

April 2009

Workflow Repository Integration with the P- GRADE Portal

Cory Douglas Stone
Worcester Polytechnic Institute

Daniel A. Alderman
Worcester Polytechnic Institute

Follow this and additional works at: <https://digitalcommons.wpi.edu/mqp-all>

Repository Citation

Stone, C. D., & Alderman, D. A. (2009). *Workflow Repository Integration with the P-GRADE Portal*. Retrieved from <https://digitalcommons.wpi.edu/mqp-all/835>

This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.

Workflow Repository Integration with the P-GRADE Portal

a Major Qualifying Project report
submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the
Degree of Bachelor of Science by

Daniel A. Alderman

Cory D. Stone

April 29, 2009

Professor Gábor N. Sárközy, Major Advisor

Professor Stanley M. Selkow, Co-Advisor

ABSTRACT

The current version of the P-GRADE Portal supports the creation, management, and execution of application workflows from a user's Portal storage space. What is lacking in the current implementation is the ability for users to share application workflows to help facilitate information dissemination and collaborative development. To provide this desired functionality, a workflow repository system was proposed for integration with the P-GRADE Portal. After analyzing various existing repository technologies, DSpace was chosen as the best candidate for integration with the P-GRADE Portal and a design plan was drafted. Next, a DSpace installation was established and configured for the needs of the P-GRADE Portal and application workflow storage. In order to fully integrate the DSpace repository with the P-GRADE Portal, additional work was needed to create a new layout tab in the Portal, along with three new portlets and their corresponding business logic to handle interactions with DSpace and uploading or downloading to/from a user's Portal storage space. After implementation of the system and rigorous testing, integration was completed with the production installation of the P-GRADE Portal, allowing Portal users to browse and share application workflows.

ACKNOWLEDGEMENTS

We would first like to thank our project sponsor, MTA SZTAKI, especially the Laboratory of Parallel and Distributed Systems (LPDS) and its laboratory head, Professor Dr. Péter Kacsuk, for providing us with the opportunity to work on an exciting and meaningful project in the field of Grid computing. On the same note, we would like to thank Worcester Polytechnic Institute for allowing us to complete our final project abroad in Hungary, which has truly been a once in a lifetime opportunity.

The following individuals deserve particular acknowledgement for their contributions to our project, as well as showing us great hospitality. As mentioned, Professor Dr. Péter Kacsuk not only provided us with the opportunity to work on this project, but also gave us guidance during the course of our time in Hungary. Also, we would like to thank Dr. Róbert Lovas for his continued help and support throughout the project, serving as our mentor, as well as making us feel at home during our stay. Gergely Sipos provided helpful feedback and also made us feel at home outside of SZTAKI, including inviting us to partake in indoor soccer with SZTAKI colleagues (which Dan greatly enjoyed). Attila Marosi was very kind and helpful during our stay and thanks are due to him for teaching Dan how to play squash. Gábor Hermann also deserves notice for checking up on how we were doing and making sure that we got the proper care when we were feeling ill.

Also, we extend our thanks to Ákos Balaskó for answering our daily questions and providing lessons, guidance, and technical support throughout the course of the project. Thanks as well to Sándor Ács for his great deal of technical support and for his hospitality during our stay in Hungary. József Patvarczki was of great help back at WPI, helping to prepare us for our trip and giving us an overview of the Grid computing basics. Finally, we would like to thank our advisor Gábor Sárközy and co-advisor Stanley Selkow for their guidance on both our project and our stay in Hungary. Professor Sárközy was very diligent in making sure that we were well equipped for our stay and constantly helped us become more acclimated to living in a new environment and culture. We especially thank Professor Sárközy for advising this project and providing us with such a great opportunity to complete our MQP abroad.

Table of Contents

Abstract	2
Acknowledgements	3
1: Background.....	7
1.1: Project Statement	7
1.2: Grid Computing	8
1.3: MTA SZTAKI	9
1.3.1: LPDS	9
1.4: P-GRADE Portal.....	10
1.5: Repository Technologies	11
1.5.1: DSpace	13
1.5.2: Fedora Repository	15
1.5.3: myExperiment.....	17
1.5.4: Archimède	19
1.5.5: ACS	21
1.5.6: NGS.....	22
1.5.7: Download.com.....	23
1.5.8: Findings	24
2: Methodology.....	25
2.1: Requirements	25
2.2: Use Cases Diagrams.....	27
2.3: Sequence Diagrams	30
2.4: Design Plan.....	35
3: Implementation	39
3.1: DSpace	39
3.2: P-GRADE Portal.....	40
3.2.1: DSpace View Portlet.....	42
3.2.2: Download Portlet	42
3.2.3: Upload Portlet.....	43
4: Testing	46
4.1: DSpace Configuration Testing	47
4.2: Portal Integration Testing.....	47
4.3: Pre-Deployment Testing	48

5: Conclusions	50
5.1: DSpace Repository.....	50
5.2: Portal Download	50
5.3: Portal Upload	51
5.4: P-GRADE Portal Integration.....	52
6: Future Work.....	53
6.1: Collaborative Development.....	53
6.1.1: User Specified Workflow Access Permissions	53
6.1.2: User Specified Workflow Administrative Rights.....	54
6.1.3: Content Locking	54
6.2: Other Functionalities	55
6.2.1: Versioning	55
6.2.2: Commenting	55
6.2.3: Rating	56
6.2.4: Automatic Workflow Submission	56
7: References	57
Appendix A: Use Case Diagrams	60
Appendix B: User Manual.....	63
Appendix C: Installation & Administration Manual.....	72

Table of Figures

Figure 1 - P-GRADE Portal interface	10
Figure 2 - Workflow editor.....	11
Figure 3 - DSpace on University of Texas at Austin (https://pacer.ischool.utexas.edu/)	14
Figure 4 - Fedora Repository UI implemented by Plus One (http://www.plusone.org).....	17
Figure 5 - myExperiment (http://www.myexperiment.org)	18
Figure 6 - Archimède on Érudit (http://www.erudit.org).....	21
Figure 7 - National Grid Service (http://www.grid-support.ac.uk/)	23
Figure 8 - Phase 1: DSpace Installation	28
Figure 9 - Phase 2: P-GRADE Portal Integration.....	29
Figure 10 - DSpace user uploading a workflow	30
Figure 11 - Anonymous user downloading a workflow.....	31
Figure 12 - Administrator creating a new DSpace collection (workflow category).....	32
Figure 13 - Portal user uploading a workflow to DSpace	33
Figure 14 - Portal user downloading a workflow from DSpace to his/her Portal storage	34

Figure 15 - Design plan for integration of DSpace repository and P-GRADE Portal	36
Figure 16 - DSpace repository for the P-GRADE Portal	40
Figure 17 - Implementation of integration of the P-GRADE Portal and DSpace repository	41
Figure 18 - DSpace View and Download Portlets	42
Figure 19 - Upload Portlet	45
Figure 20 - DSpace user creating an account	60
Figure 21 - DSpace user editing account and workflow details	61
Figure 22 - Administrator creating a new user and editing group details	62

Table of Tables

Table 1 - Repository Technology Rankings	24
Table 2 - DSpace Usage Statistics	49

1: BACKGROUND

Computational Grids enable the sharing, selection, and aggregation of a wide variety of geographically distributed computational resources and present them as a single, unified resource for solving large-scale and data intensive computing applications [17]. Due to this design, Grids lend themselves to parallel computing, that is, running various components of an application in parallel. The P-GRADE Portal, developed at MTA-SZTAKI in Budapest, Hungary, is a user-friendly interface that provides seamless access to multiple Grids that use various implementation standards. The user submits his/her application to the Portal in the form of a workflow – a directed acyclic graph that defines the dependencies between the different components of the user’s application. The job manager then uses this workflow to schedule time on various Grid resources for processing the application in the most efficient manner possible.

1.1: PROJECT STATEMENT

Currently the workflow manager of the P-GRADE Portal provides the ability to upload a workflow from the user’s local machine to a personal storage space on the P-GRADE server. From here users may then execute this workflow on the Grid of their choice. However, as far as strengthening the Grid communities is concerned, this system has much room for improvement; what if a user wishes to share his or her application workflow with other users, so that they might learn from it? What if multiple users collaborating on a project wish to work with a single workflow? With these thoughts in mind, we would like to develop a Workflow Repository for the P-GRADE Portal that allows users to do just that: upload, share, and browse application workflows in a space accessible to people everywhere.

Therefore, with this project goal in mind and some preliminary guidance from SZTAKI colleagues, the following two part project approach was adopted for the task at hand. First off, the point was made that there is no reason to reinvent the wheel for this project – thus the first step was to examine current repository technologies. Next, the group decided which one would be the best and/or easiest to integrate with the P-GRADE Portal to provide the desired repository functionality. Furthermore, once a technology was decided upon for integration, then new portlets were crafted for the P-GRADE Portal that would utilize this repository technology backend. Also, additional functionalities would likely be implemented, time permitting, at this

phase of the project once testing proved that the integration was successful and provided the desired basic functionality to upload, download, and browse application workflows.

1.2: GRID COMPUTING

Ian Foster, “Father of the Grid” [3], and Carl Kesselman, another noted name in Grid computing, defined a computational Grid as “a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities” [15]. The core focus behind Grid computing is to provide access to a non-trivial number of resources, where the user of these resources does not have to understand how they are implemented to use them. One might associate the idea of “Grid computing” with that of a power grid; it is unnecessary to know how electricity is generated and sent to one’s home – you simply plug a power cord into an outlet, and it works [18].

Originally, the Grid was proposed as a computational infrastructure that encompassed the entire globe, and could be used to tackle computationally intensive problems that even modern supercomputers could not solve in a reasonable amount of time [20]. The intended purpose of this Grid was to provide a resource sharing infrastructure for advanced science and engineering, similar to the way the World Wide Web was initially developed as a means for scientific collaboration through the sharing of information [16]. In an effort to provide this level of computational power, Grid computing combines “geographically and architecturally dispersed hardware and software resources into large virtual super-resources” [20].

The elements that make up the “virtual super-resources” of Grids are referred to as Virtual Organizations (VOs). A VO is a set of resources which may constitute an entire Grid, or only a part of one, where a “Grid middleware layer” is built on top of them, hiding low level, hardware and software-specific implementation from high-level, standardized services and protocols. In order to access the resources of a VO, one must obtain the appropriate certificate. With this certificate, an application may be run on the resources of the designated VO [20].

One typically accesses the resources of a Grid through a Portal, such as the P-GRADE Portal. Developed by the LPDS at MTA SZTAKI, the P-GRADE Portal aims to bring us one step closer to the original idea of a “World Wide Grid” [22] by hiding the different implementations of various middleware layers and providing a single access point to numerous VOs over multiple Grids.

1.3: MTA SZTAKI

MTA SZTAKI is a Hungarian acronym for “The Computer and Automation Research Institute, Hungarian Academy of Sciences” [37]. It is governed by the Hungarian Academy of Sciences which specializes in the fields of computer science, engineering, information technology, intelligent systems, process control, wide-area networking, multimedia, and Grid computing.

MTA SZTAKI also cooperates with many other technical universities in Hungary and internationally with universities in the United States and in the United Kingdom. The institute maintains business relations with many large corporations as well, such as General Electric, RICOH, and the National Aeronautic and Space Administration (NASA) in the United States. SZTAKI is comprised of over 300 full time employees and between 40 to 60 graduate and Ph.D. students at any given time. It is divided up into various departments with specialized research focuses.

1.3.1: LPDS

The Laboratory of Parallel and Distributed Systems (LPDS) is a department within MTA SZTAKI that specializes in Grid computing and related areas of research. The laboratory head is Professor Dr. Péter Kacsuk, a renowned expert in the field of Grid computing and co-editor of the Journal of Grid Computing [23]. LPDS boasts five products, the main one being the P-GRADE Grid Portal. Also, LPDS has six main research areas which they outline in their department information:

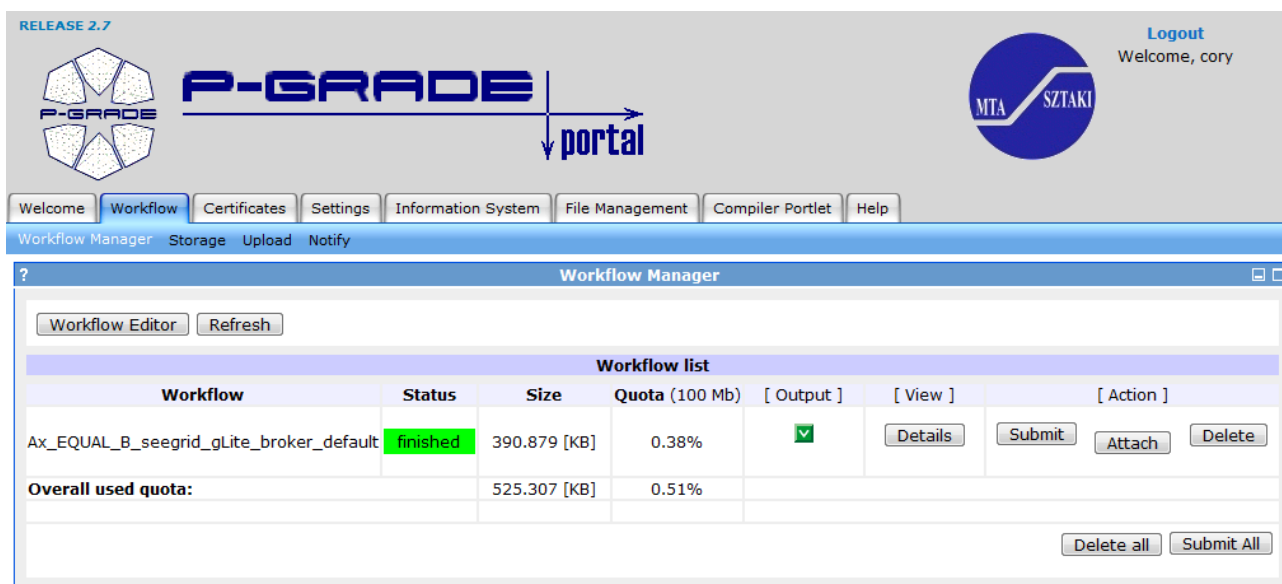
1. Parallel and Distributed Systems
2. Grid Computing
3. Graphical Software Development Environments
4. Application and Resource Monitoring
5. Performance Analysis and Visualization
6. Desktop Grid Systems

LPDS has participated in and contributed greatly to many national and international Grid research projects in the past decade. Furthermore, the Laboratory of Parallel and Distributed Computing serves as the Central-European Training Centre for EGEE (Enabling Grids for E-scienceE) and hosts Grid Summer Schools as well [37].

1.4: P-GRADE PORTAL

The P-GRADE Grid Portal, developed by the LPDS at MTA-SZTAKI, is presented as a gateway that allows users to access the resources of multiple Grids. It is a web-based application built using the GridSphere portal framework (<http://www.gridisphere.org>) that provides the ability to develop, execute, and monitor workflows on various Grid platforms [33].

As a part of The GridSphere Project, the P-GRADE Portal was constructed using the JSR 168 Portlet Specification, or Portlet API (<http://www.jcp.org/en/jsr/detail?id=168>). Portlets are web components that are deployed inside of a container (e.g. the P-GRADE Portal), and are used to generate dynamic content [19]. An image of the P-GRADE Portal interface can be seen in Figure 1.



The screenshot displays the P-GRADE Portal interface. At the top left, it shows 'RELEASE 2.7' and the P-GRADE logo. The main header features the 'P-GRADE portal' text with a blue arrow pointing right. On the top right, there is a 'Logout' link and a welcome message 'Welcome, cory' next to the MTA-SZTAKI logo. Below the header is a navigation bar with tabs: 'Welcome', 'Workflow' (selected), 'Certificates', 'Settings', 'Information System', 'File Management', 'Compiler Portlet', and 'Help'. Underneath the navigation bar, there are sub-tabs: 'Workflow Manager', 'Storage', 'Upload', and 'Notify'. The main content area is titled 'Workflow Manager' and contains a 'Workflow Editor' and 'Refresh' button. Below this is a 'Workflow list' table with columns for Workflow, Status, Size, Quota (100 Mb), [Output], [View], and [Action].

Workflow	Status	Size	Quota (100 Mb)	[Output]	[View]	[Action]
Ax_EQUAL_B_seegrid_gLite_broker_default	finished	390.879 [KB]	0.38%	<input checked="" type="checkbox"/>	Details	Submit Attach Delete
Overall used quota:		525.307 [KB]	0.51%			

At the bottom right of the workflow list, there are 'Delete all' and 'Submit All' buttons. The date 'April 3, 2009' is displayed at the bottom center of the page.

Figure 1 - P-GRADE Portal interface

Each of the tabs across the top (“Welcome,” “Workflow,” etc.) represents a Portlet. The Portlet shown in the figure is the Workflow Manager, which allows users to upload, download, execute, and view the progress of their workflows.

A workflow, in the context of the P-GRADE Portal, is a directed acyclic graph, where each node of the graph represents a particular job or application service to be carried out by the resources on the Grid. The arcs between the nodes convey dependencies between those nodes in the form of file transfers. When a node in the workflow relies on the results of another node, this dependency is represented as a directional arc pointing from the output file (displayed as a grey box) of one node to the input file (displayed as a green box) of the dependent node. The workflow runtime system of the Portal handles input and output file transferring/renaming, relieving the user of file management duties. The semantics of the workflow allows for two levels of parallelism when computing applications on a Grid: the first is among branches of the workflow that do not rely on one another and can run simultaneously, and the second is within the nodes of the workflows themselves, where there is parallel code [21]. The P-GRADE Portal provides a Workflow Editor that can be used as a graphical interface for creating application workflows. The editor, along with a sample workflow, can be seen in Figure 2.

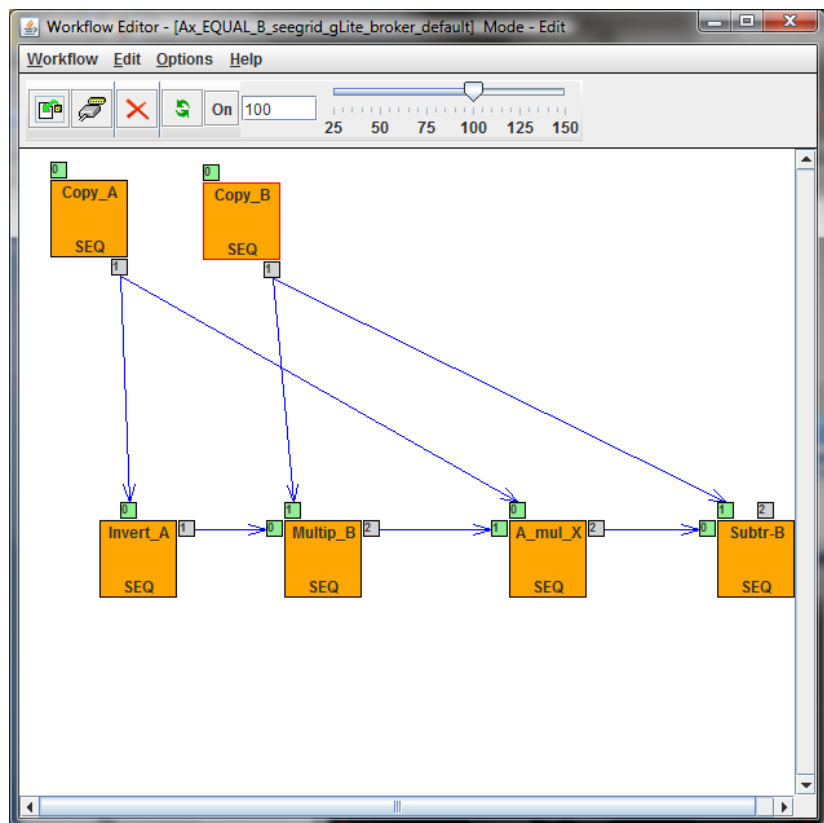


Figure 2 - Workflow editor

1.5: REPOSITORY TECHNOLOGIES

The first phase of this project was researching existing repository technologies. There was no need to reinvent the wheel; many project communities exist that are dedicated to developing repository software and services, so we wanted to find the one that best suited the

needs of the P-GRADE Portal. Five technologies were considered for integration with the Portal: DSpace, Fedora Repository, myExperiment, Archimède, and Application Contents Service. We observed two other technologies – a repository designed for the National Grid Service, and the services provided by Download.com – which could not be integrated with the Portal, but may have had features which we would have liked to implement.

