**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"**


# Overall Research Design Report

Deliverable Code: D7.2

# 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: | D7.2 |
| Deliverable title: | Overall Research Design Report |
| Contractual Date of Delivery: | 1st of February 2008 |
| Actual Date of Delivery: | 1st of May 2008 |
| Editor(s): | CNR |
| Author(s): | Paolo Manghi and Marko Mikulicic |
| Reviewer(s): | N.Manola |
| Participant(s): | |
| Workpackage: | WP7 |
| Workpackage title: | Enhancing Infrastructure Sustainability and Research Integration |
| Workpackage leader: | CNR |
| Workpackage participants: | NKUA, ICM, CNR |
| Distribution: | Public |
| Nature: | Deliverable |
| Version/Revision: | 2.0 |
| Draft/Final: | Draft |
| Total number of pages: (including cover) | |
| File name: | D7.2 |
| Key words: | Policies, rules, deployment, release, development |

# 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

**Document Description** ..... **2**
**Disclaimer** ..... **3**
**Table of Contents** ..... **4**
**Table of Figures** ..... **5**
**Summary** ..... **6**
**1 Introduction** ..... **7**
**2 Deployment of new development infrastructures** ..... **8**
**3 Software life-cycle policies** ..... **9**
3.1 VCS, build system and continuous integration ..... 9
3.2 Issue tracking policies ..... 9
3.3 API definition: contracts ..... 10
3.4 Software configuration ..... 10
3.5 Packaging ..... 11
**4 Service development coordination** ..... **12**
**5 References** ..... **13**

# Table of Figures

# Summary

This document describes policies and rules to be followed by developers in the realization, release and deployment of the DRIVER Services.

# 1 Introduction

The DRIVER infrastructure consists of a number of Services running at different DRIVER nodes, i.e. machines connected to the Internet. Such Services have been and are to be developed by the four technological partners of the DRIVER consortium: ICM, ISTI-CNR, NKUA and UniBi; nevertheless, others might be developed in the future by non-DRIVER organizations. Due to the highly distributed scenario, where several organizations might be developing at the same time Services that are to interact with each other, the overall development of the Services must be carefully organized and coordinated in order to respect the *release plan* produced in WP5. Similarly, the process of deployment of new infrastructures for development and testing reasons, as well as the process of deployment of new Services in an infrastructure, must be planned and follow special validation procedures.

This document contains the policies and rules to be followed in the development process by all partners involved, in order to end up with a uniform system, where software is packaged and released in the same way and configured according to the same methodologies.

Finally, the document describes the methodologies applied when relevant changes to the release plan are required, and explain how these might have an impact on the ongoing plan.

# 2 Deployment of new development infrastructures

Project partners and those willing to contribute with hardware, should maintain a common execution environment on the machines they provide as DRIVER nodes. In particular, the environment must support *XEN Dom0* [7] machines capable of running one or more *DomU* virtual machines, each with its own static IP address and with all ports opened in the firewall. Each virtual machine behaves as an independent hardware unit, thus supports an independent virtual running environment, i.e. a pool of threads on its own running on some established and configured hardware resources (e.g. CPU, Ram, Disk). Being virtual, DomU machines can be moved from hardware to hardware, as long as the XEN Dom0 technology is provided, and be dynamically configured to share a portion of the resources available on the physical machine. This environment eases the deployment or migration of a pool of Services from one machine to another and gives flexibility on the usage of the physical resources.

The virtual machines should obey to the following common configuration:

- execution environment. In order to guarantee the accuracy and coherency of the machines, these are installed with a common script, which can be found on the ScrewDRIVER wiki [1]. Although rarely, the script can change due to new requirements; hardware suppliers should be notified and update their machines accordingly.
- software configuration and directory layout: unless special requirements demand for a different scenario, all machines should share the same software configuration and installation. In particular, the specific software configuration instructions is to be found on the ScrewDRIVER wiki [1]; when references or instructions to some software are not present (e.g. first time installation), these should be placed on the wiki for others to use.

Development infrastructures are constituted by a number of DRIVER nodes, i.e. a XEN virtual machine. Two main rules apply:

- Development infrastructures are named according to the name of the Information Service machine running in that infrastructure;
- One Service instance shall be registered only with one development infrastructure at a time; indeed, sharing of Services between different development infrastructures should be avoided unless there are special requirements.

The allocation of the infrastructure machines to Services for various groups of partners and different aims is kept up-to-date on the ScrewDRIVER Wiki [1].

# 3 Software life-cycle policies

## 3.1 VCS, build system and continuous integration

Development activities are carried on at different partner sites. The Service software releases resulting from such activities are needed by other partners to develop their own Services, to check the APIs to interact with, internal Service implementations, and deploy further development infrastructures.

In order to ease the process of handling and reading code as well as reusing it and installing it, partners must keep their local VCS environments according to a common internal *repository layout* and *build system*. Although the reduced partners freedom may create some initial problem, the resulting benefits will help overcome most of the integration problems experienced during DRIVER-I.

We refer to the minimal unit of code (Services and libraries) in the VCS and build system as *module*. The repository layout should be constructed in such a way that each module has its own branches and tags. Each module shall keep explicit dependencies with other modules and external dependencies. External dependencies links should be explicitly versioned and handled with *Apache Ivy* [8]; the tool maintains a clean VCS repository structure by reduce wasted space[1].

A common *Ant* build system will allow automated periodic full builds (continuous integration) which will help code maintenance and development. In particular, the *trunk* of every service, i.e. the last stable running release, will be continuously built in background (combination of compilation and unit tests) and every "failed build" will be reported.

The Ant build system will provide a powerful and easy to use set of targets for managing branching and release operations, by exploiting versioned working copies using the `svn:externals` feature.

