European Middleware Initiative (EMI)

Alberto Di Meglio (CERN)
Project Director

Outline

- Definitions
- Project structure
- (Possible) Software Engineering model
- Relationships

What is EMI?

The European Middleware Initiative (EMI) project represents a close collaboration of the three major middleware providers - ARC, gLite and UNICORE, together with other software providers (dCache) - to establish a sustainable model to support, harmonise and evolve the grid middleware for deployment in EGI and other distributed e-Infrastructures

FP7 Program

FP7 Capacities Work Programme 2010: Infrastructures

Call **FP7-INFRASTRUCTURES-2010-2**

Sub-topic: 1.2.1.3 – Middleware and repositories

> Develop middleware that strengthens European presence by consolidating or even going beyond existing DCIs (e.g. exploiting emerging developments like virtualisation), while improving their stability, reliability, usability, functionality, interoperability, security, management, monitoring and accounting, measurable quality of service, and energy efficiency

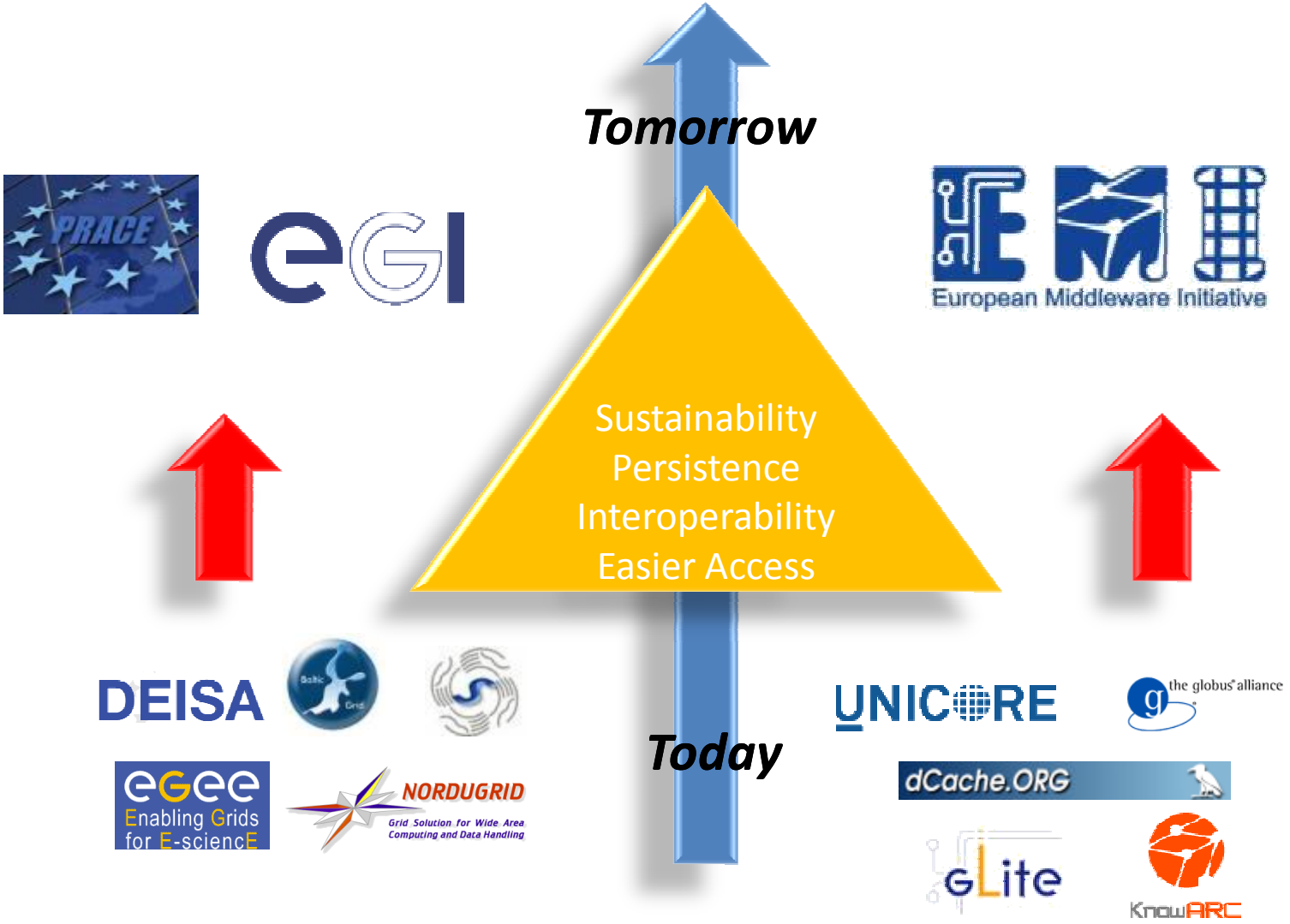
Starting date: May 1st (?)

Duration: 3 years

Total budget: 26M € (13M € from EC + 13M € from partners)

Effort: 65 FTEs/year (88% for technical activities)

A European Vision



Objectives

Consolidate

Consolidate the existing middleware distribution simplifying services and components to make them more sustainable (use of off-the-shelf and commercial components whenever possible)

