gLite Patch Release Procedure

Felix Ehm
IT-GD
White Area Lectures
What do we have?

- the gLite Middleware **software**
- components (=**metapackages**)
- RPM packages grouped into metapackages
- **bugs** in parts of the software
- **fixes** for these bugs provided as new RPMs
- **patches** introducing fixed RPMs to the Integration Procedure.
• How does the bug fix go to production?

• Well..
By the way…

- **What is a repository?**
  - Directory with RPM files + some metadata for yum

- **How can I create one?**
  - Create a directory, e.g.:
  - Copy your RPMs
  - Run `createrepo`
  - Result:

  ```
  mkdir RPMS.release
  cp *.rpm RPMS.release/
  createrepo RPMS.release
  ```

  ```
  drwxr-xr-x  2 23834 root 22528 Dec  2 14:42 RPMS.release
  drwxr-xr-x  3 23834 root  2048 Dec  4 11:34 repodata
  ```

  ```
  -rw-r--r-- 1 23834 root 291351 Dec  2 14:46 filelists.xml.gz
  -rw-r--r-- 1 23834 root  5576 Dec  2 14:46 other.xml.gz
  -rw-r--r-- 1 23834 root 313086 Dec  2 14:46 primary.xml.gz
  -rw-r--r-- 1 23834 root  951 Dec  2 14:46 repomd.xml
  ```
Integrator Work I

- **Integrator**
  - Looks for patches in „Ready for Integration“
  - Checks patch information
    - Metapackage changes field
      - *Only used for add/removals of packages from/to metapackage*
      - *Strict format, e.g. :*
        - \[ \text{R|glite-UI|glite-release||} \]
        - \[ \text{R|ALL|obsolete-package||} \]
        - \[ \text{A|ALL|a-new-package|3.1.0-0|} \]
    - RPM name(s) field
      - *Only valid rpm file names must be given (e.g. glite-release-3.1.-0.noarch.rpm)*
      - *If RPMs are external, specify additionally the download url in the external packages field*
    - Subject
      - *Must contain x86_64 in case of a 64 Bit patch*
• **Integrator**
  - Provides a RPM repository for the Certifier
    - Extracting metapackage changes from Savannah
    - Extracting RPM names from Savannah
    - Checks if patches are available in ETICS/External repository
    - Copies RPMs to new patch location
    - Creates a `cert-glite-*` rpm in case of metapackage changes
      - *Prod-lists are basis*
    - Transforms location into a repository
    - Sets the Patch state to “Ready for Certification”
Certifiers work

- **Certifier**
  - Starts a new clean SLC machine
  - Enables Node Production repository
  - In case of a patch
    - Installs & configures affected node type (Savannah information)
    - Verifies bug (attached to patch)
    - Enables patch repository
    - Installs patch
    - Verifies bug is fixed
    - Runs set of test
  - In case of a new metapackage or metapackage changes
    - Enables patch repository
    - Installs cert-glite-* package
    - Checks if node type is ok
  - Sets Savannah patch state to “Certified”
• **PPS**
  – Announces via mail which patches go to PPS
  – Generate GGUS ticket

• **Integrator**
  – Applies the patches to the PPS lists
  – Generates new metapackages depending on the new RPMs
  – Copies new files into a current PPS shadow repository and generates the yum files
  – Generates Twiki release notes page
  – Creates UI/WN TAR ball
  – Sends announcement of new repository
Like CERT to PPS, but

- Releases notes are created differently
  - A clean **SL** machine tries to install all nodetypes consecutively
  - Yum output is parsed for packages being pulled in
  - List is compared to previous release list
  - HTML page with differences is generated for each node
- RSS feed is created with update notification

Relesenotes creation gives good opportunity to check for correctness

- Installation problems can be spotted early
- However, takes very long time (~ 1.5h)
Procedure Problems (technical)

- **Past Problems**
  - Metapackage changes were done manually
  - Documentation
  - HowTos
  - Release scripts
    - Error handling
    - Stability
    - Able to handle easily new releases (e.g. 3.2, 3.3, ...)

- **Current Problems**
  - Missing Savannah Interface
  - Web page extraction for RPM names and changes by sed/awk/tr
  - Release notes creation takes too long
Procedure Problems (formal)

- **Past Problems**
  - none

- **Current Problems**
  - Savannah patches are not filled out properly
  - Developers need improved tools/help to do so
    - More use of `checkAffectedMetapackages.sh` script (see Appendix)
• **Future work**
  – gLite web pages should be revised
  – Node update pages should be generated by **repoview**
• Using Patch Repositories
  – Enable Patch repository
    • `/etc/yum.repos.d/patch1234.repo`

```
[Patch 1234]
name=Patch 1234
enabled=1
```

  – Exclude a package from a repository

```
[Patch 1234]
name=Patch 1234
enabled=1
exclude=excluded_package
```

  – Protect packages from repository to be upgraded

```
[Patch 1234]
name=Patch 1234
enabled=1
protect=1
```

This does not work when installing the packages the first time
<table>
<thead>
<tr>
<th>Repository</th>
<th>AFS</th>
<th>WEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROD</td>
<td>/afs/cern.ch/project/egee/gLite/</td>
<td><a href="http://linuxsoft.cern.ch/EGEE/gLite/R3.1/">http://linuxsoft.cern.ch/EGEE/gLite/R3.1/</a></td>
</tr>
<tr>
<td>Integration</td>
<td>/afs/cern.ch/project/gd/www/glite/integration/repository/externals</td>
<td>None</td>
</tr>
<tr>
<td>ETICS</td>
<td>/afs/cern.ch/project/etics/repository</td>
<td><a href="http://eticssoft.web.cern.ch/eticssoft/repository/">http://eticssoft.web.cern.ch/eticssoft/repository/</a></td>
</tr>
</tbody>
</table>
• General Integration Procedure
  – https://twiki.cern.ch/twiki/bin/view/EGEE/IntegrationProcedure

• How to fill out a patch
  – https://twiki.cern.ch/twiki/bin/view/EGEE/HowToFillAPatch
  – EGEE MSA 3.2

• Check affected Metapackages

• Savannah client Interface
  – https://twiki.cern.ch/twiki/bin/view/EGEE/SavannahCommandLineInterface

• EMT
  – https://twiki.cern.ch/twiki/bin/view/EGEE/EMTHome

• Node Tracker
  – https://twiki.cern.ch/twiki/bin/view/EGEE/Glite31NodeTracker