D-SA2.1: VERCE platform integration: first selection and integration of services and tools and report of the management release process

15/05/2012

Project acronym:	VERCE
Project n°:	283543
Funding Scheme:	Combination of CP & CSA
Call Identifier:	FP7-INFRASTRUCTURES-2011-2
WP:	WP6/SA2, Integration and evaluation of the platform services
Filename:	D-SA2.1.pdf
Author(s):	S.H. Leong (BADW-LRZ), P. Martin (UEDIN), H. Schwichtenberg (SCAI), A. Gemünd (SCAI) and A. Spinuso (KNMI)
Location:	http://www.verce.eu/Repository/Deliverables/RP1/
Type of document:	Deliverable
Dissemination level:	Public
Status:	Final
Due date of delivery:	15/05/2012
Reviewer:	G. Erbacci (CINECA) and I.A. Klampanos (UEDIN)
Keywords :	Evaluation, Integration, Testing, Tools, Services, Application codes, Software components, Release Management Process, PDCA Cycle

Version	Author	Date	Comments
0.1	S.H. Leong (BADW-LRZ)	09/03/2012	Initial draft for comments
0.2	P. Martin (UEDIN)	06/04/2012	ADMIRE related information
0.3	H. Schwichtenberg, A. Gemünd, S. Claus (SCAI)	13/04/2012	EGI related information
0.4	A. Spinuso (KNMI)	15/04/2012	Testing framework
0.5	S.H. Leong (BADW-LRZ)	15/04/2012	Updating the information, integrating the modifications and restructuring the content.
0.6	S.H. Leong (BADW-LRZ)	02/05/2012	Incorporating suggestions and corrections from the internal reviewers. Changes to align with D-SA1.1.
1.0	S.H. Leong (BADW-LRZ)	14/05/2012	Incorporating final comments with regards to the Executive Summary and the general format of the deliverable.

Copyright notice

Table of Content

List of figures

List of tables

Executive summary

One of the objectives of the VERCE project is to provide a service-oriented architecture and framework that wraps the data-infrastructure resources and services with a set of distributed data-aware Grid and HPC resources provided by the European e-Infrastructure and community. To this end, the tools, services and application codes, i.e. software components, which are particularly relevant to the seismologists and the Earth Science community will be considered. In addition, software components that are already adopted by the existing European e-Infrastructure and service providers like EGI, PRACE, Mapper and IGE will also be explored, evaluated and selected to be integrated into the VERCE platform.

The main aims of this reporting period are to define the release management process to enable the effective cooperation of the work packages to select and release software components, to define and plan procedures to implement the release management process and to identify an initial portfolio of software components to provide the basic functionalities to enable VERCE members to access the test bed.

The Plan-Do-Check-Act (PDCA) cycle is selected to manage the release process. Each cycle is estimated to be one year with two overlapping cycles that occur simultaneously to facilitate a six-monthly release of the platform. In order to mitigate the risk of missed releases, a release schedule and recommended work practices are defined.

SA2 procedures in accordance with the PDCA cycle are further defined to effectively evaluate and integrate the software components into the test bed. A "Software Component Release Request Form" is prepared to improve the communication between the requesting work package and SA2 when a release request is submitted. Generic and software component specific tests are also defined and planned to ensure that the accepted software components are of a recommended quality and are properly integrated into the VERCE platform. Key performance indicators that could be used to assess the quality of service of the platform are defined and shared with SA1 to assist in the definition of the monitoring metrics.

To allow VERCE members to access the test bed and perform development, evaluation and integration work, an initial (fast track) portfolio of software components is selected from a list of potentially useful software components studied by JRA1 and JRA2, and a list of currently available software components on the test bed gathered by SA1. Priority is given to common components that are currently available on the test bed to support immediate access, and development and integration work. In addition, priority is also given to components that are required to support the work of the JRAs in the next reporting period. The selected components are summarised as follows:

Secure User Access

- X.509 certificate-based authentication

Resource Management

- VERCE Gateway (web portal)
- Resource Database (possibly LDAP)

Job Submission

- Grid Middleware: GLite, Globus Toolkit or UNICORE

Data Management

- ADMIRE gateway
- OGSA-DAI
- GridFTP

Finally, the issues faced in this reporting period are shared and resolutions are proposed or made. The future work of SA2 includes refining and increasing the agility of the release management process,

managing the first official VERCE release, and evaluating and testing the proposed software components from SA3 and the JRAs.

1. Introduction

The main objective of the SA2 work package is to enhance and expand the capabilities of the VERCE research platform by integrating software components, i.e. application codes, client services and tools, in accordance with the requirements of the seismology and other Earth Science research communities. To fulfil this objective, two main activity areas were identified in the DoW:

- Assembly of tools, services and application codes
- Management of the release process

The process to assemble and release software components requires the management and coordination of activities among numerous work packages. In order to define and thus manage the release process, activities of work packages that directly interact with SA2 are elaborated in Figure 1. The activities are numbered to illustrate the workflow of the release process.

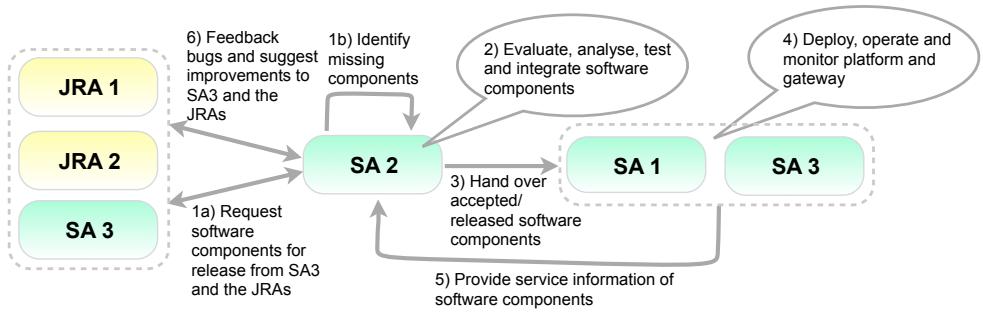

Figure 1. Workflow of the release process

SA2 receives software component release requests from SA3 and the JRAs, which in turn receive the requirements or use cases from NA2. In parallel, SA2 attempts to identify the missing components that are required to allow SA1 to operate a fully functional VERCE platform. All software components are screened and tested by SA2 before they are recommended to SA1 and SA3 for deployment. SA1 will deploy, operate and monitor the VERCE platform, i.e. the software components, that are assembled while SA3 will deploy, operate and monitor the scientific gateway. While operating the platform and the gateway, monitoring data, i.e. service information, is collected. Detailed information about the service information can be found in D-SA1.1. The information is shared with SA2 so that issues and possible improvements can be identified. The finding is fed back to the respective SA and JRAs as change or feature requests.

In order to facilitate the above activities, a release management process has to be defined to better manage the release and thus the assembly of the software components. In Section 2 of this deliverable, the release process based on Plan-Do-Check-Act (PDCA) cycle is thus defined. In Section 3, SA2 procedures adhering to the PDCA cycle are shared. In Section 4, an initial list of potential software components gathered by the JRAs and SA1 are shown. The fast track portfolio that is selected, the tests to be performed and the definition of performance indicators are defined. In Section 4, the organisation of work in SA2 is clarified. Finally in Section 5, the issues encountered and the planned work in the next cycle is discussed.

2. Release Management Process

The management of the release process is crucial to support the operation of a stable and encompassing VERCE platform. Defining a Management Release Process is essential for the efficient management of the evaluation and integration of the candidate software components, i.e. tools, services and application codes. In the process of managing a release, many issues, e.g. software defects, issues, risks, software change requests, new development requests, and deployment and packaging, have to be taken care of. As such, the process of handling a release should be agile to ensure that the issues do not delayed the release process.

To facilitate an agile management release process, the comprehensive IT Service Management framework; Plan, Do, Check, Act (PDCA cycle) as shown in Figure 2 is adopted to assist the release process. The PDCA is recommended by ISO 20000 and ITIL.

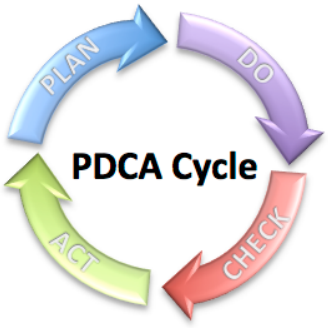

Figure 2. PDCA Cycle

The PDCA cycle is chosen, as it is known to help to reduce risks, improve communications, and increase productivity and efficiency as it leverages on a proven set of best practices.

2.1. PDCA Cycle

The PDCA[1] cycle consists of 4 phases; Plan, Do, Check and Act. The cycle assists the team to implement, operate, monitor, review, maintain and improve the software components we release. Figure 3 presents the workflow of the release process that is elaborated in Figure 1 by categorising the activities into the 4 phases of the PDCA cycle.

