EUROPEAN MIDDLEWARE INITIATIVE

SOFTWARE MAINTENANCE AND SUPPORT PLAN

EU DELIVERABLE: DSA1.1

Document identifier:	**EMI-DSA1.1-CDSREF-SoftwareMaintenanceAndSupportPlan-v0_3**
Date:	**31/05/2010**
Activity:	**SA1**
Lead Partner:	**INFN**
Document status:	**DRAFT**
Document link:	

Abstract:
This document describes the Software Maintenance and Support processes, the roles and responsibilities and the main metrics to be used for the Service Level Agreements.

Delivery Slip

	Name	Partner / Activity	Date	Signature
From	Francesco Giacomini	INFN / SA1	03/08/10	
Reviewed by				
Approved by				

Document Log

Issue	Date	Comment	Author / Partner
1			
2			
3			

Document Change Record

Issue	Item	Reason for Change
1		
2		
3		

TABLE OF CONTENTS

1. INTRODUCTION **5**
1.1. PURPOSE 5
1.2. DOCUMENT ORGANISATION 5
1.3. APPLICATION AREA 5
1.4. REFERENCES 5
1.5. DOCUMENT AMENDMENT PROCEDURE 5
1.6. TERMINOLOGY 5
2. EXECUTIVE SUMMARY **6**
3. MAINTENANCE **7**
4. SUPPORT **8**
5. CONCLUSIONS **9**

1. INTRODUCTION

1.1. PURPOSE

1.2. DOCUMENT ORGANISATION

1.3. APPLICATION AREA

1.4. REFERENCES

Table 1: Table of References

R 1	
R 2	
R 3	
R 4	
R 5	
R 6	

1.5. DOCUMENT AMENDMENT PROCEDURE

Amendments, comments and suggestions should be sent to XXX. The procedures documented in XXX have been followed.

1.6. TERMINOLOGY

Table 2: Table of Definitions

2. EXECUTIVE SUMMARY

3. MAINTENANCE

The Software Maintenance task in SA1 is responsible to coordinate the continuous maintenance of the middleware components developed within the project and included in an EMI distribution, preserving at the same time their stability in terms of interface and behavior, so that higher-level frameworks and applications can rely on them.

The term *interface* is intended in a broad sense. Interfaces include, but are not limited to, APIs, ABIs, WSDLs, DataBase schemas, network protocols, authentication and authorization mechanisms, logging formats, packaging and other deployment characteristics of a component.

A strong constraint of the software maintenance activity is that no backwards-incompatible changes are introduced in production.

3.1. EMI RELEASES

The EMI distribution will be organized in periodic major releases, tentatively delivered once a year [cite milestones], 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 the implementation of the plan is coordinated by JRA1 [cite milestones].

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) need to be preserved even across major releases, according to end-of-life policies to be defined on a case-by-case basis.

The availability of a new major release of EMI does not automatically obsolete the previous ones and multiple major releases may be supported at the same time according to their negotiated end-of-life policies.

3.2. COMPONENT 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.

  The contents of a new major release are endorsed by the PTB and included in the project technical plan. The implementation is coordinated by JRA1.

- **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.

  The contents of a new minor release are endorsed by the PTB and included in the project technical plan. The implementation is coordinated by JRA1. If the release is going to be introduced in an existing major release of EMI, the implementation is also supervised by SA1 in order to guarantee that the production quality and the backwards-compatibility are preserved.

- **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.

The type of release is reflected in the version of the corresponding package(s) [to be described in DS1.2].

3.3. INCIDENTS, PROBLEMS, CHANGES

SA1 is in general responsible for *corrective* and *adaptive* maintenance to address defects, potential defects and minor improvements found in running services in the production environments, based on requests for changes (RfC) in the code of EMI software components.

ITIL defines a **change** as the addition, modification or removal of authorized, planned or supported service or service component and its associated documentation.

One of the main sources of RfCs are the *incidents* reported by users through the support channels, notably GGUS. ITIL defines an **incident** as an unplanned interruption to an IT service or reduction in the quality of an IT service. Failure of a component that has not yet affected service is also an incident.

After investigation by the different levels of support (see Section ???), an incident may be traced to and actual *problem* in the code. ITIL defines a **problem** as the cause of one or more incidents.

If that is the case, the problem is recorded into a tracker and further processed by the team responsible for the affected component, usually leading to changes in the code.

