Web-Based Science Gateways for Structural Bioinformatics

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Inform. Sandra Gesing
aus Ahaus

Tübingen
2013

Tag der mündlichen Qualifikation:	10.04.2013
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Oliver Kohlbacher
2. Berichterstatter:	Prof. Dr. Thomas Walter

To Isabell
Being eleven years old, you are one ”smart cookie” -
full of the joys of life and very patient with me.

Slartibartfast: I must warn you, we're going to pass through, well, a sort of gateway thing.
Arthur Dent: What?
Slartibartfast: It may disturb you. It scares the willies out of me.
(Douglas Adams: The Hitchhiker's Guide to the Galaxy)

Acknowledgements

The work on this thesis was carried out during the years 2006 - 2011 at the University of Tübingen in the group and under the supervision of Prof. Oliver Kohlbacher, at the computer centre of the University of Tübingen in the group of Dr. Werner Dilling, at the e-Science Institute Edinburgh in the group of Dr. Jano van Hemert, and at MTA SZTAKI in the group of Prof. Peter Kacsuk. This thesis would have not been possible without the support and effort of many people.

I would like to deeply thank my supervisor Prof. Oliver Kohlbacher for taking the risk and offering me to join his team as a bioinformatics newbie. He aroused my interest in science gateways for structural bioinformatics and gave me the opportunity to work in this area and to write this thesis. Our lively discussions were always fruitful and he gave me the freedom to do research in those areas that fascinated me most. He supported my research visits lasting several weeks and the workshop series IWSG-Life (International Workshop on Science Gateways for Life Sciences), which is guided by me.

I am also very grateful to Prof. Thomas Walter for beneficial discussions on science gateways and reviewing this thesis. He and the former head of the computer centre Prof. Dietmar Kaletta as well as Dr. Werner Dilling gave me the possibility to work in the field of grid computing and parallel programming. It was a pleasure to work for them.

Furthermore, I owe much gratitude to Dr. Jano van Hemert and Prof. Malcolm Atkinson for the valuable collaboration and for inviting me several times to the e-Science Institute Edinburgh for long-term research visits. The scientific and technical discussions gave me new insights in the field of science gateways. Jano and I chaired together the first event of the workshop series IWSG-Life. The extension of Rapid introduced in this work would have been unthinkable without the collaboration with Dr. Jos Koetsier, the main developer of Rapid.

I am also tremendously thankful to Prof. Peter Kacsuk who supported my work on WS-PGRADE and the project MoSGrid. He gave me the opportunity to work for several weeks in his lab and has held many beneficial discussions with me. The SAML extension of WS-PGRADE, the development of the UNICORE submitter, and the Application Specific Module would have been unthinkable without the collaboration with Prof. Miklos Kozlovszky, Anna Szikszay Fabri, István Márton, and Ákos Balasko.

Many thanks go also to Prof. Shantenu Jha for the invitation to his lab for several weeks and the valuable discussions. Even though it turned out that the subject of the collaboration is beyond the scope of this work, it inspired me for the work on WS-PGRADE described in this thesis and for future projects.

My work carried out via the MoSGrid project would not have been possible without the efficient and pleasant collaboration with the following colleagues; Prof. Oliver Kohlbacher, Dr. Jens Krüger, Dr. Werner Dilling, Dr. Bernd Schuller, Richard Grunzke, Georg

Birkenheuer, Patrick Schäfer, Prof. Sonja Herres-Pawlis, Johannes Schuster, Sebastian Breuers, Prof. André Brinkmann, Ines Dos Santos Vieira, Prof. Gregor Fels, Prof. Rene Jäkel, Dr. Ralph Müller-Pfefferkorn, Dr. Thomas Steinke, and Martin Wewior. Many simulations for the performance studies were performed on the HPC-BW cluster (bwGRiD cluster), which is funded by the Ministry for Education and Research of Germany and the Ministry of Science, Research and Arts of the state Baden-Württemberg.

I would like to thank my colleagues and former colleagues in the Kohlbacher lab for fruitful discussions. Special thanks go to Jens Krüger and Muriel Quenzer. Jens worked with me in the MoSGrid project and supported me with use cases and workflows in structural bioinformatics. Muriel drafted with me the first concept for a workflow-enabled portal for bioinformatics. I am also thankful to Andreas Bertsch who supported me with his deep insights in TOPP and ideas for science gateways. I'm grateful to Marc Röttig who supported the migration of workflows from Galaxy to WS-PGRADE with his wide knowledge about Galaxy. Furthermore, many thanks go to our system administrators Jan Schulze, Muriel, and Andreas Kerzmann for their premium support.

I also would like to thank Dr. Korbinian Schneeberger, Dr. Stephan Ossowski, Jörg Hagmann, Prof. Detlef Weigel, Dr. Alexander Auch, Viktor Achter, and Lech Nieroda for the efficient and pleasant collaborations on projects which are not in the scope of this work but in the area of parallelisation of algorithms and tools.

I am grateful to Oliver, Thomas Walter, André, Jens, Andreas, Marc, Muriel, Miklos, Stefan, and Hugh for their valuable comments on the manuscript.

During my time at the Kohlbacher lab, in Werner Dilling's group, and at the labs I have visited, I have met a lot of great people. Besides the fabulous conversations at the labs, I enjoyed incomparable evenings at *Saints & Scholars*, *Last Resort*, *Collegium*, and *Coyote Cafe* in Tübingen, genial get-togethers at *The Doctors Bar* and *Native State* in Edinburgh, exquisite celebrations at *Trófea Grill* and *Morrison's Pubs* in Budapest, and amusing get-togethers in Baton Rouge. These evenings were always a great possibility to relax in pleasant company and to have breaks from work which have not seldom lead to very creative ideas for tools and projects.

Finally, I would like to wholeheartedly thank my friends and my whole family, who often suffered from my keening about the progress of my research and patiently followed my scientific endeavor.

In accordance with the standard scientific protocol, I will use the personal pronoun we to indicate the reader and the writer, or my scientific collaborators and myself.

Abstract

Structural bioinformatics applies computational methods to analyse and model three-dimensional molecular structures. These methods address data-intensive and compute-intensive problems, which demand high-performance computing (HPC) to allow data analysis in an acceptable time. Thus, structural bioinformatic applications are ideal candidates for grid and cloud computing infrastructures, so-called DCIs (Distributed Computing Infrastructures). DCIs provide access to HPC facilities and services across organisational boundaries. However, the usability of DCIs is limited and the use of the complex methods in structural bioinformatics requires a lot of experience. In addition, users mainly process and analyse data not only via single jobs but via workflows.

An approach to offer easy and intuitive access to applications on DCIs are science gateways. In general, a science gateway provides a single point of entry to a set of tools and data of a specific application domain while hiding the complex underlying infrastructure. Web-based science gateways are additionally characterised by only requiring a computer connected to the Internet and an installed web browser on the users' side. Developers of such gateways support the users with pre-configured user interfaces targeted for a specific application domain. The overall goal for creating science gateways is to increase the usability of applications.

This work is focused on workflow-enabled grid portals that are specific web-based science gateways supporting the management of workflows on DCIs. Four major aspects in the context of workflow-enabled grid portals are adressed in this work: security in portals with underlying DCIs, job and workflow management, migration of workflows between diverse science gateways, and the automatic creation of portlets for workflow management.

We have developed a granular security concept, which encloses all layers of the involved infrastructure: the user interface, the high-level middleware layer, the grid middleware layer, and the HPC facilities. We are especially focused on the role-based user management concerted for the molecular simulation community and the credential management. The workflow-enabled grid portal WS-PGRADE (Web Services Parallel Grid Runtime and Developer Environment) has been extended for the use of SAML (Security Assertion Markup Language) for trust delegation. The created credential files set the stage for the authentication processes in the connected DCIs. We chose to use DCIs connected via the grid middleware UNICORE 6 because of UNICORE's scalable service-oriented architecture and its workflow engine. For the integration in WS-PGRADE, we implemented a plugin, a so-called submitter, which allows invoking jobs on UNICORE 6. The submitter additionally supports UNICORE workflows to be invoked via WS-PGRADE and thus provides workflow interoperability. Users can seamlessly re-use existing UNICORE 6 workflows in WS-PGRADE. The Application Specific Module (ASM) has been developed to simplify the implementation of workflow-enabled portlets extending WS-PGRADE. The migration of workflows is achieved by a tool for the export of Galaxy workflows to WS-PGRADE workflows. Galaxy is a workflow-enabled portal for cloud infrastructures and for local HPC facilities. It is widely used but has the disadvantage of lacking the possibility to connect to grid infrastructures. Therefore, we implemented the export of Galaxy workflows to WS-PGRADE workflows, which can be easily imported by the users in WS-PGRADE. Furthermore, this work presents a concept for automatically generating graphical user interfaces for a WS-PGRADE portal via the software framework Rapid. Rapid allows creating portlets without programming in the traditional sense. Developers are enabled to automatically convert XML files to a fully functional portlet for WS-PGRADE.

The concepts and implementations in this work are applied in the science gateway of the project MoSGrid (Molecular Simulation Grid). It forms a use case for a science gateway for structural bioinformatics and provides a complete solution for the molecular simulation community.

Zusammenfassung

In der Strukturbioinformatik werden rechnergestützte Methoden angewandt, um Molekularstrukturen zu analysieren und modellieren. Diese Methoden befassen sich mit daten- und rechenintensiven Problemen, die für Datenanalysen in annehmbarer Zeit den Einsatz von Hochleistungsrechnern erfordern. Dadurch sind Anwendungen in der Strukturbioinformatik ideal geeignet für den Einsatz auf sogenannten Grid- und Cloud-Rechnerinfrastrukturen. Diese verteilten Rechnerinfrastrukturen bieten Zugang zu Hochleistungsrechnern und Diensten über organisatorische Grenzen hinweg. Allerdings sind die Interaktionsmöglichkeiten mit den Rechnerinfrastrukturen nicht benutzerfreundlich und die Bedienung der komplexen Methoden der Strukturbioinformatik erfordert viel Erfahrung. Zudem führen Benutzer nicht nur einzelne Jobs sondern primär Workflows zur Datenanalyse aus.

Um einen intuitiven Zugang zu Anwendungen auf verteilten Rechnerinfrastrukturen zu erreichen, werden Science Gateways eingesetzt. Im Allgemeinen bieten sie einen einheitlichen Zugang zu verschiedensten Programmen eines spezifischen Anwendungsgebiets an und verbergen die komplexe unterliegende Infastruktur vor dem Benutzer. Webbasierte Science Gateways zeichnen sich zusätzlich dadurch aus, dass sie auf der Benutzerseite lediglich einen Computer mit Internetzugang und einen Webbrowser erfordern. Entwickler solcher Science Gateways unterstützen die Benutzer mit voreingestellten Bedienoberflächen, die für das jeweilige Anwendungsgebiet angepasst sind. Der Einsatz von Science Gateways hat zum Ziel, die Bedienung von Software benutzerfreundlicher zu gestalten.

Der Fokus dieser Arbeit liegt auf workflowfähigen Grid-Portalen, die den Benutzern die intuitive Verwaltung von Workflows auf verteilten Rechnerinfrastrukturen ermöglichen. Es werden vier Kernaspekte von workflowfähigen Grid-Portalen in dieser Arbeit adressiert: die Sicherheit in Portalen mit Zugriff auf verteilte Rechnerinfrastrukturen, Job- und Workflowverwaltung, die Migration von Workflows zwischen bestehenden Science Gateways und die automatische Generierung von Portlets für Workflows.

Wir haben ein granulares Sicherheitskonzept entwickelt, das alle Ebenen der beteiligten Infrastruktur umfasst. Wir konzentrieren uns insbesondere auf eine rollenbasierte Benutzerverwaltung für die rechnergestützte Chemie-Community und die Verwaltung von Berechtigungsdateien. Das workflowfähige Grid-Portal WS-PGRADE (Web Services Parallel Grid Runtime and Developer Environment) ist für den Einsatz von SAML (Security Assertion Markup Language) erweitert worden. Die erzeugten Berechtigungsdateien bilden die Grundlage für die Authentifizierungsprozesse in den angebundenen verteilten Rechnerinfrastrukturen. Wir haben uns aufgrund der skalierbaren service-orientierten Architektur und der zugehörigen Workflow-Engine für UNICORE 6 als Grid-Middleware entschieden. Für die Integration mit WS-PGRADE wurde ein sogenannter Submitter implementiert, über den Jobs an UNICORE 6 geschickt werden können. Ausserdem ermöglicht er das Aktivieren von UNICORE-Workflows in WS-PGRADE und somit Workflow-Interoperabilität. Benutzer werden in die Lage versetzt, UNICORE-Workflows in WS-PGRADE wiederzuverwenden. Die Anwendungsschnittstelle Application Specific Module (ASM) erleichtert Entwicklern die Erstellung von workflowfähigen Portlets in WS-PGRADE. Die Migration von Workflows wird mit einem Programm realisiert, das Galaxy-Workflows in WS-PGRADE-Workflows konvertiert. Galaxy ist ein workflowfähiges Portal für Cloud-Rechnerinfrastrukturen und für lokale Hochleistungsrechner. Es ist weitverbreitet, hat jedoch den Nachteil, das es keine Verbindungsmöglichkeiten zu Grid-Rechnerinfrastrukturen bietet. Deswegen haben wir ein Programm implementiert, das Galaxy-Workflows als WS-PGRADE-Workflows exportiert, die sehr einfach von Benutzern importiert und dadurch mit wenig Aufwand wieder benutzt werden können. Ausserdem wird ein Konzept zur automatischen Generierung von graphischen Benutzerschnittstellen für WS-PGRADE über die Software Rapid vorgestellt. Rapid versetzt Benutzer in die Lage, intuitive graphische Benutzerschnittstellen zu realisieren ohne im klassischen Sinne zu programmieren. Entwickler können automatisch XML-Dateien in vollfunktionale Portlets für WS-PGRADE konvertieren.

In dem Science Gateway des Projekts MoSGrid (Molecular Simulation Grid) sind die Konzepte und Implementierungen dieser Arbeit umgesetzt. Das Science Gateway stellt einen Anwendungsfall für ein Science Gateway im Bereich der Strukturbioinformatik dar und bietet eine vollständige Lösung für die Anwender von molekularen Simulationen.

Contents

List of Figures

List of Tables

List of Algorithms

”A mouse pusher:
A person that prefers a mouse over a keyboard; originally used for Macintosh fans. The derogatory implication is that the person has nothing but the most superficial knowledge of the software he/she is employing, and is incapable of using or appreciating the full glory of the command line.”
(http://www.retrologic.com/jargon/M/mouse-pusher.html)

1 Introduction

Structural bioinformatics is concerned with computational methods to gain new insight into molecular structures, their prediction, and the analysis of their functions. These computational methods are invaluable tools in materials science, structural biology, and drug design. Structural bioinformatics addresses data-intensive and compute-intensive problems, which demand high-performance computing (HPC) to allow data analysis in an acceptable time. Hence, grid and cloud computing infrastructures, so-called Distributed Computing Infrastructures (DCIs), have been developed to provide access to HPC facilities and services across organisational boundaries.

One of the first definitions of grid computing can be found in a book edited by Foster and Kesselmann [1]: ”A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.” The differences between grid and cloud computing have been discussed for several years now and the definition of cloud computing is still evolving. One of the opinions is that cloud computing offers on-demand resource provisioning with the possibility of scaling prices according to the utilised resources [2]. Sam Johnston stated in his blog ”The Cloud is what The Grid could have been.” [3] Aside from differences between grid and cloud computing, grid middlewares and cloud software frameworks are responsible for the authentication of users, the submission and monitoring of jobs, and the data management. Accordingly, the underlying hardware, the operating systems involved, batch systems, and file systems are hidden from the users.

Even though the details of the underlying infrastructure of DCIs is hidden, the application of structural bioinformatic methods on DCIs has usability issues. On the one hand, these issues derive from the applied methods. The usability of individual tools is limited and the implemented methods are very complex, reflecting the underlying complex theory.

They thus require a lot of experience. The lack of user interfaces and of pre-configured settings deters novice users from using the tools. The users have to become acquainted with the domain-related features, of course, but employing them via command lines increases the barrier for a wide use. On the other hand, the underlying DCIs augment the complexity, especially for users who lack a computer science background. In order to access a DCI, users have to become versed in command line tools and job description languages to submit jobs via a grid middleware or a cloud software framework. Already the authentication procedure may demand the use of command line tools. In addition, users in structural bioinformatics mainly process and analyse data not only via a single job but via a workflow. A workflow can be described as a sequence of connected steps in a defined order based on its control and data dependencies. For instance, the preparation of a protein structure may include the minimisation and equilibration of energies to obtain reliable predictions about the protein's properties.

An approach to offer easy and intuitive access to applications on DCIs are science gateways. A science gateway provides a single point of entry to a set of tools for a specific scientific application domain. It hides, as far as suitable, underlying technical infrastructures and offers a single look-and-feel for diverse functionalities. The overall goal for creating science gateways is to increase the usability of applications. Usability of software is examined by many different researchers (e.g., Seffah and Metzker [4], Brinck et al. [5]) and an ISO standard [6] has been developed. The standard elucidates ergonomic requirements for office work with visual display terminals including a guidance on usability. Steve Krug states in his book "Don't make me think!: A Common Sense Approach to Web Usability" the definition "After all, usability really just means that making sure that something works well: that a person of average (or even below average) ability and experience can use the thing — whether it's a Web site, a fighter jet, or a revolving door — for its intended purpose without getting hopelessly frustrated." [7]

One of the drawbacks of a science gateway is, that it may require the installation of software and the administration of firewall properties on the users' computers. To relieve the users from these issues, a web-based science gateway offers access to pre-configured tools via a web browser. Users are enabled to customise views and tools and their preferences are stored. They only need a computer that can be connected to the Internet and an installed web browser. Hardware configuration, software installation, and firewall issues are resolved on the web servers' side. The users are thus enabled to focus on their particular research questions and interrelated computational methods. Furthermore, most of the users are familiar with the World Wide Web for various applications (e.g., booking of hotels, webmail).

The evolution and wide acceptance of the World Wide Web [8] and of web-based social networks (e.g., Facebook [9]) reflects the potential of web-based science gateways for research communities. Besides features targeted to a specific domain, science gateways may offer community features to improve the communication between the gateway's users. Research communities are mostly spread geographically and appreciate tools like chatrooms, Wikis, and repositories. Such tools allow users to build up active virtual communities and they are able to discuss online, share knowledge, and forge links for further collaborations.

Science gateways supporting virtual communities and offering access to DCIs have implemented single sign-on features with the principle of trust delegation. Single sign-on concepts allow users accessing multiple resources with a unique credential and trust delegation allows the science gateway to act on behalf of the users. The characteristics of the trust delegation in a science gateway depends on the credential management of the underlying DCI. Nowadays, the security of grid middlewares is mostly based on public key cryptography using X.509 certificates and certification authorities (CA) [10]. Established trust delegation models include assertion files in SAML (Security Assertion Markup Language) [11] or GSI proxy certificates (Grid Security Infrastructure) [12]. Optimally, a science gateway only requires the users' certificates imported in the browsers' keystores for a login or the local login maps to a federated identity (e.g., in Shibboleth infrastructures [13]). Furthermore, science gateways offer features for easily creating the additional files for trust delegation. Once these files are available in the science gateway, the security infrastructure allows the users to submit jobs and workflows to the underlying DCI.

The submission of workflows requires a workflow engine to efficiently manage the process and to invoke jobs in the right order on suitable resources. A user-friendly solution is a workflow-enabled science gateway. It assists users emulating their working steps and offers tools for the whole lifecycle of a workflow; a workflow editor to create and change workflows and a graphical interface to invoke and monitor the latter via a workflow engine. When workflows are invoked on DCIs via the science gateway, it is called a workflow-enabled grid science gateway that allows the efficient management of jobs, workflows and ideally also of data. Dependent on the size of the involved datasets, it may be more efficient to invoke tools at the data location than to transfer the data to the location of the tools. Grid and cloud file systems work with several mechanisms (e.g., distributed replicas) to accelerate data access. These mechanisms are hidden from the users and a science gateway may facilitate the interface of a distributed file system in such a way as to enable them to use the latter like a local file system.

The P-GRADE portal family [14, 15], EnginFrame [16], and Galaxy [17] have been developed to serve users with the possibility of managing the whole lifecycle of workflows including efficient data management. Whereas the first two grid portals support grid and cloud infrastructures, Galaxy is a kind of toolbox for cloud infrastructures and for HPC facilities managed by batch systems like PBS (Portable Batch System) [18] and OGE (Oracle Grid Engine) [19]. It is widely used but has the disadvantage of lacking the possibility of connecting to grid infrastructures. All three are designed for generic-purpose workflows and can be employed for single application domains. In contrast to the proprietary Galaxy, P-GRADE and EnginFrame are developed on top of a portal framework. Portal frameworks aid developers with standard features like login procedures and pre-defined so-called portlets. They are based on standards like JSR168 [20] and its successor JSR286 [21], which allow for the development of portlets just one time and deploy them in different portal frameworks (e.g., Liferay [22], Pluto [23]). However, the development of portlets requires specialised skills and is time-consuming. It oftens occupies a developer for weeks or months for a single feature. The demands on a workflow-enabled grid portal for handling jobs, workflows, and distributed data in DCIs are similar regardless of which application domain is applied. They offer graphical user interfaces but these are not particularly adapted to the application domain. Hence, there is the need to develop domain-specific portlets, which are tailored especially for an application domain.

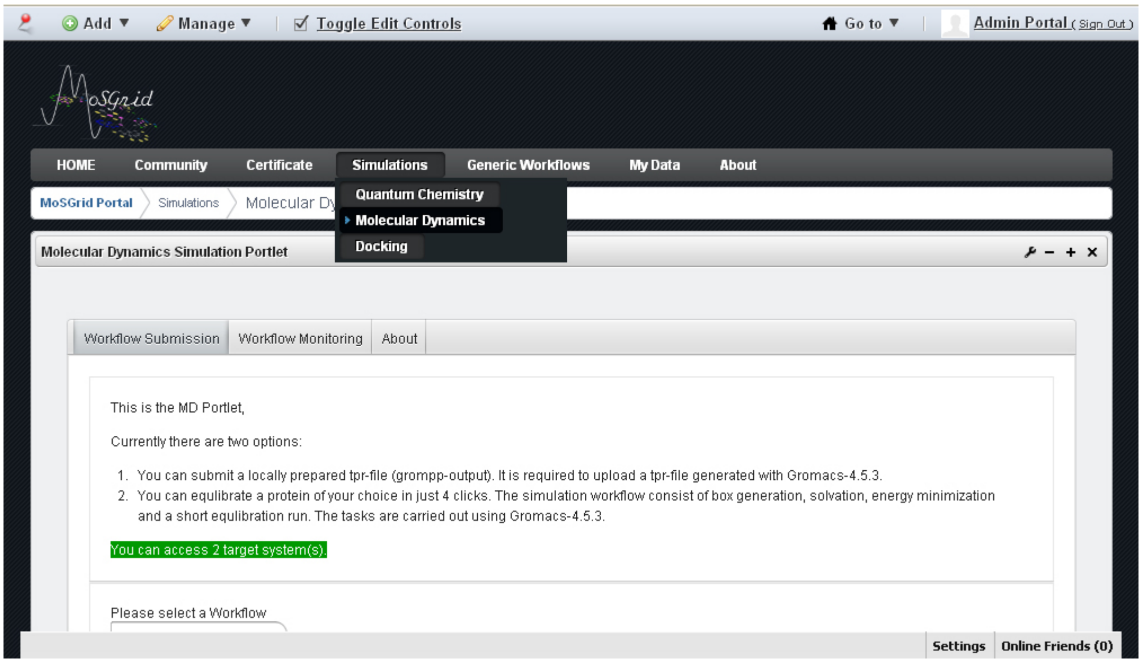
Diverse tools exist to ease the developers' work for a portlet framework. Besides Java Development Frameworks (e.g., Vaadin [24], Google Web Toolkit [25]), which aid in implementing Java web applications, three frameworks currently exist, which support the development of portlets and are actively maintained: Rapid [26], EnginFrame, and Rappture [27]. Rapid allows generating JSR168/JSR286-compliant portlets for DCIs only by creating an XML (Extensible Markup Language) [28] file. EnginFrame also offers the possibility of defining service descriptions of processes in a proprietary XML format to make a graphical interface available. In contrast to the self-contained portlets of Rapid, EnginFrame is a workflow-enabled grid portal and the process of creating portlets for DCIs relies on the overall portal framework. Analogously, Rappture is a toolkit for creating graphical user interfaces for the nanoHUB portal framework. However, the mechanisms are quite different. Rappture specifically enables each application via a proprietary infrastructure for the web. All three frameworks allow handling data-intensive and compute-intensive jobs or workflows, respectively.

Even though there are workflow-enabled grid portals and tools easing the developers' work available, it is a scientific challenge to create science gateways suitable to a specific application domain. Often communities are heterogeneous with different levels of experience with domain-related tools as well as with the use of DCIs. These different levels of experience have to be met in the science gateway. In addition, the security concept has to fit to the targeted DCIs that are available to the community. Data in structural bioinformatics is sensitive and expensive, especially in the field of drug design, and it is fundamental to protect it again misuse and to prevent data loss. Analysis of this data is mainly performed via workflows. Such workflows managed in diverse workflow engines differ from a syntactic point of view but can be re-used from a semantic point of view within a community. On the developers' side, the effort to add features in a workflow-enabled grid portal is still quite high resulting from the complex underlying infrastructure and in the case of structural bioinformatics additionally resulting from the complex theory and sophisticated methods in the application domain.

This work presents a complete solution for the structural bioinformatics community to aid end users and developers in the area of web-based science gateways. We focus on workflow-enabled grid portals and cover four major aspects in this context: security in portals with underlying DCIs, job and workflow management, migration of workflows between diverse science gateways, and the automatic creation of portlets for workflow management. The concepts, designs, and implementations have been carried out via the interdisciplinary project MoSGrid (Molecular Simulation Grid) [29, 30]. The developed web-based science gateway is based on the Liferay-version of the workflow-enabled grid portal WS-PGRADE (Web services Parallel Grid Runtime and Developer Environment), which belongs to the P-GRADE portal family.

A fundamental topic in science gateways is the authorisation and authentication of users to ensure the confidentiality, integrity, and availability of data. For this purpose, we have developed a granular security concept which covers all layers of the involved infrastructure: the user interface, the high-level middleware layer, the grid middleware layer, and the HPC facilities. The focus of this work is on the role-based user management and the credential management. We identified six main user roles ordered hierarchically regarding their authorisation level: guests, novice users, expert users, workflow developers, science gateway developers, and administrators. Guests can only access public information in the science gateway; novice users, expert users, and workflow developers are registered users with a different knowledge about the application domain and the underlying infrastructure; science gateway developers add features to the science gateway and administrators manage the science gateway. These roles within the community are mapped to technical roles in Liferay. In the area of credential management, WS-PGRADE has been extended for the use of SAML for trust delegation. SAML assertions are advantageous compared to proxy certificates since they can be limited to one entity, to a specific validity time span, and to a trust chain of a maximum length. The possibility to manage SAML assertions has been added to the certificate portlet of WS-PGRADE. The creation of the SAML assertions is solved in an integrated signed applet, which forms a flexible and secure solution only relying on a web browser and Java security.

WS-PGRADE has already provided access to various DCIs (e.g., Globus Toolkit [31], gLite [32]). However, for the MoSGrid project, we decided to use the grid middleware UNICORE 6 [33] because of its scalable, service-oriented architecture and its workflow engine. There are already user interfaces available like the UNICORE Commandline Client (UCC) [34] and the rich client [35] based on Eclipse which both demand the installation of software on the users' side. For the integration in WS-PGRADE, we have implemented a plugin, a so-called submitter, which allows invoking jobs to UNICORE 6 employing the libraries of the UCC. Furthermore, the feature has been added to support users with an automatically generated list of available tools. This feature is unique compared to other submitters of WS-PGRADE and other workflow-enabled grid portals. The submitter additionally supports UNICORE workflows to be invoked via WS-PGRADE and, thus, provides workflow interoperability. Users can seamless re-use existing UNICORE 6 workflows in WS-PGRADE. The Application Specific Module (ASM) has been developed to simplify the implementation of workflow-enabled portlets in WS-PGRADE. ASM supports to manage the whole life cycle of the execution of the workflows using the underlying DCI services via an API (Application Programming Interface). Thus, developers do not need to become acquainted with the underlying DCI services in detail but can focus on designing the layout and on implementing features for the application domain.



This is the MD Portlet,

Currently there are two options:

1. You can submit a locally prepared tpr-file (grompp-output). It is required to upload a tpr-file generated with Gromacs-4.5.3.
2. You can equlibrate a protein of your choice in just 4 clicks. The simulation workflow consist of box generation, solvation, energy minimization and a short equlibration run. The tasks are carried out using Gromacs-4.5.3.

You can access 2 target system(s).

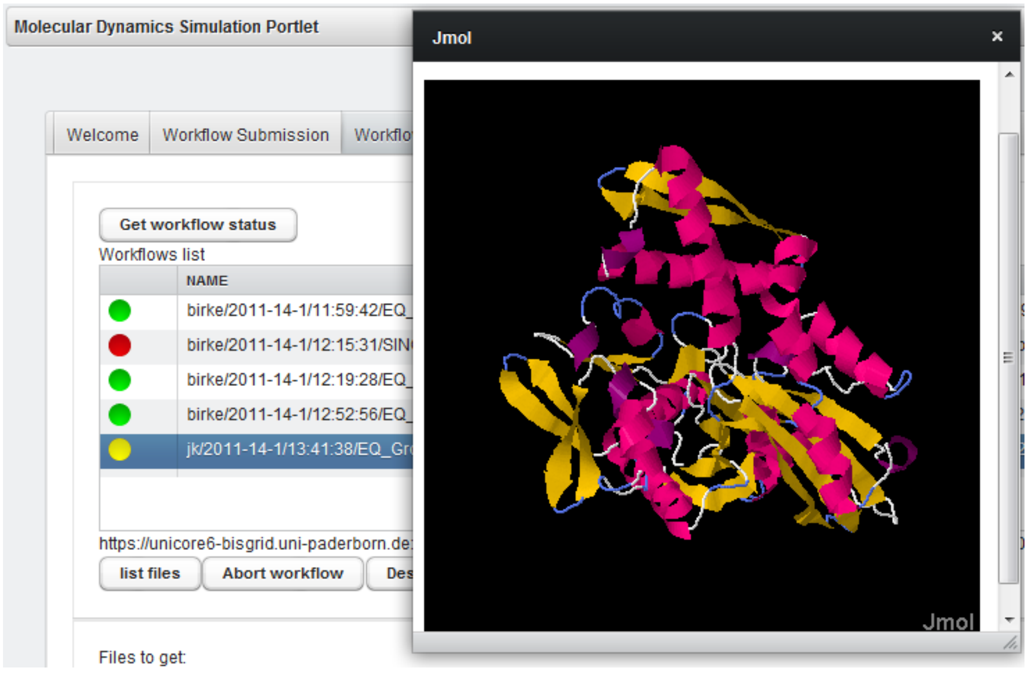
Please select a Workflow

Figure 1.1: The MoSGrid science gateway

The MoSGrid community is a heterogeneous group, which have already partially applied workflows in different workflow management systems like UNICORE and Galaxy. Migration of workflows supports the re-usability in diverse workflow management systems and relieves users from creating workflows from scratch. We achieved migration of workflows by developing a tool for the migration of Galaxy workflows to WS-PGRADE workflows. Therefore, we have implemented the export of Galaxy workflows to WS-PGRADE workflows, which can be easily imported by the users.

Furthermore, we have developed a concept for generating graphical user interfaces automatically for a WS-PGRADE portal via Rapid. Rapid has been extended to process additional to the Rapid XML file a WS-PGRADE XML workflow file. By adding this file, a portlet is created with the graphical visualisation of the workflow. Each tool can be selected in the visualisation for parameterising it for submission to the grid. If a tool also provides an XML file about its parameters to Rapid, an additional forms opens up for the specific parameters. Hence, developers are able to generate domain specific portlets including workflows only by providing XML files. They are relieved from becoming acquainted with portlet programming and can easily re-use existing workflows. Rapid with the workflow extension thus provides a complete solution for creating domain-specific portlets embedded in a WS-PGRADE portal.

MoSGrid as a D-Grid project (the German national grid initiative) [36] is employing the DCI of D-Grid and supports the utilisation of several widely-used tools and workflows in the domains of molecular dynamics, quantum chemistry, and docking (see Fig. 1.1). In addition to the features of WS-PGRADE to manage workflows and share them via a repository within the community, domain specific portlets have been developed, which are tailored to particular needs of the community. They allow submitting pre-defined domain specific workflows and the graphically visualisation of the results as well (see Fig. 1.2).



	NAME
●	birke/2011-14-1/11:59:42/EQ_
●	birke/2011-14-1/12:15:31/SIN
●	birke/2011-14-1/12:19:28/EQ_
●	birke/2011-14-1/12:52:56/EQ_
●	jk/2011-14-1/13:41:38/EQ_Gr

https://unicore6-bisgrid.uni-paderborn.de:

Figure 1.2: Visualisation of a result in the molecular dynamics portlet

Besides the possibility of managing workflows, the MoSGrid portal is being developed as a community-oriented science gateway with repositories for molecular structures and simulation results. Users of the portal will be able to share their data and results in public repositories in the cloud file system XtreemFS [37] with protection against data loss and unauthorised access.

Structure of the Thesis

This thesis is structured as follows: the theoretical background on structural bioinformatics and the distributed computing background are introduced in Chapter 2 and 3, respectively. Chapter 4 describes the evaluated and designed infrastructure for a science gateway for structural bioinformatics, which forms the basis for the developments in the subsequent chapters. Chapter 5 presents a novel credential management for science gateways facilitating SAML. An enhanced job and workflow management allowing workflow interoperability is introduced in Chapter 6. The seventh chapter provides a solution for the migration of workflows and the eights chapter goes into detail for an extension for creating automatically workflow-enabled portlets. Subsequently, we present the MoSGrid science gateway as use case for a science gateway for structural bioinformatics, which applies the concepts and developments in this work (see Chapter 9). Section 10 provides the conclusion of the presented work and gives an outlook on further steps.

2

Theoretical Background

This thesis focuses on web-based science gateways for structural bioinformatics, which is briefly presented in the following. A complete introduction is far beyond the scope of this work. Please refer to [38, 39, 40, 41, 42] for a more detailed coverage of the subject.

2.1 Structural Bioinformatics

Structural bioinformatics is a discipline combining theories and methods of various disciplines (see Fig. 2.1) to predict and analyse the structure of biological macromolecules in order to gain insights in their mechanism of function.

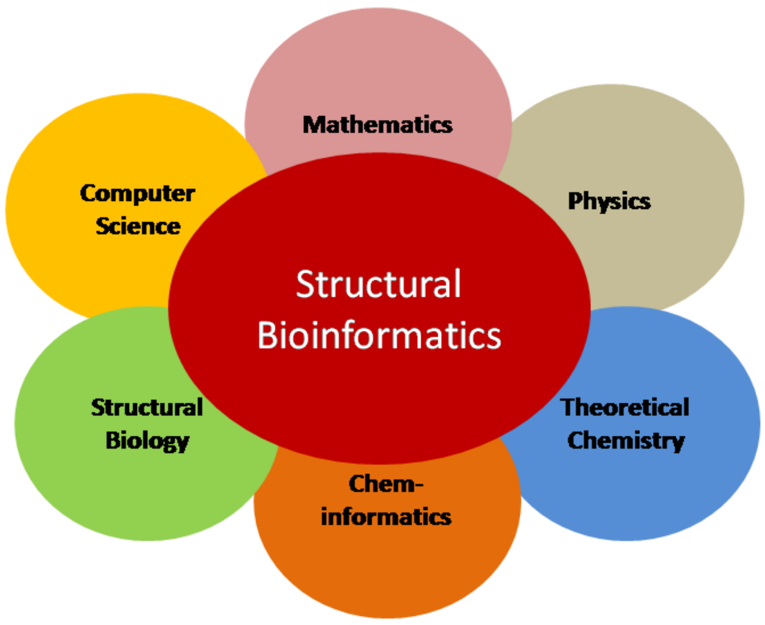


Figure 2.1: Structural bioinformatics and involved disciplines

An enormous amount of data about molecules and macromolecules is available regarding their sequence, structure, and assembly. This information is highly valuable, but only represents small part of the understanding of dynamic and functional behaviour. Gu and Bourne [38] state ”The great promise of structural bioinformatics is predicated on the belief that the availability of high-resolution structural information about biological systems will allow us to precisely reason about the function of these systems and the effects of modifications or perturbations”.

Several challenges have to be met to support researchers with computational methods for structural data. The biological data basis generated by experimental methods like X-ray crystallography, Nuclear Magnetic Resonance (NMR) spectroscopy, and electron microscopy is noisy and imperfect. It has to be examined regarding its experimental quality and to be further analysed, e.g., the assignment of peaks in an NMR spectrum to an atomic interaction is a difficult search problem. The structural models provided by crystallography are generally static, whereas the function of a molecule generally results from its dynamics. A huge amount of structural data is being experimentally created and leads to the need for databases storing the information in standardised formats and to efficiently categorise them. The Protein Data Bank (PDB) [43] is one of the first established databases. Figure 2.2 illustrates the total and the yearly growth of the number of searchable structures stored in the PDB.

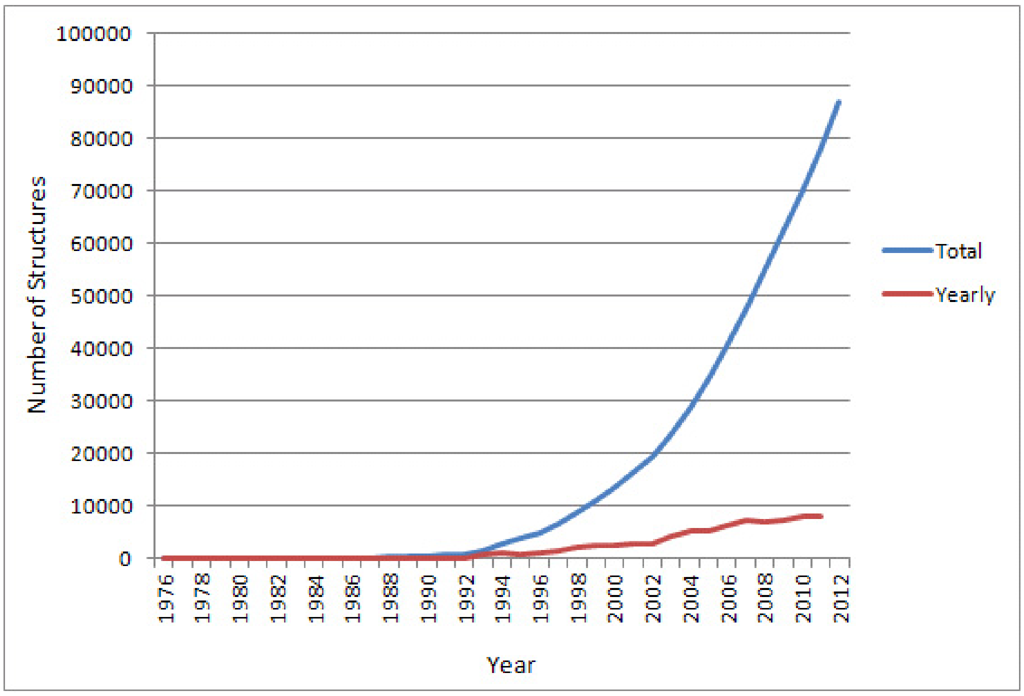


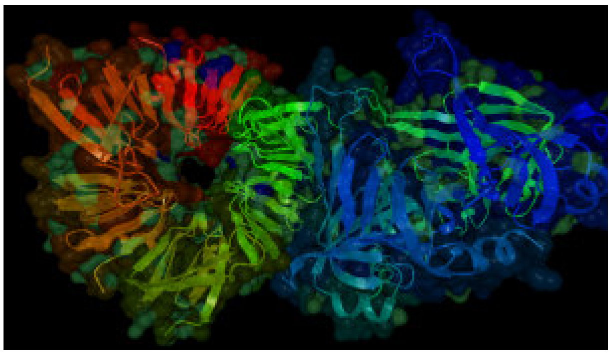
Figure 2.2: Growth of number of structures in the PDB[1]

Molecular simulations are computational methods to predict useful functional properties of chemicals, e.g., thermodynamic properties, thermochemical properties, spectroscopic properties, mechanical properties, transport properties, or morphological information. They anticipate the behaviour of molecules on basis of fundamental physical laws that determine the interaction of atoms. Dirac noted 1929 ”The underlying physical laws

[1]Numbers are taken from [44]

necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble." [45], which is still applicable nowadays. The computational methods implemented on basis of mathematical and computational models have to deal, to some extent, with a continuous search space and with NP-complete problems. Thus, heuristics and approximations are often necessary for executing molecular simulations in an acceptable time even when deployed on DCIs. Various sophisticated tools and methods have been developed for structural biology and proved to be invaluable in numerous applications like materials science and drug design.

Further challenges arise in dynamic visualisation of molecule structures and their prediction. The visualisation programs interpret and transform the raw data on basis of computational analyses and molecular simulations. It is a complex process from modeling and geometry stages to rendering stages. Various molecule viewers have been developed to support users with 2D or 3D models of structural information (e.g., BALLView [46], Jmol [47]). They offer diverse models to illustrate molecules (e.g., ball and stick model, cartoon model, surface model, see Fig. 2.3) and provide insights that cannot be obtained by simply examining raw data of structural information.


(a) Cartoon model and transparent surface

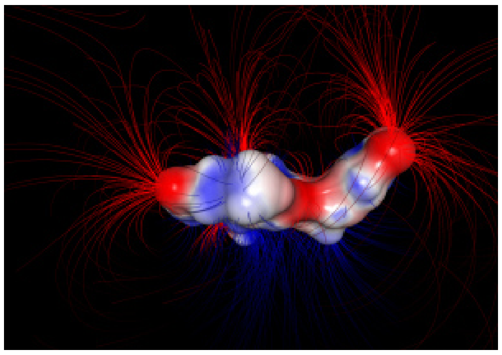

(b) Surface model with field lines, e.g., to visualize electrostatic potentials

Figure 2.3: Screenshots of molecular structures generated with BALLView[2]

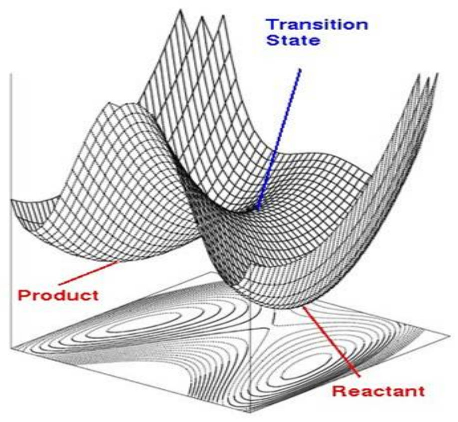
2.2 Application areas

A wide range of application areas uses structural bioinformatic methods, e.g., for prediction of protein structures or structural alignment. The focus of this work is on three selected application areas in the field of molecular simulations: quantum chemical calculations dealing with the electronic structure of molecules, molecular dynamics employing classical mechanics approaches, and docking estimating ligand-receptor interactions. Application areas of quantum chemical calculations include the explanation of observed phenomena, e.g., in spectroscopy, and the examination of the role of metal and polyester in biology. Molecular dynamics is widely used for protein structure determination and refinement. The main application of docking is in computer-aided drug design. Key problems of these three areas include

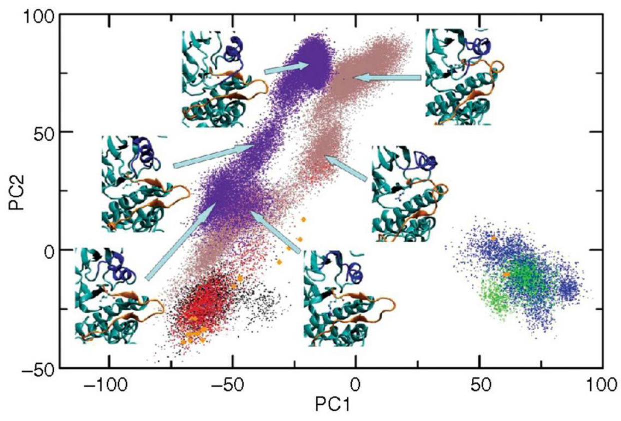
[2]Graphics are taken from http://www.ballview.org/Gallery

- the determination of the energy of chemical structures on basis of wave functions, which can be found by approximating solutions of the Schrödinger equation (quantum chemistry)
- the prediction of physical movements of atoms and molecules via molecular mechanics force fields (molecular dynamics)
- the search for a stable complex of two molecules binding to each other (docking)

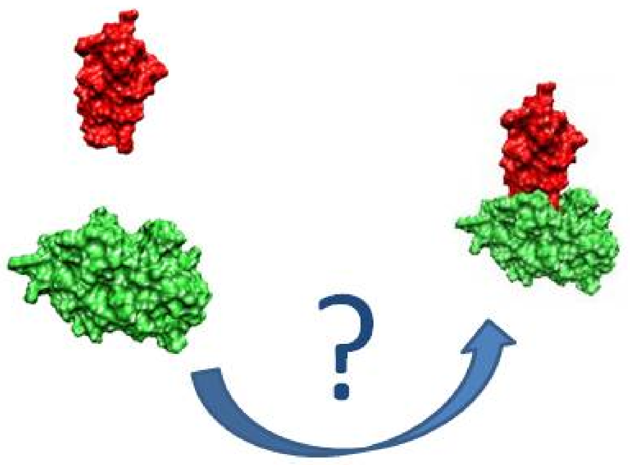
Figure 2.4 illustrates these key problems for the three application areas. The next three sections give a brief overview on the theory and methods in these application areas.



(a) Relation between relative atomic coordinates and potential energy



(b) Projections of molecular dynamics trajectories and the X-ray structures as result of principal component analysis on the EGFR kinase domain



(c) Searching for the ideal complex of the enzyme trypsin (green) with its inhibitor BPTI (red)

Figure 2.4: Key problems of the three areas quantum chemistry (a), molecular dynamics (b), and docking (c) illustrated on examples[3]

[3]Graphic (a) is taken from [48], graphic (b) is taken from [49], and graphic (c) is adapted from [50]

2.2.1 Quantum Chemistry

The scientific community using quantum chemistry encompasses inorganic, organic, and physical chemists, among others. Quantum chemistry studies and simulates electronic behaviour of molecules on basis of approximations of the Schrödinger equation [51]. The Schrödinger equation is the eigenvalue equation

$$H\Psi = E\Psi$$

where H is the Hamiltonian operator (sum of operators corresponding to the kinetic and potential energies of a system), Ψ is the wave function (characterises particles' motions, used to govern the probability of particles' properties like the location of an electron close to a nucleus), and E is the energy of the system. A few standard cases exist like a particle in a box or the hydrogen atom, which allow solving the Schrödinger equation exactly. For any poly-electronic system and, thus, any molecule, the real solution of the equation can only be estimated, which has lead to a range of different approximations. They can be divided in two major groups: *ab initio* (or first-principles) methods and semi-empirical methods. *Ab initio* calculations rely on basis sets of parameters, which are mostly not complete or partially estimated to solve equations of the underlying complex theory exactly. In contrast, semi-empirical methods approximate parts or ignore terms of equations to simplify a calculation.

The Born-Oppenheimer approximation [52] considers the motions of electrons but the nuclei as fixed, which isolates the electronic structure problem from nuclear motion. The approximation relies on the property that masses of nuclei are much larger than the masses of the electrons. Thus, electrons are assumed to react almost instantaneously on motions of the nuclei. The wave function can be expressed as the product of the nuclear wave function and the electronic wave function. The total energy of a molecule is calculated via the sum of the nuclear energy and the electronic energy.

A single-electron wave function is also referred to as an orbital. Linear combinations of atomic orbitals are applied to determine molecular orbitals. Since these are concerned only with single electrons, the specification of the location of all electrons in a molecule requires N-electron wave functions combining N orbitals. The N-electron wave functions have to consider the electrons' spins and have to be anti-symmetric, which originates from the indistinguishability of electrons. If two electrons are exchanged, the wave function is required to change sign. Good approximations of N-electron wave functions can be achieved via atomic and molecular orbitals. The variation principle allows evaluating the quality of different approximations; the better the wave function is approximated, the less energy is calculated for the system. However, the exact orbitals are mostly not known and have to be approximated as basis sets. Various methods have been developed for creating basis sets relying on experimental knowledge or knowledge about simpler systems than the targeted ones.

Hartree-Fock (HF) equations [53] are integro-differential equations, which are used to solve the minimisation problem for the energy via N equations of N spin orbitals under the constraint that the orbitals are orthonormal to each other. In this method, the wave functions can be written as the product of molecular orbitals with average charge distribution, as so-called Slater determinants [54]. The solution of an equation for one electron typically influences the solution for the other involved electrons. The self-consistent field (SCF) [55] approach generates basis sets via multiple iterations exchanging operators and lowering the total energy until the solutions for all electrons stay unchanged. The perturbation theory offers a set of approximation schemes for characterising complex systems via the knowledge about a simpler system, for example, the Møller-Plesset perturbation [56] adds electron correlation effects, coupled-cluster methods [57] add excitation operators to improve the HF method. In general, HF equations are determined differently for atoms and molecules. They can be solved numerically for atoms under the constraint that electrons are spherically symmetrical distributed. These simplified cases form the basis for analytical approximations of more complicated systems. For molecules, the linear combination of atomic orbitals (LCAO) [58] is a popular strategy, which defines each spin orbital as linear combination of orbitals. The Roothaan-Hall equations [59, 60] apply matrices instead of integro-differential equations to closed-shell molecules or atoms with N electrons in $N/2$ orbitals and, thus, permits the solution of the equations with linear algebra methods. Further approaches replace atomic orbitals by Gaussian functions. An advantage for the calculation is that the product of two Gaussian functions can be formulated as one Gaussian function. However, Gaussian functions do not have a cusp at the origin and they decay towards zero more quickly than the orbitals. The resulting errors in the approximations are unacceptable for single Gaussian functions replacing orbitals. Thus, each orbital is equated with linear combinations of Gaussian functions.

Generally, basis sets rely on calculations of atomic functions and minimal basis sets contain one basis function per atomic orbital. STO-nG (Slater-Type Orbitals) [61] are minimal basis sets in which n Gaussian functions express one orbital. Since at least three Gaussian functions are required to represent an orbital, STO-3G is the minimum for a minimal basis set. These approximations have several deficiencies like calculating the same number of basis functions for all atoms independent of the number of contained electrons or lacking the ability to describe non-spherical aspects of electronic distribution. Thus, more functions are added to the basis set. One approach is to double the functions for each orbital, called the double zeta basis. Other approaches like 3-21G, 4-31G, and 6-31G, so-called split valence basis sets [62], double the functions for valence electrons but describe core orbitals by a single function.

The computing time of *ab initio* methods is about $O(n^3)$ or worse for n atoms and most of the computing time for solving the equations is spent on calculating and manipulating integrals. Semi-empirical methods neglect or approximate some of these integrals. The zero-differential overlap approximation (ZDO) [63] sets the overlap between selected pairs of different orbitals to zero, for example orbitals of different atoms which is referred as diatomic differential overlap. The complete neglect of differential overlap (CNDO) [64] was the first implementation of ZDO and utilises Roothaan-Hall equations to model atoms. It explicitely includes only the outer valence electrons. The further developments CNDO/1 and CNDO/2 include electron-electron repulsion terms while ignoring many of them, approximating some, and filling a few with experimental data. The neglect of diatomic differential overlap model (NDDO) [65] is the basis for the modified neglect of diatomic overlap model (MNDO) [66]. In addition to ignoring the integrals for calculating orbitals of electrons in different atoms, MNDO allows only using monoatomic parameters. By extending the method with d orbitals, the so-called MNDO/d method, organometallic compounds can be analysed. MNDO has the tendency to overestimate the repulsion of atoms at close separation distances. To address this drawback, the Austin Model 1 (AM1) [67] and the parameterised model number 3 (PM3) [68, 69] have been derived from the MNDO model. AM1 and PM3 differ regarding the parametrisation of the functions. AM1 fills some parameters with spectroscopical values, whereas PM3 interprets them as optimisable values. The reliability of results generated via semi-empirical methods has been gradually improved with enhancing the methods. However, they have still some drawbacks for calculating properties of molecules often originating from missing parameterisation for a particular property. It is fundamental for users of these methods to examine carefully the results and to be aware that the calculations could deliver anomalous results for certain types of systems.

Density functional theory (DFT) [70] relies on the characteristics of systems that the ground-state energy and other properties are uniquely related to the electron density. Instead of calculating the full N-electron wave functions like HF methods, DFT methods approximate the total electronic energy and the overall electronic density distribution. DFT methods iterate over the electronic density functions via the SCF approach. The functions may incorporate Gaussian functions, Slater-type orbitals, or numerical basis functions. Numerical basis functions calculate analytical gradients of cubic spline functions derived from a spherical polar grid on each atom. The local density approximation (LDA) [71] is based on a model that the electronic density in each point in space has the same value as in a homogeneous electron gas. Thus, the total energy only derives from kinetic energy and exchange-correlation energy. DFT methods deliver excellent results for particular systems but have shown that they are mostly inadequate for isolated organic molecules. Hybrid HF/DFT methods (e.g., the B3LYP density functional [72]) have been developed to address these drawbacks.

Tools for the area of quantum chemistry include Gaussian [73] and Turbomole [74]. Gaussian, for example, supports *ab-initio* quantum chemistry methods applying Gaussian functions, a large number of basis sets, SCF, and DFT approaches. It achieves linearised computational cost via automated fast multipole methods and sparse matrix techniques. Furthermore, it is designed for shared-memory parallelisation.

All quantum chemistry methods attempt to calculate the electronic structure of atoms and molecules to gain insights into properties of the targeted systems. For large systems, quantum chemistry methods are too time-consuming even when deployed on DCIs. Methods in molecular dynamics using force fields can be applied also for large systems in a reasonable time.

2.2.2 Molecular Dynamics

The scientific community in molecular dynamics is driven by organic and bio-organic chemistry, and structural biology, while inorganic systems with metal interactions and delocalised electrons are difficult to handle by these methods. Molecular dynamics deals with studies and simulations of molecular motions. In order to access relevant time scales, a classical mechanics approach is followed, treating molecules as interacting point masses.

Molecular mechanics relies on several presuppositions. The Born-Oppenheimer approximation allows writing the energy of a system as a function of nuclear coordinates. A simple model of interactions in a system considers bond stretching, angle bending, and torsion energies. Furthermore, insights gained in smaller systems can be transferred to larger systems like macromolecules. The potential energy of a system is described in molecular mechanics by force fields, which are mathematical functions and numerous approximations derived from experimental data. The force fields rely on fundamental physical laws and are divided into bonded interactions, which occur between atoms of the same molecule, and nonbonded interactions, which occur between atoms independent of bonds in the same molecule or in different molecules. A simple functional form of the total energy is

$$E_{total} = \underbrace{E_{bond} + E_{angle} + E_{torsion}}_{\text{bonded interactions}} + \underbrace{E_{electrostatic} + E_{vanderWaals}}_{\text{nonbonded interactions}}$$

The bonded interactions include the energy E_{bond} between pairs of bonded atoms, the energy E_{angle} between three atoms in which two are bonded to the same atom, and the energy $E_{torsion}$ if a bond between atoms is freely rotatable (see Fig. 2.5). The first two energies are usually modelled by harmonic potentials; the farther the bond distances or the valence angles are displaced from the equilibrium state, the more energy is needed. The third energy is formed by torsional potentials, which reflect the changes in energy caused by bond rotations. The nonbonded interactions are calculated between all pairs of atoms located in different molecules or located in the same molecule characterised by being separated by at least three bonds (see Fig. 2.5). Electrostatic interactions $E_{electrostatic}$ are usually calculated via Coulomb's law and represent the energy resulting from the electrostatic interaction between charges in atoms and molecules depending on the distance between the charges. In case the charges possess opposite signs, the electrostatic energy is called attractive, otherwise repulsive. Van der Waals interactions $E_{vanderWaals}$ [75] are usually approximated via the Lennard-Jones potential [76] describing an attractive and a repulsive part of nonbonded interactions, which are not covered by electrostatic interactions. They include forces between two permanent dipoles, between a permanent dipole and an induced dipole, and between two induced dipoles.

