Project number:RI 211727

Project acronym: EDGeS

Project full title: Enabling Desktop Grids for e-Science

Research Infrastructures
INFRA-2007-1.2.2 Deployment of eInfrastructures for scientific communities

DSA2.1 EDGeS Application Validation Procedure

Due date of deliverable: 30/06/2008 | Actual submission date: XXXXX

Start date of project: 01/01/2008 | Duration: 24 months

CIEMAT

Dissemination Level : RE

Version: 0.4

The EDGeS project has received Community research funding

1 Table of Contents

2 Status and Change History

Status:	Name:	Date:	Signature:
Draft:	Raúl Ramos, Manuel Rubio	10/05/2008	n.n. electronically
Reviewed:			n.n. electronically
Approved:			n.n. electronically

Version	Date	Pages	Author	Modification
0.1	15/05/2008	All	RRP MRS	Creation
0.2	27/05/2008	All	RRP MRS PK	Modifs by Peter, Manuel and Raul
0.3	29/05/2008	All	RRP MSR	Modifs after feedback in project meeting
0.4	31/05/2008	All	RRP MSR	Completed draft and included templates in annexes

3 Glossary

DG. (Desktop Grid) Variant of the volunteer computing model in which an organization that manages several desktop PCs (an enterprise, educational centre, etc) uses them for setting up a distributed project. The different PC computes together for this project. In this model the organization develops and ports its own applications and all DG clients should be managed automatically.

SG. (Service Grid) Distributed computing model in which computational resources are not attached to a centralized point. Different sites (institutions, centres) may join its heterogeneous resources (i.e. computing power, storage capacity, etc.) into a Virtual Organization. Any resource belonging to this VO is available to any site being part of it.

BOINC. (Berkeley Open Infrastructure for Network Computing) Open source middleware for building volunteer computing and desktop grid projects. It was originally developed for supporting SETI@Home project. Then it was generalized to support any other distributed project.

XtremWeb. Open source software for building Desktop Grid systems. Its architecture it is not only subject to centralized models, but also P2P and hierarchical ones are supported.

gLite. Lightweight middleware for Grid Computing. It is produced by the EGEE project and it consists of an integrated set of components designed to enable Grid resource sharing. Its distribution includes software for supporting services such as Metadata catalogue, file catalogue, Resource Broker, Information system, Computing element, Storage element, Worker nodes, X509 certificate and proxy authorization and authentication, etc.

EGEE. (Enabling Grids for E-sciencE). Project funded by the European Commission in the sixth framework programme. EGEE connects institutions across many countries in Europe in order to build a multi-science Grid infrastructure. Its objectives are to build a reliable, secure and robust grid infrastructure, to develop and maintain gLite middleware and attract and supports users and institutions towards Grid technology.

Interoperability. It is the ability for systems or organizations for working together. Applied to software terms, interoperability refers to the capability of the different programs to communicate among them by exchanging formats, using the same file formats, or using protocols.

VO. (Virtual Organization). It is a group of individuals or institutions that share the computing resources of a Grid system. When a user authenticates its certificate against a VO, he can use all the resources available within this VO.

Multiplatform. It refers to computer programs, operating systems, computing or programming languages or other kind of software that can be made to work on different computer platforms (which can refer to software framework, such as Windows XP, Linux, OS X or hardware architecture, such as x86, PowerPC,etc).

4 Content

4.1 Introduction

The EDGeS project is producing the technology to make BOINC and XtremWeb Desktop Grids (DGs) interoperable with gLite based Service Grids (SGs) as deployed on the EGEE infrastructure. This technology is delivered as ***bridges*** between SG and DG middleware and users will be able to share jobs, resources and applications across desktop and service Grids, thus enlarging the possibilities for user communities to exploit infrastructures managed by different middleware. EDGeS is providing a production infrastructure, containing many desktop computers, connected to the EGEE infrastructure and thus allowing effective sharing of resources from both infrastructures to all users.

Technological interoperability constitutes the foundation for ***seamless interoperability*** that must also encompass other issues of organizational and conceptual nature. In some aspects, applications are engineered differently for DGs and SGs. For instance, applications targeted to run on DGs typically contain regular checkpoints to be able to resume work after unexpected desktop user interruptions, or applications targeted to run on SGs typically have little restrictions regarding network bandwidth. The following constitutes a non-exhaustive list of such issues that, at different levels, the EDGeS project must solve:

- ***Application models***. DG applications generally execute within more restricted sandbox environments their SG counterparts. SG applications typically have access to a wider set of resources with fewer limitations (bandwidth, memory, etc.), DG applications must take into account contingencies such as application interruption, multiplatform availability, etc. which, although significant, are not equally important in SG, etc.

- ***Authentication models***. SGs typically rely on strong VO based authentication and authorization models, whereas DGs are more prone to having anonymous or loosely identified users. This has great impact on the different policies for resource usage and access and also on other issues such as monitoring, usage accounting, etc.

- ***Virtual organizations***. SG communities organize around the notion of a virtual organization, which regulate issues such as user access, resource engagements, application requirements, etc. DG communities have more informal arrangements organized around each particular application with little regulations.

- ***Resource availability***. SGs are able to guarantee resources for a given community (VO) requiring a certain scientific production and thus, can timely plan production of scientific results. DGs rely on resources as they become available, which prevents them to guarantee a priori availability of resources. Also elements of SGs (computing nodes) typically are connected to high speed networks, which allows easy transportation of bulk input or output files. This is not the case of DGs where bandwidth is typically limited.

