SEVENTH FRAMEWORK PROGRAMME

CAPACITIES

Research Infrastructures

INFRA-2007-1.2.1 Research Infrastructures

DRIVER II

Grant Agreement 212147

"Digital Repository Infrastructure Vision for European Research II"

Software Release Plan

Deliverable Code: D6.1

Document Description

Project	
Title:	DRIVER, Digital Repository Infrastructure Vision for European Research II
Start date:	1st December 2007
Call/Instrument:	INFRA-2007-1.2.1
Grant Agreement:	**212147**

Document	
Deliverable number:	D6.1
Deliverable title:	Software Release Plan
Contractual Date of Delivery:	31st of January 2008
Actual Date of Delivery:	17th of March 2008
Editor(s):	CNR
Author(s):	Paolo Manghi
Reviewer(s):	Natalia Manola
Participant(s):	Marko Mikulicic
Workpackage:	WP6
Workpackage title:	Service Integration Management
Workpackage leader:	ICM
Workpackage participants:	NKUA, CNR, ICM, UGENT
Distribution:	Public
Nature:	Report
Version/Revision:	1.7
Draft/Final:	Final
Total number of pages: (including cover)	
File name:	
Key words:	*Services, Software, Release, Planning*

Disclaimer

This document contains description of the DRIVER II project findings, work and products. Certain parts of it might be under partner Intellectual Property Right (IPR) rules so, prior to using its content please contact the consortium head for approval.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify us immediately.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of DRIVER consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 25 Member States of the Union. It is based on the European Communities and the member states cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors. (http://europa.eu.int/)

DRIVER is a project funded by the European Union

Table of Contents

Table of Figures

Summary

The aim of this deliverable is to present the software design and production steps leading to the D-NET v1.0 release of the DRIVER Infrastructure.

1 Introduction

1.1 Purpose of this document

This purpose of this document is to present the plan to be carried on in order to achieve the production releases of the DRIVER Infrastructure software D-NET v1.0 and D-NET v2.0. The plan is based on the DoW [2] and is driven by the specification documents relative to the various Services **Error! Reference source not found.**.

D-NET v1.0 is expected at month 6, namely May 2008, while D-NET v2.0 at month 21, namely August 2009. The current deliverable is then to be updated at month 13, namely December 2008, in order to provide the plan for the second release.

1.2 Document Outline

Section 2 introduces all the overall design and development plan, by presenting the main changes expected in the Services and in the architecture logic. Next, it specifies the challenges to be faces for each Service, presented by Service layers.

Finally, Section 3 specifies the software delivery practice to be adopted for D-NET v1.0, envisaging two candidate infrastructure release steps before issuing the final production infrastructure.

2 Production plan

DRIVER-II features two main software/infrastructure milestones, D-NET v1.0 at month 6 (M6.1) and D-NET v2.0 at month 21 (M6.2). Next the production plan for D-NET v1.0 is envisaged, while the road to D-NET v2.0 will be determined after month 6, once D-NET v1.0 will be released.

The most notable changes are:

- The migration from Perl implementation of the Services to a Java implementation: the change was mainly due to performance issues, compatibility issues, and packaging/deployment issues. Indeed, Perl revealed a number of problems in dealing with SOAP and threads, and Web Services and SOAP are not fully interoperable across different languages. Besides, adopting the same language platform permits common packaging and deployment methodologies and eases the diffusion of software.
- The definition of a common core of libraries for the interaction with the infrastructure enabling layer (enabling layer client). All Services are now implementing their own special libraries to be able to participate to the infrastructure life-cycle (registration/deregistration and discovery of the services). Providing a common core of libraries will ease Service developers in their activities as well as the diffusions of possible modifications to the enabling layer.
- Improvement of the infrastructure logic: the addition of new Data Structures and Services empowering and simplifying the definition of applications as set of interacting resources. Multiple independent "applications", each made of a specific selection of potentially shared resources, will be supported.

2.1 DRIVER Core Libraries

CNR will produce a number of libraries to be used in the implementation of all Services in order to interact with the enabling layer Services. The libraries will deal with:

- IS-Registry client: Service registration, deregistration, and resource profile updates;
- IS-Lookup client (resource discovery): find a resource profile and get the respective Service stub;
- IS-SN client: subscribing a topic to the IS-SN.