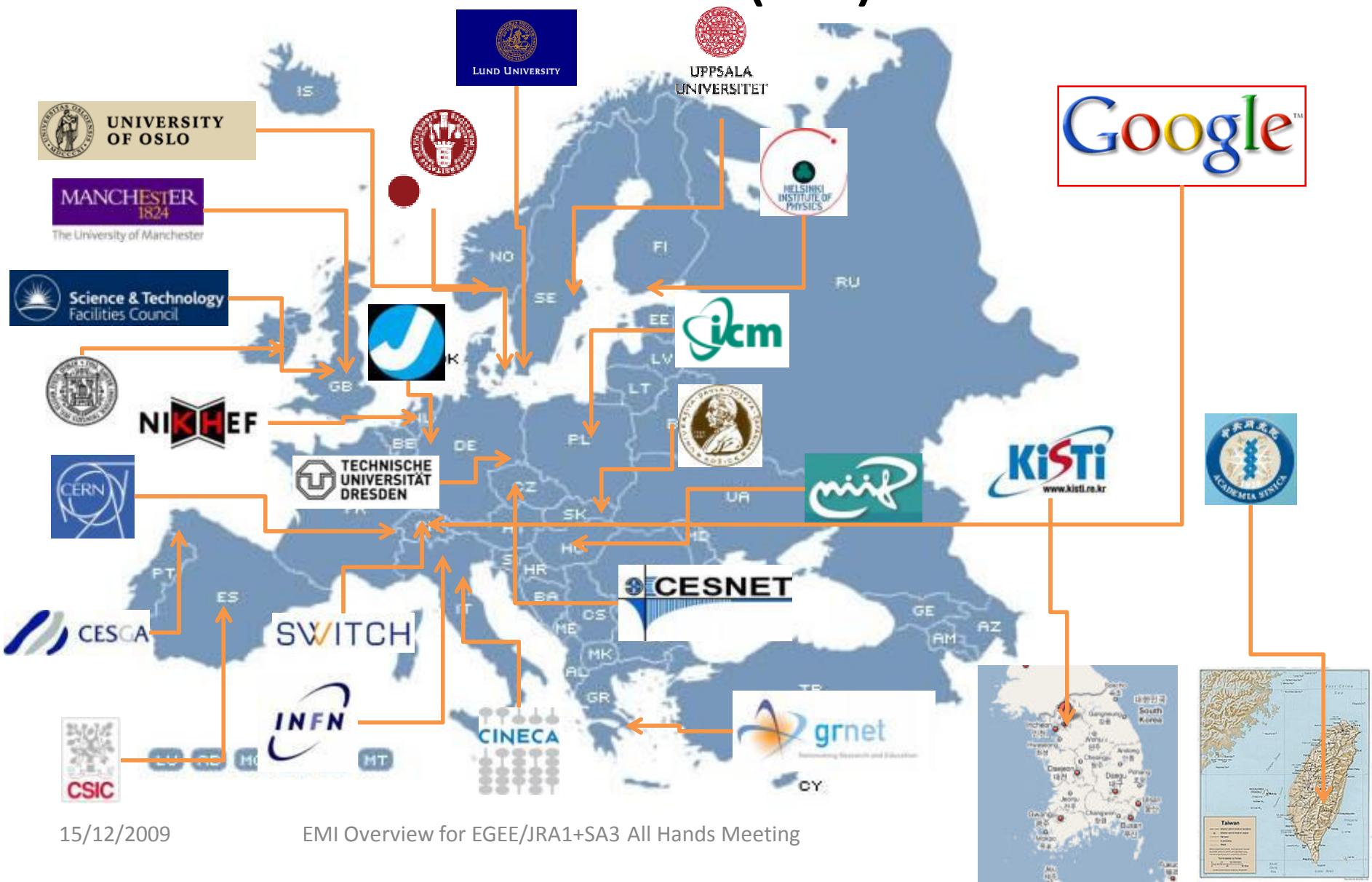
Evolve

Evolve the middleware services/functionality following the requirement of infrastructure and communities, mainly focus on operational and interoperability aspects

