OPEN Project

STREP Project FP7-ICT-2007-1 N.216552

Title of Document:	Testing and Validation Methodology
Author(s):	Grasselli Agnese, Marzorati Stefano, Mazzei Simone, Piunti Mattia
Contributor(s):	C. Santoro, R. L. Olsen, H. Klus, M. Henning, S. Labeaga, A. Nickelsen, F. Moreddu
Affiliation(s):	Vod, AAU, CNR, ClU, SAP, NEC, AD
Date of Document:	1-07-2009
OPEN Document:	D6.4
Distribution:	Public
Keyword List:	
Version:	1.6

OPEN Partners:

CNR-ISTI (Italy)
Aalborg University (Denmark)
Arcadia Design (Italy)
NEC (United Kingdom)
SAP AG (Germany)
Vodafone Omnitel NV (Italy)
Clausthal University (Germany)

Title:	**Id Number: 0**

Abstract

This document is expected to serve as a handbook for the evaluation and validation of OPEN technological solution.

Title:	**Id Number: 0**

Table of Contents

1 Introduction

This document is intended to be the baseline for the execution of testing procedures within the OPEN project. The evaluation of the OPEN migratory platform and the related applications will be divided in two different experiences, involving two sets of prototypes developed by the OPEN partners.

- The **first testing iteration** aims to evaluate the result of the first prototypes (the ones described in D2.1, D3.2 and D5.2), they don't represent completely the final outcome of the project but they can be seen as useful "Proof of concept": the prototypes are simple demonstrator of some of the basic concept that the project is aiming to implement. The expected outputs of this iteration are useful feedback for improving the final solution.
- A second phase of the development will lead to embedded applications in the "OPEN eco-system"; if correctly validated in the second testing iteration (**final testing iteration**), they will benchmark the success of such a project.

Testing involves three different evaluation areas (usability, programmability, and technology), whose testing methodology will be described in the next three sections of the document:

- **Usability** will be analyzed through three steps: an *exploratory* study on the description of some prototypes, an *assessment* of the usability level reached by some prototypes during the first testing iteration, and the following *validation*, performed during the final testing iteration. A key element of such a process is the characterization of test participants.
- Also for **programmability** an *assessment* is needed to underline where a configuration facility is implemented and how (thus analyzing context variables, migration rules, et cetera); then, a following *validation* of it is provided for all the configurable modules present in at least one available prototype.
- **Technological evaluation** will verify the level that has been reached for some technical *indicators* commonly used to determine the working of a service/product; the analysis will be enriched by checking the satisfaction of *specific requirements* for the OPEN project.

Each section will define the overall testing methodology that will be applied to a set of prototypes submitted to the testing; after that, the sections related to the three areas' test plans will shape the testing procedures to be followed for each prototype during the first stage.

The results of the evaluations will be collected in D6.5 "Evaluation results".

The second iteration testing activity will be shaped using the same guidelines and templates described in this document. Since D6.4 could only contain the test plans for the available modules and prototypes, it is worth to foresee an internal report in order to specify the test plans for the final prototype(s), which depends on future developments.

2 Usability

2.1 Methodology

During the OPEN project development cycle, three usability evaluations have been scheduled.

The first usability evaluation is an **exploratory** study about users' opinion on some important design concepts. A questionnaire about fundamental characteristics of OPEN prototypes has been proposed to a selected group of users. Furthermore, a study has been carried out to review the requirements of the OPEN project. In this case, the usability level has been evaluated by a theoretical analysis of the process used for the OPEN requirements elicitation, in a User-centered-design (UCD) optic, giving some input for further work. This is not a test to be performed on a working prototype, but a preliminary study about some fundamental design concepts.

The second usability evaluation is an **assessment** test. This kind of test is conducted either early or midway into the product development cycle, usually after the fundamental or high-level design or organization of the product has been established [HUT]. The objective of this test is to evaluate the user's feelings about the product during the execution of realistic tasks.

The last usability evaluation to be performed on the OPEN platform is the **validation** test. It will be conducted when the project development cycle will be near to the end. In this case, a more formal approach will be used, because it will be needed to individuate in a very accurate way the software usability problems to be communicated to the development team. When this test is performed it is expected that every software structural usability problem has already been solved, thanks to the results of the assessment test. So, there could be only some problems on how initial usability requirements and the requirements defined after the assessment test have been implemented.

The exploratory and assessment analysis will be carried out for the first iteration evaluation, while the validation test will be performed for the final prototype.

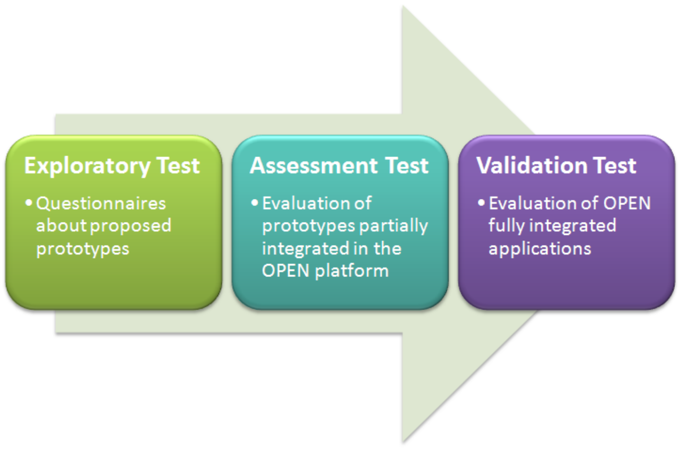

Exploratory Test
• Questionnaires about proposed prototypes
Assessment Test
• Evaluation of prototypes partially integrated in the OPEN platform
Validation Test
• Evaluation of OPEN fully integrated applications

Figure 1: Usability evaluations during the OPEN project development cycle

2.1.1 Test Participants Characteristics

In this section some guidelines will be given concerning the selection of the OPEN project usability tests participants.

First of all, it is necessary to individuate the OPEN end user characteristics. This is a not trivial issue and the success of an usability testing activity is often related to this point.

Usually, for commercial products, the marketing department performs some analysis about the end user characteristics. This kind of information is not complete for a usability test, but it is surely a good starting point. For the OPEN project it is not possible to have such information, and, moreover, the user characteristics can change with the tested application running over the middleware. In fact, the OPEN middleware is designed to support a wide range of applications (e.g. games, simulators, etc.) and for each application there is a specific target user. Thus, the better way of proceeding is to define some common characteristics of the OPEN platform user, and then, for each tested application prototype, specific characteristics will be defined. Finally, applications and middleware specifications can be used in order to find an appropriate group of users for usability evaluations.

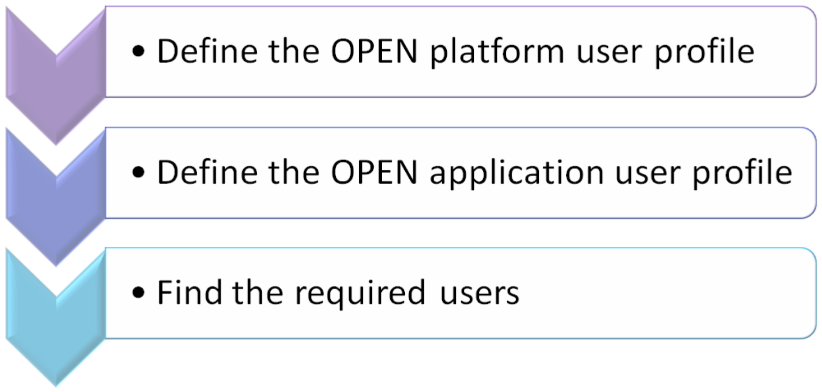

• Define the OPEN platform user profile
• Define the OPEN application user profile
• Find the required users

Figure 2: Selection process of users for OPEN usability tests

The profile of a generic user to be employed on usability evaluations of the OPEN platform includes the following characteristics (they are valid for each tested prototype):

- Technology: select participants that are familiar with used devices. It is quite important that every user is already able to use the devices where the tested applications are migrated, because we need to individuate the OPEN platform usability problems, and not the problems related to the devices capabilities or to their learnability.
- Have not been in contact with OPEN project before. The users should not be influenced by any previous knowledge, prejudice and bias gained through contact with the project.

- Are not friends/family/acquaintance of the test moderator. It will not be possible to maintain a strictly neutral attitude with someone known. This requirement is valid only for usability evaluations that require an interaction between test moderator and user (i.e. it is not valid for exploratory tests).

When first contacting the candidates, it is important that the following information is presented to them and fully understood:

- Presentation: who is the person that is contacting him/her, what this person is doing and why this person needs they to perform a testing activity (what the expectations are). This information is needed to prevent 'confusing' the respondents with wrong (and maybe undesirable) information.
- Duration: how much time they have to complete the testing activity. Make clear how much time it will be necessary. It is very important to take into account, during the calculation of the required time, also possible problems that could slow down the testing activity. For long lasting tests it is necessary to take also into account some coffee breaks. This is important because, if the user is tired, or frustrated, the test results are not accurate.
- Agreement: make clear that they need to sign an agreement stating that:
  - The sessions will eventually be audio- and/or video recorded but will only be used for internal purposes to OPEN project.
  - Any ideas that will be shown during the interview to the respondent must be kept confidential.

To make it possible to compare the data in a reliable way there must be a minimal number of respondents for each user group. For the usability questionnaires compilation that is performed during the exploratory study a number of users between 10 and 20 is required, in order to have a representation of a homogenous target.

For assessment and validation tests, instead, the minimum number of respondents required for end-users are 4 per each user group. In this way, using an informal approach, it is possible to identify 80% of the usability issues and probably all of the severe ones [HUT]. This result is considered acceptable for the OPEN project because it represents a good compromise between the obtained test accuracy and the required effort (i.e. a further increase of the test accuracy couldn't justify the required extra effort).

2.2 Exploratory Usability Study

The objective of the exploratory usability study is to get an evaluation about a software design at the beginning of the development cycle.

As foreseen in the D6.1 (Usability criteria for project phases: use cases selection, design, development, test and deployment), the first stage of usability testing is based on the output of preceding deliverables and proposed prototypes. There are some key deliverables and prototypes to be taken into account for an effective usability evaluation. For each of them, a draft of test plan has been elaborated to be then compiled in collaboration with the deliverable work team. Finally, it has been distributed to all OPEN partners for comments.

Applications described in the D5.1 (Initial application requirements and design) are very important for this study. In this case, Vodafone team, working together with Arcadia for the Social Game and with SAP for the Emergency Prototype, respectively, will deliver a questionnaire to be presented to target users of both these two services in order to preliminarily evaluate their usability. The first point to define is what to present to the responders; in this case, some screenshots and a brief description of the application have been used in order to have an immediate rendering of it, so that users can provide their opinions.

Moreover, a usability study about the OPEN project deliverable D1.1 (Requirements for OPEN Service Platform) is performed. This study is an analysis of the deliverable using a user-centred-design optic, in order to verify that during the drafting of such a requirements document the usability towards final users has been taken into account in the proper way. The following aspects are considered during such analysis: the method used for scenarios and requirements proposition, the method used to reach an agreement about proposed scenarios and requirements, the obtained requirements and scenarios.

Through this brief analysis we also want to define some guidelines for the D1.3: this deliverable, in fact, will finalize the requirements to be followed and respected during the implementation of the OPEN platform.

2.3 Assessment Usability Test

This kind of usability evaluation will be performed on OPEN prototype applications, in order to get some information about their usability level. Moreover, some suggestions from applications users will be used in order to guide the following of the development activity.

Only an informal evaluation will be performed, without any measure about the user's behaviour. This is because at this stage, a complete and integrated version of proposed applications is still unavailable and qualitative evaluations, together with discussions with users, will be more useful than tasks completion times and user's error rates in order to get some improvement suggestion. For this reason, during this test phase, a very important role will be held by the test moderator.

For every tested prototype, there are two aspects that will be evaluated: the application adaptation (when the application migration is performed between devices with different characteristics) and the migration usability.

At the end of the assessment usability test, a careful analysis of the testing results will be performed. For each application, a list of weak and strong points will be individuated. Moreover, every suggestion brought forth by users during the talking with the moderator will be communicated to the development team. In this way, a set of usability requirements will be individuated to be implemented for the validation usability test.

In this section a general methodology is described for assessment usability tests in the OPEN project. However, during the application of the methodology to tested prototypes, it could be slightly modified, in order to adapt it to the specific characteristics of each prototype. .

2.3.1 Test Moderator Characteristics

The test moderator (or test administrator) is the most important component of the testing team. He is ultimately responsible for all preparations including test materials, participant arrangements, and coordination of the testing activity [HUT].

During the execution of the task list, the test moderator is the only person allowed to interact with the end user. The moderator must not help the user to complete her/his tasks, but she/he can only interact with her/him in order to understand her/his impression and her/his difficulties. Moreover, she/he must take notes about the user behaviour, her/his errors or her/his problems. Even if the testing session is audio / video registered, it is important to take notes of the user behaviour because the analysis of registrations could take a great amount of time.

During the testing activity, if a hardware or a software problem makes it impossible for the user to complete a task, or if the user is particularly slow during the task list execution, the test moderator is allowed to modify the test plan. However, a change on the task list can be performed only for some exceptional cases and when there is no alternative solution available.

At the end of the task list execution, the test moderator must direct the debriefing session, in order to inspect carefully any problem previously noticed. During this phase, if some observers attended the test session, they are also allowed to discuss with the user.

The test moderator must not be a developer of the tested application, in order to not interfere (even unintentionally) to the test result.

During the assessment usability evaluation, the interaction between the user and the test moderator is very important. In fact, there should be a continuous interaction between moderator and user, in order to take an insight of how the application user interface is perceived. The "**thinking aloud**" method will be employed. The user, therefore, during the test execution must be encouraged to talk about his impressions and about his thoughts. It is not simple, for a user, to describe every performed action, thus the moderator must encourage her/him to share her/his point of view. Sometimes, there are users that are not able to talk continuously during every task execution, in particular when it is a quite complex action. For this kind of users, the test moderator must continue asking questions about their impressions, but without insisting when the answer is a little concise. The risk is that the user, talking with the moderator, is a little distracted from her/his principal activity (i.e. the task list execution).

When the thinking aloud method is employed, it is not possible to use the tasks execution times as a usability evaluation measure, since some users can be slowed down by the talking.

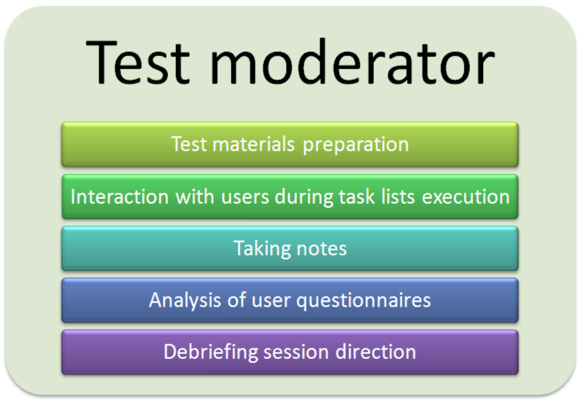

Test moderator
Test materials preparation
Interaction with users during task lists execution
Taking notes
Analysis of user questionnaires
Debriefing session direction

Figure 3: Main tasks performed by the test moderator during assessment tests

2.3.2 Application Adaptation

The OPEN platform allows the user to use applications on several devices (e.g. personal computers, mobile phones, etc.). In order make it possible to perform the same actions on devices with different capabilities, an application adaptation is performed. A usability evaluation of this aspect is needed because it strongly affects the user experience.

The usability test of the application adaptation is a complex problem, because there is not a single application to evaluate, but a mechanism for the application UI generation and, for some applications, for the logic reconfiguration (if the result of reconfiguration is reflected in the user experience).

In order to achieve an accurate evaluation of the OPEN platform, a **comparison** usability test will be performed. This type of test is used to compare two or more versions of an application. A comparison test can be performed at every level of the application development cycle. Comparison tests are usually performed in order to compare the developed product with the one offered by a competitor or to compare the new version of a product with the previous one. For each tested OPEN prototype a usability comparison test will be performed between the original version (usually running on a PC) and the adapted version (running on a different device or with a different configuration).

In the OPEN platform the different versions of tested applications are running on different devices, with different capabilities. For this reason, it is necessary to individuate , if each usability problem is related to the adaptation performed by the OPEN middleware or to the device capabilities (e.g. if the user has a problem using a touch screen device, this problem could be related to the device).

However, the adaptation performed by the OPEN platform should compensate for some limitations of the devices and the user must be able to complete the assigned task list on every device.

The user, in a comparison test, should evaluate two versions of the same product, in order to individuate usability weak and strong points in each application.

Indeed, it is often impossible to state which version is the best one, but each of them will have some positive and some negative aspects.

When the user starts using an application and s/he completes her/his tasks, s/he also learns how to use the application. So, when s/he starts using the second version, s/he can be helped by this learning (for example when tested versions are very similar) or, in some cases s/he can encounter more problems (for example, when tested versions are very different and the user thinks that assigned tasks can be completed on the second version in the same way s/he did on the previous one, s/he is misled).

In order to get more accurate results, it is necessary to have two different groups of users (say group A, and group B). Let's suppose that the tested application can run over two devices: D1, and D2.

The users of the group A will start using the application with the device D1, they will perform a migration using the OPEN platform and then they will use the device D2. At the end of each device usage (and after a migration to the other device), they will compile a questionnaire in order to evaluate it. However, for application prototypes that require a short task list, it is preferable to compile every evaluation questionnaire at the end of the test.

The users of the group B will perform the same tasks performed by the group A users, but starting with the test of the device D2.

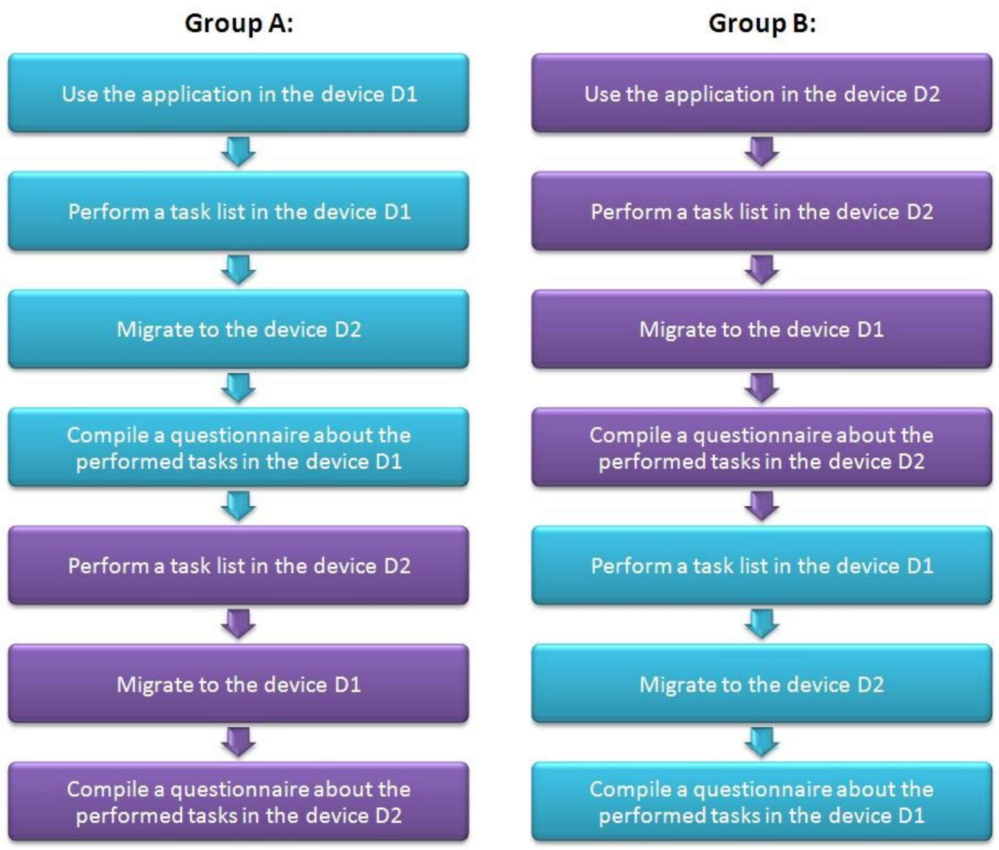

Group A:
Use the application in the device D1
Perform a task list in the device D1
Migrate to the device D2
Compile a questionnaire about the performed tasks in the device D1
Perform a task list in the device D2
Migrate to the device D1
Compile a questionnaire about the performed tasks in the device D2
Group B:
Use the application in the device D2
Perform a task list in the device D2
Migrate to the device D1
Compile a questionnaire about the performed tasks in the device D2
Perform a task list in the device D1
Migrate to the device D2
Compile a questionnaire about the performed tasks in the device D1

Figure 4: Method used in order to avoid bias problems during comparative usability tests

In this way, it is possible to identify any problem related to the learning process during the usability test.

This is the general method that will be applied during the available prototypes usability evaluation. However, some modifications will be produced to this method during its application to each prototype (for example if some features are not implemented yet).

2.3.3 Migration Process

The migration process in the OPEN platform can be manually started by the user (for example when s/he is about to leave her/his apartment s/he can migrate its application from the PC to her/his mobile phone) or automatically started by the OPEN middleware (for example, when the user is using a device with a low battery level).

The migration process must be **comprehensible** for the user, even if s/he never used the OPEN platform. It is very important that the user is able to know what application (and what application component in case of partial migrations) will be migrated and where it will be migrated. For this reason, names in human-readable format that are known by the user must be employed in order to identify applications, application components, and devices (for example, the identification of devices via IP addresses is strongly discouraged).

The effects of a migration must be **predictable**. For example, if a user is about to migrate an application, s/he must be informed about the fact that at the end of the migration the running version s/he is currently using will be terminated.

During a migration, it must not be possible to perform, by mistake, any action that will cause **data loss** for the user. Every action performed by the user about the migration must be **reversible**. For example, it the user performs by mistake a migration to a wrong device, s/he must be able to migrate again the application to the required device.

The following aspects will be considered during the migration process usability evaluation: user interface and status maintenance.

The user interface offered by the OPEN platform for the migration management must be very simple to use and it must not interfere with the usual application usage. For migrations automatically proposed by the OPEN platform, the user must be able to know what will be migrated, where it will be migrated, and why such migration is suggested (for example because a more capable device is available or because the current device battery level is too low).

At the end of the migration, a good state maintenance must be perceived by the user. She/He must therefore be able to start using the application in the target device from the point s/he abandoned it in the source device. In order to achieve a good user experience, some mechanism could be provided in order to help the user to remember what task s/he was performing and what parts of it were already completed. This is not a technical evaluation of the state maintenance, but only the feeling of the user about this aspect will be evaluated.

The migration process usability can be evaluated during the user interface adaptation testing. In fact, for every application, at least one migration for each device is needed. Moreover, in every application task list, a migration can be inserted for each migration trigger available policy (manual, or automatic with and without a confirmation prompt).

Therefore, in the usability questionnaire, the user will also be able to answer some questions about the migration process. In this way, it is also possible to check the interaction between the OPEN migration management user interface and every demo application in different devices, in order to identify different kind of problems.

During this test phase, the migration process will be still not integrated in the OPEN platform, but every tested prototype will use its own process. For this reason, the main objective of this test is to provide some usability requirements and suggestions that could be taken into account during the integration of available application prototypes in the OPEN platform.

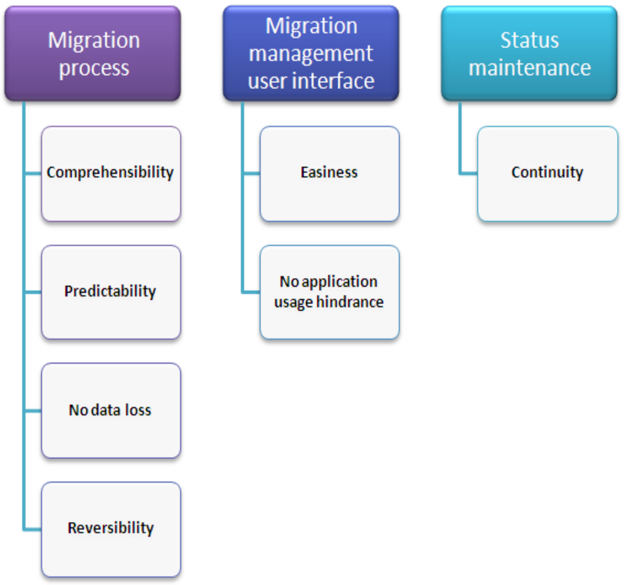


Figure 5: Migration process usability requirements

2.3.4 Execution of a pilot testing session

It is recommended to execute, before the usability test of an application, a pilot test session, in order to validate the test execution plan.

A member of the usability testing team should execute every task, in order to make sure that there isn't any problem during such operations (i.e. there are no software bugs that interfere with the task execution). Moreover, during the execution of the pilot test, it is possible to measure the execution time of every task (i.e. the time employed by a user to complete an assigned task). If this amount of time is greater than the one already communicated to the users during their recruitment (par 2.1.1), some modifications to the task list can be implemented.

However, the test plan feasibility must be evaluated before the pilot test in collaboration with the application development team. Only minor changes can be implemented in the test plan after the pilot test execution, because it is performed shortly before the test with real users.

The pilot test is also useful for the test moderator that in such way can be more prepared for the testing sessions. However, for this test phase the pilot test execution is not mandatory because an informal approach will be used during the testing activity. For tests that requires a set of measurements it is more important to perform a pilot test, in order to make sure that every parameter will be measured in an accurate way.

2.4 Validation Usability Test

The validation usability test is performed when the application development cycle is almost completed and it is more formal than an assessment test. Usually, during this test phase, only minor usability problems are individuated. Most of important usability problems, in fact, are individuated (and fixed when possible) during the assessment usability test.

During this test phase a more formal approach will be used, in order to evaluate the overall usability level of the OPEN project. Moreover, the application usability problems (for example the erroneous implementation of a usability requirement) will be found and classed according to their impact on the user experience. For severe usability problems the support of the development team will be required in order to fix them.

This section describes the proposed method for the validation usability test to be performed over the OPEN platform. This method will be slightly modified during its application for each application running over the OPEN middleware. Moreover, some additional modifications will be implemented if some platform features will be added/removed during the following of the development cycle.

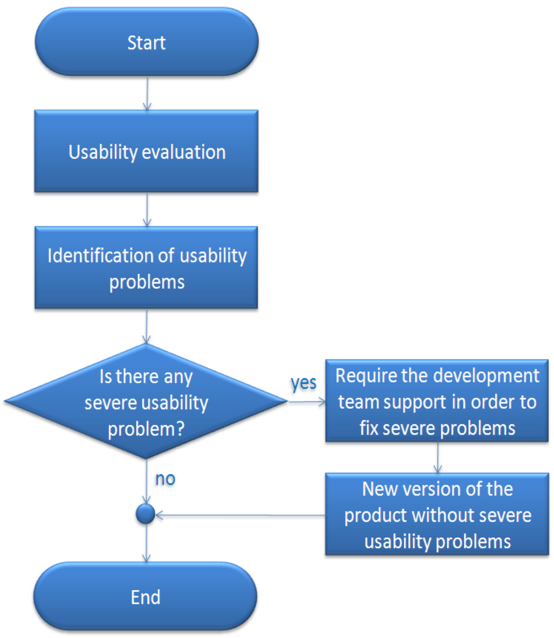

Start
Usability evaluation
Identification of usability problems
Is there any severe usability problem?
yes
Require the development team support in order to fix severe problems
no
New version of the product without severe usability problems
End

Figure 6: Flowchart of validation usability testing process

2.4.1 Test Moderator Characteristics

The role of the test moderator during this test is similar to its role during the assessment test (par. 2.3.1), with some modifications about its interaction with users.