On DRIVER nodes all Services will have to deal with automatic discovery of Services through the IS (this leads to simple configuration files).

2.2 Infrastructure logic

The logic of the Infrastructure, i.e. the way resources should interact to support special community applications, will be modified to support the notions of:

- *Regions*: set of resources defining the interaction boundaries of such resources. A resource is activated in the context of a region and can only interact with the resources in the same region at that moment in time. For example, an UI belongs to one region; a query execution can only be demanded to the Search Services available in the same region of the User Interface at search-time.
- *Groups*: set of resources bound together by particular application-oriented properties. Groups do not define any interaction boundaries between the resources they contain; a group is used to support actions on sets of resources. For example,

an Index Service should be designed to run a query on a group of Index DSs, rather than on an explicit list of such DSs. The overall organization of the system can be notably simplified by this simple and natural notion.

- *API contracts*: a Service implements an API whose *contract* establishes its method's signatures and semantics. WSDL and Java Stub of the Services are available from the support site.
- *Sub-Services*: Services implementing only a part of an API contract. The existence of sub-Services must be declared as an explicit dependency in the IS. Examples are: the embedded ResultSet of the ICM Index obeys only to a sub-part of the ResultSet Service contract, that of ResultSet access. As such, is a sub-ResultSet.
- *Moving from Repository Services* to *Repository Data Structures.* Indeed, Repositories are not DRIVER Services, but rather Data Structures to be managed by a special Service, called *Repository Management Service*. The Service operates as a generic OAI-PMH Repository harvester, returning result sets of the harvesting results. It therefore operates independently of the Aggregation Service, which will deal with input/output ResultSets, and can be reused for other purposes by other Services.

2.3 D-NET v1.0

The software in D-NET 1.0 is the outcome of refining and stabilizing the DRIVER Testbed software delivered in the DRIVER project. The aim is to release installable software packages, capable of supporting a running and stable production infrastructure, and also deploy such software on a number of production machines. In this section we detail the expected Service modifications by infrastructure layer and schedule the respective delivery dates. In particular, the following Service software is to be released:

- *Enabling Layer*
  - Information Service (CNR)
  - Manager Service (CNR)
  - Authentication and Authorization Service (ICM)
  - ResultSet Service (CNR)
- *Data Layer*
  - MDStore Service (UniBi)
  - Index Service (UniBi)
  - Browse Service (CNR)
  - Aggregator Service (UniBi)
  - Collection Service (CNR)
  - Publisher Service (CNR)
  - Search Service (NKUA)
  - Validator Service (NKUA)
  - Repository Management Service (UniBi)
- *Functionality Layer*
  - User Interface Service (NKUA)
  - User Profile Service (NKUA)
  - Community Service (NKUA)
  - Recommendation Service (NKUA)

- Text Engine Service (CNR)

It must be remarked that not all such Services will be completed in D-NET v1.0 and some of their functionality may be missing, still without compromising the delivery of a production infrastructure. The plan is articulated in

