Software Lifecycle Management for WLCG beyond EMI (Main points)

The EPEL FedoraProject provides a process to release packages and supports this process with a tool chain suited for distributed teams.

Due to the packaging rules and licensing restriction not all material needed for WLCG middleware services and clients can be integrated with EPEL. Currently these packages are provided via the EMI and egi-UMD repositories.

dCache Java packages cannot be integrated with EPEL. Sites should take the server code directly from http://dcache.org/ as it is already common practice on most sites. The clients need to be integrated with the UI and WN meta packages. This can be done via the UMD/WLCG repository. A concrete discussion on WLCG requirements for middleware repositories after emi has started with EGI.

These repositories fulfil to a large extend the functionality required by this proposal. For simplicity I will refer throughout the text to this additional needed repository as the UMD/WLCG repository since it isn't clear who will provide and manage these repositories in the future.

Many product teams will remain, but with reduced effort. Will be missing the coordination between areas (PTs, users, sites).

WLCG needs to be active in rollout testing and coordinating site upgrades.

Overall approach to use open software products – EPEL in this area. Careful management needed EPEL-test to EPEL-prod especially as user/expt. Code overlaps m/w. Version management/announcement/validation not in EPEL.

More reliance on local solutions being shared. Particularly in packaging and configuration.

Straw-man process:

Integration Point, Packaging and Repositories

If technical feasible all software has to be provided in an EPEL compliant format. Those packages that can't be included in EPEL have to be provided in the UMD/WLCG repository. Alternative packaging formats, such as used for CERNVMfs, re-locatable tarballs etc. have to be derived from EPEL-stable and UMD/WLCG repositories. The integration point is the combination of the EPEL repositories and the UMD/WLCG production repository.

Product Teams

Product teams make use of pilot services for pre-EPEL testing.

Coordination

Tracking (issues/requirements) via GGUS. Discussion via pre-GDB every 4 months. WLCG MB then endorses priority list and tracks progress.

decommissioning of components, interfaces, APIs and the move to new OS versions tracked/coordinated via twiki tables. To minimize the workload on a central team it is essential that information is maintained and updated by those who do the work.

Middleware configuration:
Small sites profit from simple tools, such as YAIM or RPM post installation scripts. For large sites integration with their fabric management tool is a must. Given the multitude of tools, Quattor, puppet, cfengine, YAIM etc., it is not reasonable to expect that PTs acquire the necessary knowledge to support all.

Sites that support specific configuration tools should be encouraged to share their Work – perhaps at pre-GDB.

Validation in Production, Staged Rollout:
This requires a significant effort from sites and experiments. Currently we are preparing an early form of this process for the validation of emi clients. Use of twiki to show progress and led by Ops coordination team.

The staged rollout will start when packages are moved from the PTs repositories to the EPEL-test repository. A sufficiently long period for the transition to EPEL-stable should be set to avoid that packages are moved without proper validation. It is the PTs' responsibility to inform the involved sites and experiments.

For packages that are not much used outside our community, such as Globus, changes have to be tested explicitly via staged rollout. An agreed list of packages that WLCG needs to watch is needed.

Rollback
The concept of rolling back to the last recent working version isn't a concept well supported by the EPEL repository. To rollback the previous version of the packages are moved into the UMD/WLCG
repository.

An approach easier for the site can be implemented via the epoch functionality of RPMs. However this concept of a hidden version number is confusing for developers, site admins and users. In addition it is in conflict with the EPEL style guide.

Managing and tracking middleware versions in the infrastructure
In the past (pre emi-2) the middleware providers produced versioned components. Clearly defined sets of RPMs, for the middleware layer, were defined by versioned meta packages and a set of repositories. The minimal acceptable versions were communicated via the WLCBaselineVersions twiki page.

The situation is more complicated for clients due to the different distribution channels.

During the build up of WLCG sites installed frequently inadequate versions of the clients. Experiments needed to provide client libs and tools via their software distribution mechanisms

With the move of the integration point to EPEL the approach changed. This is not due to a policy decision by emi, but due to the fundamental concepts of EPEL.

The Product Teams release service components through the EPEL process without versioned meta packages. EGI/WLCG creates versioned meta packages for components.
Services publish the installed version automatically with the ResourceInfoProvider into the information system. To improve security this version number should be encoded with a public key

Required Resources
All efforts are estimates assuming a significant reduction in development activity after the end of emi and should be seen as a starting point for a discussion.
Most effort is distributed and has to be seen as part of the work of the PTs.