During this usability test, in fact, besides the questionnaires compiled by users, some instrumental measures will be performed. In particular, execution times (i.e. the amount of time employed by an user to complete an assigned task) and error rates (calculated by taking into account the errors committed by the user during the application usage) will be measured during the test. For this reason, the test moderator should interact as less as possible with users, in order not to distract them from tasks execution.

Moreover, the test moderator will be responsible of all measurements that will be performed using an appropriate software tool. An alternative solution could be to manually perform every required measurement. In this case, the help of another member of the testing team could be required. In this way, the test moderator could perform the required measurements while her/his collaborator takes notes about the ongoing testing session. For long task lists an automatic tool is preferable, while for short task lists manual measurement is probably the best solution.

At the end of the task list execution, the test moderator will analyze the questionnaire compiled by the user and the performed measurements and then s/he will direct the debriefing session.

2.4.2 Application Adaptation

The method that will be used during this test phase for the application adaptation mechanism offered by the OPEN platform is the same used during the assessment test (par. 2.3.2), with some modifications. The fundamental difference is that during the validation test a more formal approach will be used and that the debriefing sessions will be probably shorter than the ones performed during the previous test phase.

Moreover, some thresholds will be defined for the maximum difference between some usability parameters (to be defined for every tested application) over different devices (such thresholds will be defined taking into account the different capabilities of the used devices). In this way it is possible to verify that the difference of usability of a application on different devices is lower than a threshold value.

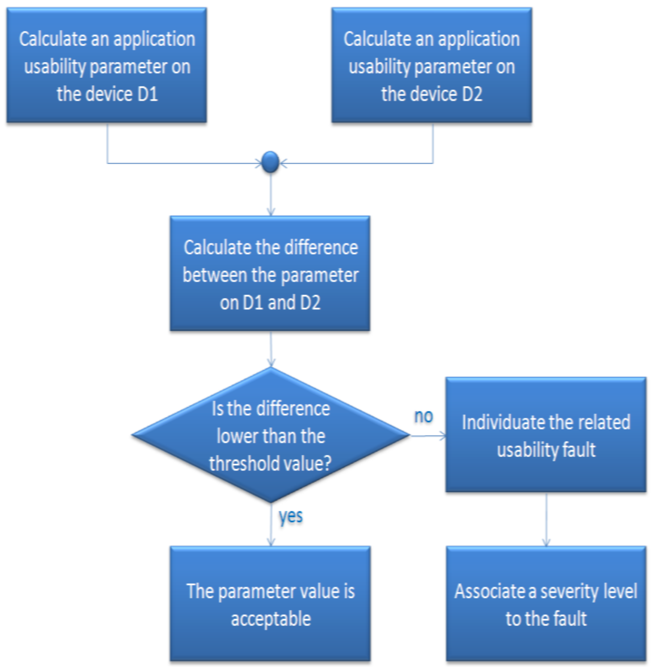

Calculate an application usability parameter on the device D1
Calculate an application usability parameter on the device D2
Calculate the difference between the parameter on D1 and D2
Is the difference lower than the threshold value?
no
Individuate the related usability fault
yes
The parameter value is acceptable
Associate a severity level to the fault

Figure 7: Flowchart of the application adaptation usability evaluation

2.4.3 Migration Process

During this test phase, it is expected that all tested applications will be integrated in the OPEN middleware. So, every migration will be performed using a complete version of the OPEN platform. This aspect and the employing of a more formal approach (with execution times and error rates measurement) are the only differences with the procedure proposed in the paragraph 2.3.3.

It is expected to obtain an excellent usability level for this component of the OPEN platform. A threshold level for some usability parameters (to be defined in the test plan of every tested application) will be used in order to identify whether the migration usability is the expected one.

2.4.4 Execution of a pilot testing session

As stated in the paragraph 2.3.4, the execution of a pilot test could be useful for every usability evaluation. Before performing validation tests of the OPEN project, it is necessary to perform a pilot test, in order to make sure that all software and hardware infrastructures are correctly configured, and that the chosen measurement tools for execution times and error rates provide accurate values. Without a pilot test session there is the risk not to get an accurate evaluation during the tests performed with the first user.

3 Programmability

As defined in the deliverable D6.2 [D6.2], programmability is the capability within hardware and software to accept a new set of variables and instructions that alter its behaviour [http://encyclopedia2.thefreedictionary.com/programmability]. In the OPEN environment, the following definitions apply:

- Variables: context information
- Instructions: rules describing the migration process behaviour depending on the context information
- Migration process:
    - Migration triggering (when to migrate) and orchestration (where, what, how to migrate)
    - Application Logic Reconfiguration
    - User Interface Adaptation

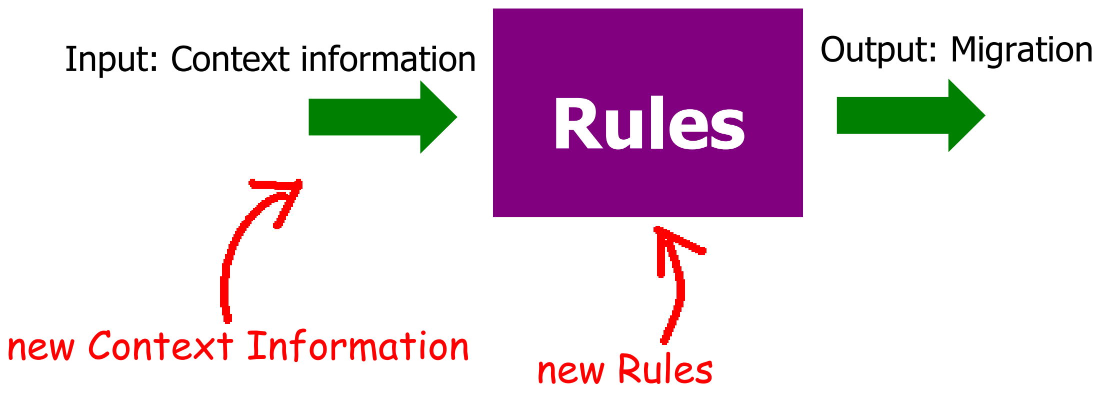


Figure 8: Programmability block diagram.

The concept of programmability (or configurability) agreed inside the consortium and described in D6.2 [D6.2] is the capability for the user of the platform (developer, platform administrator…) to define:

- new context variables (to be used for the platform behaviour definition)
- new rules (function of the context variables) describing the platform behaviour

Example:

Suppose I have a mobile phone that provides me the following information:

- Battery
- Signal strength

In the platform, migration triggering rules depending on these variables are defined.

Suppose that afterwards I have another mobile phone that can provide also the "user_location" (latitude and longitude), I should be able to:

- Instruct the middleware in order to acquire the new "user_location" variable
- Define a new rule depending on this new variable. E.g.: if "user_location"="home" then trigger the migration towards the TV.

The programmability evaluation will address the different components of the migration process, which are mapped in different middleware modules. For each module, the programmability evaluation will consider two different phases:

- Programmability Assessment
- Programmability Validation

The assessment and validation phases will be carried out for both testing iterations: the first iteration and the final iteration. The different approaches are explained in the following paragraphs.

3.1 Programmability Assessment

The first phase of the programmability evaluation process is the "**programmability assessment**". With respect to the programmability, we distinguish two different middleware tasks:

- *Context variable collection and distribution:* different middleware modules can participate in this task, as for example the Device Discovery and the Context Management. For these modules, enabling the programmability means providing the capability of managing all the available variables without constraints. E.g., a possible constraint could be: the module accepts only numerical variables: it does not handle Boolean or String; the module accepts only the variables defined in a predefined structure as for example:

```
<context variables>
        <battery>0.55</battery>
        <signal_strength>12</signal_strength >
</context variables>
```

- *Migration and adaptation rules definition:* different modules apply rules for migration triggering and orchestration, application logic reconfiguration and user interface adaptation. For these modules, enabling the programmability means providing some facilities to define the module behaviour depending on the available context information.

It could be useful to map these two tasks in the previous example, in which the platform should:

- Instruct the middleware in order to acquire the new “user_location” variable (this aspect will be evaluated in the scope of “context variable collection and distribution”)
- Define a new rule depending on this new variable. E.g.: if "user_location"="home" then trigger the migration towards the TV (this aspect will be evaluated in the scope of “migration and adaptation rules definition”).

The aim of the assessment phase is, for each OPEN middleware module, understanding which of the previous programmability aspects it should address or if it should address both, and depending on it, understanding if appropriate facilities are available.

In order to carry on this phase, the following template is proposed. Depending on the specific module that will be analysed, small changes can be done on the template in order to fit the specific needs of the related evaluation. The template will be compiled for each module during the first testing iteration and then revised during the second iteration, in order to incorporate the possible evolution.

Table 1: Programmability Assessment template.

Title:	OPEN Programmability Assessment *moduleName*		
ID:	OPEN Programmability_ *moduleName_x*		
Version	**Issue**	**Date**	**Author**
			Module owner/Vodafone team
Module name *moduleName*			
General considerations	*[to be filled by module owner]* *Please specify if the module enables the programmability in its current implementation related to both programmability aspects:* • *context variable collection and distribution* • *migration and adaptation rules definition*		
Reference prototypes	*[to be filled by module owner]* *List of prototypes using the specified modules*		
Synthetic description	*[to be filled by module owner]* User: *(developer, system manager, service provider…)* Supported context variables type: *(int, double, boolean…)* Manageable variables: *(only predefined, all variables that*		

	can be represented by an integer number, all…) Available tools for configuration *(only for modules addressing migration and adaptation rules definition)*: • *configuration file* • *graphic tool* • *workflow definition tool* • *other (specify)* Workflow patterns supported *(only for modules addressing migration and adaptation rules definition)*: • *Sequence* • *Parallel split* • *Synchronization* • *Exclusive choice* • *Simple merge* *(Refer to paragraph 3.1.3 for workflow patterns description)*
Context variable collection and distribution	*[to be filled by module owner]* *Only for modules addressing context variable collection and distribution.* *Please describe how the module is able to collect and make available context variables to other modules*
Language/tool available for the module behaviour description	*[to be filled by module owner]* *Only for modules addressing migration and adaptation rules definition.* *Please describe how the user can define the rules for module configuration*
Parametrical evaluation	*[to be filled by Vodafone evaluation team]* *For modules addressing context variable collection and distribution:* • Extensibility (capability of accepting and managing new variables): 1..5 *For modules addressing migration and adaptation rules definition:* • Conciseness (capability of specifying the module

	behavior in a synthetic way): *1..5* o weight: 1 • Fulfillment (capability of specifying the required workflow patterns): *1..5* o weight: 3 • Usability(*) (usability of the provided tool): *1..5* o weight: 2 *Note: consistency, runtime efficiency and robustness will be evaluated in the programmability validation phase*
Qualitative evaluation	*[to be filled by Vodafone evaluation team]* *A qualitative evaluation of the provided facilities will be provided*
Synthetic description of the adopted verification and validation strategies	*[to be filled by Vodafone evaluation team]* *In this field the objectives of the programmability validation phase will be highlighted and a brief description of the test cases will be provided. Log files or stub methods eventually required will be defined. The validation phase will take into consideration the following parameters:* • *Consistency* • *Robustness* • *Runtime efficiency*

(*)The evaluation will provide a qualitative analysis of the usability of the provided tool for the programmability. Since the user of this tool is not the end user but one of the actors involved in the service development (developer, system manager…) this evaluation is not part of the usability assessment.

In the "**Title**" should be indicated the name of the module, the "**ID**" field should be used to distinguish between the evaluation iteration performed for the first iteration prototype and the final evaluation. A different "**version number**" should be given to the document if substantive changes to the contents of the document have been made. Different "**issue**" numbers within a given version indicate minor changes such as spelling and grammatical corrections.

The synthetic description of the information expected for each field is described in italic. In the following paragraph the detailed description of the required information for each field is provided.

3.1.1 Programmability assessment – first iteration

The first iteration of the programmability assessment will be carried on taking into account the modules defined in D4.2 [D4.2], in order to put the basis for the final evaluation and to provide useful feedback for the development activity. As stated in the previous paragraph, the aim of the assessment phase is understanding, for each OPEN middleware module, which programmability aspects it should address, and consequently, understanding if appropriate facilities are available. In order to collect all the required information for this analysis, the proposed template should be filled.

The "**General considerations**" field must be filled by the module owner, who will specify if the module enables the programmability in its current implementation. As already stated, the module could enable the programmability with respect to:

- *Context variable collection and distribution:* the module enables the programmability with respect to the context variable collection and distribution if it supports capability of setting new variables. One of the modules that should enable this kind of programmability is the Context Management Framework. E.g.: the context information is provided by the mobile phone, which communicates to the Context Management Node the battery level and the signal strength. This information is mapped in two variables in the Context Management Node. Supposing that another mobile phone has also the location information based on GPS, the Context Management Node should be able to allocate a variable for this information. Allocating a new variable is not enough, because in order to use this variable for applying a specific logic, it is necessary to be able to trace the variable meaning.
- *Migration and adaptation rules definition:* the module enables the programmability with respect to the migration and adaptation rules definition if it supports capability of setting new rules. One of the modules that should enable this kind of programmability is the Application Logic Reconfiguration. E.g.: the programmability of the Application Logic is the capability of imposing rules depending on context information that define the application logic reconfiguration behaviour.

If the module does not support programmability, in this current implementation, there are two different ways of proceeding, depending on the reason why it does not support programmability:

- The programmability support is not required. In this case the module owner should specify why enabling the programmability is not a requirement for the module. <u>All the subsequent fields of the template will not be filled.</u>
- The programmability support is required but not still implemented yet. In this case:
  - o If the programmability approach has been already defined but not implemented, the module owner should specify the designed

approach and the subsequent field of the template will be filled providing information related to this solution.

  - If the programmability approach has not been already defined, the module owner should specify why. All the subsequent field of the template will not be filled.

The "**Reference prototypes**" field must be filled by the module owner listing the first iteration prototype using the specified module.

The "**Synthetic description**" field must be filled by the module owner and should provide the following basic information:

- User: who is the user of the provided facilities for programmability? (developer, system manager, service provider…)

- Supported context variables type: which kinds of variable are supported? (int, double, boolean…)

- Manageable variables: how many variables can be supported? (only predefined variables, all variables that can be represented by an integer number, all variables…)

- Available tools for configuration (only for modules addressing rules definition):
  - configuration file
  - graphic tool
  - workflow definition tool
  - other (specify)

- Workflow patterns supported (only for modules addressing migration and adaptation rules definition):
  - Sequence
  - Parallel split
  - Synchronization
  - Exclusive choice
  - Simple merge

The "**context variable collection and distribution**" field must be filled only for modules addressing this aspect of programmability. The module owner should describe how the module is able to collect and make available to other modules context variables.

The "**Language/tool available for the module behaviour description**" field must be filled only for modules addressing migration and adaptation rules definition. The module owner should describe how the user can define the rules for module configuration.

The subsequent fields will contain the programmability evaluation of the module given the information provided by the module owner in the previous field. This evaluation will be carried on by the Vodafone testing. These fields will be completed after the evaluation activities.

The "**Parametrical evaluation**" field will provide a first quantitative evaluation of module programmability providing a value between 1 (poor) and 5 (excellent) of the following parameters. For each identified parameters, a weight between 1 (low importance) and 3 (high importance) has been defined in order to underline that different parameters have different influence in the programmability evaluation: e.g. the Extensibility has weight "3" because the capability of accepting and managing new variable is very important for enabling the module programmability, while the Conciseness (capability of specifying the module behavior in a synthetic way) has weight "1" because it is less relevant. The quantitative evaluation will assign a value to the parameters listed below depending on the information provided in the previous fields.
The parameter identified for modules addressing context variable collection and distribution is:

- Extensibility (capability of accepting and managing new variables): 1..5
    - weight: 3

The parameters identified for modules addressing migration and adaptation rules definition are:

- Conciseness (capability of specifying the module behavior in a synthetic way): *1..5*
    - weight: 1
- Fulfillment (capability of specifying the required workflow patterns): *1..5*
    - weight: 3
- Usability (usability of the provided tool): *1..5*
    - weight: 2

The "**Qualitative evaluation**" field will be filled with a first qualitative evaluation of the provided programmability facilities.

The "**Synthetic description of the adopted verification and validation strategies**" will be filled only if the module is currently used by one of the first iteration prototype and therefore the validation phase of the programmability evaluation will be carried on for the module. In this field the objectives of the programmability validation phase will be highlighted and a brief description of the test cases will be provided.

3.1.2 Programmability assessment – final evaluation

The final evaluation of the programmability assessment will be carried out taking into account all the modules responding to the following criteria:

- Modules that in the first iteration already have been indicated as enabling the programmability, in order to verify that the feedbacks provided during the first iteration of the programmability evaluation have been incorporated where possible.
- Modules that in the first iteration were indicated as not enabling the programmability but for which the programmability supports were required, in order to verify if enhancements have been done.

The final evaluation of the programmability assessment will not involve modules that have been classified during the first iteration as modules that do not require programmability facilities. During this evaluation, the proposed template will be recompiled for the selected modules.

The "**General considerations**" field must be filled by the module owner, who will specify if the module enables the programmability with respect to:

- Context variable collection and distribution.
- Migration and adaptation rules definition.

If the module does not support programmability the module owner should specify the reason why. <u>All the subsequent fields of the template will not be filled.</u>

Refer to 3.1.1 for the other field's compilation.

3.1.3 Workflow patterns

In D4.1 [D4.1] the Workflow Patterns [WP] have been introduced as tool for evaluating the various perspectives that need to be supported by a workflow language or a business process modelling language of an orchestration tool. However, in the OPEN project, it can be used to understand the capability of the specific tool provided for the module programmability of specifying dependencies between various tasks (e.g. parallelism, choice, synchronization etc). This analysis can be applied to:

- Middleware modules implementing the migration triggering and orchestration
- Modules for the UI adaptation
- Modules for the Application Logic Reconfiguration (for both orchestration and wiring approach)

As for Workflow patterns, various perspectives can be distinguished:

1 The control-flow perspective depicts aspects related to dependencies between various tasks (e.g. sequence, parallelism, etc.). Originally, the

Workflow Pattern Initiative proposes twenty patterns, but in the latest iteration this has grown to over forty patterns.

2 The data perspective aim to capture the various ways in which data is represented and utilised in workflows (passing of information, scoping of variables, etc.)

3 The resource perspective deals with resource to task allocation, delegation, etc.

4 The exception handling perspective aims at defining the different causes of exceptions and the actions that need to be taken as a result of exceptions occurring.

In the programmability assessment, only the control-flow perspective is considered, in order to understand the capability of the specific tool provided for the module programmability of specifying dependencies between various tasks (e.g. parallelism, choice, synchronization etc). As previously mentioned, the control flow patterns are more than forty: in order to simplify the analysis, the Basic Control-flow patterns have been selected. This class of pattern captures elementary aspects of process control. The capability of representing these patterns enables a satisfying level of programmability. The basic Control-flow patterns are:

- Sequence
- Parallel split
- Synchronization
- Exclusive choice
- Simple merge

Although the workflow patterns have already been described in D4.1 [D4.1], it is relevant to add some information in order to explain their use for the programmability assessment:

- A brief explanation is provided, also through some simple examples not directly related to the methods/tasks implemented by the actual OPEN middleware modules.
- The evaluation criteria to be used for determining if the provided tool/language enables the use of the specific pattern.

Following the Basic Control-flow patterns description:

- **Sequence:** a task in a process is enabled after the completion of a preceding task. E.g.: the “verify account”(B) task executes after the “obtain credit card details”(A). The Sequence pattern is an essential building block for processes. It is used to construct a series of consecutive tasks which execute in turn one after the other. <u>Evaluation criteria</u>: support for this pattern is demonstrated by any tool/language which supports a representation of dependency between two tasks. E.g.: for evaluating if the Application Logic Reconfiguration module supports this workflow pattern, we can try to answer to the following question: after the rewiring, is the module able to execute tasks in sequence? If yes, the rewiring is able to represent this workflow pattern (e.g.: depending on the context variable x, the components A and B are wired. A execute the task "a" and then B execute the task "b")

- **Parallel Split:** the divergence of a branch into two or more parallel branches each of which execute simultaneously. E.g.: when a migration trigger is received (A), triggers the "retrieve state"(B) task and the "retrieve device information" task simultaneously. The Parallel Split pattern allows a single thread of execution to be split into two or more branches which can execute tasks concurrently. Evaluation criteria: support for this pattern is demonstrated by the provision of an implicit or explicit construct allowing the thread of control to be split into two or more concurrent branches.
- **Synchronization:** the merge of two or more branches into a single consequent branch: the thread of control is passed to the task immediately following the synchronizer once all of the incoming branches have completed. E.g.: The "start migrated application"(C) task runs immediately after both the "migration check"(A) and "receive state"(B) tasks are completed. Evaluation criteria: support for this pattern is demonstrated by any tool/language providing a construct which allows the convergence of the execution threads of two or more parallel branches in one task.
- **Exclusive choice:** the divergence of a branch into two or more branches: when the incoming branch is enabled, the thread of control is given to one of the outgoing branches based on a rule that can choose one of the outgoing branches. E.g.: Depending on the value of context information "x"- evaluated by the "check x"(A) task -, either the "trigger partial migration"(B) or "trigger total migration"(C) task is initiated. Evaluation criteria: support for this pattern is demonstrated by the provision of a construct (either implicit or explicit) that allows the thread of control at a given point in a process to be defined depending on a specific condition.
- **Simple Merge:** the merge of two or more branches into a single consequent branch: each enablement of an incoming branch results in the thread of control being passed to the consequent branch. E.g.: At the conclusion of either "trigger partial migration"(A) or "trigger total migration"(B) tasks, a "notify migration"(C) task is started. Evaluation criteria: support for this pattern is demonstrated by any tool/language providing a construct which allows different execution threads to have the same subsequent task.

The exhaustive description of the workflow patterns is out of the scope for this document, a complete description can be found in (Aalst et al., 2007) [Aalst04].

3.2 Programmability validation

The programmability validation will be carried out taking into account all the modules responding to the following criteria:

- Modules that in the programmability assessment were indicated as enabling the programmability
- Modules present in at least one available prototype

While the Programmability Assessment provides a theoretical evaluation of the module programmability, during the Validation phase the described module features will be verified and evaluated through some measurable parameters:

- Consistency: is the module behaviour compliant with the rules set using the tool/language selected? (yes/no).
- Robustness: how much is the module affected by errors in defining the module behaviours? E.g.: if there is an error in a parameter value is the module able to limit the error consequences?
- Runtime efficiency: this parameter will evaluate if the runtime efficiency is affected by changing the module configuration, e.g.: the module has different runtimes for the standard configuration and for different configurations. The runtime efficiency could be evaluated if specific measuring tools are available.

During both the first and the final iterations, for each module satisfying the previous criteria, an exhaustive set of test cases should be defined, in order to validate the correct behaviour defined for the different modules. The test cases definition should be carried on by both module owners and Vodafone testing team.

The template that will be used for these test cases is the following.

ID	*Univocal identifier of the test case*
Module	*Related module*
Description	*Objective of the test case*
Input	*Input provided to the module*
Expected output	*Expected output in terms of module behaviour*
Actual output	*Output obtained*
General considerations	*Comments derived by the test result*

For each module, the test result will be shared using the following test report:

Title:	OPEN Programmability Test Report *moduleName*	
ID:		
Version	**Date**	**Comment**

Test Case ID	Description	Status (pass, failed, fixed)	Actual behaviour	Severity
1				
2				
3				
4				
5				

3.2.1 Programmability validation – first iteration

The validation for the first iteration of the programmability evaluation will involve:

- the modules that in the programmability assessment-first iteration, were indicated as enabling the programmability
- modules present in at least one available prototype

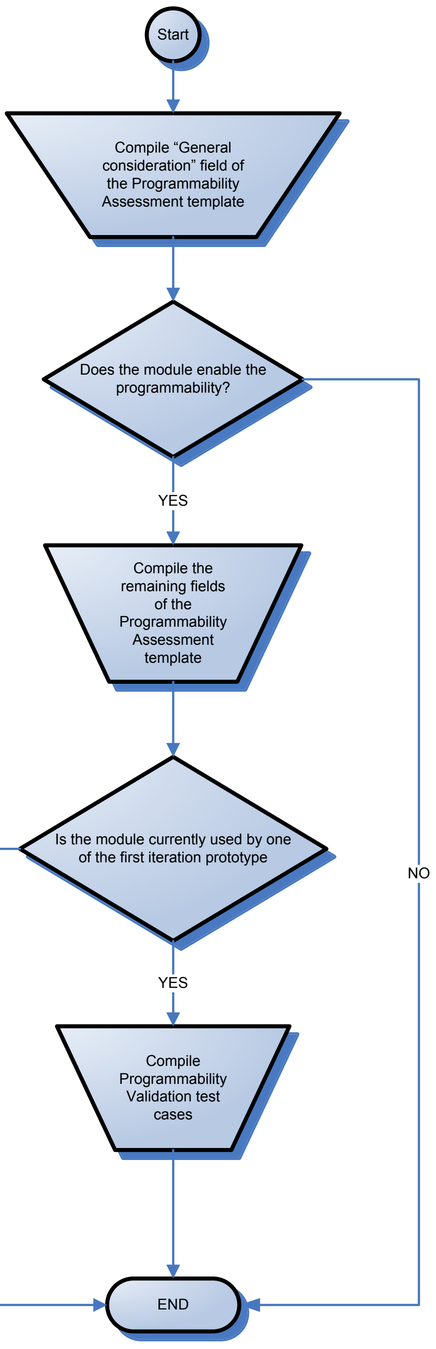
3.2.2 Programmability validation – final iteration

The validation for the final iteration of the programmability evaluation will involve:

- the modules that in the programmability assessment-final iteration were indicated as enabling the programmability
- modules present in the final prototype

3.3 Partners' contribution-first iteration

In the following workflow diagram, the contribution for the programmability evaluation requested from each module owner is depicted. It is foreseen that most of the evaluation in the first iteration will stop after the first activity (Compile "General considerations" field of the Programmability Assessment template) because most of the first iteration prototypes do not support programmability in their current implementation.



Start
Compile “General consideration” field of the Programmability Assessment template
Does the module enable the programmability?
YES
NO
Compile the remaining fields of the Programmability Assessment template
Is the module currently used by one of the first iteration prototype
YES
NO
Compile Programmability Validation test cases
END

Figure 9: Programmability evaluation: partner’s contribution.

4 Technological evaluation

The previous deliverable D6.3 [D6.3] defined the common parameters used to perform a technical evaluation and described how to realize a translation of them for testing purposes.

In particular, some specific requirements for the OPEN project were added in that deliverable, as they were considered as useful to be verified; but especially this document constitutes the basis for the test plan concerning the technical evaluation, thus shaping the realization of it and the fields to be considered and agreed among all the OPEN project partners.

