nuxeo

edos
www.edos-project.org

TULIP
Continuous testing of Linux distributions upgrade

Stefane Fermigier
Laurent Godard

Agenda

- Context - EDOS WP3
- Tulip Framework
- Results and future enhancements

Context - EDOS WP3

- Testing Framework and Quality assurance portal
- QA Tools for linux distributions actors
- Accessible & easy to use

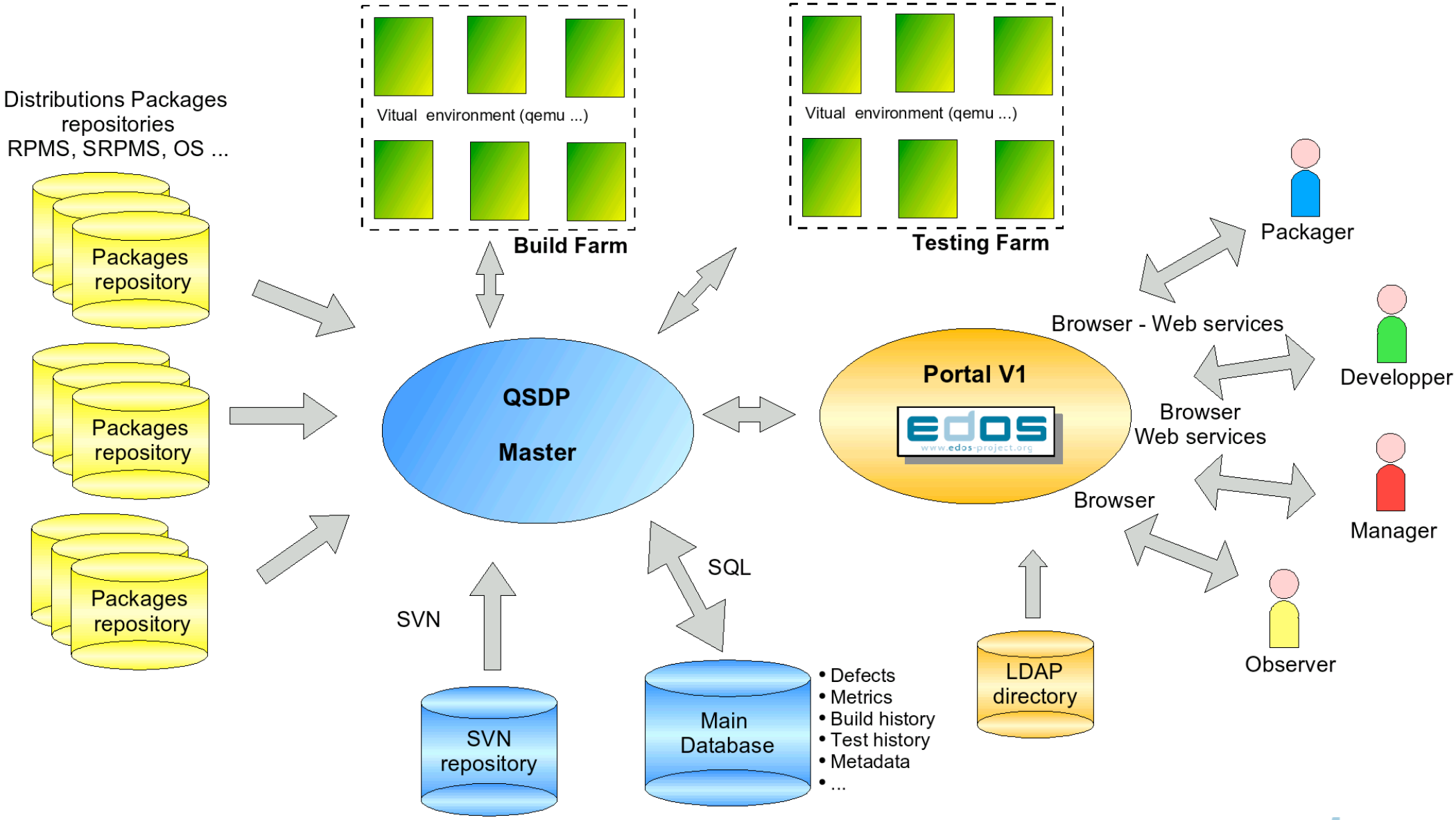
Problem to solve

- Upgrading a distribution is a risky business
  - Specially when using “unstable” branches
  - Ex: upgrading perl on Cooker usually breaks vi (!), and sometimes urpmi, as I have personally witnessed 5 times over the last 5 years
  - Users want “transparent upgrades” that work
  - Even for unstable branches
- Both a packaging and a tool (apt, urpmi...) problem (a repository replication / network issue too)

How to solve it?

- These problems can be addressed at the theoretical level (WP2)
  - Good because this is an algorithmically very complex problem
- But we need also testing tools to reasonably check that things actually work for end users
  - Must aim for rough tests (“smoke tests”) because we can’t address the whole problem space (~2^n combinations, where n = # of packages)

Software Quality Portal Schema



Portal V1

- At qa.edos-project.org
- Knowledge base of articles, software and other ressources on testing and QA
- Reports on static package analysis (Cooker)

Next steps for Portal (V2-V3)

- Create and define the testing farm
- Tools to manipulate it
  - Administration
    - New project
    - Add tests
    - Add reports
  - Consultation
    - Reports
    - Alerts

QATR – QA Test Runner

- A versatile “unit” testing framework for packages
- Prototype available
- Test the applications once installed
  - Setup/tear down environment
  - Unit tests
  - Doctests
- QA at application/package level
- Can be integrated at a higher level
- Similar to the qmtest project, still pondering if it’s sensible to base our work on qmtest or not

TULIP Framework

- **T**esting
- **U**pgrades of
- **L**inux
- **I**mages
- **P**rogram

- Drive upgrade tests of various linux distributions to ensure both fine grained QA at the package level and testing of the standard update mechanism
- Inspired by continuous integration testing frameworks (Continuum, Buildbot, Cruisecontrol)

Constraints

- Modify the system as least as possible
  - Minimize « Observer effect »

