Implementing product teams

Oliver Keeble
EGEE SA3
CERN

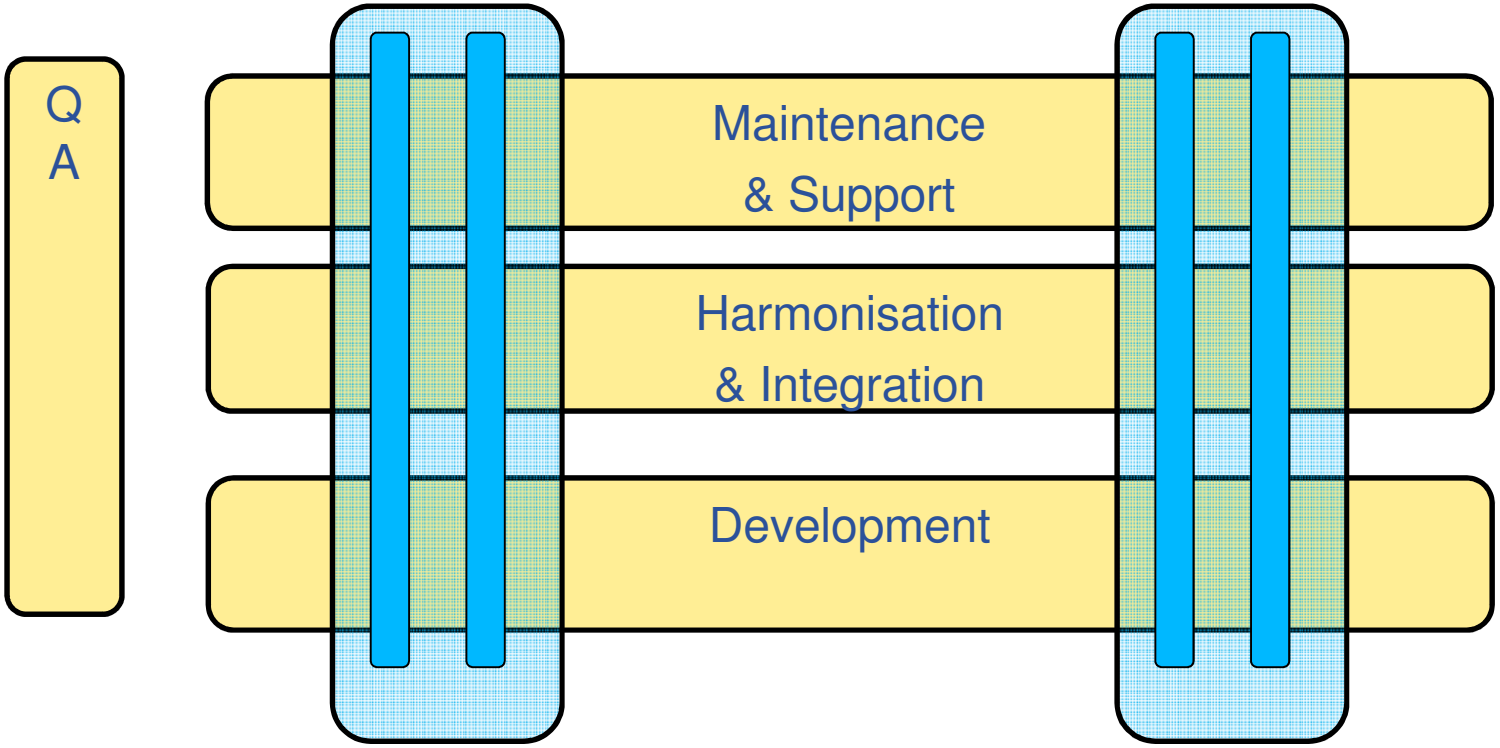
Summary

- **WHY**
  - Quick reminder about the “EGI era”
- **WHAT**
  - What are Product Teams?
- **HOW**
  - How will be begin to implement them?

This is not an EGI talk, but ...

- **The purpose of the current exercise is to orient EGEE-III towards EGI structures in order to achieve a smooth transition**
- **There are a number of players in the EGI era, particularly**
  - EGI.eu
  - EMI
  - gLite Collaboration
- **All incorporate the concept of Product Teams**
  - EGI envisages middleware provision via independent 'Product Teams' which meet software requirements described by the EGI.eu Middleware Coordination Board (MCB)
  - The EMI (European Middleware Initiative) retains the concept of Product Teams

Proposed Lifecycle in EMI



Example product teams

- **Data Management**
  - node-types: DPM, FTS, LFC, ...
  - components: gfal/lcg_util, ...
- **Security**
  - node-types: HYDRA, SCAS, ...
  - components: trustmanager, ...
- **UI**
  - node-type: UI
  - components: none

Advantages of this approach

- **It is what is envisaged within EGI**
  - ie it maps onto the devolved model of EGI
- **Reduces the total certification workload**
- **Naturally produces node-type oriented releases**
  - Will remove the necessity for ‘glite update #N’ as we can just refer to metapackages
- **Development teams work in the standard production environment**

