OPEN Project

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

Title of Document:	**Evaluation parameters for enabling the environment programmability**
Author(s):	Grasselli Agnese, Marzorati Stefano, Piunti Mattia
Contributor(s):	Santoro Carmen
Affiliation(s):	Vodafone IT, CNR
Date of Document:	10/12/2008
OPEN Document:	D6.2
Distribution:	Public
Keyword List:	Application Logic Reconfiguration, User Interface Adaptation
Version:	0.5

OPEN Partners:

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

Evaluation parameters for enabling the environment programmability	Id Number: D6.2

Abstract

The purpose of this deliverable is describing the approach that will be implemented for testing the environment programmability and presenting the evaluation parameters that will be used.

Title: 0	Id Number: 0

Table of Contents

1 Introduction

The aim of the document is describing the approach that will be implemented for the programmability evaluation of the OPEN environment.

For the scope of this document, the programmability is the capability of defining different rules that describe the migration and adaptation processes depending on the context information. Since the OPEN platform architecture is still an High Level design, the following analysis should be seen as a guidelines collection, to be further evaluated when the architecture will be refined.

The document is structured as follows: in the second chapter a brief definition of programmability applied to the OPEN migration and adaptation processes is introduced. In the chapters number 3, 4 and 5, the migration and adaptation process will be analysed, whereas the last chapters summarizes the evaluation approach.

2 The Open environment

The migration process is composed of a set of phases, which trigger and manage the application migration. A specific part of this process is the application adaptation.. Even if the adaptation is part of the migration process, in this document it is deepened in a dedicated chapter in order to better focus on its specific tasks: the application logic reconfiguration and the content adaptation. The adaptation is conditional on the models used for the application logic description and logical user interface description, to be addressed in WP2 and WP4.

In order to better understand the scope of this document, it is useful to identify who the final users of the programmability are.

In the OPEN project, we can identify at least three categories of users:

- Middleware and application developers: for this kind of users, programmability means the capability of adding new code or modifying the existing-one. This capability is strictly related to the particular technologies used for the implementation.
- Service provider: the entity hosting the middleware and the application, which deliver the service to the end users. For this kind of users, the programmability is the capability of setting rules that defines the migration and adaptation processes performed by the OPEN platform depending on the context information.
- End users: for this kind of users, programmability could be the capability of influencing the migration and adaptation process setting personal preferences. These personal preferences are considered as context information managed by the platform, in the document are not distinguished by the other context information.

In this document the programmability is the capability of setting rules that defines the migration and adaptation processes performed by the OPEN platform depending on the context information. This definition fits both to the Service Provider user and the Application developers' point of view.

2.1 Migration process

The Migration process is composed of a number of stages, which can be implemented in different ways; these stages are deeply described in D1.2. The migration process is performed by the **Middleware layer**. In the migration process have been identified the following stages:

- **Device Discovery**: it identifies the devices that are available to be involved in the migration process and their attributes that can be relevant for migration.
- **When to Migrate**: the Migration Trigger indicates when to migrate. This event can be generated by the user or the system or through a mixed initiative process (the system proposes migration and the user can decide whether to accept it). Users can request migration when they feel

it necessary, while the system can trigger it when specific events are detected (such as the device is getting out of power).

- **Where to Migrate**: once migration is triggered it is important to identify the target device for the migration process. Such target should be one of the devices available for this purpose and should be detected on the basis of its features and how well it fits in the new context of use.
- **What to Migrate**: an interactive migratory service is composed at least of two main parts: the user interface and the application logic. The former is the software dedicated to the interaction with the user while the latter is the functional core independent of how user interaction takes place.
- **How to Migrate**: since the device to access the application changes after migration some level of adaptation of the migratory service should be performed, in particular of its interactive part, in order to better exploit the new resources available while preserving usability.
- **State persistence**: one of the main reasons for migration is to continue their session through different devices. This means that the changes made by the user in the source device should not be lost when moving to the new one. Thus, it is important to carry out source state extraction and associate it to the target version.
- **Activation in the target device**. In order to obtain continuity it is important that the application on the target device is activated not at its usual starting point but at the point in which it was left off on the source device.
- Optional termination of the source version

Each migration process stage:

- Uses the information provided by the context management system
- Is supported by the communication infrastructure

2.2 Adaptation process

The different parts of a migratory service are adapted at the **Presentation and Application Logic layer.** An interactive migratory service is composed of the following main parts:

- The **User Interface,** which is composed of the presentation (the choice of the modality, layout, graphical attributes …), the dynamic behaviour (the choice of the navigation model, the dynamic activation and deactivation of interaction techniques), and content (what information is actually presented). Each of them can adapt according to a change of context.
- The **Application Logic** can be adapted by reconfiguring the access to the functionalities in order to access different implementations of some of them or increase/decrease such functionalities because of the change

of device or the change of the connectivity. For example an access to a data base to retrieve a large amount of data can be performed if the application is using a good connectivity, while the same access should be avoided if the connectivity is to poor to provide results in a reasonable amount of time. In this case it could be necessary to perform the data base access in a separate phase or in separate application hence affecting not only business logic of the migrating application but also the overall procedure the migrating application belongs**.**

- The **Network** support, since the connectivity can change then the network protocol and their quality of service may have to change.

2.3 Programmability of the Open environment

The definition of "Programmability" is: the capability within hardware and software to change; to accept a new set of variables and instructions that alter its behaviour [http://encyclopedia2.thefreedictionary.com/programmability]. Programmability generally refers to program logic (business rules), but it also refers to designing the user interface which includes the choices of menus, buttons and dialogs.

Applied to the OPEN platform, the term Programmability acquires a wider meaning. Programmability of the OPEN environment means:

- Programmability of the **migration process**: the capability to impose a new set of variables and instructions or rules that alter the migration behaviour. The programmability of the migration process must be evaluated taking into account all the different middleware components which perform the different migration stages. This part should be application independent.
- Programmability of the **adaptation process**: the capability to impose a new set of variables and instructions or rules that alter the application adaptation behaviour.

Since in the OPEN project the migration and adaptation processes are driven by context information, the variables that will be taken into account are the context information variables. For the scope of this document, the programmability is the capability of defining different rules that describe the migration and adaptation processes behaviour depending on the context information.

The modules implementing the migration and adaptation processes are those defined in the D1.2 and depicted in Figure 1. Since the OPEN platform architecture and the interaction protocols between different modules have not yet been designed in detail, the following analysis should be seen as a guidelines collection, to be further evaluated when the architecture will be refined.

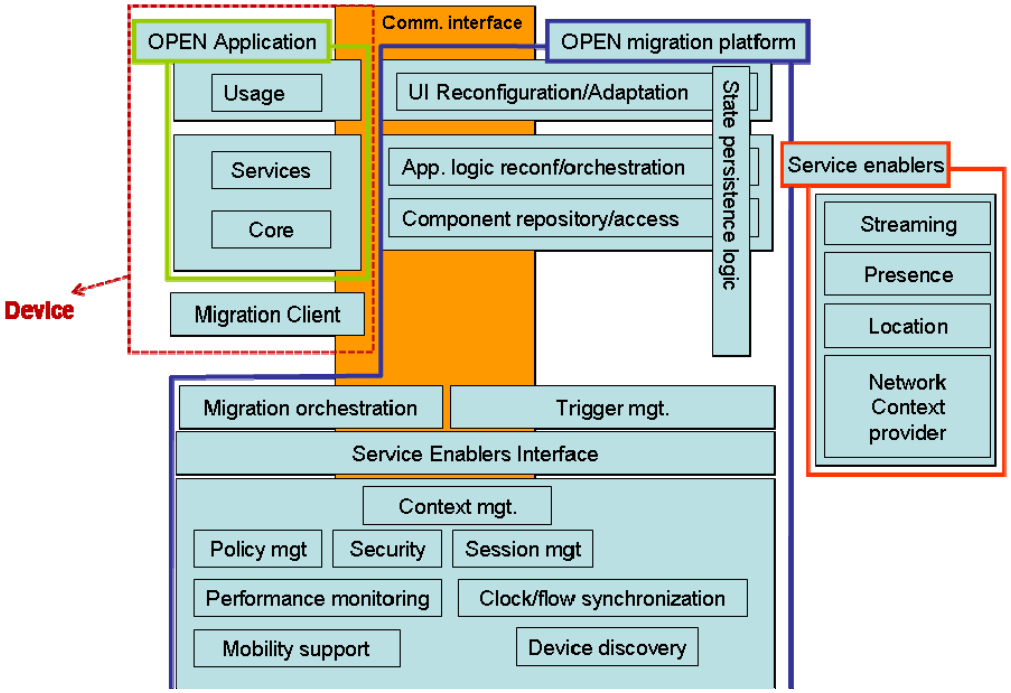

Figure 1: OPEN platform.