- ***APIs a programming tools***. Applications are transformed differently whether they are to be used on DGs or on SGs. This means that applications are built upon different API (Application Programming Interfaces) or tools.

- ***Platform availability***: DGs applications are typically available in popular end-user platforms (Windows, MacOS, Ubuntu). This makes them more ubiquitous but at the same time more prone to viruses and attacks than other platforms typically used in SGs.

For these issues, and many others, EDGeS must engineer solutions of technical and/or organizational nature. One representative scenario is authentication reconciliation. This is, as DGs users are typically loosely authenticated, the following question arises: under which EGEE identity will jobs generated in the EDGeS desktop infrastructure run on the EGEE infrastructure? For practical solutions one must take compromises. One possible compromise is to have an EDGeS virtual organization, accepted within EGEE, through which jobs coming from DGs authenticate into the EGEE infrastructure. Each DG application could then have a single EDGeS user certificate used by all jobs generated by such application. Many other approaches could exist for this and each scenario.

This is just one example, and it is envisaged that solutions of this kind might be adopted in the future to sort out the issues arising by the different natures of DGs and SGs. In general, as access to SG environments is more restrictive, any such solution will imply SG infrastructures to build ***trust*** on the DG infrastructures with which they operate, beyond their own particular trust model. In the example scenario above, the EGEE infrastructure would have to trust all the jobs generated by the one EDGeS application through the same VO (the EDGeS VO) and user certificate.

Therefore, in order to contribute building this kind of trust, in a transparent and objective manner, applications coming from the EDGeS infrastructures are required to pass a ***EDGeS Application Validation Procedure***, through which the EDGeS community will certify the applications hosted at the EDGeS infrastructure. Thus, we create a new category of applications, named ***EDGeS Certified Applications*** that, by virtue of the validation procedure, have been measured to have acceptable behaviour standards through public objective criteria.

This document describes the ***EDGeS Application Validation Procedure***, which responds to the problematic just exposed. Starting from a Candidate Application, the procedure produces and ***EDGeS Application Validation Report*** and, if successful, an ***EDGeS Certified Application***. The report will mainly describe the measured ***Application Profile*** of the candidate application, which includes issues such as its scientific purpose, number of checkpoints, typical job runtime or memory consumption, results of integrity and virus checks, etc.

4.2 Goals of the EDGeS application validation procedure

The general goal of the procedure to certify applications for running on the EDGeS infrastructure. Certification aims at obtaining the following goals:

Objective 1. To understand the nature of the application
This objective has three parts. First is to understand the scientific and computational task that the application aims at solving. This includes understanding the specific calculations the application carries out, the scientific relevance within its community of solving the problems the application is addressing, the scientific impact of the expected results, etc.

Second goal is to understand non technical issues such as ethical implications or appropriate usage of the computational resources that the application might use through the EDGeS infrastructures.

Finally, the goal is to understand the application from a technical point of view. This includes deployment information about the application itself (what are its binaries, libraries, dependencies, file layout, etc.) and information about how the application is engineered (what kind of parallelism it exploits, what workflows it belongs to, etc.)

Objective 2. To experimentally measure application behaviour
Applications to be effectively deployed on the EDGeS infrastructure will have been measured and observed for proper behaviour. This implies building knowledge on how the application uses the computational resources, obtaining experimentally metrics on the application behaviour, such as average job runtime, input/output file size, resources comsumption, etc. and running integrity checks on the application.

The goal behind this objective is to have a priori knowledge on application behaviour and to ensure, as best as possible, the applications will use the computational resources on which they are deploy in an expected manner.

Objective 3. To build confidence on the usage of the resources
Interoperability among infrastructures implies that resources are shared along with authorship and certain responsibilities. At the heart of interoperability lies the notion of trust between interoperating infrastructure. Trust is built on top to technological choices but also on top of building a common culture of understanding. The bottom line behind this procedure is to provide transparency and objectivity on how applications are to exploit the resources across the interoperable infrastructures.

It is important to note that each interoperable infrastructure has its own administrators, responsible people, users, etc. Teams from each infrastructure must share clear understanding and limitations of responsibilities. This encompasses many scenarios. If EDGeS applications are to use EGEE resources through a generic EDGeS VO, VO administrators must feel confident about the applications they allow in their VO, as they will be held responsible for their misbehaviour. If desktop users are to yield their PCs to the EDGeS infrastructures they must be given minimum guarantees that no harmful applications will end up in their PCs, etc.

The different activities enforced by the EDGeS Application Validation Procedure are aimed ultimately at building and sharing this confidence from an objective and measurable point of view.

Objective 4. To be able to plan for long term sustained production
Application validation and certification constitutes a milestone in an application's life cycle. It signals the application is ready for production and can be effectively used to produce results in a massive and sustained manner. Long term scientific production can only be plan if we can correlate a concrete need for scientific results with the computational resources needed to achieve such results. Therefore, the application validation procedure must provide quantitative criteria on application behaviour allowing future scientific productions to be properly planned over computational resources.

Specifically, the procedure must obtain a description and dimensioning of a typical scientific run with relevant significance and the computational resources required to obtain run it. This is to be built upon the measured behaviour required by Objective 2 above. For instance, if for a specific application we measure that a typical job lasts two hours on a standard CPU, and a typical data challenge required 100 jobs we can expect to require 200 CPU hours to run the data challenge.