Service	***Partner***	***Alphav1: Freezing Date 31st of March***	***Type***	***Alphav2-Beta: Freezing Date 30th of April***	***Type***	***RC: Freezing Date 15th of May***	***Type***
Enabling Layer							
Core Communication Libraries		-	-	15/04/2008		15/04/2008	**IN-I**
	CNR, NKUA	-	-	Automatic registration component	**IN-I**	-	**-**
	CNR, NKUA	-	-	IS-lookup client: no cache in D-NET v1.0, support for transparent ResultSet iteration (see NKUA code)	**IN-I**	-	**-**
	CNR, NKUA	-	-	Service locator (stub finder)	**IN-I**	-	**-**
	CNR, NKUA	-	-	IS-Registry client	**IN-I**	-	**-**
	CNR, NKUA	-	-	IS-SN client	**IN-I**	-	**-**
	ICM, NKUA	-	-	-	**-**	AAS Client	**IN-O**
	CNR, ICM	-	-	-	**-**	Node Secure communication layer: regions	**OUT**
	All	-	-	Alignment of all Java Services: packaging, common libraries, automatic registration	**IN-O**		
Hosting Node Manager Service		-	-	30/04/2008		15/05/2008	
	CNR, NKUA	-	-	HNM in Java: simple profile update	**IN-I**	-	-
	CNR	-	-			Munin-SmokePing Integration	**IN-O**
Information Service		31/03/2008		30/04/2008		15/05/2008	
	CNR	Profiles for Groups of Services and Repository Service with status	**IN**	-	-	-	-
		-	**-**	Service profile changes: 1) separate date of creation and date of update 2) Service status in profile	**IN-I**	Service profiles with status	**IN**
	CNR	Refinement of IS-SN in Perl	**IN**	-	-	-	-
		-	**-**	-	**-**	on Space Views integration: waiting fo	**OUT**
		-	**-**	-	-	Java implementation	**IN-I**
	CNR, ICM	-	-	-	-	Region profiles and management	**OUT**
	CNR	-	-	IS-Store: Java + Exist 1.2	**IN-I**	IS-Store: Java + Exist 1.2 + service profile status	**IN-I**
Manager Service		31/03/2008		30/04/2008		15/05/2008	
	CNR	Managing multiple indices and stores	**IN**	-	-	-	-
		-	**-**	-	-	rol panels and monitoring tools: re-wo	**IN**
	CNR		-	Java implementation	**IN**	-	-
Authentication and Authorization Service		-	-	-	-	15/05/2008	
	ICM	-	-	-	-	Service-2-Service secure communication	**IN-O**
ResultSet Service		-	-	-	-	15/05/2008	
	CNR	-	-	-	-	Java implementation	**IN-O**

, after an in-depth analysis of the results expected from D-NET v1.0.

2.3.1 Enabling layer

As mentioned above, all Services in the Enabling Layer will be re-implemented in Java, except from the Authentication and Authorization Service, which is Java based already.

Information Service

The Information Service (IS) consists of four components:

- IS-Store
- IS-SN
- IS-Lookup
- IS-Registry

The IS-SN will have to support parallel notification queues. This is to be achieve by implementing two extra queue management "agents", as specified in the Enabling Layer Specification [4].

The Service in the need of heavy re-design is the IS-Store, which should enforce:

- distribution/replication of storage space in order to increase robustness and scalability of the Infrastructure (see [4] for further details);
- porting to XML query engine Exist v1.2.

Manager Service

The Manager Service consists of two main components:

- MS-RO: resource orchestration deals with the orchestration activities required for the maintenance of the Information Space
    - Repository registration and validation: the action fires the creation of the MDStore DSs needed to store the metadata formats available from the Repository, the creation of the replicas of such MDStores, and the Harvesting Instance DS needed to manage the Repository Service; such notions will be replaced by the notions of Transformation Data Structure ad Repository Data Structure, respectively, as mentioned above.
    - Storing data in a DMF store: creating the relative Index DS, if it does not exist, and its replicas; then feeding them with the content of the DMF MDStore which fired the event;
- MS-RM: resource monitoring has to do with,
    - Consistency check/corrective operations, controls deal with:
        - Index Data Structures that do not target any MDstore;
        - Missing MDStore and Index DSs replicas
    - Robustness check/corrective operations, now dealing with the evaluation of the performances of DRIVER Hosting Nodes. Performance estimation is carried on by administrators interacting with the administration User Interface, using the tools provided by Munin Project [3]. Checking the various performance parameters on a given period of time, administrators can take load balancing decisions, such as installing new Services, e.g. ResultSet Services and Index Services, or moving them from one DRIVER Hosting Node to another.

Authentication and Authorization Service

At the current stage, the Authentication and Authorization Service (AAS) is used only for Users access policies. The AAS will:

- support secure Service-to-Service communications;
- verify the consistency of *region*-imposed constraints, implicitly determined by the context of the requesting Service.

ResultSet Service

The ResultSet Service will be rewritten in Java.

2.3.1.1 Steps to D-NET 1.0

The re-implementation of the Services in Java will be carried on keeping in mind the modifications required to enhance resource management in the Infrastructure. Such changes are concerned with the notions of *region, group, sub-service* and *API contract* mentioned above. All such aspects should be introduced in a step-by-step, non-invasive way, in order not to block Service refinement at the different partner sites and favor the shortest possible change-integration period.

In particular, the delicate aspect of secure Service-to-Service communication will have to be faced by avoiding the usage of heavy SOAP embedded security protocols and still preserving as much as possible the non-proprietary nature of the DRIVER architecture.

