Adapting OpenSource package distribution to HP business

Dominique Dumont
Hewlett-Packard

French Perl Workshop 2009

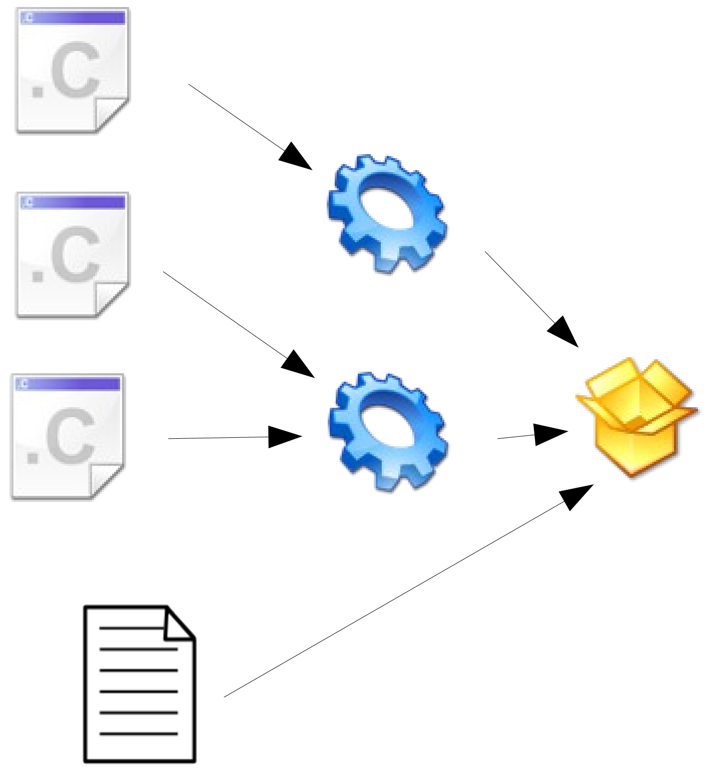
Linux package distribution for HP business

Agenda

- What is a package ?
- How the user can find good packages ?
- OSS example: Debian's distribution
- What is missing for business purpose ?
- Business package life cycle
- Tracking product version
- OpenCall package infrastructure

What is a software package ?

A way for automatic install, upgrade and removal



A package is :

- software
- instructions for Linux package manager:
  - where to install files
  - how to start service
  - how to upgrade
  - what to remove
  - special instructions for special cases: shell scripts, source of many problems

