EUROPEAN MIDDLEWARE INITIATIVE

DJRA1.1.4 - COMPUTE AREA WORK PLAN AND STATUS REPORT

EU DELIVERABLE: D5.1.4

Document identifier:	EMI-DJRA1.1.4-1277614-Compute_Area_Work_Plan_M36
Date:	30/04/2013
Activity:	JRA1
Lead Partner:	INFN
Document status:	Final
Document link:	https://cds.cern.ch/record/1277614

Abstract:
This deliverable contains the final status report of the Compute Services technical area, compliant with the overall EMI Technical Development Plan. The plan is released early in the project life and updated every year including a status report on the achievements of the past 12 months compared to the planned objectives.

I. DELIVERY SLIP

	Name	Partner/Activity	Date
From	Marco Cecchi	INFN/JRA1	08/04/2013
Reviewed by	Balazs Konya Krzysztof Bendyczak Ivan Marton	LU/JRA1 UWAR/JRA1-SA1 NIIFI/JRA1-SA1	17/04/2013
Approved by	PEB	-	17/04/2013

II. DOCUMENT LOG

Issue	Date	Comment	Author/Partner
1	03/04/2013	v1.0 sent to PEB and reviewers for review	Marco Cecchi/INFN
2	16/04/2013	v1.1 sent to PEB and reviewers v1.2 sent to PEB and reviewers	Marco Cecchi/INFN
3	17/04/2013	v1.0 PEB approved version	PEB

III. DOCUMENT CHANGE RECORD

Issue	Item	Reason for Change
1		
2		

IV. DOCUMENT AMENDMENT PROCEDURE

This document can be amended by the authors further to any feedback from other teams or people. Minor changes, such as spelling corrections, content formatting or minor text re-organization not affecting the content and meaning of the document can be applied by the authors without peer review. Other changes must be submitted for peer review and to the EMI PEB for approval.

When the document is modified for any reason, its version number shall be incremented accordingly. The document version number shall follow the standard EMI conventions for document versioning. The document shall be maintained in the CERN CDS repository and be made accessible through the OpenAIRE portal.

V. GLOSSARY

Acronym	Long Name
A-REX	ARC Resource-coupled EXecution
API	Application Programming Interface
ARC	Advanced Resource Connector
BES	Basic Execution Services
BLAH	Batch Local Ascii Helper
CE	Computing Element
CLI	Command Line Interface
CRL	Certificate Revocation List
CREAM	Computing Resource Execution And Management
DCI	Distributed Computing Infrastructure
DEISA	Distributed European Infrastructure for Supercomputing Applications
DoW	Description of Work
EGI	European Grid Infrastructure

EMI	European Middleware Initiative
EMIR	EMI Registry
EMIR SERP	EMI Registry – Service Endpoint Record Publisher
FTS	File Transfer Service
HED	Hosting Environment Daemon
HEP	High Energy Physics
HiLA	High Level API
HPC	High Performance Computing
HTC	High Throughput Computing
HPC-BP	HPC-Basic Profile
ICE	Interface to CREAM Environment
JDL	Job Description Language
JSDL	Job Submission Description Language
JMS	Job Management Service
LB	Logging and Bookkeeping
LRMS	Local Resource Management System
MCT	Minimum Completion Time
MPI	Message Passing Interface
NGI	National Grid Initiative
OGF	Open Grid Forum
OS	Operating System
PGI	Production Grid Infrastructure
PBS	Portable Batch System
PT	Product Team
RTE	Run Time Environment
SE	Storage Element
SGE	Sun/Oracle Grid Engine
SLURM	Simple Linux Utility for Resource Management
SMS	Storage Management Service
TSF	Target System Factory
TSS	Target System Service
UCC	UNICORE Commandline Client
UNICORE	UNiform Interface to COmputing Resources
XACML	eXtensible Access Control Markup Language
XNJS	eXtensible Network Job Supervisor

WG	Working Group
WLCG	Worldwide LHC Computing Grid
WMS	Workload Management System
WS	Web Service

The complete EMI glossary is available at https://twiki.cern.ch/twiki/bin/view/EMI/EmiGlossary .

VI. COPYRIGHT NOTICE

TABLE OF CONTENTS

1. INTRODUCTION

1.1. EXECUTIVE SUMMARY

This document provides a report on the final status of the Compute Area activity after the third year of the EMI project, with respect to the work done to achieve the objectives as described in DNA1.3.3 Technical Development Plan [DNA1.3.3] and DJRA1.1.3 Compute Area Work Plan and Status Report [DJRA1.1.3]. With respect to DJRA1.1.3, in particular, only the status of the work at the time of writing the document is presented. There is no work plan included, unlike the format of the previous document(s).

One of the most important goals of the EMI project, not only within the Compute Area, was to achieve a unified and consolidated middleware distribution out of the major European middleware providers. Much of the effort spent in these three years was devoted to providing a common ground for building e-Infrastructures composed of grid and cloud services developed by the excellence of the European research Institutions and Universities. Conciliating sometimes very different views about distributed computing, different paradigms as well, was not always easy, and required a great deal of work at any level, from the endless discussions about the main roadmap to be followed, to the intense development phase, needed to fulfill the huge number of development tasks planned, to the issues with delivering production quality software in a coordinated and well defined distribution. All these efforts were eventually compensated by the final outcome of seeing the initial products converge towards a unified distribution setting the stage for an interoperable, consolidated, harmonized European middleware.