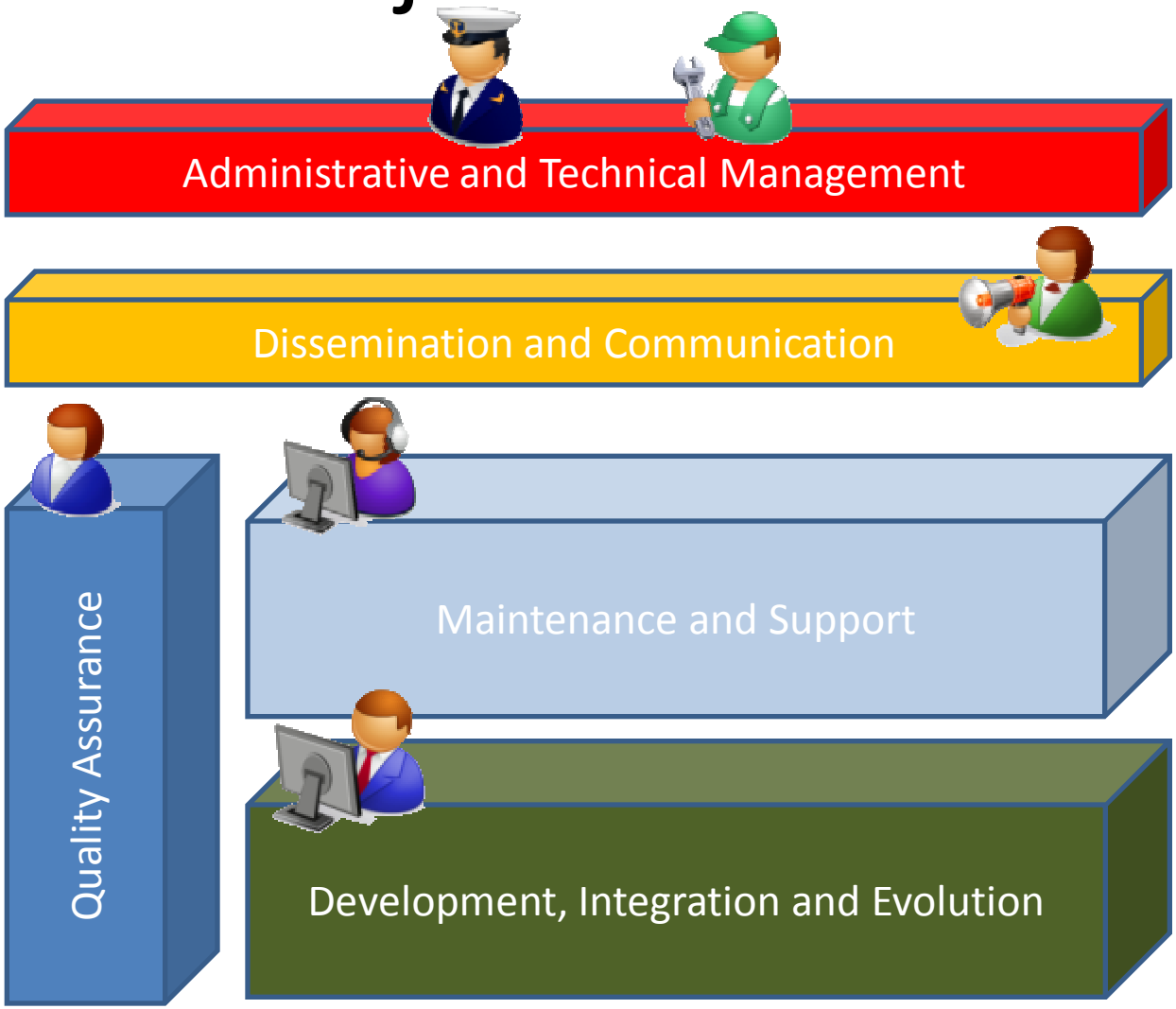
Support

Reactively and proactively maintain the middleware distribution to keep it in line with the growing infrastructure usage

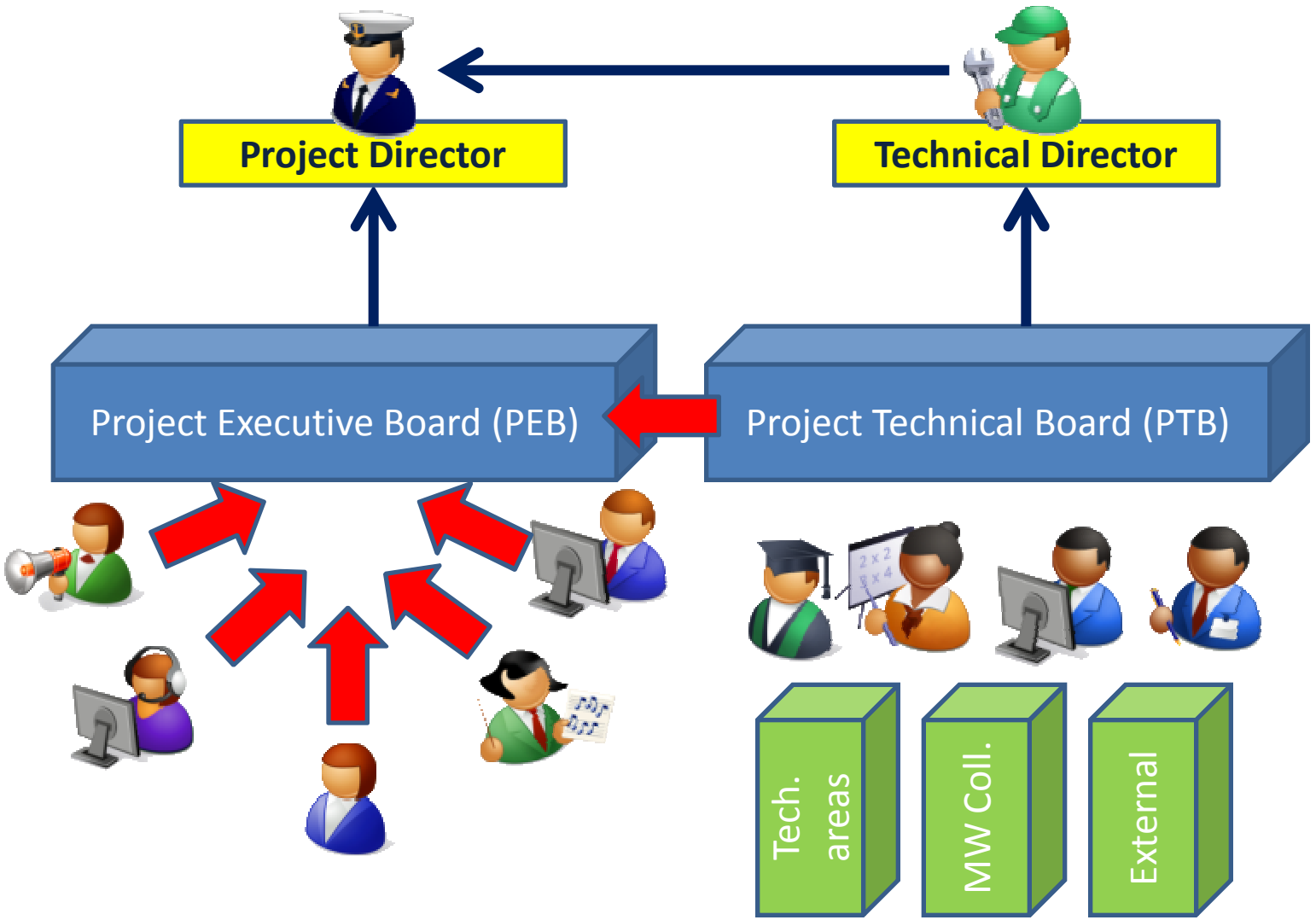
Partners (24)