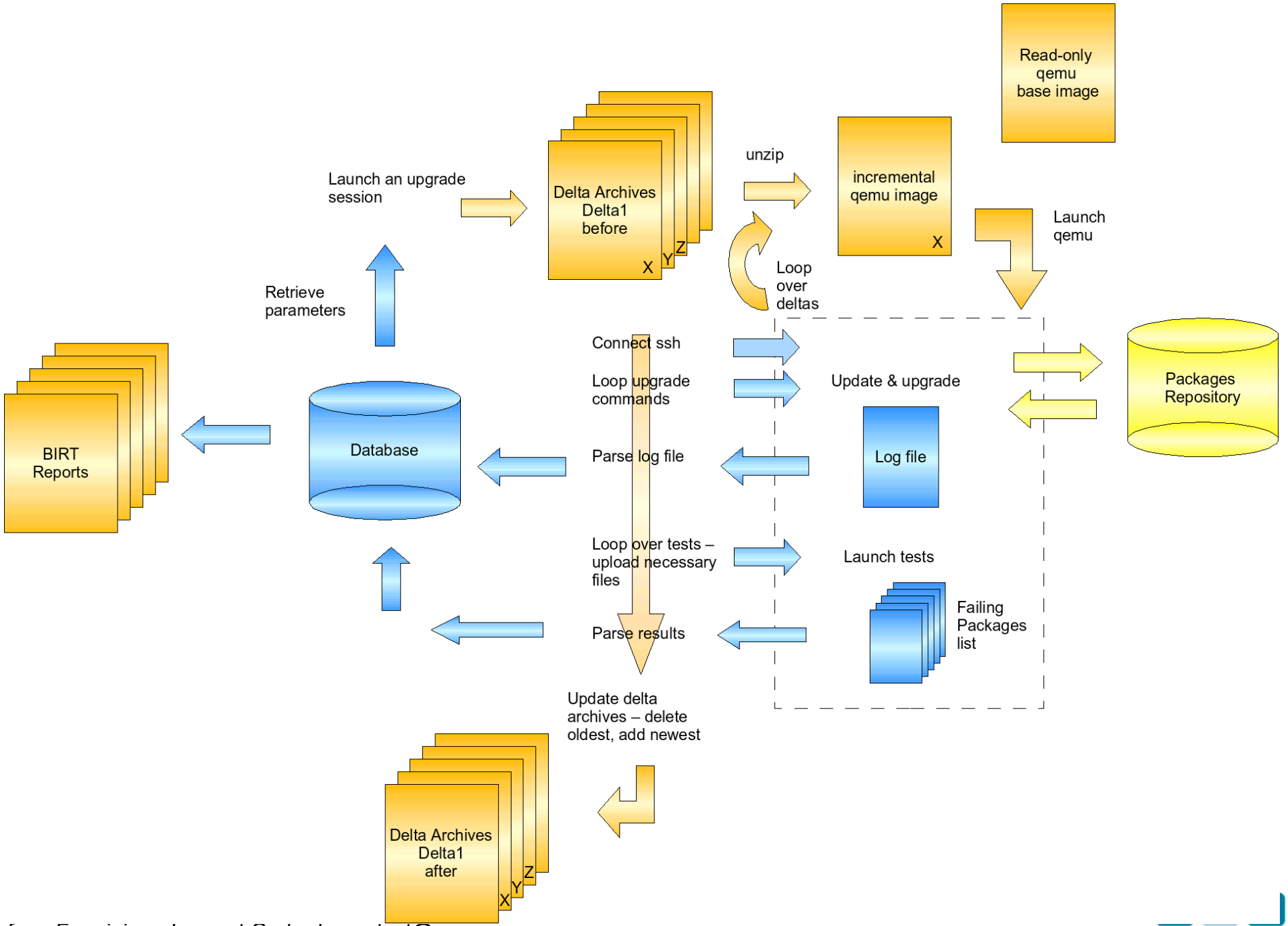
« ... instruments that by necessity alter the state of what they measure ... » source Wikipedia (http://en.wikipedia.org/wiki/Observer_effect)

- Few hypotheses on what is present/correct in the image
- Run all the projects on a daily basis max: the testing machine has to be able to respond to periodicity minima
- Storage issues

Global architecture

- Used tools
  - Qemu (incremental qemu images)
  - Python + pexpect + SQLAlchemy
  - SSH
  - Some basic console commands
  - BIRT
  - Sqlite, Postgresql at the end
- A main runner
- A project creator (local)
- A database creator (local)

Schema


Read-only
qemu
base image
Launch an upgrade
session
Delta Archives
Delta1
before
X
Y
Z
unzip
incremental
qemu image
X
Launch
qemu
Loop
over
deltas
Retrieve
parameters
Connect ssh
Loop upgrade
commands
Update & upgrade
Packages
Repository
BIRT
Reports
Database
Parse log file
Log file
Loop over tests –
upload necessary
files
Launch tests
Failing
Packages
list
Parse results
Update delta
archives – delete
oldest, add newest
Delta Archives
Delta1
after
X
Y
Z

Prerequisites on image

- Ssh and root access allowed
- Silent/non-interactive mode of upgrade
- Python
- Size problem solved with incremental images of qemu
  - Compression ratio ~ 1:4

Define a project

- Name
- Installer type
- Periodicity (daily, weekly, monthly)
- Paths (linux image, storing area)
- Ssh connection parameters
- 2 projects under test so far:
  - Debian Etch (= testing) Gnome desktop
  - Mandriva 2006 Community KDE desktop

