8 Developing Science Gateways at Various Levels of Granularity Using WS-PGRADE/gUSE

Tamás Kiss, Gábor Terstyánszky, Péter Borsody, Péter Kacsuk, and Ákos Balaskó

Abstract. Science gateways can provide access to distributed computing resources and applications at very different levels of granularity. Some gateways do not even hide the details of the underlying infrastructure, while on the other hand some provide completely customized high-level interfaces to end-users. In this chapter the different granularity levels at which science gateways can be developed with WS-PGRADE/gUSE are analysed. The differences between these various granularity levels are also illustrated via the example of a molecular docking gateway and its four different implementations.

8.1 Introduction

Science gateways, such as gateways built using the WS-PGRADE/gUSE framework [Kacsuk/2012], have the potential to offer transparent and user-friendly access to a wide variety of distributed computing resources. These tools hide the complexity of the underlying infrastructure from the scientist end-users and let them concentrate on their scientific research problem instead of requiring a steep and sometimes impossible learning curve in complex computing paradigms.

Many web and desktop-based tools have been developed in the past few years that have been labelled as science gateways. However, close examination of these tools reveals that the level of granularity at which end-users can access the applications is rather varied. There are solutions which do not aim to hide the details of the original command line interface, and simply provide web-based access to the underlying distributed computing infrastructure. On the other extreme, there are custom-built portals supporting a single or a small family of applications and providing highly intuitive graphical user interfaces incorporating visualization tools, for example. Science gateways can be developed at various levels of granularity, significantly influencing how and by which category of users these tools can be utilized.

Part of the research carried out in the SCI-BUS European project was to investigate the level of granularity of science gateways that a particular user community requires. As the WS-PGRADE/gUSE framework supports the development of sci-

ence gateways at different levels of granularity, once the required level was identified, SCI-BUS supported the development of various end-user gateways in diverse disciplines and at the required granularity levels. This chapter gives an overview of the various granularity levels offered by WS-PGRADE/gUSE, and illustrates via the example of a molecular docking gateway [Kiss/2010] the advantages and disadvantages of the different levels and approaches.

8.2 Granularity Levels Supported by WS-PGRADE/gUSE

There are two main approaches when developing science gateways: developing from scratch or adapting and customizing an existing gateway framework.

When developed from scratch, software engineers create a custom gateway solution for a particular user community with no or minimal reuse of existing components. This can result in a highly customized and specialized gateway. On the other hand, development typically requires significant time, effort, and resources. Also, these gateways are typically highly specialized, making it hard to reuse and extend them for additional user scenarios. Changing the level of granularity in case of these custom gateways is not supported since any modification of the developed solution requires major software engineering effort.

The second approach to building a science gateway is to adapt and customize a generic science gateway framework. These frameworks (for example, WS-PGRADE/gUSE [Kacsuk/2012], the Catania Science Gateway Framework [Barbera/2010], or the HubZero framework [McLennan/2010]) provide readily available services, significantly decreasing development time and effort. Additionally, if these frameworks were extended with customization methodologies then highly specific gateways could also be built on top of them. Therefore, gateways at different levels of granularity can be created requiring various levels of effort from the developers, and providing various levels of customization for end-users.

WS-PGRADE/gUSE supports the development of science gateways at four different levels of granularity. These levels are:

1. Out of the box: Deploy the generic WS-PGRADE/gUSE framework and provide it “out of the box” for end-users.
2. Predeveloped workflows: Deploy the generic WS-PGRADE/gUSE framework and also predevelop and provide the necessary workflows to be executed by the end-users.
3. End-user view: Use the end-user view of WS-PGRADE/gUSE to hide the complexity of workflow creation and parameterisation from the end-user.
4. Custom user interface: Develop a completely customized gateway using the Application-Specific Module (ASM) API or the Remote API [Balasko/2013] of the WS-PGRADE/gUSE framework.

The effort required from application developers and system administrators to create and set up a gateway increases as we move from option 1 toward option 4,

with the lowest level of developer/system administrator effort being required for the first option. On the other hand, the level of effort and expertise required from the end-user decreases significantly as we move toward option 4. The remaining part of this chapter analyses and describes the different options via the example of a molecular docking gateway.

