

PII
Deliverable D5.4

Final validation and framework evaluation

Editor:	Lars Kollecker, DTAG
Deliverable nature	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	M30 (December 2010)
Actual delivery date:	M32 (March 31, 2011)
Suggested readers:	Developers, Managers of the development process & innovation clusters, Integrators, Testers, SAC
Version:	1.0
Total number of pages:	43
Keywords:	Validation, Architecture, Infrastructure, Framework, Interconnection Gateway, Testbed manager, TEAGLE, Test, Lessons Learned

Abstract

This deliverable is the final integrated prototype with accompanying documentation of the integration of testbeds based on the set of use cases from WP1. The final integrated prototype will be delivered in the form of validation experiments. Special attention will be drawn to the collection of feedback by all stakeholders towards the update and revision of the Panlab framework

Disclaimer

This document contains material, which is the copyright of certain PII consortium parties, and may not be reproduced or copied without permission.

In case of Public (PU):

All PII consortium parties have agreed to full publication of this document.

In case of Restricted to Programme (PP):

All PII consortium parties have agreed to make this document available on request to other framework programme participants.

In case of Restricted to Group (RE):

All PII consortium parties have agreed to full publication of this document. However this document is written for being used by <organisation / other project / company etc.> as <a contribution to standardisation / material for consideration in product development etc.>.

In case of Consortium confidential (CO):

The information contained in this document is the proprietary confidential information of the PII consortium and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the PII consortium as a whole, nor a certain party of the PII consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum

Pan European Laboratory Infrastructure Implementation

Panlab II

WP5 - Integration and validation

Final Validation and framework specification

Editor: Lars Kollecker, DTAG

Work-package leader: Christos Tranoris, UoP

Estimation of PM spent on the Deliverable: 46,25

Copyright notice

© 2011 Participants in project PII

Executive summary

The result of the integration work in Panlab II project is the final integrated prototype of the PII platform. Many features, tools, protocols and automatisms have been implemented allowing PII customers to create their own testing environment very easily and to provide a federation platform for platform providers which is managing the testbed resources in a very effective and efficient way.

In this deliverable, the final integrated prototype of the integration of testbeds is evaluated. In order to integrate, validate and evaluate the overall PII platform a number of Use Cases were developed continuously during the project live time. There are four groups of Use Cases; Academic, Industrial, Large-Scale (P2P Client Use Case) and PII Partner Use Cases as well as some new Use Cases that were not presented in D1.1r, i.e. the P2P Client Use Case. The Use Cases have been utilized to serve as project integration, i.e. the provision of detailed requirements, as well as validation and evaluation experiments. The aim is to thoroughly evaluate all aspects of the PII framework from a technical point of view and draw conclusions from the testing results, lessons learned and user feedback.

The number of use cases involved and the plethora of persons and expertise involved in the testing phase of each setup, necessitates the formulation (or a compilation) of a common and unified method in narrating each testing approach. This is accomplished based on highlighting basic steps on testing/evaluating procedures in a kind of storyboard, depicting platform assets available that are under test. The means of validation are based on previously mentioned storyboards with focus on customer interaction, testbed discovery and setup, use of virtual testbeds, monitoring and acquiring results and security aspects.

The validation of the Panlab II framework and its refinement is based on the use cases, above mentioned means of validation and early experimentation. This document is identifying whether the PII framework meets specifications and if it fulfils its intended purpose. The validation process is concentrating on the PII framework, Teagle services, testbed infrastructure and interconnection, the Resource Adapter concept, controlling the experiment, Panlab processes and legal framework, software delivery and support tools and a variety of supported use cases.

Finally the lessons learned section will provide valuable feedback by all stakeholders towards the update and revision of the Panlab framework.

List of authors

Company	Author
COSMO	Georgios Korinthios, Efi Nikolitsa
BCT	Kostantinos Koutsopoulos
DT	André Steinbach, Lars Kollecker
Eurescom	Anastasius Gavras
FhG	Sebastian Wahle, Konrad Campowsky
OCTO	Jussi Mäkinen, Marko Palola, Esa Piri
TSSG	Shane Fox
UoP	Christos Tranoris
WIT	Shane Fox
Synchromedia	Mathieu Lemay, Samir Hadjout, Mohamed Cheriet

Table of Contents

Executive summary	3
List of authors.....	4
Table of Contents	5
List of figures and/or list of tables.....	7
Abbreviations	8
Glossary.....	9
1 Introduction	10
1.1 Objective of this document.....	10
1.2 Purpose	10
2 Use Case Description	11
2.1 PII academic Use Case	11
2.1.1 Introduction	11
2.1.2 Technical Environment, testbed implementation and deployment.....	12
2.1.3 Running and operating the experiment.....	14
2.2 PII industrial Use Case	16
2.2.1 Technical environment	17
2.3 P2P Client Use Case.....	19
3 Testing Approach	20
3.1 Introduction	20
3.2 Storyboard of use cases	20
3.2.1 PII academic Use Case	21
3.2.2 PII Industrial Use Case.....	22
3.2.3 P2P Client Use Case.....	23
3.3 Means of validation	24
3.3.1 Customer Interaction	24
3.3.2 Testbed Discovery & Setup.....	25
3.3.3 Using the Virtual Testbed, Monitoring and Acquiring results	26
3.3.4 Security aspects	27
4 Validation of the PII Framework.....	28
4.1 The PII framework	28
4.2 The Teagle services	28
4.3 Testbed infrastructure and interconnection	29
4.4 The Resource Adapter concept.....	30
4.5 Controlling the experiment.....	30
4.6 Panlab Processes and legal framework.....	31
4.7 Software delivery and Support tools	32
4.8 Variety of supported use cases	32
5 Lessons Learned	33
5.1 BCT	33
5.1.1 Teagle GW	33
5.1.2 PTM and RAL.....	33
5.2 DT.....	33

5.3	FOKUS.....	34
5.3.1	VCTTool	34
5.3.2	Request Processor.....	34
5.3.3	Teagle Portal.....	34
5.4	OCTO.....	34
5.5	TSSG.....	35
5.6	UoP.....	35
5.6.1	RADL	35
5.6.2	FCI.....	35
5.7	Synchromedia.....	36
5.8	Other use case.....	36
5.8.1	Real-Time Industrial Networking Infrastructure as testing environment.....	36
5.8.2	Testing solution for the real time evaluation of PLC (Power Line Communications) transmission.....	37
6	Conclusions	38
	References	40
Annex A	http://www.panlab.net/use-cases.html	41
A.1	Recent Panlab II Use Cases.....	41
A.1.1	Future use case - Using Panlab via Teagle by Starhome and FT-PSC project (By Starhome and FT-PSC project)	41
A.1.2	Testing trans-coding video through dynamic cloud allocation (By Synchromedia and Inocybe inc. and Communication Research Centre Canada)	41
A.1.3	Testing Multicast Streaming on Dynamic Networks (By Synchromedia, Inocybe inc. and Communication Research Centre Canada).....	41
A.1.4	Testing Uncompressed HD Streaming (By Synchromedia, Inocybe inc. and Communication Research Centre Canada).....	41
A.1.5	EzWeb application over TID SDPLabs: “PII Message Sender” (By Telefónica I+D) .	42
A.1.6	Testing end-to-end Self-Management in a Wireless Future Internet Environment (By Octopus network, University of Athens, and VTT Technical Centre of Finland)	42
A.1.7	Testing enhanced Web TV services over mobile phones (By Technicolor, Images & Reseaux, and COSMOTE)	42
A.1.8	Testing Adaptive admission control and resource allocation algorithms (By University of Patras)42	
A.1.9	Stress test the Open IMS core (By Fraunhofer FOKUS)	42
A.1.10	Testing a VOIP user agent (By University of Patras)	43

List of figures and/or list of tables

Figure 1: The setup for testing the algorithm	11
Figure 2: The Resource adapters of the available testbed resources	12
Figure 3: RADL definition for the RUBiS application resource.....	13
Figure 4: The RUBiS use case setup designed in the VCT tool.....	14
Figure 5: Designing the algorithm to operate resources during execution.....	15
Figure 6: Provided Resources.....	16
Figure 7: The Industrial use case setup designed in the VCT tool.....	18
Figure 8: An overview of the p2p experiment.....	19
Figure 9: Deploying the experiment.....	19

Abbreviations

DMZ	Demilitarized Zone
HSS	Home Subscriber Server (IMS component)
IGW	Interconnection Gateway
IMS	Internet Protocol Multimedia Subsystem
IP	Internet Protocol
IPTV	Internet Protocol Television
ISC	IMS Service Control
MRF	Multimedia Resource Function (IMS component)
NAT	Network Address Translation
NGN	Next Generation Network
NGOSS	New Generation Operations Systems and Software
NMTP	Negotiable Mail Transfer Protocol
PII	Pan-European laboratory Infrastructure Implementation II (2nd phase of
PTM	Panlab Testbed Manager
QoS	Quality of Service
RMON	Remote Monitoring
RA	Resource Adapter
S-CSCF	Serving Call Session Control Function
SID	Shared Information & Data (model)
SIP	Session Initiation Protocol
SLA	Services Layer Agreement
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SSA	Specific Support Action (EU FP6 project instrument)
SSH	Secure Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UC	Use Case
UI	User Interface
UML	Unified Modelling Language
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
VCT	Virtual Customer Testbed
XML	Extensible Markup Language

Glossary

User	<p>The term user is referring to a person that is actively using or testing services provided by a customer's resource inside the testbed federation.</p> <p>Differentiation between End-User and Test-User may only be useful when directly referring to the special task of active service testing or the usual task of common service usage.</p>
User Driven Innovation	<p>Inclusion of the user as part of the testing process in order to take into account early feedback. In this project two levels of User Driven Innovation (UDI) will be applied.</p> <ul style="list-style-type: none"> • The first level refers to the customer (the organisations and companies that use the federation to test their products and services). • The second level refers to the potential end-users of the services. <p>To differentiate between these two groups in this document, we will refer to the first as federation-customers, service-testers or service-providers and the second we will refer to as end-user.</p>
Provider	<p>This term refers to a party owning testbed infrastructure and has entered an agreement allowing a customer to use its testing infrastructure and resources to develop and test services.</p>
Testbed	<p>A testbed is an environment which allows experimentation and verification for research and development products. A testbed provides rigorous, transparent and replicable testing and herein it is always used in the context of new information and (tele-) communications technologies for networks and services.</p>
Testbed federation	<p>A testbed federation or federated testbeds is the interconnection of two or more independent testbeds for the creation of a richer environment for testing and experimentation, and for the increased multilateral benefit of the users of the individual independent testbeds.</p>
Customer	<p>A customer of Panlab is someone that used TEAGLE to set up testbed resources for the purpose of testing or developing services. The customer is able to directly connect to the rented resources (his VCT) using a VPN client (U3 interface).</p> <p>The customer can offer his services under the terms of Panlab user domain access (U2 interface) to external Test-/End- Users</p>
Testing session	<p>Herein the term testing session refers to a well-defined temporal and spatial relation of testing infrastructures and testing resources by the customer or user(s).</p>
TEAGLE	<p>TEAGLE is the central Pan-European Laboratory search and composition engine. It provides a web-based customer interface for browsing the Panlab federation offerings and the instantiation of a VCT.</p>
VCT	<p>A Virtual customer Testbed is the sum of all resources, including interconnections, rented by the customer. It basically is an isolated network in that the customer is able to "dial in" and directly access the items assembled in TEAGLE.</p> <p>Each customer operates inside its own VCT and has no access to other VCTs. The purpose of direct VCT access is to configure and develop but not to widely test new services; such testing is intended to be done from outside using external Test- and/or End-Users.</p>

1 Introduction

In this deliverable, the final integrated prototype of the integration of testbeds is evaluated. In order to evaluate the framework, the Use Cases that are defined in WP1 and a number of new Use Cases have been utilised to serve as project integration, validation and evaluation experiments. The aim is to thoroughly evaluate all aspects of the PII framework from a technical point of view and draw conclusions from the testing results, lessons learned and user's feedback.