Force fields are empirical, whereas their parameters may derive from *ab initio* quantum mechanical methods. Besides the functional form, the parameterisation is fundamental to predict molecular properties. An important feature is the transferability, which allows the use of the same form and parameters for modelling a series of related molecules. Established force fields include MM2 [77], MM3 [78], MM4 [79], AMBER (Assisted Model Building and Energy Refinement) [80], and CHARMM (Chemistry at HARvard Macromolecular Mechanics) [81] force fields. The MM family distinguishes between types of carbon atoms and is widely used for the conformational analysis of small organic molecules. AMBER and CHARMM force field families have been developed for molecular dynamics of molecules and, thus, consider the coordinates of atoms as free variables. AMBER is designed for macromolecules, whereas CHARMM is applied for small molecules and macromolecules. The computation of force fields lies in the complexity of $O(n)$ to $O(n^2)$ for n atoms and the most compute-intensive tasks are the calculations of nonbonded interactions. All calculations are typically parallelised and scale well even for a large number of CPUs. Nevertheless, the method suffers from a few limitations. For example, parameterisation is complex and energies are not approximated very accurately. Hybrid quantum mechanical/molecular mechanical (QM/MM) approaches aim to combine the advantages of both methods: the accuracy of quantum mechanical methods with the speed of molecular mechanical methods. They partition the system into regions or layers, whose energies are each calculated with quantum mechanical methods, molecular mechanical methods, or a combination of both. A three-layer ONIOM (our Own N-layer Integrated molecular Orbital molecular Mechanics) [82] method, for example, uses B3LYP density functional for the inner core, HF equations for the intermediate layer, and MM3 for the outer layer of a system.

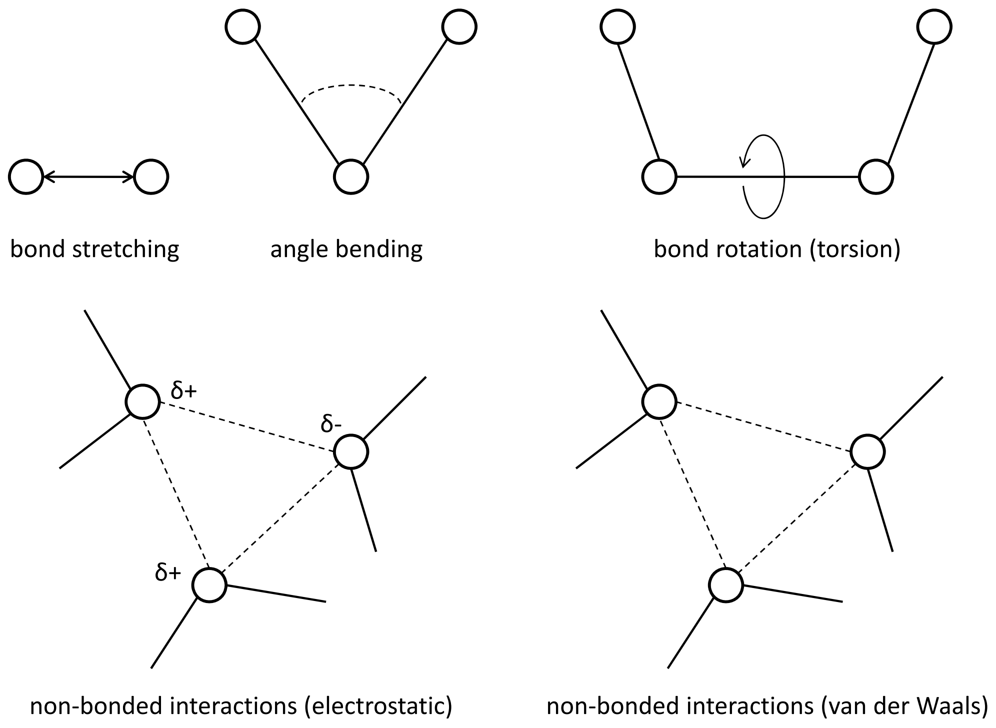

bond stretching
angle bending
bond rotation (torsion)
δ+
δ-
δ+
non-bonded interactions (electrostatic)
non-bonded interactions (van der Waals)

Figure 2.5: Schematic representation of the interactions in a typical molecular mechanics force field

Molecular dynamics is a deterministic computer simulation method, which generates configurations of macromolecules for predicting structural and thermodynamic properties with an affordable amount of computation. It applies force fields and Newton's equations of motions to analyse forces upon atoms and to calculate atomic positions as a function of time. Simulations are performed assuming in vacuo, implicit solvent, and explicit solvent states of systems. Systems in vacuo and in implicit solvent states lack information about volume and pressure in simulations, which can be addressed via computing free energies (e.g., via mean-field approaches). Explicit solvent states are simulated via adding water molecules to the system, usually performed in bounded boxes.

In molecular dynamics, time-dependent behaviour of atoms and molecules can be examined, providing insight into a system's state changing from one conformation or configuration to another for a future or past point of time. Therefore, the calculation of the forces upon atoms and their positions and velocities is performed for a series of very short time steps, typically on the order of $1 - 4$ fs. After each time step, the atoms are moved to the calculated positions and the forces are updated to determine the positions and velocities for the next time step. Thus, a trajectory is generated reflecting the temporal evolution of positions and velocities of atoms. A difficulty is to determine an initial configuration of a system with a large set of molecules and atoms and to perform all calculations. Ensembles [83] have been introduced, that consist of replications of the analysed system, considering all replications simultaneously and describing a possible state the real system might be in. A widely-used ensemble in molecular dynamics is the microcanonical or NVE ensemble that conserves the number of particles (N), the volume (V), and the energy (E). Furthermore, to reduce the complexity of the calculations of involved molecules, suitable force fields are used and/or cluster analysis is performed, which groups together homogeneous systems and chooses representatives of this group for the calculation.

Thus, running a molecular simulation consists of several steps. Firsts, an appropriate energy model has to be chosen like one of the introduced molecular mechanics models or QM/MM models. Steered molecular dynamics (SMD) [84], for example, employs external forces on proteins to manipulate their structures. It keeps parts of the configuration constant and pulls the forces on selected atoms along desired degrees of freedom. Umbrella sampling [85] supports the modification of coordinates of the potential function to avoid unsuitable configurations of a system. The resulting free energy can be calculated via the potential of mean force (PMF) [86], which describes the average force on a particle from a set of molecules. *Ab initio* molecular dynamics (AIMD) [87] uses quantum mechanical methods to calculate potentials, which causes higher compuational cost and is suitable for smaller systems and shorter periods of time only. Typically, molecular dynamics simulations have to consider effects like the cutoff of potentials and boundary effects.

After an energy model has been chosen, an initial configuration of the system has to be carefully selected to set the stage for a successful simulation. Based on the initial configuration, the equilibration of the system is performed examining the thermodynamic and structural properties. During the adjacent production phase, the simulation additionally calculates simple properties of the system and writes them to log files at regular intervals. This way, users are enabled to check whether the simulation performs promisingly. Finally, the results can be examined. In general, molecular dynamics simulations compute for the dynamics of proteins and nucleic acids nanoseconds (10^{-9} s) or microseconds (10^{-6} s), which result in 10^6 or 10^9 calculations of short time steps assuming that a time step is 1 fs long. These calculations take CPU-days to CPU-years and can be parallelised on DCIs. Good scalability is exhibited within the codes Gromacs [88] and Amber [89], e.g., tools of Gromacs are parallelised for a number of force fields, for calculating the Newton's equations and, therefore, also for the calculation of trajectories.

Methods in molecular dynamics are generally used to examine properties of macromolecules and molecules and to predict their structures. They can also be applied for analyses of ligand-receptor complexes as support for docking methods.

2.2.3 Docking

The main application of docking is in computer-aided drug design, both in academia and industry. Within the docking domain, the evaluation of ligand-receptor interactions is the main focus enhancing modern drug design. Large sets of drug-like molecules of potentially active drugs have to be managed, enabling linear scaling as the individual docking simulations are independent of each other.

Ligands are molecules interacting with receptors by direct physical interaction. Receptors are usually integral membrane proteins or soluble proteins causing a response upon the binding of a ligand. Docking methods predict the binding conformation of a ligand and a receptor and predict the binding affinity. The basic algorithm of docking methods typically include three major steps. Firsts, many plausible structures of a complex are generated. Therefore, a huge search space has to be globally or locally investigated. Global methods scan the whole receptor's surface, whereas local docking can be applied in case the binding site is already identified. Both approaches have to inspect the ligand's plausible poses around a binding site, but local docking reduces the search space on the receptor's surface and, thus, the required computing time. After generating the plausible structures, the algorithm filters out geometrically or energetically unfavourable structures and finally predicts the binding free energy ΔG of the remaining structures via scoring/energy functions. The assumption is that the better the score (the lower the calculated binding free energy), the better the remained structures approximate the real structure (see Fig. 2.6).

Scoring functions are estimations and can be grouped into three major types: scoring functions applying force fields, empirical scoring functions relying on the calculation of physical interactions, and scoring functions based on knowledge about statistical potentials of ligand-receptor complexes and their binding free energy. There is no universally valid scoring function for all plausible structures but fitting scoring functions can often be determined for specific structures.

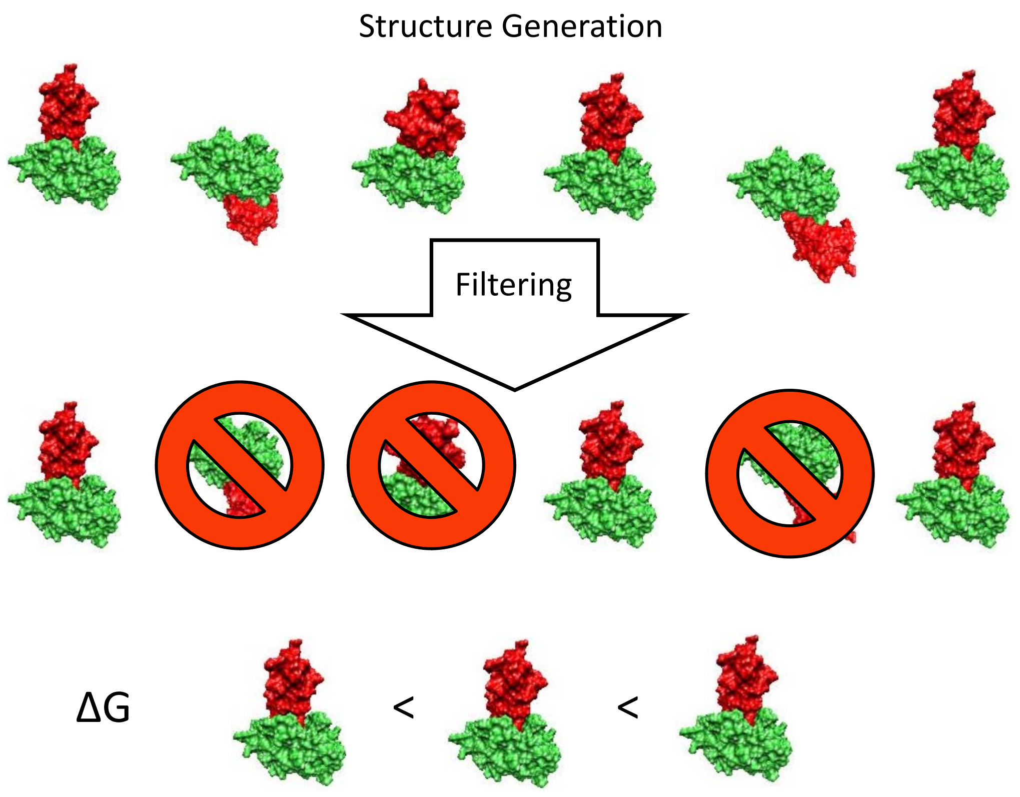


Figure 2.6: Basic algorithm of docking[4]

The lock-and-key principle [90] considers a ligand to a receptor as a key to a lock. Thus, a binding ligand possess regions of geometric complementarity to regions in the receptor's surface. In general, this geometric complementarity is described via molecular surfaces. Molecular surfaces can be defined in various ways relying on atom coordinates, e.g., van der Waals surfaces are the imaginary surfaces of the union of spheres representing

[4]Graphics are adapted from [50]

the atoms; solvent-excluded surfaces (SES) [91] contain the volume that is accessible to the solvent on van der Waals surfaces; solvent-accessible surfaces (SAS) [92] define the surface that is accessible to the center of a spherical solvent molecule. Besides relying on the geometric complementarity, rigid docking methods presume that ligands and receptors are rigid objects and preserve their structures during the binding process. The search for the best placement of the ligand to the receptor investigate six degrees of freedom (DOFs) in the motion: three translations of motions (horizontal, vertical, torsional) combined with three rotations at perpendicular axes.

Semi-flexible docking methods consider flexible ligands having additional internal DOFs, which leads to a larger search space of conformations and larger running times. The receptor is preserved as rigid, whereas domain movements and side chain flexibility is investigated for the ligand. FlexX [93] is one of the most popular codes for semi-flexible docking. It decomposes ligands into pieces and flexibly builds them up in the binding site of the receptor via a multi-greedy heuristic. FlexX offers an empirical scoring function based on the work of Böhm [94] linearly combining loss of entropy, hydrogen bonds, salt bridges, interactions between aromatic rings, and hydrophobic contacts. AutoDock [95] supports like FlexX semi-flexible docking. The 2.x releases use simulated annealing (SA) [96], whereas the 3.x releases employ a Lamarckian genetic algorithm (LGA) [97]. After the creation of a random initial population, a loop over the next generations is executed. The next generations are created via local optimisations, which apply a scoring function to select the best individuals of the last generation. The scoring function considers hydrogen bonds, van der Waals interactions, electrostatic interactions, entropy loss from ligand's DOFs, and solvation effects. To accelerate the performance of the scoring function, grid-based evaluation is employed. Glide [98] belongs to the Schrödinger Suite [99] and follows a deterministic approach for semi-flexible docking. It approximates a complete search of the positional, orientational, and comformational space available to the ligand. The search space is immensely narrowed via a grid-based approach. The energy is optimised considering fields like hydrogen bonds or hydrophobic groups. Next, the best docked poses are selected via an energy function calculating van der Waals interactions and electrostatic forces. Further refinement of the remaining few candidates is applied via a Monte Carlo sampling. CADDSuite (Computer-aided Drug Design Suite) [100, 101] as a collection of tools for drug design applies an empirical scoring function that contains terms for van der Waals interactions, electrostatic contributions, desolvation of the ligand, hydrogen bonds, and rotational entropy. Furthermore, it allows rescoring via a grid-based approach and via experimentally gained measurements of the binding free energy. Thus, the rescoring mechanism leads to more suitable scores reflecting the real docking process.

Flexible docking methods additionally consider the receptor or parts of the receptor as flexible, which increases the DOFs and the search space. FlexX-Ensemble [102], formerly known as FlexE [103], is an extension module of FlexX applying the same scoring function but considering variations of the receptor. The receptor is represented as an ensemble of structures, which is combined to a united description. The ligand is docked to the ensemble. Even though the computing time ranges in exponential complexity for some cases, the performance is better than docking the ligands against the number of singular structures represented by the ensemble.

Before a docking process can be started, it is crucial to examine whether a ligand is provided in the suitable format for the applied docking software. All semi-flexible and flexible docking tools mentioned above require a 3D conformation of the ligand. Further crucial properties depend on the docking method. AutoDock 3, for example, allows ligand files only in PDBQ format and the partial charges for all the heavy atoms and polar hydrogens must exist in the ligand. In contrast, CADDSuite supports various formats like PDB and mol2 and only data with valid bond orders can be processed. The hydrogens can be added via tools of the CADDSuite. Due to the different preparation steps, each docking software offers exact descriptions for the steps of a ligand preparation suitable to its methods. To support users within the process itself, workflow workbenches have been developed like the KoNstanz Information MinEr (KNIME) [104] and PipelinePilot [105]. They are popular in the drug design community. The Schrödinger Suite, for example, has been integrated with KNIME to provide users with standard workflows for ligand preparation, receptor preparation, and the docking process itself. However, KNIME as well as PipelinePilot only allow processing data on the users' computers. For data analysis on a large scale, workflow-enabled science gateways are needed that are able to manage workflows on DCIs.

3 Distributed Computing Background

In the following the distributed computing background of this thesis will be introduced. We present grid and cloud computing concepts and the security standards for grid computing. Furthermore, we give an introduction to workflows and web-based science gateways relevant to this work.

3.1 Grid and Cloud Computing

Structural bioinformatics as well as diverse research areas (e.g., particle physics) are characterised by data-intensive and compute-intensive problems, which demand high-performance computing to allow data analysis in an acceptable time. Hence, grid and cloud computing infrastructures have been developed to provide access to HPC facilities and services across organisational boundaries. The paradigm of grid computing evolved in the mid 1990s (e.g., Foster and Kesselmann [1]). The term grid was used in analogy to the electric power grid to indicate the main goal; to make access to computing power as easy as the access to electricity. Ian Foster extended his definition of grid computing by a three-point checklist [106], which emphasises that a grid manages distributed resources, uses standard, open, and generic-purpose protocols and interfaces, and carries out non-trivial quality of services. The demand on computing power and its efficient management has led to the development of so-called grid middlewares. Grid middlewares manage the authentication of users, the submission and monitoring of jobs, and the data. Accordingly, the underlying hardware, the involved operating systems, batch systems, and file systems are hidden from the users. The authentication procedure is based on single sign-on concepts which allow users to access multiple resources with a unique credential. The most common grid middlewares are UNICORE, Globus Toolkit, and gLite.

The most popular solutions to support data management in the grid are GridFTP [107], SRB (Storage Resource Broker) [108], SRM (Storage Resource Manager) [109], XtreemFS, dCache [110], and OGSA-DAI [111]. GridFTP was introduced by Globus whereas SRM is mostly used in gLite. SRB is popular in Globus-based grids and XtreemFS, OGSA-DAI, and dCache are available in UNICORE, Globus, and gLite infrastructures.

The paradigm of cloud computing is still evolving and various defintions can be found (e.g., 21 expert definitions in the Virtualization Journal [112]). Buyya et al. [113] describe cloud computing as ”a model on which a computing infrastructure is viewed as a cloud, from which businesses and individuals access applications from anywhere in the world on demand”. Another definition is provided by NIST (National Institute of Standards and Technology) [114]; ”Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [115]. Cloud computing provides the access to software, compute resources, and data resources as services and via virtualisations [116]. The concepts behind cloud computing are well known - from cluster computing over grid computing to utility computing [117]. The new approach is the combination of these related concepts. Cloud computing is often associated with business models, which originates from the early uptake of cloud computing services by Amazon, Google, IBM, and Microsoft. For example, Amazon provided commercialised access to their Elastic Compute Cloud (EC2) [118] in 2007. The Magellan report 2011 on cloud computing for science [119] results in that key features of cloud computing are very beneficial for science but that cloud computing needs further research and developments. However, there are already efficient, scalable, and reliable file systems for data management in clouds available like XtreemFS, Amazon Simple Storage Service (Amazon S3) [120], the Google File System (GFS) [121], and the Oracle Cloud File System [122].

3.2 Security in Grid Infrastructures

Whereas there has no security standard evolved for cloud computing so far, most of the established grid middlewares and grid file systems rely on X.509 based certificates for authentication. X.509 is an ITU standard (International Telecommunication Union) for a public key infrastructure (PKI) [123] and one of the most important standards for digital signatures. It was first issued in 1988 and relies on hierarchical certification authorities (CAs). The standard X.509 v3 has been adapted by the Internet Engineering Task Force (IETF) [124] and is commonly referred as PKIX [125].

An X.509 certificate includes the following information.

- Certificate Version
- Serial Number
- Algorithm ID
- Issuer
- Validity, Not Before, Not After
- Subject
- Subject Public Key
- Public Key Algorithm
- Subject Public Key Issuer
- Unique Identifier (Optional)
- Subject Unique Identifier (Optional)
- Extensions (Optional)

Common formats are PEM and P12 (PKCS#12). PEM files usually contain Base64 encoded certificates and P12 files usually contain certificates and password-protected private keys.

Via the IGTF (the International Grid Trust Federation) [126], a structure has been established on common policies and guidelines for managing X.509 certificates across organisations, across countries, and across continents. The coordination between grid projects on the international level ensures unique and secure authenticaton of users, services, and servers. The European part of this structure is the Policy Management Authority (PMA) EUGridPMA (European Grid Policy Management Authority) [127]. In Germany, there are two CA members of EUGridPMA: DFN (Deutsches Forschungsnetz) [128] and GridKa-Ca (Grid Computing Centre Karlsruhe Certification Authority) [129]. They centrally process the credential management for grid certificates in Germany and offer root certificates. The latter are the basis for all user certificates, service certificates, and server certificates in the German grid infrastructure and their long-term signing keys are used for the validation processes. The CAs support users with web-based tools in the creation of certificates while verification of the identities of users is handled by registration authorities (RA). Each entity (e.g., universities, companies) that wants to issue grid certificates, have to be registered as RA at one of the CAs. The CAs store the certificates, publish lists of the DNs, and email reminders to the included email address before the validity has expired (e.g., a user certificate is valid for one year). Furthermore, they manage so-called certificate revocation lists (CRL). Once a certificate is issued, it is valid until it will expire. However, for the cases that users forget the password protecting the certificate or the hardware with the private key get lost, compromised or stolen, certificates can be revoked by inserting them in CRLs. These are published and grid infrastructure providers can check whether a certificate is still allowed even if the validity is not expired. While authentication for users is handled via a certificate, the authorisation to grid middlewares is handled via VO memberships. The German NGI (National Grid Initiative) offers a web page with available VOs and the responsible VO authorities. Users can choose which VO is suitable and apply for them and for roles within the VO (e.g., developer). If the VO authority accept the application, all grid resources, which allow this VO, are available for the users.

Nowadays, there are two kinds of certificates to support the users: user certificates which are each correlated with a single user and robot certificates which are correlated to communities, to applications, or to science gateways. Robot certificates are not supported in all DCIs since their security processes rely on the distinguished names in the certificates. These are not unique for each user in a robot certificate.

However, the use of robot certificates relieves users from the multi-step application process for a user certificate. Figure 3.1 illustrates the necessary steps in the German NGI as an example for the general approach.

1. First of all, an appropriate RA has to be selected (e.g., the university the user is attending or working for) and an online request form has to be filled in.
2. Users identify themselves at the RA in person and show up with the printed and signed request. The form will be emailed automatically to the RA.
3. The RA will check the request to ensure its validity. It will be acknowledged and the users receive a private key and a user certificate.
4. For using the certificate, it has to be imported in a web browser on the users'

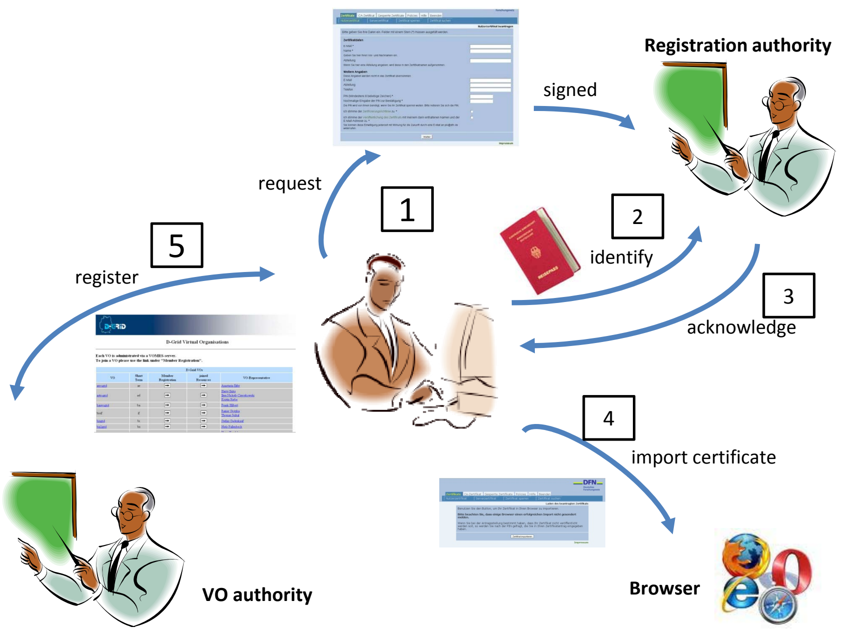


Figure 3.1: Application process for a user certificate

computers.

5. Once the certificate is imported, users can apply for a VO which fits to their application domain. If the authority of the VO accepts the application, the users will be registered for the VO.

Security concepts relying on X.509 certificates offer single sign-on to users. The user has to authenticate himself just once and gains access to all connected systems without the need for further authentication procedures. Single sign-on relies on the principle of trust delegation by which systems can be allowed to act on behalf of the user. It is used, for instance, in workflow systems, where a whole workflow consists of multiple jobs. Using trust delegation, a workflow engine acting in the name of the user, can submit individual jobs to suitable resources without further user interaction. This approach decouples job submission and user interaction.

Credentials for the authentication and authorisation of users have to be highly protected, especially in DCIs where they are used for various resources via single sign-on methods and are transferred over wide-area networks. Suitable user authentication methods ensure that credentials are not divulged to unauthorized parties and the goal of server authentication methods is to guard authentication information [130]. To additionally protect the X.509 user certificates, assertion files are used instead of the original certificates in DCIs. Nowadays, there are two main concepts available for trust delegation in DCIs; SAML assertion files and GSI proxy certificates.

SAML is an XML-based framework developed by the Security Services Technical Committee of the Organization for the Advancement of Structured Information Standards (OASIS) [131]. SAML assertion files offer a few advantages compared to proxy certificates.

A proxy certificate is always transferred along with its private key, which is extremly sensitive. Anyone, who possesses the proxy certificate, can impersonate the user. To mitigate this problem, the validity span is often severe limited, which creates new problems. Furthermore, it is impossible to reconstruct each step of a trust chain built with proxy certificates. To mitigate the problem of short validity time spans, users can upload their certificate to MyProxy servers and periodically generate proxy certificates for a certain duration of time. A MyProxy [132] server also lessens security risks since the private keys do not have to be stored on every maschine used. However, it also creates new problems. The central server has to be very well secured and it does not improve the security of GSI proxy certificates in itself. The SAML assertion file can be chained, meaning that an entity acting on the users behalf can delegate trust to yet another entity, which is then also able to act on the users' behalf. The ability of SAML trust delegation assertions to be limited to one entity, to a specific validity time span, and to a trust chain of a maximum length, are important security characteristics. Furthermore, SAML is already supported by various single sign-on infrastructures like Shibboleth which allow mapping local accounts to federated identities.

3.3 Workflows

The concept of workflows has a long history, which started with the industrialisation in the area of business workflows and has led to the development of numerous workflow management systems (e.g., Georgakopoulos et al. [133] list 50 workflow management systems). In general, a workflow can be described as a sequence of connected steps in a defined order. Each step represents a job in computational workflows. Workflow management systems can be considered as sophisticated extension of job management systems. Job management is basically concerned with the creation, submission, scheduling, and monitoring of jobs - on single computers, on clusters, and in homogeneous as well as in heterogeneous DCIs. On single computers, operating systems (e.g., Linux, Windows) take care of the allocation of system resources, process scheduling, storage management, and providing services for application programs. Clusters additionally require job schedulers or batch systems (e.g., OGE, PBS) for adequate job management and distributed file systems (e.g., Lustre [134]) for scalable storage management. Solutions for job schedulers and distributed file systems mostly also offer to be operated via wide-area networks and, thus, create homogeneous DCIs in regard to the job management. In contrast, heterogeneous DCIs necessitate grid middlewares or cloud software frameworks capable of supporting different job schedulers and file systems over wide-area networks. Workflow management systems handle additionally dependencies between jobs and are designed for creating, changing, invoking, and monitoring workflows. They depend on job management systems for invoking and monitoring of workflows, since a workflow is finally split into single jobs, which are executed and monitored on the targeted hardware.

The interest in scientific workflows increased enormously with the possibility to produce and analyse data on a large scale for various research areas and the advent of grid infrastructures. Yu and Buyya [135] stated "Scientific workflow is concerned with the automation of scientific processes in which tasks are structured based on their control and data dependencies". They point out that each workflow can be illustrated as graph and distinguish between a Directed Acyclic Graph (DAG) or a non-DAG representation of workflows. Barker and van Hemert [136] as well as Ludäscher et al. [137] compare the challenges for scientific workflows with the challenges for business workflows and give an introduction to existing workflow solutions. The main differences are seen in the goals (experimental vs. business goals), in the focus on implementation of scientific workflows vs. modeling of business workflows to develop a common understanding of a process, and in data-driven flows in science vs. process control flows in business.

Plenty of different workflow management systems and workflow languages have been developed. For example, WS-PGRADE, Galaxy, UNICORE, Taverna [138], KNIME, and PipelinePilot are widely used in the structural bioinformatics community. The workflow languages differ in their syntax and semantics. Most of the established workflow languages are XML derivatives like the proprietary workflow languages used in gUSE/WS-PGRADE and UNICORE [139]. Galaxy, in contrast, stores workflows in internal data structures and offers the possibility to export a workflow in JSON (JavaScript Object Notation) [140] format. The advantages and disadvantages of XML compared to JSON have been thoroughly discussed for years [141, 142, 143]. Both formats provide advantages over each other dependent on the purpose they are used for. XML is document-oriented whereas JSON is data-oriented. XML is characterised by flexibility, extensibility, openness, and interoperability. However, JSON posseses the same general characteristics as a format for data interchange and is simpler than XML. Many XML parsers have been developed in established programming languages, whereas JSON reflects builtin data structures of programming languages and can be created and parsed via these data structures.

The semantic of workflow languages can be examined via categorisations regarding the workflow patterns or workflow constructs the languages support. van der Aalst et al. [144] suggest to distinguish between control-flow patterns, resource patterns, and data patterns. Basic control-flow patterns, for example, include sequence, parallel split, and synchronisation. The categorisation illustrate the feasible options for workflows in each workflow language and allows comparing workflow languages on semantic level.

Besides the syntactic and semantic differences between the workflow languages, also the graphical representation of workflows differ between the workflow systems. They all use directed graphs but the form of the vertices and edges vary in several aspects. For example, WS-PGRADE illustrates vertices via square boxes in a fixed size and allows only straight lines for the edges. In contrast, the vertices in Galaxy are represented by rectangular boxes in modifiable size and the edges are represented by curved lines. In general, the user interfaces of the workflow systems provide the feature to create the graphs via a drag-and-drop mechanism. Thus, the users decide on the layout of the graph. The automatic translation of one graph representation to another may require a graph drawing algorithm. The efficient and aesthetic creation of directed graphs has being extensively studied [145]. The algorithm of Sugiyama et al. [146] is widely used, which creates a layered layout [147]. Diverse problems occuring in the algorithm are NP-hard (e.g., 2-layer crossing minimisation [148]) and have led to the developments of efficient heuristics [149]. Different criteria for the aesthetic of a graph has been followed like minimising the number of edge crossings, aligning the nodes and edges to an underlying grid, and drawing the edges as straight as possible. Purchase et al. [150] observed via empirical work that the most important criteria for users are minimising edge crossing and the aligning of the nodes and edges.

3.4 Web-based Science Gateways

In general, a science gateway can be defined as a single point of entry to a set of tools deployed across organisational boundaries. The overall goal is to increase the usability of tools and to hide the complexity of the underlying infrastructure. Web-based science gateways are characterised to require only a web browser on the user's side. They usually provide an intuitive user interface that requires much less effort to learn and use correctly. There are various peculiarities of web-based science gateways, e.g., a web page providing a collection of web services for single tools without employing a security infrastructure (e.g., the EBI web services [151]), a web page offering a collection of tools for the submission to pre-defined DCIs (e.g., WeNMR [152]), and workflow-enabled grid portals (e.g., WS-PGRADE). Workflow-enabled portals offer additionally tools for managing workflows and grid portals additionally provide access to DCIs.

Portals can be developed from scratch or via a portal framework. A portal framework combines the output of several independent web applications, so-called portlets, and arranges them in one or more customisable web pages. The output of each web application tends to be rendered surrounded by a border that gives it the appearance of residing in a window. Usually portal frameworks offer the possibility to log in to a personalised environment using a username and password. When logged in, users can customise their own pages by selecting the layout and deciding which portlets to display and where. The more advanced portals can do this using user-friendly menus and drag-and-drop.

There are a number of standardisation efforts underway to enable portlets to be used in portals from different vendors, using the principle of 'write once, deploy anywhere'. Currently there are the JSR168 standard and its successor JSR 286, WSRP (Web Services for Remote Portlets) [153] and the OpenSocial [154] standard. The choice of using these standards assures the sustainability of the developed portlets. Several open and closed source products exist that implement these standards and which can use conforming portlets (e.g., Liferay, Pluto).

Portal frameworks are not standalone applications but consist of a container with default applications and a portal interface which is deployed inside an application server (see Figure 3.2). A well-established application server is Apache Tomcat [155], which implements the Servlet 2.5 [156] and JavaServer Pages 2.1 [157] specifications and their security models. Apache Tomcat handles the access control of users and programs to resources and the integrity of data during transfers via HTTP or HTTPS. Furthermore, the application server offers role-based authorization modules and supports the login with user name and password.

User management generally deals with the authentication, authorisation, and accounting (AAA) [158] for access to and utilisation of services, local computing infrastructures, and DCIs. Authentication processes are responsible for checking the identity of a user, e.g., via a login-and-password mechanism. In contrast, authorisation processes are concerned with examining whether an action, which a user wants to perform, is allowed for the user. Some services are offered with underlying business models and users are charged for employing these services. To calculate the charge, accounting processes collect detailed data about the consumption of resources.

There are two authentication and authorisation layers in a grid portal; the first one manages the access to the portal itself, the second one manages the access to the underlying DCIs (see Fig. 3.3). Ideally, users are supported to authenticate themselves via the same

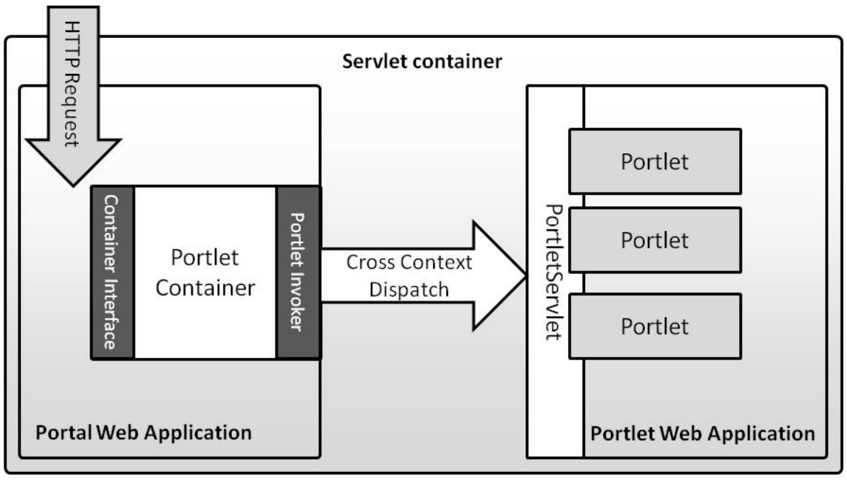


Figure 3.2: General architecture of portal frameworks

credential to the portal and the connected DCIs.

Role-based access control (RBAC) [159] is a wide-spread authorisation model in local computing infrastructures and DCIs. It allows the control of access to different services dependent on roles and simplifies the administration of large communities. Instead of assigning permissions to single users, permissions are assigned to roles. Thus, as soon as a user obtains a role, all related permisssions are granted to him. Furthermore, modifications on permission have to be performed only once for a role and not for all users affected by the modificaton. In general, all commonly used portal frameworks (e.g., Liferay, Pluto) employ a RBAC model. In a X.509 infrastructure, role-based access is primarily granted via VOs. Chadwick and Otenko implemented the infrastructure PERMIS [160] to support RBAC via attributes in X.509 certificates and storing them in LDAP. To efficiently use RBAC in a portal, it is essential to analyse which roles occur in a community using a science gateway. Roles in science gateways should consider the users' diverse levels of knowledge and experience with the application area and the maintenance and further development of a science gateway. Balasko et al. [161] present four user roles in the context of the development lifecycle of science gateways: end users, science gateway developers, grid application developers, and administrators.

3.4.1 Workflow-enabled Grid Portals

Diverse workflow-enabled grid portals have been developed, which have reached a maturity level where they can be used by various user communities in a reliable way. These portals include OGCE [162], GridPort [163], Vine Toolkit [164], P-GRADE portal family including WS-PGRADE, and Genius [165]. Grid portals typically realize a layered concept in their architecture implementing the following layers:

1. User interface layer
2. Applications, portlets layer
3. High-level (aggregation) services layer
4. Low-level DCI services layer (grid middlewares, cloud software frameworks)

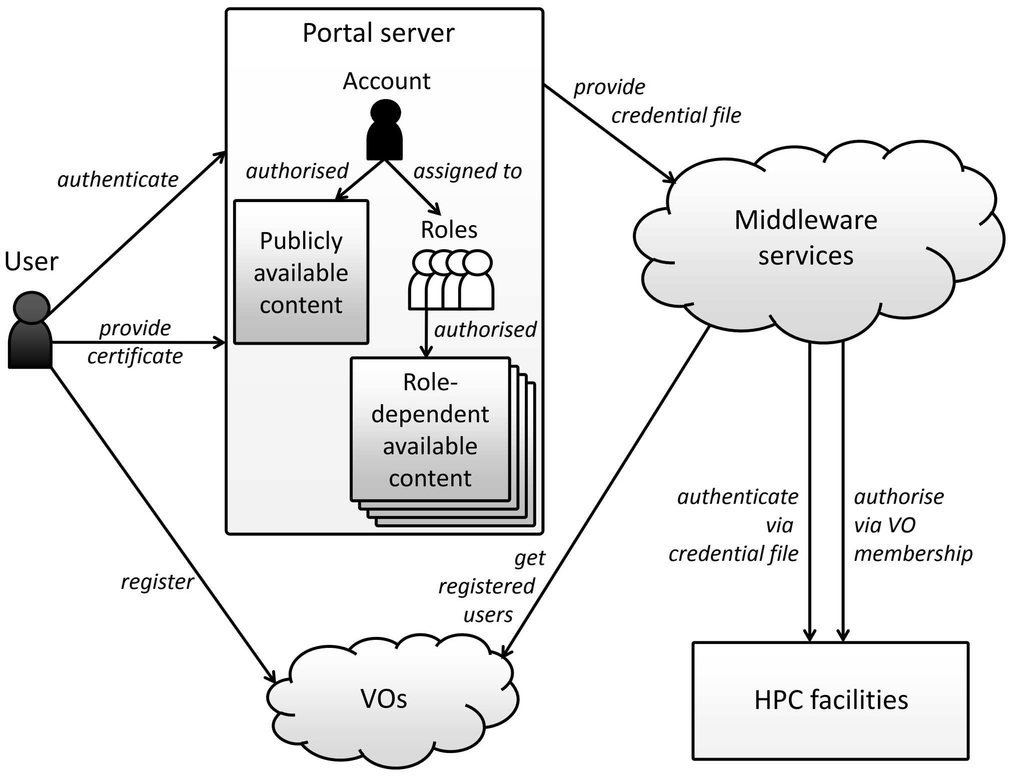


Figure 3.3: Authentication and authorisation in a grid portal

5. DCI resources layer

In fact only layers 1-3 are implemented in a portal, layers 4-5 represent the underlying DCI that is managed by the high-level services layer of the portal. A grid portal's versatility depends on how many DCIs it can manage by the high-level services layer. Once a grid portal is able support various DCI types it is another important feature if the portal can be connected to several DCIs and can run jobs simultaneously on several DCIs. This feature is particularly interesting when large parameter sweep applications should be executed and hence the parallel exploitation of several infrastructures is an advantage. This feature is especially important in heterogeneous grids like the D-Grid where Globus, gLite, and UNICORE sites are available. Unfortunately, the current grid portals are not prepared for this flexible usage of the DCIs. The only exception is P-GRADE, which can simultaneously run several jobs of a workflow in any DCI that is supported by P-GRADE.

If a portal supports several grids or VOs then it is important to provide some forms of DCI settings facilities. Unfortunately, this is an area overlooked by most Grid portals. Only Genius and P-GRADE provide some possibilities to control the usage of resources and services. Genius enables the users to select resource broker, replica location server, and MyProxy server. It could be a very useful function in an unreliable Grid infrastructure where any of these services can be down at any time. Genius also enables the connection of several VOs to the same portal and users can select the VO they would like to use. This

Table 3.1 Comparison of generic features in grid portals

	OGCE	GridPort	Vine Toolkit	P-GRADE	Genius
Grid support	Globus	Globus	Globus, gLite, UNICORE, GRIA [166]	Globus, gLite, ARC[167], BOINC [168]	gLite
Cloud support	Amazon S3, EC2	No	No	Eucalyptus [169], OpenNebula [170]	No
Simultaneous multi-DCI support	No	No	No	Yes	No
DCI settings	No	No	No	Yes	Yes
DCI data management	GridFTP, SRB	GridFTP, SRB	GridFTP, SRM, SRB, OGSA-DAI	GridFTP, SRM, SRB, OGSA-DAI	SRM
DCI resource monitoring, information service	Yes	Yes	Yes	Yes	Yes
Job execution monitoring	Yes	Yes	Yes	Yes	Yes

is very similar to P-GRADE where not only several VOs but also several types of DCIs can be connected to the same portal. It is the task of the portal administrator to configure the portal for various DCIs and VOs, then the user can add or remove resources from any of those configurations and VOs as he likes. In this way a user can customize any DCI or VO according to his preferences. For example, if a site proves to be very unreliable, then a user can remove this resource from the list of resources using his applications. Of course, it has no impact for the available resource list of other users.

DCIs are not perfectly reliable and, hence, monitoring the resources is an important feature for the users to check the status of the various resources. The monitoring information usually describes other important features of resources like capacity, load, configuration, etc. Every grid portal should be able to show this information to the users and indeed, all the investigated portals have this functionality. Furthermore, it is not only the resources the user would like to observe but also the jobs as they are executed, so application monitoring is another must in grid portals. An overview is shown in Table 3.1.

The execution of workflows is an error-prone, tedious task especially in an unreliable DCIs. Workflows enable to automatically manage these long and tedious application execution processes. A very distinguishing feature of grid portals is if they are able to support workflow development or not. Taking again the same five grid portals as example, Table 3.2 shows their workflow support characteristics. As it can be seen, only three of them provide workflow editors to develop workflow applications to be executed in the underlying DCIs.

Parameter studies on a large scale are often used in the area of molecular simulations, which can be processed via parameter sweeps. Therefore, it is also important to investigate if a portal framework is able to support parameter sweep applications both at the workflow language level and within the high-level services layer. Among the selected five grid portals

Table 3.2 Comparison of workflow features in grid portals

	OGCE	GridPort	Vine Toolkit	P-GRADE	Genius
Workflow editor	XBaya (graphical)	No	No	P-GRADE workflow (graphical)	Triana workflow (graphical)
Parameter sweep workflow execution	No	No	No	Yes	No
Workflow repository	XRegistry	No	No	DSpace [171]	No

only P-GRADE has built-in support for parameter sweep applications.

Another important aspect of generic workflow-enabled portals is whether they support the re-use of existing workflows by different members of the portal user community. This is also an often overlooked feature of portals. Usually, they are designed to support only individual users and not user communities. However, state-of-the-art portals should put emphasis on supporting user communities by enabling application developers to publish complete applications or workflow templates in a workflow repository. Then end-users can download the published applications from the workflow repository and can execute them in the connected DCIs controlled by the portal. This concept is quite new and supported only by two of the example portals. OGCE provides XRegistry as part of its workflow management system to store workflow application while P-GRADE is integrated with the open-source DSpace repository.

3.4.2 Development of Portals

Portal development is often considered expensive regarding the development time. There are several reasons for this. First, specifications are often vague, which means the development process depends on iterating through many versions to get the final version right. Second, this type of portal requires developers to know about the domain they are developing for, the general area of portal, and web development as well as to be able to work with DCIs — currently, this skill set is often not found in one person. Third, the current state of portal development tools are immature; the frameworks are often stable, but the underlying development processes are not developed as far as in other areas, making debugging and integrating components time consuming.

All this makes scientific portal development expensive relative to the user base a scientific portal base often supports. The specific goal of a scientific portal is often only of interest to hundreds of researchers, which is significantly less than would be the case for commercial portals such as Amazon, Google, and FaceBook. Therefore, the design and implementation plan of scientific portals must take into account the fact that resources for development and maintenance will be scarce and often have an end date that coincides with an externally funded project. There are three mature, actively maintained toolkits that aim to provide a cost-effective and time-efficient solution to the development and maintenance of scientific computing portals: Rapid, Rappture and EnginFrame. They offer generic solutions which can be used by researchers from any discipline to execute what-

Table 3.3 Three mature and live portlet generators that require little to no development in conventional programming languages.

	Rapid	Rappture	EnginFrame
License	GNU GPLv3	Proprietary Open Source	Commercial
Language used	XML	XML(1)	XML
Remote connection	Ssh and JSDL	VNC	VNC
Portal dependencies	JSR168/JSR286	NanoHUB	EnginFrame Server
Resource description	in portlet	not appropriate	in portal
Grid support	UNICORE, gLite, Globus Toolkit, Condor	-	UNICORE, gLite, Globus Toolkit

(1) The original application must be adapted as well

ever application they depend on. Of course, this does mean the researcher must know how to use the application and must invest significant time to learn how to go from a generic framework for job submissions to their specific task. They aim to provide a specific solution to enable existing applications on to compute and data resources with the minimal amount of development required. All three take a slightly different approach to achieving the goal of providing web-based access to applications running in DCIs. Table 3.3 shows basic information about the toolkits.

Rappture is a toolkit developed at Purdue University. Its aim is to web-enable scientific applications by providing an experiment environment specifical to each application. It combines numerical building blocks along with an infrastructure for handling user interfaces. Once the input/output is described for a simulator, Rappture handles the rest, generating a graphical interface automatically based on the description. The resulting application must then be deployed on the nanoHUB portal framework [172].

Development of a portlet takes two stages. In stage one the interface is defined in terms of inputs and outputs in an XML file. In stage two the original application is modified to include statements to allow the user interface to control the application and to show relevant output. The latter development depends on the language the original application was developed in. The connection between the remote application and the web interface is via the Virtual Networking Computing (VNC) protocol. A significant difference between Rappture and the next two solutions is that the current version is not specifically designed to enable the applications to run on large-scale compute resources.

EnginFrame is a commercial product developed by NICE s.r.l., which includes an XML language to define service descriptions of processes that are to be made available in a graphical user interface as well as the actual portal framework that provides this user interface. The framework supports many different job schedulers, which include both cluster and grid computing.

Development of an interface is through a description of its inputs, outputs, and execution specifics. Compute resources are maintained by the overall portal framework. This differs from the next solution, where each portlet is self-contained and therefore can live in any container as it knows everything it needs to successfully handle compute jobs.

Rapid is developed by the UK National e-Science Centre at the University of Edinburgh. Its philosophy is to make submitting compute jobs as easy as booking a flight or purchasing a book online. The main idea is that the whole task, the resources, and the interface are described in one XML file. This file is then translated directly into a portlet that can be used directly in a portal container. An important aspect is that Rapid allows the construction of a task that guides its users through several steps, it does not try to be a generic interface for any compute job. This way each individual portlet can be tailored to the requirements of its user base.

Portlets generated through Rapid do not depend on any specific portal platform, which is in contrast to the other two solutions and to most so-called generic job submission portals. Instead it relies on the JSR168/JSR286 standard for portal frameworks, which allows portlets created with Rapid to work in any portal framework compliant with this standard. Moreover, compute jobs are internally translated into JSDL (Job Submission Description Language) [173], which allows Rapid to connect to any system that is able to consume JSDL.

4 Evaluated and Designed Infrastructure

4.1 Introduction

The infrastructure of an intuitive and efficient web-based science gateway for structural bioinformatics using DCIs has to be carefully designed because of the complexity of the applied methods, the compute-intensive jobs, and the volume of data created and analysed. Hence, existing solutions in the area of workflow-enabled grid portals considering the different layers of the infrastructure have been evaluated. Furthermore, the infrastructure has been adhered to the interdisciplinary MoSGrid project, which aims to provide a complete solution for processing molecular simulations on grid infrastructures. This solution contains the intuitive web-based access to molecular simulations, the management of workflows, and distributed data management. In addition, the MoSGrid project intends to offer a large repository of molecular structures, chemical recipes, and results of simulations. Since MoSGrid is a D-Grid project, the consideration of the D-Grid infrastructure has been essential. Providers of resources in the D-Grid infrastructure offer access to the HPC facilities via the grid middlewares UNICORE, Globus Toolkit, and gLite. Frameworks for distributed data management set the stage for efficient data repositories on DCIs. The D-Grid infrastructure supports XtreemFS, GridFTP, dCache, and OGSA-DAI.

We evaluated and designed the infrastructure on the following four layers of the architecture of workflow-enabled grid portals:

1. User interface layer
   The user interface is adapted to the different levels of experience of the users. Suitable roles have been designed in the science gateway.
2. Application, portlets layer
   Diverse portal frameworks have been evaluated considering the users' and the administrators' side.
3. High-level (aggregation) services layer
   The available high-level services regarding existing workflow-enabled grid portals have been examined.

4. Low-level DCI services layer (grid middlewares, cloud software frameworks, distributed data management)
   The available grid middlewares and frameworks for distributed data management in the D-Grid infrastructure have been considered.

Since the layers build upon each other, the following sections present the designed infrastructure from the low-level DCI services to the high-level services layer to the application, portlets layer through to the user interface layer.

4.2 Selected Distributed Computing Infrastructure

We chose the grid middleware UNICORE 6 because of its service-oriented and reliable architecture. In contrast to Globus Toolkit and gLite, UNICORE 6 contains a powerful workflow engine developed for the computational chemistry community during the project Chemomentum [174]. To support distributed data management, we selected the efficient and reliable file system XtreemFS, which is designed for DCIs in wide-area networks. It offers optimised file distribution and access via replication features on server as well as on file level.

4.2.1 Grid Middleware UNICORE

UNICORE is an integrated grid middleware system, that includes a full software stack from clients to several server components down to components for accessing compute or data resources. Its fundamental ideas are abstraction of site-specific details, openness, interoperability, operating system independence, security, and autonomy of resource providers. Furthermore, the software is easy to install, configure, and administrate. UNICORE is being deployed and used in a variety of use cases, ranging from small projects to large multi-site infrastructures involving high-performance computing resources. UNICORE has been under development in several German and European projects since 1997. Its current version UNICORE 6 is based on web services, using many XML-based standards such as JSDL, XACML (eXtensible Access Control Markup Language) [175] and SAML 2.0.

UNICORE consists of four tiers: the client layer, the gateway layer, the services layer, and the target system layer (see Fig. 4.1). A variety of clients exist, from programming interfaces (e.g., HiLA [176]), a commandline client [34], a powerful graphical client based on the Eclipse framework [35] to portals.

UNICORE clients invoke operations as SOAP-based (Simple Object Access Protocol) [177] web services, which establish connections via the gateway. The latter is a thin authentication and routing component protecting the grid services behind it. The underlying services layer provides basic services such as registry service, resource discovery, job management, and storage access as well as higher level services like the workflow system. The services are SOAP-based web services that follow the WSRF (Web Services Resource Framework) paradigm [178]. Entities like jobs, systems, and storages are modelled as individually addressable resources. For checking whether a user is allowed to perform actions on the resources, UNICORE uses access control policies expressed in XACML. Each user identity is mapped to a set of attributes, which are then checked against a set of XACML policies. UNICORE jobs are expressed in JSDL, which supports data staging. If a job is created on a targeted UNICORE resource, UNICORE automatically provides a job work-

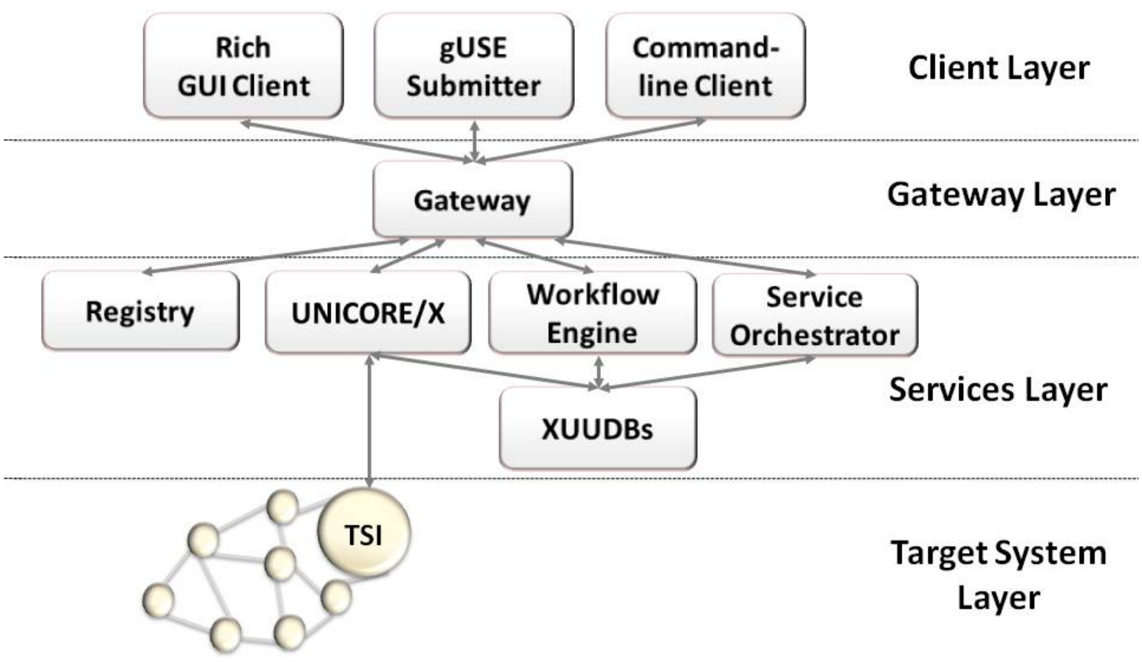


Figure 4.1: UNICORE architecture

ing directory on an HPC facility (USpace). Via the BFT (Basic File Transfer) protocol data can be uploaded to and created data can be downloaded from the USpace.

The UNICORE workflow system consists of two major services: the workflow engine deals with workflow processing, while the service orchestrator combines a resource broker with a single job execution manager. Single jobs in the workflow are passed to a service orchestrator for brokering and execution.

The target system tier connects the services from the upper tier to the local HPC facilities. It consists of the Target System Interface (TSI) installed on each connected HPC facility. It communicates with the batch system, filesystem, and local operating system to manage the jobs and the file I/O.

To support single sign-on and, thus, trust delegation UNICORE use the approach of explicit trust delegation (ETD) [179] in its dynamic style [180]. it allows the dynamic creation of jobs in the name of the user, though the trust relationships are still static. In UNICORE signed trust delegation assertions, encoded in SAML 2.0, are used to provide robust security properties. A client connects with a client-authenticated SSL (Secure Socket Layer) connection to the gateway and provides three files: the SAML assertion file, the X.509 certificate to which the trust delegation is issued by the user, and a truststore which includes the public keys of the CAs used in the UNICORE DCI. The gateway issues a SAML consignor assertion and forwards the request to the registry of the target site.

The registry checks the credential via the information provided in the assertion file created by the gateway. If applicable, the client is allowed afterwards to get a list of all available target systems and their preferences. Once a client is registered, the process of trust delegation consists of three main steps (see Fig. 4.2). First, the UNICORE/X server publishes its information to a service registry and provides information like the service address, the service type and the identity of the server. Secondly, the client gets the identity information from the service registry and finally, the client sends a request and adds the trust delegation information to the message directly to the UNICORE/X server.

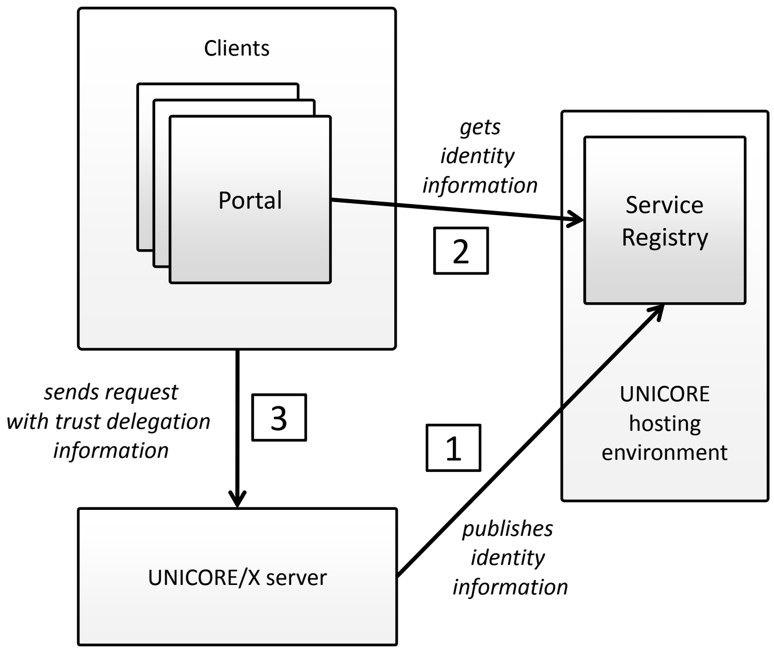

Clients
Portal
gets identity information
2
Service Registry
UNICORE hosting environment
sends request with trust delegation information
3
1
publishes identity information
UNICORE/X server

Figure 4.2: Authentication process via trust delegation with SAML assertions

SAML allows applying various security characteristics specifying the assertion in more detail. The SAML assertion file for trust delegation of a user is signed according to the XML digital signature specifications. The public part of the signing certificate is included and, thus, the security token can be validated. The DN of the issuer as well as the DN of the subject of the trust delegation are defined, the settings for the validity and the type of an assertion file can be configured.

