Why are common quality and development policies needed?

M Alandes¹, A Abad¹, L Dini¹ and P Guerrero¹

¹ European Organization for Nuclear Research, CERN CH-1211, Geneve 23, Switzerland

E-mail: maria.alandes.pradillo@cern.ch

Abstract. The EMI project is based on the collaboration of four major middleware projects in Europe, all already developing middleware products and having their pre-existing strategies for developing, releasing and controlling their software artefacts. In total, the EMI project is made up of about thirty development individual teams, called “Product Teams” in EMI. A Product Team is responsible for the entire lifecycle of specific products or small groups of tightly coupled products, including the development of test-suites to be peer reviewed within the overall certification process. The Quality Assurance in EMI (European Middleware Initiative), as requested by the grid infrastructures and the EU funding agency, must support the teams in providing uniform releases and interoperable middleware distributions, with a common degree of verification and validation of the software and with metrics and objective criteria to compare product quality and evolution over time. In order to achieve these goals, the QA team in EMI has defined and now it monitors the development work and release with a set of comprehensive policies covering all aspects of a software project such as packaging, configuration, documentation, certification, release management and testing. This contribution will present with practical and useful examples the achievements, problems encountered and lessons learned in the definition, implementation and review of Quality Assurance and Development policies. It also describes how these policies have been implemented in the EMI project including the benefits and difficulties encountered by the developers in the project. The main value of this contribution is that all the policies explained are not depending on EMI or grid environments and can be used by any software project.

1. Introduction

The EMI project (European Middleware Initiative) [1] is comprised of 28 software development teams, called product teams (PTs), who develop the 56 EMI software products. PTs are “self-organizing” teams who are responsible for the entire lifecycle of one or more software products. EMI PTs are coming from major middleware providers like ARC [2], dCache [3], gLite [4] and UNICORE [5], who have been developing software in the grid domain for the past several years, always in the framework of European projects like EGEE [6], OMII-Europe [7], KnownARC [8] and others. The following particularities characterise the corresponding development teams:

- Heterogeneous developer profiles: different backgrounds and different degrees of expertise.
- Geographically distributed.
- Different home institutes which implies different cultures, different development technologies, process and methods.
- High turnover due to the limited duration of the projects where the grid software has been developed so far.
Lack of formal tracking to manage user requirements that moreover have been evolving and changing throughout the years as grid technology became more mature and users more aware of the features that they really needed.

More focus on development activities, with limited resources, if any, available for quality assurance activities.

The EMI project aims at delivering a consolidated set of middleware software products from these middleware providers for deployment in WLCG [9], EGI [10], PRACE [11] and other Distributed Computing Infrastructures (DCIs). In order to deliver a consolidated set of middleware components, a set of documents called EMI QA Policies [12] have been defined to guide the development teams toward uniform practices and processes.

2. Quality Assurance Policies in EMI

EMI QA Policies are the documents that define the main activities of the software lifecycle in the EMI project, such as releasing, tracking, packaging and documenting the EMI software. See figure 1. EMI QA Policies are therefore the mean to guarantee an unambiguous release process within EMI, common for all product teams. EMI QA Policies are defined in detail the next subsections.

![Software Quality Assurance Diagram](image)

**Figure 1 – Documents defined in the EMI Software Quality Assurance Plan, including the EMI QA Policies**

2.1. Release Management Policy

The EMI Release Management Policy describes the process to be followed when managing EMI releases. It describes the actions to be performed by the Release Manager (technical management of releases, continuous integration, repository creation and EMT coordination, which is a weekly meeting
with product teams to prioritize and manage the upcoming releases). It also contains the definition of
the different elements that define a release (user requirements, technical development tasks, release
tasks, changes, etc) and the corresponding tracking tools used within the project.

2.2. Change Management Policy
The EMI Change Management Policy describes how to manage software changes in EMI products. It
describes the different types of EMI releases and it includes a detailed overview of the different
tracking tools to track Release Tasks (where new product versions are tracked) and Request for
Changes (bugs or defects released with a new product version) which are the main elements that PTs
deal with when preparing a release. It defines in detail the state transition diagrams, mandatory fields
that need to be present in Release Tasks and Request for Changes and how they can be included in the
different types of EMI releases.

2.3. Configuration and Integration Policy
The EMI Configuration and Integration Policy describes how to build EMI middleware. It is
comprised of the build integration policy describing the single build environment used by all PTs,
which is ETICS; and the build configuration policy describing how to configure ETICS [13] to
properly build EMI software products.

2.4. Packaging Policy
The EMI Packaging Policy describes how to create packages for the EMI software products. It
specifies the supported package formats, their contents and metadata.

2.5. Testing Policy
The EMI Testing Policy describes how to test EMI software products. It details the type of tests that
need to be run by PTs to test new product versions (Unit Tests, Deployment Tests, Functionality Tests,
Regression Tests, etc). It also contains a template to define the Test Plan for each EMI product, giving
specific instructions on what needs to be included. It also provides a template for the Testing Report
that all PTs must use to report about the results of the testing phase when releasing new product
versions.