1.1 Objective of this document

The objective of this document is to complete the D5.4 deliverable, which is the final deliverable associated with Task 5.3 "Phased Validation and Experimentation".

1.2 Purpose

Chapter 2 describes in detail the Use Cases that have been used in the validation of the PII framework. There are four groups of Use Cases; Academic, Industrial, Large-Scale (P2P Client Use Case) and PII Partner Use Cases. The list includes some new Use Cases that were not presented in D1.1r, i.e. the P2P Client Use Case. For the Use Cases that were described in D1.1r some more details about the Use Cases are provided.

Chapter 3 presents how the unified testing approach has been defined and its application to the individual Use Cases. It also defines a set of assessable events that can be used to evaluate the PII platform from a user and testbed provider point of view.

Chapter 4 describes the validation of the PII framework using the results from the Use Cases and describes refinements to the framework that came as a result of the testing.

Chapter 5 depicts the lessons learned whilst conducting the update and revision of the PII Framework based on feedback provided by the component developers and Use Case participants.

Chapter 6 concludes the document.

2 Use Case Description

This Chapter describes in detail the Use Cases that have been used in the validation of the PII framework. There are four groups of Use Cases; Academic, Industrial, Large-Scale (P2P Client Use Case) and PII Partner Use Cases. The list includes some new Use Cases that were not presented in D1.1r, i.e. the P2P Client Use Case. Additionally several new Use Cases have been developed, which are presented on the PII website. These new Use Cases are part of the Annex. For the Use Cases that were described in D1.1r some more details about the Use Cases are provided. The individual PII Partner Use Cases were not addressed again in this document, but can be found in D1.1r.

2.1 PII academic Use Case

2.1.1 Introduction

In order for one to test an adaptive admission control and resource allocation algorithm, it is necessary to set up an appropriate testbed of a distributed web application such as RUBiS benchmark, which is an auction prototype site modelled after eBay.com. It provides a virtualized distributed application that consists of three components, a web server, an application server, a database and a workload generator that produces the appropriate requests. Furthermore it can be deployed in a virtualized environment using Xen server technology, which allows regulating system resources such as CPU usage and memory, and also provides a monitoring tool, Ganglia. This can measure network metrics such as round trip time and other statistics, and also resource usage in virtual machines.

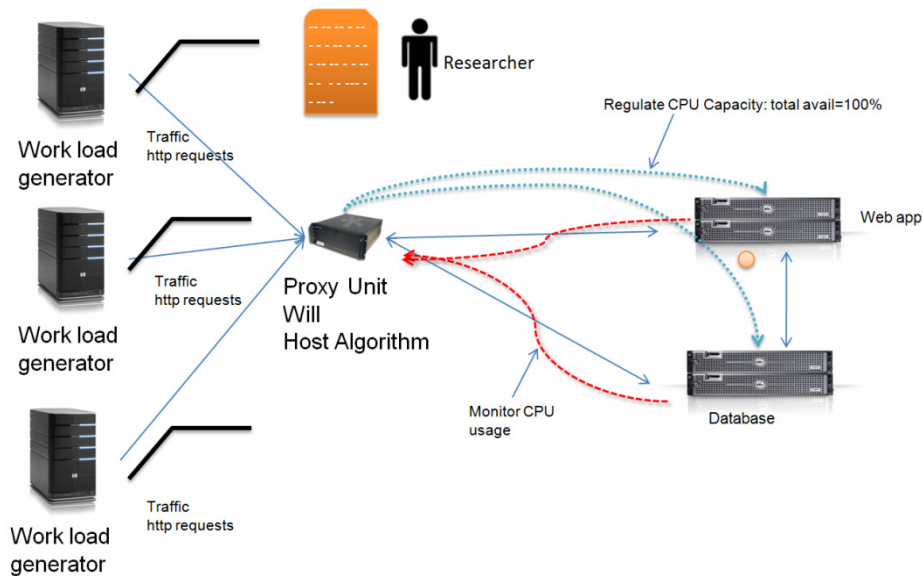


Figure 1: The setup for testing the algorithm

The adaptive admission control and resource allocation algorithm is applied to succeed the specific target of network metrics, like round trip time and throughput. This will be done by deploying a proxy-like control component for admission control and using Xen server technology to regulate CPU usage. During this scenario the adaptive admission control and resource allocation algorithm is tested against network metrics, like round trip time and throughput. RUBiS clients will produce requests to push RUBiS components to their limits in order to produce high values for resources like CPU usage and network throughput.

During the setup, the researcher wants to test http proxy software written in C programming language that implements an admission algorithm. Figure 1 displays the setup of the discussed scenario. The setup consists of 3 work load http traffic generators, making requests through a hosting unit. The algorithm, which is located at the proxy unit, needs to monitor the CPU usage of the Web application and Database machines. The algorithm should also be able to set new CPU capacity limits on both

resources. Additionally the algorithm should be able to start and stop the work load generators on demand.

2.1.2 Technical Environment, testbed implementation and deployment

From the requirements of the use case, it is evident that it would benefit from a testbed offering RUBiS resources. Moreover, the experiment needs to manage and monitor resources within the C algorithm. So the resources need to provide monitoring and provisioning mechanisms.

To support such an experiment and similar ones, a required infrastructure needed to be built. The equipment used is as follows:

- Linux machines for the RUBiS based work load generators
- A Linux machine for hosting the algorithm unit, capable of compiling C and Java software
- Linux machines for running the XEN server which runs the RUBiS Web app and database

The final user needs to provide the algorithm under test. They will simply login to the Proxy Unit, compile the software and execute it. The user will not have access to the RUBiS resources (i.e. cannot login) so there is a need to encapsulate the monitoring and provisioning capabilities. For this requirement and to make available the RUBiS resources for future testing within the Panlab federation, the so called Resource Adapters (RAs) were built.

For each resource there is a corresponding RA, which exposes configuration parameters to the end user. As displayed in Figure 2, all the components are based on Virtual Machines managed by a XEN server. The implemented RAs instantiate all these Virtual Machines and configure the internal components according to end-user needs. The work load generator exposes parameters such as: used IP for the testbed, memory, hard disk size, number of clients, ramp up time for the requests and a parameter used during the execution of the experiment called Action which accepts the values start and stop. The Proxy Unit exposes parameters such as used IP for the testbed, memory, hard disk size, username, password and IP to connect to the RUBiS application resource. The RUBiS application and the RUBiS database have similar parameters to the above and additionally a MON_CPU_UTILIZATION parameter which is used to monitor the resource and a CPU_CAPACITY used to set the max cpu capacity of the resource.

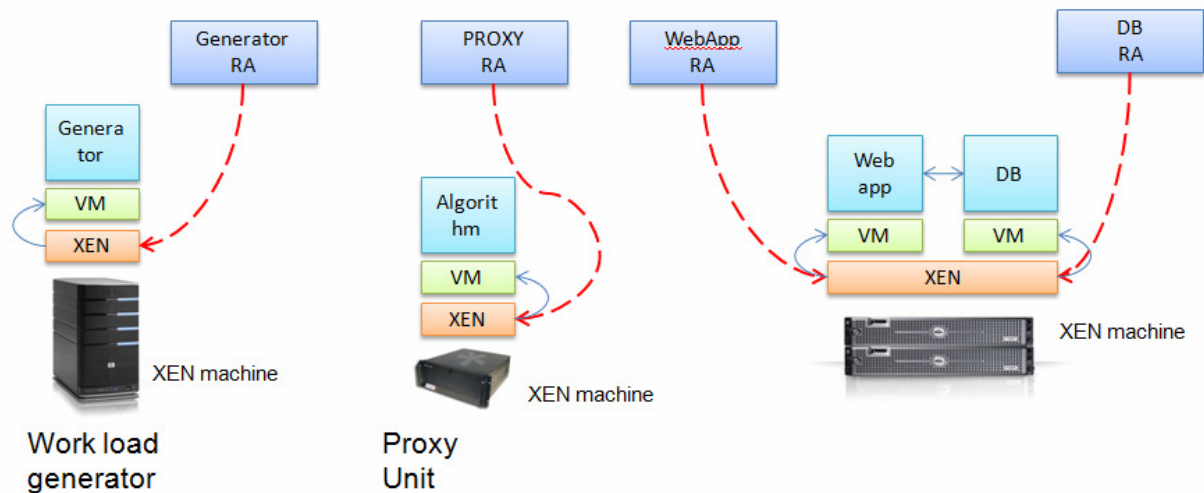


Figure 2: The Resource adapters of the available testbed resources

```
rubis_app.radl x rubis_cl.radl rubis_proxy.radl rubis_db.radl
Resource Adapter "rubis_app"
Configuration Parameters { // Visible Parameters to VCT user
    APP_VLANID;
    APP_IP;
    APP_GW;
    APP_MEM;
    APP_DISKSPACE;
    APP_DBIP;
    CPU_CAPACITY description = "Set CPU Capacity " ;
    MON_CPU_UTILIZATION description = "Readonly.Returns resource utilization";
}

Binding Parameters { // Local Parameters used for resource configuration
    admin = "root";//root
    admin_pwd = "v[REDACTED]";//[REDACTED]
    admin_ip = "150.140.[REDACTED]";//[REDACTED] (where scripts are)
    admin_port = "22";//22
}

On Update {
    ProcessOnAllConfigurationParametersComplete = NO;
    RProtocol SSH {
        Remote Machine = "admin_ip";
        RPort = "admin_port";
        RUsername = "admin";
        RPassword = "admin_pwd";
        RExecute{
            "/repo/scripts/shutdown_forget.sh appname;"
            "/repo/scripts/rubis/rubis_app.sh appname" << "APP_VLANID"
                "APP_IP" "APP_GW" "APP_MEM" "APP_DISKSPACE" "APP_DBIP">>
            "/repo/scripts/rubis/changecpu.sh dbname" << "CPU_CAPACITY" >>
        }
    }
}
```

Figure 3: RADL definition for the RUBiS application resource

The resource adapters were defined using the Panlab's Resource Adapter Description Language (RADL)[4]. RADL is a concrete textual syntax for describing a Resource Adapter based on an abstract syntax defined in a meta-model. RADL is an attempt to describe a RA in a way that decouples it from the underlying implementation code. RADL's textual syntax aims to be easier to describe a RA than code in Java or other target languages. RADL is useful in cases when there is a need to configure a resource that offers an API for configuration. The user can configure the resource through some Configuration Parameters. The RA "wraps" the parameters and together with the Binding Parameters, the RA can configure the resource. A Binding Parameter is a variable that is assigned locally by the resource provider, e.g. a local IP address. This approach was also adopted for developing the RUBiS RAs. Figure 3 displays the RADL definition for the RUBiS application server.

The Configuration Parameters section describes the exposed parameters to the end user. The Binding Parameters are used for internal purposes of the local testbed configuration. The On Update section describes what the rubis_app RA does when it receives a provisioning update command from the upper layers. The RA will use the SSH protocol to connect to the internal machine and execute scripts on it.

