9 Sharing Science Gateway Artefacts through Repositories

Gábor Terstyánszky, Edward Michniak, Tamás Kiss, and Ákos Balaskó

Abstract. Researchers want to run scientific experiments focusing on their disciplines; they do not necessarily want to know how and where the experiments are executed. Science gateways hide details by coordinating the execution of experiments using different infrastructures and workflow systems. ER-flow/SHIWA and the SCI-BUS project developed repositories to share artefacts such as applications, portlets, workflows, etc. inside and among research communities. Sharing artefacts in repositories enables gateway developers to reuse them when building a new gateway and/or creating a new application.

9.1 Introduction

Researchers simply want to run scientific experiments focusing on their disciplines. Science gateways hide details how and where experiments are run by coordinating the execution of experiments using different infrastructures and workflow systems. Using a science gateway framework significantly speeds up the gateway development process when compared to development from scratch. Most gateway frameworks provide such common services as authentication, job/workflow submission to various DCIs, monitoring and information system capabilities, or execution statistics, just to mention a few. These services are provided by the framework itself and are typically tightly coupled with the underlying technology.

Different artefacts, for example, applications, portlets, workflows, etc., that could be specific to a particular application domain or science gateway could also be efficiently shared and reused between multiple developer and research communities. Sharing these various artefacts in repositories, for example, in application, workflow, or portlet repositories enables gateway developers to reuse these existing artefacts when building a new gateway and/or creating a new application. Moreover, these building blocks can, in many cases, be utilized by developers using different gateway frameworks, and in this way facilitate an even wider collaboration between communities. Sharing and reuse of these artefacts via repositories significantly shortens the development. There are two major scenarios in sharing artefacts:

- cooperation inside a particular research disciplinary, and

- inter- or multidisciplinary research cooperation among different research communities.

The WS-PGRADE/gUSE framework [Kacsuk/2012] contains internal repositories that allow researchers of one community, who are registered with a gUSE gateway, to publish and share artefacts (Sect. 9.3). To support inter- or multidisciplinary research cooperation, the gUSE framework can be connected to external repositories containing portlets and workflows (Sects. 9.4 and 9.5). Developers can upload and publish porlets and workflows in these repositories. Both developers and researchers of different communities can access these repositories to find and download those portlets and workflows they want to use. To further improve the user experience, the framework incorporates a portal and a workflow system to make transparent the execution and infrastructure details. The portal provides a power-user view for developers to create and test experiments, and an end-user view for researchers to run these experiments. To further improve the researchers' experience, the framework offers the application-specific module (ASM) concept, which enables developers to create customized portlets that are tailored to researchers' requirements to run particular experiments.

9.2 SCI-BUS Community Gateways

The SCI-BUS gateway's environment is shown in Fig. 9.1. The key component is the customized community science gateway. The customization incorporates configuring the DCI Bridge to enable access to the infrastructure(s) required by the community and extending the gateway with ASM portlets [ASM] to run the scientific experiments in a community-specific way. SCI-BUS communities deploy, customize, and operate the community gateways. To support sharing applications, files, portlets and workflows in the development and execution phase, the environment provides internal repositories such as the application repository, file storage, workflow storage and access to external repositories such as the SCI-BUS portlet repository and the SHIWA Workflow Repository. The SHIWA workflow repository allows for management of workflows developed by communities. Workflow developers can upload workflows either automatically (export operation) through the gateway, or manually through the repository GUI. Workflow developers can search the workflow repository, and then select and download the workflows they want to incorporate in the customized ASM portlets. They can publish the portlets manually in the SCI-BUS portlet repository. Community gateway system administrators can manually download and add these customized ASM portlets to their community gateways.