Two testing timeframes have been scheduled: the first experience (currently foreseen between M15 and M20 of the OPEN project) acts as an input for a following phase of software development, aimed at correcting technical issues that will arise; this choice has been made, since technical aspects are primary for the outcoming prototypes from the project, and they can be a future proof for the final development. After that, another evaluation stage will definitively demonstrate the technical solidness of OPEN platform, in order to complete the final report by M28.

Therefore this document, for the sections concerning the technological evaluation, has the scope of:

1. Clarifying the overall methodology that will be strictly observed for the technical testing of the various prototypes within the OPEN project
2. Describing in detail the test plan (whose format has been defined in D6.3) for each first iteration prototype to be submitted to the technical evaluation
3. Putting the basis for further work, regarding both the collection/evaluation of the results and the organisation of the second testing iteration

4.1 Methodology for technical testing

The indicators to be analyzed for an effective technical evaluation have been defined in the D6.3 that followed two testing paths.

The first (and more complex) path is the evaluation and measurements of a set of parameters (mainly taken from the Description of Work for the OPEN project). This is the most complex step, because for each prototype these indicators can wear different meanings, and the analysis will not provide a pure success/fail output. The evaluated indicators and the measurements have to be compared with the expected results, with a subsequent check among the OPEN partners. Furthermore, the analysis should be repeated for each migration scenario (from a device to another and back, to verify possible degradations), when this is allowed from the prototype.

The second path is based on a set of specific OPEN requirements, elicited in the D1.1; they will be verified for each prototype, thus producing a clear outcome, since they have a Yes/No format: they are, in fact, functional requirements concentrating on *what* the system should do, while the previous indicators focus on *how well*. The requirements are surely a valuable evaluation benchmark: during the drafting of D1.1, they have been defined through consolidated and reliable methods (VOLERE, Ben Achour, etc.) and are specific for the OPEN platform and the applications embedded, so they can verify the fulfilment of the objectives of this project.

These approaches have to be adapted to the first iteration prototypes, with the goal of making sure that the software system to be tested fulfils the expectations about the indicators and the functional requirements. This could be a complicated process, if a complete testing of all software modules involved in the OPEN system is considered. However, a trade-off is needed, since it is not possible during an efficient evaluation to verify that the system responses as it is designed to do given every possible combination of inputs and resulting outputs, while an exhaustive testing would be required in order to test all logical execution paths.

Therefore, a practical goal for software testing would be to maximize the probability of finding errors using a finite number of representative test cases, to be executed with the minimum effort: this is why simulations that foresee the behaviour of OPEN platform in high load scenarios have been considered in the D6.3 as being out of scope for this technical testing experience.

An example of such an approach can be identified in the performance indicators described in the D6.3: the separation of performance measurement among different functional elements allows identifying the source of possible issues, failings, bottleneck, and so on.

This will be very useful for these first prototypes, which don't completely reproduce the end to end OPEN behaviour, but currently focus on particular features/modules: since the whole system is not still mature and available at this time, the evaluation approach (for these prototypes) will not be a completely integrated testing.

However, it will be more than a pure *module testing.* In fact, according to the British Standards Institution definition, "Module testing, also known as unit or component testing phase, is concerned with the testing of the smallest piece of software for which a separate specification exists": this meaning should be enriched basing on the technological scopes at this point (with some possible exceptions depending on the particular element /product to be tested).

So the first iteration will give focus to the prototypes, starting from some building blocks that combine individual software modules and testing them as a group. During the second stage, on the other hand, a complete integration testing will be feasible, in order to arrive to the overall product/service (this topic would be enlarged when talking about partial/system integration).

Another key point to underline concerns the difference between "*white box*" and "*black box*" testing (these are common terms from the mathematical modelling theories), of course in relation with the purpose of such an evaluation:

White box testing (a.k.a. clear box testing, glass box testing, transparent box testing, translucent box testing and structural testing) uses an internal perspective of the system to design the analysis and the test cases [WBDef]. It requires programming skills from the tester to identify all paths through the software: the inputs are chosen to especially verify paths through the code, determining the proper outputs. This means for example that if the implementation changes, the tests (based on the current one) probably will need to change, too. So a white box testing is finally more suitable to a debugging phase from the developers, and it will be considered out of scope for the technological evaluation in the OPEN project.

Black box testing, on the other hand, takes an external perspective of the product/service to derive test cases. These tests can be functional or non-functional so, again using a previous clear definition, they can verify both *what* the system should do and *how* [BBDef]. The testing design phase aims to select valid and invalid inputs and determines the correct output, without needing a strict knowledge of the test object's internal structure. The dimensioning of this *black box* is applicable to different levels of testing: the higher the level, and hence the bigger and more complex the box, the more one is forced to use such a testing to simplify, even if one cannot be sure that all the possible paths are tested (but this method can uncover parts of the specification finally unimplemented). Technological evaluation within the OPEN project will be completely referable to the black box approach.

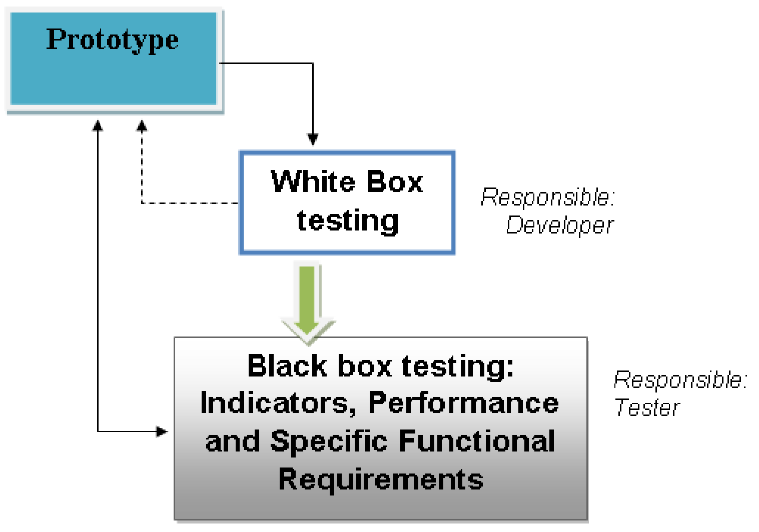

Prototype
White Box testing
Responsible: Developer
Black box testing: Indicators, Performance and Specific Functional Requirements
Responsible: Tester

Figure 10: Separation between white box and black box testing for OPEN testing

After the first iteration and the following timeframe of development, a new phase of testing will aim not to only verify the way of working of each module, but it will evaluate the level of integration [IDef] reached within the OPEN platform and its components (this recalls the so-called *bottom-up* approach, since the analysis starts from lowest levels of integration, to be then incremented). About this point, such an evaluation can imply **partial integration** or **system integration** testing:

Partial integration acts again as a black box testing, but this time it doesn't observe the level of the single module; the areas being involved in the evaluation cover several modules, through a product/service related to this set. This means that with a finite number of areas, all the modules are tested while being integrated with other ones (but not all the possible ones).

System integration is the final complete validation of the product/service the project aims at, in which all the modules are tested while contributing to it and being integrated each other. So the testing process exercises the system's coexistence with all the others.

Since the objective of the OPEN project is to realize and validate the complete migratory platform, enriched through the two final target applications, the second iteration should aim at a system integration; partial integration is somehow included in the first stage prototypes, and in the second test experience it will be previously verified only in case of specific exceptions, such as prototypes that need it, according to a clear indication from the developers. So the structure of the complete evaluation process will be the following:

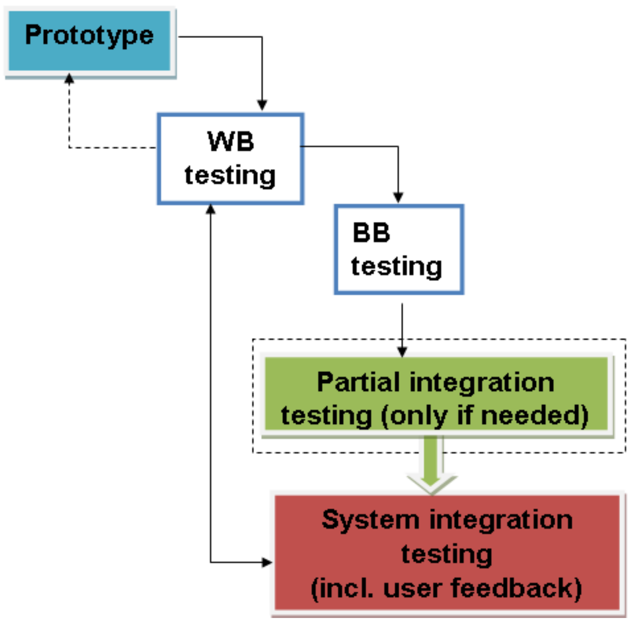

Prototype
WB testing
BB testing
Partial integration testing (only if needed)
System integration testing (incl. user feedback)

Figure 11: Level of integration in the different phases of the testing process

Further sections of this document will define the template for test cases definition, which prototypes (among the ones from the first phase) will need a technological evaluation, and especially why this can give added value to the project; but before of these specific topics, it is necessary to first define how the generic indicators or requirements presented in D6.3 could be adapted to a particular prototype.

The main point of this adaptation, from previous WP6 deliverable to the test plan, concerns the general indicators, since the same process is going to be very simple for the specific functional requirements (Section 3 of D6.3): this set needs only to be shaped to each prototype, cutting the requirements that don't concern it at all and inserting the others that are applicable in the prototype test list. The summary of indicators and requirements can be found in the Appendix D.

4.1.1 Template for test cases definition

Internally to each prototype test plan, a test list will be defined, basing it on some test cases. Since tests within the OPEN project cover not only technical evaluation, but also other areas, a common template for the test case definition will be used where possible for these test experiences, being then shaped on the particular necessity for the specific test. This simple template, as seen in Chapter 3, can satisfy these needs for both the technological and the programmability evaluation:

ID	
Module	
Description	
Input	
Expected output	
Actual output	
General considerations	

Figure 12: Generic template for test cases definition

In fact, this is a typical template for black box testing, in which we have only a description of what should be done and with what result: the template doesn't mind internal configuration of the test object, but only what results are achieved and eventually how. Of course this template can be enlarged or modified for the specific prototype/test case, to maintain a stricter adherence to the testing purposes.

4.1.2 Test plan

A test plan format has been defined in the D6.3: it is made of eighteen different sections to totally shape and lead the execution of the technical evaluation. Such a test plan is oriented to the prototype, since this is more suitable to verify *if* and *how* a product/service works; the modules involved in each prototype can be used to underline the possible origin for a fault or an unsatisfying indicator.

The format of D6.3 follows, this time it has been structured in a table that underlines for which section a contribution is needed from the developers of the prototype to reach an agreement on the testing procedures:

Test plan Identifier: *It is a unique way to refer as to the test plan, related to the ongoing project*	OPEN Technological test plan *prototype Name*		
Version	**Issue**	**Date**	**Author**
			Prototype owner/Vodafone testing group
Prototype name *Prototype name*			
References	*[by tester]* The set of documents within the ongoing project to which the test plan refers		
Lesson learned from previous experiences	*[by tester]* If testing is divided in more phases, each phase can contribute to modify the approach during the following, both at high level and in detail		
Test items	*[by tester]* It is a categorization of the whole testing evaluation in different areas *[by prototype owner]* Analysis of which indicators/requirements are feasible to test		
Risk	*[by tester]* Potential risks, which must imply mitigation actions *[by prototype owner]* Eventual risk from direct experience on the prototype		
Features to be tested	*[by tester]* In-scope functionalities *[by prototype owner]* Support for test cases definition and validation		
Features to be not tested	*[by tester]* Out of scope functionalities, so the other indicators and requirements, not listed in the previous section.		
Approach	*[by tester]* Describes the cooperation of partners executing the evaluation and their role/actions during testing *[by prototype owner]* Agreement for the assignment of possible actions		

Entry/Exit criteria	*[by tester]* Describes, if it is feasible, how a partner can join/leave an ongoing test – OUT OF SCOPE FOR OPEN PROJECT
Test environment	*[by tester]* Describes the testing environment with a special section dedicated to possible limitations
Item pass/fail criteria	*[by tester]* Identifies how to declare a test case passed/failed; usually where there are multiple steps then the case shall fail if any of the steps fail *[by prototype owner]* Expected output for test cases (e.g. indicators)
Suspension criteria and resumption requirements	*[by tester]* Describes, if it is feasible, why partners can stop and then resume testing, and with what kind of requirements – OUT OF SCOPE FOR OPEN PROJECT
Test deliverables and reporting	*[by tester]* Identifies the deliverable/report files that will be used both during and after testing to collect the results
Remaining test tasks	*[by tester]* If the testing of the same prototype is divided in more phases, this section describes what can be the added features to verify in the following ones *[by prototype owner]* Define if there are possible future developments to be further tested
Staffing and training needs	*[by tester]* Describes the resources and the know how needed to participants *[by prototype owner]* Contribute from direct experience on the prototype
Roles and responsibilities	*[by tester]* Identifies precise actions for people involved in the test experience *[by prototype owner]* Agreement/Validation
Schedule	*[by tester]* Defines the timing of testing, agreed among partners *[by prototype owner]* Agreement/Validation

Post trial analysis	*[by tester]* Describes, if there, some analysis/evaluations, to do after the closure of testing, such as tracking analysis, statistics, and so on

4.2 *Prototypes to be tested during first iteration*

In order to choose which prototypes are going to be submitted to the technological evaluation, the aspects to consider are:

- The real added value that could be given from a technical testing
- The feasibility of a testing based on the approaches previously described
- The relevance of the prototype for the following development within the OPEN project
- The key OPEN functionalities demonstrated in the prototype

Depending on this line, three prototypes will be part of the first stage of the technological evaluation: the web migration application (from D2.1), the mobility support (from D3.2) and the device selection map (again D3.2). So the indicators and the requirements to test will be shaped on these prototypes to realize the test plans in a suitable and feasible way.

Web migration has been chosen since this will be the basis for the final middleware of the OPEN platform: the middleware, as from the definition, "automatically supports the main functionalities, adaptation and state persistence across multiple devices with various interaction resources". So this prototype includes the seamless migration concept for an interactive application (the web shopping list) between different devices, maintaining the current state; this already involves many different modules working together:

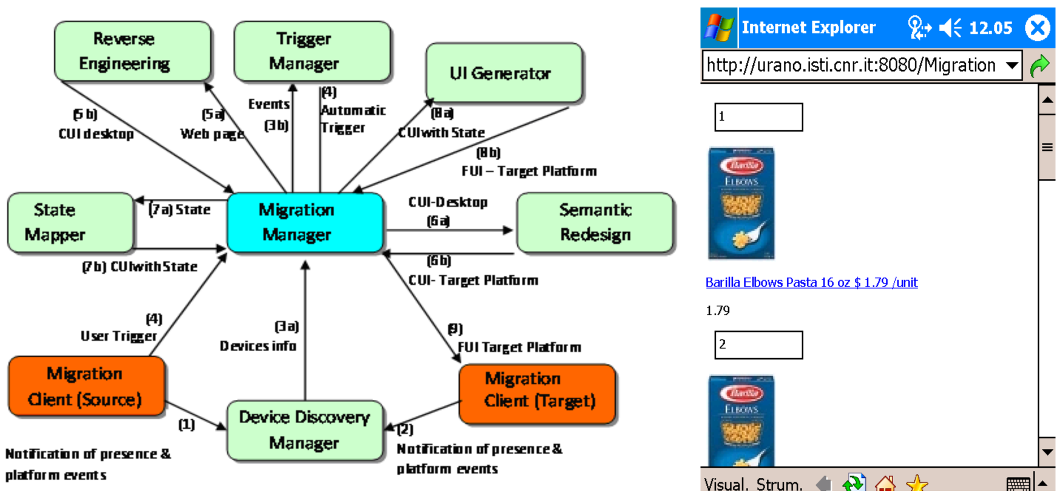


Figure 13: Modules involved in the web migration / An adapted web page

The other two prototypes aim to support the application migration; there is a third prototype within this stream (included in the D3.2), but it mainly concerns the configurability area and especially the application logic reconfiguration. The two solutions to be considered for a technological evaluation are currently working more or less independently, but they represent a strong basis for further work, since in the future they will work together in the integrated OPEN platform.

Mobility support and context information management concerns the migration of a video-streaming application based on context information (e.g. user location), so it is a context aware management of the mobility (for instance the volume can depend on the numbers of users):

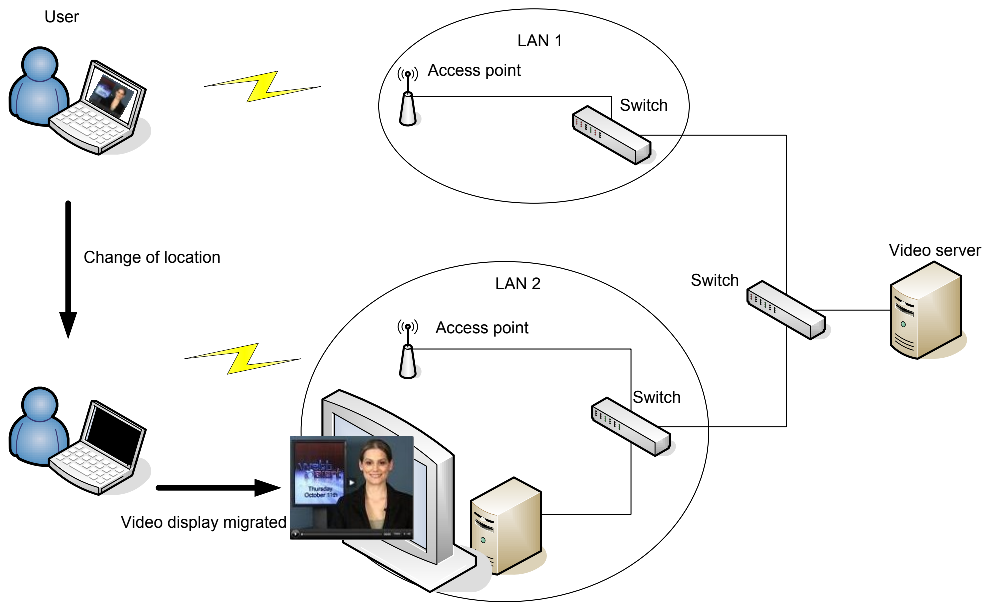


Figure 14: Mobility support for video-streaming migration

Such a prototype, like the web migration, needs the interaction of different functional elements to correctly perform the functionalities which it is aimed at.

Device selection map is, from the description of the prototype itself, “a graphical-interaction component embedded on the OPEN-client that may facilitate it in discovering the available target devices, their capabilities and their state”. Of course to do this, the device selection map has to be context-aware concerning user location, movements and so on, thus being a context provider. Some examples of different graphic follow:

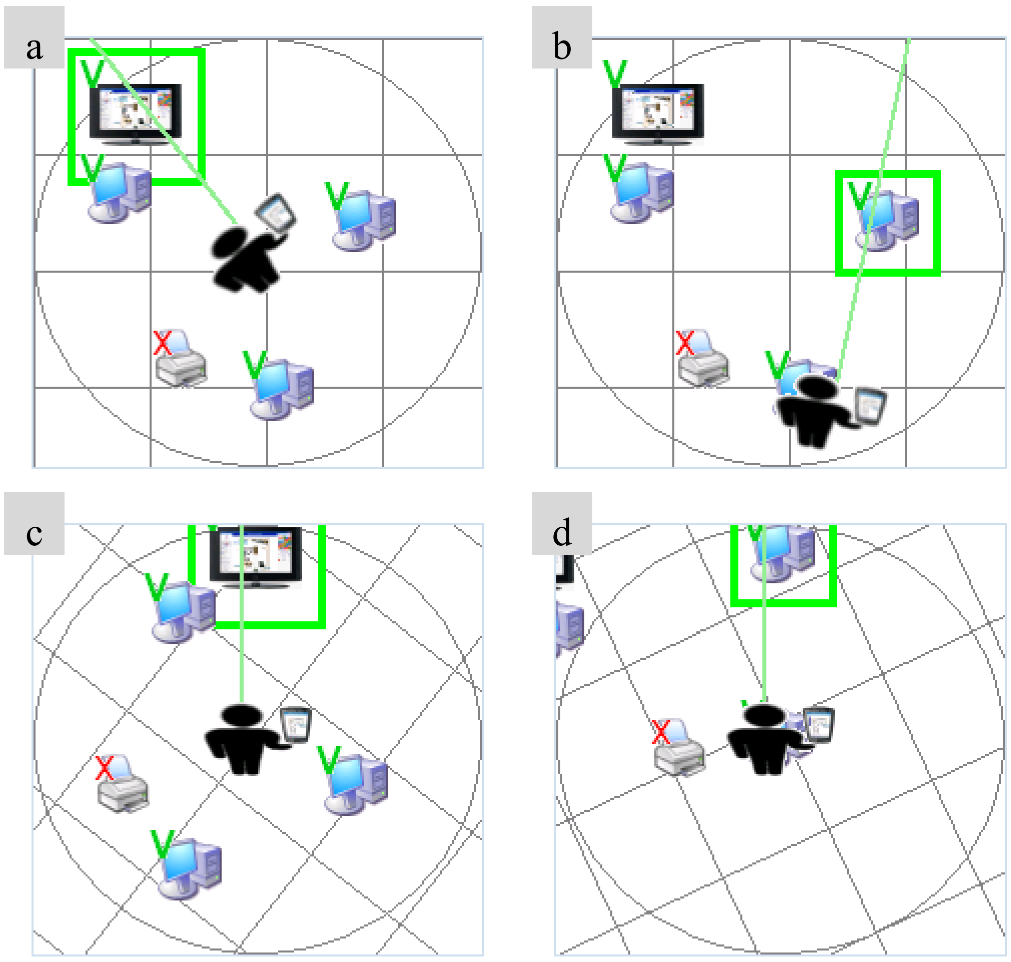


Figure 15: Graphical choices for device selection map

5 Usability test plans

5.1 From D1.1: OPEN Requirements

Note: This is not a standard test plan because it represents an exploratory study about the process of eliciting OPEN requirements. In particular, no product/service intended for the final user is evaluated by a group of users, but only a theoretical analysis is performed.

Test plan:

1. Description: This is an exploratory usability study about the OPEN project deliverable D1.1 (Requirements for OPEN Service Platform).
2. Purpose: This study is an analysis of the deliverable using a user-centred-design optic, in order to verify that during the elicitation of such a requirements document the usability towards final users has been taken into account in the proper way. Through this brief analysis we also want to define some guidelines for the D1.3: this deliverable, in fact, will finalize the requirements to be followed and respected during the implementation of the OPEN platform.
3. Schedule: The analysis has been performed during May 09 (M16), before the finalization of D1.3 in order to provide a useful input for people working on it.
4. Owner: The analysis is performed by VF-IT team.
5. Approach: The following aspects are considered during such analysis
   - Method used for scenarios and requirements proposition
   - Method used to reach an agreement about proposed scenarios and requirements
   - Final requirements and scenarios obtained in the D1.1
6. Outcome: The analysis will be communicated to the owner of D1.3 (SAP) and after that to the OPEN partners with a report file.

5.2 From D5.1: Social Game Application Design

5.2.1 Purpose, Goals and Objective of the Test

D5.1 (Initial application requirements and design) is a key part of exploratory usability study. This test plan aims at evaluating the usability of the Social Game application only through a description and some screenshots of it, to be presented to possible target users of such a solution. The result is intended to be useful as a feedback for the future development of this application.

5.2.2 Participant Characteristic

Participants to the testing are typical user of such an application, so people 18-35 aged, basically keen on gaming, with no particular technical background and know-how. Their number is intended to be from 10 to 20, since they quite represent a homogenous target.

5.2.3 Method

Vodafone team, working together with Arcadia, delivers a questionnaire to be presented to participants. The first point to define is what to present to the responders; in this case some screenshots and a brief description of the application have been used in order to have an immediate rendering of it, so that users can provide and write their opinions. The questionnaire has a dedicated section in this test plan.

5.2.4 Test Environment, Equipment and Logistics

Questionnaire can be provided both directly and from remote to the participants, the same is happening to collect the answers as they finish.

5.2.5 Data to Be Collected and Evaluation Measures

The usability evaluation focuses on user feedback about graphics and format of the social game application, comments on some UI options, and suggestions for the future development, such as other functionalities, similar products and so on.

The answers, classifiable as preference data, are to be collected; they can be analyzed and described with proper diagrams, and provided to developers to help their further work.

In the next paragraph, the template of the usability questionnaire is reported.

5.2.6 Usability Questionnaire

Dear user, you'll find here the "grand vision" of a migratory game concerning Formula 1. Through this application, you'll be able to switch the social race game between different devices both as you wish and automatically (when particular conditions are present) in a seamless way.