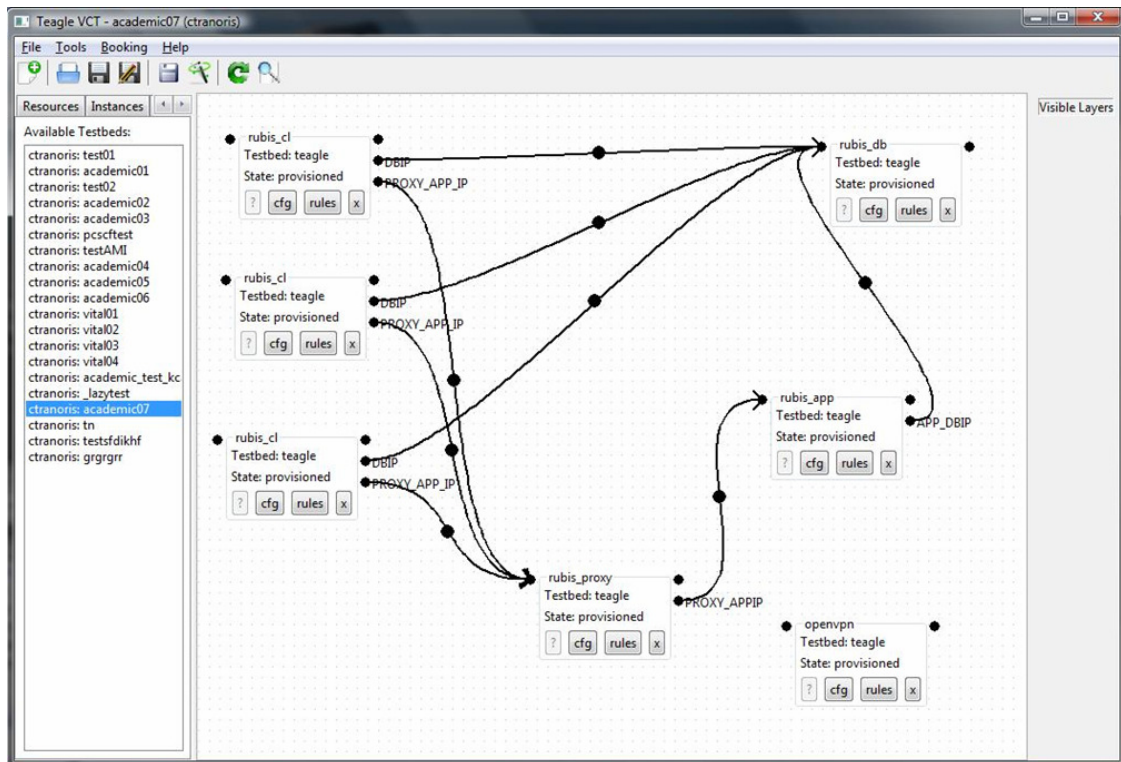


Figure 4: The RUBiS use case setup designed in the VCT tool

The tools to deploy, monitor and run the experiments are those offered by the Panlab architecture. The Resource Adapters are loaded in the testbed PTM and made available through Panlab's repository. Figure 4 displays the use case setup as can be done inside the VCT tool of Panlab. The resources are available in the left side of the tool. Three rubis clients can be seen to have been selected, along with one rubis proxy, one rubis application and one rubis database. Interconnections were made between these components in order to assign reference values to all resources. For example, the RUBiS clients need to know the IP of the proxy which hosts the algorithm. The proxy needs to know the IP of the RUBiS application which also needs a reference to the RUBiS database.

2.1.3 Running and operating the experiment

The scenario during the experiment utilizes the Federation Computing Interface (FCI) API that Panlab provides [5]. Federation Computing Interface (FCI) is an API for accessing resources of the federation. It is an SDK for developing applications that access VCT requested resources through the Panlab office services during the operation of testing. It is quite easy to embed it into your application/SUT in order to gain control of the requested resources during testing. The FCI is delivered to the customer after the generation of the SLA and not only does it contain the necessary libraries but also the alias of the resources that are used in the VCT scenario. This allows the User-Application/SUT to access the testbed resources during execution of the experiment in order to manage and configure various environment parameters or to get the status of the resources.

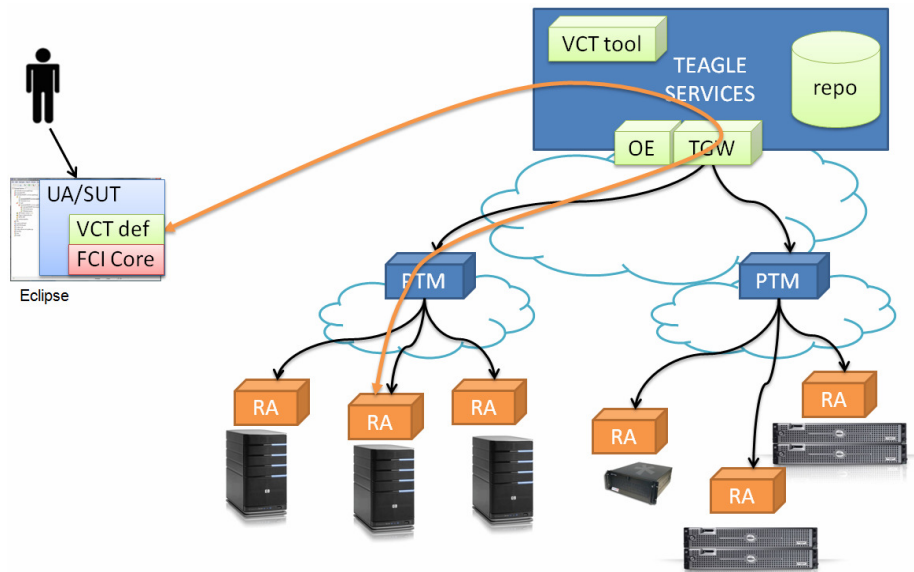


Figure 5: Designing the algorithm to operate resources during execution

In our testing scenario there is a need to configure resources or even get monitoring status data after the VCT is provisioned and while the testing is in progress. Figure 5 displays this condition where the System Under Test (SUT) is our algorithm. FCI automatically creates all the necessary code that the end user can then inject inside the algorithm's code. The end-user needs just to enter his credentials in order for FCI to generate the necessary wrapper classes and functions that are capable of accessing the reserved, provisioned resources. An example is given in the following code listing in Java:

```
//an example Java federation program
public class Main {
    public static void main(String[] args) {
        //An example for VCT: academic07
        academic07 myvct = new academic07();
        myvct.getuop_rubis_cl_91().setRAMP_UP_TIME("55000");
        myvct.getuop_rubis_cl_91().setACTION("start");
        myvct.getuop_rubis_cl_91().setNUM_CLIENTS("300");
        int moncpu = myvct.getuop_rubis_db_33().getMON_CPU_UTILIZATION();
    }
}
```

Assuming that we have given the name academic07 for our VCT definition, the java listing displays how we can access the resources of this VCT. FCI creates a java class, called academic07() that we can instantiate in order to get access to the resources. Additionally, for each resource that participates in the VCT, java classes are able to provide access. For example the command `myvct.getuop_rubis_cl_91().setACTION("start");` starts the RUBiS client of the rubis_cl_91 resource. The command `myvct.getuop_rubis_db_33().getMON_CPU_UTILIZATION();` is able to give back the CPU usage of the database resource.

2.2 PII industrial Use Case

Web TV is a common application over the open internet and enhancements of this service are contemplated thanks to additional components permitting to add Quality of Service.

Typically, real time video adapters and transcoders are under investigation to ensure more efficient delivery of the video streams over the internet. The adapters offer to resize the video content with respect to terminal characteristics. The transcoders permit to face different network bandwidth capabilities.

The two components proposed here are the starting and ending points of the web TV service.

The particular use case setup offers the possibility to perform tests upon:

- A video on demand service platform
- End users in a real «mobile operator» context.

Potential service customers include e.g. a firm developing an IMS compatible MRF (Media Resource Function) providing buffering and video transcoding capabilities or an operator that intends to differentiate from ordinary WebTV service and trial the aforementioned MRF.

In this context, the setup is providing:

- The service platform as needed by the customer (service is including to build and maintain a list of available web tv services)
- The end user network, terminals and the related application to run the service.

The Virtual Customer Testbed (as shown in Figure 6) includes the following resources:

- Web TV service data server (C1): a software component for the delivery of WEB TV channels service data. All request and delivery procedures follow most of the TISPAN IMS specification for IPTV services.
- IMS core (C2): a software component acting as an IMS router interfacing the end user through a proxy CSCF and interfacing the WebTV application servers via the standardized ISC interface.
- Web TV client (C3): a software component capable of connecting to the Web TV service and to activate an audio video rendering client such as Videolan. The client is compliant with Web TV server component and as such following most of the TISPAN IPTV specifications.
- The customer component (C4): not available.

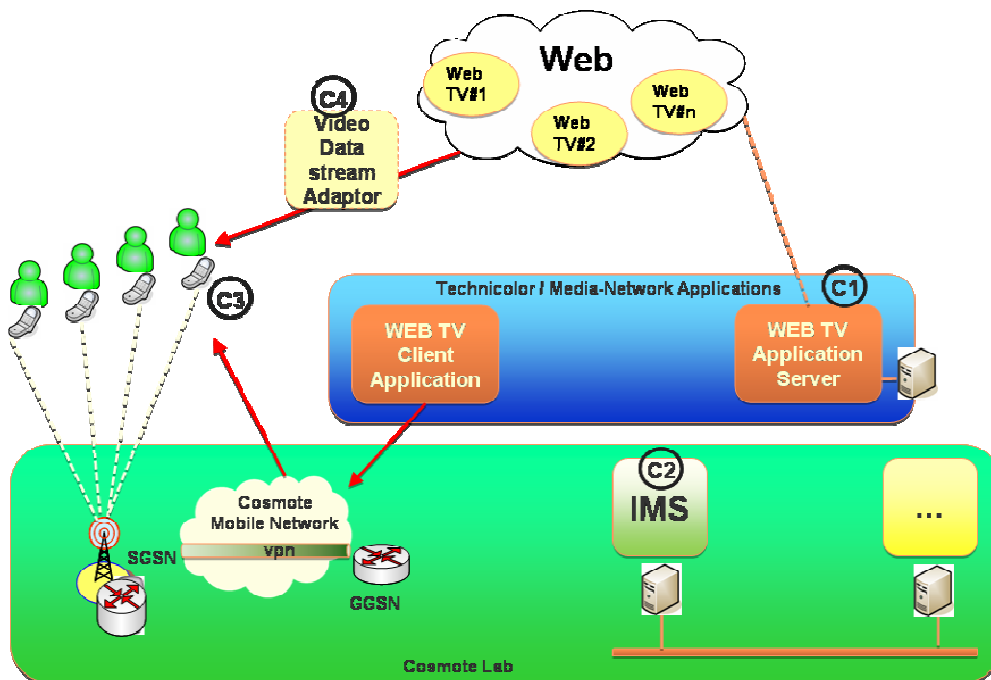


Figure 6: Provided Resources

The developed federated testbed environment offers the possibility to instantiate new IMS users and provides them (over-the-air) with the necessary client application and configuration settings. The mobile user (trial customer) can then (from the application GUI) register to the IMS platform, get the list of available Web TV channels with an HTTP request and SIP INVITE to watch a channel and assess Quality of Service that is understood from the mobile user perspective.

2.2.1 Technical environment

Putting the whole concept into Panlab's context; apart from establishing new or configuring existing resources the potential customer (i.e. firm developing MRF) should utilise the tools to deploy, monitor and run the experiments that are offered by the Panlab architecture. A Web Portal is available where customers and providers can access services, a visual Creation Environment called the "Virtual Customer Testbed" (VCT) tool allows the customer to define the requested services. Experimenters can browse through the resource registry content and can select, configure, deploy and access reserved resources. Another part of Teagle is the Teagle Gateway, the component that is responsible for transferring provisioning and configuration commands to selected resources lying in various administrative domains.

Some additional components for integrating testbeds that belong to various administrative domains, are: the Panlab Testbed Manager (PTM) and Resource Adapters (RA). The PTM is responsible for configuring the domain's resources. It implements the Resource Adaptation Layer where Panlab partners "plug-in" their Resource Adapters (RA). A Resource Adapter (a concept similar to device drivers) wraps a domain's resource API in order to create a homogeneous API defined by Panlab.

The VCT provided for the particular use case is comprised of the following resources (in the context of Panlab) that are available through the VCT tool:

- Host: this is the access to the machine running the Technicolor or Media Network Cluster PTM.
- Web TV server: performing the start and stop of the application server for web TV services. Potentially customer specific domains could be created.
- WebTV testing client: a first implementation of a client in a fixed environment. The client is a machine offering access to upload files and to execute SSH commands. The RA reads relevant parameters from the machine, and then creates a specific configuration file for this machine, sending a java file plus the configuration file. The machine is then ready to run the WebTV application.
- Web TV Mobile client: this collects information from the WebTV server and IMS provisioning resource and constructs the WebTV client configuration file. It then notifies the subscriber via SMS or email of the URL to download the software/configuration file from.
- IMS subscriber provisioning: adds an IMS subscriber to the open-ims.cosmo realm.

