

Chapter 2

Introduction to the WS-PGRADE/gUSE Science Gateway Framework

Tibor Gottdank

Abstract WS-PGRADE/gUSE is a gateway framework that offers a set of high-level grid and cloud services by which interoperation between grids, clouds, and scientific user communities can be achieved. gUSE is also a workflow system that enables scientific communities to compose and execute a series of computational or data manipulation steps in a scientific application on Distributed Computing Infrastructures (DCIs). The current chapter summarizes the most important features of WS-PGRADE/gUSE.

2.1 Introduction

The Grid and Cloud User Support Environment (gUSE), also known as *WS-PGRADE (Web Service—Parallel Grid Run-time and Application Development Environment)*¹/*gUSE*, is a renowned European science gateway (SG) framework² that provides users with convenient and easy access to grid and cloud infrastructures as well as to data storage.

WS-PGRADE/gUSE provides a specific set of *enabling technologies* (Lovas 2013) as well as front-end and back-end services that together build a generic SG. An enabling technology provides the required software stack to develop SG frameworks and SG instances (that is, to provide a simple user interface that is tailored to the needs of a given scientific community). Typical examples of such enabling technologies are: web application containers (Tomcat, Glassfish, etc.), portal or web application frameworks (Liferay, Spring, etc.), database management

¹ WS-PGRADE is the graphical user interface of gUSE. See a detailed description of WS-PGRADE in Sect. 2.5.

² gUSE is the *Most Visited SG Framework* and *Most Visited Workflow System* by the EGI Applications Database (<https://appdb.egi.eu/>).

T. Gottdank (✉)

Laboratory of Parallel and Distributed Systems, Institute for Computer Science and Control,
Hungarian Academy of Sciences, Budapest, Hungary
e-mail: gottdank.tibor@sztaki.mta.hu

systems (MySQL, etc.), and workflow management systems (WS-PGRADE/gUSE, MOTEUR, etc.). With help of gUSE, scientific communities can compose and execute a series of computational or data manipulation steps in a scientific application on *Distributed Computing Infrastructures (DCIs)*.

This chapter introduces the key features, the architecture, the common user-level components, and the customization modes of the gUSE framework.

2.2 What gUSE Offers

gUSE provides a transparent, web-based interface to access distributed resources, extended by a powerful general-purpose workflow editor and enactment system, which can be used to compose scientific applications into data-flow based workflow structures (Balasko 2013a). gUSE is the only SG framework in Europe that offers a comprehensive and flexible workflow-oriented framework that enables the development, execution, and monitoring of scientific workflows. In addition, the nodes of these workflows can access a large variety of different DCIs, including clusters, grids, desktop grids, and clouds (Kacsuk 2012).

This SG framework can be used by *National Grid Initiatives (NGIs)* to support small user communities who cannot afford to develop their own customized SG. The gUSE framework also provides two Application Programming Interfaces (APIs), namely the *Application-Specific Module API* and the *Remote API*, to create application-specific SGs according to the needs of different user communities.

A relevant requirement in the development of gUSE was to enable the simultaneous handling of a very large number of jobs, even in the range of millions, without compromising the response time at the user interface. In order to achieve this level of concurrency, the workflow management back-end of gUSE is implemented based on the *web service* concept of *Service Oriented Architecture (SOA)* (Kacsuk 2012).

2.3 Key Features

Among many other features, the main five capabilities of gUSE are as follows:

1. gUSE is a *general-purpose* SG framework under which users can access more than twenty different DCIs³ via the DCI Bridge service, and six different data storage types (HTTP, HTTPS, GSIFTP, S3, SFTP, and SRM) via the Data Avenue service. Both DCI Bridge and Data Avenue were developed as part of the WS-PGRADE/gUSE service stack, but they can also be used as independent services enabling their use from other types of gateways and workflow systems.
2. WS-PGRADE/gUSE is a *workflow-oriented system*. It extends the *Directed Acyclic Graph (DAG)*-based workflow concept with advanced *parameter sweep*

³ The full list of supported DCIs is in Sect. 2.4.

(PS) features by special workflow nodes, condition-dependent workflow execution, and workflow embedding support. Moreover, gUSE extends the concrete workflow concept with the concepts of *abstract workflow*, *workflow instance*, and *template* (see the details in Sect. 2.5).