3 Programmability of the migration process

The programmability of the migration process is analysed taking into account the different stages previously listed and their mapping into the modules of the architecture described in D1.2.

3.1 Context management system

The different stages of the migration process interact with the context management system because the migration process is driven by the context information. The programmability of the context management system can be evaluated using the following parameters:

- **Capability of storing and managing new context variable**. E.g.: context information is provided by the mobile phone, which communicates to the context manager the battery threshold and the signal strength. This information is mapped in two variables in the context management system. Supposing that the mobile phone has also the location information based on GPS, the context management system 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.
- **Capability of setting different logic** for the interaction with other modules: the context management system should interact for example with the migration trigger. The logic implemented for managing the information exchange between context management system and other modules should be expandable/changeable.

3.2 Communication system

The communication system is the Communication interface depicted in Figure 1. It is an enabler for the communication between the OPEN modules; no programmability evaluations are required for it.

3.3 Device Discovery

This module is able to manage:

- Device discovery (device presence network)
- Service discovery (which services are provided by a specific device)
- Resource discovery (battery lifetime, processing power...)

The Device Discovery module is a provider of context information, because available devices and their characteristics are a particular kind of context information.

The programmability of the device discovery module can be evaluated using the following parameters:

- **Capability of storing and managing new variable**. E.g.: a PDA provides to the device discovery module the battery threshold and the screen dimension. In the device discovery module there are the relative two variables. Supposing that the PDA has also the location information based on GPS, the device discovery module 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.
- **Capability of setting different logic** for the interaction with the context management system. The logic implemented for managing the information exchange between the modules should be expandable/changeable.

3.4 Trigger management

The "When to migrate" stage of the migration process involves the Trigger management module. This module analyses contextual information changes and decides whether or not a migration should be activated through issuing triggers.

The programmability of the trigger management module can be evaluated using the following parameters:

- **Capability of storing and managing new variable**. As soon as the context management system is able to mange new context variables, the trigger management module should be able to use these new variables in order to implement new triggers.
- **Capability of setting different logic**. The logic implemented for issuing triggers should be expandable/changeable. Simple thresholds on context information values can be used or even function of different pieces of context information may apply.

3.5 Migration orchestration

The module indicated as "Migration orchestration" controls the migration process from received trigger to successful use of the migrated service. This modules implements different stage:

- Where to migrate
- What to migrate
- How to migrate

This module, even if associated to the migration process, could be affected by the approach used for the application modelling. The decision of "Where" to migrate and "What" should be taken accordingly with the different functions and requirements of the different application components which are going to migrate. "How" to migrate is a decision that put in to effects the results of the previous decision, depending on the particular scenario capabilities (involved devices, networks capabilities, and requirements to be satisfied…).

Again for this module, the programmability can be evaluated using the following parameters:

- **Capability of storing and managing new variable**.
- **Capability of setting different logic**. The logic implemented for selecting “Where”, “What”, “How” should be expandable/changeable.

3.6 Session Management

The session management function helps ensure that sessions can continue during migration, no programmability evaluations are required for it.

4 Programmability of the adaptation process

4.1 Introduction

The adaptation process should support the adaptation of both Application Logic and User Interface, and the capability of switching between different available networks.

4.2 Network

There are different use cases in which the network support can change:

- Migration between devices using different network supports (game migrating from the STB using xDSL access network to the mobile using 3G access network): in this case, the network switching is a consequence of device change and is part of the migration process. The supported network is a particular device information collected in the device discovery.
- Switching between different access networks but still using the same device: in this case, the handover between two different access network can be performed:
  - By the device: the mobile phone autonomously performs the handover between 2G (GSM) and 3G (UMTS/HSPA) coverage. No platform support is required.
  - By the platform: in an hybrid STB, which has both xDSL and DTT (Digital Terrestrial Television) or Satellite interface, live TV could migrate from IP to broadcast. It is the need for network handover which triggers the migration process, since different network elements can be involved in the service delivery changing the network support. In this case the trigger can use network related context information. This kind of context information can be provided to the Context management system by the Performance monitoring.