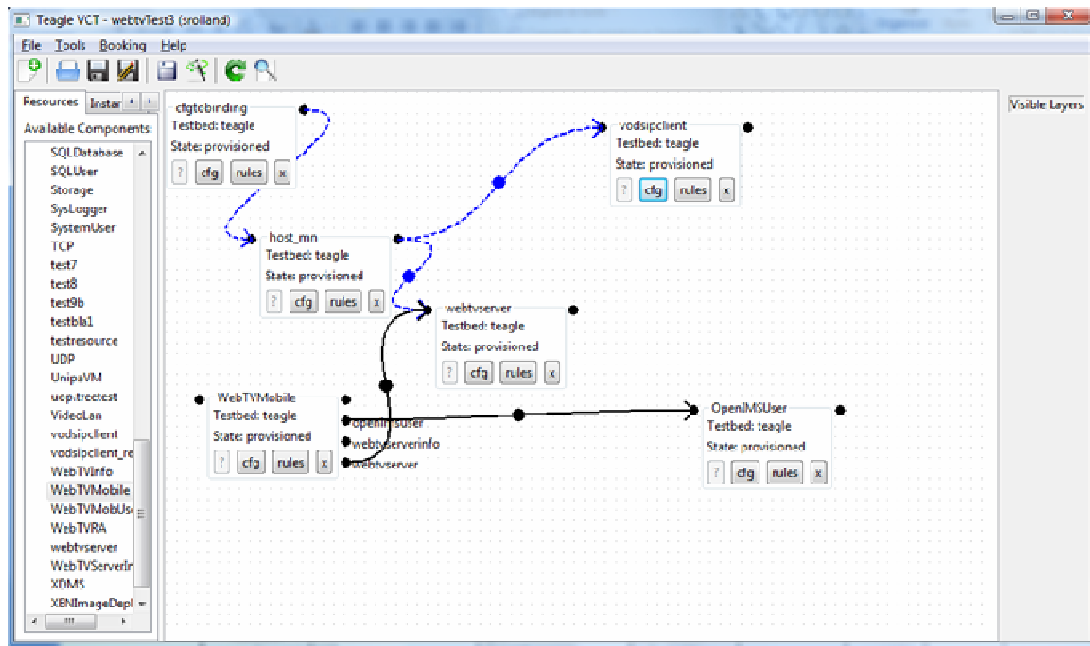


Figure 7: The Industrial use case setup designed in the VCT tool

The lack of a customer developing MRF is prohibiting, concluding to results regarding the usability of the service in a QoS trialling endeavour. Yet, the particular setup has shown encouraging efficiency in terms of federating services like WebTV and IMS from different testbeds and providing (over-the-air) all the necessary communication between end users, providers of the service and potential customer component's under test.

The resources for creating similar scenarios are going to be available under the Panlab Office offerings. Potential users after incorporating their system under test to the Panlab framework would simply have to utilize the VCT tool to build their own VCT utilising the Resources available.

2.3 P2P Client Use Case

This use case tests a p2p application over a large number of virtual machines located across the PII provider testbeds.

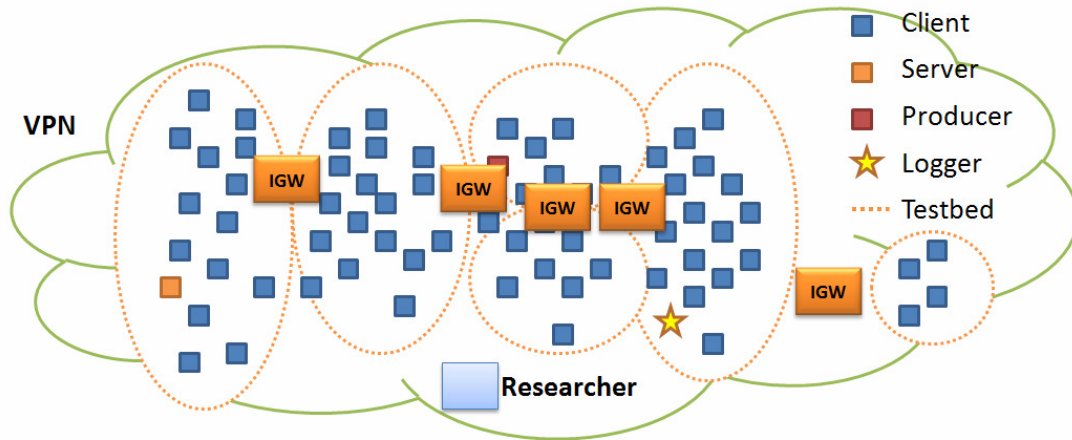


Figure 8: An overview of the p2p experiment

Figure 8 shows an overview of the experiment and the core components. The researcher provides two VM images: One with the p2p client and the server bundled and one with the logger. The researcher creates the scenario by requesting a series of VMs (e.g. 30) that will be instantiated with the client and selects also another machine which is going to play the role of the Server. Finally, a machine is selected that will host the logging services requested by the user.

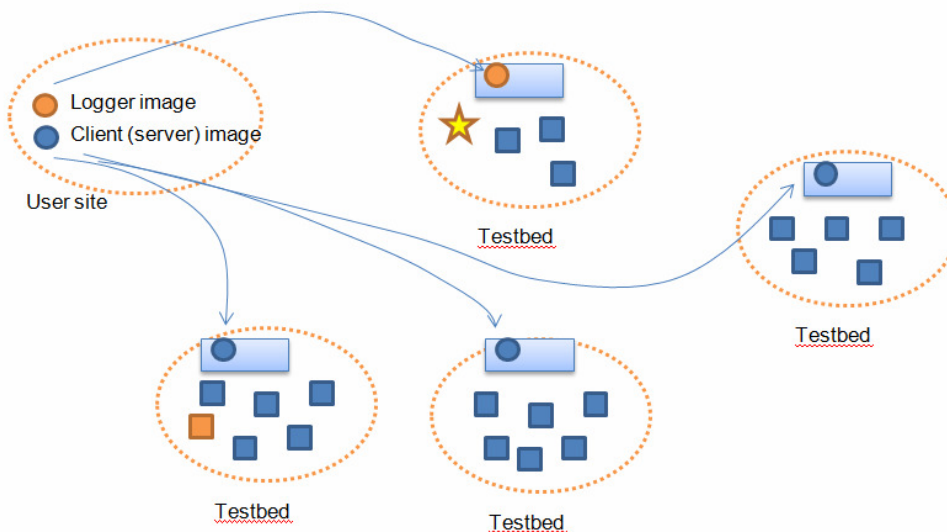


Figure 9: Deploying the experiment

Figure 9 depicts how the deploying of the experiment is done through Resource Adapters. In the experiment only two resource adapters are used, namely the XENImageStore and the XENVMDeploy. Both Resource Adapters are located at each testbed. Moreover, each testbed is capable of providing some XEN host physical machines. Each testbed configures its own RA with the necessary binding parameters.

The XENImageStore is used to speed up the process of copying the VMImage from one location to each testbed. So the XENImageStore copies locally the image from the user's site to its own testbed. From there the XENVMDeploy located in each testbed will use the VM copy to instantiate the VMs. Thus, we can deploy in parallel many machines across the federation of testbeds.

Finally, the IGWs of each testbed are configured to create a VPN between the instantiated VMs of each testbed and the user.

3 Testing Approach

3.1 Introduction

The number of use cases involved and the plethora of persons and expertise involved in the testing phase of each setup, necessitates the formulation (or a compilation) of a common and unified method in narrating each testing approach.

This is accomplished by highlighting the basic steps of the testing/evaluating procedures and depicting platform assets available that are under test.

3.2 Storyboard of use cases

This section draws a story board from each use case based on chapter 2 in order to highlight and represent a certain procedure or event of the use case.

In each use case, the final user should perform certain actions/steps necessary for building, operating and assessing the functionality of each virtual testbed. The representation of each story board is depicted in the table schema presented in Table 1. In the table headline, “**Event**” refers to the specific action, step or procedure that the user (or tester) intends to perform. The “**Real life Object**” column represents the actual object that the user (tester) acts upon. In the “**Object Representation in PII**” column, the translation of its real life object into PII semantics is given.

Table 1: Use cases story board representation schema

Event	“Real life” Object	Object Representation in PII
Provides the system under test	SUT	Resource(s). RAs built straight in Java (through available PII RA templates) or utilising RADL developed in PII.
Configure the necessary resource parameters in the VCT tool (provisioning), according to the intended test (needed number/type of resources, links, parameters and references between them)	VCT	Set of Resources (and respective RAs)
Book the designed VCT and run the experiment	Actual Booking & Operation of the resources	Successful booking summary & ad-hoc operation of resources
Monitor Status	Operating status of the VCT resources	Ad-hoc collection of Resource(s) status or through FCI
Acquire Results	Test results	Ad-hoc collection from the Resources
(Optionally) repeat the procedure with different setup/parameters.	VCT	Set of Resources (and respective RAs)
(Optionally) repeat an old/saved in the repository test to reproduce the results	VCT	Set of Resources (and respective RAs)

3.2.1 PII academic Use Case

In this use case an adaptive admission control and resource allocation algorithm (which is the System Under Test) is tested against a testbed of a distributed web application like RUBiS benchmark. The user sets up a Virtual Customer Testbed (VCT) that consists of four components, a web server, an application server, a database and workload generators, which produce the appropriate requests and can be situated in different physical testbeds.

The PII academic use case can prove the scalability virtues of the PII framework by applying large scale experiments utilizing high numbers of workload generators from diverse test environments (testbed, server capabilities).

Moreover, one can exploit FCI to monitor the status of the resources or alter parameters on demand after the resources been initially provisioned, thus operating the test on demand.

Event	“Real life” Object	Object Representation in PII
Provides the system under test	Admission control and Resource allocation algorithm	RAs built utilising RADL developed in PII.
Configure the necessary resource parameters in the VCT tool (provisioning), according to the intended test (needed number/type of resources, links/parameters references between them)	VCT with Web Application, Proxy, Database, Workload generators RAs	Set of Resources
Book the designed VCT and run the experiment	Actual Booking & Operation of the resources	Successful booking summary & ad-hoc operation of resources
Monitor Status	Operating status of the VCT resources	Collection of Resource(s) status through FCI
Acquire Results	Test results	Ad-hoc collection from the Resources
(Optionally) repeat the procedure with different setup/parameters.	VCT	Set of Resources
(Optionally) repeat an old/saved in the repository test to reproduce the results	VCT	Set of Resources

3.2.2 PII Industrial Use Case

The PII Industrial Use Case develop the means to test video adapters and transcoders on their efficiently adapting and delivering video streams over the internet in a variety of user clients.

In this context, the setup is providing:

- The service platform as needed by the customer (the service includes building and maintaining a list of available web tv services), containing the Web TV service data server.
- The end user network, terminals and the related application to run the service, containing an IMS core and the Web TV clients.