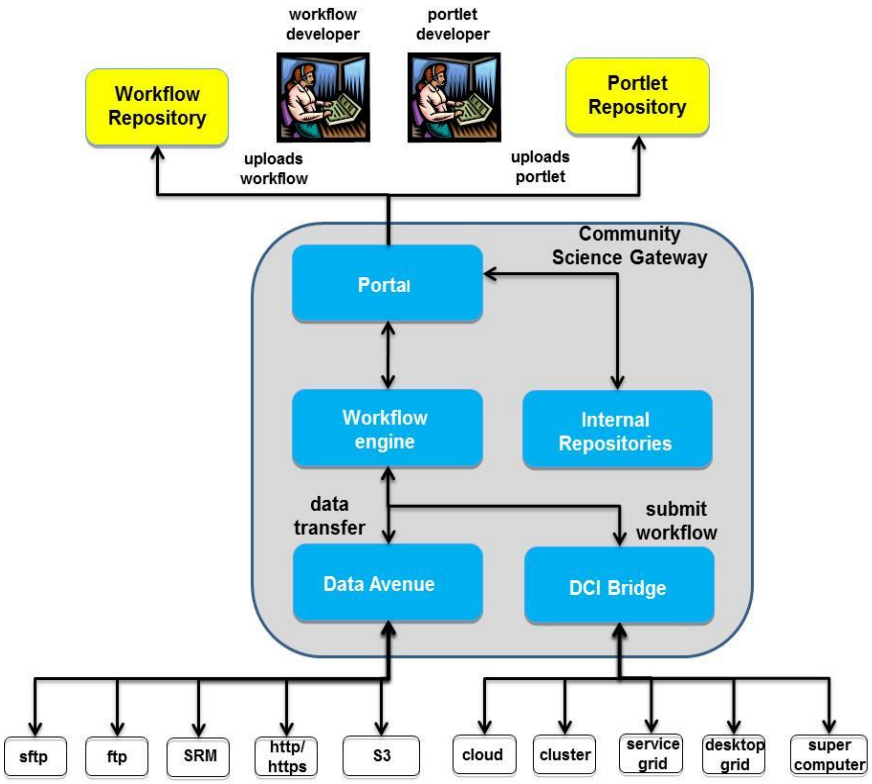

Fig. 9.1 SCI-BUS environment

There are two possible SCI-BUS development scenarios. In the first scenario, workflow developers first create workflows on the community science gateway and then upload them to the workflow repository. Next, the portlet developers elaborate ASM portlets. They either select workflows from the workflow repository, or develop them from scratch, and incorporate the workflows into ASM portlets. Finally, they publish these portlets in the portlet repository. In the second scenario a different community develops its own community portal. They can either develop all portlets and workflows from scratch, or search the repositories and find portlets and workflows they want to reuse. Reusing portlets and workflows that have been published in repositories can simplify and speed up the science gateway development process. Therefore, the main aim of the repositories is to support developers and system administrators both inside one community and among different research communities to share portlets and workflows.

9.3 Sharing Workflows

The gUSE gateway framework enables sharing of workflows inside a community using the gateway's local storage services (Sect. 9.3.1) and among communities using the external SHIWA Workflow Repository (Sect. 9.3.2). Having these two types of storage services, users can upload (or export), search, find, select, and download (or import) workflows.

9.3.1 Workflow Storage in the gUSE Gateway

There are three internal repositories in the gUSE framework: the application repository, the File Storage and the workflow storage. They manage workflows and their data, and also support sharing WS-PGRADE workflows among the science gateway users. These repositories provide basic functionalities such as uploading or downloading files and workflows.

File storage manages the uploaded input files and executables, and it additionally may store the generated output files in the gateway's file system using its folder structure.

Workflow storage (WFS): The workflow configuration is stored in a database handled by the WFS. It manages a database consisting of several tables that store the workflow's property set, such as the required resource, the type of workflow node (for example, binary, service or workflow), etc. It means that the workflow description itself is not stored explicitly; instead, it is generated on-demand by WFS when the user downloads a workflow.