In particular, the implementation of the final EMI Execution Service specification in the three Computing Elements can proudly be considered an outstanding achievement for the Compute Area work plan and for the whole project. It was not easily obtained, it involved all the major topics about distributed computing and required intense cooperation among different people and teams. The EMI-ES interface was so vast to include a specification for user's requirements, the Activity Description Language, encompassing and unifying a wide spectrum of operations that span from HPC to HTC. This allowed, for example, to define a framework for the execution of parallel jobs solely based on the abstractions defined by the EMI-ES, successfully enabling in this way seamless execution of MPI applications on both HPC and HTC resources. In a similar manner, the EMI-ES also contributed to be the driving force towards the consolidation of the clients and APIs, and it is with great satisfaction that the EMI software distribution will now be able to provide two different consolidated APIs, one for C++ and one for Java, that will be both able to expose job submission and management through the EMI-ES. A tighter integration with security and infrastructure has also contributed, along the same direction, to providing a consolidated distribution built upon the very same basic components that constitute the core of the EMI software stack. This included a common library for authentication, a single site service for authorization, a common service for endpoint discovery and a unified record format for accounting.

1.2. PURPOSE AND SCOPE

This document covers in detail the progress in the development of the Compute Area work plan during the third project year. It is meant to be a reporting/planning document, its style is not the one of an in-depth technical paper. Where relevant a description of the technical aspects of the issue is provided with references to technical documentation. Generally speaking, this document is supposed to mainly focus on management matters.

1.3. DOCUMENT ORGANIZATION

An executive summary of the document can be found in Section 1. Section 2 reports on the progress with respect to the objectives for Compute Area over the third year. Then, there are two more Sections about conclusions and future work. Finally, Section 5 contains the references.

2. COMPUTE AREA STATUS REPORT

This section reports in detail about the progress achieved during the third year with respect to the technical objectives as defined in the Technical Development Plan (DNA1.3.3) [DNA1.3.3] for the Compute Area. Each subsection describes the status with respect to a specific objective.

2.1 CONSOLIDATION/HARMONIZATION OF COMPUTE AREA CLIENTS (DNA1.1.3 C10)

In the reporting period, the task force devoted to developing a consolidation strategy for the compute area clients met in a meeting in Brussels and had several subsequent discussions with the target of a wider evaluation of the portfolio offered by the EMI software and its collaborations in terms of available clients. The unit delivered a detailed and exhaustive design document evaluating several pathways for consolidation [CLIENT_SURVEY]. This was useful to get an overall understanding of the starting point, and will also constitute a valid research for future reference. Among all the possibilities that were identified (which also comprised the SAGA APIs [SAGA_API] and the gLite L&B service [L&B]), the team had to eventually agree on one workable and shared solution. Three scenarios were first selected as the most appropriate ones for an efficient and effective consolidation strategy. After the involvement of the affected product teams and an evaluation of the pragmatic implementation aspects, the final agreement was presented at the All-Hands Meeting in Hamburg; therein work commenced to develop the plan. This plan initially foresaw the adoption of the ARC client and APIs [LIBARCCLIENT] as the reference implementation for the C/C++ consolidated API and command line interface, plus a consolidated Java API to be written to map as close as possible to the ARC API.

The starting point for the C++ library *libarcclient* was a complete yet not consolidated interface, offering added-value features such as resource discovery and resource brokering. It must be noted that its original implementation was not conceived with the intention to expose a high-level API up to the end-user. For this reason, *libarclient* underwent a major re-write in this phase, to make it even more modular (and part of the more general ARC SDK [ARC_SDK]), and also more extensible to any combination of supported interfaces to accommodate support for EMI-ES. Its name also changed to *libarccompute* [LIBARC_CHANGES]. Initially, documentation of the *libarccompute* API was meant for internal use only. This documentation is now available [ARC_API] and integrated in the EMI 3 release for the ARC software [ARC_EMI3]. The *libarccompute* component, released in the EMI 3 distribution, is exposing a consolidated and high level C++ API for end users with full support for the EMI-ES and offering a fully-fledged command line client.

During the documentation phase of *libarccompute*, it turned out the API as it stood could not be mapped to Java directly due to the complexity of the API itself and to certain incompatibilities in the two languages (e.g. C++ class templates versus Java generics). This would have required a further harmonization work to make both APIs as equivalent as possible deep down their internals. It was agreed to pursue a more workable complete and high-level solution. HiLA [HILA_EMIES], the UNICORE high-level API for job submission and management was chosen to be the reference implementation for the EMI consolidated Java API. HiLA is an API for accessing grid resources of different middleware in a consistent manner. It has been implemented for UNICORE versions 5 and 6. An OGSA-BES implementation is also available. HiLA is in use in various places in different grid access libraries. It provides a resource oriented approach to the definition of the Java interfaces and the API uses a *factory* mechanism to dynamically load particular implementations such as the one for the EMI-ES. Location of the HiLA resources are based on URI's where the scheme is used to select a particular implementation. The EMI-ES enabled HiLA was successfully tested against two CREAM and A-REX test instances [EMIES-INTEROP]. From a thin HiLA client built on top of the these APIs, it was possible to issue a delegation from a VOMS proxy certificate, start activities and properly handle input and output, preserving the original VOMS roles. The work on handling delegations required the availability of the EMI common library for authentication purposes, because it is well

known that UNICORE never really used proxies within its own infrastructure, so that this feature had to be implemented from scratch. This caused some delays in the implementation. The development task for the consolidated Java API can be considered as completed. The handling of delegated certificates from HiLA was not delivered with the EMI 3 distribution, while all the rest of the interface could be included in the Monte Bianco release.