[1] http://labspace.open.ac.uk/mod/resource/view.php?id=346003

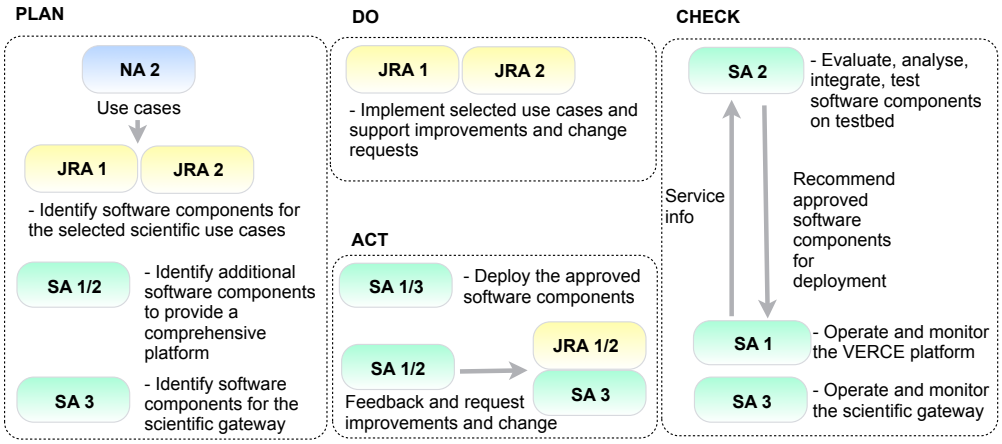

Figure 3. Release Management Process: PDCA cycle

In the "Plan" phase, the seismic and platform requirements, i.e. new features, bugs and improvements, are fed back by NA2, SA1 and SA2. NA2 will focus on providing the scientific use case requirements while SA1 and SA2 will address the requirements to provide a comprehensive distributed research platform. The identified requirements and change requests are forwarded to SA3 and the JRAs to plan the work for that cycle. The work to be done in this phase is estimated to be short since it mainly involved the prioritisation of requirements for that cycle. The duration of this phase is thus estimated to be approximately one month.

The "Do" phase involves only the JRAs. It is the phase where JRA1 will work on harnessing the selected application codes while JRA2 will adapt and/or develop the selected high-level integration services and tools. Since actual development work is performed in this phase, more time is proposed. This phase is estimated to last five months.

In the "Check" phase, SA2 will evaluate, integrate and test the selected software components while SA1 will be responsible for operating and monitoring the platform. The software components that have passed the tests are accepted and handed over to SA1 for deployment. Software components that failed the tests will be handed back to the respective JRA. If the JRA is able to perform a quick fix within this phase, the component will be re-evaluated for acceptance and thus deployment. In parallel, SA1 and SA3 will monitor the already deployed software components from the previous PDCA cycle and collect service information (refer to Section 3.5) into the platform and gateway respectively. This information will be shared with SA2 to identify issues and improvements. Taking into account the amount of work planned and the time required to collect service information, more time is allocated. The duration of this phase is approximated to require four months.

In the final "Act" phase, SA1 and SA3 will act on the approved list of software components from SA2 and deployed them. SA1 and SA2 will analyse the collected service information from the previous phase and request SA3, JRA1 and JRA2 to suggest feature improvement, bug fixes, etc., as per required. This final phase is estimated to require approximately two months.

Given the aforementioned time estimates, a PDCA cycle will take a year to complete. However, a new release or update of the platform should take place every six months. To achieve that, two PDCA cycles will run simultaneously as illustrated in the next section.

2.1.1. Overlapping PDCA Cycles

In order to have software components released or updated every six months, two overlapping PDCA cycles will run simultaneously, as shown in Figure 4. In total, seven PDCA cycles, corresponding to seven releases should take place during the project lifetime, i.e. four years.

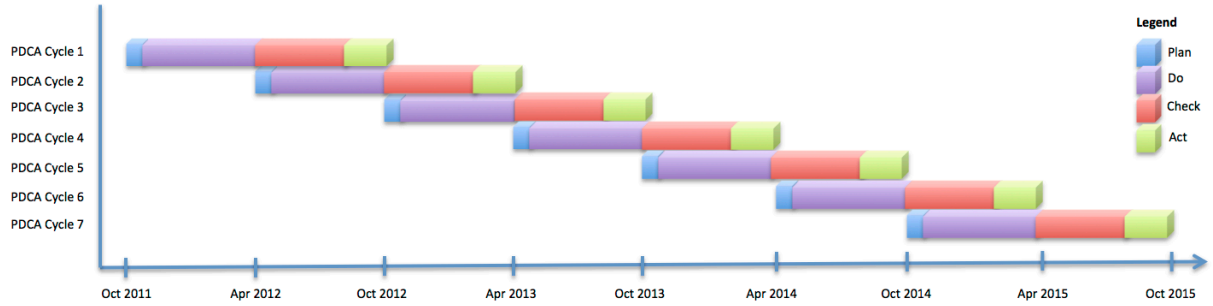

Figure 4. PDCA cycles distribution during project lifetime

It is possible to have overlapping PDCA cycles as the JRAs are mainly involved in the "Plan" and "Do" phases, while the SAs are principally involved in the "Check" and "Act" phases, as illustrated in Figure 5. This arrangement leverages on the fact that the SAs and the JRAs can work independently. In addition, it ensures that there is no gap in the release process where the SAs or the JRAs have to wait redundantly for one another.

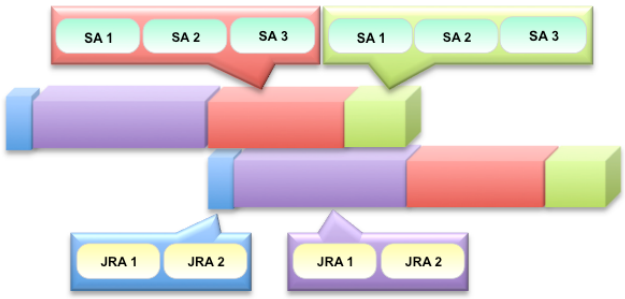

Figure 5. Overlapped PDCA phases

This clearly defined release process however entails a certain risk. The lapse of one work package in the PDCA cycle could have serious repercussions on the other work packages, ultimately resulting in a missed platform release. In order to mitigate the risk, a certain level of agility will be necessary from each work package while following the process. The recommended work practices to gain agility (Leffingwell, 2007) that are of relevance to us are as such:

- The release schedule is communicated and known to all affected development teams and work package leaders well in advance.
- Frequent prototype releases of in-house software components by the development teams that are independent of the VERCE release process should be made.
- Constraining teams to the release schedule means that implemented functionalities for the components in each release must be flexible. If some planned functionalities are not ready by the schedule date, the component should still be released.
- Infrastructure components must be track ahead or made available in advance so that the in-house developed software components can be readily integrated when ready.
- Each component team must be committed to meeting the release schedule date by having a primary plan and a fallback plan. The fallback plan can be as simple as to ship with an older version of the software component as opposed to the latest one.

With the work practices and the PDCA cycle in mind, the release schedule for the next release will be planned. After the first release, a review of management release process, i.e. the PDCA cycle, duration of each phase and the recommended work practices, with the affected work packages will be planned.

3. SA2 Procedures in the PDCA Cycle

SA2 procedures to carry out the actual release process adhering to the PDCA cycle are elaborated in the following sections. The "Do" phase will not be elaborated since SA2 has no activities in this phase.

3.1. Plan Phase

In the "Plan" phase, SA3 and the JRAs will submit their selected software components to SA2 for evaluation, integration and testing. A "Software Component Release Request Form" is thus prepared.

3.1.1. Software Component Release Request

To facilitate communication between SA2 and the requestors, SA3 and the JRAs a "Software Component Release Request Form" has been created. This form is based on DEISA's and PRACE's DECI proposal template[2]. Two test usages of the form to evaluate the release management process took place. The test requests for evaluation of the tool/service, GridFTP, and the application code, SeisSol, can be found in Appendix A. The form will be an integral part of SA2's Release Management Process. It facilitates the first step for SA3 and the JRAs to get in contact with SA2, to formally request an analysis and test of a selected software component for release into the VERCE platform.

3.2. Check Phase

In the "Check" phase, SA2 will evaluate, integrate and test the selected software components. Suitable resources and tests thus have to be defined, chosen and planned.

3.2.1. Evaluation and Integration Form

With the combined effort of SA1 and SA2, the information of the resources provided by the partners is gathered. Eleven resources are currently identified, as shown in Figure 6. These resources will be steadily integrated by SA1 to be a part of the VERCE test bed for development, analysis, integration and testing.

