EGI-InSPIRE

SOFTWARE PROVISIONING PROCESS

EU MILESTONE: MS512

Document identifier:	EGI-MS512-SA2.2.doc
Date:	**26/07/2012**
Activity:	**SA2**
Lead Partner:	**CESGA**
Document Status:	**FINAL**
Dissemination Level:	**PUBLIC**
Document Link:	https://documents.egi.eu/document/1135

Abstract

This document describes the process by which components will be deposited in the EGI Software Repository, mainly by external software providers, processed and released for deployment into production. This document is a revision of the MS508 milestone. It describes the assessment process and the criteria that will be applied to software components and outlines some of the component specific tests that may be applied as part of the software validation process.

I. COPYRIGHT NOTICE

II. DELIVERY SLIP

	Name	Partner/Activity	Date
From	Álvaro Simón	CESGA / SA2	24/5/2012
Reviewed by	**Moderator:** Andrea Ceccanti **Reviewers:** Michel Jouvin	INFN CNRS	13/7/2012
Approved by	**AMB & PMB**		26/7/2012

III. DOCUMENT LOG

Issue	Date	Comment	Author/Partner
1	16/05/2012	ToC adapted from previous milestone	Enol Fernández / CSIC (TSA2.2) Álvaro Simón / CESGA (TSA2.3)
2	24/05/2012	Quality Criteria, ToC changes	Enol Fernández/CSIC (TSA2.2)
3	25/05/2012	Stage Rollout, Software provisioning changes	Mario David/LIP (TSA1.3)
4	27/05/2012	Verification process changes	Álvaro Simón/CESGA (TSA2.3)
5	01/06/2012	UMD section 5	Kostas Koumantaros/GRNET (TSA2.4)
6	08/06/2012	UMD section update	Kostas Koumantaros/GRNET (TSA2.4)
7	12/06/2012	Document comments, Section changes,	Michel Drescher/EGI
8	12/06/2012	Section changes, QC section changes.	Enol Fernández/CSIC (TSA2.2)

9	09/07/2012	Comments from external reviewers	Mario David/LIP (TSA1.3) Alvaro Simon/CESGA (TSA2.3)
10	23 /07/2012	Final comments from external reviewers	Mario David/LIP (TSA1.3)

IV. APPLICATION AREA

This document is a formal deliverable for the European Commission, applicable to all members of the EGI-InSPIRE project, beneficiaries and Joint Research Unit members, as well as its collaborating projects.

V. DOCUMENT AMENDMENT PROCEDURE

Amendments, comments and suggestions should be sent to the authors. The procedures documented in the EGI-InSPIRE "Document Management Procedure" will be followed:
https://wiki.egi.eu/wiki/Procedures

VI. TERMINOLOGY

A complete project glossary is provided at the following page: http://www.egi.eu/about/glossary/.

PROJECT SUMMARY

To support science and innovation, a lasting operational model for e-Science is needed − both for coordinating the infrastructure and for delivering integrated services that cross national borders.

The EGI-InSPIRE project will support the transition from a project-based system to a sustainable pan-European e-Infrastructure, by supporting 'grids' of high-performance computing (HPC) and high-throughput computing (HTC) resources. EGI-InSPIRE will also be ideally placed to integrate new Distributed Computing Infrastructures (DCIs) such as clouds, supercomputing networks and desktop grids, to benefit user communities within the European Research Area.

EGI-InSPIRE will collect user requirements and provide support for the current and potential new user communities, for example within the ESFRI projects. Additional support will also be given to the current heavy users of the infrastructure, such as high energy physics, computational chemistry and life sciences, as they move their critical services and tools from a centralised support model to one driven by their own individual communities.

The objectives of the project are:

1. The continued operation and expansion of today's production infrastructure by transitioning to a governance model and operational infrastructure that can be increasingly sustained outside of specific project funding.
2. The continued support of researchers within Europe and their international collaborators that are using the current production infrastructure.
3. The support for current heavy users of the infrastructure in earth science, astronomy and astrophysics, fusion, computational chemistry and materials science technology, life sciences and high energy physics as they move to sustainable support models for their own communities.
4. Interfaces that expand access to new user communities including new potential heavy users of the infrastructure from the ESFRI projects.
5. Mechanisms to integrate existing infrastructure providers in Europe and around the world into the production infrastructure, so as to provide transparent access to all authorised users.
6. Establish processes and procedures to allow the integration of new DCI technologies (e.g. clouds, volunteer desktop grids) and heterogeneous resources (e.g. HTC and HPC) into a seamless production infrastructure as they mature and demonstrate value to the EGI community.

The EGI community is a federation of independent national and community resource providers, whose resources support specific research communities and international collaborators both within Europe and worldwide. EGI.eu, coordinator of EGI-InSPIRE, brings together partner institutions established within the community to provide a set of essential human and technical services that enable secure integrated access to distributed resources on behalf of the community.

The production infrastructure supports Virtual Research Communities (VRCs) − structured international user communities − that are grouped into specific research domains. VRCs are formally represented within EGI at both a technical and strategic level.

VII. EXECUTIVE SUMMARY

EGI-InSPIRE Software Provisioning Process has experienced several changes and improvements in the last project year. This document is a revision of the MS508 milestone. It covers the main issues found in the provisioning process during the last months and the actions taken to mitigate their impact. EGI will initially use software components provided by different Technology Providers like the European Middleware Initiative project (EMI) or the Initiative for Globus in Europe (IGE) project, but more Technology Providers (TP) products will be included in a near future. The inclusion of new TPs and the requirements from user communities (to include the new software into EGI Unified Middleware Distribution in a short time) have led some major changes. This document describes in detail the technical and workflow changes applied during PY2 by EGI SA2 and SA1 teams. The main PY2 achievements for EGI Software Provisioning are listed below:

- EGI Quality Criteria (QC) was aligned to UMD Capabilities: 100% coverage through 204 quality criteria
- QC change management: 6 monthly revisions, 2 public reviews per revision.
- The new verification SA2 testbed is a private cloud based on OpenNebula 3.2 framework (supports multi-OS deployments, SL5, SL6 or debian6).Verification process is a community effort: Currently there are working actively 16 verifiers, from 12 different institutes and 7 countries.
- During PY2 7 UMD releases were published. 104 updates for 57 products (EMI, IGE).
- Development of several tools to streamline the Software Provisioning Process.
- New Early Adopters have committed to more products (for instance, services based on Oracle database backend) into UMD.