Time-wise, the first version of the Enabling Layer will respect the alpha and beta candidate release deadlines pointed out above.

2.3.2 Data layer

MDStore Service

The MDStore needs to improve its stability in the presence of large files and needs optimization in the file indexing process.

Index Services

Two implementations of the Index Service are available, one from ICM (Java-based, based on Yadda index technology developed at ICM Labs) and one from UniBi (Per-based, built on top of Lucene).

Both implementations need "search refinement" to be better specified and implemented.

UniBi requires instead a re-development in Java language. The need of such implementation and its delivery in D-NET 1.0 or D-NET 2.0 is to be evaluated based on the human and time resources available to the partner.

Browsing Service

The Browsing Service needs to be re-developed in Java. Its current implementation supports one-level browsing and is delivered by CNR.

A further version of the Browsing Service, to be developed at UniBi, should support drill-in functionality. The need of such implementation and its delivery in D-NET 1.0 or D-NET 2.0 is to be evaluated based on the human and time resources available to the partner.

Aggregator Service

The current implementation of the Aggregator Service obeys to the specification of version 1.0.1 available at: http://technical.wiki.driver.research-infrastructures.eu/index.php/Aggregator_Service. Aggregator manager users are capable of defining through a User Interface the mappings (*transformations*) from Dublin Core to DMF records. In D-NET v1.0 the Aggregator Service should extend its import boundaries to Repositories that are not necessarily OAI-PMH compliant; e.g. FTP based.

The next version of the Service v1.2.0 to be released in D-NET v2.0 will instead:

- Accept special input ResultSets, where each entry consists of an *Object Record* (http://technical.wiki.driver.research-infrastructures.eu/index.php/Object_Records), i.e. a record containing a number of different records.
- No longer be based on the notion of Harvesting Instance DSs, but on the notion of Transformation Data Structures. Such DSs describe the input metadata formats of the transformation, the output metadata format, and the set of rules in between.
- Enable the definition of a set of rules combining fields from all the input records in each entry so as to generate one output record.
- Harvesting will be delegated to Repository Management Services (see below).

Thus, to the aim of D-NET v1.0, the Aggregator Service should at least:

- Delegate OAI-PMH harvesting to Repository Management Services.
- Work with an input Object Record ResultSet and an output metadata record ResultSets for DMF.
- Replace Harvesting Instance DSs with simplified Transformation DSs, conceived to contain generic mapping rules, but initially limited to representation of mappings from DC to DMF. This is to say that in D-NET v1.0 we might not target the generic mappings from format to format.

Collection Service

The Collection Service has to be re-implemented in Java.

Publisher Service

The Publisher Service needs to be re-implemented in Java. It offers an OAI-PMH interface (http://oai.driver.research-infrastructures.eu/oai) to the DRIVER Information Space, plus a method to access any harvested or DRIVER generated record by its unique DRIVER identifier. The Service resolves DRIVER absolute identifiers to fetch and deliver the records to the calling Services.

Search Service

The Search Service will integrate the notion of Information Space Views (ISV) and Index Map Data Structures. The former, as explained in ScrewDRIVER wiki [1], are configured by administrator to specify how the records in the same Information Space should be organized by the available indexes. The result of distributing the MDStores, i.e. the records they contain, over the different Indices according to the ISV specification returns a number of pairs Index-MDStore whose status is maintained in the Index Map Data Structure.

Validator Service

The Service performs a quality check of the functionalities of OAI-PMH Repositories and of the content they offer. It can be configured to test different aspects of the OAI-PMH

protocol default as well as special rules specifying the expected content of the harvested records; e.g. what range of values are expected for a certain metadata field, the XML structure of the records. In the context of the DRIVER Information Space Application it is used to check whether the Repositories registered to DRIVER are compliant to the DRIVER Guidelines for Content Providers.

The Service will have to update the Repository Data Structure profiles with information about their current status of validation.

Repository Management Service

The Service operates as a generic Repository harvester, returning result sets of the OAI-PMH harvesting results for a certain Repository DS. It therefore operates independently of the Aggregation Service, which will deal with input/output ResultSets, and can be reused for other purposes by other Services.

If new kinds of data sources, accessible through different protocols, will have to be introduced in DRIVER a corresponding Service-Data Structure pattern will have to be created. For example: FTP Site Service and FTP Site Data Structures.

