# EGI: Managing the Software Process

Steven Newhouse, Francesco Giacomini, Oliver Keeble, Markus Schulz, Aleš Křenek, Achim Streit, Oxana Smirnova, Farid Ould-Saada

22.3.2009

Revision 9.5

# 1 Executive Summary

This document expands on the high-level view provided in the EGI Blueprint to clarify the relationship between the EGI middleware unit and the software providers that it works with to provide a production quality software infrastructure for the NGIs. It covers the relationship between members of staff within EGI.org and its advisory boards in addition to its software providers. The majority of the document covers the maintenance issues related in the work of EGI.org, but information is also provided as to how new components, developed by the software providers, would move into production.

# 2 Introduction

## 2.1 The Need for UMD and its benefits

UMD is an opportunity to consolidate the experience from nearly a decade of European distributed computing middleware developments to simplify for resource providers the provision of e-infrastructure and to provide common access mechanisms for application communities. This consolidation has already been underway for several years enabled through projects such as OMII-Europe and OGF-Europe. The EGI, and its associated funding for middleware, provides an opportunity to accelerate this work.

Work within the scope of UMD includes:

- the maintenance of the currently deployed middleware in production use across Europe that supports a diverse and active application community
- the convergence of widely deployed components from different providers to a single, ideally standardized interface, that will enable communities to utilize services from any European middleware provider
- the rationalization of European middleware components to eliminate duplication where there are components with converged interfaces that are deployed to meet identical use cases
- the continued development of new functionality to meet the evolving needs of its application community and to ensure that Europe continues to be recognized as a leading provider of e-infrastructure software

By the three middleware consortia coming together through UMD to meet the middleware needs of EGI, we expect to:

- have an increasingly stronger view within standards bodies such as the OGF (see the recent activity within the GLUE and PGI working groups)
- provide standard interfaces to common services and provide defined environments through synchronised development activities to provide a coherent architecture
- simplify the life for application developers and users by enabling transparent access to resources regardless of the deployed middleware implementation
- by converging and rationalising the European middleware reduce the long-term software

maintenance costs for the European e-infrastructure

## 2.2 Software activity within the EGI

The software deployed in EGI will come from a number of different software providers. Some software providers are large and currently contribute a significant proportion of the software used in production infrastructures within Europe (e.g. ARC, gLite and UNICORE). Other large software providers will only be contributing a small proportion of their software output to the production infrastructure (e.g. VDT, Globus, ...). A third category is the small software providers which provide vital 'niche' functionality for the production infrastructure (e.g. dCache). EGI needs to form partnerships and define a software release process that encompasses new releases from all of these software providers.

Within the context of EGI, UMD needs to maintain a stable reliable scalable production-ready software base that will be deployed as e-infrastructure within Europe. Alongside the software maintenance activity needed to maintain this software base, there is a need for further software development to enable components to converge towards agreed standards, to work on rationalizing middleware components where there is duplication and meet the needs of new requirements with new functionality.

Generating a clear definition between software maintenance and software development is challenging when the context in which EGI is operating, and the needs of its different communities, are still evolving. Regardless of the source, any software components that expect to be deployed on the production infrastructure will have to enter through the same qualification process as any 'maintained' components, have the same demonstrated quality, and a clear community need. Requirements from the user and operations community will be maintained and used to prioritise maintenance tasks and to inform the wider software development community as to the broader software needs of EGI.

Any new services or significant changes in functionality are seen to be out of scope of the core EGI maintenance model, and will have to be funded through separate projects (either EU or national) or from within the interested communities.