The test-bed resources can be categorised as departmental/institutional resources, EGI resources and HPC/PRACE resources. The departmental/institutional resources are resources owned by the Earth Science community and seismologists at their institutions. The HPC/PRACE and EGI resources computing resources on the existing European e-Infrastructure, where the seismologists and Earth Science community hope to utilise now and in the future. The departmental resources can represent both the local computing resources available to the seismologists and Earth Science community, and the client resources that the scientists would access the European e-Infrastructure from. In the scope of SA2, the departmental resources will especially be useful to analyse and test components, which have to be available on the scientists' local resources. The HPC and EGI resources will be used to evaluate the suitability of the software components on the selected EGI/PRACE resources and thus the European e-Infrastructure. With such a setup, we hope to ensure that the selected software components could be integrated on most if not all existing European e-Infrastructure and departmental resources of scientists.

[2] http://www.prace-project.eu/IMG/doc/deci9-proposal-acronym-final.doc

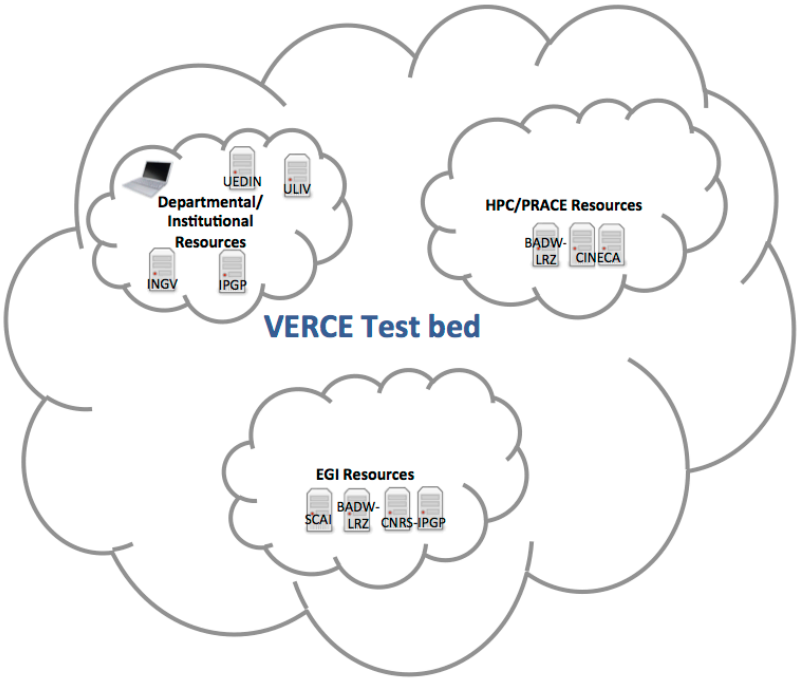

Figure 6. VERCE Test bed

A sample of the GridFTP Evaluation and Integration Form, listing the resources that are possibly available for the evaluation and integration tests is shown in Appendix B. Two resources, SuperMIG and UEDIMI, were selected to test the component. A SA2 member was put in charge for each of the selected resources to evaluate, integrate and test GridFTP. The purpose of this form is to keep track of the resources chosen to evaluate a particular software component. The selection of which resources to use for the evaluation and test will be based on the recommendation as per the Software Component Release Request Form in the field "Preferred/Priority test architecture". For each software component, at least one departmental resource and one HPC/EGI resource from the testbed will be recommended to closely represent the actual use case.

3.2.2. Testing

Proposed software components, in particular in-house developments, have to be properly documented and packaged before they can be accepted as a part of the VERCE platform. A set of manual generic tests to ensure that the released software components are of basic recommendable quality is thus defined. Additionally, the components have to be tested to verify that all expected features are functional and that new or updated components can be properly integrated with the platform. A second set of tests, the software component specific tests, is thus defined.

The specific tests should be automated as a part of the continuous integration process since the software components developed within VERCE should be frequently tested to ensure their effectiveness and compliance to the infrastructural policies and constraints of the VERCE test bed. The variety of the components expected to be a part of the VERCE platform also suggests that the integration should proceed regularly to ensure that the interoperability of the components in the platform is not compromised. This is particularly important after a change in any of the components. Extensive automated testing is thus recommended to minimise the integration effort and allow development effort to remain as undisrupted as possible while ensuring that a release is shipped within the scheduled deadline.

In conclusion, a continuous automated integration system is beneficial to the development and release process in multiple ways and thus would be adopted. It fosters the identification of bugs through self-test builds, diff-debug and consequently prevent cumulative bugs. It helps in reducing the risk of having blind

spots in the projects where the bugs of one software component result in the failure of another seemingly unrelated software component. The availability of the automated test framework encourages frequent prototype releases, which could act as a fallback release in event that the planned release failed (refer to Section 3.1.1). VERCE in-house developers can thus focus on coding while allowing the automated integration and testing system to check their releases on the test bed.

The ten rules of Fowler for Continuous Integration[3] will be evaluated since it serves as a good guideline to reduce integration issues:

- Maintain a Single Source Repository.
- Automate the Build
- Make Your Build Self-Testing
- Everyone Commits To the Mainline Every Day
- Every Commit Should Build the Mainline on an Integration Machine
- Keep the Build Fast
- Test in a Clone of the Production Environment
- Make it Easy for Anyone to Get the Latest Executable
- Everyone can see what's happening
- Automate Deployment

More rationale and properties of these tests are discussed in the following sections.

3.2.2.1. Generic Tests

A set of generic tests is identified to analyse the tools, services and applications codes. By studying the EGI's UMD certification tests, the generic tests are defined. A sample form containing the tests is shown in Appendix C. These generic tests should ensure that the basic documents, licenses and release notes are included in each release. The generic tests are especially applicable to VERCE in-house developed software components. The tests include main tests that referred to the integration and functionality tests, also known as the software component specific tests, as elaborated in the following section. Service reliability and robustness, security and performance tests are also defined in the generic tests.

3.2.2.2. Software Component Specific Tests

In order to provide an automated environment for continuous integration, available software solutions have been explored. One of them is Jenkins-CI[4], a continuous integration tool written in Java[5], which runs on a servlet container[6] such as Apache Tomcat or the GlassFish[7] application server. It supports Source Control Management[8] tools such as CVS[9], Subversion[10], Git[11] and Clearcase[12] and can execute Apache Ant[13] and Apache Maven[14] based projects, as well as arbitrary shell scripts and Windows batch commands.

---

3 http://martinfowler.com/articles/continuousIntegration.html
4 http://jenkins-ci.org/
5 http://en.wikipedia.org/wiki/Java_%28programming_language%29
6 http://en.wikipedia.org/wiki/Java_Servlet#Servlet_containers
7 http://en.wikipedia.org/wiki/Glassfish_Application_Server
8 http://en.wikipedia.org/wiki/Source_Control_Management
9 http://en.wikipedia.org/wiki/Concurrent_Versions_System

The Acceptance Testing Driven Development (ATDD)[15] approach will be adopted to determine whether a software component is accepted or rejected. This will enable an in-depth analysis of the requirements defined in the scientific use cases by facilitating the definition of tests based on the actual inter-workings of the software components on the test bed. This helps to promote the development of working, high quality software components that are reliable, easy to maintain and integrate.

There are freely available open source tools to support the automated ATDD approach. One worth mentioning is the Robot framework[16], which is used by IGE. This framework is a generic test automation framework for acceptance testing and acceptance test-driven development. It has an easy-to-use tabular test data syntax and utilises the keyword-driven testing[17] approach. It also provides a command line interface and XML based outputs for integration into continuous integration systems. This framework can be easily configured to run as one of the Jenkins-CI tasks. In-house development software components will be recommended to have tests utilising such a framework.

An example for specific GridFTP tests defined by SA2 is shown in Appendix D. The tests will be shared with IGE and proposed to be included in their GridFTP test-suite. In general, integration tests between software components will be coordinated and planned within SA2 to check the interoperability among the components in the platform.

3.3. Act Phase

In the "Act" phase, SA1 and SA2 will check the service information collected by SA1 and provide feedback to the JRAs on issues found and possible improvements. To do so, the definition of performance indicators that can be used to define the monitoring metrics is required. The monitoring metrics will in turn be used to assess the quality of service.

3.3.1. Definition of Performance Indicators and Assessment of Quality of Service

In order to define the performance indicators and to assess the quality of service in VERCE production platform, a strong collaboration between SA1 and SA2 is required. SA1 has to commit to monitoring the VERCE platform, i.e. the software components. As such, an initial list of possible key performance indicators that can be used to measure the performance and assess the quality of service is identified. These indicators are shared with SA1 to assist them in defining monitoring metrics for the platform.