2.3.2.1 Steps to D-NET 1.0

The majority of the Services are responsibility of UniBi. The MStore Service should be re-implemented, the Aggregator Service re-designed and re-implemented, while the Repository Management Service should be designed and implemented. These activities are quite committing and may not all be accomplished for D-NET 1.0. In order of priority:

1. MDStore Service: java;
2. Migrating to Repository Data Structures: new profiles and integration with Validator Service results;
3. Designing and implementing the Repository Management Service;
4. Designing and implementing the new Aggregator Service according to the strategy above.
5. Full-text indexing capabilities should be enforced. This can be done by implementing special index management policies at aggregation time, but still tools for document extraction and document preservation must be devised. At the same time full-text search will have to be added to the User Interface.

2.3.3 Functionality layer

User Interface Service

The UI is now customized to DMF, which means that it cannot be automatically adapted to a different format: advanced searches and display of search results are bound to DMF record structure. D-NET 1.0 will deliver an automatically configurable interface, useful to automatically serve DRIVER to all communities willing to use it, independently from the metadata format they adopt.

Community Service

Nothing to be signaled.

Recommendation Service

The Recommendation Service can send users, either via email or by display on login to DRIVER, "announcements" relative to the registration and validation of new Repositories into DRIVER. Other more sophisticated recommendations are still under development and will not be released in V1.1.0. At such stage, for example, users will be notified of the harvesting of documents from their favorite Repositories

Text-management Service

In order to build full-text Index Services, text-related operations should be performed; e.g. keywords have to be extracted from the text of a document, in order to enable full-text searches. Text management tools are made available through a specific Service, to be used by Aggregator Service instance, but also available to other Services which might join the infrastructure in the future. The Service needs to be re-designed in Java.

Repository Workflow Interface

Repository Managers, Aggregator Manager and Country Correspondents should have access to an UI through which Repositories can be registered, edited in their properties and properly validated before being introduced into the DRIVER Information Space. The validation status of a Repository determines its position in the workflow, which ends in harvesting, aggregation and indexing of its content. The specification of such a process can be found in the technical wiki [1].

2.3.3.1 Steps to D-NET 1.0

Full priority is to be given at the User Interface, which should be fully operative and generic for D-NET v1.0.

The Repository workflow should be fully supported by the relative User Interface.

2.4 D-NET v2.0

The DoW [2] draws the attention on two main activities: the enhancement of the Services of the DRIVER Testbed and the realization of new Services, to serve Complex Objects based applications. D-NET v1.0 accomplishes part of the first task, including the policies and rules needed to maintain a running infrastructure alive. D-NET v2.0 will determine the completion of both the activities. In particular, the following extra Services are expected:

- *Enabling Layer*
  - Information Service: fully scalable version with low-level distributed IS-Store (CNR)
  - Manager Service: supporting automatic robustness methodologies (CNR)
  - Authentication and Authorization Service: distributed version coping with efficient Service-to-Service secure communication (ICM)
- *Data Layer*
  - Content Services: typed complex object Repository (CNR)
  - Access Services: evaluating references to external objects, when references are used within complex objects, i.e. enabling access to "external object surrogates" (CNR)
  - Advanced Harvesting Service: harvesting complex objects into Content Services typed containers (UniBi)
  - Advanced Aggregator Service: transforming complex objects into complex objects (UniBi)

- Publisher Service: exporting complex objects through ORE interfaces (CNR)
- Search Service: dealing with Content Services complex object data model (NKUA)
- Reference Inference Service: automatic inference of references between sets of publications (ICM)

- *Functionality Layer*
  - Advanced User Interface Services: user interfaces to create, delete, modify and search complex objects in Content Services (NKUA)
  - Active Information Discovery Service: user interfaces to create, delete, modify and search Enhanced Publications represented as complex objects in Content Services (NKUA)
  - Advanced User Profile Service: including extra user profiling functionalities (NKUA)
  - Advanced Community Service: including extra community-oriened functionalities (NKUA)
  - Advanced Recommendation Service: adapting to complex objects nature (NKUA)

3 Logistic

3.1 Candidate releases management

The delivery of the D-Net 1.0 software release is to be preceded by two pre-releases (one *alpha* and one *beta),* a *first release candidate* (*rc1*) and optionally a *second release candidate*.