2.2 CLOUD STRATEGY IN THE COMPUTE AREA (DNA1.1.3 C11)

The EMI project has always been committed since its very beginning, to studying and evaluating the application of new and emerging computing models to be used side by side with the classic grid computing paradigm. The difficulties in implementing this strategy were mainly in finding a concrete work that could lead to providing a cloud-based solution as part of a 'ready-to-market' software distribution such as the ones provided by EMI already. After careful evaluation process carried out by both a specific task force and the whole project management board, it was decided to include in the EMI distribution a promising service developed at the INFN to provide computing centers with on-demand virtual resource provisioning. Worker Nodes on Demand Service [WNODES] acts at the LRMS level to dynamically provide virtualized and customized computing resources on demand, allowing for full integration with existing computing resource scheduling, policing, monitoring and accounting. Virtualized resources managed by WNoDes can be used to run applications software, interactive analysis, software development, services and so on. WNoDeS lets local, grid and cloud computing converge upon Dynamically Provided Virtualized Resources. In December 2011, WNoDeS was added as a service part of the compute area and its core components, not the full distribution, were included in the EMI 2 release. The full distribution, including components responsible for providing cloud resources were integrated in the EMI 3 release. Cloud provisioning through WNoDeS was shown in the 2012 EGI workshop [EGIWNOD_EGI] and in international events like SC2012. WNoDeS was included in the EGI Fed Cloud solution [EGICLOUD]. Recently, the IGI portal [IGIPORTAL] has started integrating this solution to offer seamless cloud and grid access to its users and scientific communities.

2.3 EMI-ES SERVER SIDE IMPLEMENTATION (C5)

EMI-ES CLIENT-SIDE IMPLEMENTATION (C6)

These two development objectives refer to the implementation of the EMI Execution Service [EMIES] in the three reference EMI Computing Elements and in the EMI clients. For ease of discussion they will be reported altogether. The server-side implementation has always been a strategic commitment of the EMI project and one of its main goals. As a result of the intense work involving both development and finalization of the specification at the same time, all the interfaces to computing resources managed by the three middlewares are now able to handle requests through the EMI Execution Service. This is a major achievement of the project. The implementation followed an iterative approach, as changes in the specification had to be brought while the tests proceeded. After the interoperability tests already started in the second year and the discussions at the 5th EMI all-hands meeting in Budapest (29th - 31st October 2012), there were other rounds of testing triggered by each revised version of the specification. The latest update, which produced version 1.16, was released on 16th November 2012. That could be considered a “bug fix” release of the specification that brought some final touches. At that time, the majority of design issues had already been addressed, so that all the Computing Elements were able to produce in a short time full implementations of the latest release of the EMI-ES specification in the A-REX, CREAM and UNICORE/X computing services. A full report of the interoperability tests can be found here [EMIES-INTEROP]. It addresses method by method the behavior of each client against each server, reporting the issues found and fixes needed.

For the client-side implementation of the EMI-ES, not all the teams were able to provide a fully-fledged integration into their clients for those products less involved in the client consolidation plan.

This objective was affected by the outcome of C10, which was devoted to identifying a roadmap for the consolidation and unification of all the compute clients. The consolidation plan identified two main areas of work for EMI middleware to converge into a consolidated set of client APIs. These were ARC libarcclient and UNICORE HiLA. This new slant taken by the consolidation task relaxed the requirements for CREAM and UNICORE UCC to provide their own full implementations on the client-side implementation of the EMI-ES. These two resulted to be partial and mainly meant at testing interoperability, because the unified client would have been in charge of providing the full implementation of the EMI-ES, as subsequently agreed. WMS, a metascheduler and not a computing service, was not directly involved in the development and testing that started last year, as the specification was premature at that time. In this case the iterative approach could not be followed and the team decided to better focus on other development tasks. The final release of the specification was delivered too late to start working on it.

2.4 COMMON PARALLEL EXECUTION FRAMEWORK (C13)

According to the plan proposed in the second year, the purpose of this task was to implement a common execution framework for parallel jobs making use of the EMI Execution Service to provide a unified interface for job submission and management and offering a common semantics for describing both user activities and resources. The plan described in DJRA1.1.2 foresaw the adoption of the *ParallelEnvironment* element of the *Resources* port-type to create the correct invocation for the parallel environment requested by the user.

The back-end proposed and utilized for the tests was mpi-start [MPI_START]. After steering the development of the EMI-ES for *ParallelEnvironment*, and having waited for the implementation of the whole EMI-ES to be completed and deployed in an integration testbed, the task force started the interoperability tests with the three reference services. Since the first tests, the UNICORE computing service was able to interpret and fulfill requests for parallel jobs expressed and submitted through the EMI-ES. ARC did not immediately prove to support the *ParallelEnvironment* element, instead. Submitting a parallel job was possible, yet through another element of the EMI-ES, *RunTimeEnvironment*, which reflected more closely the ARC internal architecture, although if not in the agreed way. After these tests, the A-REX product team brought the needed changes to enable parallel jobs via the common parallel execution framework, yet this development could not be made part of the first EMI 3 release. The CREAM team also enabled the submission of parallel jobs through the *ParallelEnvironment* element, as requested, only after the content of the EMI 3 release was established, according to the release schedule. So, even for CREAM, the team would need another update to fully include the work developed and tested on the common parallel framework in an EMI release.

2.5 GLUE 2.0 SUPPORT IN JOB MANAGEMENT SERVICES (C8)

In the modular WMS architecture, this work consisted in the implementation of components in the WMS engine able to query over LDAP a GLUE 2.0 enabled BDII, cache the bits of information and enable the match-making based on GLUE 2.0 attributes. These modules are called 'purchasers' in the WMS architecture. A conversion from LDIF to Classad [CLASSAD] is performed by the WMS engine and the results are then stored into an internal cache, the so called Information Supermarket (ISM).

The LDAP GLUE2.0 purchasers have been implemented in the WMS, also designed is such a way that no endpoint publishing both v. 1.3 and 2.0 GLUE is accounted twice. This feature required a further profiling activity in order to implement the necessary agreements to identify a unique queue from any *ObjectClass*.

The work for Storage Elements (*objectclass GLUE2StorageService*) was completed by adding support for the missing GLUE 2.0 attribute *StorageAccessProtocols*.