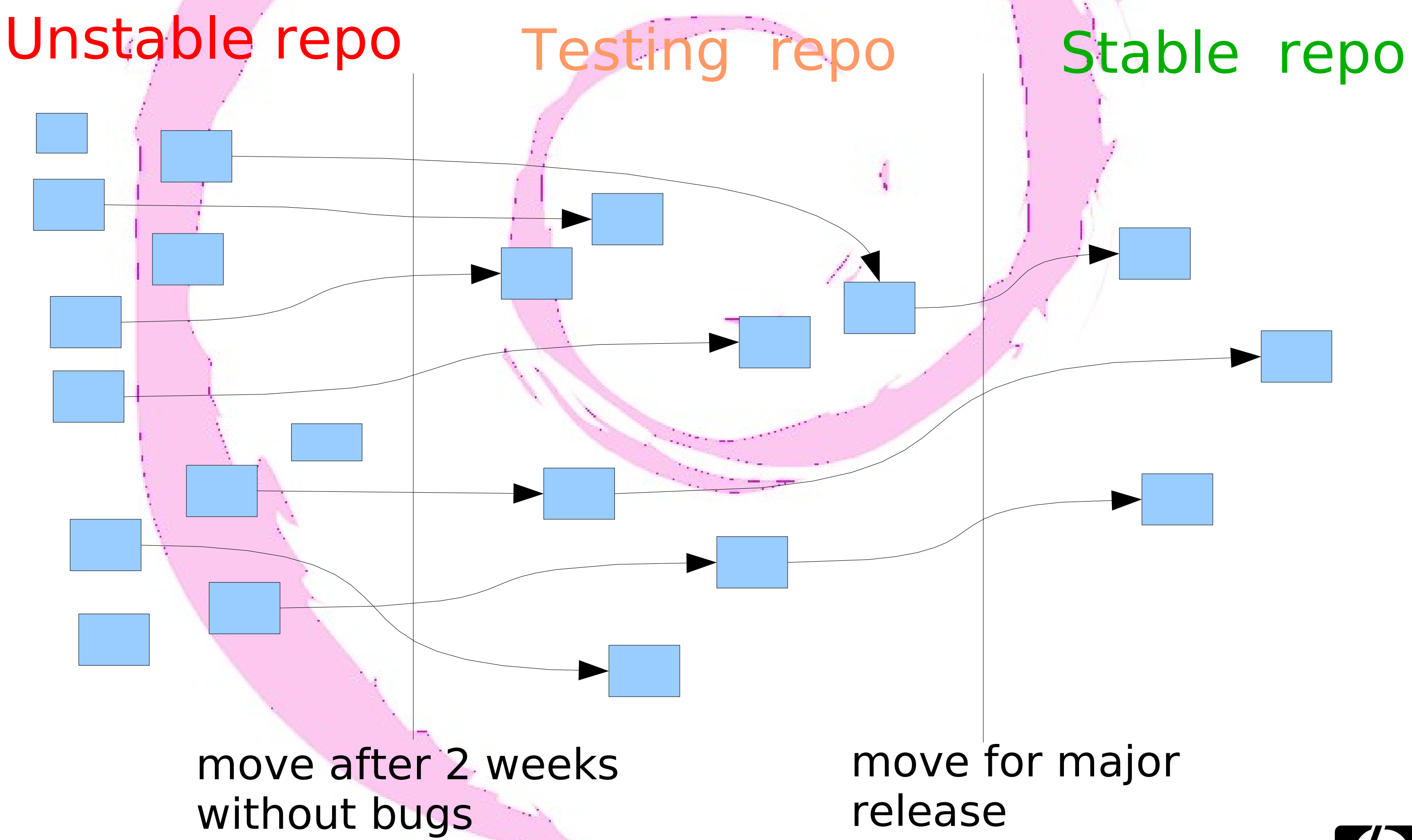
How OSS user can find good packages ?

Organize repositories in quality levels

- Development can generate lot of packages
- Most of them are bad (that's why we test them)
- Advanced users or testers will use and test most of them
- Bad packages will be discarded
- Good packages are moved to “stable” repositories
- User will get good packages from these repositories
- User will get *latest* packages

Package distribution: Debian example

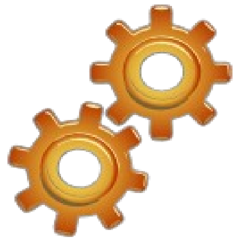
3 quality levels: unstable, testing and stable



What is missing for business purpose ?

Track the package versions that make a product version

- OSS users are satisfied with latest available version
- Business want the product version that was *qualified*
- Cannot create a repository per qualified version:
  - too many versions
  - duplication of common packages
- Cannot use a meta-package: can't patch product
- Product versions must be tracked as a set of packages/versions



Package life cycle

From integ to qa_ready to pre_release to release to obsolete

From our practices, several stage are defined

- integ: distributable within lab
- qa_ready: ready for serious QA tests
- pre-release: tests ok, can be shipped to customers for early access
- release: Send to customers
- obsolete: end of product life, will be deleted

i n v e n t

Tracking product version

with package set

- A package set is a group of packages that defines the software part of a product. I.e. a list of packages with specific version (may include source package)
- Package set also have a life cycle
- Package set are managed outside of the native packaging system (E.g. rpm)
- Must provide tools to check installed packages versus package set
- Source packages are “tied” to binary packages

Package set properties

Package set have quality levels

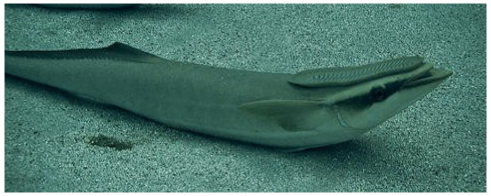
- No dependencies between package sets
- Package set content is stored in a database (the list, not the packages)
- Package set content does not change:
  - Evolution is done by creating a new version of a package set
- Package set for *Product* are required.

