mancoosi
managing software complexity

Final version of the tools for transactional package upgrades

Deliverable 3.3

Nature : Deliverable
Due date : 23.05.2011
Start date of project : 01.02.2008
Duration : 40 months

SEVENTH FRAMEWORK PROGRAMME

Information Society Technologies

Specific Targeted Research Project
Contract no.214898
Seventh Framework Programme: FP7-ICT-2007-1

A list of the authors and reviewers

Project acronym	Mancoosi
Project full title	Managing the Complexity of the Open Source Infrastructure
Project number	214898
Authors list	John Thomson <john.thomson@caixamagica.pt> André Guerreiro <andre.guerreiro@caixamagica.pt> Susana Nunes <susana.nunes@caixamagica.pt> Vasco Silva <vasco.silva@caixamagica.pt> Pedro Rodrigues <pedro.rodrigues@caixamagica.pt> Paulo Trezentos <paulo.trezentos@caixamagica.pt>
Internal review	Roberto Di Cosmo and Anne-Sophie Refloc'h
Workpackage number	WP3
Deliverable number	3
Document type	Deliverable
Version	1
Due date	23/05/2011
Actual submission date	23/05/2011
Distribution	Public
Project coordinator	Roberto Di Cosmo

Abstract

The Mancoosi project aims at solving the upgradeability problem that users of Free and Open Source Software distributions experience when trying to install, remove, or upgrade packages. The specific aim of Workpackage 3 was to study and develop transactional rollback utilities.

These tools take RPM and APT as the basis for developing improvements to the state of the art rollback utilities. A first insight into the tools and utilities was carried out and performed in Deliverable D3.1 [BTLD08]. The tools were then developed on the basis of this investigatory work. This document enhances the description of the new developments, integrations and refinements that have occurred during the last period of the Mancoosi project.

Contents

List of Figures

Chapter 1

Introduction

Mancoosi Workpackage 3 objective was to develop tools and softwares to improve the transactional upgrade problem. This was investigated in the current state of the art in Deliverable 3.1 [BTLD08] and in Technical Report 4 [TT09]. The WP3 software attempts to integrate the work performed by the simulators and failure detector in WP2 and to link it with the tools that have been modified for WP3.

WP3 efforts have resulted in the generation of complementary pieces of software:

- RPM 4.6 with rollback and DSL
- APT-RPM with DSL rollback features and DSL
- Website for the automatic generation of DSL enabled packages
- Pulse2 remote management tools for performing package rollbacks across a network
- RPM test framework for checking the improvements of RPM versions

All these complementary pieces of software attempt to improve the state of the art of transactional upgrades.

Chapter 2

RPM DSL

2.1 RPM-DSL usage guide

RPM v4.6 – Caixa Mágica fork, has been modified to take advantage of DSL properties and to add extra features for rollback.

RPM EIU: corresponds to Erase/Install/Upgrade, the typical commands used with RPM. It is used for conciseness when the feature corresponds to all the common operations.

RPM EIU "–tid": Allows the user to manually specify the transaction for which the operation should correspond to. This is limited to the last TID or the last+1 TID. This allows the user to merge rpm operations into a single transaction. It is used by APT-DSL to link multiple RPM operations together into one transaction.

RPM EIU "–rb_tid": Allows the user to specify which transaction they wish to rollback. It is used with the package operation to find the package and the corresponding transaction ID to use the DSL properties for rollback. It is used by APT-DSL to specify the rollback DSL operations to use.

If no "–tid" is provided, then RPM will automatically use the next Transaction ID for the operation. This is to ensure that when packages are upgraded, the DSL History is maintained.

`URL for the software is:` https://gforge.info.ucl.ac.be/svn/mancoosi/trunk/cms/rollback/rpmrb/

2.2 DSL usage scripts

DSLstart is a BASH script used by the rollback tools. It should be placed somewhere in the $PATH directory for the system.

A RPM has been modified with DSLtags and is installed through RPM. When it reaches a dslstart tag it launches a script passing along parameters as prepared. DSLstart uses these tags to populate a DSL History database stored in /var/lib/rpm/rollback/. There are two DBs used. The first is for maintaining the TID that can be used by RPM and APT, called rb3.sqlite. The second is rbdsl.sqlite linking the dslIDs with the Package, its EVR and the associated templates and arguments.