Application Repository: The application repository enables the users of the same gateway to publish their workflows internally or to import workflows that others have exported. In technical terms, exporting a workflow into the application repository means first getting the workflow description from the WFS and its files from the file storage, and then sending them as a zip file to the application repository which saves this file in a particular folder on the server. There is no role management implemented in this repository: the users have the same privileges, so everyone can see the workflows exported by everyone else. Beside its main scope, the application repository plays a role in two additional scenarios. In the first scenario it supports sharing workflows among gateway users who run workflows through end- or power-use views. In the second scenario users execute shared workflows through ASM portlets. In both scenarios the workflow systems imports workflows from the application repository to enable their execution.

Usage scenarios: We distinguish between development and execution scenarios. In the first case, as shown in Fig. 9.2 these storage entities have different roles in the three phases of the workflow development: graph creation, concrete workflow configuration, and workflow exportation. Creating the abstract graph of the workflow using the graph editor defines the workflow skeleton by adding its nodes and the relations among them. This step does not indicate any file transfer. Therefore, this step uses the workflow storage only, which stores the given infor-

mation in the database. In the workflow configuration phase users can add arguments to specify the resources to be used or the types of workflow nodes and so on. In this phase input files and executables can be also uploaded. These operations invoke the file storage to transfer these files into the proper folders. Information storage in the workflow configuration step is supported by both the File Storage and the workflow storage services. Finally, the export process requires cooperation of all three storage components. First of all, the workflow configuration and its data must be collected from the file storage and the workflow storage. Next this data is compressed and is exported to the application repository as a single zip file.

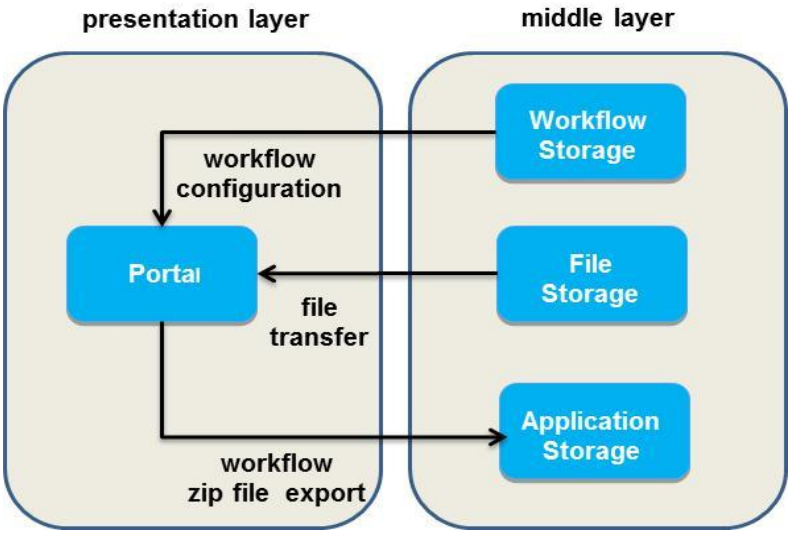

Fig. 9.2 gUSE workflow storage

In the execution scenario the Application Repository supports sharing workflows among gateway users. They import the workflows from the Application Repository in order to parameterize and execute them.

9.3.2 SHIWA Workflow Repository

The "Sharing Interoperable Workflows for Large-Scale Scientific Simulations on Available DCIs" (SHIWA) project [SHIWA/2014] developed and deployed the SHIWA Repository. The repository allows users to manage workflows including upload, upgrade, and delete workflow operations and use workflows including browse, search, and download workflow operations.

Repository users: There are three types of repository users: domain researchers, workflow developers and repository system administrators. Workflow developers are familiar with workflow systems and infrastructures where workflows are executed. They can create workflows and can support researchers to run these work-

flows. To achieve this, they define and publish workflows in the workflow repository. Researchers can browse/search the workflow repository, select and download workflows from the repository, and run these workflows on DCIs. The repository system administrators manage the repository. They have the highest privileges among the repository users.