The following screenshots will help you to understand the user approach to this kind of application; four situations have been identified, each of them involving different types of migrations:

1. From college library to home
2. From the living room to the kitchen
3. Change of user

4. At the pub

Thomas and Brad, two young gamers, will join the game with you in this storyboard, by showing the levels of the game and the different scenes and goals:

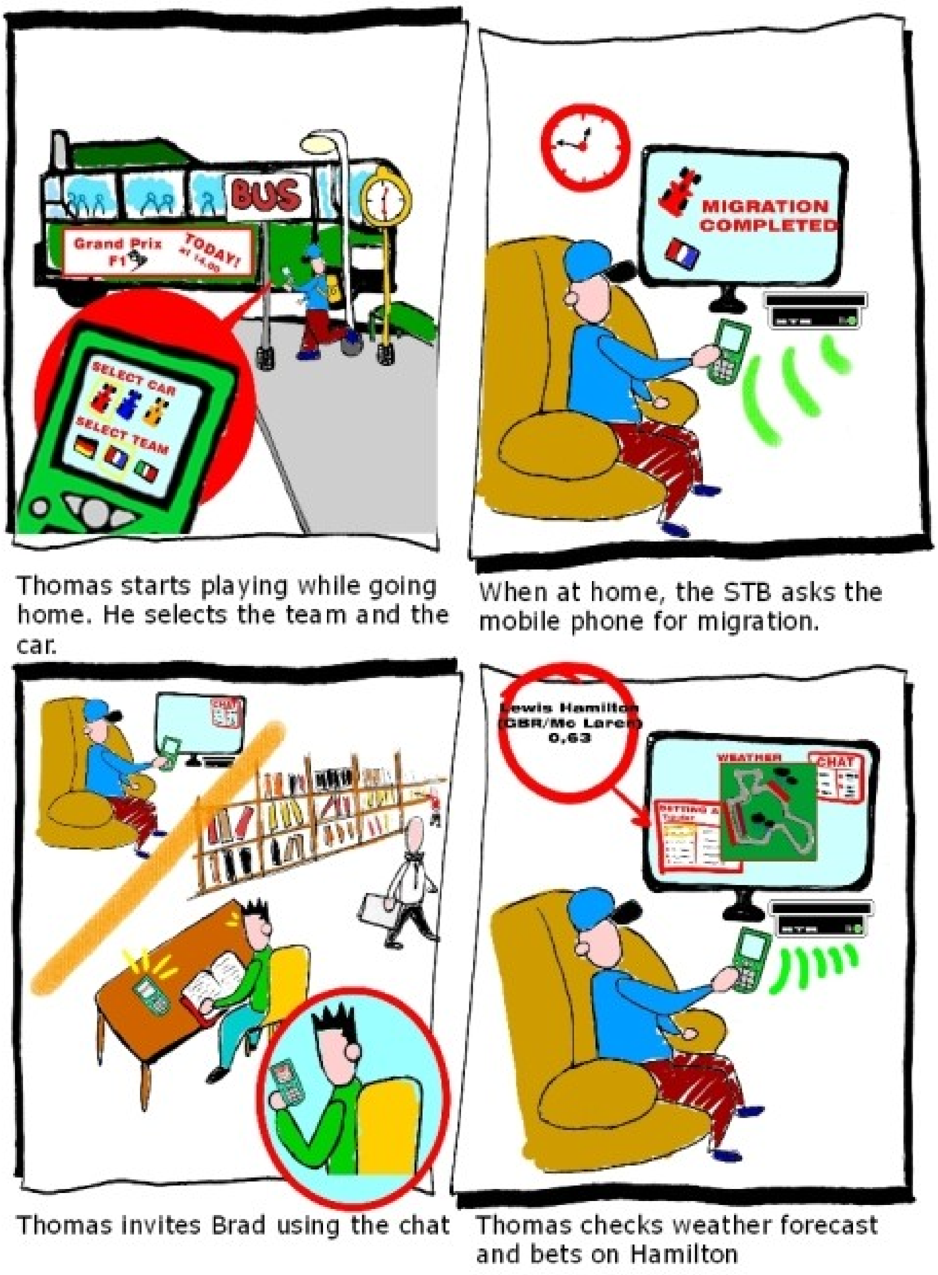

BUS
Grand Prix F1
TODAY! at 14.00
SELECT CAR
SELECT TEAM
Thomas starts playing while going home. He selects the team and the car.
MIGRATION COMPLETED
When at home, the STB asks the mobile phone for migration.
CHAT
Thomas invites Brad using the chat
Lewis Hamilton (GBR/Mc Laren) 0,63
WEATHER
CHAT
Thomas checks weather forecast and bets on Hamilton

Figure 16: Scenario (part 1)

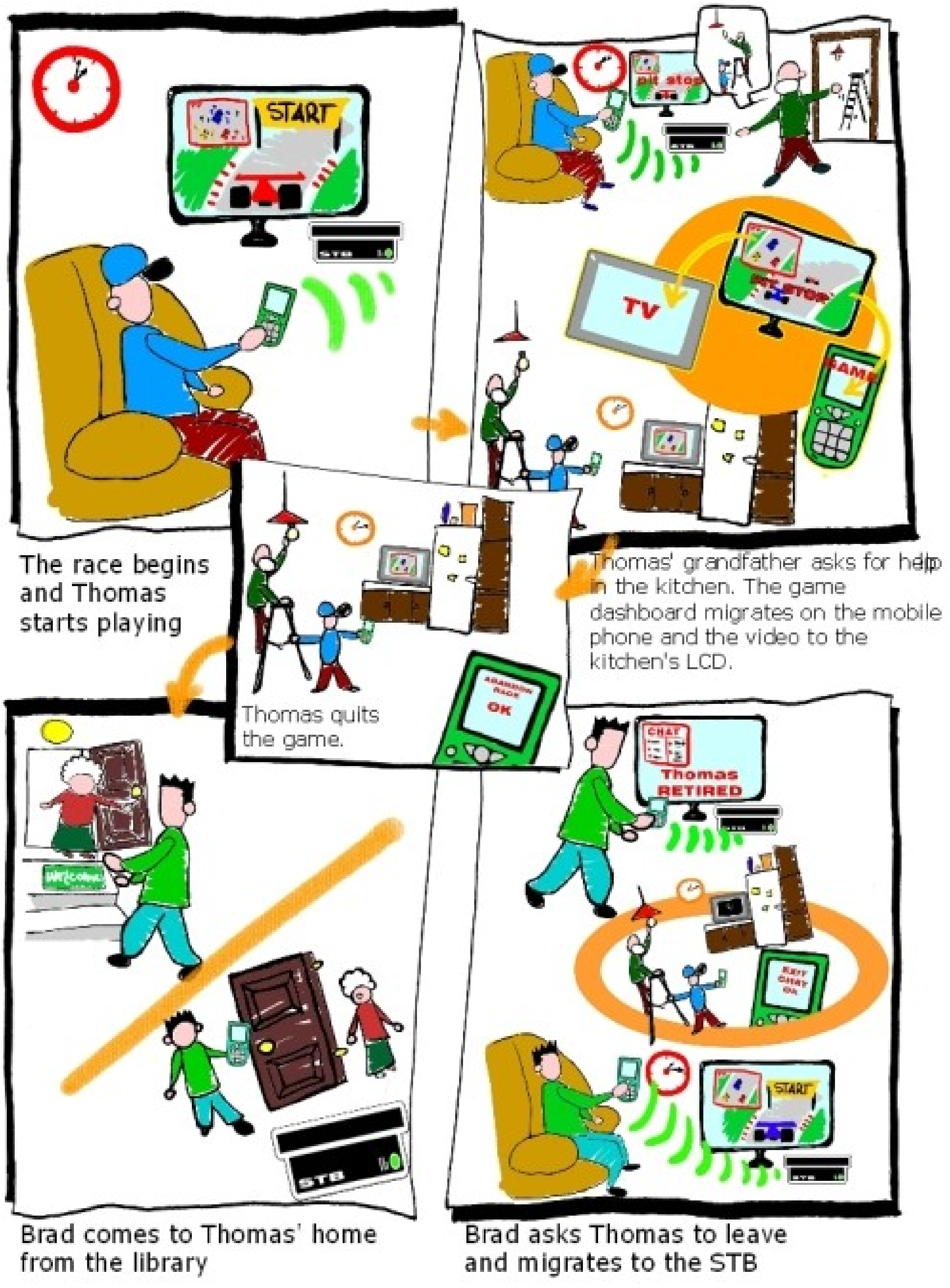

START
STB
pit stop
TV
PIT STOP
GAME
The race begins and Thomas starts playing
Thomas' grandfather asks for help in the kitchen. The game dashboard migrates on the mobile phone and the video to the kitchen's LCD.
ABANDON RACE
OK
Thomas quits the game.
WELCOME
STB
CHAT
Thomas RETIRED
EXIT CHAT OK
START
Brad comes to Thomas' home from the library
Brad asks Thomas to leave and migrates to the STB

Figure 17: Scenario (part 2)

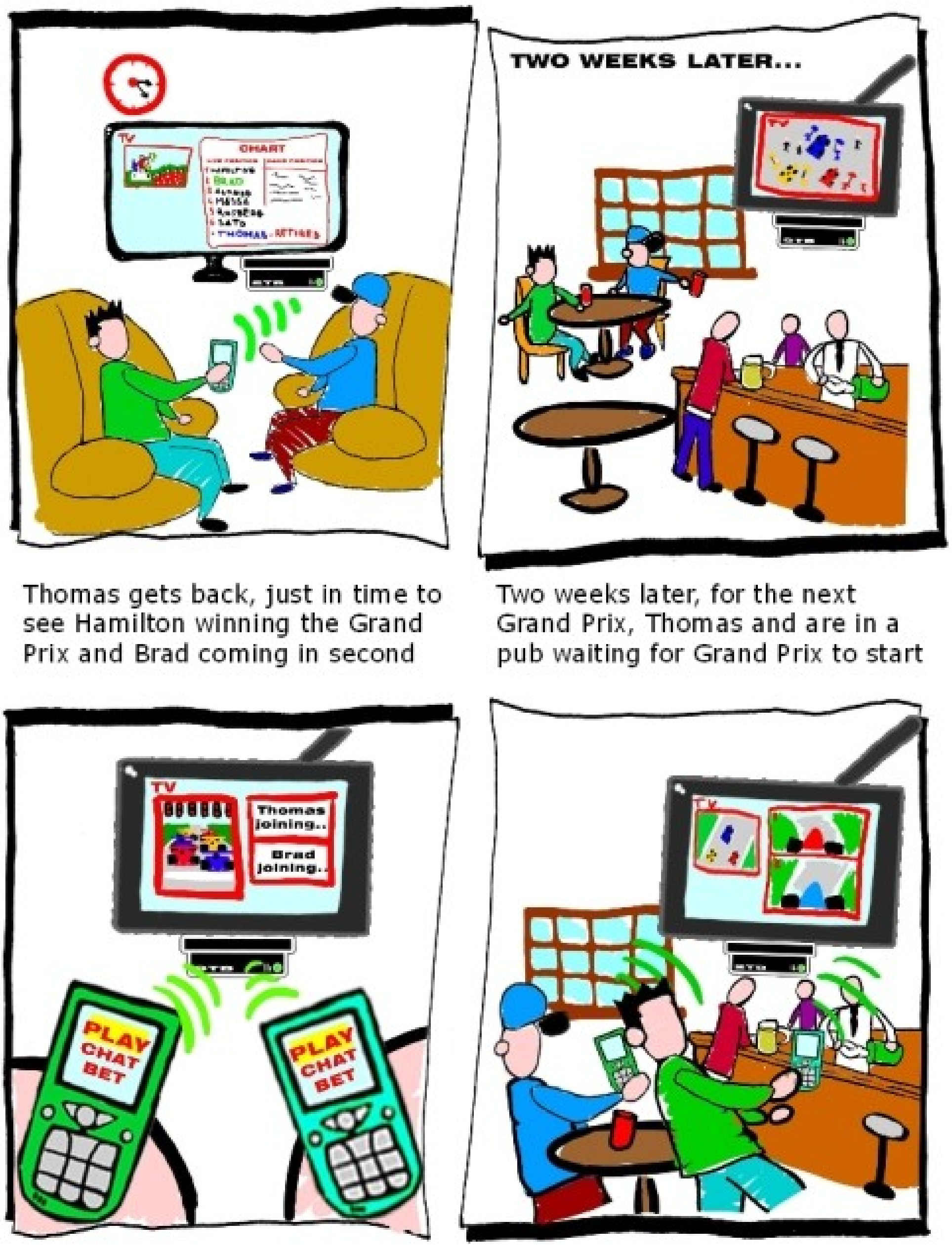
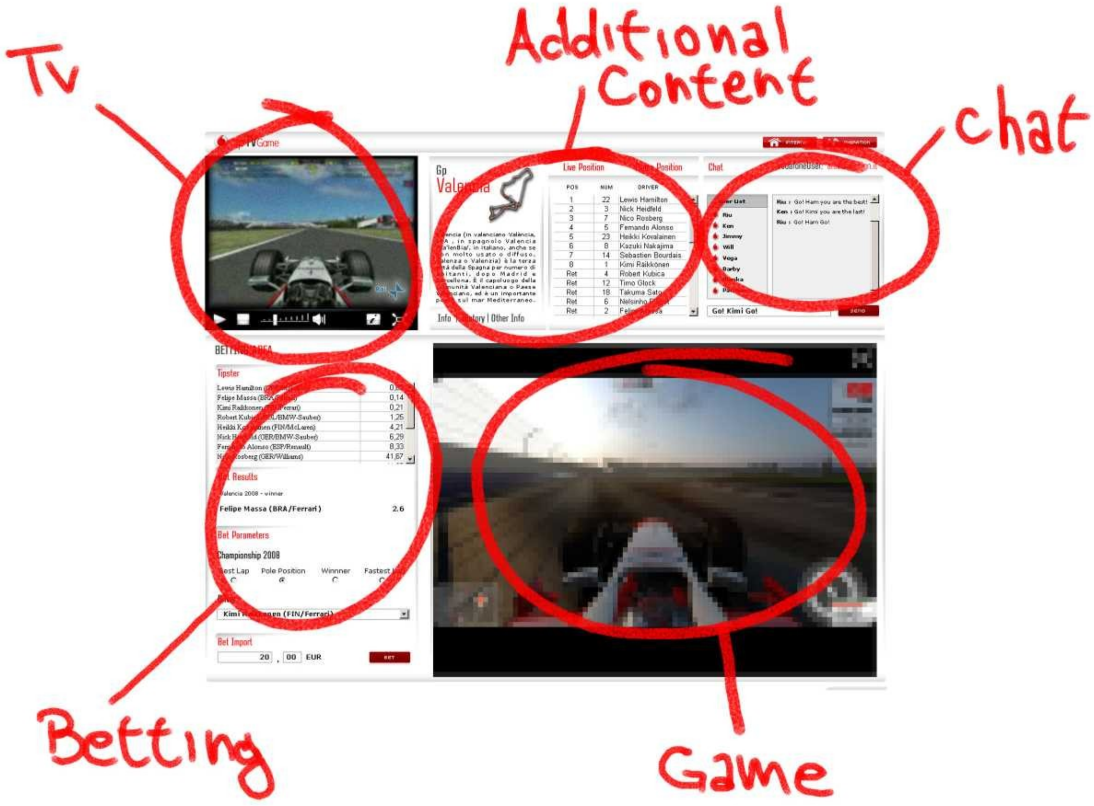

TV
CHART
Thomas gets back, just in time to see Hamilton winning the Grand Prix and Brad coming in second
TWO WEEKS LATER...
TV
Two weeks later, for the next Grand Prix, Thomas and are in a pub waiting for Grand Prix to start
TV
Thomas joining..
Brad joining..
PLAY
CHAT
BET
PLAY
CHAT
BET
Thomas and Brad join the race with their mobile phone
TV
Thomas and Brad play on the big screen and control the game with the mobile phone

Figure 18: Scenario (part 3)

Furthermore, to help you better understanding the game, this is a screenshot of how the application could outcome:



POS	NUM	DRIVER
1	22	Lewis Hamilton
2	3	Nick Heidfeld
3	7	Nico Rosberg
4	5	Fernando Alonso
5	23	Heikki Kovalainen
6	8	Kazuki Nakajima
7	14	Sebastien Bourdais
8	1	Kimi Räikkönen
Ret	4	Robert Kubica
Ret	12	Timo Glock
Ret	18	Takuma Sato
Ret	6	Nelsinho [illegible]
Ret	2	Fel[illegible]

Riu : Go! Ham you are the best!
Ken : Go! Kimi you are the last!
Riu : Go! Ham Go!

Go! Kimi Go!

Lewis Hamilton [illegible]	0,[illegible]
Felipe Massa (BRA/[illegible])	0,14
Kimi Raikkonen [illegible]/Ferrari)	0,21
Robert Kubi[illegible]/BMW-Sauber)	1,25
Heikki Ko[illegible]nen (FIN/McLaren)	4,21
Nick H[illegible]ld (GER/BMW-Sauber)	6,29
Fer[illegible] Alonso (ESP/Renault)	8,33
N[illegible]osberg (GER/Williams)	41,67

Valencia 2008 - winner
Felipe Massa (BRA/Ferrari) 2.6

Championship 2008
Best Lap ( ) Pole Position [✓] Winnner ( ) Fastest [illegible] ( )

Kimi R[illegible]nen (FIN/Ferrari)

20 , 00 EUR

Figure 19: Social Game user interface

Finally here is to you a simple and fast questionnaire to collect your feedback about the application that has been presented to you; the first part will help us to understand your approach to this kind of services, while the second one will be useful to improve it.

The complete version of the questionnaire is available in Appendix A (par. Social Game Questionnaire)

5.3 From D5.1: Emergency Application Design

5.3.1 Purpose, Goals and Objective of the Test

D5.1 (Initial application requirements and design) is a key part of exploratory usability study. This test plan aims at evaluating the usability of Emergency application only through a description and some screenshots of it, to be presented to possible target user of such a solution. The result is intended to be useful as a feedback for the future development of this application.

5.3.2 Participant Characteristic

In this case, participants to the testing present stricter requirements than the previous case, since they have to be possible target user of a business emergency application, i.e. with a particular technical background and know-how. Their number is intended to be about 10 (due to the strict requirements, in case of availability problems this number can be restricted to 8), with a classification based on their skills and background:

- Professionals with strong informatics know how
- People accustomed to manage emergency situations
- People without such skills, but whose company can face such scenarios

5.3.3 Method

Vodafone team, working together with SAP, delivers a questionnaire to be presented to participants. The first point to define is what to present to the responders; in this case some screenshots and a brief description of the application have been used in order to have an immediate rendering of it, so that users can provide and write their opinions. The questionnaire has a dedicated section in this test plan.

5.3.4 Test Environment, Equipment and Logistics

Questionnaire can be provided both directly and from remote to the participants, the same is happening to collect the answers as they finish.

5.3.5 Data to Be Collected and Evaluation Measures

The usability evaluation focuses on user feedback about graphics and format of the emergency application, comments on some UI options, and suggestions for the future development, such as other functionalities, similar products and so on.

The answers, classifiable as preference data, are to be collected; they can be analyzed and described with proper diagrams, and provided to developers to help their further work.

5.3.6 Usability Questionnaire

This section describes the usability questionnaire in order to qualitatively assess the usability of the OPEN migration platform for the Emergency Scenario. The general aim of usability tests is to observe people using the product to discover errors and areas of improvement and to measure how well test subjects respond in four areas: efficiency, accuracy, recall, and emotional response. In this case, since this is still a usability exploratory study, users will observe only a description of the application tasks, enriched by some pictures and screen shots: the results of this first study can be treated as a baseline or control measurement; all subsequent tests can then be compared to the baseline to indicate improvement.

Hypotheses

The test hypotheses describe a reasoned proposal predicting a possible causal correlation among multiple phenomena.

1. The OPEN platform facilitates the smooth operation in an emergency operations centre (EOC).
2. The migration services are easy to use.
3. The migration services are self-explained.
4. The migration services enable new usable application formats.
5. The new application formats boost the usability of emergency applications

Tasks

The main representative tasks for the EOC application are now provided to carry out the usability study. After finishing reading the task descriptions, the user has to fill in the questionnaire in order to state his personal opinion about the product.

1. Register a device (video wall) to the OPEN platform
2. Start the flood simulation application on the PC
3. Choose the video wall as migration target
4. Initiate the migration towards the smart wall

Figure 20 sketches the test scenario. The migration of the graphical user interfaces of both simulations could be triggered by detecting the nearby smart wall via a personal area network like Bluetooth. Over Bluetooth the OPEN system can negotiate the migration policies of a laptop and the smart wall. The smart wall has a high resolution screen that allows the display of different application views at the same time. In order to take full advantage of modern rich Internet applications, the smart wall is also equipped with an OPEN-enhanced Mash-Up generator that combines the outputs of different applications into a coherent view (see Figure 22-Figure 23). Also the controls of the different applications have to be merged in a usable manner. The simulation applications can still simultaneously be controlled via the PDA or laptop as well as via the smart wall.

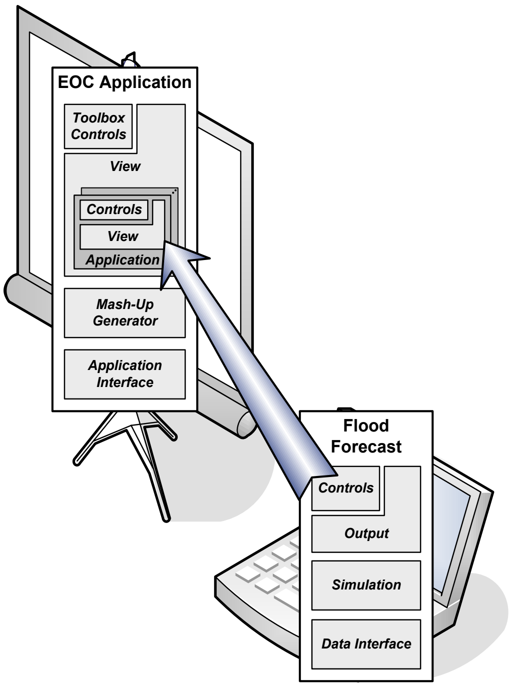


Figure 20: Migration test scenario

Sample Application

The following pictures show screenshots of the OPEN migration platform as well as of the emergency scenario applications to the according tasks:

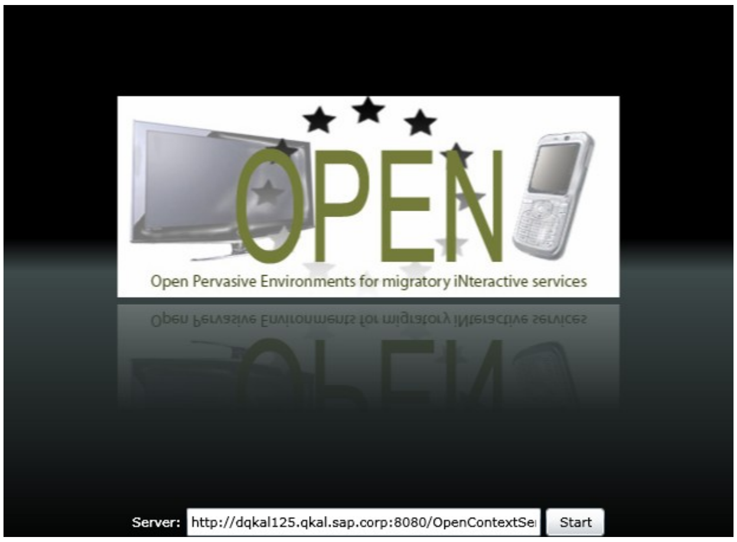

OPEN
Open Pervasive Environments for migratory iNteractive services
Server: http://dqkal125.qkal.sap.corp:8080/OpenContextSe
Start

Figure 21: Register a device (video wall) to the OPEN platform through the address of the OPEN server

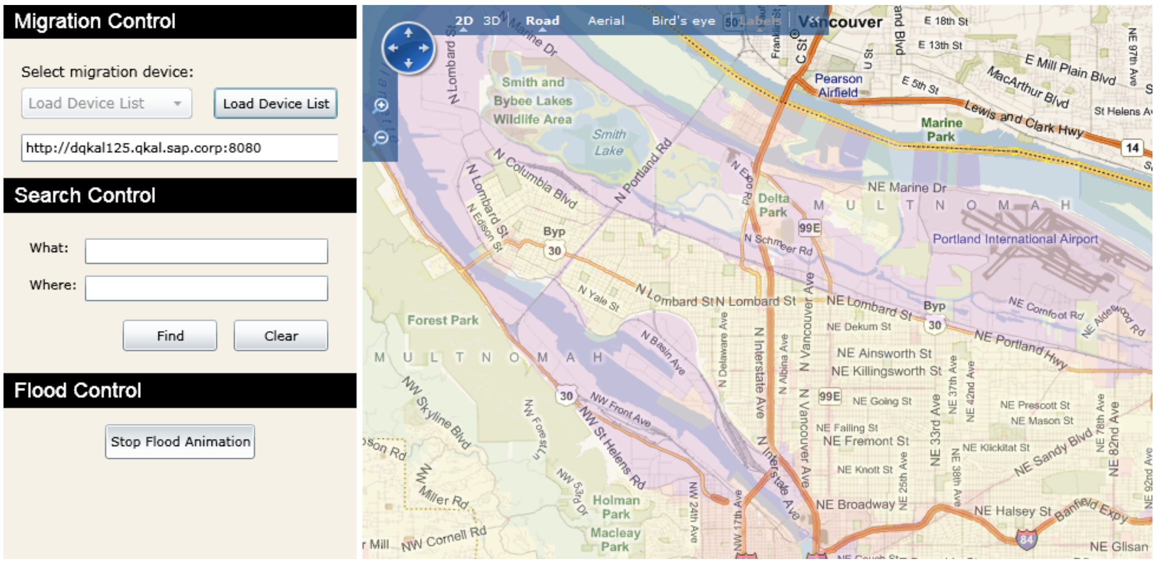
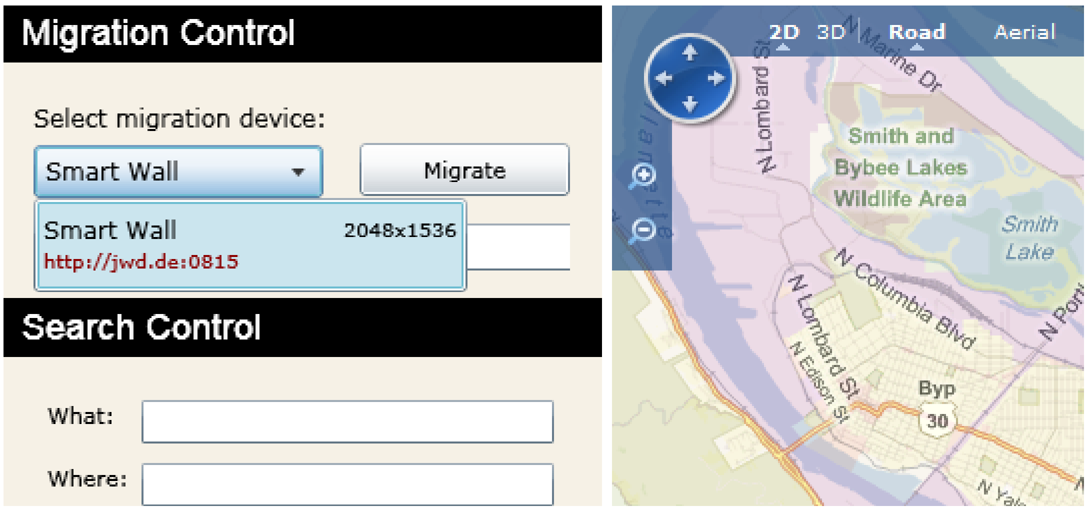


Figure 22: Start the flood simulation application


Migration Control

Select migration device: Smart Wall

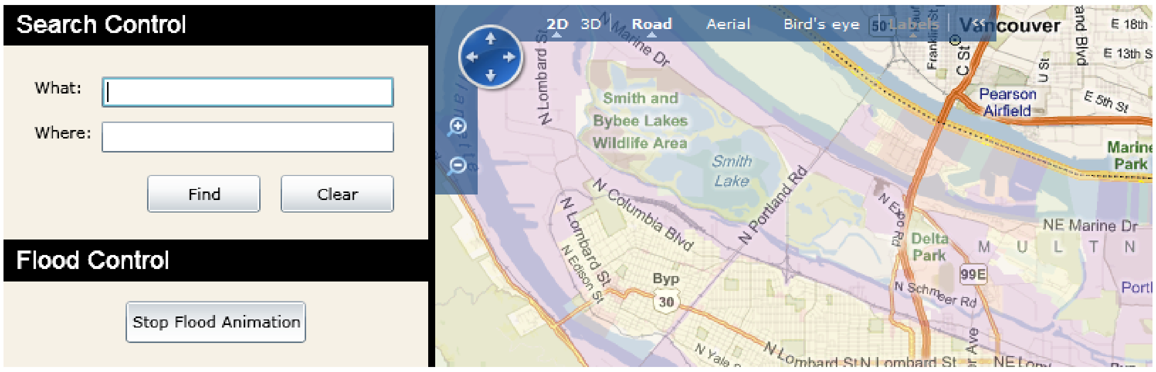
Smart Wall 2048x1536
http://jwd.de:0815

Search Control

What:

Where:

Figure 23: Choose the video wall as migration target


Search Control

What:

Where:

Flood Control

Figure 24: Video Wall after migration

The questionnaire proposed to the user is available in Appendix A (par.)

5.4 From D2.1 and D3.2: Web Migration with Device Selection Map

5.4.1 Purpose, Goals and Objective of the Test

The web applications migration prototype [D2.1] offers the option to use the same web application from different devices, with a user interface adaptation. Moreover, the migration functionality is offered, and thus it is possible to start using the application on a device and to continue using it in another one, while maintaining the current state.