The source code and binaries of each release (including *pre-releases*) shall be correctly packaged and stored on the official release package repository, but only the *release candidate* and D-Net 1.0 release code shall be available for public download from the repository.

Once an *pre-release* is issued, the current public visible Testbed infrastructure (known as *production infrastructure*) will be upgraded with the respective *pre-release* code base.

The *release candidate* code base instead will not be applied over the current public production infrastructure, but a new set of machines will be dedicated for it's deployment. The new production deployment will start with the *release candidate* software release and will be subsequently updated until it becomes the new public production infrastructure running the final D-Net 1.0 release code base.

Here is a more detailed view of each release step:

- The *alpha* release is to be issued at the end of March 2008. The release will contain the latest stable version of the services, namely that shown for the review. Some of the functionalities may be missing, but the resulting infrastructure will be stable for demos and Information Space maintenance. Note: alpha v1 coincides with the final production of the DRIVER Testbed, developed by the DRIVER Project.
- The *beta* release is to be issued at the end of April 2008. The release will contain the latest stable version of the services, independently from their current status of development, available for all partners to develop in stable environments. The idea is to start deploying the services and initiate testing with relevant and significant data. If some component is stable enough it may be used to update the current production infrastructure.
- The *first release candidate 1* (*rc1*) is to be issued the 15th of May 2008. The release will contain the last stable version of the Services, for those Services that reached stability, deployed on new production machines according to a specific plan and available for download and installation to the public. Except for unexpected blocking emergencies, such Services should only be maintained up and running. In such context we expect to harvest, store and migrate the real Information Space data. This release will become full production when all components prove to be stable and behave correctly, and all relevant data will be available.
- If the *first candidate release* does not prove to be stable enough and mayor updates are needed to fix the outstanding issues, a *second release candidate* 2 (*rc2*) should be released before the the final D-Net 1.0 release.

The features planned for each release stage are shown in tables Table 1, Table 2 and Table 3.

Each (pre-)release is divided into several steps:

1. The source code of all the components which participate in each (pre-)release enters in the *feature freeze* state. During the *feature freeze* stage developers are not allowed to apply changes which introduce new features or otherwise disrupt the interaction of software components in a sensible way.
2. The software is packaged (sources, binaries and configuration). The packaging shall be tested and eventually the build/packaging system shall be updated to correct eventual issues found during this stage.
3. The packaged software shall be deployed on the production infrastructure. In case of pre-releases, the existing production infrastructure shall be updated. The production deployment is already organized in such a way that an incremental software update is possible (as documented on the wiki [1]).
4. Once the code proved to behave correctly in the production infrastructure (whether the upgraded production of pre-releases or the new production infrastructure of the release candidate) the the code is enters in the *code freeze* state. During the *code freeze* stage developers are not allowed to apply any change to the code. The complete source code is branched in a *release branch.*
5. Code is checked out from the *release branch* and officially packaged and uploaded to the official package repository. Each package shall be digitally signed
6. The production infrastructure shall be upgraded with the official packages.

Moreover, in the delivery date of a release:

- We should analyze which functionality among those expected by the release plan has been fully developed and tested per Service.
- We should analyze which functionality could instead not be developed or not tested, the reasons which led to the failure and provide solutions to avoid that the next release is affected by the same problems.
- New Service software packages (fully or partially completed) which were not part of the given release will be internally released for the partners to work on a running infrastructure;
- The plan of the next (pre-)release will be modified according to the tasks still to be accomplished.

The underlying principle is that "something will go wrong" and we must cope with it without blocking the overall development process.

3.2 Development tools

The development and testing stages will be supported through the following tools:

- Tasks and bugs assignment: moving from *Savane* to *Trac*;
- Software versioning: *SVN*;
- Testing: *ICM testing suite*.
- Continuous integration: *Hudson, LuntBuild or CruiseControl*

Policies and rules to be followed under development are presented in 3.3 and 3.3.

3.3 DRIVER Node architecture and Service configuration

The whole architecture will be Java-based. The choice does not hinder external technologies, built within different platforms, to be integrated into DRIVER. On the other hand it enables a helpful and uniform way of designing, developing and deploying DRIVER Services. D-NET v1.0 will deliver common Service classes to be implemented, a set of common libraries implementing core functionalities, and the definition of a common DRIVER Node architecture.