TABLE OF CONTENTS

1 INTRODUCTION

This document describes the process by which components from the software providers are included in the UMD software repository [R 3] and proposes a set of changes aimed at improving the current workflow. The software provisioning process was detailed in MS409 (TSA1.3) [R 1] and MS508 (TSA2.3) [R 2] and has been technically implemented and proven to work during PY2, ensuring the software meets the defined acceptance criteria, and is made available to the staged rollout process before deployment into production. During the last year, EMI 1 and IGE 1 products and updates have gone through the process and have been included in UMD. Also SAM/Nagios updates and IGTF CA releases undergo the same process and are available in the EGI repositories.

The document first describes the current provisioning workflow and the changes implemented are described in Section 2. The following Sections describe the updates from previous milestones to the stages of the provisioning workflow: Quality Criteria in section 3, Verification process in section 4, Staged Rollout in section 5 and the provisioning infrastructure in section 6. Finally metrics and conclusions are drawn in Sections 7 and 8 respectively.

2 SOFTWARE PROVISIONING PROCESS IN EGI

The software provisioning workflow is the process by which software products are included in the UMD software repository [R 3]. All software, either developed by external Technology Providers, such as EMI, IGE, SAGA and StratusLab, or internally within EGI-InSPIRE, such as the operational tools, goes through the process before being released in production. A detailed description of the workflow can be found in MS409[R 2] and MS508[R 1]. Figure 1 shows the main stages of the process:

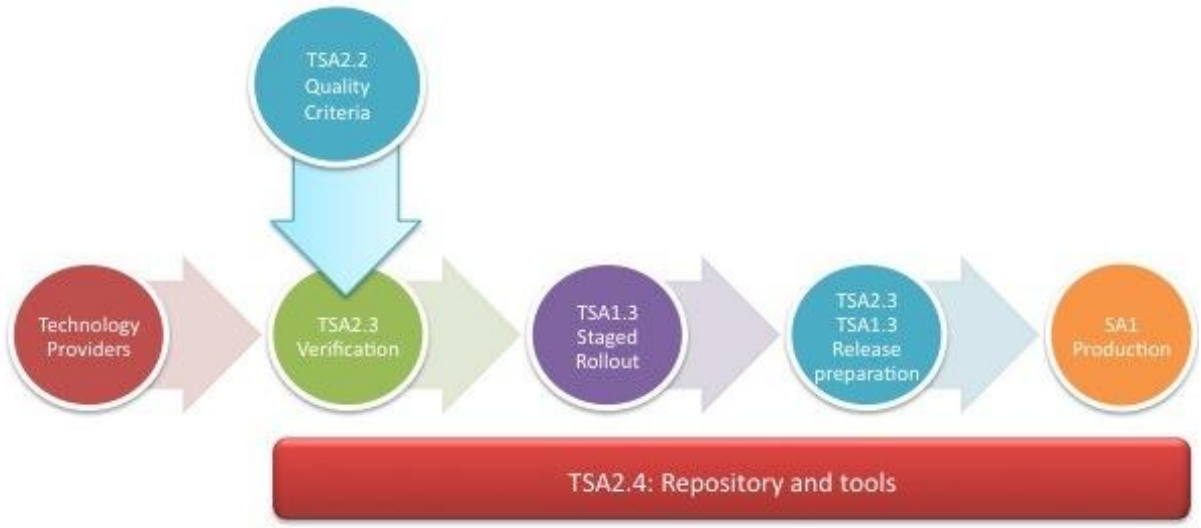

Figure 1. The Software Provisioning Workflow in EGI.

The Technology Providers start the process by submitting information about a new release into GGUS. The products are verified by the TSA2.3 following the Quality Criteria defined by TSA2.2. After successful validation, the products undergo the Staged Rollout phase by TSA1.3. The UMD release candidate is prepared by TSA2.3 and TSA1.3 teams and announced for production. All phases of the process are supported by the repository and tools provided by TSA2.4.

The GGUS ticket created by the technology provider contains the full description of the release in an xml file called "release.xml". This file contains, among other information, the software (meta-packages), dependencies and documentation for each product. The release.xml file allows the automatic management of dependencies and independent product verification and staged rollout phases. This is accomplished through the creation of independent repositories for each product. In the final stage, the independent repositories are merged again into the UMD production repositories, a "base" repository for the initial UMD release and an "updates" repository for the subsequent updates.

Some assumptions, facts and issues with the current process are given below:

- The process assumes that the technology provider produces the "release.xml". The process described here is triggered by the availability of this file. External entities were required to produce it but EMI will not be providing the "release.xml" for EMI2 onwards.

- The provisioning of such file containing full information about a release, both documentation and packages, by the Technology Provider was justified by the fact that it's the entity with best knowledge for such task.
- EGI TSA2.3 and TSA1.3 have, during the past year, acquired sufficient know-how to be able to produce the required information for each release.

- The products and components are publicly available in the technology provider's repositories, the moment they do the release, while in general, it takes some time to produce the "release.xml". The average time from including a given update in the EGI software provisioning to the process completion is about 1 month. Moreover only the components that underwent the full cycle with success will be included in UMD.

2.1 Changes to the SW provisioning workflow

EMI aims to distribute the software through the EPEL (for RHEL and Fedora distributions) and Debian repositories. EMI is already distributing some packages in EPEL. IGE is also distributing the Globus packages through the EPEL repositories. This is part of EMI and IGE strategy for future sustainability plans, towards an open community supported software.

The changes described next, are some of the steps being taken by EGI, to provision the UMD software taking into account the TP plans described above as well as towards a simplification of the process. More detailed justification is given below. The current SW provisioning workflow is described in Figure 2 (see MS508 [R 1] section 4 for more info):