The first purpose of this usability test is to evaluate the user interfaces that the OPEN platform generates from original web applications for PDA devices.

This is not a standard usability test because there is not a single user interface to evaluate, but a mechanism for user interfaces generations (par. 2.3.2).

A comparative test will be performed between the original web applications accessed from PC and the corresponding adapted versions displayed on the mobile device. For the latter, a little usability decrease is accepted only if it is related to the devices capabilities (for example if it is related to the device screen size or to the available bandwidth). A usability evaluation will also be performed for the original web application accessed from a PDA device (i.e. without using any component of the OPEN platform), in order to determine the OPEN web interface adaptation perceived usefulness.

The second purpose of this usability test is an evaluation of the migration process offered by the OPEN platform. In particular, a user interface is available, that allows to perform a migration of the current web application from the current device to another one. The selection of the target device is performed using a "**Device Selection Map**" [D3.2].

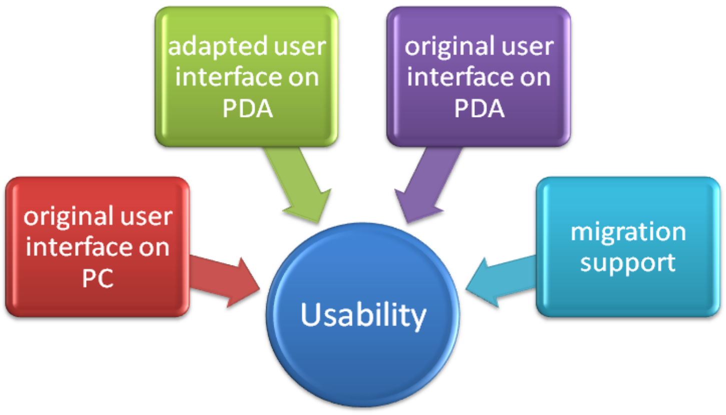

adapted user interface on PDA
original user interface on PDA
original user interface on PC
migration support
Usability

Figure 25: usability evaluations that will be performed on the web migration prototype

In the next paragraphs a complete description about the proposed testing activity will be provided.

5.4.2 Research Questions

This paragraph contains a list of aspects that will be considered during the usability evaluation of the tested prototype.

The following research questions will be addressed to web applications adaptation. These questions are referring to the usage of the adaptation module on a PDA device. At the end of the usability test it will be possible to answer all of these questions.

- Do adapted web pages generated by the OPEN platform offer an acceptable level of readability? In order to answer this question, it is necessary to compare adapted web pages with their original version.
- Is it possible to recognise every component of a web page (for example titles, forms, etc.) in the adapted version generated by the OPEN platform?
- Are adapted web pages generated by the OPEN platform more usable than the original ones (displayed on the same device)?

The following research questions will be addressed about the migration process:

- Is the migration process intuitive? (i.e. is it possible to know in an intuitive way what will be migrated and what is the destination device of the migration?)
- Is the migration process reversible? This question is needed because when the user commits an error during a migration (for example when he selects a wrong target device) he must be able to roll-back to the previous situation.
- Is the perceived continuity acceptable for the end user?

5.4.3 Participant Characteristic

This paragraph contains the OPEN web migration user profile. The user selection will be performed using this profile and the generic OPEN user profile.

- Age. This application's users can have an age between 18 and 60 years. There are not particular constraints on the user's age.
- Sex. There are no constraints about the user sex. It could be preferable to have almost the same number of male and female users.
- Internet usage. This application's users are very familiar with web sites and web applications. It could be preferable that they access the web on a daily basis.

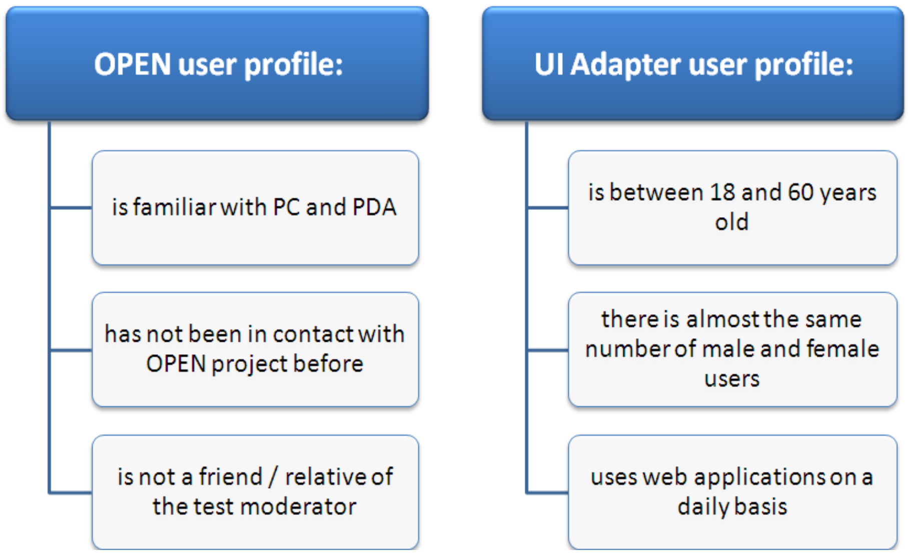

OPEN user profile:
is familiar with PC and PDA
has not been in contact with OPEN project before
is not a friend / relative of the test moderator
UI Adapter user profile:
is between 18 and 60 years old
there is almost the same number of male and female users
uses web applications on a daily basis

Figure 26: OPEN and web migration user profiles

5.4.4 Method

The application will be tested with three web applications. A complete test list will be executed for the first two web sites, and an informal evaluation will be provided for the last web site.

Web applications containing only html and JavaScript code will be taken into account. Moreover, in order to make sure that the UI adapter does not encounter any problem during the original web pages elaboration, only web pages containing well formed html code will be taken into account. Such condition will be verified using the W3C Validator (http://validator.w3.org). This tool allows to check if an html page, even if it is correctly displayed by web browsers, contains some errors (for example a tag that is never closed). All of these verifications have been performed before the task list creation, in order to avoid any adaptation problem during the usability test.

The web applications will be tested on a PC, on a PDA using the OPEN platform, and on a PDA without using the migration platform (in the last case no evaluation will be performed on the migration process).

As explained in the par. 2.3.2, two groups of users will be employed. The first group will execute a task list on the PDA without using the OPEN platform, then will execute it on a PC, and finally will migrate the application, in order to execute the task list on the PDA (using, this time, the OPEN platform).

The second group of users will perform the same operations, but starting from the PDA with the OPEN platform, then migrating on a PC and in the end using a PDA without the OPEN platform.

The task list execution on the PDA device without using the OPEN platform is useful only for the Web UI Adaptation module evaluation. In this case, indeed, it is it isnot possible to perform a migration and the state maintenance is not offered.

It is worth noting that this method does not completely solve the bias issue since, for both of the user groups, the PC is used after another device. However, considering that only simple and intuitive web applications will be used, and that an informal study will be performed, this effect could be considered acceptable. An alternative solution could be to employ a third group of users, but the advantages that could be obtained from such an approach would not justify the required extra-effort.

At the end of the testing activity, a debriefing session will be performed with the user, the moderator and test observers. In order to identify at least the 80% of usability problems, each testing group must contain at least 4 users.

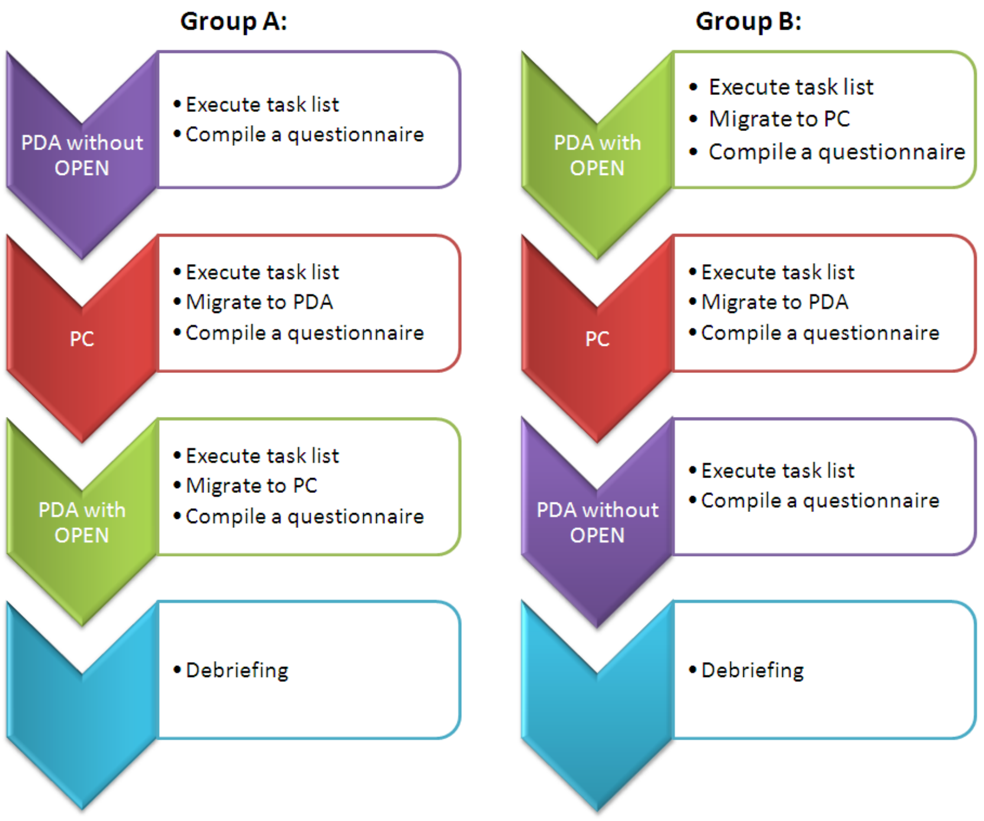


Figure 27: usability testing activity for web migration prototype

A few days before the usability test, users will compile a questionnaire about their experience on web applications (for example, what kind of web sites they prefer, how much time they spend surfing on the internet, etc.). Moreover, each user will provide a list of her/his most used web sites. These answers will be then used to individuate for every user a web site considered very significant for her/him and that is supported by the OPEN web migration prototype. During the test, the user will perform some tasks on this web site and s/he will provide a qualitative feedback. However, this phase can be skipped if it is not possible to identify a web site that can be correctly adapted and migrated.

The first application taken into account is a web site proposed by the application developers and it is called: "**Shopping Assistant**". It is an e-commerce web site that offers the option to focus on the migration client capability to maintain the application status during migrations (in particular when they are performed during not completed transactions) and on the web pages navigability.

The second web site tested is one of the most famous applications available on the internet, i.e. **wikipedia.org**. The Italian version will be taken into account (at least for Italian users). This web site, with its several articles, offers the option to analyze, in particular, the web user interface readability.

The last web site to be tested will be individuated using the user's initial questionnaire. This site represents a web application that the user considers very useful and it is, obviously, different for each user. For this reason, during this test phase, a less formal approach will be used. Then, user comments or suggestions will be taken into account more carefully than numerical parameters (it will not be possible to calculate mean usability parameters for user groups, because every user will use, in this phase, a different web site).

During all the testing activity, the "**thinking aloud**" method will be employed: the user will be encouraged to explain her/his thoughts during all the testing activity.

At the end of a user testing activity, a mean level of usability will be calculated for each web application and for each device (using the parameters defined in par. 5.4.8). This value is obtained calculating the arithmetic mean of the usability parameters expressed by the user during the usability questionnaires compilation.

The expected result is that the PDA-adapted UI usability is a little less than the UI displayed on the PC usability (because the PC has a larger screen, a higher computational power and usually a greater bandwidth than the PDA) and greater than the PDA not adapted UI usability.

This calculation will be performed before the debriefing session, so every unexpected result will be discussed with the user, in order to get some additional information.

At the end of all the testing sections, a mean usability level will be calculated by taking into account Shopping Assistant and Wikipedia tests. For the third web site it is not possible to calculate a mean usability value for all of the users, because it is different for each user.

If we indicate with U_{PDA-A} the usability on the PDA with OPEN UI adaptation, with U_{PDA-N} the usability on the PDA without the OPEN UI adaptation, and with U_{PC}, the usability on the PC, the following result is expected:

$$U_{PDA-A} > U_{PDA-N}$$

$$U_{PDA-A} < U_{PC}$$

The usability on the PDA with the OPEN web user interface adapter must be greater than the one obtained on the same device without user interface adaptation. Moreover, the usability of the user interface displayed on the PC can be slightly greater of the PDA usability with adaptation, but their usability difference should be minimal (a not formal analysis will be performed, then a threshold difference hasn't been defined).

About the usability evaluation of the migration, an informal evaluation will be performed and the application usability will be considered good if the most of the user will consider it good. A mean usability parameter will be calculated, but there isn't any condition about its value.

The test plan proposed in this document will be validated during the execution of a pilot test. During this phase, either the test list or the expected usability results could be modified (for example if a test page is no longer available, or if the application is not able to adapt or to manage its migration in the expected way).

5.4.5 Task List

In Appendix A (par. Web Migration Task Lists) the proposed task lists for the usability tests performed by the user group A and by the user group B are available. A few tasks will be added when the preliminary questionnaire of every user will be analyzed. This task list could be subjected to some modifications during the pilot test execution.

5.4.6 Test Environment, Equipment and Logistics

The following devices will be used for this usability evaluation:

- PC. A PC connected to a LAN through a WiFi access point is required. It must be able to reach the migration server (that has a public IP address) using the TCP protocol and a specific port. Moreover, it must be able to communicate with other devices on the same LAN using TCP and UDP protocols.
- PDA. A PDA using Windows Mobile 6.0 will be employed. Dot Net Compact Framework 3.5 will be installed on the device in order to execute the application prototype. The device shall have a WiFi connection that will be used in order to access the same LAN where the PC is located. TCP and UDP protocols will be used to communicate with the PC and TCP protocol will be used for the communication with the migration server.

Preliminary questionnaires (to be compiled some days before the task list execution) will be compiled in an electronic format.

All of the questionnaires to be used during the task list execution will be printed and then compiled by the user using a pen.

The moderator will take notes during the task list execution using a proper module.

No audio/video registration will be performed during this test and no specific logging tool will be used.

5.4.7 Test Moderator Role

Before the testing activity, the moderator should make sure that the required test environment is correctly configured and that there are no network problems. Moreover, he should verify that required test pages are reachable.

During the task list execution, the moderator should talk with the user in order to know his thoughts. He should have a copy of the test list in order to take notes for every executed task.

At the end of the task list execution, the moderator must analyze the user questionnaires and direct the debriefing session. Also in this case, he should take notes of the discussion.

Moreover, the test moderator could modify the test list in the case of an unexpected event during the task list execution (for example if a tested web application is no longer reachable, or if a web page has been modified and its new version contains an error that makes impossible to use it in the OPEN platform).

5.4.8 Data To Be Collected and Evaluation Measures

This paragraph contains a list of aspects to be evaluated during the task list execution. Parameters related to the web user interface will be evaluated for each tested version (using a PC, a PDA with OPEN, or a PDA without OPEN).

Parameters related to the web user interface:

- Text readability. The text content displayed on the user interface must offer an acceptable readability. This is especially important for medium length text content (for examples news, scientific articles, and other information available on the internet), that often have a poor readability on PDA devices. Every user employed in the testing activity will evaluate this parameter using a numeric value from 1 (very poor) to 5 (very good).
- Visibility of titles. The user must be able to identify, in thedisplayed web pages, each title. They could be displayed in a greater size than normal text, in a different color, or in bold. The important thing is that they are recognizable, even for users that have never used the tested web application. Also for this parameter a numerical evaluation between 1 (very poor) and 5 (very good) will be provided.
- Visibility of links. The user must be able to identify each link contained in the visited web pages. Two evaluations will be provided for this parameter. A value between 1 (very poor) and 5 (very good) decided by the user, and the number of errors (i.e. clicks on texts that are not links) measured by the test moderator.

- Images rendering. Images shall be correctly displayed in the device screen. Images shall be neither too large, nor too small in comparison with the screen size. A numerical evaluation between 1 (very poor) and 5 (very good) will be provided.
- Web forms usability. Web form elements (i.e. text inputs, password inputs, selection lists, buttons, etc.) must be correctly displayed. In particular, it must be possible not only to read the content of web form elements, but also to edit it (when it is possible). Moreover, disabled or not editable elements must be recognizable. A numerical evaluation between 1 and 5 will be provided, with some (optional) comment about elements whose rendering is not optimal.
- Navigability. The user must be able to navigate with ease in the web application and to go back to the previous page without using the web browser back option. A numerical evaluation between 1 and 5 will be provided.

Parameters related to the migration process usability:

- OPEN UI user friendly. The OPEN client (the application used to start a migration) shall be very easy to use even if the user has never used it before. A numerical evaluation between 1 and 5 will be provided.
- Device selection map usability. The map displayed to the user in order to select the migration target device must be clear even for users that never used it before. A numerical evaluation between 1 and 5 will be provided.
- Web browser interaction. The OPEN client shall not create any problem during the normal web browser usage (i.e. it shall not cover any part of the current web page). A numerical evaluation between 1 and 5 will be provided (1 for the worst case, 5 for the best case).
- Continuity. After the migration from a device to another one, the state must be kept and the user shall be able to continue using the web application in another device. This is not a technical evaluation, but this parameter represents the application continuity perceived by the user. For example, if the migration is correctly performed, but it takes a long amount of time, even if the state is correctly saved, the end user will perceive a poor continuity. A numerical evaluation between 1 and 5 will be provided.

About the reversibility of the migration process, it is evaluated during the task list execution, and if there will be a problem about this point it will be noticed by the test moderator and discussed during the debriefing session.

5.5 From D5.2: Social Game Prototype

5.5.1 Purpose, Goals and Objective of the Test

The Social Game prototype is a complex web application, described in detail in the document D5.2 (Initial prototype applications).

The following functionalities are offered:

- Chat. It is possible to chat with other users of the Social Game.
- IPTV. In the current implementation, a preconfigured clip is displayed instead of a real IPTV. In the complete implementation of this scenario, it should be possible to view a real Formula one race while playing the Racing Game.
- Racing Game. A formula one racing game is available. Every lap completed by the player is timed and her/his best times are compared with those of the other players. In the complete implementation of this scenario players' times would be compared with real drivers' times.
- Betting. It is possible to bet an amount of money on the real race results.

Since in the current implementation only a simulation of the migration of the controls of the Racing Game is provided, the main purpose of this testing activity is to evaluate the feeling of the migration process, as well as the application adaptation for the supported component. Nevertheless, in order to provide some suggestions regarding possible evolutions of the Social Game development, a usability evaluation will be also performed for the components that in the current implementation do not support migration yet.

5.5.2 Research Questions

This paragraph contains a list of aspects to be considered during the usability evaluation that will be performed on the tested prototype.

The following research questions will be addressed about the prototype migration features:

- Has the Racing Game the same usability level when it is controlled by a PC and when it is controlled by a mobile phone?
- Are racing indicators (for example speed, acceleration, etc.) clear enough when the racing game is controlled by a PC and when it is controlled by a mobile phone?
- Is the migration process intuitive and simple to use?

The following research questions will be addressed about the social game features that at the moment cannot be migrated. This could be useful for the following of the application development:

- Is the chat component readable and simple to use?
- Is the betting component readable and simple to use?
- Is the IPTV simulator correctly rendered and simple to use?

5.5.3 Participant Characteristic

This paragraph contains the Social Game user profile. The user selection will be performed using this profile and the generic OPEN user profile.

- Age. This application's users can have an age between 18 and 35 years. The social game users should be quite young, because this application is addressing people keen on games and social web.
- Racing games. This application's users have some experience on racing games.
- Chat applications. This application's users are very familiar with chat and instant messaging software.

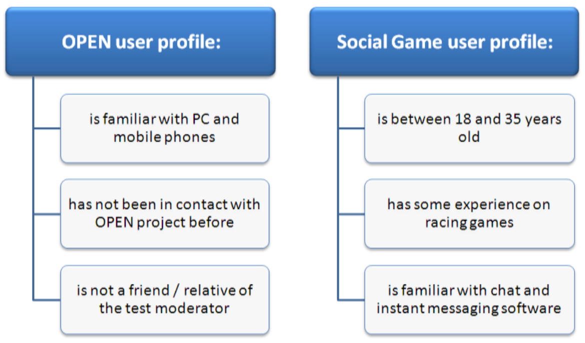

OPEN user profile:
is familiar with PC and mobile phones
has not been in contact with OPEN project before
is not a friend / relative of the test moderator
Social Game user profile:
is between 18 and 35 years old
has some experience on racing games
is familiar with chat and instant messaging software

Figure 28: Social Game user profile

5.5.4 Method

As stated in the paragraph about the method proposed for usability tests [par. 2.3], a comparative evaluation will be performed for the features that can be migrated. In the current prototype version, only Racing Game controls and some indicators are migrated from a PC to a mobile phone.

In order to take into account the learning process during the Racing Game usage and then to avoid any bias, two groups of users will be employed, who will complete the task list with a different execution order. For this application prototype the task list is quite short and then, it is useful for users to compile only one questionnaire at the end of the application usage.

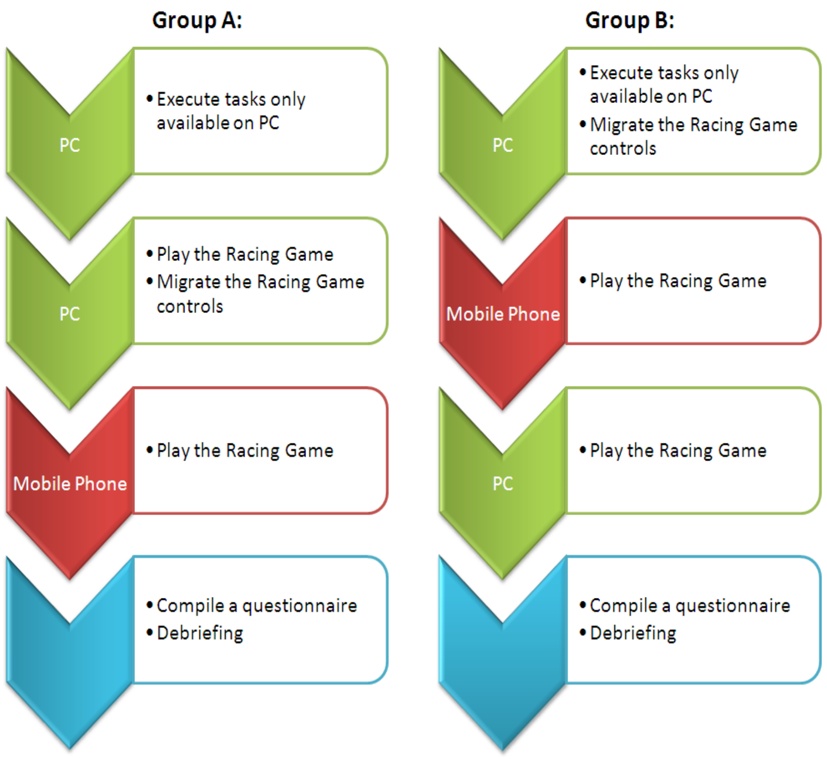


Figure 29: Social Game usability testing procedure

Before the task list execution, an extract of the D5.2 will be provided to the users in order to give them some basic information about the Social Game characteristics and, in particular, about the Racing Game usage (for example, the fact that in order to restart the current lap it is necessary to press the "T" key).

After the task list execution and the questionnaire compilation, a debriefing session will be held, in order to discuss with the user about her/his impressions on the prototype usability level.

As stated in the usability methodology paragraph [par. 2.3], there are no strict usability requirements in this phase because application prototypes are still not completely developed and the usability testing team will be more focused on finding requirements/suggestions for the following of the development cycle. However, users employed in the usability evaluation will fill a questionnaire by answering some questions with a numerical value (between 1 and 5). At the end, a mean usability value will be calculated for several aspects.

The only usability requirement for this testing phase is that, for the functionalities that can be used via PC and via mobile phone, there is a similar usability level.

For all of the other aspects of the social game, a careful informal analysis will be performed.

5.5.5 Task List

In Appendix A (par. Social Game Task Lists) the proposed task lists for the usability test performed by the user groups A and B are available.

The task lists are very similar (only the execution order is slightly different).

5.5.6 Test Environment, Equipment and Logistics

The following test environment [D5.2] will be used during this usability evaluation:

- PC. The same device will be used as Game Client, Game Logic Server, and Physics Server.

  The following characteristics are required: Intel or AMD CPU running at 2Ghz with 1Gb RAM, ATI or NVIDIA GPU supporting Shader Model 2.0, and 100Mb of free disk space, Display with 1280x1024 resolution, WiFi network connection; Mozilla Firefox 3.x with JavaScript enabled running on Windows XP SP2 or Vista operating system.
- Mobile Phone. A Nokia N95 mobile phone will be employed. The tested application could be compatible also with other devices, but the development team already used the application on this mobile phone, so, in order to avoid instability problems, usability tests will be performed on this device.
- Network configuration. Each application needs a network connection with at least 2 Mbit bandwidth and a set of open TCP and UDP ports. Moreover, an internet connection is needed. A WiFi access point will be employed in order to connect the PC and the mobile phone on the same LAN.

The questionnaire will be printed and then compiled by every user using a pen.

The moderator will take notes during the task list execution using a proper form.

No audio/video registration will be performed during this test and no specific logging tool will be used.

5.5.7 Test Moderator Role

Before the testing activity, the moderator should make sure that:

- The game Physics and Logic Server are correctly configured and running on the test PC. The PC, moreover, should be correctly configured in order to be used as the social game client.
- The Racing Game application is correctly installed and configured on the mobile phone.
- The document with the application description and the required questionnaire are available.

During the task list execution, the moderator should talk with the user in order to know his thoughts. He should have a copy of the test list in order to take notes for every executed task.

At the end of the task list execution, the moderator will analyze the user questionnaire and direct the debriefing session. Also in this case, he should take notes on the discussion.

Due to the small number of tasks contained in the task list, it is suggested to avoid any modification of the task list. If it is impossible (for example for an unexpected network problem) to execute a task, this fact must be taken into account during the questionnaire compilation (for example, if it is impossible to execute the task on the chat, the user will not answer any question about this feature).

5.5.8 Data To Be Collected and Evaluation Measures

This paragraph contains a list of aspects to be evaluated during the task list execution.

Parameters that can be evaluated on both the PC and the mobile phone:

- Racing Game controls. The usability level of the racing game controls will be evaluated by the user with a numerical value between 1 and 5.
- Racing Game Indicators. The clearness of the indicators displayed when the user is playing the racing game will be evaluated by the user with a numerical value between 1 and 5.

Parameters that will be evaluated only on PC:

- Chat Evaluation. An overall usability evaluation about the chat tool will be provided by the user. A numerical value between 1 and 5 will be used.
- Betting Evaluation. An overall usability evaluation about the betting tool will be provided by the user. A numerical value between 1 and 5 will be used.
- IPTV Evaluation. An overall usability evaluation about the IPTV simulator will be provided by the user. A numerical value between 1 and 5 will be used.
- Migration Evaluation. A usability evaluation about the commands used in order to simulate the migration of the game commands from the PC to the mobile phone will be provided. Also in this case, a numeric value between 1 and 5 will be used.