4.3 Roles and definitions

EDGeS Application Validation Procedure	Procedure by which a Candidate Application becomes an EDGeS Certified Application apt to run on the EDGeS Production Infrastructure
EDGeS Candidate Application	Application considered to be certified to use the EDGeS Production Infrastructure. New releases of already certified applications are considered also as candidate applications and, therefore, must pass the EDGeS Application Validation Procedure.
EDGeS Certified Application	Application that has gone successfully through the EDGeS Validation Procedure and thus, it can be deployed on the EDGeS Production Infrastructure for production usage.
EDGeS Infrastructure Administrator (or Manager)	Person responsible of the EDGeS infrastructure, regulating its usage, evolution, applications deployment, access, etc.
EDGeS Application Certifier	Person who controls the EDGeS Application Validation Procedure and, at last, guarantees a Candidate Application has successfully become (or not) an EDGeS Certified Application by passing the EDGeS Application Validation Procedure.
EDGeS Test Infrastructure	EDGeS infrastructure used for running the EDGeS Application Validation Procedure
EDGeS Production Infrastructure	EDGeS infrastructure used for running only EDGeS Certified Applications for production usage.
Application Profile	Qualitative and quantitative characterization of an application
Application Provider	Owner of an application. It is typically the research group or individual that has developed the application and will be used by her community. She has the knowledge on how the application has been built and how it can be used for scientific production
Application Metrics	Definition of the quantitative measures of the behaviour of an application. It includes metrics such as average run time, input file size, etc.
Measurement	Effective measurement of applications metrics on a given infrastructure
Scientific Production	Sustained run of an application on an infrastructure to produce a certain set of scientific results.
Volumetry	Aggregated metrics for a certain scientific production. For instance, running an application X to obtain N results requires 100Gb of input data, produces 1Tb of output data and needs 1000 CPU hours.
Test Scientific Production	Sample of a scientific production used for testing and metrics measurements purposes.

4.4 Application profile

An application profile contains all significant information to understand what an application does, what resources it will need to obtain certain scientific results and how it will use the infrastructure.

The aim is that, starting from a set of requirements for a scientific production and an application profile, we can know how much resources the application will need to produce the expected scientific results. For instance, for an application for statistical analysis we might need to process, say, 1 million samples for an established data challenge. Knowing how much CPU time the application takes in analysing one sample, the amount of memory it needs, the amount of disk space each sample takes, etc. we are in a position of knowing how many computational resources will be needed to make the data challenge.

Additional qualitative information about the application itself (such as the degree of parallelism, user interaction or the dependencies among the analysis of each sample) provides better understating on the application behaviour and allows better estimation of the computational resources needed.

Considering the particularities of the EDGeS infrastructure, this information is especially valuable in determining how appropriate would be to deploy and run an application on the EDGeS infrastructure. For instance, applications having large input files are difficult to run on desktop computers as there are little guarantees for bandwidth availability. Likewise, applications with large computing times with no checkpoints are not adequate to desktop computers as there is great probability of user interruption.

In order to measure how much an application is adequate for EDGeS we define a *Target Application Profile* that encapsulates a model EDGeS application. Therefore, adequateness of applications is measured against this *Target Application Profile*.

In general, we structure an ***Application Profile*** as containing two parts, the ***Application Description*** and the ***Application Metrics***. The *Application Description* is a qualitative characterization of the application, including its scientific purpose, what it calculates, the kind of input files it processes, the degree of parallelism that it can support, etc. The *Application Metrics* provide a quantitative characterization of the application. It contains metrics such as average job run time, expected input file size, number of checkpoints, etc.

The following three sections are structured as follows. The specification of the two parts of an *Application Profile* is described in sections 4.4.1 and 1.1. Then, section 1.1 describes the *Target Application Profile* as the model application for EDGeS.

4.4.1 Application description

This section gathers information for help the EDGeS infrastructure administrator and the EDGeS application certifier to know in detail the most relevant aspects of the application such as operation, management, architecture and needs. So, being aware of the cuantitative application details, this section provides a complete idea about the candidate application profile.

4.4.1.1 Application purpose

This section includes information about the scientific purposes for the application. Basic information such as the scientific field (i.e. biomedicine, physics, hardware development, etc) and the concrete application purposes (description about the application objective) should be provided in order to help catalogue properly the application.

There also is desirable to know the nature of results obtained (i.e. statistic analysis, images processed, etc) and an estimation about the benefits to having the application ported to the EDGeS platform.

4.4.1.2 Technical description

Documentation

The documentation description deals about some issues needed to completely know how to use the application and produce results:

- Installation instructions. The application provider should give details explaining how to install the application. I.e. operating system, path, environment variables, software prerequisites, un-package commands, etc.
- Running instructions. How a normal operation for the application should be launched. The application provider will specify the application command (or interface details), the parameters needed and what they mean, or another related information.
- Result management. A description of the results produced and where they are stored when the execution ends, as well as an estimation of required disk space. If the results were needed as input for other applications, post-processing details would be desirable.

Platforms

Platform(s) in which the application is available. For a complete list of platform, see ANEX IV.

If the application is multi-platform, specify what is the best one in terms of performance and/or stability.

If the application runs on single platform should be desirable an effort estimation about porting it to other platforms. The following levels are given as examples:

- Straightforward. A recompilation in the new platform suffices.