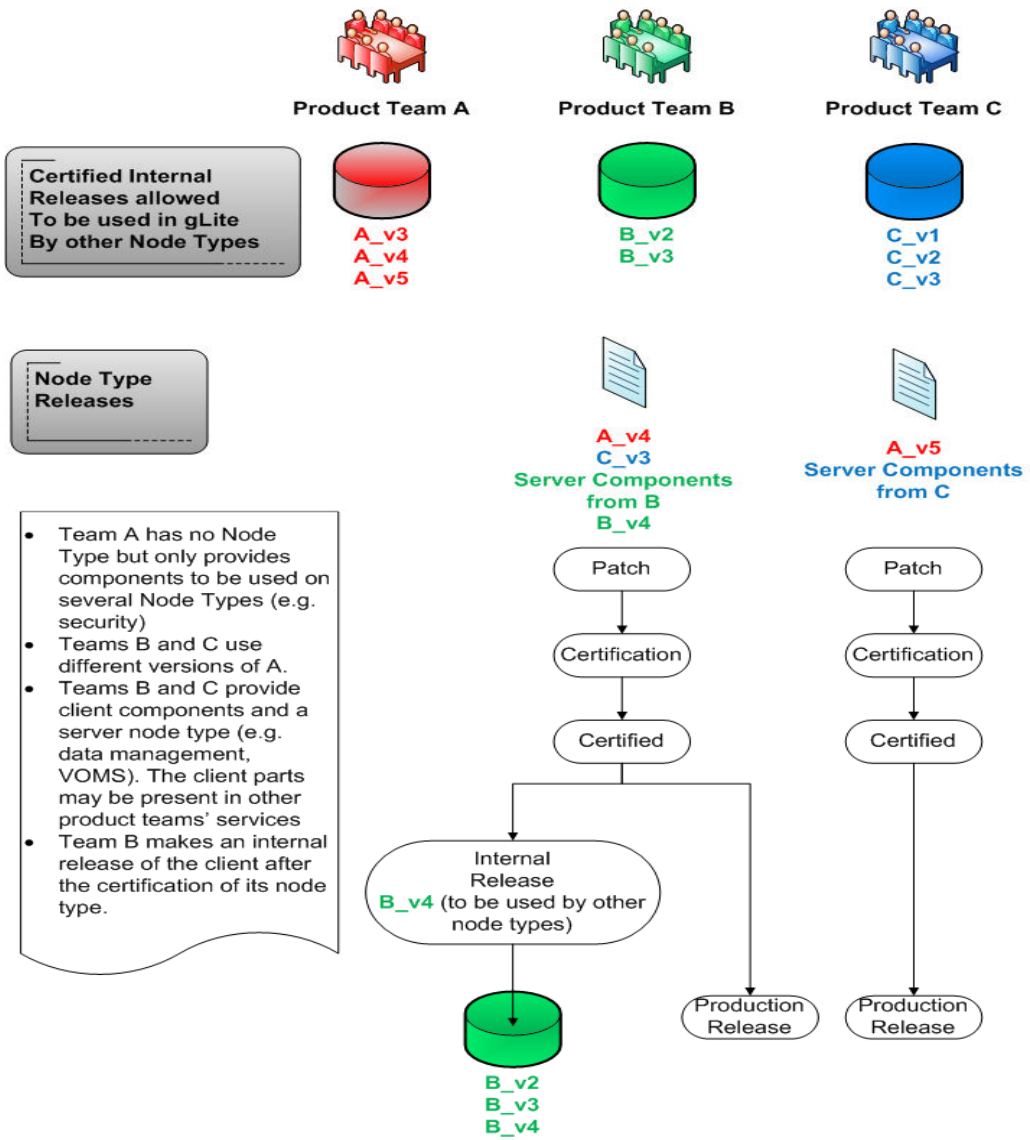
PT's responsibilities in EGEE-III

- **A product team is a group responsible for producing quality-ensured middleware releases and updates of self-contained products.**
  - **Maintain** and enhance the **codebase** according to agreed JRA1 workplan.
  - **Maintain** the integrated node-type (where relevant), including all necessary **documentation**
  - To produce an update, identify the complete rpm list and define the required build
  - **Perform** the **build** against the glite-SDK
  - Resolve general build issues, including those related to multi-platform support, in conjunction with the porting coordinator
  - **Run** deployment **tests** and regression tests and any other 'pre certification' smoke tests.
  - Create 'internal' patches to advertise internally generated changes of relevance to other product teams
  - **Create** 'production' **patches** when triggered by the availability of relevant updates (generated both internally and externally to the team).
  - **Update** configuration and produce any necessary **YAIM** packages
  - **Write** the **patch release notes**, incorporating relevant information on changes generated by other product teams
  - **Certify** the internally generated **patches**. This involves running specific tests in a controlled and recorded environment, resulting in a 'pass' or a 'fail. The tests and acceptance criteria are already available and documented.
  - **Provide** new **tests** for the services in question where judged appropriate or where requested. Note that a large and documented body of tests exists already.