Several aspects of each of the five main repository technologies were examined and compared to determine which application would be best suited for use with P-GRADE:

- Functionality – What can this application do for the P-GRADE Portal?
- API – Does this application have a well-defined API? Is it clearly documented?
- GUI – Does this application have an intuitive user interface?
- Standards – What standards in repository technology does this application use?
- License – Is this project open source?
- Version – How long has this project been under development?
- Control – Is this a standalone application, or is it a web service? Is it free?
- Installation – How long will it take to install? Is the installation process complicated? What kind of hardware and software is needed?
- References – What kind of community is behind this project? Has the application been successful? Has it been used previously with Grids?

An ideal repository for the P-GRADE Portal is one that is simple to use, and easily customizable. It should have a clearly documented and easy to understand API. A template user interface would be preferable; even better if it is based on Java Server Pages, or JSP (as the P-GRADE Portal is). The repository should adhere to standards in repository technology. An open source, standalone application would be best suited for P-GRADE. The installation process should be relatively quick, easy, and have clear, detailed instructions. Finally, an ideal repository for the P-GRADE Portal is one that has a large, active community, with plenty of sources that can vouch for the quality of the project, and provide support with the repository application.

1.5.1: DSPACE

The first project we analyzed was DSpace (<http://www.dspace.org>). DSpace was developed by MIT Libraries and Hewlett-Packard with a design goal of easy adaptation and customization in mind. It provides the following features [9]:

- A digital object model that allows any format of data to be stored
- A user interface that can be used as is or rearranged to suit one's needs
- The ability to change the format of metadata
- Configurable browsing and searching features
- Supports over twenty different languages
- Bit integrity checking
- A History system for logging changes
- Access control and authentication features

DSpace is well-documented and has an easy-to-use API based on Java Servlet and JSP technology. The API is divided into three groups: a Content Management API for reading and manipulating content stored in the DSpace system, a Search API for extracting information about items modified within a particular timeframe and scope, and a Browse API for maintaining indices of dates, authors, titles, and subjects, and allowing callers to extract parts of these. DSpace also has a default web interface, an example of which can be seen in Figure 3.

For standards in repository applications, DSpace uses Dublin Core, OAI-PMH, and OpenURL technologies. Dublin Core (<http://dublincore.org/>) is a standard format for metadata, developed by the Dublin Core Metadata Initiative, a non-profit organization dedicated to the development of “interoperable online metadata standards” that can be used in a wide range of purposes [10]. OAI-PMH, or the Open Archives Initiative Protocol for Metadata Harvesting (<http://www.openarchives.org/>) is a “low-barrier mechanism for repository interoperability” that provides a standard for metadata harvesting [30]. Finally the OpenURL standard is a protocol that is used to define links that lead a user to appropriate resources [31].

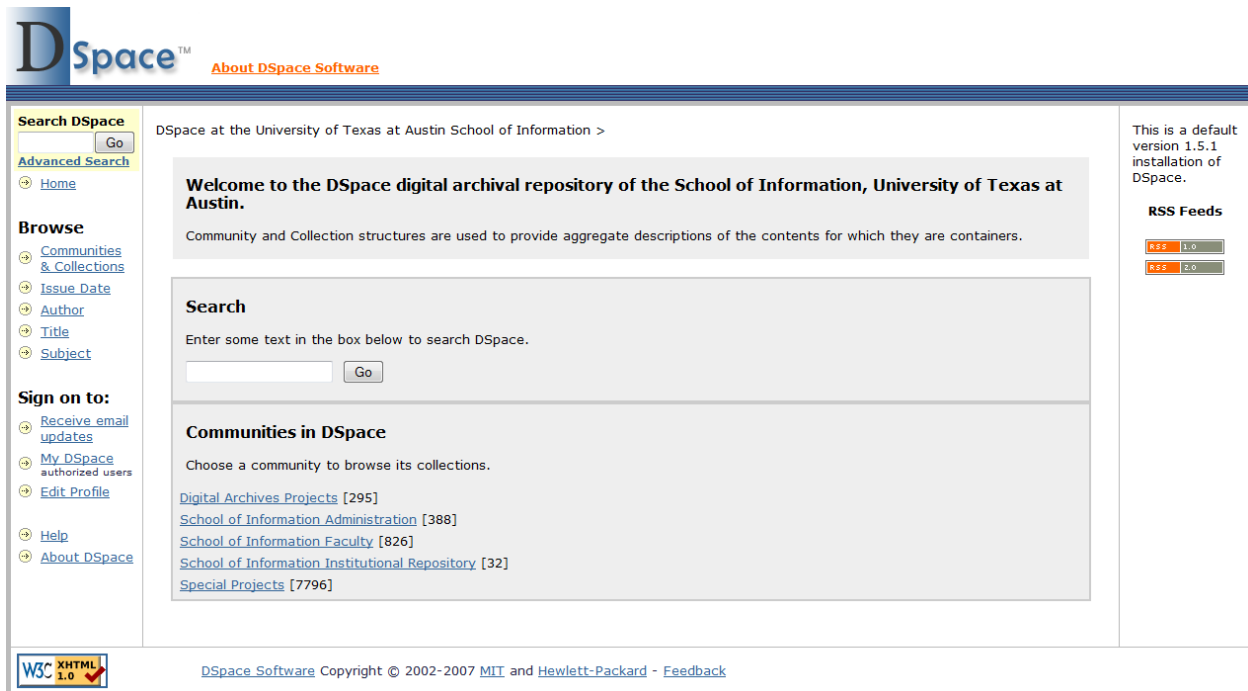


Figure 3 - DSpace on University of Texas at Austin (<https://pacer.ischool.utexas.edu/>)

The DSpace project uses a BSD license: it is open source and can be freely used and modified. It is provided as a standalone application. At the time of this writing, the currently released version of DSpace is 1.5.1. With its first release in 2002, DSpace has been under development for around 8 years.

DSpace has an easy-to-follow installation procedure, and lays out the time it should take to fully install the system: one day for prototype installation, between one day and one week for exploring the software, and one more day for a production installation with basic software [7]. The system requires a reasonably good server with a decent amount of memory and disk space. The following are the specifications for a couple of machines in use by the DSpace community that are running the application [6]:

- HP Server rx2600, dual Intel 64-bit processors (900 MHz), 2 GB RAM, 26 GB disk space, HP Storage Works msa1000 with high-performance controller: \$40,000
- Dell PowerEdge 2650 with dual Xeon processors (2.4 GHz), 2 GB RAM, two 73 GB scsi disks, 2.5 TB Apple XServe, DLT tape library: \$10,000

The DSpace system also has the following software requirements [6]:

- JDK 5 or later, Ant 1.6.2 or later, Maven 2.0.8 or later
- PostgreSQL or Oracle
- A servlet engine (e.g. Tomcat or Jetty)
- Perl

DSpace has a large, active community with 80 developers contributing code and 15 lead developers planning releases and integrating features and fixes suggested by the community. It has been very successful, with over 250 institutions, including research institutions, universities, and governments currently using DSpace within their organization [7]. In fact, DSpace was preferred in a 2007 survey of Institutional Repositories in the United States [4]. After reading of these many DSpace success stories and its ability to be highly customizable, it appeared DSpace would be one of the most promising technologies to integrate with the P-GRADE Portal.

1.5.2: FEDORA REPOSITORY

The next repository technology we examined was the Fedora Repository (<http://www.fedora-commons.org>), developed by Fedora Commons, a non-profit organization aiming to provide “sustainable technologies for sharing and preserving digital content.” The goals of Fedora Commons are similar to those of DSpace, in that they wish to provide a flexible repository application that can be suited to anybody’s needs. Fedora boasts the following features [13]:

- Powerful digital object model
- Extensible metadata management
- Expressive object-to-object relationships
- Web-service integration
- Access control and authentication
- Content versioning
- A number of features involving digital preservation

Fedora is clearly documented and has an API that is exposed as web services. The API is divided into a number of groups that provide various functionalities [14]:

- Management API – This provides an administrative interface
- Access API – An interface for accessing objects in the repository
- Various APIs for searching based on metadata, object relationships, etc.

These APIs are provided as SOAP protocols, but the Management and Access APIs also have “light” versions that provide a RESTful interface. As described by w3schools, “SOAP [Simple Object Access Protocol] is a simple XML-based protocol to let applications exchange information over HTTP.”[36] RESTful interfaces are ones that employ Representational State Transfer (REST), which is

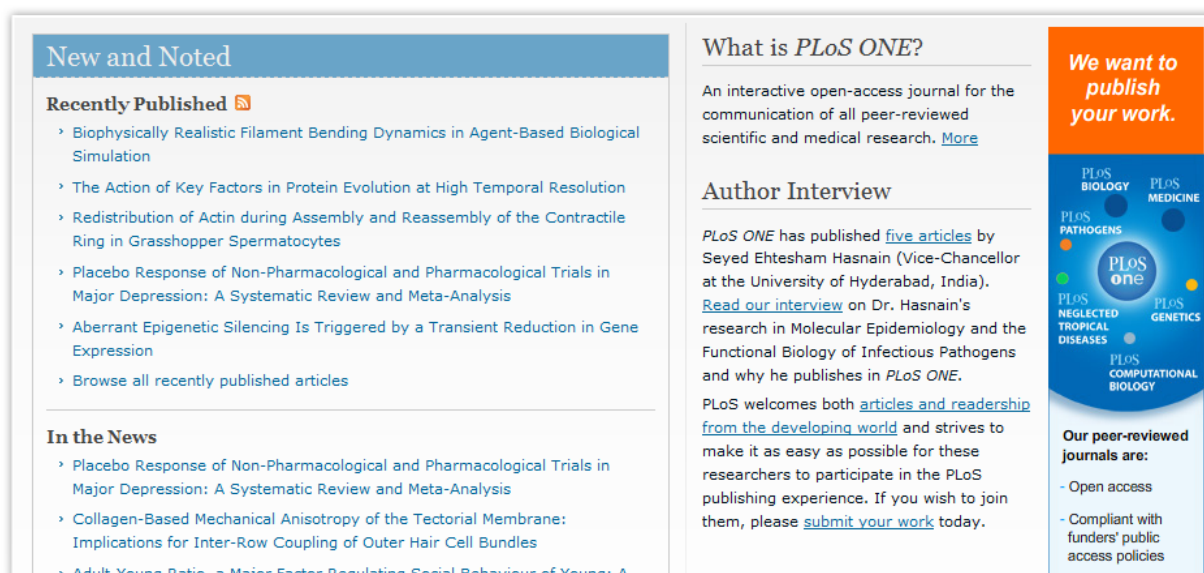
“... a key design idiom that embraces a stateless client-server architecture in which the web services are viewed as resources and can be identified by their URLs. Web service clients that want to use these resources access a particular representation by transferring application content using a small globally defined set of remote methods that describe the action to be performed on the resource.”[34]

At the time of this writing, Fedora is also working on an experimental “REST API” that would replace the light APIs. The Fedora Repository does not have a default GUI. However, an example implementation of a web interface for this application can be seen in Figure 4. Fedora adheres to the Dublin Core and OAI-PHM standards.

The Fedora Commons project is licensed under a Creative Commons license, and can be freely used and modified. It is provided as a standalone application. At the time of this writing, the currently released version of the Fedora Repository is 3.1. With its first release in 1997, Fedora has been under development for about 13 years.

The installation process for Fedora is well documented and appears relatively simple, with options for “Quick” or “Custom” installations. Quick installation will install Fedora with all default options, and is an easy way to get a prototype up and running while the Custom installation allows the installer to tailor all the features of Fedora to his or her needs. No specific hardware requirements were presented, but the following software is required [14]:

- Windows or UNIX OS
- JDK 5 or later, Ant 1.7 or later
- MySQL, PostgreSQL, Oracle, or McKoi database



New and Noted

Recently Published

- Biophysically Realistic Filament Bending Dynamics in Agent-Based Biological Simulation
- The Action of Key Factors in Protein Evolution at High Temporal Resolution
- Redistribution of Actin during Assembly and Reassembly of the Contractile Ring in Grasshopper Spermatocytes
- Placebo Response of Non-Pharmacological and Pharmacological Trials in Major Depression: A Systematic Review and Meta-Analysis
- Aberrant Epigenetic Silencing Is Triggered by a Transient Reduction in Gene Expression
- [Browse all recently published articles](#)

In the News

- Placebo Response of Non-Pharmacological and Pharmacological Trials in Major Depression: A Systematic Review and Meta-Analysis
- Collagen-Based Mechanical Anisotropy of the Tectorial Membrane: Implications for Inter-Row Coupling of Outer Hair Cell Bundles
- Adult-Young Ratio, a Major Factor Regulating Social Behaviour of Young: A

What is PLoS ONE?

An interactive open-access journal for the communication of all peer-reviewed scientific and medical research. [More](#)

Author Interview

PLOS ONE has published [five articles](#) by Seyed Ehtesham Hasnain (Vice-Chancellor at the University of Hyderabad, India). [Read our interview](#) on Dr. Hasnain's research in Molecular Epidemiology and the Functional Biology of Infectious Pathogens and why he publishes in PLoS ONE.

PLOS welcomes both [articles and readership from the developing world](#) and strives to make it as easy as possible for these researchers to participate in the PLoS publishing experience. If you wish to join them, please [submit your work](#) today.

We want to publish your work.

Our peer-reviewed journals are:

- Open access
- Compliant with funders' public access policies

Figure 4 - Fedora Repository UI implemented by Plus One (<http://www.plusone.org>)

Fedora has an active and fairly large community, second only to DSpace. There is a community of 22 developers, and 156 known major institutions using the Fedora Repository. There is also a core community of vendors who provide Fedora software integration services [12]. Overall, Fedora is a well developed repository technology that only appeared to be lacking in providing a default graphical user interface, thus making it a major candidate for integration.

1.5.3: MYEXPERIMENT

The third repository technology we researched was myExperiment. It was developed in the United Kingdom by the University of Manchester and the University of Southampton. myExperiment is designed for the ease of use for its end users [29]. This is accomplished with a simple, intuitive layout to the software's user interface (as seen below in Figure 5) and Wiki documentation for the software [25]. Overall though, this software is more of a social networking tool with the ability to share workflows. Nonetheless, myExperiment is part of the myGrid Consortium which develops the Taverna Workflow Workbench, a tool for creating and executing scientific workflows directly from myExperiment.

The screenshot shows the myExperiment website interface. At the top, there is a navigation bar with the myExperiment logo and links for About, Mailing List, Publications, Log in, Register, Give us Feedback, and Invite. Below this is a secondary navigation bar with Home, Users, Groups, Workflows (selected), Files, and Packs. A search bar is located below the navigation. The main content area displays a list of workflows, with a highlighted section for 'Kegg_DrugID (v1)'. This workflow is created by 'Rory' and has a description: 'This workflow accepts looks up drug identifiers from KEGG given a pathway identifier. You can enter a pathway ID in the form path:map07026'. It includes a workflow diagram, a license (Creative Commons Attribution-Share Alike 3.0 License), and statistics such as Rating (0.0 / 5), Versions (1), Reviews (0), Comments (0), Citations (0), and internal views/downloads. On the right side, there is a sidebar with 'New/Upload' options, 'Log in / Register' forms, and 'Popular Tags'.

Figure 5 - myExperiment (<http://www.myexperiment.org>)

Upon our research, we ascertained that myExperiment had the following primary functionalities [29]:

- Content Versioning
- User Profiles
- Resource Sharing
- Tags
- Credits and Attributions
- Messaging
- News Feeds

All of these functionalities were encompassed in a well laid out, easy to understand API that is complete with clear documentation. The graphical user interface component of myExperiment has a web-based RESTful interface and also supports a custom interface design. Despite all of the great features, functionality, and visually appealing user interface, myExperiment was deemed more of a social networking site which has many features that are unsuitable for our desired integration with the P-GRADE Portal. However, we did note that there were a few components of myExperiment that we wanted to incorporate in our implementation, regardless of what repository technology we chose.

The rest of our analysis into myExperiment produced the following details about its status, installation procedures, and community. First, myExperiment is free as it uses a BSD open source license. It is currently in a beta release version and is available as both a standalone application and a free online service. Installation requires a reasonably powerful Linux server running Debian or Ubuntu version of Linux with many additional packages and software installed. Finally, the myExperiment community entails a small number of developers and a fair number of individual users. Currently there are nine myExperiment developers contributing code, four of which are lead developers in charge of controlling releases and coordinating development. myExperiment also boasts 1,624 users at the time of this writing, which is considered fairly small for a social networking application. myExperiment even has a Facebook plug-in, allowing it to leverage existing social networks for sharing workflows [29]. Therefore, myExperiment is a fairly well-rounded repository technology which focuses more on the social aspect of information sharing.

1.5.4: ARCHIMÈDE

Another repository technology that we uncovered during our preliminary research was Archimède. Archimède is a repository developed in Canada by Laval University specifically as an institutional repository for the use of its faculty members and research community. Laval noted that this repository was inspired by the DSpace model. Archimède provides some more advanced repository functionality, some of which is lacking in the current DSpace implementation. These functionalities include [2]:

- Fine-grained Security
- Versioning

- Locks for Exclusive Content Editing
- Support for Portal Integration (JSR-168 Portlets)
- Custom Metadata Formats
- Multiple Language Support

Also, the Archimède API consists of well laid out Java documentation, and the following GUI was used with the actual repository implementation for Laval University (Figure 6). The Archimède repository implementation also supports OAI-PMH, Dublin Core, and Java Content Repository (JSR-170) standards.

As far as using Archimède is concerned, it is licensed using the GPL Open Source license, so it is freely available for use, modification, and integration. However, Archimède has a standalone release version of 2.0 and is no longer under development. The currently released version at the time of this writing had clearly defined procedures for installation which appear relatively simple. Archimède will run on any operating system and only requires the Torque database from Apache and the following additional software installations, JDK 1.4.1, J2EE Servlet 2.3, and Apache Ant [2]. As far as current installations are concerned, there was no mention of Archimède being used anywhere else besides the Laval University Library system and no Grid communities had considered it or attempted to integrate it with existing Grid technology. Finally, as far as the community for Archimède is concerned, it is no larger than developers and users at Laval University, at which development appears to have ceased in 2005. Also, at the time of this writing there were broken links on the website for Archimède, leading us to believe that it had not been worked on or updated since it was successfully launched by Laval University, which is unfortunate since it appeared to be a very promising and functional repository system.