## 2.3 Organisational Structure

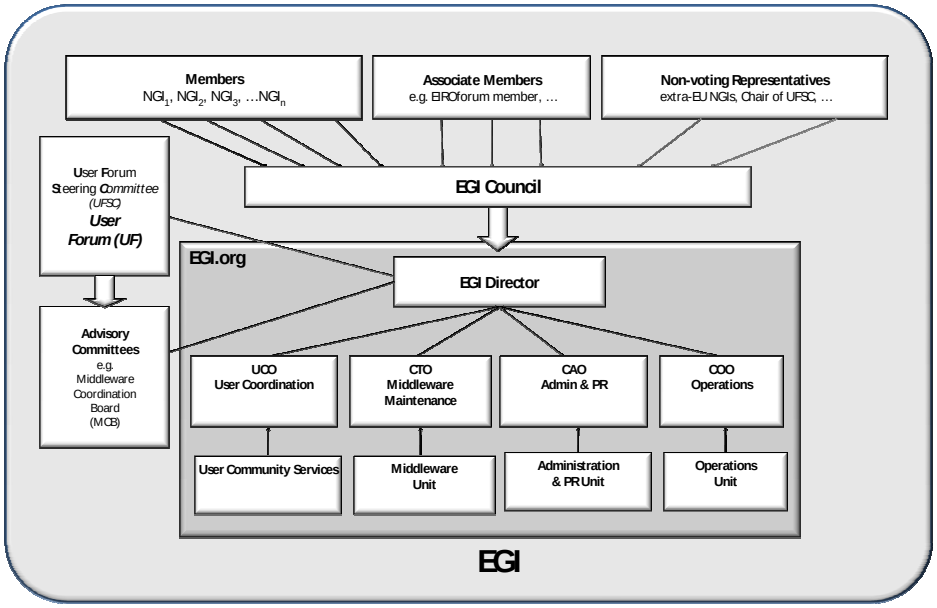


*Figure 1: EGI Organisational structure*

A tentative mapping is made between the major activities defined within this document and the staff tables taken from the EGI_DS Blueprint. This is indicated by [Staff: MU1] – details as to the available staff effort can be seen in the table towards the end of the document.

## 2.4 Definition Used in this Document

MCB: Taken from the EGI_DS Blueprint, the Middleware Coordination Board sets the technical priorities and makes all decisions concerning the maintenance, support and evolution of the middleware deployed on the EGI e-Infrastructure. The MCB is composed of representatives of the following areas, appointed in agreement with the EGI.org management:

- the main middleware developers of the components in use in the EGI e-Infrastructure as the three European Middleware Consortia;
- the operation function representing globally the operational requirements of EGI.org, NGIs and Resource providers;
- the User Community Services (UCS) teams on behalf of the Specialised Support Centres (SSC), representing the various user communities organised in thematic disciplines.

It is expected that the CTO, COO and UCO will attend the MCB as observers and represents their units within the MCB. Representatives of 3rd party software providers might also have observer status.

MU: EGI.org Middleware Unit. An organisational unit within EGI.org consisting of technical personnel with the necessary expertise to manage the middleware development, testing, documentation and integration that will need to take place within the software providers.

OU: EGI.org Operations Unit. An organization unit within EGI.org that is responsible for coordinating the NGI operations.

UCS: EGI.org User Community Services. An organizational unit within EGI.org responsible for coordinating the individual specialized support centres (SSC) and their user communities.

CTO: EGI.org Chief/Central Technical Officer and head of the MU and the main entry point towards the MU.

COO: EGI.org Chief/Central Operations Office and head of the OU.

UCO: EGI.org User Coordination Officer and head of the User Community Services unit.

*Software Element:* A software element may consist of single or multiple software components that are assembled to deliver specified functionality to the MU. This could range from a single service with client or service components such as BDII, or a complete system such as the ARC-CE. As such it forms the basic 'unit' that a site might install onto a machine.

*Software Provider:* In this example the software provider is an engineering team responsible for the complete delivery of a software element. It is expected that UMD components will come from the three largeconsortia (ARC, gLite and UNICORE), other major providers (such as Condor, Globus and VDT) and smaller providers focused on specific components (such as dCache or SRB).

*Software Engineering*: The act of implementing the change request from the MU and in any associated test suite.

*Integration*: The integration of the software element into the deployment infrastructure and deployment environment specified by the MU, i.e. operating system, package manager, etc.

*Software Certification*: This activity ensures that the software element works in the deployment environment without interfering with the running of any other software elements, i.e. is the software fit for release to a production environment.

*Verification*: This activity ensures that the software element satisfies the specification defined by the MU by executing the test plan agreed with the MU.

*Platform*: A combination of operating system, architecture, libraries, software elements and compiler that defines a unique environment that components will be built and certified against.