Run a session

- Unzip qemu image of the last session
- Launch qemu image (or equivalent) (with parameters like tcp redirection or memory size)
- Connect as root through ssh
- Loop over update and upgrade commands
- Retrieve result (log file)
- Loop over tests
  - Download and run tests
  - Upload tests results
- Shutdown image

Delta Sessions

- Depending on periodicity, N past images are kept
  - Daily 6 images
  - The Delta 1 is the current (most recent) image
- Moved down in the hierarchy at each session: the new upgraded image becomes Delta 1 and the other decreased by one.
- Allow more large steps in upgrading where more changes occur on mirrors at a time

Running post-install tests

- Verify that all (or targeted) packages/software are operational after the upgrade
- Binary dependancy (script using ldd), explicit QA scripts (QATR) ...
- Synopsis
  - All in a main loop
  - Tests files are uploaded (engine if necessary)
  - Tests return a list of faulty packages
  - List retreived back to the session-pilot and parsed in database
- Caution: may be time-consuming

Collected informations

- Depending on installer and distribution
- Upgraded Package name
- Version
- Previous version
- Success/failure/name of the test
- Global execution time
- ...

BIRT reports

- Eclipse tool
- Create reports bound to a data source
- Graphical and list representation depending on parameters
- Output as PDF, HTML or displayed using integrated viewer
- Built-in parameters selections

Session Report

Parameter

Parameters marked with * are required.

Parameters

The session ID: *

195 - debian - Wed Jun 28 2006 14:38:39

BIRT Report Viewer

BIRT Report Viewer

Showing page 1 of 1

TULIP Project Activity - Session re

Session Reference 195 — **Date** 28-jui

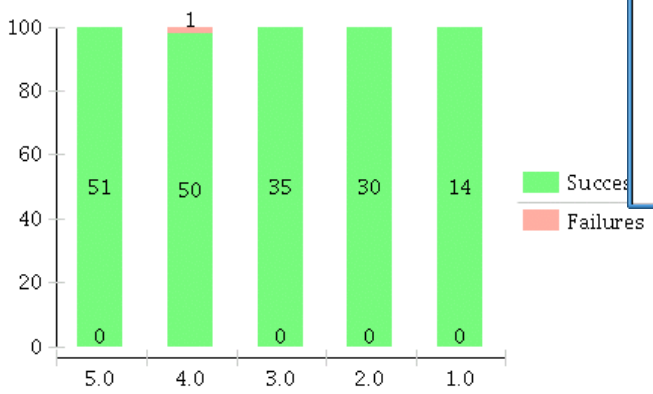
Project Name debian — **Project Id** 11 — **Periodi** daily

Number of packages 14 — **Nur** 0

Processed Packages List for the session (delta

hicolor-icon-theme modutils grub nano mount libsys
gnome-mime-data bsdutils libc6 gzip util-linux locale

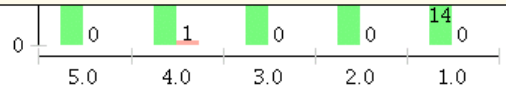
Images Delta Upgrades results



BIRT Report Viewer

BIRT Report Viewer

Showing page 1 of 1 — Go to page:



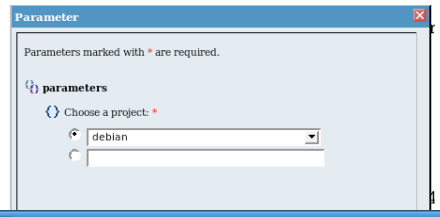
Delta Upgrades errors

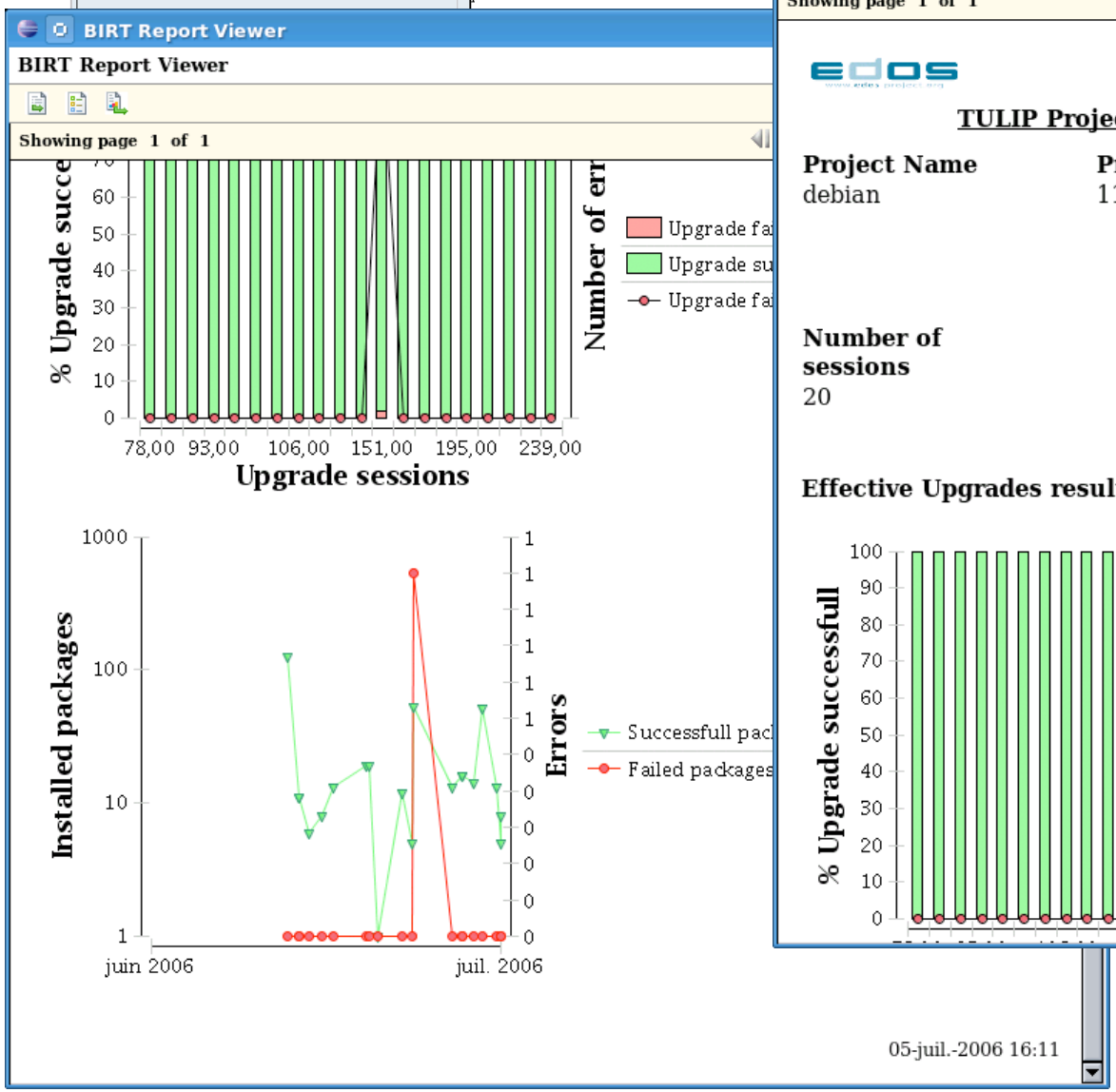
Delta Rank	Package name or error	Session ID
4	error-dpkg	198