Event	“Real life” Object	Object Representation in PII
Provides the system under test	Video adapters and transcoders & Web TV clients	RAs built straight in Java (through available PII RA templates) or utilising RADL developed in PII (only Web TV clients are available).
Configure the necessary resource parameters in the VCT tool (provisioning), according to the intended test (needed number/type of resources, links/parameters references between them)	VCT containing the Web TV service data server, IMS Core with IMS Subscriber sign in mechanism, Subscriber Notification mechanism (via SMS or email), Web TV clients	Set of Resources (and the respective RAs), i.e. webtvserver, mobileclient etc.
Book the designed VCT and run the experiment	Actual Booking & Operation of the resources	Successful booking summary & ad-hoc operation of resources – i.e. viewing different content on Web TV clients.
Monitor Status	Operating status of the VCT resources	Ad-hoc collection of Resource(s) status, i.e. accessing servers or clients’ logs or through FCI
Acquire Results	Test results	Ad-hoc collection from the Resources’ logs
(Optionally) repeat the procedure with different setup/parameters.	VCT	Set of Resources (and respective RAs)
(Optionally) repeat an old/saved in the repository test to reproduce the results	VCT	Set of Resources (and respective RAs)

3.2.3 P2P Client Use Case

This use case tests a p2p application over a large number of virtual machines located across the PII provider testbeds. The researcher provides two VM images: One with the p2p client and the server bundled and one with the logger. The researcher creates the scenario by requesting a series of VMs (ie 30) that will be instantiated with the client and selects another machine which is going to play the role of the Server. Finally, a machine is selected that will host the logging services requested by the user.

Event	“Real life” Object	Object Representation in PII
Provides the system under test	A P2P application	Resource(s). RAs built in RADL developed in PII.
Configure the necessary resource parameters in the VCT tool (provisioning), according to the intended test (needed number/type of resources, links/parameters references between them)	VCT	A Resource copying images from provided sites and a resource deploying an image to a XEN VM Host
Book the designed VCT and run the experiment	Actual Booking & Operation of the resources	Operating the resources through FCI
Monitor Status	Operating status of the VCT resources	Specialized log file server provided by user
Acquire Results	Test results	Results from the log file server

3.3 Means of validation

This section defines the evaluating items based on the use case storyboards' template from the previous section. The assessment is accomplished by qualitatively evaluating the PII platform's desirable overall merits, as expressed in the former PII requirements deliverables, a collection of functionality requirement tickets in Trac (web-based software project management tool) and ad-hoc communications between partners.

3.3.1 Customer Interaction

In this particular set of cases the means and methods of interacting with the PII platform both as test user and testbed provider are tested. Initial use/installation of software available and the complexity of developing the needed components, as well as the support available for these are assessed.

Event	Evaluating Items
Provides the system under test	<ol style="list-style-type: none"> 1. Usability of testbed/resources registration mechanisms 2. Ease in integrating a new resource into the framework, including installing necessary software and developing glue logic to integrate with PII. 3. Interconnection parameters/setup clarity (and fulfilment), including setup needed on the user side to accomplish integration. 4. Comparison of accessing/altering own/new resource parameters through PII (and necessary steps needed) vs straightforward access. 5. Technical support available (and means of communicating with) for each of the components.
Configure the necessary resource parameters in the VCT tool (provisioning), according to the intended test (needed number/type of resources, links/parameters references between them)	<ol style="list-style-type: none"> 1. Feedback on the use of the graphic tool 2. Feedback on the use of FCI 3. Feedback on the use of RADL
Book the designed VCT and run the experiment	<ol style="list-style-type: none"> 1. Booking functionality assessment 2. Running experiments needed in SUT, draw conclusions on the usability of the interface
Monitor Status	<ol style="list-style-type: none"> 1. Evaluate available means of monitoring in terms of user-friendliness and efficiency
Acquire Results	<ol style="list-style-type: none"> 1. Evaluate available collection mechanisms (usability, trustworthiness)
(Optionally) repeat the procedure with different setup/parameters.	<ol style="list-style-type: none"> 1. Ability to alter specific parameters (during operation or in idle time) 2. Ability to include new resources or alter federated ones and repeat the experiment
(Optionally) repeat an old/saved in the repository test to reproduce the results	<ol style="list-style-type: none"> 1. Reproducible results (trustworthiness)

3.3.2 Testbed Discovery & Setup

The following items evaluate proper insertion/deletion of resources and/or alterations in their parameters and successful update of the framework to include amendments, being discoverable by the built-in tools.

Event	Evaluating Items
Provides the system under test	<ol style="list-style-type: none"> 1. Register resource(s) and confirm through VCT designer and/or access repository to validate assertion in the federation 2. New Testbeds assertion does not lead to framework or existing resources/testbeds/PTMs redesign/altering parameters/interconnection details.
Configure the necessary resource parameters in the VCT tool (provisioning), according to the intended test (needed number/type of resources, links/parameters references between them)	<ol style="list-style-type: none"> 1. Configure/update resource parameters and verify parameters existence/alteration 2. Discover federated resources (from various testbeds), build VCT and configure accordingly, assess usability and parameters proper assertion 3. Evaluate available procedures for automating (making transparent to the user) interconnection of resources 4. Are there any “broken”/non existent resources/testbeds, yet discoverable? 5. Complexity hiding aspects: disperse testbeds can be integrated in a transparent (or near-transparent) way, no need to know the testbed-specific semantics
Book the designed VCT and run the experiment	<ol style="list-style-type: none"> 1. Booking speed 2. Evaluate booking summary and logs vs. parameters stored in resources and conceivability of narration 3. Reservation mechanism assessment
Monitor Status	-
Acquire Results	-
(Optionally) repeat the procedure with different setup/parameters.	<ol style="list-style-type: none"> 1. Alter specific parameters (during operation or in idle time) and evaluate alteration 2. Update VCT design and repeat experiment (expandability of testing and VCTs)
(Optionally) repeat an old/saved in the repository test to reproduce the results	<ol style="list-style-type: none"> 1. Save VCT, close and reopen graphic tool, repeat tests

3.3.3 Using the Virtual Testbed, Monitoring and Acquiring results

The following evaluating items qualitatively assess the platforms ability to assist running, monitoring and acquiring results from the SUT.

Event	Evaluating Items
Provides the system under test	-
Configure the necessary resource parameters in the VCT tool (provisioning), according to the intended test (needed number/type of resources, links/parameters references between them)	1. Configure/update resource parameters during runtime and verify parameters existence/alteration
Book the designed VCT and run the experiment	<ol style="list-style-type: none"> 1. Perform parallel sessions with use of the same resources (different instance of the same RA) 2. QoS preservation mechanisms 3. SLA structure available
Monitor Status	1. Mechanisms available for reporting of resource/testbed status
Acquire Results	1. Results' representation means available
(Optionally) repeat the procedure with different setup/parameters.	-
(Optionally) repeat an old/saved in the repository test to reproduce the results	-

3.3.4 Security aspects

A federated environment should offer individual platform secure separation (and transparency in interconnection details) as well as secure means of registering with it, running experiments, logging and analysing results.

Event	Evaluating Items
Provides the system under test	<ol style="list-style-type: none"><li data-bbox="730 443 1418 510">1. Secure access to testbed/resources registration mechanisms<li data-bbox="730 517 1418 584">2. Secure testbed interconnections (new and existing ones)
Configure the necessary resource parameters in the VCT tool (provisioning), according to the intended test (needed number/type of resources, links/parameters references between them)	<ol style="list-style-type: none"><li data-bbox="730 602 1418 669">1. Hide (if wanted) parameters being provisioned to resources<li data-bbox="730 676 1418 743">2. Hide networking complexity (and information) of individual testbeds
Book the designed VCT and run the experiment	<ol style="list-style-type: none"><li data-bbox="730 790 1418 857">1. Prohibit tampering in information flows and interfaces<li data-bbox="730 864 1418 931">2. Preserve user's information/traffic flows (if needed)
Monitor Status	<ol style="list-style-type: none"><li data-bbox="730 949 1418 983">1. Secure access to monitoring mechanisms
Acquire Results	<ol style="list-style-type: none"><li data-bbox="730 999 1418 1032">1. Secure preservation of logs and results
(Optionally) repeat the procedure with different setup/parameters.	-
(Optionally) repeat an old/saved in the repository test to reproduce the results	<ol style="list-style-type: none"><li data-bbox="730 1133 1418 1167">1. Secure storage of VCT information

4 Validation of the PII Framework

This section describes the validation of the framework and its refinement based on the use cases and early experimentation. It will identify if the PII framework meets specifications and that it fulfills its intended purpose.

4.1 The PII framework

The PII framework promised to offer a centralized resource brokering service, offering heterogeneous resources for supporting a rich set of use cases. In general the Resource Adapter concept managed to wrap a wide range of resources, like physical or virtual computing resources, IMS infrastructures, applications like webtv transcoders, clients and sms servers. The Teagle services realized the centralized resource brokering service exposing unified interfaces to experimenters.

Moreover, the Panlab Testbed Manager is able to host Resource Adapters wrapping a wide taxonomy of resources and handle any intra-testbed management issues. PII also promised to offer inter-testbed connectivity, which is realized through the IGW component.

4.2 The Teagle services

The main tasks of the core Teagle services comprise the following:

- To provide information about the various entities which play a role in the operation of the Teagle framework. This information is provided to both the outside and also towards other Teagle components. These entities include user accounts, participating organisations, domains, domain managers, available resource types, existing resources and their configurations, etc.
- To provide Panlab users with a way to create and subsequently book VCTs in a flexible yet convenient way.
- To provide Panlab partners with ways to describe the resources offered by their domains towards the Teagle framework and to provide further administrative information
- To interpret VCT definitions, breaking them up into individual provisioning requests and subsequently route them to the appropriate domain manager for execution.
- To provide usage information and guidance to both users and partners.

The informational tasks mentioned above are fulfilled by the Teagle Repository which is implemented as a web application and exposes its functionalities through a RESTful interface. The Teagle Repository is a critical component in the Teagle architecture, whose contents are queried and relied upon by virtually all other components. While its domain model has continuously evolved, the Repository itself as well as its flexible interface has proven invaluable for the operation of the Teagle platform.

Teagle allows its users to create and book VCTs through the use of the VCT Tool, which has been implemented as a Java-Webstart application specifically for this purpose. The implementation in Java means that the VCTTool can be used on a variety of different platforms while still providing a rich graphical user interface. On the other hand, the deployment as a Java-Webstart application means that the VCT Tool can be conveniently started directly from a web browser.

The VCT Tool has been continuously updated and developed further, mainly driven by user's feedback and the evolution of capabilities of other Teagle components. Even though great progress has been made both in terms of flexibility as well as usability, many more changes could be imagined. Eventually it might even make sense to provide separate creation environments for specialised tasks. Please see the relevant section in the next chapter for a discussion on this topic.

The task of facilitating the actual booking of VCTs – meaning the breakup of VCTs in individual provisioning requests and routing them to the domain managers - is performed by a combination of components. These are the Request Processor, the Orchestration Engine, and the Teagle Gateway. The Request Processor is the entry point for a user's booking request. Among its tasks is too choose a domain for the resources the user did not choose a domain themselves. Furthermore, it performs access

control checks and policy evaluation (by enlisting the help of the Teagle Policy Engine) on the request. When these checks return successful the request is relayed to the Orchestration Engine which performs the complex task of breaking up a VCT into individual requests. This is not trivial since a booking can mean more than just the creation of new instances or the reservation of existing instances. Much rather, it can also mean the reconfiguration or deletion of existing instances. After individual provision requests have been determined, these are communicated to the Teagle Gateway whose task is to route them to the appropriate domain manager for executing the request.

Finally, the Teagle Portal provides facilities for domain owners (Panlab partners) to manage information about their domains and the resources they provide. Partners can create resource types and specify the configuration parameters they expose, and provide information on their domain managers, their API entry points and the resource types they support.

Customers can browse the list of available resources to see which resource types are available at which domains and which existing resource instances are available at certain times.

Furthermore, the Portal provides some sections with general information, like FAQ and news section or documentation on the Platform.

4.3 Testbed infrastructure and interconnection

There are a number of problems to be solved for interconnecting the Panlab testbeds. One is to ensure that only specific resources are allowed to communicate to each other, another problem is to hide the network topology of a site by applying network addresses and protocol translations. Quote from PanLab1 D2.2 Section 4.1 : “In order to establish connectivity, a Virtual Private Network must be set up ... hiding the complexities and allowing Panlab partners to dynamically provision multiple overlay networks“. The PII architectural component dedicated to support interconnectivity is the IGW.

