mancoosi
managing software complexity

UPDB infrastructure to collect traces of upgradeability problems in CUDF format

Deliverable 5.3

Nature : Deliverable
Due date : 31.01.2011
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	Pietro Abate Ralf Treinen
Workpackage number	WP5
Deliverable number	3
Document type	Deliverable
Version	1
Due date	31/01/2011
Actual submission date	03/02/2011
Distribution	Public
Project coordinator	Roberto Di Cosmo

Abstract

One of the objectives of the Mancoosi project is to resolve some of the problems that users of Free and Open Source Software distributions experience when trying to install, remove, or upgrade packages installed on their machines. The specific goal of Workpackage 5 is to to build a data base of problem reports generated from such user requests to a meta-installer, which then will be used by the Mancoosi project, and the research community in general, to develop better algorithms to compute upgrade paths.

We have in earlier work defined CUDF as a common format for describing package upgrade problems in a common, vendor-independent way [TZ08], and different vendors have set up the tools and infrastructure to collect upgrade problem reports from their respective users [AGL+10]. This deliverable describes the infrastructure to build a common data base of upgrade problems at a central, project-wide repository.

Contents

List of Figures

Chapter 1

Introduction

Modern software systems are deployed in the form of a collection of components from which users may chose the software packages that they wish to install on a machine. This choice is not permanent and is subject to modification when a user applies updates, installs additional software in order to satisfy new requirements, removes unwanted software, or replaces one software component by an alternative component that provides the same functionality. In Free and Open Source Software (FOSS) collections, the set of available software components itself is rapidly changing, which adds further dynamics to the scenario.

In the FOSS world, software components are commonly called *packages*, and a collection of packages is called a *distribution*. It is the job of the *distribution editor* to create and maintain a coherent distribution. Each distribution editor chooses his format of *metadata* that describes some abstract properties of the packages in the distribution. The most important pieces of information in the package metadata are the name and version number of a package, its requirements, what it conflicts with, and the features it provides.

One of the objectives of the Mancoosi project is to resolve some of the problems that users experience when trying to modify the installation status of packages on their machine. In particular, Workpackage 4 of the Mancoosi project aims at developing better algorithms for finding solutions (in terms of versions of packages to install and remove) of such a user request. In Workpackage 5, we build a database of problem reports regarding failed attempts to modify the installation status of the packages that where produced with instrumented versions of currently popular tools. This database will then be available as data set to to researchers working on better algorithms, both in Workpackage 4 of the Mancoosi project, and researchers who do not participate in Mancoosi.

In order to build such a database we implemented the infrastructure to collect and store upgrade problems and make them accessible to the research community. The architecture of our infrastructure, as defined in Deliverable 5.1 [TZ08], is summarized in Figure 1.1:

1. The meta-installer available on user machines is instrumented so that they can generate a report of a failed package upgrade attempt. The format of the report is specific to the software distribution used, but follows a general project-wide scheme called DUDF (Distribution Upgradeability Description Format) [TZ08].
2. Problem reports are uploaded to a server specific to the software distribution.
3. Collected reports are translated by the distribution editor into a common format called CUDF (Common Upgradeability Description Format) [TZ08].

4. Translated reports in CUDF format are transferred from the distribution editor's server to a central server of the Mancoosi project, where they will be used in the construction of a project-wide problem database.