Lund University
Uppsala Universitet
University of Oslo
Google
Manchester 1824
The University of Manchester
Helsinki Institute of Physics
Science & Technology Facilities Council
icm
NIKHEF
Technische Universität Dresden
CERN
KiSTi
www.kisti.re.kr
CESNET
CESGA
SWITCH
INFN
CINECA
grnet
CSIC
IS
NO
SE
FI
RU
EE
LV
LT
PL
GB
IE
NL
BE
DE
CZ
SK
UA
PT
ES
IT
HR
BA
CS
ME
MK
AL
GR
CY
GE
AM
AZ

Project Structure



Project Execution



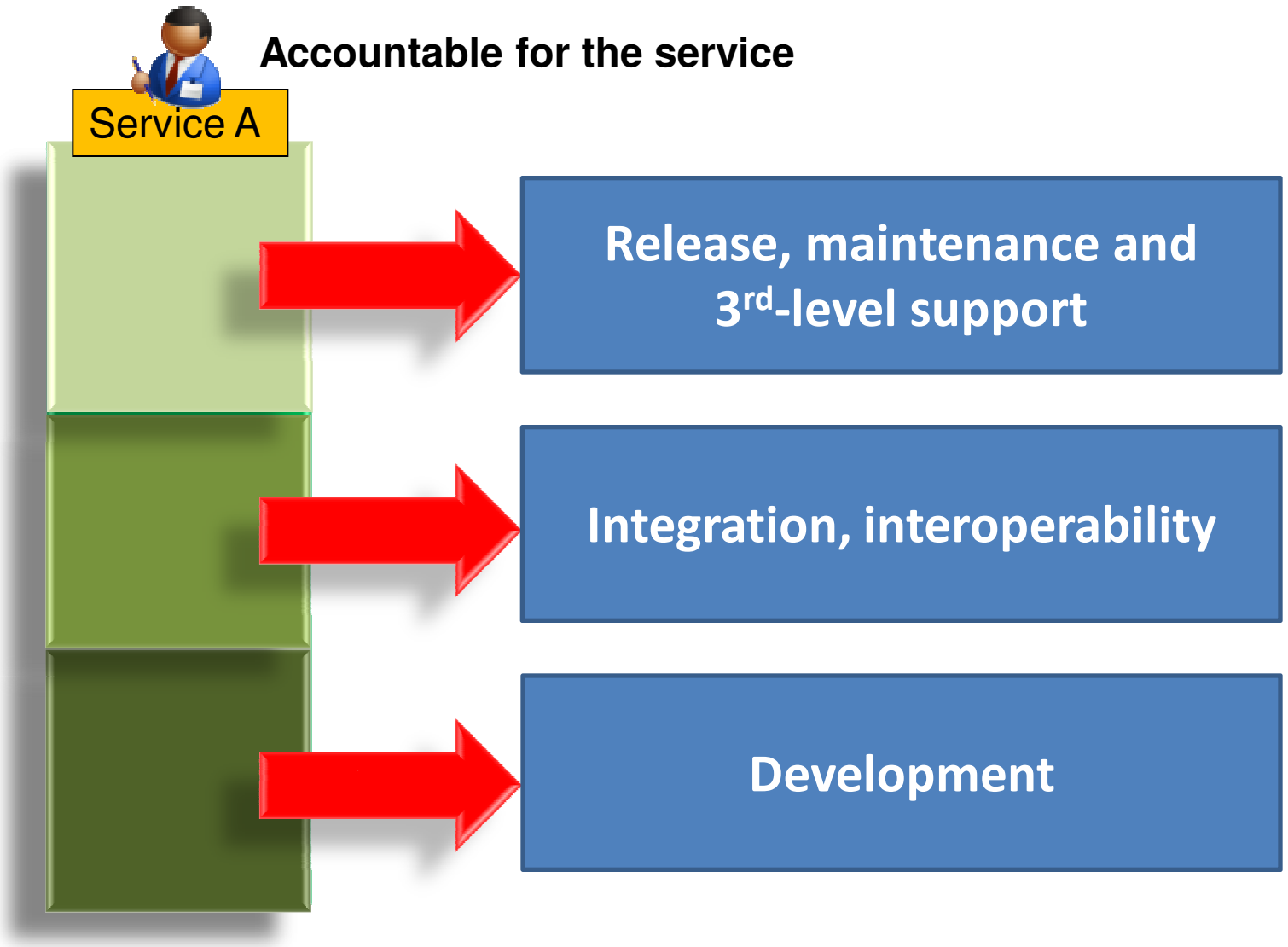
Technical Areas

Compute Services	**A-REX, UAS-Compute, WMS, LB, CREAM, MPI**
Data Services	**ARC SE, dCache, StoRM, UAS-Data, DPM, LFC, FTS, Hydra, AMGA**
Security Services	**UNICORE Gateway, UVOS/VOMS/VOMS-Admin, ARGUS, SLCS, glExec, Gridsite, Proxyrenewal**
Infrastructure Services	**Messaging, accounting, monitoring, virtualization/clouds support, Information systems and providers**