The main purpose of IGW is to interconnect Panlab2 testbeds with each other and components inside the testbed with the internet (End User “public access”). Technically seen this component is a Border gateway function (BGF) and ingress-egress point in each site for the IP communication (media plane) for the signalling (this node can decide when and where the signalling shall be routed). For a given packet, characterized by a 5-tuple (IP source, port number source, IP destination, port number destination, protocol type), it is allowed or not to pass the border gateway. The most complete methodology is to correlate the control plane and the transport plane: i.e. authorize the transit of media packets whose addresses match with the media information carried in signalling.

From the PTM’s point of view the IGW is “just another component” in the testbed. There can be one or more IGWs per testbed. PTM utilizes plain SIP to instruct the IGW about creating or deleting connection states inside the testbed and to other testbeds. The IGW server is expected to have several interfaces, one towards the PTM, another towards the internal network of the partner who is offering the resources of the testbed. Further interfaces can be connected to a demilitarized zone (DMZ) or wireless interfaces (e.g. UMTS).

The establishment of a VPN up to now has required intervention by specialized network staff for such operations as setting configuration information in a VPN router, setting key information for encrypting the communication between routers, and changing the peer settings at the time of VPN use. As a result, setting up a VPN has been a costly process requiring a certain amount of time from start to finish. Special consideration must be given in the NATing involved at the IGW.

Since Customers should generally only have access to administration tools „rented resources“ and services he is part of one VPN (per customer, the customer’s VCT) that connects all these points and protects different customers and the platform against security problems and interference with each other. These functions are put dynamically together and managed by ad-hoc VPN establishment functionality triggers by SIP-based messages.

It is very unlikely to terminate a VPN at every kind of lab device so Panlab uses the solution of one VPN per customer (the VCT) connecting all his resources and protects the customer and the platform against security problems and interference with each other.

The per customer's VCT is put together dynamically and managed by IGW ad-hoc VPN establishment functionality, which can be managed by SIP/IMS or other protocols from TEAGLE/PTM. All VCTs are independent VPNs, completely isolated from each other on the same platform.

Since the interconnection gateway is the one dedicated resource to establish connectivity between separated Panlab2 physical test sites, it is used for all use cases that utilize this ability. IGWs are foreseen to mesh automatically with each other and therefore establish connections to other peer IGWs. An important design criterion was to make them as self-configuring as possible and enable instant low-overhead meshing of all IGWs that are part of a specific VCT. Of course the P2P use case with five or more interconnected physical testbeds is the best demonstration field for the concept of such ingress-egress points that allow intra-testbed-communication and shows that it fulfilled its initial requirements. Such dynamically configurable hubs also allow isolation of local testbed devices including different access and quality of service rules set by orchestration tools for each single testbed site. Especially enforced isolation of local testbed collision domains was a key feature requested by project partners and is ensured by this concept. For high-speed or high-bandwidth experiments this of course may be at some point a theoretical bottle neck, but may be worked around by connecting specific IGWs with more dedicated connections or using more than one IGW with separate connections per test site.

4.4 The Resource Adapter concept

The implementation's initial requirements were to allow sharing resources beyond domain boundaries. An important design criterion was that the resources to be shared should be highly heterogeneous in nature. Resources that are currently supported range from general-purpose virtual and physical machines, to Cloud Computing resources, services, and specialized devices. Via dedicated Resource Adaptors (RA, similar to device drivers) that plug into the Panlab Testbed Manager, PTM, arbitrary resources can be controlled making it a generic control framework that strongly supports federation. Experimenters can browse through the resource registry content and can select, configure, deploy, and access booked resources in order to execute network and application layer experiments.

The resources are described according to a common model. This allows for sophisticated resource management across the boundaries of organizational domains. This concept allows us to provide a large pool of federated resources that can be used in any meaningful combination

Resources can exist as Physical- or Logical-Resources where resource providers can define a list of resource instances as specific subtypes based on the model to represent their federation offerings.

The Resource Adapter Description Language (RADL) is introduced to help developers implement resource adaptors. It is critical for the Panlab federation to maintain a rich collection of resources that can be offered in order to attract many experiments to be run on the federated facility. Therefore, making it easy to offer resources via Panlab is a main concern. Hence, the development of RADL.

The resource adapter concept seems to fulfil its initial requirements through the diverse set of implemented use case. PII managed to wrap Resource Adaptors from Virtual Machine hypervisors and hosts, to complete services like mail or SMS servers and transcoding services.

4.5 Controlling the experiment

Many use cases needed to configure resources or even get monitoring status data correctly after the VCT is provisioned and while the testing is in progress. What is critical with the operational part of a VCT is the proper and valid run-time configuration of the participating resources. The reason is that, while the VCT is operated by the federation's customer, the federation must ensure all terms of the requested SLA of the VCT are fulfilled and nothing is violated or falls out of the scope of the SLA. To this end, the SLA must be constantly audited (or monitored) for different parameters (i.e. metering, service quality, security, etc.).

The controlling of resources, through Resource Adaptors was a requirement of the academic use case and thus the FCI was built. Federation Computing Interface (FCI) is an API for accessing and controlling resources participating in an experiment. It is an SDK for developing applications that access an experiment's requested resources through the PanLab office services during operation of

testing. It is quite easy to embed it into a System Under Test (SUT) in order to gain control of the requested resources during the life-time of testing. The FCI is delivered to the customer after the generation of the SLA and not only does it contain the necessary libraries but also the alias of the resources that are used in the experiment scenario. This allows the User-Application/SUT to access the testbed resources during execution of the experiment in order to manage and configure various environment parameters or to get the status of the resources.

4.6 Panlab Processes and legal framework

In accordance with results of the PII WP4, which are summarised in the PII deliverable D1.5 “Updated Version of the Panlab Framework”; the Panlab/PII operational processes are grouped in the following five main areas:

- Business Management (BM) defines all processes concerning requirement management, selection of testbed solution and contract negotiation.
- Testbed Management (TM) comprises all processes to manage entire testing sites including adding a testbed to the Panlab framework and verifying its basic PII compatibility. The Panlab Testbed Manager (PTM) serves as a nucleus for any PII compliant testbed and thus its handling is the focus of TM.
- Resource Management (RM) defines all processes to add physical or logical resources to a testing site. Testing resources are added incrementally to the PTM and are controlled via an abstract configuration layer defined by well-defined resource adapters (RA).
- Session Management (SM) depicts all processes required to conduct individual testing sessions. This includes the process of defining an individual testing session and its binding to the legal contract accompanying each testing session, monitoring and documentation.
- User Management (UM) enables a Panlab partner to manage individual users, grant or deny rights to conduct specific actions on behalf of the PanLab partner, e.g. defining or executing test sessions, manage testing sites and testbeds, and review raw data obtained during test runs.

Processes related to the Business Management cover all the necessary preparations related to inclusion of the required testbeds and testing resources into the Panlab Federation offer; involvement of users of the federated testing resources by defining the required testing configurations; and establishment of the necessary formal agreements. Accordingly, all the stated preparations have been made, in order to ensure that the use cases, described in Chapter 3, can be successfully executed. Furthermore, from the legal and contract points of view, all use case participants agreed on the final PII legal framework, as described in PII deliverable D1.5. With it, Panlab/PII Business Management processes can be considered as validated by the preparations carried out for each of the testing use cases.

Testbed Management and Resource Management processes have been validated in the first stage of the implementation of the use cases, as described in Chapter 3 within the corresponding tables. These present a “use case story board” for each of the use cases, under the event title “Providing the system under test”.

The main part of the framework validation in respect to Panlab/PII operational processes was dedicated to Session Management. In accordance with conclusions from WP4, as summarised in PII deliverable D1.5, the Session Management consists of the following three main tasks:

- Definition of a (new) testing configuration
- Scheduling of a testing session
- Access to testing results

The first Session Management task “Definition of a testing configuration” has been carried out for all the implemented use cases under the event title “Configure the necessary resource parameters in the VCT tool, according to the intended test”, as described in Chapter 3 within corresponding tables. Scheduling of a testing session was performed under the event title “Book the designed VCT and run the experiment”, whereas the event entitled “Acquire results” has been performed, in order to get access to the testing results. Of course, the application of event “Monitor status” was necessary to observe if the defined and selected testing configurations were operating as expected.

Finally, processes related to the User Management have been validated by first ensuring the necessary access for individual users to the Panlab/PII portal and tools (e.g. VCT tool) and allowing them to include testbeds and testing resources within the Panlab/PII offer (performed by testbed providers), as well as select, configure, and execute defined tests (performed by customers).

4.7 Software delivery and Support tools

The PII Software is currently Open Source. There is a public location for downloading the source code: <http://svn.panlab.net/PII/repos/Software/sources/>. One can find there all the source code that comprises the Teagle services: the Orchestration Engine, the Repository, the VCT Tool and the Teagle Gateway. The source code of FCI is also available there. Binaries are also available for individuals that want to just install the software.

The validation of the software components and supported tools was possible through the support and execution of the different set of use cases. All services were used during the development of the use cases: RADL was used to develop the source code for the Resource Adapters. The portal was used to register and manage resources and testbeds. Simultaneously the repository was also tested. All the use cases used the Teagle Services: the VCT tool to create the VCTs, the Orchestration Engine used to provision the VCTs and the Teagle GW to communicate with the individual testbeds. So, all the above components fulfilled their initial requirements.

4.8 Variety of supported use cases

The available PII tools were greatly utilized through a diverse set of use cases. Through the resource adapters it is possible to offer a wide range of resources, thus making available a variety of use cases. It seems possible to perform testing on different domains of IT. By exploiting the available interfaces of existing resources, one can expose his testbed resources easily, regardless of the functionality of the resource. The only requirement is the availability of a software interface and an API that can be wrapped through the RA and exposed to the repository.

5 Lessons Learned

This chapter will collect feedback by all stakeholders towards the update and revision of the Panlab framework.

5.1 BCT

BCT was heavily involved in the design and implementation of a number of components both in Teagle and the PTMs. More specifically, BCT implemented the Teagle GW and the PTM, defined the RAL environment and provided the core management bundles for it and finally, offered a Netbeans module that produced the skeleton for an RA so that developers can easily start writing resource specific code without dealing in depth with the environment details.

5.1.1 Teagle GW

Throughout the lifecycle and evolution of the gateway the experimentation of the partners revealed many issues mainly relating to error tracing and resolution. This need lead to the introduction of extensive logging facilities, however, the overall concept of error tracing should have been dealt with from the beginning of the design so that errors can be isolated and reported with accuracy.

The mechanisms offered at the PTM level for installation of RAs as well as the events an RA may generate drove the design of the TGW as far as the Teagle Repository interaction is concerned. This led to the introduction of mechanisms that allow RAs to be deployed in a “plug and play” schema. The TGW collects the information of the events and acts upon the repository entries. E.g. if an RA is plugged in the RAL, then the next time the VCT Tool enumerates the repository entries, it includes the new type in the list of available resources.

5.1.2 PTM and RAL

Error tracing in the PTM case has also been an issue. Currently, Application Server logging and error reproduction remains the only means for resolving erroneous situations. This, however, requires a developer’s involvement for clarification of the situation. Additional checkpoints should be introduced so as to allow for more functional detection of the issues.

Future thoughts for the evolution of the PTM and RAL involve integrating both in an OSGi environment. This would allow for greater flexibility with respect to maintenance, management and deployment as well as error control.

5.2 DT

Using dedicated IGWs to interconnect resource virtualization has established some best practices that were developed over time to ensure usability and efficiency of design and testing process. Especially handling of inter-domain delays, different bandwidth and other phenomena of multi site experiments heavily influence the experiment results and these limits can require additional domain specific knowledge at experimentation and design time. Many setups created by the Teagle framework and the underlying resource management environment encapsulate stand-alone functions, such as a P2P network simulation, and can be tweaked in advance on a local basis. Furthermore, this allows holding them on standby in a pre-orchestrated state to avoid a full re-orchestration at experimentation run time. If needed, virtualized resources or building blocks can be duplicated for scalability in a pre-configured and optimized way on other sites. Teagle supports easy setup and interconnectivity of such heterogeneous replicated capabilities relying on a common control framework. Resources that are predictable in performance or less performance critical can be allocated to remote locations, however the experimenter needs to keep in mind that such circumstances may be detected by modern (mesh- / routing- / P2P-) algorithms and taken in to account during the experiment.