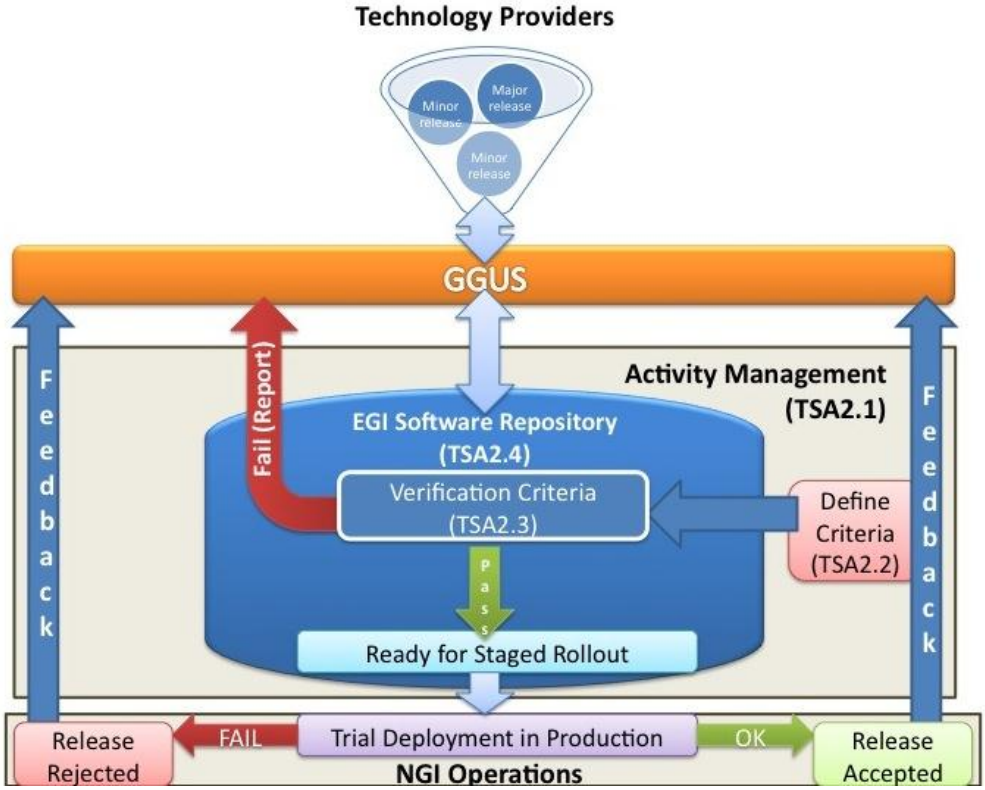

Figure 2 - Overview of the software provisioning workflow

- When a new software release is available, its Technology Provider (TP) notifies EGI through GGUS ticket system.
    - **Change:** SA2.3 and SA1.3 are notified when the technology providers produce a release or update. This done in periodic meetings where technology providers representatives, gives a status of future updates and release schedules.
- **Phase 1 – Delivery**: TP produce a *"release.xml"* file containing the list of the new products and its dependencies.
    - **Change:** SA2.3 and SA1.3 produce the *"release.xml"* and submit it to the "sa2-umd-rel" RT queue.
- The previous action triggers the "bouncer" tool to split the "sa2-umd-rel" into several "Product-Platform-Architecture" (PPA) "sw-rel" child tickets.
- The new release is split in several PPAs and pulled in to the "**unverified**" repository.
    - **Change:** The new packages are pulled into the "**untested**" repository.
- Each "sw-rel" ticket tracks the three remaining phases of the SW provisioning: Verification, Staged Rollout and Publication.
- **Phase 2 – Verification**: starts when the new PPA RT tickets are created and appear as "**unverified**" state.
- **Change:** Verification officially starts the moment the "sw-rel" tickets are created and the packages are in the "**untested**" repository. It doesn't mean that the actual work starts since this will depend on the availability of the verifiers. Verifiers assess all the Quality Criteria to certify the new product.
    - **Change:** Verification concentrates on the documentation, release notes, and what special bugs or vulnerabilities that the Early Adopters performing Staged Rollout, should pay special attention to.
    - **Justification:** lower the load of the Verifiers. More thorough tests are, anyway performed by the Early Adopter teams.
- Verifiers write the verification report and publish it in the EGI docDB with links to the corresponding RT ticket.
- Verifiers and Early Adopters use different volatile repositories tracking each product's packages during software provisioning process:
    - **Change:** Setting the ticket field "**RolloutProgress**" from "In Verification" to "StagedRollout" triggers the movement of packages of a given product from the "**untested**" repository to the "**testing**" repository.
    - **Justification:** The two repositories mentioned in the previous bullet are fixed, publicly available and contain all packages (all products) that are under a given phase of the process. These repositories are disabled by default but allow immediate availability of the new releases in the UMD area. Sites can have access to it if they wish to. Removes the need to configure other repositories to get access to the newer versions of the software.
- **Phase 3 - Staged Rollout**: starts after the previous step is completed.
- The Early Adopters are notified that a new product is ready for Staged Rollout.

- **Changes:**
  - Selecting a list of Early Adopters (EAs) from "sw-rel" ticket field "EATeams" performs this action. This field has been imported from the "staged-rollout" RT queue.
  - EA teams may choose more freely, than in the current process, when to do a Staged Rollout of any given component.
  - One advantage of having a fixed, publicly known "testing" repository containing all products, is that any site can enable that repository to deploy and perform tests, even if not officially committed has an EA team.
  - The RT queue "staged-rollout" has been already deprecated and the necessary custom fields in this queue have been merged or added to the "sw-rel" queue. This is already a simplification, allowing a closer communication between the Verifiers and the Early Adopters.
- **Justification**: simplification of the workflow as well as closer contact and communication between Verifiers and Early Adopter teams.

- When an EA finishes the Staged Rollout, it will write a report and attach it to the respective RT ticket.
- When all Staged Rollout reports are collected for a given product, the ticket field "**RolloutProgress**" should be changed from "StagedRollout" to "UMDStore".
- **Change:** The previous action triggers the movement of the corresponding product packages from the "testing" repository to the "candidate" repository.
- UMD releases depend on TP releases.
  - **Change:** EGI will do UMD releases or updates every quarter containing all products that are in the "candidate" repository at the time of the release.
  - **Justification:** Some sites and user communities expect some stability and don't want frequent updates to the middleware. On the other hand, sites or user communities that want or need newer versions of the middleware may enable the "untested" or "testing" repository to get them, without the need to configure repositories outside the UMD area.
- The "candidate" repository is freezed 1 week before the planed UMD release date.
  - This is the only period where this repository is freezed and no packages may be moved here.
  - Even if staged rollout activities occurs and finishes during this period, the actual movement from the testing repository will have to wait until the current UMD release is finished.
- At any given time, SA2 or SA1.3 may decide to schedule a release containing urgent updates or security vulnerability fixes. The urgency and criticality of the update is discussed with the Software Vulnerability Group (SVG) in the case of security vulnerabilties.