Source package

Managed almost like binary package

Source package in products:

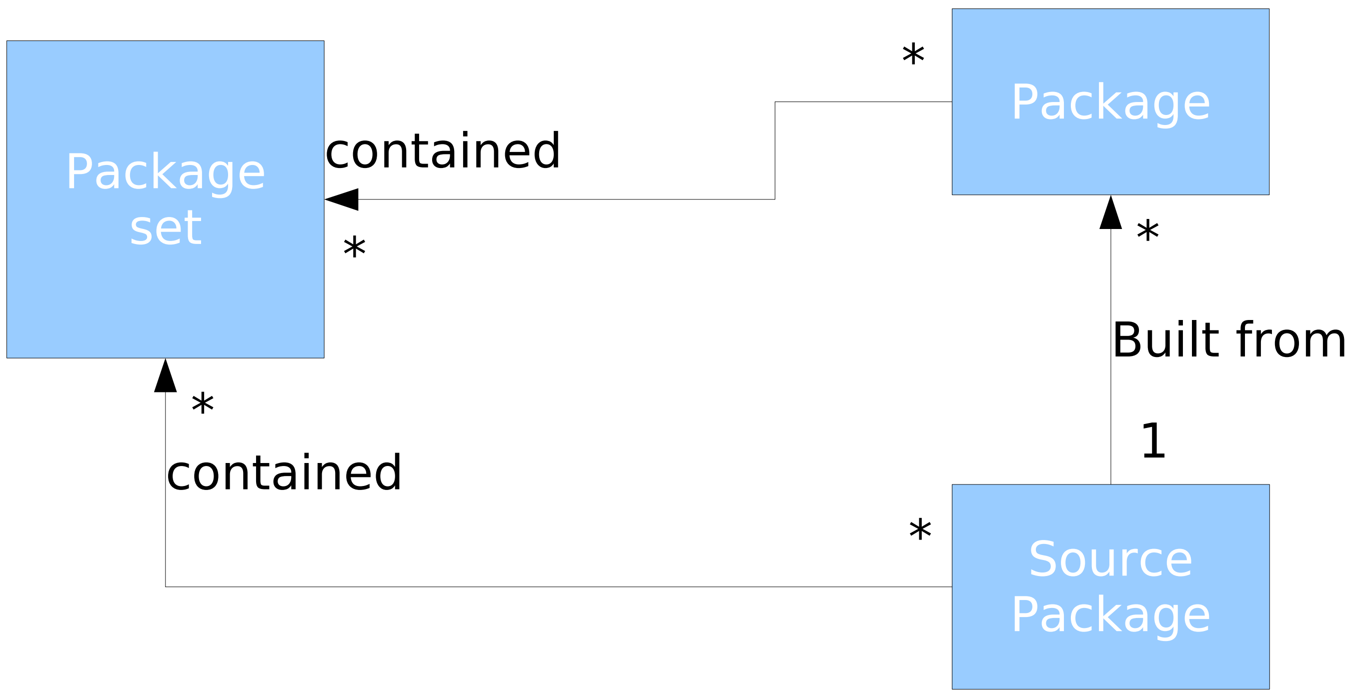
- To deliver kernel modules
- To satisfy license requirements



Manage source package:

- Package set can include source package
- Source package are “tied” to its binary package(s) (built from the source package)
- Follow promotion of “tied” package