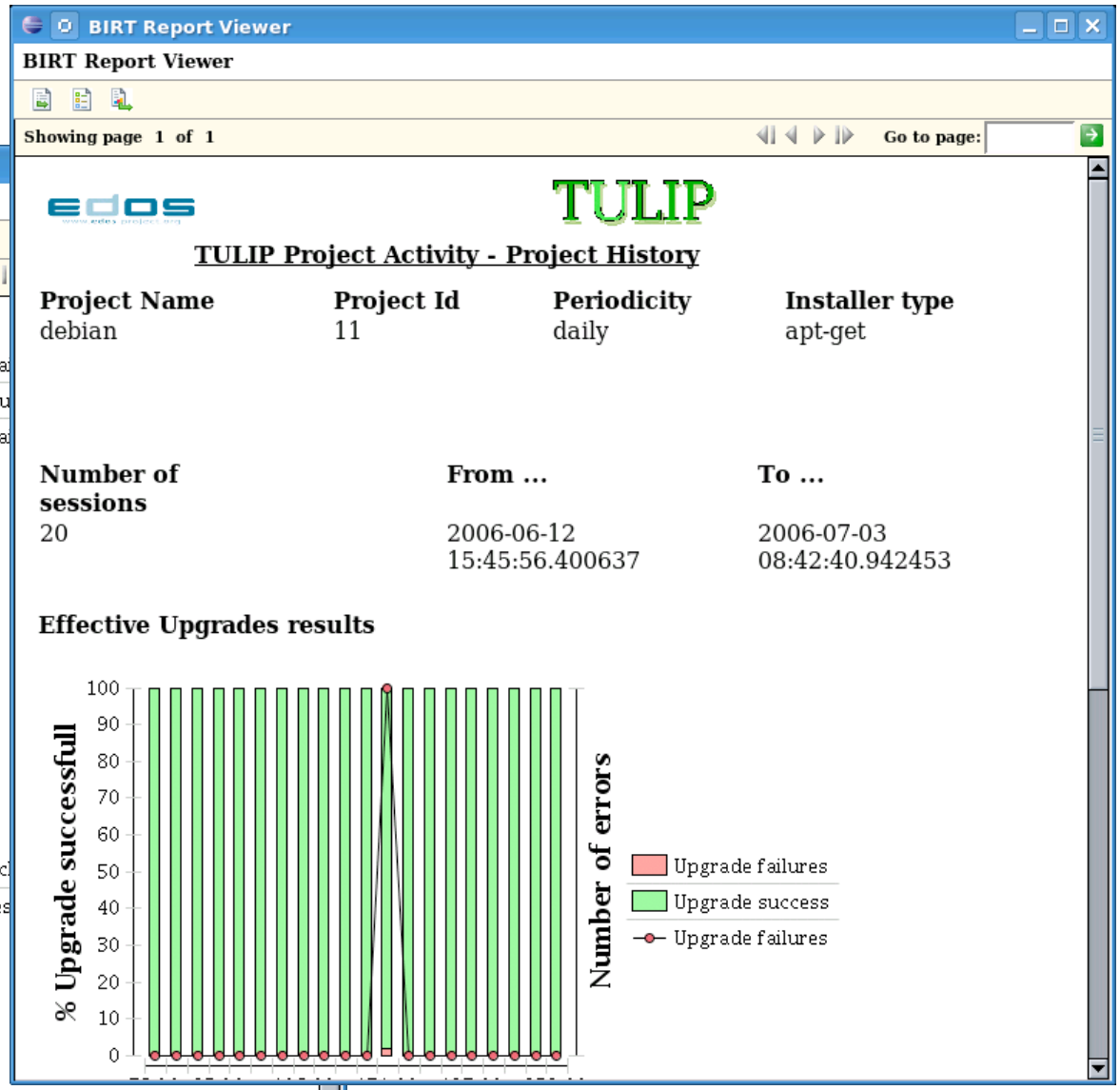
Detail of installed packages

detail_id	log_id	delta_rank	Package Name	Status	version
505	195	1	hicolor-icon-theme	ok	0.8-4
506	195	1	modutils	ok	2.4.27.0-6
507	195	1	grub	ok	0.97-11
508	195	1	nano	ok	1.3.11-3
509	195	1	mount	ok	2.12r-10
510	195	1	libsysfs2	ok	2.0.0-7
511	195	1	libc6-dev	ok	2.3.6-15
512	195	1	libc6-amd64	ok	2.3.6-15
513	195	1	gnome-mime-data	ok	2.4.2-2
514	195	1	bsdutils	ok	1:2.12r-10
515	195	1	libc6	ok	2.3.6-15
516	195	1	gzip	ok	1.3.5-14
517	195	1	util-linux	ok	2.12r-10
518	195	1	locales	ok	2.3.6-15
519	196	2	debconf	ok	1.5.2
520	196	2	hicolor-icon-theme	ok	0.8-4
521	196	2	modutils	ok	2.4.27.0-6
522	196	2	grub	ok	0.97-11
523	196	2	libgnomeui-common	ok	2.14.1-2
524	196	2	console-data	ok	20060609
612	198	4	libc6-dev	ok	2.3.6-15
613	198	4	libc6-amd64	ok	2.3.6-15
614	198	4	error-dpkg	FAIL	dpkg --configure -a
615	198	4	gnome-mime-data	ok	2.4.2-2
616	198	4	bsdutils	ok	1:2.12r-10
617	198	4	libc6	ok	2.3.6-15
618	198	4	util-linux	ok	2.12r-10

Project History Report







Some figures

- Typical daily project size: 8 Gb
  - (6 Gb base + 2 Gb incremental images zipped)
- Duration: ~ 2 hours
- Number of daily upgraded packages: 0-10
- Number on weekly basis (Delta 6): 10-50

First results

- Still under development
- No package errors detected on “stable” distributions testing
  - On upgrade
  - Regarding binary dependancy checking
- Urpmi log file is too weak (more a screen dump than a log) – Hard to parse reliably
- Some connection issues regarding --curl default of urpmi
  - Use of --wget switch recently

To do

- Run on “unstable” distribution branches (Cooker, Debian Unstable)
- Integrate to portal
  - Create projects, launch manually, add or see reports ...
- Re-work urpmi analysis as log file is not reliable: first simulate then upgrade and finally compare. Is a generalization useful ?