Availability of services:

- Service availability
- Number of service interruptions
- Duration of service interruptions

---

[10] http://en.wikipedia.org/wiki/Subversion_%28software%29
[11] http://en.wikipedia.org/wiki/Git_%28software%29
[12] http://en.wikipedia.org/wiki/Clearcase
[13] http://en.wikipedia.org/wiki/Apache_Ant
[14] http://en.wikipedia.org/wiki/Apache_Maven
[15] http://www.methodsandtools.com/archive/archive.php?id=72
[16] http://code.google.com/p/robotframework/
[17] http://robotframework.googlecode.com/hg/doc/userguide/RobotFrameworkUserGuide.html?r=2.7#different-test-case-styles

- Availability monitoring
- Number of service failures (in terms of tickets submitted by users)

Tool/Service/Software Releases:

- Percentage of failed release component acceptance tests
- Percentage of failed service validation tests
- Number of incidents

Security:

- Number of major security incidents
- Number of major changes
- Number of emergency changes

Quality of Support:

- Number of incidents (submitted tickets)
- Average initial response time
- Incident resolution time
- Number of service reviews

The service information collected for "Availability of Services", "Tool/Service/Software Releases" and "Security" will be of particular interest for SA2. The service information, "Quality of Support", will be of more relevance for SA1 and SA3. With the collected service information, SA2 will be able to perform the final "ACT" phase in the PDCA cycle to feedback to the JRAs on how they could improve their tools, services and application codes.

In addition, it has been noted that EMI has defined a policy, tools and a rather sophisticated dashboard for checking their performance, assess their quality of service and displaying and reporting their service information. VERCE can probably benefit from their experience and available tools.

4. Assembly of Software Components

Tools, services and simulation codes, i.e. software components, are the fundamental building blocks of the VERCE platform. The selection of the software components is a conscientious process that leverages on the expertise of the related work packages. The tools and application codes required to realise the scientific use cases are expected to be selected by JRA1. The tools and services that are necessary to support the selected workflow management tool to execute the selected scientific use cases are expected to originate from JRA2. The tools and services required to operate the scientific gateway is expected to come from S3. Tools and services that are required to operate the platform will be requested by SA1 and SA2. The stability and suitability of some of the proposed software components will be tested and analysed, and a suitable set of software components will be assembled and recommended to SA1 for deployment.

4.1. Initial Proposed List of Software Components

In this first reporting period, a list of the currently available software components provided by VERCE partners has been collected (refer to D-SA1.1). Similarly, a list of software components that are of relevance to enable JRA2 to adapt and/or develop high-level integration services and tools are gathered and studied (refer to D-JRA2.1). JRA1 also provides a list of application codes (refer to D-JRA1.1) that can be adopted to fulfil the scientific use cases. By integrating these lists of software components, SA2 is able to derive an initial list of potentially useful software components for the platform. The list of potentially useful software components, categorised by their functionalities, is shown in Table 1. This list will serve as the basis for the selection of the initial (fast track) portfolio of software components, in particular to support and allow the basic functionalities to access and work on the test bed (refer to Section 3.2).

Tools/Services	Comments
Authentication	
X.509 certificate based	Most computation resources contributed by partners are either existing PRACE or EGI resources, which require X.509 based authentication. X.509 is thus a reasonable choice that enables VERCE to easily integrate with the existing e-Infrastructure.
Username/Password based	Default access method for most resources. This is however not as useful and manageable as X.509 based method when users require to access multiple resources and services.
Access to resources	
Globus GSI-SSH	Interactive X.509 access to resources. This is required to compile, install, evaluate and test tools, services and application codes on the test-bed and production resources.
UNICORE	UNICORE 6 rich client provides an SSH interface to support interactive access to the resources. This is required to compile, install, evaluate and test tools, services and application codes on the test bed and production resources.
Data transfer and management	
Globus/gLite GridFTP	Optimised for high-bandwidth, secure, reliable and high performance data transfer protocol based on FTP. It includes a server and a command-line UI. It is supported on both EGI and PRACE infrastructure. ADMIRE will be using it as one of the protocols for accessing data from seismic data resources. It is a part of EGI's Universal Middleware Distribution (UMD).
ArcLink	ArcLink is a protocol similar to SeedLink for real-time data transfer. It is based on TCP and uses simple commands in ASCII coding. Data are requested based on time windows. It is possible to use ArcLink for accessing different data archives.
OGSA-DAI	Supports distributed data access and management. It is being used by ADMIRE to access data from distributed database, e.g. relational SQL DB and XML DB. It will be a part of EGI's UMD.
ObsPy	An open-source project to provide a Python framework for processing seismological data. ObsPy is used by ADMIRE to access some seismic database. It can be used in both

	seismic use cases.
iRODS	iRODS is a community-driven, open source and data grid software solution to manage (organise, share, protect and preserve) large sets of files (even in the scale of petabytes). It supports high-performance network data transfer, a unified view of disparate data, backup and replication, manages metadata, controlled access, data workflows and management of large collections.
Workflow management	
ADMIRE/DISPEL	A service for translating logical workflow specifications into a choreographed set of executable task pipelines delegated to distributed resources. ADMIRE provides a favoured dataflow language, Dispel. Provides visual programming tools to define the workflow. Supported by VERCE as an in-house product.
UNICORE	Provides a graphical and command line interface for defining workflows. Provides seamless, secure and intuitive access to distributed Grid resources such as HPC, cluster systems and information stored in databases. Supported by both EGI and PRACE. It is a part of EGI's UMD.
GridSpace2	A top-level user interface designed to suit the requirements of domain scientists and allow them to exploit distributed computing platforms. A (MAPPER, 2011) Web based Experiment workbench that supports workflow definition.
Kepler	Scientific workflow application used by GEOsciences Network (GEON) project in the US and by a lot of communities, e.g. Astrology.
SHIWA	A platform to support workflow interoperability when different workflow engines/languages are used. This could be a useful tool for ADMIRE to integrate with, in order to provide interoperability with other workflow languages.
Job management	
gLite WMS	A workflow management system that comprises a set of grid middleware components responsible for the distribution and management of tasks across grid resources such that the applications are conveniently, efficiently and effectively executed. It provides a command line job submission and management interface. It uses local batch scheduler at the backend. It is supported on many EGI resources. A part of EGI's UMD.
Globus GRAM	Enable users to locate, submit, monitor and cancel remote jobs on Grid-based compute resources. Enables execution management in contexts where reliable operations, stateful monitoring, credential management and file staging are important. Command line job submission and management interface. Uses local batch scheduler at the backend. Supported on many PRACE resources. A part of EGI's UMD.
UNICORE 6	A ready to run Grid system including client and server software. Provides a GUI interface for job submission and management. Uses local batch scheduler at the backend. Supported on PRACE resources. A part of EGI's UMD.
SAGA	A standardised API for developing distributed applications that can run on grid and cloud infrastructures. Emphasis is given on job handling and monitoring, file transfer and management, and distributed orchestration mechanisms. Most importantly, it provides backend support for many middleware distributions, including Globus Toolkit, UNICORE, gLite and ARC.
Application codes	
AXISEM	A parallel spectral-element method to solve 3D global acousto-elastic wave propagation
SES3D	Regular grid spectral element in spherical coordinates – possibility to calculate sensitivity kernels.
SeisSol	Tetrahedral grid based on discontinuous Galerkin method.
SPECFEM3D	A parallel code based on the spectral-element method. Used to compute sensitivity kernels for gradient-based inversions.
RegSEM	A versatile code based on the spectral-element method to compute seismic wave propagation at a regional scale.

Visualisation	
SDX	Analysis and visualisation of seismic waveform. Supported by VERCE partner, ULIV.
Client-sided	
RAPID	A web portal technology that allows fast generation of portlets that adheres to the industry standard for portal interfaces as well as to HPC and Grid standards for job execution. It will be evaluated as a potential VERCE scientific gateway.
GSI-SSHTerm	A platform independent Java-based X.509 terminal client that will allow a user to generate both VOMS and normal proxy to interactively access a resource. It is the recommended tool for interactive access on PRACE Tier-1 resources. Various EGI resource providers in the UK and Germany that support interactive access also recommend it. In the US, NCSA and XSEDE also provide support for this tool.
Globus Online	Globus Online is a browser-based interface to use GridFTP services (reliable file transfer) via the cloud. It provides an easy to install client side GridFTP server, Globus Connect. Globus Online is based on a set of REST services and thus can be used by any client including ADMIRE. In addition, a command-line interface that provides at least as many features, as the Web-based one, is also available.