3 QUALITY CRITERIA

The software verification process is based on the Quality Criteria defined by the SA2.2 Task. These Quality Criteria are classified into generic criteria, i.e. criteria that any software product to be verified must meet, and specific criteria, i.e. criteria valid for a particular software product only. The main artefacts produced by the Quality Criteria Definition Task are the Quality Criteria Documents. These documents contain the criteria and are available in two different formats:

- Quality Criteria documents are organized according to the UMD capabilities [R 4] as described in the previous milestone [R 1]. These documents cover the complete list of UMD capabilities that are currently defined.
- Per-Product Quality Criteria documents. For each product undergoing the verification process, a document containing only the applicable Quality Criteria is provided to aid the verification process. New products entering into the software provisioning process trigger the creation of a new document.

Quality Criteria definition is a continuous process driven by the requirements of users and operation communities, however the verification of new software releases is preformed against stable Quality Criteria documents that are updated every 6 months.

Between these releases, SA2.2 produces public draft versions available for review by the community. This allows the verification team to plan ahead the effort required for each product in the verification process and makes Technology Providers aware of quality assurance process. At any given time there are 3 different possible states for the Quality Criteria documents:

- **FINAL**; documents are actively used for verification of software products, every 6 months a new release is created.
- **DRAFT**; documents that are in preparation and will be used in the future for verification.
- **DEPRECATED**; for documents that are no longer used in verification.

Any change introduced in the Quality Criteria by the definition team is recorded in the release notes of the criteria documents available in the wiki [R 5]. The changes are triggered by one of the following sources:

- Requirements from User Community.
- Requirements from Operations Community (especially software issues or bugs found in production).
- Deficiencies in criteria found in Verification or Stage Rollout.
- Recommendations and issues found by the Software Vulnerability Group.
- Analysis of reference implementations of UMD Capabilities defined in the Roadmap.
- Review and analysis of feedback from Technology Providers.

Each criterion includes also a *Related Information* field that includes any link to direct source of change for the criterion (e.g. RT ticket with user requirement) and a *Revision Log* field that records the historic information about the changes in the criterion.

The wiki also includes pointers to the current FINAL version of the documents [R 6] and the current DRAFT [R 7]. A detailed roadmap with estimated dates for the availability of reviews and final versions for the next two releases is also available.

4 SOFTWARE VERIFICATION

The main objective of the TSA2.3 is to verify the quality of the software provided by the TP based on the EGI Quality Criteria designed by TSA2.2 team (section 3). A detailed description of the verification process is included into MS508 document [R 1]. This section will describe the main changes introduced during PY2.

4.1 Verification Effort

In order to improve the efficiency of the process of products undergoing the EGI software provisioning workflow, some changes have been implemented in the verification level of testing. These changes stem from the fact that different types of releases should have different levels of testing, thus different levels of effort by the Verifiers. Currently, all new TP products are assigned to one of the following release types:

- **Major releases:** May not be backwards compatible (due to OS or middleware major changes). In this case, verifiers must actively assess all assigned QCs (from the QC verification product templates), test the new features and install the new product from scratch (or upgrade if it's supported by the product).
- **Minor releases:** Are backwards compatible. In this case, verifiers only check QCs affected by update changes (must be described in TP release notes). Verifiers must enter the results of these QC checks into the QC_Verification_Template excel file, the rest of QCs should be left in blank or just add a comment: "Minor release, this QC was already verified". Verifiers still perform a package update and installation from scratch.
- **Revision releases:** Are backwards compatible. These releases include only bug fixes (without new features or major changes). The verifier only checks new package installation and upgrades, for revision releases no further testing is needed. Optionally verifiers can verify the new bug fixes. Staged rollout is always performed. Exceptions are evaluated in a case by case basis, when there are very urgent fixes or security vulnerabilities.

The release type is assigned to each new PPA. A new RT field "ReleaseType" was created to identify which level of testing is required in each case. The verifier must read this value and act accordingly (it's up to the verifier to identify which QCs must be assessed based on TP release notes).

4.2 Verification Process

The verification process is the same as described in MS508 [R 1]. Only a few changes were included to simplify the current process and to help the verifiers work. A new set of QC verification templates was created based on the newer QCv3 products mapping [R 9]. The verification templates [R 10] are generated automatically using the new QC service mapping which assign specific and generic QCs to each product provided by the TPs. When a new product is included, SA2.2 team only has to include the new mapping, afterwards the new template is generated automatically using a new python script.

The verification templates have not changed, but new QCs where included since MS508. Verifiers must fill two documents when the verification is finished. These documents should be merged into one final PDF Verification Report. The version number represents the QC version used to create these files (Version 3 at the time of writing this milestone).

If the Verifier finds problems or issues, either they are clarified within the ticket by the verification team or, if the problems need interaction with the Technology Provider then a GGUS ticket is opened by the Verifiers with the following characteristics:

- **Priority**: All new GGUS tickets should be created with the default priority, the short description field must begin with "UMD Verification:" and then the description of the issue. (Notification mode should be set to "*On every change"*). Except if the verifier finds a critical or showstopper issue. In this case, the GGUS ticket must be set with a higher priority.
- **Criticality**: The ticket should describe the criticality of the issue in the ticket body. If it's a showstopper or not, or if there are possible workarounds.
- **Routing**: The Deployed Middleware Support Unit (DMSU) will route it to the technology providers as they see fit.
    - The ticket resolution should have the following workflow:
        - For critical or showstopper issues the Verifier should change the RT ticket **RolloutProgress** to the *Waiting for Response* status.
        - All GGUS ticket links must be included into the **RelatedGGUSTickets** RT field. Verifiers should also include links to all GGUS tickets as reply into the verification RT.
        - Once the problem is solved, the **RolloutProgress** must be changed again to "*In verification*".
- **Referencing**: Each GGUS ticket link must be included into the Verification Executive Summary (GGUS tickets section).

4.3 EGI Verification Testbed

CESGA SA2 partner maintains a verification testbed. An OpenNebula cloud framework was updated and included more Virtual Machine (VM) golden copy images to be used in the future (for the upcoming EMI and IGE releases).

Currently there are 3 new images that can be used by the verifiers: SL5, SL6 and debian 6. All the new images were generated from scratch using qcow2 utilities developed by Qemu team [R 11]. Qcow storage optimization is used by the SA2 testbed because it offers the following features: images grow as data is added, and support AES encryption and transparent decompression. VM images are smaller than older raw images and they can be instantiated in few minutes on any host.

The new VM generation mechanism will allow integrate virtualised resources in EGI. SA2.3 team is collaborating with EGI Federated Cloud task force [R22] to provide pre-configured grid service after SA2 verification process (see section 8 for more information).

