OPEN Platform for Migration of Interactive Services:
Architecture and Evaluation

Anders Nickelsen, Aalborg University, Denmark

Fabio Paternò, ISTI-CNR, Italy

Agnese Grasselli, Vodafone, Italy

Kay-Uwe Schmidt, SAP Research, Germany

Miquel Martin, NEC Europe Ltd., Germany

Francesca Mureddu, Arcadia Design, Italy

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, an integrated Migration Service Platform (MSP) is proposed, where all the common migration functions are centralised. The authors 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. The authors show how migratory applications can interact with the platform and thereby do not have to contain migration functions themselves. The authors 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: *Context-Awareness, Middleware, Migration Service Platform, Service Continuity, State Adaptation*

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

DOI: 10.4018/jaras.2012040102

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 are applications 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 (Dey, 2001), 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 State of the art, 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 Scenarios and requirements, 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 is described in the State of the Art Section. Then, we present the architecture of our proposed migration platform, and finally we draw the conclusions.

SCENARIO 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.

Two representative scenarios from the domains are presented and the requirements to the platform are derived from the scenarios subsequently.

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.

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 analyze 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 synchronize the source and target devices so that virtual discussions how the emergency situation is eventually going to change are much easier to follow.

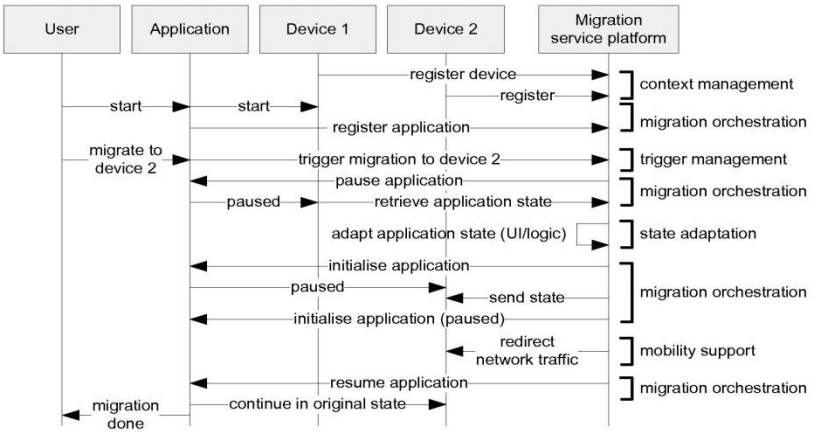
Migration Process Overview

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 migrations.

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

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



remote servers are redirected to the target device seamlessly.

Requirements to the Migration Platform

Various important requirements need to be fulfilled by a platform that provides common functions to facilitate migration. 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 networks. 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 applications 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 straight-forward, 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 is 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.

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.

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 (Gajos, 2005) 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 (2005) proposed summary thumbnails, which consist in a thumbnail view of the original Web page, but the texts are summarised enabling a good legibility. WebSplitter (Han, Perret, & Naghshineh, 2000) 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 (1997) 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 (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. 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.

Application Reconfiguration

Dynamic reconfiguration of applications and especially their behaviour deals with the adaptation of application logic during run-time based on continuously changing usage environments (Friedberg, 1987; Kramer & Magee, 1985). 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 behaviour should be seamlessly adapted. One way to realise such adaptability is to build the application out of interacting components. Kramer and Magee (1990) 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 (Aksit & Choukair, 2003). The characteristics of certain components are either known during development time or evaluated during run-time like in Camara, Canal, and Salaun (2009) 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 (Loke, 2006). Among others, Floch et al. (2006) 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 fulfill these requirements during run-time and to take action if needed (Maia, 2009).

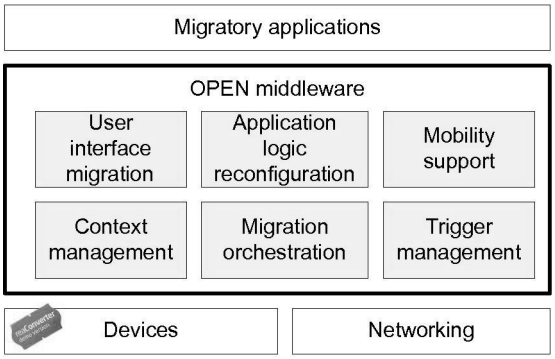
The integration of such adaptation mechanisms into a migration platform is important in order to provide the most appropriate behaviour of an application before and after migration. How the module which realises such adaptability can be integrated in such a platform will be shown in later.

Mobile Code, Sessions and Terminals

An obvious candidate research area for migratory services 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 (Fuggetta, Picco, & Vigna, 1998) 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 user 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 (Perkins, 2002; Johnson, Perkins, & Arkko, 2004), 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 (Stewart, 2007; Koh, Chang, & Lee, 2004), 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) (Rosenberg et al., 2002), which operates at the application layer and can overcome terminal mobility (Wedlund & Schulzrinne, 1999) 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 Munkongpitakkun, Kamolphiwong, and Sae-Wong (2007) where a HTTP session is moved between Web browsers on different devices, and in Chen and Wang (2008) 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 Wedlund and Schulzrinne (1999) and Schulzrinne and Wedlund (2000) and while

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



all these are working solutions they assumes that both end-points of the communication are SIP enabled, hence it is not suited for OPEN.

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 (Martin, 2009).

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.

Deployment of the MSP Middleware

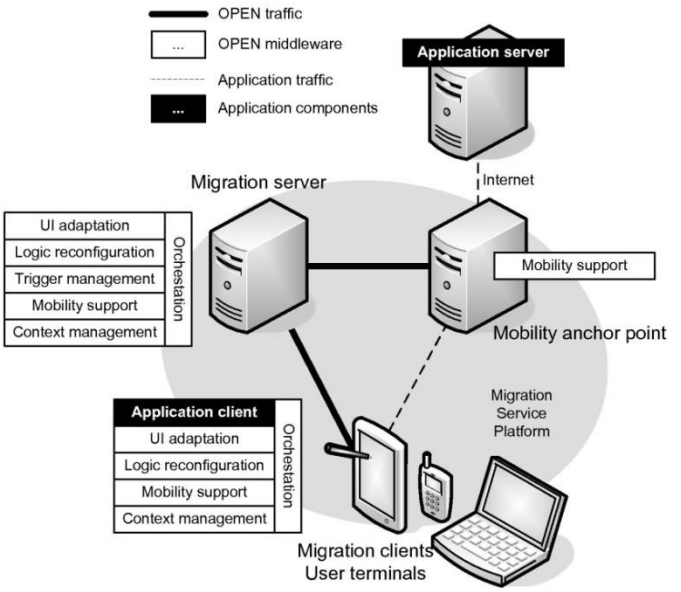
A client-server architecture is employed in the OPEN middleware, which is illustrated in Figure 3.