8.3 A Gateway for Molecular Docking

Molecular docking simulation programs have significant potential to contribute to a wide area of molecular and biomedical research, including drug design, environmental studies, or psychology. AutoDock [Morris/1998] is one example of a program which allows in silico modeling of intermolecular interactions. Emerging literature shows that AutoDock can be successfully utilized in research strategies for the study of molecular interactions in cancer [Ali/2007] and for designing drug inhibitors for HIV [Teixeira/2007], for example. AutoDock is a suite of automated docking tools. It is designed to predict how small molecules, such as substrates or drug candidates, bind to a receptor of known 3D structure. AutoDock currently comprises two discrete generations of software: AutoDock 4 and AutoDock Vina. The latter provides several enhancements over the former, increasing average simulation accuracy while also being up to two orders of magnitude faster.

Autodock Vina is particularly useful for virtual screening, whereby a large set of ligands can be compared for docking suitability with a single receptor. In this instance parallelism is achieved by first breaking the set of all ligands into equal sized disjoint subsets. Each computing job then uses a different subset as an input. The ligands in each subset are simulated/docked sequentially on the computing node using the single receptor, while a postprocessing stage can be used to compare the results from all computing jobs.

AutoDock 4 is typically used to accurately model the molecular docking of a single ligand to a single receptor. In this instance the process is composed of three discrete stages. First, a low complexity sequential preprocessing stage defines a random starting location in 3D space for both the ligand and receptor. This is achieved using a tool within AutoDockTools (ADT) called AutoGrid. The second stage can comprise many parallel jobs, each receiving a copy of the ligand and receptor starting locations which form the input to a genetic algorithm. The algorithm acts to randomly rotate/reposition the ligand and then determine likely docking/binding sites based upon energy levels which are calculated from the original starting locations. This process can be considered a parameter sweep, where the varied input parameter is the initial random rotation of the ligand. Finally, a single low complexity sequential post-processing stage can be used to identify the most likely binding site by comparing energies from all jobs of the preceding stage.

Above described scenarios are supported in four different granularity level implementations, using the WS-PGRADE/gUSE framework described next.

8.4 Granularity Level 1: Out of the Box

Granularity level 1 simply means installing the generic WS-PGRADE/gUSE framework by a system administrator (including the connection and configuration of the gateway to suitable distributed computing resources), and providing access for potential end-users in the form of power-user accounts. After getting access to the gateway, it is the task of the end-users to design and develop the necessary workflow applications to run the docking scenarios.

WS-PGRADE/gUSE provides an intuitive and high-level user interface for this scenario that supports complex workflow development without needing to deal with low-level details, such as job submission mechanisms, job monitoring, file transfers, etc. There is no need to write complex programs or to use command line interfaces and understand the low-level details of various DCIs. On the other hand, this scenario is still well above the expertise of most bioscientists and requires a specific and rather long training period. As the target end-users are scientists, they typically do not wish to be diverted from their research with such extra activities and requirements.

The typical tasks needed to be carried out by the user in this scenario are illustrated on Fig. 8.1. The graph editor is required to design the workflow graph, and then the concrete workflow needs to be configured, including the upload of executables, definition of command line parameters, selection of DCIs, definition of ports, etc. Figure 8.1 shows a three-job AutoDock Vina workflow and its configuration, as a representative example. The first job of this workflow is the Generator that runs only once and prepares the necessary input files for the simulation. The second job is the actual docking simulation application, in this case AutoDock Vina. This job runs as many times there are input ligands uploaded to the workflow. In each run, a different ligand is docked on the target receptor molecule. The right-hand side of the figure shows the configuration of this job that is mapped to a BOINC-based desktop grid resource. Using the desktop grid the parameter sweep can be effectively parallelized. Finally, the third collector job analyses the results of all docking simulations and selects the required number of best docking results.

While significant effort and expertise is required from the end-users, the tasks and responsibilities of gateway operators and application developers are kept at the minimum in this scenario. If in the target community are experienced workflow and application developers, then a suitable gateway can be set up very quickly by installing the framework as it comes “out of the box”.

