Improving the quality of EMI Releases by leveraging the EMI Testing Infrastructure

C Aiftimiei[1,*], A Ceccanti[2], D Dongiovanni[2], A Di Meglio[3] and F Giacomini[2]

[1]INFN-Padova - Ist. Naz. di Fisica Nucleare, Via Marzolo 8, 35131 Padova (Italy)

[2]INFN-CNAF - Ist. Naz. di Fisica Nucleare, Viale Berti Pichat, 6/2 40126 Bologna (Italy)

[3]CERN - European Organization for Nuclear Research, 1211 Geneva (Switzerland)

[*] on leave from NIPNE-HH, Romania

e-mail: cristina.aiftimiei@pd.infn.it

Abstract. What is an EMI Release? What is its life cycle? How is its quality assured through a continuous integration and large scale acceptance testing? These are the main questions that this article will answer, by presenting the EMI release management process with emphasis on the role played by the Testing Infrastructure in improving the quality of the middleware provided by the project. The European Middleware Initiative (EMI) is a close collaboration of four major European technology providers: ARC, gLite, UNICORE and dCache. Its main objective is to deliver a consolidated set of components for deployment in EGI (as part of the Unified Middleware Distribution, UMD), PRACE and other DCIs. The harmonized set of EMI components thus enables the interoperability and integration between Grids. EMI aims at creating an effective environment that satisfies the requirements of the scientific communities relying on it. The EMI distribution is organized in periodic major releases whose development and maintenance follow a 5-phase yearly cycle: i) requirements collection and analysis; ii) development and test planning; iii) software development, testing and certification; iv) release certification and validation and v) release and maintenance. In this article we present in detail the implementation of operational and infrastructural resources supporting the certification and validation phase of the release. The main goal of this phase is to harmonize into a single release the strongly inter-dependent products coming from various development teams through parallel certification paths. To achieve this goal the continuous integration and large scale acceptance testing performed on the EMI Testing Infrastructure plays a key role. The purpose of this infrastructure is to provide a system where both the production and the release candidate product versions are deployed. On this system inter-component testing by different product team testers can concurrently take place. The Testing Infrastructure is also continuously monitored through Nagios and exposed both to automatic testing and to usage by volunteer end-users. Furthermore the infrastructure size is increased with resources made available by volunteer end-users that are interested in implementing production-like deployments or specific test scenarios.

EMI Middleware

The European Middleware Initiative (EMI) [1] is a close collaboration of four major European technology providers: ARC [2], gLite [3], UNICORE [4] and dCache [5]. Its main objective is to deliver a consolidated set of components for deployment in the European Grid Infrastructure (EGI)

[6] (as part of the Unified Middleware Distribution, UMD [7]), PRACE [8] and other Distributed Computing Infrastructures (DCIs). The harmonized set of EMI components thus enables the interoperability and integration between Grids. EMI aims at creating an effective environment that satisfies the requirements of the scientific communities relying on it.

The EMI software stack currently consists of selected components provided by the ARC, gLite, UNICORE and dCache middleware consortia. At the end of the project, the EMI stack will also include some new products developed through common efforts. Such new products include the EMI Registry service (EMIR), common clients for data management, job submission and authorization. The maintenance and development of the EMI components (or products) is carried out by their respective Product Teams by following well-defined procedures and policies described by the project.

The EMI software stack is currently composed of 56 products with more than three millions single lines of code.

The four middleware providers produce software falling in the category of grid middleware. The generally accepted architecture of a grid adheres to the so-called Service Oriented Architecture (SOA) and follows the "hourglass model" as shown in Figure 1.

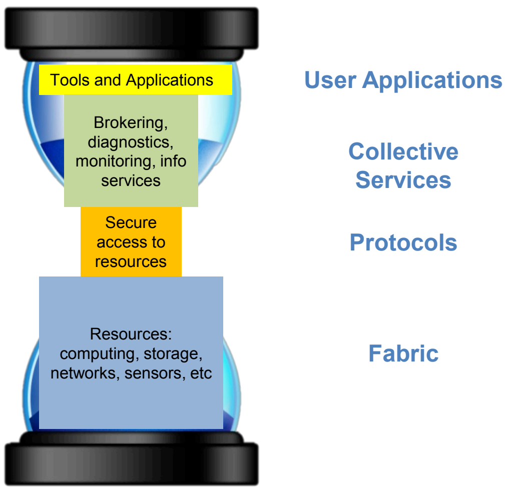

Tools and Applications
Brokering, diagnostics, monitoring, info services
Secure access to resources
Resources: computing, storage, networks, sensors, etc
User Applications
Collective Services
Protocols
Fabric

Figure 1: The grid hourglass model

The EMI services fall in the Fabric and Collective Services categories and make use of a number of protocols and interfaces, some of which are recognized open standards, while others are proprietary to specific EMI services.