4.2.2 Cloud File System XtreemFS

XtreemFS is an object-based grid and cloud file system, which addresses the challenges induced by high latencies and complex failure cases in DCIs. Such failure cases include hardware defects and network splits, which prevent the accessibility of servers and, thus, of the data stored on these servers. XtreemFS ensures the data integrity via replications and allows storing data among many servers. It is characterised by global namespaces and excellent scalability. According to the object-based concept, the file content and metadata are stored on different servers. XtreemFS has been developed in the EU project XtreemOS [181] since 2006 and is being further developed in other projects (e.g., MoSGrid).

The architecture of XtreemFS consists of three layers, the client layer, the server layer, and the metadata storage layer (see Figure 4.3). XtreemFS offers a FUSE (Filesystem in USErspace) [182] client, which allows implementing a fully functional filesystem in a userspace program. The development of further clients is supported via the two available

APIs, the Java API and the C++ API. The server layer contains three different types of servers: the metadata and replica catalogs (MRCs) manage all metadata of the file system (e.g., the filename, the DN of the owner, the directory tree) and keep a list of replicas; the directory service (DIR) forms the central registry of XtreemFS that provides the list of the available volumes, of the registered services, and of the available object storage devices (OSDs); the OSDs store the content of files as objects split into fixed-size chunks of data. The persistent data of DIR and the metadata are stored in an efficient key-value store applying Log-Structured Merge-Tree (LSM-Tree) [183] methods. Clients allow mounting the volumes of MRCs from anywhere in the Internet. They translate file system calls in requests to the DIR, to the MRCs, and to the OSDs and, thus, connect MRCs and OSDs. The servers and clients can be distributed worldwide.

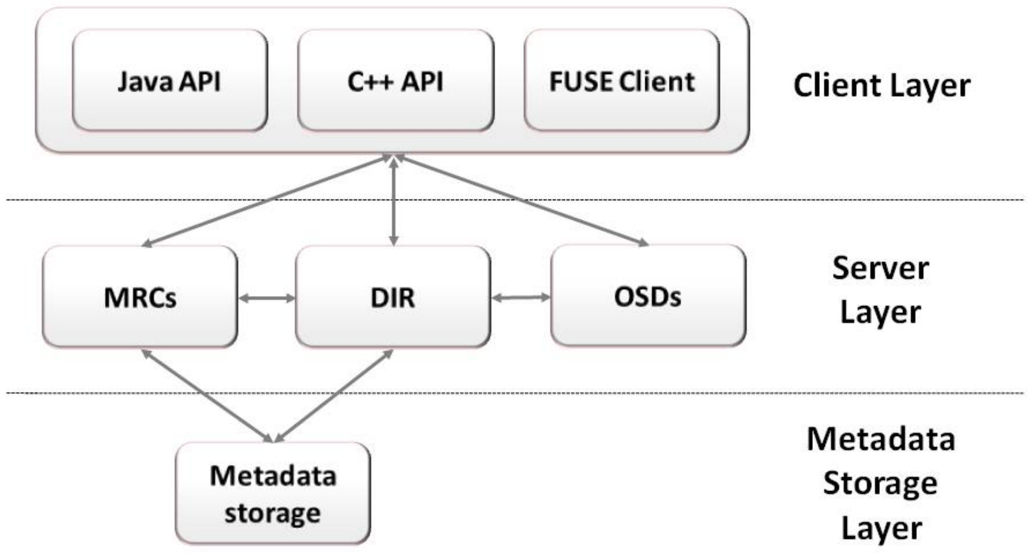


Figure 4.3: XtreemFS architecture

XtreemFS supports additionally striping of file content [184]. Data striping chops the file content in fixed-size segments, which can be accessed on different storage devices. The striping scheme RAID 0 (Redundant Array of Independent Disks or Redundant Array of Inexpensive Disks) [185] is applied to increase the performance and allows to parallelly process read and write requests for a single file. RAID 5 additionally secures the data by adding parity segments. Different OSDs may handle different parts of the striped file. The management of the striping schemes and the calculation of the parity segments is processed by the FUSE client.

XtreemFS supports X.509 certificates and authentication is handled by both OSD and MRC, while authorisation is handled by the MRC only. There is no communication between MRC and OSDs when authorising file access to a client. Instead the client requests a lease on a file from the MRC. A lease represents the authorization level and is valid for a limited timespan, and as such has to be renewed periodically. To access a file stored on an OSD, the client sends a lease confirming that it is allowed to access the file.

4.3 Workflow-enabled Grid Portals

There are various mature workflow-enabled grid portals available like P-GRADE, Genius, Vine Toolkit, OGCE, and GridPort. We compared the features of these five grid portals in Section 3.4.1. Due to the flexible DCI support, the graphical workflow editor, and the possibility to process parameter sweeps, we chose to use WS-PGRADE, the second generation of the P-GRADE portal family, as basis for the aimed web-based science gateway for structural bioinformatics. gUSE forms the high-level middleware employed by WS-PGRADE and the next section gives an overview on the architecture of gUSE and WS-PGRADE.

4.3.1 gUSE and WS-PGRADE

gUSE provides a collaborative, community-oriented application development environment, where developers and end-users can share sophisticated (layered and parameter sweep-enabled) workflows, workflow graphs, workflow templates, and ready-to-run workflow applications. The workflows are able to use a large set of virtualised, high-level DCI services. This environment is capable to provide interoperation among classical service and desktop grids, clouds and clusters, and unique web services in a scalable way.

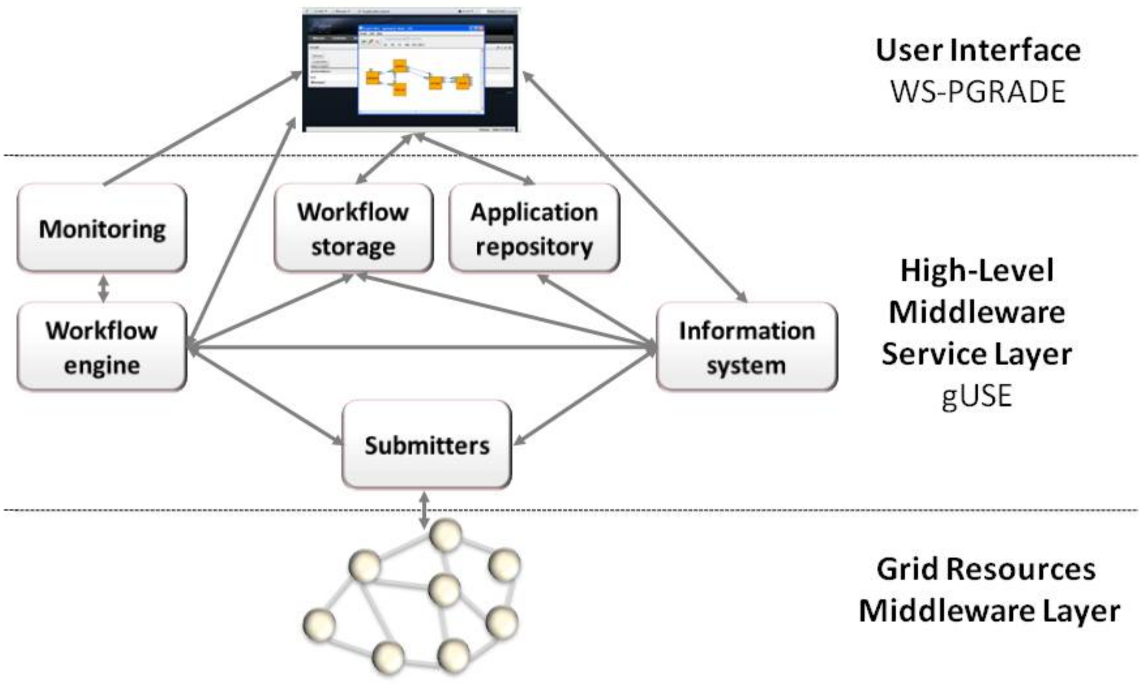


Figure 4.4: gUSE architecture

Internally gUSE is implemented as a set of web services that bind together the different components like a workflow storage, an application repository, an information system, and a monitoring system. The architecture (see Fig. 4.4) contains a workflow engine (called Zen), which provides seamless enactment of data-driven workflows and also supports extra features such as embedded workflows, timed workflows, web service type inputs, and database access. The workflow engine facilitates the management of workflows, which are based on DAGs.

The workflow engine encapsulates the single steps and invokes so-called submitters for each job. Submitters are responsible for the authentication mechanism to the underlying DCI and the management of the jobs. For each supported middleware for underlying DCIs (e.g., Globus Toolkit, gLite) a specialised submitter has been developed to provide access. Jobs within the same workflow may be configured for diverse DCIs or local resources and the workflow engine invokes each with an appropriate submitter.

WS-PGRADE is an easily usable, highly flexible, graphical user interface of gUSE. The current WS-PGRADE version employs Liferay and, thus, additionally supports JSR168- and JSR286-compatible portlets. It uses the client APIs (Application Programming Interface) of gUSE services to turn user requests into sequences of gUSE-specific web service calls. The collaborative, community-oriented application development environment of WS-PGRADE offers a graphical workflow editor and enables users to manage the whole lifecycle of workflows, from creating to changing over invoking or aborting to monitoring workflows. The creation of a workflow implies several steps. First of all, a basic graph representing the structure have to be drawn in the intuitive workflow editor. It facilitates drag-and-drop mechanisms to add jobs with input and output ports illustrating the data and connecting lines illustrating the dependencies between the jobs. As soon as a graph is uploaded to the portal server, it constitutes the basis for a workflow. It can be used to create a concrete workflow or a workflow template. The different types of workflows are configured via setting the properties of each included job, e.g., a binary, parameters, input and output files. WS-PGRADE enables users to easily administrate all preferences of a portal instance, e.g., available DCIs, VOs, storage quota for users of a portal. On basis of these preferences, lists of available DCIs and their properties are proposed via drop-down menus for the configuration of a job. Existing workflows, workflow graphs, workflow templates, and sophisticated workflow applications can be shared via a local repository.

The P-GRADE portal family has been mainly developed at MTA SZTAKI in Budapest since 2003. WS-PGRADE portals are operating and serving numerous user communities and international projects, providing access to multi-institutional grids and grid-based virtual organizations as generic DCI portals or e-Science gateways within Europe. gUSE and WS-PGRADE have become open-source in 2011 and the code is available under the GNU General Public License (GPL) at Sourceforge [186].

4.4 Evaluated Portal Frameworks

In 2010, the MoSGrid project evaluated existing open-source portal frameworks implementing the standards JSR168/JSR286. We compared Liferay, Pluto including the extension Jetspeed 2 [187], and GateIn [188] as joint project between the JBoss Community Portal Project and eXo [189]. In addition, GridSphere [190] has been a widely-used portal framework in the DCI community at that point of time. Due to the fact that the portal framework has not been actively maintained any more, it was not taken in consideration for the evaluation.The evaluation criteria were divided into user and administrator side. Since usability, acceptance and establishment of a portal by a user community is essential, the criteria on the user side (e.g., visual appearance, performance, reliability, security) were prioritised compared to the administrator side (e.g., effort for installation and maintenance, reliability, security, support by providers, documentation). Nevertheless, the observation of the administrator side contributes to the sustainability and longevity of a portal. The comparison between the three portal frameworks Liferay, Pluto/Jetspeed 2, and GateIn resulted in Liferay as favoured portal regarding look-and-feel, features, security, monitoring, and support. A small drawback was the lower performance compared to Pluto and GateIn. Performance is an important aspect in the context of acceptance by the user community. Nevertheless, the performance of the portlets is satisfying, especially in proportion to the job duration and the user interaction with the portal. Currently, Liferay is the most widely used portal framework for grid portals in the DCI community.

4.5 User Management for the Molecular Simulation Community

Portal frameworks support role-based authorisation to adapt the available features to the users' needs. Pages in a portal contain portlets or web content and, thus, contain among others the domain related features of a portal. Liferay supports various authentication standards and protocols out of the box, e.g., LDAP (Lightweight Directory Access Protocol) [191], single sign-on with CAS (Central Authentication Service) [192], and Open ID [193]. Unfortunately, it lacks features for authentication via X.509 certificates.

Liferay extends the role-based authorisation of Apache Tomcat with more granular security mechanisms for three different scopes: portal roles, organisation roles, and community roles. Portal roles and assigned permissions affect the whole portal with all its included features. Organisations reflect hierarchical structures and may present various divisions or various locations of a company. Private and public pages can be assigned to organisations as well as to communities. In contrast to organisations, communities are designed for allowing access across organisational boundaries or to pages which are applicable for all users of a portal.

The access to pages is handled fine-grained. It allows to distinguish between whether a role has access rights to a page at all or has rights for viewing, editing, or deleting content. Pages in the three mentioned scopes can only be added by administrators and are configured with default preferences. Registered users are enabled to adapt the look-and-feel and preferences like the preferred language. Furthermore, they can add private pages to the portal which are only visible to them.

To meet the needs of the molecular simulation community, the community management and the organisation management is configured in the MoSGrid science gateway. The basic view of the portal and its features are provided in the MoSGrid community view. The configuration of the organisations MoSGrid consortium users and MoSGrid developers automatically generates different views with their own portal workspaces selectable via a drop-down menu (see Fig. 4.5). We implemented six roles via Liferay: guests, novice users, advanced users, consortium users, developers, and administrators. The scope of the role guest as well as the role administrator apply to the whole portal, the roles novice users and advanced users are valid across the community MoSGrid, and consortium users and developers are configured hierarchical in organisations.

Guests are characterized by lacking an account for the science gateway. However, they can obtain information via an integrated Wiki about the project and about essential steps for getting access to the MoSGrid science gateway and its features. Currently, some demonstrations are being developed which will illustrate typical workflows in the three application domains molecular dynamics, quantum chemistry, and docking. The users are able to get a first impression of the user interface for the specific workflows and WS-PGRADE features. Additionally, there will be the option to submit pre-defined workflows

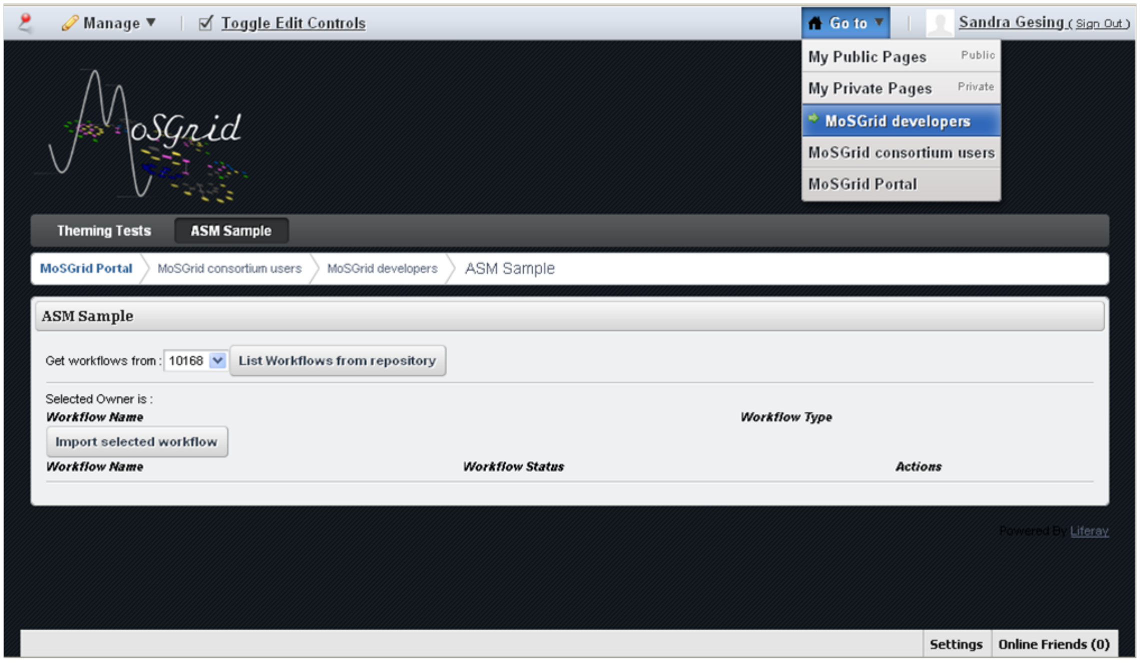


Figure 4.5: Selectable organisational views

to the local resources on the portal server. Liferay offers the option that an account can be created by an unknown user. As soon as a user has a login created for the MoSGrid science gateway, he can apply for the MoSGrid community via email and his account will be classified in the roles MoSGrid user or MoSGrid advanced user.

Novice users, in terms of novice users of tools in structural bioinformatics, are classified as MoSGrid users. This role enables to choose pre-defined workflows to become acquainted with the tools and domain-specific workflows. The latter are offered via intuitive graphical user interfaces with default parameters. The users are allowed to access pages to change the input and parameters, to invoke, and monitor the workflows.

The access to additional pages for creating and changing workflows is granted to the MoSGrid advanced user. A portlet is being developed that enable the users to switch between the role as novice user and as advanced user. They are self-dependent regarding which level of functionalities they have access to in the scope of the community features. Additionally, the administrators are relieved from the effort for assigning the role to advanced users.

The users within the consortium are assigned to advanced users and consortium users. Thus, they are additionally allowed to change grid and gUSE settings (e.g., add new DCIs for selection or configure thresholds like quotas for storage each user is allowed to occupy). These options are not shown in the community view but in the auxiliary organisation view of the portal. The users are enabled to easily switch between the views and obtain well-arranged pages with the role-dependent features.

Similarly, an additional organisation view is selectable for the MoSGrid developers. They additionally gain the same rights like the consortium users. While the view for consortium users applies new features to them, the developer view allows the deployment of new features. This view is invisible for MoSGrid users and consortium users and allows to test without involving directly the community. Even though there is a cloned test environment of the portal infrastructure available for the developers of MoSGrid, they may first add a new or extended portlet on pages of this view. They can ensure that the deployment process of a portlet including the insertion into a page can be finished without any defects. Since Liferay allows only administrators to add new pages, we have decided that MoSGrid developers are also assigned to the role administrators to gain access to this feature. Hence, they are also enabled to manage all credentials including users, organisations, and communities. However, from the organisational, theoretical point of view, they do not need most of the administrative rights but they are thus enabled to support the community even more efficient.

4.6 Basic Infrastructure

We evaluated and designed the infrastructure for the MoSGrid science gateway under consideration of the application domain, existing standards, and the D-Grid infrastructure. Due to the flexible DCI support, the graphical workflow editor, and the possibility to process parameter sweeps, the workflow-enabled grid portal WS-PGRADE using the web services of gUSE has set the stage for the science gateway. Until 2010 WS-PGRADE was developed on top of the portal framework GridSphere. In course of the collaboration between MTA SZTAKI and the MoSGrid project, developers of MTA SZTAKI have ported WS-PGRADE to Liferay, the preferred portal framework of the MoSGrid project. On the low-level DCI layer the grid middleware UNICORE 6 was chosen because of its service-oriented and reliable architecture and the powerful embedded workflow engine. XtreemFS was selected for distributed data management because of its efficiency and reliability in wide-area networks.

Furthermore, we presented in this section the role-based user concept we implemented for the molecular simulation community in Liferay.

Table 4.1 Hierarchical user roles in the MoSGrid science gateway

User role	Wiki	Demo	Workflows	Domain-specific pages	DCI settings
Guest	read	invoke	-	-	-
MoSGrid user	read	invoke	invoke, monitor	read	read
MoSGrid advanced user	read	invoke	edit, invoke, monitor	read	read
MoSGrid consortium user	read, edit	invoke	edit, invoke, monitor	read	configure
MoSGrid developer/ Administrator	read, edit	edit, invoke	edit, invoke, monitor	read, edit, deploy	configure

Via the user management of Liferay we are able to map the existing user roles in the MoSGrid community to technical roles in the portal framework. Dependent on the knowledge, the function, and responsibility for the portal, users obtain different levels of access rights. Additionally, users may have various responsibilities within the community like developing new features for or the administration of the science gateway. Hence, the technicals roles are adapted to the users' roles in the community. Table 4.1 outlines the different roles with their access rights to different features in the science gateway. Even though the basic infrastructure is tailored for the German molecular simulation community, it is not limited to one research area and re-usable on international level.

5 A Novel Credential Management for Science Gateways

5.1 Introduction

The security in grid computing infrastructures mostly relies on X.509 certificates. The certificates require that users go through a multistage application process to receive their user certificates. Therefore, users have to create essential files like SAML assertions or proxy certificates to enable the trust delegation. These procedures are time-consuming and may discourage users from using DCIs. Therefore, several approaches have been proposed to simplify the application process or to automatically generate the essential credential files.

WS-PGRADE offers features for the management of assertion files in a graphical user interface. It allows to automatically create these files via a portlet.

La Rocca et al. [165] suggest to use robot certificates for user communities. These certificates authorise users for specific tools and resources. They are handed over to the users on smartcards which demands card readers on the users' computers. Users are authenticated via login and password in a grid portal and are granted access afterwards via automatically generated proxy certificates and the smartcard to DCIs. This solution has two major drawbacks. First, the need for additional hardware on the users' side. Secondly, the additional effort for processes in grid security infrastructures, such as mapping user DNs to local accounts on HPC facilities.

The Java library GridCertLib [194] supports users of web-based science gateways to automatically obtain X.509 certificates and to use proxy certificates. The prerequisite is that the science gateway has access to a SAML assertion of a previously successful Shibboleth authentication to the science gateway. A similar concept has been implemented by the UK project SARoNGS [195]. However, the generation of a proxy certificate in the portal still needs the interaction of the user and a web service, which demands Shibboleth authentication. Both approaches are applicable for Shibboleth-based infrastructures.

There are several projects using federated access control based upon Shibboleth technologies, which are working with SAML. Users are authenticated by their home organisation Identity Provider (IdP) server and the Service Provider (SP). Shibboleth, for instance, only needs to trust the limited number of IdPs for authentication purposes. If the user is authenticated by his home authentication server, a SAML authentication assertion message is sent to the SP. Sinnott at al. [196] introduce two case studies, the DAMES portal and the EuroDSD portal. The DAMES project developed secure portals for occupational data management, educational data management, and e-Health data management. The EuroDSD portal supports a secure upload of clinical case information for disorders of sex development (DSD) and the search in this data.

The management of credentials for grid-based DCIs is not a standardised feature in portal frameworks. It demands centralised tools and services, which can be facilitated in grid middlewares and on portal servers. All briefly introduced solutions are concerned with the credential management in a portal relying upon a secure and stable authentication process in DCIs.

5.2 Credential Management Using SAML Assertions

Independent of the credential type, a basic management tool for credentials in a grid portal offers to create, browse, and delete assertion files derived from provided X.509 user certificates. An assertion file stored by the portal server sets the stage for the complete authentication process to the underlying infrastructure and, thus, supports a single sign-on process to all DCI components applying the same technology and granting access to the initiating user.

We chose SAML assertions for the science gateway because of their advantages over proxy certificates and for sustainability reasons. SAML is well-established in various security systems and the maintenance of this technology is guaranteed for many years.

The MoSGrid security infrastructure consists of four layers: the grid portal as an intuitive user interface, the high-level middleware service layer including gUSE and XtreemFS, the grid middleware layer with UNICORE, and suitable HPC facilities in the D-Grid infrastructure (see Fig. 5.1). SAML assertions had to be integrated into the MoSGrid infrastructure in several places; in WS-PGRADE, in the gUSE services, and the cloud file system XtreemFS. The focus in this section is on the extended credential management of WS-PGRADE.

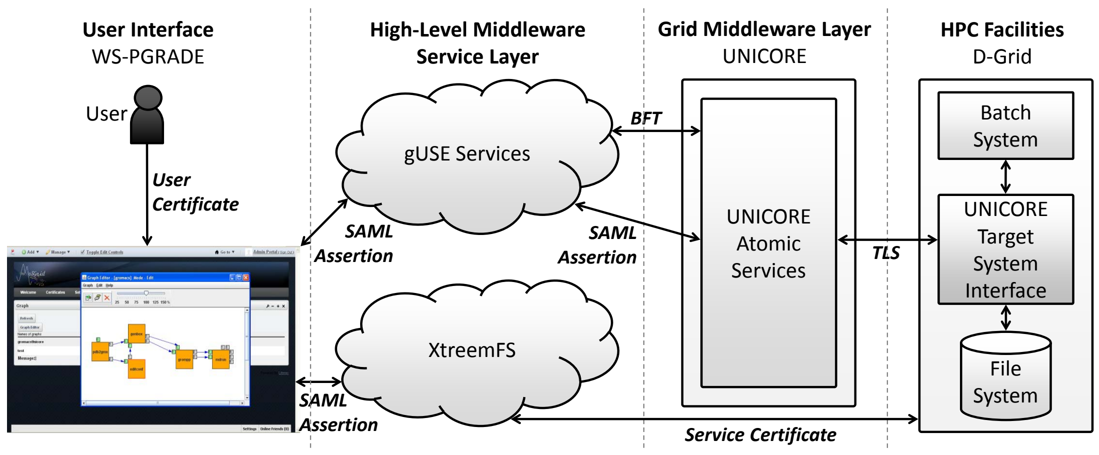
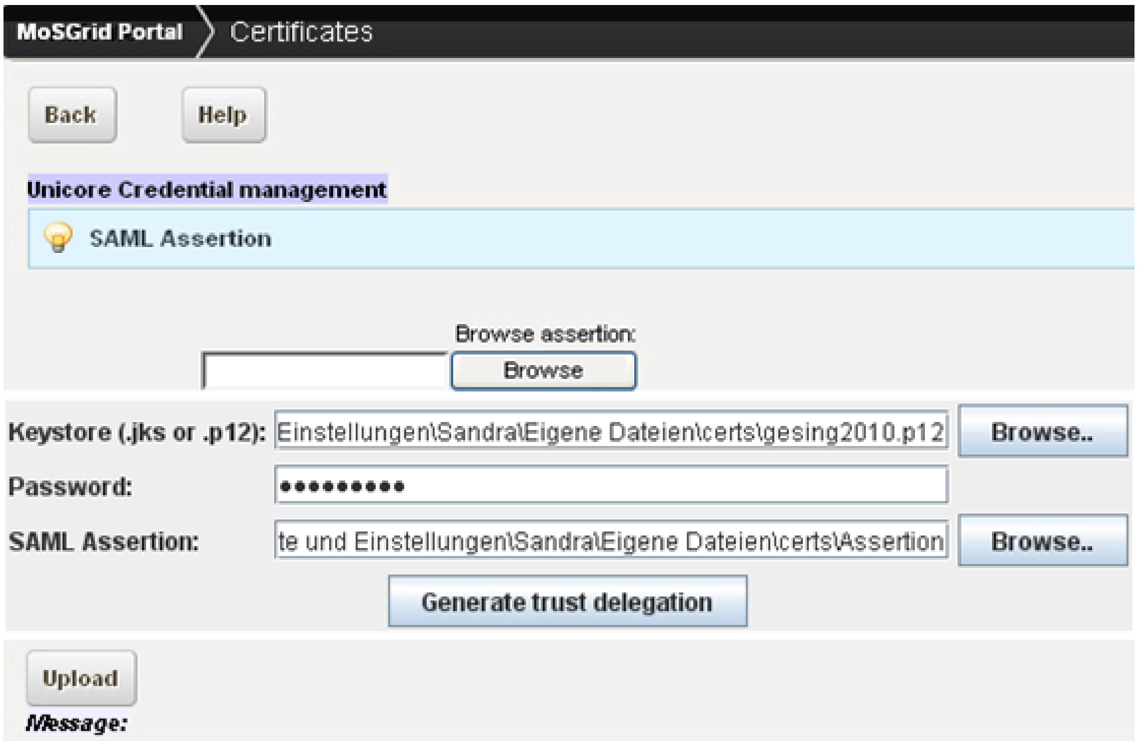


Figure 5.1: Security infrastructure of the MoSGrid science gateway

To protect the users' certificates and their corresponding credentials, it is fundamental to minimise the necessary transfers in the authentication process and the locations where they have to be stored. WS-PGRADE achieves this goal by offering a certificate portlet for credential management without uploading the certificates to the portal server. It offers to generate SAML assertions on the users' computers (see Fig. 5.2) or to upload certificates to a MyProxy server. The SAML extension has been developed in this work. The previous version of the certificate portlet only supported proxy-certificate-based authentication with MyProxy servers. A new certificate portlet was created which provides features for the diverse credential formats SAML, PEM, and P12.



Keystore (.jks or .p12): Einstellungen\Sandra\Eigene Dateien\certs\gesing2010.p12
Password: •••••••••
SAML Assertion: te und Einstellungen\Sandra\Eigene Dateien\certs\Assertion

Figure 5.2: Trust delegation generation with integrated certificate portlet

5.3 Implementation

Three options for the technical implementation in the certificate portlet were considered; the Java portal library, an applet, and a Java Web Start (JWS) applicaton. One fundamental demand for the design of the extension was that the X.509 user certificates are neither transferred to nor stored on the portal server. This has led to the elimination of the first option, the Java portal library, since this library always requires to upload data to the portal server for processing it. An applet and a JWS application allow processing data on the users' computer without involving the portal server. Both technologies are easily integrated into portlets and necessitate minimal configuration steps on the users' side. Both are relying on Java security [197] whereas applets can only be executed from within a web browser and not directly from an operating system. JWS applications have to be installed on the users' computers. Since comparisons [198] of further characteristics have not shown disadvantages of applets, the need to install JWS applications on the users' side have led to the choice of an applet.

For security reasons and since the applet needs to access files on the users' computers, we have used a signed applet. Signed applets are characterised by a digital signature, which ensures that the origin of an applet can be proven by the users. Web browsers usually accept signed applets that rely on CAs like VeriSign [199] or thawte [200]. These CAs sell SSL certificates world-wide and affirm that the certificate is valid for a person or an instance. Proprietary certificates like D-Grid certificates cause the web browers to prompt the users whether they trust the signing person or instance if the applet is called for the first time.

Hence, users are enabled to accept or deny the access to files on their computer. Via this mechanism, signed applets additionally ensure the integrity of the origin of the applet. In the MoSGrid project, the X.509 certificate of the MoSGrid science gateway is taken for signing the applet. Thus, the users can be sure to utilise an applet provided by MoSGrid. Furthermore, the MoSGrid certificate fulfills the supplemental purpose to be the recipient of the users' trust delegation in the generated SAML assertion.

UNICORE developers provided an applet as open-source, which allows to set the following options:

- location of the keystore
- password of the keystore
- name of the subject for the SAML assertion
- location of the SAML assertion
- alias for the keystore
- validity of the SAML assertion
- length of the trust chain

In this work, the applet was adapted for the use in the MoSGrid portal. The user only has to specify the location of his certificate on his computer, the corresponding password, and the location on his computer where the generated assertion file should be stored. Figure 5.3 shows a Unified Modeling Language (UML) [201] class diagram of the implementation of the applet (ETDApplet - Explicit Trust Delegation Applet). The resulting SAML assertion file adheres to the SAML 2.0 standard for X.509 certificates and sets the stage for the authentication processes in UNICORE and XtreemFS. Moreover, the SAML assertion file can be used for all other DCIs that support this standard.

The signed applet has been integrated into the certificate portlet for generating SAML assertion files locally on the user's computer. It automatically generates a SAML assertion file with the same validity as the user's certificate, which can be easily changed to a shorter validity period. An assertion file can be uploaded to the portal via the certificate portlet and will be available as the user's security token.

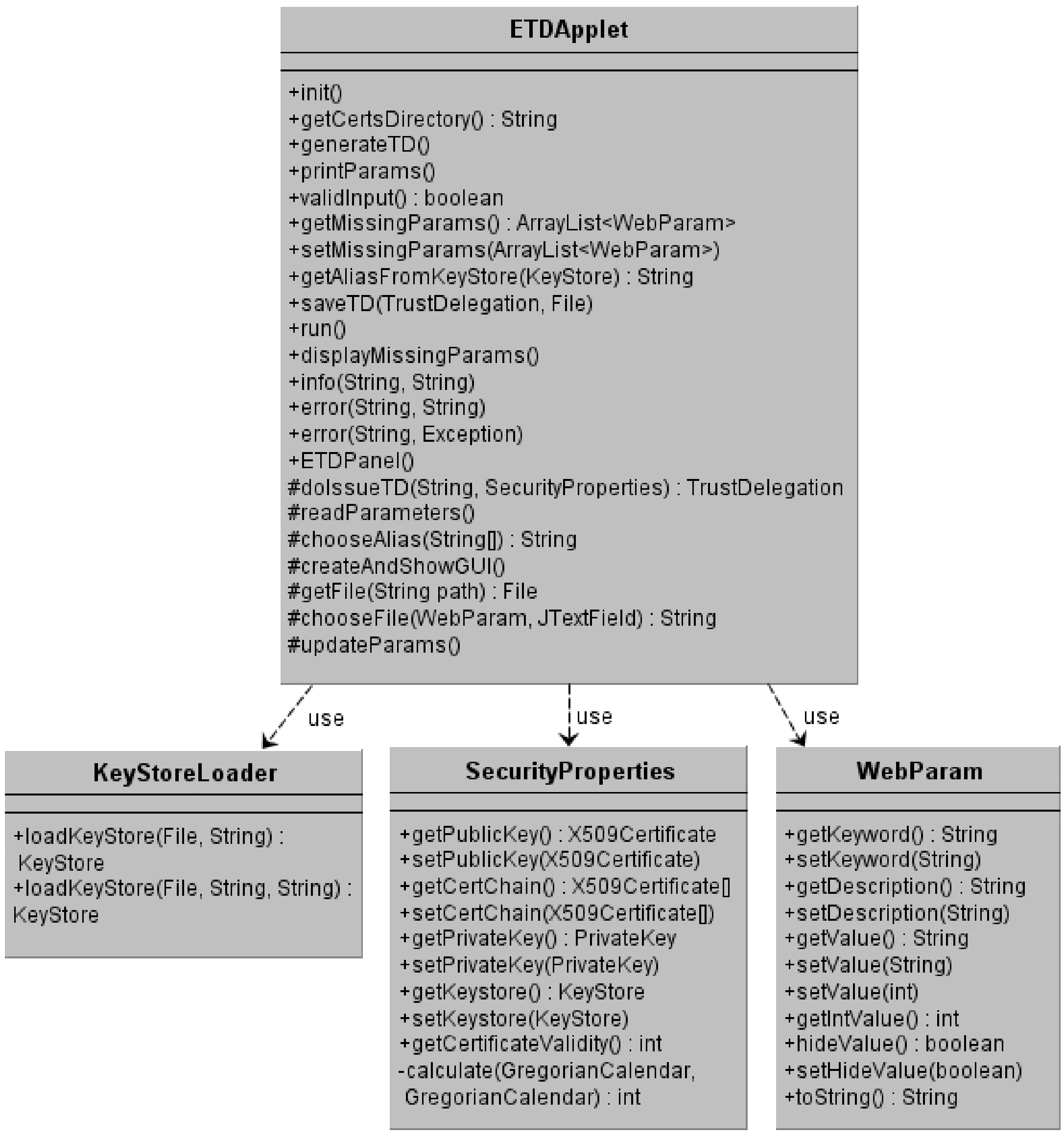

ETDApplet
+init()
+getCertsDirectory() : String
+generateTD()
+printParams()
+validInput() : boolean
+getMissingParams() : ArrayList<WebParam>
+setMissingParams(ArrayList<WebParam>)
+getAliasFromKeyStore(KeyStore) : String
+saveTD(TrustDelegation, File)
+run()
+displayMissingParams()
+info(String, String)
+error(String, String)
+error(String, Exception)
+ETDPanel()
#doIssueTD(String, SecurityProperties) : TrustDelegation
#readParameters()
#chooseAlias(String[]) : String
#createAndShowGUI()
#getFile(String path) : File
#chooseFile(WebParam, JTextField) : String
#updateParams()
use
use
use
KeyStoreLoader
+loadKeyStore(File, String) : KeyStore
+loadKeyStore(File, String, String) : KeyStore
SecurityProperties
+getPublicKey() : X509Certificate
+setPublicKey(X509Certificate)
+getCertChain() : X509Certificate[]
+setCertChain(X509Certificate[])
+getPrivateKey() : PrivateKey
+setPrivateKey(PrivateKey)
+getKeystore() : KeyStore
+setKeystore(KeyStore)
+getCertificateValidity() : int
-calculate(GregorianCalendar, GregorianCalendar) : int
WebParam
+getKeyword() : String
+setKeyword(String)
+getDescription() : String
+setDescription(String)
+getValue() : String
+setValue(String)
+setValue(int)
+getIntValue() : int
+hideValue() : boolean
+setHideValue(boolean)
+toString() : String

Figure 5.3: UML class diagram of the ETDApplet implementation. *ETDApplet* is the main class of the applet. It uses *KeyStoreLoader* to load the certificate in a specific format (e.g., PKCS#12), *SecurityProperties* to store properties like the private key and the certificate chain, and *WebParam* to store the keyword, description, and value of an applet parameter.

Since the MoSGrid science gateway only offers access to UNICORE infrastructures via SAML assertions, the certificate portlet of WS-PGRADE has been also adapted to simplify its use. The involved source files are shown in Figure 5.4. Users do not have to distinguish between diverse options and are enabled to use all relevant options regarding a SAML assertion file. Hence, the file can be generated, the generated or an existing file on the user's computer can be uploaded to the portal server, the credential can be deleted on the portal server, and its details can be browsed, for example to inspect the duration of the validity.

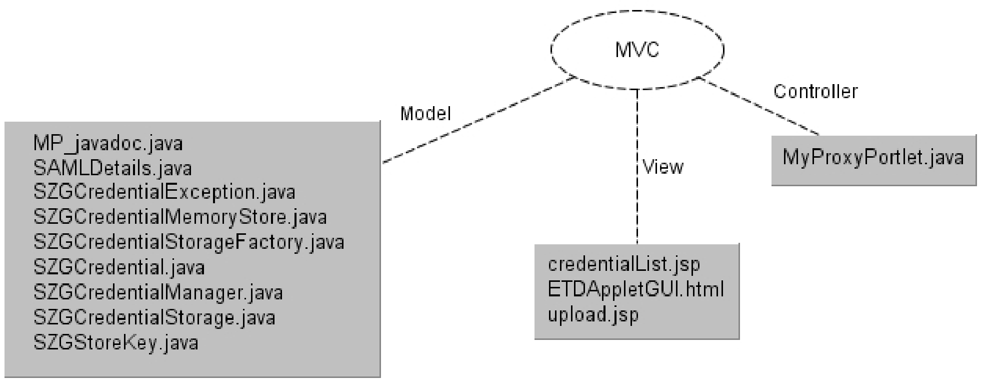


Figure 5.4: Source files involved in the adaption of the certificate portlet

5.4 Results and Discussion

The credential management for science gateways relies on the security infrastructure of the connected DCIs. This work is focused on grid-based DCIs, which authentication mechanisms are based on X.509 certificates. Two main types of secure short-lived credentials exist: SAML assertion files and GSI proxy certificates. The use of SAML assertions is advantageous compared to proxy certificates. SAML trust delegation assertions can be limited to one entity, to a specific validity time span, and to a trust chain of a maximum length. Furthermore, SAML is applied in various single sign-on infrastructures like Shibboleth. We showed a solution to easily create SAML assertion files in a web-based science gateway via a signed applet integrated into a portlet. A signed applet secures the creation process and the users' certificates can remain on the users' computers without being transferred to the science gateway, a proxy server, or the DCI. Thus, the necessary data transfers and the number of locations where the credentials are stored, are minimised. This solution based on an applet is very flexible and advantageous compared to the use of the Java portal library or a JWS application. Applets only depend on the web browsers and on Java security without the need for further installations on the users' side. In addition, an applet can be easily integrated into every web-based science gateway. However, the management of the SAML assertion has to be supported in the science gateway. Every grid portal employing DCIs via single sign-on and trust delegation has to have features for managing short-lived credentials. The credential portlet of WS-PGRADE has been extended to manage SAML assertions. Only the mandatory fields have to be filled in by the users. This simplifies the process of the necessary generating and uploading steps. Hence, the credential management fulfills the security demands while offering an intuitive solution to the community.

However, there is still room for improvement. The user certificate embedded in the browser can be used for the creation of SAML assertions, which relieves the user from choosing the certificate from the local hard drive. There are solutions available or underway, which allow the automatic creation of short-lived credentials during the login procedure eliminating the user interaction for this process. The Java library GridCertLib supports users of web-based science gateways by automatically obtaining X.509 certificates and using proxy certificates. The prerequisite is that the science gateway has access to a SAML assertion of a previous successful Shibboleth authentication. This library could be adapted for the use of SAML assertions and employed in the science gateway in case it is used in an environment offering federated identities based on Shibboleth. Barbera et al. [202] developed a prototype that automatically creates SAML assertions based on robot certificates. The methods of this prototype could be applied in the science gateway as well. A drawback of such solutions is that a user is not able to influence the properties of a created SAML assertion nor to create diverse SAML assertions employing different DCIs in a single portal.

Representatives of resource providers, of CAs, and of RAs of the German NGI currently discuss to change the security infrastructure employing X.509 user certificates and to offer access to DCIs via Shibboleth. The users would be relieved from the multistage process for obtaining X.509 certificates. The foundation EGI.eu (European Grid Infrastructure) [203] presented in October 2010 the result of a questionnaire about requirements for authentication and authorization infrastructures for DCIs. The questionnaire was answered by a number of projects from different domains. One result was that the key technologies include SAML and X.509 certificates and that the goal is to bridge security domains by using for example Shibboleth. Since the science gateway already applies SAML in its infrastructure, only minor changes have to be performed for the authentication relying on Shibboleth instead of X.509 certificates.

6 Enhanced Job and Workflow Management

6.1 Introduction

Multiple workflow-enabled science gateways have been developed for the job and workflow management in DCIs. UNICORE and Taverna, for example, provide intuitive workbenches but require software installation on the users' side. Also diverse excellent workflow-enabled grid portals are available to support users via web browsers, e.g., WS-PGRADE, Genius, OGCE (see Section 3.4.1 for a comparison).

All systems apply different workflow languages and serve various user communities. The capability to invoke and nest workflows from different systems inside one system is called workflow interoperability. Workflow interoperability allows to re-use existing workflows and to dynamically target different workflow management systems. Efforts towards a standard in this field reach back to the 1990s [204]. Nowadays, it still lacks a standard but several solutions are underway to support workflow interoperability.

The project SHIWA (SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs) [205] suggests a platform based on WS-PGRADE. Within SHIWA two different approaches for workflow interoperability [206] are distinguished: coarse-grained interoperability and fine-grained interoperability. Coarse-grained interoperability applies to nested workflows of different systems, which are invoked via one system applying service calls. This approach has been already integrated into the SHIWA platform for several workflow engines, e.g. Taverna, and requires that the connected DCIs include the targeted workflow engines. Fine-grained interoperability includes the translation of a workflow language into another workflow language, which can be processed via one workflow engine. The translation can be performed directly from one workflow language into another workflow language. The translation can also be performed within a multistage process using an intermediate representation of the workflow. The SHIWA project suggests the workflow language IWIR (Interoperable Workflow Intermediate Representation) [207] for a multistage process.

A similar approach with an intermediate representation is followed by Abouelhoda et al. [208]. These authors have extended Galaxy to Tavaxy and have developed a workflow language that allows nesting of workflows. Tavaxy focuses on the integration of Galaxy and Taverna workflows. Galaxy workflows can be directly integrated into Tavaxy workflows, whereas Taverna workflows can be invoked via web service calls.

Alqaoud et al. [209] follow a message passing approach and propose the Scientific Workflow Interoperability Framework (SWIF). SWIF offers a publish/subscribe asynchronous messaging system based on web services. Workflows on a workflow engine can be published via the SWIF client. If a workflow is published, the SWIF client offers services for workflows on other workflow engines to subscribe the workflow. As soon as the published workflow is invoked, the registered workflows are notified. Even though this approach is generic, it requires a SWIF client on the users' side and the feature in the workflow engines to receive and analyse the message about the status of the published workflow and its results.

Workflow interoperability enables users to re-use workflows of different systems in one user interface. Such a user interface for the management of workflows is usually generic in workflow-enabled grid portals and offers the same look-and-feel for diverse workflows independent of the application. To extend the portal with further features and domain-specific user interfaces, additional portlets have to be developed. Ideally, a workflow-enabled grid portal provides an interface for employing the underlying workflow engine in an easy fashion. EnginFrame and nanoHUB, for example, include appropriate frameworks (see Section 3.4.2) for this purpose. Since the workflow engines and technologies facilitated in the portals are different from each other and are not based on a standard, these frameworks based on the portals are also different from each other. Hence, a framework cannot be re-used for a portal based on a different technology.

This chapter goes into detail for job and workflow management via gUSE and WS-PGRADE. We have added the support of UNICORE jobs as well as UNICORE workflows and, thus, offer workflow interoperability. Furthermore, we introduce the Application Specific Module (ASM) that provides an API for managing workflows to portlet developers, who want to extend WS-PGRADE by additional features.

6.2 Job Management

To support the possibility for users to invoke workflows on UNICORE infrastructures via WS-PGRADE, we extended gUSE as well as WS-PGRADE. A UNICORE submitter for gUSE and a plugin module for UNICORE parameters in WS-PGRADE have been developed. They employ the UCC (UNICORE Commandline Client) libraries, which are able to perform all UNICORE features available for a UNICORE client. In contrast to programming interfaces like HiLA, the UCC libraries enable clients to invoke UNICORE workflows. The option for directly using the UCC instead of its libraries inside the extensions would have led to the additional demand to offer and maintain a UCC in the infrastructure for gUSE and WS-PGRADE.

6.2.1 UNICORE Submitter

The workflow engine of gUSE performs the complete workflow management and employs submitters for the management of single jobs. One mandatory information in a job description is the targeted type of DCI, which allows the workflow engine to invoke the appropriate submitter and to supply it with the complete job configuration. The workflow engine offers a general interface for submitters. These submitters have to provide the following methods to manage the whole lifecycle of a job:

- actionJobSubmit: Submission of a job including authentication, authorization, and data staging
- actionJobAbort: Cancel a job
- actionJobOutput: Get the output of a job
- actionJobStatus: Query the status of a job
- actionJobResource: Return the resource, where the job was submitted to

The essential feature of a submitter is the submission of a job to UNICORE as illustrated by the algorithm below.

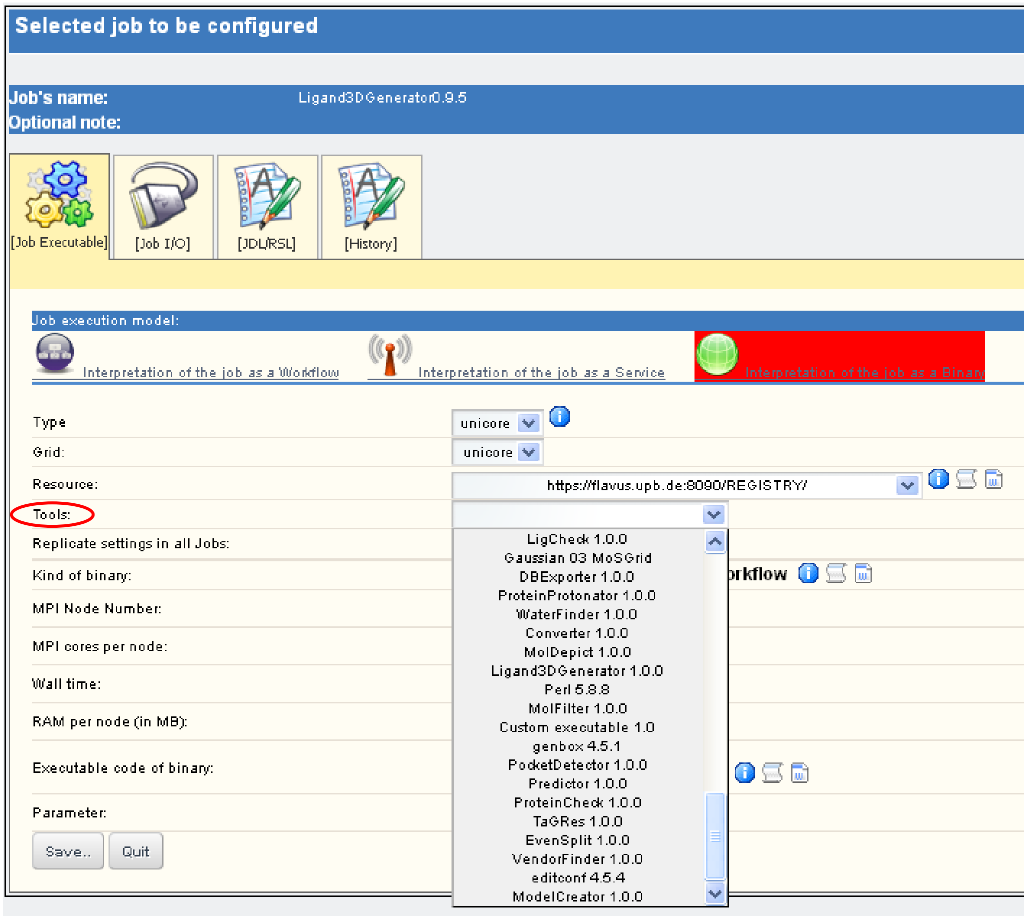
Algorithm 6.1 Submission of a UNICORE job

```
set trust delegation tokens
initialise a registry client
find/create a fitting target system service
if not fitting target system available then
    abort job submission
    exit
end if
generate JSDL
create UNICORE job
if executable is provided then
    import executable to USpace
end if
if local input data is provided then
    import data to USpace
end if
start UNICORE job
```

Three files are required to set the trust delegation tokens for a user invoking a job; the SAML assertion file created via the certificate portlet, the X.509 certificate of the portal server to which the trust delegation is issued by the user, and a truststore, which includes the public keys of the CAs used in the UNICORE infrastructure. The SAML assertion file is unique for each user, whereas the other two files are identical for all users of the same portal. As soon as a user uploads his SAML assertion file via the certificate portlet to the portal server, the submitter has access to this file. In order to create the SAML assertion file, the public key of the portal is used by the certificate portlet.

An administrator of the portal ensures that the X.509 certificate used for the trust delegation as well as the truststore are available to the submitter on the portal server. Based on these essential files, the security properties for the connection to UNICORE are settled and the submitter tries to establish a connection to a given registry. The registry checks whether the trust delegation token is valid and for which connected resources (target systems) the user is authorised. The information about authorised users is managed by an XUUDB server that maps user credentials (an X.509 certificate or a DN) to a set of security attributes fitting to the target systems. A typical security attribute is a Unix login, a role, or a Unix group.

If the user credential grants access to at least one target system, the submitter examines the job properties in order to find a fitting target system. The basic characteristics in the job definition is the application that the job has to perform. The application may either be a binary executable or an executable script. The UNICORE submitter offers a unique additional feature compared to all other submitters in gUSE. It allows users to simply provide a name of a tool instead of expecting an executable file in the job configuration. This feature is driven by the Incarnation Database (IDB, see Appendix B), a database in UNICORE that maintains the properties of a target system and its implemented tools.


Selected job to be configured

Job's name: Ligand3DGenerator0.9.5
Optional note:

[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Job execution model:
Interpretation of the job as a Workflow
Interpretation of the job as a Service
Interpretation of the job as a Binary

Type	unicore
Grid:	unicore
Resource:	https://flavus.upb.de:8090/REGISTRY/
Tools:	
Replicate settings in all Jobs:	
Kind of binary:	orkflow
MPI Node Number:	
MPI cores per node:	
Wall time:	
RAM per node (in MB):	
Executable code of binary:	
Parameter:	

LigCheck 1.0.0
Gaussian 03 MoSGrid
DBExporter 1.0.0
ProteinProtonator 1.0.0
WaterFinder 1.0.0
Converter 1.0.0
MolDepict 1.0.0
Ligand3DGenerator 1.0.0
Perl 5.8.8
MolFilter 1.0.0
Custom executable 1.0
genbox 4.5.1
PocketDetector 1.0.0
Predictor 1.0.0
ProteinCheck 1.0.0
TaGRes 1.0.0
EvenSplit 1.0.0
VendorFinder 1.0.0
editconf 4.5.4
ModelCreator 1.0.0

Figure 6.1: Configuration of a job in WS-PGRADE including the unique feature to select an available tool from an automatically generated list

Besides the executable file or tool name, the job configuration may also define various parameters for the application (e.g., the expected wall time, number of required nodes, whether it uses Message-Passing Interface (MPI) [210] for parallel execution) and command line parameters. For input and output files, two main principles are implemented: management of local files and of remote files. Local files have to be provided in the job configuration. In contrast, remote files are defined by a URL with the schema "xtreemfs://<DN of the user>/" and the file handling is taken over by UNICORE and XtreemFS. If XtreemFS is mounted locally at a target system, downloading or uploading files from or to XtreemFS can be performed by a simple local copy command by UNICORE. If an XtreemFS remote storage is available to the target system, the UNICORE/X server can be configured to access this storage.

If a target system is compatible with the requirements of the job, a JSDL file with the job information is generated and a UNICORE job is created. As a result, UNICORE automatically provides a working directory (USpace) for the created job on the target system. The USpace is solely accessible to the user who invokes the job. If an executable or local input files are provided, the submitter uses the BFT (Basic File Transfer) protocol for uploading these files belonging to a job to the USpace. In contrast, the file handling of remote files is taken over by UNICORE and XtreemFS.