There is a corresponding dslstop BASH script that is just a symbolic link to the dslstart. This

is used to close off, once the associated scriptlet code has been completed. In a completely DSL enabled package this dslstart is not required, because only templates would be used.

To create the initial Databases there is also a script called dbinit.sh that creates the database and the tables with the associated scheme.

URL for the software is: https://gforge.info.ucl.ac.be/svn/mancoosi/trunk/cms/rollback/maintdslinjector/trunk/

Chapter 3

APT DSL

3.1 APT-DSL User's guide

This document outlines the use of APT-DSL explaining the rollback commands and, from a user perspective, what commands are needed to perform the common operations.

Apt-get install $PKG: installs a package. It uses the modified RPM-DSL as a basis to install the package.

Apt-get remove $PKG: removes a package. It uses the modified RPM-DSL as a basis to remove the package. It saves the configuration data related to the transaction ID for later rollbacks.

Apt-get upgrade $PKG: upgrades a package to the latest version. It uses the modified RPM-DSL as a basis to upgrade the package. It saves the configuration data related to the transaction ID for later rollbacks.

Apt-get rollback-hist: shows the history of the upgrades performed on the system through a set of transaction IDs. Each Transaction ID corresponds to a single apt command, where an operation has been performed.

Sample:

+————————————————————+

Transaction ID: 284

Date: 2011-04-21 - 12:11:48

Packages:

wine64(1:1.2-3mdv2010.1) REMOVE

wine(1:1.2-0.rc4.1mdv2010.1) INSTALL wine-doors(0.1.2-4mdv2009.0) INSTALL

+————————————————————+

For this transaction that was performed on 2011-04-21 – 12:11:48 as indicated by the time-stamp, one package was removed and two were installed. If we had to rollback this transaction what would happen is that, wine and wine-doors would be removed and wine64 would be installed.

Apt-get rollback $TID: Rolls a transaction back to the state before that transaction.

Sample: Using Transaction ID 284 above will remove two packages (wine, wine-doors) and

install one new package (wine64).

Rolling back TID 284; two packages would be removed (wine and wine-doors) and one would be installed (wine64). Performing the rollback of a transaction generates another Transaction ID. A such, we could revert this operation by getting the history of the transaction just after a rollback and undo the functions performed by the TID.

`URL for the software is:` https://gforge.info.ucl.ac.be/svn/mancoosi/trunk/cms/rollback/

Chapter 4

APT contributions (others)

4.1 APT-PBO

4.1.1 Introduction

The apt-pbo solver is a prototype meant to demonstrate the usefulness of a Pseudo-Boolean Solver applied to software upgradeability problems. Apt-pbo works by encoding the CUDF problem, as a Pseudo Boolean Optimization Problem, and calling an opb-compatible solver like Minisat+ http://minisat.se/MiniSat+.html to give a solution according to the requested optimization criteria (paranoid/trendy) defined in the rules of the MISC Competition http://www.mancoosi.org/misc/. Finally, it returns a solution in the form of a CUDF Document containing the package universe, which is the output of the proposed solution.

4.1.2 User Interface

This tool is command-line based and expects the following arguments:

```
./aptpbo cudfin=FILE cudfout=FILE criterion=(paranoid,trendy)
  cudfin : CUDF Input file
  cudfout : CUDF Output File
  criterion: One of "trendy" or "paranoid"
```

4.1.3 Technical Details

Apt-pbo is implemented as a perl script and, additionally, it needs a modification of the apt-get utility. This adds a new command to apt-get which, for a set of package installation status and package repositories, returns an opb encoding of the installation problem suitable for the actual pbo solver (minisat, wbo, etc.) The source code for apt-pbo can be obtained using the Bazaar SCM Tool issuing the following command:

```
bzr branch lp:~aguerreiro/apt/apt-pbo-misc
```

The source to the modified APT, needed to run apt-pbo, can be fetched by running:

```
bzr branch lp:~aguerreiro/apt/apt-pboinstall-misc
```

There is no included solver in apt-pbo itself but minisat+ is fully compatible and available as FOSS.

http://aptpbo.caixamagica.pt

4.2 APT-CUDF

4.2.1 Introduction

In Mancoosi, several data formats have been defined to capture the information related to installation and upgrade problems, in particular with the goal of enabling the MISC international competition in WP5. The first format of interest for us is DUDF, which is used to submit reports about an upgrade problem from the user to the distribution editors, who are in the best position to fix them; for this reason, DUDF has a general uniform structure, but contains a lot of information that is distribution specific. There is also the need to have a common format, distribution and package management systems-independent, to ebable the MISC competition, and to develop standard, interoperable solver plugins: the Common Upgradeability Description Format (CUDF). The mapping between DUDF and CUDF is performed using other tools produced in the project notably those included in the dose3 framework.