To support the new EGI VM images provisioning process, the SA2 testbed VM golden copies are generated following the Joint Security Policy Group and StratusLab best practices for creating base images [R 12]. The new procedure also allows the upload of testbed VM images into the StratusLab Marketplace [R 23] after the verification process.

5 STAGED ROLLOUT

The EGI staged rollout is a procedure by which certified and verified updates of the supported middleware are first released to and tested by Early Adopter sites before being made available to all sites through the production repositories.

It permits testing an update in a production environment that is also more heterogeneous than what is possible during the certification and verification phases. It allows for potential issues with an update to be discovered at a small scale and potential workarounds to be added to the release notes. In some cases, an update may be rejected at this stage.

The staged rollout objective is to increase the confidence in the quality of the updates, such that the vast majority of the sites should experience smooth updates. Volunteer Resource Infrastructure Providers (sites) participate in the staged rollout for services that they have a particular interest in and that they already run in production, with the proviso that they may need to debug issues with a particular update and are required to report their findings.

5.1 Second project year

During the 2nd year of the project, the first UMD release has been done plus seven updates. In each update, either new products or updates from two Technology Providers (EMI and IGE) have been through the SW provisioning process.

One of the major challenges regarding the Staged Rollout was to have Early Adopter teams that can cover the majority of products available from the Technology Providers.

The other major challenge comes from the recent release of EMI2, implying that existing Early Adopter teams had to be redistributed between EMI1 and EMI2 products. A campaign as been undertaken to have more teams from the SA1 sites to act as Early Adopters.

To help out the SA2.3 team in the Verification process, some of the more active and experienced Early Adopter teams have been participating in this effort, particularly for either major UMD releases or for specific products where SA2.3 has less experience.

At the time of writing, some of the new products released by both EMI and IGE do not have Early Adopter teams assigned to them yet. SA1.3 has the responsibility to call SA1 as well as the user communities interested in deploying such products in the production infrastructure.

The mitigation of this issue is not easy; it passes always by presented this in the EGI operation meetings, or fora as well as direct contact with sites and user communities. Although slowly, products without EA coverage have been getting commitment from new teams, this was the recent cases of LFC Oracle and FTS. Presently, coverage is missing for following products:

- EMI2:
  - CLUSTER
  - EMIR
  - GLEXEC
  - Pseudonymity
  - WNODES
- IGE:
  - GridSAM
  - OGSA-DAI

6 PROVISIONING INFRASTRUCTURE

The EGI Software Repository Portal provides a unified point of access to the Universal Middleware Distribution (UMD) and operational tools. UMD includes products developed for EGI.eu by external Technology Providers such as EMI [R 16], IGE [R 17], SAGA [R 18] and Stratuslab [R 19]. The EGI Software Provisioning infrastructure and its current status, are described next.

6.1 EGI Software Provisioning process

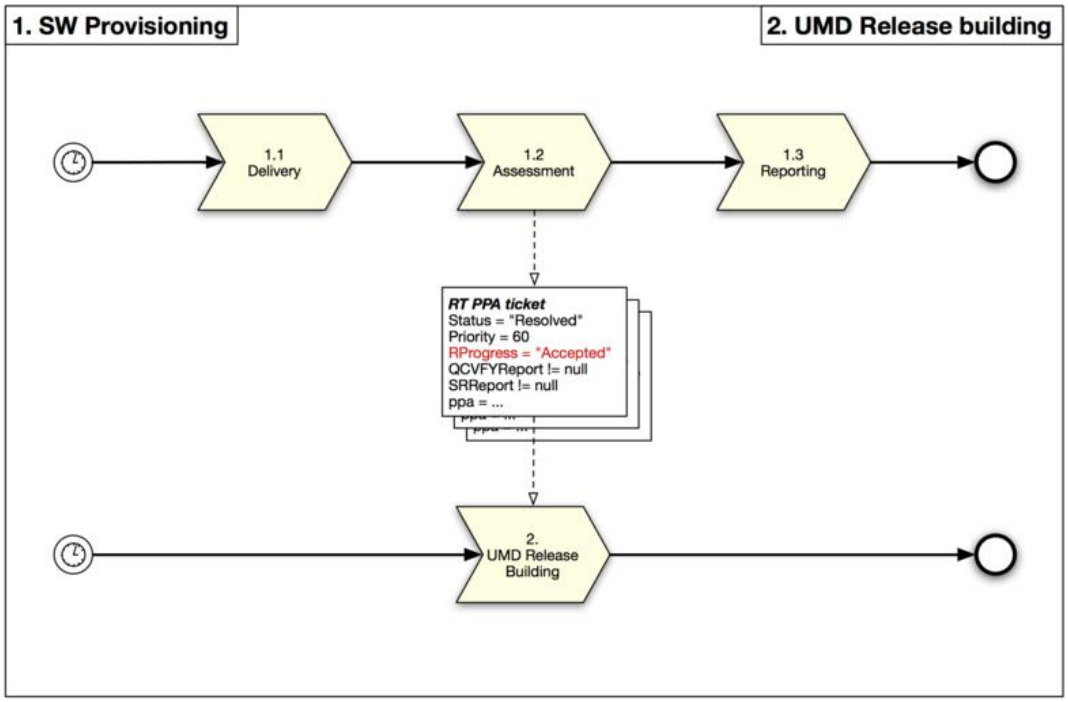

Figure 3 EGI Software Release Process

The architecture of EGI software provisioning tools is described in MS506 [R 15] this paragraph describes only the implementation of the proposed changes in chapter 2 plus the changes made to provide Debian support. As was described in chapter 2, one of the most significant changes in EGI software provisioning process is that the Technology providers will not provide the aforementioned "release.xml" from PY 3 onwards. In order to address this issue TSA2.4 developed a "Release XML editor tool" that allows to construct and validate "release.xml" files. Figure 4 shows a screenshot of the main menu of the tool.