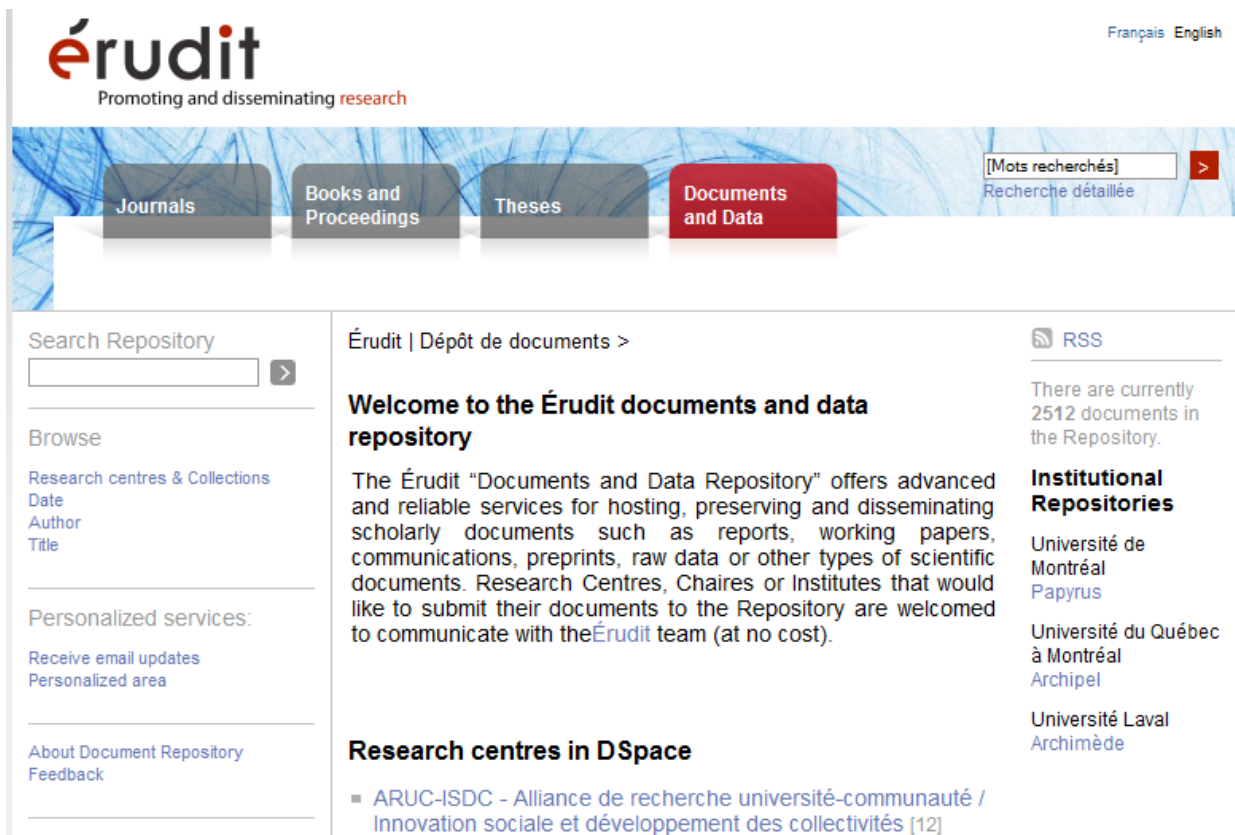


Figure 6 - Archimède on Érudit (<http://www.erudit.org>)

1.5.5: ACS

The final repository technology we analyzed in consideration for integration with the P-GRADE Portal was Application Contents Service, or ACS (<http://forge.ogf.org/sf/projects/acs-wg>). At first ACS appeared to be a promising candidate; it is a workflow repository designed for the Business Grid Computing Project in Japan. The goal of the developers was to create a repository for application-related information that meets the standards of the Open Grid Services Architecture and is not dependent on Grid implementation. ACS even provided the following major functionalities which we were looking for in our integration [1]:

- Application Lifecycle Management
- Versioning Control
- Ability to Store Resource Properties (Metadata)

While this sounded ideal for the P-GRADE Portal, the route of design and implementation taken by the project turned out to be unsuitable for our needs.

ACS has plenty of documentation, although it is lacking a clear description of the API, making application customization a difficult task. There is a command-line interface available, but no graphical interface is included.

The ACS project uses the Apache License Version 2.0, so it can be freely used and modified. It is provided as a standalone application. At the time of this writing, there was one release of ACS available, listed as a prototype. The application was developed between 2005 and 2007, but the project does not appear to be in development any longer.

ACS provided few details on the installation process, and as a result it is uncertain what kind of knowledge is necessary to install the system. A server with at least 1 GHz CPU, 512 MB RAM, and 10 GB disk space is recommended. The following software is required for ACS:

- RedHat Linux 9.0
- Java SDK 1.4.2
- Apache Ant 1.6.2
- Globus Toolkit 4

The ACS community is small, encompassing only a handful of developers, and is inactive. ACS is integrated with the NAREGI Infrastructure Middleware, which is used by the Business Grid Computing Project, but no working examples of the application could be found. The lack of a large community, poor documentation, and middleware integration restrictions ultimately led us to discard ACS from potential repositories for integration with the P-GRADE Portal, despite its initially promising appearance and similar approach to our project's goals.

1.5.6: NGS

The two other repository applications examined were the National Grid Service (NGS) Application Repository, and Download.com. NGS (<https://portal.ngs.ac.uk>), seen in Figure 7, has implemented a repository that contains single jobs, rather than entire workflows. The repository is not provided as an open source project, but rather is provided as a service only to be used with NGS; thus it can't be integrated into the P-GRADE Portal. There are features of the NGS repository that are desirable for P-GRADE, however. In particular, NGS provides the ability for anyone to explore the Application Repository and view available jobs, but only authenticated

users may submit those jobs to the Grid. Also, some nice features are supplied for browsing through the available job applications and filtering them by category and job status (e.g. “submitted,” “completed,” “failed,” etc.). Another interesting feature to note is that NGS supports the submissions of jobs to the Grid directly from the repository. Thus, this feature and a few other NGS functionalities were considered for inclusion with our implementation while the use of the repository as a whole was ruled out since its source code is not readily available to the public and it was intended to use single jobs rather than entire workflows.

Application Name	Version	Job Name/Detail	Modified	Delete	MyJob
DL_POLY_2	2.16	DL_POLY_2 Bench8	Nov 1, 2007	<input type="checkbox"/>	load
R	2.4.1	R Example	Nov 1, 2007	<input type="checkbox"/>	load
File Stage Demo	1.0	stageExample	Nov 2, 2007	<input type="checkbox"/>	load
BLAST_TOOLBOX_NCBI	25/03/2007 re	BLAST_TOOLBOX_NCBI (serial application)	Nov 1, 2007	<input type="checkbox"/>	load

Figure 7 - National Grid Service (<http://www.grid-support.ac.uk/>)

1.5.7: DOWNLOAD.COM

Download.com (<http://www.download.com>) was also briefly examined. However, upon inspection it became fairly obvious that this service would not be appropriate for the purposes of the P-GRADE Portal. Download.com is an online service intended to be a means of software distribution. Any submitted content is controlled by Download.com, and must be approved by Download.com before being made available. The workflow repository being designed for P-GRADE would ideally be under the control of P-GRADE staff, and allow instant access to the content submitted by its users. Therefore, Download.com was also deemed unsuitable for our desired implementation since it would not provide the administrative controls and rapid access to uploaded materials, along with lacking many advanced functionalities that the other repository technologies provide.

1.5.8: FINDINGS

In order to help with the decision on which repository technology would be most suitable for integration with the P-GRADE Portal, the five main technologies examined were ranked for each of the comparison criteria where a ranking was possible. The results of this ranking can be seen in Table 1.

	DSpace	Fedora	myExperiment	Archimède	ACS
Functionality	4	5	3	2	1
API/Documentation	5	4	2	3	1
GUI	3	2	5	4	1
Version/Development	4	5	3	2	1
Installation	5	4	2	3	1
References/Community	5	4	3	2	1
Totals:	26	24	18	16	6

Table 1 - Repository Technology Rankings

After thoroughly analyzing and ranking all the pertinent repository technologies on the categories specified above, the group reached the following conclusions. First, DSpace and Fedora are the two best choices for integration with the P-GRADE Portal. This brought up the interesting idea of combining both for implementation since DSpace and Fedora are currently undertaking a joint venture to merge their respective technologies. After much discussion with our SZTAKI colleagues on this topic, it was determined that DSpace should be used for the implementation while incorporating the versioning functionality from Fedora. Also, while other technologies were ruled out, it was agreed that some of their best features were to be considered for inclusion in the final implementation if feasible, such as some graphical user interface (GUI) components of myExperiment and NGS.

2: METHODOLOGY

In order to determine how the Workflow Repository for the P-GRADE Portal should be implemented, we took several steps in examining what key features were required by Portal users, how those users would interact with the repository, and how the system should be designed to behave accordingly. We first drafted the requirements for the workflow repository and discussed them with our colleagues at SZTAKI. From those requirements we developed several use case and sequence diagrams. Finally we considered what design should be used in our implementation.

2.1: REQUIREMENTS

Our requirements were divided into two sections based on our plan of implementing the workflow repository in two phases: in the first phase, we would install the workflow repository as a standalone application on its own server; the application would not require the user to have an account on the P-GRADE Portal. This first phase only required the default functionality supplied by DSpace. The second phase involved adding new portlets to the P-GRADE Portal which would allow Portal users to upload and download application workflows directly between the repository and their user space on the Portal server; for this phase, we required that the user interface have a setup similar to the standalone repository service.

PHASE ONE:

- **User Specific**
 1. Users may create an account
 2. Users may delete an account
 3. Users may modify account details
 4. Users may choose to upload to local machine
 5. Users may download to local machine
 6. Users may browse through workflows in the repository by name or category
 7. Users may search for workflows based on specific criteria
 8. Users may add tags to their workflows
 9. Users may delete the tags associated with their workflow
 10. Users may give a summary of their workflow
 11. Users may update the summary of their workflow
 12. Users may give a detailed description of their workflow
 13. Users may update the description of their workflow

14. Users may place workflow in a certain category (collection in DSpace)
15. Users may move workflow to another category (collection in DSpace)
16. Users who are not logged in shall be treated as anonymous users and only given basic privileges

- **Administrator Specific**

1. Administrators shall be able to create user accounts
2. Administrators shall be able to delete user accounts
3. Administrators shall be able to set user permissions
4. Administrators shall be able to modify user account details
5. Administrators shall be able to delete any workflow
6. Administrators shall be able to define user groups
7. Administrators shall be able to set permissions for user groups
8. Administrators shall be able to modify user group details
9. Administrators shall be able to delete user groups
10. Administrators shall be able to create categories (collections)
11. Administrators shall be able to modify categories (collections)
12. Administrators shall be able to delete categories (collections)
13. Administrators shall be able to put restrictions on file size

PHASE TWO:

- **User Specific**

1. Portal users may additionally choose to upload from Portal space
2. Portal users may additionally choose to download to Portal space

- **Application Specific**

1. Portal application shall have similar UI to standalone application

2.2: USE CASES DIAGRAMS

The system design for our repository integration with the P-GRADE Portal consisted of two phases, with the second phase expanding on the functionality of phase one. The first phase involved setting up a default DSpace installation with minor settings or layout customizations to fit our requirements. This phase required only the basic functionalities provided by DSpace.

The three actors in the use cases for this first phase are the basic actors associated with typical DSpace use cases; an anonymous user, a regular user, and an administrator. An anonymous user is only associated with simple use cases such as browsing, searching, or downloading an application workflow to his/her local directory. A regular user has many more options such as account related use cases and the ability to upload workflows. Finally, phase one provides administrator use cases which involve account and permissions administration, and many other functionalities specified by our requirements such as overseeing the storage of application workflows. A use case diagram for the DSpace installation can be seen below (Figure 8).

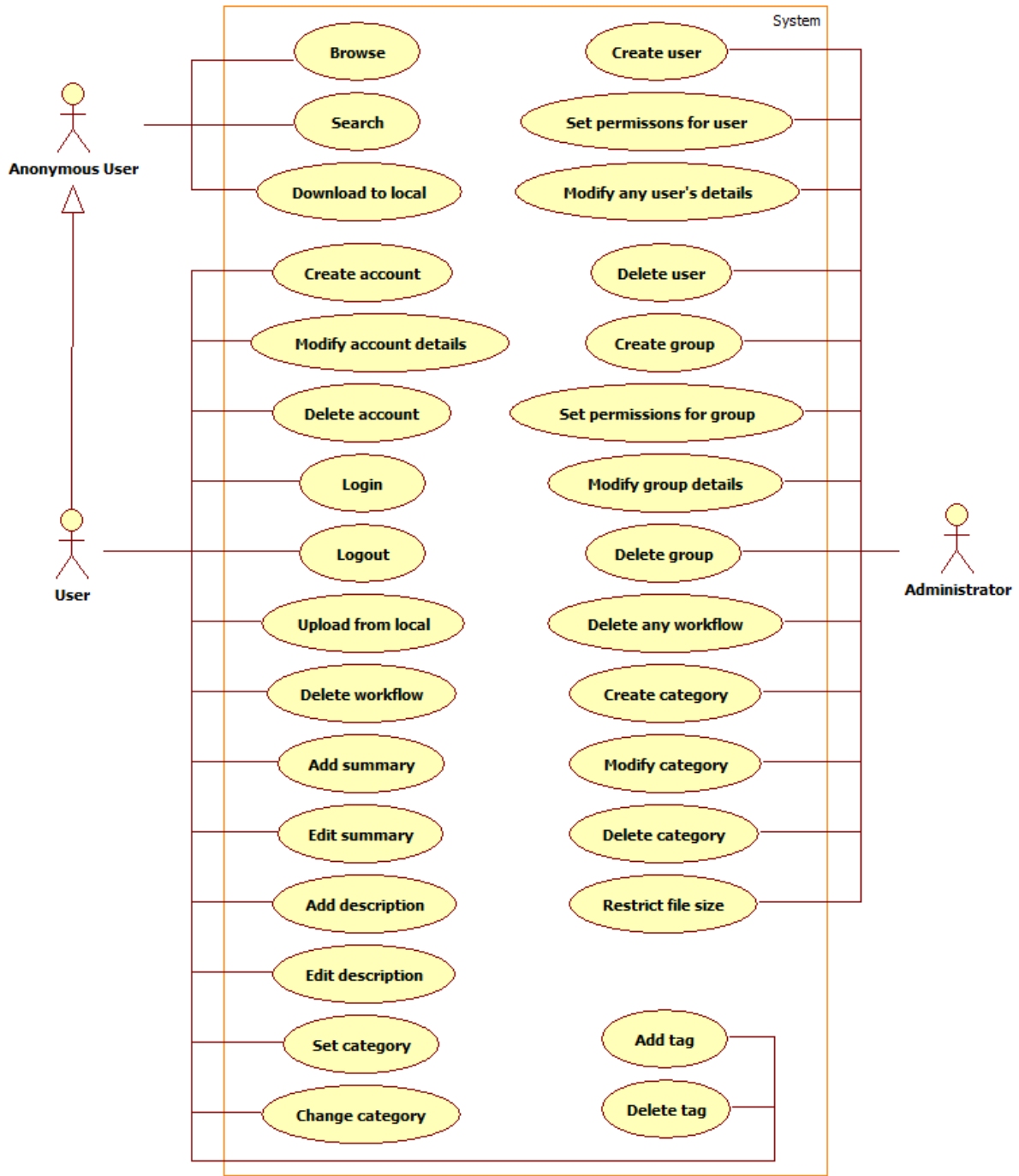


Figure 8 - Phase 1: DSpace Installation

Phase two of the system design comprised of two additional Portal related use cases and the addition of a Portal user actor who would be able to carry out these use cases. In this phase, integration with the P-GRADE Portal was actually achieved and the Portal user was able to carry out use cases for uploading and downloading application workflows to and from his/her Portal storage space. This second phase of the system works as a compliment or addition to phase one and adds in the Portal related functionality as shown in the diagram below (Figure 9).

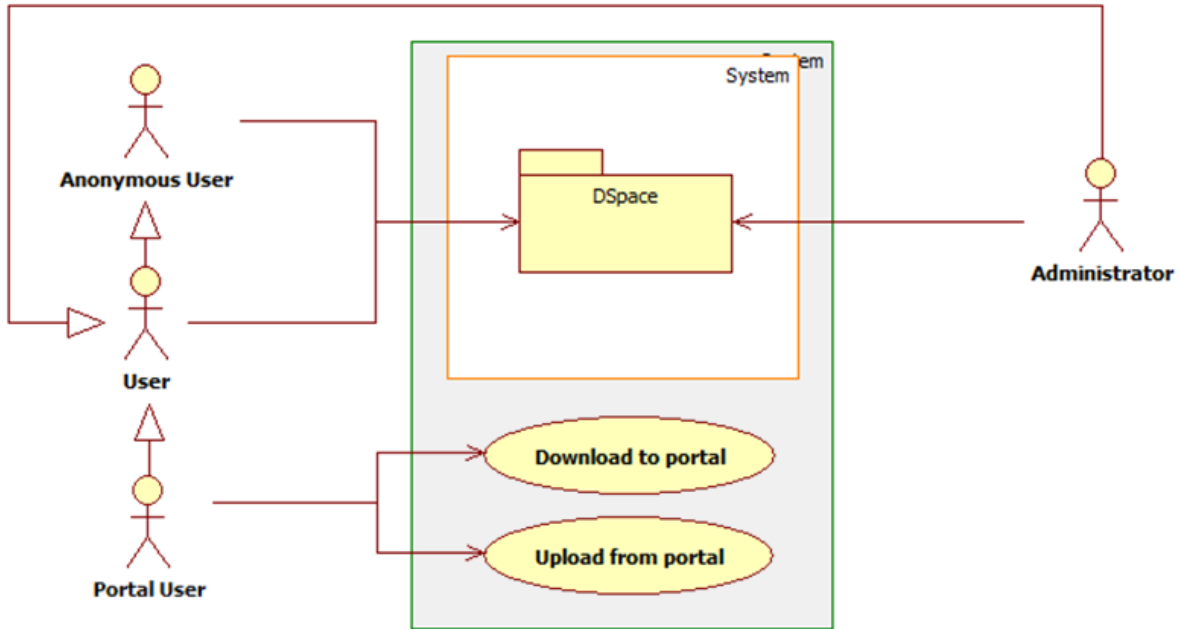


Figure 9 - Phase 2: P-GRADE Portal Integration

2.3: SEQUENCE DIAGRAMS

The following sequence diagrams are specific to the first phase of our system design. They represent the basic functionalities provided by DSpace and illustrate how users, anonymous users, and administrators interact with the system via the DSpace web user interface. Only a few functionalities are represented here; see the Appendix for more sequence diagrams.