The Meta-installer present in Caixa Mágica up until version 15 is apt-rpm so our contribution to the MANCOOSI virtuous cycle was best accomplished making our package manager fully equipped and conformant with the mentioned tools and standards.

4.2.2 User Interface

A user of the apt-rpm package present in Caixa Mágica 15 has two ways to opt-in DUDF reporting: the first one is targeted for experienced Linux users, and the second one is meant to be easier for novices. The first option is to switch the APT::Dudf::Store-Report configuration option in the file /etc/apt/apt.conf.d/caixamagica.conf which is initially set to False. The second option is offered at the distribution upgrade "wizard" application level where the user is asked if he wants to opt-in to installation problem reporting.

4.2.3 Technical Details

Support for DUDF problem reporting was implemented by linking apt-get binary with an XML library (libxml2) and by adding the appropriate hooks to handle different situations, needed for this reporting (installation success, installation failure due to inconsistent RPM Database, due to failure of dependency solving, etc.). The following configuration options were also added to apt-rpm:

```
APT::Dudf::Store-Report ''true'';
APT::Dudf::Store-Report-Success ''true'';
Dir::Dudf ''/var/lib/apt/dudf'';
```

http://contribsoft.caixamagica.pt/repo/packages/cm15/apt/current

Chapter 5

DSL enabled web site

The DSL contrib website allows users to upload SRPMs from http://contribware.mancoosi.caixamagica.pt. In the background, the website uses some regular expressions, matching tools and a backend database to examine and identify DSL template matches. Once detected, the corresponding scriptlet code is removed and the DSL template is used instead. The users of this website can log in and associate different templates to the maintainer scripts, in order to try and see if the packages can be linked to templates or different DSL Instances.

The administrators of the webside can also modify the templates, in that case all associated packages will be rebuilt: this is a feature that can be quite useful when an error is detected.

As a contributor, the steps to be followed are as follows:

5.1 Using the site

Access the website http://contribware.mancoosi.caixamagica.pt and log in at the top of the page, as shown in Figure 5.1. After logging in, the user is redirected to the "Contributions" page. This page allows the user to see his contributions to SRPMs in the website so far.

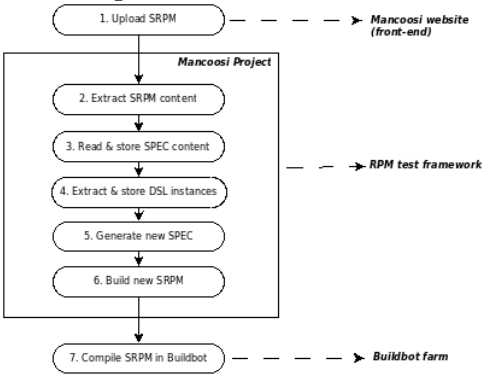
Figure 5.1: Login to website

5.2 Contributing a package

Access the menu “Contribute” and upload a SRPM, as shown in Figure 5.2: in this page the user has to select the SRPM to upload, to enter the description, if wanted (this field is optional), and to press the “Submit” button.

Figure 5.2: Upload to website

Figure 5.3: Process Flow



While the SRPM is uploading, in the background some tasks are taken as shown in Figure 5.3:

1. Check if the SRPM is a valid package, using rpmlint command:

   rpmlint -i <src-package.rpm>

2. Copy the SRPM to the shared directory /var/rpm-mancoosi/cm15/ and, with the usage of RPM query tags, obtain and store some information in a PostgreSQL database (package name, version, release, epoch, architecture, summary, description, distribution, repository, . . .):

   rpm -q –nomanifest –queryformat “%{NAME}\n%{VERSION}\n%{RELEASE}\n %{SOURCEPACKAGE}\n%{URL}\n%{GROUP}\n%{SUMMARY}\n %{DESCRIPTION}” -p <src-package.rpm>

   When stored in the database, it is necessary to set in the “release” table the field “status” to “building” and the field “active” to “1”, these are the fields that will be checked by Buildbot to see if there is a new SRPM to be built.