Match-making and GLUE2.0 interoperability were tested with the EMI 2 version with no issues. Also, as part of the restructuring of some internal data structures required by the newest design, some inefficiencies were spotted and fixed. This work was beneficial for the overall performance, even with the GLUE 1.3 purchasers that also populate the ISM. This objective can be considered achieved at 90%. The remaining 10% concerns the ability to automatically select resources on both compute and data requirements. During the development it was observed the amount of data produced collecting GLUE 2.0 information from a production BDII (egee-bdii.cnaf.infn.it) was high as GLUE 1.3, of the order 130,000 entries (about 112,000 entries for 5000 Shares and about 22.000 entries for GLUE2StorageShare). The WMS caches all the ISM in memory and, with the advent of 64-bit architectures, always suffered from an excessive growth of the memory footprint (VSZ, virtual memory size) especially under high load. This fact is also stressed by the use of a multi-threaded architecture that insist of the same data structure. Memory fragmentation is a well-known side-effect. With GLUE 2.0, the amount of data produced was high enough that this issue became so evident to impact on the quality of the service, as an excessive memory consumption leads in general to swap activity in the node and causes a *denial of service* from the user point of view. A filter expression was added in the configuration, in order to mitigate the problem and provide administrators with the ability to exclude information from VOs other than the managed ones, just as an example, or from any other unrequired GLUE 2,0 parameter. That said, implementing the missing feature to query both for compute and data requirements altogether, however, would have further increased the size of the ISM, in a multiplicative fashion, given that each *ComputingShare* would have to be linked to the representations of all the *StorageElements* that are bound to it. To conclude, the situation is bearable with the present implementation and a deeper restructuring of the WMS caching mechanisms would be required to deal with the volume produced by GLUE 2.0. The current implementation of match-making with data would have led to scalability issues that would have compromised the whole development task and the reputability of the GLUE schema as a whole, from a user's point of view.

2.7 ACCOUNTING RECORD IMPLEMENTATION (C9)

This development objective is about the implementation of the specification for the EMI Compute Accounting Record [CAR] produced in the previous reporting period. The implementation took place as planned in the third year, and all the affected components were able to fulfill their development task and released with EMI 3. The implementation phase also triggered a new round of discussions that led to an updated release of the specification, version 1.2.

2.8 SUPPORT FOR EXCLUSIVE NODE OR MULTI-CORE ALLOCATION (C14)

This development objective was cancelled. The cancellation does not imply the present middleware stack is not able to provide support for multi-core jobs and/or exclusive allocation. It means no further harmonization work could be accomplished. It has to be noted the EMI-ES specification provides mechanisms to specify both multi-node and exclusive allocation in the *Resources* port type through the attributes *ExclusiveExecution*, *NumberOfSlots* and *SlotsPerHost*, as required by the task. The EMI Execution Service can be considered, as it happened for the parallel framework and the client consolidation tasks, the main harmonization effort devoted to interfaces to computing resources. Using the EMI-ES for harmonizing the access to different computing paradigms and scenarios is already shown to be the correct way to go, as it successfully happened for objective C13, for example.

2.9 PROCESS JOBS WITH DIFFERENT CHARACTERISTICS (C15)

This development objective was cancelled. The situation is similar to the task above. This low priority objective was not considered worth spending effort, because, even in this case, the *Resources* port type allows for passing environment variables on to the LRMS through the attribute *RuntimeEnvironment*. The harmonization work that was required by the supplier of this requirement was met in the design

and development of the EMI-ES. The EMI-ES will allow for passing Resource Constraint Tags down to the LRMS in an agreed way, allowing sites to customize scheduling if they desire so.

2.10 INCREASE PERFORMANCE (X15)

As mentioned in previous reports, this development objective was left generic enough to allow evaluating specific requirements from the user communities at any time. If accepted it could lead to a new development task. The already existing documentation task about deployment, best practices and performance of the WMS required by the EGI project was addressed in the reporting year [WMS_PRACTICES], while, in the meantime, A-REX was requested to improve its handling of cache and session directories, to enable faster job processing. This was partially achieved by splitting status information per state in the directory structure and by implementing a limiter mechanism that controls the maximum number of active tasks.

2.11 EVOLVE EMI COMPONENTS (X16)

In this objective, several tasks to deliver various improvements of the existing services were grouped together. These are: enabling logs from CREAM by the L&B service, implement a high-availability cluster solution for CREAM and improve the A-REX back-end scripts.

The first task refers to the ability to support logging compute related events to the L&B service from the CREAM Computing Element. On the Logging and Bookkeeping (L&B) part, the pre-existing (pre-EMI) CREAM state machine implementation was extended to cover additional job state attributes and support jobs that were submitted to CREAM through the WMS. L&B then receives events from traditional (WMS, job wrapper) as well as new (CREAM) sources. For the latter, events are correctly interpreted and events logged by CREAM are used to provide more fine-grained status information for the job. To simplify deployment, the L&B's Java client package was split into two (local logging code to be used by CREAM and remote logging/querying operations that depend on Java-based authentication libraries and are not required on a CREAM machine). Minor adjustments had to be done to L&B's *interlogger* configuration to allow simultaneous access to the spooling directories by CREAM and the *interlogger*.

On CREAM side, logging calls had to be integrated, to generate L&B events at correct times and set correct attribute values. Logging to L&B on CREAM can be configured and activated/deactivated by the administrator. Since CREAM does not use any service registries for automated configuration, the appropriate L&B endpoint must be set by the site administrator manually. The native CREAM event monitoring capability is present in all currently supported L&B server releases, i.e., v3.2 available from EMI 1 and EMI 2, and v4 available from EMI 3. The appropriately modified Java client component will be released as an update of EMI 3. The native CREAM event logging capability will be supported by the upcoming update of CREAM, to be released with the first EMI 3 update.