5.6 From D5.2: Emergency Prototype

5.6.1 Purpose, Goals and Objective of the Test

The Emergency Scenario application is intended to be used when two or more emergency management experts are working on different aspects of the same problem and they need to merge their results.

This prototype is a web application that offers the option the play a simulation (i.e. a geo-referenced time-sequenced dataset) on a map and to migrate it to a smart wall. When two simulations are migrated to the smart wall, they are shown on the same map in a merged view [D5.2].

The objective of this test is to evaluate the usability level offered by the migration and when merging of two simulations. In the current implementation the prototype does not offer the option to split two simulations from the smart wall to two PCs.

5.6.2 Research Questions

This paragraph contains a list of aspects to be considered during the usability evaluation that will be performed on the tested prototype.

The following research questions will be addressed:

- Is the migration process intuitive and simple to use?
- Is a simulation with one dataset displayed in a comprehensible way?
- Is a simulation with two datasets displayed in a comprehensible way?

5.6.3 Participant Characteristic

This paragraph contains the Emergency prototype user profile. The user selection will be performed using this profile and the generic OPEN user profile [par. 2.1.1].

- Age. This application's users are older than 25 years, because this scenario belongs to the business domain.
- Work Experience. This application's users have a good experience either in the IT field or in some other field related to emergency management.

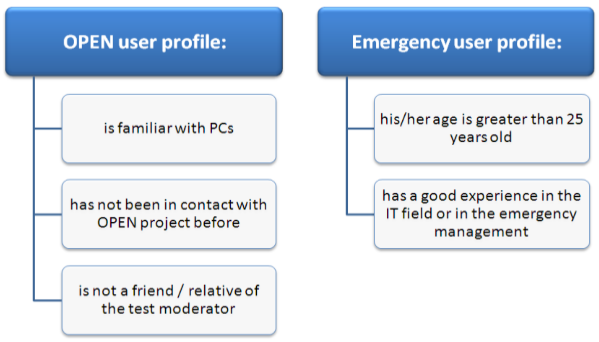

OPEN user profile:
is familiar with PCs
has not been in contact with OPEN project before
is not a friend / relative of the test moderator
Emergency user profile:
his/her age is greater than 25 years old
has a good experience in the IT field or in the emergency management

Figure 30: OPEN user profile and Emergency prototype user profile

5.6.4 Method

During this usability test, as stated in the paragraph 2.3, two aspects of the prototype will be addressed. A comparative evaluation will be performed between the simulation of a single dataset and the simulation of two datasets (this is the feature that is modified during the migration) and a usability evaluation will be performed on the migration process.

For this application it is not possible to employ two groups of users (as stated in par. 2.3.2) for the task list execution in a different order. In fact, in order to apply this methodology, users of the second group should start using the application with the simulation of two datasets and then migrate one of them to another device. This, with the current implementation, is not possible. So a single group of 4-5 users will be employed during the test. Moreover, the smart wall usage does not modify the user experience related to the migration and the merging process. So two common PCs will be employed instead of one PC and a smart wall.

At the end of the task list execution, the user will compile a questionnaire and a debriefing session will be held, in order to discuss with the user about his impressions on the prototype usability level.

During the assessment test there are no strict usability requirements to accomplish because application prototypes are still not completely developed and the usability testing team will be more focused on finding requirements/suggestions for the following of the development cycle. However, users employed in the usability evaluation will fill a questionnaire by answering some questions with a numerical value (between 1 and 5). At the end, a mean usability value will be calculated for several aspects.

5.6.5 Task List

In Appendix A (par. Emergency Prototype) the proposed task list for the usability test on the emergency scenario prototype is available. The application will be tested by a single group of users and on two different PCs. At this stage, the migration toward a smart wall is not needed, because the simulation with two datasets is correctly displayed on a common PC.

5.6.6 Test Environment, Equipment and Logistics

The following test environment [D5.2] will be used during this usability evaluation:

- 2 Client PCs. Two PCs connected to the same LAN, able to access to the PC used as web server, and to the Internet will be employed by the users. Used PCs must have a web browser that supports Silverlight (for example Internet Explorer or Mozilla Firefox, with the required plugin).
- Web Server PC. A PC to be used as a web server is needed for the prototype testing. It must be accessible by the client PCs. Apache Tomcat web container and Emergency web application must be installed on this PC.

The questionnaire will be printed and then compiled by every user using a pen.

The moderator will take notes during the task list execution using a proper module.

No audio/video registration will be performed during this test and no specific logging tool will be used.

5.6.7 Test Moderator Role

Before the testing activity, the moderator should make sure that:

- The web server is up and running.
- The web application correctly runs on both of the client PCs.
- The required questionnaire is available.

During the task list execution, the moderator should talk with the user in order to know his thoughts during the test list execution. He should have a copy of the test list in order to take notes for every executed task.

At the end of the task list execution, the moderator will analyze the user questionnaire and direct the debriefing session. Also in this case, he should take notes of the discussion.

5.6.8 Data To Be Collected and Evaluation Measures

This paragraph contains a list of aspects to be evaluated during the task list execution.

- Simulations readability. The user will indicate with a numerical value between 1 and 5 the readability of the displayed simulation (in the case of a single dataset and in the case of two merged datasets).
- Migration process usability. The user will indicate the migration process usability with a numerical value between 1 and 5. Moreover, a user comment could be provided in order to individuate some usability problems.

5.7 Further work: reporting and second testing iteration

As stated in the paragraph 2.3, at the end of assessment usability tests, an analysis of the testing results will be performed (it will be contained in the D6.5).

In particular, for every tested application prototype a usability report will be compiled. It will contain a qualitative usability evaluation, a list of usability weak and strong points, and some suggestions for the following of the development cycle.

Validation usability tests will be performed when the OPEN project will be near to the end of its development cycle. Required test plans will be drawn up when it will be possible to individuate the final list of features that will be offered by the tested prototypes. Moreover, the experience gained during assessment tests could lead to some modifications of the validation test methodology described in this document.

An internal report will be provided with an update of the employed methodology (in the case it will be modified) and a detailed description of the test plans that will be applied during validation tests.

6 Programmability test plans

In this chapter the programmability test plans are collected. For each module, the Programmability Assessment template has been filled and a brief description of the Programmability Validation phase will be introduced if foreseen for the specific module.

6.1 Context Management Framework

Title:	OPEN Programmability Assessment Context Management Node		
ID:	OPEN Programmability_Context_Management_Node_1		
Version	**Issue**	**Date**	**Author**
1	1	08-06-2009	AAL/ Vodafone team
Module name Context Management Node			

General considerations	The context management framework (CMF), consisting of one management node and several agents distributed in the network, is able to collect, distribute and provide easy access to context information. The key points with respect to programmability of the CMF are: • **The collection** is done via small software components, called retrievers, interacting with raw sources of data, e.g. device discovery and Device Selection Map (or DiscoveryMap-see D3.1) and the CMF, and by processing units which may produce/infer non-measurable data types. The Trigger Management will utilize the processing capability as to infer the right moment in time, space and context for a potential service migration. • **The distribution** is handled by the internal of the CMF as needed, but can be influenced by ***scoping*** from the user, and support ***synchronous*** as well as ***asynchronous access*** to context information. • **The information model** used is the key to the interaction, and is extensible in terms of attributes or entities that may be produced or provided. • **Configuration of the CMF** is done via XML files which allow a flexible setup of the agents in the network, allowing also overlay network types to cross network domains. The configuration of an Agent is required for both the Context

	Management Node (CMN), and for each of the clients in the network connected to the CMN, but with different settings. • **The installation of the CMF** is ensuring that the required java packages are also installed, but do requires the user to setup an environmental variable "CMF_PATH" to whatever installation path that has been chosen. Currently this is not automatic. The intention at a later stage to run the CMF in an OSGi environment, simplifying the processing and retriever component setup procedure significantly and dynamically, but for now this is done statically via XML configuration files. By implementing the retrievers and processing units as OSGi service bundles, these can easily be *installed*, *started*, *stopped* and *uninstalled* as needed. By utilising remote-OSGi those bundles can even be found in remote, centralized repositories from where Context Agents can locate relevant bundles as needed making the Context Agent a full automatized, autoconfigured entity providing access to any information needed in a distributed environment.
Reference prototypes	"mobility support and context information management prototype" described in D3.2
Synthetic description	User: developer Supported context variables type, Manageable variables: variables described by the XML file Available tools for configuration: XML files
Context variable collection and distribution	Since the CMF is responsible for distribution and access to any general information types, the model used is key important to benefit from the system. The information model is relying on each information element having some or all of the following elements • **EntityIdentifier**: A unique identifier of the information element • **EntityType**: A type element that describes the information type

- **AttributeName** and **Value**: A set of attributes and values (in pairs). There may be many attributes per information. Currently simple values, like integers, floats, strings, etc. are supported values.
- **Metadata**: additional information about the information, e.g. timestamp.

So, for example, the battery voltage of a device could be described by

```
<Entity>
  <EntityIdentifier>someDeviceIdentifier</EntityIdentifer>
  <EntityType>Device </EntityType>
  <AttributeName>
    <name>BatteryVoltage</name>
    <type>float</type>
    <value><float>12.5</float></value>
    <metadata>
      <name>unit</name>
      <type>string</type>
      <value><string>Volt</string></value>
     </metadata>
     <metadata>
      <name>timestamp</name>
      <type>long</type>
      <value><long>1242208577546</ long></value>
     </metadata>
  </Attribute>
</entity>
```

The setup, as mentioned is currently done via XML files, and is too comprehensive to detail in this document. However, the intention is to move the CMF to OSGi framework, from where retrievers and processing units can be handled like separated services, hence leading to a minimum of setup via XML files.
A major requirement to the context management framework, is its ability to collect, distribute and provide access to general information.

With respect to the collection of information, the context management framework addresses this by allowing:

- Extensible collection method by allowing specifically written retrievers components to be plugged into the framework, which converts raw, measured data into a common data description model
- Extensible approach of inferring, deriving new types of context information based on measured information, by processing unit. Similar system to

	the retrievers, processing units are plugged in, and may provide any relevant information not directly measurable by retrievers. With respect to distribution, the framework o Informs the Context Management Node (CMN, the central entity in a CMF network configuration) about the availability of information at a given Context Agent. Subsequently, the CMN may be inquired about the location of context information and then followed by a direct request to the relevant entity for the value of the information. In this way, only information needed to be exchanged/communicated over the network is exchanged, hereby limiting the network traffic to only the absolute needed. With respect to the access of information o The access of information happens via a dedicated query language (Context Access LAnguage, CALA). This offers several ways of conducting searches for relevant information, mainly by *EntityIdentifier* and/or *EntityType* plus any additional *Attributes* that may be desired. o Furthermore, scoping of context queries, e.g. a query may be scoped via network domain, location or time. The CMF then takes this into account when accessing the rightful device. o Finally, the CMF offers synchronous and asynchronous access via either a request/response model or subscription/notification based approaches (either periodic or event based, with possibility of defining events in the subscription query).
Language/tool available for the module behaviour description	NA
Parametrical evaluation	*[to be filled by Vodafone evaluation team]* • Extensibility (capability of accepting and managing new variables): 1..5 *Note: consistency, runtime efficiency and robustness will be evaluated in the programmability validation phase*

Qualitative evaluation	*[to be filled by Vodafone evaluation team]* *A qualitative evaluation of the provided facilities will be provided*
Synthetic description of the adopted verification and validation strategies	The proposal is to verify the "consistency" of the CMF in the variable handling. In this context, consistency means the CMF ability of collecting and making available the context information in a consistent way with respect to its configuration. The proposed approach foresees the use of Siafu (a context generation tool): • Siafu will generate a new context variable • The CMF is configured using a proper XML in order to acquire this new variable • A CALA query is used in order to verify that the variable is correctly handled by the CMF

The proposed approach foresees the use of Siafu (a context generation tool):

- Siafu will generate a new context variable (e.g.: BatteryVoltage)
- The CMF is configured using a proper XML in order to acquire this new variable

Please refer to Appendix B for the foreseen test cases.

6.2 Migration Orchestration

Title:	OPEN Programmability Assessment Migration Orchestration		
ID:	OPEN Programmability_Migration_Orchestration_1		
Version	**Issue**	**Date**	**Author**
1	1	22/05/2009	Vodafone
Module name Migration Orchestration			

General considerations	This component synchronizes all of the actions that must be performed during a migration. The Migration Orchestration starts the migration when a trigger is

	received from the Trigger Manager or when an user requests a migration and then he makes sure that all of the required actions are performed either on migration source (i.e. the device where the application element is currently running) and migration target (i.e. the device where the application element will be migrated). The list of actions to be performed during a migration is not subjected to be modified/configured in order to avoid an unexpected behaviour of the platform.
Reference prototypes	NA
Synthetic description	NA
Context variable collection and distribution	NA
Language/tool available for the module behaviour description	NA
Parametrical evaluation	NA
Qualitative evaluation	NA
Synthetic description of the adopted verification and validation strategies	NA

The Migration Orchestration receives from the Trigger Manager or from an OPEN client the request for migration and carries out all the tasks in order to perform the migration: Since the list of tasks to be performed during a migration is not subjected to be modified/configured, the migration orchestration is not supposed to support the programmability, neither related to context variables collection and distribution nor related to rules definition. For this reason, for the Migration Orchestration no programmability validation will be performed.

6.3 Trigger Management

Title:	OPEN Programmability Assessment Trigger Management		
ID:	OPEN Programmability_Trigger_Management_1		
Version	**Issue**	**Date**	**Author**
1	1	12/05/2009	AAU/Vodafone
Module name Trigger Management			

General considerations	The ultimate programmability goal of the trigger management function is to be able to input new triggering rules into the module during deployment or even during runtime. This has, however, not been addressed in the currently implemented version. The envisioned programmability of new rules would be in the form of functions that map trigger management input, i.e. context information, to configuration scores. A configuration is a specification of devices in use, network technologies in use and placement of applications components in the currently running architecture. The trigger management generates triggers if the currently highest ranking configuration is different from the configuration in place. Each configuration rank is calculated from scores mapped from context information. The functions to be input in the trigger management to enable programmability would most probably be implemented as processing components in the context management framework. The new rules could have several objectives; • To provide new thresholds or reaction patterns for already available context information, in the end resulting in score values different from other or previous functions. • To allow triggers to be generated based on new types of context information. Basically this is a necessity in order for the platform to be able to react to new types of context information. Different ways of specifying rules and configuration exist: • Pure software implementation of processing components, typically programmed by the system

	developers or deployment managers. • One type of processing component may be a general one reading XML files into rules, which can then be specified by either a user or a developer. Whether this specification is done directly in XML or via graphical user interface outputting XML is currently undefined. • Specifications of new configurations would most probably be done through a graphical user interface, most probably by a system manager. The data would be stored in XML, which is then input to the trigger management.
Reference prototypes	"mobility support and context information management prototype" described in D3.2
Synthetic description	NA
Context variable collection and distribution	NA
Language/tool available for the module behaviour description	NA
Parametrical evaluation	NA
Qualitative evaluation	NA
Synthetic description of the adopted verification and validation strategies	NA

The Trigger Management is the key module for enabling the OPEN platform configurability. In fact, the user of the platform (application developer, system manager) should be enabled to set new triggering rules: e.g.: a new application of traffic news is implemented on top of the OPEN platform and the following triggering should be added: if there is an available device with a GPS positioning system, then migrate the traffic news application in this device. Different solutions could be implemented in order to enable the module configurability, as described in the "General considerations" field.

Different solutions address different users and enable different degree of configurability.

Since currently no programmability solutions are implemented, in the first iteration no programmability validation will be performed for the module.

6.4 Policy Enforcement

Title:	OPEN Programmability Assessment Policy Enforcement		
ID:	OPEN Programmability_Policy_Enforcement_1		
Version	**Issue**	**Date**	**Author**
1	1	22/05/2009	Vodafone team
Module name Migration Orchestration			
General considerations	The Policy Enforcement will be used in order to allow/deny a migration according to a set of rules. This module will support the following aspects of the programmability: • definition of variables to be considered. For example, a privacy level could be considered. In this case a required privacy level must be associated to application elements and an offered privacy level must be associated to every device. Each variable shall have a data type (int, double, String, etc.). • definition of rules applied to defined variables in order to allow/deny a migration. In the example of the privacy level, a rule could be: "if the target device offered privacy level is greater than the migrated application element required privacy level then allow the migration". A simple way of defining a rule is to allow the migration when the following expression is TRUE: "deviceVariable OPERATOR applicationElementVariable". OPERATOR can be one of the following operators: o "==" (for every type of variable) o ">", ">=", "<", "<=" (for numeric variables) o "OR", "AND", "NOT AND", "NOT OR" (for Boolean variables). For example, if the variable "*deny migration on mobile phone*" is defined for the migrated application and the variable "*is a mobile phone*" is defined for the device, the		

	migration must be denied only when both of these variables are true. The resulting rule will be: “*deny migration on mobile phone*” NOT AND “*is a mobile phone”*. E.g.: “*deny migration on mobile phone*”: True “*is a mobile phone”*: True Allow migration= True NOT AND True =False The Policy Enforcement module is still under definition, so at the moment its implementation is unavailable.
Reference prototypes	NA
Synthetic description	NA
Context variable collection and distribution	NA
Language/tool available for the module behaviour description	NA
Parametrical evaluation	NA
Qualitative evaluation	NA
Synthetic description of the adopted verification and validation strategies	NA

The Policy Enforcement will be used in order to allow/deny a migration according to a set of rules. This module should enable the programmability: the user of the platform (application developer, system manager) should be enabled to set new policy. Since currently no programmability solutions are implemented, in the first iteration no programmability validation will be performed for the module.

6.5 Mobility Support (Server side)

Title:	OPEN Programmability Assessment Mobility Support (Server side)		
ID:	OPEN Programmability_Mobility_Support_1		
Version	**Issue**	**Date**	**Author**
1	1	12/05/2009	AAU/Vodafone team

Module name Mobility Support	
General considerations	The mobility support module is not enabling programmability. It provides a static set of connectivity solutions based on the underlying network architecture. If programmability should be enabled in the mobility support module, it would be to allow for new connectivity solutions to be introduced in the set after development as plug-ins or alike, enforcing the migration system to be able to use them. But as this functionality is out of the scope of the considered use cases of the project, such requirements are not considered for the mobility support module.
Reference prototypes	NA
Synthetic description	NA
Context variable collection and distribution	NA
Language/tool available for the module behaviour description	NA
Parametrical evaluation	NA
Qualitative evaluation	NA
Synthetic description of the adopted verification and validation strategies	NA

The Mobility Support module is not enabling programmability. For this reason, for the Mobility Support no programmability validation will be performed.

6.6 Web UI Adaptation

Title:	OPEN Programmability Assessment – Web UI Adaptation		
ID:	OPEN Programmability_Web_UI_Adaptation		
Version	**Issue**	**Date**	**Author**
1	1	08/06/2009	ISTI-CNR team / Vodafone Team
Module name Web UI Adaptation			

General considerations	This module aims to adapt the user interface of the source device to the characteristics of the target device. In order to do this, this module accepts a CUI (Concrete User Interface Description), specified in XML-based language and describing the user interface of the application rendered on the source device *at a concrete level*: this means for instance to specify the UI elements not referring to a specific final implementation language, but in more abstract terms, just referring to the specific platform considered. Then, the Web UI Adaption module transforms a CUI (designed for the source device) into another CUI adapted to the characteristics of the target device. This transformation is performed by following a cost-based algorithm that calculates the cost of every UI element in a presentation (for instance, the cost of a textual string is the number of pixels occupied by it on the screen) and then, depending on the total cost of the various presentations composing a UI, calculates a new CUI that is more suitable for the characteristics of the target device. In addition, from this new calculated CUI, this module generates the final user interface using a specific implementation language for delivering the UI on the target device. It is worth pointing out that, in order to enable programmability features on this Web UI adaptation module, it is possible for the user of the OPEN platform to e.g. specify/change the costs associated to various UI elements. Therefore, depending on the values specified by the user (e.g. the costs of some UI elements), the adaptation can deliver different results. Regarding the programmability features, this module allows for modifying the adaptation rules both for the *mapping* and the *splitting* transformations,

	which are defined below: • The **mapping** rules basically allow for transforming a specific UI element into another UI element. As an example of such rules we might consider the transformation of a radio-button (desktop platform) into another UI element, which is deemed more suitable to be rendered onto a mobile platform: for instance, if the cardinality of a radio-button is higher than a certain threshold, the radio-button might be transformed into a pulldown-menu (on a mobile platform), since the latter element occupies less screen space. Therefore, in order to have programmable mapping rules, such transformation rules should be modifiable. • Through the **splitting** rules, this module can change the structure of a presentation. Then, with such rules, a single presentation (e.g.: for a desktop platform) can be translated into multiple, smaller presentations, which will be rendered onto a mobile platform. Then, the splitting rules specify the rules according to which a certain presentation will be split into multiple presentations (since e.g. the original presentation does not fit the capabilities of the target device). Therefore, in order to have programmable splitting rules, such rules should be modifiable.
Reference prototypes	Web migration prototype, described in D2.1. It can handle desktop web applications that are well-designed (e.g. they comply with W3C standards) and can be specified at a concrete level.
Synthetic description	User: service provider, migration platform administrator Supported context variables: the Web UI Adaptation module basically handles variables that model the various aspects managed by the adaptation algorithm. Such aspects basically refer to device-related characteristics like e.g. the 'cost' of a graphical UI element (e.g.: the number of pixels needed for rendering it on the screen), the number of characters that can be contained in a single line visualized on the device screen, the interaction capabilities of a certain device, etc.. Therefore, this module basically manages variables that can be represented by integer values.

	Available tools for configuration: a graphical tool is available, together with a configuration file. They are both currently subject to further improvements. Workflow patterns supported: NA
Context variable collection and distribution	• User agent information (for identifying the target device) + WURFL repository (for retrieving more detailed information about the various characteristics of a certain device) • Device description file specifying the characteristics of a certain device (this information is supposed to be exchanged between devices during the device discovery phase)
Language/tool available for the module behaviour description	A graphical tool is available for manipulating the variables that can be handled in a programmable way by the Web UI Adaptation module, through the mapping rules and the splitting rules. Among such variables we cite e.g. variables like the cost of the various elements of a graphical UI, the tolerance (number of allowed scrollings within a single graphical presentation), the number of characters that can be contained in a single line, etc.. In addition, such variables (together with their current values) are also specified in a configuration file.
Parametrical evaluation	*[to be filled by Vodafone evaluation team]* • Consistency (capability of specifying the module behavior in a synthetic way): *1..5* o weight: 1 • Fulfillment (capability of specifying the required workflow patterns): *1..5* o weight: 3 • Usability (usability of the provided tool): *1..5* o weight: 2
Qualitative evaluation	*[to be filled by Vodafone evaluation team]* A qualitative evaluation of the provided facilities will be provided
Synthetic description of the	Some test cases will be executed on the configuration tool.

adopted verification and validation strategies	A group of meaningful mapping and splitting rules will be tested in order to evaluate the tool consistency (by checking that the adapter behavior is the expected one). Moreover, an informal usability evaluation will be performed.

The proposal is to test the web UI Adaptation programmability: a group of meaningful mapping and splitting rules will be tested in order to evaluate the tool consistency (by checking that the adapter behavior is the expected one). Moreover, an informal usability evaluation will be performed.

Please refer to Appendix B for the foreseen test cases.

6.7 Server side Application Logic Reconfiguration

Title:	OPEN Programmability Assessment Server side Application Logic Reconfiguration		
ID:	OPEN Programmability_Application_Logic_Reconfiguration_1		
Version	**Issue**	**Date**	**Author**
1	1	08/06/2009	ClU/Vodafone team
Module name Application Logic Reconfiguration			

General considerations	This module is responsible for the adaptation of the application logic during runtime. Application logic is realized by components which interact through interfaces. Thus, the task of this module is to change the wiring of the components and their internal behaviour like introduced in deliverable D4.1. This module is the key module for enabling the OPEN applications programmability (refer to the example after the template).
Reference prototypes	The PacMan prototype as described in D4.3
Synthetic description	User: application developer Supported context variables type: variables used for the module configuration are currently hard-coded, so it is possible to use any type of variable and object supported by the programming language. Manageable variables: in the current prototype, the module is not able to handle context information in a

generic way. If context information has to be used, it has to be hard-coded into the components. In fact, these variable types have to be defined during development time of the application.
In the future version, it will be possible to make use of all kinds of context variables.

Available tools for configuration: in the current version, no tool support is provided. In future versions, some kind of configuration files can be used to describe the reconfiguration rules, also considering reconfiguration conditions based on context information, like for example:

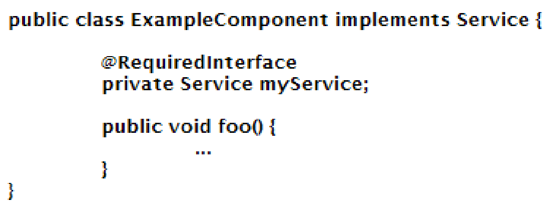
- use components on those devices where battery>50%

The goal is also to provide a tool which shows a graphical representation of the current system configuration in order to ease application administration.

Workflow patterns supported:

- Sequence
- Parallel split
- Synchronization
- Exclusive choice
- Simple merge

As already mentioned before, the application logic is built out of interacting components, currently implemented based on OSGi and Java. What the application developer does is to implement the components and define their required and provided interfaces within the code of the component. Required interfaces are given by an annotated variable, and provided interfaces by implementing the according interfaces:



```
public class ExampleComponent implements Service {

            @RequiredInterface
            private Service myService;

            public void foo() {
                        ...
            }
}
```

Furthermore, the developer can define an integer value representing the quality of service of the interface

	implementation. The ALR component will automatically inject the required instance into the given variable as soon as an according instance becomes available. Furthermore, it will replace an instance, if another instance with a higher priority becomes available. As soon as all required instances are injected into the according variables, the ALR module will notify and start the component. Every time a new component becomes available, the ALR will check if a rewiring of components or a replacement of a component is necessary. The result is an application logic implementation which changes its behavior during runtime. For these reasons, each type of workflow patterns can be realized. However, they have to be implemented by the component developers. The parallel split pattern for example can be realized by just calling methods at two different components. The synchronization pattern on the other hand can be realized by a component which waits until all execution threads to synchronize have called a method at that component. A component can implement the exclusive choice pattern by deciding which component to call next based on available information. It is not intended for the OPEN project to integrate a workflow specification language into the ALR module. Thus, the specification of the workflow will still take place in the code of the components. But for a future version it is intended to have an application specification where rules can be specified defining how the components are wired and adapted based on context information, like already mentioned above.
Context variable collection and distribution	NA
Language/tool available for the module behaviour description	NA
Quantitative evaluation	NA
Qualitative evaluation	NA

Synthetic description of the adopted verification and validation strategies	NA

The Application Logic reconfiguration is the key module for enabling the OPEN applications programmability. In fact, the user of the platform (application developer, system manager) should be enabled to define the application logic depending on the context variable collected by the OPEN middleware. E.g.: a new application of traffic news is implemented on top of the OPEN platform and the following logic should be implemented:

- if the device in which the application is running does not provide the user location, then visualize the traffic news from the newest to the oldest.
- if the device in which the application is running provides the user location, then visualize the traffic news form the nearest to the farthest.

Currently, the application logic is embedded in the code and no tools enabling the programmability are implemented. So, in the first iteration, no programmability validation will be performed for the module.