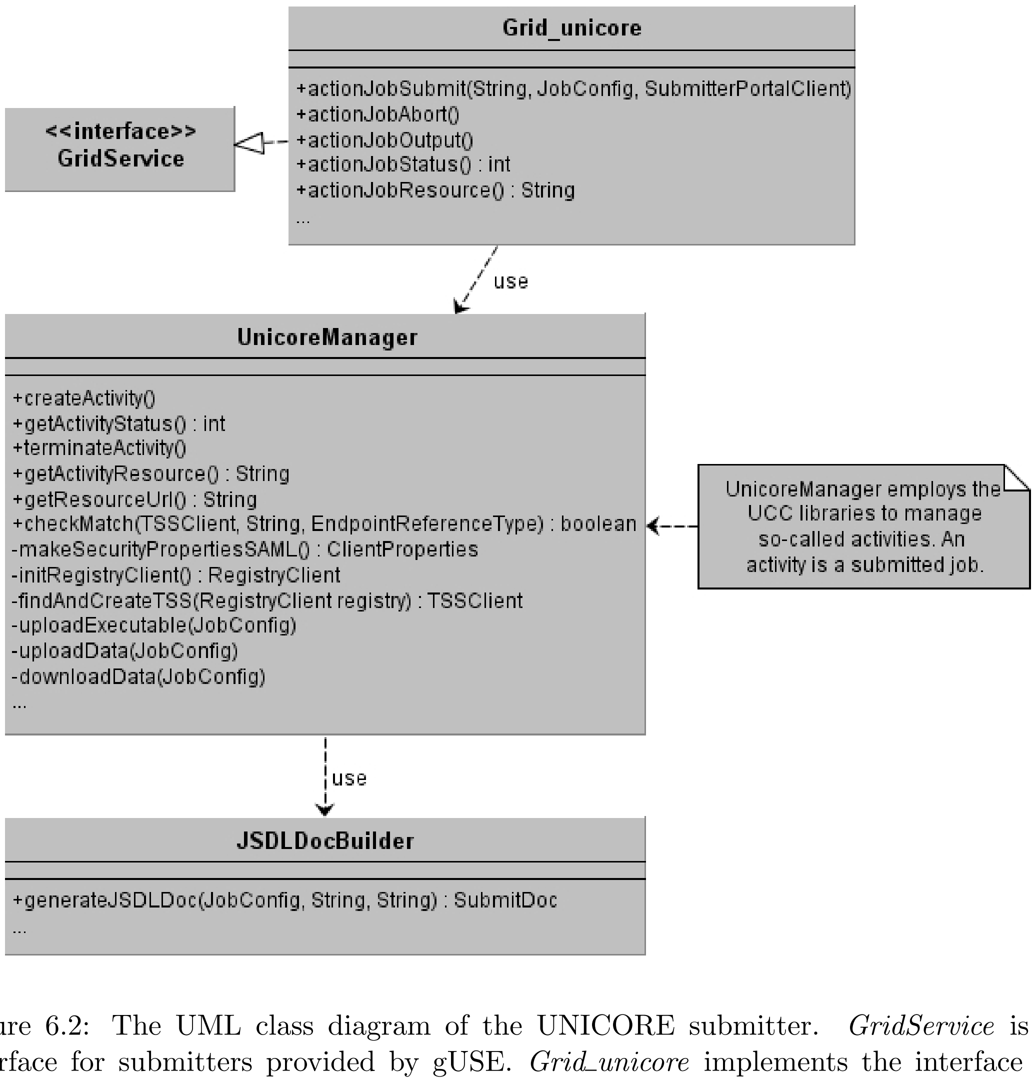
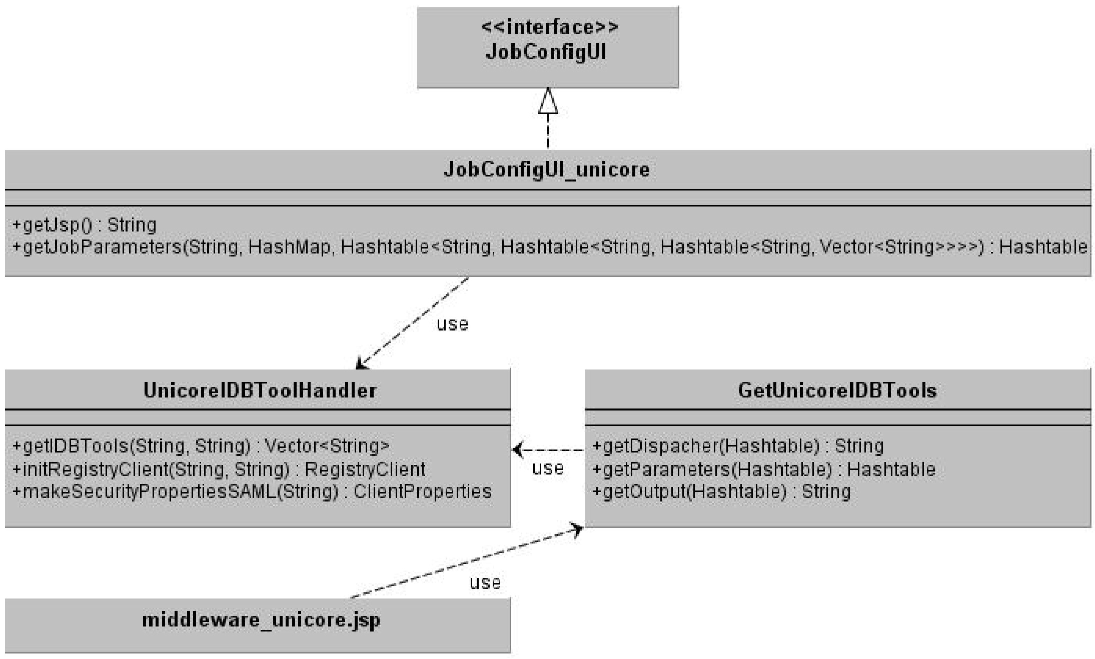

Grid_unicore
+actionJobSubmit(String, JobConfig, SubmitterPortalClient)
+actionJobAbort()
+actionJobOutput()
+actionJobStatus() : int
+actionJobResource() : String
...
<<interface>>
GridService
use
UnicoreManager
+createActivity()
+getActivityStatus() : int
+terminateActivity()
+getActivityResource() : String
+getResourceUrl() : String
+checkMatch(TSSClient, String, EndpointReferenceType) : boolean
-makeSecurityPropertiesSAML() : ClientProperties
-initRegistryClient() : RegistryClient
-findAndCreateTSS(RegistryClient registry) : TSSClient
-uploadExecutable(JobConfig)
-uploadData(JobConfig)
-downloadData(JobConfig)
...
UnicoreManager employs the UCC libraries to manage so-called activities. An activity is a submitted job.
use
JSDLDocBuilder
+generateJSDLDoc(JobConfig, String, String) : SubmitDoc
...

Figure 6.2: The UML class diagram of the UNICORE submitter. *GridService* is the interface for submitters provided by gUSE. *Grid_unicore* implements the interface and uses *UnicoreManager* to start, monitor, and abort a UNICORE job on a suitable target system. *JSDLDocBuilder* creates a JSDL file with the provided parameters for the job including the data staging.

After the potential data-staging, the UNICORE job is invoked as web service. The workflow engine of WS-PGRADE starts to periodically poll the submitter about the status of the web service. When the web service finishes, the local output files and standard output files like STDOUT and STDERR are downloaded from the USpace and the resources of the job are deleted on the UNICORE instance. The implementation details are described via the UML diagram in Figure 6.2.

6.2.2 UNICORE Plugin in WS-PGRADE

WS-PGRADE allows the intuitive configuration of the characteristics and the parameters of a job. The display of the choice of parameters is handled in a flexible and scalable way with the aid of AJAX (Asynchronous JavaScript and XML) [211] and via a naming convention recognised by gUSE. Parameters can be added to the portlet with the use of JSPs (JavaServer Pages). If their names follow the schema `job_<parametername>`, they are automatically added as parameters to the job properties and their values can be requested by a submitter.



<<interface>>
JobConfigUI
JobConfigUI_unicore
+getJsp() : String
+getJobParameters(String, HashMap, Hashtable<String, Hashtable<String, Hashtable<String, Vector<String>>>>) : Hashtable
use
UnicoreIDBToolHandler
+getIDBTools(String, String) : Vector<String>
+initRegistryClient(String, String) : RegistryClient
+makeSecurityPropertiesSAML(String) : ClientProperties
use
GetUnicoreIDBTools
+getDispacher(Hashtable) : String
+getParameters(Hashtable) : Hashtable
+getOutput(Hashtable) : String
use
middleware_unicore.jsp

Figure 6.3: The UML class diagram of the UNICORE plugin for WS-PGRADE. *JobConfigUI_unicore* implements the general interface for displaying the parameters of a job configuration for UNICORE. It uses *UnicoreIDBToolHandler* to get a list of all tools maintained in the IDBs of the connected target systems. *middleware_unicore.jsp* uses AJAX to update the list of tools suitable to a new selected UNICORE registry via *GetUnicoreIDBTools*.

As long as there are no properties configured for a job, a choice of generic parameters valid for each DCI is provided to the user, e.g., a drop-down menu with available types of DCIs, an option to upload an executable, and a field to define commandline parameters. As soon as a type of DCI is selected by the user, the choices are extended by a DCI-related view created via plugins. Each type of DCI obtains an own plugin. In case of UNICORE, users are enabled to choose the preferred UNICORE registry out of a generated list and to provide additional parameters like the designated number of nodes for MPI jobs. The selection of the registry induces a unique mechanism applied for UNICORE compared to all other supported DCIs in WS-PGRADE. The trust delegation tokens are set and the selected registry is initialised (see the previous section for the details on this process). All tools, which are maintained in the IDBs of connected target systems, are automatically added to a drop-down menu (see Fig. 6.1). Thus, WS-PGRADE provides up-to-date information to users about the accessible tools by selecting a UNICORE registry. The implementation of the plugin is illustrated in Figure 6.3.

6.3 Workflow Interoperability of gUSE with UNICORE

The UNICORE submitter additionally offers to invoke UNICORE workflows exported from the UNICORE Rich Client (URC). Instead of starting only a single UNICORE job, a UNICORE workflow is invoked. However, UNICORE workflows appear to gUSE like a single job. Figure 6.4 illustrates this concept of workflow interoperability.

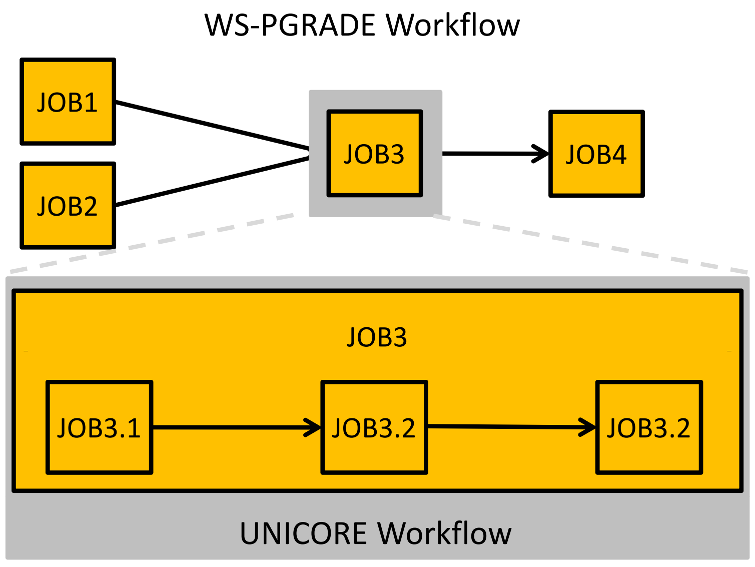


Figure 6.4: Illustration of an example of a WS-PGRADE workflow containing a UNICORE workflow in the third job

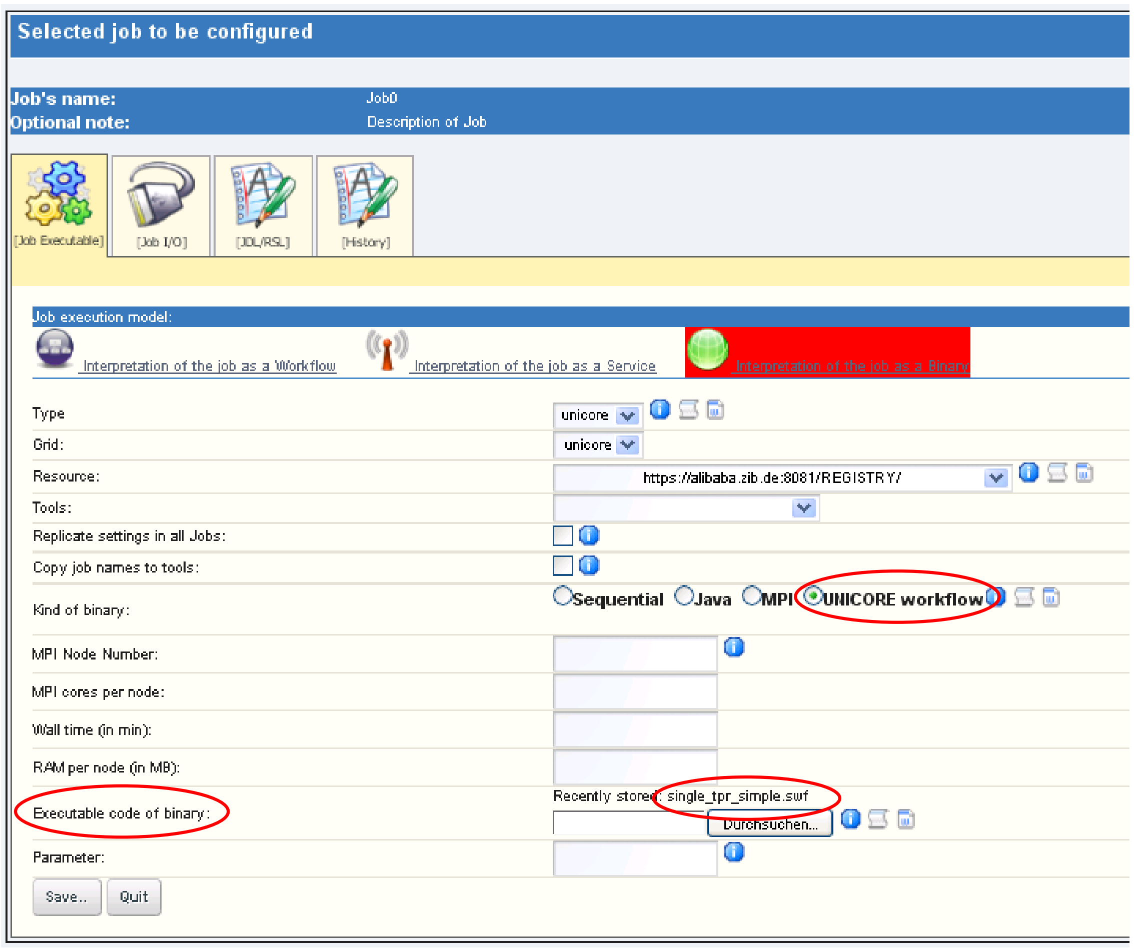
The workflow engine supplies the UNICORE submitter with the UNICORE workflow. The submitter distinguishes between single UNICORE jobs and UNICORE workflows. To submit a workflow, the following algorithm has to be performed.

Algorithm 6.2 Submission of a UNICORE workflow

```
set trust delegation tokens
initialise a registry client
if workflow data storage available at registry then
    create workflow data storage client at registry
else if shared data storage available then
    create workflow data storage client for shared data storage
else
    abort workflow submission
    exit
end if
find fitting workflow site
load workflow from file
if workflow syntax not correct then
    abort workflow submission
    exit
end if
create workflow client
create rules document
create workflow wrapper
start UNICORE workflow
```

Setting up the security properties and initialising a registry is identical to the submission of a UNICORE job. A fundamental prerequisite for invoking workflows is the accessibility of a data storage via the chosen registry. The submitter examines first whether a local data storage is available at the registry. If this is not the case, the submitter requests a shared data storage from the registry. UNICORE developers recommend to install a data storage at each registry that provides a workflow system. However, this is not mandatory but the UNICORE workflow system needs at least one type of accessible data storage. If neither a local nor a shared data storage is provided, the workflow system runs into an error. If a storage is available, the submitter requests a fitting workflow site from the registry. Afterwards, the workflow is loaded from the provided workflow file and the syntax is checked for correctness. If the syntax is correct, the submitter creates a workflow client that fits to the workflow site. Furthermore, the submitter generates a rules document that contains role-based access rights in XACML. The loaded workflow and the rules document are wrapped for the UNICORE workflow system and the resulting services are used to invoke a UNICORE workflow as web service.

The UNICORE workflow is handed over to the UNICORE workflow system that provides two major services: the workflow engine deals with workflow processing, while the service orchestrator combines a resource broker with a single job execution manager. Single jobs in the workflow are passed to a service orchestrator for brokering and execution. The configuration of a job as a UNICORE workflow has been added to WS-PGRADE as one additional job property in the UNICORE plugin. The property can be checked by the submitter. For the purpose of uploading the workflow file and, thus, providing it in the job properties, the existing upload option for executables is used (see Figure 6.5). This approach guarantees that the gUSE workflow engine treats the UNICORE workflow



Selected job to be configured

Job's name:	Job0
Optional note:	Description of Job

[Job Executable]
[Job I/O]
[JDL/RSL]
[History]

Job execution model:

Interpretation of the job as a Workflow
Interpretation of the job as a Service
Interpretation of the job as a Binary

Type	unicore
Grid:	unicore
Resource:	https://alibaba.zib.de:8081/REGISTRY/
Tools:	
Replicate settings in all Jobs:	☐
Copy job names to tools:	☐
Kind of binary:	Sequential Java MPI [✓ UNICORE workflow]
MPI Node Number:	
MPI cores per node:	
Wall time (in min):	
RAM per node (in MB):	
Executable code of binary:	Recently stored: single_tpr_simple.swf
Parameter:	

Figure 6.5: Configuration of a job as UNICORE workflow in WS-PGRADE

simply like a UNICORE job.

The adaption of the submitter for workflows affected only the class `UnicoreManager` of the submitter. Figure 6.6 illustrates the changed and additional methods. The other already introduced classes and methods remained the same.

6.4 Workflow Management via the Application Specific Module

gUSE provides a sophisticated web-based way to create, configure, and execute grid applications on various types of DCIs. WS-PGRADE employs gUSE to offer a generic user interface for workflows. To tailor the user interface to specific applications, additional portlets are needed in WS-PGRADE. To hide the complex architecture from developers, a new component called ASM has been developed that can be used as an API. Thus, developers are enabled to focus on creating domain-specific portlets, e.g., the possibility to create user-friendly input masks or to allow some post-processing steps after the execution such as generating pictures or charts from the results. Authentication on grid and cloud infrastructures, submission, and monitoring of workflows are handled jointly by various web-service components of gUSE.

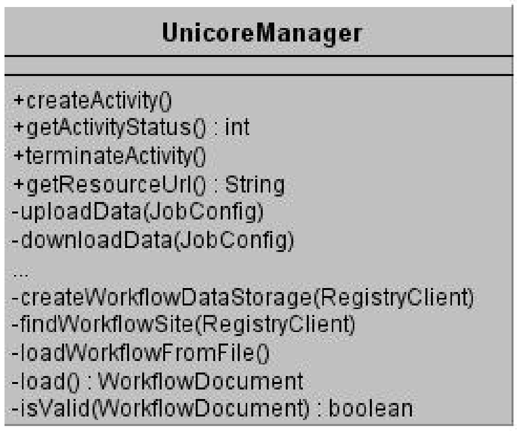

UnicoreManager
+createActivity()
+getActivityStatus() : int
+terminateActivity()
+getResourceUrl() : String
-uploadData(JobConfig)
-downloadData(JobConfig)
...
-createWorkflowDataStorage(RegistryClient)
-findWorkflowSite(RegistryClient)
-loadWorkflowFromFile()
-load() : WorkflowDocument
-isValid(WorkflowDocument) : boolean

Figure 6.6: The UML class diagram of the workflow extension in the UNICORE submitter

Every application included in the local repository of gUSE can be re-used via a portlet using the ASM libraries. Applications in gUSE consist of workflows and corresponding parameters as well as input and output files. ASM provides various interfaces for the management of applications and contains methods that enable the management of the whole execution lifecycle. These methods include the following:

- list the users who have exported applications to the local repository
- list the exported applications for a specified user
- import an application to the user space
- manage input and output files for upload and download
- manage remote files
- set and get the command line arguments of a specified job
- submit the application
- get the status of the application
- abort an application
- delete the application

In this way, the original but generic solutions for creating and managing applications are hidden from the users. Users can use well-tested applications via interfaces that were tailored to their needs. To assure that the users can set and execute an exact copy of tested and shared applications, ASM does not provide any method to make structural modifications on the workflows, for instance adding or deleting jobs.

ASM has been developed in Java and supports AJAX actions. Details of the implementation are illustrated in Figure 6.7. The Java methods can be called from portlets, which in turn can use any technology and visualisation method fitting to the applications' needs, independently from the underlying solution. The security mechanisms rely on the implemented submitters in gUSE. Hence, portlets developed employing ASM can use a submitter via the configured applications.

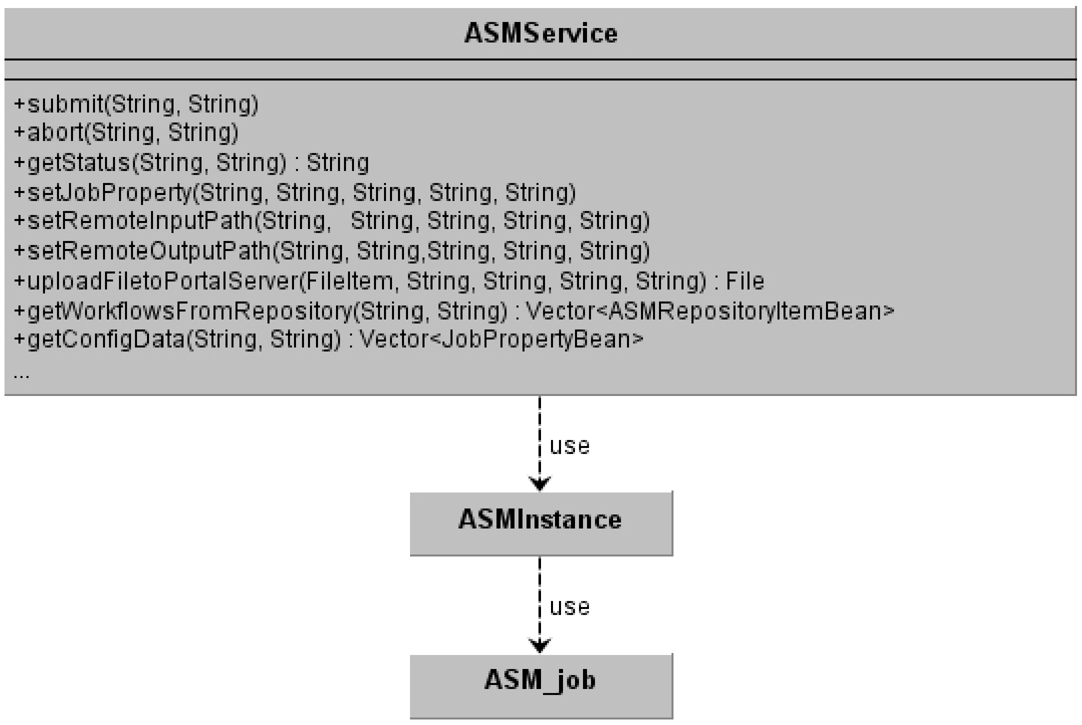

ASMService
+submit(String, String)
+abort(String, String)
+getStatus(String, String) : String
+setJobProperty(String, String, String, String, String)
+setRemoteInputPath(String, String, String, String, String)
+setRemoteOutputPath(String, String,String, String, String)
+uploadFiletoPortalServer(FileItem, String, String, String, String) : File
+getWorkflowsFromRepository(String, String) : Vector<ASMRepositoryItemBean>
+getConfigData(String, String) : Vector<JobPropertyBean>
...
use
ASMInstance
use
ASM_job

Figure 6.7: UML class diagram of the most important ASM classes. The class *ASMService* provides all methods for the whole execution lifecycle of an application. It uses *ASMInstance* for managing the properties of the current imported workflow. *ASM_job* includes the methods to set and get the properties of a job.

6.5 Results and Discussion

The workflow-enabled grid portal WS-PGRADE employs a variety of DCIs via gUSE. In this work, the job and workflow management of WS-PGRADE and gUSE was enhanced on different levels. First, the management of jobs on UNICORE DCIs was added. The developed submitter increases the number of supported DCIs and, thus, the flexibility for the users applying WS-PGRADE. In addition, an automatically generated list of available tools on the connected UNICORE DCIs is offered to the users. Hence, they are able to choose from this list during the configuration of a workflow. During the execution of a workflow, the single jobs are invoked via the submitter on a suitable resource. The users are relieved from uploading any scripts or executables to the science gateway. In particular, in case of complex software packages, users do not need to know on which resources the software packages are actually provided. This is a unique feature compared to the other submitters of gUSE and to other workflow-enabled grid portals. The current solution creates the list of available tools dynamically for each user. This is performed by reading the entries of the IDBs connected via the chosen UNICORE registry during the configuration of a workflow.

The submitter for UNICORE additionally supports workflow interoperability. Users are able to apply and re-use UNICORE workflows in a WS-PGRADE workflow. The workflow engine of gUSE treats the UNICORE workflow like a UNICORE job and hands it over to the submitter. The submitter distinguishes between UNICORE jobs and UNICORE workflows and invokes them accordingly. This type of interoperability requires that the user submitting a workflow has access to a DCI containing the targeted workflow engine. This requirement is fulfilled in the MoSGrid portal since it offers access to UNICORE DCIs. The same concept for workflow interoperability between WS-PGRADE and UNICORE is applied by the SHIWA platform based on WS-PGRADE. The UNICORE submitter has set the stage for the integration of UNICORE into SHIWA. Tavaxy offers the same approach for workflow interoperability with Taverna via web service calls. Since Tavaxy lacks the support of security features for X.509 certificates and SAML assertions, UNICORE workflows cannot be integrated in Tavaxy via web service calls.

ASM is an API developed in this work, which provides methods for the whole life cycle of a workflow submission. It is used for providing gUSE workflow features in portlets. It supports developers of domain-specific portlets to focus on the layout and domain-related features. The API can be applied via different Java Development Frameworks like Vaadin and the Google Web Toolkit as well as frameworks supporting the development of portlets like Rapid (see Section 8). The workflows are managed via the ASM API, which internally use the gUSE services and, thus, the submitters. If the implementation of the gUSE services or the underlying security infrastructure changes, the use of ASM assures that the portlets are not affected. Hence, the use of ASM simplifies the development of portlets integrated into WS-PGRADE.

7

A Novel Method for Workflow Migration

7.1 Introduction

In general, workflow-enabled science gateways support users creating, changing, and invoking workflows in an intuitive way. Ideally, users are offered a graphical user interface without becoming acquainted with the syntax of a workflow language or with the technical aspects of the managing workflow engine. They can focus on the applications of a workflow, which is independent of the underlying workflow technology. The migration of workflows allows the sharing of workflows within communities and between related communities, which do research in the same area but not necessarily use the same science gateways. Migration of workflows is concerned with the conversion of workflows of one workflow language to another following a fine-grained workflow interoperability approach (see Section 6). It does not involve the different workflow systems themselves but offers the possibility to map existing workflows to a different technology and to import and export workflows. Migration of workflows between web-based science gateways can be applied on different levels.

1. Migration between two instances of the same technology and version
   This case implies the export and import of existing workflows from one gateway instance to another gateway instance of the same technology and version. Some science gateways (e.g., Galaxy) support the migration by exporting the graph characteristics and job dependencies of a workflow. They relieve the users of creating the workflow from scratch and they retain the logical flow and the parameters. Several gateways additionally offer options to export an instance of a workflow including the executables, log files, input files, and output files (e.g., WS-PGRADE).
2. Migration between two instances of the same technology but in different versions
   Additionally to the demands of the first case, the migration of workflows between different versions obliges to examine whether the definition of workflows has been involved in changes. If not, this case is identically to the first one. Otherwise the different definitions have to be compared regarding the possibility of mapping a

workflow to the targeted version. In general, users aim to apply a technology with more supported workflow constructs than to limit the workflows via a migration. Thus, it is very probable that the targeted version for the migration is capable of at least the same workflow constructs. Otherwise the demands of the following case applies.

3. Migration between different science gateways
To migrate workflows between different science gateways, it is necessary to analyse the two different workflow languages and technologies of storing workflows and dependent data. An essential part is the investigation of the semantics of the diverse workflow languages (Kradolfer et al. [212]). If the workflow languages vary in their supported constructs, it is essential whether the targeted workflow language or the origin workflow language support more constructs. If the targeted workflow language is capable of more constructs than the origin one, the migration can be performed smoothly. Otherwise, each workflow and the used workflow constructs has to be checked whether it is suitable for an export in the other workflow language. In addition to the conversion of a workflow, a complete migration requires the export of the workflow from the originating science gateway and an import of the workflow at the target science gateway. There are three possibilities to place the conversion task during the migration process. It can be executed during the export, in a stand-alone tool, or in combinations with the import.

In this work, we focus on the third type of migration of workflows between Galaxy and WS-PGRADE. Both science gateways are widely used by various communities. Galaxy is a workflow-enabled portal for cloud infrastructures (e.g., Amazon EC2) and for HPC facilities managed by batch systems like PBS. It is designed as a kind of toolbox and administrators have to setup a Galaxy instance for each DCI they want to support. Additionally, it lacks the possibility to connect to grid infrastructures. To allow re-using Galaxy workflows more flexibly in different DCIs, we implemented the export of Galaxy workflows to WS-PGRADE workflows.

7.2 Workflows

WS-PGRADE as well as Galaxy apply a data-driven workflow engine. To compare the semantics of the two workflow languages and, thus, the supported constructs, we refer to the workflow patterns defined by van der Aalst et al. [144]. The following workflow constructs form the subset of the defined workflow patterns that is suitable for at least one of the two workflow languages.

- Sequence:
Two connected jobs of a workflow are a sequence if one job is not started until the preceding job is finished.
- Parallel split:
A workflow is split parallelly if a job forms a sequence with multiple subsequent jobs, which are not connected to each other and can be executed concurrently.
- Synchronisation:
A workflow is synchronised if a job forms a sequence with multiple preceding jobs.

- Exclusive choice:
  It is a conditional parallel split in which only one branch of the workflow is executed.
- Multiple instances with a priori run-time knowledge:
  A job or a branch of a workflow is instantiated multiple times and executed concurrently. The number of instances is known before the execution of the workflow and the workflow has to be synchronised when the multiple instances are finished.
- Recursion:
  It allows a job to invoke itself during its execution.

The analysis of the workflow languages has led to the following result of supported workflow patterns.

Table 7.1 The supported workflow patterns in WS-PGRADE and Galaxy

Workflow pattern	WS-PGRADE	Galaxy
Sequence	true	true
Parallel split	true	true
Synchronisation	true	true
Exclusive choice	true	false
Multiple instances with a priori run-time knowledge	true	false
Recursion	true	false

Thus, all workflow patterns of Galaxy can be converted to workflow patterns in WS-PGRADE. The syntactic format of the workflows are quite different and characterised in the next sections.

7.2.1 WS-PGRADE Workflows

The data-driven workflows in WS-PGRADE are defined by two parts; i) the graphical representation of the steps as a DAG with so-called ports for input files and output files and ii) the concrete characterisation of the steps with well-defined input files and output files. To create a workflow, first the graph has to be defined in the graph editor. Figure 7.1 illustrates the graphical representation of a workflow in WS-PGRADE. Each yellow box represents a job, a little green box represents an input file and a little grey box represents an output file. Connections (the red lines) can be only established between an input file and an output file. Via this logic, the subsequent order of the workflow is assessed. The graph editor allows additionally inserting names for the jobs, descriptions and names of the input files and output files.

By storing the graph, the graph editor automatically creates an XML file that includes the graph properties. The tag `<workflow>` is the root element for the definition of the whole workflow, whereas `<graf>` configures the name of the graph (see Fig. 7.2).

The subjacent tags `<job>` define the properties of the jobs: their names, the coordinates for the yellow boxes and the properties of the input files and output files. Via `<input>` the input name, the coordinates of the ports and the logical flow of the workflow is represented. It includes the name of the parent job (`prejob`) and which output port of `prejob` is taken as input for the current job. All ports are automatically numbered with an integer starting with 0. `preoutput` contains the number of the output port of the parent job connected to the input port of the current job.

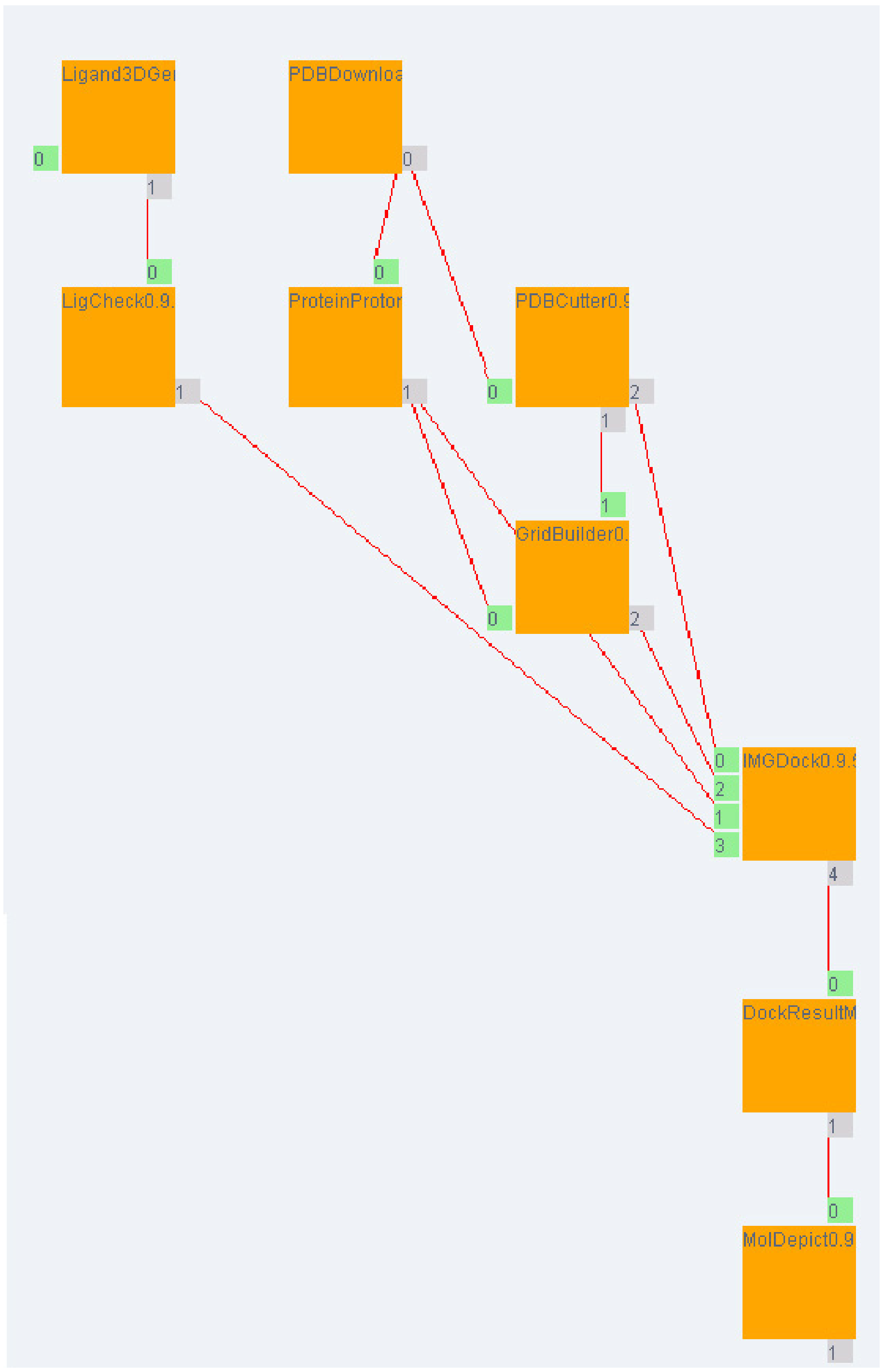
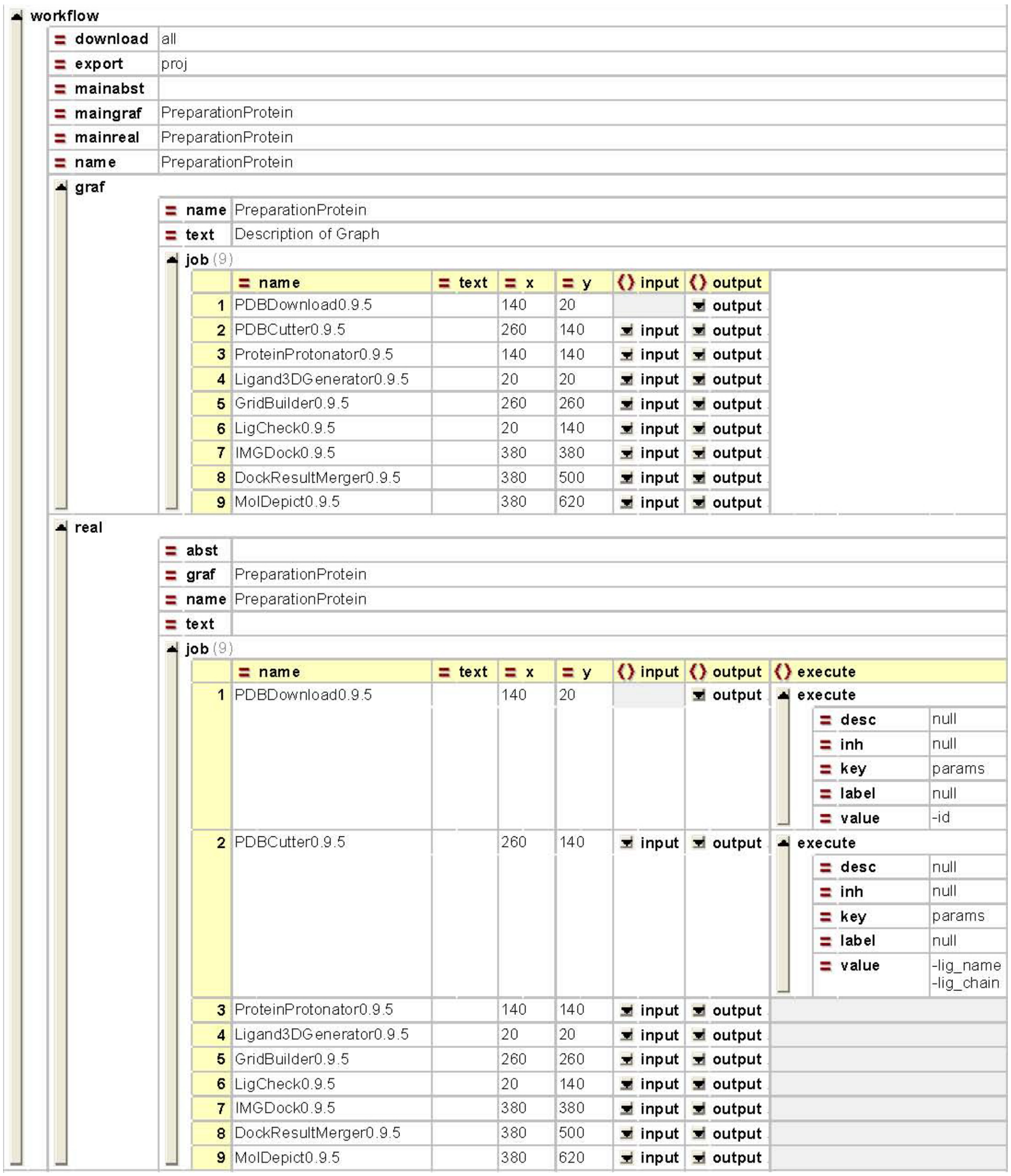


Figure 7.1: The graphical illustration of a workflow for docking of a protein in WS-PGRADE

Further parameters for the workflow and each job can be assigned after the graph has been expanded into a concrete workflow. By selecting this option, the XML file is automatically extended with the tag `<real>` and all existing job properties are copied under this tag (see Fig. 7.2). Under the `<real>` section, `<job>` may include the tags `<execute>` and `<history>` (history of changes in the workflow). Via `<execute>` the job properties like parameters are stored (see Fig. 7.3).



workflow

download	all
export	proj
mainabst	
maingraf	PreparationProtein
mainreal	PreparationProtein
name	PreparationProtein

graf

name	PreparationProtein
text	Description of Graph

job (9)

	name	text	x	y	input	output
1	PDBDownload0.9.5		140	20		output
2	PDBCutter0.9.5		260	140	input	output
3	ProteinProtonator0.9.5		140	140	input	output
4	Ligand3DGenerator0.9.5		20	20	input	output
5	GridBuilder0.9.5		260	260	input	output
6	LigCheck0.9.5		20	140	input	output
7	IMGDock0.9.5		380	380	input	output
8	DockResultMerger0.9.5		380	500	input	output
9	MolDepict0.9.5		380	620	input	output

real

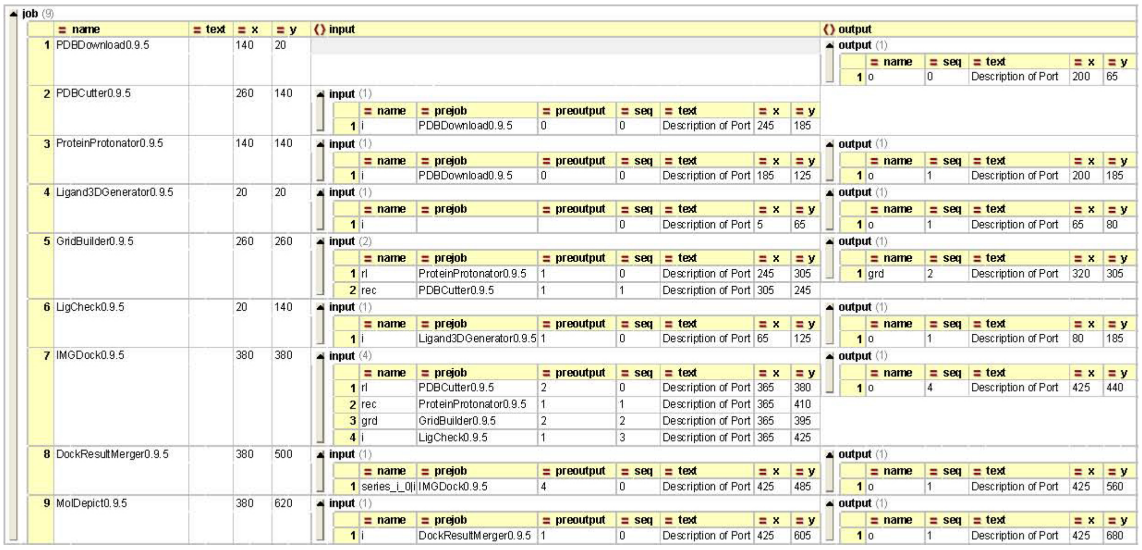
abst	
graf	PreparationProtein
name	PreparationProtein
text	

job (9)

	name	text	x	y	input	output	execute
1	PDBDownload0.9.5		140	20		output	execute: desc null; inh null; key params; label null; value -id
2	PDBCutter0.9.5		260	140	input	output	execute: desc null; inh null; key params; label null; value -lig_name -lig_chain
3	ProteinProtonator0.9.5		140	140	input	output	
4	Ligand3DGenerator0.9.5		20	20	input	output	
5	GridBuilder0.9.5		260	260	input	output	
6	LigCheck0.9.5		20	140	input	output	
7	IMGDock0.9.5		380	380	input	output	
8	DockResultMerger0.9.5		380	500	input	output	
9	MolDepict0.9.5		380	620	input	output	

Figure 7.2: XML illustration of the data structure of WS-PGRADE workflows on the example of a workflow for docking of a protein

While the XML file contains all logical information about the workflow, WS-PGRADE



job (9)

	name	text	x	y	input								output					
						name	prejob	preoutput	seq	text	x	y		name	seq	text	x	y
1	PDBDownload0.9.5		140	20									1	o	0	Description of Port	200	65
2	PDBCutter0.9.5		260	140	1	i	PDBDownload0.9.5	0	0	Description of Port	245	185						
3	ProteinProtonator0.9.5		140	140	1	i	PDBDownload0.9.5	0	0	Description of Port	185	125	1	o	1	Description of Port	200	185
4	Ligand3DGenerator0.9.5		20	20	1	i			0	Description of Port	5	65	1	o	1	Description of Port	65	80
5	GridBuilder0.9.5		260	260	1	rl	ProteinProtonator0.9.5	1	0	Description of Port	245	305	1	grd	2	Description of Port	320	305
					2	rec	PDBCutter0.9.5	1	1	Description of Port	305	245						
6	LigCheck0.9.5		20	140	1	i	Ligand3DGenerator0.9.5	1	0	Description of Port	65	125	1	o	1	Description of Port	80	185
7	IMGDock0.9.5		380	380	1	rl	PDBCutter0.9.5	2	0	Description of Port	365	380	1	o	4	Description of Port	425	440
					2	rec	ProteinProtonator0.9.5	1	1	Description of Port	365	410						
					3	grd	GridBuilder0.9.5	2	2	Description of Port	365	395						
					4	i	LigCheck0.9.5	1	3	Description of Port	365	425						
8	DockResultMerger0.9.5		380	500	1	series_i_0|i	IMGDock0.9.5	4	0	Description of Port	425	485	1	o	1	Description of Port	425	560
9	MolDepict0.9.5		380	620	1	i	DockResultMerger0.9.5	1	0	Description of Port	425	605	1	o	1	Description of Port	425	680

Figure 7.3: XML illustration of the data structure of jobs in a WS-PGRADE workflows on the example of a workflow for docking of a protein

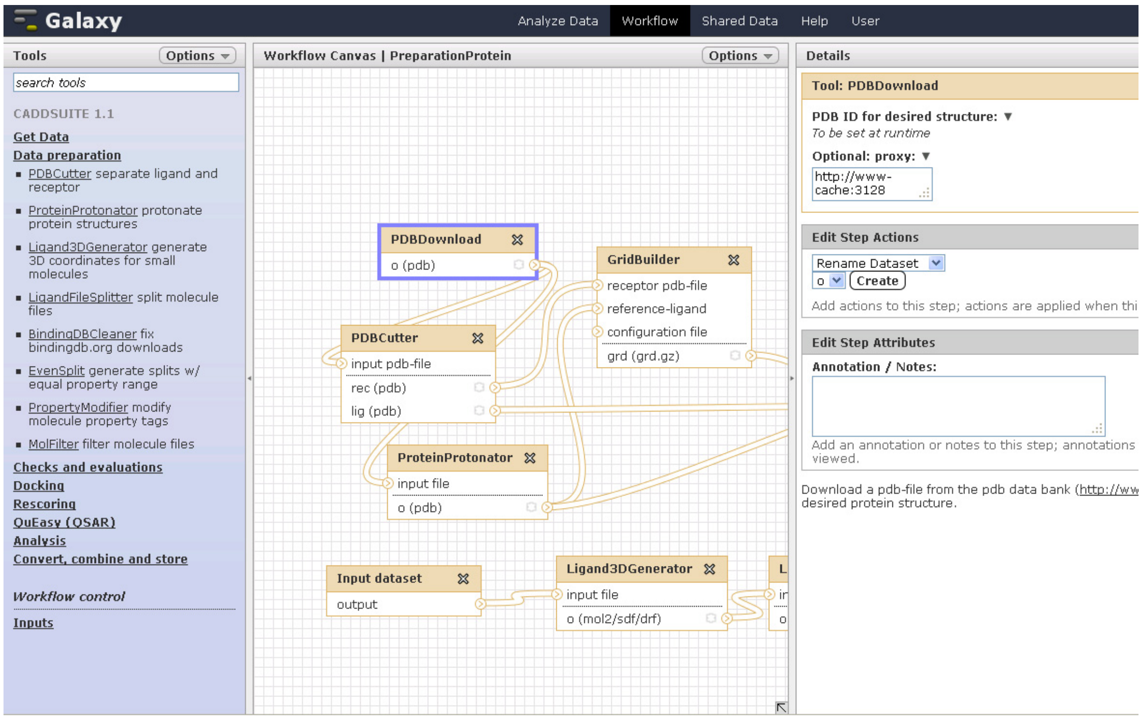
also stores all data of submitted workflows including local executables, local input files, local output files, and log files in a directory on the portal server. A workflow can always be re-used and input files can easily be changed for further invocation. Users are able to perform the creation, possible changes and execution of workflows without managing and knowing the underlying files. Each step is supported by graphical user interfaces and the XML file is not shown in the science gateway.

WS-PGRADE offers different download and upload features for workflows. Users are enabled to download or upload only a graph, a workflow or an instance of a workflow with all corresponding files like executables, input files, and output files. A graph is directly re-usable in another WS-PGRADE instance without any changes. A workflow or an instance of a workflow may need corrections dependent on the underlying infrastructure (e.g., available UNICORE registries).

7.2.2 Galaxy Workflows

The representation of Galaxy workflows is quite different from the representation of WS-PGRADE workflows. The workflows of both systems are data-driven and they can be represented by DAGs. However, the goal of Galaxy is to offer a toolbox for workflows in a configured infrastructure (e.g., PBS, Amazon EC2). Administrators of a Galaxy instance configure tools, including the necessary paths and environments for the execution. They can pre-define parameters and can additionally allow to change parameters for a tool for each execution. These tools are stored in a database and are offered to the users. Thus, the users can simply select them for a workflow in a graphical user interface (see Figure 7.4).

In the graphical user interface of Galaxy the steps in a workflow are represented by boxes, which include the name of the tool in the header and the input files and output files as lists below. The logical flow of a workflow is symbolised by connecting an input file to an output file of the parent job. Boxes in Galaxy may also represent input datasets.



CADDSUITE 1.1

Get Data

Data preparation

- PDBCutter separate ligand and receptor
- ProteinProtonator protonate protein structures
- Ligand3DGenerator generate 3D coordinates for small molecules
- LigandFileSplitter split molecule files
- BindingDBCleaner fix bindingdb.org downloads
- EvenSplit generate splits w/ equal property range
- PropertyModifier modify molecule property tags
- MolFilter filter molecule files

Checks and evaluations
Docking
Rescoring
QuEasy (QSAR)
Analysis
Convert, combine and store

Workflow control

Inputs

Tool: PDBDownload

PDB ID for desired structure: ▼
To be set at runtime

Optional: proxy: ▼
http://www-cache:3128

Add actions to this step; actions are applied when thi

Annotation / Notes:

Add an annotation or notes to this step; annotations viewed.

Download a pdb-file from the pdb data bank (http://ww desired protein structure.

Figure 7.4: The graphical user interface for creating workflows in Galaxy

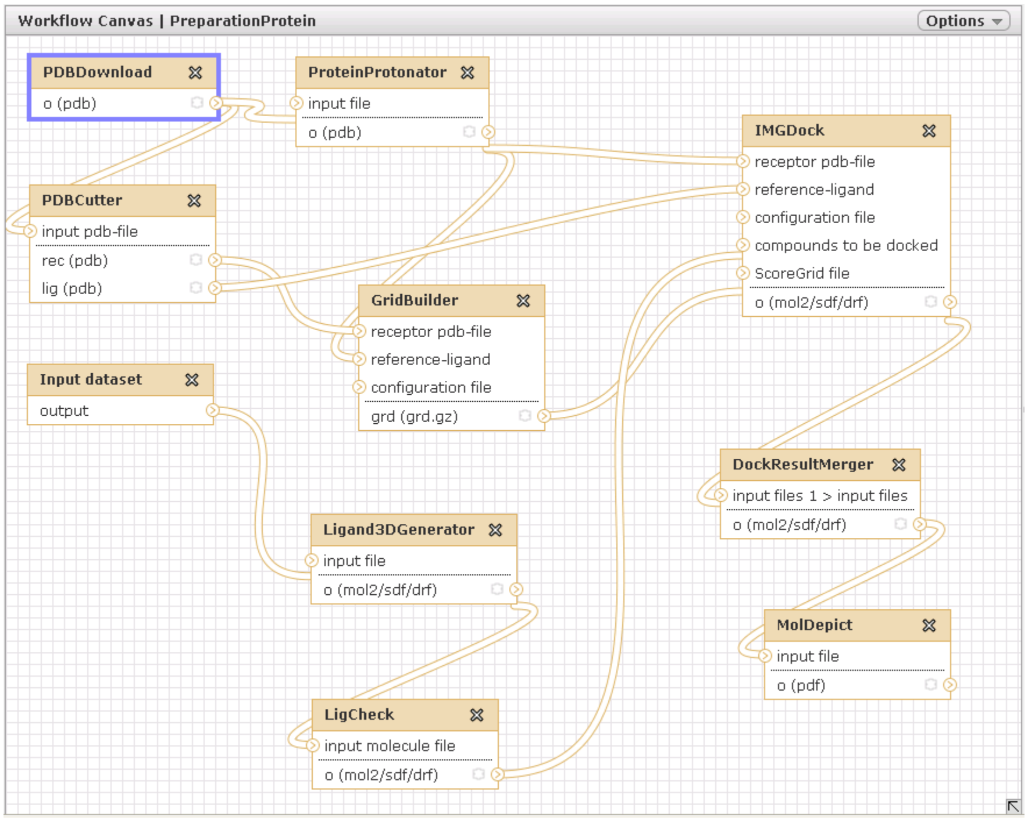


Figure 7.5: The graphical illustration of a workflow for docking of a protein in Galaxy

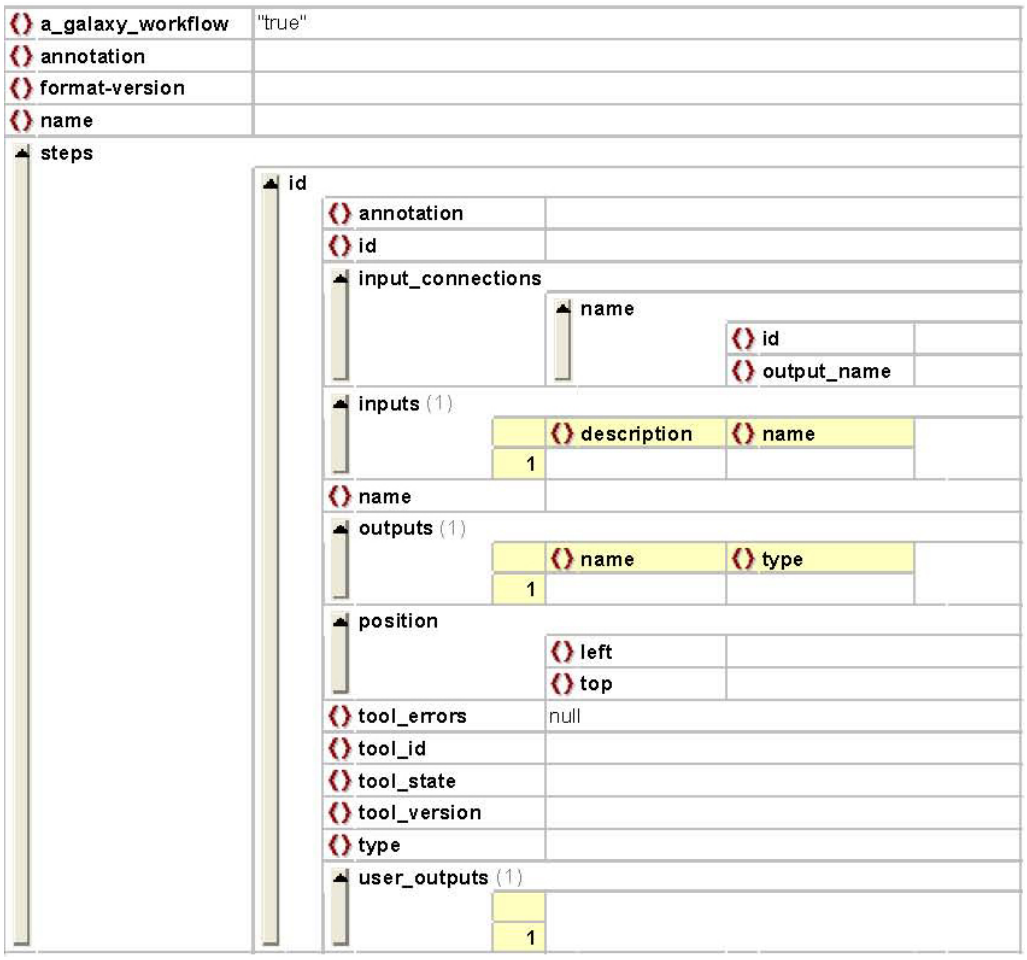


Figure 7.6: JSON illustration of the data structure of Galaxy workflows

They are used for input files which are independent of other tools and are provided by the user. Figure 7.5 represents the same workflow in Galaxy, which was shown in WS-PGRADE in Figure 7.1.

A workflow and its graphical representation is stored in Galaxy in a Python [213] dictionary. This datastructure includes fields for the name of the workflow, annotation, and definitions for each step like the tool id, the job parameters, and information about the parent jobs. It preserves the logical flow of the workflow and the coordinates of the boxes. Since the tools are directly integrated in a Galaxy instance with a specific underlying infrastructure, there is no information provided about this infrastructure (e.g., the underlying batch system). Figure 7.6 illustrates the JSON data structure.

A workflow structure can be uploaded or downloaded via Galaxy in a JSON format. A workflow is directly re-usable in case the same tools with the same tool ids are configured in the targeted Galaxy instance. If this is not the case, Galaxy reports an error when a workflow is invoked.

7.3 Migration of Galaxy Workflows to WS-PGRADE Workflows

WS-PGRADE as well as Galaxy offer features for importing and exporting workflows in their specific format. For the migration, we analysed these specific formats (see above). A WS-PGRADE-based workflow can be uploaded to a WS-PGRADE instance as ZIP file. The ZIP file contains an XML file *workflow.xml* which defines the structure of the workflow and its name. Furthermore, a directory is included which is named after the workflow's name. A correct workflow for WS-PGRADE can be created only from the XML file and the directory. Galaxy supports the download of workflows in a JSON format representing a Python dictionary and the export of workflows to myExperiment sandboxes in SCUFL (see Karasavvas et al. [214] for the developed migration). Both formats differ from the workflow format for WS-PGRADE. Since a workflow conversion from SCUFL to WS-PGRADE format would have been as complex as the conversion from Galaxy format to WS-PGRADE format and would involve an additional technology and dependencies on it, the preferrable choice is to directly convert Galaxy workflows to WS-PGRADE workflows for the migration.

There are three possible locations to perform the conversion; i) a stand-alone tool for the conversion of the Galaxy-based JSON file to a WS-PGRADE-based ZIP file, ii) an option to download a WS-PGRADE-conformable ZIP file in Galaxy, and iii) an option to upload a Galaxy-conformable JSON file in WS-PGRADE. The main drawback of a stand-alone tool is on the users' side. They would have to install an additional tool for the conversion. Therefore, the decision was made that the conversion takes place in one of the involved science gateways. We have preferred to extend Galaxy for maintainability reasons. The workflow format of WS-PGRADE is well-established in the P-GRADE portal family. Furthermore, the workflow interoperability with various workflow-enabled science gateways has been developed via the project SHIWA. The change of this format is less probable than a change of the format of a JSON file reflecting an inner datastructure in Galaxy. As long as the datastructure in Galaxy will only be extended and essential fields will be not changed or removed, the integrated conversion is still executable.