Workflow data: The repository manages four data entities: abstract workflows (or workflows), concrete workflows (or implementations), configurations and workflow engines. The abstract workflow describes the workflow behavior. It specifies the workflow graph, including workflow nodes and edges, node inputs and outputs. However, it does not actually contain any binaries or data needed to run the workflow because any abstract workflow may have been implemented in different workflow systems. Concrete workflows strictly follow the definitions of the abstract workflow. They contain binaries/executables, and input data or references for input data are defined by the configurations. The configurations contain data, parameters, and files, for example, default or sample data or references to these data entities. The workflow engine includes or references files and any other data required to execute a workflow engine on an infrastructure.

Repository domains: The repository has a private and public domain. In the private domain developers can manage workflows having both read and write access rights. They can specify, upload, modify and delete them. The content of this domain is available only for registered users. In contrast, the public domain does not require any registration. It allows browsing/searching workflows in the repository, and also downloading them.

Repository views: The SHIWA Repository offers two views: researcher (or browse) and developer (or table) views. The researcher view (Fig. 9.3) presents workflow data assuming a basic user-level understanding of workflows. It enables users to find workflows they need to run scientific experiments. In this view the repository displays the summary (domain, application, owner, description, graph, etc.), the number and types of inputs and outputs, data sets, and details of the existing concrete workflows. Users can search workflows, either selecting a domain via the domain list or by specifying workflow names. The developer view allows workflow developers to upload workflows manually, edit workflow data, and delete workflows. This view displays the name, owner, status, and description of the workflow, and also the group to which the workflow belongs in a table format. After selecting a particular abstract workflow, further details such as attributes and its implementations are also displayed.

Repository usage scenarios: Users can use the repository in three scenarios. In the first scenario workflow developers can upload, upgrade, and delete workflows. They can publish workflows either automatically or manually. In the first case they upload workflows from the WS-PGRADE/gUSE-based science gateway using the export operation. In the second case they enter workflow data manually using the developer (or table) view. The repository offers two major operations to find workflows: browse and search operations. The browse operation enables checking the list of workflows, selecting them, and displaying their details. The

search operation allows users to specify search criteria, for example, domain name or workflow name to filter the search operation. Users can browse and/or search the repository in both the researcher (or browse) and developer (or table) view, and select workflows they want to execute. They can access the repository through either the repository GUI or the repository portlet of the WS-PGRADE/gUSE-based science gateway. In the third scenario users can automatically or manually download workflows from the repository. The WS-PGRADE/gUSE-based science gateway enables users to automatically import workflows from the repository. Workflow developers can also manually download workflows through the developer (or table) view.

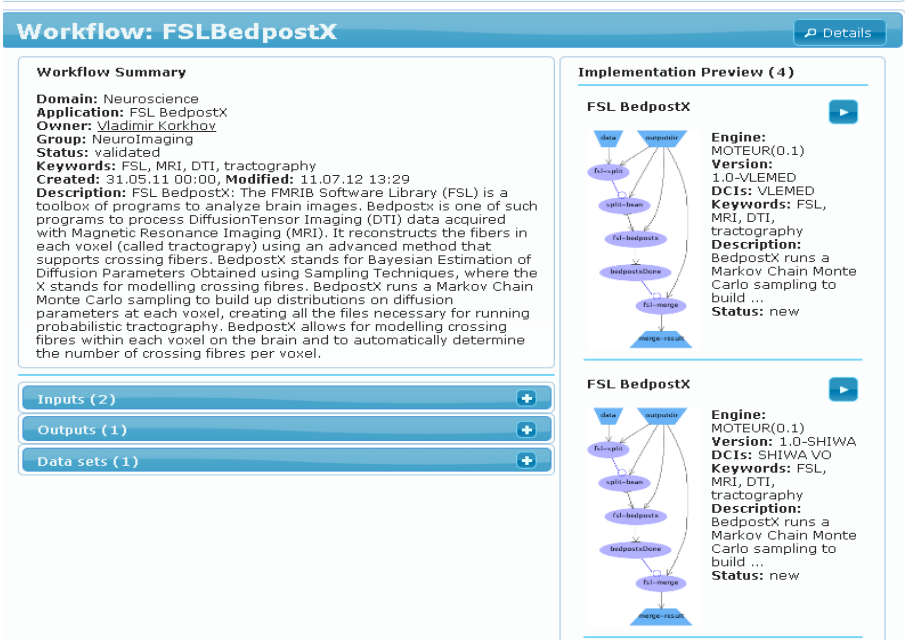

Fig. 9.3 Researcher (or browse) view of the SHIWA repository, containing description and metadata about the abstract workflow (*left*) and two implementations (*right*)