As a general reference, the currently planned EMI products can be categorized in one (in a few cases more than one) of four major classes or Technical Areas: Compute Management, Data Management, Security and Infrastructure. The first three areas mainly refer to services in the Fabric architectural category, while the fourth area mainly refers to services in the Collective Services category. In addition, a product can be classified as a:

- **Service**: provides a well-defined set of features (behaviors or capabilities) through a published interface. Both interfaces and behavior must be publicly documented, supported and maintained and are subject to public transition and lifecycle processes
- **Client**: provides capabilities to interact with the services through their published interface and a separate user interface accessible with command-line or graphical commands. The user interface must be publicly documented, supported and maintained and are subject to public transition and lifecycle processes
- **Library**: provides capabilities to interact with the services through their published interface or implements a set of tools and utilities. It provides a programmatic interface with bindings for one or more programming languages. The programmatic interface must be publicly documented, supported and maintained and is subject to public transition and lifecycle processes
- **Internal component**: a sub-element of a service, client or library that does not expose its interfaces to users or external programs. Its interfaces must be documented, but are not

subject to public transition or lifecycle processes and can be changed at any time provided the change does not introduce changes in the published interface or behavior of services, clients, and libraries using them.

This classification is important from the maintenance point of view, since the different roles have different maintenance and support constraints and lifecycles.

1.1 EMI Release process

During the three year duration of the EMI project, the EMI developers and engineers work together to consolidate, harmonize and support the existing software products, evolving and extending them based on existing and new requirements. Redundant or duplicate services resulting from the merging are deprecated; new services can be added to satisfy user requirements or specific consolidation needs. Input for the development activities is taken from users, infrastructures projects, standardization initiatives or changing technological innovations. The software components are adapted as necessary to comply with standard open source guidelines to facilitate the integration in mainstream operating system distributions.

The maintenance and development of the EMI services is based on a 5-step yearly cycle (Figure 2) composed of:

1. **The Requirements Analysis phase**: input collected from the EMI collaboration activities or from direct user submission in the EMI User Support system is analysed and prioritized based on the Severity assigned by the users, the urgency, impact, cost and available effort. The result of the analysis is compiled in the form of the EMI Technical Plans defining the project technical objectives. The plans are defined at the beginning of the project and refined at every cycle based on the new input. The Project Technical Board (PTB) and the specific Technical Areas coordinate the requirements analysis and the overall technical plan. The Technical Area leaders and the Standardization Task leader within Joint Research Activity Work Package (JRA1) coordinate the Standardization Plan.
2. **The Development and Test Plans phase**: based on the latest version of the project Technical Plans, the Development and Test Plans for the current cycle are defined. The plans outline which of the technical objectives can be included in the cycle, which components are involved, which platforms and operating systems can be targeted, external and internal integration constraints, development, deployment and testing timelines, etc. The plans are centrally coordinated by the JRA1 Work Package leader and distributed to the EMI Product Teams (PTs) for implementation. A complementary Release Plan is established by the Support Activity Work Package (SA1) with the details of the timelines to be applied to each release cycle (like code freeze date, release date, any technical preview release date) and the outline of the set of acceptance criteria to be fulfilled by the components (like documentation and specific categories of tests).
3. **The Development, Testing, and Certification phase**: based on the Development and Test Plans, the various PTs develop the new functionality and perform unit, integration, deployment and functional tests under the overall coordination of JRA1 and monitoring of the PTB. Once a piece of functionality has passed the foreseen set of acceptance criteria (set out in the Software Quality Assurance Plan (SQAP)) and has been certified, it is released as Release Candidate to SA1.
4. **The Release Certification and Validation phase**: as Component Releases are transitioned from JRA1 to SA1, the final certification phase starts. During this phase, the software components are validated against the set of acceptance criteria that were defined by the customers. Technical Previews are made available to users and projects taking part in the "Works with EMI" technical collaboration program and the final packaging and signing is performed. Components that do not pass the criteria are rejected back to JRA1 for revision.
5. **The Release and Maintenance phase**: once the software components have passed all acceptance criteria, they are released and uploaded to the official EMI Software Repository from where they can be picked up by users and infrastructure operators. The continuous maintenance phase starts at this point, any defect found by users in production environments and submitted to the EMI User Support system (GGUS), or to the technology providers specific bug trackers, are analysed, prioritised and addressed as revision or minor releases. Any component that has not passed all criteria by the time the final release is due it is rejected and reschedule for another release cycle.