The modules involved in the use case described are:

- Device discovery
- Context management system
- Performance monitoring

The first two modules have been analysed in the previous chapter.

For the Performance monitoring module, the programmability can be evaluated using the following parameters:

- Capability of storing and managing new variable.
- Capability of setting different logic.

4.3 Application Logic and User Interface

The programmability of the Application Logic is the capability of imposing rules depending on context information that define the application logic reconfiguration behaviour.

The programmability of the User Interface is the capability of imposing rules depending on context information that define the user interface adaptation behaviour.

The programmability should be enabled by specific tools/languages that the Service Provider can use in order to impose desired behaviours. The analysis of these tools/languages is out of the scope of this document, but should be addressed in proper deliverables, as for example the D4.1 for the application logic reconfiguration.

For the programmability evaluation it is necessary to specify some parameters which can be used to verify that the selected tools are proper.

In the evaluation of the application logic reconfiguration programmability, the parameter that will be used is the consistency: if the application logic reconfiguration behaviour is compliant with the rules set using the tool/language selected, the reconfiguration is programmable.

In the evaluation of the user interface adaptation programmability, one of the parameters that will be used is again the *consistency*: if the user interface adaptation behaviour is compliant with the rules set using the tool/language selected, the adaptation is programmable. Another parameter that will be used for the UI programmability is the provided *control* of the UI: namely, to what extent the adaptation rules are able to cover all the different aspects of a user interface (e.g.: navigation, presentation, ..).

4.4 Application Logic

The Application Logic reconfiguration/adaptation module is in charge of adapting the application logic during the adaptation process. In this section, an introduction to the available approaches enabling the application logic reconfiguration is done. For different approaches, different tools/languages for defining the rules describing the module behaviour can be used. Two different approaches can be implemented:

- **Orchestration:** specification of an executable process that involves message exchanges with other systems, such that the message exchange sequences are controlled by the orchestration designer. The orchestration approach is mainly used for business processes: a business process can be modeled as a sequence of services, e.g. web services, with a specific language, as BPEL (Business Process Execution Language) [BPEL]. This model is used by an orchestration engine, as ActiveBPEL [Abpel], which create an instance of the process. The engine calls the different web services involved in the process, maintaining the control of the process during all the time in which it is running.

- **Configuration:** arrangement of functional units. A reconfiguration module run a specific reconfiguration algorithm and produces a particular arrangement of the available functional units. In this case the configuration module is not involved in the application execution.

Choosing between the two approaches should use implementation and architectural evaluation. In the current stage of the OPEN project, the approach has not already been defined, therefore in this chapter both the approaches will be considered.

4.5 Orchestration

In Figure 2 a general approach to orchestration is described. The main functional elements are:

- **Web services:** a software system designed to support machine-to-machine interaction over a network. Web services are frequently just Web APIs that can be accessed over a network and executed on a remote system hosting the requested services. In the picture below, some examples of Web services are: SMSC/MMSC for the SMS and MMS delivery, streaming server, chatting and betting server, Gaming application server.
- **Service orchestration:** a dedicated module performs the service orchestration.
- **Content adaptation:** the orchestrated service can be accessed through different devices. A Content adaptation module can be used in order to adapt the UI depending on the user's device(s).
- **Devices:** the user can access to the service using a browser.