3. Copy the SRPM to the temporary directory /var/rpm-mancoosi/srpmsbuild/, extract the SPEC file and store its content in the database, that will be used later when rewriting the SPEC file with DSL-enabled:

   rpm2cpio <src-package.rpm> | cpio -imd

4. Call the python script template-matcher.py to parse the SPEC file and extract the DSL instances, and then try to match with the templates that are already stored in the database (see "RPM Test Framework" chapter);

5. Rewrite the SPEC file, now DSL-enabled (see "RPM Test Framework" chapter);

6. Rebuild the SRPM with the new SPEC file and replace the uploaded SRPM by this in /var/rpm-mancoosi/cm15/;

   When the upload process is finished, the user will be redirected to his "Contributions" page, where he will be able to see the status of the SRPM, as shown in Figure 5.4.

Figure 5.4: Uploaded Contributions

URL for the software is: http://projects.caixamagica.pt/projects/mancoosi/browser/contribdsl/branches

Chapter 6

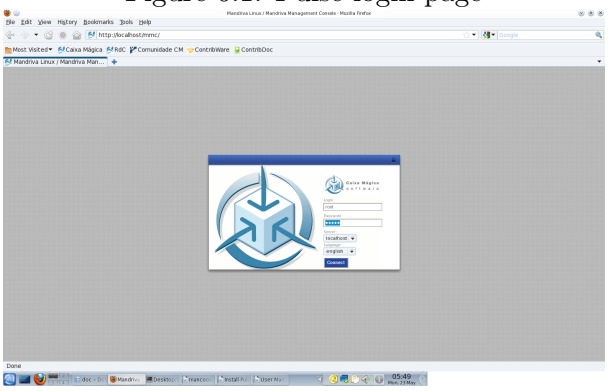
Pulse2 remote management of upgrades

Apt-Pulse Mancoosi, is an extension of Mandriva MMC – Pulse2 Project. This extensions aims at enabling the Systems Administrators to install packages remotely, to an entire network, through an easy and efficient way.

The management page is accessible from an internet browser at the following address:

http://<Server-Address>/mmc:

Figure 6.1: Pulse login page



After logging in, the following window will appear. This is the main page of MMC Mandriva/Pulse. Here we will just focus on the software deployment feature.

To access the software Deployment, the user has to follow the following steps:

1. Select Computers button on Top Bar:
2. A list of available computers will appear.
3. Select the computer on which you want to execute apt command and click on the “software Deployment” button.
4. Window 6.3 will appear. We have developed the new tabs: “Launch Apt-get” and “Launch apt-rollback”