Package and package set relations



Package and source package follow the life cycle of the package set

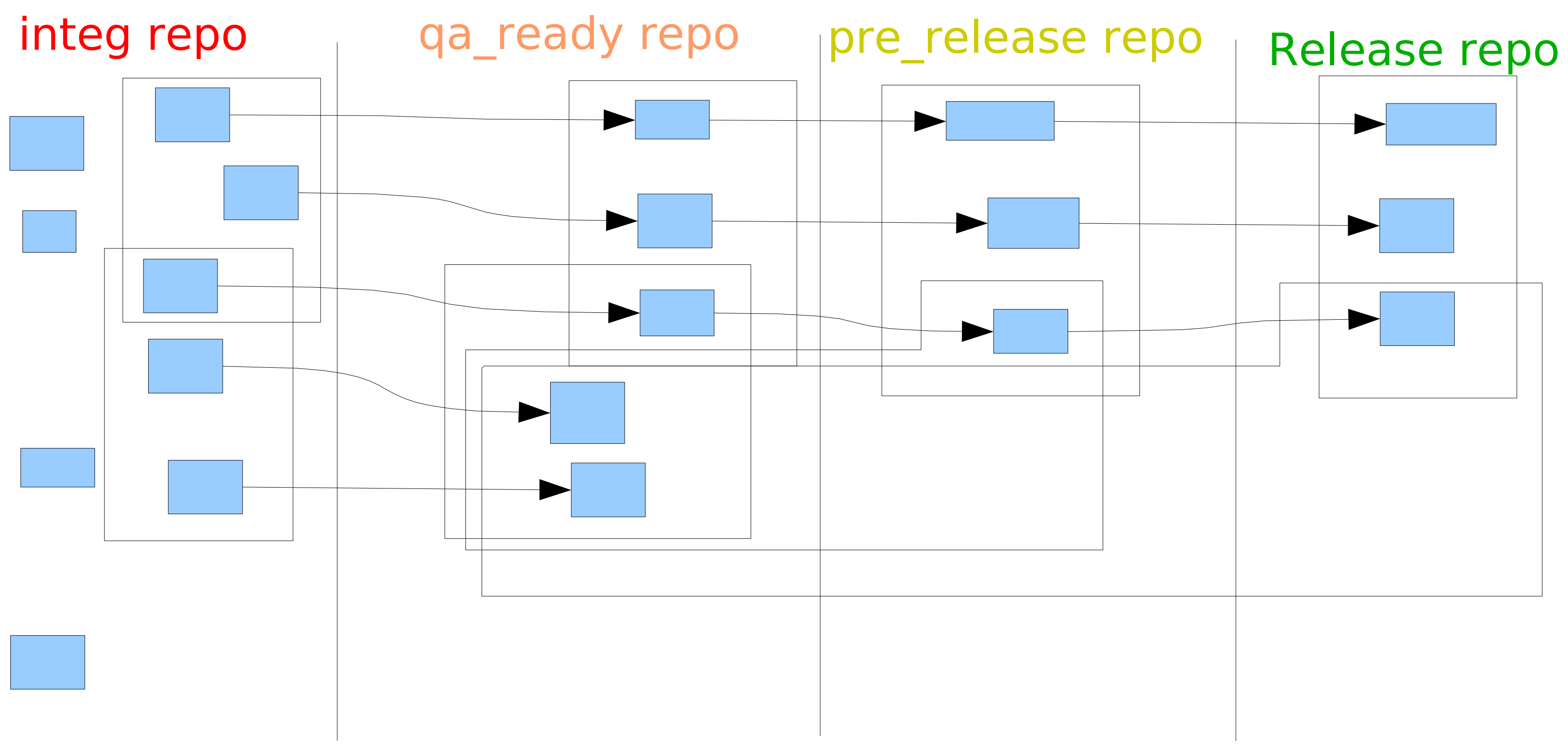
Package promotion

Efficient life cycle management

- Only package set are promoted by user
- Promotion level depend on user capabilities
- Individual package are promoted with their package set
- Source package are promoted:
  - With their package set
  - Or with their binary package(s)