Table 1. Initial list of potentially useful software components

4.2. Initial Portfolio of Software Components (Fast Track)

One of the milestones of SA2 in this reporting period is to select an initial portfolio of the software components to provide the basic functionalities, secure user access, resource management, job submission and data management, required to operate the VERCE test bed. The selection of the fast track components will not follow the defined PDCA cycle that is introduced in Section 2 but will instead focus on gathering currently available software components on partner resources to quickly provide the required basic functionalities. In addition, priority is given to components that are required to support the work of the JRAs. The motivation behind the initial portfolio is to support as quickly as possible the basic functionalities required to enable VERCE members to commence their development and integration activities on the test bed. With those aims in mind, currently available software components that are in production on partner resources, in particular EGI and PRACE resources, are leveraged and the components required by the JRAs are prioritised. The initial portfolio of software components selected from Table 1 is elaborated as follows:

Secure User Access

X.509 certificate-based authentication is the official access method to existing EGI and PRACE Tier-1 infrastructures. X.509 is a public key infrastructure (PKI) standard developed by the ITU Telecommunication Standardisation Sector (ITU-T). This certificate-based authentication enables EGI and PRACE to protect the users' privacy and support the secure sharing of data across multiple distributed resources.

Both EGI and PRACE endorse the certificates issued by the Policy Management Authorities (PMAs) that complies with policies and guidelines established by the International Grid Federation Trust (IGTF). VERCE will adopt similar strategies to guarantee European-wide access to the computing resources.

To support certificate-based interactive access, GSISSH, a service provided by Globus Toolkit in PRACE is chosen. The Globus Toolkit is also a part of EGI's UMD, contributed by IGE. On the client side, there is a platform-independent Java Web Start terminal client, GSI-SSHTerm, recommended by PRACE and some EGI resource providers, which supports the generation of both normal and VOMS proxies. VOMS proxies are required in order to access the EGI resources.

Resource Management

The management of resource across resource providers typically varies. In order to efficiently manage the resources, a unified resource management tool that is synchronised with the individual VERCE resource providers' local resource management systems is required.

PRACE uses numerous distributed LDAP directories, operated by the resource providers, in order to share and manage the information and thus effectively allocate and manage the resources. Each partner/site administrator has the responsibility to update and synchronise, on daily basis, the gathered information in the LDAP directories with their local resource management system.

EGI adopts a similar procedure. Due to the complexity of resources available in EGI, two types of data sources are used to store resource information, Berkeley Database Information Index (BDII) and Grid Configuration Database (GOCDB). BDII, an LDAP-based information index, is used to store resource information while GOCDB is used to share general information about the sites participating in the production Grid (Aeschlimann, 2010). On top of the data sources, EGI provides a central operations web portal to share and manage the resource information in BDII and GOCDB. As opposed to PRACE method of updating the resource information by site administrators, the virtual community representatives update their respective user/virtual organisation information. In the backend, each resource provider/site administrator is responsible for synchronising the information shared in BDII and/or GOCDB with their local resource management systems.

VERCE plans to adopt a similar VO management strategy as EGI. A central portal will be provided by SA3 to support the sharing of required information to manage the VOs. The resource providers in VERCE will be responsible in synchronising this information with their local systems.

For the fast track portfolio, the EGI VO and resource management framework is used. Since many of the resources provided by the partners are EGI resources, VERCE members are able to access on the EGI resources via this method. On HPC and departmental resources, accounts will be created using the respective local user administrator systems of the partners.

Job Submission

PRACE and EGI provide various middlewares, Globus GRAM and UNICORE 6, to support job submission. Globus GRAM provides a command-line interface to enable users to locate, submit, monitor and cancel jobs on Grid-based computing resources. GRAM enables execution management in contexts, which reliable operation, stateful monitoring, credential management and file staging are important. It provides a single protocol for communicating with different batch/cluster job schedulers. (The Globus Alliance Wiki, 2005). In contrast, UNICORE 6 provides a graphical, Unicore Rich Client (URC), and a command line, Unicore Commandline Client (UCC), interfaces for job submission and management. It supports the use of standardised Job Submission and Description Language (JSDL) and OGSA Basic Execution Services (OGSA-BES) to improve interoperability with other technologies to enable cross-Grid job submission and improve resource management capabilities. It also provides workflow support, including a graphical workflow editor, workflow monitoring features, a workflow engine and a service orchestrator layer.

EGI resource providers mainly provide gLite Workload Management System (WMS) for job submission on the server end. There are some resource providers that also support Globus GRAM and UNICORE 6. GLite WMS comprises a set of grid middleware components responsible for the distribution and management of tasks across grid resources in such a way that applications are conveniently and efficiently executed. It provides a command-line interface for end-users and a matchmaking process between submission requests and available resources on the grid to assign jobs to appropriate computing elements for execution. (GLite, 2005) It also supports the use of CREAM clients in addition to the command line interface.

As such, in order to serve users coming from infrastructures like EGI and PRACE, gLite, Globus and UNICORE are selected as the job submission tools in the fast track portfolio. All three middlewares are a part of EGI's Unified Middleware Distribution (UMD). Each VERCE resource provider is recommended to provide at least one of these three job submission middlewares. In the case where a resource provider is unable to support any of the middleware, support for a limited list of common local batch schedulers will be considered in future releases.

Data Management

To support data management, a critical service in VERCE, ADMIRE is the selected (refer to D-JRA2.1). Several dependent software components, ADMIRE gateway, OGSA-DAI, Apache Tomcat[18], GridFTP, Grid Security Infrastructure (GSI)[19] and Java Runtime, are required to enable the ADMIRE architecture. Three of the software components, ADMIRE gateway, OGSA-DAI and GridFTP are of particular importance to support data management development work of JRA2 and are thus selected as a part of the initial portfolio.

The ADMIRE gateway is a service for translating logical workflow specifications into a choreographed set of executable task pipelines, delegated to distributed resources, before they are presented to some enactment process. The principal advantages of ADMIRE lie in its inherent extensibility and support for a given model of user / system interaction, which is currently favoured by VERCE. It allows a variety of heterogeneous tools and services to submit specifications of tasks as workflows via a common gateway and then to be implemented and deployed automatically across a range of heterogeneous processes and resources. An ADMIRE gateway can serve as the "hub" to which various existing services, tools and resources are connected. It will act as a lightweight system that can be adapted easily to the requirements of the project, which then allows, through various simple, custom interfaces, the easy assimilation of the aforementioned existing services and code libraries. ADMIRE provides a favoured dataflow language, Dispel.

OGSA-DAI is an open-source framework that allows for distributed data access and management. It is the favoured enactment platform for ADMIRE. It allows ADMIRE to plug in activities with specific functionality to access to various data resources, such as relational databases, web resources, e.g. ORFEUS and remote filesystems. It also provides data processing, such as mining and filtering, functionalities.

GridFTP is the reliable high performance data transfer protocol that will be supported by ADMIRE to allow huge data transfer. It is a proven FTP based protocol that is supported by both Globus and gLite in PRACE and EGI respectively. GridFTP is a part IGE's contribution to EGI's UMD.

[18] http://tomcat.apache.org/

[19] http://en.wikipedia.org/wiki/Grid_Security_Infrastructure

5. Organisation of Work

To manage SA2 efficiently, task leaders are identified to lead the critical tasks and regular meetings are organised.

5.1. SA2 Task Organisation

Five partners, BADW-LRZ (leader), CNRS-IPGP, KNMI, SCAI and UEDIN, are involved in the work package WP6/SA2. Four critical tasks have been identified in the DoW.

- Task 1: Coordination and management of integration efforts *[Siew Hoon Leong, BADW-LRZ]*
- Task 2: Assembly of tools, services and simulation codes *[Amy Krause, UEDIN]*
- Task 3: Testing *[Alessandro Spinuso, KNMI]*
- Task 4: Definition of performance indicators and assessment of the quality of service *[Geneviève Moguilny, CNRS-IPGP]*

The leadership of each task has been organised to leverage on the expertise of the representatives from each partner. Since BADW-LRZ is in charge of SA2, the work packager leader also leads the effort in Task 1 to coordinate and manage the integration efforts. Taking into consideration that UEDIN leads JRA2, the key work package that will contribute to the list of tools and services that will be assembled, the representative from UEDIN is chosen as Task 2 leader to leverage on her expertise. The representative from KNMI is picked to be Task 3 leader so as to exploit his strong IT experience to provide an automatic testing framework. Finally, since SA1 is led by CNRS-IPGP, the provider of the production performance indicators indices, the representative from CNRS-IPGP is selected to be Task 4 leader to provide the bridge in coordinating the collection of useful production monitoring data for feedback and improvements. Representatives from SCAI offer the final lever to provide the much needed technical competency to ensure that the requirements of the existing e-Infrastructure, in particular EGI, are always taken into consideration in our analysis of tools, services and application codes.