3. WS-PGRADE/gUSE supports the *development and execution of workflow-based applications*. Users of gUSE define their applications as workflows. They can share their applications among each other by *exporting* them to the internal Application Repository. Other users can import such applications and execute or modify them in their user space.
4. gUSE supports the fast development of SG instances by a *customization* technology. gUSE can serve different needs, according to the community requirements about the computational power, the complexity of the applications, and the specificity of the user interface to fit the community needs and to meet its terminology.
5. The most important design aspect of gUSE is *flexibility*. Flexibility of gUSE is expressed
 - *in exploiting parallelism*: gUSE enables parallel execution inside a workflow node as well as among workflow nodes. It is possible to use multiple instances of the same workflow with different data files. See details in Chap. 3.
 - *in the use of DCIs*: gUSE can access various DCIs: clusters, cluster grids, desktop grids, supercomputers, and clouds. See details in Chap. 4.
 - *in data storage access*: gUSE workflow nodes can access different data storage services in different DCIs via the *Data Avenue Blacktop* service. Therefore, the file transfer among various storages and workflow nodes can be handled automatically/transparently. See details in Chap. 5.
 - *in security management*: For secure authentication it is possible to use users' personal certificates or robot certificates. See details in Chap. 6.
 - *in cloud access*: A large set of different clouds (Amazon, OpenStack, OpenNebula, etc.) can be accessed by WS-PGRADE/gUSE either directly (see Chap. 4) or via the CloudBroker Platform (see Chap. 7).
 - *of supported gateway types*: gUSE supports different gateway types: general-purpose gateways for national grids (e.g., for Greek and Italian NGIs), general-purpose gateways for particular DCIs (e.g. EDGI gateway), general-purpose gateways for specific technologies (e.g. SHIWA gateway for workflow sharing and interoperation, see Chap. 9) and domain-specific science gateway instances (e.g. Swiss proteomics portal, MoSGrid gateway, Autodock gateway, Seismology gateway, and VisIVO, see Part 2 of the book).⁴ This aspect of WS-PGRADE/gUSE is described in detail in Sect. 2.6 and in Chap. 8.
 - *in use of workflow systems*: Users can access from the SHIWA Workflow Repository many workflows written in various workflow languages and use these workflows as embedded workflows inside WS-PGRADE workflow nodes. This feature of WS-PGRADE/gUSE gateways is described in detail in Chap. 9.

⁴ The domain specific science gateways are discussed in Part 2.

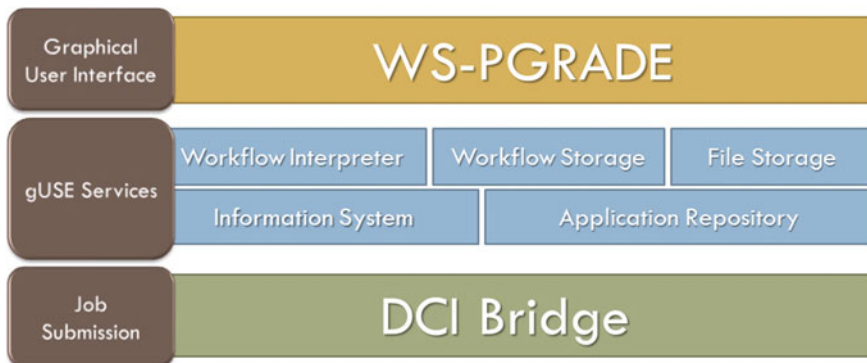


Fig. 2.1 The three-tier architecture of WS-PGRADE/gUSE

2.4 Architectural Overview

The main goal of designing the *multitier architecture* of WS-PGRADE/gUSE was to enable versatile access to many different kinds of DCIs and data storage by different kinds of user interfaces. This access can be technically performed through the *DCI Bridge* job submission service which is in the bottom within the gUSE architectural layers as shown in Fig. 2.1, and via the Data Avenue Blacktop service that is an independent service provided by SZTAKI (see Chap. 5).

DCI Bridge⁵ is a web service-based application providing standard access to various DCIs. It connects through its DCI plug-ins to the external DCI resources. When a user submits a workflow, its job components are submitted transparently into the various DCI systems via the DCI Bridge service using its standard *OGSA Basic Execution Service 1.0 (BES)* interface. As a result, the access protocol and all the technical details of the various DCI systems are totally hidden behind the BES interface. The job description language of BES is the standardized *Job Submission Description Language (JSDL)*. See further details on DCI Bridge in Chap. 4.