# 3 What is UMD?

## 3.1 A definition

The Universal Middleware Distribution (UMD) is a collection of compiled software components with integrated configuration from a single repository that add a layer of Grid functionality on top of operating systems (similar in concept to a Linux distribution). As such it is composed of multiple production quality software components originating from different providers each with their own separate software development infrastructure (bug tracking, source code management, testing, mailing lists, etc.). A UMD release has:

- A unique version number defining the set of components and software elements contained within the distribution
- The components are used to install and configure one or more software elements

Each software element in a release has:

- A unique name and version number

- Been integrated and certified to deliver a defined capability allowing its components to be managed and configured as though a single entity
- Inter-operational dependencies between software elements (i.e. one version of a software element will only work with specified versions of other software elements)

Each component within a release:

- Has a unique name and version number assigned by the software provider
- If the component consumes or implements defined specifications or protocols then the name and version must be recorded
- May have an individual platform specific release cycle and may not be released on all platforms
- Must satisfy a set of technical criteria prior to being accepted to UMD (see below)
- Must pass a set of rigorous tests before any new release is accepted into the distributed
- Release cycle of UMD follows a roadmap which broadly defines direction of evolution and allows for inclusion of new components

The version numbering scheme should follow the conventional major.minor.patch (e.g. 3.1.4) format. A patch release must maintain the same interfaces (CLI and API) and protocol/specification versions. If any of these interfaces or protocol/specification is extended, but retains compatibility with earlier releases, then that difference is recognized by increasing the minor release number. If a release is not backwards compatible then the major release number is increased.

## 3.2 The Released Distribution

The software that the MU releases to production will be defined by sets of platform specific binary packages available from a single repository that have been certified by the software provider and verified by the MU to have the specified quality.

NGIs are able to take uncertified/unverified software components from this repository, directly from the software provider or from any other source in response to their own needs or those of the VOs supported by their sites.

NGIs are of course free to deploy any additional software on their infrastructure that does not affect any of the core systems. The support that EGI will be able to provide to unverified unreleased software components will be limited.

Any software deployed independently by the NGIs that replaces the software certified by EGI.org must comply with the operational and functional behaviour expected and defined by the OU.

## 3.3 Criteria for Production Quality Software

The MU will maintain documentation relating to the development of high quality software that will be developed through feedback from the MCB. These criteria will cover:

- *Generic criteria:* interoperability, completeness, extensibility, deployability, simplicity, product documentation, platform portability, lightweight (i.e. only install the additional components that need to be installed), co-existence (i.e. require minimal, ideally none, changes to any other commonly installed software components), ... [Staff: MU1]
- *Component/element specific criteria:* performance, stability, scalability, standards compliance, command line syntax, public API, graphical interface... [Staff: MU3a]

Many of these criteria cannot be defined expect through representative use by a user community within a production environment. The OU will liaise with the VOs and NGIs [Staff: OU2] to identify sites willing to be involved in providing resources (hardware and applications) to support

the definition of detailed quality criteria for new software components. This will be done in conjunction with the relevant software provider. The ability to provide such support would be seen as an important demonstration of the need for any new functionality by a community.

## 3.4 UMD Roadmap

The software available within EGI for use by the production infrastructure will be described by a roadmap detailing the expected evolution of UMD over the next 18-24 months. As the effort within EGI itself will be dedicated to maintaining the current software base, software containing new functionality can only come through the availability of production ready software from EGI's partner software development projects. The procedure for the introduction of new components into production is covered later in this document. The roadmap will indicate:

- When the MU expects such software to first become available from its software development partners
- When it expects to be able to verify that the software is ready for production
- For each software element in the roadmap it will also specify:
    - The funded support mechanisms (including the expected response time) provided through any maintenance contract
    - The un-funded support mechanisms provided by the community around the software element
    - The testing, certification and integration work that the software element has undergone, including results, to meet the generic or element specific criteria.
    - The projected changes in functionality that could be utilized by other software elements and their expected minor/major release dates
    - Any dependencies on other software elements (and their versions) and compiler versions