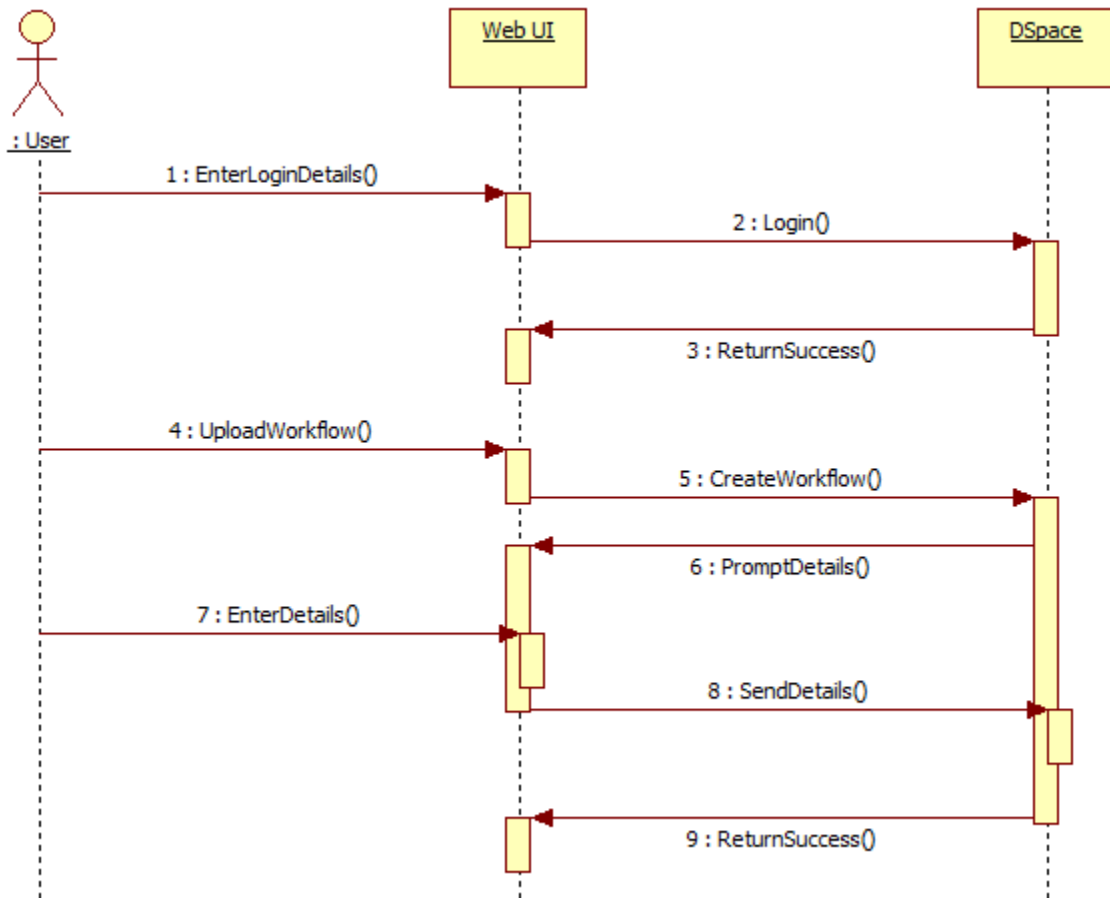


Figure 10 - DSpace user uploading a workflow

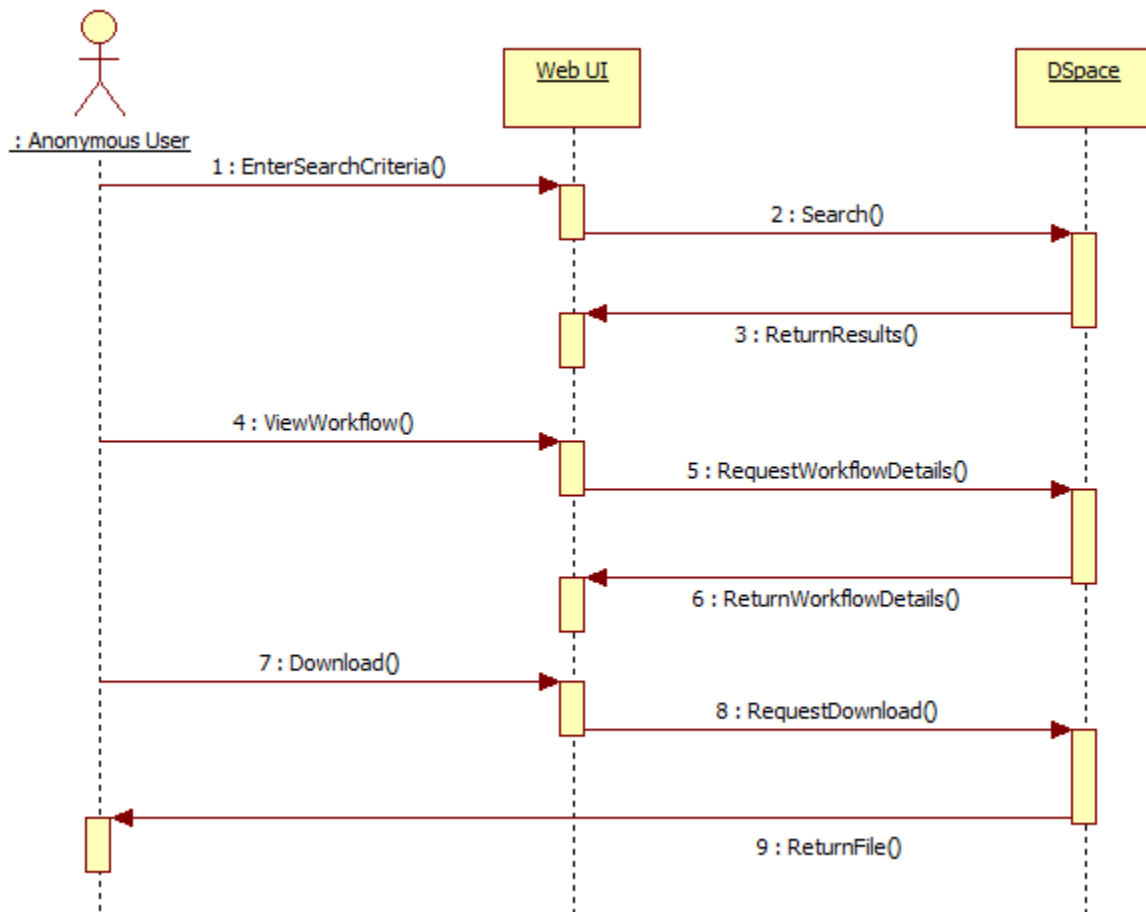


Figure 11 - Anonymous user downloading a workflow

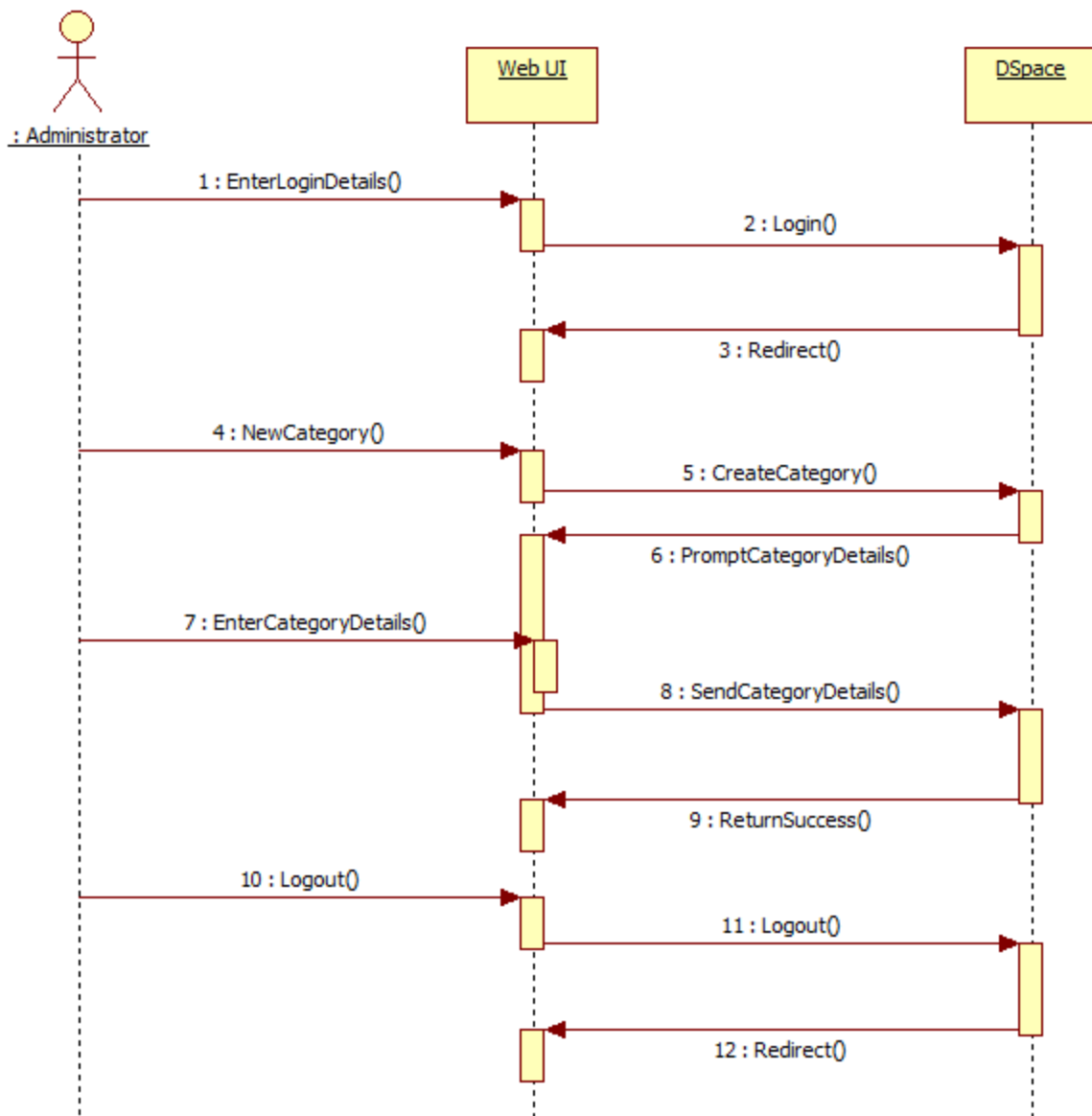


Figure 12 - Administrator creating a new DSpace collection (workflow category)

The following sequence diagrams (Figures 13 and 14) illustrate how the second phase of our system integrates with the P-GRADE Portal. These diagrams also detail how the P-GRADE Portal and DSpace interact with each other to conduct a Portal user upload or download of a workflow.

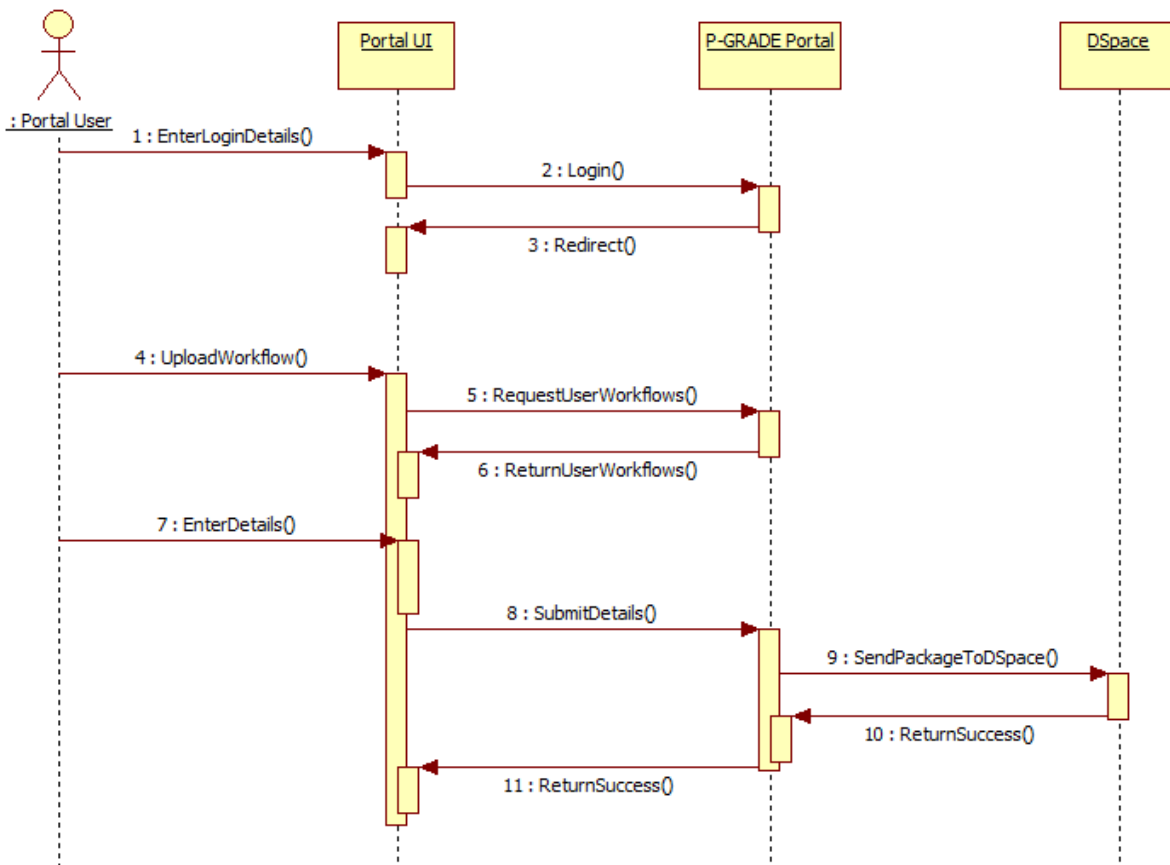


Figure 13 - Portal user uploading a workflow to DSpace

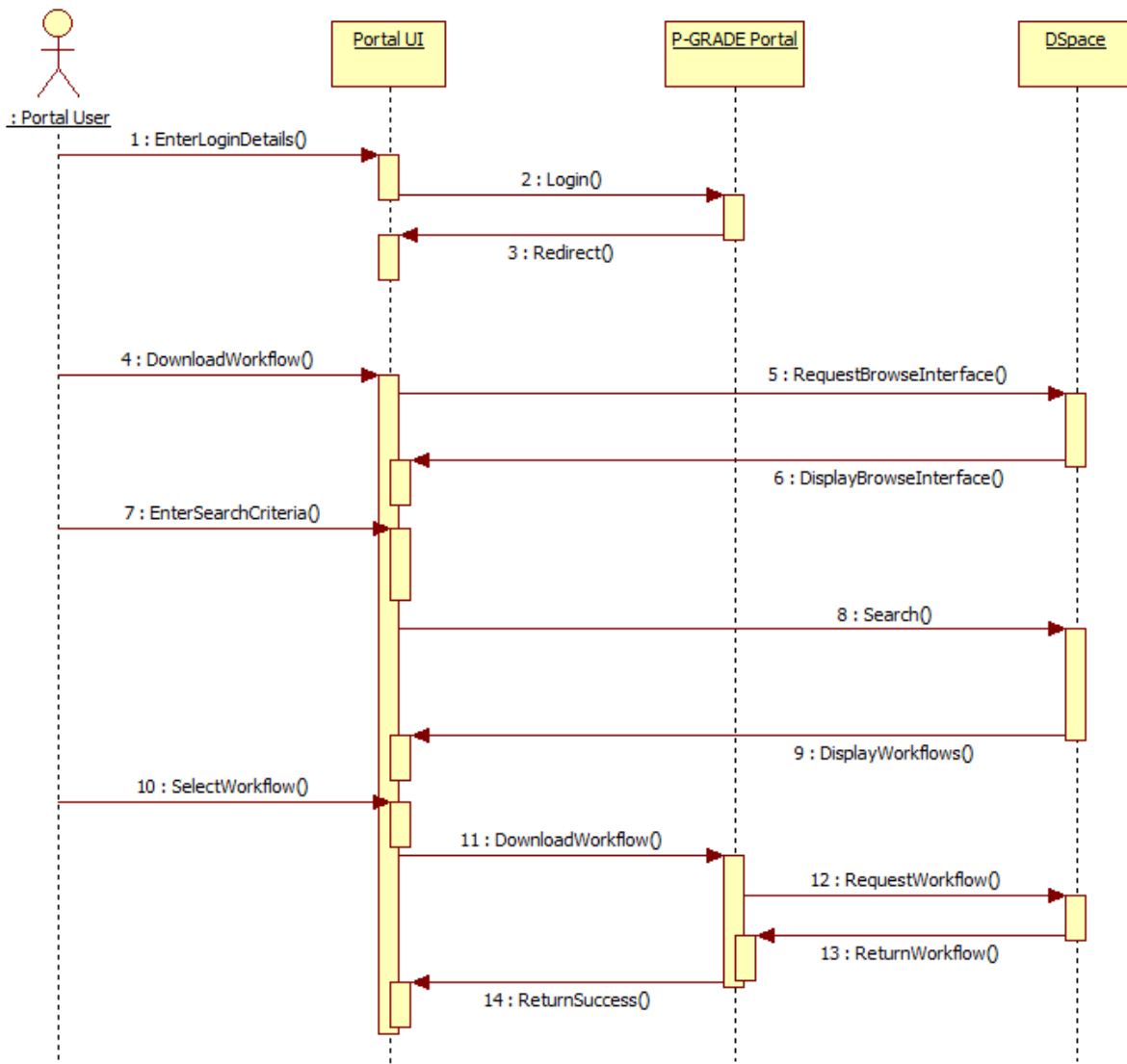


Figure 14 - Portal user downloading a workflow from DSpace to his/her Portal storage

2.4: DESIGN PLAN

The first phase of our design involved setting up a default DSpace installation and tailoring its layout and settings to meet our initial requirements. DSpace provides the ability to create what are referred to as “collections,” which are different categories into which users may upload workflows. For this project, we created collections called “Application Workflows,” “Jobs (Executables),” and “Results.” These were the collections used in all interactions between the P-GRADE Portal and DSpace. The Application Workflows collection was for any workflow a user wishes to upload. The Jobs collection was for single jobs (a node in the workflow). The Results collection was for storing the results of a workflow application that has been executed.

Phase two of our integration efforts to incorporate the DSpace repository technology with the P-GRADE Portal required allowing the Portal users to upload and download application workflows to and from their workspaces. After much analysis of documentation and source code for DSpace and the P-GRADE Portal, we identified the following issues which had to be addressed in the design of the second phase of our integrated system.

First, authentication was an important factor since there are multiple ways which DSpace can authenticate users and create DSpace user accounts whereas the P-GRADE Portal has its own user account and authentication methods. This means that there could be separate authentications set up for DSpace and the P-GRADE Portal, the use of certificates, the potential for X.509, or other methods for authenticating users for the integrated system. There were three potential solutions to this problem. The first was to take the certificates that the P-GRADE Portal uses for authentication and use them with DSpace as well. Another would be to automatically authenticate the user in DSpace if they were currently logged into the P-GRADE Portal by storing and associating DSpace usernames and passwords with Portal users, then passing that information along to DSpace. A third solution, which is a simpler implementation of the previous, and the one we decided to use, was to simply require a Portal user to provide a DSpace username and password each time they wish to upload a file to DSpace.

Another design-related issue dealt with accessing a Portal user’s space on the P-GRADE Portal in order to upload and download workflows to and from the repository. The DSpace system provides the ability to browse directories on a local machine in order to locate a file to upload, or specify a directory to download to. This posed a problem because Portal users need

the ability to upload and download workflows from and to remote storage space on the P-GRADE Portal server which can only be accessed by P-GRADE. This issue stemmed from the fact that the DSpace and the P-GRADE Portal would run on separate servers, while both had their own server-side APIs. The two systems would run independent of each other, but there was a need for coordination to make the integration of DSpace successful. One solution to this problem was to retrieve the user's workflows and select the desired workflow for upload using the P-GRADE Portal API, and then store the files for upload in a temporary directory which is known and accessible by DSpace.

However, the method we decided to pursue was to use a technology called Lightweight Network Interface (LNI). This technology provides a limited but simple remote API that maps to the DSpace Object API, giving access to basic functionalities required for integration with the P-GRADE Portal [24]. LNI provides the ability to build an application which can remotely access integral parts of the DSpace API.

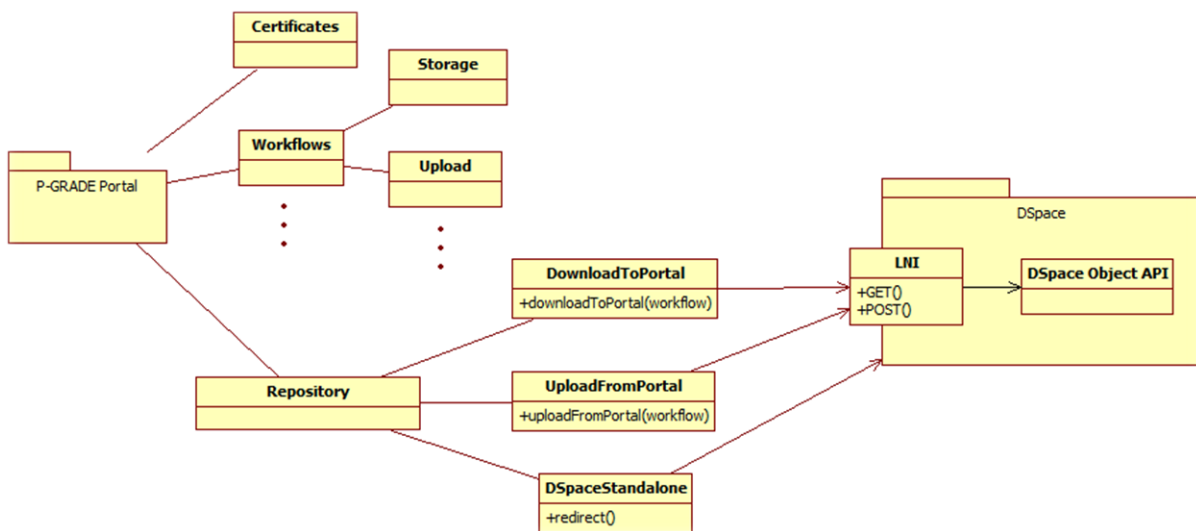


Figure 15 - Design plan for integration of DSpace repository and P-GRADE Portal

Therefore, the design plan for phase two of our integration efforts involved constructing a new “tab” for the P-GRADE Portal called Repository, in which we would create three portlets: Upload (from Portal), Download (to Portal), and DSpace View (the DSpace standalone application). The Upload and Download portlets would use LNI to interface with our DSpace installation, allowing Portal users to upload/download workflows from/to their Portal storage space. The DSpace installation would still serve as a standalone application for regular users and could still be used by Portal users as a standalone application – the DSpace View portlet would redirect Portal users to the DSpace Web UI. This design solved our identified issues: LNI provides the ability to pass a DSpace username and password with calls to the DSpace API. Thus a Portal user would be able to supply his or her DSpace login information, which could be passed through LNI to authenticate that user with DSpace. This also fixed the problem of accessing a Portal user’s workflows. Since LNI was installed with DSpace and accessed from a P-GRADE Portal portlet, a user’s workflows could be retrieved using the P-GRADE Portal API methods and then uploaded using the LNI API. As mentioned before, it was still possible to use DSpace as a standalone application with this implementation, although Portal users were also provided with basic DSpace functionalities needed for phase two of our requirements.