9.4 SCI-BUS Portlet Repository

Much like the SHIWA Repository, the SCI-BUS portlet repository offers a service that aims to simplify the way in which developers and researchers use distributed computing infrastructures. This repository increases the availability of portlets, providing a service to aid their discoverability, uploading, and downloading.

Repository users: There are three main user groups for this service; portlet developers, portlet users, and repository administrators. Portlet developers use the

repository to publish their portlet. From the interface they can describe the portlet, assign its attributes, relate its dependencies, and release – and upload files for – different versions of their portlet. The portlet users can search, investigate, and ultimately download and install a portlet of their choice through the repository GUI. Finally, the repository system administrator has super-user powers over all the information stored and the users registered. The overall design of the repository is centered on assumptions made about each user's specific knowledge domain. Again, like the SHIWA Repository, certain views are only applicable to specific user types.

Repository data: The portlet repository and the SHIWA Repository share many of the design concepts, and the back-end infrastructures are similar. There are three main data constructs used in the portlet repository: portlet, portlet version (or implementation), and attributes that can be assigned to both the portlets and their versions. The portlet can be seen as an abstract definition of the application's behavior. This is aided by a set of attributes to store a description and several URLs that point to support or documentation. In addition, developers can assign their portlet to a category or define a set of attributes (or tags), both of which aim at helping the users search for and discover portlets relating to their specific knowledge domain or need. To aid the process of selection and discoverability the developer can also upload screenshots of their portal in use (Fig. 9.4). The portlet version specifies a concrete implementation of the portlet. This has a version number, a set of dependencies, and any files or other information that help describe this particular version. The dependencies describe the environment that the user must have in order to install and run the portlet. They could be a gUSE version, a Liferay portal version, or a link to a workflow on the SHIWA Repository.

Portlet visibility: Both a portlet and its versions have a visibility status attached to them, which can be either public or private. After creation this value is always private assuming the portlet is under development. Next, the developers have to set manually the visibility to public to enable it to be seen by users, effectively publishing their portlet. This can work the other way around, in order to hide a portlet or version from public viewing.

Repository usage scenario: There are two main usage scenarios. In the first scenario a portlet developer wants to publish a portlet. The repository gives him/her the ability to create a new portlet entity, specify its attributes, upload its files, and manage any data he/she decides to store. After creating and describing any portlets or their versions, they can associate tags and a category to portlets and their versions to help repository users browse and search effectively. In addition to this the portlet repository can act as a hub for releasing updates to maintained portlets throughout the development lifecycle. The second scenario involves a user, either a researcher or a portlet developer, who does not have to be registered in the portlet repository. They can click on the "Portlets" tab and can browse all the public portlets. They can also search by knowledge domain (category) or by keywords (tags) to find a suitable portlet. They then have access to the files for each publically available version and links to documentation or support.