# 4 Players, Roles, and Decision Making

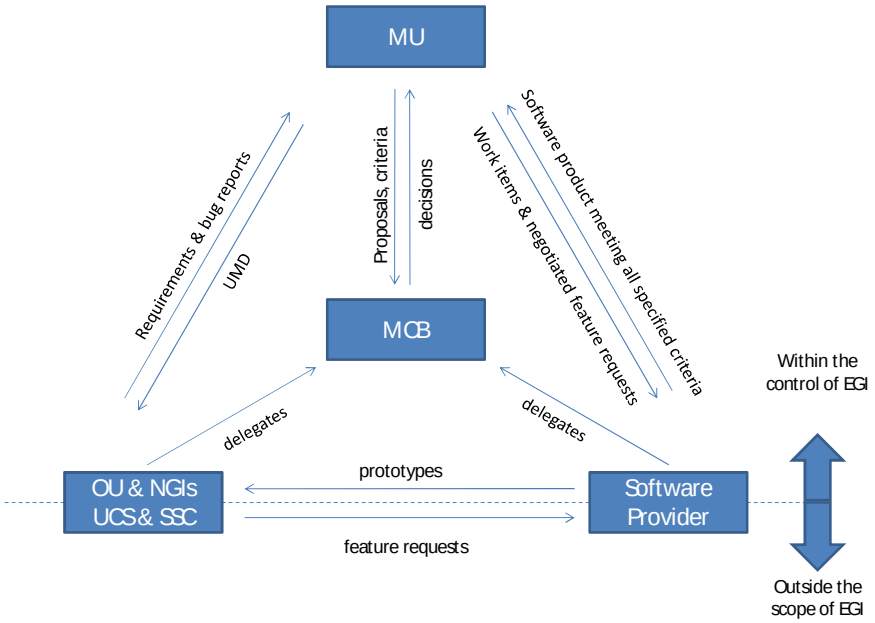


*Figure 2: UMD process and key actors and actions*

The primary actors (defined earlier) in the production of software maintenance releases in EGI are identified as (see also Figure 1). Key UMD customers are:

- Operations Unit (OU)
- NGIs
- User Community Services (UCS)
- Specialised Support Centres (SSC)

Decisions pertaining to UMD support, maintenance and evolution can be of two kinds:

- *of tactical nature*, such as MU staffing, technical issues with test facilities, details of evaluation procedures etc
- *of strategical nature*, such as UMD composition, selection criteria, roadmap etc

While the tactical decisions are left to the prerogative of the MU and the CTO as its head, strategic decisions require approval by the MCB.

MCB is expected to deploy an appropriate decision-making procedure involving voting whenever a consensus cannot be reached. MCB elects a chairperson (rotation period 1 year), who is in charge of organising the Board meetings and liaising with other actors.

The MCB receives input from the MU in the form of the proposals, such as:

- proposals for UMD selection criteria and their prioritization
- proposals for UMD composition
- evaluation results for eventual newly proposed components
- policy proposals
- other documents bearing global consequences for the UMD users and contributors

These proposals and documents are duly evaluated and in case of approval are implemented by the MU, and otherwise are returned to the MU with feedback regarding necessary modifications.
The MCB may also receive input from the observers delegated by the third party software providers. Proposals to include a new component into UMD are forwarded to the CTO for further

evaluation. Evaluation results are reported back to the MCB, which makes the final judgment.

The CTO receives middleware related requirements and feature requests from the UMD customers, either directly, or via MCB. The CTO may also receive information about possible new components directly from the third party software providers. Generally, feature requests, as well as requests for documentation and technical details, are forwarded by the CTO immediately to the middleware consortia. Implemented features and the newly proposed components are evaluated by the MU and results are reported to the MCB. Depending on the outcome, the component implementing a feature may or may not be included into UMD. In case when different consortia offer different solutions for the same functionality, all satisfying the UMD requirements, it is up to MCB to decide whether all or only some implementations are added to the UMD.

Middleware consortia may receive feature requests directly from the customers which they are free to work on using resources not provided through EGI. Developed prototypes may be delivered outside of the UMD release process for the users to test and provide direct feedback to the relevant developers. EGI is clearly not able to support activities and software that does not meet its quality criteria and has not passed through its release mechanism. When satisfied with the new functionality, the users may submit a request for inclusion of such new component into UMD as described above. These restrictions are not imposed in order to restrict advances. They effectively allow the software provider to maintain software for EGI on behalf of its (EGIs) user community through a maintenance contract, but also to work directly with consumers of its software to develop new innovative functionality using resources outside of those funded through any maintenance agreement that it can then offer to EGI.

# 5 Processes for Accepting Releases of Software Components

The categorisation (i.e. development vs. maintenance issue) and the prioritisation of maintenance issues (i.e. requests such as reported bugs, support for new platforms, performance problems, etc.) reported in production use takes place between representatives of EGI.org's middleware, operations and user units. This is done using the established requirements, priority and procedures endorsed by the MCB. These requirements are maintained by EGI following collection from the operations and user communities (through [Staff: UCS1 and OU1]). Following this categorisation and prioritisation, high priority maintenance work items will be assigned to the relevant software provider using the resources available under the agreed maintenance agreement.

The CTO will be empowered on behalf of the Director and the Council to make any final decision regarding prioritisation and the assignment of maintenance work to a software provider within the policies agreed with the MCB. Where an issue falls outside an area not defined by the policy the CTO will work with the relevant bodies within EGI (e.g. the MCB, the Council) to establish a policy.