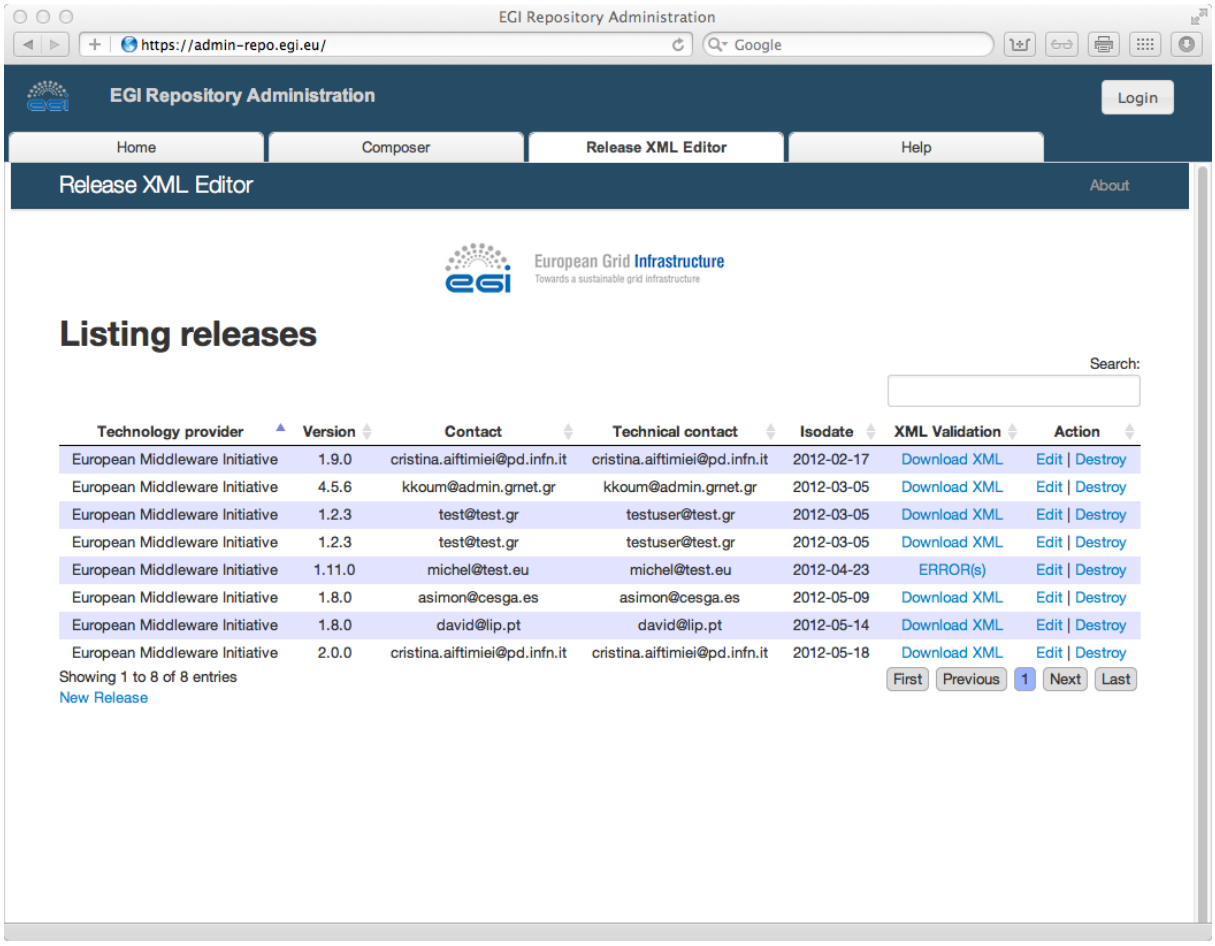

Figure 4 Screenshot of the Release XML Editor

In order to address the proposed changes described in chapter 2, the Repository backend engine was adapted in order to provide also the following repositories:

- **Untested:** includes everything that is currently in the Unverified or In Verification State:
- URL: http://repository.egi.eu/sw/untested/umd/
- **Testing:** includes everything that is currently in the Stage Rollout or UMD-Store State:
- URL: http://repository.egi.eu/sw/testing/umd/
- **Release Candidate:** includes all products that underwent successfully through the Software Provisioning process and are about to be published:
- URL: http://repository.egi.eu/sw/production/umd/candidate/
- **Production:** includes products released to production:
- URL: http://repository.egi.eu/sw/production/umd/

The above repositories follow the <UMD MAJOR>/<OS>/<ARCH> directory structure that allows for more than one UMD Major to be supported at any given time and similarly each UMD Major may support more that one operating system / architecture combination (e.g UMD-2 supports SL5/x86_64, SL6/x86_64 and Debian6/Amd64) based on the main Linux distributions repository organization.

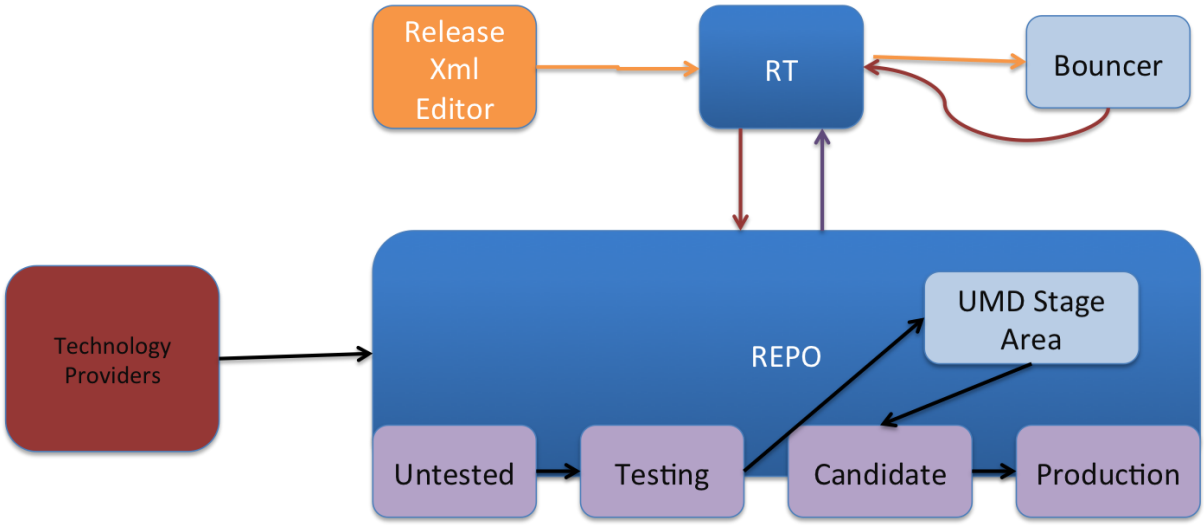

Figure 5 Interactions of EGI's Software Provisioning Tools

In order to add Debian support in EGI software provisioning tool, the following changes where performed:

- Customise the repository backend in order to be able to accept and manage debian packages.
- Create a mechanism (agent) for apt, debian-based, repositories.
- Adapt internal tools in order to be able to construct debian-like UMD releases.
- Add a Debian package dependency resolver.