5.2. Meetings

To efficiently coordinate and manage the work in SA2, regular bi-weekly Skype calls have taken place since November 2011. Since SA1 and SA2 have to collaborate closely to support and operate the VERCE platform, a Platform Service coordinator, SA2 leader, was officially assigned in March. The role of the coordinator is to ensure that the work performed in SA1 and SA2 are aligned. Regular monthly Skype calls will be proposed.

6. Issues and Future work

In the first six months of VERCE, we identified in more details the activities and the plans of each release cycle. While doing so, a number of issues have been identified. One of the issues is that there are insufficient expertise and resources in SA2 to develop the many specific tests for the core tools/services that must be offered in the VERCE platform. A good example in this case is GridFTP. While attempting to come up with reasonably thorough test-suite for GridFTP, detailed knowledge of the functionalities, features and bug fixes of each GridFTP releases is important to update the test-suites to ensure that required functionalities and interoperability of the software components are not adversely affected. To this end, a negotiation with Initiative for Globus in Europe (IGE) to define a MoU to share their automatic functionality and regression test-suites has commenced. Support for a number of IGE components, e.g. the two core data management tools, GridFTP and OGSA-DAI from the fast track portfolio, are also requested. More collaboration with other FP7 projects, e.g. EMI, will be considered.

Another issue that needs resolving is related to the multitude of licenses associated with the application codes offered by JRA2. A generic way of handling application codes with restrictive licenses has to be considered when offering application codes as a service. The experience of the HPC centres in handling software licenses, e.g. in PRACE, will be leveraged.

Finally, another issue identified is the non-uniform version of gLite across the EGI infrastructure. The requirements of the gLite middleware are mainly driven by the High Energy Physics community. As such, there are no generic rules/requirements for EGI resources to upgrade their gLite version to the latest version offered by EMI. Since older gLite versions are packaged with an older version of GridFTP, many new and useful features, such as the resumption of file transfer, is unavailable. JRA2 has since been informed and will take that into consideration while developing the GridFTP services in ADMIRE. Partners in VERCE with EGI resources will be encouraged to update their gLite/GridFTP to the latest released version.

In addition to tackling the above-mentioned issues and managing the first official VERCE release, the future work of SA2 includes the refinement and improvement of the agility to manage the release process. A meeting with the affected work packages to review the release management process will be planned after a VERCE platform release. The generic tests are to be further evaluated in the next release to explore room for improvement. The automatic test framework, Jenkins-CI, will be deployed to facilitate the continuous integration of software components. The definition of the performance indicators will be reviewed to assess its usability and suitability after the first “Act” phase in the next reporting period.

Appendix A: Software Component Release Request Form

GridFTP Request Form:

VERCE Tools, Services and Application Codes Release Request Form	
(To be submitted to sa1@verce.eu and sa2@verce.eu upon completion)	
Request Origin:	JRA2
Contact Person:	Paul Martin
Email address:	pmartin at staffmail.ed.ac.uk
Request Submission Date:	5th March 2012
Code/Tool name:	GridFTP
Code/Tool version:	Globus Toolkit 5.2.0
Purpose of Code/Tool:	Efficient transfer of large data files between remote locations.
Description of operation	GridFTP clients request data files from GridFTP servers upon provision of suitable security credentials (preferably an X509 certificate signed by a trusted Certificate Authority). Clients can also upload data and order the transfer of data between two remote GridFTP servers.
Role on the VERCE platform	Use-case workflows require the staging and unstaging of large datasets on computational resources. GridFTP is being proposed as the primary mechanism of file transfer between geographically remote resources, to be invoked by gateways either on request (initially) or based on information provided by the data curation / replication service (later).
Technical comments	GridFTP uses GSI (Grid Security Infrastructure), also part of Globus Toolkit 5.2.0, for security. GridFTP permits third-party data transfer, partial data transfer and parallel data transfer over multiple data channels.
Architecture already installed/used:	Has been installed on EDIM1 for testing purposes; EDIM1 is now being opened for external GridFTP requests. Further internal tests under an updated configuration are also to be performed soon.
Known/Possible architecture problems:	Proper use of GridFTP requires configuration of clients and servers, which requires a properly configured security infrastructure to be in place; integration of GSI into ADMIRE (specifically under the OGSA-DAI enactment platform) has not yet been completed.
Preferred/Priority test architecture:	EDIM1.
Licence info:	Apache Public License 2.0
	Globus Toolkit Public License 3.0
	OpenSSL license (for OpenSSL): http://www.openssl.org/source/license.html
Download url:	Globus Toolkit 5.2.0: http://www.globus.org/toolkit/ (installation of GridFTP alone possible)
Test suite availability:	Globus has some test suites but it has broken in 5.2
Tool's Specific Information	
Programming language:	C
Dependent software: (e.g. JAVA, PERL,..)	Within the Globus Toolkit: Non-WS (General) Authentication & Authorization, C Common Libraries, XIO.
	3rd party: OpenSSL.
Memory requirement:	About 34 MB (2 MB base + 16 MB TCP buffer + 16 MB user space buffer) per server instance -- see http://www.globus.org/toolkit/docs/5.2/5.2.0/gridftp/admin/#id2468929.
Additional open ports: (e.g. 2811 for GridFTP)	2811 (default) for GridFTP control channel; additional ports for data channels required.
Any additional information/requirements:	
Application Code Specific Information (requirements of a typical run)	
Programming language:	

Parallel Communication: (Pure MPI, OpenMP or Hybrid)	
Maximum no. of cores:	
Minimum no. of cores:	
Minimum memory per code (GB) at maximum number of cores:	
Total memory for smallest target problem:	
Total memory for largest target problem:	
Temporary disk space requirement during a single run:	
Storage required:	
Library requirements:	
Any additional information/requirements:	

Request arrived and received
Person in charge: Siew Hoon Leong (Cerlane)
Date: 05/03/2012

Seissol Request Form:

VERCE Codes and Tools Release Request Form	
(To be submitted to sa1@verce.eu and sa2@verce.eu upon completion)	
Request Origin:	JRA1
Contact Person:	Marek Simon
Email address:	marek.simon@geophysik.uni-muenchen.de
Request Submission Date:	
Code/Tool name:	SeisSol
Code/Tool version:	
Purpose of Code/Tool:	Calculation of wavefield propagation with ADER-DG scheme
Description of operation	calculate synthetic seismogramms for earthquake events on continental, regional and fault scale; basic research on earthquake faulting; applications for gas and oil industry;
Role on the VERCE platform	HPC pilot application
Technical comments	
Architecture already installed/used:	SeisSol runs on PC, MAC, UNIX systems and various HPC cluster: e.g. Shaheen (BlueGene/P, IBM), SuperMIG, HLRB2, BSC, Huygens
Known/Possible architecture problems:	not known
Preferred/Priority test architecture:	with IBM compilers
Licence info:	Not under open source available due to restrictions of industry partners. Code can be accessed under a simple and cost free EULA (?).
Download url:	http://svn.geophysik.uni-muenchen.de/svn/seissol
Test suites availability:	Over night-build runs on local institute cluster. This test could potentially be ported.
Tool's Specific Information	
Programming language:	
Dependent software: (e.g. JAVA, PERL,..)	
Memory requirement:	
Additional open ports: (e.g. 2811 for GridFTP)	
Any additional information/requirements:	
Code's Specific Information (requirements of a typical run)	
Programming language:	Fortran 90, some F77 routines, MPI
Parallel Communication: (Pure MPI, OpenMP or Hybrid)	MPI
Maximum no. of cores:	48000 (in principle no restrictions)
Minimum no. of cores:	1
Minimum memory per code (GB) at maximum number of cores:	estimated to ~500MB (not tested)

Total memory for smallest target problem:	Depends highly on the model and we use the flexibility of SeisSol to simulate very different models.
Total memory for largest target problem:	Usually, rather low (1-10% of memory per node).
Temporary disk space requirement during a single run:	No temporary disk space required. Just memory for results necessary.
Storage required:	Depends on desired output and ranges between 10-1000MB. In the case, that high-resolution wavefield snapshots are requested the amount of data can be 30-50GB per single snapshot (very rarely used case).
Library requirements:	No strict dependencies. BLAS (MKL) and ESSL optional. (pre- and postprocessing scripts that are included in the software package: Matlab, Maple, Python, Paraview, Tecplot, DGVisu. METIS, SConstruct)
Any additional information/requirements:	Requires pre- and postprocessing: Model generation (individual), meshing (ANSYS ICEM (not supporting all feature of SeisSol), currently on search for a better suited since the standard mesher Gambit is not available anymore), result processing with matlab, python, tecplot, paraview, DGVisu (self written fortran program). Partitioning is done with METIS (http://glaros.dtc.umn.edu/gkhome/views/metis) controlled by matlab scripts. For compiling SConstruct is used (http://www.scons.org).