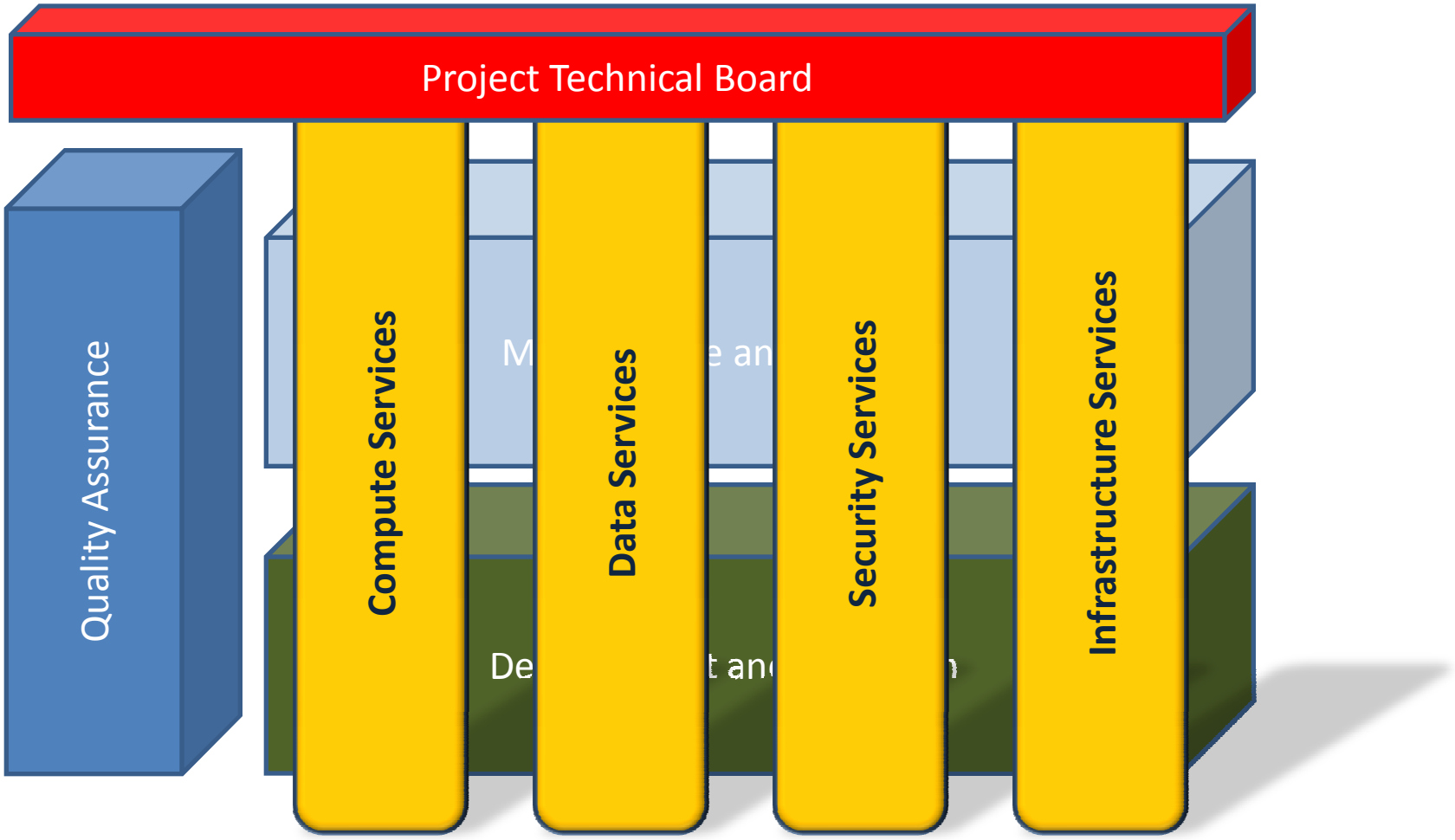
Product Teams

- Product teams are the services implementation teams within each area responsible to deliver software releases and all associated material.
- They perform the required technical tasks from **design to release through implementation, testing and certification and** provide **3rd-level support**.
- Product Teams are flexible, in the sense that they can be formed or closed as the corresponding products are introduced or obsoleted in EMI and allow adding or removing services as needed even from external contributors
- They provide a transparent and direct method to assign responsibility for a service