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 exists an adaptor for all functions 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 functions.

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 [1]through a mobility anchor point to make the migration seamless for the application server.



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 (Paterno, Santoro, & Scorcia, 2010; Stuhmer, Anicic, Sen, Ma, Schmidt, & Stojanovic, 2009).

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 behaviour 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 (Niebuhr, Rausch, Klein, Reichmann, & Schmid, 2009).

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.

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. An example scenario 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 behaviour 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. Traditional terminal mobility challenges are addressed by deploying MobileIP in the infrastructure.

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 for:

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

The Orchestration implements the migration through the following phases:

1. Application stop on the source device.
2. Platform modules orchestration: each adaptation module can contribute to the

overall migration process (e.g., for user interface or logic adaptation).
3. State maintenance: the Orchestration saves and transfers the state to the target device;
4. 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 re\-source-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 (Nickelsen, Martin, & Schwefel, 2010).

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 fulfill 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 an 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 run-time (Nickelsen, Olsen, & Schwefel, 2010).

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, (Bauer, Olsen, Jacobsen, Sanchez, Imine, & Prasad, 2006), 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) (Bauer,

Olsen, Jacobsen, Sanchez, Imine, & Prasad, 2006) 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 Olsen, Figueiras, Rasmussen, and Schwefel (2010) 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.

EVALUATION

The OPEN MSP addresses two main categories of users:

1. Migratory application end-user
2. Migratory application developers

The evaluation activities have taken into account both categories. For the former, a usability evaluation has been carried out, in order to understand end-users' perception of migration features. For the latter, a programmability evaluation has been performed in order to understand MSP flexibility and configurability. Moreover, a technological evaluation aimed at understanding the MSP requirements fulfillment and performances. All results have been created experimentally based on implemented prototypes. Below, the evaluation methodology is described and the main results are presented.

Usability Evaluation

Usability Evaluation Methodologies

The usability of a particular tool, following the ISO (International Organization for Standardization (2000) guidelines, can be evaluated in terms of:

1. Effectiveness: users' capability of completing the predefined tasks with the tool, percentage of successful and failed tasks
2. Efficiency: user's required effort for completing the task, the required time for task execution
3. Satisfaction: users' feelings during the interaction with the tool

The OPEN MSP usability evaluation shaped the analysis focusing on migration features, although these features are implemented by the middleware platform, applications are needed for exploiting the interaction between the user and the platform. During the OPEN project some prototypes have been developed in the scenarios previously described (migratory game and migratory emergency application), these prototypes have been used for the testing activity. In particular, a comprehensive task list has been created for each prototype, driving the user during the migration steps: context change (e.g., device change), adaptation and continuity.

Two different task lists drive two groups of users (groups A and B) during evaluation: same tasks but different device order to compensate any misleading effect given by using one specific device order. This approach is an adaptation for the OPEN project of one of the methods proposed by Jeffrey and Chisnell (2008) for the comparison test concerning the evaluation of multiple product versions.

For each user, during the task list execution the moderator (please refer to Jeffrey & Chisnell, 2008, for moderator characteristics) effectiveness and efficiency information.

At the end of each testing session, a questionnaire is given to the user for collecting user's feedback and feelings about the solution for the satisfaction evaluation.

The questionnaire selected for satisfaction evaluation is the SUS (System Usability Scale) described in (Brooke, 1986): it is based on ten standard sentences; the user indicates the degree of agreement/disagreement in 5 point scale. The sentences are in a predefined order that alternates positive and negative sentences, so that it is possible to mitigate the effect of

undecided testers. A score is assigned to each answer, and a predefined SUS scoring algorithm is applied in order to obtain the overall value of System Usability (SU) within the range of 0% to 100%. In the considered scale 0% means 'hard to use' while 100% means 'easy to use'.

In addition to the migration usability evaluation, during the OPEN project a part of the evaluation effort was dedicated to a competitor analysis with respect to the adaptation part of the migration. In fact, some commercially available products already implements adaptation respect to different devices, e.g., the web page adaptation performed by Opera Mini. For the competitor analysis, the task list drives the user in the interaction with the selected OPEN prototypes and the commercial products. Also for this evaluation, the approach follows the one described for comparison test in Jeffrey and Chisnell (2008). Data for effectiveness and efficiency evaluation are collected and a specific questionnaire aimed at stressing some usability aspects of the adaptation have been developed and proposed to the user.

Usability Results

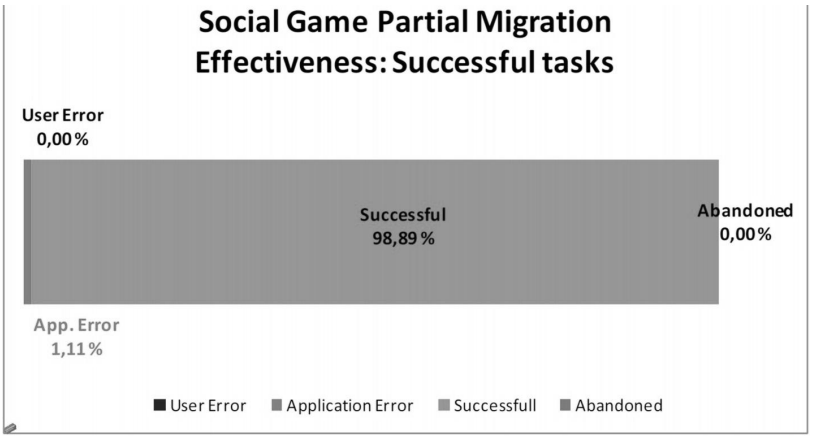
We conducted usability tests for all prototypes according to the guidelines described above in Usability evaluation methodologies. In order to provide concrete examples we detail the results of the usability test of the Social Game. The other usability tests had similar results. Details can be found in the Testing and Validation Results report (Grasselli, 2009).

Using the Social Game application for the OPEN MSP evaluation means to test an example of entertainment application whose experience is enriched by the migration features enabled through the OPEN platform. In particular the partial Web migration has been considered in this test.

Evaluation tests are performed in 2010 by involving 10 people, equally divided into two subgroups A and B. All people involved in the tests are familiar with the use of the computer because they are university students or people involved in academia activities. In the sample both men and women were considered in order to have a representative situation of the possible application end user. Group A tested the partial migration before testing the total migration and group b vice versa. Only the results of the partial migration usability test are detailed in the following paragraphs.