6.8 Multicore GUI Toolkit

Title:	OPEN Programmability Assessment Multicore GUI Toolkit		
ID:	OPEN Programmability_ Multicore_GUI_Toolkit_1		
Version	**Issue**	**Date**	**Author**
			NEC/Vodafone team
Module name Namuco (Multicore GUI Toolkit)			

General considerations	The Namuco GUI toolkit is a Java framework that allows the creation of graphical user interfaces according to capabilities of the target device and context information (e.g. battery state). Namuco will offer three distinct configurations regarding GUI widget and font sizes, which will be chosen according to the display resolution of the target device. Additionally context information regarding CPU performance determines whether animation effects will be enabled and to which extent.
Reference prototypes	NA

Synthetic description	We distinguish two kinds of users. One is an application developer who uses Namuco to build a GUI for his program. The second one is an end user who uses an application which is written for the OPEN platform and makes use of Namuco. The Namuco library provides several means for configuration: the prototype application and Namuco GUI appearance can be configured to use certain GUI features. This can be done on several levels: 1. In source code, by setting specific flags that disable/enable certain behaviours/features. 2. At run-time via user interaction/input: the user can use several GUI elements to configure the behaviour of the running application. This interface is provided by the prototype application implementation. 3. The system can be extended to support configuration files (which are written, e. g., in XML) that describe available capabilities of the target device or user preferences regarding the GUI appearance. Rules for adaptation are implemented inside certain Namuco modules. The current configuration settings for the GUI appearance and behaviour will be stored in a dedicated Java class in the form of configuration variables. Upon application start-up a module inside Namuco will retrieve all required information about device capabilities which do not change at run-time and set the corresponding configuration variables. During run-time another module will keep track of all relevant dynamic context variables (which change during run-time (e.g. battery state)) and will disable or enable certain GUI features accordingly (e.g. disable power- and performance-consuming animations).
Context variable collection and distribution	Namuco itself will not distribute context information but rather request context data from other OPEN platform modules. These context variables will be mainly related to the device on which the application that uses Namuco will run on. While variables like screen resolution and certain CPU characteristics (e.g. performance indicators like clock frequency or benchmark results) will not change at run-time, other information like number of available CPU cores or battery state is dynamic and may trigger changes in the behaviour of the Namuco library

	and int the appearance of the application using it. The Namuco library will not provide an interface to retrieve hardware and context information, as this information should be provided by other dedicated parts of the OPEN platform.
Language/tool available for the module behaviour description	NA
Quantitative evaluation	NA
Qualitative evaluation	NA
Synthetic description of the adopted verification and validation strategies	NA

For the first iteration, no programmability validation is foreseen for this module.

6.9 Further work: reporting and second testing iteration

The results of the first iteration of the programmability evaluation in term of strengths and weaknesses of the programmability approaches proposed for the OPEN modules will be reported in D6.5. The aim of the evaluation is to provide useful feedbacks and inputs for further improvement of the modules.

The Programmability Validation phase will be carried out for:

- CMF
- Web UI Adaptation

Moreover, a deeper analysis on the solution implemented by the Application Logic Reconfiguration module will be performed, in order to figure out possible enhancements.

For the Trigger Management and Policy Enforcement modules, which have been recognized as key for enabling the programmability, some possible solutions will be analyzed.

When the final OPEN prototype will be available, the second (and final) evaluation phase will verify the quality of the final programmability solution. The template introduced in this document can also be used for the final evaluation iteration.

7 Technological test plans

This section defines the test plans of the three prototypes that will be evaluated from a technological point of view during the first iteration. The complete set of test cases is listed in the Appendix C.

7.1 From D2.1: Web migration test plan

Test plan Identifier:	OPEN Technological test plan *Web migration*	
Version	**Date**	**Author**
3.0	09/06/09	Vodafone Italy / CNR
Test plan section		
References	OPEN D6.3 – Indicators for technical evaluation - Test plan format OPEN D2.1 – Early infrastructure for migratory interfaces - The prototype definition	
Lesson learned from previous experiences	Currently none	
Test items	Testing is based on some test cases, following the format previously described. It will involve a subset of the technical indicators and of the specific requirements listed in the testing methodology description	
Risk	To be verified before starting: connectivity and security policies that could impact testing. To avoid other issues: ▪ Devices have to be attached to the same LAN. ▪ Web contents have to follow W3C specs.	

Features to be tested	• These technical indicators will be tested: Availability, Reliability, Performances, Accessibility and Adherence to standards. • This subset of requirements from D1.1 can be applied to the prototype: 86-6-82-62-157-54-34-162-163-106-74-61-115-90-156-80-66-40-20-33. • Additional requirements: A1 - Image size must fit the screen of every kind of device allowed A2 - Page has to be entirely loaded for a good user experience
Approach	The test plan has been written based on CNR (owner of the prototype) contribution about the product description and the necessary info for testing purposes about it. Test cases are drafted, checked and approved by CNR and Vodafone Italy.
Test environment	Testing will be executed in VF-IT innovation labs, remotely linked to CNR server. Prototype version available at the beginning of testing will be frozen for the whole testing timeframe. The devices involved will be a PC and a PDA provided by VF-IT.
Item pass/fail criteria	Indicators will be evaluated after the execution of testing and after the analysis of logs. Requirements have simple Y/N passing criteria.
Test deliverables and reporting	Reporting will follow the testing experience, generating a section of next D6.5.
Remaining test tasks	Future developments of the prototype are currently foreseen. The decision of what to test in the future will depend on what is the prototype status at the testing timeframe and on the results of the evaluation.
Staffing and training needs	People involved in testing needs an overall technical background, with at least high level network and informatics skills, and a detailed knowledge about the OPEN project.

Roles and responsibilities	VF-IT testing team will execute the testing, while CNR will support them for the setup and in case of issues, necessities, problems, etc.
Schedule	Proposed timeframe: W28-30 (6th – 24th of July). Tests will be within the window from around 9 to 18 CET, from Monday to Friday.
Post trial analysis	Evaluation of indicators measured is necessary to provide feedback for future developments.

7.2 From D3.2: Mobility support and context information management test plan

Test plan Identifier:	OPEN Technological test plan *Mobility support*	
Version	**Date**	**Author**
4.0	17/06/09	*Vodafone Italy / Aalborg*
Test plan section		
References	OPEN D6.3 – Indicators for technical evaluation - Test plan format OPEN D3.2 – System support for application migration - The prototype definition	
Lesson learned from previous experiences	Currently none	
Test items	Testing is based on some test cases, following the format previously described. It will involve a subset of the technical indicators and of the specific requirements listed in the testing methodology description.	
Risk	Currently not particular ones	
Features to be tested	• A preliminary test will be performed to validate the basic prototype operations. • These technical indicators will be tested: Availability, Reliability, Performances. • This subset of requirements from D1.1 can be applied to the prototype: 7-86-82-54-74-61-131-80-66-91. • Additional requirements: A1 - Image size must fit the screen of every kind of device allowed A2 - The offline-online migration must be triggered by battery too	

Approach	The test plan has been written basing on Aalborg (owner of the prototype) contribution about the product description and the necessary info for testing purposes about it. Test cases are drafted, checked and approved by Aalborg and Vodafone Italy.
Test environment	Testing will be executed in VF-IT innovation labs. Prototype version available at the beginning of testing will be frozen for the whole testing timeframe. The devices involved will be a laptop and a fixed workstation.
Item pass/fail criteria	Indicators will be evaluated after the execution of testing and after the analysis of logs. Requirements have simple Y/N passing criteria.
Test deliverables and reporting	Reporting will follow the testing experience, generating a section of next D6.5.
Remaining test tasks	Future developments of the prototype are currently foreseen, since it is not fully completed. The decision of what to test in the future will depend on what is the prototype status at the testing timeframe and on the results of the evaluation.
Staffing and training needs	People involved in testing needs an overall technical background, with at least high level network and informatics skills, and a detailed knowledge about the OPEN project.
Roles and responsibilities	VF-IT testing team will execute the testing, while Aalborg will support them for the setup and in case of issues, necessities, problems, etc.
Schedule	Proposed timeframe: W37-39 (7th – 25th September). Tests will be within the window from around 9 to 18 CET, from Monday to Friday.
Post trial analysis	Evaluation of indicators measured is necessary to provide feedback for future developments.

7.3 From D3.2: Device selection map test plan

Test plan Identifier:	OPEN Technological test plan *Device selection map*	
Version	**Date**	**Author**
3.0	09/06/09	Vodafone Italy / CNR
Test plan section		
References	OPEN D6.3 – Indicators for technical evaluation - Test plan format OPEN D3.2 – System support for application migration - The prototype definition	
Lesson learned from previous experiences	Currently none	
Test items	Testing is based on some test cases, following the format previously described. It will involve a subset of the technical indicators and of the specific requirements listed in the testing methodology description	
Risk	To be verified before starting: connectivity and security policies that could impact testing. Devices have to be attached to the same LAN. Web contents have to follow W3C specs.	
Features to be tested	• These technical indicators will be tested: Availability, Reliability, Performances, Accessibility and Adherence to standards. • This subset of requirements from D1.1 can be applied to the prototype: 86-157-61-63-90-20-33.	
Approach	The test plan has been written basing on CNR (owner of the prototype) contribution about the product description and the necessary info for testing purposes about it. Test cases are drafted, checked and approved by CNR and Vodafone Italy.	

Test environment	Testing will be executed in VF-IT innovation labs, remotely linked to CNR server. Prototype version available at the beginning of testing will be freeze for the whole testing timeframe. The devices involved will be a PC and a PDA provided by VF-IT.
Item pass/fail criteria	Indicators will be evaluated after the execution of testing and after the analysis of logs. Requirements have simple Y/N passing criteria.
Test deliverables and reporting	Reporting will follow the testing experience, generating a section of next D6.5.
Remaining test tasks	Future developments of the prototype are currently foreseen. The decision of what to test in the future will depend on what is the prototype status at the testing timeframe and on the results of the evaluation.
Staffing and training needs	People involved in testing needs an overall technical background, with at least high level network and informatics skills, and a detailed knowledge about the OPEN project.
Roles and responsibilities	VF-IT testing team will execute the testing, while CNR will support them for the setup and in case of issues, necessities, problems, etc.
Schedule	Proposed timeframe: W28-30 (6th – 24th July). Tests will be within the window from around 9 to 18 CET, from Monday to Friday.
Post trial analysis	Evaluation of indicators measured is necessary to provide feedback for future developments.

7.4 Further work: reporting and second testing iteration

Consequently to the execution of the technical evaluation (following the previous test plans), a reporting phase will lead to the drafting of the D6.5 for the first evaluation summary of results. This will be an input for corrections and for further work of developers. When the second set of prototypes, representing the final product of the OPEN project, will be available, a second evaluation phase will test the complete set of features in a (at least partially) integrated system. Since D6.4 could only shape the current technical validation, it is worth to foresee an internal report to draft for the second set of prototypes an updated test plan, which depends on future developments.

Note that the first experience can also underline some aspects related to the pure execution of tests (e.g. configuration matters, equipment necessity), which have eventually to be included and mentioned in the first report phase (action point for the reporting team) to eventually update the methodology in an introductive section of this internal report. Timeline foreseen for the report should currently be after the conclusion of the second phase of development (M24). The only requirement that can currently be presented to the second iteration developers is, of course when is possible, to keep developing internal log/tools to monitor and record the technical indicators previously described.

Appendix A

This appendix collects usability questionnaires and usability task lists.

I. *Social Game Questionnaire*

Question	Description	Answers
1	How often do you play video games?	□ Never □ Rarely □ Weekly □ Almost daily
2	Which kind of game do you prefer?	□ First Person Shooter □ Racing □ Strategy □ Arcade □ Other, specify -
3	Which kind of device do you prefer?	□ Portable console □ Fixed console □ PC □ Mobile phone
4	Do you usually play online or offline?	□ Always online □ Both □ Always offline
5	How often do you use other services like chat, betting, video streaming?	□ Never □ Rarely □ Weekly □ Almost daily
6	Are you keen on Formula 1 and other racing sports	□ Not so much □ A bit □ A lot
7	Would you be interested in mixing these two application areas?	□ Not so much □ A bit □ A lot □ It depends on… specify
8	Is it clear the game concept?	□ yes □ no, specify why?
9	Are the services around the game (chat, betting, TV, web) clear?	□ yes

		□ no, specify why?
10	Are there additional functionalities you would like?	□ yes, specify □ no
11	Do you like the look and feel?	□ yes □ no, specify why?
12	How do you consider the following UI options: User is able to play and chat/browse/watch TV in the meanwhile	□
13	How do you consider the following UI options: User can share a screen (e.g. pub) without sharing private info	□
14	How do you consider the following UI options: Users can communicate each other before, during and after the game	□
15	Do you appreciate the level of security? Has it an acceptable impact on the applications proceeding (betting security, access to STB and screens…)?	□ yes □ no, specify why?
16	Would you like to extend the game to other sports?	□ yes, specify – □ no
17	Are there additional relevant environments to describe?	□ yes, specify – □ no

II. Emergency Application Questionnaire

Question	Description	Rationale	Answers
1	Which job category do you belong to?		□
2	How many years of work experience did you make?		□ 0-5 □ 5-10 □ 10-20 □ More than 20
3	How would you define your technical background?		□ Very basic □ Average □ Good □ Expert
4	Which kind of device do you use at work?		□ Mobile phone □ PC □ PDA □ Other, specify
5	Are you ever in dangerous situations at work, requiring a fast reaction?		□ Never □ Almost never □ Often
6	What do you think about the importance (in such situations) of simulations, remote access and control, coordination		□ Not so high □ High □ Extremely high
7	Is it clear the application concept?	General user feeling	□ yes □ no, specify why?
8	How satisfied are you with the OPEN migration?	The user is asked for his overall impression of the usability of the OPEN migration platform.	□ 1 - Very satisfied □ 2 - Satisfied □ 3 - Average □ 4 - Disappointed □ 5 - Very disappointed
9	How usable did you find the registration of devices to the OPEN migration platform?	This more concrete question aims at subjective feelings of how easy it was to register the video wall.	□ 1 - Very easy □ 2 - Easy □ 3 - Average □ 4 - Hard □ 5 - Very hard

10	Would you like to migrate user interfaces in our daily applications?	In this question we ask the user whether the OPEN migration platform as a whole package is useful for his/her work.	□ yes □ no
11	What application elements are missing in order to boost performance in the EOC?		□
12	Which RIA platform do you prefer?		□ AJAX □ FLEX □ Silverlight □ Other - Specify □Indifferent/Don't know
13	Do you like the look and feel of EOC-application?		□ yes □ no, specify why?
14	Are there additional functionalities you would like to add?		□ yes, specify □ no
15	Would you like to enlarge the application to other environments?		□ yes, specify – □ no
16	Comments	Here the user can enter free style comments. User comments might give useful hints towards specific usability problems, not foreseen in the design of the evaluation or of the original OPEN migration platform.	□

III. Web Migration Task Lists

Group A Task List				
Task ID	**Device / Conditions**	**Web Application**	**Task Description**	**Comments**
SH-A-1	PDA – without OPEN	Shopping Assistant	Buy a specific product	In order to evaluate the web application usability, the task will not contain the exact name of the product, but a description of a product characteristic. In this way the user is forced to navigate some pages of the web application in order to find the required product. Moreover, during the product purchase, the user will be able to evaluate the web form rendering.
-	-	Shopping Assistant	Compile a questionnaire about the shopping assistant UI used on the PDA without OPEN	
SH-A-2	PC	Shopping Assistant	Insert in the cart the cheapest product	The purpose of this task is to make sure that the user visits several pages of the web site. In such way it is possible to obtain a good evaluation of the web site

				navigability.
SH-A-3	PC	Shopping Assistant	Migrate to the PDA	This task is performed to make sure that the web site state is correctly maintained during the migration.
SH-A-4	PDA	Shopping Assistant	Remove the product from the cart	With these tasks, the state maintaining during the migration PDA -> PC is tested.
SH-A-5	PDA	Shopping Assistant	Migrate to the PC	
SH-A-6	PC	Shopping Assistant	Choose a product and start the purchase operation (i.e. compile a part of the “purchase” form)	This additional migration is used in order to test the state maintenance during a form compilation.
SH-A-7	PC	Shopping Assistant	Migrate to the PDA	
SH-A-8	PDA	Shopping Assistant	Complete the transaction	
-	-	Shopping Assistant	Compile a questionnaire about the web UI used on the PC	The usability test of the web application from a PC is completed.
SH-A-9	PDA	Shopping Assistant	Insert the most expensive product in the cart	Task used for navigability evaluation.
SH-A-10	PDA	Shopping	Start the purchase	Forms and state

		Assistant	operation (i.e. compile a part of the "purchase" form)	keeping evaluation.
SH-A-11	PDA	Shopping Assistant	Migrate to the PC	
SH-A-12	PC	Shopping Assistant	Complete the transaction	
-	-	Shopping Assistant	Compile a questionnaire about the shopping assistant UI used on the PDA Compile a questionnaire about the migration functionalities	
W-A-1	PDA without OPEN	Wikipedia	Open the page: http://it.wikipedia.org/wiki/5_settembre (for not Italian users: http://en.wikipedia.org/wiki/5_September)	
W-A-2	PDA without OPEN	Wikipedia	Select "**Edit**" for the section "**Holidays and observances**", write some text and then select "**Cancel**"	

W-A-3	PDA without OPEN	Wikipedia	Select the link “**Freddie Mercury**” and read the page for a few minutes.	
-	-	Wikipedia	Compile a questionnaire about the Wikipedia UI used on the PDA without OPEN	
W-A-4	PC	Wikipedia	Open the page: http://it.wikipedia.org/wiki/5_settembre (English version: http://en.wikipedia.org/wiki/5_September)	
W-A-5	PC	Wikipedia	Select “**Edit**” for the section “**Holidays and observances**”	Forms evaluation
W-A-6	PC	Wikipedia	Write some text in the proposed text field	State keeping test
W-A-7	PC	Wikipedia	Migrate to PDA	
W-A-8	PDA	Wikipedia	Select “**Cancel**” in order to go back to the previous page.	

W-A-9	PDA	Wikipedia	Select the link "**Freddie Mercury**" and read the page for a few minutes.	
W-A-10	PDA	Wikipedia	Migrate to PC	
W-A-11	PC	Wikipedia	Continue to read the page for a few minutes.	
-	-	Wikipedia	Compile a questionnaire about the web UI used on the PDA Compile a questionnaire about the web UI used on the PC Compile a questionnaire about the migration functionalities	For usability evaluation over this web application, usability questionnaires are compiled at the end of the test, because they takes less time than the tests performed on the Shopping Assistant. Moreover, in order to get an accurate evaluation of the OPEN migration procedure, there is not a clean separation between PC and PDA usage.

Group B Task List

Task ID	**Device / Conditions**	**Web Application**	**Task Description**	**Comments**
SH-B-1	PDA	Shopping Assistant	Insert in the cart the cheapest product	The purpose of this task is to make sure that the user visits several pages of the web site. In such way it is possible to obtain a good evaluation of the web site navigability.
SH- B-2	PDA	Shopping Assistant	Migrate to the PC	This task is performed to make sure that the web site state is correctly maintained during the migration.
SH- B-3	PC	Shopping Assistant	Remove the product from the cart	With these tasks, the state maintaining during the migration PC -> PDA is tested.
SH- B-4	PC	Shopping Assistant	Migrate to the PDA	
SH- B-5	PDA	Shopping Assistant	Choose a product and start the purchase operation (i.e. compile a part of the "purchase" form)	This additional migration is used in order to verity the state keeping during a form compilation.
SH- B-6	PDA	Shopping Assistant	Migrate to the PC	
SH- B-7	PC	Shopping Assistant	Complete the transaction	

-	-	Shopping Assistant	Compile a questionnaire about the web UI used on the PDA	The usability test of the web application from a PC is completed.
SH- B-8	PC	Shopping Assistant	Insert the most expensive product in the cart	Task used for navigability evaluation.
SH- B-9	PC	Shopping Assistant	Start the purchase operation (i.e. compile a part of the "purchase" form)	Forms and state keeping evaluation.
SH- B-10	PC	Shopping Assistant	Migrate to the PDA	
SH- B-11	PDA	Shopping Assistant	Complete the transaction	
-	-	Shopping Assistant	Compile a questionnaire about the shopping assistant UI used on the PC Compile a questionnaire about the migration functionalities	
SH- B-12	PDA – without OPEN	Shopping Assistant	Buy a specific product	In order evaluate the web application usability, the task won't contain the exact name of the product, but a

				description of a product characteristic. Moreover, during the product purchase, the user will be able to evaluate the web form rendering.
-	-	Shopping Assistant	Compile a questionnaire about the shopping assistant UI used on the PDA without OPEN	
W- B-1	PDA	Wikipedia	Open the page: http://it.wikipedia.org/wiki/5_settembre (English version: http://en.wikipedia.org/wiki/5_September)	
W- B-2	PDA	Wikipedia	Select “**Edit**” for the section “**Holidays and observances**”	Forms evaluation
W- B-3	PDA	Wikipedia	Write some text in the proposed text field	State keeping test
W- B-4	PDA	Wikipedia	Migrate to PC	
W- B-5	PC	Wikipedia	Select “**Cancel**” in order to go back to the	

			previous page.	
W- B-6	PC	Wikipedia	Select the link "**Freddie Mercury**" and read the page for a few minutes.	
W- B-7	PC	Wikipedia	Migrate to PDA	
W- B-8	PDA	Wikipedia	Continue to read the page for a few minutes.	
-	-	Wikipedia	Compile a questionnaire about the web UI used on the PC Compile a questionnaire about the web UI used on the PDA Compile a questionnaire about the migration functionalities	For usability evaluation over this web application, usability questionnaires are compiled at the end of the test, because they takes less time than the tests performed on the Shopping Assistant. Moreover, in order to get a accurate evaluation of the OPEN migration procedure, there is not a clean separation between PC and PDA usage.
W- B-9	PDA without OPEN	Wikipedia	Open the page: http://it.wikipedia.org/wiki/5_settembre (for not Italian users: http://en.wikip	

			edia.org/wiki/5_September)	
W- B-10	PDA without OPEN	Wikipedia	Select “Edit” for the section “Holidays and observances”, write some text and then select “Cancel”	
W- B-11	PDA without OPEN	Wikipedia	Select the link “Freddie Mercury” and read the page for a few minutes.	
-	-	Wikipedia	Compile a questionnaire about the Wikipedia UI used on the PDA without OPEN.	

IV. Social Game Task Lists

Group A Task List			
Task ID	**Device / Conditions**	**Task Description**	**Comments**
SG-A-01	PC	Login to the Social Game web application.	These functionalities of the Social Game are tested only on the PC because in the current prototype implementation it is not possible to migrate them on the mobile phone.
SG-A-02	PC	Send a message to the chat	
SG-A-03	PC	Bet 25 € on Felipe Massa	
SG-A-04	PC	Answer the question: who completed the fastest lap?	
SG-A-05	PC	Start the full screen view of IPTV simulator	
SG-A-06	PC	Change IPTV channel	
SG-A-07	PC	Close the full screen view of IPTV simulator	
SG-A-08	PC	Start playing on the Racing Game	In the group A, users start using the Racing Game from the PC
SG-A-09	PC	Migrate the controls of the Racing Game to the mobile phone	
SG-A-10	Mobile Phone	Continue the race by controlling the car from the mobile phone	
-	-	Compile a questionnaire	

Group B Task List			
Task ID	**Device / Conditions**	**Task Description**	**Comments**
SG-B-01	PC	Login to the Social Game web application.	These functionalities of the Social Game are tested only on the PC because in the current prototype implementation it is not possible to migrate them on the mobile phone.
SG-B-02	PC	Send a message to the chat	
SG-B-03	PC	Bet 25 € on Felipe Massa	
SG-B-04	PC	Answer the question: who completed the fastest lap?	
SG-B-05	PC	Start the full screen view of IPTV simulator	
SG-B-06	PC	Change IPTV channel	
SG-B-07	PC	Close the full screen view of IPTV simulator	
SG-B-08	PC	Migrate the controls of the Racing Game to the mobile phone	In the group B, users start using the Racing Game from the mobile phone (the game control via mobile phone, however, is enabled using the PC interface of the Social Game).
SG-B-09	Mobile Phone	Start playing on the racing game controlling the car from the mobile phone	
SG-B-10	PC	Continue playing on the racing game controlling the car from the PC keyboard	
-	-	Compile a questionnaire	

V. Emergency Prototype

Task ID	Device / Conditions	Task Description	Comments
BS-01	PC1	Start the flooding simulation	The fist available dataset regards a flooding simulation [D5.2]
BS-02	PC1	Migrate the flooding simulation to the second PC	
BS-03	PC1	Start the traffic simulation	The second available dataset regards a traffic simulation [D5.2]
BS-04	PC1	Migrate the traffic simulation to the second PC	
BS-05	PC2	Start the merged simulation	
-	-	Compile a questionnaire	

Appendix B

In this appendix the test cases for the Programmability Assessment are collected.