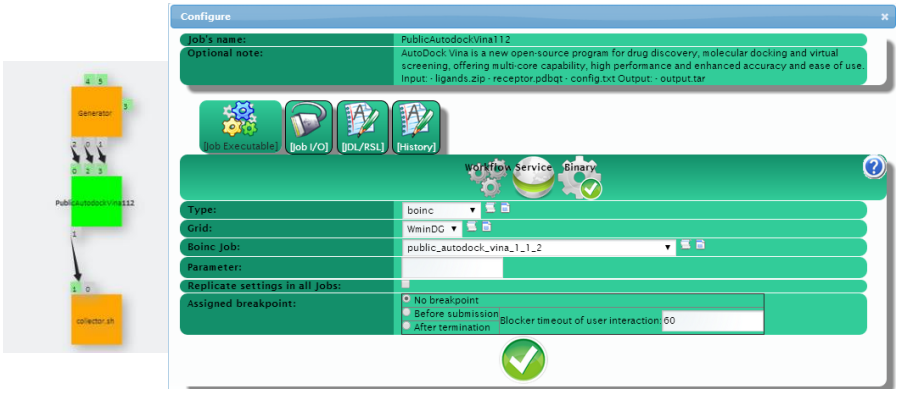

Fig. 8.1 Graph creation and concrete workflow configuration are done by end-user at granularity level 1

8.5 Granularity Level 2: Predeveloped Workflows

Granularity level 2 is an extension of the first scenario when not only is the gateway installed for the end-users, but the necessary workflows are also predeveloped and exported to a suitable workflow repository. WS-PGRADE offers access to both an internal workflow repository and the external SHIWA repository (see chapter 9 for details).

In this scenario, end-users need to import the predeveloped workflows to their accounts, parameterize and execute them. Users also have the possibility to customize, modify, or even extend the workflows if they wish. End-users do not have to design or implement the workflows as these are precreated by workflow development experts. However, users still need to be familiar with some concepts of the gateway framework, e.g., they do need to understand the workflow concept and should have some awareness of distributed computing infrastructures. We recommend that at least an introductory gateway course is recommended to be undertaken. Figure 8.2 illustrates that the user only needs to configure the workflow (typically providing input files only) at this granularity level, but no workflow creation is required. However, the configuration interface (as it facilitates full workflow configuration) is rather complex, with different panels for jobs and ports.

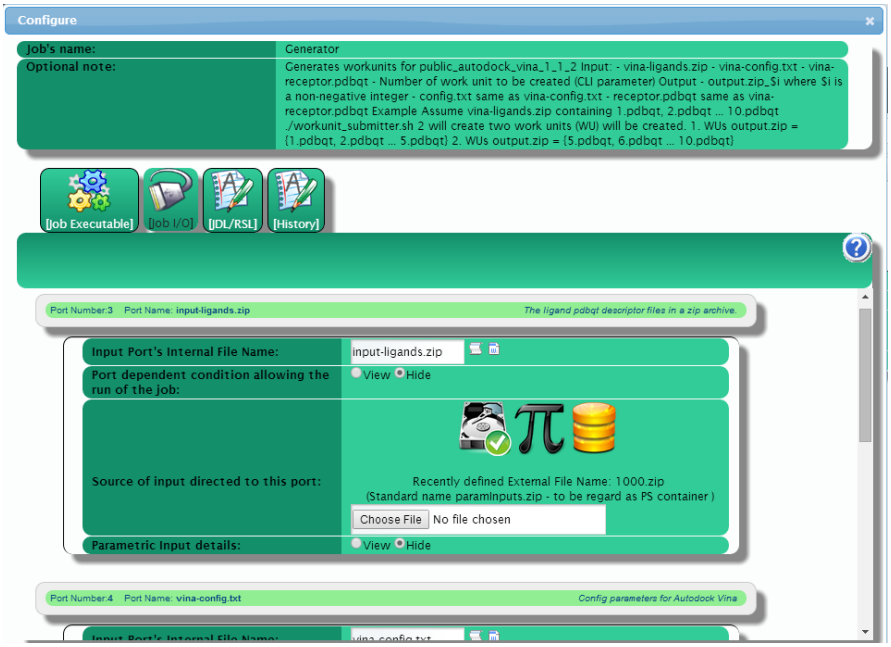

Fig. 8.2 Only concrete workflow configuration is carried out by the end-user at granularity level 2

On the provider side, this scenario, besides system administrators who deploy the gateway, also requires specialized workflow application developers who design, implement and maintain the workflows for the users.

The common feature of the first two granularity levels is that no custom interfaces are created for the applications. Only the standard WS-PGRADE/gUSE submission interfaces are used in the form that these features were described in the previous chapters of this book.

8.6 Granularity Level 3: End-User View

Granularity level 3 utilizes the end-user view of WS-PGRADE/gUSE. This view enables the generation of customized user interfaces without writing additional code.