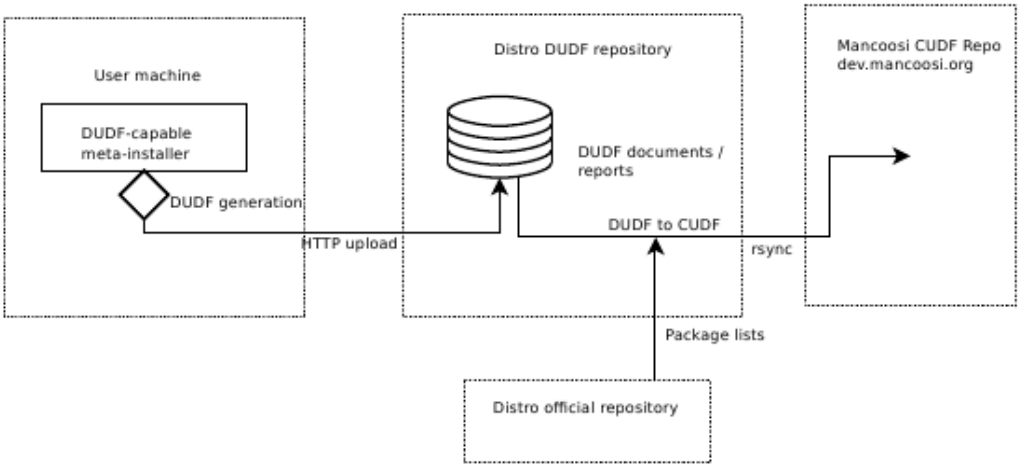

Figure 1.1: Reporting infrastructure

Deliverable 5.2 [AGL+10] described the submission infrastructure from end-users to distribution editors and the translation from DUDF to CUDF. The current deliverable describes the process of transferring CUDF documents to the project-wide server and the establishment of the problem data base.

Chapter 2

Central Cudf Database

The final point of collection for all end-user upgrade problems is the central CUDF database. This database receives documents in the CUDF format that are generated and collected by all distribution editors. CUDF documents are distribution independent per design, and therefore constitute an homogeneous corpus of problems ready to be used by the research community.

2.1 Database Backend

Figure 2.1 details the implementation of the database backend. The submission mechanism is based on the rsync service[1], which is a very common tool available on most UNIX platforms. It is efficient, and as a UNIX tool it can be easily used in scripts. We have chosen a *push* model where all distribution editors periodically send new CUDF documents to the central database. In order to avoid abuse of the service, each distribution editor is authenticated via a simple asymmetric encryption schema.

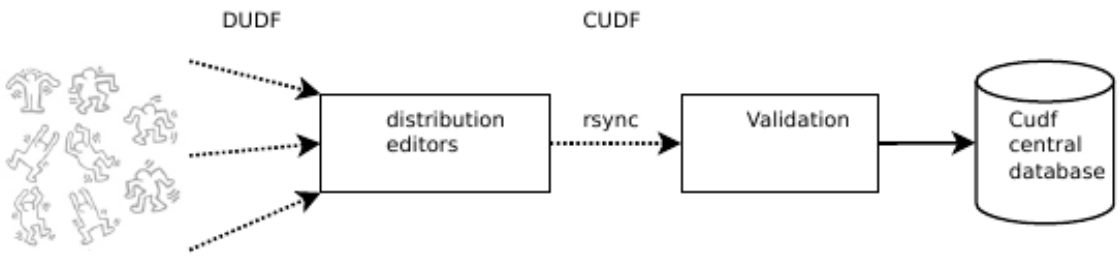

Figure 2.1: Database Backend and Validation

[1] http://rsync.samba.org/

2.1.1 Validation

The central CUDF database aims to establish a collection of difficult problems that can be used by the research community. Consequently, before selecting a CUDF problem for long term storage we perform a number of tests to check the correctness of the document and evaluating the semantic coherence.

We define coherence as the ratio of the number of broken (that is, not installable) packages to the total number of packages present in a CUDF document. Informally, a package is said to be broken if it is not possible to satisfy all it's dependencies and conflicts in an empty universe. By empirical evaluation on a large sets of CUDF documents that we have analyzed, we noticed that a coherence index greater then 0.3, that is more then 30% of broken packages, is often an indication of a problem in the CUDF document. These kind of problems, despite originating from syntactically correct CUDF, can often be easily linked to an error in the translation process from DUDF to CUDF.