The DCIs supported by DCI Bridge are the followings:

- Clusters (PBS, LSF, MOAB, SGE)
- Grids (ARC, gLite, GT2, GT4, GT5, UNICORE)
- Supercomputers (e.g., via UNICORE)
- Desktop grids (BOINC)
- Clouds (via CloudBroker Platform, GAE, as well as EC2-based Cloud Access).

The *middle tier* of the gUSE architecture contains the high-level gUSE services. The *Workflow Storage* stores every piece of information that is needed to define a workflow (graph structure description, input files pointers, output files pointers,

⁵ DCI Bridge is discussed in Chap. 4.

executable code, and target DCI of workflow nodes) except the input files of the workflow. The local input files and the local output files created during workflow execution are stored in the *File Storage*. The *Workflow Interpreter* is responsible for the execution of workflows, which are stored in the *Workflow Storage*. The *Information System* holds information for users about workflows running and job status. Users of WS-PGRADE gateways work in isolated workspace, i.e., they see only their own workflows. In order to enable collaboration among the isolated users, the *Application Repository* stores the WS-PGRADE workflows in one of their five possible stages. (Physically all the five categories are stored as zip files.) The five categories of stored workflows are as follows, and the collaboration among the gateway users is possible via all these categories:

- Graph (or abstract workflow) containing information only on the graph structure of the workflow.
- Workflow (or concrete workflow) containing information both on the graph structure and on the configuration parameters (input files pointers, output files pointers, executable code and target DCI of workflow nodes).
- Template: a workflow containing information on every possible modifiable parameter of the workflow if they can be changed by the users or not. These play an important role in the automatic generation of executable workflows in the end-user mode of a WS-PGRADE/gUSE gateway (Sect. 2.6).
- Application is a ready-to-use workflow that contains all the embedded workflows, too. It means that all the information needed to execute this workflow application is stored in the corresponding zip file.
- Project is a workflow that is not completed yet and can be further developed by the person who uploaded it into the *Application Repository* or by another person (so collaborative workflow development among several workflow developers is supported in this way).

At the top of the three-tier structure, the *presentation tier* provides WS-PGRADE, the graphical user interface of the generic SG framework. All functionalities of the underlying services are exposed to the users by portlets residing in a *Liferay* portlet container, which is part of WS-PGRADE. This layer can be easily customized and extended according to the needs of the SG instances to be derived from gUSE. The next section introduces the essential user-level elements of WS-PGRADE.

2.5 Introduction to WS-PGRADE

Most users of gUSE come into contact with WS-PGRADE portal interface. The WS-PGRADE portal is a *Liferay* technology-based web portal of gUSE. It can be accessed via the major modern web browsers like Chrome, Firefox, etc.

2.5.1 User Roles

A member of a gUSE community can be a *power user* or an *end-user* in the WS-PGRADE portal environment. The *power user* or, in other words, *workflow developer* develops workflows for the *end-user scientists* (chemists, biologists, etc.). The power user understands the usage of the underlying DCI and is able to develop complex workflows. This activity requires editing, configuring, and running workflows in the underlying DCI as well as monitoring and testing their execution in the DCIs. In order to support the work of these power users, WS-PGRADE provides a GUI through which all the required activities of developing workflows are supported. When a workflow is developed for end-user scientists, it should be uploaded to a repository where scientists can download from and execute it. In order to support this interaction between power users and end-users, gUSE provides the earlier-mentioned Application Repository service in the *gUSE services-tier*, and power users can upload and publish their workflows for end-users via this repository.

The end-user scientists are generally not aware of the features of the underlying DCI nor of the structure of the workflows that realize the type of applications they have to run in the DCI(s). For these users, WS-PGRADE provides a simplified end-user GUI where the available functionalities are limited. Typically, end-user scientists can download workflows from the Application Repository, parameterize them, and execute them on the DCI(s) for which these workflows were configured to run. They can also monitor the progress of the running workflows via a simplified monitoring view. Any user of WS-PGRADE can login to the portal either as a power user or an end-user and according to this login she/he can see either the developer view or the end-user view of WS-PGRADE.