Depending on the issue assigned to the software provider a revised specification and test suite may need to be defined by the MU for the software provider to deliver against (to be specified by [Staff: MU3a]).

Two aspects expected from a software provider (software engineering and software certification) are defined separately. It is an internal function of the software provider how they implement the capability – it is shown below to illustrate the actions expected by the MU of its software suppliers.

It is envisaged that the software providers will be responsible for delivering to EGI an integrated solution that includes all development, integration, testing and certification of the software element in the environment specified by the MU which will be representative of the expected environment found in production deployments. The MU will work with all software providers to ensure their components are of a high quality and are deployable into the environments used by the NGIs. It is

recognised that for some software such certification work may need to be done by the MU or by another qualified software provider.

It is the MU that coordinates and monitors the delivery of the revised software component by liaising with the relevant software provider. They do not control the internal process of the software provider.

## 5.1 Component Patch Release

Only bug fixes are covered by the maintenance agreement. The process should be lightweight allowing for fast turnaround.

1. MU/OU/UCS (Categorisation and Prioritisation) [Staff: MU5a, UCS2 and OU3]
    - In: Issues from the production users of EGI software: User and Operations Community
    - Out: Assignment of prioritised issue to appropriate software provider
2. Software Provider (Software Engineering)
    - In: Bug report(s) prioritised by EGI.org
    - Out: Component(s) implementation which fixes the specified bugs and adds no other functionality to the baseline minor release of the components, and regression tests of the bugs (if applicable).
3. Software Provider (Software Certification)
    - In: Software from Software Provider development team
    - Out: Certification to EGI.org that the software component in the repository compiles and runs in the Development Environment and satisfies the test suite and test plan.
4. Middleware Unit (Verification) [Staff: MU3b]
    - In: Notification from a software provider that they have contributed a patch release into the repository.
    - Out: Notification that the release meets the same criteria that were required for the baseline minor release, and that the specified bugs were fixed.

In the case of patch releases the verification step is optional and asynchronous – it cannot block release of the components to production (given the limited manpower of MU it would become bottleneck). Therefore the main burden and responsibility for certification remains with software provider.

## 5.2 Component Maintenance Release

A maintenance release is a non-trivial change in the software component covered by the EGI.org's maintenance agreement with the software provider that will still provide backwards compatibility (through the public command line or public APIs as applicable).

1. MU/OU/UCS (Categorisation and Prioritisation) [Staff: MU5a, UCS2 and OU3]
    - In: Issues from the production users of EGI software: User and Operations Community
    - Out: List of issues categorised into feature requests (out of scope of EGI) and prioritised maintenance issues
2. Middleware Unit (Specification Development) [Staff: MU3a]

- In: Maintenance Issue requiring work that relates to a change in public interface or behaviour.
- Out: Specification and test plan defined by the MU for approval by the MCB and with direct feedback from the relevant software providers on delivery time, development effort, etc.

3. Software Provider (Software Engineering)
    - In: Specification and test plan from EGI.org
    - Out: Component implementation and test suite implementation in separate packages). Documentation for users, operators (e.g. service dependencies and configuration management information) and developers (e.g. build instructions)
4. Software Provider (Software Certification)
    - In: Software from Software Provider development team
    - Out: Certification to EGI.org that the software component in the repository compiles and runs in the Development Environment and satisfies the test suite and test plan. The test plan may require deployment and functionality tests in environments typically found in production using NGI resources provided through the OU.
5. Middleware Unit (Verification) [Staff: MU3b]
    - In: Notification from a software provider that they have contributed a release into the repository that meets the provided specification as demonstrated through its associated test plan. NB: If EGI depends on software from a provider that it does not have a maintenance contract with to undertake the Software Certification activities then they will have to be done internally or by another 3rd party.
    - Out: Notification to Middleware Unit (Release Management) that a software component has been delivered that meets the general and any component specific specifications defined by the MU:
        - It deploys and runs on the Development Environment and satisfies the specified test criteria required for this component(s)
        - It does not interfere with the running of other components
        - The architecture and implementation of critical areas (e.g. security) has been reviewed by a dedicated organisation/team. This may be an auditable activity within the software provider or a review by a group outside of the original development team such as another software provider or independent 3rd party.
        - This verification will generally be by inspection. Critical deliveries or complex components may deserve additional effort.
6. Middleware Unit (Release Management) [Staff: MU4]
    - In: A set of components of known behaviour and demonstrated functionality stored in the repository.
    - Out: List of components that need to be installed to deliver different deployment environments including documentation relating to deployment guidelines, compatibility issues, ...