Another major source of RfCs is the continuous stream of user requirements, notably from the EGI MCB and similar bodies.

Finally, RfCs, both to fix defects and to introduce improvements, can also be generated internally in the project or even in the Product Team itself in charge of a given component.

3.4. PRIORITY-DRIVEN DEVELOPMENT

Since SA1 addresses only *corrective* and *adaptive* maintenance, it is necessary to define how to select RfCs that qualify as corrective or adaptive.

The criterion for such classification is the priority of the RfC, where the priority is the result of the composition of a number of factors:

- severity: a measure of the degradation of the quality of service of the affected component;
- impact: a measure of the effect of the degradation of the quality of service of the affected component;
- urgency: a measure of how long it will be until the quality of service of the affected component is not significantly degraded;
- cost: a measure of the resources needed for the management of the change, including the risks associated to the degradation of the quality of the affected component in case the change is not fully successful.

The evaluation of the priority of an RfC results in one of four possible logical values. Each level implies a very specific behavior for the management of that RfC.

The four priority levels and the corresponding behaviors are:

- **Immediate**

  The RfC needs to be addressed as soon as possible, in all affected EMI major releases.

  A release containing immediate-priority changes can contain only immediate-priority changes. Multiple immediate-prioritychanges can be included in the same release, provided that any change does not delay the releasesignificantly.

- **High**

  The RfC will be addressed in a next release of the affected component, in all affected EMI major releases.

- **Medium**

  The RfC will be addressed in the release of the affected component that will be shipped with the next EMI major release.

- **Low**

  There is no target date for addressing the RfC.

Each logical value needs to be mapped to a specific value in the tracker used by each PT.

The priority of an RfC can be suggested by the proponent, but the PTB (or should it be the EMT?) is the ultimate responsible for setting it.

3.5. TRACKING OF CHANGE REQUESTS

Each RfCs needs to be tracked with an appropriate tool. Each PT is free to choose the one to use for the product or products it delivers, provided the tool satisfies the constraints described in this section.

For each RfC the following information should be available or recorded:

- a unique identifier;
- a URL pointing to a description of the RfC. If the RfC concerns a security vulnerability the URL should point to a private page;
- the affected component;
- the priority;
- defect vs feature;
- the state of the RfC. The minimal set of logical states is: Open (just submitted), Accepted (assessment has been done and RfC accepted), Fixed (change committed to the VCS), Closed (change released or won't fix/unreproducible/invalid/etc.);
- detection area (production, testing, development, etc.);
- target component release;
- target EMI major release(s) (or ARC, dCache, gLite, UNICORE major releases in the transition period).

If the same RfC is going to be applied to two different component releases (i.e. to two different component configuration items) or to two different EMI major releases, then a corresponding number of RfC is created.

An RfC cannot stay in the Open state for a long period (how much?). Once accepted it should be immediately associated to a next release of the affected component.

The above information will be collected periodically (of the order of every week) to monitor the progress of the Software Maintenance task.

3.6. ROLES

TBD

3.7. PKIS

PKIs relevant for the maintenance task and how they will be monitored is presented.

TBD

3.8. TRANSITION

Since the first EMI major release is foreseen at PM 12, the existing separate middleware distributions still need to be maintained. This section will briefly describe how that will be done.

TBD

4. SUPPORT

The reliability of the EMI distribution and the reputation of the EMI project itself depend critically on the ability to provide, together with EGI, effective support in case a user requests assistance about one of the services provided by EMI. The request can concern anything from documentation to configuration, from receiving advice to asking for a new feature, but the following description will concentrate mainly on incidents[1], because that is the type of request that will involve the EMI support task.

4.1. SUPPORT MODEL

The EMI support model integrates in the overall support structure adopted in EGI, which foresees an organization in three levels:

1. The EGI Helpdesk represents the main contact point for a user where to get support. Within the Helpdesk the Ticket Processing Management (TPM) is responsible for the monitoring and routing of all active tickets to the appropriate support units (SUs).

   In EGI the Helpdesk is a distributed infrastructure consisting of a central Helpdesk interconnected with a collection of local NGI or EIRO Helpdesks.

   If the Helpdesk is unable to resolve the incident, this is escalated for further investigation to a 2nd-level support unit.

2. The Deployed Middleware Support Unit (DMSU) ensures the availability of more specialized skills than those offered by the Helpdesk in the investigation and resolution of incidents. The DMSU includes people that together can cover all middleware areas: job and compute management, data management, security, information systems, accounting, etc.