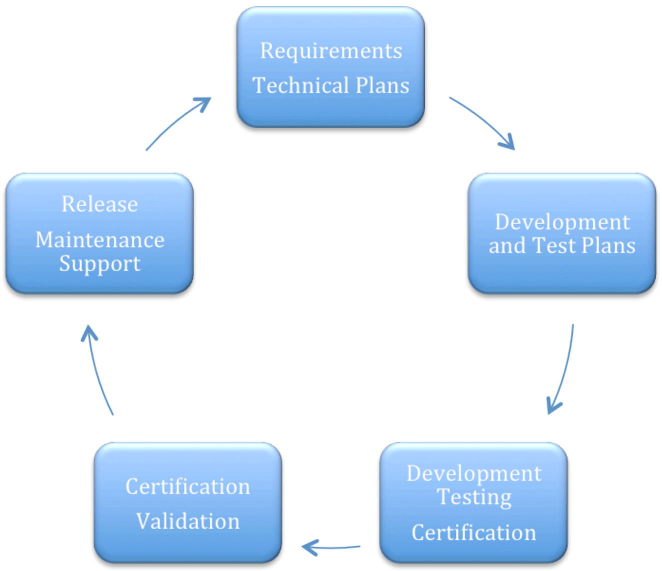


Figure 2: EMI Release Cycle

The EMI software development process is driven by two concurring demands:

- Software evolution, harmonization and consolidation, as defined in the EMI Technical Development Plan, to address user requirements for new functionality and rationalize the EMI software stack. The plans outline which of the technical objectives can be included in the cycle, which components are involved, which platforms and operating systems can be targeted, external and internal integration constraints, development, deployment and testing timelines, etc. The plans are centrally coordinated by the JRA1 Work Package leader and distributed to the EMI PTs for implementation. A complementary Release Plan is established by SA1 with the details of the timelines to be applied to each release cycle (like code freeze date, release date, any technical preview release date) and the outline of the set of acceptance criteria to be fulfilled by the components (like documentation and specific categories of tests).
- Software adaptive and corrective maintenance, to address problems reported by the middleware users or changes in the software operating environment.

In both cases, development is implemented in the context of Product Teams. It is worth noting here that:

- Each PT has the responsibility to implement the project-wide software engineering process defined by EMI;
- The EMI project imposes a configuration and integration process built around the ETICS build system [9]. This integration process is described in detail in the EMI Configuration and Integration Policy [10].

The EMI distribution is organized in periodic *major releases* tentatively delivered once a year providing a good balance between the conflicting requirements of stability and innovation.

An EMI major release is characterized by well-defined interfaces, behavior and dependencies for all included components, available on a predefined set of platforms. What is included in a new EMI major release is defined by the PTB and included in the yearly Technical Development Plan and the implementation of the plan is coordinated by JRA1

Backward incompatible changes to the interface or to the behavior of a component that is part of the EMI distribution can be introduced only in a new EMI major release. Changes to interfaces that are visible outside the node where the component runs (e.g. a WSDL or a network protocol) need to be preserved even across major releases, according to end-of-life policies to be defined on a case-by-case basis.

1.1.1 Components Releases

An EMI distribution includes all the components that are developed within the project and that have reached production quality. Within an EMI major release, only one version of a given component is maintained.
Four types of releases have been identified for a given component:

- **Major Release**: A major release for a component is characterized by a well-defined interface and behavior, potentially incompatible with the interface or behavior of a previous release. New major releases of a component can be introduced only in a new major release of EMI.
- **Minor Release**: A minor release of a component includes significant interface or behavior changes that are backwards compatible with those of the corresponding major release. New minor releases of a component can be introduced in an existing major release of EMI.
- **Revision Releases**: A revision release of a component includes changes fixing specific defects found in production and represents the typical kind of release of a component during the lifetime of an EMI major release.
- **Emergency Releases**: An emergency release of a component includes changes fixing only Immediate-priority defects found in production, typically security-related.