For what concerns the implementation of a highly-available cluster solution for the CREAM CE, a design document explaining the architecture of the solution was written. This document was also forwarded to EGI for comments, on the third project year. This document [CREAM_HA] identifies all the components that need to be made able to work in a distributed environment with no single point of failure. For what concerns third-party technologies (i.e. shared file systems, DNS-based load balancing, etc.), it suggests possible solutions and layouts that can be implemented directly by the system administrator. Then, it proceeds analyzing two components in the global CREAM architecture that are more directly affected by this development task and need explicit work. They are the front-end Java application that runs in an *Axis2* container deployed in a *Tomcat* application server, and the back-end BLAH, a lightweight modular service that directly interacts with the LRMS. The front-end accepts requests, queues them in a local priority queue and stores static and dynamic job information in the database, including proxies while the job sandboxes that contain all input and output data files accessed and produced during the job's life cycles are stored in the local file system. With this latest

development, the command queue can now be shared by more CREAM instances and is not exclusively local. Every instance will process requests independently by picking up requests from the same location. This capability was fully implemented and no further development is needed. Furthermore, given that by design CREAM is a stateless web service which treats each request independently, no explicit session replication is required. The work on the back-end had to be faced in a different and more complex way to respect BLAH's requirement of lightweight implementation that ships with other distributions, as well limits the adoption of third-party software components (e.g. SQL databases). In a HA cluster environment, each BLAH daemon running on different CREAM nodes kept its registry synchronized with all other nodes. A new feature was added to avoid proliferation of the queries to the LRMS. Whenever it refreshes the local registry from the LRMS, a BLAH instance now also sends the updated information either to a multicast address or to a list of unicast addresses where other instances will listen and, in turn, update their registry without having to send new requests to the batch system. This synchronization mechanism guarantees that new BLAH instances are updated whenever they are instantiated using a custom solution relying on low level libraries only. From what has been described, there is still some work remaining to finalize the front-end. More precisely, a cluster solution for the MySql database has to be identified, configured and thoroughly tested. Secondly, the CREAM front-end still needs more development. The CREAM cluster architecture is not a share-nothing system, and in order to minimize to load on the underlying layers, a single instance needs to be elected as the master node, that is the one and only node that responsible for querying the BLAH pool. This last work is still under development. The improvements to the front-end and back-end are being distributed in the EMI 3 release, but they cannot be considered to be a fully-functional to provide a highly-availability cluster for CREAM. This will only happen when a complete solution can be deployed and tested. This will still require thorough testing of the MySQL Cluster Database Engine, and some missing login in the CREAM interface itself.

Another development planned for this task was the support to the SLURM batch system in CREAM. At the time of writing, the CREAM product team has developed in its back-end BLAH the necessary translations to provide support for job submission and management to the SLURM batch system. The work is being released in EMI 3 and it comes will all the required configuration tools to be considered self-contained and completed. The work to provide CREAM with accounting sensors for SLURM assigned to the APEL PT was not provided as the team experienced delays in the development of the SLURM parser. As a result, the CREAM team decided to write the accounting sensors on behalf of APEL, so as to consider the support for SLURM fully functional. This work required a change to the planning to address delays introduced by the APEL team. As a result, it will only be released by a subsequent update.

For ARC, a cleanup work of all the A-REX back-end scripts that interface with the LRMS was planned and carried out. This activity consisted in making sure that these scripts interpret the input variables in the same way, in identifying functionality common to most of the back-ends and factorizing it out. This improved the sustainability of the service as a whole.

2.12 MIGRATION TO AUTHENTICATION LIBRARY (X17)

This development objective mandates that each EMI compute service and its clients has to handle authentication and all related operations based on the EMI authentication library [EMIAUTHLIB] (for example, security token handling, such as X.509 certificates or SAML assertions, for issuing and checking activity requests, checking of the credentials etc.). How backwards compatibility is addressed is left to each product team. The Compute Area products affected by this development are the following:

1. ARC: A-REX, ARC clients
2. gLite: L&B, CREAM, Gridsite, proxy renewal
3. UNICORE: UNICORE/X, UCC, HiLa

For UNICORE, the interface UNICORE/X, the clients UCC and the high-level API HiLA have been fully ported to the EMI authentication library in this period. They will be in the EMI 3 release.

For ARC, a core module is being utilized by A-REX and other services. This component, ARC-core, is not under direct responsibility of the compute area, however A_REX relies on it. The work on the migration to the EMI authentication library caNl++ can be considered to be still at an early stage and not ready to be released in EMI 3.

For gLite, CREAM completed the development work but the team won't be able to release it in EMI 3. From the development point of view, nonetheless, the task can be considered to be accomplished and can be released as a subsequent update. For WMS, it turned out that no direct work was affecting this task as WMS fully relies on L&B, CREAM APIs and Gridsite for authentication. In turn, these needed to be ported. The adoption of caNl in L&B was not done in time for EMI 3. However, L&B's GSS layer was implemented previously by the caNl-c implementation team and both products share features and code. Thus, appropriate functions are implemented by L&B already. caNl adoption is still being considered as a way to reduce code and maintenance effort. Gridsite has fully achieved the goal and it will also distribute it in the EMI 3 release. L&B and caNl-c security features are gradually being aligned to allow for seamless replacement in L&B's. The team was waiting for the EMI 3 version of caNl to be available, and cannot release the full work for EMI 3. Proxy renewal as well still needs to be completed. GridSite was heavily refactored using caNl, resulting in a GridSite 2.0 release. All code duplicated by caNl was removed. Further code reduction was achieved by removing rarely used and issue-prone functions such as gsexec, slashgrid, gridsite-copy.cgi and gridsite-storage.cgi. The benefits of adopting caNl in ProxyRenewal were negligible and the team decided to cancel this task.