5.3 FOKUS

Fraunhofer FOKUS was responsible for developing the main components which customers and partners alike used to access the mechanisms offered by the Teagle platform. These components are the VCTTool, the Teagle Portal and the Request Processor.

5.3.1 VCTTool

The VCTTool is the main graphical interface for customers to create their virtual testbeds (VCTs).

Feedback from users has shown that although the VCTTool is a very flexible tool that allows for the design and provisioning of arbitrary testbed layouts, the use of the tool can become somewhat awkward when creating large and complex layouts. This is mainly due to the fact that the said flexibility is achieved by directly exposing the huge amount of configuration options different resources offer to the end user.

The problem has already been somewhat mitigated by introducing a number of mechanisms that either automatically choose configuration options for certain parameters or provide additional information to users about how certain resources can be sensibly configured.

However, in the long run the solution here would be to provide at least some specialized creation environments for complex tasks, e.g. for provisioning large testbeds comprised of a limited number of resource types.

5.3.2 Request Processor

The Teagle request Processor provides a backend for the VCTTool (or any other creation environment) to trigger the actual instantiation of created VCTs. It provides for both synchronous and asynchronous operations. The current implementation has shown to be fully sufficient to perform its tasks.

5.3.3 Teagle Portal

The Teagle Portal provides functionalities to both Teagle customers and Teagle partners. Apart from providing tutorials and documentation, the Portal serves as the primary entry point for customers to access the Teagle platform. From here, the customer can either browse the repository of existing resources or launch the VCTTool to start creating testbed designs.

Teagle partners additionally use the portal to register resource types, manage their PTMs and indicate which resource types are supported by their respective domains.

These functionalities have been continuously updated to keep up with the evolution of the Teagle platform mechanisms.

One issue that has been criticized by users was the initially slow operation of certain aspects of the portal. This was due to a somewhat naive implementation of the data retrieval from the repository. The issue has meanwhile been partially mitigated through heavy employment of caching and threading when accessing the repository contents. The behaviour could be further improved if the repo would provide an interface allowing the execution of some kind of search query.

5.4 OCTO

Octopus implemented use case testing end-to-end Self-Management in a Wireless Future Internet Environment, which allowed for the provisioning of a live WiMAX air interface to a customer. The downlink and uplink traffic of WiMAX was tunnelled to the customer's own testbed. In addition Octo implemented additional BS control software which allows for the dynamic collection of WiMAX link information from the BS to control Quality of Service (QoS) settings on the fly. The BS control software changes QoS service classes by setting a new configuration to the BS using the Simple Network Management Protocol (SNMP). After development of the BS control software and setting up routing and tunnelling manually, Octo implemented RAs for setting tunnelling and routing parameters for the two routers at the Octopus Network and the two routers at the customer premises. Resource Adapter Description Language (RADL) was used to generate source code for each RA. Octo decided

to use a separate RA for each IP tunnelling machine, BS and SS. The RAs managing the tunnelling, send commands to the respective machines via SSH to setup both tunnelling and routing. The default values are stored in each RA.

Implementation of RAs was straightforward and easy using RADL with Eclipse Helios. The customer using the VCT tool needs to input only public IP addresses and user credentials for the two routers in order to setup the IP tunnels and routes.

5.5 TSSG

TSSG was responsible for developing the Repository component of the PII framework. The Repository stores data about PII partner's testbeds, PII users and their requested testbed setups. This data is made available to all components within the framework, e.g. Teagle, PTM, Orchestration engine, etc.

The Repository has proven to be a critical component of the whole PII architecture due to most components within the system needing access to it. For this reason, simple and easily accessible APIs were foremost in the design requirements. REST was chosen as the best approach to implement these needs, as its lightweight, requires no toolkits to build and is a simple interface for clients to use. REST's flexibility can however have some drawbacks; it does not enforce data type checking which can lead to problems with invalid requests. This problem has been alleviated by the provision of XML Schema files.

The original data model for the Repository has evolved to keep track with developing semantics within the framework. The most notable changes in the data model are resource reservations and information about which resource types can be instantiated at a given domain. The interface on the other hand has remained stable. While it has proven that the current interface implementation of the Repository is very flexible it can lack in terms of performance. This is the reason why possibilities for implementing an advanced query interface are currently being evaluated.

5.6 UoP

5.6.1 RADL

After developing a few RAs for resources and since the PTM is implemented in Java, it was evident that templates of code could be used in order to speed up a RA implementation. RADL [7] is a concrete textual syntax which describes a Resource Adapter and is an attempt to describe an RA in a way that decouples it from the underlying code. RADL's textual syntax aims to be easier to describe a Resource Adapter than code in Java. With RADL one defines the exposed interface of a resource and code is automatically generated that wraps the resource and exposes its interface to the Teagle services.

RADL was used in many resources used in the presented use cases. The interfacing used to wrap resources was mostly ssh. Further implementation is under way to wrap for example XML-RPC or http calls.

RADL evolved during the development of the use cases and now supports the full life-cycle of a resource. That is the creation, updating and reading of resource attributes and deletion.

5.6.2 FCI

Many use cases (such as the academic one) need to configure resources or even get monitoring status data properly after the VCT is provisioned while the testing is in progress. What is critical with the operational part of a VCT is the proper and valid run-time configuration of the participating resources. The reason is that, while the VCT is operated by the federation's customer, the federation must ensure all terms of the requested SLA for the VCT are fulfilled and nothing is violated or falls out of the scope of the SLA. To this end, the SLA must be constantly audited (or monitored) for different parameters (i.e. metering, service quality, security, etc).

Federation Computing Interface (FCI) [8] was implemented for this purpose. FCI is an API for accessing resources of the federation and more or less control experiments. It is an SDK for

developing applications that access VCT requested resources through the Panlab office services during operation of testing. It is quite easy to embed it into your application/ SUT in order to gain control of the requested resources during testing.

FCI is really straightforward to use. FCI evolved after the first initial versions and currently also supports the request, reservation and full management of resources. That is during an experiment, one can request, reserve and manage resources on demand. So it is possible now to create much more complex scenarios. Given the fact that FCI is written in Java, one can write code requesting and managing concurrently as many resources as the experiment demands. Large scale scenarios involving large numbers of resources of great complexity are now possible.

5.7 Synchronmedia

In the context of Panlab, the Synchronmedia consortium implemented different cloud use cases where media resources were being used. The consortium also acted as a gateway to other testbeds notably Phosphorus FP6, HPDMnet, FEDERICA and GreenStar Network testbeds. This involved implementing many different types of resources adapters in different ways throughout the project.

For these uses cases different resource adapters were implemented using a variety of methods. Initially, the first resource adapters worked with the FOKUS PTM and were implemented in Java and Python. These early resource adapters allowed control of cloud (storage, network, and compute) resources for OpenNebula environments. Additional use cases which included VideoLAN support, FFMpeg and FEDERICA interactions were developed using the Netbeans plugins and tools developed by BCT and University of Patras. Both resource adapters behave a little differently and the programming models are also different. The experience learned was that integrating third party libraries in the FOKUS PTM based resource adapters was easier to do than having them in the BCT PTM environment. However the BCT environment allowed class loading isolation through its use of OSGi technology. The latest resource adapters were created using the RADL tooling and were much easier and faster to implement than the earlier manual implementations. However when doing these, instead of having API level integration between the Resource Adapter and the client implementation an external command line client was usually being invoked via SSH commands.

In addition to the implemented Resource Adapters, the team implemented its own Canadian Panlab Testbed Manager (PTM). This was due to the fact that current Canadian work involved having its own virtual resources and definitions based on the IaaS Framework. This resource description framework and its related tools are being used to manage the testbeds internally and as such it is more convenient to implement interaction with the other Panlab components at an interface level. This is why the T1 service was implemented as well as a client to the TeagleGW. This allows bidirectional communication between the systems in a seamless fashion. The purpose being to avoid implementing any future resource adapters by having direct translation between the description formats.

Finally, a lot was learned during this project, and we will strive at keeping resources Panlab compatible in the future using the tools (PTM-CA, Resource Adapters) developed under this project.

5.8 Other use case

5.8.1 Real-Time Industrial Networking Infrastructure as testing environment

The testbed for the proposed use case comprises an industrial networking facility. The installation and integration of the core PII components (PTM and IGW) into the networking facility was completed successfully and without any serious problem. The major obstacle imposed, was the integration of legacy software and hardware. Therefore, interoperability issues between the proprietary software and the overall seamless integration to the PII framework arose. Moreover, since one of the distinct characteristics of this particular testbed is the facilitation of a fully deterministic environment, certain time constraints have to be satisfied. Obviously the character of such an environment can be only guaranteed locally, i.e., inside the testbed. Therefore, there is an issue regarding the remote access of the resources and at the same time the retention of the deterministic behaviour of the topology.

However, the infrastructure is established and integrated to the PII platform (PTM and IGW). Furthermore a set of RAs have been developed through which the RT infrastructure can be partially utilized by potential users. Additional effort has to be made in order to find solutions in the area of integrating legacy resources. In any case, since the testbed is established some of its resources will take part in the large scale demo.

5.8.2 Testing solution for the real time evaluation of PLC (Power Line Communications) transmission

Use case proposed by FT/Orange relies on a PLC evaluation platform including two PLC modems and a real time channel emulator to test the robustness and performances of system components at different OSI levels, for transmission over the electrical network in a home environment.

The channel emulator is equipment based on X5-400M PCI Express module from Innovative Integration, including Virtex5 FPGA from Xilinx. It emulates the effect of various perturbations (multi path, Gaussian noise, impulse noise...)

Originally, configuration and control of the channel emulator is provided by a local PC (in the same box as the channel emulator) under Windows XP.

For integration into Panlab architecture, the principle is to include the Panlab Testbed Manager (PTM), implementing the Resource Adaptation Layer, and the specific Resource Adapter (RA) inside the configuration and control PC.

Software packages and instructions to implement PII components have been downloaded from PII TRAC and have been very helpful.

The following components have been installed and configured on the PC under Windows XP:

- Java Development Kit
- Glassfish ESB V21 (first configuration) then ESB V22 (second configuration)
- MySQL community server
- PTM Manager (PTM)
- Resource Adapter Layer (RAL)

With the first configuration using ESB V21 it was not possible to configure T1Adapter with https protocol, to secure the T1 connection between TeagleGW and PTM. The problem was solved with the second configuration using ESB V22.

The main difficulty was the way to define and integrate a Resource Adapter (RA) inside the PLC channel emulator. The original configuration software is developed with Microsoft Visual C++ environment. Click buttons functions are used to set channel parameters and to start/stop the emulation.

Among various options, an idea was proposed by UoP to create a small server and to integrate it in Visual C++ program to listen for remote requests. Then the RA, acting as a proxy would connect to the server and dispatch the request when it receives an UPDATE command

Several source codes for small servers have been found on the Web, fitted and successfully compiled with Visual C++. But in all cases the compiling and/or linking options for those servers were not compatible with the original options required for the configuration program, and it was not possible to aggregate the codes.

So the development of the RA was not completed, given the fact that no important effort was anticipated for that use case software integration in WP5.

6 Conclusions

This deliverable has summarized the experiences made through the final integration of the PII framework and its individual components. The experiences gathered provide an evaluation of the Panlab infrastructure that aimed at delivering automated tools and agreed processes for the deployment of testbed environments for research experimentation and system testing.

The objective of the project was to deliver such a framework that allows an easy matching of offered resources by testbed providers and requested resources by testbed users. Automated tools allow the easy testbed service orchestration and provisioning and also the runtime interaction with the used resources.

The framework was evaluated through the implementation of a number of use cases that were defined in the beginning of the project and which were used to extract functional and non-functional requirements that the PII framework had to satisfy.