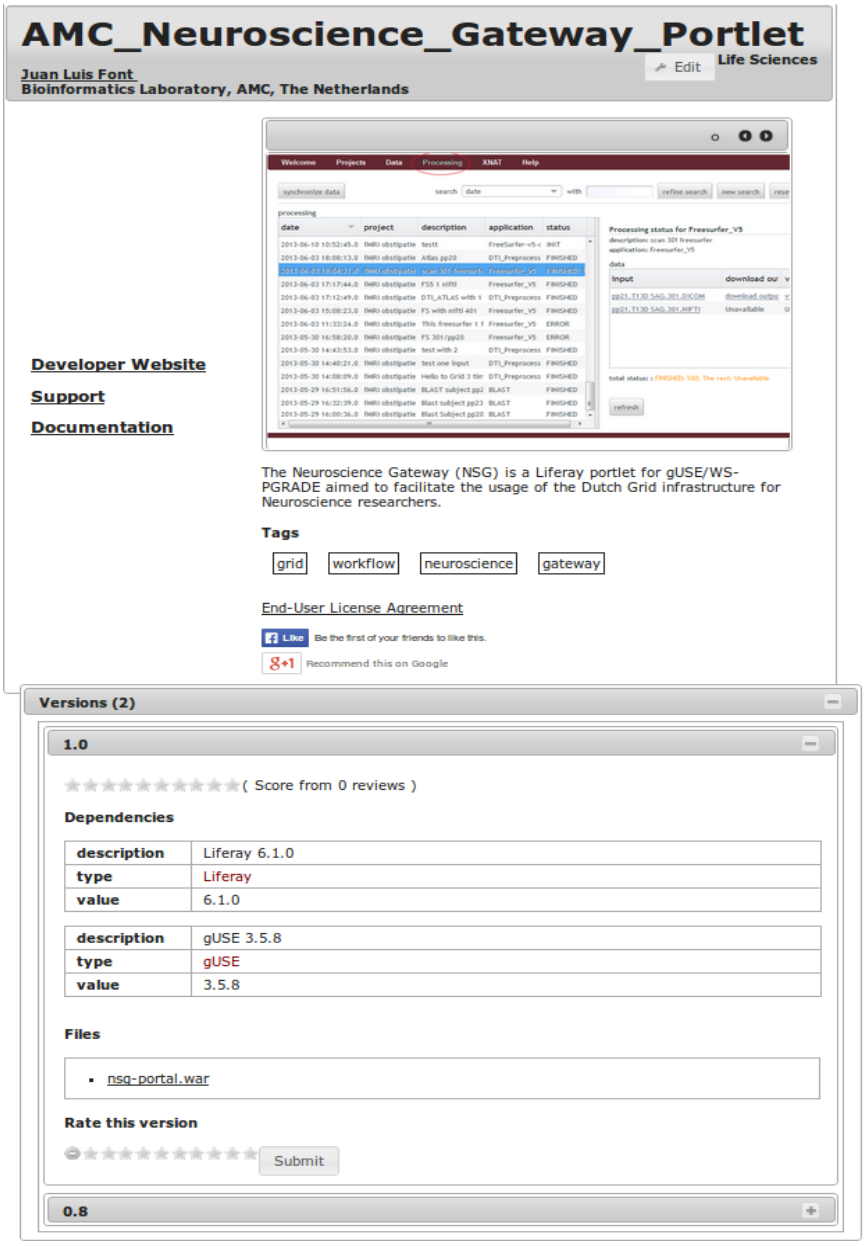

Fig. 9.4 User view of a publicly accessible portlet

9.5 Supporting Workflow Interoperability

To address workflow interoperability the SHIWA project developed the coarse-grained interoperability (CGI) approach [Terstyanszky/2014]. SHIWA created and deployed a production-level CGI service, called the SHIWA Simulation Platform (SSP) [Korkhov/2013] to enable execution of workflows created in different workflow systems and executed on different distributed computing infrastructures (DCI). Several research communities use the CGI concept to create, integrate, share and run workflows.

CGI concept: CGI is based on workflow engine integration approach. It manages non-native workflows as black boxes. These workflows are described as legacy applications and their descriptions are uploaded to the SHIWA Repository. These descriptions identify the workflow engine that can execute the workflow. The CGI concept manages two workflow types: native and non-native workflows. Workflows of the host workflow system are called native workflows, while all others are considered as non-native ones. In the gUSE gateways, the native workflow system is the WS-PGRADE workflow system. All others such as Galaxy, Kepler, MOTEUR, Taverna, etc., are managed as non-native workflows. According to the CGI concept, the native workflow engine (workflow engine A) contacts a submission service when it identifies a workflow of a non-native workflow engine (workflow engine B) and forwards the workflow ID to the submission service. It retrieves the non-native workflow from the repository and associates it with the workflow engine that can run it. Finally, the submission service forwards the workflow to the associated workflow engine, which then executes the workflow. To support the CGI concept, a gateway based on WS-PGRADE can be connected to two SHIWA services: SHIWA Repository, to publish and import workflows, and SHIWA Submission Service, to run non-native workflows.