The option to download a WS-PGRADE workflow out of Galaxy has required changes at the user interface and the creation of an additional feature for workflow conversion. Galaxy has been developed in Python and the template library Mako [215]. The additional option in the user interface has been integrated into the existing Mako file for downloading and exporting workflows. The underlying class for downloads of WS-PGRADE workflows has been added to the web controller for workflows in Galaxy (see Fig. 7.7). The controller has three main features: the coordinates for the WS-PGRADE graph are calculated based on the structure of a workflow, necessary fields of the Python dictionary are used for creating the XML file, and the resulting XML file as well as an empty directory with the workflow's name is zipped for download.

After the workflow has been read from the Galaxy database and assigned to a Python dictionary, coordinates are calculated for the WS-PGRADE graph. Afterwards fields in the Python dictionary are used to map to tags in an XML file describing a WS-PGRADE workflow (see Table 7.2). The created XML file is zipped together with an empty directory, which is named like the workflow. This ZIP file can be downloaded to the users' computers and then uploaded to a WS-PGRADE instance.

Via the XML file a correct graph of the workflow is created. A real workflow is also prepared but it is not fully functionable in this state in WS-PGRADE. For a fully functional

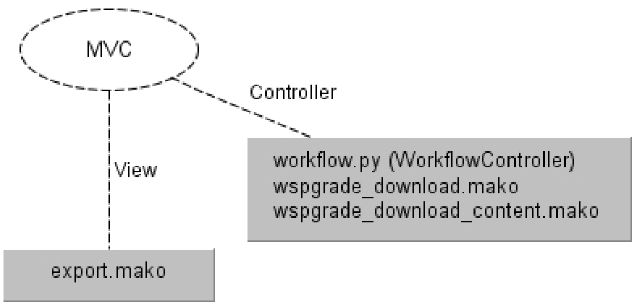

MVC
Controller
View
workflow.py (WorkflowController)
wspgrade_download.mako
wspgrade_download_content.mako
export.mako

Figure 7.7: Source files involved in the download of workflows in WS-PGRADE format

Table 7.2 Mapping of the Python dictionary to XML tags

Python	XML tag	Use

workflow[name]	<workflow>
	
	<graf>
	<real>
		maingraf=”[name]” mainreal=”[name]”
		name=”[name]”
		name=”[name]”
		graf=”[name]” name=”[name]”

steps[name] and steps[tool version]	<job>	name=”[name]”+”[tool version]”
input connections[name]	<input>	name=”[name]”
input connections[id]	<input>	determine the name of the prejob via [id]
input connections[output name]	<input>	determine the number of the port via [output name]
outputs[name]	<output>	name=”[name]”
inputs[name]	<execute>	key=”params” value=”-[name] ”

workflow, additional information or files are needed (e.g., the targeted DCI, executables). WS-PGRADE supports the execution of tools via diverse methods, e.g., via uploaded executables or addresses of web services. Additionally, the UNICORE submitter allows selecting tools in the available HPC resources. The job name in the XML file is chosen like the name of tools in the IDB of UNICORE. Users can select which DCI they want to use for one job or for the whole workflow. If they have chosen to select one configuration for the whole workflow, all jobs are automatically configured. However, the steps lack still a job definition. The configuration for UNICORE enables another automatic step to finish each job definition. Via an option for UNICORE, users are enabled to copy the job name as tool name for each step in the workflow. Thus, a fully functional workflow is created and can be invoked.

7.4 Graph Algorithm

Galaxy workflows as well as WS-PGRADE workflows can be represented as DAGs. However, the graph representation and the coordinates of Galaxy workflows and WS-PGRADE workflows are quite different from each other. Therefore, the stored coordinates in Galaxy workflows cannot be used for the conversion. Furthermore, input files, that are provided by the user and do not result out of a parent job, are stored as single steps in Galaxy. WS-PGRADE displays them as ports of a job. Additionally to the coordinates of the boxes representing jobs, WS-PGRADE needs coordinates for the small boxes illustrating the input files and output files. To calculate the coordinates for a clearly arranged graph in WS-PGRADE with as few intersections of edges as possible, we developed an algorithm, which allows to calculate the coordinates of the boxes incrementally. The boxes for the jobs and the ports have fixed measures and the boxes of the ports are drawn directly next to the boxes of the corresponding jobs (see Fig. 7.1).

The steps of a Galaxy workflow are first sorted in topological order which sort all parent steps before their children. The steps are represented by vertices. We have used a Python code [216], which implements the algorithm of Tarjan [217] for topological order of DAGs. They are many solutions available for DAGs (e.g., in Sedgewick "Algorithms" [218]) to create a topological order. The applied algorithm of Tarjan was the first one that solved the problem for a Graph $G = (V, E)$ in linear running time $O(|V| + |E|)$. The algorithm for creating a topological order consists of three main tasks. First a datastructure for a directed graph is created (Task 1), then all root elements are filtered (Task 2) as starting point for Task 3. In Task 3 all root elements are added to a list as long as a root element exists. This list is built dynamically and if a root element is examined, it is emitted from the graph and the children are probed whether they are now roots. This processing ensures that each step is checked exactly one time and added to a list, which is sorted in a topological order afterwards.

The algorithm for creating a graph out of a workflow distinguishes between steps of the type *tool* and *dataInput* in the Python dictionary when generating the graph (see Algorithm 7.1).

Algorithm 7.1 Create a graph out of a workflow

```
#create stepOrder, inputFiles and partialOrder
for all steps in workflow do
    if type of step equal "tool" or NONE then
        append step to stepOrder
    end if
    if type of step equal "dataInput" then
        append step to inputFiles
    end if
end for
for all steps in workflow do
    for all inputConnections of step do
        if inputConnection not in inputFiles then
            append (inputConnection, step) to partialOrder
        end if
    end for
end for
#execute topological sort
stepOrder = topologicalSort(stepOrder, partialOrder)
```

```
#sort parents directly before children
for all steps in stepOrder do
    if distance between parent step and first child step $> 1$ then
        move parent before child
    end if
end for
```

The type *tool* characterises a job, the type *dataInput* characterises an input file. By processing the jobs in this order, it is assured that the coordinates of all parent jobs are already calculated before the children receive their coordinates. Furthermore, the algorithm assures that all parent jobs are sorted directly before their children if the structure of the graph allows it. For example, the order for the jobs
PDBDownload → *Ligand3DGenerator* → *PDBCutter* → *ProteinProtonator* → *LigCheck* (see Figure 7.5) can be determined as already topologically sorted. For the graph in WS-PGRADE, the algorithm sort the jobs in this order
Ligand3DGenerator → *LigCheck* → *PDBDownload* → *ProteinProtonator*→ *PDBCutter*. The advantage of this order is that the intersections between edges can be avoided as far as possible.

The coordinates of each box of a job is calculated dependent on the coordinates which are already assigned (see Algorithm 7.2). If a job has no parent, the box is located at the top, left from the so far maximal coordinates (see *PDBDownload* and *Ligand3DGenerator* in Figure 7.1). Boxes of children are placed under the boxes of their parents. If the coordinates directly under a parent are already occupied, the coordinates to the left are tested until free coordinates are found (see *ProteinProtonator* and *PDBCutter* in Figure 7.1). If there are more parents than two and at least two on the same vertical coordinates, the child box is placed lower and left of the maximal coordinates (see *IMGDock*in Figure 7.1). The boxes have the fixed measures 60 × 60 pixels and the algorithm calculates a fixed space between the boxes.

Algorithm 7.2 Calculate coordinates of the boxes representing steps

```
$xMax = xStart$
for all steps in stepOrder do
    if root then
        $xMax = xMax + space$
        $x = xMax$
        $y = space$
    else
        assign ccoordinates of first parent to $x$ and $y$
        if step has one parent then
            $y = y + space$
        end if
        for parents starting with second parent do
            if $x$ equal x coordinate of parent then
                $x = x + space$
            else if $x <$ x coordinate of parent then
                assign x coordinate of parent to $x$
            end if
            if $y$ equal y coordinate of parent then
                $y = y + space$
            else if $y <$ y coordinate of parent then
                assign y coordinate of parent to $y$
            end if
            while $x$, $y$ already coordinates of a step do
                $x = x + space$
            end while
            $xMax$ = maximum of ($x$, $xMax$)
        end for
    end if
end for
```

The coordinates of the ports are determined by placing the ports on the frame of the box which belongs to the corresponding job. In general, input files are represented at the left or at the top and output ports at the bottom or at the right. These positions avoid the intersection of the edge with the current parent box and child box. If there are so many ports configured that this rule cannot be followed (e.g., more input files than eight), the ports are placed on an available space at the box. The order of the configured connections between ports is not stringently sorted like the topological order which is created by the algorithm. To avoid intersections of edges connected to the same job box, the edges are tested for intersections. For the test of intersections, we consider the coordinates of input ports and output ports as points in a plane with x and y coordinates and utilise a geometric algorithm [218, 219] to test whether points are located in a counter-clockwise order. It relies on a formula for the calculation of the area of a triangle via the determinant of a 3×3 matrix. Let A, B, and C be points in 2D. Then the calculation for their order is performed in the following way.

$$area(A,B,C) := \begin{vmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{vmatrix} = (C_y - A_y)(B_x - A_x) - (B_y - A_y)(C_x - A_x)$$

$$area(A,B,C) \begin{cases} < 0 & \text{if } A,\ B,\ C \text{ in clockwise angle} \\ = 0 & \text{if } A,\ B,\ C \text{ collinear} \\ > 0 & \text{if } A,\ B,\ C \text{ in counter-clockwise angle} \end{cases}$$

Figure 7.8 illustrates the orientation of points. The points A, B, and C are drawn in counter-clockwise order; A, B, and D are drawn in clockwise order.

The test whether two edges intersect with each other, is based on the counter-clockwise order of the vertices defining them. Edges are considered as line segments in a plane. Let (A,B) and (C,D) be two line segments. These intersect if and only if the points A and B lie on different sides of (C,D) and the points C and D lie on different sides of (A,B). This means that A, C, and D are oppositely oriented than B, C, and D and that A, B, and C are oppositely oriented than A, B, and D. Thus, the below stated Algorithm 7.3 allows calculating the intersection between edges in constant time.

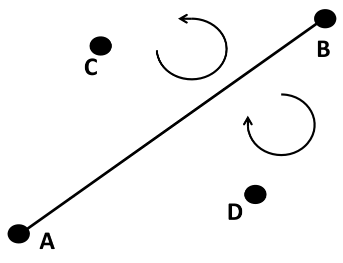


Figure 7.8: Orientation of the points A, B, C, and D in 2D

If edges of input ports of one step intersect, the coordinates of the involved ports are swapped (see *IMGDock* and the ports 1 and 2 in Figure 7.1), which is the final action in the graph algorithm to calculate the coordinates for a WS-PGRADE workflow graph.

Algorithm 7.3 Methods to check intersections of edges

```
#method returns true if coordinates of A, B, C are counter clockwise ordered
function CCW(A, B, C)
    return (C.y-A.y)(B.x-A.x) > (B.y-A.y)(C.x-A.x)
end function

#method returns true if edge (A,B) intersects with edge (C,D)
function INTERSECT(A, B, C, D)
    return ccw(A,C,D) ≠ ccw(B,C,D) and ccw(A,B,C) ≠ ccw(A,B,D)
end function
```

7.5 Results and Discussion

The concept of workflow migration between science gateways supports the re-usability and the exchange of workflows between the communities of science gateways. Thus, users are relieved from creating identical workflows in different technologies. In contrast to workflow interoperability introduced in Section 6, migration of workflows includes the translation of a workflow language into a targeted workflow language. Thus, the workflow can be used in the targeted science gateway without the need to have access to the source workflow engine. We have presented in this section the migration of Galaxy workflows to WS-PGRADE workflows.

The migration of the workflows has been embedded in Galaxy and consists of several steps. The essential steps for the workflow language translation are the graph algorithm for calculating the coordinates of the graphical representation of the WS-PGRADE workflows and the mapping of suitable workflow constructs of the Galaxy workflow language to workflow constructs of the WS-PGRADE workflow language. Due to the different graphical representations of workflows in Galaxy and WS-PGRADE, the coordinates of the Galaxy workflows cannot be re-used for the WS-PGRADE workflows. We developed a graph algorithm, which avoids edge crossing as far as possible and aligns the vertices and edges to an underlying grid. These two criteria for the aesthetics of a graph have been classified as the most important criteria for users [150]. Galaxy workflows are represented by DAGs and, thus, the vertices can be sorted in topological order. The topological order is extended by sorting parent vertices directly before their child vertices if the resulting order is still a valid topological order. The coordinates are incrementally calculated so that each child vertex is placed in a horizontal layer below the parent vertex. The algorithm creates all coordinates in linear running time $O(|V| + |E|)$.

A plethora of graph drawing algorithms exists, which follows a layered layout approach [147]. The algorithm of Sugiyama et al. [146] is widely used. It consists of four phases for the drawing of directed graphs: cycle removal, layer assignment applying chains of dummy vertices for long edges, crossing reduction of edges, and x-coordinate assignments to vertices while deleting dummy vertices. These phases include diverse NP-hard problems (e.g., 2-layer crossing minimisation [148]). Thus, a number of efficient heuristics [149] has been developed. The algorithms consider more criteria for the aesthetic of a graph than our algorithm, e.g., the minimisation of the number of layers. Our algorithm is efficient but it may lead for large graphs to a large number of layers and long edges. For example, the illustration of a workflow consisting only of sequences results in a graph with one vertex per layer.

Galaxy as well as WS-PGRADE are widely-used and are intuitive science gateways supporting a large number of communities with overlapping research fields. However, Galaxy lacks the support of grid-based DCIs and requires the installation of a Galaxy instance per underlying DCI. Thus, the migration of Galaxy workflows to WS-PGRADE workflows allows flexibly using existing Galaxy workflows for various DCIs. The same approach is followed by Karasavvas et al. [214] migrating Galaxy workflows to myExperiment sandboxes, which use SCUFL as workflow language. The SHIWA project [205] as well as the Tavaxy developers [208] suggest to perform workflow language translation via an intermediate presentation of the workflow in a multistage process. The resulting workflows of our solution can be integrated into the SHIWA platform since WS-PGRADE workflows are supported for the language translation to the intermediate workflow language IWIR [207].

8 Automatic Creation of Portlets for Workflows

8.1 Introduction

The creation of portlets is a time-consuming process that demands specialised skills. Besides portal frameworks, several tools and standards have been developed to aid developers in the portlet creation process. Vaadin and Google Web Toolkit, for example, allow creating JSR168/JSR286-compliant portlets. Developers of portlets supporting workflows to DCIs have to consider well-defined security features and embedded workflow management systems. So far, there is no established standard for workflow-enabled portlets. However, reliable and fully featured portals like WS-PGRADE and Genius are available and are based on the JSR168/JSR286 standard. Thus, deploying additional portlets to the portals can be easily performed — the sophisticated task is the development of portlets integrated with the workflow features for DCIs.

Frameworks like Rapid [26], EnginFrame [16], and Rappture [27] aim to provide a specific solution to enable applications to utilise DCIs with a minimal amount of development. All three frameworks are well-designed, actively maintained, and offer access to DCIs (see Section 3.4.2). Rapid supports the standard JSR168/JSR286 and, thus, is more flexible regarding the underlying infrastructure than EnginFrame and Rappture. The latter two rely on specific portals, EnginFrame on an EnginFrame server and Rappture on nanoHub, respectively. The philosophy of Rapid is to deliver customized portlets via specifications of the interface, tasks, and resource descriptions in an XML file. Even more flexible logic in the portlets can be achieved via plugins using Jython [220]. With the XML file and optional plugin files as basis, Rapid allows the automatic creation of deployable portlets for various portal frameworks such as Liferay and Pluto. Hence, there is no need for becoming acquainted with portlet programming. These features as well as the flexible and scalable design of Rapid made us choose it for an extension for workflows.

We developed a first prototype [221] for analysing mass spectrometry data via TOPP (The OpenMS Proteomics Pipeline) [222] and used the Jython plugin. After this prototype has been deployed in a Liferay portal framework, it allowed researchers to handle pipelines for analysing data on a large scale. The users have been able to select different TOPP tools, to monitor the pipeline, and visualise the results (see Fig. 8.1). However, to integrate a fully functional workflow management system, the decision was made to design an extension in Rapid that supports an existing workflow-enabled grid portal. Thus, we have developed an extension in Rapid for the use of WS-PGRADE workflows.

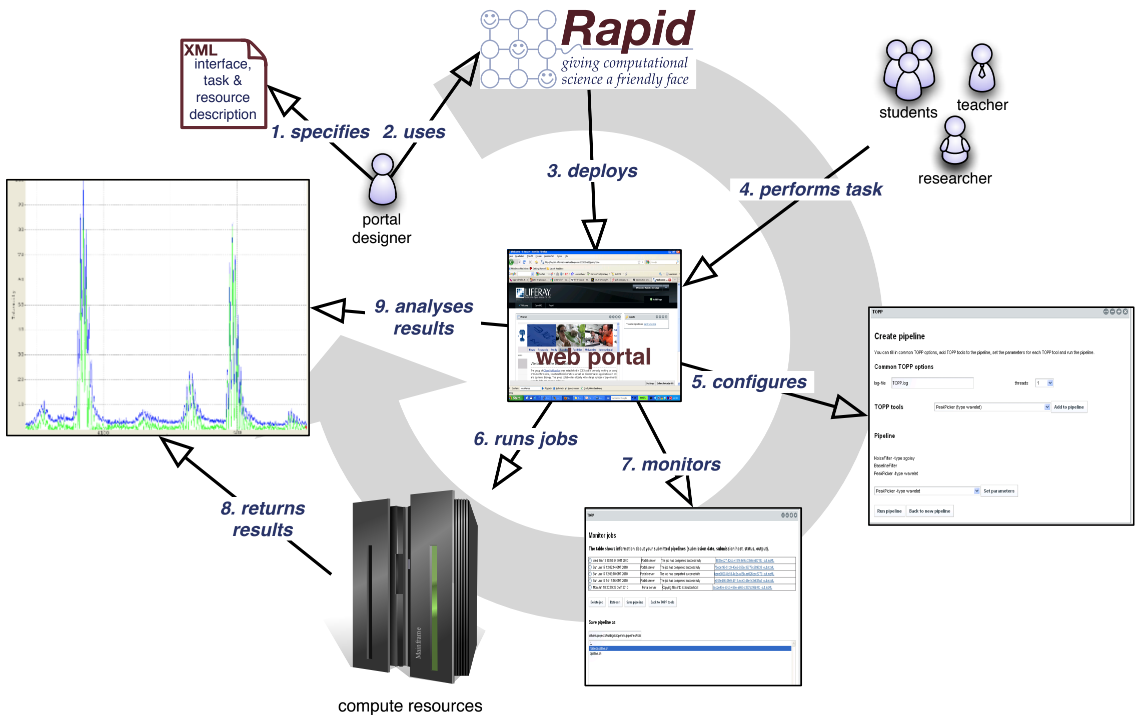


Figure 8.1: Process of creating and using TOPP portlets via Rapid

8.2 Rapid Framework

As described above, Rapid is designed for automatic creation of portlets from XML files and optional Jython plugins. These XML tags and Jython plugins are converted to Java and JSP source files. The creation process is independent of a portal framework, whereas the resulting portlet relies on JSR168/JSR286-compliant portals (e.g., Liferay, Pluto).

The conceptual architecture is shown in Figure 8.2. It reflects the separation between the layout and the job submission included in each resulting portlet. The layout is modeled via XHTML [223] and Rapid tags. Users may employ XHTML tags for tables or different fonts, the Rapid tags allow to integrate parameters like text boxes, drop-down menus, and radio buttons. Furthermore, a Java-based text editor can be inserted. The model of the job manager has been designed with regard to the standards Basic Execution Service (BES) [224] and JSDL. Hence, Rapid supports to define several stages of job submission; staging in and staging out of data, execution of a job including pre-processing and post-processing, and monitoring of jobs. Jobs can be invoked to several pre-configured DCIs.

Additionally, Rapid is able to invoke jobs via JSDL to GridSAM [225] services, a BES implementation. GridSAM supports grid middlewares like UNICORE, Globus Toolkit, and gLite.

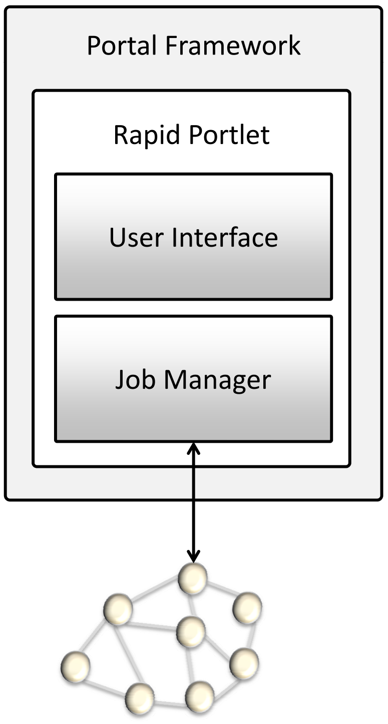


Figure 8.2: Conceptual architecture of Rapid

The available XML constructs in the Rapid format can be divided into two groups: XML tags for the layout and XML tags for job and data management. The root element of each Rapid document is the tag `<rapid>`. Via the child elements, the targeted DCIs, the file handling, the details of the job submission, and the layout are defined. The following child elements exist.

1. <condor> defines Condor [226] as targeted DCI for the job submission (optional)
2. <sungridengine> defines OGE (formerly SGE) as targeted DCI for the job submission (optional)
3. <pbs> defines PBS as targeted DCI for the job submission (optional)
4. <fork> defines a local resource as target for the job submission (optional)
5. <local> defines a local file system (optional)
6. <ftp> defines an (s)ftp file system (optional)
7. <http> defines an http file system (optional)
8. <gsiftp> defines the gsiftp file system (optional)
9. <initialise> initialises a new job template

10. <page> defines the user interface
11. <persistence> defines state preservation (optional)

1. - 4. reflect the different pre-configured types of DCIs, which can be targeted for the job submission (GridSAM is accessible via a job template), 5. - 8. contain the supported protocols for file handling, 9. defines the job properties, 10. defines the user interface, and 11. allows to set persistence characteristics. Each element allows further definitions and all available Rapid tags are provided in "Manual: Writing a portlet using the Rapid tool" on Sourceforge [227].

8.3 Extending Rapid for Workflows

For the extension of Rapid, ASM (see Section 6.4) is applied. It provides a possibility to add workflow functionality to portlets tailored for WS-PGRADE. However, if it is solely used, the developer still has to become acquainted with implementing portlets. The extension of Rapid using ASM supports an intuitive way to create portlets via XML files. Features developed with ASM require that the processed workflows exist in the repository of the executing WS-PGRADE instance. Thus, the developer first has to create the workflow that he wants to integrate in a portlet and to add it to the repository. The XML file of the workflow is used as additional input file for the portlet creation.

Furthermore, extra Rapid tags have been defined for the integration. The tag <ws-pgrade> contains the name of the WS-PGRADE workflow and is mandatory for initialising further steps for the workflow integration. Dependent on the features the developer wants to define in the portlet, the following tags can be added as child elements of <wspgrade>.

- <import> imports an application to the user space
- <upload> and <download> manage input and output files for upload and download
- <remoteinput> and <remoteoutput> manage remote files
- <configure> configures the properties of a specified job
- <submit> submits the application
- <status> gets the status of the application
- <abort> aborts an application
- <delete> deletes an application

If the Rapid process detects <wspgrade>, it automatically copies prepared portlet snippets in the resulting Java source code for the specified actions like importing the workflow to the user space as well as submitting and monitoring a workflow. These snippets only provide the feature and not the layout. The layout still relies on the extensive features of Rapid. In case the developer wants the same layout for the configuration of jobs as it appears in WS-PGRADE, the tag <standardview> can be added and the source code including the DCI-specific WS-PGRADE plugins (see Section 6.2.2) is automatically copied for the generation of the portlet.

Additional features can be added via supporting XML files for tools in the workflow. Each XML file reflects the parameter options for a tool, e.g., a list of choices, free text, or files. When the XML file is parsed, the parameters are mapped to suitable Rapid tags for the layout.

8.4 Results and Discussion

This section introduced a construction method for workflow-enabled grid portlets using Rapid that allows creating intuitve graphical user interfaces without programming in the traditional sense. Developers are enabled to automatically convert XML files (a Rapid XML file, a WS-PGRADE workflow XML file, optional XML files for tools) to a fully functional portlet for WS-PGRADE. The process of creating such portlets implies that the developer additionally creates a workflow in WS-PGRADE, import it to the repository of the executing instance, and downloads the workflow XML file before the portlet can be generated via Rapid. The generation method applies ASM and relieves developers from becoming acquainted with details about the underlying workflow system gUSE and about DCIs. All DCIs available in the executing WS-PGRADE instance can be applied. Thus, developers are enabled to focus only on the layout, which is maintained easily and efficiently in XML files.

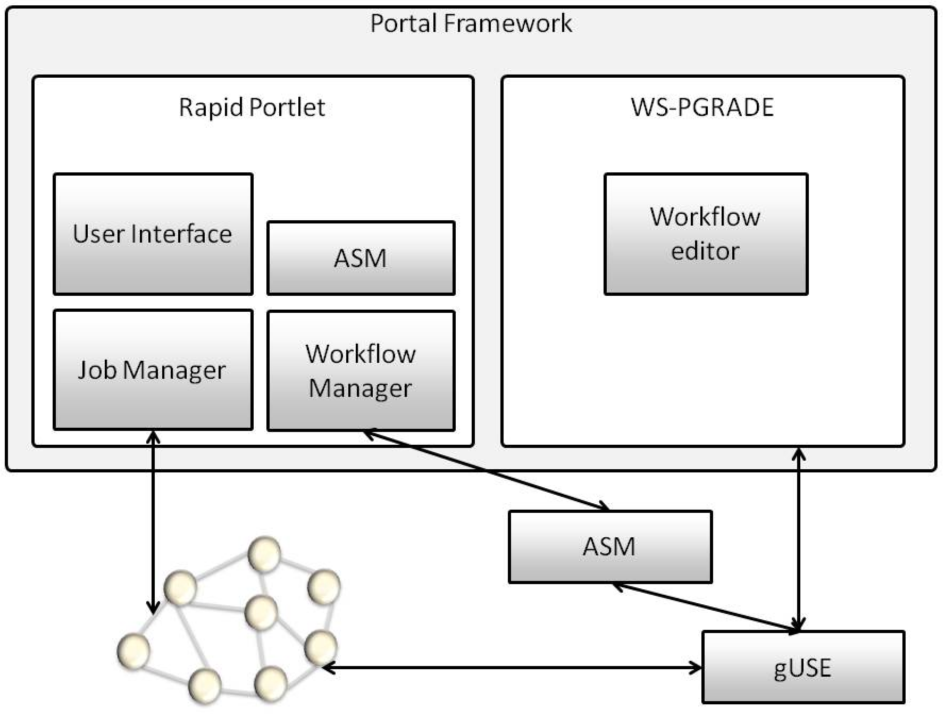


Figure 8.3: Extended conceptual architecture of Rapid for workflow-enabled grid portlets. ASM is integrated in Rapid for the generation of the workflow-enabled portlets. In addtion, gUSE and ASM have to be available for the execution of the workflow manager.

The integration of WS-PGRADE workflows into Rapid is an efficient solution for developers. A minor drawback is the loss of independence of Rapid from portal frameworks and workflow-enabled grid portals via the workflow extension. Rapid for WS-PGRADE workflows requires a more complex underlying infrastructure. The architecture of Rapid without the extension is self-contained and portlets can be generated in every JSR168/JSR286-compliant portal framework. The extension with the workflow manager using ASM always requires WS-PGRADE, gUSE, and ASM in the grid portal infrastructure (see Fig. 8.3).

Hence, Rapid for workflow-enabled grid portals only supports the portal framework that is applied by WS-PGRADE, which is currently Liferay. Even though Rapid including the workflow extension has lost its independence, the possibilities for the design of the portlets are more powerful compared to EnginFrame and Rappture. Both software frameworks only offer pre-configured layouts for creating new user interfaces via XML files, whereas Rapid supports XHTML and Rapid-specific tags for the layout.

9 MoSGrid - Bringing It All Together

Parts of this chapter have been previously published in [29].

9.1 Introduction

Structural bioinformatic tools have become indispensable tools in many fields of research in life sciences and chemistry. Quantum chemical methods, molecular dynamics methods, and protein-ligand docking provide deep insights into the structure of biomolecules and their interactions and are essential tools in such diverse areas as material science and drug design. While very powerful, most of the tools and applications used for computational chemistry calculations reflect the complexity of the underlying scientific theories. Using these tools requires a lot of experience. Their usability is often limited and frequently deters novice users. The computational complexity of these methods, the jobs' running times, and the huge amount of data make them ideal candidates for DCIs. However, DCIs have usability issues of their own, which limit their acceptance in the scientific community. Overcoming these usability issues and thus enabling the use of DCIs for a broader user community was the key goal when the MoSGrid project was conceived. It is part of the D-Grid and is designed to address the requirements of both commercial and academic users.

Currently, the MoSGrid community consists of about 100 users or working groups, respectively. At this stage, the science gateway is opened for beta testing to about 20 users from academia and industry whose feedback and demands are invaluable for further development. It is planned to offer the science gateway to the whole community in the near future. Novice and advanced users are able to run their computations on DCIs. They are assisted by graphical user interfaces with different levels of sophistication to accommodate both user groups. Additionally, standard methods for specific problem classes are offered. MoSGrid provides a framework for developing, storing, and providing simple and complex workflows.

The infrastructure of the MoSGrid science gateway and its underlying DCIs is reflected in this work. The science gateway is based on the workflow-enabled grid portal WS-PGRADE on a version which employs Liferay 6 and Apache Tomcat 6. The authentication and authorisation to the portal is performed via the user management of Liferay (see

Section 4.5) and to the DCIs via SAML assertions generated with the extended certificate portlet of WS-PGRADE (see Section 5). Even though WS-PGRADE offers the access to several DCIs and D-Grid supports computing infrastructures utilising UNICORE, Globus Toolkit, and gLite, the MoSGrid portal only allows access to UNICORE infrastructures and local resources. This decision has been made because of the additional features of UNICORE with the integrated workflow system and the IDB to store preferences of resources and installed tools. Thus, the developed submitter in gUSE including the workflow interoperability with UNICORE is applied (see Section 6).

Furthermore, users will be enabled to archive the results of their calculations in a repository and share them with other users. The repositories will be stored in XtreemFS whose security infrastructure has been extended for the use of SAML assertions. To provide an efficient data management between UNICORE jobs and XtreemFS, UNICORE has been enhanced to work together with shared and remote data storages controlled by XtreemFS. The included repositories will store data in MSML (Molecular Simulation Markup Language). It has been designed for specifying structural information of small and large molecules and results of various molecular simulation tools and docking tools.

Besides these features, portlets for domain-specific workflows have been implemented which make use of the ASM libraries and are tailored to the specific demands in the targeted domain. The development has been started right at the beginning of the MoSGrid project and has been continuously adapted to meet the community's feedback. Whereas a first prototype for the quantum chemistry domain [228] evaluated and employed ICEFaces [229], the first prototype for handling data management with XtreemFS evaluated and employed Vaadin. The evaluation resulted in using Vaadin for further developments. In parallel, the ASM library has been developed and the below introduced portlet for molecular dynamics (see Section 9.2.1) was the first one leaving the prototype state while applying ASM and Vaadin. We will additionally introduce a new designed portlet for quantum chemistry and a prototype for docking tools. Finally, Section 9.3 goes into detail for case studies of the community regarding the running time on DCIs.

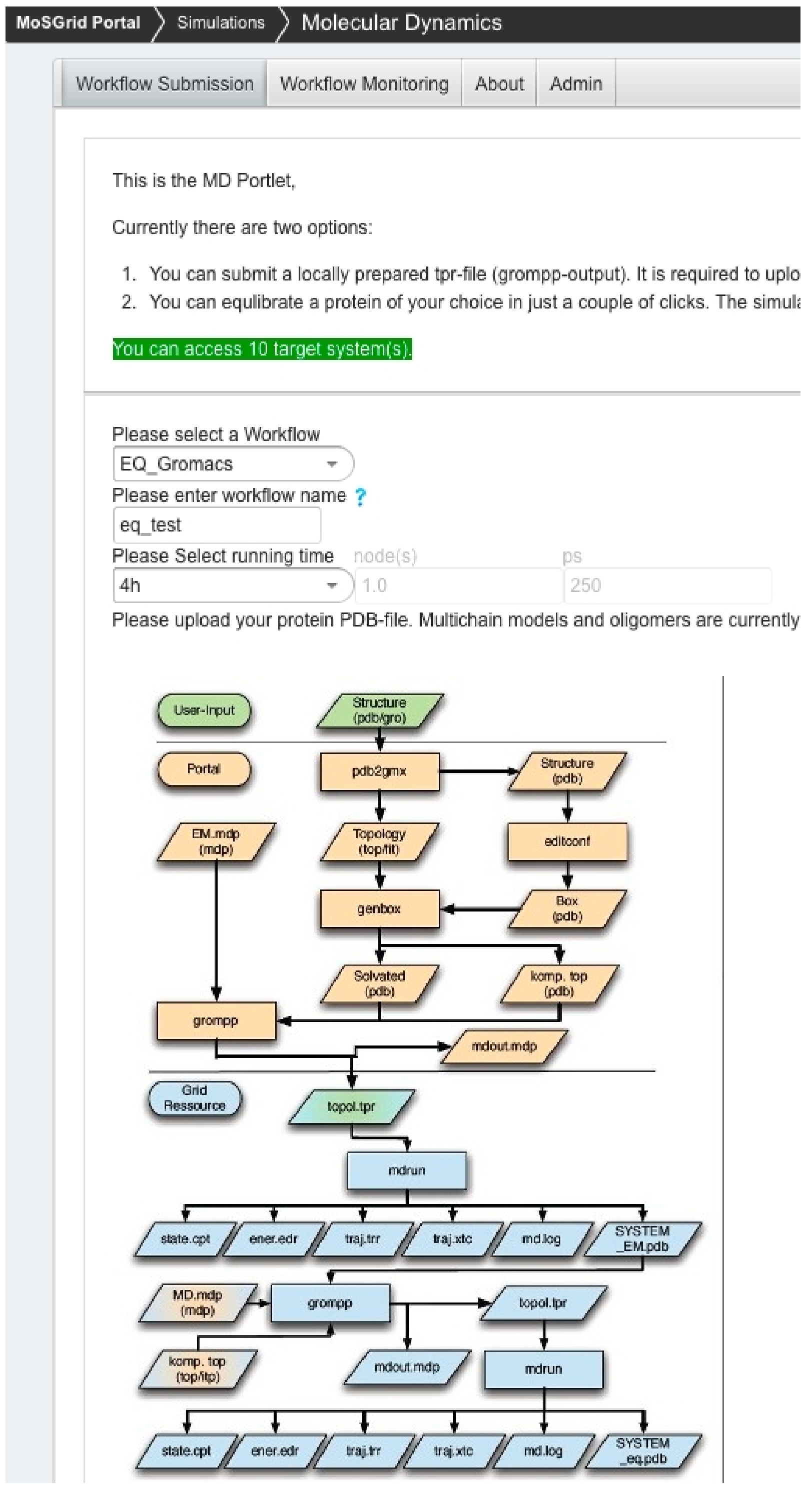
9.2 Domain-specific Workflows

In addition to the WS-PGRADE-based workflow-oriented instruments to use DCIs, MoSGrid aims to provide intuitive ways for novice users to run chemical simulations. To serve this purpose, the chemical simulation codes, workflows, and DCIs are hidden. The user accesses portlets that directly offer instruments to start and manage simulations for different subjects of structural bioinformatics.

Currently, MoSGrid offers specific portlets for molecular dynamics, quantum chemistry, and protein-ligand docking. The portlets utilise the ASM library together with the gUSE services like the UNICORE submitter. This enables the developers to focus on the domain-related features to further improve the user experience. The design and functionality of the domain specific portlets are described in the following.

9.2.1 The Molecular Dynamics Portlet

The Molecular Dynamics (MD) portlet enables chemists to easily access molecular simulation codes. The portlet allows chemists to simulate the processes of frequently used standard recipes.



MoSGrid Portal 〉 Simulations 〉 Molecular Dynamics

Workflow Submission | Workflow Monitoring | About | Admin

This is the MD Portlet,

Currently there are two options:

1. You can submit a locally prepared tpr-file (grompp-output). It is required to uplo
2. You can equlibrate a protein of your choice in just a couple of clicks. The simula

You can access 10 target system(s).

Please select a Workflow: EQ_Gromacs

Please enter workflow name ?: eq_test

Please Select running time: 4h — node(s): 1.0 — ps: 250

Please upload your protein PDB-file. Multichain models and oligomers are currently

Figure 9.1: Preparing a recipe with the MD portlet - at the top parameters can be adjusted and a recipe chosen and in the bottom the whole recipe is shown

These recipes are mapped to UNICORE workflows and are available in the portal. This way, the portal should on the one hand ease the work of experienced users and on the other hand lower the hurdle on using molecular simulations on DCIs for novice users. The scientists can submit molecular simulations without knowledge of the DCI. The MD portlet is organized in two main sections. The first section covers the recipe configuration and submission and the second section covers the monitoring of running simulations.

Submission - The MD submission section is designed to provide a molecular dynamics service on multiple levels. Figure 9.1 shows the design. When the user selects the MD portlet, he/she is welcomed with a short description of the portlet's abilities. In addition, the number of available cluster systems is displayed. To get this information, the portlet has taken the user's credentials to connect to the underlying DCI and detect which computing facilities can be accessed. If the user cannot connect to any facility, maybe due to missing SAML assertions, he/she is informed about the problem.

The MD simulation allows the user an easy use of standard workflows. In the current state, the user is enabled to submit a single simulation using a directly uploaded job description. Alternatively, the user can run a complex recipe that includes an energy minimization and a subsequent equilibration. This recipe is an indispensable prerequisite for all kinds of production runs. The portal supports the user with a description of the purpose and requirements of the selected simulation.

For each simulation the user has to upload a file containing either the job description (Gromacs TPR format) or the structural information (PDB format). In the background, the portlet automatically checks the correctness of the job description. Unnecessary input information is automatically removed. The user is notified about obvious problems like missing residues in the input structure.

Even if the portlet minimises the necessary user input as far as possible, it still requires further information. First of all, it is hard to guess how long a molecular simulation should run [230]. Therefore, the user has to define the simulation length. Secondly, the user has to define the resources for the simulation. This includes the number of parallel nodes and the maximum duration of the simulation (wall time). When all information is given and checked, the user can submit the job to the MoSGrid infrastructure.

The job submission uses the grid connection through the WS-PGRADE and gUSE infrastructure and the UNICORE submitter. The MD portlet employs the user's SAML assertion for the authentication and accesses the submitter through the ASM module. Alternatively, the connection can be established to the UNICORE grid middleware directly.

The UNICORE submitter checks the IDBs of the connected grid clusters. Each IDB contains meta information about the simulation codes, available in its cluster system. This allows to select the appropriate cluster for executing the recipe's substeps.

Monitoring - After job submission, the user can monitor the job process in the second main section. The jobs are either named after a user given identifier, or if the user did not specify a job name, the jobs are named after the user login on the portal, combined with the submit time, and name of the workflow recipe. A traffic light for each simulation entry shows the status of the simulation.

Yellow indicates that the job is still executing or queued.

Green means that the job has been successful.

Red is used for a failed job.

Further information, e.g., about the underlying HPC facility, which the job uses, is hidden.

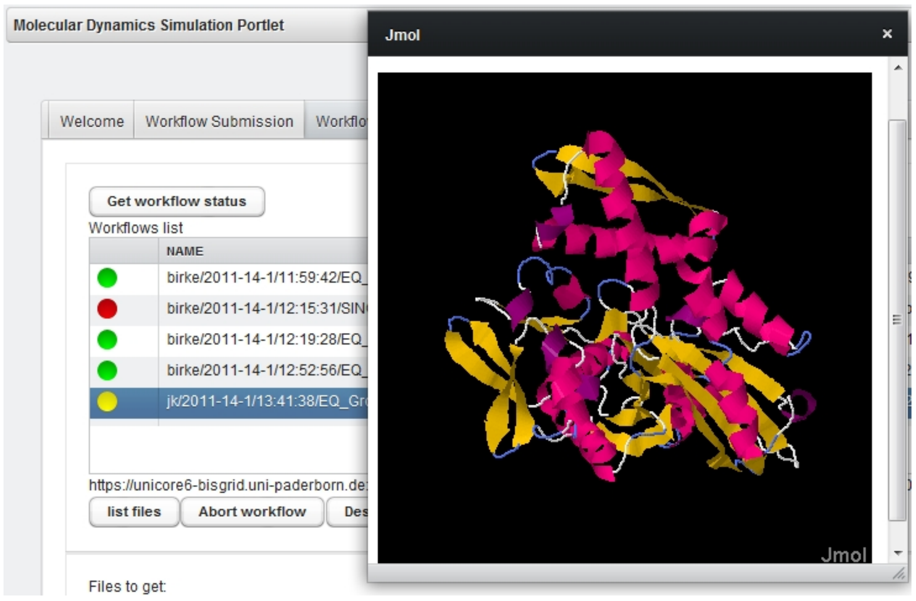

Molecular Dynamics Simulation Portlet
Welcome
Workflow Submission
Get workflow status
Workflows list
NAME
birke/2011-14-1/11:59:42/EQ_
birke/2011-14-1/12:15:31/SIN
birke/2011-14-1/12:19:28/EQ_
birke/2011-14-1/12:52:56/EQ_
jk/2011-14-1/13:41:38/EQ_Gr
https://unicore6-bisgrid.uni-paderborn.de
list files
Abort workflow
Files to get:
Jmol
Jmol

Figure 9.2: MD portlet - monitoring and view of a molecule file in Jmol

For each recipe the user can query the output files, even for an ongoing simulation. The files are displayed in a tree-like shape, reflecting the substeps of the running workflow. The resulting or intermediate file of the workflow can be downloaded or displayed in the portal. The MD portlet shows either plain text, pictures or figures, and in case a molecule file is selected, a 3D view in Jmol (see Fig. 9.2).

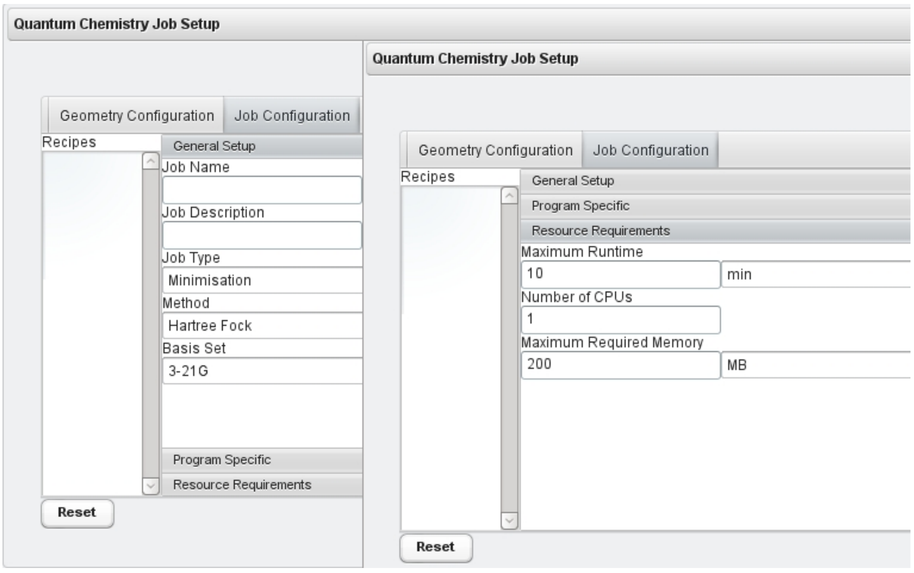
In the future the MD portlet will be enhanced with more simulation codes, additionally to the currently supported Gromacs.

9.2.2 The Quantum Chemistry Portlet

The Quantum Chemistry (QC) portlet implements a complete quantum chemical workflow. The platform enables researchers to submit their molecular simulations, monitor the progress, and retrieve the results. It is designed for users both experienced and inexperienced with computational chemistry. Therefore, pre- and post-processing routines are available. Amongst others, these can be used to export the output of the simulation tools in a standardised format.

The chosen software design is open for extensions. Instead of directly accessing the UNICORE command line client (UCC), the portlet is implemented to make use of ASM. With the use of ASM, the authentication via SAML is processed by the UNICORE submitter. To be able to use the ASM it was required to create WS-PGRADE workflows. First simple workflows include the execution of the specific tool and parsers for data evaluation. The support for MSML/CML [231, 232, 233] is fully functional, but relying on preliminary definitions, which are not yet standardised. So, while being able to generate valid MSML, the input files are still generated and submitted in tool specific formats.

Taking the users' feedback into account, the user interface was split into two portlets; the first is specialised to creating and submitting the workflows, the second portlet monitors the workflows and visualises the results.



Job Name

Job Description

Job Type: Minimisation

Method: Hartree Fock

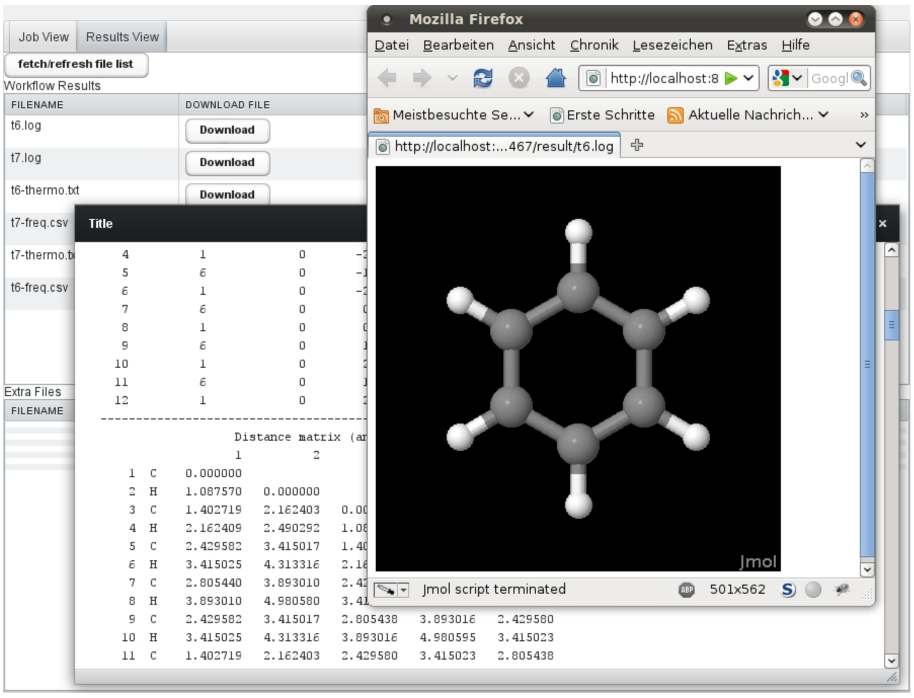
Basis Set: 3-21G

Maximum Runtime: 10 min

Number of CPUs: 1

Maximum Required Memory: 200 MB

Figure 9.3: QC portlet - configuration of a workflow



Workflow Results

FILENAME	DOWNLOAD FILE
t6.log	Download
t7.log	Download
t6-thermo.txt	Download
t7-freq.csv	
t7-thermo.tx	
t6-freq.csv	

```
      4          1           0        -
      5          6           0        -
      6          1           0        -
      7          6           0
      8          1           0
      9          6           0
     10          1           0
     11          6           0
     12          1           0
 ---------------------------------------
                    Distance matrix (a
                     1          2
   1  C    0.000000
   2  H    1.087570   0.000000
   3  C    1.402719   2.162403   0.0
   4  H    2.162409   2.490292   1.0
   5  C    2.429582   3.415017   1.4
   6  H    3.415025   4.313316   2.1
   7  C    2.805440   3.893010   2.4
   8  H    3.893010   4.980580   3.4
   9  C    2.429582   3.415017   2.805438   3.893016   2.429580
  10  H    3.415025   4.313316   3.893016   4.980595   3.415023
  11  C    1.402719   2.162403   2.429580   3.415023   2.805438
```

Figure 9.4: QC portlet - result view with Jmol

Creating and submitting a workflow follows the structure-centric approach. First, the user selects the geometry of a molecule. Currently, there is support for uploading files and entering the information directly. OpenBabel [234] is used to support various geometry file formats.

The second step, presented to the user in a second tab, is the configuration of the workflow (see Fig. 9.3). Here, parameters can be selected in three categories. The first category summarises the generic parameters of QC calculations. This includes the job type specification (e. g., optimization, energy minimization) and the selection of the simulation method. The second catagory allows the user to select a specific tool for the simulation and to set up tool specific parameters. The third and last tab offers access to hardware-specific settings like runtime, number of processors, and the amount of memory needed.

The monitoring and display of results was transferred to a second, independent portlet. Originally it was designed for the monitoring of all tools from all domains. With the ASM integration the listing of workflows uses a tree-like structure. For the current single-step-workflows there is a list entry for each workflow. The status is represented by the well-known items queued, running, successful, or failed. On the right, the most important information for each job is summarised. In case of a successful execution, the exit code of the tool is shown as well as the information that result data is available.

This data can be viewed on the second tab of the monitoring portlet, where detailed information and visual representation is provided (see Fig. 9.4).

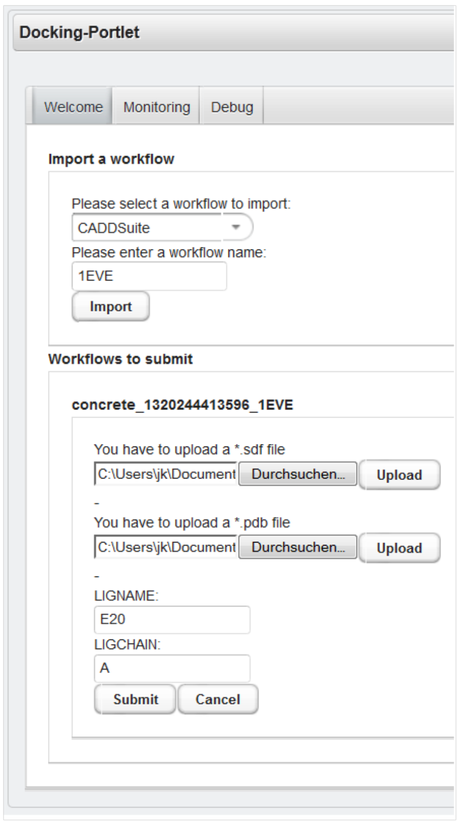
Depending on the input data, the native output format of the simulation tool, specific values and the trend of these values are plotted to files, which can be viewed, downloaded, and processed in common spreadsheet applications. Visualisation of the data is provided by Jmol.

9.2.3 The Docking Portlet

The docking portlet supports the use of docking tools via workflows and visualises the results. At the moment, it is available as a prototype and offers access to pre-defined workflows with CADDSuite tools. The design originates from lessons learned via the development of the MD portlet and its integration of the ASM libraries. The users are provided with three views; a welcome view for importing workflows and submitting them (see Fig. 9.5), a monitoring view (see Fig. 9.6), and a debug view. This layout will be the model for further developments of portlets in MoSGrid and its ASM features will be used as a kind of template.

The welcome view allows to import workflows from the workflow repository of WS-PGRADE via ASM to the user space. This way, users are supported with pre-defined workflows. Since advanced users are additionally able to create a WS-PGRADE workflow from scratch and to export these to the repository, the integration of a new workflow in the welcome view is easily possible. Once a user has imported a workflow, it is offered for submission, input data can be uploaded, and several parameters can be filled, e.g., the name of the ligand. The status of workflow is shown in the monitoring view (queued, running, successful, or failed). Workflows can be deleted and in case, a workflow is finished, the results can be downloaded and visualised in Jmol. For a more flexible use, the browsing of intermediate results will be integrated in the future similar to the MD portlet.

Currently, a workflow is provided that performs protein-ligand docking via an iterative multi-greedy docking method and rescoring algorithm. The molecules are prepared with several steps like separating ligand and receptor and finding strongly bound water molecules. By splitting the molecule file of the ligands, steps for the chemical sanity check of the ligand and the docking can be parallelised. Finally, the docking results are merged and filtered.



Docking-Portlet

Welcome | Monitoring | Debug

Import a workflow

Please select a workflow to import:
CADDSuite

Please enter a workflow name:
1EVE

Import

Workflows to submit

concrete_1320244413596_1EVE

You have to upload a *.sdf file
C:\Users\jk\Document | Durchsuchen... | Upload

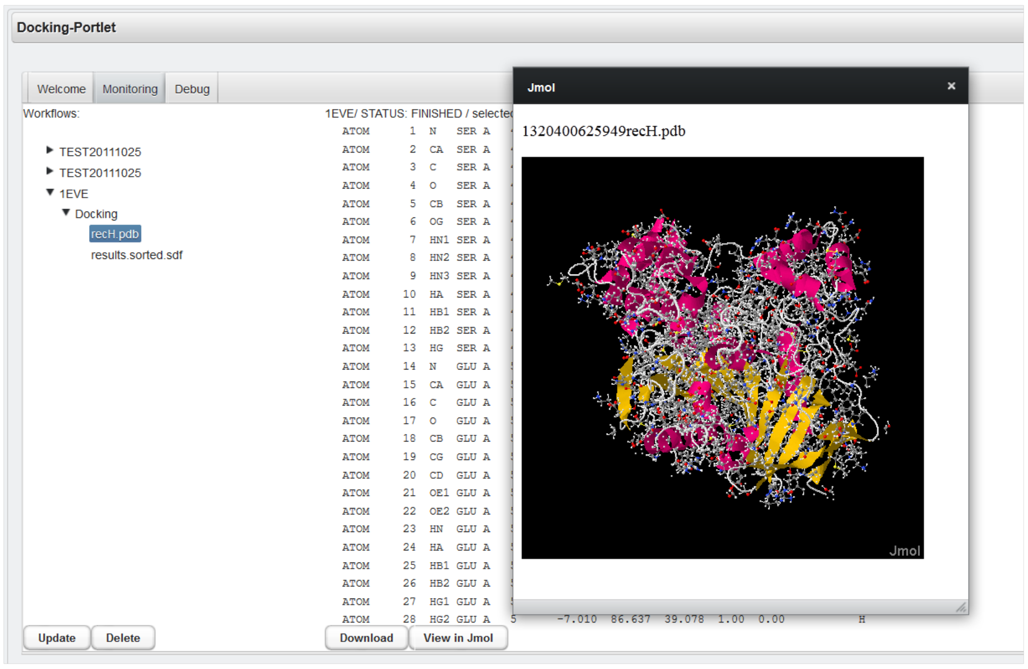
You have to upload a *.pdb file
C:\Users\jk\Document | Durchsuchen... | Upload

LIGNAME:
E20

LIGCHAIN:
A

Submit | Cancel

Figure 9.5: Docking portlet - configuration of a workflow



Docking-Portlet

Welcome | Monitoring | Debug

Workflows:

- TEST20111025
- TEST20111025
- 1EVE
  - Docking
    - recH.pdb
    - results.sorted.sdf

1EVE/ STATUS: FINISHED / selected

```
ATOM      1  N   SER A
ATOM      2  CA  SER A
ATOM      3  C   SER A
ATOM      4  O   SER A
ATOM      5  CB  SER A
ATOM      6  OG  SER A
ATOM      7  HN1 SER A
ATOM      8  HN2 SER A
ATOM      9  HN3 SER A
ATOM     10  HA  SER A
ATOM     11  HB1 SER A
ATOM     12  HB2 SER A
ATOM     13  HG  SER A
ATOM     14  N   GLU A
ATOM     15  CA  GLU A
ATOM     16  C   GLU A
ATOM     17  O   GLU A
ATOM     18  CB  GLU A
ATOM     19  CG  GLU A
ATOM     20  CD  GLU A
ATOM     21  OE1 GLU A
ATOM     22  OE2 GLU A
ATOM     23  HN  GLU A
ATOM     24  HA  GLU A
ATOM     25  HB1 GLU A
ATOM     26  HB2 GLU A
ATOM     27  HG1 GLU A
ATOM     28  HG2 GLU A   5      -7.010  86.637  39.078  1.00  0.00           H
```

Update | Delete | Download | View in Jmol

Jmol

1320400625949recH.pdb

Figure 9.6: Docking portlet - monitoring of a workflow and visualisation of the result with Jmol

The next steps for the extension of the portlet include the visualisaton via WebGL [235] tools, which allow not only the visualisation of molecules but provide a fully powered molecule editor.