1.1.2 EMI Updates cycle

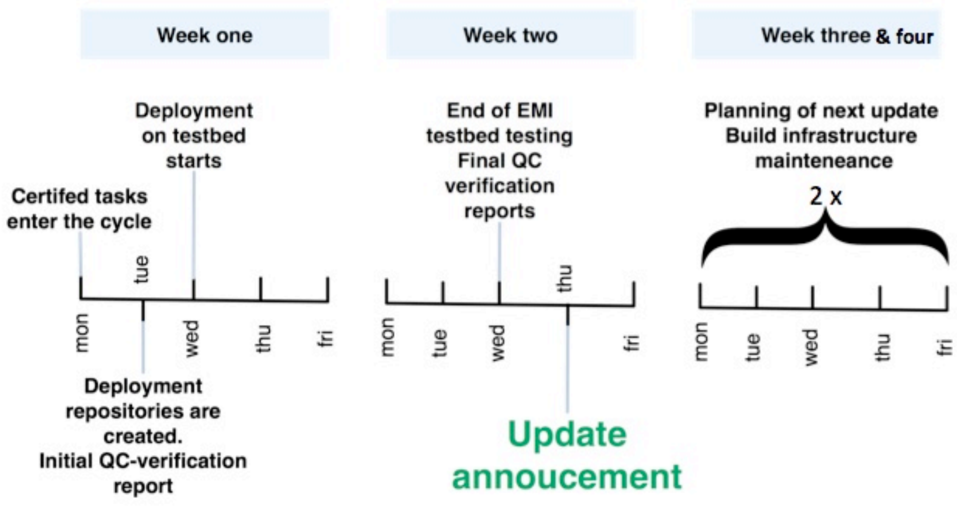

Week one
Week two
Week three & four
Deployment on testbed starts
Certifed tasks enter the cycle
End of EMI testbed testing Final QC verification reports
Planning of next update Build infrastructure mainteneance
2 x
mon
tue
wed
thu
fri
Deployment repositories are created. Initial QC-verification report
Update annoucement

Figure 3 The EMI updates release cycle

The natural output of the maintenance activity is the release of the updated components in production following the *EMI updates release process*. Currently, this release process is organized in four-week cycles.

On Monday of the first week, the tasks in the release tracker that are considered *certified* (i.e., ready to be released) by the PTs enter the cycle. The SA2 Quality Control (QC) team starts the verification for these components, which are then deployed on the EMI testbed [11] for additional integration testing. After one week of testing on the testbed, the products that satisfy the EMI production release criteria [12] and pass the QC verification are released into production. On Thursday of the second week the EMI release manager announces the new release using appropriate communication channels (e.g., dedicated announce mailing lists, EMI web site). The last two weeks of the cycle is dedicated to the planning the next release, by assessing submitted Request for Changes (RfCs) in the context of the PTB or EMT weekly meetings and preparing the build and release tools for the next cycle.

EMI Testing Infrastructure

2.1 How to harmonize 56 products into 1 Release?

In EMI decentralized software development model, different PTs are responsible for testing and certification of one or more software components. Once the certification and testing of the product functionalities in insulation has been successfully carried out, all release candidate components are deployed in the EMI central testbed. This central testbed represents the first point of contact among different products and it is the place where cross-product functionality is checked. The main goal of this testing phase is to harmonize into a single release the strongly inter-dependent products coming from parallel certification paths.

2.2 Inter-component testing testbed

As mentioned in Section 1, EMI project periodically releases updates for its software products: monthly minor release updates and yearly major releases. As shown in Fig.4, all component release candidates to enter the next EMI update must pass an inter-component testing certification step, which is performed on EMI central inter-component testing infrastructure.

Inter-component testing is defined as the part of certification of an EMI software product where product functionalities and expected behaviour is tested against other EMI software products interacting with the product considered. As shown in Figure 5, inter-component testing is performed on release candidate components, that is product components that have passed certification tests (with component in isolation) and quality control verification steps. Quality controls verify that candidate components are compliant to release policy that all required test reports in proper format are available and product documentation is compliant to agreed quality standards.