6.2 UMD Repository Software distribution during PY2

During PY2 the EGI software provisioning tools were deployed in production and where used for the following releases (Figure 6 shows a snapshot of the current status of the Products in the workflow):

- 9 UMD, that include 56 Products and 49 Updates
- 7 SAM Updates
- 9 EGI-IGTF updates

EGI operations have been working on tools to monitor the uptake of UMD/EMI software by the site in production; this can be done using the GOCDB on the one hand and the Top-BDII information system on the other hand, for all services released by EMI technology provider. A first preliminary assessment of the UMD uptake can be found in [R23]

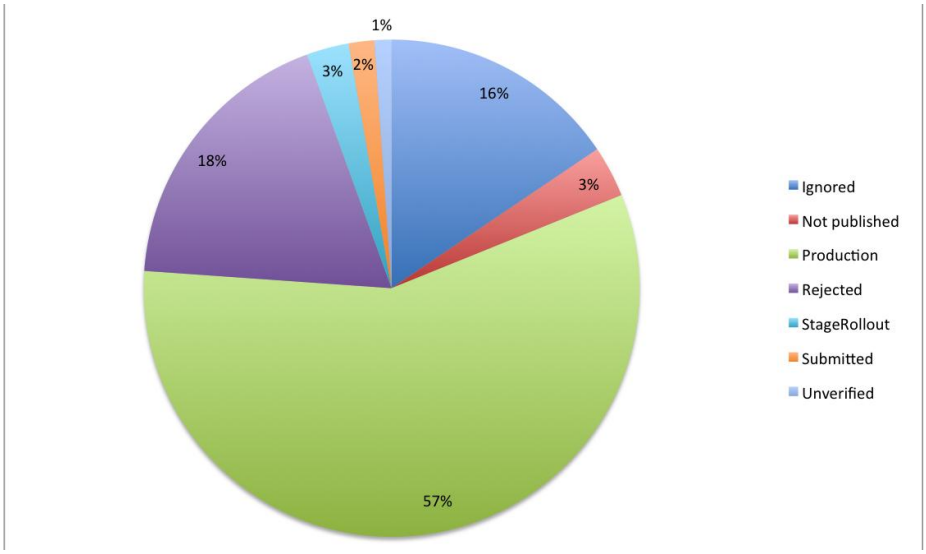

Figure 6 Products Per rollout Progress

Figure 7 shows the usage of the Production repository indicating that sites are migrating to UMD-1 distribution and take the benefits of SA2 software provisioning process.

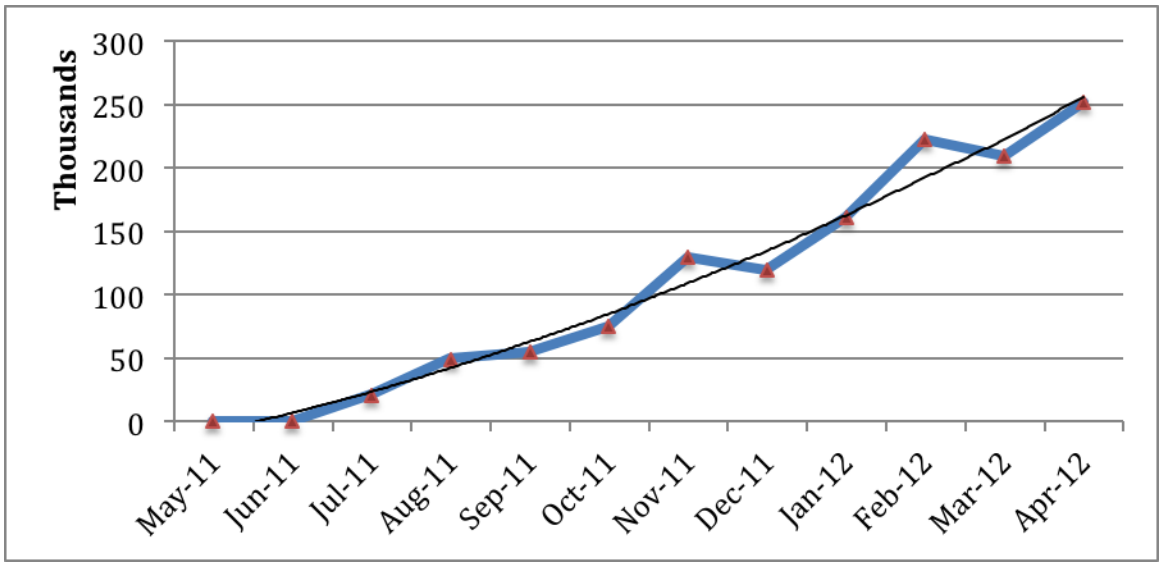

Figure 7 Total Downloads Per Month

7 METRICS

7.1 Quality Criteria

During PY2, SA2.2 has included a new internal metric that reports the number of issues that produce changes in the Quality Criteria, quantifying the work of the criteria definition team. This metric is collected using the detailed change-log of the criteria documents where all the changes and the sources of changes for each criterion are listed and the Revision Log field of the criteria where more detailed information is provided. The collection of this metric serves also for the collection of the other two metrics defined for SA2.2, which did not have any changes during PY2.

7.2 Verification

Metric M.SA2-6 [R14] collects the mean time taken to validate a Product release by the verification team. The changes introduced in the process (see section 4.1) to improve and fasten the software provisioning workflow (see section 2.1), allowed the effort spent during verification to decrease significantly in the last UMD releases.

The collection of M.SA2-6 is performed using the different fields of the RT tickets for the software provisioning process. Each ticket reports the number of hours dedicated to the validation of the software and the release type of the product (major, minor or revision). These fields are processed into an excel file that include all data used to produce the SA2 Verification Metrics [R 8]. This excel file is updated periodically allowing a fine grained analysis of the effort spent in each of product for each type of release.

Figure 8 shows the verification effort during PY2 per type of release. For revision releases, verifiers only check the new changes and the process is finalized in a short time. After UMD major releases most of the TP updates include only minor changes.