2.6. Documentation Policy
The EMI Documentation Policy describes how to document EMI software products. It details the type
of documents that have to be provided by each EMI software product (user guide, functional
description, system administrator guide, release notes, troubleshooting guide, man pages, etc).
Different audiences have been identified (End users, system administrators and developers) and
documents have been classified accordingly. Formatting and versioning have been also defined to
have homogeneous set of documentation across the project.

3. Quality Assurance Policies in practice: encountered problems and solutions

3.1. Compromise between existing processes and new EMI policies
EMI QA Policies were defined by the QA team of the EMI project. This was done in collaboration
with product teams and EMI project management. Product teams were involved since the beginning at
the early stages of the project. One of the main issues found during the definition stage, was to make
sure that policies were flexible enough to co-exist as much as possible with existing development
methods, since product teams were reluctant to change their current ways of working. Creating
awareness of the need of a common and homogeneous process was crucial to get a commitment from
the product teams in the use of the policies.
3.2. Deviations from the policies
The QA team is responsible for monitoring the application of *EMI QA Policies* throughout the different stages of the software lifecycle. This is done in collaboration with the Quality Control (QC) team. When deviations from the policies are detected, the QA team discusses with PTs to understand why policies have not been respected. It could be that a policy is obsolete and does no longer reflect the real needs of the project and therefore the product teams find no motivation in following the policy. In this case, the QA team should update the relevant policies accordingly (a well-defined process exists to create new versions of the policies). It could also be that the policy is up to date but it is not being followed by the product team. In this case, the QA team reports to the release manager, who follows up with the concerned product teams.

3.3. Effort to know the policies
Product teams usually find it difficult to keep up to date with policies. Following the policies and spending time to be familiar with them could be sometimes a burden. The QA team carries out regular dissemination activities to help product teams be familiar with the policies or to keep up with changes. A release check list is also provided so that product teams can easily follow all the necessary steps to prepare a release without knowing in detail the contents of the different policies.

4. Benefits of applying Quality Assurance Policies

4.1. Coherent and homogeneous releases
The main achievement of defining the *EMI QA policies* was the successful release of 54 products in EMI 1, the first release of the EMI project. The EMI 1 release is a coherent and homogeneous set of EMI software products in terms of documentation, test reports and packaging that was released according to the specified timing constraints. Policies played a fundamental role in this achievement directing product teams, release manager and QA team in the release process. Even if not all the mandatory criteria defined in the policies was actually covered by all products, the policies themselves also helped to track the deviations and to work with the concerned PTs for future improvements.

![Figure 2 - Organization of the EMI release according to the EMI QA policies. Trackers allow to monitor the state of the release and to perform measurements on the quality of the product and the process.](image-url)
4.2. Up to date information on the status of the release

Another important aspect of the policies is the possibility of accessing up to date information on the status of the release. The Release Management Policy defines the way of tracking user requirements and technical objectives. The Change Management Policy defines the way of tracking the next releases of the EMI software products. Thanks to this, it is possible to know whether certain technical objective has been implemented or not, whether a user requirement has been accepted for inclusion in a next release, whether certain version of a product has been released or not, etc. See figure 2.

The Change Management Policy also defines a common layer for the different software defect/feature trackers used by the different product teams. Thanks to this, the QA team has developed a tool, the RfC tracker [14], who is able to query all these trackers and offer a unique view of the status of the upcoming releases. See figure 3.

![Figure 3 – RfC Dashboard offering a single entry point to track defects and features of EMI products.](image)

4.3. Measurable processes and software products

Finally, policies are fundamental to be able to measure the release process and the EMI software products in terms of quality. The QC team controls that for each released software product, policies have been applied by checking testing and certification reports. The QA team maintains the Production Release Criteria [16], which is what the QC team actually checks, and that basically contains the set of mandatory criteria from each policy. The Production Release Criteria (i.e. mandatory tests, mandatory documents, mandatory fields in the defect trackers, etc) eases the task of the QC and metrics activity. A QC Verification Dashboard [15] has been developed to automatically collect data from new versions of software products, showing whether a software product meets the minimum criteria to be released in production or not. See figure 4.
5. Lessons learned

5.1. Start early
The definition of QA policies must be a top priority for any software project. They play a fundamental role in the definition of the software development process. The earlier policies are defined, the better. The work of development teams and release manager will be easier since everybody will have a clearer picture of what to do. This will avoid misunderstandings and everybody will move forward in the same direction. Moreover, the effort to define common policies in a big software project with many development teams should not be neglected. It requires a considerable amount of time to coordinate with so many people and agree on important aspects of the process that affect everybody. This is also a reason to focus on the definition of the policies in the first stages of the project.

In the case of EMI, some policies took months to be agreed upon. The Documentation Policy was approved very close to the release date and this cause many confusion within the product teams.

5.2. Involve the necessary people
All the people who are going to be affected by the policies in one way or another should be involved in their definition. This normally affects developers, release manager, but also the QA and QC teams and the project management. Involving all the parties, makes sure the concerns of everybody are taken into account and that all aspects of the software development lifecycle are covered.