- High cost requiring. (I.e. a complete and proof understanding of the application is needed)
- Non-possible due to libraries availability in the other platforms.
- Non-possible due to source code is not available.
- Other reasons.

Security access

In this section the application provider should indicate the possible security issues the application has. For example, login details, certificate provision, key-signed files, usage of a concrete VO, etc.

Job submission

The information to be included here is about the parallelization of the application. In other words, in which the distributed version of the application consists of. This information can be grouped in the following parts:

- Distributed architecture. The most common one is to launch the jobs in parallel way, but alternative ways should be considered, such as standalone execution, special workflows, etc.
- Kind of parallel jobs. This subsection explains how tasks are run in the distributed architecture. Most common types are:
  - Parameter sweeping. This kind of parallelization sends a set of jobs based on the same executable, each job receiving a different set of parameters.
  - Data chunking. In this case a large amount of data has to be analyzed or processed, and the parallelization consists of dividing the data into small chunks to be analyzed, sending each one to a remote node.
  - Statistical simulation. This model is useful for massive computations that obtain statistical results, and the representativeness and accuracy of these results depend of how big is the population of samples simulated.
  - DAG Jobs. This model represents a case in which dependences between jobs are presented. In other words, it is needed to finish a job before starting a new one, because its output is used as the input for the next one.
  - Interactive jobs. They need the user interaction in certain moments during the execution, so the user performs a direct connection with the stdin.
  - Other. There may appear other execution models different to the ones described before, i.e. complex workflows or combinations between Parameter sweeping and Data chunking.

Data management

This section will include all the information related with data storage and accessibility. The application provider should detail how its application accesses to the data and provides information such as:

- Database usage. The application may perform connections to a database engine during its execution, so the application provider should specify the connection details (engine, login, etc.)
- Metadata catalogue usage. (AMGA, OGSADay, etc) Metadata usage in Grid environments is used in most of the case for describing pieces of data stored in the File Catalogue. If the application uses some metadata cataloguing system, its details (catalogue system, connection details, etc.) should appear here.
- File catalogue usage. This is a service used for storing large files on the Grid Storage Elements and retrieving them. It is also used for replicate the files due to security or performance reasons. If the application uses it, the application provider should describe which kind of data is stored on the catalogue.
- Access rights and data encryption. If any of the models described before store the data in encrypted way or require special access rights, the application provider should provide information either for the information encrypted or the access rights.

Information system and check pointing

It is possible the application gives feedback of its status to the user periodically. This fact should be registered since this status information could be useful as a trickle message or as a log message. If the application does check pointing this should be also reflected here.

Network requirements

This section gathers information about the network usage the application may need. So, information about the following points should be provided:

- Need to access to network. Some services such as database, catalogue or login will need network access. These services should be named here.
- Data encryption during communications. This point reflects if the data to be sent through network is to be encrypted. This is a desirable feature for all applications.
- Communications topology. The application provider should describe briefly the communications topology its application has. Example of communication topologies are master-worker (or star) model, bus model, mesh model, etc. Master-worker model is the most appropriate for DG applications.

4.4.2 Application metrics

This section measures the most relevant aspects of the application from the technical, resource requirements and performance points of view. Part 1 of validation report (Application profile) must contain how the metrics are to be measured for a particular application. Part 3 of the report (Certification) will contain the different metrics measured values for the application on the EDGeS test infrastructure. The following table lists the points to measure, and shows a brief explanation for each one:

M1 Files size All file sizes can be obtained by executing "ls -lh <file> | cut -f 5 -d " ""	
M1.1 Executable file size	Executable file size must be measured.
M1.2 Additional libraries size	Is the size addition of all auxiliary libraries (software prerrequisites excluding OS libraries) the application needs.
M1.3 Input sandbox size (Average)	Size addition of all files belonging to the input sandbox.
M1.4 Output sandbox size (Average)	Size addition of all files belonging to the output sandbox.
M2 Distributed execution	
M2.1 Number of CPUs needed (Jobs in parallel)	This value depends of the expected volumetric for the scientific production.
M2.2 Job running time (Average in seconds)	Time measurement (performed through *time* command) for a typical job. The result should be obtained as the average of ten executions.
M2.3 Job memory usage (Average)	This value represents the real memory size of the process. It can be obtained (in bytes) with *ps* command: *ps aux*. The memory value is under the RSS (6th) column. This value should be obtained when the job has been running during 20% of its average job running time. Since it is an average, it should be calculated by performing 10 executions.
M2.4 Free disk space needed during execution (Average)	This value does not represent the space needed to store the executable or the libraries, but represents, for example, the space needed for intermediate files or swap space in case they exist. It is an average value.
M3 Information system	
M3.1 Check pointing frequency (In seconds)	If the DG application provider has enabled checkpoint feature, its frequency value should be provided here.
M3.2 Trickle messages frequency (In seconds)	If the DG application provider has enabled trickle messages in the application, its frequency value should be provided here.
M3.3 Log size	Total amount of bytes occupied by the log generated during the execution, in case it exists.
M4 Data management	
M4.1 Database usage	Brief description of the database usage the application performs. i.e. number of times, frequency, etc.
M4.2 Metadata catalogue usage	Brief description of the metadata catalogue usage the application performs.