Figure 6.2: Pulse login page

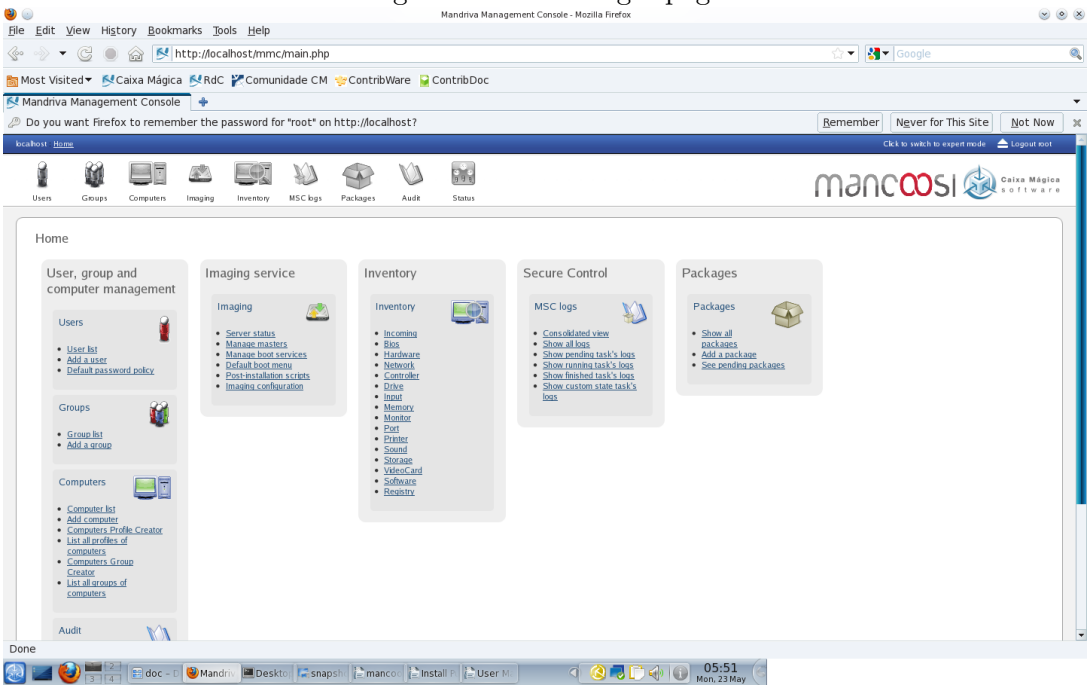
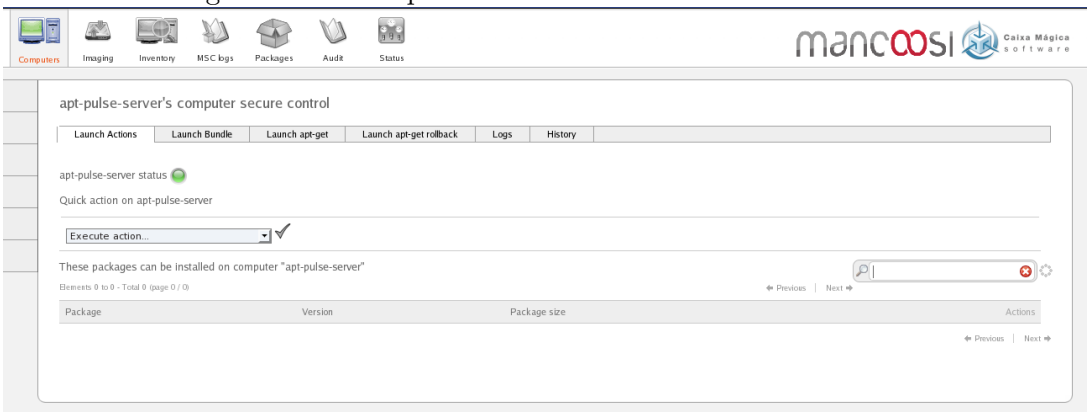


Figure 6.3: Pulse perform an action on a remote machine



In the Launch apt-get tab, the system allows the user to push the most used apt-commands to the client computer. The available commands are: “apt-get update”, “apt-get upgrade”, “apt-get dist-upgrade”, and the user can also specify the commands to install (“apt-get install”), or to remove (“apt-get remove”).

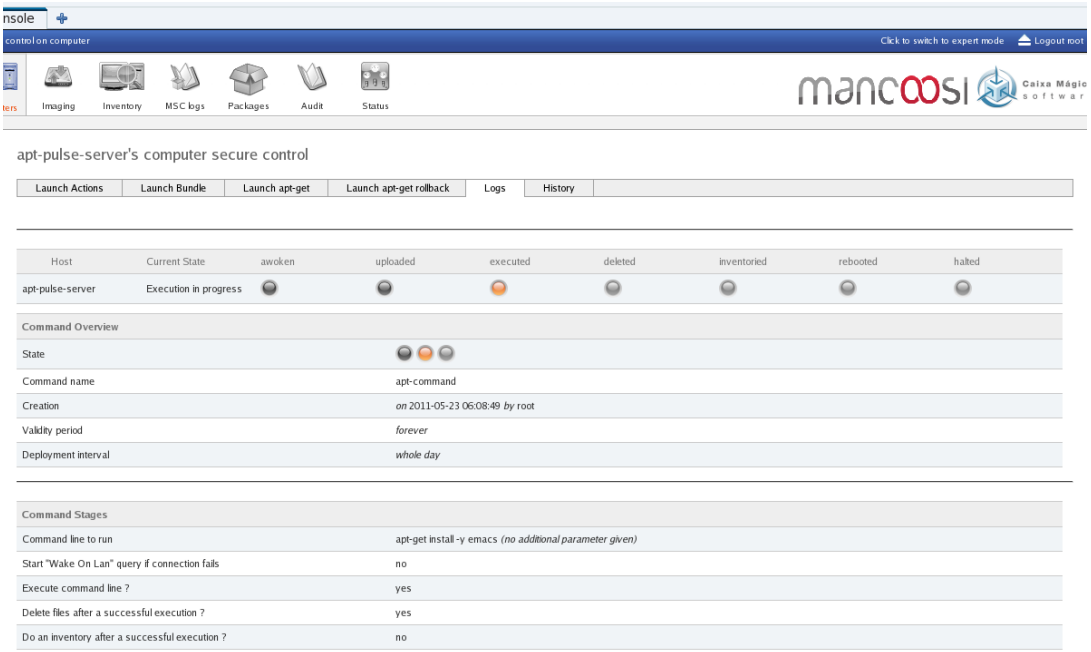
To install a package the user should select the option “I want to specify package(s) to install”, and click on “Launch Apt-get”.