In order to correctly execute the Social Game usability test the following equipment are necessary: two laptop devices, one mobile device (iPhone), a wireless network infrastructure available, one "Social Game prototype tutorial" printed which the testers can consult every time that they consider it necessary.

Figure 4. Social Game partial migration effectiveness results



The tasks the test users had to fulfill were all of the kind: Migrate a part of the application running on one laptop to another laptop or to a mobile device.

Effectiveness

The main findings of the effectiveness results depicted in Figure 4 are:

1. A very small percentage of tasks were not completed (1.11%). As showed in the plot they are all due to application error. Note that it is relevant to distinguish the reason for a failed task because, from the usability point of view, the degraded user experience is caused by prototype problems and not interface or presentation layer problems.
2. No task has caused the test abandonment.
3. Almost all tasks (98.89%) were successfully completed.

Efficiency

Efficiency is the second quantitative indicator about usability related to the task execution time. In Figure 5 the execution time distribution for Social Game is reported. The plot considers the task list execution classes on x-axis and the number of people that fall in each range on y-axis. The colour bar in overlay to x-axis ticks shows the estimated time to execute the task list.

The distribution histogram can be interpreted as follows:

1. Recorded times are coherent with the proposed estimation.
2. All data fill into two classes. It is positive because different users answer to the test more or less in a close time and it can be a positive indicator of how the platform is usable because perceived in the same way by different users.
3. There is not any appreciable difference between the people that execute first the Social Game (group A) respect to the ones that executes Social Game after have tried web page total migration.

Satisfaction and system usability scale: The system usability scale results are obtained using the SUS questionnaire meant for the evaluation of the user satisfaction with the partial migration experience. The evaluation methodology follows the classification described in detail in Usability evaluation methodologies. Figure 6 depicts the classification obtained during the performed tests. During the tests the average SUS score obtained is 75% with a standard deviation of 11%. Please note that the obtained score is a very good one especially when considering the standard deviation associated to it. Regarding the observation of the user comments and behaviours it is interesting to note that the support of migration applied to the Social Game was well perceived by all end users.

Usability Evaluation Conclusions

The partial migration of web applications has been evaluated by the test users in a positive way. The main finding was that the social game partial migration is an interesting and easy to use feature. Moreover also the effectiveness (successful tasks: 98.89%) and efficiency (task list completed within the estimated time: 100%) results highlight positive performances.

Similar results were also obtained from the usability tests with the Emergency Application. Even if the Emergency Application is still a prototype, a strong positive result has been obtained (effectiveness - successful tasks: 97.28% and efficiency - task list completed within the estimated time: 100%).

After all we can conclude that the usability tests attested that our migratory prototypes are usable and that they do not expose any major usability issues.

Programmability Evaluation

Programmability Evaluation Methodologies

While the usability evaluation addressed end-users' perception of migration feature, the programmability evaluation aims at understanding the migration application developers point of

Figure 5. Social Game partial migration efficiency results

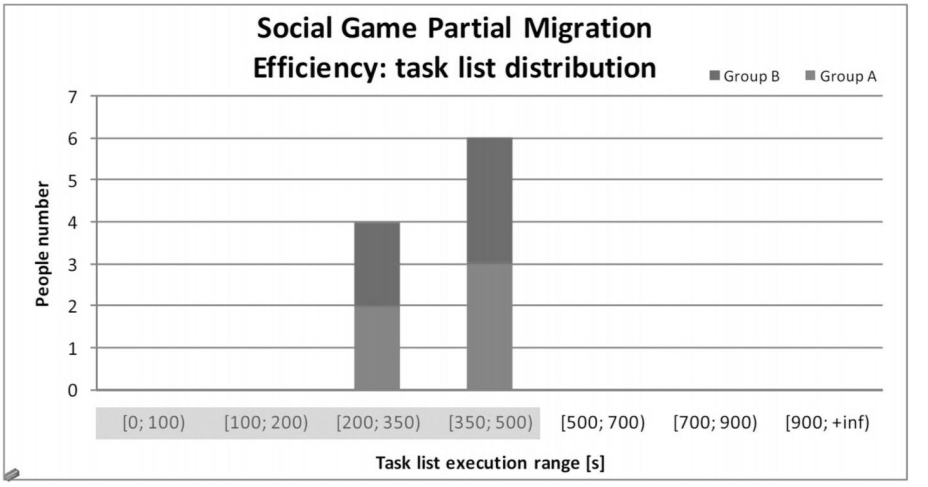
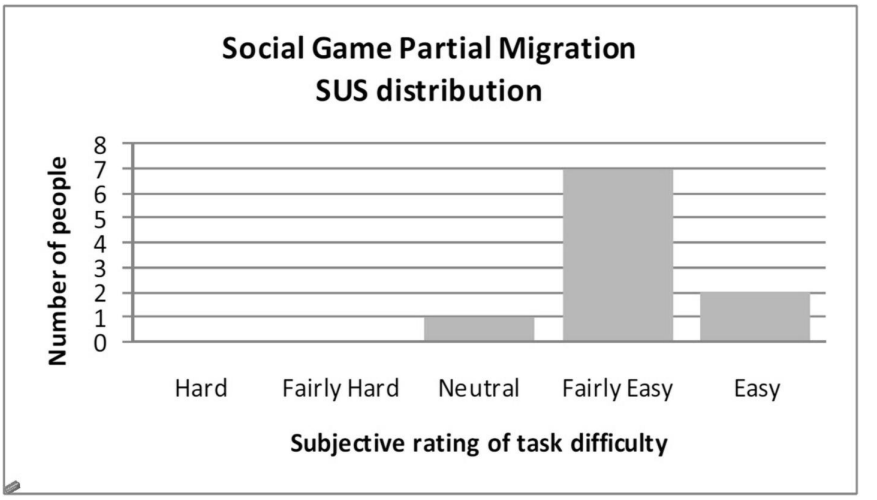


Figure 6. Social Game partial migration SUS results



view. In fact, each new application would need a customized migration and adaptation and the MSP should provide the required flexibility and configurability in order to allow the proper customization.

Referring to the general procedure and necessary steps for performing migration from a source device to a target device as described in Figure 1, the main steps that need to be configurable are:

1. Devices registration: during the registration procedure, the OPEN client communicates to the OPEN MSP device and network capabilities as well as application's requirements to devices and networks. Different applications may need to use different variables, hence the modules handling registration must be configurable, in order to accept and handle different variables.