2.5.2 The Three-Phase Process of Workflow Development

The WS-PGRADE power users (workflow developers) typically perform a three-phase operation sequence (workflow edit, workflow configure, and workflow execution) as shown in Fig. 2.2. This step sequence covers the life-cycle of a workflow. The life-cycle of a WS-PGRADE workflow is the following:

1. During the *editing* phase, the user creates the abstract graph of the workflow.
2. In the workflow *configuring* phase the executable, the input/output files, and the target DCI of the workflow nodes representing the atomic execution units of the workflow are specified.
3. Finally, in the *submitting* phase, the workflow is submitted resulting in a workflow instance.

The following section gives a detailed description about what happens in the three phases.



Fig. 2.2 The three generic workflow development phases in WS-PGRADE

2.5.2.1 The Editing Phase: Creation of the Workflow Graph

The users construct their *abstract workflows* in this phase. Practically, it covers the workflow graph creation by the interactive, online workflow graphical designer and visualizer tool, the *Graph Editor* of WS-PGRADE (Fig. 2.3). The structure of WS-PGRADE workflows are represented by *directed acyclic graphs (DAGs)* as shown in Fig. 2.3. The DAG-based structure is the static skeleton of a workflow in WS-PGRADE. The nodes of the graph are abstract representations of jobs (or service calls). Each job must have a name, and job names are unique within a given workflow. The job communicates with other jobs of the workflow through *input* and *output ports*. An output port of a job connected to an input port of a different job is called a *channel*. Channels are *directed edges* of a graph, directed from the output ports toward the input ports. A single port must be either an input or an output port of a given job.

A job in a workflow may have *single* and *parametric input ports* (which should be specified in the next, the *configuring* phase of workflow development when *concrete workflow* is defined from abstract workflow). If a node has only single input ports, it is executed only once as a single instance processing the single inputs of every input ports. These nodes are called *normal nodes*. If a node has at least one parametric input port it is called *parametric node*. If a parametric node has one parametric input port, it will be executed in as many instances as the number of files that arrive on the parametric input port (Manual 2014).

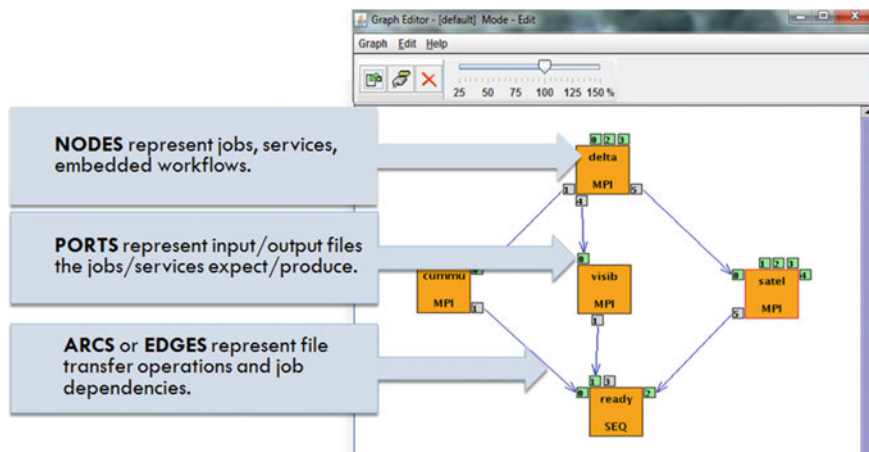


Fig. 2.3 Directed acyclic graph-based structure of a sample workflow in WS-PGRADE graph editor

A special but widely used workflow type also supported by WS-PGRADE/gUSE is the so-called *parameter sweep* or *parameter study* (PS) workflow, which is typically used for simulations where the same simulated workflow should be executed with many different input sets. DCIs are ideal for PS executions, and therefore their most frequent usage scenario is performing such PS workflows.

A typical PS-workflow contains three nodes (jobs) as shown by Fig. 2.4:

1. the *generator job* generates the necessary parameter set;
2. the *parametric job* (this is the *call* job in the example of Fig. 2.4) executes a specific application in as many instances as there were outputs generated by the generator job; and,
3. the *collector job* collects and processes the results of the parametric job (for example, by creating statistics based on the results of the different executions).

If the output port of a generator job is connected to the parametric input port of a parametric job, then this parametric job will be executed for every file generated by the generator job.