I. *Context Management Framework*

ID	Programmability_CMF_1
Module	CMF
Description	The objective of the test is verifying that the CMF properly collect and make available the context information.
Input	• Siafu will generate a new context variable (e.g.: BatteryVoltage) • The CMF is configured using a proper XML in order to acquire this new variable (e.g.: the XML provided in the example)
Expected output	A CALA query to the CMF should return the value provided to the CMF by the Siafu
Actual output	Output obtained by the Query
General considerations	If the result of the query is the value provided by the Siafu, the variable collection and distribution is correct

II. Web UI Adaptation

ID	Programmability_WebUIAdaptation_1
Module	Web UI Adaptation
Description	The objective of the test case is verifying the effect of modifying a **mapping rule** in the Web UI Adaptation module when passing from a desktop platform to a mobile one. This example rule can e.g. transform a (desktop) radiobutton into a different UI object (e.g. a pulldownmenu, which occupies less screen space) depending on the cardinality of the possible choices of the considered radiobutton element. The concerned rule takes in input the maximum number of options that radiobuttons can have (let's call it **MaxCard**). If the cardinality of the selection items of the considered radiobutton (let's call it **Card(radioButton)**) is higher, the radioButton is transformed into a pull-down menu onto the mobile platform; otherwise the radiobutton is maintained in the mobile platform.
Input	**MaxCard** An integer value representing the maximum cardinality of options in radiobutton elements
Expected output	For each radioButton existing in the desktop UI: If Card(radioButton)>MaxCard the radiobutton is transformed into a pull down menu. Then, a pulldownmenu should appear on the target mobile device UI instead of the original radioButton. If Card(radioButton)<=MaxCard the radiobutton is maintained in the mobile platform (no transformation of the UI object takes place)
Actual output	Output obtained

General considerations	Comments derived by the test result

ID	Programmability_WebUIAdaptation_2
Module	Web UI Adaptation
Description	The objective of the test case is verifying the effect of modifying a **mapping rule** in the Web UI Adaptation module (when passing from desktop to mobile). In particular, the concerned example rule allows for specifying the maximum dimension that images can have when doing the adaptation from a desktop UI to a mobile one. In particular, the concerned rule takes in input **Image_MaxDim**, which is supposed to represent the **maximum dimension that images can have on the mobile UI=(MaxWidth, MaxHeight).** This rule can transform the various images existing in the desktop UI by resizing them according to the specified size value **Image_MaxDim**, defined in terms of width and height). This resizing process should be carried out in such a way that the original image aspect ratio will be preserved, and the size of the adapted image should be equal (or less) of the specified parameter **Image_MaxDim.**
Input	**Image_MaxDim** Maximum image dimension that images can have on the mobile device. It is specified through a couple of integers (Image_MaxWidth, Image_MaxHeight) representing the maximum width and the maximum height (in pixels) that an image can assume on the mobile target platform.
Expected output	For each image belonging to the desktop UI: if the dimension of the considered image is bigger than Image_MaxDim, the considered image is resized

	according to the specified dimension Image_MaxDim. Otherwise, the considered image maintains its own dimension on the mobile target platform (then, no resizing transformation is performed).
Actual output	Output obtained
General considerations	Comments derived by the test result

ID	Programmability_WebUIAdaptation_3
Module	Web UI Adaptation
Description	The objective of the test case is to understand the effect of modifying a **splitting rule** in the Web UI Adaptation module (when passing from desktop to mobile). Supposing that **Height_Resolution** is the resolution of the actual height of the mobile screen expressed in pixels. This example rule allows for modifying the height associated with the mobile target device, by multiplying the device screen's actual height of a factor (namely, the **Tolerance** parameter). In this way, the height associated with the device screen is considered as "extended" through this tolerance factor to the screen height, so as to be able to include more UI objects in the same presentation, and also enabling a number of (vertical) scrolling actions on a presentation. If the total cost of the presentation (namely, the sum of the costs of the various objects contained in the same presentation, let's call it **Total_Cost**) exceeds the "extended" capability of the mobile device, then a splitting transformation is carried out. This means that multiple pages are created on the mobile platform (starting from the original desktop single presentation), together with additional links for navigating between the newly created pages;

	otherwise the presentation is not split.
Input	**Tolerance** This integer value represents the factor according to which the height resolution of the mobile device screen is multiplied. The goal is to enable an increased tolerance towards the device screen's height, in order to allow that more UI objects are contained in the same presentation, which can be accessed through a number of scrolling actions (on the vertical axis). Then, if Tolerance>1, a vertical scrolling is enabled in the target mobile UI.
Expected output	If ((**Tolerance** * **Height_Resolution**)> **Total_Cost**) then the presentation is split in multiple presentations, otherwise a single presentation is maintained on the target device.
Actual output	Output obtained
General considerations	Comments derived by the test result

ID	Programmability_WebUIAdaptation_4
Module	Web UI Adaptation
Description	The objective of the test case is to understand the effect of modifying a splitting rule in the Web UI Adaptation module (when passing from desktop to mobile). In particular, the concerned example rule focuses on the adaptation of a presentation containing a textual interactor, and how it can vary in a programmable way. More specifically, in this example rule it is shown to what extent the cost of a certain textual element can vary depending on the values of

	the specified programmable input parameters manipulated by the concerned splitting rule (namely, such parameters are ExpectedScreenWidth and LineCost, see below). Such a variation of the cost associated with the textual interactor might lead to a *possible* splitting (since such a splitting also depends on the costs of the other UI elements included in the presentation) of the presentation containing such interactor.
Input	ExpectedScreenWidth The number of characters that are expected to be contained in a single line of the screen. This property might be found within the list of device capabilities, but the user can also specify a value for it. LineCost The cost assigned to a single line. Generally it corresponds to the height resolution of the text expressed in pixels, but the user can also specify a value for it.
Expected output	If we consider a textual interactor (for instance, a quite long text) we can calculate the cost of the textual interactor in the following way: TextualInteractorCost = (NumLines * Line Cost) where NumLines is the number of characters contained in the textual interactor, divided by ExpectedScreenWidth. Since the total cost of the presentation containing the textual interactor can also contain other UI elements, the total cost of the presentation can be calculated in the following way: TotalCost = TextualInteractorCost + (Cost of the remaining part of the presentation) If TotalCost > (HeightResolution of the mobile device screen) then a splitting is carried out, and the original single presentation is transformed into multiple presentations on the mobile device. Otherwise no splitting action is performed.

Actual output	Output obtained
General considerations	Comments derived by the test result

Appendix C

I. Web Migration test cases

ID	OPEN Technological test plan Web migration TC1
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 86 - Migration should be triggered by the user
Input	
Expected output	
Actual output	
General considerations	

ID	OPEN Technological test plan Web migration TC2
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 6 - System should be able to trigger a migration
Input	

Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC3
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 82 - Migration should be automatic / system triggered. Based on previous settings by the user
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC4
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its

	functionalities are correctly performed. This test case aims to verify this requirement: 62 - Users want to use the migration process for triggering application actions, e.g. for joining a game
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC5
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 157 - The OPEN platform should be installed and listening for any device requesting migration
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC6
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 54 - It must be possible to continue my current service seamlessly across multiple devices
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC7
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 34 - Service content should be provided in a context aware manner
Input	
Expected output	
Actual output	

General consideration	

ID	OPEN Technological test plan Web migration TC8
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 162 - The OPEN platform should be able to maintain the data inserted by the user in the source device and show them in a consistent way after migration on the target device
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC9
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 163 - The OPEN platform should present the last

	data inserted by the user on the source device in the first presentation provided to the user in the target device
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC10
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 106 - OPEN should let me know where my data is. After it has migrated several times
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC11

Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 61 - The user does not want to care about networking aspects when trying to migrate
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC12
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 115 - OPEN enables the user to get, what s/he individually can handle, i.e. the information remains not only complete, but in terms of perceived complexity understandable after a migration
Input	
Expected output	
Actual output	

General consideration	

ID	OPEN Technological test plan Web migration TC13
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 156 - The input devices must be able support the same actions
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC14
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 80 - Users must be able to accept or deny a migration from a to b

Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC15
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 20 - Users need to discover devices in the vicinity.
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC16
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to

	check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: A1 - Image size must fit the screen of every kind of device allowed
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC17
Item	Specific requirements
Description	Web migration testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: A2 - Page has to be entirely loaded for a good user experience
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC18
Item	Availability
Description	Availability is monitored recording possible failures and their lasting while executing the prototype.
Input	Prototype is actively used during the whole working day and left in background during the following night, with an internal tool recording every kind of issue: migration failures, application failure, wrong device discovery, and so on
Expected output	There is no target value; the result will be evaluated after the closure of the testing timeframe.
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC19
Item	Reliability
Description	Reliability is monitored recording possible failures during a complete E2E execution of the prototype.
Input	A complete execution of the prototype (e.g. the access to a product and a following migration) is performed as many times as possible during the whole working day, with an internal tool recording every kind of issue. The percentage of complete executions without issues is the final result.
Expected output	There is no target value; the result will be evaluated after the closure of the testing timeframe.
Actual output	

General consideration	

ID	OPEN Technological test plan Web migration TC20
Item	Performance
Description	Timings and possible failures will be monitored and recorded for: Triggering, Migration, Application
Input	Execution of the prototype is performed while recording the internal logs during the whole working day. Values to measure: triggering time, migration time, application delay and jitter. Events to record: trigger failures, migration failures.
Expected output	There is no target value; the result will be evaluated after the closure of the testing timeframe.
Actual output	
General consideration	

ID	OPEN Technological test plan Web migration TC21
Item	Accessibility
Description	This test case aims to underline possible lacks of Web content accessibility.
Input	Web contents of the application are submitted to the accessibility analysis of Magenta tool; if no lacks of accessibility raise, further tools from D6.3 can be applied (W3C Validator, WAVE Web Access Evaluation Tool, Web Access Checker at ATRC).
Expected	These tools should not discover any accessibility

output	issues.
Actual output	Outcome from Magenta tool:
General consideration	

ID	OPEN Technological test plan Web migration TC22
Item	Adherence to the Standard
Description	This test case verifies the respect of W3C standards concerning (X)HTML tags.
Input	Web content is checked through the W3C website, verifying possible errors towards its specs.
Expected output	No incompatibilities with W3C specs should arise.
Actual output	Outcome from W3C website:
General consideration	

II. Mobility support test cases

ID	OPEN Technological test plan Mobility support TC0
Item	Prototype Initial Test
Description	This test case aims to validate the basic prototype operation. It consists of a set of YES/NO questions that verify if the prototype is carrying out or not the task which was designed for: video migration. The result of the test case will be PASSED if the response to all the questions are yes or NOT

	PASSED if any of them is no.
Input	Connect the source and destination prototype devices to the test scenario network by following carefully the prototype setting up instructions. Answer the following set of questions with YES or NOT: i) Is it possible to visualize the streaming video at the source device? ii) Is the streaming audio synchronized with the video at the source device? iii) Does the video/audio stream skips, cuts out or buffers? Trigger the migration application as indicated in the instructions and answer the following questions with YES or NOT: iv) Does the migration take longer than 4 seconds [1]? v) Is it possible to visualize the streaming video at the target device? vi) Is the streaming audio synchronized with the video at the target device? vii) Does the video/audio stream (at the target device) skips, cuts out or buffers?
Expected output	The expected output is PASSED.
Actual output	
General consideration	Any unexpected change in the setting up should be reported to the owner of the project before starting with the test plan. Any kind of unexpected error or delay during the test case should be included in the prototype test cases results.

ID	OPEN Technological test plan Mobility support TC1
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 7 - The user must be enabled to watch a program using his set top box and multiple screens
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC2
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 86 - Migration should be triggered by the user
Input	
Expected output	

Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC3
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 82 - Migration should be automatic / system triggered. Based on previous settings by the user
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC4
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test

	case aims to verify this requirement: 54 - It must be possible to continue my current service seamlessly across multiple devices
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC5
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 74 - Users must be able to migrate identified parts of the application to other devices e.g. high score list
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC6
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 61 - The user does not want to care about networking aspects when trying to migrate
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC7
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 131 - The offline-online migration must be triggered by network QoS parameters too
Input	
Expected output	

Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC8
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 80 - Users must be able to accept or deny a migration from a to b
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC9
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test

	case aims to verify this requirement: 66 - The user must be able to specify migration policies, e.g. automatic migration when switched off
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC10
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: 91 - OPEN should predict the data and applications needed when going mobile. Possible migration also for non-OPEN service providers
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC11
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: A1 - Image size must fit the screen of every kind of device allowed
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC12
Item	Specific requirements
Description	Mobility support testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test case aims to verify this requirement: A2 - The offline-online migration must be triggered by battery too
Input	
Expected output	

Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC13
Item	Technical Measurements: Availability
Description	The goal in this test case is to measure the performance of the system in terms of: - Availability → % of time in which the service is available. This performance will be measured under 3 different network conditions: dedicated network (no traffic), shared network and overloaded network.
Input	General input for indicators testing: Execution of the prototype is performed while recording the internal logs during the whole working day. Values to measure: triggering time, discovery time, migration time and application delay, jitter and synchronization. Events to record: jitter, number of fails, trigger failures, discovery failures, and migration failures. Repeat the previous step but adding the following network conditions: - Network utilization by other applications: 50% - Network utilization by other applications: 90% Traffic can be generated using LAN Tornado, Paessler Net Flow generator or other traffic generation software.
Expected output	Measurements of the commented parameters. (No expected target values in this phase).

Actual output	
General consideration	

ID	OPEN Technological test plan Mobility support TC14
Item	Technical Measurements: Reliability
Description	The goal in this test case is to measure the performance of the system in terms of: - Reliability → % of successful migrations This performance will be measured under 3 different network conditions: dedicated network (no traffic), shared network and overloaded network.
Input	General input for indicators testing: Execution of the prototype is performed while recording the internal logs during the whole working day. Values to measure: triggering time, discovery time, migration time and application delay, jitter and synchronization. Events to record: jitter, number of fails, trigger failures, discovery failures, and migration failures. Repeat the previous step but adding the following network conditions: - Network utilization by other applications: 50% - Network utilization by other applications: 90% Traffic can be generated using LAN Tornado, Paessler Net Flow generator or other traffic generation software.
Expected output	Measurements of the commented parameters. (No expected target values in this phase).
Actual output	

General consideration	

ID	OPEN Technological test plan Mobility support TC15
Item	Technical Measurements: Performances
Description	The goal in this test case is to measure the performance of the system in terms of: - Delay → Timings and possible failures will be monitored and recorded for: trigger, discovery, migration and application. - Jitter This performance will be measured under 3 different network conditions: dedicated network (no traffic), shared network and overloaded network.
Input	General input for indicators testing: Execution of the prototype is performed while recording the internal logs during the whole working day. Values to measure: triggering time, discovery time, migration time and application delay, jitter and synchronization. Events to record: jitter, number of fails, trigger failures, discovery failures, and migration failures. Repeat the previous step but adding the following network conditions: - Network utilization by other applications: 50% - Network utilization by other applications: 90% Traffic can be generated using LAN Tornado, Paessler Net Flow generator or other traffic generation software.
Expected output	Measurements of the commented parameters. (No expected target values in this phase).

Actual output	
General consideration	

III. Device selection Map Test cases

ID	OPEN Technological test plan Device selection map TC1
Item	Specific requirements
Description	Device selection map testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test aims to verify this requirement: 86 - Migration should be triggered by the user
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Device selection map TC2
Item	Specific requirements
Description	Device selection map testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its

	functionalities are correctly performed. This test aims to verify this requirement: 157 - The OPEN platform should be installed and listening for any device requesting migration
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Device selection map TC3
Item	Specific requirements
Description	Device selection map testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test aims to verify this requirement: 61 - The user does not want to care about networking aspects when trying to migrate
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Device selection map TC4
Item	Specific requirements
Description	Device selection map testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test aims to verify this requirement: 63 - OPEN should work with and without internet connection
Input	
Expected output	
Actual output	
General consideration	

ID	OPEN Technological test plan Device selection map TC5
Item	Specific requirements
Description	Device selection map testing will start by verifying the specific requirements, since they are primary to check what the prototype does, and if its functionalities are correctly performed. This test aims to verify this requirement: 20 - Users need to discover devices in the vicinity.
Input	
Expected output	

Actual output	
General consideration	

ID	OPEN Technological test plan Device selection map TC6
Item	Availability
Description	Availability is monitored recording possible failures and their lasting while executing the prototype.
Input	Prototype is actively used during the whole working day and left in background during the following night, with an internal tool recording every kind of issue.
Expected output	There is no target value; the result will be evaluated after the closure of the testing timeframe.
Actual output	
General consideration	

ID	OPEN Technological test plan Device selection map TC7
Item	Reliability
Description	Reliability is monitored recording possible failures during a complete E2E execution of the prototype.
Input	A complete execution of the prototype (e.g. the access to a product and a following migration) is performed as many times as possible during the whole working day, with an internal tool recording

	every kind of issue. The percentage of complete executions without issues is the final result.
Expected output	There is no target value; the result will be evaluated after the closure of the testing timeframe.
Actual output	
General consideration	

ID	OPEN Technological test plan Device selection map TC8
Item	Performance
Description	Timings and possible failures will be monitored and recorded for: Device discovery.
Input	Execution of the prototype is performed while recording the internal logs during the whole working day. Values to measure: discovery time. Events to record: discovery failures.
Expected output	There is no target value; the result will be evaluated after the closure of the testing timeframe.
Actual output	
General consideration	

Appendix D

This appendix summarizes from D6.3 the indicators considered as necessary for the technological evaluation and the specific OPEN requirements to be verified.

Indicators:

- Availability
- Reliability
- Performance
- Accessibility
- Scalability
- Security
- Adherence to the standards

Note that possible exceptions are not considered here, because additional indicators could be specified in the specific test plan, depending on the precise prototype to evaluate. How can the indicators be adapted to a prototype? The translation for testing purposes (cfr. D6.3 section2) shall be applied to its specific context.

I. Availability

The percentage of time during which the prototype is correctly working should be measured and monitored for a predefined lasting of time; of course this means that the **expected result** from the execution of the prototype should be agreed from the test team with the developers.

The most basic solution to define how to measure the availability is to continually execute the prototype for a predefined time; of course this is a baseline, which can be improved in two ways:

- With a proper tool, either internal to the prototype itself or from other free software; this would be probably the best way to perform an availability measurement
- Starting the prototype and leave it executing (both managing context info and performing migrations) without a continuous interaction (if possible)

The feasibility of these two options should be checked between the developers and the testers: lot of tools are available (also used in other European projects), and the opportunity to use some of them can be evaluated. About the second option, the feasibility especially depends on the features of the prototype itself and of the application running on it.

About the comparison with the environment usual parameters (e.g. 3G network), it would give a real added value only in case of a very low difference between the network/environment availability and the one desired by the prototype; it is possible to omit this point when the desired availability is much lower than it; of course this is another topic to be first discussed for each prototype.

II. Reliability

Reliability concerns the persistence of the availability, for a predefined lasting time, related to the execution of the necessary functionalities of a product; for example, if the prototype was a racing application, this timeframe could be dimensioned on a complete race, otherwise it can be related to a complete migration process; this can be decided for each prototype.

A preliminary discussion about the expected results is needed, in the same way as the availability; about the testing procedures, the basic option is also in this case to launch the execution of the prototype as many times as possible during the testing timeframe, with the alternatives of using: 1) Some tools 2) Automatic executions (not simulations that are more proper in a development phase).

III. Performance

Evaluation of performance will be based on the measurement of some Key Performance Indicators; D6.3 listed 1-2 KPI for each functional element of the OPEN platform, while the other element to verify (if feasible) is possible performance degradation after the migration (additional KPI could be inserted in the single prototype test plan):

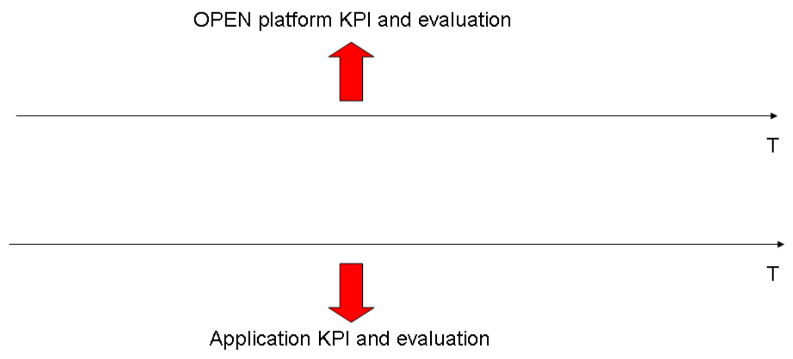

OPEN platform KPI and evaluation
T
T
Application KPI and evaluation

Figure 31: Two paths of performance evaluation: migration and application

About the first class of records, for each prototype the functional elements involved (within the complete set) will take part to the performance evaluation:

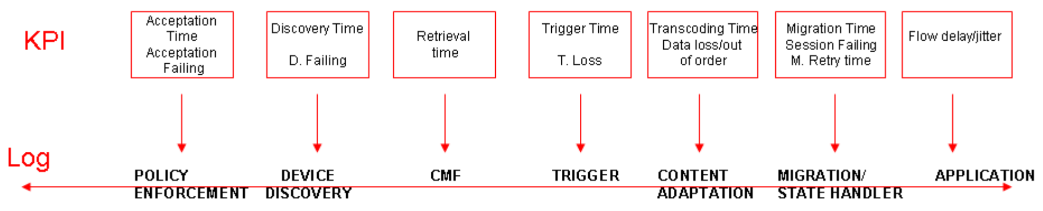


Figure 32: KPI to collect during the performance test, depending on the functional element

The collection of KPI would be more reliable if an internal log/tracer would be provided in the prototype itself, in order to avoid that external measurements can impact the precision or misunderstand the events considered as a reference. So prototypes with such a facility will be evaluated (from a performance point of view) by using values internally measured.

After the measurement, the results should be analyzed to summarize an overall evaluation of the parameters.

IV. Accessibility

The D6.3 listed some tools to apply to web pages and items to verify the eventual lacks of accessibility: for this indicator, developers and testers will agree the content on which to execute the analysis and the most suitable tools.

V. Scalability

The necessity to measure the increment of traffic, CPU usage and so on when new users join the system leads to the same conclusions than performance: it is worth to use internal measurements from the prototype if possible, and the results should be evaluated with the developers. Further kinds of measure could be added in the test plans depending on the specific prototype.

VI. Security

Security can be very basically tested with checks about the AAA for users allowed/not allowed, about the use of secure protocols (IPSec/IKE). More specific tests can involve the use of tools related to this topic, while further ones can be added by the developers for each prototype.

VII. Adherence to the standards

For this indicator the first evaluation is very simple: the prototype should not impact the environment, breaking the reference specifications. Other kinds of tests can be added for each prototype after an agreement between developers and testers, depending on the standards involved both in the product and in the environment.

VIII. Specific requirements

As said before, D6.3 identified a set of specific requirements for the OPEN project, depending on:

- Their critical importance and necessity for a correct functioning of a migration ”ecosystem”, made of the interaction of its components with device, applications and so on
- The feasibility of an easy way of testing these requirements, from the observation of the applications execution and of the migration
- Their relevance for testing purposes
- Their contribute to a general platform evaluation, in order to avoid requirements too context –specific

The requirements are now listed, classifying them basing on the functional element involved for their satisfaction; the classification has been revised in order to map them to the modules defined in the D4.2 (some requirements are mapped to more than one OPEN module):

Application

Reference from D1.1	Requirement	Typology
7	The user must be enabled to watch a program using her/his set top box and multiple screens	Migration Service Platform
78	Gaming anywhere, anytime, anyhow	User Interface /MSP
163	The OPEN platform should present the last data inserted by the user on the source device in the first presentation provided to the user in the target device	UI/MSP

117	OPEN enables the viewing and browsing of information for different users with different devices at the same time	UI
144	The OPEN platform should be able to handle, e.g. co-ordinate and synchronize, inputs from multiple-users, not only in gaming scenarios, but for others application too	MSP
152	When several users share the same screen in a multiplayer game, there must be a perfect synchronism in the input elaboration	UI/MSP
74	Users must be able to migrate identified parts of the application to other devices e.g. high score list	MSP/UI
Additional	Periodic actions of the applications maintain their phasing	MSP/Network
Additional	User status for Presence service maintained after migration	MSP/Network

Migration orchestration

Reference from D1.1	Requirement	Typology
6	System should be able to trigger a migration	MSP
86	Migration should be triggered by the user	MSP/UI
62	Users want to use the migration process for triggering application actions, e.g. for joining a game	MSP/UI
157	The OPEN platform should be installed and listening for any device requesting migration	MSP
54	It must be possible to continue my current service seamlessly across multiple devices	MSP
74	Users must be able to migrate identified parts of the application to other devices e.g. high score list	MSP/UI
Additional	User status for Presence service maintained after migration	MSP/Network

CMF

Reference from D1.1	Requirement	Typology
34	Service content should be provided in a context aware manner	UI/Application Logic
106	OPEN should let me know where my data is. After it has migrated several times	MSP/UI/AL

Web UI adaptation

Reference from D1.1	Requirement	Typology
115	OPEN enables the user to get, what s/he individually can handle, i.e. the information remains not only complete, but in terms of perceived complexity understandable after a migration	UI
156	The input devices must be able support the same actions	UI
75	Users must be able to push and pull user interfaces	MSP
117	OPEN enables the viewing and browsing of information for different users with different devices at the same time	UI
Additional	Image size must fit the screen of every kind of device allowed	MSP/Network
Additional	Page has to be entirely loaded for a good user experience	MSP/Network

Generic UI Adaptation

Reference from D1.1	Requirement	Typology
115	OPEN enables the user to get, what s/he individually can handle, i.e. the information remains not only complete, but in terms of	UI

	perceived complexity understandable after a migration	
156	The input devices must be able support the same actions	UI
75	Users must be able to push and pull user interfaces	MSP

Trigger management

Reference from D1.1	Requirement	Typology
131	The offline-online migration must be triggered by network QoS parameters too	Network
82	Migration should be automatic / system triggered. Based on previous settings by the user	MSP/UI
6	System should be able to trigger a migration	MSP
Additional	The offline-online migration must be triggered by battery too	Network

Application logic reconfiguration

Reference from D1.1	Requirement	Typology
87	I should be able to migrate more than the user interface, i.e. codec, computation tasks…	MSP/AL
81	Binary implementations of the services must be downloadable into the target device – A downlink is required	Network

Open Client Daemon with UI

Reference from D1.1	Requirement	Typology
86	Migration should be triggered by the user	MSP/UI
80	Users must be able to accept or deny a migration	MSP/AL

	from a to b	
90	The user must be able to select which content he wants to migrate to the low-end device	MSP

Policy enforcement

Reference from D1.1	Requirement	Typology
79	The user must be able to instruct the system, not to be interrupted, e.g. by somebody waiting to join. The user wants to control who can join the game, e.g. at play time by a list	MSP/AL
66	The user must be able to specify migration policies, e.g. automatic migration when switched off	MSP/AL

Generic/Web State handler

Reference from D1.1	Requirement	Typology
43	Recording of sessions	MSP/Network
123	OPEN enables the users to have a complete ex-post emergency analysis	UI
162	The OPEN platform should be able to maintain the data inserted by the user in the source device and show them in a consistent way after migration on the target device	UI/MSP

Mobility support

Reference from D1.1	Requirement	Typology
61	The user does not want to care about networking aspects when trying to migrate	Network
91	OPEN should predict the data and applications	Network/MSP

	needed when going mobile. Possible migration also for non-OPEN service providers	

Device discovery

Reference from D1.1	Requirement	Typology
20	Users need to discover devices in the vicinity.	Network
33	Devices in geographical range (but not network range) should be usable to migrate to	Network

Platform (requirements that involve the whole platform)

Reference from D1.1	Requirement	Typology
63	OPEN should work with and without internet connection	Network
38	My private information should be kept safe	MSP/UI
Additional	Use of secure protocol (e.g. IPSEC/IKE)	Network

References

[1]	Martin, J., Principles of Data Communication. Englewood Cliffs, NJ: Prentice Hall, 1988
[Aalst04]	W. Aalst, K. Hee. Workflow Management, Models, Methods, and Systems. First MIT Press paperback edition, 2004.
[BBDef]	Black box test definition: http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf http://en.wikipedia.org/wiki/Black_box
[D1.1]	Requirements for OPEN Service Platform
[D1.3]	Final Requirements for OPEN Service Platform
[D2.1]	Early infrastructure for migratory interfaces
[D3.2]	System support for application migration
[D4.1]	Solution for Application Logic reconfiguration
[D4.2]	Migration service platform design
[D4.3]	Prototype for Application Logic Reconfiguration
[D5.1]	Initial application requirements and design
[D5.2]	Initial prototype applications
[D6.1]	Usability criteria for project phases: use cases selection, design, development, test and deployment
[D6.2]	Evaluation parameters for enabling the environment programmability
[D6.3]	Indicators for technical evaluation
[DoW]	Description of Work – open – VERY LAST
[HUT]	Wiley Publishing, Inc. 2008: Jeffrey Rubin and Dana Chisnell:

	Handbook of Usability Testing.
[IDef]	Integration test definition: http://www.bsi-global.com/ http://www.testingstandards.co.uk/bs_7925-2.htm http://en.wikipedia.org/wiki/Integration_testing http://en.wikipedia.org/wiki/Unit_testing http://hissa.nist.gov/HHRFdata/Artifacts/ITLdoc/235/chapter7.htm
[WBDef]	White box test definition: http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf http://en.wikipedia.org/wiki/White_box_(software_engineering)
[WP]	Workflow Patterns initiative. http://www.workflowpatterns.com/patterns/index.php