UNIVERSITEIT VAN AMSTERDAM

MASTER OF SCIENCE IN GRID COMPUTING

Problem Solving Environment for Medical Image Analysis

A thesis submitted to the Board of Examiners in partial fulfillment of the requirement of the degree of Master of Science in Grid Computing

Ketan C Maheshwari

Supervisors:
Sìlvia Olabarriaga and Adam Belloum

Contents

Acknowledgment

I am grateful to my supervisors Silvia Olabarriaga and Adam Belloum for their invaluable guidance. Charl, Jeroen, Johan, Piter T. de Boer. Stanley. opensource developers community. Dhiraj, friends and family.

Abstract

The development of Medical Image Analysis (MIA) applications that can successfully be applied in clinical practice is difficult for several reasons, one of them being the large amount and variety of resources involved (people, data, methods, computing). The application goes through several phases (development, parameter optimization, evaluation and clinical deployment) usually supported by different systems. The lack of support for information flow from phase to phase puts extra logistics burden on the lifecycle of MIA applications. The present report describes efforts to develop a Problem Solving Environment (PSE) for MIA applications using the three systems available at the proof-of-concept environment of the Virtual Laboratory for e-Sciences project. The proposed PSE implements data provenance mechanisms that support information flow among systems, facilitating navigation across phases of the application lifecycle.

The thesis is partly published in and presented at the IEEE Computer Based Medical Systems (CBMS) symposium 2007 as "Problem Solving Environment for Medical Image Analysis" Authors Ketan C Maheshwari, Silvia D Olabarriaga, Charl P Botha, Jeroen G Snel, Johan Alkemade and Adam Belloum

CHAPTER 1

Problem Definition

1.1 Introduction

Medical Image Analysis (MIA) [1] is a field that enables the knowledge extraction through computational post-processing of digital medical images. Processing of digital images using MIA modalities, greatly assist in disease monitoring, diagnosis and preoperative planning. Application of High Performance Computing (HPC) has become an essential part of MIA [2]. Grid Computing is increasingly used as a source of HPC and large scale data storage infrastructure in the hospitals. However, a suitably facilitated computational framework is yet to be fully realized. Such a framework would not only facilitate the effective use of HPC, it would also greatly enhance the user experience and patient comfort in a hospital setup. The present report describes the analysis, design and implementation of an interoperable environment, wherein a series of phases are involved.

A diverse range of software systems are often used by users to perform the clinical research and experiments. Workflow management systems (WfMS)[3] and Problem Solving Environments (PSE) with varying degree of specialization are two such candidates to tackle the requirements in such a scenario. Every system that is used has its own domain of specialties that caters to a subset of these requirements. Ideally an interoperable combination of these systems is needed that facilitates a smooth transition of data and control across the system. However, these systems are not necessarily inherently interoperable. Yet, they are all required together to perform a set of interrelated tasks. This makes it necessary to join these WfMS and/or PSE to avoid technical difficulties and incompatibilities.

1.1.1 Medical Image Analysis (MIA)

Figure 1.1 shows a high level overview of a MIA method. Medical image from an image acquisition modality are sent to the MIA methods for enhancements and the resulting image is sent to the users' workstation for further analysis and diagnosis. PACS (Picture Archiving and Communication System) is a tailormade system at the hospital for image archival and distribution purposes. The box representing MIA in the figure is a high level view of a whole range of MIA subsystems that work in coordination to achieve a specific task. The viewing workstation can also be user's personal computer.

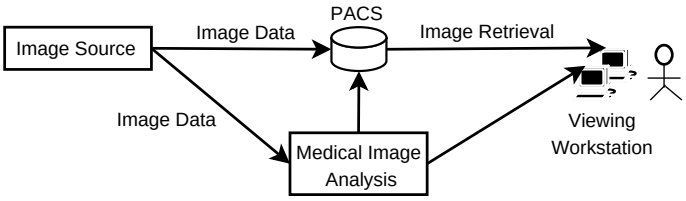


Figure 1.1: General Scheme of an Ideal Medical Image Analysis in a Clinical Setting

1.2 Motivation

This work is motivated by the proposed Medical Image Analysis Support Systems (MIASS) model, as proposed in the article "Integrated Support for Medical Image Analysis Methods: from Development to Clinical Application" by Olabarriaga et. al. in [4]. This section summarizes the MIASS model. MIA applications are compute intensive and time consuming process involving heterogeneous skills and resources. To fully support MIA multiple software systems are needed. There is a need for information organization and management in order to increase the interoperability among these systems. This in turn also accelarates the adaptation of new MIA techniques and tools.

1.2.1 MIA Support System(MIASS)

MIASS is viewed as an ideal environment to provide the necessary technical and human support to the development and deployment of MIA methods in a clinical setup. The vision of MIASS is guided by a set of requirements. It consists of several well defined phases that will be explained in the subsequent sections. MIASS brings in the desired computing and process development system to the MIA methods. A diverse set of users are identified based on their technical background and clinical expertise. The necessity of MIASS is motivated by several factors that include among others (a) a user-friendly interface to HPC infrastructure, (b) a seamless environment that binds together the computing, storage and instrumentation devices within the hospital, (c) a collaborative environment for the users involved.

1.2.2 MIA Development Phases

During its lifecycle, a MIA method evolves through phases. Figure 1.2 depicts the phases of MIASS with the information transmission. A MIASS lifecycle starts with the development of components and network of components. This is followed bychoosingg optimal parameters for the developed network and using statistical as well as qualitative evaluation in the evaluation phase. These phases are developed using the test data. Once the evaluation phase clears the MIASS method, it reaches the clinical routine where it could be used with the clinical data. During any phase the control can shift backwards to the component development in the form of feedback. The purpose of feedback is to enhance and tune the Developed Component in order to suit one or many requirements. Described below is each of these phases.