2. Then, migration can be triggered, either manually by the user or automatically by the application or as a reaction to contextual changes. Different applications may allow different migration triggers, hence the modules handling migration must be configurable, in order to set the required migration. In case of platform triggered migration as reaction to context change, the application developer must be able to set the proper rule for the trigger, depending on the application specific variables.
3. 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. Also the adaptation rules could be customized for different application, so the application logic and the user interface adaptation must be configurable.

The programmability evaluation has been carried out from the single modules point of view and addressed two main topics:

1. Handling new variables
2. Managing different migration rules

The aim of the programmability analysis is to verify if variables and migration rules can be correctly managed by the OPEN MSP. This is done by considering single modules, each one with its programmability features, and then with an overall point of view of the whole platform. The goal of the programmability evaluation is to obtain a classification of the programmability features considering two dimensions:

Extensibility. A module is extensible if and only if it allows adding and removing variables and/or rules able to control its behaviour without changing the module source code.

Configurability. A module is configurable if it supports changing already defined variables and/or rules in order to modify how the system answers to different situations. Also in this case changing configurations should be possible without changing the module source code.

The assessment starts from each module by identifying if configuration facilities are available and what can be configured and eventually extended. The second step considers how well the extensions and configurations that are available according to the OPEN MSP implementation. In this case two pass/fail dimensions are considered which are:

Robustness towards errors in the configuration files that are managed by the platform. For example new behaviours are not enabled, but the entire platform does not crash, but continues to be available. Robustness is evaluated modifying with no valid parameters or deleting the configuration and check if the platform notifies the configuration error continuing to work without denial of service.

Consistency checks that configuration changes effectively change the platform behaviour. Consistency is evaluated by changing a configuration and checking that the configured behaviour is the one exercised by the platform.

Programmability Evaluation Results

In the following are described the results of evaluating programmability for the modules in the migration platform that contain variables or rules that can be changed.

The modules are: context management, trigger management, application logic reconfiguration and user interface adaptation.

The trigger management module is developed in a well-structured way but configurability is not yet supported, as the configurations are hard-coded and do not consider any external configuration file. The reason is that all the development efforts about this module were addressed on the module integration inside the OPEN Migration Service Platform rather than on its configurability features. The module is not considered further in this programmability evaluation.

Context Management

The context management module is the MSP module that manages the available context data in the environment. It is important to point out that module itself provides the infrastructure to collect information independently from data types.

During the first test iteration the programmability assessment identifies the CMF as a programmable module considering the possibility to add variables and to configure how the CMF itself can obtain information from the environment. So, the improvement proposals were focused on the use of the OSGi technology that makes possible to add/remove a retriever without the restart of the complete CMF module enhancing the module flexibility. The last OPEN MSP implementation follows the previously exported recommendations in fact the CMF is implemented through OSGi methodology using Equinox which is a certified implementation of the OSGi R4 core framework. Thanks to the OSGi architecture model it is possible to add, remove, start, stop and update single retrievers without restarting the entire CMF module.

The module is composed of retrievers, which are the components that collect information from sensors and transducers in order to make them available to the applications by the context management module. Each retriever manages a single piece of information like for example battery level, environment temperature, GPS positions, etc. Each of these pieces of information is so considered a variable which is managed by CMF. So, adding a variable means in practical to add a retriever able to manage a new piece of information which was not previously considered. According to the features described the context management module can be classified as extensible and configurable:

1. Extensible because it can be enabled to manage new variables simply creating the retriever module and it can be added and directly started to a running instance without any module restarting. The module restart is not necessary also when some changes are implemented in the retriever, because it can simply update to the new version through the update functionality of the framework.
2. Configurable because it is always possible to edit the method in which the information are retrieved through the XML file that contains the queries subscription specifications. At the same time the module can be configured through another XML file to be used on different network nodes with custom ports.

Application Logic Reconfiguration

The application logic reconfiguration module is devoted to the change of the application logic: it modifies the application behaviour by changing how the application takes decisions and actuates them. The platform implementation provides an XML configuration file to set up how the application module can be configured and rewired. For each application component the corresponding XML component descriptor file is produced in order to describe the input and output interfaces required and provided by the considered application module. According to the features provided, using the XML configuration file the ALR module can be classified as extensible and configurable:

1. Extensible because once the new application modules to consider are available they can be plugged to the logic simply adding them with their wiring constraints to the XML configuration files.
2. Configurable because you can always change the constraints configuration in the XML files.

About robustness the module results are quite good even if it is suggested to improve the exception explanation in order to make them better comprehensible by the user. Another thing to remember is to restart the ALR once the XML configuration file has to be edited. From the consistency point of view the module results are good because the behaviours

declared during the configuration phase results effectively reproduced during the application execution. This is a good result because the suggestions exported by the first test iteration are now implemented and working.

User Interface Adaptation

Analyzing the UI Adaptation module it is possible to classify this module as not extensible but highly configurable:

1. Not Extensible because the parameters are internally fixed by the module and if new parameters are considered they must be added by hard coding their management rules and recompiling the entire module code.
2. Configurable because it is possible to configure a large number of parameters for the desktop to mobile adaptation, selecting which value they have to assume in a predefined range.

In this context one important aspect is control on the rules that drive adaptation to the various platforms (the most common case is desktop-to-mobile adaptation). For example, the adaptation engine is able to split the desktop pages when they require considerable amount of interaction resources but some users may like to have more control on the splitting algorithm.

The module provides a web based user interface that allows end users to configure the adaptation process, as exemplified in Figure 7.

The various parameters are grouped according to the user interface aspect considered. For the fonts, it is possible to specify the minimum and maximum font in the target device and the associated measure unit, which is the default font with its family. For the radio buttons it is possible to indicate whether they should be transformed into an interaction that supports the same semantics but with using less space screen. In this case, it is possible to specify the threshold, in terms of number of choice options, which should trigger the transformation and the type of interaction to use for its replacement. Similar parameters are available for the list boxes. Other parameters concern the maximum number of characters for a text, maximum and minimum dimensions for images. Finally the page splitting options (Figure 7) are reported giving the possibility to the user to enable or disable the page splitting features.