LNI ingests data to be stored in the DSpace repository in the form of Submission Information Packages (SIPs) [8]. A SIP contains a manifest, digital objects (i.e. the workflow), and any associated metadata. Thus the Upload portlet was to be implemented as a form that requested all the relevant information that the DSpace standalone application would normally require a user to provide. Specifically, the Upload portlet would provide the following:

- A space to enter the user’s DSpace username and password
- A list of workflows in the user’s Portal space, from which one may be chosen for upload
- A space to enter the name of the author(s) of this workflow
- A space to enter the name of the workflow
- A space to enter any keywords/tags to be associated with the workflow
- A space to enter any specific Grids the workflow was designed for
- A space to enter any specific VOs the workflow was designed for

- A list of possible workflow types to select from (“Full application”, “Template”, “Abstract Workflow”)
- A list of languages to select from that signifies what language the workflow documents are in
- A space to enter a brief abstract for the workflow
- A space to enter a more detailed description of the workflow

The information gathered from this form would be used to create a DSpace SIP and then sent to DSpace using LNI.

3: IMPLEMENTATION

In this stage of the project, we sought to implement a working system that integrated the functionalities of DSpace with the P-GRADE Portal. We first installed and configured DSpace as a standalone application on a server that would be dedicated to the repository. We then installed the latest version of the P-GRADE Portal (version 2.7) on another server, which gave us a sandbox environment where we could develop extensions to the Portal that would add the functionality of DSpace. Three new portlets were created for the P-GRADE Portal: one for navigating the DSpace website, one for downloading a workflow from DSpace to the Portal, and one for uploading a workflow from the Portal to DSpace.

3.1: DSPACE

The DSpace repository was installed on its own server. As per the DSpace manual [6], the following software was installed on the DSpace server:

- Scientific Linux 4.7 (<https://www.scientificlinux.org/>)
- Java JDK 1.6.0_12 (<http://java.sun.com/>)
- Apache Maven 2.1.0 (<http://maven.apache.org/>)
- Apache Ant 1.7.1 (<http://ant.apache.org/>)
- PostgreSQL 8.3.604 (<http://www.postgresql.org/>)
- Apache Tomcat 5.5.27 (<http://tomcat.apache.org/>)
- Perl 5.8.5 (<http://www.perl.org/>)

The DSpace system was then configured to use the appropriate server hostname and mail server settings. By default, DSpace is intended to be used with a Handle Server, and links to items within the repository using the Corporation for national Research Initiatives' Handle System (<http://www.handle.net/>). To avoid confusion for users, the system was configured to link to items within the repository using the URLs that point to their actual locations on the DSpace server.

We also customized the header file that appears atop all pages on the DSpace website to display a special logo for the P-GRADE Portal which was designed by Róbert Lovas of the LPDS. An image of the landing page for the DSpace repository can be seen in Figure 16.

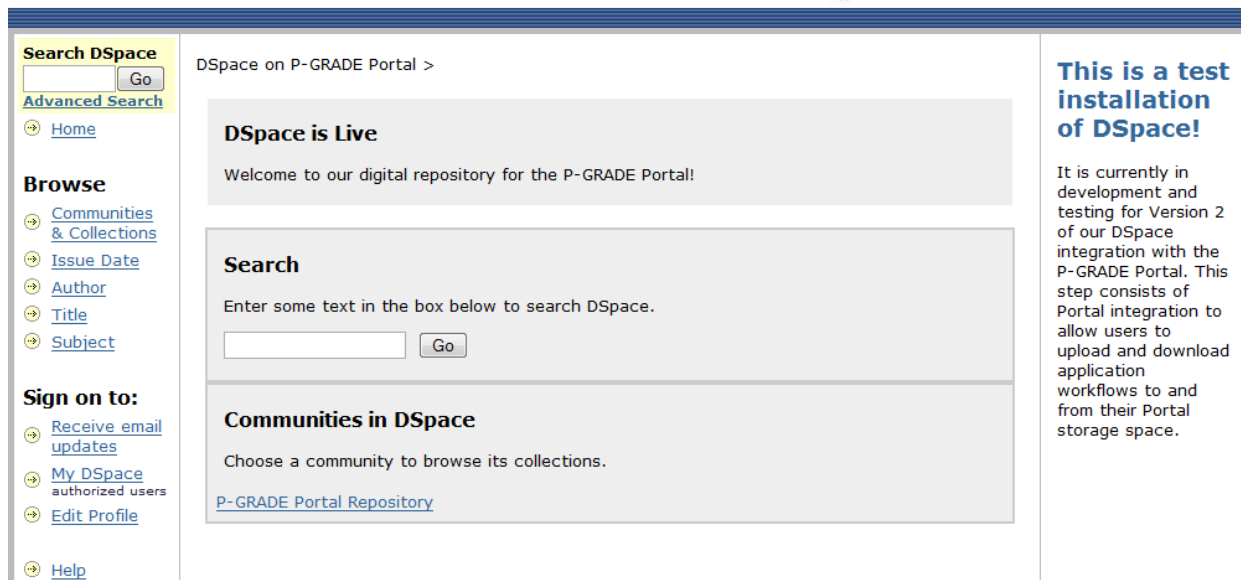


Figure 16 - DSpace repository for the P-Grade Portal

After the testing of our DSpace installation was complete, the system was configured to use an external database, as this was to be used in the production installation.

3.2: P-GRADE PORTAL

On another server, the P-Grade Portal v2.7 system was installed as a testing ground for our implementation efforts. The P-Grade Portal was developed using JSP technology, and we had to make use of this technology in order to integrate the DSpace repository with it. Three JSP files were created, one for each of three new portlets we developed for the Portal: a DSpace View portlet, a Download portlet, and an Upload portlet. Three corresponding Java files were also created to handle the backend logic associated with the functionality of these portlets. The DSpace View portlet was only intended for convenience: it displays the DSpace website within a frame, so that users may browse DSpace without leaving the P-Grade Portal. The Download and Upload portlets were created to do what their names imply: download and upload workflows from/to the DSpace repository.

Two other Java resources were used to facilitate the integration of DSpace with the P-Grade Portal. One was the DSpace SIP Toolkit [8], which provided a method for easily

producing valid SIPS (see section 2.4). The other, which was most critical to the communication between the P-GRADE Portal and DSpace, was the Simple LNI Client, developed by Larry Stone [35]. This LNI client implements parts of the LNI API (see section 2.4) to provide POST and GET methods for uploading and downloading items to and from the DSpace repository. A diagram displaying how the system was implemented can be seen in Figure 17.

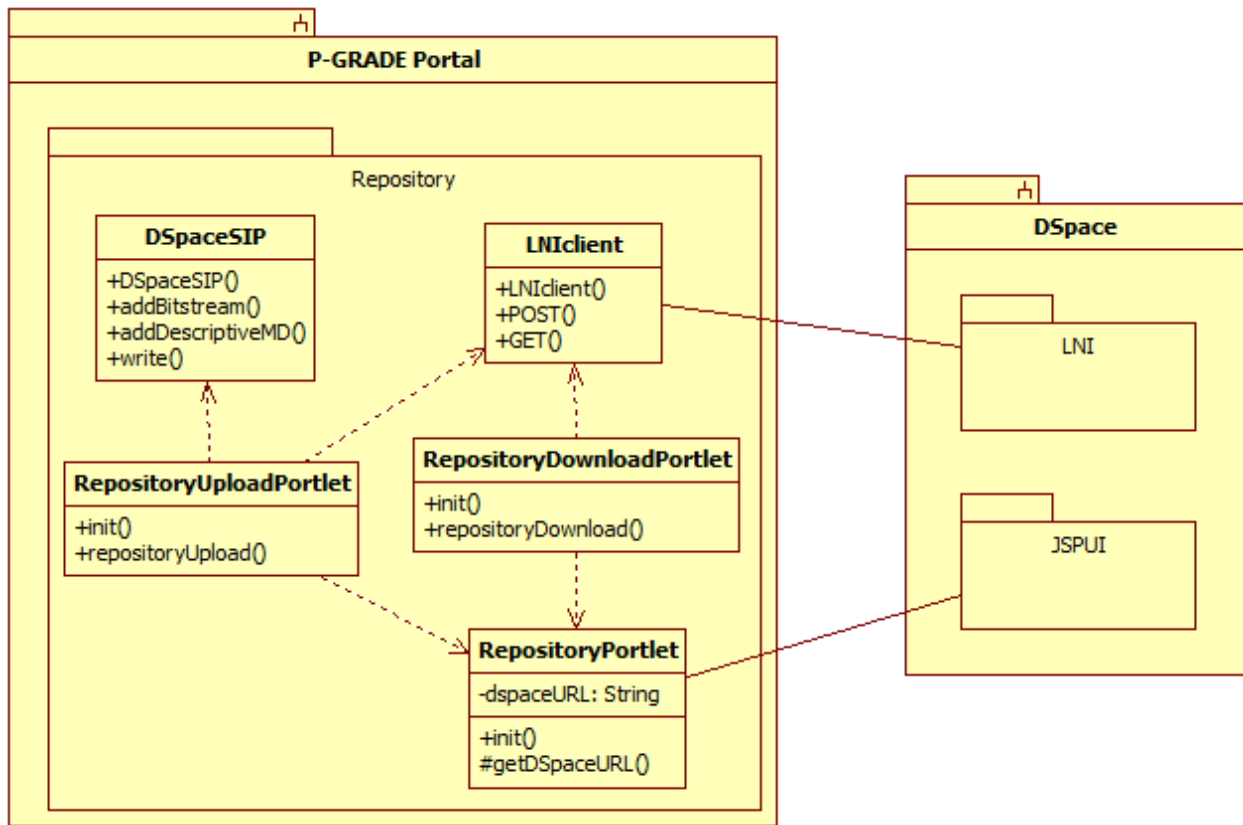


Figure 17 - Implementation of integration of the P-GRADE Portal and DSpace repository

To display our new portlets, 2 XML files were edited: one that controls the individual portlets and one that controls the website's tabs and which portlets are displayed in them (as seen in Figure 1 of Section 1.4). A new tab called DSpace Repository was created, under which two more sub-tabs were created: Download/DSpaceView and Upload. This new tab can be seen in Figure 18. Within the Download/DSpace View sub-tab, we placed the Download portlet at the top, with the DSpace View portlet underneath. The reason for this was to incorporate the browsing and searching functionalities of DSpace as easily as possible, such that users can find the information needed for the Download portlet in a simple manner.

3.2.1: DSPACE VIEW PORTLET

The DSpace View portlet was the simplest to create. This portlet displays the actual DSpace web interface within a frame the P-GRADE Portal. We accomplished this by creating a JSP page that contained a single MessageBean. When the Download/DSpaceView page is requested, the server initializes the DSpace View portlet and sets the value of this bean to HTML code that will display the DSpace website using an iframe. This allows users to access all the functionalities of the DSpace repository without ever leaving the P-GRADE Portal website.

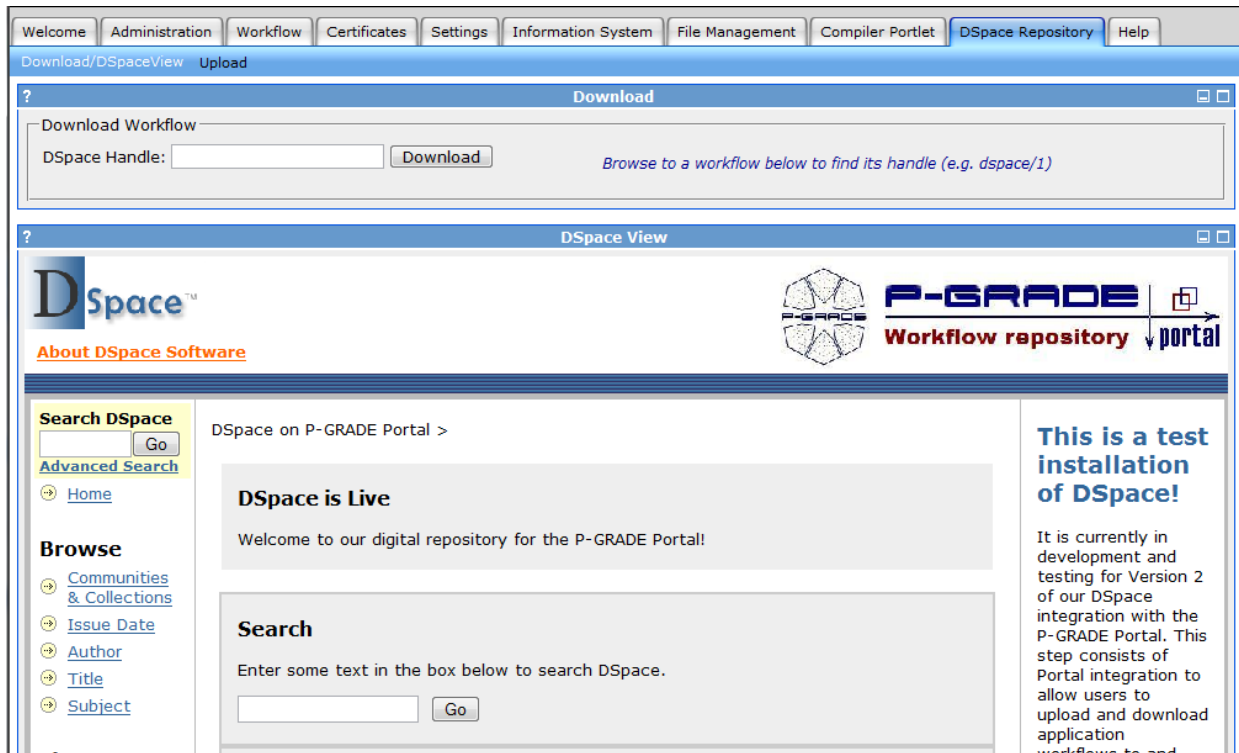


Figure 18 - DSpace View and Download Portlets

3.2.2: DOWNLOAD PORTLET

Due to limitations in time and the abilities of the LNI client we used, the implementation of the Download portlet was quite elementary. This portlet provides a text field in which the user may enter the handle of a DSpace Item he/she would like to download to his/her Portal storage space, a button to click that will submit the request, and a MessageBean in which to display messages to the user (instructions, error messages, etc.). This portlet is displayed above the DSpace View portlet, giving the users easy access to the browsing abilities of DSpace. The

only drawback of this method is that once a user has located the workflow to download to his/her workflow space, that user must physically copy the DSpace Item's handle and paste it into the text field on the Download portlet. While this method was admittedly a bit crude, it allowed us to provide the users with all the functionality of DSpace's repository browsing utilities within a short timeframe.

When a handle is submitted by the user, the portlet calls the LNI client's startGet() method, which will find the Item on DSpace associated with the given handle, if it exists, and return a data stream carrying the Dissemination Information Package (DIP) [32]. A DIP is an archive file that contains a compressed file (e.g. a workflow), an XML file representing any metadata associated with the Item, and the license agreement that the author agreed to when the Item was originally uploaded to the repository. The compressed workflow is the file of interest, so that item is extracted from the DIP. The workflow is then decompressed, and stored in the user's Portal storage space.

The Download Portlet can be seen above in Figure 18.

3.2.3: UPLOAD PORTLET

The Upload portlet was more difficult to implement than we originally evaluated it to be, and thus required the most of our time to make it functional. This portlet displays a simple form which requests information from the user that is typical of the DSpace upload process, but laid out in a way that best suits the P-GRADE user. The first section of the form requires that the user enters the e-mail address and password for his/her DSpace account. Below that is a list of the user's workflows on the Portal, from which one must be selected. The next section requires that the user enter the author of the workflow they wish to upload (presumably the user). Following that is a section that provides a space to enter the title of the workflow, a comma-separated list of keywords/tags to be associated with the workflow, any Grids or VOs the workflow was designed for (which will also be passed along as keywords/tags), the type of workflow that is being uploaded ("Full application," "Template," "Abstract workflow," or "Other"). Finally, there are two more fields to enter information about the workflow: one is a text area for an abstract summary of the workflow; the other is a text area for a more detailed description of the workflow.

When the information on the form is submitted, the portlet uses the DSpaceSIP class to create a SIP. The workflow that was selected is then located in the user's workflow space and compressed into a single archive file, which is added to the SIP. The difficulty in implementing this portlet came with representing the information collected from the form as metadata. The DSpaceSIP represents metadata in an XML file that follows the Library of Congress's Metadata Encoding and Transmission Standard (METS). METS is used for "encoding descriptive, administrative, and structural metadata regarding objects within a digital library" [26]. Within the METS document, the metadata gathered from the user-submitted form is represented. However, at the time of implementation there was no standard XML schema for defining the elements of Dublin Core (DC), which is what DSpace uses for storing metadata in the repository. For this reason, LNI expects an XML file that uses the Metadata Object Description Schema (MODS) [27] instead. For our first attempt at implementing this portion of the upload process, we used a chart that displays how various elements of MODS map to elements of DC [28]. However, certain elements that we required could not be mapped from MODS to DC, thus another method had to be used. What we found was not another method entirely, but rather an extension of the MODS to DC mapping. An e-mail in the DSpace developers Email Archive described the same issues we were having with our implementation, and gave a solution that involved editing a file within the DSpace system that converts the XML file using MODS to one that uses a Dublin Core schema [11]. Leveraging this method, we were able to successfully create a valid METS XML file using MODS to represent the metadata elements we desired.

After the METS file is completed and added to the SIP, the LNI client is initialized with the given user credentials, and the startPut() function is called, which transfers the constructed SIP to DSpace and adds it to the Application Workflows Collection. A link is then provided to the user displaying the location of the new submission on DSpace.

Welcome	Administration	Workflow	Certificates	Settings	Information System	File Management	Compiler Portlet	DSpace Repository	Help
---------	----------------	----------	--------------	----------	--------------------	-----------------	------------------	--------------------------	------

Download/DSpaceView Upload

Upload

DSpace Login

*E-mail: *Password:

UPLOAD SUCCESSFUL!
[CLICK HERE TO VIEW THIS ITEM ON DSPACE](#)

[Click here to register for a DSpace account](#)

Upload a Workflow

Select	Name
<input type="radio"/>	Ax_EQUAL_B_seegrid_gLite_broker
<input type="radio"/>	LM_9_DEMO-fromscratch
<input checked="" type="radio"/>	MatrixTest_fromscratch

Author

*First: *Last:

Information

*Title:

Keywords: Comma-separated list (e.g. Keyword1, Keyword2)

Grid: (e.g. EGEE, SEE-GRID)

VO: (e.g. GILDA, Seismology VO)

Type: ▼

Language: ▼

Abstract

Figure 19 - Upload Portlet

4: TESTING

Testing was a vital part of our repository integration with the P-GRADE Portal and a part of the project which we were careful not to overlook or take lightly. At the beginning of the design stage of our project, we decided that we would adopt an iterative development and testing approach. This meant that each individual step of development would be tested before proceeding to the next step. Furthermore, each phase of the project was thoroughly tested before progressing to the next phase. The iterative development and testing approach was favored and turned out rather successful for two main reasons. First, it ensured that the system testing was not overlooked or omitted due to a lack of time at the end of the implementation or that a specific aspect would not be tested since it was forgotten about by the time the testing phase occurred. Also, this approach was beneficial since most development on the project relied heavily on the previously developed parts functioning and communicating properly.

As far as the type of testing is concerned, our group mainly utilized manual testing. Our particular repository system set itself up for manual testing since there was complex communication between the two servers and various APIs along with functionalities that truly relied on human interaction for testing. Therefore, most testing of our integrated repository system consisted of the group implementing a certain functionality or part of the system and then posing as a system user or administrator to verify that it was functional and met our requirements. In order to make this manual testing possible, the group was provided with our own sandbox P-GRADE Portal installation where we could edit the source code for development and utilize for testing purposes without risk of damaging the production deployment of the P-GRADE Portal software.

One other important testing note is that the group heavily relied on logging and debugging functionalities provided by Tomcat's output log file called Catalina.out. Catalina.out provided us with the ability to write status and debugging information to file using Java's standard output, which was often analyzed to detect errors and locate problematic sections of code. Status messages were also useful to ensure various functions were executed and that the proper information was retrieved or sent by the system.

4.1: DSPACE CONFIGURATION TESTING

After the initial installation of DSpace for phase one of our system integration, it was necessary to set up the repository structure that we desired. This required creating a community and some collections, as well as adjusting the settings pertaining to this repository infrastructure. After this was accomplished, it was necessary to test the DSpace installation to make sure a local user could interact with DSpace and upload or download files. Also, basic administrative settings were tailored at this phase of the project, and these settings and permissions were tested as well to validate that administrative tools were functioning properly and that user accounts and groups were behaving as expected.