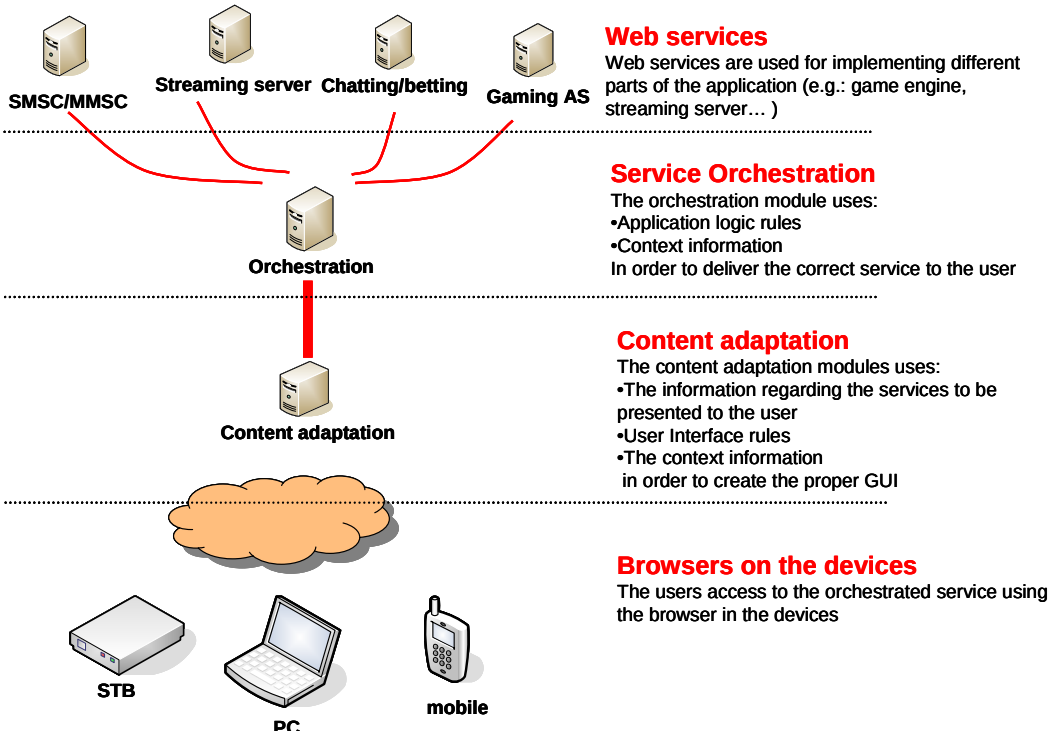

Figure 2: orchestration scenario.

4.5.1 Application Logic modelling

In order to select the proper tool for the application logic modeling, an analysis of the state of art for workflow languages and business process modeling language should be performed. The workflow language is a language that specifies the rules for connecting tasks to produce workflows. The Business Process Modeling Language (BPML) [BPEL] is language for the modeling of business processes.

Afterwards, the evaluation of the following approach could be done:

- **Top-down:** the aim is, given the existing workflow language and business process modelling language, to select one of these, which better fit to the OPEN project needs.
- **Bottom-up:** given the OPEN platform and applications, design a dedicated modelling language.

If the orchestration approach will be selected, the approach selection should be brought to completion during the WP4 work.

In order to compare different modelling language, we refer to Workflow Patterns [WP, Aalst04], which provide a thorough examination of the various perspectives (control flow, data, resource, and exception handling) that need to be supported by a workflow language or a business process modelling language. Workflow Patterns are widely used for examining the suitability of a particular process language or workflow system for a particular project, assessing relative strengths and weaknesses of various approaches to process specification, implementing certain business requirements in a particular process-aware information system,

and as a basis for language and tool development. Analysing the OPEN scenarios from this point of view, it is possible to extract the expressivity needed by the OPEN middleware.

In process-aware information systems various perspectives can be distinguished.

- The control-flow perspective captures aspects related to control-flow dependencies between various tasks (e.g. parallelism, choice, synchronization etc). Originally the Workflow Pattern initiative proposes twenty patterns for this perspective, but in the latest iteration this has grown to over forty patterns.
- The data perspective deals with the passing of information , scoping of variables, etc.
- The resource perspective deals with resource to task allocation, delegation, etc.
- The exception handling perspective deals with the various causes of exceptions and the various actions that needs to be taken as a result of exceptions occurring.

The workflow pattern can be used also for evaluating the programmability of the Application Logic reconfiguration/orchestration module: more workflow patterns a language is able to represent, higher is its capability of representing new applications. The selection of the proper modelling tool is key for the module programmability. In fact, this tool should be able to correctly model the application which will be implemented, but it should be in general a good modelling tool, in order to not restrict the application which could use the OPEN platform. This evaluation remains theoretical.