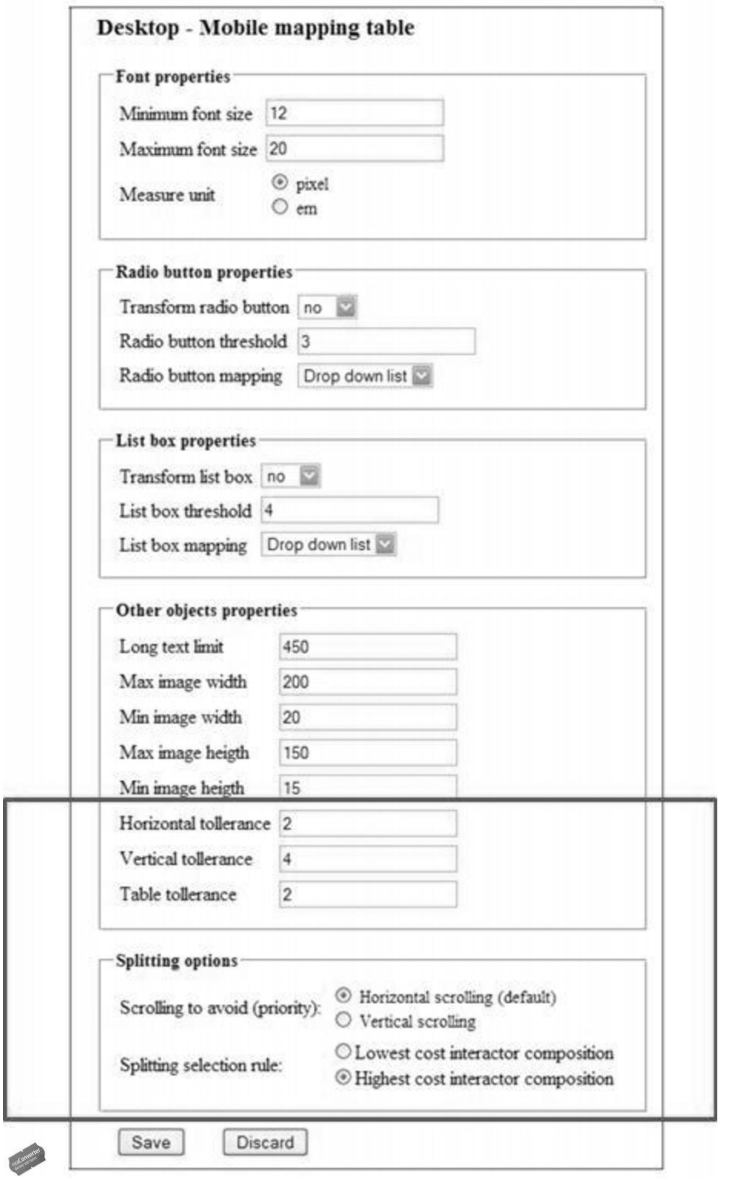
All these parameters determine the cost of rendering a presentation. This cost is compared with the overall sustainable cost in the target device, which is given by the screen resolution multiplied by horizontal and vertical tolerance. The higher the tolerance coefficient values are, the more scrollable the generated user interface will be. This means that end users have the possibility to specify to what extent the adapted content will be scrollable in the target device. The table tolerance provides an additional factor to consider when calculating the sustainable cost. In practice, this means that when there are tables more scrolling will be acceptable before deciding to split the presentation. The customization interface also allows the user to indicate what type of scrolling (horizontal or vertical) to avoid has the priority and the algorithm splitting version to apply. Through the considered interface it is possible to configure and store the specification of user preferences regarding web user interface adaptation.

As previously described in the programmability methodology after the theoretical evaluation carried out on the basis of the available documentation it is now possible to test how much the interface configurable features are robust and consistent.

About the robustness a user interface is available to the user, so data validation is intrinsically checked by the user before saving the configuration. Note that using a user interface for some parameters means that the configuration simply consists of choosing what value to assign among the available values. Thanks to the web interface the user has no visibility and access to any configuration files and this is good to ensure that UI Adaptation module does not export files which are potentially vulnerable.

At the end considering the UI Adaptation module it is confirmed how it results highly

Figure 7. Web UI Adaptation programmability: Desktop to Mobile configuration table



configurable. Considering the extensibility it is not kept into account because it is clear that once a new configurable parameter is considered it is necessary to edit the parser and adapted web page builder features. This can be done by changing the module code implementation. A possible future solution that could be investigated could consider implementing the WebUIAdaptation module using an approach like the ALR one. So that it may be extensible adding the necessary code component to manage the new adaptation behaviour. Then how the module interacts with the rest of the system could be done through an XML file.

From the consistency point of view the module results are good because the configurations declared in the desktop-mobile configuration results effectively reproduced by the UI Adaptation module behaviours. Another important thing to remember is that it is not necessary to restart the UI Adaptation module once the adaptation configuration is changed because it is hot loaded by the module and it becomes immediately effective.

Concluding, this is a good result because with respect to the previous version tested during the first test iteration now the module can be configured considering many parameters and at the same time it results more robust. At the same time it is necessary to keep in consideration that this is one of the more complex modules that compose the OPEN platform and its actual state is a milestone about User Interface adaptation functionalities. Sure next improvements efforts can be focused on its extensibility features.

Technological Evaluation

Technological Evaluation Methodologies

The technological evaluation methodology consists of an analysis of performance indicators:

1. Functional evaluation analysis: for each considered platform requirement it provides a yes/no answer (similar to a standard acceptance test).
2. Indicators evaluation analysis: it focuses on a set of indicators useful to give feedback about different aspect of the OPEN MSP.

The functional evaluation process considers the requirements elicited during the first phases of the OPEN project (contextualizing/specifying some of them if required). Since the OPEN MSP is a middleware, it is necessary to use the prototype applications in order to verify that it correctly implements the requirements elicited. The requirements have been classified as:

1. SUPPORTED: the requirement is available in the current implementation of the OPEN MSP platform and could be:
2. VERIFIABLE: there exists at least a prototype or an ad-hoc application which is able to verify, through a test case, if the requirement is satisfied or not. After the test case execution, the requirement could be:
3. PASSED: The requirement is testable and it successfully passed the test case.
4. NOT PASSED: The requirement is testable but it doesn't pass the test case.
5. NOT VERIFIABLE: no prototype exists able to verify whether or not the requirement is satisfied even if it is implemented in the current version of the middleware.
6. NOT SUPPORTED: The OPEN MSP does not implement the requirement in the current version, but it is implementation can be considered in future if the requirement still represents a strategic point.
7. NO ADDED VALUE: The requirement does not add any value, considering the objective of the OPEN MSP. It happens that some aspects that in the early stage of the development have been considered as platform requirements result more precisely as already implemented in other software/middleware that can be efficiently used together with the OPEN MSP. It makes in fact more sense to delegate this features to infrastructures and systems specialized to manage this issues (e.g., Network aspect, database management). Typically, these requirements refer to an issue that works at lower level respect to the OPEN MSP.