Request arrived and received
Person in charge: Siew Hoon Leong (Cerlane)
Date: 12/03/2012

Appendix B: Evaluation & Integration Allocation Form

Summary of integration, testing and assessment on VERCE resources

(Refer to http://www.verce-project.eu/projects/verce1/repository/entry/verce/All/SA/SA1/T1b1/comp_res.html for more info abut the resources)

	Resource (Computation)	Site	Due date	Person in charge (Testing)	Status	Comments
1	SuperMIG/SuperMUC	LRZ	3/20/2012	Siew Hoon Leong	Testing	PRACE
2	Linux Cluster (>2800cores)	LRZ				EGI
3	Blue Gene/Q	CINECA				PRACE
4	PLX GPU Linux Cluster	CINECA				PRACE
5	EGI Cluster	SCAI				EGI
6	Parallel Cluster (IBM x3550)	IPGP				Departmental
7	EGI Cluster	IPGP				EGI

	Resource (Data)	Site		Person in charge (Testing)	Status	Comments
1	COHERSIS (110TB)	IPGP				Departmental
2	Linux Cluster (2.8TB)	ULIV				Departmental
3	EDIMI (Data intensive cluster)	UEDIN	3/20/2012	Paul Martin	Testing	Departmental
4	ELIOS cluster	INGV				Departmental

Appendix C: GridFTP Generic Tests

	Integration Process Checklist (EGI)	Description	Applicability	Pass/Fail	Comments
1	Download/Receive code/software		All	Pass	Recommendation for GridFTP: If available, install using IGE pre-compiled packages. http://www.ige-project.eu/downloads/software/releases/downloads Download and install from source only when pre-compiled packages are unavailable for the platform.
2	Source code availability		Open source software	Pass	http://www.globus.org/toolkit/survey/index.php?download=gt5.2.0-all-source-installer.tar.gz
3	Installation/Administration Documentation	Administration guide describing installation, configuration and operation	All	Pass	IGE: Installation: http://www.ige-project.eu/downloads/documents/guide/component-installation-guide Administration: http://www.ige-project.eu/downloads/documents/guide/component-installation-guide Globus.org: Installation: http://globus.org/toolkit/docs/5.2/5.2.0/admin/install/#q-bininst Administration: http://globus.org/toolkit/docs/latest-stable/gridftp/admin/#gridftpAdmin
4	Functional Description	Code/Tool should provide a functional document describing its functionality	All	Pass	http://www.globus.org/toolkit/docs/latest-stable/gridftp/#gridftp
5	Release Notes	Include all changes in release: bug fixes and new features	All	Pass	http://www.globus.org/toolkit/docs/latest-stable/gridftp/rn/#gridftpRN
6	User Documentation	A document/readme describing how to use it.	All	Pass	http://www.globus.org/toolkit/docs/latest-stable/gridftp/user/#gridftpUser
7	Online help (man pages)	End user command line tools must include man pages or online help	End user command line tools	Pass	Checking for man page, local or on-line, or '-h' argument derivatives for each command line tool of the component. globus-url-copy -help
8	Software License	Code/Service should have a license to allow usage on shared resources (EGI/PRACE infrastructure) For Open Source products, compatible licenses are those accepted by the Open Source Initiative and categorised as "Popular and widely used or with strong communities": - Apache License, 2.0 (Apache-2.0) - BSD 3-Clause "New" or "Revised" license (BSD-3-Clause) - BSD 3-Clause "Simplified" or "FreeBSD" license (BSD-2-Clause) - GNU General Public License (GPL) - GNU Library or "Lesser" General Public License (LGPL) - MIT license (MIT) - Mozilla Public License 1.1 (MPL-1.1) - Common Development	All	Pass	Apache License 2.0 On SuperMIG: /lrz/sys/grid/globusToolkit/globus_superm/GLOBUS_LICENSE

		and Distribution License (CDDL-1.0) - Eclipse Public License (EPL-1.0) Other licenses accepted by the Open Source Initiative and listed as "Special Purpose" are compatible with the infrastructure (when applicable): - Educational Community License - IPA Font License (IPA) - NASA Open Source Agreement 1.3 (NASA-1.3) - Open Font License 1.1 (OFL-1.1) Any other license, and non Open Source products will have to be evaluated.			
9	Service reference card	For each of the services that a product runs, document its characteristics with a reference card. The document must exist and contain the following information for each service: ServiceName - Description: Description of the service - Init scripts: List of init scripts for the service, expected run levels - Daemons: List of daemons needed for the service - Configuration: List of configuration files used by the service - Logs: List of log files used by the service - Open ports: List of ports the service uses - Cron: List of crons used by the service - Other information: Any other relevant information about the service.	Services	Pass	Available from IGE: http://www.ige-project.eu/downloads/documents/service-reference-cards/globus-gridftp-server
10	API Documentation	Public API of product/appliances must be documented.	Open source software	Pass	http://www.globus.org/toolkit/docs/latest-stable/gridftp/pi/#gridftpPI

	Main Tests	**Additional Info**		**Pass/Fail**	**Comments**
1	Release changes testing	All the changes in a release should be tested, especially bug fixes	All		
2	Integration and Functionality Test	Interaction with other software modules should be tested.	All		
3	Regression Tests	Ensure that old bugs are still resolved and if new bugs appear.	All		
4	Backwards compatibility	Minor/Revision releases of a product must be backwards compatible.	All		
5	Service Control and Status	Services run must provide a mechanism for starting, stopping and quering the services following the OS init scripts conventions	Service	Pass	/etc/xinetd.d/gsiftp To change/query status: /etc/init.d/xinetd {start/stop/restart/status}

6	Service logs	All services should create log files where the service administrator can trace most relevant actions taken.	Service	Pass	Control Channel: /lrz/sys/grid/globusToolkit/var/log/superm/gridftp.log /lrz/sys/grid/globusToolkit/var/log/superm/gridftp-transfer.log Data Channel /lrz/sys/grid/globusToolkit/var/log/superm/gridftp-be.log /lrz/sys/grid/globusToolkit/var/log/superm/gridftp-transfer-be-login01.log /lrz/sys/grid/globusToolkit/var/log/superm/gridftp-transfer-be-login02.log

	Service tests	**Additional Info**		**Pass/Fail**	**Comments**
1	Service Reliability	Services must maintain a good performance and reliability over long periods of time with normal operation. Service must not show performance degradation during a 3-day period. The most important parameters to check are: - stable memory usage - throughput and/or response times remain stable during the period of activity (they should be as good or better than at the beginning of the test for similar requests)	Service		
2	Service Robustness	Services should not produce unexpected results or become uncontrollable when taxed beyond normal capacity. Services taxed beyond normal capacity: - should not become unresponsive to normal start/stop operations - must be able to start after a forceful stop - must not expose (potentially sensitive) memory contents to other processes - must not leave sensitive data in world-readable files - must not accept connections that would be refused under normal operating conditions	Service		

	Security tests	**Additional Info**		**Pass/Fail**	**Comments**
1	World writable files	Products must not create world-writable files or directories. - Start the service under test and initiate a standard client to use the service. Then using lsof check for open files by the service and test each one for world writable files. - Check for the files installed by the package and check each file/folder for world writable permissions	All	Pass	To find all world-writable files and directories, execute: # find . -path /proc -prune -o -perm -2 ! -type l -ls To find all files that are not owned by any user or group, execute: # find . -path /proc -prune -o -nouser -o -nogroup

2	System initialisation Files	System initialisation files (/etc/rc* , /etc/init.d/*, /etc/rc?.d) of the service should be protected such that only the root/service owner can write to them.	Service	Pass	

	Performance tests	**Additional Info**		**Rating**	**Comments**
1	Ease of installation	How easy is it to install this component. Rating: 1-5 (1 is very easy and 5 is very difficult)	All	Pass	
2	Performance of application codes	How many percent of peak performance. Rating 1-5 0-1%: 5 1-2.9%: 4 3-5%: 3 5%-10%: 2 >10%: 1	application codes	N/A	