4.5.2 Execution

The orchestration approach required an orchestration engine, as ActiveBPEL [Abpel], which creates an instance of the modeled process. The engine uses the context information in order to perform decision and call the different web services involved in the process, maintaining the control of the process during all the time in which it is running. The programmability evaluation will not consider an engine evaluation, but the testing of the consistency of the OPEN application models running.

4.6 Configuration

The (re)configuration is the arrangement of functional units. A reconfiguration module runs a specific reconfiguration algorithm and produces a particular arrangement of the available functional units. In this case the configuration module is not involved in the application execution.

The dynamic-adaptive systems approach defined in [D1.2] is a reconfiguration approach. It considers dynamic-adaptive systems: systems built from a set of components that work together to perform some kind of tasks that are useful for application users. To established component-based applications, the behaviour of

dynamic-adaptive applications adapts during runtime to the needs of the current user and his environment.

Components are software entities that realize specific services that are described by and accessed through interfaces. Other components can define dependencies to provided services by defining them as required. Provided and required services have to be linked together in order to enable for example method calls from one component to another. The Figure 3 shows a set of component instances running on a PC that offer and require different services, described by interfaces. Required services are depicted as semi-circles, while provided services are depicted as circles. That notion follows the UML 2 standard [UML]. In order to run the application, according provided and required services have to be connected. The decision of which services to connect will be made in the middleware.

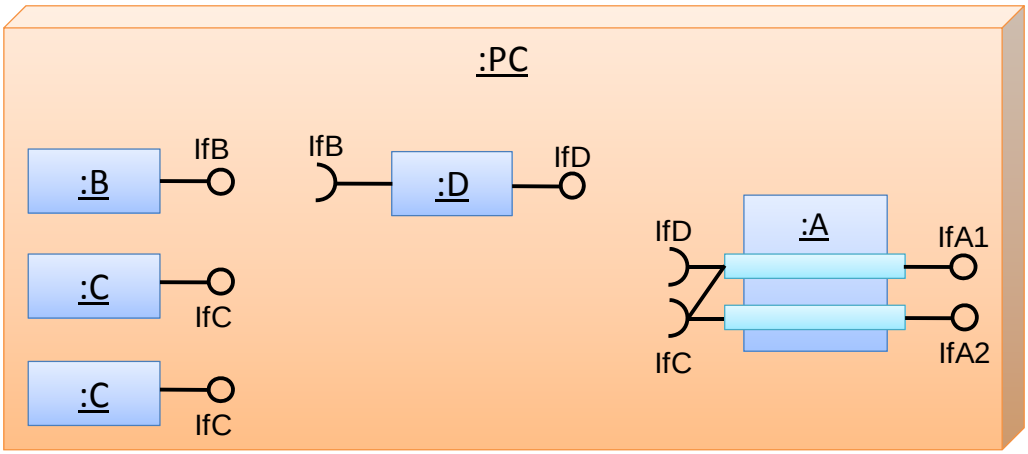

Figure 3: Component instances running on a PC ready for wiring and interacting together. If a component instance holds all required references to service implementations, it becomes runnable and will provide the given services to other components.

Therefore, the task of the middleware is to ensure that the wiring of component instances always fits to the user's needs on the one hand, and to the usage context on the other hand. The wiring can be changed during runtime based on context information, user preferences, or currently available component instances.

The rewiring aim is to use the information collected in the model to ensure that the wiring of component instances always fits to the application logic. The wiring can be changed during runtime based on context information, user preferences, or currently available component instances.

The programmability evaluation will test the consistency of the OPEN application running.

4.7 User Interface

As far as the user interface is concerned, the programmability of the migration platform mainly regards the capability to enable the user of the platform to customise/modify the aspects that have an impact on the user interface behaviour of the migrating application, according to various aspects (e.g. changed requirements/goals, different context, etc.). One of the modules that affects the

user interface part in the migration platform is the user interface **adaptation** module, which is in charge of adapting the user interface depending on changing contextual conditions. The programmability of this module implies that the user of the migration platform should be able to customise the platform behaviour by specifying different rules (or adding further rules) according to which the user interface adaptation will be carried out. Such rules might affect both the presentation part and the dialogue part of the user interface. Examples of such rules might be for instance the possibility for the user of the migration platform to decide the maximum number of presentations that should be generated by the migration platform as a consequence of a splitting, e.g. when passing from a desktop graphical device to a mobile graphical one, or the maximum number of elements to be included in a single adapted presentation. Another aspect can regard the possibility to use (or not) some specific types of media in the presentations generated after migration (for instance: using (or not) audio, using (or not) video ...), an additional one could cover the possibility of using or not some presentation techniques or the possibility to use some specific interaction techniques during adaptation.