9.3 Performance Studies

There have been a plenty of benchmarks and performance studies about tools in computational chemistry regarding the increasing efficiency via parallelisations on DCIs (e.g., for Gromacs [236, 237, 238, 239], for Gaussian [240, 241, 242, 243], for FlexX [244, 245, 246]). Vendors of computer systems as well as independent research facilities often publish performance values. The overall result is that each tool has to be examined per computer architecture and per use case for the best rate of parallelisation. Above a certain number of parallel processes and simultaneously applied cores, the overhead of parallelisation causes parallel processing to be less efficient than serial processing. In [238], a benchmark for Gromacs parallelised on 128 cores is introduced, which shows that there is still a speedup for the application. However, already 14 % of the running time was needed for communicating. Gaussian speedups with 32 concurrently used cores for selected applications [241]. Borcz et al. [247] developed workflows via the UNICORE Rich Client which reduces the necessary analysis time for researchers at least six times.

Users of the MoSGrid science gateway have currently access to seven clusters for managing their workflows (see Table 9.1).

Table 9.1 Technical specifications of the clusters Alibaba, Bisgrid, Deimos, Emilia, Flavus, HPC-BW, SuGI, and Atlas

Cluster Name	CPU	Nodes	Cores per Node	Network
Alibaba	Dual-core Intel Xeon 2.8 GHz	80	6	Infiniband
Bisgrid	Dual-core AMD Opteron 2.8 GHz	8	8	Infiniband
Deimos	Dual-core AMD Opteron 2.6 GHz	384	2	Gigabit
	Dual-core AMD Opteron 2.6 GHz	230	4	Gigabit
	Dual-core AMD Opteron 2.6 GHz	112	8	Gigabit
Emilia	Quad-core Intel 3.0 GHz	43	8	Gigabit
Flavus	Quad-core Intel i7 2.8 GHz	8	4	Gigabit
HPC-BW	Quad-core Intel Xeon 2.83 GHz	140	8	Infiniband
SuGI	Quad-core Intel Xeon 2.33 GHz	32	8	Infiniband
Atlas (1)	16-core AMD Opteron 2.2 GHz	92	64	Infiniband

(1) It is planned to offer access to Atlas in the second quarter of 2013.

Each job of a workflow is invoked via the UNICORE submitter on a suitable resource (e.g., suitable to the selected tool, suitable to the requested number of cores). Often the number of simultaneously allowed cores and processes, size of memory and of storage of DCIs are limited to each user by computer centres serving various user communities at the same time. Additionally, available software licenses have to be considered for the choice of resources. For example, the SuGI cluster in Cologne is the only cluster of the MoSGrid resources, which owns the suitable license for MoSGrid users to perform Gaussian jobs. The use case on conformational analysis of guanidine zinc complexes presented by Herres-Pawlis et al. [248] was carried out using 110 Gaussian 03 jobs on the SuGI cluster. 80

jobs could have started in parallel, whereas 30 conformations were calculated dependent on the first 80 results. However, the number of allowed nodes for one user at the same time is restricted to 20. Each run occupying eight cores took about 3 days on the cluster. A similar calculation on a standard PC was aborted after 14 days.

A standard workflow for energy minimisation of a molecular structure (see Fig. 9.7) offered by the MoSGrid science gateway is applied for the following performance study. This workflow is often used in molecular dynamics for the preparation of molecular structures before performing further molecular simulations. The preprocessing step includes the choice of a force field and the generation of an appropriate initial configuration for a molecular system. Then the system is inserted into a periodic box and solvated with water. Afterwards the total charge is neutralised by the addition of counter ions. The workflow ends with the energy minimisation of the molecular structure. We employed Gromacs 4.5.5 and the first four steps of the workflow are processed serially and the fifth step can be performed in parallel. The workflows were invoked via the science gateway to HPC-BW and Flavus with the structures of the proteins Acetylcholinesterase complexed with Galanthamine containing 4210 atoms (PDB ID [43] 1DX6) and Hemagglutinin in complex with ligand LSTa containing 45750 atoms (PDB ID 3UBJ) as input. Figure 9.8 illustrates the performance achieved on the two clusters. The running time for 1DX6 decreases almost linear to eight cores on HPC-BW and decreases further until 128 cores in sublinear manner. The running time for 1DX6 on Flavus decreases from one core to 32 cores in sublinear manner, whereas the running time for 3UBJ decreases only until 16 cores sublinear and then increases for 32 cores. The running time on HPC-BW for 3UBJ decreases even until 64 cores but increases again on 128 cores. The performance study shows that applying Gromacs in its parallelised version is beneficial for both molecular structures on the selected computer architectures at least up to 16 cores. The running time for 1DX6 was even decreasing until 128 cores on HPC-BW.

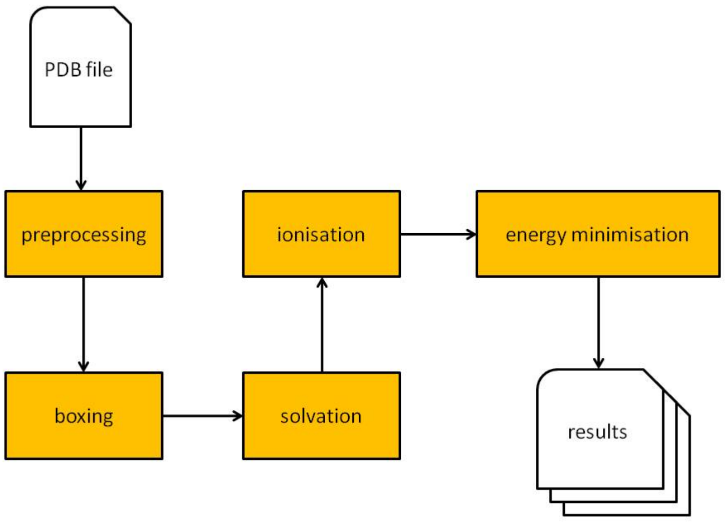


Figure 9.7: A workflow to minimise the energy of a molecular structure

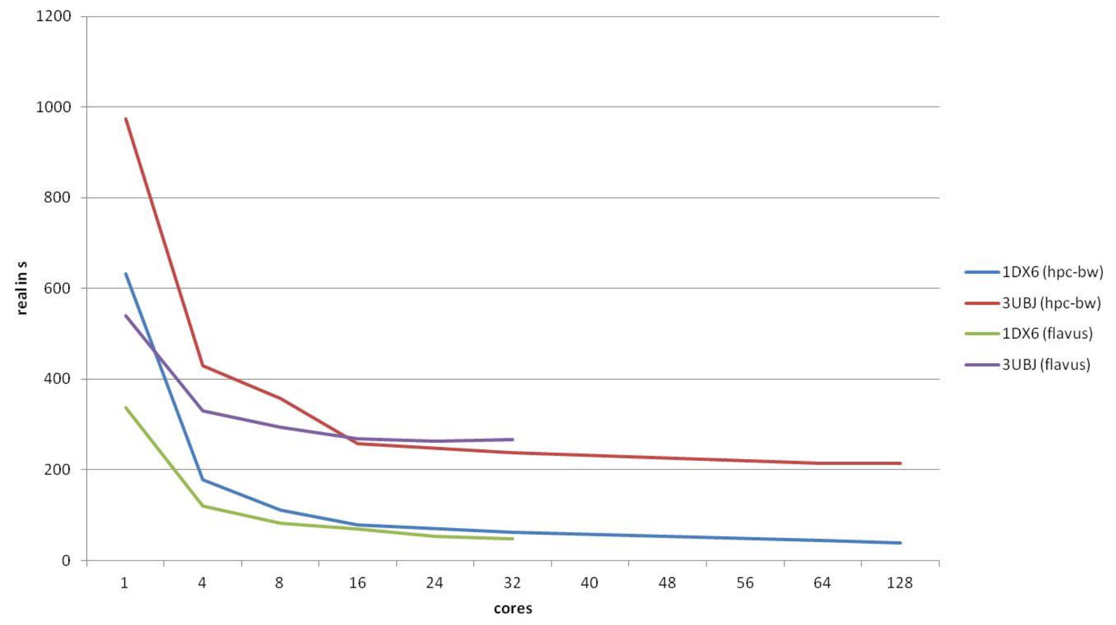


Figure 9.8: The running times of the energy minimisation for the structures 1DX6 and 3UBJ on HPC-BW and Flavus

For the following performance study we applied the docking workflow using CADDSuite, which is also offered in the docking portlet. Figure 9.9 illustrates this workflow, which includes the steps for the preparation of a receptor and the steps for the preparation of the ligands before the docking is performed. PDBCutter is used to split a complex into its receptor and ligand parts. The extracted ligand forms the reference ligand for defining the binding pocket. Missing hydrogen atoms are added to the receptor structure via Protein-Protonator. The ouput of PDBCutter as well as ProteinProtonator is used in GridBuilder to build an interaction grid for the binding pocket of the receptor. The resulting grid and receptor file set the stage to dock a set of ligands with IMGDock. In general, the ligands have to be prepared for docking by generating 3D conformations and adding hydrogens to them (Ligand3DGenerator). Furthermore, a sanity check (LigCheck) is performed. The sanity check examines, for example, whether the ligands are properly protonated, possess suitable bond lengths and properly assigned binding orders. The LigandFileSplitter allows to split the ligand file into a number of subsets. Thus, the workflow can be parallelised based on the number of splits.

As input we used a benchmark data set for the protein kinase ABL (PDB ID 2HZI) offered by DUD-E (A Dabase of Useful Decoys: Enhanced) [249]. The benchmark data set includes a molecule file with 10885 ligands. Additionally, a second set of files (lead-like subset) [250] containing about 5 million structures has been downloaded from the ZINC database [251], which is generally used for virtual screening. Virtual screening is applied in drug discovery to search large data sets of chemical structures in order to examine the structures' binding probability to a drug target. The structures in the data sets have been already prepared for docking. Thus, we skipped the two steps Ligand3DGenerator and LigCheck in the workflow. The resulting workflow has been submitted four times to

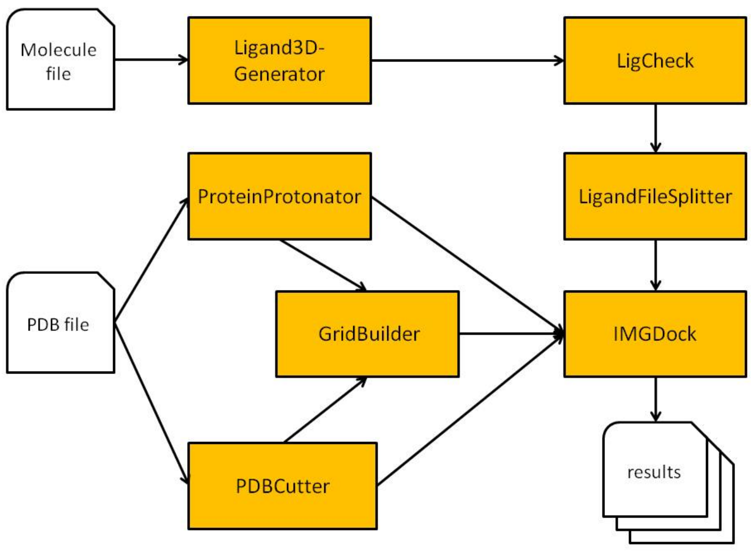


Figure 9.9: A docking workflow including the preparation of the receptor and the preparation of the ligands

Atlas. For the first three submissions we used the benchmark data set offered by DUD-E as input and split it into 10, 100, and 1000 subsets via LigandFileSplitter. The screening data set consisting of 41 single files formed the input for the fourth submission and these 41 files have been split each in 100 subsets. The CPU average time for a docking process per ligand (see Fig. 9.10) was lower and, thus, performed better the higher the number of molecules have been in the molecule file. However, the overall time for processing the different workflows, the time which is recognised by the users, resulted in a gain of performance in sublinear manner for the higher number of parallel jobs. For example, the number of ligands processed via the screening data set are about 500 times more than the number of DUD-E ligands but the duration of the complete docking process only took 67 times longer than the best running time with 1000 splits on the DUD-E ligands (see Fig. 9.11).

9.4 Results

The MoSGrid project serves the molecular simulation community with an intuitive web-based science gateway for the research areas molecular dynamics, quantum chemistry, and docking. The science gateway is based on the Liferay-version of WS-PGRADE including the extensions regarding the role-based user management (see Section 4.5), the novel credential management based on SAML (see Section 5), and the enhanced job and workflow management via WS-PGRADE (see Section 6).

Furthermore, the science gateway was extended with specific workflows and portlets for the community. The development of the portlets was an interactive process considering the feedback from the community. Thus, the user interface is especially tailored to the targeted domains and pre-configured workflows with default values are provided. Users are able to select from several options and are led through the whole process of editing,

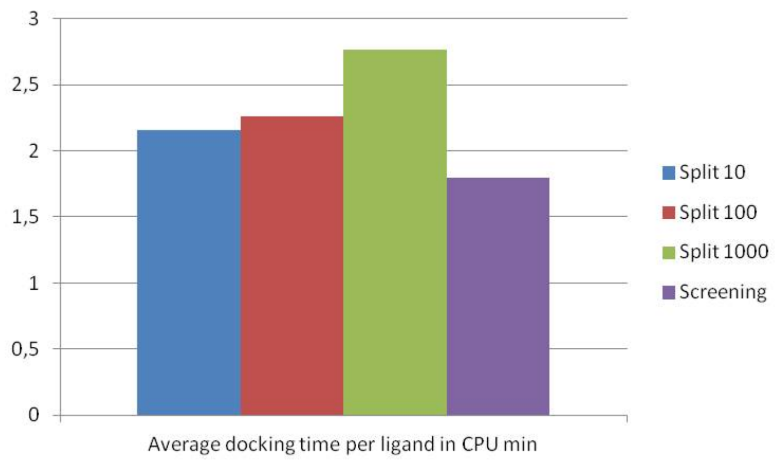


Figure 9.10: Performance study for docking the protein kinase ABL

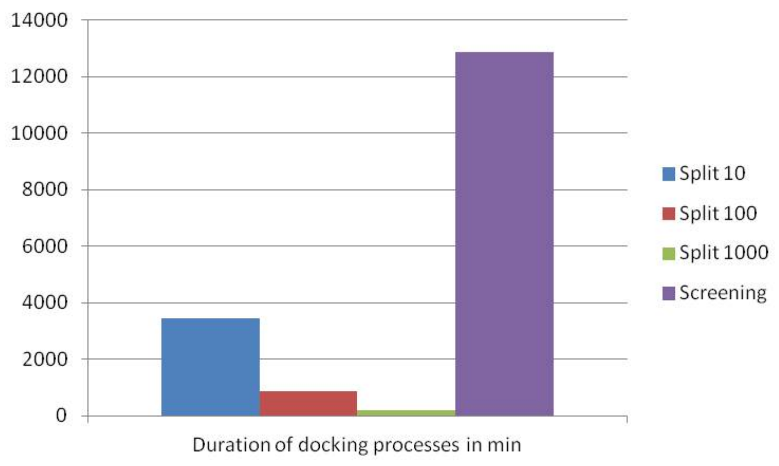


Figure 9.11: Performance study for docking the protein kinase ABL

invoking, and monitoring a workflow to process data on a large scale. The results can be visualised via Jmol. Advanced users, consortium users, and developers are additionally able to create workflows and share them via a workflow repository within the community. Also data repositories will be offered to the community. MSML was designed to store structural information on small as well as large molecules and results from various molecular simulation tools and docking tools. Besides the valuable information for the community to share knowledge about workflows, structural data, and molecular simulations, MSML ensures interoperability of different tools through a consistent data representation.

The science gateway provides access to the D-Grid infrastructure. Currently, seven clusters in different computer centres are integrated and users are able to employ them via a single sign-on mechanism. The performance studies show that parallelisation of tools and workflows using DCIs is beneficial for the performance of molecular simulations. The gain of performance submitting workflows via the science gateway compared to scripts starting the same tools via the commandline on these clusters depends on various factors.

First, the access to the clusters is handled by the science gateway via the single sign-on approach. The users are relieved from applying for accounts for each resource they want to use. In addition, the resources are heterogeneous regarding their architecture and batch systems, which requires creating scripts for the diverse batch systems (e.g., OGE, PBS). The science gateway supports diverse resources hiding the differences from the user. Furthermore, jobs of a workflow can be invoked on diverse connected clusters via the science gateway whereas scripts are bound to one cluster. Thus, users can access more resources and distribute their workflows. They are less dependent on the availability of one dedicated cluster.

In the context of science gateways, also soft measures for performance are important regarding the usability of infrastructures and the support of users. The ISO standard [6] defines usability as ”the effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in a particular environment”. Research of usability connected to science gateways covers a wide range of aspects like design aesthetics [252], peculiarities of science gateways [253], and testing of usability [254].

The MoSGrid portal was designed to target the usability aspects in various areas. The role-based user management meets authentication and authorisation requirements, but it additionally provides an appropriate view of features taking the knowledge level of users into account (e.g., the difference between novice users and expert users). The credential management via a certificate portlet, whose view is reduced to mandatory fields, ease its usability. The project adapts the user interfaces continuously to the feedback of users and adds domain-related workflows. For example, the possibility to automatically extract relevant information out of a Gaussian file for further analysis or to prepare molecules for docking tools via pre-defined workflows, saves time for the users and is sensed as comfortable by users.

10 Conclusion and Outlook

Parts of this chapter have been previously published in [29, 255].

Web-based science gateways with underlying DCIs have gained increasing popularity in various compute-intensive and data-intensive research fields for about ten years now. The overall goal is to enhance the usability of sophisticated tools and complex infrastructures. The main goal of this thesis was to provide a complete solution to aid end users and developers in the field of web-based science gateways for structural bioinformatics. The complexity of the applied methods and the volume of data created and analysed in structural bioinformatics clearly suggest the use of workflow-enabled grid portals. The interdisciplinary project MoSGrid has been conceived to support users in the field of molecular simulations, especially in the application areas quantum chemistry, molecular dynamics, and docking. The concepts, designs, and implementations presented in this work have been developed under consideration of this D-Grid project.

We evaluated existing workflow-enabled grid portals and their infrastructures considering the application domain, standards, and the D-Grid infrastructure (see Chapter 4). Criteria for the evaluation of the systems have been their embedded features, efficiency, reliability, and usability. The evaluation resulted in the service-oriented grid middleware UNICORE 6, the object-based grid and cloud file system XTreemFS, and the workflow-enabled grid portal WS-PGRADE employing gUSE services for managing workflows on DCIs. WS-PGRADE has been developed on top of the portal framework Liferay, which implements the widely-used standards JSR168/JSR286. We analysed the different user roles in the molecular simulation community and applied the role-based user management of Liferay to reflect these roles via technical roles in the science gateway. Even though the resulting infrastructure has been designed for the German molecular simulation community, it is re-usable on international level for diverse research areas.

Security is a key aspect for science gateways offering access to DCIs. The security of most established grid middlewares is based on X.509 certificates and the short-lived credentials SAML assertions or GSI proxy certificates. The advantages of SAML assertions compared to proxy certificates led to the decision to apply SAML in the science gateway. The science gateway's infrastructure was adapted in order to allow trust delegation from the user interface layer across the high-level middleware layer and the grid middleware layer down to the HPC facilities via SAML. The portlet for credential management of WS-PGRADE was extended for managing SAML assertions in this work. We integrated and adapted a signed applet for the automatic creation of these short-lived credentials (see Chapter 5). We showed that an applet is advantageous for the creation of credentials compared to the JAVA portlet library or a JWS application. The applet is a flexible and secure solution, which can be integrated into every web-based science gateway managing short-lived credentials for DCIs. The options in the applet and the certificate portlet were minimised to the mandatory options due to simplify their use for the community.

Other approaches like Murri et al. [194] and Barbera et al. [202] suggest to automatically generate short-lived credentials during the login procedure. Thus, the users are relieved from any interaction for creating a short-lived credential. However, they are not able to change properties of the short-lived credential nor to manually create diverse short-lived credentials. The latter option is interesting if a user has access to diverse DCIs with different requirements on the properties of the SAML assertion (e.g., allowed duration of validity). Thus, the design decision for a portlet is beneficial for a community having access to various DCIs, whereas the integration in the login procedure is suitable for one DCI. It is planned to offer the automatic creation during the login procedure to the German molecular simulation community in the MoSGrid science gateway. The open-source version of the MoSGrid science gateway will be offered for download including the integrated portlet to keep the flexibility. Representatives of the German NGI as well as of European NGIs discuss to apply security infrastructures based on Shibboleth facilitating SAML in the future. Since the science gateway already uses SAML, its security infrastructure can be easily extended to rely on Shibboleth instead of certificates for user authentication.

A second key aspect of workflow-enabled grid portals addressed in this work is the job and workflow management in DCIs. We enhanced the job and workflow management of WS-PGRADE and gUSE on different levels (see Chapter 6). First, the UNICORE submitter and the UNICORE plugin were added to allow job management on UNICORE DCIs. Secondly, we implemented the unique feature compared to other submitters in gUSE and other workflow-enabled grid portals to select available tools via an automatically generated list. Thus, users receive the complete information on the software packages installed on the connected UNICORE DCIs and maintained in the IDBs. The submitter ensures that the selected tools are invoked on suitable resources. To apply this beneficial approach for DCIs in general, the gUSE information system can be extended with a database table storing all available tools on the different DCIs. The methods developed in this work could be used to automatically keep the database up-to-date for UNICORE. UNICORE is the only grid middleware, which offers the information about the available software. However, the gUSE information system could be manually maintained for other DCIs. The supply of software in DCIs is not a dynamic process requiring daily maintenance of the database. In contrast, the users would benefit from selecting available tools during their daily work.

We additionally extended the submitter for workflow interoperability. The users can seamlessly re-use UNICORE workflows in WS-PGRADE and integrate these workflows into WS-PGRADE workflows. UNICORE workflows are treated as black box in WS-PGRADE and can be easily inserted by the users. To invoke the UNICORE workflow via WS-PGRADE, a UNICORE DCI has to be accessible for the user via WS-PGRADE. The UNICORE submitter has set the stage for the same approach followed for workflow interoperability between WS-PGRADE and UNICORE in the SHIWA platform [205]. The methods of the submitter for invoking and monitoring UNICORE workflows can be re-used in science gateways allowing workflow interoperability via web services. The prerequisite is that the science gateways additionally support credential management for SAML assertions.

Another enhancement of the workflow management of gUSE affects the flexibility of user interfaces extending WS-PGRADE. WS-PGRADE as well as workflow-enabled science gateways in general offer generic user interfaces for the workflow management. To support users with domain-specific portlets and to ease the implementation of such user interfaces, we developed the ASM API. It internally uses the gUSE services but offers an intuitive API for the management of the whole life cycle of workflows. Thus, the use of ASM supports the developers to focus on the layout and domain-specific features. The developers are relieved from becoming acquainted with the gUSE services in detail. They can employ any Java development framework they prefer (e.g., Vaadin, Google Web Toolkit) for the development of portlets.

The solution presented in Chapter 7 contributes to the re-usability of workflows via workflow language translation and migration of workflows between different science gateways. The migration assures that workflows are fully integrated in the targeted science gateway. The workflow-enabled portal Galaxy serves the structural bioinformatics community with a wide range of workflows. Galaxy is a kind of toolbox for cloud infrastructures and for HPC facilities, but lacks the support of grid-based DCIs. We embedded the export of Galaxy workflows as WS-PGRADE workflows in Galaxy. The essential tasks of the workflow language translation are formed by the graph algorithm calculating the coordinates of the WS-PGRADE workflow graph and semantic mapping of the suitable workflow constructs. The graph algorithm avoids edge crossing as far as possible and aligns vertices and edges to an underlying grid in linear running time. Purchase et al. [150] categorise these two criteria for the aesthetics of a graph as the most important criteria for users [150]. However, compared to other algorithms following a layered layout approach [147], our algorithm could be improved regarding additional criteria for aesthetics. For example, the number of layers could be minimised as in the widely-used algorithm of Sugiyama et al. [146]. The consideration of additional criteria would increase the time complexity of the algorithm for a graph $G = (V, E)$ at least to $O((|V| + |E|)log|E|))$ using efficient heuristics developed for Sugiyama's algorithm [149].

The workflow language translation from Galaxy workflow to WS-PGRADE workflows enables the WS-PGRADE community to re-use Galaxy workflows on diverse grid-based DCIs. Workflow migration including workflow language translation is used by several approaches. Galaxy also offers the option to migrate Galaxy workflows directly to myExperiment sandboxes [214]. Abouelhoda et al. [208] extended Galaxy to Tavaxy and designed an intermediate representation of workflows. Galaxy workflows can be directly integrated into Tavaxy workflows. The SHIWA project [205] also suggests to perform workflow language translation via an intermediate presentation of workflows. They use IWIR [207] for workflow language translation in a multistage process. Since the SHIWA platform is based on WS-PGRADE, the resulting workflows of our solution can be seamlessly integrated into the SHIWA platform.

In Chapter 8 we tackle the problem that portlet development is a time-consuming process, which demands specialised skills. The development of portlets for workflow-enabled grid portals increases the complexity significantly. Rapid [26] is a self-contained portal framework for automatic creation of portlets accessing DCIs without the need of programming in the traditional sense. Due to the lack of a standard for workflow-enabled portlets, we extended Rapid especially tailored for the management of WS-PGRADE workflows. Thus, developers are able to create domain-specific portlets for WS-PGRADE by simply providing XML files. A deployed portlet additionally requires the workflow in the repository of the executing WS-PGRADE instance. EnginFrame [16] and Rappture [27] provide similar solutions based on XML files for pre-configured layouts: EnginFrame for the EnginFrame server and Rappture for nanoHUB [172]. The advantage of the workflow extension of Rapid over EnginFrame and Rappture lies in the flexibility of the layout. Rapid supports XHTML and Rapid-specific tags for the generated portlets. The workflow extension of Rapid can be further developed for diverse workflow-enabled grid portals applying the JSR168/JSR286 standard.Therefore, these portals have to be examined regarding their interfaces for the workflow management. Thus, a more generic interface can be developed that supports different workflow-enabled grid portals.

The developments in this work are applied in the MoSGrid science gateway. The introduced specific portlets and workflows are especially developed for the MoSGrid community considering their feedback (see Chapter 9). The performance studies demonstrate that using DCIs is beneficial for the performance of molecular simulations. However, the portlets and workflows are re-usable on international level by the structural bioinformatics community. Partners in MoSGrid intend to address the international community by participating in the EU project SCI-BUS (Scientific gateway Based User Support) [256] and the EU project ER-flow (Building a European Research Community through Interoperable Workflows and Data) [257]. Furthermore, they are collaborating with the EU project EDGI (European Desktop Grid Initiative) [258]. Several communities from a number of application fields participate in SCI-BUS. Partners of SCI-BUS and ER-flow are also members of the VO Life Science that is one of the first established virtual user communities. The close collaboration between these communities allows the exchange of experiences, workflows, results, and molecular structures. Thus, they can benefit from each other without spending time on calculating the same molecular simulations or repeating unsuccessful simulations. The data is sensitive and the analysis of molecular data on a large scale is time-consuming and expensive. The output of a survey in the MoSGrid community shows that 70% [259] would share their results and molecular structures in a repository after they have published them or own a patent. We assume that researchers of the other communities would agree to share data in a repository at a similar rate.

The willingness to share tools and workflows is quite higher (nearly to 90%). Thus, the developers of services for the communities can also profit from the synergies. They can coordinate domain-related work as well as general developments for the science gateways. SCI-BUS focuses on further coordinated developments of science gateways and on offering portlet repositories, whereas ER-flow concentrates on workflow interoperability, workflows, and workflow repositories. EDGI brings in new technologies with desktop grids. The participation in and collaboration with these three projects allows developers of domain-related tools and workflows to concentrate on the specific demands, whereas general developments (e.g., security) can be completed by partners who are working on the infrastructure of science gateways. This leads to the provision of advanced services better tailored to the users' needs and supporting innovation and efficiency in the scientific discovery process.

The MoSGrid science gateway will be extended in SCI-BUS by a semantic search for workflows and simulation results. The visualisation of grid-computed results and the possibility of interaction with a 3D molecule editor based on WebGL [235] will support the users to process the full life-cycle of data analysis within the science gateway.

Currently, the MoSGrid community consists of more than 100 German users. However, the MoSGrid science gateways and its infrastructure with the integrated tools and workflows as well as generated results are valuable for the international community. The design of the science gateway is open for international use and the software will be provided as open-source software. SCI-BUS will advance a generic-purpose science gateway technology that will provide access to DCIs and their services in Europe. Additionally, life-sciences-related communities contribute to the project and partners of the communities will further develop these gateways. Several user community workshops will bring together users, developers, and providers of the science gateways. The collaboration with ER-flow and EDGI forges links to more related communities and opens up the possibility for the molecular simulation community to influence workflow interoperability and desktop grids on European level.

A

Abbreviations

A-WARE	An easy Way to Access grid REsources
AAA	Authentication, Authorisation, and Accounting
AIMD	Ab Initio Molecular Dynamics
AJAX	Asynchronous JavaScript and XML
AM1	Austin Model 1
Amazon EC2	Amazon Elastic Compute Cloud
Amazon S3	Amazon Simple Storage Service
AMBER	Assisted Model Building and Energy Refinement
API	Application Programming Interface
ASM	Application Specific Module
BES	Basic Execution Service
BFT	Basic File Transfer
BPEL	Business Process Execution Language
CA	Certification Authority
CADDSuite	Computer-aided Drug Design Suite
CAS	Central Authentication Service
CHARMM	Chemistry at HARvard Macromolecular Mechanics
CNDO	Complete Neglect of Differential Overlap
CRL	Certificate Revocation List
DAG	Directed Acyclic Graph
DCI	Distributed Computing Infrastructure
DFT	Density Functional Theory
DFN	Deutsches Forschungsnetz
DN	Distinguished Name
DOF	Degree of Freedom
DSD	Disorder of Sex Development

EBI	European Bioinformatics Institute
EDGI	European Desktop Grid Initiative
EGI	European Grid Infrastructure
ER-flow	Building an European Research Community through Interoperable Workflows and Data
ETD	Explicit Trust Delegation
EUGridPMA	European Grid Policy Management Authority
Gap-SLC	Gap Short Lived Credentials
GFS	Google File System
GPL	General Public License
GridKa-Ca	Grid Computing Centre Karlsruhe Certification Authority
GridSAM	Grid Job Submission and Monitoring Web Service
GSI	Grid Security Infrastructure
gUSE	grid User Support Environment
HF	Hartree-Fock
HiLA	High-level API
HPC	High-Performance Computing
I2MI	Internet2 Middleware Initiative
IDB	Incarnation DataBase
IdP	Identity Provider
IETF	The Internet Engineering Task Force
IGTF	International Grid Trust Federation
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JSDL	Job Submission Description Language
JSON	JavaScript Object Notation
JSP	JavaServer Page
JWS	Java Web Start
LCAO	Linear Combination of Atomic Orbitals
LDA	Local Density Approximation
LDAP	Lightweight Directory Access Protocol
LGA	Lamarckian Genetic Algorithm
MD	Molecular Dynamics
MNDO	Modified Neglect of Diatomic Overlap
MoSGrid	Molecular Simulation Grid
MPI	Message Passing Interface
MRC	Metadata and Replica Catalog
MSML	Molecular Simulation Markup Language
NDDO	Neglect of Diatomic Differential Overlap
NGI	National Grid Initiative
NIST	National Institute of Standards and Technology
NMR	Nuclear Magnetic Resonance
OASIS	Organization for the Advancement of Structured Information Standards
OGCE	Open Grid Computing Environments
OGE	Oracle Grid Engine
ONIOM	our Own N-layer Integrated molecular Orbital molecular Mechanics
OSD	Object Storage Device

PBS	Portable Batch System
PDB	Protein Data Bank
PDP	Policy Decision Point
PKI	Public Key Infrastructure
PM3	Parameterised Model Number 3
PMA	Policy Management Authority
PMF	Potential of Mean Force
QC	Quantum Chemistry
QM/MM	Quantum Mechanical/Molecular Mechanical
RA	Registration Authority
RAPPTURE	Rapid APPlication InfrastrucTURE
RBAC	Role-based Access Protocol
SA	Simulated Annealing
SAML	Security Assertion Markup Language
SAS	Solvent-Accessible Surfaces
SES	Solvent-Excluded Surfaces
SCF	Self-Consistent Field
SCI-BUS	SCientific gateway Based User Support
SCUFL	Simple Conceptual Unified Flow Language
SHIWA	SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs
SMD	Steered Molecular Dynamics
SOAP	Simple Object Access Protocol
SP	Service Provider
SRB	Storage Resource Broker
SRM	Storage Resource Manager
SSL	Secure Socket Layer
STO	Slater-Type Orbitals
SWIF	Scientific Workflow Interoperability Framework
TOPP	The OpenMS Proteomics Pipeline
TSI	Target System Interface
UCC	UNICORE Commandline Client
URC	UNICORE Rich Client
UML	Unified Modeling Language
UNICORE	Uniform Interface to Computing Resources
VNC	Virtual Networking Computing
VO	Virtual Organisation
WeNMR	A worldwide e-Infrastructure for NMR and structural biology
WS-PGRADE	Web Services Parallel Grid Runtime and Developer Environment
WSRF	Web Services Resource Framework
WSRP	Web Services for Remote Portlets
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
YAWL	Yet Another Workflow Language
ZDO	Zero-Differential Overlap

B Appendix

Unified Modeling Language

We use the UML [201] in this work to illustrate the object-oriented design of the extended or implemented components. Due to clarity, the figures in this work include only the extended or implemented components. We omit dependencies to components which remained the same in the extended science gateways and tools, except for the interfaces provided.

In 1997, UML [260] was developed from object-oriented methods, modeling language design, and architectural description languages to represent software models. Its goal is to provide tools for analysis, design, and implementation of object-oriented software. We have visualised extended or newly implemented classes for science gateways and tools via UML class diagrams. In these diagrams, classes are represented by rectangles, which include the name of the correponding class and may contain attributes and operations. To illustrate different types of attributes and operations, they are preceded with a single character (”+” relates to public, ”#” relates to protected, and ”-” relates to private). Relationships between classes are represented by lines or arrows. In this work, two different kinds of relationships are illustrated; the dependency between two classes and the realisation of a specification. The latter usually designates the implementation of an interface. Whereas dependencies are drawn as dashed lines with an arrow, realisations are represented by a dashed line with a hollow triangle at the end of the specification. A comment is shown as a rectangle with the upper right corner bent. It is connected via a dashed line with the associated class. The figure below shows an example for a class `ClassA` which implements an interface `InterfaceA` and uses methods of the class `ClassB`. Additionally, comments are added to `ClassB`.

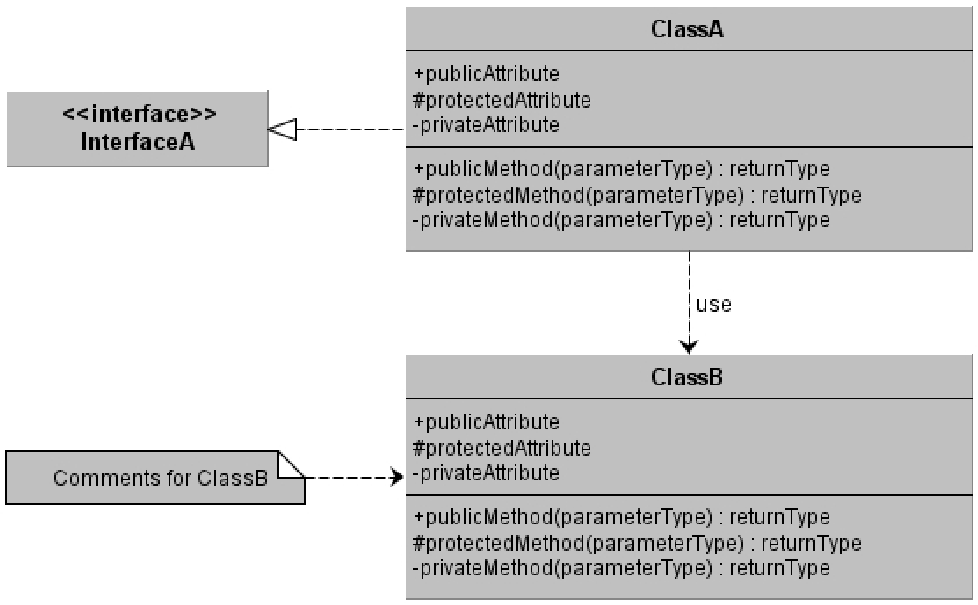

ClassA
+publicAttribute
#protectedAttribute
-privateAttribute
+publicMethod(parameterType) : returnType
#protectedMethod(parameterType) : returnType
-privateMethod(parameterType) : returnType
<<interface>>
InterfaceA
use
ClassB
+publicAttribute
#protectedAttribute
-privateAttribute
+publicMethod(parameterType) : returnType
#protectedMethod(parameterType) : returnType
-privateMethod(parameterType) : returnType
Comments for ClassB

Figure B.1: Example for an illustration of two associated classes in UML

Model-View-Controller

The MVC [261] design pattern separates the visualisation of a program from its logic and the respective data. The science gateways and tools, that we have extended or developed, are following the MVC pattern. To illustrate the involved implemented files and their purpose, we included figures with the definition whether the specified files belong to the model, the view, or the controller. The illustration is adapted from the UML [201] representation of design patterns. The dashed ellipes contain the name of the pattern and the dashed lines connect the involved source files listed in rectangles. The lines are labeled with ”Model”, ”View”, or ”Controller”. The figure below shows a basic example. The controller as well as the model contain one single file, whereas the view includes a Java file, a JSP file, and a HTML file.

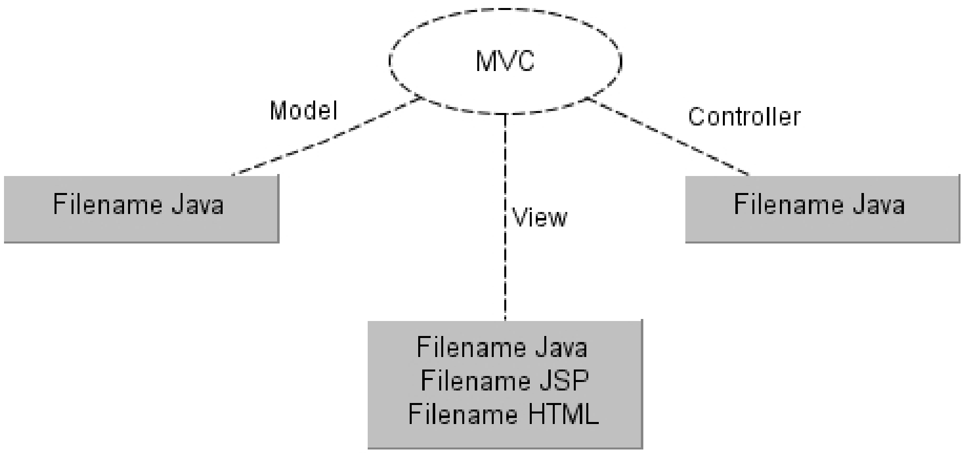
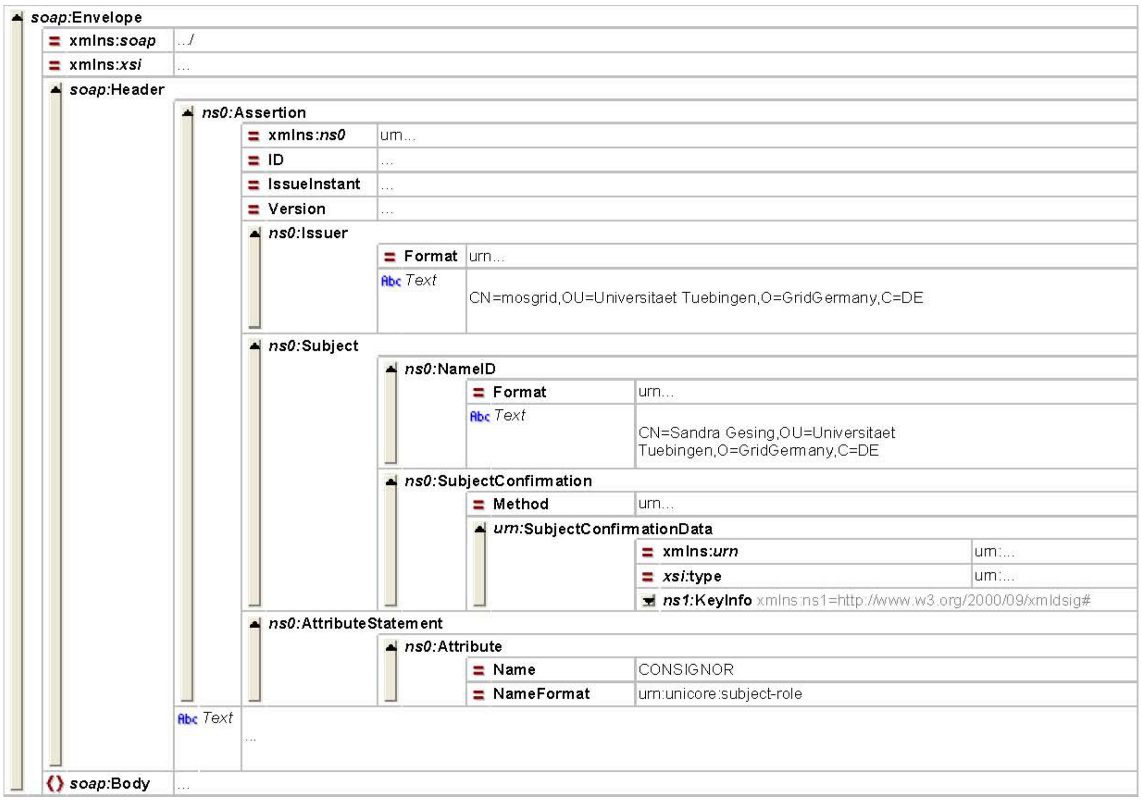


Figure B.2: Example for an illustration of the MVC pattern

SAML Assertions

Here are examples provided for SAML assertion files used in the submission process of jobs in UNICORE 6.

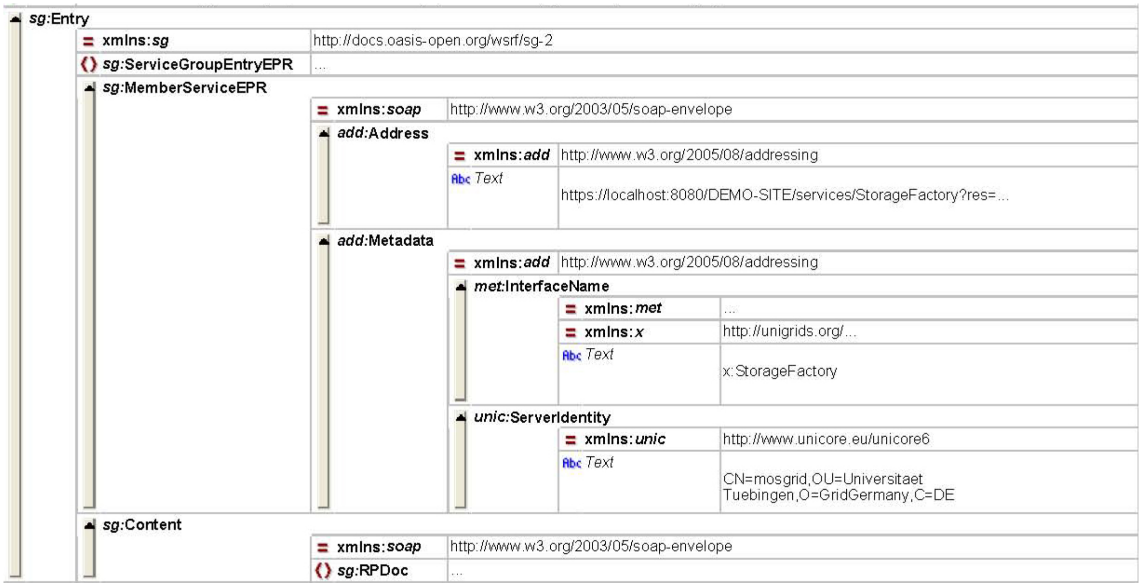
Figure B.3 illustrates a SAML consignor assertion issued by a gateway and forwarded to the registry of the target site. It uses SOAP for exchanging information between resources and contains a header part and a body part. The header defines the metadata about the involved resources whereas the body part contains the identifier of the targeted resource. The assertion is included in the header. It is characterised by the information on the gateway in `<ns0:Issuer>`, by the properties of the user in `<ns0:Subject>`, and by the type of assertion in `<ns0:AttributeStatement>`. The properties of the users inlcude the DN of the user and the X.509 certificate.



```
soap:Envelope
  = xmlns:soap   .../
  = xmlns:xsi    ...
  soap:Header
    ns0:Assertion
      = xmlns:ns0      urn...
      = ID             ...
      = IssueInstant   ...
      = Version        ...
      ns0:Issuer
        = Format   urn...
        Abc Text   CN=mosgrid,OU=Universitaet Tuebingen,O=GridGermany,C=DE
      ns0:Subject
        ns0:NameID
          = Format   urn...
          Abc Text   CN=Sandra Gesing,OU=Universitaet
                     Tuebingen,O=GridGermany,C=DE
        ns0:SubjectConfirmation
          = Method   urn...
          urn:SubjectConfirmationData
            = xmlns:urn   urn:...
            = xsi:type    urn:...
            ns1:KeyInfo xmlns:ns1=http://www.w3.org/2000/09/xmldsig#
      ns0:AttributeStatement
        ns0:Attribute
          = Name         CONSIGNOR
          = NameFormat   urn:unicore:subject-role
    Abc Text
      ...
  soap:Body   ...
```

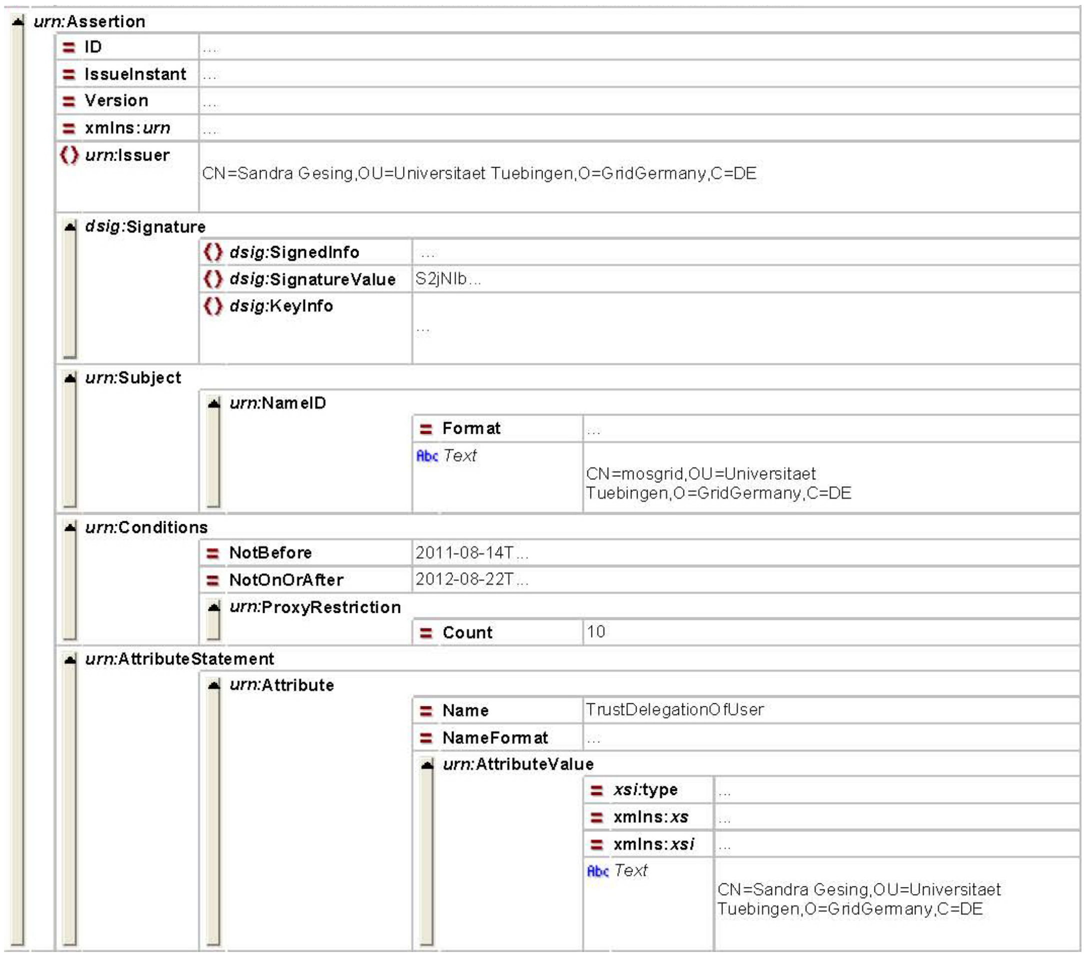
Figure B.3: XML illustration of an example for a SAML consignor assertion

See Figure B.4 for an example of a registry entry which is provided by the UNICORE/X server to the service registry. `<sg:MemberServiceEPS>` contains the characteristics of the service. The service address is included in `<add:Address>`, the service type in `<met:Interface>`> and the identity of the UNICORE/X server in `<unic:ServerIdentity>`.



```
sg:Entry
    xmlns:sg                      http://docs.oasis-open.org/wsrf/sg-2
    sg:ServiceGroupEntryEPR       ...
    sg:MemberServiceEPR
        xmlns:soap                http://www.w3.org/2003/05/soap-envelope
        add:Address
            xmlns:add             http://www.w3.org/2005/08/addressing
            Text                  https://localhost:8080/DEMO-SITE/services/StorageFactory?res=...
        add:Metadata
            xmlns:add             http://www.w3.org/2005/08/addressing
            met:InterfaceName
                xmlns:met         ...
                xmlns:x           http://unigrids.org/...
                Text              x:StorageFactory
            unic:ServerIdentity
                xmlns:unic        http://www.unicore.eu/unicore6
                Text              CN=mosgrid,OU=Universitaet
                                  Tuebingen,O=GridGermany,C=DE
    sg:Content
        xmlns:soap                http://www.w3.org/2003/05/soap-envelope
        sg:RPDoc                  ...
```

Figure B.4: XML illustration of an example for a registry entry



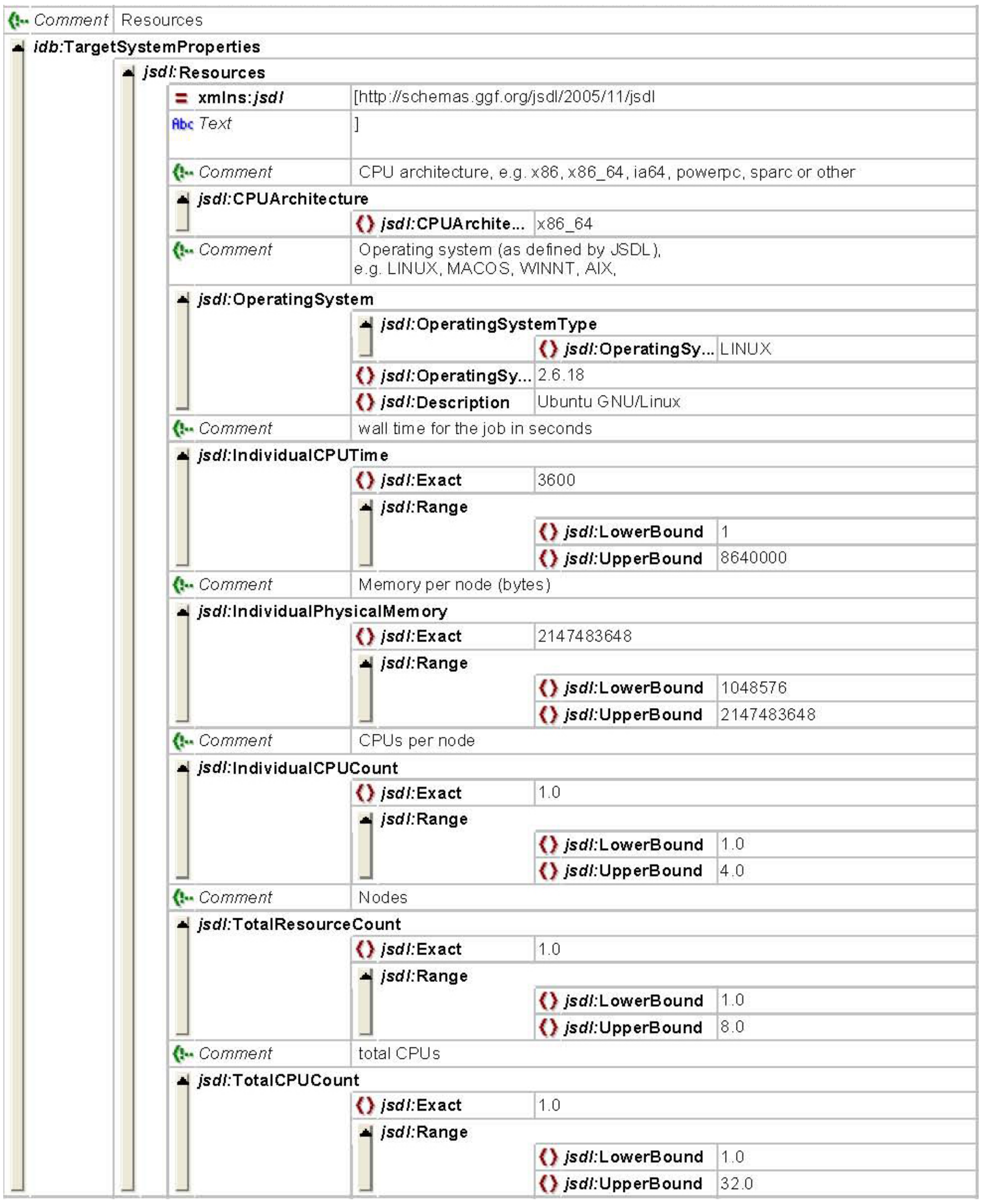
```
urn:Assertion
    ID                            ...
    IssueInstant                  ...
    Version                       ...
    xmlns:urn                     ...
    urn:Issuer                    CN=Sandra Gesing,OU=Universitaet Tuebingen,O=GridGermany,C=DE
    dsig:Signature
        dsig:SignedInfo           ...
        dsig:SignatureValue       S2jNIb...
        dsig:KeyInfo              ...
    urn:Subject
        urn:NameID
            Format                ...
            Text                  CN=mosgrid,OU=Universitaet
                                  Tuebingen,O=GridGermany,C=DE
    urn:Conditions
        NotBefore                 2011-08-14T...
        NotOnOrAfter              2012-08-22T...
        urn:ProxyRestriction
            Count                 10
    urn:AttributeStatement
        urn:Attribute
            Name                  TrustDelegationOfUser
            NameFormat            ...
            urn:AttributeValue
                xsi:type          ...
                xmlns:xs          ...
                xmlns:xsi         ...
                Text              CN=Sandra Gesing,OU=Universitaet
                                  Tuebingen,O=GridGermany,C=DE
```

Figure B.5: XML illustration of an example for a SAML assertion

SAML allows to apply various security characteristics specifying the assertion more detailed. Figure B.5 illustrates a SAML assertion file for trust delegation of a user which is signed according to the XML digital signature specifications. The public part of the signing certificate is included in `<dsig:Signature>` and thus the security token can be validated. The DN of the issuer as well as the DN of the subject of the trust delegation are defined in `<urn:Issuer>` or `<urn:Subject>`, respectively. In `<urn:Conditions>`, the settings for the validity of an assertion file can be configured (e.g., the duration of validity). Finally, `<urn:AttrributeStatement>` defines the type of a SAML assertion.