Appendix D: GridFTP Specific Tests

	GridFTP Tests	Description
1	Start server via script (e.g. /etc/init.d/gsiftp start or /etc/xined.d/gsiftp start)	Test with command that server is started successfully >telnet localhost {port} Expected result: E.g. Trying 127.0.0.1... Connected to localhost. Escape character is '^]'. 220 GridFTP Server mldev.mcs.anl.gov 2.0 (gcc32dbg, 1113865414-1) ready.
2	Stop server via script (e.g. /etc/init.d/gsiftp stop or /etc/xined.d/gsiftp stop)	Test with command that server is stopped successfully >telnet localhost {port}
3	Restart server via script (e.g. /etc/init.d/gsiftp restart or /etc/xined.d/gsiftp restart)	Test with command that server is started successfully >telnet localhost {port} Expected result: E.g. Trying 127.0.0.1... Connected to localhost. Escape character is '^]'. 220 GridFTP Server mldev.mcs.anl.gov 2.0 (gcc32dbg, 1113865414-1) ready.
4	Status of server via script	
5	Transfer a 1GB file from local to local	Create a 1GB file in $HOME cd $HOME dd if=/dev/zero bs=1024 count=1000000 of=myfile.$USER Transfer the file globus-url-copy -vb file:///$HOME/myfile.$USER \ file:///tmp/
6	Transfer a 1GB file from local to remote	Transfer the file globus-url-copy -vb file:///$HOME/myfile.$USER \ gsiftp://{hostname:port}/tmp/
7	Transfer a 1GB file from remote to local and rename the file	Transfer the file globus-url-copy -vb gsiftp://{hostname:port}/tmp/myfile.$USER \ file:///$HOME/myfile2.$USER
8	Transfer a 1GB file from remote to remote	Transfer the file globus-url-copy -vb gsiftp://{hostname:port}/tmp/myfile2.$USER gsiftp://{hostname:port}/tmp/
9	Resume a 10GB file from local to remote. - Stop the transfer and restart it (Transfer should resume) - Verify that transferred file is not corrupted	Create a 10GB file in $HOME cd $HOME dd if=/dev/zero bs=1024 count=10000000 of=myfile.$USER Transfer the file globus-url-copy -vb -df transfer.out file:///$HOME/myfile.$USER \ gsiftp://{hostname:port}/tmp/ While transfer is ongoing, stop the transfer (e.g. with Ctrl+C) Open another terminal to check the "half-transferred" file. watch --interval=0.1 ls -la /tmp/myfile.$USER Continue transfer globus-url-copy -vb -df transfer.out Check transferred file that is on remote side is "continued" and not restarted. Move original file to remote side and do a "diff" between both files to ensure content is still the same.

10	Transfer a 100GB file from remote to remote - Stop the transfer and restart it (Transfer should resume) - Verify that transferred file is not corrupted	Create a 100GB file in /tmp/ cd /tmp dd if=/dev/zero bs=1024 count=100000000 of=myfile.$USER Transfer the file globus-url-copy -vb -df transfer.out gsiftp://{hostname:port}/tmp/myfile.$USER \ gsiftp://{hostname:port}/tmp/ While transfer is ongoing, stop the transfer (e.g. with Ctrl+C) Open another terminal to check the "half-transferred" file. watch --interval=0.1 ls -la /tmp/myfile.$USER Continue transfer globus-url-copy -vb -df transfer.out Check transferred file that is on remote side is "continued" and not restarted. Move original file to remote side and do a "diff" between both files to ensure content is still the same.
11	Performance Test: Transfer a 100GB file from local to remote with multiple streams. Performance should improve in most case.	globus-url-copy -vb -p 5 gsiftp://{hostname:port}/tmp/myfile.$USER \ gsiftp://{hostname:port}/tmp/myfile2.$USER
12	Stress test: Start 40 instances of local to remote 10G transfers	
13	Stress tests: directory with many small files	
14	Stess tests: Recursive directories E.g. With depth >3	
15	Feature tests: Test trasfer with flag -udt to test the udt protocol	

References

Aeschlimann, A. (2010, December 20). *GOCDB - EGIWiki*. Retrieved April 7, 2012 from egi.eu: https://wiki.egi.eu/wiki/GOCDB

EGI Wiki. (2010, December 20). *GOCDB - EGIWiki*. Retrieved April 7, 2012 from egi.eu: https://wiki.egi.eu/wiki/GOCDB

GLite. (2005). *glite-WMS*. Retrieved April 12, 2012 from glite.cern.ch: http://glite.cern.ch/glite-WMS/

Leffingwell, D. (2007). *Scaling Software Agility: Best Practices for Large Enterprises*. USA: Addison-Wesley Professional.

MAPPER. (2011). *D4.1 Review of Applications, Users, Software and e-Infrastructures*. London: MAPPER.

The Globus Alliance Wiki. (2005, October 21). *GRAM - Globus*. Retrieved April 12, 2012 from globus.org: http://dev.globus.org/wiki/GRAM

Glossary and Links

ADMIRE	Architectures for Data Intensive Research http://www.admire-project.eu/
ArcLink	A protocol for data transfer based on time windows http://www.seiscomp3.org/wiki/doc/applications/arclink
AXISEM	A parallel spectral-element method http://www.seg.ethz.ch/software/axisem
BADW-LRZ	The Bavarian Academy of Sciences and Humanities - Leibniz Supercomputing Centre http://www.lrz.de/english/
BDII	Berkeley Database Information Index
CREAM	Computing Resource Execution And Management
DCIs	Distributed Computing Infrastructures
DECI	Distributed European Computing Initiative / DEISA Extreme Computing Initiative
DEISA	Distributed European Infrastructure for Supercomputing Applications
DISPEL	The language for ADMIRE to describe complex data-intensive workflows http://www.admire-project.eu/data-computing/index.html
DoW	Description of Work
EGI	European Grid Infrastructure http://www.egi.eu/
gLite	Lightweight Middleware for Grid Computing http://glite.cern.ch/
EMI	European Middleware Initiative http://www.eu-emi.eu/
GUI	Graphical User Interface
Globus	Open source Grid software http://www.globus.org/
Globus Toolkit	Open source software toolkit used for building grids http://www.globus.org/toolkit/

Globus Online	A http://www.eu-emi.eu/ cloud-based reliable, high performance and secure service for managing file transfers https://www.globusonline.org/
GridSpace2	Provides a Web 2.0-based Experiment Workbench for joint development and execution of virtual experiments by groups of collaborating scientists. https://gs2.plgrid.pl/
GSI-SSHTerm	A Java based terminal client for accessing the Grid http://www.grid.lrz.de/en/mware/globus/client/gsissh_term.html
Kepler	A free and open source scientific workflow application https://kepler-project.org/
iRODS	Integrated Rule-Oriented Data-management System https://www.irods.org/
ITU	International Telecommunication Union
ISO 20000	The international standard for IT Service management http://20000.fwtk.org/iso-20000.htm
ITIL	Information Technology Infrastructure Library http://www.itil-officialsite.com/
JRA1	Equivalent to Work Package 8 (WP8)
JRA2	Equivalent to Work Package 9 (WP9)
LDAP	Lightweight Directory Access Protocol
MAPPER	Multiscale Applications on European e-Infrastructure http://www.mapper-project.eu
MoU	Memorandum of Understanding
NA2	Equivalent to Work Package 2 (WP2)
NCSA	National Center for Supercomputing Applications http://www.ncsa.illinois.edu/
ObsPy	A Python Toolbox for seismology/seismological observatories http://obspy.org/
OGSA-DAI	An innovative solution of distributed data access and management http://www.ogsadai.org.uk/
PRACE	Partnership for Advanced Computing in Europe

	http://www.prace-project.eu/
RAPID	Rapid portals for Seismological Waveform Data
	http://research.nesc.ac.uk/node/423
RegSEM	A Spectral Element Method code to compute seismic wave propagation
	http://www.ipgp.fr/~paulcup/RegSEM.html
SA1	Equivalent to Work Package 5 (WP5)
SA2	Equivalent to Work Package 6 (WP6)
SA3	Equivalent to Work Package 7 (WP7)
SAGA	A Simple API for Grid Applications
	http://www.saga-project.org/
SDX	Seismic Data eXplorer
SeisSol	A simulation software based on the Discontinuous Galerkin Finite Element Method
	http://www.geophysik.uni-muenchen.de/~kaeser/SeisSol/
SES3D	Programme package for simulation of elastic wave propagation in a spherical section and the computation of Frechet kernels
	http://www.geophysik.uni-muenchen.de/Members/fichtner/ses3d
SHIWA	Sharing Interoperable Workflows for large-scale simulations on Available DCIs
	http://www.shiwa-workflow.eu/
SPECFEM3D	A simulation software code based on the spectral-element method for 3D seismic wave propagation
	http://www.seg.ethz.ch/software/specfem3D
UMD	Universal Middleware Distribution
UNICORE	Uniform Interface to Computing Resources
	http://www.unicore.eu/
VOMS	A system to classify users that are a part of a Virtual Organisation (VO)
	http://vdt.cs.wisc.edu/VOMS-documentation.html
The PDCA cycle	The Plan-Do-Check-Act cycle
	http://labspace.open.ac.uk/mod/resource/view.php?id=346003
XSEDE	Extreme Science and Engineering Discovery Environment
	https://www.xsede.org/