In the case of the first versions of EMI QA policies, product teams were represented by at least one member of their team. A QA team member was coordinating the definition of each policy. The project management was giving the final approval to the policy once a final version was ready. In further versions of the policies, there is a two week window open for everybody in the project for comments, after which the policies are considered approved.

5.3. Evolve
Policies become sometimes obsolete because the “theory” turns out not to match the real practices or because the needs of the project change and they are not reflected in the policies. The QA team is
responsible for following up closely the changes in the project and how they could affect policies. Policies cannot be defined at the beginning of the project and later on forgotten. It is necessary to review them throughout the life of the project.

In the case of EMI, a review of the policies is done after each major release, applying lessons learned (i.e. removing too rigid or unrealistic criteria) and making sure the policies are going to be valid also for the next major release (i.e. new platforms).

6. How to reuse EMI QA policies
EMI QA policies have been defined in the context of the EMI project. However, they describe the foundations of a software lifecycle and therefore they could be easily adapted to be applied in other software projects. Moreover, EMI QA policies have already taken into account a heterogeneous development environment as it is the one of the EMI project. The EMI project integrates four different middleware distributions comprising many development teams with their own well established development process. The EMI release lifecycle has been defined with a high level of flexibility, trying to respect and co-exist as much as possible with different development choices. This means that applying EMI QA policies in other software projects is very easy due to the versatile approach of the policies.

The next sections explain how the different EMI QA policies could be easily adapted in a software project of a different scope than that of the EMI project.

6.1. Release Management Policy
The EMI Release Management Policy describes the process to be followed when managing EMI releases. Although the document contains internal project elements, it could be a useful example to be used as a guide for any project interested in defining the release management task.

6.2. Change Management Policy
The EMI Change Management Policy describes how to manage software changes in EMI products. It has a project specific part linked to the EMI release tracker and a very generic part that could be used in connection to any software change tracking tool, as it is already the case in EMI where each product team can use the tracking tool of their choice. The part of the policy linked to the EMI release tracker can be easily adapted to work with any other tool. It is therefore a very useful document for any software project interested in organising and documenting how to implement software changes.

6.3. Configuration and Integration Policy
The EMI Configuration and Integration Policy is a more project specific document since it is tightly linked to the project build system ETICS. However, EMI is evolving to let product teams choose the build tool of their choice and the policy has been updated to reflect this tendency. It is therefore an interesting document for other software projects which need to write guidelines on how to carry out the build process. Its structure and contents can be a good starting point for other projects who may want to follow a similar approach.

6.4. Packaging Policy
The EMI Packaging Policy is very closely tight to EPEL and Debian packaging guidelines. It could be a useful document for other software projects willing to adopt some specific packaging rules.

6.5. Testing Policy
The EMI Testing Policy is a very comprehensive guide on how to organise the testing activity within a project. It details the basic set of tests types that any software project should consider running as part of its testsuite. This includes unit tests, deployment tests, functionality tests or regression tests. It
contains a template to define a test plan which can be adopted by any software project. The Testing Report template is also generic and it could be applied to any software.

6.6. Documentation Policy

The EMI Documentation Policy could be easily adopted in any other software project. The policy describes a set of best practices in terms of contents and structure for all type of software related documents like system administrator guide, user guide, installation and configuration manual, functional description, release notes, etc. In particular, the policy gives a detailed table of contents to be used for the system administrator guide.

7. Acknowledgements

This work was partially funded by the EMI project under European Commission Grant Agreement INFSO-RI-261611.

8. References

[1] European Middleware Initiative project (EMI) http://www.eu-emi.eu/
[2] Advanced Resource Connector (ARC) http://www.nordugrid.org/arc/
[3] dCache http://www.dcache.org/
[4] gLite http://glite.cern.ch/
[5] Uniform Interface to Computing Resources (UNICORE) http://www.unicore.eu/
[6] Enabling Grids for e-Science http://www.eu-egee.org/
[7] Open Middleware Infrastructure Institute for Europe (OMII-Europe) http://omii-europe.omii.ac.uk/
[8] KnownARC http://www.knowarc.eu/
[9] Worldwide LHC Computing Grid (WLCG) http://wlcg.web.cern.ch/
[10] European Grid Infrastructure (EGI) http://www.egi.eu/
[11] Partnership for Advance Computing in Europe (PRACE) http://www.prace-project.eu/
[12] EMI Quality Assurance Policies https://twiki.cern.ch/twiki/bin/view/EMI/SA2
[13] eInfrastructure for Testing, Integration and Configuration of Software (ETICS) http://etics.web.cern.ch/etics
[14] EMI RfC Tracker Dashboard http://emi-rfc.cern.ch/
[15] EMI Verification Dashboard http://emi-dashboard.cern.ch/
[16] EMI Production Release Criteria https://twiki.cern.ch/twiki/bin/view/EMI/ProductionReleaseCriteria