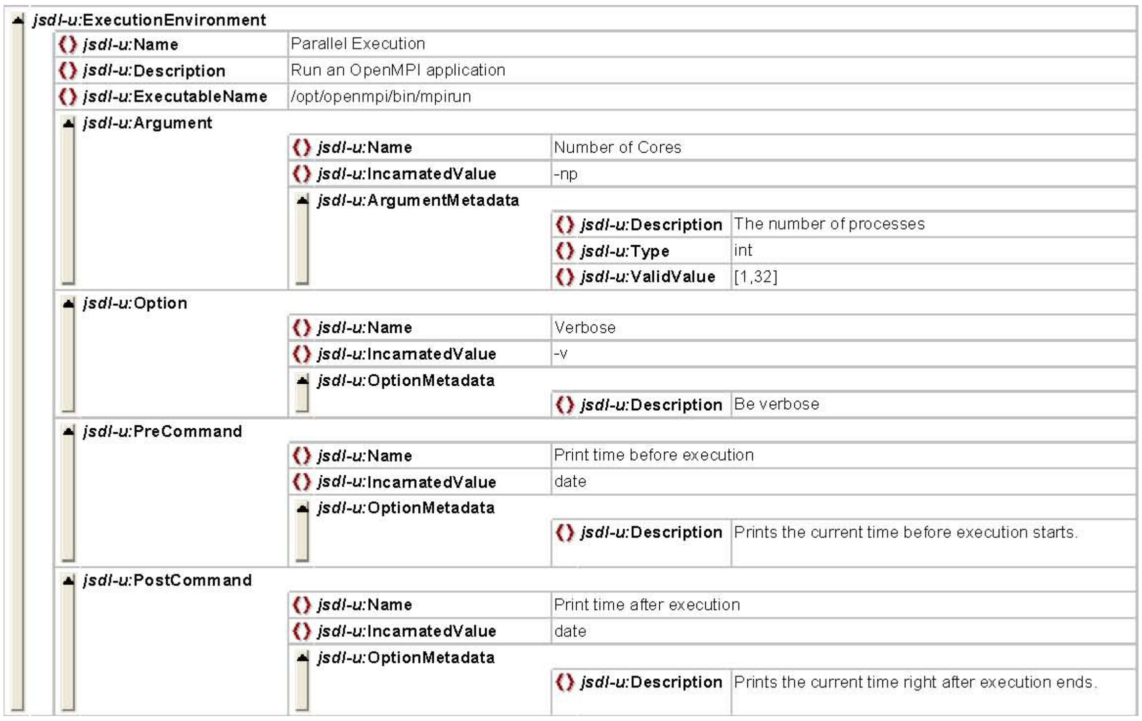
Incarnation Database (IDB)

UNICORE offers the possiblity to map abstract job definitions to real executables via the IDB. It allows defining specifications for resources, the execution environment, and tools. The corresponding XML schema as well as further explanations are provided at [262]. Here are examples listed which are configured in D-Grid facilities. Values for resources (e.g., architecture, number of CPUs), for execution environments (e.g., number of cores, precommands), and for tools (e.g., name, version, parameters, interactive parameters) can be provided.



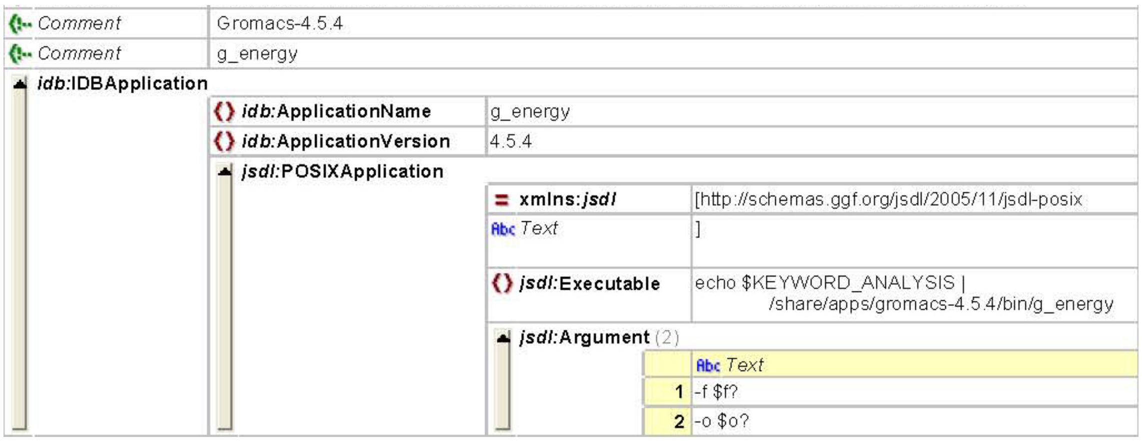
```
Comment  Resources
idb:TargetSystemProperties
    jsdl:Resources
        xmlns:jsdl  [http://schemas.ggf.org/jsdl/2005/11/jsdl
        Text        ]
        Comment     CPU architecture, e.g. x86, x86_64, ia64, powerpc, sparc or other
        jsdl:CPUArchitecture
            jsdl:CPUArchite...  x86_64
        Comment     Operating system (as defined by JSDL),
                    e.g. LINUX, MACOS, WINNT, AIX,
        jsdl:OperatingSystem
            jsdl:OperatingSystemType
                jsdl:OperatingSy...  LINUX
            jsdl:OperatingSy...  2.6.18
            jsdl:Description     Ubuntu GNU/Linux
        Comment     wall time for the job in seconds
        jsdl:IndividualCPUTime
            jsdl:Exact  3600
            jsdl:Range
                jsdl:LowerBound  1
                jsdl:UpperBound  8640000
        Comment     Memory per node (bytes)
        jsdl:IndividualPhysicalMemory
            jsdl:Exact  2147483648
            jsdl:Range
                jsdl:LowerBound  1048576
                jsdl:UpperBound  2147483648
        Comment     CPUs per node
        jsdl:IndividualCPUCount
            jsdl:Exact  1.0
            jsdl:Range
                jsdl:LowerBound  1.0
                jsdl:UpperBound  4.0
        Comment     Nodes
        jsdl:TotalResourceCount
            jsdl:Exact  1.0
            jsdl:Range
                jsdl:LowerBound  1.0
                jsdl:UpperBound  8.0
        Comment     total CPUs
        jsdl:TotalCPUCount
            jsdl:Exact  1.0
            jsdl:Range
                jsdl:LowerBound  1.0
                jsdl:UpperBound  32.0
```

Figure B.6: XML illustration of an example for the configuration of a target system in the IDB



```
jsdl-u:ExecutionEnvironment
    jsdl-u:Name                      Parallel Execution
    jsdl-u:Description               Run an OpenMPI application
    jsdl-u:ExecutableName            /opt/openmpi/bin/mpirun
    jsdl-u:Argument
        jsdl-u:Name                  Number of Cores
        jsdl-u:IncarnatedValue       -np
        jsdl-u:ArgumentMetadata
            jsdl-u:Description       The number of processes
            jsdl-u:Type              int
            jsdl-u:ValidValue        [1,32]
    jsdl-u:Option
        jsdl-u:Name                  Verbose
        jsdl-u:IncarnatedValue       -v
        jsdl-u:OptionMetadata
            jsdl-u:Description       Be verbose
    jsdl-u:PreCommand
        jsdl-u:Name                  Print time before execution
        jsdl-u:IncarnatedValue       date
        jsdl-u:OptionMetadata
            jsdl-u:Description       Prints the current time before execution starts.
    jsdl-u:PostCommand
        jsdl-u:Name                  Print time after execution
        jsdl-u:IncarnatedValue       date
        jsdl-u:OptionMetadata
            jsdl-u:Description       Prints the current time right after execution ends.
```

Figure B.7: XML illustration of an example for a configuration of a JSDL execution environment



```
Comment                              Gromacs-4.5.4
Comment                              g_energy
idb:IDBApplication
    idb:ApplicationName              g_energy
    idb:ApplicationVersion           4.5.4
    jsdl:POSIXApplication
        xmlns:jsdl                   [http://schemas.ggf.org/jsdl/2005/11/jsdl-posix
        Text                         ]
        jsdl:Executable              echo $KEYWORD_ANALYSIS |
                                         /share/apps/gromacs-4.5.4/bin/g_energy
        jsdl:Argument (2)
                                     Text
                                  1  -f $f?
                                  2  -o $o?
```

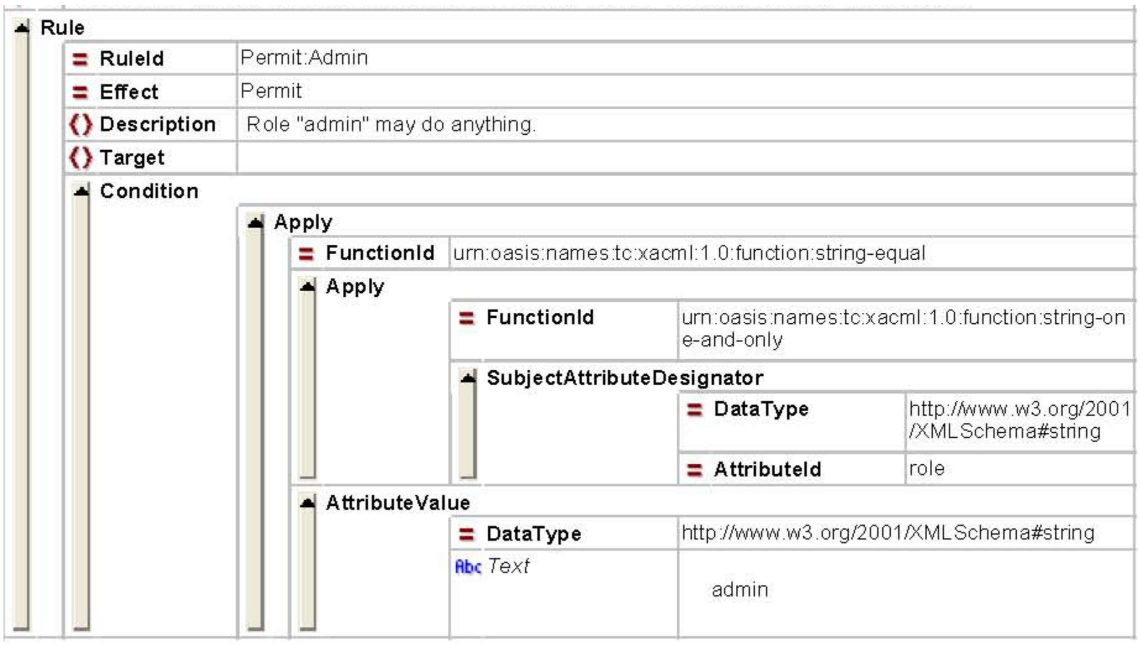
Figure B.8: XML illustration of an example for a configuration of an IDB application

XACML (eXtensible Access Control Markup Language)

An XACML entity is part of each UNICORE/X server and forms a Policy Decision Point (PDP), which communicates with the XUUDB. UNICORE supports different XACML versions (1.x and 2.x) [263] and the here given examples illustrate 2.x versions. For invoking a job or a workflow, a request is generated with following attributes.

- the UNICORE service which is targeted (e.g. job management service)
- the name of the action being invoked
- the DN of the user initiating the request
- the role of the user as defined in XUUDB

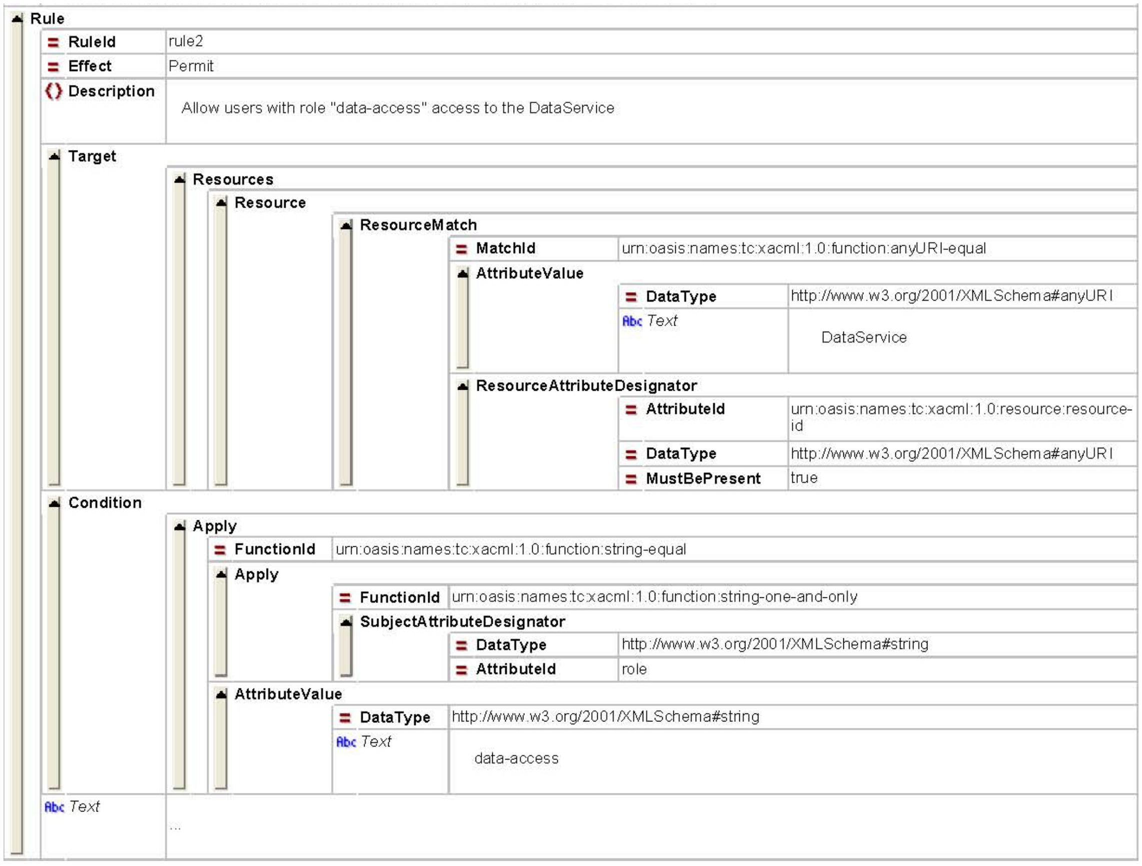
The UNICORE/X server processes the request and checks it against a policy file or a set of policy files. Policy files constitute rules for a request and support to dynamically add policies to the processed authorisation. XACML supports role-based access to services and the following example demonstrates the rule that the role admin allows access to all services.



```
Rule
    RuleId          Permit:Admin
    Effect          Permit
    Description     Role "admin" may do anything.
    Target
    Condition
                    Apply
                        FunctionId   urn:oasis:names:tc:xacml:1.0:function:string-equal
                        Apply
                                     FunctionId   urn:oasis:names:tc:xacml:1.0:function:string-on
                                                  e-and-only
                                     SubjectAttributeDesignator
                                                  DataType      http://www.w3.org/2001
                                                                /XMLSchema#string
                                                  AttributeId   role
                        AttributeValue
                                     DataType     http://www.w3.org/2001/XMLSchema#string
                                     Text
                                                    admin
```

Figure B.9: XML illustration of an example for a rule for the role admin

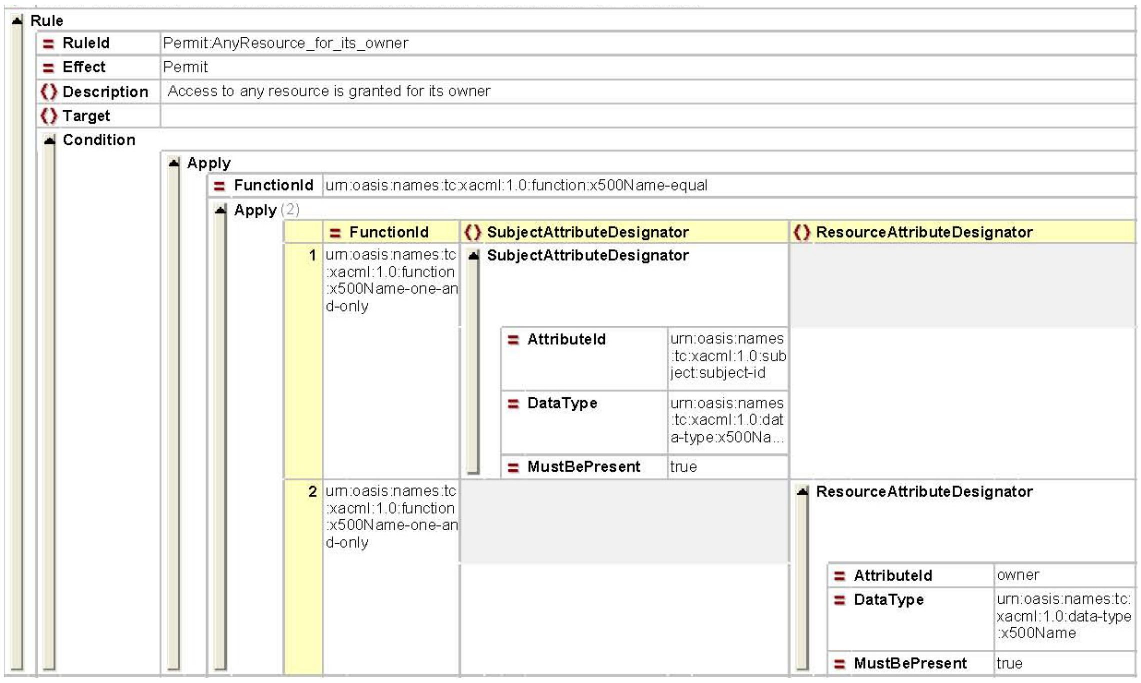
For limiting access to a certain service, the tag <Target> identifies the targeted service and the tag <Condition> under which conditions the access is granted.The following example illustrate that the service DataService allows access for users with the role data-access.



```
Rule
  RuleId        rule2
  Effect        Permit
  Description   Allow users with role "data-access" access to the DataService
  Target
    Resources
      Resource
        ResourceMatch
          MatchId          urn:oasis:names:tc:xacml:1.0:function:anyURI-equal
          AttributeValue
            DataType       http://www.w3.org/2001/XMLSchema#anyURI
            Text           DataService
          ResourceAttributeDesignator
            AttributeId    urn:oasis:names:tc:xacml:1.0:resource:resource-id
            DataType       http://www.w3.org/2001/XMLSchema#anyURI
            MustBePresent  true
  Condition
    Apply
      FunctionId    urn:oasis:names:tc:xacml:1.0:function:string-equal
      Apply
        FunctionId  urn:oasis:names:tc:xacml:1.0:function:string-one-and-only
        SubjectAttributeDesignator
          DataType     http://www.w3.org/2001/XMLSchema#string
          AttributeId  role
      AttributeValue
        DataType    http://www.w3.org/2001/XMLSchema#string
        Text        data-access
  Text          ...
```

Figure B.10: XML illustration of an example for a condition for users with the role data-access

Access to service instances like jobs or the USpace is expressed via conditions. The following sets access only to the owner of the service instance.



Rule

RuleId	Permit:AnyResource_for_its_owner
Effect	Permit
Description	Access to any resource is granted for its owner
Target	

Condition

Apply

FunctionId	urn:oasis:names:tc:xacml:1.0:function:x500Name-equal

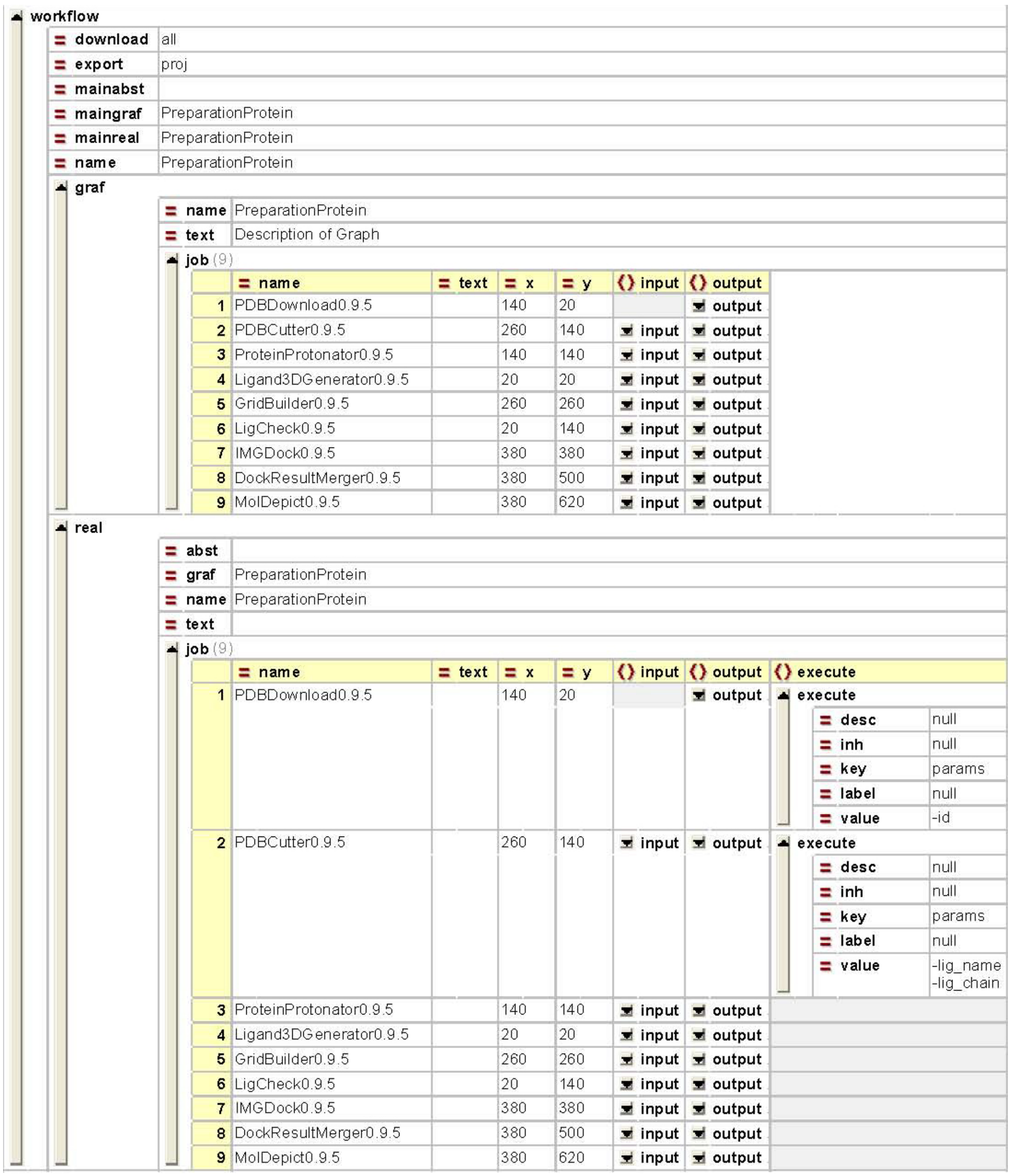
Apply (2)

	FunctionId	SubjectAttributeDesignator	ResourceAttributeDesignator
1	urn:oasis:names:tc:xacml:1.0:function:x500Name-one-and-only	SubjectAttributeDesignator: AttributeId urn:oasis:names:tc:xacml:1.0:subject:subject-id; DataType urn:oasis:names:tc:xacml:1.0:data-type:x500Na...; MustBePresent true	
2	urn:oasis:names:tc:xacml:1.0:function:x500Name-one-and-only		ResourceAttributeDesignator: AttributeId owner; DataType urn:oasis:names:tc:xacml:1.0:data-type:x500Name; MustBePresent true

Figure B.11: XML illustration of an example for a rule for the owner of a service instance

WS-PGRADE Workflows

The overall structure of WS-PGRADE workflows is described in Section 7.2.1. The following defines the example used for the illustration of the migration of workflows (see Section 7). The dependencies of the jobs result out of the Galaxy workflow, the names of the jobs are generated analogously to IDB tool names, and the coordinates are calculated via the graph algorithm (see Section 7.4).



workflow

download	all
export	proj
mainabst	
maingraf	PreparationProtein
mainreal	PreparationProtein
name	PreparationProtein

graf

name	PreparationProtein
text	Description of Graph

job (9)

	name	text	x	y	input	output
1	PDBDownload0.9.5		140	20		output
2	PDBCutter0.9.5		260	140	input	output
3	ProteinProtonator0.9.5		140	140	input	output
4	Ligand3DGenerator0.9.5		20	20	input	output
5	GridBuilder0.9.5		260	260	input	output
6	LigCheck0.9.5		20	140	input	output
7	IMGDock0.9.5		380	380	input	output
8	DockResultMerger0.9.5		380	500	input	output
9	MolDepict0.9.5		380	620	input	output

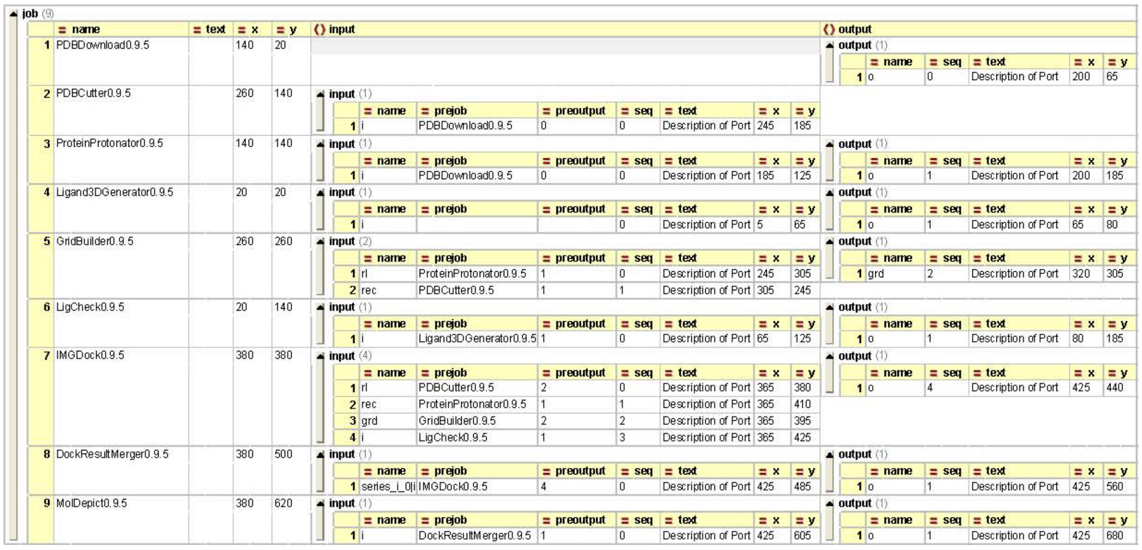
real

abst	
graf	PreparationProtein
name	PreparationProtein
text	

job (9)

	name	text	x	y	input	output	execute
1	PDBDownload0.9.5		140	20		output	execute desc null inh null key params label null value -id
2	PDBCutter0.9.5		260	140	input	output	execute desc null inh null key params label null value -lig_name -lig_chain
3	ProteinProtonator0.9.5		140	140	input	output	
4	Ligand3DGenerator0.9.5		20	20	input	output	
5	GridBuilder0.9.5		260	260	input	output	
6	LigCheck0.9.5		20	140	input	output	
7	IMGDock0.9.5		380	380	input	output	
8	DockResultMerger0.9.5		380	500	input	output	
9	MolDepict0.9.5		380	620	input	output	

Figure B.12: XML illustration of an example for a WS-PGRADE workflow



job (9)

	name	text	x	y
1	PDBDownload0.9.5		140	20
2	PDBCutter0.9.5		260	140
3	ProteinProtonator0.9.5		140	140
4	Ligand3DGenerator0.9.5		20	20
5	GridBuilder0.9.5		260	260
6	LigCheck0.9.5		20	140
7	IMGDock0.9.5		380	380
8	DockResultMerger0.9.5		380	500
9	MolDepict0.9.5		380	620

input

		name	prejob	preoutput	seq	text	x	y
2 input (1)	1	i	PDBDownload0.9.5	0	0	Description of Port	245	185
3 input (1)	1	i	PDBDownload0.9.5	0	0	Description of Port	185	125
4 input (1)	1	i			0	Description of Port	5	65
5 input (2)	1	rl	ProteinProtonator0.9.5	1	0	Description of Port	245	305
	2	rec	PDBCutter0.9.5	1	1	Description of Port	305	245
6 input (1)	1	i	Ligand3DGenerator0.9.5	1	0	Description of Port	65	125
7 input (4)	1	rl	PDBCutter0.9.5	2	0	Description of Port	365	380
	2	rec	ProteinProtonator0.9.5	1	1	Description of Port	365	410
	3	grd	GridBuilder0.9.5	2	2	Description of Port	365	395
	4	i	LigCheck0.9.5	1	3	Description of Port	365	425
8 input (1)	1	series_i_0|i	IMGDock0.9.5	4	0	Description of Port	425	485
9 input (1)	1	i	DockResultMerger0.9.5	1	0	Description of Port	425	605

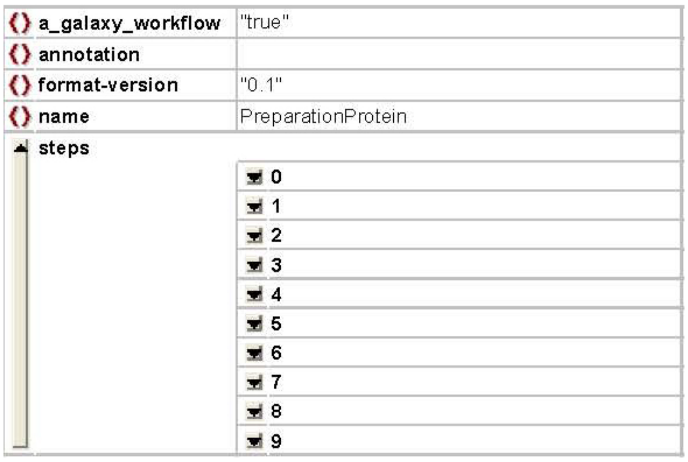
output

		name	seq	text	x	y
1 output (1)	1	o	0	Description of Port	200	65
3 output (1)	1	o	1	Description of Port	200	185
4 output (1)	1	o	1	Description of Port	65	80
5 output (1)	1	grd	2	Description of Port	320	305
6 output (1)	1	o	1	Description of Port	80	185
7 output (1)	1	o	4	Description of Port	425	440
8 output (1)	1	o	1	Description of Port	425	560
9 output (1)	1	o	1	Description of Port	425	680

Figure B.13: XML illustration of an example for jobs of a WS-PGRADE workflow

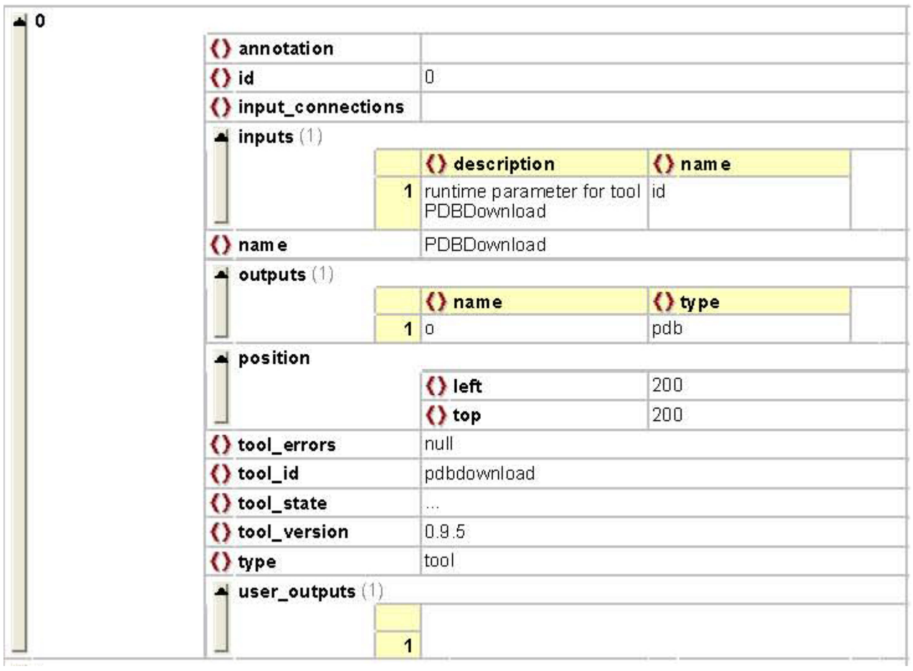
Galaxy Workflows

The overall structure of Galaxy workflows is described in Section 7.2.2. The following defines the example used for the illustration of the migration of workflows (see Section 7).



a_galaxy_workflow	"true"
annotation	
format-version	"0.1"
name	PreparationProtein
steps	0 1 2 3 4 5 6 7 8 9

Figure B.14: JSON illustration of an example for a Galaxy workflow



0

annotation	
id	0
input_connections	
inputs (1)	
name	PDBDownload
outputs (1)	
position	
tool_errors	null
tool_id	pdbdownload
tool_state	...
tool_version	0.9.5
type	tool
user_outputs (1)	

inputs (1)

	description	name
1	runtime parameter for tool PDBDownload	id

outputs (1)

	name	type
1	o	pdb

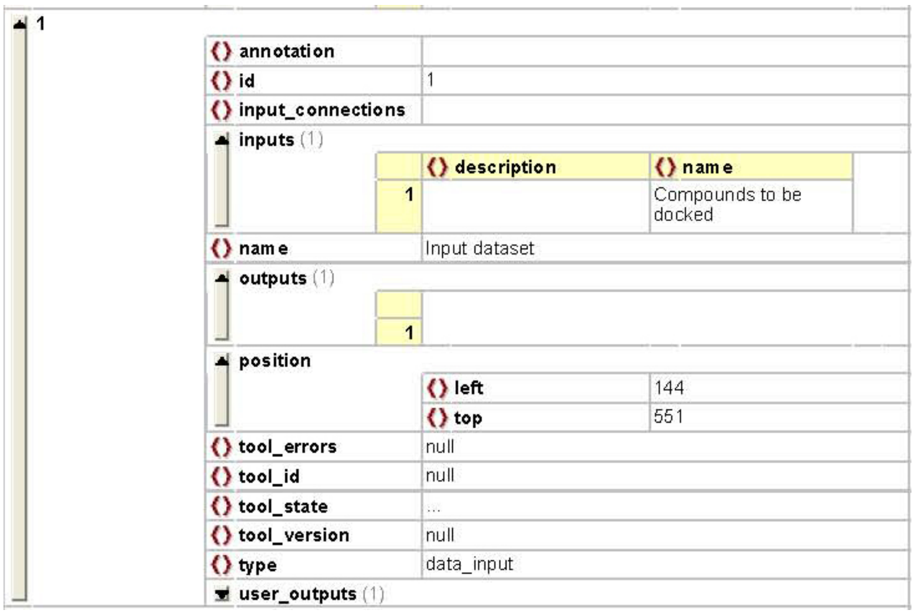
position

left	200
top	200

user_outputs (1)

1

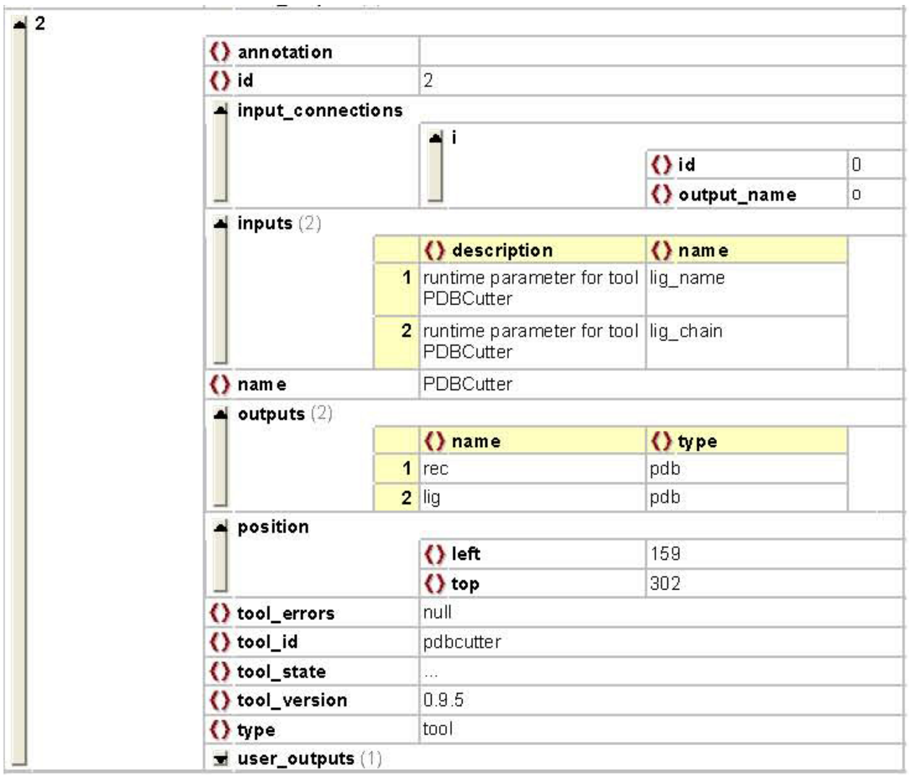
Figure B.15: JSON illustration of an example for a parent step of a Galaxy workflow



```
1
    annotation
    id                  1
    input_connections
    inputs (1)
                        description     name
                    1                   Compounds to be
                                        docked
    name                Input dataset
    outputs (1)

                    1
    position
                        left            144
                        top             551
    tool_errors         null
    tool_id             null
    tool_state          ...
    tool_version        null
    type                data_input
    user_outputs (1)
```

Figure B.16: JSON illustration of an example for a data input step of a Galaxy workflow

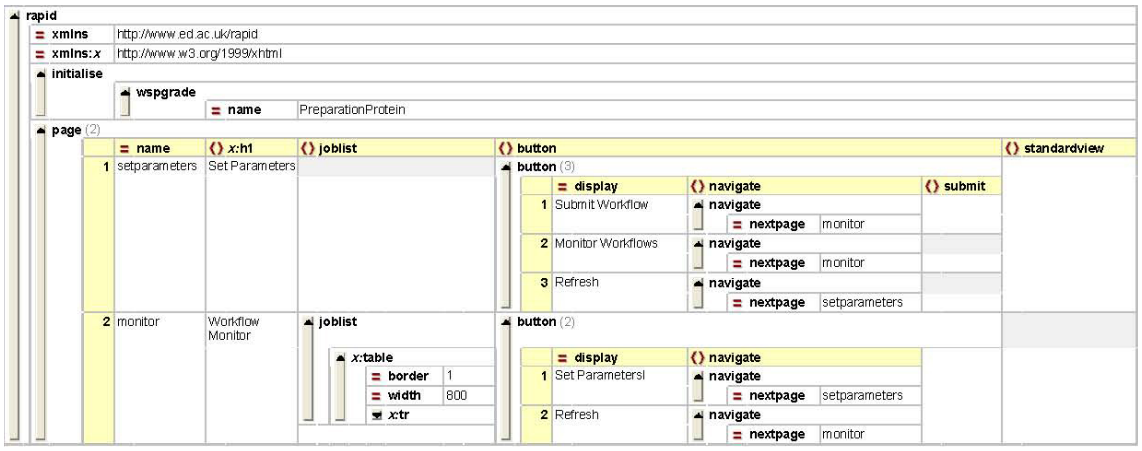


```
2
    annotation
    id                  2
    input_connections
                        i
                                        id              0
                                        output_name     o
    inputs (2)
                        description             name
                    1   runtime parameter for tool   lig_name
                        PDBCutter
                    2   runtime parameter for tool   lig_chain
                        PDBCutter
    name                PDBCutter
    outputs (2)
                        name            type
                    1   rec             pdb
                    2   lig             pdb
    position
                        left            159
                        top             302
    tool_errors         null
    tool_id             pdbcutter
    tool_state          ...
    tool_version        0.9.5
    type                tool
    user_outputs (1)
```

Figure B.17: JSON illustration of an example for a child step of a Galaxy workflow

Rapid Files

Rapid supports to create JSR168/JSR286-compliant portlets via defining an XML and optionally via addtional plugins implemented in Jython. All available Rapid options except for the extension for WS-PGRADE are provided in [227]. The following example reflects the integration of a WS-PGRADE workflow.



```
rapid
  xmlns      http://www.ed.ac.uk/rapid
  xmlns:x    http://www.w3.org/1999/xhtml
  initialise
      wspgrade
          name    PreparationProtein
  page (2)
         name           x:h1            joblist          button                                                     standardview
      1  setparameters  Set Parameters                   button (3)
                                                                display            navigate                 submit
                                                              1 Submit Workflow    navigate
                                                                                     nextpage  monitor
                                                              2 Monitor Workflows  navigate
                                                                                     nextpage  monitor
                                                              3 Refresh            navigate
                                                                                     nextpage  setparameters
      2  monitor        Workflow        joblist          button (2)
                        Monitor
                                          x:table               display            navigate
                                            border  1         1 Set ParametersI    navigate
                                            width   800                              nextpage  setparameters
                                            x:tr              2 Refresh            navigate
                                                                                     nextpage  monitor
```

Figure B.18: XML illustration of an example for an integration of a WS-PGRADE workflow in Rapid

Lists of Source Code Files

The developments in this work have been integrated into the software frameworks WS-PGRADE [186], gUSE [186], Galaxy [264], and Rapid [227]. The following lists illustrate the classes, which have been added or extended in these frameworks.

List for **Section 5 ”A Novel Credential Management for Science Gateways”**

The credential portlet

```
hu/sztaki/lpds/pgportal/services/credential/SZGCredential.java
hu/sztaki/lpds/pgportal/services/credential/SZGCredentialException.java
hu/sztaki/lpds/pgportal/services/credential/SZGCredentialManager.java
hu/sztaki/lpds/pgportal/services/credential/SZGCredentialMemoryStore.java
hu/sztaki/lpds/pgportal/services/credential/SZGCredentialStorage.java
hu/sztaki/lpds/pgportal/services/credential/SZGCredentialStorageFactory.java
hu/sztaki/lpds/pgportal/services/credential/SZGStoreKey.java
```

ETDApplet

```
de/fzj/unicore/security/etd/ETDApplet.java
de/fzj/unicore/security/etd/KeyStoreLoader.java
de/fzj/unicore/security/etd/SecurityProperties.java
de/fzj/unicore/security/etd/WebParam.java
```

List for **Section 6 ”Enhanced Job and Workflow Mangement”**

UNICORE submitter

```
hu/sztaki/lpds/submitter/grids/Grid_unicore.java
hu/sztaki/lpds/submitter/grids/unicore/JSDLDocBuilder.java
hu/sztaki/lpds/submitter/grids/unicore/UNICOREManager.java
```

UNICORE plugin

```
hu/sztaki/lpds/pgportal/servlet/ajaxactions/GetUnicoreIDBTools.java
hu/sztaki/lpds/pgportal/servlet/ajaxactions/grids/JobConfigUI_unicore.java
hu/sztaki/lpds/pgportal/servlet/ajaxactions/grids/JobConfigUtils.java
hu/sztaki/lpds/pgportal/util/resource/ResourceXMLHandler.java
hu/sztaki/lpds/pgportal/util/resource/UNICOREIDBToolHandler.java
```

Application Specific Module

```
hu/sztaki/lpds/pgportal/services/asm/ASM_Instance.java
hu/sztaki/lpds/pgportal/services/asm/ASM_job.java
hu/sztaki/lpds/pgportal/services/asm/ASMInputPort.java
hu/sztaki/lpds/pgportal/services/asm/ASMManager.java
hu/sztaki/lpds/pgportal/services/asm/ASMObjectBase.java
hu/sztaki/lpds/pgportal/services/asm/ASMOutputPort.java
hu/sztaki/lpds/pgportal/services/asm/ASMPort.java
hu/sztaki/lpds/pgportal/services/asm/ASMRepositoryItemBean.java
hu/sztaki/lpds/pgportal/services/asm/ASMRepositoryItemType.java
hu/sztaki/lpds/pgportal/services/asm/ASMService.java
hu/sztaki/lpds/pgportal/services/asm/ASMUploadThread.java
hu/sztaki/lpds/pgportal/services/asm/EPortalClientInformation.java
hu/sztaki/lpds/pgportal/services/asm/EStatusConstants.java
hu/sztaki/lpds/pgportal/services/asm/InstanceStatusBean.java
hu/sztaki/lpds/pgportal/services/asm/PortalClientInformation.java
hu/sztaki/lpds/pgportal/services/asm/StatusColors.java
hu/sztaki/lpds/pgportal/services/asm/StatusConstants.java
hu/sztaki/lpds/pgportal/services/asm/exceptions/ASMException.java
```

List for **Section 7 "A Novel Method for Workflow Migration"**

The extension of Galaxy for the migration of WS-PGRADE workflows

```
galaxy/lib/galaxy/web/controllers/workflow.py
galaxy/template/workflow/export.mako
galaxy/template/workflow/wspgrade_download.mako
galaxy/template/workflow/wspgrade_download_content.mako
```

List for **Section 8 "Automatic Creation of Portlets for Workflows"**

The extension of Rapid for WS-PGRADE workflows

```
uk/ac/ed/rapid/jsp/output/Output.java
uk/ac/ed/rapid/jobsubmission/jobmanager/workflow/AbstractWorkflowFactory.java
uk/ac/ed/rapid/jobsubmission/jobmanager/workflow/WorkflowManager.java
uk/ac/ed/rapid/jobsubmission/jobmanager/workflow/WorkflowPlugin.java
uk/ac/ed/rapid/jobsubmission/jobmanager/workflow/WorkflowProperties.java
uk/ac/ed/rapid/jobsubmission/jobmanager/workflow/WorkflowState.java
```

C Contributions

Chapter 1 - Introduction
Small parts of this chapter have been previously published in the manuscript [255].

Chapter 2 - Theoretical Background
Small parts of this chapter have been previously published as joint work with Jens Krüger and Sonja Herres-Pawlis in the background section of the manuscript [29].

Chapter 3 - Distributed Computing Background
Parts of this work have been previously published as manuscript in *Concurrency and Computation: Practice and Experience* [265]. The manuscript is a joint work with Jano van Hemert (JV), Peter Kacsuk (PK), and Oliver Kohlbacher (OK). SG drafted most of the manuscript as discussed with OK. JV contributed to the expert opinion on portal development toolkits and PK contributed to the expert opinion on portals in HPC facilities, grid and cloud infrastructures.

Chapter 4 - Evaluated and Designed Infrastructure
Parts of these sections have been previously published in [266] and [29]. The insights in UNICORE 6 have been previously published as background in the manuscripts [266] and [29]. It is a joint work with Richard Grunzke and Bernd Schuller. SG evaluated the portal frameworks and designed the Liferay roles as discussed with MoSGrid consortium members.

Chapter 5 - A Novel Credential Management for Science Gateways
This work is part of a manuscript that has been previously published in [29]. It is a joint work with Miklos Kozlovszky (MK), Bernd Schuller (BS), Valentina Huber (VH), and Anna Szikszay Fabri (AS). SG drafted the specifications for the credential management in collaboration with MK and BS. AS extended the credential management for WS-PGRADE by SAML as discussed with SG and MK. SG extended the basic version of the integrated applet, which was created by VH and by AS.

Chapter 6 - Enhanced Job and Workflow Management
The job and workflow management via gUSE and WS-PGRADE as well as the workflow interoperability is part of a manuscript that has been previously published in the proceedings of IWSG 2010 [266]. It is a joint work with Bernd Schuller (BS), Istvan Marton (IM), Miklos Kozlovszky (MK), Patrick Schäfer (PS), and Oliver Kohlbacher (OK). SG designed and implemented the UNICORE submitter and corresponding plugins as discussed with BS, IM, MK, and OK. PS extended the UNICORE submitter for additional parameters and remote files. The Application Specific Module is part of a manuscript that has been previously published in [29]. It is a joint work with Miklos Kozlovszky (MK) and Akos Balasko (AB). SG and MK drafted the specifications for the Application Specific Module whereas AB designed and implemented it.

Chapter 8 - Automatic Creation of Portlets for Workflows
Part of this work has been published as manuscript at the CCGrid 2010 [221]. It is a joint work with Jano van Hemert (JV), Jos Koetsier (JK), Andreas Bertsch (AB), and Oliver Kohlbacher (OK). SG designed and implemented the prototype for TOPP as discussed with OK and JV. JK supported with his deep insight in Rapid the development of the Jython plugin and AB supported with his knowledge about TOPP the offered pipelines.

Figures of the manuscript [29] appear throughout this thesis. The figure of the manuscript [221] is used in Chapter 8.

D Curriculum Vitae

Personal Details

Date of Birth:	24.03.1972
Place of Birth:	Ahaus, Germany
Marital Status:	Divorced
Nationality:	German

Education and Qualifications

since 2006	**PhD student** in Oliver Kohlbacher's group ”Applied Bioinformatics Group”, University of Tübingen, Germany
1997 - 2005	Extramural studies in computer science (minor business management) at FernUniversität Hagen, Germany Degree **Diplom-Informatiker**
2001	**Student** at UCLA, Los Angeles, US
1991 - 1994	**Apprenticeship** Mathematisch-technische Assistentin at Westfälische Wilhelms-Universität Münster, Germany Degree **IHK-Abschluss**
1991	**University-entrance diploma** (Abitur) at Pius-Gymnasium in Coesfeld, Germany

Research Visits

Sep 2012	Research visit in Jarek Nabrzyski's group **Center for Research Computing**, University of Notre Dame, US
Feb 2007 - Aug 2011	Seven research visits in Malcolm Atkinson's group (formerly Jano van Hemert's group) at the **e-Science Institute Edinburgh**, UK
Jul - Aug 2010	Research visit in Peter Kacsuk's group at **MTA SZTAKI**, Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, Hungary
Sep - Oct 2008	Research visit in Shantenu Jha's group at **Center for Computation & Technology**, LSU, Baton Rouge, US

Work Experience

since Sep 2012	**diamedi UG** (haftungsbeschränkt), Berlin, Germany
	Managing Partner
	Design and implementation of the diamedi UG internet platform
	Marketing and accounting
since Oct 2011	**Freelancer**, Tübingen, Germany
	Senior Consultant for web-based applications and project management
	Design and implementation of webpages (PHP, MySQL, TYPO3)
	Search Engine Optimization (SEO)
May 2006 -	**University of Tübingen**, Germany
Sep 2011	**Research Associate**
	Workpackage leader ”Portal” in MoSGrid (Molecular Simulation Grid)
	Administration of clusters in bwGRiD (Baden-Württemberg Grid)
	Design and implementation of portals for bioinformatics
	Parallel programming (GPU, OpenMP, OpenMPI)
	Teaching
	Writing proposals
	Chairing and organisation of workshops and conferences
Dec 2004 -	**LVM Versicherungen a.G.**, Münster, Germany
Apr 2006	**Head** of the group ”Application systems”
	Infrastructure for web-based applications
	Authentification via smartcards
	Migration of Lotus Notes
May 2000 -	**LVM Versicherungen a.G.**, Münster, Germany
Nov 2004	**System programmer** (C, Perl, Unix scripts) und
	Administrator (MQSeries, SAP, Lotus Notes)
	Integration of ”'GDV-Schadennetz”' with MQSeries
	Migration of SAP R/2 to SAP R/3
Sep -	**Internship** at **InterGest North America**, New York, US
Dec 2001	Application programming in Java and Visual Basic
	Training of employees in SAP R/3 BC
Oct 1996 -	**LVM Versicherungen a.G.**, Münster, Germany
May 2000	**Application developer** (PL/I, IMS, DB2)
	Design and implementation of a new accident insurance dialog
	Design and implementation of an automatic request processing
Sep 1994 -	**Coesfelder Holzwerke GmbH & Co KG**, Coesfeld, Germany
Sep 1996	**Administrator** (SAP, Network) and **Programmer** (ABAP/4, C)
	Design and implementation of a sales information system
	Configuration and installation of the network infrastructure

Additional Occupation

Oct 1998 -	**Tutor** at the study center Coesfeld of the **FernUniversität Hagen**
Mar 1999	Course guidance
	Organisation of events

Talks

- ”Web-based Science Gateways for Structural Bioinformatics”, Rutgers Discovery Informatics Institute, Rutgers University, New Brunswick, US, 2012
- ”Web-based Science Gateways for Structural Bioinformatics”, Center for Research Computing, University of Notre Dame, US, 2012
- ”Web-based Science Gateways for Structural Bioinformatics”, Center for Clinical and Research Informatics, NorthShore University HealthSystem, Evanston, US, 2012
- ”Granular Security for a Science Gateway in Structural Bioinformatics”, IWSG-Life 2011, London, UK, 2011
- Invited talk on ”The MoSGrid Portal - A workflow-enabled Grid Portal for Molecular Simulations”, 7th Black Forest Grid Workshop, Freiburg, Germany, 2011
- ”A Science Gateway for Molecular Simulations”, EGI User Forum, Vilnius, Lithuania, 2011
- Invited talk on ”UNICORE integration into a science gateway for MoSGrid”, Supercomputing’10, New Orleans, US, 2010
- ”Workflow Interoperability in a Grid Portal for Molecular Simulations”, IWSG2010, Catania, Italy, 2010
- ”A gUSE submitter for MoSGrid”, MTA SZTAKI, Computer and Automation, Research Institute, Hungarian Academy of Sciences, Budapest, Hungary, 2010
- ”Requirements on a portal for MoSGrid”, PUCOWO, ETH Zürich, Switzerland, 2010
- ”EuSGE - European Science Gateways for e-Science”, University of Westminster, London, UK, 2009
- Invited talk on ”A Workflow-enabled Grid-Portal for Bioinformatics”, NorduGrid 2008, Budapest, Hungary, 2008
- ”A Workflow-enabled Grid-Portal for Bioinformatics”, CCT, LSU, Baton Rouge, US, 2008
- ”Workflow Engines in a Bioinformatic Portal”, e-Science Institute Edinburgh, UK, 2007

Academic services

Chair of IWSG-Life 2012, May 2012, Amsterdam, The Netherlands
Chair of IWPLS'09, September 2009, Edinburgh, UK

PC member of IWSG 2013, June 2013, Zurich, Switzerland
PC member of PDP 2013, February 2013, Belfast, Northern Ireland
PC member of HealthGrid Conference 2012, May 2012, Amsterdam, The Netherlands
PC member of IWSG-Life 2011, June 2011, London, UK
PC member of PUCOWO 2011, June 2011, London, UK
PC member of GWW 2011, March 2011, Cologne, Germany
PC member of IWSG 2010, September 2010, Catania, Italy
PC member of PUCOWO 2010, June 2010, Zürich, Switzerland

Editor of Proceedings of the HealthGrid Conference and IWSG-Life 2012
Guest editor of Concurrency and Computation: Practice and Experience, editor of the Special Issue: IWPLS 2009
Editor of the Proceedings of IWPLS'09, Edinburgh, UK, September 2009

Reviewer for the CCGRID 2009, May 2009, Shanghai, China
Reviewer for the Journal of Grid Computing

E

List of Publications

*These authors contributed equally to the respective work.

Published manuscripts

- **Gesing, S.***, Grunzke, R.*, Krüger, J., Birkenheuer, G., Wewior, M., Schäfer, P., Schuller, B., Schuster, J., Herres-Pawlis, S., Breuers, S., Balasko, A., Kozlovszky, M., Szikszay Fabri, A., Packschies, L., Kacsuk, P., Blunk, D., Steinke, T., Brinkmann, A., Fels, G., Müller-Pfefferkorn, R., Jäkel, R., and Kohlbacher, O. "A Single Sign-On Infrastructure for Science Gateways on a Use Case for Structural Bioinformatics". *Journal of Grid Computing*, 10(4):769-790, 2012.
- **Gesing, S.**, Herres-Pawlis, S., Birkenheuer, G., Brinkmann, A., Grunzke, R., Kacsuk, P., Kohlbacher, O., Kozlovszky, M., Krüger, J., Müller-Pfefferkorn, R., Schäfer, P., and Steinke, T. "A Science Gateway Getting Ready for Serving the International Molecular Simulation Community". *Proceedings of Science*, PoS(EGICF12-EMITC2)050, 2012.
- Schlemmer, T.*, Grunzke, R.*, **Gesing, S.***, Krüger, J., Birkenheuer, G., Müller-Pfefferkorn, R., and Kohlbacher, O. "Generic User Management for Science Gateways via Virtual Organizations". *In: EGI Technical Forum 2012*, September 2012, Prague, Czech Republic, 2012.
- Herres-Pawlis, S., Birkenheuer, G., Brinkmann, A., **Gesing, S.**, Grunzke, R., Jäkel, R., Kohlbacher, O., Krüger, J., and Dos Santos Vieira, I. "Workflow-enhanced conformational analysis of guanidine zinc complexes via a science gateway". *Studies in Health Technology and Informatics*, 175:142-151, IOS Press, 2012.
- **Gesing, S.** and Krüger, J. "Workshop Series IWSG-Life". *Studies in Health Technology and Informatics*, 175:115-118, IOS Press, 2012.
- **Gesing, S.**, Herres-Pawlis, S., Birkenheuer, G., Brinkmann, A., Grunzke, R., Kacsuk, P., Kohlbacher, O., Kozlovszky, M., Krüger, J., Müller-Pfefferkorn, R., Schäfer, P., and Steinke, T. "The MoSGrid Community - From National to International Scale". *In: EGI Community Forum 2012*, March 2012, Munich, Germany, 2012.