To do (II)

- Integrate “unit” tests (QATR, others)
- Other distributions and other installers
  - Ready (using urpmi or apt-get) Ubuntu, Kubuntu
  - yast, yum: only need a parser for installed packages and command lines

Planned enhancements

- Projects
  - More installation profiles (desktop/server, stable/unstable)
  - Test upgrades from stable to unstable/testing
- More reports
  - Follow a package through time
- Allow pre-update & pre-upgrade scripts
  - Change sources, add new packages
  - Repair a broken image
- Alerts
  - Email, RSS feed, Jabber, Nabaztag...

Possibilities

- Collect more information
  - Hardware stress statistics
- Test other categories with new projects
  - Mirrors availability/out-of-sync
  - A reference repository and verify selected mirrors are providing the same results
- Even more reports

Conclusion on TULIP

- Framework for testing upgrade of various linux images
- Incremental delta sessions increase possibilities coverage
- Can be enhanced easily
  - More installers (and distributions)
  - More install typologies
  - More tests
  - More reports

Perspectives for Portal V3

- Alerts
- The testing farm is on the road
  - Validate collected metadata
  - Define reports and metrics
- A project manager for piloting the testing farm
  - Interactive job definition and scheduling
- A interactive report manager for adding and calling new BIRT reports

Credits

- Tulip inspired by prior work by Nexedi (Umigumi/ Umitester) and Caixa Magica (eqatool)
- Qatr inspired by qmtest (from CodeSourcery) and the unit-testing movement (Cunningham, Beck...)
- Discussion with other EDOS projects members
- Contributors at Nuxeo: Laurent Godard, Stefane Fermigier, Benoit Delbosc, Tarek Ziadé, Olivier Grisel...