In this scenario, system administrators deploy the gateway, workflow developers develop the necessary workflows, and then the end-user interfaces are automatically created by WS-PGRADE/gUSE, based on the workflow settings provided by the developers. The framework enables workflow developers to define templates on top of concrete workflows (by differentiating between fixed and open parameters from the end-user's point of view), and how to create applications from these templates. Once these applications are exported to the application

repository, in end-user view the gateway presents these applications as simple web-forms for parameterization by the scientist.

This view completely hides the complex details of workflows and DCIs from the end-user. On the other hand, creating an application suitable for end-user view is only a few more clicks when compared to granularity level 2. As a drawback, the automatically generated forms are relatively rigid, and do not allow much customization. Also, the user still needs to import the workflows from the internal repository to the individual account. Altogether, the end-user view provides a viable solution for quickly developing customized science gateways without any programming or code development. Therefore, this option is suitable to develop end-user oriented gateways.

Figure 8.3 illustrates the user interface of the previously introduced molecular docking experiments using the end-user view. The major difference between the this view and the concrete workflow configuration windows utilized at Granularity Levels 1 and 2 is that in the end-user view scientist end-users are restricted to provide and upload input files and additional other parameters that are required to run the application. More specifically, users cannot edit the workflow, specify the DCIs these workflows will be executed on, or upload executables. These characteristics of the application are all pre-defined and fixed. However, users can upload and define all input files and parameters in a more user-friendly way. Command line parameters are separated and their long names can be provided. Files are uploaded from the same form. When such a simple form is compared to the complex configuration interface demonstrated in Fig. 8.2, it well shows why such a simplified input form is beneficial for end-users.

In the random docking example shown in Fig. 8.3, bioscientists can upload the necessary input molecule files (receptor and ligand files in PDB format), and also the grid and docking configuration files (docking.gpf and docking.dpf). They also specify some other parameters such as the number of work units to be created (specifying the number of random docking experiments), and the required number of best (lowest energy level) solutions that they wish to receive back.

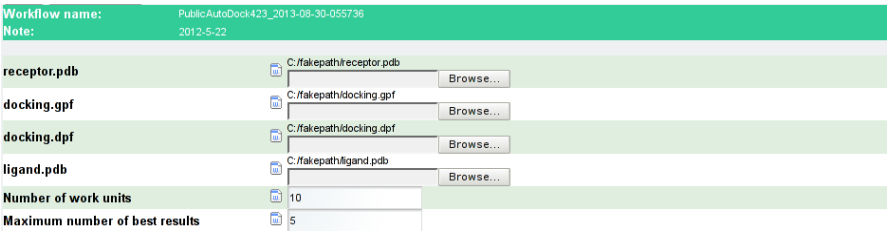

Fig. 8.3 User interface for molecular docking based on the end-user view

8.7 Granularity Level 4: Custom User Interface

WS-PGRADE/gUSE offers specific APIs that allow the connection of existing user interfaces to various DCIs via predefined workflows (remote API), or the development of new custom portlets within the WS-PGRADE/gUSE framework (ASM API). Granularity level 4 represents custom gateways that were developed using these APIs that provide access to low level WS-PGRADE/gUSE functionalities. Using the ASM or the remote API, customized gateways that incorporate visualization tools and highly specific user interfaces can be built with reasonable development effort (typically 2–4 weeks development time for a customized gateway). In this section, first we provide short overviews of the ASM and remote APIs, followed by the example of the custom user interface developed for the molecular docking gateway.

8.7.1 Application-Specific Module API

The aim of the application-specific module (ASM) API is to hide complex workflows from end-user scientists and provide for them the most convenient, application-oriented interface via a domain-specific portlet. In order to achieve this goal, the ASM API provides access to a set of well parameterized Java methods to utilize low-level gUSE services. This component enables passing information between customized portlets and the gateway framework without requiring complex algorithms or web-service calls, thus significantly simplifying the development of such custom interfaces. Portlets developed with the help of the ASM API enable end-users to upload new input files, specify parameters, or visualize results. However, end-users cannot edit or modify workflow structures, or define executables behind workflow nodes.