A dialog box, to specify the name of the packages the user whishes to perform the installation with, will appear. Then the user has to click on the “launch” button.

After this action, the user will see the status of installation as shown in Figure 6.4:

`URL for the software is:` http://projects.caixamagica.pt/projects/mancoosi/browser/contribdsl/branches

Figure 6.4: Result of executing an action shows the current status



Chapter 7

RPM Test Framework

Ensuring that a package is correct requires different testing phases.

Unit-tests are performed at the build stage of the Package Management System, thus uncovering problems relating to the construction of a package.

To uncover package installation errors, we are currently trying to install each package in isolation and checking whether errors are reported from RPM; this process is non-standardised and depends on if the packager wishes to implement such tests.

The proposed test framework also checks the database to indicate when a failure occurred during the installation, this is not currently reported by RPM.

The RPM Test Framework implemented as a Python command-line tool, so it could be easily used in automated testing scenarios, is similar to what is normally performed in continuous integration or automated testing infrastructures. Dependencies for the script tool:

- Python version >= 2.4
- RPM (Tested with both RPM 4.6 and various versions from the rpm5.org branch)
- Python-rpm bindings (recommended but not required)

7.1 Operation

There are two major modes of operation for the script test tool: single package transactions or multiple packages transactions. They are both useful and independent test scenarios because we wanted to identify if there were any limitations on consistency, for interrupted or failed transactions, on the RPM ACID Implementation, present in the RPM5 version. The package set mode available with -package-set option was implemented in order to have a more reproducible and rigorous method of testing, keeping variables (like size of packages) from affecting the results. As a consequence, in this mode, we repeatedly test the various RPM versions, with fixed sets of packages, currently hardcoded in the actual script.

7.2 Implementation

The RPM Test Framework is implemented as follows: First, the rpm test tool tries to install or upgrade the specified packages, and after waiting a configurable amount of time, it forcefully interrupts the RPM transaction by sending a SIGKILL signal. The waiting time amount should be short enough to guarantee that the RPM process is never allowed to terminate normally. After an RPM Interruption, the tool performs a number of checks to verify the integrity of the transaction at the level of the RPM database, checking for erroneous conditions, such as multiple versions of the same package installed, and at the Filesystem level checking for files that were left over on the disk, even though the package to which they belong wasn't correctly installed.

`URL for the software is:` http://projects.caixamagica.pt/projects/pmstfw/browser/trunk

Chapter 8

Conclusion

The software from WP3, when integrated with the tools from WP2, provides an improvement of the state of the art features and allows users to:

- View their package upgrades history
- Simulate a package upgrade transaction using simulator from WP2
- Perform an out-of-order rollback
- Create new DSL enabled packages
- Manage installations and upgrades across a network

With the background tools for creating and editing DSL packages, including a community contribution site and the ability to build packages using the DSL templates by uploading SRPMs, the community will quickly be able to make use of the DSL features and perform testing before any upgrades, resulting in less inconsistent systems configurations. Ultimately, this will lead to an increase in the likelihood that a package upgrade will succeed for systems with many more packages. Also, with tools such as the test framework in the background, we are hoping to promote more reliable installers that are less prone to failures and capable to perform operations transactionally; and as such make use of the rollback tools we are promoting to return the system into a consistent state.

Bibliography

[BTLD08] Paulo Barata, Paulo Trezentos, Ines Lynce, and Davide Di Ruscio. Survey of the state of the art technologies for handling versioning, rollback and state snapshot in complex systems. Mancoosi Project deliverable D3.1, November 2008. http://www.mancoosi.org/reports/d3.1.pdf.

[TT09] John Thomson and Paulo Trezentos. Transactional upgrade rollback: the dsl approach. Technical Report 4, The Mancoosi Project, December 2009. http://www.mancoosi.org/reports/tr4.pdf.