Phase one of the project also entailed customizing the look and feel of DSpace, particularly the descriptions that accompany the various pages and sidebars in the DSpace user interface. All of these customizations had to be viewed and tested to ensure proper display and readability. Also from a usability perspective, it was important to test and ensure that descriptions were meaningful and conveyed the necessary information to the end user. To accomplish this phase of testing, we asked a few members of the LPDS to interact with our installation of DSpace and give us feedback on how intuitive the user interface was, and where they thought it could use improvement.

Finally, all of the aforementioned testing for the basic DSpace installation was part of a larger requirements verification that we performed at the completion of phase one implementation. This requirements verification entailed testing every requirement that we set forth for the first phase of our system and ensuring that it was functional and behaved as we had expected.

4.2: PORTAL INTEGRATION TESTING

The first step in the Portal integration involved creating the necessary JSP and Java files along with customizing the P-GRADE Portal portlet and layout XML files to incorporate our new portlets into the Portal architecture. Therefore, after changing the XML files and after significant layout changes to the JSP files, we ensured that the three project-related portlets and DSpace Repository tab were displaying correctly and provided a user-friendly layout for Portal users. Afterwards, the actual business logic was added to process the information gathered from these portlets, and thus there was corresponding testing for changes to this logic as well. After

all changes to form processing or other phase two specific implementation tasks were complete, it was important to analyze the uploading, downloading, and DSpace related functionalities that might have been affected. This testing step involved attempting to upload or download a workflow, or performing interactions with DSpace from within the P-GRADE Portal. The group found this testing process particularly fruitful in identifying small errors in logic or information transmission.

Following the implementation of all phase two related functionality, requirements verification was performed again with a focus on phase two specific requirements. This involved conducting full use case realizations for Portal users uploading one of their workflows to DSpace or downloading a workflow to their Portal storage space. Testing Portal user functionality was perhaps the most essential testing for the project since the Portal users were the main target audience of this repository integration with the P-GRADE Portal. Again, we employed the help of LPDS members for testing the usability of this phase of the system.

4.3: PRE-DEPLOYMENT TESTING

After all implementation was concluded, a final pre-deployment testing phase was needed to ensure a high level of usability and performance for our integrated repository. The first part of this testing stage focused on checking the robustness of the system. Therefore, it was necessary to perform use cases and intentionally induce errors in various parts of the processes to analyze the error checking and handling provided by the system. By inducing errors in testing, we were able to determine if users were given proper warnings and help while interacting with the application. Also, it was imperative to make certain that user errors or other potential system errors would not lead to disruption of normal system operations or internal server errors.

Following the testing of error checking and handling, it was necessary to conduct full scale testing of the complete system and our test Portal setup since this was the final implementation that would be added to the live P-GRADE Portal software. This full Portal integration testing stage involved creating user accounts, testing all applicable administration features, and performing full use cases that a typical user would encounter when interacting with our system. Also, to avoid any testing or usability bias related to our development of the system, we also located various third party testers to serve as potential users or administrators. These testers were selected from our MTA SZTAKI colleagues and a few outside parties based on who

would likely be administrating our system and who would be potential users. Choosing these additional testers and conducting these third party tests ensured that our integrated repository system could be used by Portal users without significant technical knowledge. We also made sure that potential P-GRADE Portal administrators could easily understand and perform the necessary tasks that they would need to maintain the system.

Finally, after comprehensive testing was completed, we analyzed the statistics that DSpace provides to better visualize the results gathered from testing of the system and the user interactions which occurred with it. A portion of these results can be seen in the following simple table which further serves to validate the functionality and success of our repository integration with the P-GRADE Portal.

Action	Number of times
Item Updated	84
browse	68
Community List Views	57
Bitstream Views	54
Workspace Item Views	51
browse_by_item	47
browse_mini	46
Collection Views	36
authenticate	33
Bitstream Updates	30
Item Views	29
browse_by_value	24
Bundle Updates	22
User Home Page Views	22
Bundles Added	16
EPerson Record Updated	14
User Logins	13
Community Views	10
User Logouts	9
Bitstreams Added	8
Bitstreams Created	8
Bundles Created	8
User Profile Views	7
Workspace Items Created	7
Registration Tokens Sent	7

Table 2 - DSpace Usage Statistics

5: CONCLUSIONS

In this project, the DSpace repository was successfully integrated with the P-GRADE Portal. This was accomplished by creating a new layout tab in the Portal, along with three new portlets and their corresponding business logic to handle interactions with DSpace and uploading or downloading to/from a user's Portal storage space. This repository system was built as an extension to the P-GRADE Portal architecture and mostly required adding files to the existing hierarchy while leaving the rest of the system untouched. The exception to this was the necessity to change two of the XML files in order for the Portal to display the DSpace Repository tab and corresponding JSP files which were added for repository integration.

5.1: DSPACE REPOSITORY

First, the DSpace repository was successfully implemented as a standalone application and still can be utilized as one in our final implementation. This added important flexibility to the system, allowing both non-Portal users and Portal users to access DSpace from outside the P-GRADE Portal with the ability to browse and download workflows to their local machine as an anonymous user. Portal users can also log in to this standalone DSpace installation, assuming they have a DSpace login already set up and could browse or upload and download workflows using their local machine.

While this flexibility was a beneficial feature, the more important accomplishment was the integration of DSpace with the P-GRADE Portal, allowing Portal users to access DSpace from within the DSpace Repository tab on the Portal. This DSpace view within the Portal was accomplished by embedding DSpace as an iframe within the DSpaceView portlet JSP, providing the user with convenient access to DSpace right from the P-GRADE Portal. Also, displaying DSpace within the P-GRADE Portal was imperative to provide the download functionality for Portal users, allowing them to locate the desired item handle within DSpace which can be easily copied into the download bar at the top of their screen.

5.2: PORTAL DOWNLOAD

As mentioned above, the Portal download bar was a simple but effective portlet displayed directly above the DSpace view within the DSpace Repository tab. While initial design plans had called for a browse and search feature as part of the download portlet, time and LNI client

restrictions forced the group to utilize this simple, yet fully functional method for downloading workflows to a user's Portal storage. However, since DSpace provided excellent browsing and searching capabilities, no functionality is lost by utilizing this method of displaying the DSpace view below the download bar. Also, since this is a simple and straightforward procedure to download a workflow, complete with an instruction message in the download bar, we feel that it is not a major limitation.

5.3: PORTAL UPLOAD

The upload portlet was implemented as a single form rather than the multi-page form that DSpace utilizes, although all information that is pertinent to the P-GRADE Portal workflows is retrieved using this form. Also, the available workflows list was automatically populated from the Portal user's storage directory and certain information such as the user's name or email were automatically filled out in the form if they were readily available. Additionally, on a failed submission attempt, most form data was preserved and an appropriate error message was given to the user. This form and its features were designed in such a way as to provide an easy and user-friendly approach for uploading a workflow to DSpace from Portal storage, thereby helping users to avoid discouragement from having to complete a tedious and difficult process. Furthermore, upon completing a successful upload, the Portal user is returned a link to the item in DSpace so that they can view the workflow and make any edits to the metadata or other information if necessary.

While the upload portlet was the most difficult and time consuming to implement due to complex server and API communications needed for the P-GRADE Portal to interface with DSpace, it is also the most vital part of the repository integration with the Portal. Without this portlet, Portal users would be forced to work with the standalone DSpace installation and complete a long and complex process of downloading their workflow from Portal storage to their local machine and then uploading that using the standard DSpace upload process. Instead, Portal users are presented with a readily accessible and much more straightforward process that is tailored to the needs of the P-GRADE Portal and its users.

5.4: P-GRADE PORTAL INTEGRATION

Given the success of this project and the help of our SZTAKI colleagues, workflow repository integration has been included in the current version of the P-GRADE Portal and is now available to all of its users. Not only is it beneficial that the functionalities of the P-GRADE Portal have been expanded by this integration, it is also a significant step in the direction of collaborative development among Portal users. The rationale behind this project approach was that by providing Portal users with user-friendly, P-GRADE Portal tailored repository functionalities, those users would be more likely to actively share their application workflow projects and knowledge while benefiting from others as well. There is no doubt after extensively testing this system and observing our colleagues interacting with it that repository integration with the P-GRADE Portal will facilitate an increase in collaborative development and the dissemination of application workflow related information.

6: FUTURE WORK

After spending much time researching and brainstorming ideas for integration of a repository into the P-GRADE Portal, we identified a few features which would be a nice addition to our work. We anticipated that these features could not be implemented due to our very strict time constraints on this project and their increased level of complexity. However, we wanted to identify these anyway as an area of future work and improvements to our integration efforts.

6.1: COLLABORATIVE DEVELOPMENT

The access control policies detailed in the following sections would be especially helpful for providing greater support for collaborative development among P-GRADE users.

6.1.1: USER SPECIFIED WORKFLOW ACCESS PERMISSIONS

One additional feature to improve collaborate development support is to allow users to control their own workflow access permissions. This means that users could specify what user groups or individual users they would like to allow access to their workflow and what permissions they would like to allow these users or groups. Essentially this is enabling users to have permission administration control over their own workflows. This is especially important for collaborative development as it allows users to specify who can modify their workflow such as creating a team within the repository which can work on the workflow together. Furthermore, this would give users more control over their own workflows and decentralize some of the administration powers although administrators could still perform or override these permissions as well.

The implementation of this feature could be accomplished by creating a similar permissions policy editor to the one which the administrator has for setting the authorizations for a repository item. This would allow the owner of the workflow to act as an administrator of his/her own workflow and add or edit a policy which would allow other users or user groups to access or edit the workflow.

6.1.2: USER SPECIFIED WORKFLOW ADMINISTRATIVE RIGHTS

This feature is an extension of the feature above which would allow users to designate other users as an administrator for their workflow. This means that users could be granted administrative rights that normally only the user who owns the workflow and DSpace administrators would have such as editing tags, version, status, descriptions, etc. Adding this feature would allow a workflow owner to specify someone else as the workflow administrator or “lead developer” who could edit the workflow information in DSpace along with editing the files too. This is convenient because it allows other users to update workflow information and coordinate releases of a workflow rather than relying on the actual workflow owner having to perform all these actions for them.

Implementing this feature would mean extending the current DSpace access controls so that a user can set permissions upon workflow upload not only for downloading or editing the workflow, but also for changing the DSpace specific information on the workflow. One possible way to accomplish this would be to add an Administrative permission setting to the DSpace permissions list of Read, Write, Add, and Remove. This setting would allow the workflow owner to designate another user or user group as an administrator for his/her workflow.

6.1.3: CONTENT LOCKING

Another additional feature to enhance collaborative development is content locking for workflows. This feature would allow users to lock the contents of their workflow for exclusive editing. Content locking is very important since it would ensure that multiple users who are working collaboratively on a workflow are not editing the same workflow at the same time. Simultaneous editing by multiple users could lead to parallel versions and other issues stemming from one user overwriting or interfering with the other user’s work, potentially without even knowing they have done so.

The actual implementation for this feature would entail adding a button or setting to a user’s workflow information which the user could click or set to lock his/her workflow content for exclusive editing. Also, any other users with permission to edit the workflow should have the same ability to lock the content assuming that it is not already locked by another user. This means that when a user attempts to lock a workflow, there needs to be a check that it is not

already locked by another user and there should be a notification if the content is currently locked. Also, it might be helpful to auto generate an email to all users who have permissions to edit the given workflow, notifying them that the content has been locked by another user. Finally, content locking would necessitate that a workflow is checked to see if it is locked for editing before another user can modify it and an appropriate message should be given if a user attempts to download and modify a workflow that is locked by another user.

6.2: OTHER FUNCTIONALITIES

The features described below are currently implemented in other repository services and would be useful for DSpace/P-GRADE users.

6.2.1: VERSIONING

Content versioning in DSpace would prove quite useful, especially if some of the collaborative development features detailed in the previous sections were to be incorporated into the repository system. This would enable users to track changes in application workflows, and roll back to a previous version should a problem arise in the latest release of a workflow.

6.2.2: COMMENTING

Commenting is another interesting feature that was suggested by our SZTAKI colleagues. While we were not able to implement this due to time constraints, we did identify and briefly research a promising commenting add-on for DSpace [5]. This commenting add-on is released by the DSpace Dev group at the University of Minho. According to the documentation, the commenting add-on provides the functionality that we envisioned and appears to be relatively easy to incorporate via minor code modifications and rebuilding the DSpace installation. The main concern with this add-on is that it was developed for DSpace 1.1 and 1.2. There is mention that it may not work with previous or future versions of DSpace. Since we utilized DSpace 1.5.1, it would be necessary to test the commenting add-on with our version of DSpace to see if it could be used. If this add-on works with DSpace 1.5.1, it is highly recommended to install and incorporate this add-on as it provides the ability for users to give feedback on other user's application workflows and further expand the communication between users of the repository.

6.2.3: RATING

The myExperiment workflow repository [29], as well as many other user content-driven services, provides a rating system where users may vote on the quality of a particular entry. While the workflow repository on P-GRADE is not as focused on the social networking aspects of myExperiment, a rating system would help give the most intriguing and useful workflows recognition among researchers.

6.2.4: AUTOMATIC WORKFLOW SUBMISSION

This feature would provide similar functionality to the NGS workflow repository which allows users to submit their workflow right from DSpace to be run on the Grid. While this would likely be very hard to implement, it would greatly expedite the process of testing and executing workflows since a user could run a workflow right from DSpace without have to upload the workflow to a Grid storage directory from DSpace, log into the P-GRADE Portal, locate the workflow, and then execute the workflow the typical way one would do so using the P-GRADE Portal. Instead this could be implemented with a button next to the workflow listing in DSpace which would allow the user to click to submit the workflow to the Grid and then guide them through any necessary steps to run the workflow without leaving DSpace.

7: REFERENCES

- [1] ACS-WG. SourceForge. [Online] [Cited: 15 Apr. 2009] <<http://forge.org/sf/projects/acs-wg>>.
- [2] Archimède: A Canadian software solution for institutional repositories. *Laval University.* [Online] [Cited: 3 Apr. 2009] <<http://www.bibl.ulaval.ca/archimede/index.en.html>>.
- [3] Braverman, Amy M. "Father of the Grid." *The University of Chicago Magazine.* Apr. 2004.
- [4] Census of Institutional Repositories in the United States. *Council on Library and Information Resources.* [Online] February 2007. [Cited: 3 April 2009] <<http://www.clir.org/pubs/reports/pub140/pub140.pdf>>.
- [5] Commenting Add-on. DSpace Dev @ University of Minho. [Online] [Cited: 21 Apr. 2009] <http://dspace-dev.dsi.uminho.pt:8080/en/addon_commenting.jsp>.
- [6] DSpace 1.5.1 Manual. *DSpace Foundation.* [Online] 2008. [Cited: 3 April 2009] <http://www.dspace.org/1_5_1Documentation/>.
- [7] DSpace FAQs. *DSpace Foundation.* [Online] [Cited: 3 April 2009] <<http://www.dspace.org/index.php/FAQs/>>.
- [8] DSpace SIP Toolkit. *DSpace Foundation.* [Online] 25 June 2008 [Cited: 13 April 2009] <http://wiki.dspace.org/index.php/DSpace_SIP_Toolkit>.
- [9] DSpace. *DSpace Foundation.* [Online] [Cited: 3 April 2009] <<http://www.dspace.org/>>.
- [10] Dublin Core Metadata Initiative (DCMI). *Dublin Core Metadata Initiative.* [Online] 31 March 2009 [Cited: 3 April 2009] <<http://www.dublincore.org/>>.
- [11] Email Archive: dspace-devel. *Sourceforge.net.* [Online] 23 October 2008 [Cited: 20 April 2009] <http://sourceforge.net/mailarchive/forum.php?thread_name=7d8be55c0810170128nd31315t23f465db3feb5aca%40mail.gmail.com&forum_name=dspace-devel>.
- [12] Fedora Commons Community. *Fedora Commons.* [Online] [Cited: 3 April 2009] <<http://www.fedora-commons.org/community/>>.
- [13] Fedora Commons. *Fedora Commons.* [Online] 2 April 2009 [Cited: 3 April 2009] <<http://www.fedora-commons.org/>>.
- [14] Fedora Repository 3 Documentation. *Fedora Commons.* [Online] [Cited: 3 April 2009] <<http://fedora-commons.org/confluence/display/FCR30/Fedora+Repository+3+Documentation>>.

- [15] Foster, Ian, and Carl Kesselman, eds. The GRID: Blueprint for a New Computing Infrastructure. San Francisco: Morgan Kaufmann, 1999.
- [16] Foster, Ian, and Carl Kesselman, eds. The GRID 2: Blueprint for a New Computing Infrastructure. Amsterdam: Morgan Kaufmann, 2004.
- [17] Grid Computing Information Centre (GRID Infoware). *GRID Infoware*. [Online] [Cited: 2 April 2009] <<http://www.gridcomputing.com>.>
- [18] Joseph, Joshy, and Craig Fellenstein. Grid computing. Upper Saddle River, N.J: Prentice Hall Professional Technical Reference, 2004.
- [19] JSR 168: Portlet Specification. *Java Community Process*. [Online] 6 February 2009 [Cited: 3 April 2009] <<http://www.jcp.org/en/jsr/detail?id=168>.>
- [20] Kacsuk, Péter, and Gergely Sipos. "Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal." *Journal of Grid Computing* (2006).
- [21] Kacsuk, Péter, Gergely Sipos, Adrián Tóth, Zoltán Farkas, Gábor Kecskemáti, and Gábor Hermann. "Defining and Running Parametric Study Workflow Applications by the P-GRADE Portal." *Cracow Grid Workshop'06* (2007): 75-83.
- [22] Kacsuk, Péter. "Towards a World Wide Grid: Integrating Service Grids and Desktop Grids." *Parallel, Distributed and Grid Computing for Engineering*. Ed. Topping, B.H.V., and P. Iványi. Stirlingshire: Saxe-Coburg Publications, 2009. 49-82.
- [23] Laboratory of Parallel and Distributed Systems. *MTA SZTAKI*. [Online] [Cited: 3 Apr. 2009] <<http://www.lpds.sztaki.hu/>.>
- [24] LightweightNetworkInterface. *DSpace Wiki*. [Online] [Cited: 15 Apr. 2009] <<http://wiki.dspace.org/index.php/LightweightNetworkInterface>.>
- [25] Main Page - myExperiment. *University of Manchester and University of Southampton*. [Online] [Cited: 03 April 2009. [Cited: 3 Apr. 2009] <http://wiki.myexperiment.org/index.php/Main_Page.>
- [26] Metadata Encoding and Transmission Standard. *The Library of Congress*. [Online] 17 April 2009 [Cited: 20 April 2009] <<http://www.loc.gov/standards/mets/>.>
- [27] Metadata Object Description Language. *The Library of Congress*. [Online] 13 March 2009 [Cited: 20 April 2009] <<http://www.loc.gov/standards/mods/>.>
- [28] MODS to Dublin Core Metadata Element Set Mapping Version 3.0. *The Library of Congress*. [Online] 7 June 2005 [Cited: 20 April 2009] <<http://www.loc.gov/standards/mods/mods-dcsimple.html>.>
- [29] MyExperiment. *University of Manchester and University of Southampton*. [Online] [Cited: 3 Apr. 2009] <<http://www.myexperiment.org/>.>

- [30] Open Archives Initiative Protocol for Metadata Harvesting. *Open Archives Initiative*. [Online] [Cited: 3 April 2009] <<http://www.openarchives.org/pmh/>>
- [31] OpenURL. *Ex Libris*. [Online] [Cited: 3 April 2009] <<http://www.exlibrisgroup.com/category/sfxopenurl>>.
- [32] PackagerPlugins. *DSpace Foundation*. [Online] 30 January 2007 [Cited: 20 April 2009] <<http://wiki.dspace.org/index.php/PackagerPlugins>>.
- [33] P-GRADE Grid Portal. *MTA-SZTAKI*. [Online] 2007. [Cited: 3 April 2009] <<http://portal.p-grade.hu>>.
- [34] RESTful Web Services. *Developer Resources for Java Technology*. [Online] [Cited: 15 Apr. 2009] <<http://java.sun.com/developer/technicalArticles/WebServices/restful/>>.
- [35] Simple LNI Client. *DSpace Foundation*. [Online] 25 June 2008 [Cited 22 April 2009] <http://wiki.dspace.org/index.php/Simple_LNI_Client>.
- [36] SOAP Tutorial. *W3Schools Online Web Tutorials*. [Online] [Cited: 15 Apr. 2009] <<http://www.w3schools.com/soap/default.asp>>.
- [37] SZTAKI: Department Information. *MTA SZTAKI*. [Online] [Cited: 3 Apr. 2009] <<http://www.sztaki.hu/department/LPDS/>>.