Another useful characteristic of WS-PGRADE workflows is the possibility to *embed workflows* into workflow nodes. Thus, instead of running, for example, an executable inside a workflow node, another WS-PGRADE workflow may run inside the parent workflow node. To embed workflows, users need to apply workflows created from the so-called *templates*. A template is a generic workflow where some configuration parameters are fixed. It can be used to serve as a base of creating the definitions of new workflows.⁶

⁶ The gUSE workflow concept is discussed in detail in Chap. 3.



Fig. 2.4 A graph of a sample parameter study (PS) workflow in WS-PGRADE

2.5.2.2 The Configuring Phase: Setting of Workflow Nodes

The abstract workflow created in the first (editing) phase represents only the structure (graph) of a workflow, but the semantics of the nodes are not defined yet. The abstract workflow can be used to generate various *concrete workflows* in the *configuring phase*.

The concrete workflows are derived from abstract workflows by exactly specifying the workflow nodes and the DCIs where the various nodes should be executed. The concrete workflows generated from a certain graph can be different concerning the semantics of the workflow nodes, and the input and output files associated with ports.

The node configuration includes:

- algorithm configuration: determines the functionality of a node;
- target DCI resource configuration: determines where this activity will be executed;
- port configuration: determines what input data the job needs and how the result (s) will be forwarded to the user or to other jobs as inputs.

A typical node configuration in WS-PGRADE contains the following generic functions (see also Fig. 2.5):

1. The properties of the node can be defined by the *Job executable configuration* function (see Fig. 2.5).
2. The *Port configuration* function helps users to define the file arguments of the required calculation. Each port configuration entry belonging to the current node is listed and can be made visible.
3. By using the *JDL/RSL function* users can add or remove ads. Removing an ad happens by the association of an empty string to the selected key.
4. The state and history of the node configuration can be checked by the *Job configuration history* function.
5. The error free state of the configuration can be checked by the *Info* function.

Users can also define *breakpoints* for every node during workflow configuration in order to control the runtime execution of the created job instances. All instances of a node marked by a breakpoint can be tracked in the job submission process (Manual 2014).

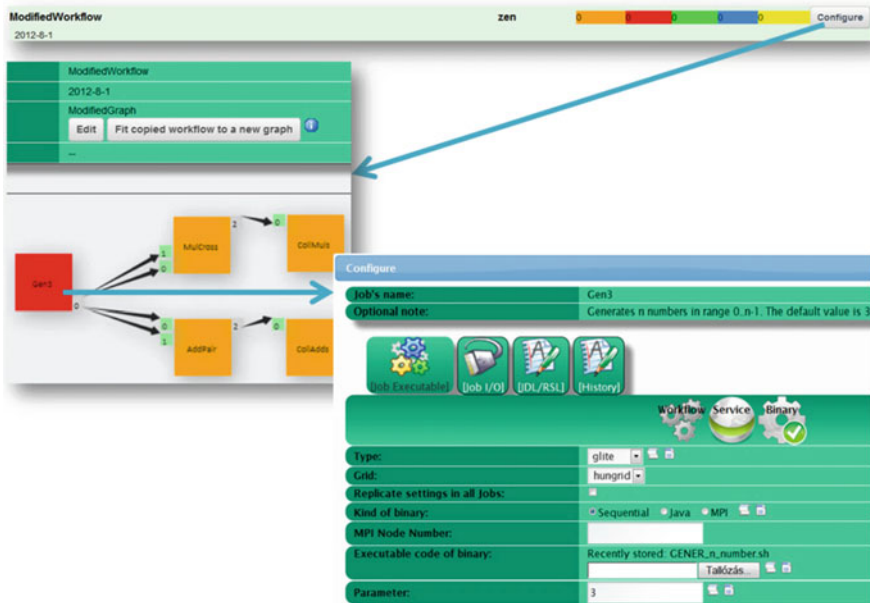


Fig. 2.5 The main elements of the configuration phase in WS-PGRADE

2.5.2.3 The Submitting Phase: Workflow Execution

After all the properties of the workflow have been set, it can be submitted, resulting in an *instance of the workflow*. A concrete workflow can be submitted several times (for example, in case of performance measurements), and every submission results in a new instance of the same concrete workflow.

The execution of a workflow instance is data driven. The order of execution is forced by the graph structure: a node is activated (the associated job is submitted or the associated service is called) when the required input data elements (usually a file, or a set of files) become available at every input port of the node. This node execution is represented as the instance of the created job or service call. One node can be activated with several input sets, and each activation results in a new job or service call instance. The job or service call instances also contain status information and in the case of successful termination the results of the calculation are represented in the form of data entities associated to the output ports of the corresponding node.