In particular a DRIVER Node consists of Java Servlet container (for example, Apache Tomcat) running a Hosting Node Manager Service (HNM). The Service communicates information about the node status to the infrastructure Information Service; such information regards network measures as well as local performance measures.

DRIVER Services

DRIVER Services implement a special DRIVER class Service (but may also not) and run on a DRIVER Node as Servlet Container WebApps. Each Service is packaged as a WAR file, containing the Service classes, resources (property files, spring bean definitions) and dependencies (including the DRIVER Core Libraries, see Section 2.1). Finally, stub classes files are to be placed in a special directory in the WebApps (details can be found at: http://technical.wiki.driver.research-infrastructures.eu/index.php/Software_Development on ScrewDRIVER 3.3).

Such scenario is the one that requires minimal modifications to the existing Java DRIVER Services, while leaving the door open to future, more elegant, solutions. One of this would be that of having all Services running inside the same WebApp, with the Hosting Node Manager Service responsible for their life-cycle and their communications with the outside world. In such scenario Services would we packaged in a DRIVER Archive (DAR) and could be deployed on-the-fly as well as switched-off or restarted automatically by the infrastructure (through the HNM) on demand. Not only, incoming and outgoing communications could be filtered and optimized by the HNM: e.g. Services running on the same node could communicate without HTTP/SOAP protocols. Note that, due to the IOC development strategy currently adopted by the DRIVER Java developers, such changes turn out to be quite natural.

Property files

We shall consider two main phases with respect to the alignment of the Services: development and production. The difference between these emerges in the property file of Services, which in development phase may contain references to all Services it needs to refer, while in production it should only contain an indirect reference to the Information Service (probably a constant domain and a reference to a DNS; this way the Services will not even need a special Information Service address) and discover other services dynamically.

During development the property-based configuration of all Services should follow the same nomenclature in order to simplify the operation of deployment of the Services. The

structure and organization of these files can be found at: http://technical.wiki.driver.research-infrastructures.eu/index.php/Software_Development, on ScrewDRIVER 3.3.

3.4 D-NET v1.0 development plan

The plan defines the precise delivery dates, establishing also what Services will be released in the D-NET v1.0. Three main releases are expected before the production: alpha, beta and release candidate.

IN	*surely part of D-NET v1.0 and to be used by others*
IN-I (Independent)	*surely part of D-NET v1.0, not to be necessarily used by others*
OUT	*it was considered for D-NET v1.0, but left out for D-NET 2.0*
IN-O (Ongoing)	*ongoing development activity, the result is not required by others in D-NET v1.and there is no certainty that it will be delivered, but it will in D-NET v2.0*

Legenda for Table 1, Table 2, Table 3

Service	*Partner*	*Alphav1: Freezing Date 31st of March*	*Type*	*Alphav2-Beta: Freezing Date 30th of April*	*Type*	*RC: Freezing Date 15th of May*	*Type*
Enabling Layer							
Core Communication Libraries		-	-	15/04/2008		15/04/2008	**IN-I**
	CNR, NKUA	-	-	Automatic registration component	**IN-I**	-	**-**
	CNR, NKUA	-	-	IS-lookup client: no cache in D-NET v1.0, support for transparent ResultSet iteration (see NKUA code)	**IN-I**	-	**-**
	CNR, NKUA	-	-	Service locator (stub finder)	**IN-I**	-	**-**
	CNR, NKUA	-	-	IS-Registry client	**IN-I**	-	**-**
	CNR, NKUA	-	-	IS-SN client	**IN-I**	-	**-**
	ICM, NKUA	-	-	-	**-**	AAS Client	**IN-O**
	CNR, ICM	-	-	-	**-**	Node Secure communication layer: regions	**OUT**
	All	-	-	Alignment of all Java Services: packaging, common libraries, automatic registration	**IN-O**		
Hosting Node Manager Service		-	-	30/04/2008		15/05/2008	
	CNR, NKUA	-	-	HNM in Java: simple profile update	**IN-I**	-	-
	CNR	-	-			Munin-SmokePing Integration	**IN-O**
Information Service		31/03/2008		30/04/2008		15/05/2008	
	CNR	Profiles for Groups of Services and Repository Service with status	**IN**				
		-	**-**	-	-	-	-
				Service profile changes: 1) separate date of creation and date of update 2) Service status in profile	**IN-I**	Service profiles with status	**IN**
	CNR	Refinement of IS-SN in Perl	**IN**				
		-	**-**	-	-	-	-
		-	**-**	-	**-**	ıation Space Views integration: waiting for G	**OUT**
		-	**-**	-	-	Java implementation	**IN-I**
	CNR, ICM	-	-	-	-	Region profiles and management	**OUT**
	CNR	-	-	IS-Store: Java + Exist 1.2	**IN-I**	IS-Store: Java + Exist 1.2 + service profile status	**IN-I**
Manager Service		31/03/2008		30/04/2008		15/05/2008	
	CNR	Managing multiple indices and stores	**IN**				
		-	**-**	-	-	-	-
				-	-	ontrol panels and monitoring tools: re-worke	**IN**
	CNR		-	Java implementation	**IN**	-	-
Authentication and Authorization Service		-	-	-	-	15/05/2008	
	ICM	-	-	-	-	Service-2-Service secure communication	**IN-O**
ResultSet Service		-	-	-	-	15/05/2008	
	CNR	-	-	-	-	Java implementation	**IN-O**