APPENDIX A: USE CASE DIAGRAMS

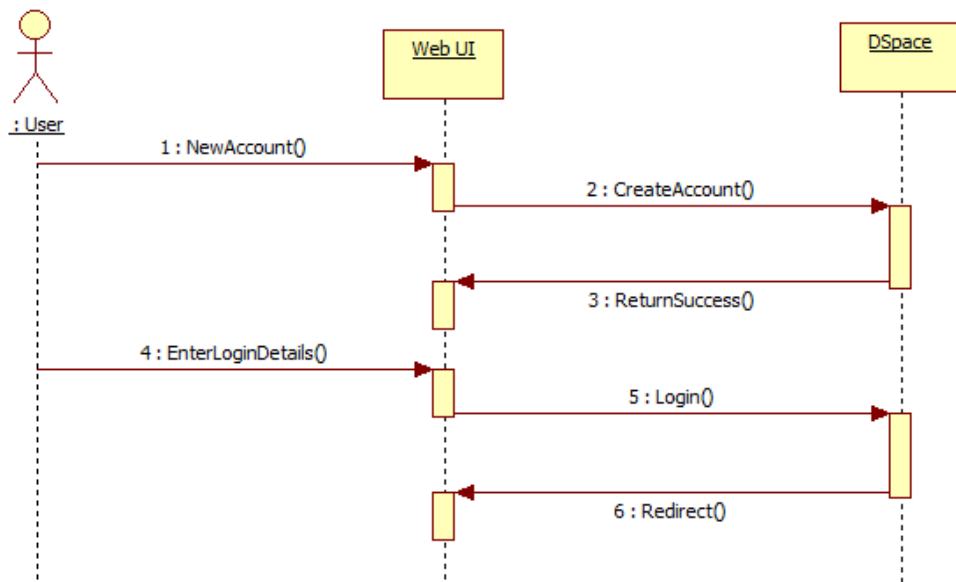


Figure 20 - DSpace user creating an account

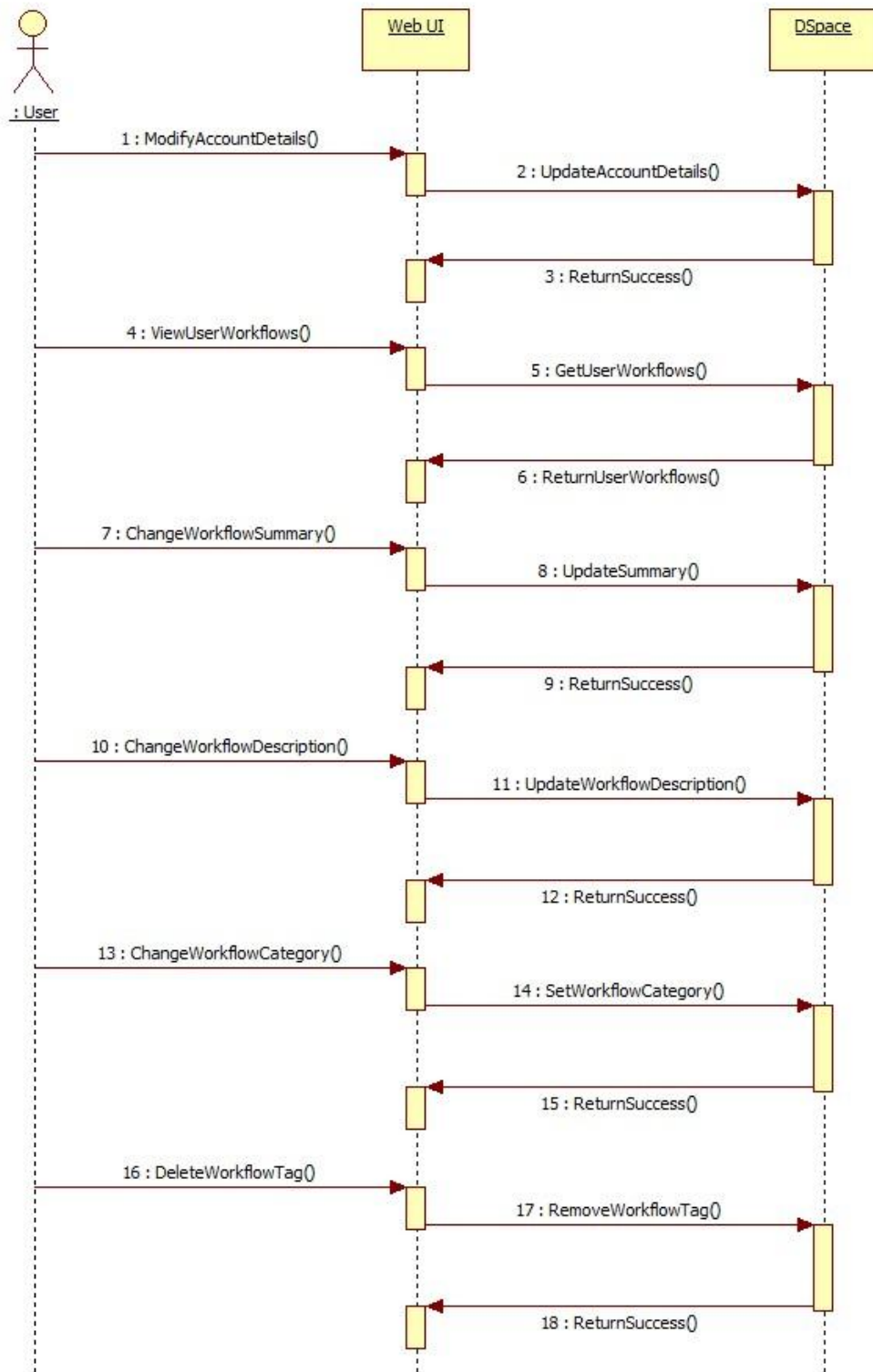


Figure 21 - DSpace user editing account and workflow details

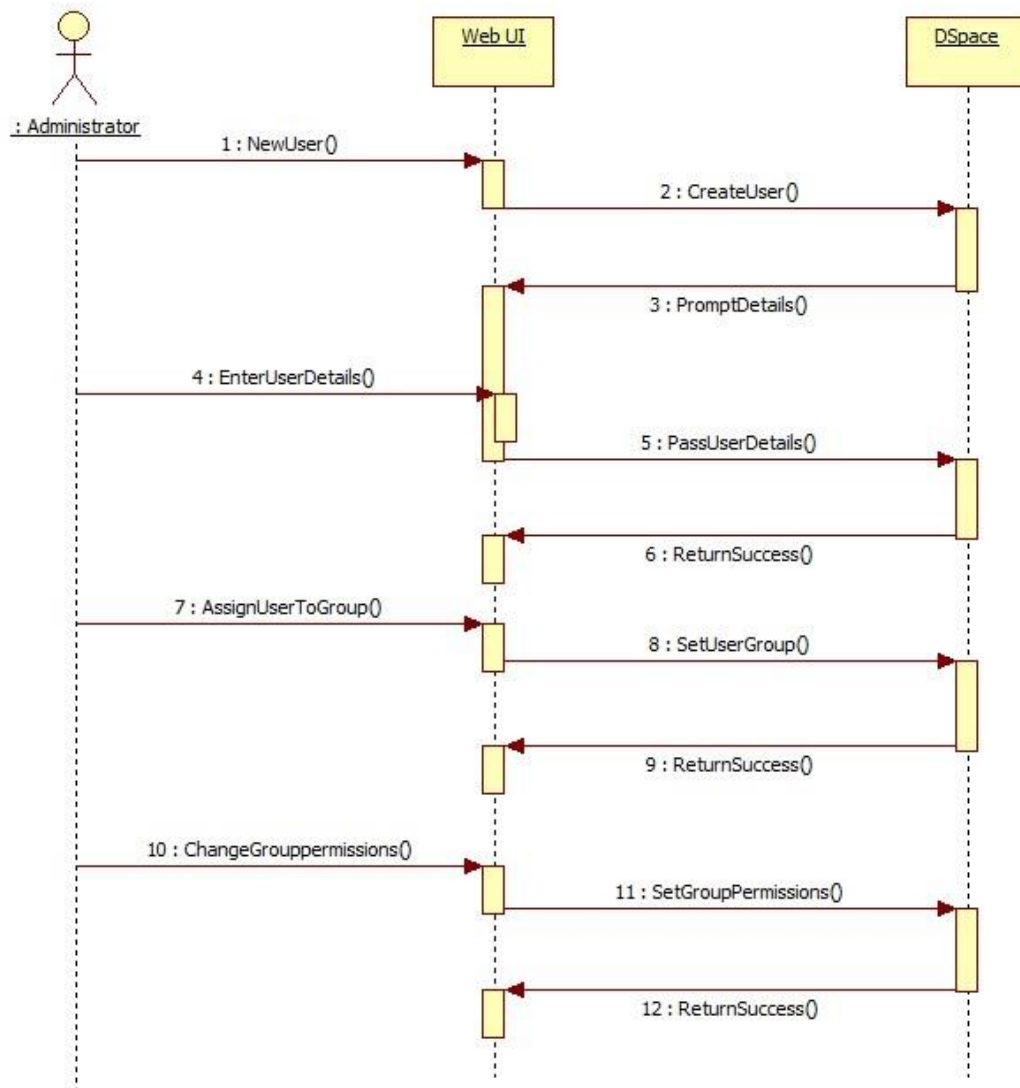


Figure 22 - Administrator creating a new user and editing group details

APPENDIX B: USER MANUAL

P-GRADE Portal – DSpace Repository User’s Manual



1 Introduction

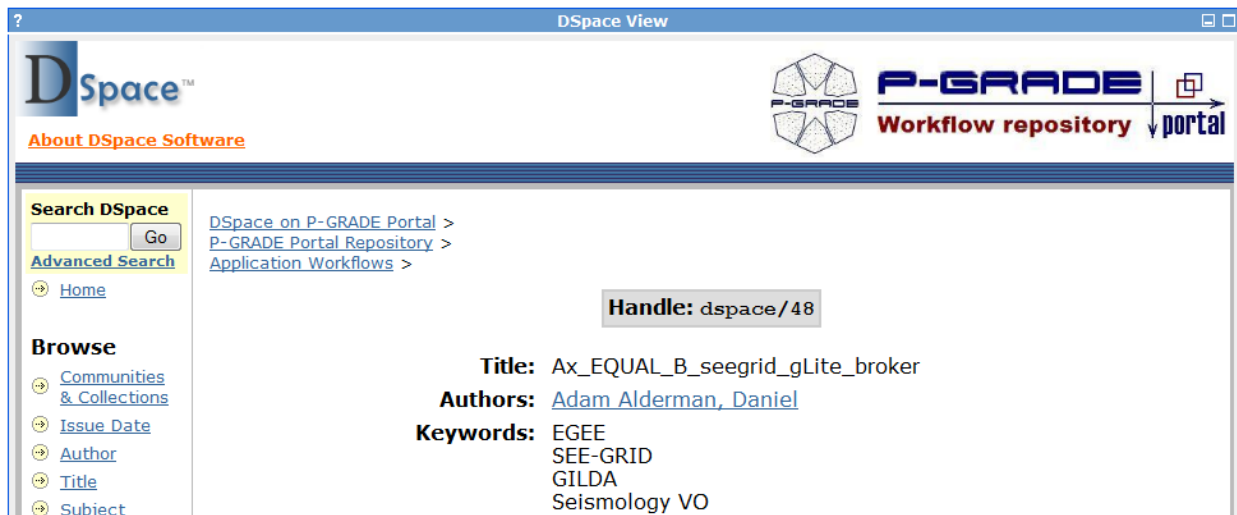
The aim of the DSpace Repository is to facilitate collaborative development and information dissemination among P-Grade Portal users. This is accomplished by allowing users to browse, search, download, and upload application workflows and other files to/from their local machine or Portal storage space. The DSpace Repository is presented to the Portal user as three portlets contained on two pages within the DSpace Repository tab. The default page view upon clicking the DSpace Repository tab is the Download/DSpaceView page which contains the Download portlet (download bar) and the DSpaceView portlet. The other page option presented within the DSpace Repository tab contains the Upload portlet. Descriptions and usage information regarding these portlets can be found below.

2. DSpaceView Portlet

When using the Download/DSpace page within the DSpace Repository tab, the DSpaceView portlet is displayed below the Download portlet (download bar) as seen in the screenshot above. The DSpaceView is a simple portlet that provides Portal users with access to DSpace within the P-GRADE Portal. This portlet should be used for all interactions with DSpace with the exception of downloading or uploading an application workflow to/from the user's Portal storage, which can be accomplished using the other two portlets described below. For help and information using DSpace from within the DSpaceView portlet, please reference the DSpace user's manual at http://www.dspace.org/1_5_1Documentation/. All further questions or comments regarding use of the DSpaceView portlet should be sent to the P-GRADE Portal administrators using the feedback link provided at the bottom of the DSpace portlet or by directly emailing pgportal@lpds.sztaki.hu.

3. Download Portlet

When using the Download/DSpace page within the DSpace Repository tab, the Download portlet is displayed at the top of the page above the DSpace view. This portlet allows Portal users to download application workflows shared by other users in the DSpace Repository and store them in their Portal storage space. The DSpace view should be used to locate the desired workflow for download to the user's Portal storage space, using either browsing or searching. Once the workflow is located in DSpace, the DSpace item handle corresponding to this workflow should be copied into the Download portlet (download bar). An example of such a DSpace item handle is "dspace/48," as shown in the screenshot below, and is also provided as an example in the information message on the right side of the Download portlet (download bar).



After successfully locating and copying the item handle for the application workflow in DSpace, the user should paste the handle into the DSpace handle box and click the Download button. Upon clicking the Download button, the download status or appropriate error messages will be displayed to the right of the download button. The potential status and error messages are listed below with a brief description of what they mean and how the user should proceed.

Browse to a workflow below to find its handle (e.g. [dspace/1](#)) – This is the default information message provided to the user which contains an example of a DSpace handle. As mentioned in the message, the user can locate the workflow handle using the browsing or searching capabilities of DSpace as provided in the DSpaceView.

Download successful! – This status message signifies that the user has successfully downloaded the application workflow with the given handle to his/her Portal storage space. This workflow is now accessible in the Workflow Manager and Storage portlets in the Workflow tab of the P-GRADE Portal.

The selected workflow is already available among your workflows – This alert message means that the user attempted to download a workflow that is already in his/her available workflows list. In this case, the download is not performed and this message is displayed. The

user can verify that this workflow is indeed in his/her storage by checking the Workflow tab of the Portal.

Invalid handle – This error message means that the download could not be performed because the DSpace handle supplied by the user was invalid. This likely means that the user copied or typed the handle incorrectly or that the handle refers to a workflow that is no longer available in DSpace (deleted item). If the user is presented with this error message, they should check the accuracy of the handle that they entered and also verify the handle still exists in DSpace using the DSpace view below the download portlet if necessary.

No handle entered – This error message is displayed above the DSpace Handle label if no handle was entered into the box before the Download button was clicked.

The download file is not a valid workflow package! – This error message means that the user specified the handle of a DSpace item that is not an application workflow. Since the Portal only supports the uploading of workflows, this message is displayed and the download is cancelled. If this message is displayed, the user should verify that the handle they entered actually corresponds to an application workflow that they wish to download.

Download failed, your quota is low – This error message means that the workflow could not be downloaded to Portal storage because the user has an insufficient amount of storage space left. The user should delete unneeded or unwanted workflows from his/her Portal storage space or request additional storage space from P-GRADE Portal administrators (pgportal@lpds.sztaki.hu) if necessary before attempting to download the workflow again.

Download Failed! – This error message signifies that the download failed for an unknown reason, and thus the desired application workflow was not uploaded to the user's Portal storage space. If the user receives this message they should attempt the download again after verifying that the handle they specified is correct. If this problem persists, the user should file a report with the P-GRADE Portal administrators using the Feedback link at the bottom of the DSpaceView portlet or by directly emailing pgportal@lpds.sztaki.hu so that the problem can be investigated and resolved.

4. Upload Portlet

The Upload portlet is displayed on its own page within the DSpace Repository tab and provides Portal users with a form for uploading one of their application workflows from their Portal storage to the DSpace repository. This allows Portal users to share their workflows and help facilitate the potential for collaborative development or information dissemination among P-GRADE Portal users.

When viewing the Upload page, users will be presented with a form they must fill out, designating which workflow is to be uploaded, and providing details pertinent to application workflows. When the user clicks the Submit button, the selected workflow, along with the supplied information, will be uploaded to the Application Workflows category on DSpace, granted that no errors occur. If the workflow is successfully uploaded, the user will be provided with a link that directs them to the item's new location on DSpace. A screenshot, as well as a brief description of each field, can be found below.

Upload

DSpace Login

*E-mail: *Password:

[Click here to register for a DSpace account](#)

Upload a Workflow

Select	Name
<input type="radio"/>	Ax_EQUAL_B_voce_gLite_broker_default
<input checked="" type="radio"/>	JavaDemoThirdPower_seggrid_gLite
<input type="radio"/>	LM_9_DEMO-fromscratch
<input type="radio"/>	MatrixTest_fromscratch

Author

*First: *Last:

Information

*Title:

Keywords: Comma-separated list (e.g. Keyword1, Keyword2)

Grid: (e.g. EGEE, SEE-GRID)

VO: (e.g. GILDA, Seismology VO)

Type:

Abstract

Description

*Required information

DSpace Login

The Portal user must enter his/her DSpace account information here. The given account must have permission to upload to the Application Workflow category on DSpace. When a workflow is uploaded to DSpace, it will be associated with this DSpace account. If the user does not have a DSpace account, they may use the link titled “Click here to register for a DSpace account” to create one. Note that permission to upload to the Application Workflow category is not automatically granted when a DSpace account is created; an administrator must first approve the account before anything can be uploaded.

E-mail – Required field: the e-mail address the user registered with on DSpace. This field will automatically be filled out with the e-mail address associated with the user’s P-GRADE Portal account.

Password – Required field: the password to the user’s DSpace account.

Upload a Workflow

This section displays a list of workflows that are available from the Portal user’s workflow storage space. The user may select the workflow he/she wishes to upload to DSpace.

Author

The Portal user must enter an author for the workflow. The ability to specify only a single author is currently available; however it may be possible to add more in the future.

First – Required field: the first name of the workflow author. This field will be automatically filled out with the user’s first name, if available (as specified by Full Name in the User Profile settings).

Last – Required field: the last name of the workflow author. This field will be automatically filled out with all names besides the user’s first name, if available (as specified by Full Name in the User Profile settings).

Information

The Portal user may enter various details about his/her workflow here, including: Title, Keywords, Grid, VO, and Type. Users may search by Keyword, Grid, VO, and Type on DSpace by selecting Browse by Subject.

Title – Required field: the title of the workflow being uploaded.

Keywords – A comma-separated list of any keywords/tags to be associated with the workflow.

Grid – Grid(s) the workflow has been designed to run on.

VO – Virtual Organization(s) the workflow has been designed to run on.

Type – The type of workflow (Full application, Template, Abstract workflow, or Other).

Abstract

The Portal user may enter an abstract about his/her workflow here. Note that line breaks are not taken into account.

Description

The Portal user may enter a more detailed description of his/her workflow here. Note that line breaks are not taken into account.

Upon filling out the form and clicking the Submit button, the upload status or appropriate error messages will be displayed within the DSpace Login section. The potential status and error messages are listed below with a brief description of what they mean and how the user should proceed.

You must select a workflow – This alert message is displayed if the user does not select a workflow to upload from the list of available workflows.

Upload successful!

[Click here to view this item on DSpace](#) – This two part status message is displayed when a

workflow is successfully uploaded. Clicking the link will bring the user to the location of the uploaded workflow on DSpace.

Invalid e-mail or password – This alert message is displayed if the provided DSpace Login details are incorrect, or if that particular DSpace account does not have permission to upload to the Application Workflows category.