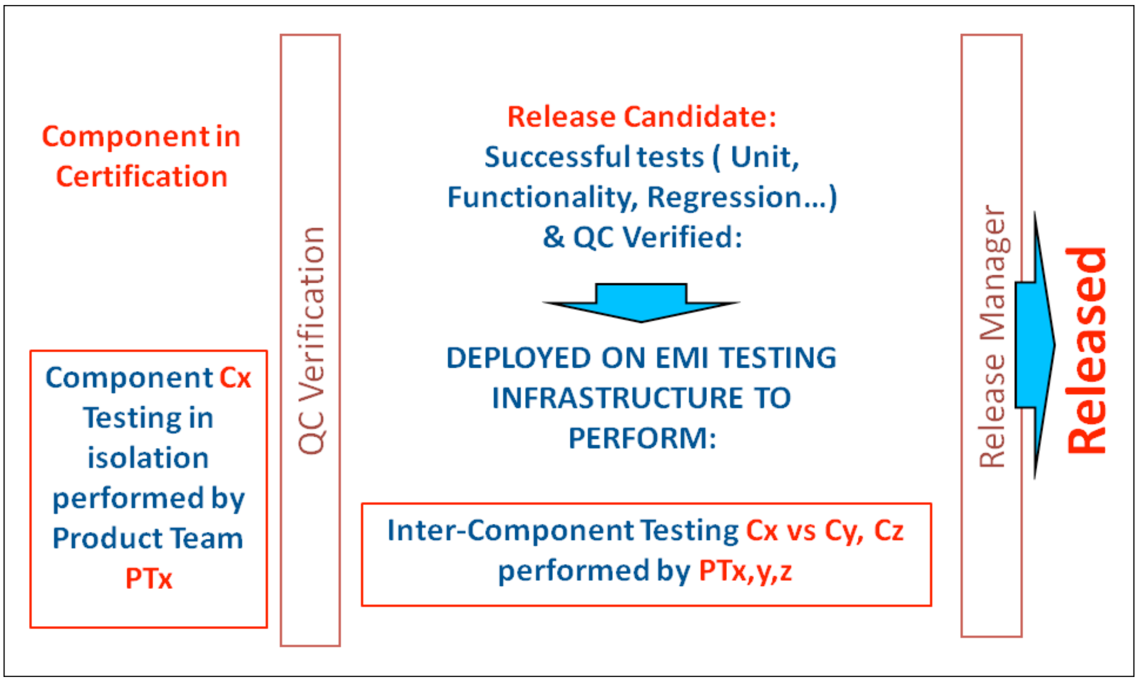

Component in Certification
Component Cx Testing in isolation performed by Product Team PTx
QC Verification
Release Candidate:
Successful tests (Unit, Functionality, Regression…) & QC Verified:
DEPLOYED ON EMI TESTING INFRASTRUCTURE TO PERFORM:
Inter-Component Testing Cx vs Cy, Cz performed by PTx,y,z
Release Manager
Released

Figure 4: ***Final part of product release process: release components are tested for deployment and integration in the Inter-Component Testing infrastructure before being released.***

To understand the requirements on the testing infrastructure deriving from procedure depicted in figure 4 it is necessary to get insight in the evolution process of EMI releases as illustrated in figure 5. EMI releases are divided into annual major releases and monthly minor update releases. Each major relapse can introduce new features or break backward compatibility, while minor release updates must be fully backward compatible. This evolution pattern has some implementative implications for the testing infrastructure that must be setup for the inter-component tests. In fact to let component testers verify the correct integration functionalities of a new Release Candidate (RC) component versus other products production and candidates versions the infrastructure must deploy:

- All product production versions for all supported major releases
- Release Candidate components under test
- All supported platform for both production and release candidate versions.

The resulting infrastructure must be a dynamic snapshot of all released product in both production and release component version for all supported platforms. The current implementation of inter-component infrastructure counts for more than 180 instances providing a snapshot of: i) pre-EMI products from the four partner middleware converging into EMI (most of which test backward compatibility with batch systems and worker nodes still existing in communities sites); ii) EMI-1 Kebnekaise production version release; iii) EMI-2 Matternhorn production versions; iv) instances for tool testing or platform testing. These instances are geographically distributed across 7 EMI participant institutes (i.e. CERN, CESNET, INFN-CNAF, DESY, JUELICH, KOSICE, NIIF). Moreover, a flexible grouping strategy of product instances in *testbed views* is implemented by information system services for production and RC versions for different middleware. Building custom testbed views including both EMI central resources and product team local resources is also possible by composing testbed views as for example by republishing resources in cascade across multiple information system services. As operational facilities we also provide two Virtual Organizations (testers.eu-emi.eu and testers2.eu-emi.eu) and a support unit through GGUS [13] support portal. Detailed documentation on both infrastructural and operational resources can be found at [14].