Package distribution: sort by quality

1 repository per quality level



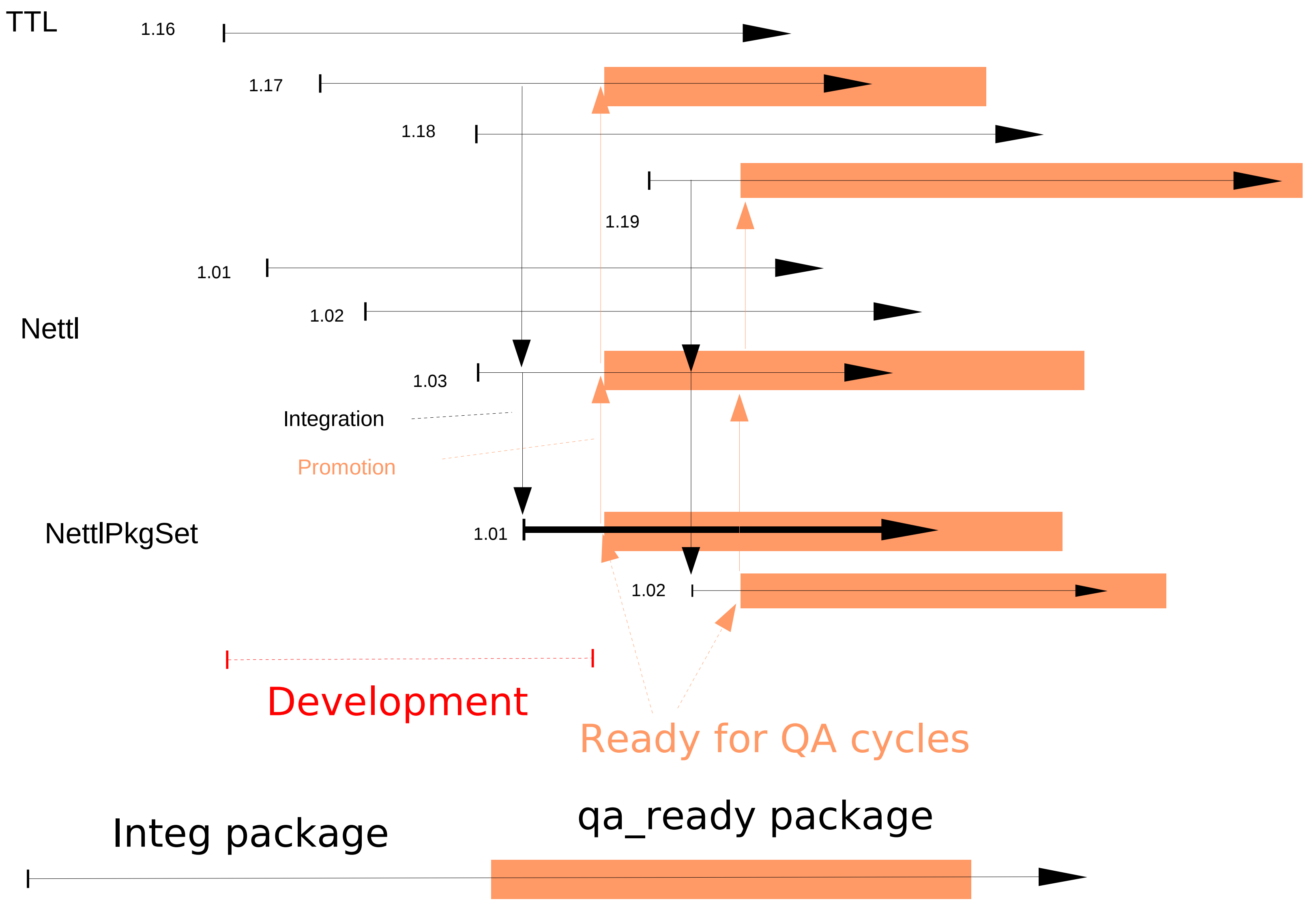
promoted on integ team decision

Promoted by QA

Promoted on checkpoint C

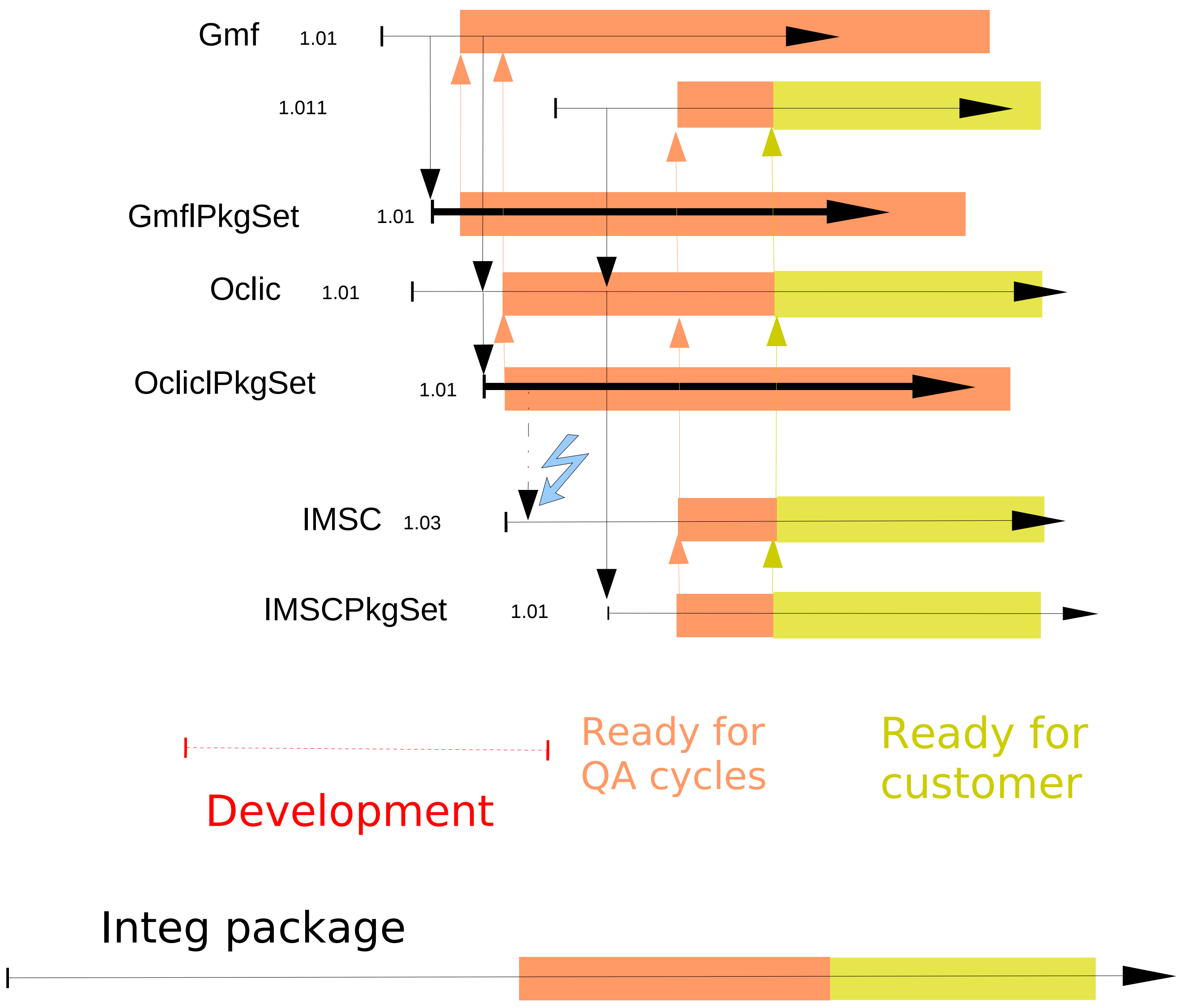
Normal use case

Development – Delivery



Use case – Integration

Development – Delivery – again – Promotion



Roles and trigger

When to move package set ? Who will move them ?

- Packages are promoted by promoting a package set

Level	Who decides	When
Integ	Integ team	Package usable by other integration teams
qa_ready	Integ team	Package are good enough for QA tests (bits ready)
pre_release	QA or release mgr	QA functional tests succesful
release	release mgr	All QA tests done, product released to customers
obsolete	CPE or release mgr	End of support life