A typical submission scenario contains three main parts (Fig. 2.6): starting and monitoring the submission as well as obtaining the submission result:

1. A workflow submission can be started by clicking on the *Submit* button.
2. Monitoring and observing the submission: the progress of workflow instance submission can be checked by the *Details* function. The result is the list of the

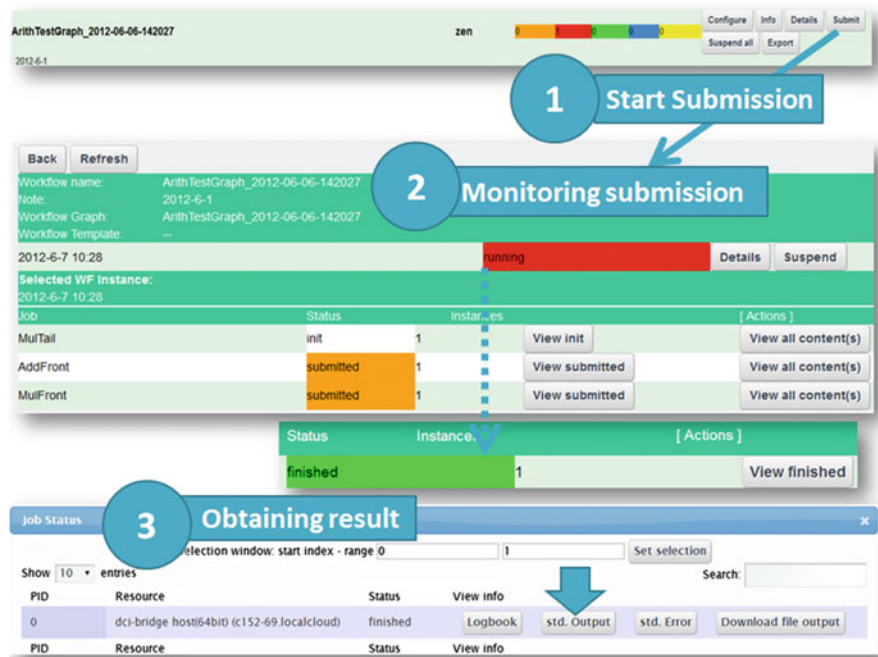


Fig. 2.6 The three main parts of the submission phase in WS-PGRADE

- component nodes with their status information. Users may suspend or abort the execution of a running instance. A previously suspended workflow instance can be resumed at any time.
3. Once the status of the component node turns to “finished”, the result files can be accessed.

Users can interact with the workflow instance in runtime at the earlier defined breakpoints. The workflow interpreter detects the defined breakpoints during the elaboration of the job instances and suspends the execution. It then notifies the user about the temporary suspension of the job instance. The user may decide separately about the future of each suspended job instance by enabling or prohibiting its progress. Moreover, the user may change the configuration of the non-executed part of the workflow after a global *Suspend* command issued in a “break pointed” state.

In order to submit workflows it is necessary to authenticate with the corresponding certificate. Grid and cloud authentication require username and password or user proxies.⁷

⁷ The authentication mechanism of gUSE is discussed in detail in Chap. 6.

A special authentication solution is the use of *robot certification*. Instead of identifying users, the robot certification identifies trusted applications which can run by workflows from WS-PGRADE. The main advantage of this certification is to run workflows without any direct authentication data upload; thus end-users just need to import from the Application Repository the previously uploaded robot certificate-aware workflows for submissions.

WS-PGRADE also provides *statistical* information about job executions from the viewpoint of portal, DCI, user, and workflow. This solution is helpful for administrators in resource settings, and for users in middleware, resource selection and in job execution time estimation.

2.6 Customizing WS-PGRADE/gUSE

In order to create application-specific gateways from WS-PGRADE/gUSE four *customization methods* have been developed depending on the user community demands (see also Fig. 2.7):

Mode 1: Power user/end-user mode: This customization mode contains two variants:

1. In *power user mode* the power users mentioned in Sect. 2.5.1 create and share applications (workflows), and the end-users typically import and execute them. Using this solution, development of science gateways technically means the development of new workflow applications that specialize the original generic WS-PGRADE/gUSE gateway toward the needs of the user community.