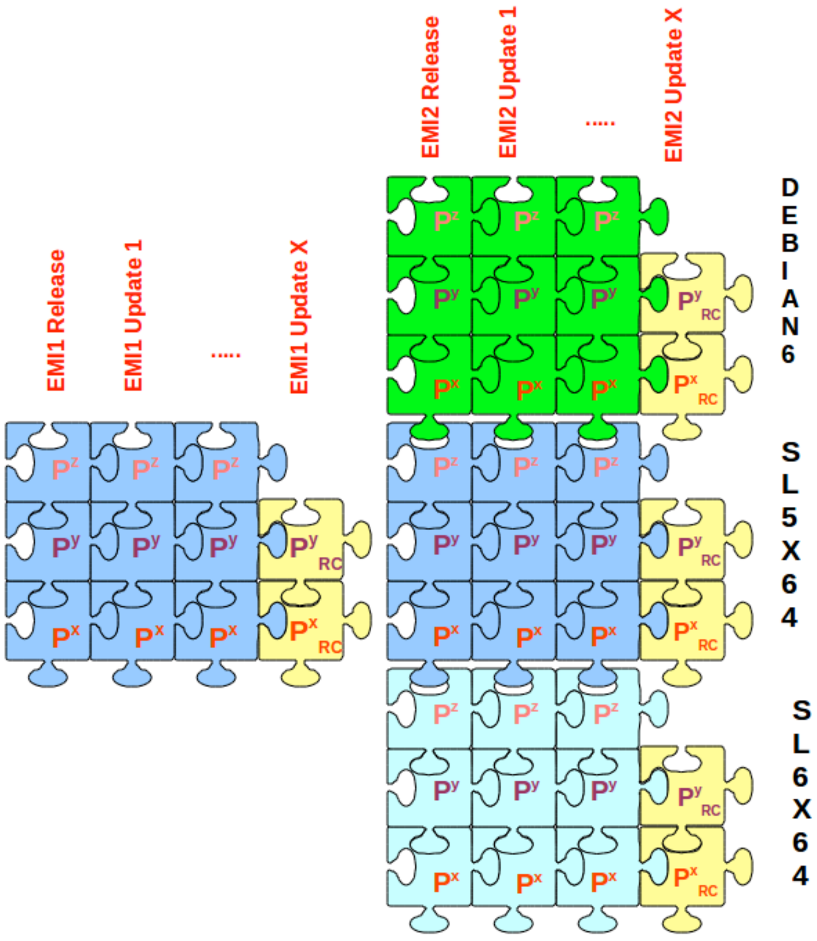


Figure 5: Evolution of EMI release. When new components candidates are ready for release they are tested for integration both with other products release candidates and for production versions to assure backward compatibility.

To reduce the impact of multiplatform deployment (starting from EMI 2) on the testbed dimension, whenever possible we opted for a serialization of components testing on each platform. In particular, the release cycle has been extended so that a three weeks (1 platform per week) deployment and inter-component testing cycle can take place. This allows for virtual machine disks replacements each week without changing services endpoints and cross configuration. Inter-component tests imply cross product team interaction and coordination. To enforce the required cross-product coordination, dedicated time slots for inter-component testing were defined together with a list of mandatory integration tests (25 tests at present time). These mandatory tests are in charge to a responsible product team that must report about them before involved products can be released. The inter-component testing campaign follows the deployment of release components on the EMI testing infrastructure for each platforms in a 3-week cycle with 1 week per supported platform. At present time cross-platform inter-component testing is not performed unless specifically requested.

2.3 Large-scale testbed and preview activities
Production environment is the ultimate realistic test for product quality assessment, but it is generally available for testing when it is too late to fix bugs. Therefore, part of our effort is devoted to involving end-user communities in previews of EMI products or specific scenario testing campaigns. In particular for EMI1 release, the first EMI major releases merging ARC, Unicore, gLite and dCache products, a preview campaign was proposed to user communities to receive early feedback on EMI products. EMI provided deployment expertise and infrastructural facilities such as a common Virtual Organization and Information system services to collect the network of deployed products. The EMI preview campaign involved 16 EMI partners providing useful feedback on documentation. The evolution of this activity was twofold: i) some partners signed a Memorandum of Understanding with EMI projects as part of EMI collaboration program [15] to have access to EMI testing infrastructure, ii) other partners contributed to the EMI large scale infrastructure for acceptance tests.
The goal of large scale infrastructure is to provide a testbed for scalability and interoperability testing of EMI components. To differentiate this activity from common staged rollout activities at production sites, our focus is on early testing of specific scenarios requested either by product team developers (difficult to cover in local testing environment) or by users communities wishing to test real use cases experience on upcoming EMI product versions. The last outreach activity we mention here is the provisioning of an infrastructure for training and dissemination activities on EMI products.