If a required field is left blank, an error message will be displayed, informing the Portal user that he/she must fill out this part of the form before the upload can be completed. Users with questions or comments regarding uploading a workflow should contact the P-GRADE Portal administrators using the Feedback link at the bottom of the DSpaceView portlet or by directly emailing pgportal@lpds.sztaki.hu.

APPENDIX C: INSTALLATION & ADMINISTRATION MANUAL

INSTALLING AND CONFIGURING DSPACE

For details on the installation process, see the DSpace Installation Guide at <http://www.dspace.org/1.5.1Documentation/ch03.html>. A PDF version of this installation guide has been included in this package as well (DSpace-Manual.pdf).

SERVER SETUP

The following software was used for the test installation of DSpace (note that PostgreSQL should not be required if an external database is used):

- Scientific Linux 4.7 (<https://www.scientificlinux.org/>)
- Java JDK 1.6.0_12 (<http://java.sun.com/>)
- Apache Maven 2.1.0 (<http://maven.apache.org/>)
- Apache Ant 1.7.1 (<http://ant.apache.org/>)
- PostgreSQL 8.3.604 (<http://www.postgresql.org/>)
- Apache Tomcat 5.5.27 (<http://tomcat.apache.org/>)
- Perl 5.8.5 (<http://www.perl.org/>)

See the DSpace Installation Guide for specific information about the configuration of each component. Note some important changes that need to be made to some Tomcat configuration files if that is the web server being used. The files used for the test installation of DSpace are included in this package ('tomcat/tomcat.conf' and 'tomcat/server.xml').

INSTALLING DSPACE

After all the appropriate software is installed, you can begin the installation process. For a more detailed explanation of each step, see the Installation Guide.

The following steps were taken during the test installation of DSpace:

1. Create a user on the server called 'dspace'
2. Download and extract 'dspace-1.5.1-release' (not 'dspace-1.5.1-src-release') from <http://sourceforge.net/projects/dspace/>. This file has also been included in this package

(dspace-1.5.1-release.tar.gz). On the test server, this file was downloaded to the /home/dspace directory.

3. Special configuration for the database (see DSpace Installation Guide).
4. Edit 'dspace-1.5.1-release/dspace/config/dspace.cfg' with the appropriate information. The configuration file used in the DSpace test installation is included with this package (dspace-1.5.1-release/dspace/config/dspace.cfg). The DSpace Installation guide points out the parts of the configuration file that should be altered; the dspace.cfg file in this package may be used, but here are some important items to be aware of:
 - dspace.dir – the directory where DSpace will be installed ('/home/dspace/install' on test installation)
 - dspace.url – complete URL of this server's DSpace home page ('http://n38.hpcc.sztaki.hu:8080/jspui' on test installation)
 - dspace.hostname – fully-qualified domain name of web server ('n38.hpcc.sztaki.hu' on test installation)
 - db.url – URL for connecting to database ('jdbc:postgresql://localhost:5432/dspace' on test installation)
 - db.username – username for database ('dspace' on test installation)
 - db.password – password for database (should be changed from that of test installation)
 - mail.server – fully-qualified domain name of your outgoing mail server ('localhost' on test installation)
 - mail.admin – mailbox for DSpace site administrator (should be changed from that of test installation)
 - registration.notify – mailbox for emails when new users register. Optional, but could be useful; since new users are not automatically given permission to upload files, this will alert a DSpace administrator that a user has registered ('\${mail.admin}' on test installation)
 - handle.prefix – prefix to appear in DSpace Handle ('dspace' on test installation)
 - handle.canonical.prefix – IMPORTANT: this item must be added to the config file in order to display the location of items using the DSpace URL instead of the CNRI Handle system ('\${dspace.url}/handle/' on test installation, should not change)

- upload.max – maximum size of uploaded files in bytes ('104857600' [100Mb] on test installation)
5. Create directory where DSpace will be installed ('/home/dspace/install' on test installation).
 6. As the 'dspace' UNIX user, go to the directory 'dspace-1.5.1-release/dspace/' and enter the command 'mvn package' to generate the installation package.
 7. As the 'dspace' UNIX user, go to the directory 'dspace-1.5.1-release/dspace/target/dspace-1.5.1.dir/' and enter the command 'ant fresh_install' to install DSpace.
 8. Tell the web server where to find the DSpace web applications 'jspui' and 'lni'. Assuming Tomcat is being used as the web server, add the lines below in the <Host> section of '[tomcat]/conf/server.xml', where [tomcat] is the installation directory for Tomcat and [dspace] is the installation directory for DSpace ('/home/dspace/install' on the test installation). The server.xml file used for the test installation can be found in this package ('tomcat/server.xml'):

```
<!-- DEFINE A CONTEXT PATH FOR DSpace JSP User Interface -->
<Context path="/jspui" docBase="[dspace]\webapps\jspui" debug="0"
reloadable="true" cachingAllowed="false" allowLinking="true"/>

<!-- DEFINE A CONTEXT PATH FOR DSpace LNI Interface -->
<Context path="/lni" docBase="[dspace]\webapps\lni" debug="0"
reloadable="true" cachingAllowed="false" allowLinking="true"/>
```

9. Enter the command '[dspace]/bin/create-administrator' to create an initial administrator account, where [dspace] is the installation directory for DSpace ('/home/dspace/install' on the test installation).
10. Restarting the web server should enable you to view the default installation of DSpace at the base URL of the server ('<http://n38.hpcc.sztaki.hu:8080/jspui/>' on the test installation).

CONFIGURING DSPACE

Note that whenever a change is made to a configuration file,

1. If DSpace has been previously installed and you skipped the Installing DSpace step, then you should edit '[dspace]/config/dspace.cfg' with the appropriate information, where [dspace] is the installation directory for DSpace ('/home/dspace/install' on the test installation). The configuration file used in the DSpace test installation is included with

this package ('dspace/config/dspace.cfg'). The dspace.cfg file in this package may be used, but here are some important items to be aware of:

- dspace.dir – the directory where DSpace will be installed ('/home/dspace/install' on test installation)
- dspace.url – complete URL of this server's DSpace home page ('http://n38.hpcc.sztaki.hu:8080/jspui' on test installation)
- dspace.hostname – fully-qualified domain name of web server ('n38.hpcc.sztaki.hu' on test installation)
- db.url – URL for connecting to database ('jdbc:postgresql://localhost:5432/dspace' on test installation)
- db.username – username for database ('dspace' on test installation)
- db.password – password for database (should be changed from that of test installation)
- mail.server – fully-qualified domain name of your outgoing mail server ('localhost' on test installation)
- feedback.recipient – mailbox for feedback mail (should be changed from that of test installation)
- mail.admin – mailbox for DSpace site administrator (should be changed from that of test installation)
- registration.notify – mailbox for emails when new users register. Optional, but could be useful; since new users are not automatically given permission to upload files, this will alert a DSpace administrator that a user has registered (should be changed from that of test installation)
- handle.prefix – prefix to appear in DSpace Handle ('dspace' on test installation)
- handle.canonical.prefix – **IMPORTANT**: this item must be added to the config file in order to display the location of items using the DSpace URL instead of the CNRI Handle system ('http://n38.hpcc.sztaki.hu:8080/jspui/handle/' on test installation)
- upload.max – maximum size of uploaded files in bytes ('104857600' [100Mb] on test installation)

2. The configuration files ‘register’ and ‘change_password’ should be changed to display the appropriate messages. They are located in ‘*[dspace]/config/emails/*’, where *[dspace]* is the DSpace installation directory (‘/home/dspace/install’ on test installation). The files used in the test installation are included with this package (‘dspace/config/emails’).
3. The DSpace Installation Manual gives some Advanced Installation guides (http://www.dspace.org/1_5_1Documentation/ch03.html#N1090F) on setting up ‘cron’ jobs, deploying a multilingual version of DSpace, and configuring DSpace to use HTTPS.
 - One ‘cron’ job that was not set up for the test installation of DSpace, but should be quite useful for the production installation, is a script that generates statistical reports. In order to set this up, enter the command ‘crontab –e’ as the ‘dspace’ UNIX user, and add the following lines:

```
# Run stat analyses
0 1 * * * [dspace]/bin/stat-general
0 1 * * * [dspace]/bin/stat-monthly
0 2 * * * [dspace]/bin/stat-report-general
0 2 * * * [dspace]/bin/stat-report-monthly
-c
```

SETTING UP DSPACE

- 1) Log into DSpace as the administrator and create a new user group (‘Basic Users’ on the test installation). This group will serve as the group of users that have permission to upload files to DSpace.
 - a) Click ‘Administer’ in the menu on the left
 - b) Click ‘Groups’ in the menu on the left
 - c) Give the group a name
 - d) Click ‘Select E-people’
 - e) Add the users who should have permission to upload files to DSpace
 - f) Click ‘Update Group’
- 2) Create a community (‘P-GRADE Portal Repository’ on the test installation).
 - a) Click ‘Communities/Collections’ in the menu on the left
 - b) Click ‘Create Top-Level Community’
 - c) Fill out the form about community information

- d) You do not need to edit the Community's Authorizations
 - e) Click 'Create'
- 3) Next, create a collection within this community where application workflows being uploaded/downloaded from/to the P-GRADE Portal should be stored ('Application Workflows' on the test installation).
- a) Click 'Communities & Collections' in the menu on the left
 - b) Click on your newly-created community
 - c) Click 'Create collection'
 - d) Click 'Next'
 - e) Fill out the form about collection information
 - f) Click 'Next'
 - g) Click 'Select Groups'
 - h) Add the groups that should have permission to upload to this collection
 - i) Click 'Next'
 - j) Click 'Update'
 - k) You may want to create other collections as well for items that are not application workflows. On the test installation, we also created 'Jobs', 'Results', and 'Documentation and Publications'
- 4) A user called "anonymous" needs to be created (this will be for the download process on the P-GRADE Portal, so remember the password used).
- a) Register a DSpace account in the normal way, using an e-mail address you have access to
 - b) Log in as the administrator and click 'Administer' in the menu on the left
 - c) Click 'E-people' in the menu on the left
 - d) Click 'Select E-person'
 - e) Click the 'Select' button next to the new account you registered
 - f) Click 'Edit'
 - g) Erase all information about this user, and set the Email field to 'anonymous'

- h) Click 'Save'
- 5) Add the '.zip' and '.gz' file extensions to the Bitstream Registry so that they may be recognized as Application Workflows (this step is optional, these file types may still be uploaded regardless).
- a) Click 'Bitstream Registry' in the menu on the left
 - b) Click the 'Add New' button at the bottom of the page
 - c) At the bottom of the page, add this information in the new fields: "compressed/application-workflow", "Application Workflow", "Compressed Zip File", "Known", "zip, gz, tar, tar.gz"
 - d) Click the 'Update' button next to the new field

ADDING/EDITING FILES TO DSPACE

1. Add the file 'pgradeheader.jpg' (located in this package under 'dspace/webapps/jspui/image') to '[dspace]/webapps/jspui/image' on the DSpace server, where [dspace] represents the DSpace installation directory (/home/dspace/install on the test installation).
2. Overwrite the file '[dspace]/webapps/jspui/display-item.jsp', where [dspace] represents the DSpace installation directory (/home/dspace/install on the test installation), with the file 'dspace/webapps/jspui/display-item.jsp' in this package. This new file changes the way DSpace displays an Item handle.
3. Overwrite the file '[dspace]/webapps/jspui/layout/header-default.jsp', where [dspace] represents the DSpace installation directory (/home/dspace/install on the test installation), with the 'dspace/webapps/jspui/layout/header-default.jsp' in this package. This new file changes the header of the DSpace website to display the P-GRADE/DSpace logo.
4. Overwrite the file '[dspace]/config/crosswalks/mods-submission.xml', where [dspace] represents the DSpace installation directory (/home/dspace/install on the test installation), with the file 'dspace/config/crosswalks/mods-submission.xml', in this package. This new file has changes for converting metadata from MODS to DC (when uploading from P-GRADE Portal).

ADMINISTRATING DSPACE

1. Click 'Administer' in the menu on the right. Here you will see many functions for administrating DSpace. See the DSpace Documentation for more information about these.

2. Whenever a new user registers, they must be added to the Basic Users group before they can begin uploading. It is also possible to “pre-register” users. The user must still go through the registration process in order to use DSpace; the advantage of “pre-registering” a user is that they may be added to other user groups ahead of time.
 - a. Click ‘Administer’ in the menu on the right
 - b. Click ‘E-people’ in the menu on the right
 - c. Click ‘Add EPerson’
 - d. Fill out information about Email, Name, Phone, and Language
 - e. Do NOT select ‘Can Log In’ or ‘Require Certification’
 - f. Click ‘Save’

INSTALLING AND CONFIGURING NEW P-GRADE COMPONENTS

1. The following libraries, all of which are included in this package (‘jars’), need to be added to the build path of the P-GRADE Portal Java project, as well as in ‘*[tomcat]/webapps/szupergrid/WEB-INF/lib*’ on the P-GRADE Portal server, where *[tomcat]* is the installation directory of Tomcat on the Portal server (‘*home/cory/pgportal/apache-tomcat-5.5.25*’ on the test installation):
 - Mets Java Toolkit 1.5 (mets.jar) - <http://hul.harvard.edu/mets/>
 - JDOM 1.1 (jdom.jar) - <http://jdom.org/>
 - Note: an older version of this library is currently in use – it was necessary to replace it with a newer version for this project
 - Apache Jakarta Commons HttpClient 3.1 (commons-httpclient-3.1.jar) - <http://hc.apache.org/httpclient-3.x/>
 - Apache Commons Codec 1.3 (commons-codec-1.3.jar) - <http://commons.apache.org/codec/>
 - Apache Commons CLI 1.2 (commons-cli-1.2.jar) - <http://commons.apache.org/cli/>
2. Create a new package in the Java project under the ‘src/java/portal/szupergrid/src/’ folder to contain the new Java files. On the test Portal this package was called ‘hu.sztaki.lpbs.repository’. Place the following Java files in the new project package (found under ‘repository/Java/’ in this installation package):
 - RepositoryPortlet.java

- IMPORTANT: the global variable ‘dspaceURL’ in this file MUST be changed to the home page of the DSpace repository
 - RepositoryUpload.java
 - IMPORTANT: the global variable ‘collection’ in this file MUST be changed to the handle of the “Application Workflows” collection on DSpace. For example, on the test installation of DSpace, this handle was ‘dspace/44’. However, if DSpace is reinstalled/this collection is recreated, this handle is likely to change.
 - RepositoryDownload.java
 - IMPORTANT: the global variables ‘anonDSpaceUser’ and ‘anonDSpacePass’ correspond to the ‘anonymous’ user created in the Setting Up DSpace section above, and should be set accordingly.
 - Note: this portlet assumes that the workflow being downloaded was originally compressed in “.zip” format. Should look into methods for ascertaining different file types (if it’s possible that there may be others) and handling them appropriately.
 - DSpaceSIP.java
 - LNIclient.java
3. Create a new directory called ‘hu/sztaki/lpds/repository’ under ‘[tomcat]/webapps/szupergrid/WEB-INF/lib’ on the P-GRADE Portal server, where [tomcat] is the installation directory of Tomcat on the Portal server (‘home/cory/pgportal/apache-tomcat-5.5.25/’ on the test installation). Compile the Java project and copy the compiled class files to this new folder. The newly compiled files should be called:
- DSpaceSIP.class
 - DSpaceSIP\$PackageFile.class
 - LNIclient.class
 - LNIclient\$PropfindHandler.class
 - LNIclient\$PropfindMethod.class
 - RepositoryDownloadPortlet.class

- RepositoryPortlet.class
 - RepositoryUploadPortlet.class
4. Create a new folder called ‘repository’ under ‘src/java/portal/szupergrid/webapp/jsp’ within the Java project, as well as under ‘[tomcat]/webapps/szupergrid/jsp/’ on the P-GRADE Portal server, where [tomcat] is the installation directory of Tomcat on the Portal server (‘home/cory/pgportal/apache-tomcat-5.5.25/’ on the test installation). Place the following JSP files in these new folders (found in ‘repository/jsp/’ of this package):
- DSpaceView.jsp
 - DSpaceDownload.jsp
 - DSpaceUpload.jsp
5. Edit the ‘layout.xml’ file (located under ‘src/java/portal/szupergrid/webapp/WEB-INF’ in the Java project, as well as under ‘[tomcat]/webapps/szupergrid/WEB-INF/’ on the P-GRADE Portal server, where [tomcat] is the installation directory of Tomcat on the Portal server (‘home/cory/pgportal/apache-tomcat-5.5.25/’ on the test installation) to include the following text. Where this text is placed within the file affects where the DSpace Repository tab appears on the Portal (the file used on the test Portal can be found under ‘repository/WEB-INF/’ in this package):

```

<portlet-tab>
  <title lang="en">DSpace Repository</title>
  <portlet-tabbed-pane style="sub-menu">
    <portlet-tab label="dspaceDownload">
      <title lang="en">Download/DSpaceView</title>
      <table-layout>
        <row-layout>
          <column-layout width="50%">
            <portlet-frame>
              <portlet-
class>szupergrid#RepositoryDownloadPortlet</portlet-class>
              </portlet-frame>
            </column-layout>
          </row-layout>
          <row-layout>
            <column-layout width="50%">
              <portlet-frame>
                <portlet-
class>szupergrid#RepositoryPortlet</portlet-class>
                </portlet-frame>
              </column-layout>
            </row-layout>
          </table-layout>
        </portlet-tab>
      <portlet-tab label="dspaceUpload">
        <title lang="en">Upload</title>

```

```

        <table-layout>
            <row-layout>
                <column-layout width="50%">
                    <portlet-frame>
                        <portlet-
class>szupergrid#RepositoryUploadPortlet</portlet-class>
                        </portlet-frame>
                    </column-layout>
                </row-layout>
            </table-layout>
        </portlet-tab>
    </portlet-tabbed-pane>
</portlet-tab>

```

6. Edit the 'portlet.xml' file (located under 'src/java/portal/szupergrid/webapp/WEB-INF' in the Java project, as well as under '[tomcat]/webapps/szupergrid/WEB-INF/' on the P-GRADE Portal server, where [tomcat] is the installation directory of Tomcat on the Portal server ('home/cory/pgportal/apache-tomcat-5.5.25/' on the test installation) to include the following text (the file used on the test Portal can be found under 'repository/WEB-INF/' in this package):

```

    <portlet>
        <description xml:lang="en"> DSpace View within P-GRADE Portal
</description>
        <portlet-name>RepositoryPortlet</portlet-name>
        <display-name xml:lang="en">DSpace View</display-name>
        <portlet-
class>hu.sztaki.lpds.repository.RepositoryPortlet</portlet-class>
        <expiration-cache>60</expiration-cache>
        <supports>
            <mime-type>text/html</mime-type>
            <portlet-mode>help</portlet-mode>
        </supports>
        <supported-locale>en</supported-locale>
        <portlet-info>
            <title>DSpace View</title>
            <short-title>DSpace View</short-title>
            <keywords>DSpace</keywords>
        </portlet-info>
    </portlet>
    <portlet>
        <description xml:lang="en"> Upload files to DSpace
</description>
        <portlet-name>RepositoryUploadPortlet</portlet-name>
        <display-name xml:lang="en">Upload</display-name>
        <portlet-
class>hu.sztaki.lpds.repository.RepositoryUploadPortlet</portlet-class>
        <expiration-cache>60</expiration-cache>
        <supports>
            <mime-type>text/html</mime-type>
            <portlet-mode>help</portlet-mode>
        </supports>
        <supported-locale>en</supported-locale>
        <portlet-info>

```

```

        <title>Upload</title>
        <short-title>Upload</short-title>
        <keywords>DSpace</keywords>
    </portlet-info>
</portlet>
<portlet>
    <description xml:lang="en"> Download files from DSpace
</description>
    <portlet-name>RepositoryDownloadPortlet</portlet-name>
    <display-name xml:lang="en">Download</display-name>
    <portlet-
class>hu.sztaki.lpds.repository.RepositoryDownloadPortlet</portlet-
class>
    <expiration-cache>60</expiration-cache>
    <supports>
        <mime-type>text/html</mime-type>
        <portlet-mode>help</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
        <title>Download</title>
        <short-title>Download</short-title>
        <keywords>DSpace</keywords>
    </portlet-info>
</portlet>

```

7. Restart the P-GRADE Portal and it should now display a 'DSpace Repository' tab with the new workflow repository functionality in place.