Functionalities of the ASM API fall into three different categories:

1. Methods covering application management issues and getting information about workflows stored in the application repository, such as getting a list of application developers and applications according to a specified developer ID, importing an application to local user space, and getting a list of applications that have already been imported.
2. Methods that can be used for input/output data manipulation, such as uploading a file to a specified port, setting a file that currently exists on the portal server as input for a job, setting command-line parameters for a job, and fetching outputs of calculations.
3. Methods for handling user activities during execution such as workflow submission; getting workflow execution status in simple or in detailed format; and for aborting, rescuing, or removing a workflow.

As portlets in general are deployed in portlet containers that supervise the most common user activities and manage user sessions, ASM does not have to

provide any security features with the exception of issues related to the underlying infrastructures and complex systems. These security features are provided via the certificate management capabilities of gUSE (see Chap. 6).

The ASM API also helps to separate the work of the workflow developers and portlet developers. Workflow developers create the ready-to-use workflows and publish them in the gUSE internal application repository. Portlet developers can develop the customized, application-oriented portlets that use and hide these workflows taken from the gUSE internal application repository. In this way, end-user scientists do not have to be even aware that behind their customized portlet a complex workflow is running on several DCIs. The ASM API is intensively used by almost every user community who builds customized science gateways based on WS-PGRADE/gUSE. Good examples can be found in Chaps 10–14 and Chap. 17. Detailed description of the actual usage of the ASM API's features can be found in the "ASM Developer Guide" document.[1]

8.7.2 Remote API

The Remote API facilitates the execution of complex workflows from an existing graphical user interface, circumventing the original WS-PGRADE GUI. If a community already has a science gateway (called the primary gateway) built with the help of a different technology and can access only one particular DCI, and there is no workflow development and execution facility built-in, then such a primary gateway can be extended with a secondary WS-PGRADE/gUSE gateway that is used for two purposes:

1. To develop the workflows needed for the community and to provide access to all those new DCIs needed for the community.
2. To execute the workflows developed on the secondary WS-PGRADE/gUSE gateway but that are launched from the existing primary community gateway.

Technically the remote API is a component of the WS-PGRADE web application enabling such remote workflow execution on the secondary WS-PGRADE/gUSE gateway. With the help of the Remote API client, existing gateways and other kind of programming environments can be extended with the rich capabilities of executing WS-PGRADE/gUSE workflows and utilizing various DCIs.

Remote API is implemented as a simple servlet that is available on every installed WS-PGRADE/gUSE gateway. This servlet can be switched on if the WS-PGRADE/gUSE gateway is intended to be used as a secondary gateway. Notice

[1] Latest release at the time of writing this book: http://sourceforge.net/projects/asmsp.guse.p/files/3.4.10/ASM_Developer_Guide_v3.4.10.pdf/download; generic format of URL: http://sourceforge.net/projects/guse/files/<VersionNumber>/Documentation/ASM_Developer_Guide_v3.4.10.pdf/download

that the same gateway can be used both as primary and secondary gateway if the remote API servlet is switched on. This feature is exploited in the SHIWA Simulation Platform gateway (Chap. 9).

The switched on Remote API feature on the secondary WS-PGRADE/gUSE gateway enables the execution of previously created WS-PGRADE workflows only. Moreover, all workflows must be available on the primary community gateway beforehand. The API call on the primary gateway automatically creates a new temporary portal user on whose behalf the workflows are executed. Once the user downloads the output of the calculation, the entire temporary user environment is cleared up on the primary gateway.

Each call of the Remote API servlet can go with different parameterization. The API includes methods for workflow submission, querying the status of a submitted workflow (e.g., submitted, running, finished, error), suspending, rescuing and aborting workflows, as well as downloading the results of a workflow in a zip file that contains both the output and the log files. The remote API servlet is intensively used by several user communities in their gateways as described in Chaps 13, 15, and 19. Detailed description of the actual usage of the remote API feature can be found in the "Remote Access Configuration Manual" document.[2]

8.7.3 Molecular Docking Gateway with Custom User Interfaces

At Granularity Level 4 the generic framework can be fully customized using the previously described high level APIs to provide an attractive and rich user environment for task execution, monitoring and visualization. Although gateway development effort is required to provide this customized solution, the utilization of the generic framework and its APIs significantly reduces the development time and effort that is needed. When compared to developing the same gateway from scratch, relying on the generic gateway framework typically results in a scalable and extendable solution with less than 10% of the effort required otherwise.