2.4 Testing activities overview and perspectives
EMI central testing infrastructure is involved in many types of release level tests, and in most of bridge activities between single product teams and release management or between EMI developers units and EMI external partners or user community representatives. Among these testing activities we mention:

1. Product deployment tests for most common scenarios;
2. Inter-component tests. Other than the 25 mandatory tests agreed across involved product teams, other tests are regularly performed as part of certification from each product team;
3. Feedback provisioning on product documentation;
4. Testing of new EMI version of SAM-NAGIOS probes, in collaboration with EGI SAM-NAGIOS development team;
5. Specific products migration paths required by user communities (ex. gLite -> EMI migration for WLCG community);
6. Release level bug fix and or workaround verification;
7. Repository functionality testing;
8. Large scale testing activities. Examples fo these activities are: i) the scalability testing of new EMI registry service (EMIR); ii) stress tests on ARGUS centralized authorization service when interacting with many computing elements or worker nodes at several sites; iii) the planned IPV6 assessment test on EMI, HEPiX [16] and EGI testing infrastructures.

First two types of tests are performed at each release update cycle (15 EMI 1 Updates delivered, 1 EMI 2 Release), while other tests are not periodic.
Focusing on perspective activities the main objective of future effort is on increasing automated testing coverage. It is worth to note here that we deal with a very fragmented reality in EMI product testing approaches. In fact the achieved standardization of 4 pre-existing middlewares into one EMI release, mainly regarded building and release process. On the other hand, product testing implementation and tools are still very dishomogeneous and difficult to change over time.

Therefore our strategy will focus on making the testing infrastructure more flexible and dynamic so to facilitate its exposure to automatic testing frameworks used by partner communities, developers and volunteers end users. This will be achieved by automating deployment and configuration, in addition to the modularity and flexibility provided by the *views* implemented by nested level of registry services.

Type of Activity	Quantitative Indicator	Description	Notifications to Developers / Release Manager
Minor Release Update Cycle - Deployment tasks	33 Cycles 155 Tasks	• Product clean installation • Product update deployment • Service reconfiguration • Functionality checks	• 155 Test Pass/Fails feedback • Feedback to RM for known issues reporting • 18 Minor problems notifications
Large Scale Testbed	2 Campaigns 8 Tasks 21 Deployment Scenarios webpages	• Testbed setup • Functionality checks • Demo/Tutorial activities	• Feedback to Developers on documentation • Feedback to quality assurance on documentation policies: new administrator's guide template designed to convey all needed information for product deployment in common production scenarios
Workaround Verifications	11 Tasks	Release level workaround efficacy testing on testbed	
EMI1 Major Release Candidate Version Deployment	171 Tasks over 3 Release Candidate cycles	• Product clean installation • Service configuration • Documentation Feedback • Functionality checks	• 17 blocking problem notifications
EMI2 Major Release Candidate Version Deployment	~ 400 Tasks over 3 Release Candidate cycles	• Product clean installation for 2 platforms • Product Update from EMI1 • Service configuration • Documentation Feedback • Functionality checks	• 51 blocking problem notifications • Feedback to RM for known issues reporting
Incident reports	23 Tasks	Single or Cross-product problems requiring some PT/Testbed staff co-debugging/investigation. They may result in bugs.	• 23 Problem notifications

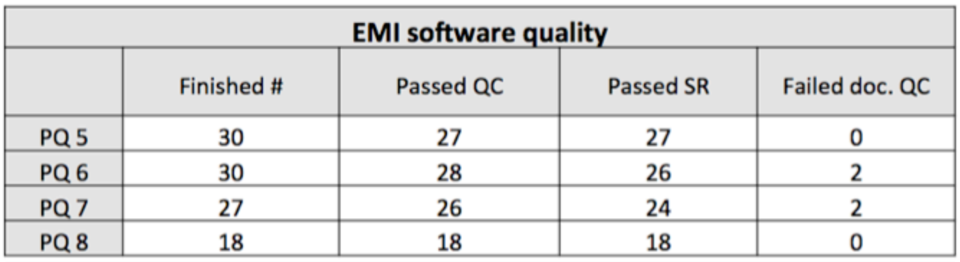
Table.1 Statistics on EMI testing infrastructures activities impact on products quality.