2.13 PORT, RELEASE AND SUPPORT ON AGREED PLATFORMS (X7)

All Compute Area components are required to fulfill this task for each and every platform endorsed by the project. The following platforms were identified by the project: Debian 6, Scientific Linux 5, and Scientific Linux 6. The vast majority of the compute area components were able to release their software for the aforementioned platforms. A summary table at the end of this section is provided for reference. WMS and its UI have not been released on Debian 6. They both underwent an intense restructuring of the build system that led to a significant reduction of build modules and produced artifacts and to a more simple and effective way of producing the build itself. All this in view of the retirement of the ETICS [ETICS] build system. Before this work, WMS was fully relying on ETICS to produce a build. Now a simple script can be downloaded to produce both native and 'mocked builds' in both Red Hat and Debian style. A migration from autotools to cmake was also accomplished for the WMS UI and to a great extent for the server as well. The work to produce Debian 6 artifacts was also completed for the UI. Even though locally tested, this could not be released in EMI 3 due to lack of availability of some dependencies in the EMI 3 repository. The team was very satisfied with this work which greatly helps in sustainability. Each present and future platform will now be managed far more easily than what happened in the past, as long as the external dependencies will be made available. L&B also is functional in 32-bit and 64-bit SL5, 32-bit and 64-bit SL5, and 64-bit Debian 6. The only issue concerns YAIM (resource BDII) on Debian, where support is missing for the glite-info-provider-service package. Tests based on an alien conversion of that package, however, have passed.

Component	SL5	SL6	Deb6
A-REX	Yes	Yes	Yes
ARC Clients	Yes	Yes	Yes
EMI-UI	Yes	Yes	No

Gridsite	Yes	Yes	Yes
L&B	Yes	Yes	Yes
Proxy Renewal	Yes	Yes	Yes
U. HILA	Yes	Yes	Yes
U. TSI	Yes	Yes	Yes
U./X	Yes	Yes	Yes
WMS	Yes	Yes	No
WMS - UI	Yes	Yes	No

Table 1: Availability of each EMI component for the operating systems Scientific Linux 5 (SL5), Scientific Linux 6 (SL6) and Debian 6 (Deb6)

2.14 ADHERE TO OPERATING SYSTEM STANDARDS (X6)

For EMI major releases and updates, i.e. EMI 2 and EMI 3, the EMI SA1 release and SA2 quality assurance processes assure that the delivered software complies with certain well stated acceptance criteria. Among these, the compliance to the operating system standards for the location of log files, configuration, libraries, and executables are checked against the FHS specification [FHS] each time a new product is released. This applies with no exception to the Compute Area components and services.

2.15 EMIR ROLLOUT (X18)

Following on from the first release of EMIR with EMI 2, a Domain Service Registry was installed on the testbed and it has been verified that all EMI services in the EMI 3 release will be able to publish to EMIR. The SERP [EMIR_SERP] has been produced in the Infrastructure Area to allow EMI service to publish its existence to the EMI Registry. For ARC, a feature was added so that the service record could be obtained by extracting the required information from the ERIS, while the same was done for gLite in order to extract service record information from the BDII. This reduced the configuration overhead for integration. All the Compute Area services and their clients were able to integrate the EMIR producers in their distribution, providing detailed instructions for administrators on how to enable the publication of the service records.

2.16 MANDATORY CONFIGURATION VARIABLES (X19)

As requested by the EGI user community, all the EMI services should document in detail the mandatory variables for configuration, their type, meaning, any default values for the non-mandatory ones and all that is needed to successfully configure a service. No incidents have to be reported for this task, and all the teams were able to provide what was needed.

2.17 STANDARD SUPPORT FOR CRL HANDLING (X21)

This development objective was added to the work plan to make sure each service is able to import Certificate Revocation Lists (CRL) from the EMI distribution and each accepted request is checked

against such CRL during authentication. CRL updates quite frequently, a few times within the day, and their refresh must be done without causing service interruption. Either through the integration of caNl that supports this feature in the API it offers or other means, all the involved services were able to satisfy this requirement. For the Java based gLite services, a component called TrustManager is responsible for re-loading the CRL in the application. TrustManager was adapted to implement this functionality using the features offered by caNl. In WMS, a C++ gLite service, a graceful restart of the Apache container will cause to flush all the pending requests, update the CRL, and accept new requests with the newly loaded CRLs. UNICORE and ARC support for 'hot CRL re-loading' was achieved through the EMI caNl component.

2.18 SERVICE MIGRATION AND HOT SWAPPING (X22)

This involved producing guidelines addressing how to migrate a service while preserving its state and, wherever supported, how to set up a 'hot-swap' infrastructure, that is the ability replace, turn on/off single instances of a service without compromising its overall availability. This is of interest to the EGI operations. After a first assessment of the situation, it turned out a general review of the documentation to provide a unified and simple explanation of these topics was needed. All EMI computing services, A-REX [AREX_MIGRATION], CREAM [CREAM_MIGRATION], L&B, WMS [WMS_MIGRATION], UNICORE/X [UX_MIGRATION] are now able to offer clear documentation, stating if they support service migration/hot swap or high-availability and load sharing, and what operations are required to a site administrator (i.e. what file system mounts need to be persistent, where paths are specified in the configuration, etc.) to install the required layout, to recover in case of failures and where special hardware is needed.

2.19 DEFINE CONSOLIDATED API SET (X23)

The Consolidated API set for the Compute area services has been identified as the harmonized client API (C10) and the EMI-ES API (C6). Refer with the relevant sections in this document.

2.20 INTERACTIVE ACCESS IN COMPUTING ELEMENT (C3)

Referring to the possible directions for investigation described in DJRA1.1.2, this task is about documenting what features the computing services support to allow interactive access in the Computing Element. A survey revealed the three stacks are all able to provide interactivity in various ways to the extent provided by their security policies. ARC, gLite and UNICORE allow to inspect the output files while the job is still running. For the provisioning of shell-like access that can be accomplished through well-known third-party tools it is within reach with the present software stack whenever connectivity to/from the worker node is allowed and wherever the security mechanisms in place at sites allow it. There are policies in the middleware itself that prevent from having interactive access with a running job. gLite, WMS and CREAM support shell-like access without constraints other than file peeking. This can be done through a third-party tool [I2GLOGIN] or via a simple job script provided in the documentation. For ARC the only kind of interactive access is done through its session directory using the URL supplied during job submission and the subsequent data management tools to deal with it. This is well documented in the provided references, and there are specific tools that allow to watch standard output and error for a job. In the very same way, UNICORE allows interactive access to the job session directory, as described in the UCC user manual [UCC-DM].