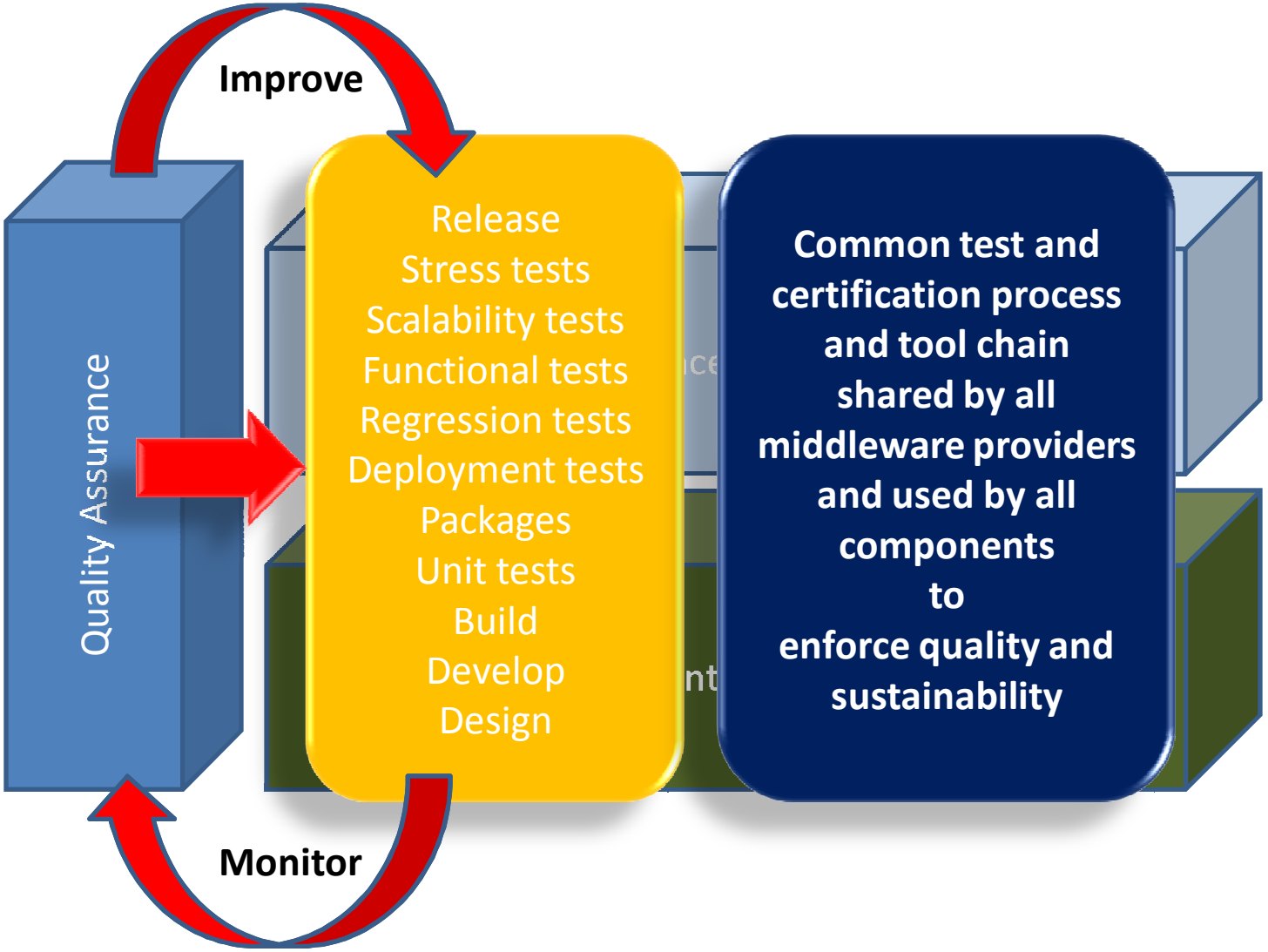
Product Teams



Technical Structure



Quality Assurance



Releases and Release Policies

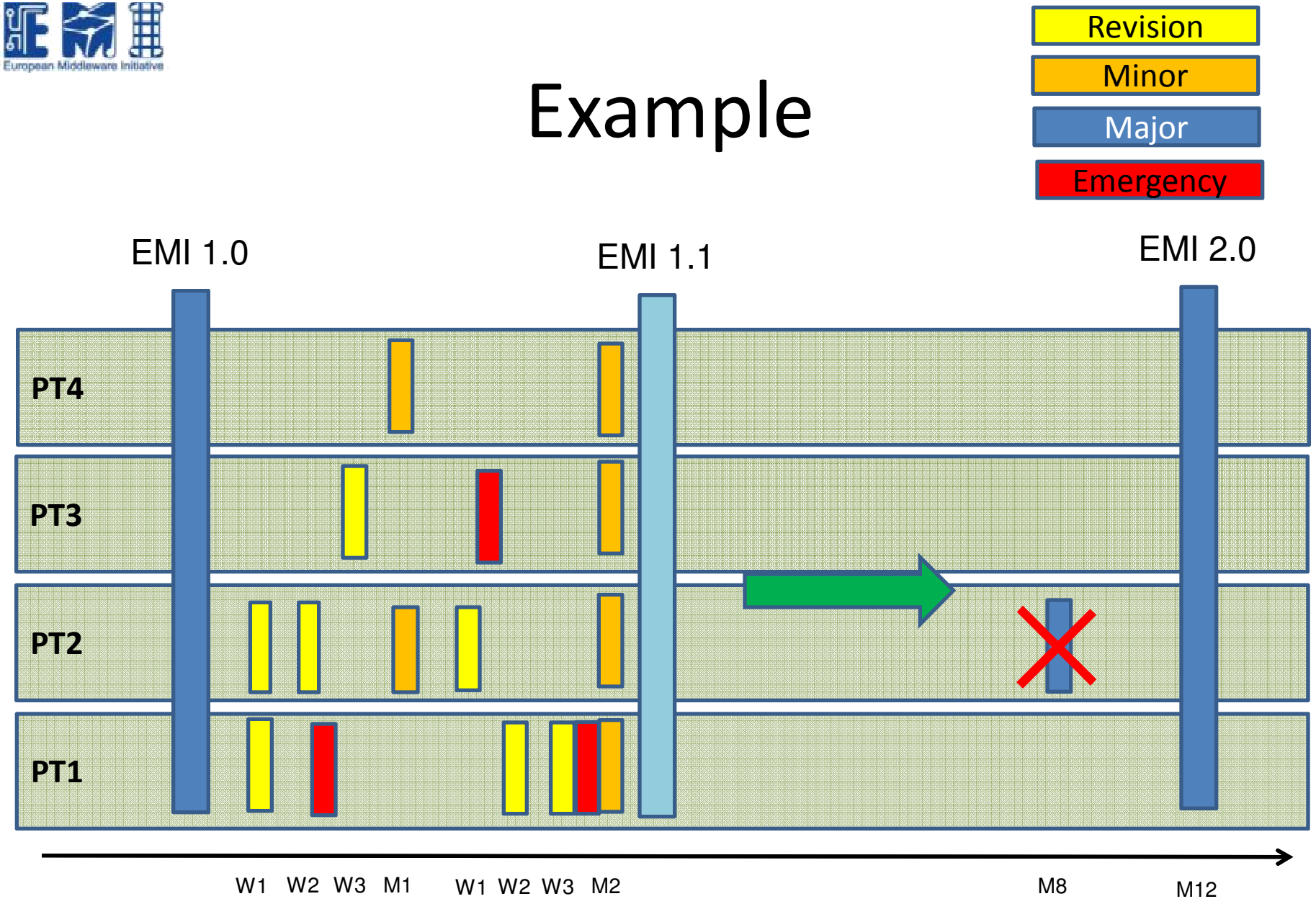
This (mainly) applies to EMI releases

- **Major releases:** once or twice per year, may contain non-backward compatible changes