M4.3 File catalogue usage	Brief description of the file catalogue usage the application performs.
M5 Network requirements	
M5.1 Bandwidth usage (In kbps)	If the application provider knows its application reach certain bandwidth usage, this value it should be given here.
M5.2 Periodicity of communications (Average, in times per minute)	How many connections the application does in a minute? (i.e. for using the database, catalogue, communicate with other process, etc). It is an average value.

-

4.4.3 Target application profile

This section describes the ideal profile of application to be run on the EDGeS platform. It can be used as a criteria for giving the application certification or denial. For each sub section of 4.4.1 and for each measurable point reflected in section 4.4.2 values or ranges of values are provided. In the case of the application metrics, the values don't reflect an existing application, but have been chosen taking in account the best profile for both DG and SG infrastructures.

This set of reference values is not fixed, but it is subject to evolution beyond what is stated in this document. As well as new applications are analyzed and hosted in the EDGeS infrastructure new values for this profile can be set, so further application profiles will be published. A document called "EDGeS Target Application Profile" will be created for this purpose.

Application description

Purpose

Scientific or commercial applications that fit in the EDGeS objectives. No harmful or undesirable purposes can be accepted, such as reverse engineering, cryptography issues such as decryption, brute force password breaking, military purposes, etc.

Platforms

The best option is to have the application available in Linux, Windows and MAC. If it is not the case, the application should be available in at least one of them.

Security access

No security access should be present when running the application under DG environments, since the DG client as to be plenty aware for this issue.

Job submission. (Distributed architecture and kind of parallelization)

The best-distributed architecture is a massive parallelized one in which lots of independent processes run separately and have no interactions (they do not communicate and they have no dependences) between them.

Data management

Since DG clients doesn't have access to databases or SEs, the only input and output of data for the application should be based on text or binary files.

Information system and check pointing

For those applications having long time consuming tasks frequent reporting of its status should be performed. In a DG case this may be done through a trickle message shown in the client manager or in the screensaver. In order to avoid errors and to reduce the volatility of DG running a check pointing system should be implemented.

Network requirements

The best case for running applications in DG clients is to minimize the network usage, so applications having light input and output files and having a low frequent communication rate represent the best profile for this case.

Application metrics

These are the proposed metrics for the profile application.

M1 Files size	
M1.1 Executable file size	< 2MB
M1.2 Additional libraries size	< 2MB
M1.3 Input sandbox size (Average)	< 1MB
M1.4 Output sandbox size (Average)	< 1MB
M2 Distributed execution	
M2.1 Number of CPUs needed (Jobs in parallel)	No limits are to be set in this point since usage of large DG and SG system are planned.
M2.2 Job running time (Average in seconds)	< 45 min (DG system has high volatility, so a limited upper bound for running time must be set)
M2.3 Job memory usage (Average)	< 512 MB (Since DG clients are mostly based on home desktop PCs, a low value must be set here in order not to interfere with other client's processes)
M2.4 Free disk space needed during execution (Average)	<2 GB (Again, this value has been chosen taking in account the limited resources of a common DG client)
M3 Information system	
M3.1 Check pointing frequency (In minutes)	Each 10 – 30 minutes
M3.2 Trickle messages frequency (In seconds)	Each 30 – 120 minutes
M3.3 Log size	< 1MB

M4 Data management	
M4.1 Database usage	No database usage is desirable (taking in account the DG model)
M4.2 Metadata catalogue usage	No metadata catalogue is desirable (taking in account the DG model)
M4.3 File catalogue usage	No file catalogue is desirable (taking in account the DG model)
M5 Network requirements	
M5.1 Bandwidth usage (In kbps)	< 512 kbps
M5.2 Periodicity of communications (Average, in times per minute)	Communications should be only performed at the beginning and end of the process, as well as for the Information System services. No additional communications are desirable, since the DG case has to be again considered.

4.5 *Procedure overview*

Starting from a candidate application, the *Application Validation Procedure* produces an *Application Validation Report* which (1) describes and measures the application and its behaviour and (2) certifies the application conformance to run on the EDGeS infrastructure. Once the application has gone successfully through the procedure it becomes an **EDGeS certified application**. We consider an application as an unmodifiable unit, i.e., new releases of the application are required to pass this procedure, probably reusing validation reports from precedent releases of the application but, in any case, preserving the aims of the procedure.

The procedure is structured in four phases, each one producing one part of the report. This is represented in the following figure:

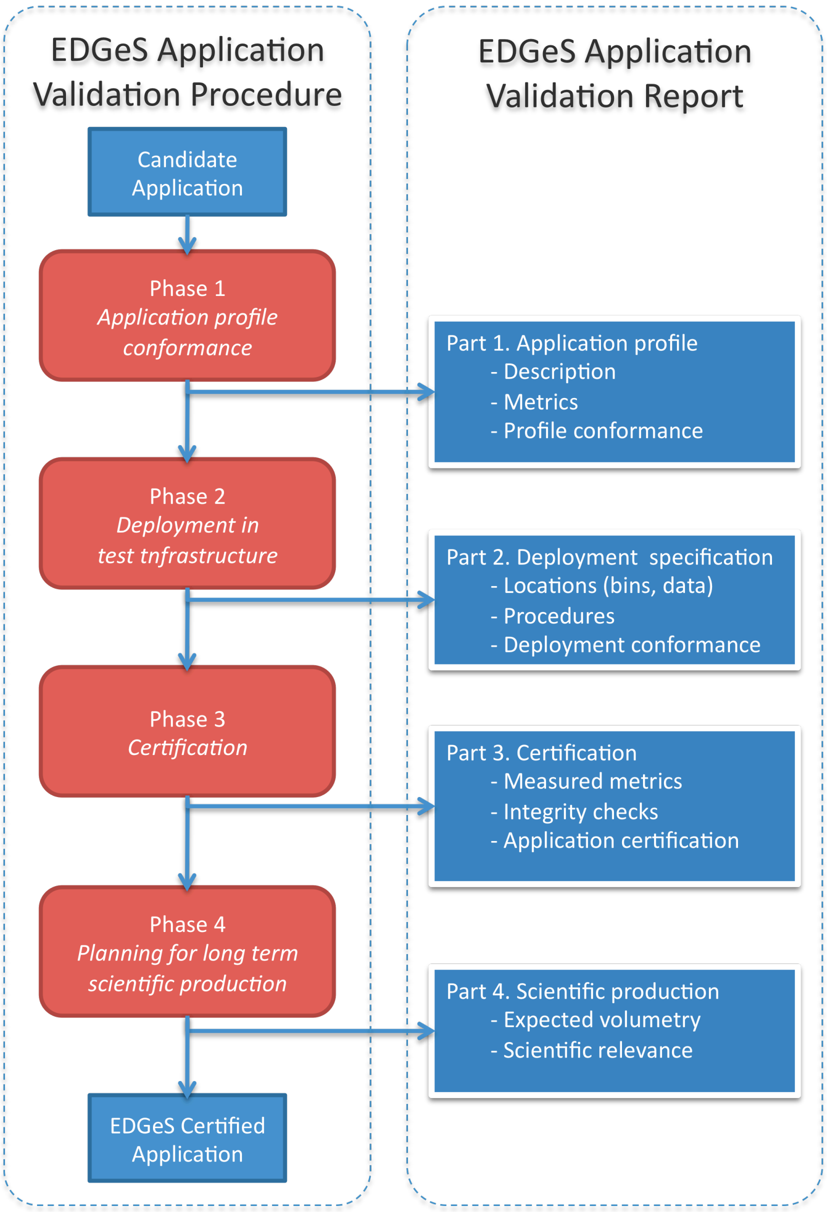
EDGeS Application Validation Procedure
EDGeS Application Validation Report
Candidate Application
Phase 1 *Application profile conformance*
Phase 2 *Deployment in test tnfrastructure*
Phase 3 *Certification*
Phase 4 *Planning for long term scientific production*
EDGeS Certified Application
Part 1. Application profile - Description - Metrics - Profile conformance
Part 2. Deployment specification - Locations (bins, data) - Procedures - Deployment conformance
Part 3. Certification - Measured metrics - Integrity checks - Application certification
Part 4. Scientific production - Expected volumetry - Scientific relevance

Figure 1: Application validation procedure outline

The following sections describe in short each phase:

Phase 1: Application profile conformance. Aims at obtaining the candidate application profile and an evaluation judgement on conformance with respect to the Target Application Profile

Phase 2: Deployment in test infrastructure. Aims at deploying the candidate application on the test infrastructure, gaining knowledge on how to deploy the application and obtaining conformance with respect to the infrastructure

Phase 3: Certification. Aims at measuring the application metrics on the candidate application deployed on the test infrastructure and perform integrity and health checks on the application

Phase 4: Planning for long term scientific production. Aims at gathering volumetry information for sustained scientific production on the EDGeS production infrastructure yielding to significant scientific results.

Each phase is described in terms of:

• **Purpose**	main goal of the phase
• **Actors**	who participates in the phase
• **Inputs**	what are the inputs of the phase
• **Procedure**	how the phase is to be executed
• **Output**	what the phase produces

4.6 Phase 1: Application profile conformance

Purpose	1. To obtain the Application Profile of the Candidate Application 2. To evaluate how closely the application fits the Target Application Profile 3. Certify conformance with respect to the Target Application Profile
Actors	Application provider EDGeS certifier
Inputs	1. The description of the Target Application Profile contained in this document 2. The Candidate Application, including information, documents and informal knowledge owned by the Application Provider.
Procedure	1. For the Application Description, the Application Provider gives information and explanation for each issue mentioned in section 4.4.1 above 2. For the Application Metrics, the Application Provider describes HOW each metric defined in section 1.1 above is to be measured for the candidate application. For instance, input file size is obtained by summing up files named initial-conditions.dat and data.dat whose typical contents are such and such, or maximum memory usage is obtained after 10 minutes of application running. 3. The EDGeS Certifier assists the Application Provider in points 1 and 2 above. 4. The EDGeS Certifier evaluates how much the Application Profile given by the Application Provider conforms the Target Application Profile 5. The EDGeS Certifier gives/denies conformance
Outputs	Part 1 of the Application Validation Report, containing: • The Application Profile of the Candidate Application, containing the Application Description and the Application Metrics. • The evaluation of conformance with respect to the Target Application Profile • The grant or denial of conformance