## 5.3 Introducing New or Major Component Releases

The development of new software components or new functionality in existing components (i.e. major releases) will be treated as follows:

1. *Notification:* EGI.org may be informed from many different sources about the availability of new software components that may be suitable for inclusion in the release. Potential source of information include:
   - The software provider
   - User communities
   - Operations communities

   All such requests must be directed to the CTO to assess the viability of the request before any resource within the MU is devoted to its assessment. All request and resulting actions undertaken by the CTO must be reported to the MCB.
2. *Prioritisation:* The CTO will make a non-technical assessment of the software component to see how it aligns with the priorities established by the EGI stakeholders. If it appears to meet an established need it will be passed on for assessment by the MU. If it does not its assessment prioritisation will be referred to the MCB for guidance.
3. *Assessment:* The MU [Staff: MU3] will make a technical assessment of the new software component. This may include establishing new criteria in conjunction with the user and operations community. A report will be provided to the MCB and the software provider on the suitability of the component for production use and the steps needed to make it ready.
4. *Inclusion in the roadmap:* Following positive feedback from the MCB the CTO will negotiate with the software provider to establish when any corrections required in the software component will be made and define a maintenance agreement. Maintenance will only start once a component is delivered to an acceptable level that meets production quality.
5. *Inclusion in the distribution:* The process for including the software in the distribution should follow the procedures for accepting a maintenance release of a software component but skipping phase 1.

# 6 Enabling Infrastructure

- *Package Repository:* Source of labeled (by release and platform) source code and binary packages or links to them. All software providers will contribute releases of their software to the MU using this mechanism that they (the software providers) believe have met the generic and specific requirements required by the MU. Packages in the repository may be classified as being ready for release, experimentation, integration, rejected or deprecated [Staff: MU4]
- *Development Environment:* A set of packages and platform that must be used as the baseline environment for development and testing by the middleware provider and the middleware unit. [Staff: MU2]
- *Build Process:* A mechanism (potentially automated) that takes source (from version control repository, tarball, etc...) and provides source and binary packages in the repository
- *Test Process:* A mechanism (potentially automated) that takes binary packages, deploys them in to the designated environment, configures and then verifies that the resulting software is operating as expected.

- *Test Environment:* Unit and functional testing can generally be executed on *adhoc* hardware co-located with the middleware provider. Stress and deployment testing generally needs more dedicated hardware that will be provided by the NGIs. [Staff OU2]

Essential, but not defined here, an issue tracker, document repository, etc. A source code control system may be needed by the MU to maintain code used internally, but most changes will be pushed back to the software providers as issues. [Staff: MU2].

# 7 EGI.org Staff Identification

| Id | Position | Allocated Effort |
|---|---|---|
| MU1 | Process and Generic Quality Criteria | 1 |
| MU2 | Process Tools | 1 |
| MU3a | Define Component Specific Criteria | 3 |
| MU3b | Validate Component Specific Criteria | |
| MU4 | Package Repository | 2 |
| MU5a | Execute Plans | 1 |
| MU5b | Standards | |
| UCS1 | Coordination of SSC activities (Requirements – task 1) | 1 (partial) |
| UCS2 | Grid Planning and Technical Coordination | 2 (partial) |
| OU1 | Requirements Gathering | 0.5 |
| OU2 | Testbed support for middleware deployment | 1 |
| OU3 | User Support | 2 |

# 8 Additional Documents

This document concentrates on the high-level software process within the EGI model. Detailed documentation needs to be developed relating to:

- Establishing Maintenance Agreements. Defining a framework for EGI to consume staff time with a software provider for engineering work on patch and minor releases. Major releases are negotiated separately. This is not a closed club and an open process will need to be defined for applying, validating and managing maintenance contracts.
- Funding context for e-infrastructure middleware activities. Provide a short document that describes our mission statement.

Work on the following items will be delayed until the Technical WG has completed its work (estimated April F2F meeting):

- The UMD Roadmap – its structure and its content. Work on the structure of the UMD Roadmap could start now and a high-level view of the structure is given earlier in this document.
- Generic technical criteria and specific technical specifications for all software components & integration. Progress in this area requires the standards criteria from the Technical WG. Once this document has been delivered the composition of the Technical WG will change to include Oliver & Anders. This work will start in May and report back in June.