The programmability of the user interface part of the Migration platform will be better supported if such rules are described using specific languages through which such rules will be modelled. Although for this adaptation module a decision about which tools/languages will be used, some preliminary ideas can be already mentioned to this regard. Indeed, since the user interface can be described at various levels using some XML-based languages, a possibility for describing the adaptation rules is using for instance XML-based techniques like XSLT [XSLT] transformations allowing specifying such adaptation rules.

Regarding some specific evaluation parameters of the programmability of the platform as far as the user interface part is concerned, we can mention the following ones:

- Assessing how effectively and consistently the migration platform will render the specified rules within the resulting adapted user interface (consistency = yes/no).
- Assessing how much control the service provider of the migration platform will have on the different aspects of the user interface (UI control = complete, partial…).

One additional aspect to verify is how easy it will be for the service provider of the migration platform to specify and edit adaptation rules. This topic will be addressed as a usability evaluation in the deliverable D6.4.

In the following table the tools and parameters related to the application logic reconfiguration and user interface adaptation modules are summarized.

Table 1: Tools and parameters for AL and UI.

		Tools for enabling the programmability	Parameters for testing the programmability
AL	*Orchestration*	workflow language or BPML, to be better addressed in D4.1	workflow patterns consistency=yes/no
	Reconfiguration	reconfiguration algorithm, to be better addressed in D4.1	consistency
UI		specific definition tool to be better addressed in WP2	consistency UI control=complete, partial…

5 Evaluation parameters and testing

In the Table below, the different evaluation approaches are summarised.

Table 2: Evaluation approaches.

Process	Migration	Application Logic Adaptation		User Interface Adaptation
		Orchestration	*Reconfiguration*	
Programmability	- capability of storing and managing new variables, e.g.: x=battery level - capability of setting different logics, e.g.: f(x)=battery level < 12%	capability of defining different rules that describe the application logic depending on context variable	capability of rewiring the components depending on context variable	capability of defining different rules that describe the user interface adaptation depending on context variable
Parameters	consistency=yes/no	- workflow patterns (only for theoretical evaluation) - consistency=yes/no	consistency=yes/no	UI control= complete, partial… consistency=yes/no
Test	Verify the system capability of managing the addition of a variable evaluating the application running consistency	evaluate the consistency of the application running with the defined rules	evaluate the consistency of the application running with the defined algorithm	evaluate the consistency of the user interface running with the defined rules

The parameter useful for programmability evaluation is the consistency. This assumption is explained by the following example.

Assuming that the orchestration approach is used, a gaming application is modelled using a defined language. Depending on the value of the variable “CPU power”, the orchestration module decides to use a specific web service, e.g.: if “CPU power”=high the orchestration module creates the service using the “chatting/betting” web service and the “game server” web service, which sends the players position to the user, if “CPU power”=low, the orchestration modules creates the service using the “chatting/betting” web service and the “streaming game application” web service, which streams to the user the game images.

In this example, programmability means that, if the application is described as function of the variable “CPU power”, when the context variable change the application behaviour will change accordingly. The application consistency with the application description is the parameter useful to test this feature.

6 References

[Aalst04]	W. Aalst, K. Hee. Workflow Management, Models, Methods, and Systems. First MIT Press paperback edition, 2004.
[Abpel]	http://www.activevos.com/community-open-source.php
[BPEL]	http://www.oasis-open.org/specs/
[D1.2]	Initial OPEN Service Platform architectural framework
[UML]	Object Management Group. UML 2.1 Superstructure and Infrastructure Specifications. November 2007.
[WCFP]	N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, BPMcenter.org, 2006.
[WP]	Workflow Patterns initiative. http://www.workflowpatterns.com/patterns/index.php
[XSLT]	http://www.w3.org/TR/xslt