4.7 Phase 2: Deployment in test infrastructure

Purpose	1. To make a procedure on how to deploy the Candidate Application on the EDGeS Test Infrastructure 2. To effectively deploy the Candidate Application on the EDGeS test infrastructure and ensure it runs correctly 3. To certify conformance for deployment on the infrastructure
Actors	Application provider EDGeS Test Infrastructure Administrator EDGeS certifier
Inputs	1. The Candidate Application Profile obtained in Phase 1 2. The EDGeS Test Infrastructure description
Procedure	1. The Application Provider together with the EDGeS Test Infrastructure Administrator install the candidate application on the EDGeS Test Infrastructure 2. The Application Provider together with the EDGeS Test Infrastructure Administrator test expected behaviour of the application 3. EDGeS Test Infastructure Administrator notifies EDGeS Certifier on correct deployment so that he can give conformance.
Outputs	Part 2 of the Application Validation Report, containing: • How the application is deployed, providing information about operating system, path, environment variables, software prerequisites, location of the deployed components, etc. • How to use the deployed application. This describes a normal operation for the application, including how to send jobs, access the data, etc. • Result management. Where the results are stored. If the results are used as the input for other application, provide post-processing details. • The grant or denial of conformance

4.8 Phase 3: Application certification

Purpose	1. To measure and validate on the EDGeS Test Infrastructure the Application Metrics of the Candidate Application as defined in Phase 1 2. To run integrity checks application multi platform portability, for virus, etc. 3. To run health checks for memory leaks, unsustained CPU consumption, etc. 4. To run a sample scientific production To effectively deploy the Candidate Application on the EDGeS test infrastructure and ensure it runs correctly 3. To certify conformance for deployment on the infrastructure
Actors	Application provider EDGeS Test Infrastructure Administrator EDGeS Certifier
Inputs	1. The Candidate Application Profile obtained in Phase 2 2. The Candidate Appication deployed on the EDGeS Test Infrastructure
Procedure	1. The Application Provider together with the EDGeS Certifier define a sample scientific production to be run on the EDGeS Test Infrastructure. The goal of this production is to obtain concrete values for the application metrics. The EDGeS Test Infrastructure Administrator must validate the size of this sample production as appropriate to obtain meaningful values for the application metrics. 2. The sample scientific production is launched once and the application metrics monitored and measured 3. The sample scientific production is launched a second time and it is monitored for healthy sustained resources consumption (memory leaks, CPU, temporary files, etc.). This is to be performed on the different platforms that may participate in the application running. This must include, gLite jobs, Boinc/XtremWeb clients on the different platforms and accessory applications such as validators, assimilators, etc. 4. The different components of the Candidate Application are validated against relevant integrity checks. For instance, Windows client components are run against standard antivirus 5. The application is inspected for proper existence and management of checkpoints with respect to the Target Application Profile 6. EDGeS Test Infastructure Administrator notifies EDGeS Certifier on measurements and results so that he can give final application certification.
Outputs	Part 3 of the Application Validation Report, containing: • The definition of the sample scientific production • Measured values of the application metrics of the sample scientific production run on the EDGeS Test Infrastructure as described in point 2 of the procedure. • The results of the sustained resources consumption as described in point 3 of the procedure • The results of the integrity checks as described in point 4 of the procedure • The results of the checkpoints assessment as described in point 5 of the procedure • EDGeS Certification grant or denial

4.9 Phase 4: Planning for long term scientific production

Purpose	1. To prepare for production usage of the Certified Application on EDGeS Production Infrastructure 2. To determine the volumetry of a typical run with scientific significance (for instance a Data Challenge) 3. To determine the resources required to serve the scientific community behind the application on a sustained basis.
Actors	Application provider EDGeS Certifier
Inputs	1. The Candidate Application Profile obtained in Phase 1 2. The results obtained in Phase 3
Procedure	1. The Application Provider dimensions, in terms of scientific results, the typical size of a scientific run providing significant scientific results. In many fields this is done in terms of a Data Challenge 2. The Application Provider together with the EDGeS Certifier assess the expected volumetry of a typical run. This consists of expected values of the application metrics as defined previously. For instance, number and size of input files, number of jobs, etc. 3. The Application Provider together with the EDGeS Certifier assess how many resources are needed to serve the volumetry determined above. 3. The Application Provider together with the EDGeS Certifier assess the volumetry and resources needed for a sustained scientific production over the EDGeS Production Infrastructure.
Outputs	Part 4 of the Application Validation Report, containing: • A dimensioning for the volumetry and resources for sustained scientific production • An evaluation of the scientific impact of the results obtained through such volumetry

5 Other considerations

We are considering the development of a lightweight validation procedure for selected applications. Trust is built through other sources (for instance SETI@home).

6 Annex I. Template for EDGeS Application Validation Report

The following template constitutes the EDGeS Application Validation report and is to be produced throughout the execution of the validation procedure described in this document.

GENERAL INFORMATION			
Application name:		**Date:**	
Application provider:			
University /research group:			
Application summary:			
PART 1. APPLICATION PROFILE			
Application purpose			
PART 1.1 Application description			
1.1.1 Platforms			
Platforms:		**Preferred platform:**	
Effort estimation in case of porting to other platforms:			
1.1.2 Security access			
Security access details:			
1.1.3 Job submission			
Description of distributed architecture:			
Kind of parallel jobs:			
1.1.4 Data management			
Database usage:			
Metadata catalogue usage:			
File catalogue usage:			
Access rights/ data encryption:			
1.1.5 Network requirements			
Network usage:			
Data encryption:			
Communications topology:			