SHIWA architecture: The simulation platform (Fig. 9.5) contains a portal (SHIWA Portal), a submission service (SHIWA Submission Service), and a workflow repository (SHIWA Repository). The SHIWA portal is a general purpose gateway based on the WS-PGRADE/gUSE framework. It has a built-in workflow system: the WS-PGRADE workflow system, which is used as the native workflow engine. The SHIWA Repository stores the formal description of abstract and concrete workflows and data needed to execute them. Workflow developers can describe, modify, and delete workflows through the repository GUI.

To support non-native workflow execution, the SHIWA Submission Service imports the previously uploaded non-native workflow from the SHIWA Repository and associates with the non-native workflow engine that can execute this workflow. This service either invokes locally or remotely predeployed workflow engines, or submits workflow engines with the workflow to local or remote resources to execute workflows. The “Building an European Research Community through Interoperable Workflows and Data” (ER-flow) project [ER-flow/2014], which is the follow-up of the SHIWA project, has been managing the SHIWA Simulation Platform since September 2012. ER-flow and SCI-BUS offer a combi-

nation of a development and an execution environment. ER-flow provides the SHIWA Simulation Platform as a development environment where workflow developers can create workflows, including native, non-native, and meta-workflows. SCI-BUS offers the technology to create and run SCI-BUS community gateways as an execution environment. Researchers can execute workflows through the end-user interface of the SHIWA Portal or through ASM portlets in the SCI-BUS community gateways.