For all the requirements that are "Verifiable", a test case is built using the fixed structure reported in Table 1. Once the test is executed, the result is clearly readable in the "Status" field: it can be only "Passed" or "Not passed."

ID & Test case IDExecutor & Test executorItem & Migration aspect evaluatedPrototype & Considered prototype(s)Status & Passed/Not

Table 1. Test case template

ID & Test case ID
Executor & Test executor
Item & Migration aspect evaluated
Prototype & Considered prototype(s)
Status & Passed/Not passed
Input & Input for test execution
Expected output & In case of success execution
Actual output & Output of the executed test
General considerations & Additional notes
Execution date & Execution test date
Status & Passed/Not passed

passedInput & Input for test executionExpected output & In case of success execution-Actual output & Output of the executed test-General considerations & Additional notesExecution date & Execution test dateStatus & Passed/Not passed

The indicators evaluation process aims at providing some quantitative data for an overall analysis of platform behaviour. The performance evaluation addressed different platform aspect:

1. Key Performance Indicators (KPI): the evaluation aim at understanding platform performances (e.g., migration time)
2. Non functional requirements: Availability, Reliability, Scalability, Security
3. Accessibility and adherence to the standards.

Part of the technological evaluation efforts have been addressed also for some modules evaluation. For the module performances evaluation the focus is on modules that mostly influence the platform performances, e.g., modules whose behaviours are most impacted by platform operative conditions. The selected modules are:

1. Context Management
2. Application Logic Reconfiguration
3. User Interface Migration

Moreover, multi-network analysis is carried out considering the OPEN MSP coupled with Mobility Support module necessary to enable the multi-network application migration.

Technological Evaluation Results

Many detailed technological results were obtained in the project, in particular regarding performance of the internal modules of the MSP. More details on these can be found in Grasselli (2009). The examples shown in the following regard the performance of the MSP as a whole, measured in the developed prototypes.

The majority of the requirements are testable; it means that a test case can be produced to evaluate them.

1. A successful result is always obtained (0% test cases fail).
2. Only 13% of the initial requirements are not supported and not implemented in the final version of the platform.
3. A very small amount of requirements (5%) is “Not Testable”. It means that the

prototypes developed on top of the OPEN platform guarantee an optimal coverage of the use case in which the OPEN MSP can be used.

All indicators where measured on the developed prototypes, called "Emergency services" and "Web migration".

To evaluate availability and reliability 20 migration were performed successfully over 3 days. The platform was operating correctly during the entire period.

Migration time (T2-T1) is recorded starting from the migration orchestration log with a sensibility of 1 millisecond.

1. T1 = migration triggered on Orchestration Server
2. T2 = migration reported as completed on Orchestration Server

Emergency prototype (Figure 8): The mode is 1 second; this is a good result because it means that the most migrations require a very short amount of time to complete.

Social game (Figure 9): The results show that the required time for migration process depends on the number and the kind of information that is requested to migrate, considering that it is also necessary to adapt each one of the migrating components. This aspect is clear from the diagram, which considers migration of different web pages (or pieces of them) obtaining different times, on the basis of the content and structural complexity of the page considered. Please note that the times recorded here are greater than the ones recorded in other tests because here the device change requires the UI adaptation step, not considered in previous cases.

In general, the average time of the main modules involved in user interface adaptation process (Proxy, State Mapper, Reverse, Semantic Redesign and UI Generator), indicates that the most time consuming modules are the Semantic Redesign and the Reverse (Figure 10).

Figure 11 shows that the application logic reconfiguration time depends on the number of registered components. The reconfiguration is local if source and target components are located on same device, and otherwise remote. The registering of the first component takes in both setups (local and remote) much more time than for integrating the following instances. The reason for this is that the first instance not only triggers the integration of that component, but also the instantiation of the other components. This is because, in the prototype, the application descriptor specifies that at least two components have to be connected in order to start the first component. The results show also that the reconfiguration time is strictly proportional in the remote and local case, while the gradient in the remote case is higher.

Figure 8. Migration time measured on the "Emergency Services" prototype

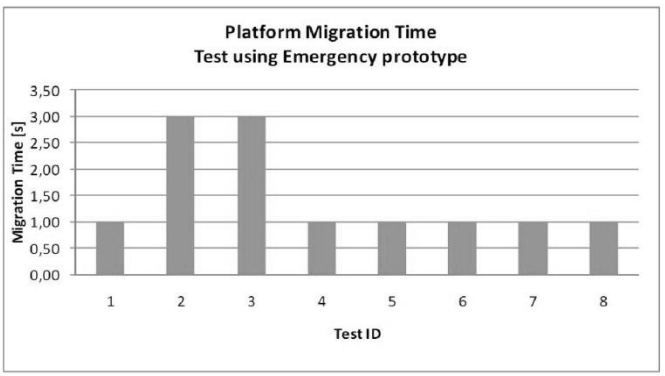


Figure 9. Migration time measured on the "Web migration" prototype

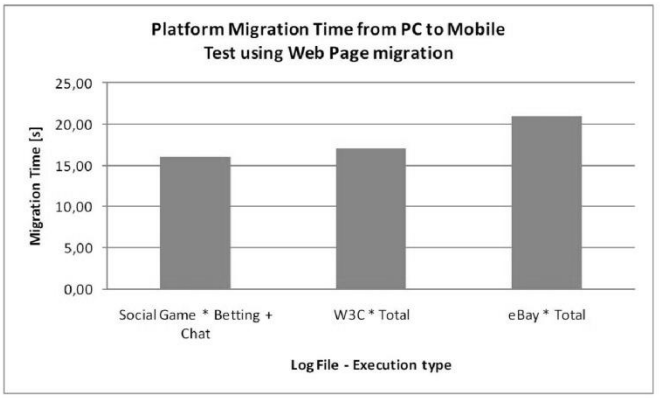
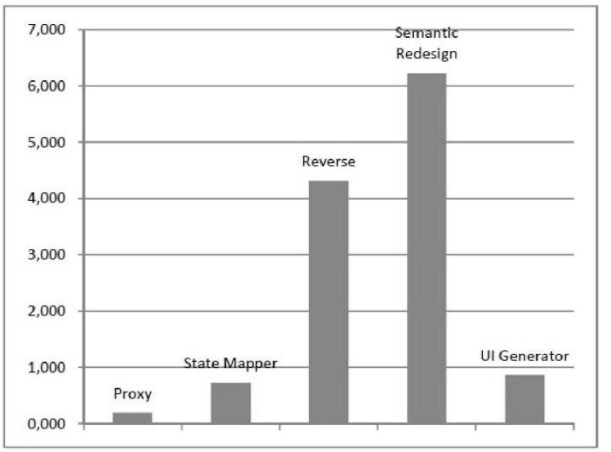