Table 1 – Enabling Layer Execution plan

Service	Partner	Alpha v1: Freezing Date 31st of March	Type	Alpha v2 -Beta: Freezing Date 30th of April	Type	RC: Freezing Date 15th of May	Type
Data Layer							
MDStore Service		-	-	-	-	15/05/2008	
	UniBi	-	-	-	-	Java implementation - large file storage	**IN-O**
Index Service		-	-	30/04/2008		15/05/2008	
	UniBi	-	-	Full-text solution	**IN-O**		
	UniBi	-	-	Search refinement	**IN-O**		
	UniBi			Sequential scan optimization (e.g. queries for browsing purposes, Index DS behaving as MDStore DS, horizontal	**IN-O**		
	ICM, UniBi	-	-	-	-	ICM code passed to UniBi	**IN-O**
Browse Service		-	-	30/04/2008		15/05/2008	
	CNR	-	-	One level Browse: Java implementation	**IN**		
	UniBi	-	-			Multi-dimensional browse	**OUT**
Aggregator Service	UniBi	-	-	-	-	15/05/2008	
		-	-	-	-	Java v1.1.0	**IN-O**
Collection Service	CNR	-	-	-	-	15/05/2008	
		-	-	-	-	Java implementation	**IN-O**
Publisher Service	CNR	-	-	-	-	15/05/2008	
		-	-	-	-	Java implementation	**IN-O**
Search Service	NKUA	31/03/2008		-	-	-	-
	NKUA, CNR	-	-	Integrating Information Space Views: one per Index Service at the	**IN**	-	-
Validator Service	NKUA	-	-	-	-	15/05/2008	
		-	-	-	-	Service	**IN**
Repository Management Service	UniBi	-	-	-	-	15/05/2008	
		-	-	-	-	Service	**OUT**

Table 2 – Data Layer Execution plan

Service	Partner	Alpha v1: Freezing Date 31st of March	Type	Alphav2-Beta: Freezing Date 30th of April	Type	RC: Freezing Date 15th of May	Type
Functionality layer							
User Interface Service	NKUA	-	-	-	-	15/05/2008	
		-	-	-	-	Generic interface	**IN-O**
User Profile Service	NKUA	-	-	-	-	15/05/2008	
		-	-	-	-	New features	**OUT**
Community Service	NKUA	-	-	-	-	15/05/2008	
		-	-	-	-	New features	**OUT**
Recommendation Service	NKUA	-	-	-	-	15/05/2008	
		-	-	-	-	New features	**OUT**
Text Engine Service	CNR	-	-	-	-	15/05/2008	-
		-	-	-	-	Java implementation	**IN-O**

Table 3 – Functionality Layer Execution plan

References

[1] *ScrewDRIVER* Wiki, http://technical.wiki.driver.research-infrastructures.eu

[2] DRIVER Annex I - "Description of Work", Proposal no. 212147.

[3] *Munin* Project. http://munin.projects.linpro.no

[4] D7.1 Enabling Services Enhancement Specification

[5] D6.2 Supporting Tools and Databases

[6] D7.2 Overall Research Design Report