Among a fairly large number of quite heterogeneous use cases, two use cases did attract a major attention, since they exhibited the bulk of the requirements. The first of these use cases that required the provisioning of a testbed environment for an adaptive admission control and resource allocation algorithm, exhibited the need for a mechanism allowing the system under test or the experimenter to directly interact with the testbed environment and alter its characteristics, e.g. the available computing capacity. The second of these use cases required the provisioning of a testbed environment for mobile Web TV, a rather common application. However the distinguished requirement of this use case was the capability to provision user subscriptions as part of the deployment of the testbed environment. This capability is a small first step towards the inclusion of users in an experiment, and specifically as part of the testbed environment.

Further use cases were implemented that demonstrated the feasibility and the usability of the framework. It is noteworthy that the architecture and concepts allow the integration and provision of very heterogeneous resources as demonstrated e.g. by the use case on integrating real-time industrial networking infrastructure as testing environment or by the testing solution for the real time evaluation of Power Line Communications transmission.

Finally the framework was evaluated by external use cases that were provided by "customers" of the framework. Among others, these use cases include the provisioning of a live network for testing end-to-end self-management in a wireless Future Internet environment provided by the Self-NET¹ project and the deployment of a peer to peer application over a large number of nodes provided by the VITAL++² project.

In a nutshell the PII framework consists of several components that all have their role in the provisioning process of a requested environment. The main components are: (i) the Teagle tools and services, (ii) the testbed interconnection capabilities, and (iii) the Panlab testbed manager and resource adapter concept.

The Teagle tool and services allow the flexible definition of testbeds through a graphical tool, called VCTtool, as well as programmatically. A testbed configuration is presented via a Virtual Customer Testbed (VCT) formal description allowing the reproducibility of an environment. The programmability allows the definition of very large scale testbeds that cannot be practically drawn on the canvas of the VCTtool. A number of other services that facilitate the provisioning are hidden to the customer, but play an important role in overall framework. These are, among others, the Resource Repository, the Request Processor, the Orchestration Engine, the Policy Engine and the Teagle Gateway. The details of these functions are reported in other deliverables.

The interconnection capabilities, largely implemented by the Interconnecting Gateway (IGW) allow the isolation of VCTs across different domains and infrastructures. The IGW is deployed when

¹ <https://www.ict-selfnet.eu/>

² <http://www.ict-vitalpp.upatras.gr/>

necessary through the Teagle services and autonomously establishes a virtual private network that interconnects all resources that are part of a VCT.

Finally the Panlab testbed manager and resource adapter concept enable the controlled sharing of resources across domain boundaries and the incorporation of resources in a technology agnostic manner. The intrinsic knowledge of the capabilities and control/management properties of a resource is allocated to the resource adapter thus allowing a resource agnostic handling of a testbed environment at the Teagle level. Furthermore, and because the Panlab testbed manager is deployed at domain boundaries, allows for policy enforcement at a single point.

Throughout the implementation of all use cases, the basic concepts and components of the framework proved their effectiveness to serve the main objective of the project. Certainly a number of improvements and extensions are possible. A number of these improvements are related to ease of use and better guarantees of the properties of the provisioned testbed environment. These are subject to future work and will be added as appropriate by the individuals and organisations that see a value in doing so. An example of better usability is the automatic detection of resource properties by the Panlab testbed manager, utilizing some sort of automatic service discovery mechanism. Another example is the semantic integration of alternative resource description languages that exist, so that a given resource does not need to be re-described using the Panlab resource description language (an adapted version of DEN-ng), given that it is already described via another modelling language. Finally a better utilisation of potentially existing networking capabilities by the IGW is possible.

Taken together, these results suggest that the framework can well serve the purpose for which it was built, namely the provisioning of heterogeneous testbed environments. Nevertheless the versatility of the framework can further be used for provisioning and managing diverse next generation network infrastructures and provisioning of dedicated service platforms that offer a closer match of the required functionality to the available capability. Towards this end the concept of a Virtual Customer Testbed (VCT) is perfectly aligned with the general trend to virtualisation of resources.

References

- [1] <http://www.panlab.net>
- [2] Self-NET project, <http://www.ict-selfnet.eu>
- [3] Apostolos Kousaridas, Gérard Nguengang, Julien Boite, Vania Conan, Vangelis Gazis, Tilemachos Raptis, Nancy Alonistioti, “An experimental path towards Self-Management for Future Internet Environments”, Book Chapter In: “Towards the Future Internet - Emerging Trends from European Research” [ISBN 978-1- 60750-539-6], Edited by Georgios Tselentis, Alex Galis, Anastasius Gavras, Srdjan Krco, Volkmar Lotz, Elena Simperl, Burkhard Stiller pp. 95 - 104, 2010.
- [4] Website of Panlab and PII European projects, supported by the European Commission in its both framework programmes FP6 (2001-2006) and FP7 (2007-2013): <http://www.panlab.net>
- [5] Octopus Network test facility <http://www.octo.fi>
- [6] IEEE 802.16 Working Group, editor. IEEE Standard for Local and Metropolitan Area Networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems. IEEE Std. 802.16-2004, October 2004.
- [7] <http://trac.panlab.net/trac/wiki/RADL>
- [8] <http://trac.panlab.net/trac/wiki/FCI>

Annex A <http://www.panlab.net/use-cases.html>

A.1 Recent Panlab II Use Cases

A.1.1 Future use case - Using Panlab via Teagle by Starhome and FT-PSC project (By Starhome and FT-PSC project)

In general a roaming broker enables roaming between two networks that do not have direct roaming relationships, by using an identity of a 3rd mobile network which does have roaming relations with both the visited and the home networks.

Starhome has built an IMS roaming broker that should be tested in various roaming scenarios. Since it bridges between different IMS networks, the compatibility with different IMS vendors is most important. The FT-PSC project (Federated Testbeds - Public Safety Communications) will provide few IMS test beds for conducting roaming tests. However it is highly recommended to increase the number of IMS test beds via external resources. Panlab exactly provides an access to a number of IMS labs maintained by different vendors. The Teagle tool facilitates the access to those labs by providing a single interface. Starhome has already started to investigate this tool and certainly would like to use it in the near future (2011).

A.1.2 Testing trans-coding video through dynamic cloud allocation (By Synchronmedia and Inocybe inc. and Communication Research Centre Canada)

On the fly, transcoding of video content or streams is becoming an increasing requirement for the media industry. This is mainly due to the increase in the number of different devices and platforms that can be used to play the video content as well as the numerous numbers of available codecs. Cloud computing is a natural way to do on the fly transcoding of these streams as it is possible to create a temporary computer farm needed to run these transcoding jobs without having to configure them manually. Therefore, depending on the size of these video files, the content can be delivered to any type of device, seamlessly. In this use case, the Panlab facility is used to transcode a high quality HD file used to stream live HD content to powerful computers to a more portable Mpeg 4 format that can be viewed on portable devices.

A.1.3 Testing Multicast Streaming on Dynamic Networks (By Synchronmedia, Inocybe inc. and Communication Research Centre Canada)

Multicast is becoming a ubiquitous technology however it still remains a challenge to deploy in some commercial networks. With offerings like IPTV becoming predominant Internet content it is required to see the impacts of multicast traffic on the network when having high latency environments such as international networks. However, configuring multicast capable networks across all these management domains and having the layer 1 connectivity on-demand, can be a lengthy process. That is the experimental international dynamic network called HPDMnet is being used in this experiment. Using this facility provides all the required substrate to do these low level multicast capable networks on-demand.

A.1.4 Testing Uncompressed HD Streaming (By Synchronmedia, Inocybe inc. and Communication Research Centre Canada)

High Definition media is becoming important in the industry because of the needs in the movies and entertainment industry. Streaming uncompressed HD requires high bandwidth dedicated links. Such links are hard to obtain in multi-domain networks. In this use case different cinematic studios want to share and stream HD content over a high speed network. This use case is interesting to run via the Panlab facilities because these networks are not available everywhere. In this use case a customer can create a dynamic link on the fly within HPDMnet network between to site. Then, the customer can control uncompressed live HD streaming session by controlling Ultragrid software through Panlab tools (Teagle and VCT Tool).

A.1.5 EzWeb application over TID SDPLabs: “PII Message Sender” (By Telefónica I+D)

Telefónica is opening some of its network functionalities to third party companies through a SDP (Service Delivery Platform) exposing a number of open APIs (SOAP and REST based) that can be used by companies and developers to build a new generation of telecommunication enabled applications and services. The TID SDPLabs is an environment meant for development and experimentation, which can be offered to third parties companies and developers around the world to develop their applications in an environment that offers real and simulated core telecommunication network capabilities. A third party company using the testbed TID SDPLabs will be able to make use of the following initial set of Telefónica’s network capabilities: Messaging, Contacts and Presence.

A.1.6 Testing end-to-end Self-Management in a Wireless Future Internet Environment (By Octopus network, University of Athens, and VTT Technical Centre of Finland)

The Self-NET project has designed the architecture and the software for communication networks self-management based on the so called closed control loop or Monitor-Decide-Execute Cycle. The scope of this use case is to experiment on the improvement of QoS features (e.g., packet loss, delay, jitter) using the Self-NET software for self-management over a live network environment and exploiting monitoring and configuration capabilities that different administrative domains provide (i.e. access network and service layer). The effectiveness and the feasibility of various parameters optimization of existing network protocols avoiding manual effort is also tested.

A.1.7 Testing enhanced Web TV services over mobile phones (By Technicolor, Images & Reseaux, and COSMOTE)

Web TV is a common application over the open Internet but enhancements of this service is contemplated thanks to additional components permitting to add Quality of Service. Typically real time video adapters and transcoders are under investigations to ensure more efficient delivery of the video streams over the internet. The adapters offer to resize the video content with respect to terminal characteristics. The transcoders permit to face different network bandwidth capabilities. The two components proposed here are the starting and ending points of the web TV service. The particular use case setup offers the possibility to perform tests upon:

- A Video on demand Service platform
- End users in a real «mobile operator» context.

A.1.8 Testing Adaptive admission control and resource allocation algorithms (By University of Patras)

In order for one to test an adaptive admission control and resource allocation algorithm, it is necessary to set up an appropriate testbed of a distributed web application like RUBiS benchmark, an auction site prototype modelled after eBay.com. It provides a virtualized distributed application that consists of three components, a web server, an application server, a database and its workload generator, which produces the appropriate requests. Furthermore it can be deployed in a virtualized environment using Xen server technology, which allows regulating system resources such as CPU usage and memory, and provides also a monitoring tool, Ganglia, that measures network metrics, such as round trip time and other statistics, and resource usage in virtual machines.

A.1.9 Stress test the Open IMS core (By Fraunhofer FOKUS)

This use case illustrates the design steps via Teagle for an environment in which one can execute stress testing the Fraunhofer FOKUS open IMS core. It starts with configuring a low end server that is exposed to a specified traffic load via the SIPNuke tool. It obviously breaks under the load within a short time. Following the breakdown of the server the use case uses again Teagle to configure an environment that uses a higher capacity server in a cloud computing environment. This environment now survives the specified stress test.

A.1.10 Testing a VOIP user agent (By University of Patras)

A Federation Customer wants to test his own VOIP client application against operation conformance. Specifically the application's performance regarding:

- Sound and recording capabilities and conformance
- Dialtone correct operation and conformance

The customer wants a simple setup of an Asterisk server and 2 VOIP user agents configured to use this VOIP server. Then he will test his VOIP user agent with the Asterisk service and the configured VOIP user agents. The Sound and recording capabilities and conformance will be accomplished through and ECHO VOIP service and the dialtone correct operation and conformance against a DTMF VOIP service. The customer's test suite consists of the following two test cases:

1. From the customer VOIP application a call to an extension number ie #301 will be made towards the Asterisk server in order then to be redirected to the ECHO VOIP service.
 2. From the customer VOIP application a call to an extension number ie #401 will be made towards the Asterisk server in order then to be redirected to the DTMF VOIP service....
-