Types of release

- **A product team will typically produce two different types of release, each represented initially by a savannah patch**
  - Internal releases are a way of informing other product teams that components they depend on have been updated
    - DM produce an internal patch of gfal/lcg_util which the UI and WN product teams could pick up
  - Production releases are certified versions of node-types with one or more updated components, originating with one or more product teams
    - DM produce a production patch of glite-DPM_mysql which includes new DPM components but also a new gridftp server and resource BDII
    - This is the “product”

Workflow


Product Team A
Product Team B
Product Team C
Certified Internal Releases allowed To be used in gLite By other Node Types
A_v3
A_v4
A_v5
B_v2
B_v3
C_v1
C_v2
C_v3
Node Type Releases
A_v4
C_v3
Server Components from B
B_v4
A_v5
Server Components from C
Team A has no Node Type but only provides components to be used on several Node Types (e.g. security)
Teams B and C use different versions of A.
Teams B and C provide client components and a server node type (e.g. data management, VOMS). The client parts may be present in other product teams' services
Team B makes an internal release of the client after the certification of its node type.
Patch
Certification
Certified
Internal Release
B_v4 (to be used by other node types)
Production Release
Patch
Certification
Certified
Production Release
B_v2
B_v3
B_v4

To get started

- **What follows is a first pass at implementing PTs**
  - It will be reviewed and changed/extended on the basis of experience
- **Build is as now, with ETICS**
  - will be generalised to an official build environment
  - working source rpms will be obligatory
- **Change tracking is savannah**
  - will need two types of patch, 'internal' and 'production'
- **Version Control**
  - for the PT to decide
    - CERN will ultimately replace the CVS service with SVN
- **The end result of the 'product team' will be a 'production patch' in the state 'certified'**
  - this will already contain the necessary meta-package
  - the current central team will then take over

Internal Patches

- These serve as input to 'production patches‘
- For example, a trustmanager update which would be picked up by other PTs but is not incorporated into any node-types managed by that PT
- A PT should maintain a web page with its latest releases and the recommended, certified versions of its products
- A central repo will be maintained where all such releases can be found
  - central team will do this when an internal patch is created

Production Patches

- **This is an update to a self-contained product, ie a node-type**
- **The PT is accountable for the release**
  - even if the changes were generated in other PTs
- **This will look like a familiar savannah patch**
  - the metapackage will already be there
- **A production release is NOT necessarily triggered every time there is a change in any constituent component**
  - changes accumulate and an update is triggered by a sufficiently important one
- **Central SA3 will provide the tools necessary for a PT to steer a patch through the familiar states**

Implementation of production patches

- PT creates a patch for a single node-type
- Add the new packages for which they are responsible
- Generate list of all other packages which have been updated
- Add this to the patch
  - -> ‘ready for integration’
- Create certification repository and metapackage
  - -> ‘ready for certification’
- Certify patch
  - (-> ‘in certification’) -> ‘certified’
- Central team now takes over, starting with signing the rpms
- Production repository will be split per node-type

Certification

- **Happens within the product team**
- **Must have equivalent coverage to now**
  - Process and tests are documented and available
  - SA3 CERN people will be logically assigned to PTs
- **PTs should contribute reference services to the testbed**
  - CERN central coordination will remain
- **A PT is not expected to certify its components in all contexts**
  - eg the VOMS PT does not certify the VOMS clients on all other node-types
- **Rejection**
  - The certifying PT should alert the supplying PT that their component has been rejected
    - the production patch is rejected
    - the supplying PT may choose to reject the corresponding internal patch
    - the supplying PT should update their webpage to indicate the rejection

"Known issues"

- **Diverging build environments – if each node-type can evolve independently, will one gLite build environment suffice?**
  - Inbuilt convergence via internal release repository
- **Generating a node-type can require rebuilds for statically linked components**
  - This can be done as long as the PT is aware there is an update available
- **Test coverage for full service tests ('end-to-end')**
  - Central testbed
- **A new version of a client is only guaranteed to be tested against a new version of a server if it is done at the 'internal testing' before the internal release.**
  - This should be a certification requirement
- **There is a risk that a product team would incorporate, unknowingly, a component already rejected by another product team.**
  - These could be remove from the internal repository
- **Release process metrics will have to be rethought**

Summary

- **Product Teams are inherent in the forthcoming lifecycle models**
- **We will implement them in EGEE-III to gain experience and to ensure a smooth transition**
- **A PT takes full responsibility for a production release**
  - In the first instance, up until the state ‘certified’
- **Exact levels of autonomy to be established**
- **They will be composed of JRA1 & SA3 people**
- **Central SA3 will provide tools, services and support**
- **Next step:**
  - When the tooling is finished, we need a small number of PTs to start testing the process