3. CONCLUSIONS

There are several considerations that can be expressed to conclude on the results of the work accomplished in both this final year and in the whole project lifetime. The overall degree of achievement of the development objectives, taken one by one and matched with the initial planning, can satisfactorily be considered to have reached a very high level of completion compared to the number and complexity of all the items to be honored in the initial 'wish list'. More work would indeed be needed for some low priority or not completed objectives. There are no technical obstacles to finalizing the details of the remaining tasks, as these have been discussed in the third year. A full and tested completion of the remaining work would probably require another three to six months. This process has already started at the time of this writing and an update of EMI 3 distribution is being delivered.

There are additional successes to be accounted for. The inclusion WNoDeS as cloud-based resource provisioning solution for sites is a clear illustration of how proactive the project was in coping with evolving demand in terms of new paradigms for distributed computing. The design of EMI-ES, the result of the harmonization work of the major three European middleware providers, is paving the way for a wider standardization able to span across several paradigms - from single batch processes to multi/many-core applications to be run in highly specialized HPC environments or in cloud-enabled IaaS/SaaS scenarios. This can be further fostered by the adoption of the EMI-ES as the basis for submission frameworks and portals that could possibly be evolved and specialized, able to tap into the processing power of a wider range of platforms, able to run efficiently by communicating configuration setups about both jobs and resources to and from the LRMS.

There are details at various levels to be defined to deliver a sound reference for present and future infrastructures that want to inter-operate and effectively support a wide range of applications. EMI has presented a valid technological basis. In an ever-changing world, where especially in ICT new challenges are encountered daily and require prompt reaction, the outcome of the current work goes hand in hand with the way future technological upgrades can be handled.

4. FUTURE WORK

The advent of the grid and more than ten years of European projects providing technological solutions for distributed computing have led to the development of the EMI consolidated middleware stack composed of a plethora of now more than ever coordinated and standard services that made and still makes possible the execution of important scientific computations. This also led to results of paramount importance, such as the detection of Higgs boson-like particles at the LHC, something EMI is proud of. EMI has contributed to the processing of huge volumes of data all across the world. It is undeniable a great deal of advances in all science and engineering domains come from computing. The High Energy Physics experiments that shaped the architecture of the present grid middleware, is composed of large and federated collaborations with demanding requirements in terms of both computing and storage. This has led to the development of cutting-edge technology on one side, but it also requires high-skills and non-trivial effort to be set up and maintained. This simple fact reveals one main direction for the future evolution of the middleware. Big scientific collaborations steered the development of highly-specialized and performing services, yet this was not always achieved with usability for small/medium sized collaborations in mind. The evolution of the distributed computing paradigms coming from the industry is representing an outstanding opportunity to provide even higher levels of efficiency while keeping the software open, standard and easy to maintain. In this respect, the roadmap for the next years is twofold: decrease the *digital divide* among the various scientific communities on one side, evolve the middleware towards new and more dynamic computing paradigms on the other one.

Consider the security infrastructure needed to set up a Virtual Organization, for example. Although providing a sound security framework and enabling federation, single sign-on, etc., more dynamic computing paradigms like *clouds* will require an easier yet equally safe approach to providing a security framework. Nonetheless, this is not something that present *cloud-based infrastructures* are able to offer and represents an open challenge. Re-consider the role of the present grid Computing Element with its well-established authentication mechanisms, potentially able to provide a more general and dynamic access to computing and storage than the classic abstraction based on the concept of *job*, might be a viable option for the near future. Current grid middleware computing services were mostly conceived as interfaces to the traditional batch systems. These, in turn, are constantly evolving due to changes in the underlying hardware technologies such as virtualization, system software, and usage patterns. The CE will have to evolve with the underlying resources and use cases. Inevitable requirements to make efficient use of multi/many-core architectures should result in functionality that is natively built into the middleware, not merely added as an external component. In this respect, the work of EMI to create a *de-facto* and pragmatic standard for the European Compute Elements puts our software in an advantageous position with respect to other, less limited, initiatives.

Taking into account the complex machinery needed to set up a VO-wise workload management framework, all the major HEP experiments utilize a centralized framework to gather and process all the requests coming from their users. The framework is kept agnostic of the complexity of the underlying systems in terms of computing paradigm (grid, cloud, volunteer, etc.), technology stack, and data distribution protocols. Resource provisioning is always decoupled from job management, recurring to the so called 'late-binding' approach in most cases. Various frameworks are nowadays used by major HEP experiments, yet there is no standard handling of security nor a common protocol for the implementation of late-binding. Moreover, they have evolved independently, oftentimes have grown outside the mainstream middleware projects such that these have developed without complying with well-defined standards. They basically provide the same solution, each with different dependencies and technology stacks, and have complex architectures. For a small institution to perform accurate workload on its payloads, it would require time, complex hardware and network configurations, and devoted people to properly set up a VO framework. Under this light, there could be great work to be done towards a consolidation and standardization of the workload management frameworks. This is not immediately felt as urgent by each single major community, as long as maintenance costs are not immediately visible. It will become a hot topic in the next few years, also in the context of a world-wide and European economic recession phase. The present middleware must be

made able to cope with more dynamic designs, such as the ones offered by most VO frameworks already. Only a global initiative could provide the required momentum towards consolidation, being able to act in a more common and shared interest. In both cases there is room for improvement.

On the client side, the development of user-friendly interfaces adapted for a variety of popular platforms is a major challenge. Given the multitude and variety of scientific applications, it is unlikely that a single tool suitable for all cases will ever be created. A solution could be the provision of a Grid Software Development Kit, which would simplify and speed up development of science-specific end-user tools. EMI has started work in all these fronts, which creates a solid basis for future developments.