This (mainly) applies to individual components

- **Minor releases**: a few times per year, fully backward compatible, may contain new functionality and fixes
- **Revisions**: every week or two weeks, only bug fixes
- **Emergency**: as needed, only very specific bug fixes, use emergency release procedures

Example


Revision
Minor
Major
Emergency
EMI 1.0
EMI 1.1
EMI 2.0
PT4
PT3
PT2
PT1
W1 W2 W3 M1
W1 W2 W3 M2
M8
M12

Repositories (1)

- As standard and intuitive as possible
- One repository for each EMI Major Release
  - Contains all packages and metapackages compatible with the major release
  - May contain 'external' packages not available elsewhere
- Each repository is split in (something like):
  - **Base**: the base release
  - **Updates**: all packages released after the base release (minor and revision releases)
  - **Tweaks**: all packages released after the base release and not fully supported or targeted to specific users (sites)
  - **Sources**: source packages

Repositories (2)

- Repositories are not only for production use
- Versioned repositories are used also for internal development
  - For example, if the next major release 2.0 is scheduled to be out in 6 months, the repository for 2.0 can be created at the beginning and populated with new packages are they become available
- Development repos can be used by external users for their own testing

ETICS (1)

- ETICS will be used to provide a common environment for building and testing
  - Especially testing
  - Builds can be done in other ways if the products comply with the agreed formats and guidelines
  - However, for testing it is necessary to have some form of 'presence' within ETICS
- An EMI Major Release is modelled as a **locked project configuration**
  - It must build completely on all supported platforms (portability)

ETICS (2)

- The project configuration for a major release (base) must contain:
  - **Configurations** for all EMI (= ARC, gLite, UNICORE, dCache) **packages** agreed to make part of that release
  - All necessary **metapackages**
  - All necessary dependency information (**DEFAULTs**), using version constraints or named configurations
- All updated configurations after the base can be built against the base (assumption of backward compatibility)
- Periodically (or when there are important changes) new project configurations (**roll-ups**) are created with an incremented minor or revision number (by cloning, equivalent to tagging)
  - However components and metapackages are released independently, not as part of an EMI release
- Further component configurations **can** be built against the minor or revised project (roll-ups) configurations to enforce specific constraints

ETICS (3)

- Packages repositories are created automatically by ETICS
- If the schema discussed earlier can be implemented directly with ETICS is better
- Otherwise some additional step to copy packages from the ETICS Repository to the final repositories may be necessary

Release Process (1)

- Releases are as much as possible planned in advance
- The first step of certification is done within the Product Teams, middleware leaving the PT is assumed to be certified
  - It must be proven on the basis of the agreed criteria (tests, documentation, etc)
- PTs announce the availability of a revision, minor or major release (release candidate)
- The integration teams in JRA1 (for development releases) and SA1 (for production releases) put together all contributions
  - it can be anything from a single package to a full release according to the planned schedule of fixes and new functionality

Release Process (2)

- The Release Candidates are checked against the external acceptance criteria (integration, stress, scalability and interoperability tests as necessary)
- The tests must be as much as possible automated to minimize the required effort
- When an RC is announced a grace period starts during which other PTs can validate the RC. At the end of the grace period:
  - The RC becomes a release if no problems have been found
  - Is rejected if problems are found (this may happen as soon as problems are found depending on the severity)

Testing (1)

- Done with ETICS as much as possible
- Manual if no other choice, but it must be fully documented
- It includes a number of test actions such as:
  - Unit tests during the builds
  - Deployment tests of the metapackages created during the builds
  - Configuration and integration tests
  - Standard compliance tests
  - Regression tests
  - Functional tests
  - Interoperability tests
  - Stress and scalability tests
- All tests have to be reproducible
- The test results have to be recorded and released with the software in support of the SLAs

Testing (2)

- Product Teams are responsible to test their software and maintain their specific test suites in ETICS
- Product Teams can contribute to the test suites of other Product Teams with specific integration, interoperability or usage tests, which will become part of the standard testing of the other PTs
- Wider scale interop tests are done by dedicated people in JRA1
- EGI, PRACE, OSG, other DCIs may require specific acceptance tests to be performed
- Users, application providers, external collaborators can provide additional tests or test cases

Process discussions

- Most of the process still to be defined and discussed
- This is just the starting point
- I'd like to see how we can converge from the current EGEE/gLite process to the EMI process making any necessary change on both sides
- We need to avoid at all costs to have a process valid until April 30th and a different process valid from May 1st
- Similar discussions will take place with ARC and UNICORE to have everybody agree on a common process

Relationships with Operating Systems

- EMI will try as much as possible to push the middleware into mainstream OSs like Fedora and Ubuntu
- Need to understand how to manage the differences in adoption speed
  - The versions in Fedora or Ubuntu may be considerably older than the latest official EMI releases

Selection Criteria

EMI includes:

- Middleware services and components from the participating providers
- All services that are or will be in production by next year
  - EGI, PRACE operations must be fully supported
- Services not currently in production, but already known to replace existing services
  - Ex: ARGUS to replace LCAS/LCMAPS
- New services satisfying identified new user needs
  - EMI requirements are user-driven, satisfy the needs of the user communities filtered through the coordination of major initiatives (EGI, PRACE, WLCG, SSCc projects)

New Development

- New services or functionality are introduced as needed based on:
  - New user requirements
  - Need to evolve existing services or replace older technologies to support the growing infrastructures
  - Special focus on integration of virtualization, monitoring interfaces, accounting, support for gateways and portals
- Trade between new research and stability of the infrastructure operations
  - Large-scale realistic testing (in collaboration with the infrastructure providers), phase-out/migration plans, support policies