Other network issues:	
PART 1.2 Application metrics (Define here how these metrics will be obtained):	
M1 Files sizes (specify here WHAT files are to be measured for size)	
M1.1 Executable file size	
M1.2 Additional libraries size	
M1.3 Input sandbox size	
M1.4 Output sandbox size	
M2 Distributed execution (describe here HOW to measure these metrics)	
M2.1 Number of CPUs needed (Jobs in parallel)	*What process to monitor for CPU usage*
M2.2 Job running time (Average in seconds)	*What process to monitor for time*
M2.3 Job memory usage (Average)	*What process to monitor for mem usage*
M2.4 Free disk space needed during execution (Average)	*What directories to monitor*
M3 Information system (describe here HOW to measure these metrics)	
M3.1 Check pointing frequency (In seconds)	*How do we know that a checkpoint has been made (temp files, etc.)*
M3.2 Trickle messages frequency (In seconds)	*How do we know a trickle message has been sent*
M3.3 Log size	*What files contain the logs*
M4 Data management (describe here HOW to measure these metrics)	
M4.1 Database usage	*What are the addresses of DB servers, login info, etc..*
M4.2 Metadata catalogue usage	*What are the addresses of metacatalogs, login info, etc.*
M4.3 File catalogue usage	*What are the addresses of file catalogs, login info, etc.*
M5 Network requirements (describe here HOW to measure these metrics)	
M5.1 Bandwidth usage (In kbps)	*What connections should be monitored*
M5.2 Periodicity of communications (Average, in times per minute)	*What connections should be monitored*
Application profile conformance:	(yes/no/comments)

PART 2. APPLICATION DEPLOYMENT	
PART 2.1 Deployment information	
Operating system:	
Path:	
Environment variables:	
Software prerequisites:	
Location of deployed components:	
Other issues:	
PART 2.2 Application operation	
Command line executable:	
How to send jobs:	
Access data information:	
Other issues:	
PART 2.3 Results management	
Results location:	
Post-processing details:	
Application deployment conformance:	(yes/no/comments)

PART 3. APPLICATION CERTIFICATION	
PART 3.1 Description of sample scientific production	
Description of sample:	
Nature of input data:	
Nature of results:	
Size of input data:	
Size of results:	
Number of jobs needed to process input data:	
Hours-CPUs needed:	
Post-processing details:	
Other issues:	
PART 3.2 Measured values for application metrics	
M1 Files size (measures taken following specifications in PART 1.2)	
M1.1 Executable file size	
M1.2 Additional libraries size	
M1.3 Input sandbox size (Average)	
M1.4 Output sandbox size (Average)	
M2 Distributed execution (measures taken following specifications in PART 1.2)	
M2.1 Number of CPUs needed (Jobs in parallel)	
M2.2 Job running time (Average in seconds)	
M2.3 Job memory usage (Average)	
M2.4 Free disk space needed during execution (Average)	
M3 Information system (measures taken following specifications in PART 1.2)	
M3.1 Check pointing frequency (In seconds)	
M3.2 Trickle messages frequency (In seconds)	
M3.3 Log size	
M4 Data management (measures taken following specifications in PART 1.2)	
M4.1 Database usage	

M4.2 Metadata catalogue usage	
M4.3 File catalogue usage	
M5 Network requirements (measures taken following specifications in PART 1.2)	
M5.1 Bandwidth usage (In kbps)	
M5.2 Periodicity of communications (Average, in times per minute)	

PART 3.3 Sustained resources consumption

	gLite jobs	**Boinc client**	**XtremWeb client**	**Accesory applications**
CPUs needed:	*How many CPUs / clients are used on a typical sample scientific production*			
Job running time:	*How much time takes a typical job running on a single node*			
Memory usage:	*Amount of main memory required for a typical job running on a single node*			
Temporary files:	*How many auxiliary or temporary files the job generates on a single node*			
Disk space needed:	*Amount of free space required during an execution (i.e. for auxiliary files, temporary results, files to be part of input/output sandbox, etc)*			
Other resources:	*Describe here other possible resources the application may consume in a common execution workflow*			

PART 3.4 Integrity checks

Virus checking:	*Write here the results of checking the different application files with an antivirus program. Results may be virus free, virus erased or non-erasable virus*
Application stability:	*Stability refers to the probability for the application to fail during its average job running time. These fails may be due to segmentation faults, programmatic errors, file faults, etc. Value to put here should be calculated by performing at least 10 common executions (i.e. how many times the application failed divided by 10).*
Other integrity checks:	*Other issues that could affect a typical application execution considering how it was built.*

PART 3.5 Check points assessments

Check point space required:	*Disk space needed for the files that support the check point*
Check point stability:	*Probability for the application to fail due it is not able to recover from a check point. This value should be calculated after at least 10 check points*
EDGeS certification grant or denial:	(yes/no/comments)

PART 4 PLANNING FOR SUSTAINED SCIENTIFIC PRODUCTION	
PART 4.1. Characterization for a data challenge *(use values in PART 3 for estimation)*	
Goals of data challenge:	
Nature of input data:	
Nature of results:	
Average size of input data:	
Average size of results:	
Number of jobs needed to process input data:	
Hours-CPUs needed:	
Post-processing details:	
Other issues:	
PART 4.2. Evaluation scientific impact of the produced results	

7 Annex II. References and further information

1. EGEE NA4 requirements reference:

   http://egee-na4.ct.infn.it/requirements/list.php

2. BOINC possible platforms (it is a subset of all the platforms available for the BOINC plaftorm):

   windows_intelx86

   windows_x86_64

   i686-pc-linux-gnu

   x86_64-pc-linux-gnu

   powerpc-apple-darwin

   i686-apple-darwin