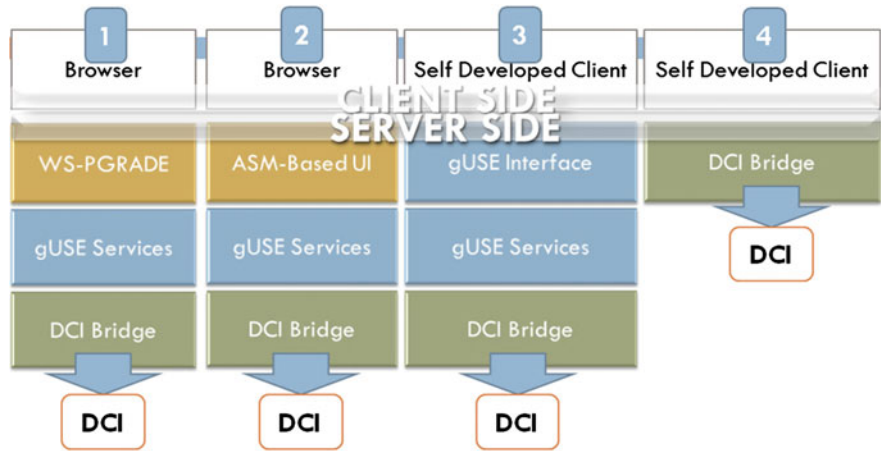


Fig. 2.7 Customization modes of gUSE

2. If the community needs just a generic but simple interface, gUSE can be tailored automatically for each application by setting it to the *end-user mode*. In this solution users are not allowed to create new workflows, but they have access to the local Application Repository to import already developed workflows. These workflows should be stored as templates, and after importing such a shared workflow, the generic end-user interface automatically and dynamically generates a parameterization interface based on the imported workflow's different properties that were set with read and write permissions when the workflow template was created.

Mode 2: Customization by ASM: In order to support the development of application-specific UI, gUSE provides the *Application-Specific Module (ASM)* API by which customized portlets can easily and quickly be created. Once this has happened, scientists who require such customization can run their workflow applications on various DCIs via the application-specific portlets developed by means of the ASM API. In this case the original WS-PGRADE UI is replaced or extended with the customized application-specific UI, and this new UI can directly access the gUSE services via the ASM API.

Mode 3: Customization by Remote API: gUSE can be extended to let the community use its own user interface to send a complete WS-PGRADE/gUSE workflow via HTTP protocol without using the original WS-PGRADE interface. Technically this extension is a new web service component called *Remote API*, which should be deployed on the WS-PGRADE/gUSE portal server, and be registered among the general gUSE components. In this case the existing community user interface can access the gUSE services via a direct API and run WS-PGRADE workflows directly via this API.

Mode 4: Own interface for workflow submission: It is also possible to move the *DCI Bridge* service from the gUSE tiers and make this service directly accessible via the standard BES job submission interface. In this case WS-PGRADE/gUSE is not used; only the stand-alone version of the DCI Bridge that can submit jobs to the connected DCIs and the existing community gateway or workflow system should be enabled to submit jobs via the standard BES interface of DCI Bridge.

2.7 Conclusions

The main objective of WS-PGRADE/gUSE was to develop a gateway framework that enables a large set of different communities to build their gateway instances by simply customizing the framework according to their needs. In order to support many communities, flexibility was a major design goal. This flexibility enables access to all the major DCIs and data storages with all the available authentication methods and enables parallel/distributed execution of workflows at several parallelization levels. Subsequent chapters explain in detail those features that contribute to this high degree of flexibility in WS-PGRADE/gUSE.

The success of these design principles resulted in WS-PGRADE/gUSE becoming one of the leading science gateway frameworks in Europe. WS-PGRADE/gUSE is actively used by more than 30 science communities and several commercial companies. WS-PGRADE/gUSE is open-source software based on an Apache license and can be freely downloaded from SourceForge (SF 2014). Detailed WS-PGRADE/gUSE documentation has also been published on SourceForge.

Science Gateways for Distributed Computing

Infrastructures

Development Framework and Exploitation by Scientific

User Communities

Kacsuk, P. (Ed.)

2014, XIX, 301 p. 94 illus., 84 illus. in color., Hardcover

ISBN: 978-3-319-11267-1