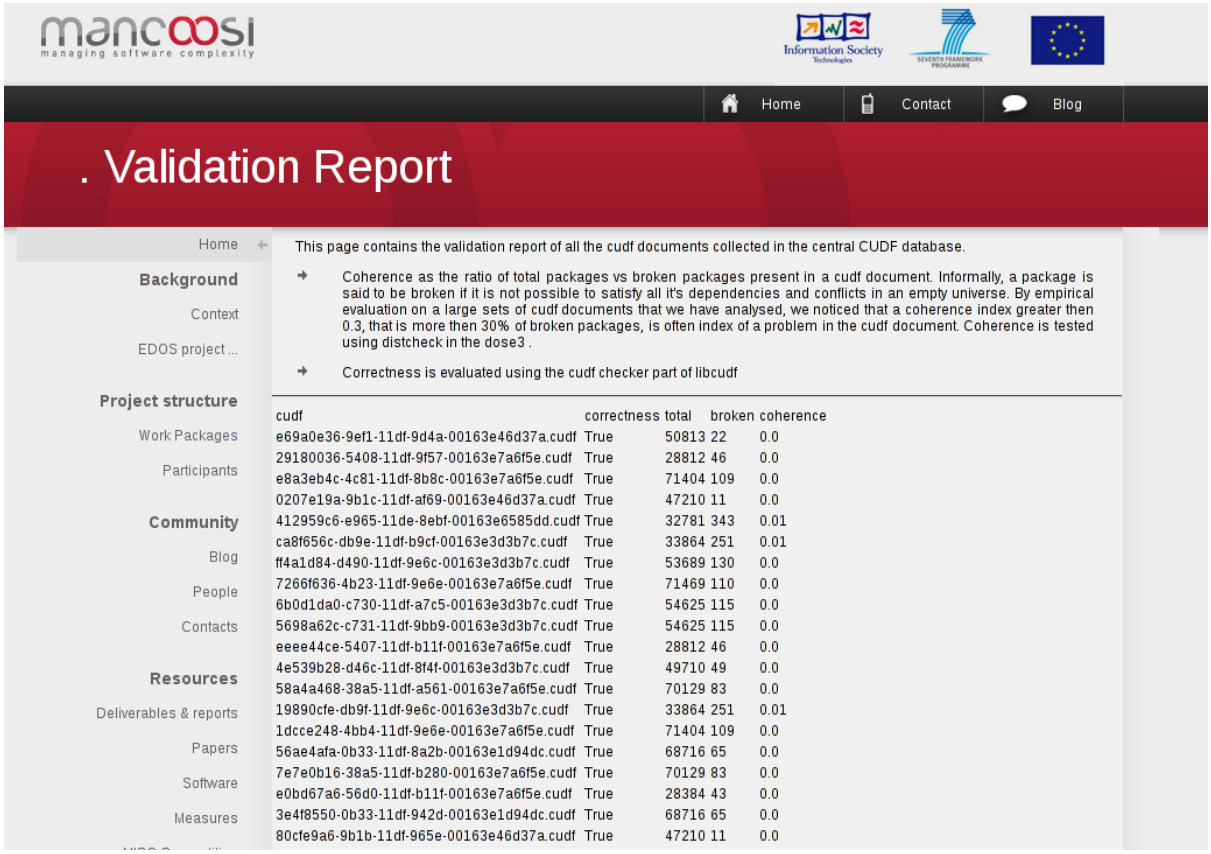

Figure 2.2: Cudf database validation report

It is important to notice that semantic coherence in this context can only be used to guess which documents are more likely to create problems. For this reason, our tests are not fully automatic, but generate a report containing the suspicious problems to be periodically reviewed by a human operator.

There are a number of tools involved in the validation process:

Correctness. Each Cudf document is checked using a syntax checker developed as part of the Cudf library. `cudf_check` takes as input a CUDF document and returns `true` if

the document is compliant with the CUDF syntax and semantic as specified in [TZ08]. Otherwise, it returns a detailed error message specifying the localization of the problem.

Coherence. Document coherence is checked using the tool `distcheck` which is part of the library `dose3`[2]. `distcheck` reads a CUDF document and returns the coherence index as described above, plus a detailed analysis of the problems encountered.

```
Cudf : debian/e69a0e36-9ef1-11df-9d4a-00163e46d37a.cudf
broken: 22
total: 50813
Coherence : 0.00
Correctness : True

Cudf : debian/29180036-5408-11df-9f57-00163e7a6f5e.cudf
broken: 46
total: 28812
Coherence : 0.00
Correctness : True
...
```

2.2 Database Frontend

Since the CUDF central database is still in a prototype phase at the time of this writing, the front end of the database is composed of a number of static HTML pages with direct link to the CUDF problems and a summary of their validation report. We also give users full access to the database to download it and mirror. The database is accessible online at http://data.mancoosi.org/cudf/db/.

2.3 State of the Data Base

As of February 3, 2011, the data base contains 434 problem reports, coming from Caixa Mágica and debian users.

[2] http://www.mancoosi.org/software/#index2h1

Bibliography

[AGL+10] Pietro Abate, André Guerreiro, Stéphane Laurière, Ralf Treinen, and Stefano Zacchiroli. Extension of an existing package manager to produce traces of ugradeability problems in CUDF format. Deliverable 5.2, The Mancoosi Project, August 2010. http://www.mancoosi.org/reports/d5.2.pdf.

[TZ08] Ralf Treinen and Stefano Zacchiroli. Description of the CUDF format. Deliverable 5.1, The Mancoosi Project, November 2008. http://www.mancoosi.org/reports/d5.1.pdf.