Figure 10. Average time in seconds spent in the different modules during a web migration



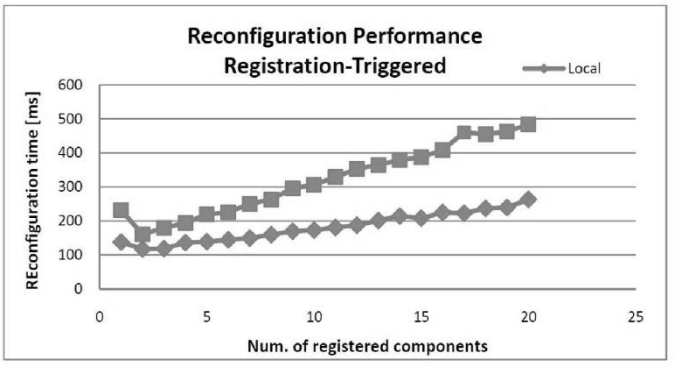
The indicator evaluations highlight that the OPEN MSP performances are quite good taking in consideration that it is still a prototype. In particular, the migration time, which is a very critical performance indicator confirmed a short platform response time.

Application Example: Digital Signage

In order to illustrate some of the advantages of a migration platform, the following illustrates how they can best be put to use in the context of Digital Signage installations. In short, work on Digital Signage aims at replacing paper signs with digital displays. Technologically speaking, POPAI, The Global Association for Marketing at-Retail describes it as "A network of digital displays that are centrally managed and addressable for targeted information, entertainment, merchandising, and advertising" ("The Evolution of a New Media," n.d.).

One of the main challenges the industry faces is the retention of customer attention, whereby numerous studies point out the need to engage the user in the installation. While content personalization is a good first step,

Figure 11. Time in milliseconds required for logic reconfiguration



the logical follow-up is to actually enable users to use part of the installation as their own. A typical scenario would involve a waiting room be it at the doctor's or an airport. In it, users are allowed to take control of a display and transfer some of the applications on their mobile phone to it, effectively migrating or partially migrating them.

The technologies presented in this paper constitute an appropriate basis for such scenarios: not only does the migration platform provide the tools required, but it also considers the user's context and provides hooks for appropriate privacy mechanisms.

Consider the combination of the Context Management and the Trigger Management modules: together, they provide the local intelligence required to adapt actions to the user and its surroundings. The demonstrator shown by NEC (Information Surfaces) at ISE 2010, a major Digital Signage fair illustrates how a camera, RFID sensors, accelerometers and a pick-up sensor can be used to this end: the application logic is customized (e.g., increasing information density as the user approaches the display), and new interaction mechanisms are provided (by putting an RFID-tagged car on a picture to get further information). Furthermore, the context information can be used to create audience reports, which further justify the owner's investment.

Bringing your own applications embodies the very idea of application migration from the user's device, to the additional screen, bringing in the rest of the migration platform and all the experiences gained in it. Applications can be social, as in the Social game prototype, or for the sole benefit of using a larger screen, such as those that currently display news or fairly static content in an airport. The possibilities are virtually endless, limited only by the user's desired level of privacy.

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.

ACKNOWLEDGMENT

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

REFERENCES

Aksit, M., & Choukair, Z. (2003). Dynamic, adaptive and reconfigurable systems overview and prospective vision. In *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops* (pp. 84-89).

Bauer, M., Olsen, R., Jacobsen, M., Sanchez, L., Imine, M., & Prasad, N. (2006). Context management framework for MAGNET beyond. In *Proceedings of the IST Mobile and Wireless Summit Workshop on Capturing Context and Context Aware Systems and Platforms*.

Bharat, K. C. (1997). Migratory applications . In Vítek, J., & Tschudin, C. (Eds.), *Mobile object systems towards the programmable Internet* (pp. 131–148). New York, NY: Springer. doi:10.1007/3-540-62852-5_11

Brooke, J. (1986). SUS - A quick and dirty usability scale . In Jordan, P. W. (Ed.), *Usability evaluation in industry*. Boca Raton, FL: CRC Press.

Camara, J., Canal, C., & Salaun, G. (2009). Behavioural self-adaptation of services in ubiquitous computing environments. In *Proceedings of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* (pp. 28-37).

Chen, M.-X., & Wang, F.-J. (2008). Session mobility of SIP over multiple devices. In *Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities* (pp. 1-8).

Dey, A. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, *5*, 4–7. doi:10.1007/s007790170019

Floch, J., Hallsteinsen, S., Stav, E., Eliassen, F., Lund, K., & Gjorven, E. (2006). Using architecture models for runtime adaptability. *IEEE Software*, *23*(2), 62–70. doi:10.1109/MS.2006.61

Friedberg, S. (1987). *Transparent reconfiguration requires a third-party connect* (Tech. Rep. No. 220). Rochester, NY: Computer Science Department, University of Rochester.

Fuggetta, A., Picco, G. P., & Vigna, G. (1998). Understanding code mobility. *IEEE Transactions on Software Engineering*, *24*, 342–361. doi:10.1109/32.685258

Gajos, K. A. (2005). Fast and robust interface generation for ubiquitous applications. In *Proceedings of the International Conference on Ubiquitous Computing* (pp. 37-55).

Grasselli, A. (2009). *D6.7: Testing and validation results*. Brussels, Belgium: European Commission.

Han, R. P. (2000). WebSplitter: a unified XML framework for multi-device collaborative Web browsing. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (p. 230). Information Surfaces. (n.d). *NEC and Instoremedia*. Retrieved from http://www.youtube.com/watch?v=ibqBtF3PGwU

International Organization for Standardization (ISO). (2000). *ISO 9241 11: Ergonomic requirements for office work with visual display terminals*. Geneva, Switzerland: Author.

Jeffrey, R., & Chisnell, D. (2008). *Handbook of usability testing*. Geneva, Switzerland: International Organization for Standardization.

Johnson, D., Perkins, C., & Arkko, J. (2004). *RFC 3755: IP mobility support in IPv6.* Retrieved from http://www.ietf.org/rfc/rfc3775.txt

Klus, H., Niebuhr, D., & Rausch, A. (2007). A component model for dynamic adaptive systems. In *Proceedings of the International Workshop on Engineering of Software Services for Pervasive Environments: In conjunction with the 6th ESEC/FSE Joint Meeting* (p. 28).