Developers are strongly discouraged from committing casual changes to trunk, while encouraged to use branches heavily for their day to day activities[2].

## 3.2 Issue tracking policies

It is crucial, for the collaborative development of the Services between them, for each partner to keep the others informed about the status of his/her activities and to notify other partners of bugs or misbehaviors relative to their code. To this aim, the partners make use of a common *Issue Tracking* software instance, to be used according to the rules described in the following.

In the project, each module shall be assigned to one *responsible person* at the partner site and all "tickets", i.e. tasks, associated to a given module will be automatically assigned to him/her. To complete the organization, each technical partner should have a local *technical coordinator* who is responsible for supervising[3] all tickets assigned to his group members, check that the issue tracker tool is used properly and that tasks are being timely handled.

---

1 Currently 914 Mb in the VCS repository is unnecessary and slows down VCS operations

2 Following the rule of thumb: "Commit early, commit often"

3 The *trac* timeline provides a powerful tool to follow daily activities.

Each developer should use the issue tracker tool to account for tasks being carried on even if they were not planned in advance. Developers receiving a ticket should not "accept" preemptively the ticket, but instead only when the underlining activity has started. The progress status of an activity should be kept up to date. Requests for new features or other kind of actions to other developers should result in a ticket, either before or shortly after direct discussions with the interested party.

## 3.3 API definition: contracts

A further important aspect is that of inter-Service communication APIs. Each Service has the important responsibility to publish an API with a specific and expected behavior. Such behavior, i.e. semantics, should be published in the form of a *contract* and should be independent from any implementation of such a contract. Consumers of Services, as well as developers willing to provide a new implementation, must be guaranteed the correctness of the contracts.

In the project, APIs are formalized by defining *Java Interface based contracts*, which will replace WSDL based contracts. Java Interface based contracts will be defined using common guidelines based on primitive types focusing on guaranteed SOAP framework portability. The contracts are intended to define the low level transport interface; while higher level interfaces shall be built on top of them. In particular, the exact semantics of every interface should be written in JavaDoc comments along with Unit Tests defining details regarding the input and output.

## 3.4 Software configuration

In order to ensure uniform installation process of the Services and of the infrastructure, Services software should follow common configuration methodologies. Software configuration falls into two categories:

- Service local configuration
- Infrastructure access configuration

During the transition phase, i.e. until Services are ported to the new common libraries which will be used to provide dynamic and auto-configured infrastructure access (e.g. Service Locator), each Service will have to maintain local configuration options containing the entry points of every other service it needs to access.

Java Services from different partners already use the *Spring IoC* [9] container framework in order to wire their internal components as well as to provide configuration points for both local configuration and infrastructure access configuration.

In order to ease the installation and maintenance of multiple instances of the same service on different machines, Java services should be able to expose their configuration settings using the Spring "PropertyPlaceholderConfigurer", which allows to override specific configuration options without needing to build a web application archive for each Service deployment, easing thus maintenance, bulk deployment and update, and guaranteed uniformity of the infrastructure-wide software deployments.[1]

[1] It should be possible to ensure that all instances of a given services are running from exactly the same code base, in order to be able to exclude that eventual erroneous and asymmetric behaviour may be ascribed to confusion during build and/or deployment.

## 3.5 Packaging

Packaging is another important aspect of the DRIVER distributed development scenario. It is important for software packages to agree on the same naming and to be uniquely and securely associated to the corresponding developer.

### Nomenclature

During the first phase of the project, while the software development has not yet been moved to a common build system, the nomenclature for the software packaging should be:

`driver-<partner>-DNet-<version>-<phase>-[<patchlevel>]`

For example: `driver-cnr-dnet-1.0-alpha1-p2`. Here `version` is the DNet software version, `phase` is “alpha”, “beta”, “rc1,rc2,...”, “final”, and `patchlevel` is “p1,p2,...”.

Patch levels apply only for trivial bug fixes or updates, for example the change of a web page title, or the correction of a spelling error.

During the second phase of the project, when the software development has been integrated in a common build system and organization, the nomenclature will be module based:

`DNet-<version>-<modulename>-<moduleversion>-<phase>-[<patchlevel>]`

The packaging nomenclature shall maintain the distinction between the module version and the main software version.

### Digital signatures

Every partner should digitally sign (using GPG/PGP) every other partner's release package. This ensures integrity of the released packages, and avoids future confusions. Private keys should not be kept on shared project's servers.

# 4 Service development coordination

The main issue of this activity is that of coordinating the work of the different packages so as to deliver the different system releases (Task T7.3 in the DoW). This activity entails the following tasks:

1. ensuring that the policies listed above are correctly followed;
2. ensuring that the accepted tickets are respected;
3. ensuring that the delivery of the release plan is respected;
4. when the release plan cannot be respected, due to delay or re-design of the Services architecture, re-organizing the work and re-formulate it into new tickets to be distributed to the different partners involved;
5. delivering a monthly Technical Report, to be published on the ScrewDRIVER wiki [1], on what of such activities have not been respected.

Among such tasks, the number 4 is the most committing. Release plans are normally expected to fail in some parts. Such scenarios are to be dealt with particular care, so as to minimize the diversion from the original goals and correctly organize the different partners for the new cooperation which might be required. This work is carried on by CNR, who is responsible to follow the development, hence the main issues, all Services in order to optimize and orchestrate their interaction, thus their development at the different partner sites.

# 5 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] D6.1 Software Release Plan

[5] D6.2 Supporting Tools and Databases

[6] D7.2 Overall Research Design Report

[7] XEN Web Site. http://www.xen.org

[8] Apache Ivy Web Site. http://ant.apache.org/ivy

[9] Spring Framework Web Site. http://www.springframework.org