The work in providing more usable and consolidated services has been started in EMI. Making the software more usable and efficient does not only involve middleware providers. It is a complex activity that requires a tighter integration among technology providers, integrators, all the various DCIs and in particular user communities. Several models have been applied until now, but any future work cannot ignore evolving towards even more fruitful pathways for collaboration.

5. REFERENCES

ARC_API	https://docs.google.com/document/d/1n7QDqSbE-m6RWtFcgDgaQvMVuMoHVm65-vM6-9Az1rQ
ARC_SDK	http://www.nordugrid.org/documents/code/13.02/
ARC_EMI3	http://www.nordugrid.org/arc/releases/12.05/release_notes_12.05.html
A-REX	www.nordugrid.org/documents/arex_tech_doc.pdf
AREX_MIGRATION	http://svn.nordugrid.org/trac/nordugrid/export/25326/doc/trunk/manuals/CE_sysadm/arc-ce-sysadm-guide.pdf
CAR	https://twiki.cern.ch/twiki/bin/view/EMI/ComputeAccounting
CLIENT_SURVEY	https://twiki.cern.ch/twiki/bin/view/EMI/EmiJra1T2ComputeClientSurvey
CREAM	http://grid.pd.infn/it/cream
CREAM_HA	https://documents.egi.eu/public/ShowDocument?docid=1494
CREAM_MIGRATION	https://wiki.italiangrid.it/twiki/bin/view/CREAM/SystemAdministratorGuideForEMI3#4_Migration_of_a_CREAM_CE_servic
DJRA1.1.1	http://cdsweb.cern.ch/record/1277608
DJRA1.1.2	http://cdsweb.cern.ch/record/1277610
DNA1.3.1	http://cdsweb.cern.ch/record/1277540
DNA1.3.2	http://cdsweb.cern.ch/record/1277543
DNA1.3.3	https://cdsweb.cern.ch/record/1277544
EGIWNOD	Elisabetta Ronchieri, Giacinto Donvito, Paolo Veronesi, Davide Salomoni, Alessandro Italiano, Gianni Dalla Torre, Daniele Andreotti, Alessandro Paolini, "Resource Provisioning through Cloud and Grid Interfaces by means of the Standard CREAM CE and the WNoDeS Cloud Solution", PoS (EGICF12-EMITC2) 124, 2012.
EGICLOUD	https://wiki.egi.eu/wiki/Fedcloud-tf:FederatedCloudsTaskForce
CLASSAD	http://research.cs.wisc.edu/condor/classad/
ARGUS	https://twiki.cern.ch/twiki/bin/view/EGEE/AuthorizationFramework
GLUE20	http://www.ogf.org/documents/GFD.147.pdf

EMIR	https://twiki.cern.ch/twiki/bin/view/EMI/EMIRegistry
EMIES	https://twiki.cern.ch/twiki/bin/view/EMI/EmiExecutionService Specification: https://twiki.cern.ch/twiki/pub/EMI/EmiExecutionService/EMI-ES-Specification_v1.0.odt Schema https://twiki.cern.ch/twiki/pub/EMI/EmiExecutionService/es-schemas-1.0-src.tar
EMIES-INTEROP	https://twiki.cern.ch/twiki/bin/view/EMI/TestPlan26
EMI_JAVA_APIS	https://github.com/eu-emi/compute-client-java
EMIAUTHLIB	https://twiki.cern.ch/twiki/bin/view/EMI/EmiJra1T4SecurityCommonAuthNLib
EMIR_SERP	https://twiki.cern.ch/twiki/bin/view/EMI/SERP
ETICS	etics.cern.ch:8443/etics
LIBARC_CHANGES	http://wiki.nordugrid.org/index.php/API_changes#libarccompute
LIBARCCLIENT	http://www.nordugrid.org/documents/client_technical.pdf
FHS	http://www.pathname.com/fhs
GRIDSITE	http://www.gridsite.org
IG2LOGIN	http://grid.ifca.es/wiki/Middleware/i2glogin
IGIPORTAL	M. Bencivenni, et al, "A portal for an easy access to the IGI grid infrastructure," poster 9 at EGI TF, 19-23 September 2011, Lyon, France
LSF	http://www.platform.com/workload-management/high-performance-computing
L&B	http://egee.cesnet.cz/cs/JRA1/LB/
HILA	http://unicore-dev.zam.kfa-juelich.de/documentation/hila-2.4.0-SNAPSHOT/hila-grid-api/hila.html
HILA_EMIES	http://unicore-dev.zam.kfa-juelich.de/documentation/hila-2.4.0-SNAPSHOT/hila-grid-api/hila.html#_emi_execution_services
MPI_START	https://github.com/IFCA/mpi-start
PBS	http://www.openpbs.org
SLURM	https://computing.llnl.gov/linux/slurm
SAGA_API	http://saga-project.github.com/saga-cpp/

UNICORE/X	http://www.unicore.eu/documentation/manuals/unicore6/files/unicorex-6.4.0/unicorex-manual.html
WMS_PRACTICES	https://wiki.italiangrid.it/twiki/bin/view/WMS/WMSBestPractices
WMS_MIGRATION	https://wiki.italiangrid.it/twiki/bin/view/WMS/WMSSystemAdministratorGuide#3_Service_Migration
UCC-DM	http://unicore.eu/documentation/manuals/unicore6/files/ucc/ucc-manual.html#ucc_datamanagement
UX_MIGRATION	https://unicore-dev.zam.kfa-juelich.de/documentation/unicorex-6.6.0/unicorex-manual.html#_migration_of_a_unicore_x_server_to_another_physical_host
WNODES	Davide Salomoni, Alessandro Italiano and Elisabetta Ronchieri, "WNoDeS, a tool for integrated Grid and Cloud access and computing farm virtualization," J. Phys.: Conf. Ser. 331 052017, 2011, doi:10.1088/1742-6596/331/5/052017