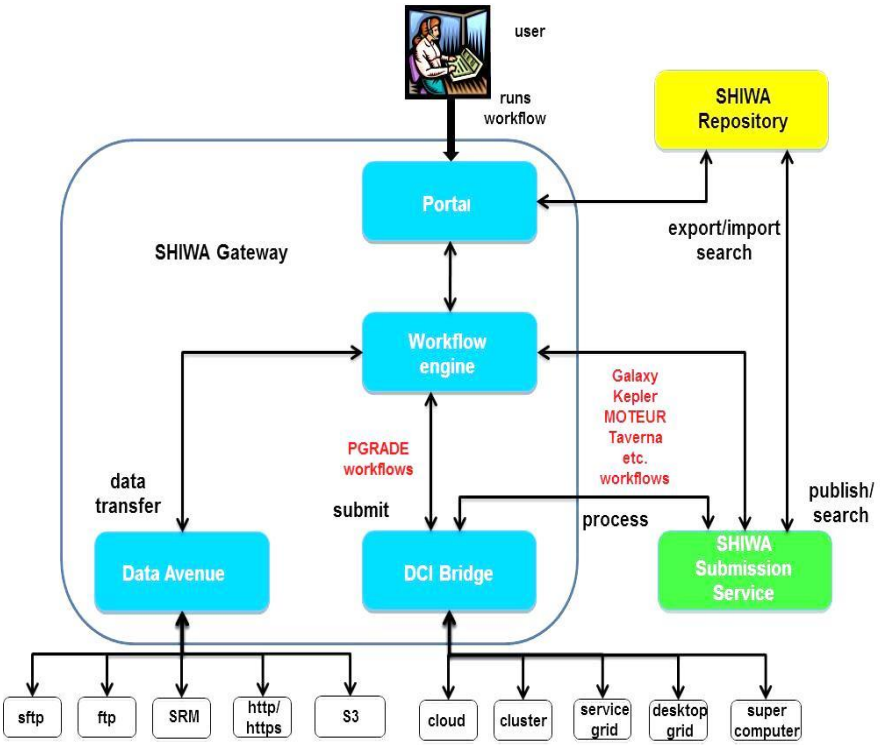

Fig. 9.5 SHIWA Simulation Platform

SHIWA usage scenario: The simulation can be used in SCI-BUS community gateways to run non-native workflows based on the CGI concept. First, users search the SHIWA Repository and select a workflow they want to execute. Next they download the selected workflow from the repository to the SHIWA portal using the import operation. They either embed the non-native workflow as a single job in a WS-PGRADE workflow, or create a meta-workflow combining WS-PGRADE jobs and workflows with non-native workflows using the portal's workflow editor. Then they submit this workflow to the gateway's WS-PGRADE workflow engine, which forwards it to the DCI Bridge. The DCI Bridge sends the submission request of the non-native workflow to the SHIWA Submission Service. This service retrieves the workflow from the SHIWA Repository, associates

it with the workflow engine that executes it, and returns it to the DCI Bridge. Finally, the DCI Bridge either sends the workflow to a predeployed workflow engine or submits the workflow engine with the workflow to be executed on the target infrastructure.

9.6 Sharing Portlets and Workflows: A Case Study

In this section we presented a case study in which various components of an existing gateway published in the SCI-BUS repositories were reused for the development and customization of another one. These repositories facilitate sharing various artefacts (applications, workflows, and portlets) of science gateways to speed up the development of new applications and gateways.

The University of Westminster developed the Westminster Desktop Grid Gateway to support various local user communities. This gateway is connected to the University of Westminster Local Desktop Grid (WLDG), utilizing the free computing capacity of up to 2000 laboratory computers. The gateway offers custom ASM portlets for molecular docking and 3D animation rendering used in both teaching and research. The workflows and portlets developed for this gateway have been uploaded to the SHIWA Workflow Repository and the SCI-BUS Portlet Repository, respectively.

Molecular docking has been identified as one of the key application areas that could be supported by publicly available science gateways that would attract potentially large user communities. However, supporting users in an open policy is not acceptable in a closed university resource such as the WLDG. Also, a public gateway requires significant resources to serve the expected large number of end-users. Within the SCI-BUS project, MTA SZTAKI developed and set up a customized molecular docking gateway based on the WS-PGRADE/gUSE framework that is operated on the EDGeS@home public desktop grid. Instead of developing the gateway from scratch, SZTAKI utilized various repositories to download, install, and customize the applications, workflows, and portlets already available.

First, SZTAKI selected the three molecular docking portlets that form part of the University of Westminster Desktop Grid Gateway. Although the Westminster gateway also includes additional portlets, the docking portlets are separate entities and are uploaded independently to the SCI-BUS Portlet Repository. Therefore, these portlets can be installed on other gateways too. The docking portlets call three workflows that are available from the SHIWA Workflow Repository. Next, SZTAKI developers downloaded these workflows and applied some necessary transformations to the workflow jobs. As SZTAKI planned to operate a public gateway, they remapped all jobs running on the local submitter (which may be disadvantageous from both performance and security points of view in a public gateway) to run on the EDGeS@home desktop grid. Such remapping of workflow jobs does not interfere with the ASM-based portlets that call the workflows.

Therefore, no further programming or modification of the user interface was required. Finally, the workflows call BOINC desktop grid applications that were downloaded and installed on the EDGeS@home desktop grid server.

9.7 Conclusions

Internal and external repositories used by gUSE gateways enable and support sharing and reusing applications, portlets, and workflows. As a result, the gateways support two different levels of research cooperation: inside a community and among communities belonging to different disciplines. Sharing and reuse of these artefacts via repositories significantly shortens the development time and improves the user experience.

The current repositories have three limitations. First, they do not manage provenance as expected and required by the research communities. Second, the workflow repository does not have proper support for workflow execution on the cloud. Workflow developers can upload virtual images of workflows as files to the workflow repository but there is no GUI to manage them properly. Third, the portlet repository does not enable automatic portlet export and import operations. These limitations should be addressed by MTA SZTAKI and the University of Westminster to further improve sharing portlets and workflows inside and among research communities.