   The DMSU is an integral part of EGI.

   If the DMSU is unable to resolve the incident, this is escalated for further investigation to a 3rd-level SU.

3. 3rd-level SUs offer the most specialized skills needed for the investigation and resolution of an incident and are typically represented by the developers of the affected software component.

   3rd-level SUs are not normally part of EGI but are integrated in the organization of the software providers, such as EMI.

This industry-standard model provides the most effective use of resources, for it involves the ultimate technical experts only when their detailed knowledge is indispensable for the investigation of an incident.

Support tickets should not normally flow from the Helpdesk directly to the EMI SUs, unless it is evident that the incident is caused by a software problem.

The tool adopted by EGI to track support requests is GGUS. Incidents occurring to users on the production infrastructure should always be reported through GGUS and their processing tracked

1 ITIL defines an incident as “an unplanned interruption to an IT service or reduction in the quality of an IT service.”

through GGUS tickets. This would allow to compute user-oriented metrics completely from GGUS data.

Different support models for other DCIs will be evaluated on a case-by-case basis.

4.2. THE EMI SUPPORT UNITS

Within EMI many SUs are established. Their exact number and scope can change during the course of the project, but approximately an SU is foreseen for each software product that has user visibility and is registered in GGUS, typically high-level middleware services.

Since multiple software products can be under the responsibility of the same Product Team, it may happen that the memberships to two or more SUs are in fact overlapping or even coincident.

When a product becomes registered in GGUS, a SU must be established and should provide:

- an e-mail address;
- a FAQ describing the SU;
- ...

The internal organization of an SU is left to the responsibility of the corresponding PT, provided it is adequate to satisfy the SLAs established between EMI and EGI.

The following metrics will be computed for every EMI SU:

M1) Number of incidents per week

M2) Incident resolution time

4.3. THE CATCH-ALL SUPPORT UNIT

A special EMI SU is established, the Catch-All SU, whose purpose is to intercept and quickly re-assign all GGUS tickets for which the EGI support units are not able to properly identify a specific EMI SU.

The organization of the C-A SU is under the responsibility of the SA1 User Support task. Considering that a) the number of GGUS tickets that require the intervention of the 3rd-level is relatively low (e.g. about 40 tickets have been received by all the gLite SUs during the period January-April 2010) and b) the assignment of a ticket to the C-A SU should be in turn an exceptional situation, the organization of the C-A SU should be lightweight. Initially it will simply consist of a mailing list (emi-support@eu-emi.eu) that will re-direct all ticket notifications to the general support mailing list of ARC, dCache, gLite and UNICORE. It is then expected that subscribers to those lists will process the tickets and re-assign them to the specific SUs. If needed, particularly complex issues can be discussed within the EMT.

The tickets arriving at the C-A SU will be properly monitored (by whom?) to guarantee that they are promptly re-assigned. If not, the problematic ticket should be brought to the attention of the EMT.

The following metrics will be computed for the C-A SU:

A) Number of incidents per week

B) Re-assignment time

If, contrary to the expectations, A) or B) are consistently high (how much???), the organization of the C-A SU will be reviewed with the introduction of shifts among all the SA1 members.

4.4. RESOLUTION OF INCIDENTS

In case an incident is reported, the goal of the user support activity is to restore normal service operation as quickly as possible, thus ensuring that the best possible levels of service quality and availability are maintained. What "normal service operation" means is defined in the Service Level Agreement (SLA).

An incident may or may not be caused by a problem[2]. If it is, a corresponding entry shall be created by the SU in the bug tracking tool specific to the affected software product and the two should be cross-linked. In a GGUS ticket this is done using the "Related issue" field.

The incident should stay open until a satisfactory (for the user) solution is found. The solution may not necessarily require that the causing problem is fully fixed, for example an acceptable (according to the SLA) workaround may exist. If the incident resolution instead does require that fix, then the ticket should stay open until the fix becomes part of a new product release.

4.5. SUPPORT TIMELINE

It is foreseen that only the latest two EMI major releases are supported at a time. Within an EMI major release only the latest version of a component is supported. More extensive coverage will be evaluated on a case-by-case basis together with the users requesting it.

4.6. KEY PERFORMANCE INDICATORS

TBD

2 ITIL defines a problem as "the cause of one or more incidents." In practice it is a software bug.

5. CONCLUSIONS