Koh, S. J., Chang, M. J., & Lee, M. (2004). mSCTP for soft handover in transport layer. *IEEE Communications Letters*, *8*(3), 189–191. doi:10.1109/LCOMM.2004.823432

Kramer, J., & Magee, J. (1985). Dynamic configuration for distributed systems. *IEEE Transactions on Software Engineering*, *11*(4), 424–436. doi:10.1109/TSE.1985.232231

Kramer, J., & Magee, J. (1990). The evolving philosophers problem: Dynamic change management. *IEEE Transactions on Software Engineering*, *16*, 1293–1306. doi:10.1109/32.60317

Lam, H. B. (2005). Summary thumbnails: readable overviews for small screen web browsers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 681-690).

Loke, S. (2006). *Context-aware pervasive systems: architectures for a new breed of applications*. Boston, MA: Auerbach. doi:10.1201/9781420013498

Luyten, K. C. (2005). Distributed user interface elements to support smart interaction spaces. In *Proceedings of the Seventh IEEE International Symposium on Multimedia* (p. 8).

Maia, M. L. (2009). Requirements and challenges for building service-oriented pervasive middleware. In *Proceedings of the International Conference on Pervasive Services* (pp. 93-102).

Martin, M. a. (2009). *D4.2: Migration service platform design*. Brussels, Belgium: European Commission.

Munkongpitakkun, W., Kamolphiwong, S., & Sae-Wong, S. (2007). Enhanced web session mobility based on SIP. In *Proceedings of the Conference on Mobile Technology, Applications, and Systems*, Singapore (pp. 346-350).

Nickelsen, A., Martin, M., & Schwefel, H.-P. (2010). Service migration protocol for NFC links. In *Proceedings of the 16th EUNICE/IFIP WG 6.6 Conference on Networked Services and Applications: Engineering, Control and Management* (pp. 41-50).

Nickelsen, A., Olsen, R., & Schwefel, H.-P. (2010). Model-based decision framework for autonomous application migration. In *Proceedings of the 18th International Conference on Analytical and Stochastic Modeling Techniques and Applications* (pp. 55-69).

Niebuhr, D., Rausch, A., Klein, C., Reichmann, J., & Schmid, R. (2009). Achieving dependable component bindings in dynamic adaptive systems-a runtime testing approach. In *Proceedings of the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems* (pp. 186-197).

Olsen, R., Figueiras, J., Rasmussen, J., & Schwefel, H.-P. (2010). How precise should localization be? - A quantitative analysis of the impact of delay and mobility on reliability of location information. In *Proceedings of the IEEE International Conference on Global Telecommunications* (pp. 1-6).

Paternò, F., Santoro, C., & Scorcia, A. (2008). User interface migration between mobile devices and digital tv. In *Proceedings of the 2nd Conference on Human-Centered Software Engineering and 7th International Workshop on Task Models and Diagrams* (p. 292).

Paternò, F., Santoro, C., & Scorcia, A. (2010). Ambient intelligence for supporting task continuity across multiple devices and implementation languages. *The Computer Journal*, *53*(8), 1210–1228. doi:10.1093/comjnl/bxp014

Perkins, C. (2002). *RFC 3344: IP mobility support for IPv4.* Retrieved from http://www.ietf.org/rfc/rfc3344.txt

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., et al. (2002). *RFC 3261: SIP: Session Initiation Protocol.* Retrieved from http://www.ietf.org/rfc/rfc3261.txt

Schulzrinne, H., & Wedlund, E. (2000). Application-layer mobility using SIP. *ACM SIGMOBILE Mobile Computing and Communications Review*, *4*, 47–57. doi:10.1145/372346.372369

Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., & Kozuka, M. (2007). *RFC 5061: Stream control transmission protocol (SCTP) dynamic address reconfiguration.* Retrieved from http://tools.ietf.org/html/rfc5061

Stuhmer, R., Anicic, D., Sen, S., Ma, J., Schmidt, K., & Stojanovic, N. (2009). Lifting events in rdf from interactions with annotated web pages. In *Proceedings of the 8th International Semantic Web Conference* (pp. 893-908).

The Evolution of a New Media. (n.d.). *Tratto da POPAI*. Retrieved from http://www.popaidigitalblog.com/blog/articles/The_Evolution_Of_A_New_Media-537.html

Wedlund, E., & Schulzrinne, H. (1999). Mobility support using SIP. In *Proceedings of the 2nd ACM International Workshop on Wireless Mobile Multimedia*, Seattle, WA (pp. 76-82).

Anders Nickelsen is a Quality Assurance Engineer at Tradeshift in Copenhagen, Denmark. He received his PhD on service migration in dynamic and resource-constrained networks from Aalborg University in 2011. He is now working with application of research on cloud-computing technologies to improve software development processes.

Fabio Paternò is Research Director and Head of the Laboratory on Human Interfaces in Information Systems at CNR-ISTI in Pisa, Italy. His research interests are Ubiquitous Interfaces, Methods and Tools for Multimodal User Interface Design and Evaluation, Accessibility, User Interfaces for Mobile Devices, Model-Based Design of Interactive Systems, and End-User Development. He has been the scientific coordinator of five EU projects, including OPEN. He is member of the IFIP Technical Committee 13 on Human Computer Interaction, and is the chair of IFIP WG 2.7/13.4. In 2010 he was appointed ACM Distinguished Scientist.

Agnese Grasselli works as Technology Strategy Specialist at Vodafone Omnitel NV at Milano. As part of the Service Architecture team, she is involved in technologies evaluations and assessments, and innovative services feasibility studies.

Kay-Uwe Schmidt works as Senior Researcher at SAP Research in Karlsruhe. His research topics are adaptive user interfaces, and, recently, billing systems for the emerging market segment of ElectroMobility.

Miquel Martin is a senior researcher at NEC Laboratories Europe, Heidelberg, Germany. He focuses on novel uses of context information, specializing in Digital Signage solutions and pushing the boundaries of today's Social Networks. He has a passion for web applications and their specific need for scalable, high-availability systems, as well as the platforms required for large-scale, distributed data mining.

Francesca Mureddu is a R&D Project Manager at Arcadia Design. In 1999 she took her Master Degree in Electronic Engineer and she joined the game company Mediola as software developer. In 2001 she joined Arcadia Design where she designed and developed mobile games and applications for different platforms. She worked in FP6 projects (OLGA, OpenInterface) as lead developer, and in FP7 project OPEN as Workopackage leader.