- Birkenheuer, G., Blunk, D., Breuers, S., Brinkmann, A., dos Santos Vieira, I., Fels, G., **Gesing, S.**, Grunzke, R., Herres-Pawlis, S., Kohlbacher, O., Krüger, J., Lang, U., Packschies, L., Müller-Pfefferkorn, R., Schäfer, P., Steinke, T., Warzecha K., and Wewior, M. (2012). ”MoSGrid: Efficient Data Management and a Standardized Data Exchange Format for Molecular Simulations in a Grid Environment”. *Journal of Cheminformatics*, 4(Suppl 1):P21, 2012.
- Birkenheuer, G., Blunk, D., Breuers, S., Brinkmann, A., Fels, G., **Gesing, S.**, Grunzke, R., Herres-Pawlis, S., Kohlbacher, O., Krüger, J., Lang, U., Packschies, L., Müller-Pfefferkorn, R., Schäfer, P., Schuster, J., Steinke, T., Warzecha, K., and Wewior, M. ”MoSGrid: Progress of Workflow driven Chemical Simulations”. *Proceedings of Grid Workflow Workshop 2011*, Cologne, Germany, March 4, 2011, CEUR Workshop Proceedings, Vol-826, ISSN 1613-0073, 2012.
- **Gesing, S.**, Hemert, J. v., Kacsuk, P. and Kohlbacher, O. ”Special Issue: Portals for life sciences - Providing intuitive access to bioinformatic tools”. *Concurrency and Computation: Practice and Experience*, 23: 223-234, 2011.
- **Gesing, S.***, Grunzke, R.*, Balasko, A., Birkenheuer, G., Blunk, D., Breuers, S., Brinkmann, A., Fels, G., Herres-Pawlis, S., Kacsuk, P., Kozlovszky, M., Krüger, J., Packschies, L., Schäfer, P., Schuller, B., Schuster, J., Steinke, T., Szikszay Fabri, A., Wewior, M., Müller-Pfefferkorn, R., and Kohlbacher, O. ”Granular Security for a Science Gateway in Structural Bioinformatics” *Proc. of 3rd Workshop IWSG-Life 2011*, London, UK, June 8-10, 2011, CEUR Workshop Proceedings, Vol-819, ISSN 1613-0073, 2011.
- **Gesing, S.**, Kacsuk, P., Kozlovszky, M., Birkenheuer, G., Blunk, D., Breuers, S., Brinkmann, A., Fels, G., Grunzke, R., Herres-Pawlis, S., Krüger, J., Packschies, L., Müller-Pfefferkorn, R., Schäfer, P., Steinke, T., Szikszay Fabri, A., Warzecha, K., Wewior, M., and Kohlbacher, O. ”A Science Gateway for Molecular Simulations”. *In: EGI User Forum 2011*, Book of Abstracts, pp. 94-95, ISBN 978 90 816927 1 7, 2011.
- Krüger, J., Birkenheuer, G., Blunk, D., Breuers, S., Brinkmann, A., Fels, G., **Gesing, S.**, Grunzke, R., Kohlbacher, O., Kruber, N., Lang, U., Packschies, L., Müller-Pfefferkorn, R., Herres-Pawlis, S.Schäfer, P., Schmalz, H., Steinke, T., Warzecha, K., and Wewior, M. ”Molecular simulation grid”. *Journal of Cheminformatics*, 3(Suppl 1):O17, 2011.
- Birkenheuer, G., Blunk, D., Breuers, S., Brinkmann, A., dos Santos Vieira, I., Fels, G., **Gesing, S.**, Grunzke, R., Herres-Pawlis, S., Kohlbacher, O., Krüger, J., Lang, U., Packschies, L., Müller-Pfefferkorn, R., Schäfer, P., Schmalz, H., Steinke, T., Warzecha, K., and Wewior, M. ”A Molecular Simulation Grid as new tool for Computational Chemistry, Biology and Material Science”. *Journal of Cheminformatics*, 3(Suppl 1):P14, 2011.
- **Gesing, S.**, van Hemert, J.I., Koetsier, J., Bertsch, A, and Kohlbacher, O. ”TOPP goes Rapid - The OpenMS Proteomics Pipeline in a Grid-enabled Web Portal”. *CCGrid*, pp.598-599, 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010.
- **Gesing, S.**, Marton, I., Birkenheuer, G., Schuller, B., Grunzke, R., Krüger, J.,

Breuers, S., Blunk, D., Fels, G., Packschies, L., Brinkmann, A., Kohlbacher, O., and Kozlovszky, M. ”Workflow Interoperability in a Grid Portal for Molecular Simulations”. *In: Proceedings of the International Workshop on Science Gateways (IWSG2010)*, pp. 44-48, Consorzio COMETA, 2010.

- Wewior, M., Packschies, L., Blunk, D., Wickeroth, D., Warzecha, K., Herres-Pawlis, S., **Gesing, S.**, Breuers, S., Krüger, J., Birkenheuer, G., and Lang, U. ”The MoS-Grid Gaussian portlet - Technologies for Implementation of Portlets for Molecular Simulations”. *In: Proceedings of the International Workshop on Science Gateways (IWSG2010)*, pp. 39-43, Consorzio COMETA, 2010.
- Krüger, J., Birkenheuer, G., Breuers, S., **Gesing, S.**, Wewior, M., Brinkmann, A., Blunk, D., Kohlbacher, O., Packschies, L., and Fels, G. ”Workflows and Analysis Approaches for Molecular Dynamics Simulations”. *In: Proceedings of the International Workshop on Science Gateways (IWSG2010)*, pp. 61, Consorzio COMETA, 2010.
- Birkenheuer, G., Breuers, S., Brinkmann, A., Blunk, D., Fels, G., **Gesing, S.**, Herres-Pawlis, S., Kohlbacher, O., Krüger, J., and Packschies, L. ”Grid-Workflows in Molecular Science”. *Software Engineering 2010*, Grid Workflow Workshop, pp. 177-184, GI-Edition - Lecture Notes in Informatics (LNI), P-160, ISSN 1617-5468, 2010.
- **Gesing, S.**, Kohlbacher, O., and Hemert, J. v. ”Portals for Life Sciences - a Brief Introduction”. *Proc. of 1st Workshop IWPLS’09*, Edinburgh, UK, September 14-15, 2009, CEUR Workshop Proceedings, Vol-513, ISSN 1613-0073, 2009.
- Schneeberger, K.*, Hagmann, J.*, Ossowski, S.*, Warthmann, N., **Gesing, S.**, Kohlbacher, O., and Weigel, D. ”Simultaneous alignment of short reads against multiple genomes”. *Genome Biology*, 10(9):R98, 2009.

Accepted manuscripts

- Packschies, L., Birkenheuer, G., Blunk, D., Breuers, S., Brinkmann, A., de la Garza, L., dos Santos Vieira, I., Fels, G., **Gesing, S.**, Grunzke, R., Herres-Pawlis, S., Kohlbacher, O., Krüger, J., Kruse, M., Lang, U., Müller-Pfefferkorn, R., Schäfer, P., Schlemmer, Schärfe, C., T., Steinke, T., Warzecha, K., and Zink, A. ”The MoSGrid-e-Science Gateway: Molecular Simulations in a Distributed Computing Environment”. *GCC 2012*, accepted.
- Grunzke, R., Birkenheuer, G., Blunk, D., Breuers, S., Brinkmann, A., **Gesing, S.**, Herres-Pawlis, S., Kohlbacher, O., Krüger, J., Kruse, M., Müller-Pfefferkorn, R., Schäfer, P., Schuller, B., Steinke, T., and Zink, A. ”A Data Driven Science Gateway for Computational Workflows”. *UNICORE Summit 2012*, accepted.

Submitted manuscripts

- Balasko, A., **Gesing, S.**, Kozlovszky, M., Kranyecz, Cs., Karoczkai, K., Kohlbacher, O., and Kacsuk, P. ”Workflow interoperability by using automatic workflow language translation”. *Future Generation Computer Systems*, submitted.

Edited journals/proceedings

- **Gesing, S.**, Glatard, T., Krüger, J., Delgado Olabarriaga, S., Solomonides, T., Silverstein, J., Montagnat, J., Gaignard, A., and Krefting, D. (eds.) ”HealthGrid Applications and Technologies Meet Science Gateways for Life Sciences”. *Studies in Health Technology and Informatics*, Volume 175, IOS Press, 2012.
- **Gesing, S.** and Hemert, J. v. (eds.) ”Special Issue: IWPLS 2009”. *Concurrency and Computation: Practice and Experience*, 23: 223-291, 2011.
- **Gesing, S.** and Hemert, J. v. (eds.) ”Proceedings of the International Workshop on Portals for Life Sciences 2009”. CEUR Workshop Proceedings, Vol-513, ISSN 1613-0073, 2009.

References

[1] I. Foster and C. Kesselmann. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[2] J. Myerson. "Cloud computing versus grid computing". *IBM developerWorks*, 2009.

[3] S. Johnston. "The future of cloud computing - an army of monkeys?". `http://samj.net/2008_07_01_archive.html`, 2008.

[4] A. Seffah and E. Metzker. "The obstacles and myths of usability and software engineering". *Commun. ACM*, 47(12):71–76, 2004.

[5] T. Brinck, D. Gergle, and S. D. Wood. *Usability for the Web: Designing Web Sites that Work* . Morgan Kaufmann, 2001.

[6] International Organization for Standardization. "ISO 9241, Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability". `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57992`, 1998.

[7] S. Krug. *Don't make me think!: A Common Sense Approach to Web Usability.* New Riders, 2005.

[8] T. Berners-Lee. *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. HarperOne, 1999.

[9] "Facebook website". `http://www.facebook.com`.

[10] International Telecommunication Union (ITU). "ITU-T Recommendation X.509". `http://www.itu.int/rec/T-REC-X.509/en`, 1988.

[11] OASIS. "Security Assertion Markup Language (SAML) V2.0". `http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip`, 2002.

[12] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. "A Security Infrastructure for Computational Grids". In *CCS '98 Proceedings of the 5th ACM conference on Computer and communications security*, 1998.

[13] R. L. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein. "Federated Security: The Shibboleth Approach". *EDUCAUSE Quarterly*, 27(4):12–17, 2004.

[14] P. Kacsuk. "P-GRADE portal family for grid infrastructures". *Concurrency and Computation: Practice and Experience*, 23(3):235–245, 2011.

[15] Z. Farkas and P. Kacsuk. "P-GRADE Portal: a generic workflow system to support user communities". *Future Generation Computer Systems*, 27(5):454–465, 2011.

[16] L. Torterolo, I. Porro, M. Fato, M. Melato, A. Calanducci, and R. Barbera. "Building Science Gateways with EnginFrame: a Life Science example". In *Proceedings of International Workshop on Portals for Life Sciences (IWPLS'09)*, volume 513. CEUR Workshop Proceedings, 2009.

[17] J. Goecks, A. Nekrutenko, J. Taylor, and Galaxy Team. "Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences". *Genome Biology*, 11(8):R86, 2010.

[18] J. P. Jones. "Beowulf cluster computing with linux". chapter PBS: portable batch system, pages 369–390. MIT Press, Cambridge, MA, USA, 2002.

[19] G. Borges, M. David, J. Gomes, C. Fernandez, J. Lopez Cacheiro, P. Rey Mayo, A. Simon Garcia, D. Kant, and K. Sephton. "Sun Grid Engine, a new scheduler for EGEE middleware". In *IBERGRID - Iberian Grid Infrastructure Conference*, 2007.

[20] A. Abdelnur and S. Hepper. "JSR 168: Portlet Specification". `http://www.jcp.org/en/jsr/detail?id=168`, 2003.

[21] M.S. Nicklous and S. Hepper. "JSR 286: Portlet Specification 2.0". `http://www.jcp.org/en/jsr/detail?id=286`, 2008.

[22] Inc. Liferay. "Liferay". `http://www.liferay.com`.

[23] Apache Software Foundation. "Pluto". `http://portals.apache.org/pluto/`.

[24] M. Grönroos. *Book of Vaadin.* Oy IT Mill Ltd, 2010.

[25] M. Hahn. "The Google Web Toolkit: a deeper look and Extensions for GWT". `http://www.dark-bit.de/wp-content/uploads/2009/07/paper_marcel_hahn_final.pdf`, 2008.

[26] J. Koetsier and J. van Hemert. "Rapid Development of Computational Science Portals". In *Proceedings of International Workshop on Portals for Life Sciences (IWPLS'09)*, volume 513. CEUR Workshop Proceedings, 2009.

[27] M. McLennan and R. Kennell. "HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering". *Computing in Science & Engineering*, 12(2):48–52, 2010.

[28] W3C. "Extensible Markup Language (XML) 1.0 (Fifth Edition)". `http://www.w3.org/TR/REC-xml/`, 2008.

[29] S. Gesing, R. Grunzke, J. Krüger, G. Birkenheuer, M. Wewior, P. Schäfer, B. Schuller, J. Schuster, S. Herres-Pawlis, S. Breuers, A. Balasko, M. Kozlovszky, A. Szikszay Fabri, L. Packschies, P. Kacsuk, D. Blunk, T. Steinke, A. Brinkmann, G. Fels, R. Müller-Pfefferkorn, R. Jäkel, and O. Kohlbacher. "A Single Sign-On Infrastructure for Science Gateways on a Use Case for Structural Bioinformatics". *Journal of Grid Computing*, 10(4):769–790, 2012.

[30] G. Birkenheuer, S. Breuers, A. Brinkmann, D. Blunk, G. Fels, S. Gesing, S. Herres-Pawlis, O. Kohlbacher, J. Krüger, and L. Packschies. "Grid-Workflows in Molecular Science". In *Software Engineering 2010, Grid Workflow Workshop*, volume P-160, pages 177–184. GI-Edition - Lecture Notes in Informatics (LNI), 2010.

[31] I. Foster. "Globus Toolkit Version 4: Software for Service-Oriented Systems". *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag(LNCS 3779):2–13, 2006.

[32] C. Ragusa, F. Longo, and A. Puliafito. "Experiencing with the Cloud over gLite". In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, CLOUD '09, pages 53–60. IEEE Computer Society, Washington, DC, USA, 2009.

[33] A. Streit, P. Bala, A. Beck-Ratzka, K. Benedyczak, S. Bergmann, R. Breu, J. M. Daivandy, B. Demuth, A. Eifer, A. Giesler, B. Hagemeier, V. Huber S. Holl, N. Lamla, D. Mallmann, A. S. Memon, M. S. Memon, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, T. Schlauch, A. Schreiber, T. Soddemann, and W. Ziegler. "Unicore 6 - Recent and Future Advancements". *JUEL-4319*, 2010.

[34] "UNICORE Commandline Client". `http://www.unicore.eu/unicore/architecture/client-layer.php#anchor_ucc`, 2011.

[35] "UNICORE Rich Client". `http://www.unicore.eu/unicore/architecture/client-layer.php#anchor_urc`, 2011.

[36] "D-Grid". `http://www.d-grid-gmbh.de/index.php?id=1&L=1`.

[37] F. Hupfeld, T.i Cortes, B. Kolbeck, J. Stender, E. Focht, M. Hess, J. Malo, J. Marti, and E. Cesario. "The XtreemFS Architecture - A Case for Object-based File Systems in Grids". *Concurrency and Computation: Practice and Experience*, 20(17):2049–2060, 2008.

[38] J. Gu and P. E. Bourne. *Structural Bioinformatics.* Wiley-Blackwell, 2009.

[39] A. R. Leach. *Molecular Modelling: Principles and Applications (2nd Edition).* Prentice Hall, 2001.

[40] N. Chandra, P. Anand, and K. Yeturu. "Structural Bioinformatics: Deriving Biological Insights from Protein Structures". *Interdisciplinary Sciences: Computational Life Sciences*, 2(4):347–366.

[41] E. B. Fauman, A. L. Hopkins, and C. R. Groom. *Structural Bioinformatics in Drug Discovery*, chapter 23, pages 477–497. Wiley-Liss Inc., Hoboken, New Jersey, 2003.

[42] T. Lengauer, R. Mannhold, H. Kubinyi, and H. Timmerman. *Bioinformatics - From Genomes to Drugs. Vol.1: Basic technologies*, volume 14 of *Methods and principles in medicinal chemistry.* Wiley-VCH, 2002.

[43] H. M. Berman, K. Henrick, H. Nakamura, J. Markley, P. E. Bourne, and J. Westbrook. “Realism about PDB”. *Nature biotechnology*, 25(8):845–846, 2007.

[44] PDB. “Yearly Growth of Total Structures”. `http://www.rcsb.org/pdb/statistics/contentGrowthChart.do?content=total&seqid=100`, 2012.

[45] P. A. M. Dirac. “Quantum Mechanics of Many-Electron Systems”. *Proceedings of the Royal Society*, A123:714–733, 1929.

[46] A. Moll, A. Hildebrandt, H. Lenhof, and O. Kohlbacher. “BALLView: A tool for research and education in molecular modeling”. *Bioinformatics*, (3):365–366, 2006.

[47] B. McMahon and Robert M. Hanson. “A toolkit for publishing enhanced figures”. *Journal of Applied Crystallography*, 41(4):811–814, 2008.

[48] J. Harvey. “Molecular Electronic Structure”. `http://www.chm.bris.ac.uk/pt/harvey/elstruct/hf_2.html`.

[49] S. Wan, D. W. Wright, and P. V. Coveney. “Mechanism of Drug Efficacy Within the EGF Receptor Revealed by Microsecond Molecular Dynamics Simulation”.

[50] O. Kohlbacher. *New approaches to protein docking.* PhD thesis, Universität des Saarlandes, 2000.

[51] E. Schrödinger. “Quantisierung als Eigenwertproblem (Erste Mitteilung)”. *Annalen der Physik*, 4 (79):361–376, 1926.

[52] M. Born and J. R. Oppenheimer. “Zur Quantentheorie der Molekeln”. *Annalen der Physik*, 389(20): 457–484, 1927.

[53] V. Fock. “Näherungsmethode zur Lösung des quantenmechanischen Mehrkörperproblems”. *Zeitschrift für Physik*, 61:126–148, 1930.

[54] J. Slater and H.C. Verma. “The Theory of Complex Spectra”. *Physical Review*, 34(2):1293–1295, 1929.

[55] V. Fock. “”Selfconsistent field” mit Austausch von Natrium”. *Zeitschrift für Physik*, 62(11-12): 795–805, 1930.

[56] C. Møller and M.S. Plesset. “Note on an Approximation Treatment for Many-Electron Systems”. *Physical Review*, 46:618–622, 1934.

[57] H.G. Kümmel. “A biography of the coupled cluster method”. In *Recent progress in many-body theories, 11th international conference*, pages 334–348. World Scientific Publishing, 2002.

[58] J. E. Lennard-Jones. “The electronic structure of some diatomic molecules”. *Transactions of the Faraday Society*, 25:668–686, 1929.

[59] C.C.J. Roothaan. “New Developments in Molecular Orbital Theory”. *Reveiws of Modern Physics*, 23:69–89, 1951.

[60] G.G. Hall. “The Molecular Orbital Theory of Chemical Valency VIII. A Method for Calculating Ionisation Potentials”. *Proceedings of the Royal Society (London)*, A205:541–552, 1951.

[61] J. Slater. “Atomic Shielding Constants”. *Physical Review*, 36:57–64, 1930.

[62] M.M. Francl, J.S. Binkley, M.S. Gordon, D.J. DeFrees, and J.A. Pople. “Self-consistent molecular orbital methods. XXIII. A polarization-type basis set for second-row element”. *Journal of Chemical Physics*, 77:3654–3665, 1982.

[63] R. Pariser and R.G. Parr. “A Semi-Empirical Theory of the Electronic Spectra and Electronic Structure of Complex Unsaturated Molecules. I.”. *Journal of Chemical Physics*, 21(3), 1953.

[64] J.A. Pople and G.A. Segal. “Approximate Self-Consistent Molecular Orbital Theory. II. Calculations with Complete Neglect of Differential Overlap”. *Journal of Chemical Physics*, 43:S136–S149, 1965.

[65] J.A. Pople, D.P. Santry, and G.A. Segal. “Approximate Self-Consistent Molecular Orbital Theory. I. Invariant Procedures”. *Journal of Chemical Physics*, 43:S129–S135, 1965.

[66] M.J.S. Dewar and W. Thiel. “Ground states of molecules. 38. The MNDO method. Approximations and parameters”. *Journal of the American Chemical Society*, 99(15):4899–4907, 1977.

[67] M.J.S. Dewar, E.G. Zoebisch, E.F. Healy, and J.J.P. Stewart. “AM1: A new general purpose quantum mechanical molecular model”. *Journal of the American Chemical Society*, 107(13):3902–3909, 1985.

[68] J.J.P. Stewart. “Optimization of parameters for semiempirical methods I. Method”. *Journal of Computational Chemistry*, 10(2):209–220, 1989.

[69] J.J.P. Stewart. “Optimization of parameters for semiempirical methods II. Applications”. *Journal of Computational Chemistry*, 10(2):221–264, 1989.

[70] R.G. Parr. “Density Functional Theory”. *Annual Review of Physical Chemistry*, 34:631–656, 1983.

[71] R.G. Parr and W. Yang. *Density-Functional Theory of Atoms and Molecules*. Oxford University Press, 1994.

[72] S.H. Vosko, L. Wilk, and M. Nusair. “Accurate Spin-dependent Electron Liquid Correlation Energies for Local Spin Density Calculations: A Critical Analysis”. *Canadian Journal of Physics*, 58:1200–1211, 1980.

[73] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, Jr. J. A. Montgomery, T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, , and J. A. Pople. “Gaussian 03, Revision C.02”, 2004. Gaussian, Inc., Wallingford CT.

[74] “TURBOMOLE v6.2 2010, A Development of University of Karlsruhe and Forschungszentrum Karlsruhe Gmbh, 1989-2007, TURBOMOLE Gmbh, since 2007”. `http://www.turbomole.com`.

[75] J. D . Van der Waals. “The equation of state for gases and liquids”. *Nobel Lecture*, 1910.

[76] J. E. Lennard-Jones. “On the Determination of Molecular Fields”. *Proceedings of Royal Society London*, A 106(738):463–477, 1924.

[77] N.L. Allinger. “Conformational Analysis. 130. MM2. A Hydrocarbon Force Field Utilizing V_1 and V_2 Torsional Terms”. *Journal of the American Chemical Society*, 99:8127–8134, 1977.

[78] J. Lii and N.L. Allinger. “Molecular Mechanics. The MM3 Force Field for Hydrocarbons. 2. Vibrational Frequencies and Thermodynamics”. *Journal of the American Chemical Society*, 111:8566–8582, 1989.

[79] N.L. Allinger, K. Chen, and J. Lii. “An Improved Force Field (MM4) for Saturated Hydrocarbons”. *Journal of Computational Chemistry*, 17:642–668, 1996.

[80] S.J. Weiner, P.A. Kollman, D.A. Case, U.C. Singh, C. Ghio, G. Alagona, S. Profena, and P. Weiner. “A New Force Field for Molecular Simulation of Nucleic Acids and Proteins”. *Journal of the American Chemical Society*, 106:765–784, 1984.

[81] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, and M. Karplus. “CHARMM: A program for macromolecular energy, minimization, and dynamics calculations”. *Journal of Compational Chemistry*, 4(2):187–217, 1983.

[82] M. Svensson, S. Humbel, R.D.J. Froese, T. Matsubara, S. Sieber, and K. Morokuma. “ONIOM: A Multilayered Integrated MO + MM Method for Geometry Optimizations and Single Point Energy Predictions. A Test for Diels-Alder Reactions and $Pt(P(t\text{-}Bu)_3)_2 + H_2$ Oxidative Addition”. *Journal of Physical Chemistry*, 100(50):19357–19363, 1996.

[83] M.E. Tuckerman and G.J. Martyna. “Understanding Modern Molecular Dynamics: Techniques and Applications”. *Journal of Physical Chemistry*, 104:159–178, 2000.

[84] J. Gullingsrud, R. Braun, and K. Schulten. “Reconstructing potentials of mean force through time series analysis of steered molecular dynamics simulations”. *Journal of Computational Physics*, 151: 190–211, 1999.

[85] G.M. Torrie and J.P. Valleau. “Nonphysical Sampling Distributions in Monte Carlo Free-Energy Estimation: Umbrella Sampling”. *Journal of Computational Physics*, 23:187–199, 1977.

[86] B. Roux. “The Calculation of the Potential of Mean Force using Computer Simulations”. *Computer Physics Communications*, 91(1-3):275–282, 1995.

[87] D. Marx and J. Hutter. *Ab Initio Molecular Dynamics: Basic Theory and Advanced Methods*. Cambridge University Press, 2009.

[88] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl. “GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation”. *Journal of Chemical Theory and Computation*, 4(3):435–447, 2008.

[89] D.A. Case, T.A. Darden, III T.E. Cheatham, C.L. Simmerling, J. Wang, R.E. Duke, R. Luo, R.C. Walker, W. Zhang, K.M. Merz, B. Roberts, S. Hayik, A. Roitberg, G. Seabra, J. Swails, A.W. Goetz, I. Kolossvai, K.F. Wong, F. Paesani, J. Vanicek, R.M. Wolf, J. Liu, X. Wu, S.R. Brozell, T. Steinbrecher, H. Gohlke, Q. Cai, X. Ye, J. Wang, M.-J. Hsieh, G. Cui, D.R. Roe, D.H. Mathews, M.G. Seetin, R. Salomon-Ferrer, C. Sagui, V. Babin, T. Luchko, S. Gusarov, A. Kovalenko, and P.A. Kollman. “AMBER 12”. *University of California, San Francisco*, 2012.

[90] E. Fischer. “Einfluss der Konfiguration auf die Wirkung der Enzyme”. *Berichte der deutschen chemischen Gesellschaft*, 27:2985–2993, 1894.

[91] M.L. Connolly. “Solvent-accessible surfaces of proteins and nucleic-acids”. *Science*, 221:709–713, 1983.

[92] B. Lee and F.M. Richards. “The interpretation of protein structures: estimation of static accessibility”. *Journal of Molecular Biology*, 55(3):379–400, 1971.

[93] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. “A Fast Flexible Docking Method Using an Incremental Construction Algorithm”. *Journal of Molecular Biology*, 261:470–489, 1996.

[94] H.J. Böhm. “The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure”. *Journal of Computer-Aided Molecular Design*, 8(3):243–256, 1994.

[95] G.M. Morris, D.S. Goodsell, R.S. Halliday, R. Huey, W.E. Hart, R.K. Belew, and A.J. Olson. “Automated Docking Using a Lamarckian Genetic Algorithm and and Empirical Binding Free Energy Function”. *Journal of Computational Chemistry*, 19:1639–1662, 1998.

[96] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. “Optimization by Simulated Annealing”. *Science*, 220 (4598):671–680, 1983.

[97] D. Whitley, S. Gordon, and K. Mathias. “Lamarckian Evolution, the Baldwin effect and function optimization”. In *Parallel Problem Solving from Nature - PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 6–15. Springer-Verlag, 1994.

[98] T. A. Halgren, R. B. Murphy, R. A. Friesner, H. S. Beard, L. L. Frye, W. T. Pollard, and J. L. Banks. “Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening”. *J. Med. Chem.*, 47:1750–759, 2004.

[99] “Schrödinger Suite”. `http://www.schrodinger.com/`.

[100] “CADDSuite (Computer-aided Drug Design Suite)”. `http://www.ball-project.org/caddsuite`.

[101] M. Schumann and O. Kohlbacher. “IMGDock and TaGRes: Efficient and Open-Source Docking and Consecutive Target-Specific Rescoring”. (submitted).

[102] BioSolvIt. “FlexX-Ensemble”. `http://www.biosolveit.de/FlexX-Ensemble/`.

[103] H. Claussen, C. Buning, M. Rarey, and T. Lengauer. “FlexE: efficient molecular docking considering protein structure variations”. *Journal of Molecular Biology*, 308(2):377–395, 2001.

[104] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel. “KNIME: The Konstanz Information Miner”. In *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*, 2007.

[105] “Pipeline Pilot”. `http://accelrys.com/products/pipeline-pilot/`.

[106] I. Foster. “What is the Grid? A Three Point Checklist”. *GRIDtoday*, 1(6), 2002.

[107] J. Bresnahan, M. Link, G. Khanna, Z. Imani, R. Kettimuthu, and I. Foster. “Globus GridFTP: what’s new in 2007”. In *Proceedings of the first international conference on Networks for grid applications*, number 19 in GridNets ’07, pages 1–5. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[108] A. Rajasekar, M. Wan, and R. Moore. “mySRB and SRB, Components of a Data Grid”. In *11th IEEE International Symposium on High Performance Distributed Computing 2002*, pages 301–310, 2002.

[109] P. Saluja, BB. Prahlada Rao, V. Shashidhar, A. Paventhan, and N. Sharma. “An interoperable & optimal data grid solution for heterogeneous and SOA based Grid - GARUDA”. *IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, 2011*, 0:1–8, 2010.

[110] A.S. Rana, F. Wurthwein, T. Perelmutov, R. Kennedy, J. Bakken, T. Hesselroth, I. Fisk, P. Fuhrmann, M. Ernst, M. Lorch, and D. Skow. “Introducing Advanced Fine-grained Security in dCache-SRM for PetaByte-scale Storage Systems on Global Data Grids: gPLAZMA grid-aware PLuggable AuthoriZation MAnagement System”. In *Nuclear Science Symposium Conference*, pages 632–636, 2006.

[111] M. Jackson, M. Antonioletti, B. Dobrzelecki, and N. Chue Hong. “Distributed data management with OGSA-DAI”. *Grid and Cloud Database Management*, pages 63–86, 2011.

[112] J. Geelan. “Twenty-One Experts Define Cloud Computing”. `http://virtualization.sys-con.com/node/612375`, 2009.

[113] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”. *Future Generation Computer Systems*, 25:599–616, 2009.

[114] “National Institute of Standards and Technology (NIST)”. `http://www.nist.gov`.

[115] P. Mell and T. Grance. “The NIST Definition of Cloud Computing”. `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`, 2011.

[116] A. Weiss. “Computing in the clouds”. *netWorker*, 11(4):16–25, 2007.

[117] G. Perry. “How Cloud & Utility Computing Are Different”. `http://gigaom.com/2008/02/28/how-cloud-utility-computing-are-different/`, 2008.

[118] Amazon Web Services. “Amazon Elastic Compute Cloud (Amazon EC2)”. `http://aws.amazon.com/ec2/`.

[119] US Department of Energy (DOE). “The Magellan Report on Cloud Computing for Science”. `http://www.nersc.gov/assets/StaffPublications/2012/MagellanFinalReport.pdf`, 2011.

[120] Amazon Web Services. “Amazon Simple Storage Service (Amazon S3)”. `http://aws.amazon.com/s3/`.

[121] S. Ghemawat, H. Gobioff, and S. Leung. “The Google File System”. In *SOSP’03*. Bolton Landing, New York, USA, 2003.

[122] “Oracle Cloud File System”. `http://www.oracle.com/us/products/database/cloud-file-system/`.

[123] OASIS. “PKI Technical Standards”. `http://www.oasis-pki.org/resources/techstandards/#organisations`.

[124] “The Internet Engineering Task Force (IETF)”. `http://www.ietf.org/`.

[125] R. Housley, W. Polk, W. Ford, and D. Solo. “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (RFC 3280)”. `http://www.ietf.org/rfc/rfc3280.txt`, 2002.

[126] IGTF. “The International Grid Trust Federation”. `http://www.igtf.net/`.

[127] EUGridPMA. “European Grid Policy Management Authority”. `http://www.eugridpma.org/`.

[128] DFN. “Deutsches Forschungsnetz”. `https://www.pki.dfn.de/grid/`.

[129] GridKa-Ca. “Grid Computer Centre Karlsruhe Certification Authority”. `https://gridka-ca-sec.fzk.de/`.

[130] The Internet Engineering Task Force (IETF). “Securely Available Credentials (SACRED) - Credential Server Framework”. `http://tools.ietf.org/pdf/rfc3760.pdf`, 2004.

[131] OASIS. “Organization for the Advancement of Structured Information Standards”. `http://www.oasis-open.org`, 2011.

[132] S. Tuecke, V. Welch, and J. Novotny. “An Online Credential Repository for the Grid: MyProxy”. In *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, pages 104–111. IEEE Press, 2001.

[133] D. Georgakopoulos, M. Hornick, and A. Sheth. “An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure”. *Distributed and Parallel Databases*, 3: 119–153, 1995.

[134] “Lustre”. `http://wiki.lustre.org/`.

[135] J. Y. and R. Buyya. “A Taxonomy of Workflow Management Systems for Grid Computing”. *Journal of Grid Computing*, 3(3-4):171–200, 2005.

[136] A. Barker and J. van Hemert. “Scientific Workflow: A Survey and Research Directions”. In Roman Wyrzykowski and et al., eds., *Seventh International Conference on Parallel Processing and Applied Mathematics, Revised Selected Papers*, volume 4967 of *LNCS*, pages 746–753. Springer, 2008.

[137] B. Ludaescher, M. Weske, T. McPhillips, and S. Bowers. “Scientific Workflows: Business as Usual?”. In *7th Intl. Conf. on Business Process Management (BPM)*, 2009.

[138] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn. “Taverna: a tool for building and running workflows of services”. *Nucleic Acids Research*, 34, Web Server issue:729–732, 2006.

[139] UNICORE Team. “UNICORE Workflow System Manual”. `http://unicore.eu/documentation/manuals/unicore6/files/workflow/workflow-manual.pdf`, 2012.

[140] Internet Engineering Task Force (IETF). “A JSON Media Type for Describing the Structure and Meaning of JSON Documents”. `http://tools.ietf.org/pdf/draft-zyp-json-schema-03.pdf`, 2010.

[141] M. Sporny and N. Walsh. “Web Services: JSON vs. XML”. `http://digitalbazaar.com/2010/11/22/json-vs-xml/`.

[142] D. Crockford. “JSON: The Fat-Free Alternative to XML”. `http://www.json.org/fatfree.html`.

[143] S. Goessner. “Converting Between XML and JSON”. `http://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html`, 2006.

[144] W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. “Workflow Patterns”. *Distributed and Parallel Databases*, 14(3):5–51, 2003.

[145] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[146] K. Sugiyama, S. Tagawa, and M . Toda. “Methods for Visual Understanding of Hierarchical System Structures”. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[147] O. Bastert and C. Matuszewski. “Layered Drawings of Digraphs”. In M. Kaufmann and D. Wagner, eds., *Drawing Graphs: Methods and Models*, number 2025, pages 87–120. Springer, 2001.

[148] P. Eades and N.C. Wormald. “Edge crossings in drawings of bipartite graphs”. *Algorithmica*, 11: 379–403, 1994.

[149] M. Eiglsperger, M. Siebenhaller, and M. Kaufmann. “An Efficient Implementation of Sugiyama’s Algorithm for Layered Graph Drawing”. *Journal of Graph Algorithms and Applications*, 9(3):305–325, 2005.

[150] H.C. Purchase, C. Pilcher, and B. Plimmer. “Graph Drawing Aesthetics - Created by Users, Not Algorithms”. *Visualization and Computer Graphics, IEEE Transactions on*, 18(1):81–92, 2012.

[151] H. McWilliam, F. Valentin, M. Goujon, W. Li, M. Narayanasamy, J. Martin, T. Miyar, and R. Lopez. “Web services at the European Bioinformatics Institute-2009”. *Nucleic Acids Res.*, 37(Web Server issue):6–10, 2009.

[152] T. A. Wassenaar, M. van Dijk, N. Loureiro-Ferreira, G. van der Schot, S. J. de Vries, C. Schmitz, J. van der Zwan, R. Boelens, A. Giachetti, L. Ferella, A. Rosato, I. Bertini, T. Herrmann, H. R. A. Jonker, A. Bagaria, V. Jaravine, P. GÃ$\frac{1}{4}$ntert, H. Schwalbe, W. F. Vranken, J. F. Doreleijers, G. Vriend, G. W. Vuister, D. Franke, A. Kikhney, D. I. Svergun, R. Fogh, J. Ionides, E. D. Laue, C. Spronk, M. Verlato, S. Badoer, S. Dal Pra, M. Mazzucato, E. Frizziero, and A. M.J.J. Bonvin. “WeNMR: Structural Biology on the Grid ”. In *Proceedings of the International Workshop on Science Gateways for Life Sciences (IWSG-Life 2011)*, volume 819. CEUR Workshop Proceedings, 2011.

[153] OASIS. “Web Services for Remote Portlets Specification v2.0 (WSRP)”. `http://docs.oasis-open.org/wsrp/v2/wsrp-2.0-spec.html`, 2008.

[154] OpenSocial and Gadgets Specification Group. “OpenSocial Specification 1.0”. `http://opensocial-resources.googlecode.com/svn/spec/1.0/OpenSocial-Specification.xml`, 2010.

[155] The Apache Software Foundation. “Apache Tomcat”. `http://tomcat.apache.org/tomcat-6.0-doc/`.

[156] Java Community Process. “Java Servlet 2.5 Specification”. `http://jcp.org/aboutJava/communityprocess/mrel/jsr154/index.html`.

[157] Java Community Process. “JavaServer Pages 2.1”. `http://jcp.org/aboutJava/communityprocess/final/jsr245/index.html`.

[158] Internet Engineering Task Force (IETF). “Guidance for Authentication, Authorization, and Accounting (AAA) Key Management”. `http://tools.ietf.org/pdf/rfc4962.pdf`, 2007.

[159] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. “Role-Based Access Control Models”. *IEEE Computer (IEEE Press)*, 29(2):38–47, 1996.

[160] D. W. Chadwick and A. Otenko. “The PERMIS X.509 role based privilege management infrastructure”. *Future Generation Computer Systems*, 19(2):277–289, 2003.

[161] A. Balasko, M. Kozlovszky, A. Schnautigel, K. Karòckai, I. Màrton, T. Strodl, and P. Kacsuk. “Converting P-GRADE Grid Portal into E-Science Gateways”. In *Proceedings of the International Workshop on Science Gateways (IWSG10)*, pages 1–6. Consorzio COMETA, 2010.

[162] C. Zhang, I. Kelley, and G. Allen. “Grid portal solutions: a comparison of GridPortlets and OGCE”. *Concurrency and Computation: Practice & Experience - Workshop on Grid Computing Portals (GCE 2005)*, 19(12), 2007.

[163] M. Thomas, J. Boisseau, M. Dahan, C. Mills, S. Mock, and K. Mueller. “Development of NPACI Grid Application Portals and Portal Web Services”. *Cluster Computing*, 6(3):177–188, 2003.

[164] D. Szejnfeld, P. Dziubecki, P. Kopta, M. Krysinski, T. Kuczynski, K. Kurowski, B. Ludwiczak, T. Piontek, D. Tarnawczyk, M. Wolniewicz, P. Domagalski, J. Nabrzyski, and K. Witkowski. “Vine Toolkit - Towards Portal Based Production Solutions for Scientific and Engineering Communities with Grid-Enabled Resources Support”. *Scalable Computing: Practice and Experience*, 11(2):161–172, 2011.

[165] R. Barbera, G. Andronico, G. Donvito, A. Falzone, J.J. Keijser, G. La Rocca, L. Milanesi, G. P. Maggi, and S. Vicario. “A Grid Portal with Robot Certificates for Bioinformatics Phylogenetic Analyses”. *Concurrency and Computation: Practice and Experience*, 23(3):246–255, 2011.

[166] E. R. Watkins, M. McArdle, T. Leonard, and M. Surridge. “Cross-Middleware Interoperability in Distributed Concurrent Engineering”. In *Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, E-SCIENCE ’07, pages 561–568. IEEE Computer Society, Washington, DC, USA, 2007.

[167] M. Ellert, M. Gronager, A. Konstantinov, B. Konya, J. Lindemann, I. Livenson, J.L. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen. “Advanced Resource Connector middleware for lightweight computational Grids”. *Future Generation Computer Systems*, 23:219–240, 2007.

[168] D. P. Anderson. “Public Computing: Reconnecting People to Science”. In *Conference on Shared Knowledge and the Web*, pages 17–19, 2003.

[169] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. “The Eucalyptus Open-Source Cloud-Computing System”. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID ’09, pages 124–131. IEEE Computer Society, Washington, DC, USA, 2009.

[170] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster. “Virtual Infrastructure Management in Private and Hybrid Clouds”. *IEEE Internet Computing*, 13(5):14–22, 2009.

[171] M. Smith, M. R. Barton, M. Branschofsky, G. McClellan, J. Harford Walker, M. J. Bass, D. Stuve, and R. Tansley. “DSpace: An Open Source Dynamic Digital Repository”. *D-Lib Magazine*, 2003.

[172] G. Klimeck, M. McLennan, S.P. Brophy, G.B. Adams III, and M.S. Lundstrom. “nanoHUB.org: Advancing Education and Research in Nanotechnology”. *Computing in Science & Engineering*, 10 (5):17–23, 2008.

[173] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. “Job Submission Description Language (JSDL) Specification”. `http://www.ggf.org/documents/GFD.56.pdf`, 2005.

[174] B. Schuller, B. Demuth, H. Mix, K. Rasch, M. Romberg, U. Maran, S. Sild, P. Bala, E. del Grosso, M. Casalegno, N. Piclin, M. Pintore, W. Sudholt, and K. K. Baldridge. “Chemomentum - UNICORE 6 based infrastructure for complex applications in science and technology”. In *Euro-Par 2007 Workshops: Parallel Processing (Lecture Notes in Computer Science 4854)*, pages 94–103. Springer-Verlag Berlin Heidelberg, 2008.

[175] OASIS. “eXtensible Access Control Markup Language (XACML) Version 2.0”. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`, 2005.

[176] B. Hagemeier and R. Menday. “HiLA 2.0 - Evolutionary Improvements”. In *UNICORE Summit 2010*, volume 5 of *IAS Series*, pages 45–50, 2010.

[177] W3C. “Simple Object Access Protocol (SOAP) Version 1.2”. `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/`, 2007.

[178] OASIS. “Web Services Resource Framework (WSRF) - Primer v1.2”. `http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf`, 2006.

[179] D. Snelling, S. van den Berghe, and V. Li. “Explicit Trust Delegation: Security for Dynamic Grids”. In *Fujitsu Scientific and Technical Journal*, pages 282–294, 2004.

[180] K. Benedyczak, P. Bała, S. van den Berghe, R. Menday, and B. Schuller. “Key Aspects of the UNICORE 6 Security Model”. In *Future Generation Computer Systems*, number 27, pages 195–201, 2011.

[181] I. Johnson, A. Lakhani, B. Matthews, E. Yang, and C. Morin. “XtreemOS: Towards a Grid Operating System with Virtual Organisation Support”. In *UK eScience All Hands Meeting*, 2007.

[182] “FUSE”. `http://fuse.sourceforge.net`.

[183] P. E. O’Neil, E. Cheng, D. Gawlick, and E. J. O’Neil. “The Log-Structured Merge-Tree (LSM-Tree)”. *Acta Inf.*, 33(4):351–385, 1996.

[184] J. Stender, B. Kolbeck, F. Hupfeld, E. Cesario, E. Focht, M. Hess, J. Malo, and J. Martí. “Striping without sacrifices: maintaining POSIX semantics in a parallel file system”. In *First USENIX Workshop on Large-Scale Computing*, LASCO’08, pages 6:1–6:8. USENIX Association, Berkeley, CA, USA, 2008.

[185] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. “RAID: High-Performance, Reliable Secondary Storage”. *ACM Computing Surveys*, 26:145–185, 1994.

[186] “grid User Support Environment”. `https://sourceforge.net/projects/guse/`.

[187] “Jetspeed 2”. `http://portals.apache.org/jetspeed-2/`.

[188] “GateIn”. `http://www.jboss.org/gatein`.

[189] “eXo”. `http://www.exoplatform.com/`.

[190] J. Novotny, M. Russell, and O. Wehrens. “GridSphere: an advanced portal framework”. In *Euromicro Conference, 2004. Proceedings. 30th*, pages 412–419, 2004.

[191] Internet Engineering Task Force (IETF). “Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map”. `http://tools.ietf.org/pdf/rfc4510.pdf`, 2006.

[192] D. Mazurek. “CAS Protocol”. `http://www.jasig.org/cas/protocol`, 2005.

[193] openid.net. “OpenID Authentication 2.0 - Final”. `http://openid.net/specs/openid-authentication-2_0.html`, 2007.

[194] R. Murri, P. Z. Kunszt, S. Maffioletti, and V. Tschopp. “GridCertLib: A Single Sign-on Solution for Grid Web Applications and Portals”. *Journal of Grid Computing*, 9(4):441–453, 2011.

[195] X. D. Wang, M. Jones, J. Jensen, A. Richards, D. Wallom, T. Ma, R. Frank, D. Spence, S. Young, C. Devereux, and N.l Geddes. “Shibboleth Access for Resources on the National Grid Service (SARoNGS)”. In *Fifth International Conference on Information Assurance and Security*, volume 2, pages 338–341, 2009.

[196] R.O. Sinnott, T. Doherty, J. Jiang, S. McCafferty, A. Stell, and J. Watt. "Security-oriented Portals for the Life Sciences". In *Proceedings of International Workshop on Portals for Life Sciences (IWPLS'09)*, volume 513. CEUR Workshop Proceedings, 2009.

[197] S. Oaks. *Java Security (2nd Edition)*. O'Reilly Media, 2001.

[198] "Applets vs JWS vs Applications". `http://mindprod.com/jgloss/javawebstart.html#APPLETSVSJWS`.

[199] "VeriSign Authentication Services". `http://www.verisign.com/`.

[200] "thawte". `http://www.thawte.de/`.

[201] Object Management Group. "Unified Modeling Lanuage Specification". `http://www.omg.org`, 1998.

[202] R. Barbera, M. Fargetta, and R. Rotondo. "A Simplified Access to Grid Resources by Science Gateways". Proceedings of Science, 2011.

[203] EGI. "European Grid Infrastructure". `http://www.egi.eu/`.

[204] J. G. Hayes, E. Peyrovian, S.l Sarin, M. Schmidt, K. D. Swenson, and R. Weber. "Workflow Interoperability Standards for the Internet". *IEEE Internet Computing*, 4:37–45, 2000.

[205] "SHIWA - SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs". `http://www.shiwa-workflow.eu/project`.

[206] D. Krefting, T. Glatard, V. Korkhov, J. Montagnat, and S. Olabarriaga. "Enabling Grid Interoperability at Workflow Level". In *Proceedings of Grid Workflow Workshop 2011*, volume 826. CEUR Workshop Proceedings, 2012.

[207] K. Plankensteiner, J. Montagnat, and R. Prodan. "IWIR: A Language Enabling Portability Across Grid Workflow Systems". In *WORKS'11*, pages 97–106, 2011.

[208] M. Abouelhoda, S. Alaa, and M. Ghanem. "Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support". *BMC Bioinformatics*, 13(77), 2012.

[209] A. Alqaoud, I. Taylor, and A. Jones. "Scientific workflow interoperability framework". *International Journal of Business Process Integration and Management*, 5(1):93–105, 2010.

[210] W. Gropp, E. Lusk, and A. Skiellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface (Scientific and Engineering Computation)*. The MIT Press, 1994.

[211] R. M. Riordan. *Head first Ajax*. O'Reilly, 2008.

[212] M. Kradolfer and A. Geppert. "Dynamic Workflow Schema Evolution Based on Workflow Type Versioning and Workflow Migration". In *Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems*, COOPIS '99, pages 104–114. IEEE Computer Society, Washington, DC, USA, 1999.

[213] "The Python Language Reference". `http://docs.python.org/reference/`, 2011.

[214] K. Karasavvas, K. Wolstencroft, E. Mina, D. Cruickshank, A. Williams, C. Goble D. De Roure, and M. Roos. "Opening new gateways to workflows for life scientists". In *HealthGrid Applications and Technologies Meet Science Gateways for Life Sciences*. IOS Press, 2012. (in print).

[215] "Mako Templates for Python". `http://www.makotemplates.org/`.

[216] O. Faigon. "Topological Sort in Python". `http://www.bitformation.com/art/python_toposort.html`, 2005.

[217] R. Tarjan. "Depth-first search and linear graph algorithms". *SIAM Journal on Computing*, 1(2): 146–160, 1972.

[218] R. Sedgewick. *Algorithms*. Addison Wesley, 1983.

[219] B. Boe. "Line Segment Intersection Algorithm". `http://www.bryceboe.com/2006/10/23/line-segment-intersection-algorithm/`.

[220] "Jython: Python for the Java Platform". `http://www.jython.org/`.

[221] S. Gesing, J.I. van Hemert, J. Koetsier, A. Bertsch, and O. Kohlbacher. "TOPP goes Rapid - The OpenMS Proteomics Pipeline in a Grid-enabled Web Portal". In *CCGrid 2010, 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 598–599, 2010.

[222] O. Kohlbacher, K. Reinert, C. Gröpl, E. Lange, N. Pfeifer, O. Schulz-Trieglaff, and M. Sturm. “TOPP - the OpenMS Proteomics Pipeline”. *Bioinformatics*, 23(2):e191–e197, 2007.

[223] W3C. “XHTML 1.0 The Extensible HyperText Markup Language”. `http://www.w3.org/TR/xhtml1/`.

[224] I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pcikles, D. Pulsipher, C. Smith, and M. Theimer. “OGSA Basic Execution Service Version 1.0”. `http://www.ogf.org/documents/GFD.108.pdf`.

[225] W. Lee, A. McGough, and J. Darlington. “Performance Evaluation of the GridSAM Job Submission and Monitoring System”. In *UK e-Science All Hands Meeting*, pages 915–922, 2005.

[226] D. Thain, T. Tannenbaum, and M. Livny. “Distributed computing in practice: the Condor experience”. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[227] “Rapid Portal Development”. `https://sourceforge.net/projects/rapidportlet/`.

[228] M. Wewior, L. Packschies, D. Blunk, D. Wickeroth, K. Warzecha, S. Herres-Pawlis, S. Gesing, S. Breuers, J. Krüger, G. Birkenheuer, and U. Lang. “The MoSGrid Gaussian Portlet – Technologies for the Implementation of Portlets for Molecular Simulations”. In Roberto Barbera, Giuseppe Andronico, and Giuseppe La Rocca, eds., *Proceedings of the International Workshop on Science Gateways (IWSG10)*, pages 39–43. Consorzio COMETA, 2010.

[229] “ICEFaces”. `http://www.icesoft.org/`.

[230] J. Krüger and G. Fels. “Ion Permeation Simulations by Gromacs − an Example of High Performance Molecular Dynamics”. *Concurrency and Computation: Practice and Experience*, 23(3):279–291, 2011.

[231] P. Murray-Rust and H. S. Rzepa. “Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles”. *Journal of Chemical Information and Computer Sciences*, 39(6):928–942, 1999.

[232] P. Murray-Rust and H. S. Rzepa. “Chemical Markup, XML and the World-Wide Web. 2. Information Objects and the CMLDOM”. *Journal of Chemical Information and Computer Sciences*, 41(5):1113–1123, 2001.

[233] P. Murray-Rust and H. S. Rzepa. “Chemical Markup, XML, and the World Wide Web. 4. CML Schema”. *Journal of Chemical Information and Computer Sciences*, 43(3):757–772, 2003.

[234] “Open Babel: The Open Source Chemistry Toolbox”. `http://openbabel.org/`, 2011.

[235] C. Leung and A. Salga. “Enabling WebGL”. In *Proceedings of the 19th international conference on World wide web*, WWW ’10, pages 1369–1370. ACM, New York, NY, USA, 2010.

[236] CC. Gruber and J. Pleiss. “Systematic benchmarking of large molecular dynamics simulations employing GROMACS on massive multiprocessing facilities”. *Journal of Computational Chemistry*, 32 (4):600–606, 2011.

[237] AMD, Dell, and Mellanox. “GROMACS Performance Benchmark and Profiling”. `http://www.hpcadvisorycouncil.com/pdf/GROMACS_Analysis.pdf`, 2009.

[238] University of Florida High Performance Computing Center. “Gromacs Benchmark”. `http://wiki.hpc.ufl.edu/index.php/Gromacs`.

[239] ScalaLife (Scalable Software Services for Life Sciences). “Gromacs”. `http://www.scalalife.eu/book/export/html/227`.

[240] R. Yang, J. Cai, A. Rendell, and V. Ganesh. “Use of Cluster OpenMP with the Gaussian Quantum Chemistry Code: A Preliminary Performance Analysis”. In *Evolving OpenMP in an Age of Extreme Parallelism*, volume 5568 of *Lecture Notes in Computer Science*, pages 53–62. Springer Berlin / Heidelberg, 2009.

[241] The National Center for Supercomputing Applications (NCSA). “Gaussian Benchmark”. `http://include-test.ncsa.illinois.edu/UserInfo/Resources/Software/CHEM/g98bench.html`.

[242] C. Redaelli. “Gaussian03 on NEC SX platforms”. `http://www.maint.ch/redaelli/docs/CSCS-NEC-G03.pdf`, 2003.

[243] C. Redaelli. “Gaussian03 on the CSCS IBM MPP”. `http://www.maint.ch/redaelli/docs/CSCS-MPP-G03.pdf`, 2003.

[244] C. Abdelouahab and B. Abderrahmane. “Comparative Study of the Efficiency of Three Protein-Ligand Docking Programs”. *Journal of Proteomics & Bioinformatics*, (1):161–165, 2008.

[245] B. D. Bursulaya, M. Totrov, R. Abagyan, and C. L. Brooks. “Comparative study of several algorithms for flexible ligand docking”. *Journal of computeraided molecular design*, 17(11):755–763, 2003.

[246] L. Olsen, I. Pettersson, L. Hemmingsen, H. Adolph, and F. S. Jø rgensen. “Docking and scoring of metallo-β-lactamases inhibitors”. *Journal of Computer-Aided Molecular Design*, 18:287–302, 2004.

[247] M. Borcz, R. Kluszczynski, K. Skonieczna, T. Grzybowski, and P. Bala. “Processing the biomedical data on the grid using the UNICORE workflow system”. `https://sites.google.com/site/iwsglife2012/abstract-3`. IWSG-Life 2012.

[248] S. Herres-Pawlis, G. Birkenheuer, A. Brinkmann, S. Gesing, R. Grunzke, R. Jäkel, O. Kohlbacher, J. Krüger, and I. Dos Santos Vieira. “Workflow-enhanced conformational analysis of guanidine zinc complexes via a science gateway”. In *HealthGrid Applications and Technologies Meet Science Gateways for Life Sciences*, volume 175 of *Studies in Health Technology and Informatics*, pages 142–151. IOS Press, 2012.

[249] M. M. Mysinger, M. Carchia, J. J. Irwin, and B. K. Shoichet. “Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking”. *Journal of Medicinal Chemistry*, 55(14):6582–6594, 2012.

[250] “Lead-like subset #1”. `http://zinc.docking.org/db/bysubset/1/`.

[251] John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. “ZINC: A Free Tool to Discover Chemistry for Biology”. *Journal of Chemical Information and Modeling*, 52 (7):1757–1768, 2012.

[252] A. Sonderegger and J. Sauer. “The influence of design aesthetics in usability testing: Effects on user performance and perceived usability”. *Applied Ergonomics*, 41:403–410, 2010.

[253] H. Pienaar. “Design and Development of an Academic Portal”. *Libri*, 53:118–129, 2003.

[254] J. Rubin and D. Chisnell. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. Hoboken: John Wiley & Sons, 2008.

[255] S. Gesing, S. Herres-Pawlis, G. Birkenheuer, A. Brinkmann, R. Grunzke, P. Kacsuk, O. Kohlbacher, M. Kozlovszky, J. Krüger, R. Müller-Pfefferkorn, P. Schäfer, and T. Steinke. “A Science Gateway Getting Ready for Serving the International Molecular Simulation Community”. In *Proceedings of Science*, PoS(EGICF12-EMITC2)050, 2012.

[256] “SCI-BUS (Scientific gateway Based User Support)”. `http://www.sci-bus.eu/`.

[257] “ER-flow - Building a European Research Community through Interoperable Workflows and Data”. `http://www.erflow.eu/`.

[258] “EDGI (European Desktop Grid Initiative)”. `http://edgi-project.eu/`.

[259] S. Gesing, S. Herres-Pawlis, G. Birkenheuer, A. Brinkmann, R. Grunzke, P. Kacsuk, O. Kohlbacher, M. Kozlovszky, J. Krüger, R. Müller-Pfefferkorn, P. Schäfer, and T. Steinke. “The MoSGrid Community - From National to International Scale”. In *EGI Community Forum 2012*, 2012.

[260] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley Longman, 1999.

[261] G. E. Krasner and S. T. Pope. “A Cookbook for Using the Model-View Controller User Interface Paradigm in Smalltalk-80”. *Journal of Object-Oriented Programming*, 1(3):26–49, 1988.

[262] “XNJS IDB Configuration”. `http://unicore.eu/documentation/manuals/unicore6/unicorex/xnjs-idb.html`.

[263] “Guide to XACML security policies”. `http://unicore-dev.zam.kfa-juelich.de/documentation/unicorex-6.4.0-rc1/policies.html`.

[264] “Galaxy”. `https://bitbucket.org/galaxy/galaxy-central`.

[265] S. Gesing, J. van Hemert, P. Kacsuk, and O. Kohlbacher. “Special Issue: Portals for life sciences - Providing intuitive access to bioinformatic tools”. *Concurrency and Computation: Practice and Experience*, 23(3):223–234, 2011.

[266] S. Gesing, I. Marton, G. Birkenheuer, B. Schuller, R. Grunzke, J. Krüger, S. Breuers, D. Blunk, G. Fels, L. Packschies, A. Brinkmann, O. Kohlbacher, and M. Kozlovszky. “Workflow Interoperability in a Grid Portal for Molecular Simulations”. In *Proceedings of the International Workshop on Science Gateways (IWSG10)*, pages 44–48. Consorzio COMETA, 2010.