3. Impact on EMI product quality

The described release process and the range of test and validation activities performed on EMI testing infrastructures have the ultimate goal of increasing products quality. Quality improvement in this context refers to: i) products with less defects or early detection and quicker fix of those defects ii) products better matching user communities expectations.

As indicators of the impact of adopted procedures and central testing practices on the two quality

objectives above we report in Table.1 some statistics of the EMI testing infrastructures activities and related visible outcome on product teams or EMI release manager. The second column in Table.1 report some quantitative indicator of the amount of tests / activities performed while the last column reports the outcome of this activity. Notice that last column does not report the commonly expected test passed notifications to release manager.
Measuring the impact of release process on whether EMI products better match user communities expectations is much harder and closer to a customer satisfaction survey than to the scope of this paper. However we can report EMI products quality improvement trend as observed by one of our main customers, EGI, that has evaluated all EMI released products through the EGI software provisioning process [17] reaching a 100% acceptance of EMI products into UMD releases in the last period (see fig. 6)



EMI software quality				
	Finished #	Passed QC	Passed SR	Failed doc. QC
PQ 5	30	27	27	0
PQ 6	30	28	26	2
PQ 7	27	26	24	2
PQ 8	18	18	18	0

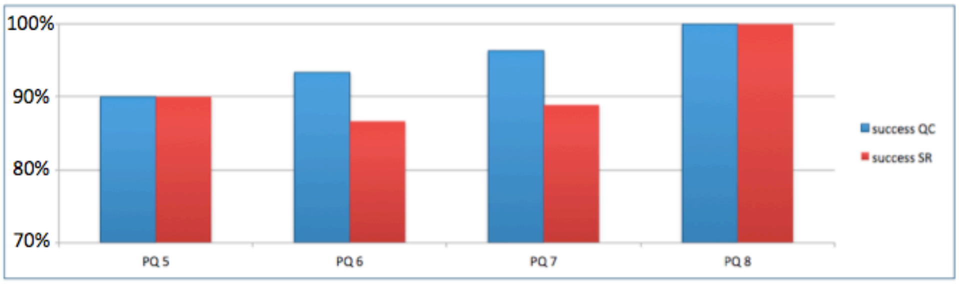


Figure 6: EGI – EMI technology provider increased performance (Source: TCB-11, http://go.egi.eu/TCB-11)

4. Conclusion

In this paper, the EMI project release process and the central testing infrastructure activities and facilities were presented. EMI release process has been designed, implemented and optimized to assure an improving level of quality of all EMI software products over evolution cycles attempting to match new end-user communities' requirements.
EMI testing infrastructure and activities play a key role for the harmonization and interoperability assessment of the EMI sixty products. This centralized infrastructure allows for continuously checking inter-component functionalities for all supported releases and platforms in most common deployment scenarios. Also a large-scale infrastructure and a program of bridge activities with end-user communities have been implemented to enhance the coverage of possible usage scenarios for tested product.

5. References

[1] European Middleware Initiative (EMI) http://www.eu-emi.eu/
[2] ARC Project http://www.knowarc.eu/
[3] gLite Project http://www.glite.eu/
[4] UNICORE Community http://www.UNICORE.eu/
[5] dCache Project http://www.dcache.org/
[6] European Grid Infrastructure http://www.egi.eu/
[7] UMD release http://repository.egi.eu/category/umd_releases/

[8] PRACE http://www.prace-project.eu/
[9] The ETICS build system https://twiki.cern.ch/twiki/bin/view/ETICS/WebHome
[10] EMI Configuration & Integration policy https://twiki.cern.ch/twiki/bin/view/EMI/EmiSa2ConfigurationIntegrationPolicy
[11] EMI Testbed https://twiki.cern.ch/twiki/bin/view/EMI/TestBed
[12] EMI Production Release Criteria https://twiki.cern.ch/twiki/bin/view/EMI/ProductionReleaseCriteria
[13] GGUS: www.ggus.org/
[14] EMI Integration Testbed https://twiki.cern.ch/twiki/bin/view/EMI/TestBed
[15] EMI collaboration program https://twiki.cern.ch/twiki/bin/view/EMI/EmiCollaborationPrograms
[16] HEPiX project https://www.hepix.org/
[17] EGI – Technology Provider Performance: https://indico.egi.eu/indico/materialDisplay.py?sessionId=15&materialId=0&confId=960

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