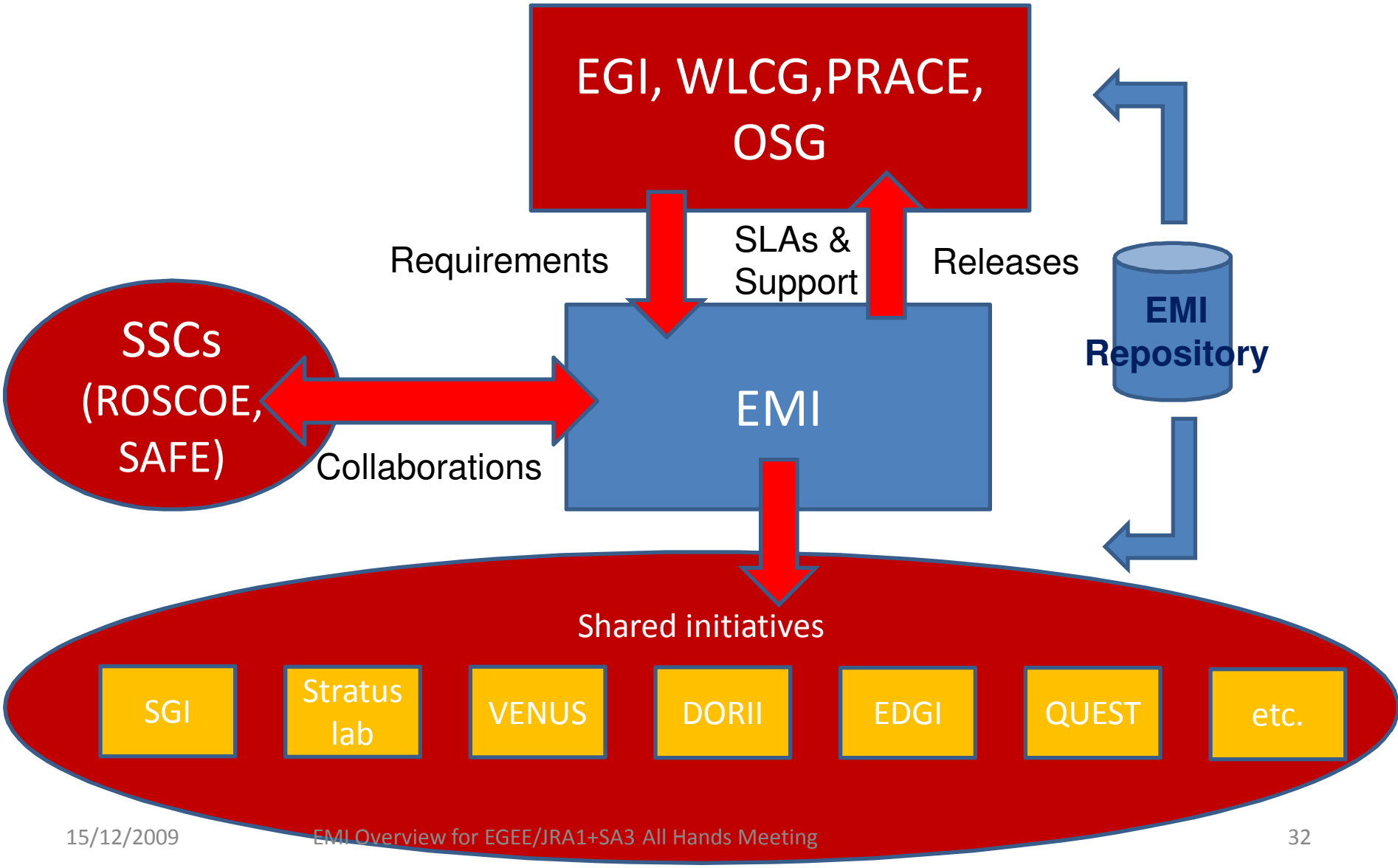
Middleware Evolution Targets

- **Consolidation of security models** across the three middleware stacks
- **Messaging**: messaging service part of the middleware architecture, used for various information tasks (monitoring, accounting, policies, etc). Based on activeMQ or MRG
- **Accounting**: minimum necessary modifications to the most used sensors and collectors to use messaging and a standard data format. Accounting tools are not in scope
- **Monitoring**: monitoring and instrumentation interfaces following an agreed standard (QMF?) to be used by higher level monitoring apps
- **SRM and security** 'harmonization' and **common storage accounting** across SE services, **POSIX support** extension for data access services
- **Consolidation of clients/APIs** (keep backward compatibility, but simplify)
- **MPI**: support added as necessary, convergence HTC/HPC?
- **Virtualization and clouds access**: support for existing or new systems (depends on what systems will be most used)
- **Interoperability** between HTC and HPC and between HTC stacks (also desktop grids, clouds)

Standardization

- Very important goal
- All services must:
  - Implement the ‘best’ relevant standards
  - Implement them in the same way
- ‘Best’ means:
  - A community standard if it is useful, usable or can be easily improved
  - A ‘de facto’ standard if no community standard exists or what exists is clearly not usable
- EMI will drive the standardization, not passively adopt it
- The Technical Areas have a major role in this, since all services within a TA may share ideas, experiences, implementations and tests

Collaborations



‘Works with EMI’

- A technical collaboration program for application providers or middleware contributors
- Members of the program get advance technical previews and support and commit to give feedback and test the EMI software
- Two goals:
  - Ensure that when new EMI components are released, the applications using them will have NO surprises
  - Create a strong identity for the EMI software as THE middleware for grids (and more)

Thank you