At Granularity Level 4 the task of system administrators and application developers is rather complex as they have to develop the fully customized gateway (including gateway deployment, workflow development, and user interface development or connection using the ASM or Remote APIs). However, these tasks are significantly supported by the gateway framework. On the other hand, end-users can fully concentrate on their research tasks and require no training to use or understand the gateway.

Figure 8.4 illustrates the custom user interface of the molecular docking gateway that was developed using the ASM API. The customized gateway enables bi-

[2] Latest release at the time of writing this book: http://sourceforge.net/projects/guse/files/3.6.7/Documentation/RemoteAPI_Config_Manual.pdf /download; generic format of URL: http://sourceforge.net/projects/guse/files/<Version Number>/Documentation/RemoteAPI_Config_Manual.pdf/download

oscientists to easily parameterize, submit, and monitor docking experiments, and they can also visualize input and output molecules.

The upper panel of the figure shows the configuration phase, where input parameters and files of the docking experiment are specified and uploaded. The interface provides detailed description of the required file or parameter and performs basic checks regarding the provided values. The lower panel shows the results of the simulations. The generated output and log files can be easily downloaded by the user for further analysis. Moreover, the molecules can be visualized providing, a more intuitive way for bioscientists to analyze the results.

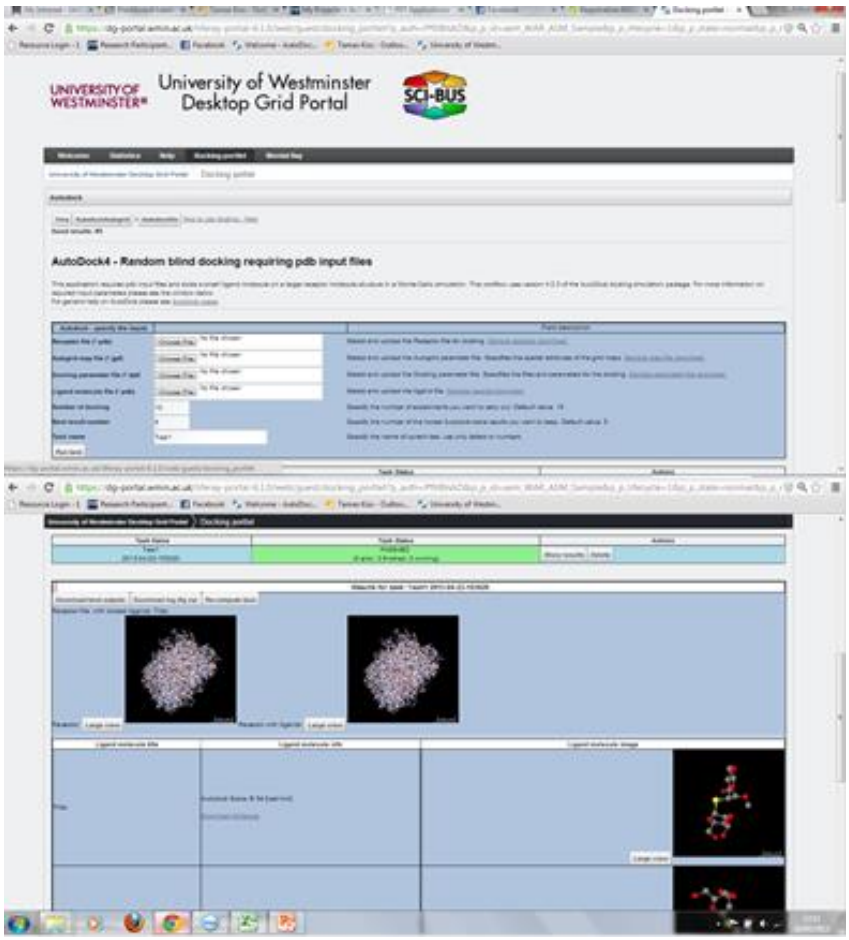

Fig. 8.4 Custom user interface for the molecular docking gateway

8.8 Summary and Conclusions

This chapter explained how gateways at different levels of granularity can be developed utilizing the WS-PGRADE/gUSE framework. Required granularity is one of the first questions that needs to be addressed when developing a science gateway. Selecting the right level of granularity is crucial for providing a usable tool for the targeted community without engaging in unnecessary and time-consuming development efforts while still providing end-users with suitable solutions that best fit their needs and requirements. As WS-PGRADE/gUSE supports gateway development at various granularity levels, it fits well to a very large number of scenarios and enables gateway developers to select and apply the best approach.