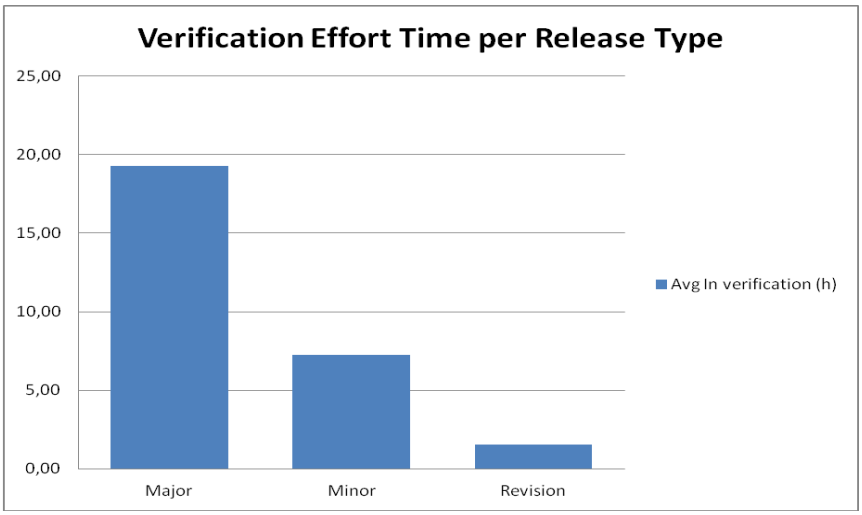

Figure 8 Verification effort time in hours.

Only one major release per year is produced by the TP. This new procedure has many advantages, the verifiers only focus their efforts in the new software changes and the new updates could be in Staged Rollout in a shorten time.

7.3 UMD package downloads statistics system

The UMD package downloads statistics system is comprised of 3 sub-systems:

a) The logging aggregation subsystem, which is based on a syslog-ng implementation. This subsystem collects centrally access logs from all web servers that serve the UMD repository.
b) The log publishing subsystem. This is a workflow that uploads the logs from central system log facility to the user interface database.
c) The user interface subsystem: Is a web application that processes the logs and provides query functionality to the user.

The logging aggregation subsystem is implemented according to the current best practices for central system logging. The central system logger has two destinations for the logs, one for simple files for archival and future processing and one towards the log publishing subsystem.

The log publishing subsystem uses the REST interface of the user interface web application to submit logs to it. In case there is a failure on the submission (either receiving an non-OK reply from the user interface or having a connection time out), the submitted log is written to a separate file for future manual upload.

The user interface consists of four tables.

1. “unprocessed_entries” table: holds the logs as they are submitted from the publishing subsystem. Each entry in this table includes both the reporting web server and the raw message itself. This table allows the publishing and processing systems to work asynchronously and to avoid blocking of publishing operation due to processing of the data. Periodically a script is run on the primary web server which parses the entries of "unprocessed_entries" table and fills in the second table.
2. "log_entries" table: the log entries are split into several fields (timestamp, client IP, requested URL). For each unique client IP that is found in the "log_entries" table an entry on a third table.
3. “ip_metadata” table: is created and filled using a GeoIP database with the relevant information (country, city, ASN).
4. A script aggregates the results from the last two tables (log_entries and ip_metadata) as well as all daily information (by design, this is the smallest time fraction allowed) to achieve better performance. This last table is queried based on the user's filters on the web front end used thereafter for displaying the requested information.

8 CONCLUSIONS AND FUTURE PLANS

The Software Provisioning Process is similar to an ITIL Continual Service Improvement process. It has experienced significant changes in the last year to fix some issues and improve its performance.

During PY2 the interaction with external TPs were coordinated using the GGUS ticketing system. This interaction between the TPs and GGUS system (including *"release.xml"* generation) has introduced a bottleneck into software provisioning process. Besides two new TPs software will be included for PY3 (Genesis II [R20] and QosCosGrid [R21]). The aim of the new proposal and technical changes is to remove this bottleneck or reduce its impact during the software provisioning process. The new UMD2 update and release.xml creation will provide more flexibility.

Another important task for the next months is the collaboration between SA2 and EGI Federated Task Force. During PY2, EGI started a federated cloud taskforce aiming to pave the way to then next generation of its infrastructure based on clouds. SA2 was adapted to the new challenge by offering a VM marketplace and appliance repository based on software developed by the Stratuslab Project [R 19].

EGI's VM marketplace aims to be a central repository for the metadata of the VMs used in the federated cloud infrastructure. The VM marketplace allows more than one appliance repository to be used concurrently thus the plan is to use EGI appliance repo to store only VMs created during EGI's

Resource providers/NGIs and Technology providers should offer their own appliance repositories with the VMs/appliances they wish to offer and upload their metadata in the central marketplace.

9 REFERENCES

R 1	MS508: Software Provisioning Process https://documents.egi.eu/document/505
R 2	MS409 - Deploying software into the EGI Production Infrastructure https://documents.egi.eu/document/478
R 3	EGI Software Repository: http://repository.egi.eu
R 4	EGI Roadmap and Technology: https://wiki.egi.eu/wiki/EGI_Roadmap_and_Technology
R 5	EGI Quality Criteria wiki: https://wiki.egi.eu/wiki/EGI_Quality_Criteria_Dissemination
R 6	Quality Criteria v3: https://documents.egi.eu/document/718
R 7	Quality Criteria v4: https://documents.egi.eu/document/1153
R 8	SA2 Verification Metrics: https://rt.egi.eu/rt/SA2/sa2-sw-rel-verification-metrics.xls
R 9	Quality Criteria Products Service Mapping: https://documents.egi.eu/public/ShowDocument?docid=418
R 10	Quality Criteria Verification Template Reports: https://documents.egi.eu/document/417
R 11	QEMU: http://wiki.qemu.org/Main_Page
R 12	StratusLab VM images best practices: http://www.stratuslab.eu/doku.php/tutorial:baseimagecreation
R 13	StratusLab Marketplace: http://www.stratuslab.eu/doku.php/install:marketplace
R 14	SA2 WP5 Metrics: https://wiki.egi.eu/wiki/InSPIRE-SA2:WP5_Project_Metrics
R 15	MS506: EGI Software Repository - Architecture and Plans https://documents.egi.eu/document/503
R 16	European Middleware Initiative (EMI): http://www.eu-emi.eu
R 17	Initiative for Globus in Europe (IGE) : http://www.ige-project.eu/
R 18	Simple API for Grid Applications (SAGA): http://forge.gridforum.org/projects/saga-rg/
R 19	Stratuslab Project: http://stratuslab.eu/
R 20	Genesis II: http://genesis2.virginia.edu/wiki/
R 21	QosCoGrid: http://www.qoscosgrid.org/trac/qcg
R 22	EGI FedCloud Task Force: https://wiki.egi.eu/wiki/Fedcloud-tf:FederatedCloudsTaskForce
R 23	D4.5 Annual Report on the EGI Production Infrastructure: https://documents.egi.eu/document/1059