Note: Promotion to last levels must not be done by development team

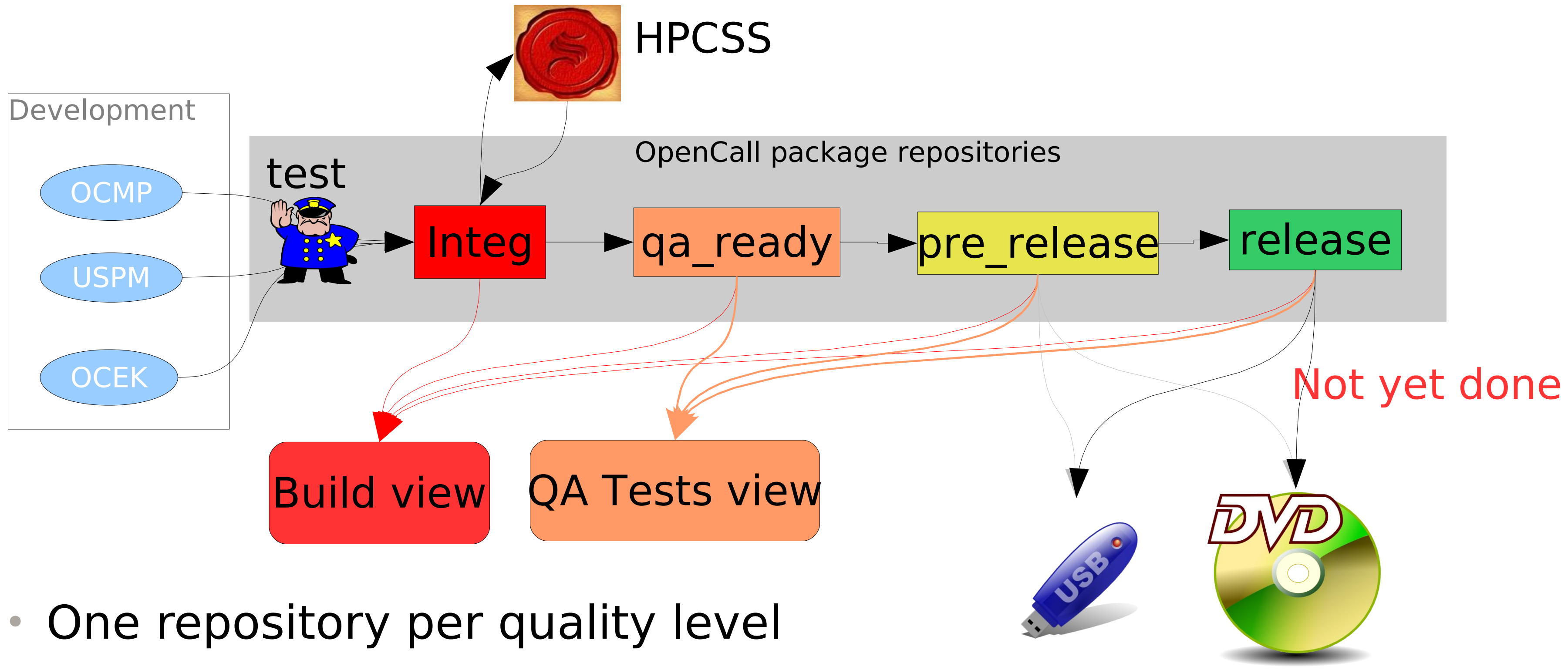
Cleanup

What goes in must go out

- Redundant packages set are deleted
- Too old packages set are deleted
- Package set in state “release” are not deleted
- Likewise for *orphan* packages
- Cleanup is done in 2 phases:
  - Warning sent by mail
  - Deletion 2 weeks later

OpenCall package infrastructure

Streamlined process



- One repository per quality level
- Build takes pkg from all 4 levels
- QA takes pkg from only 3
- Useless with low-quality packages

- Packages are signed on the fly

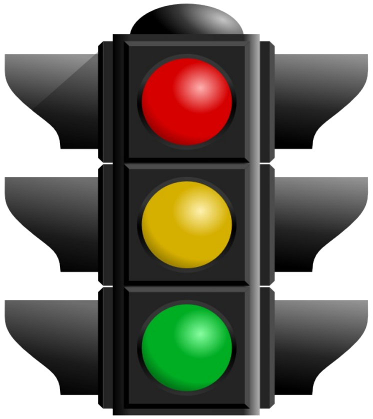
Systematic package tests

Every package is tested on entry:

- Rpmlint tests
- Custom tests to enforce our policies

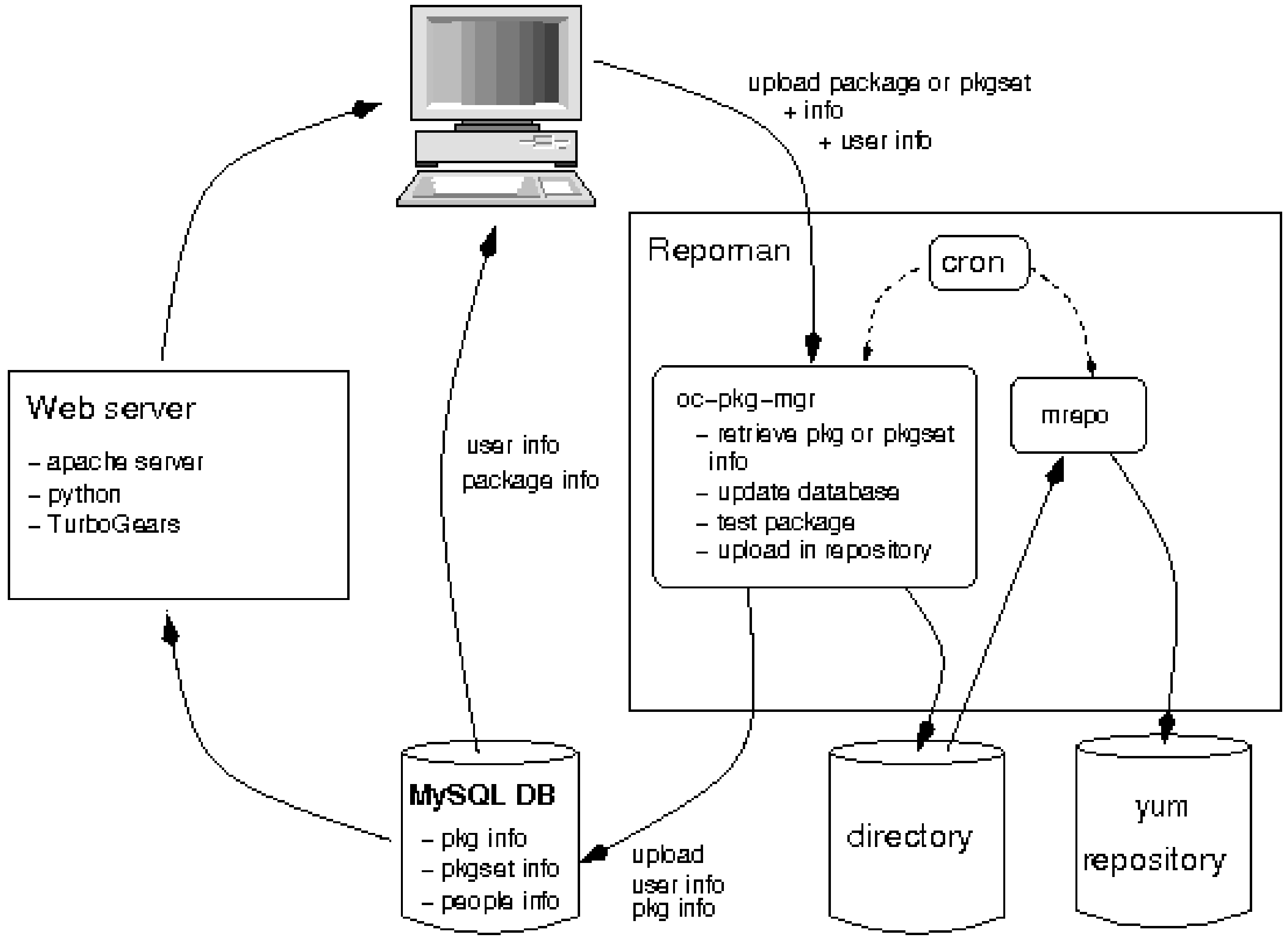
Business mandates to handle legacy:

- Framework for exceptions:
  - By package or by file
  - Time or version number limit



Implementation

MySQL + Perl + Apache + Python



Implementation

Several tools as command lines

- Oc-pkg-upload: upload packages
- Oc-pkgset-upload: scan dependencies, and upload list
- Oc-pkgset-promote: manage package set life cycle
- Oc-pkgset-install: install or download package set

Package infra advantages

Facilitate teamwork and re-use

- Integration teams are able to choose already “proven” components by picking the versions that were promoted: risk assessment is easier
- Re-use of common component in the organization is facilitated by sharing information regarding the achieved quality levels of packages
- Teams are able to monitor the usage of the components they deliver
- Source package for OSS components are handled and delivered automatically with their binary packages