Component Development/Network Development A new MIA method is developed in this phase. This could be a new basic algorithm or a combination of algorithms to perform a series of computational and analytical tasks. As figure 1.3 (a) shows the component accepts the data as input and performs processing transferring them to the output or a next component. These

Figure 1.5: MIASS UML Usecase Diagram CD=Component Developer, ND=Network Developer, PO=Parameter Optimizer, EU=Evaluation User, CU=Clinical User

3. **Advanced Search** Advanced search is based on a combination of attributes. A logical "or" or "and" combination of the above search options could be provided in the advanced search.

A search returns results that could be handled by the system depending upon the context. A search for run, for instance must enable user to rerun it or update the its attributes. Similarly, a search for an application must enable the user to update the application.

Fetch Run A user should be able to retrieve an existing run from the database and create a new run using existing one. The user must be provided with the information if a different version of the application is compatible with the version used in existing run. A rerun executes in the same manner as the run and same information as the run is stored in the log. A rerun may be executed with exactly the same information as the existing run or it can have change in parameters and/or application version.

an extensible problem solving environment.

1.4.3 Technical Requirements

User Authentications

Extensible Externally activable methods should be available.

Notifications automated, user notification.

Distributed Resource Usage The system should be able to transparently use the available distributed resources for storage and execution. These resources include distributed database and filesystem as well as a grid based computational power. A transparent and stable environment for the storage and automated (as opposed to manual) transfer of large amount of data. The environment should Provide execution control management as per the user requirements.

1.5 Organization of Thesis

The rest of this thesis is organized as follows: Chapter 2 reviews the literature on various aspects of Problem Solving Environments and Workflow Management Systems. Chapter 3 briefly

describes the infrastructure available within the VL-e. Chapter 4 describes the proposed architecture that addresses the problem and its features. Chapter 5 describes the implementation and test cases in order to verify the implementation. Chapter 6 concludes the thesis with concluding remarks and future work.

CHAPTER 2

Literature Review

2.1 Introduction

As more and more science is done with the help of HPC infrastucture (e-science or enhanced Science), the complexity of experiments and the number of required computing tools has increased. These include hardware tools like storage and network devices, instruments and their adapters as well as software tools like Problem Solving Environments(PSE), Workflow Management Systems(WfMS) and an integrated framework that enables the utilization of software and distributed infrastructure in a transparent fashion. This chapter presents a literature review of various software tools and techniques involved in doing e-science. This literature review is motivated by the problem requirements as mentioned in chapter 1. The envisioned environment requires tools, techniques and concepts associated with PSE, their integration and interoperability support. Figure 2.1 shows a partial classification and properties of a PSE. The second

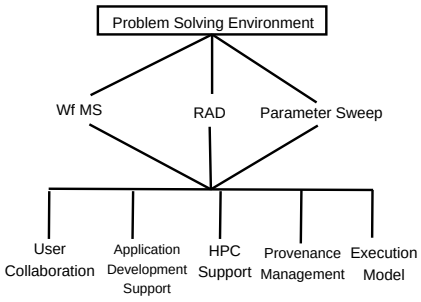


Figure 2.1: Classification of the Problem Solving Environments

level of the hierarchy in Figure 2.1 presents the conceptual classification for the integration and interoperability requirements. User collaboration has increasingly become evident and relavant in large-scale scientific applications [5]. Domain scientists form what is known as virtual organisations in order to collaborate to achieve short or long term goals. Execution Model or Execution Specification describes how the system in question allows user to specify and represent the executable portion of an application. The scope of literature review includes the study of concepts and systems that relate to the classification as shown in figure 2.1. Provenance management,

being a vast area in its own right, is treated as a separate section for literature review. This literature review is motivated by the investigation of possibilities for integration and interoperability of several systems under the hood of a single PSE.

2.2 Problem Solving Environments

Architecture There is no single architecture for a Problem Solving Environment (PSE). The components that make up a PSE depends upon the range of problems a PSE is intended to solve. Hence the architecture differs from situation to situation and depends largely upon the problem domain [6]. However, certain components are must in a PSE using HPC and/or Grid support. Figure 2.2 attempts to capture a set of common components and their role in a PSE and Figure illustrates a "specialized" PSE for computational chemistry. An intuitive user interface that makes the human-computer interaction more productive is a salient feature of PSE. Application Development Tools enable users to develop prototype applications rapidly to solve simple or complex scientific problem at hand. These applications serve as reusable off the shelf components that could reside in a library. Middleware acts as a broker for scheduling and managing the underlying HPC resources. Distributed agents acts a glue to tie the distributed applications and provides auxiliary support eg. provenance, debugging, logging among others. The last layer comprises of the hardware devices and instruments. These devices are controlled and interfaced through the low-level system software.
A PSE is a computational framework that solve a target class of problems, support rapid prototyping and can be used at the frontiers of science [6]. A PSE brings together all the capabilities a given domain of problems might require.
Triana [7] is described as a PSE capable of acting as a complete integrated computing environment for composing, compiling and executing application for a specific problem domain. It is dominantly used as a WfMS in scientific community. The architecture of Triana is based on a third-party interfacing to Grid and p2p components. It is capable of visually developing workflow based applications that can use the underlying grid and/or p2p resources.
It provides a suit of tools to perform computational Chemistry integrated within a PSE. The Ecce

ordinate activities between applications. This provides a strong basis for collaborative work environments. A notification agent serves messages to users as well as interested applications whenever the output of subscribed job changes. LabGrid [8] provides an easy-to-use and potentially collaborative web-browser based environment. Triana does not seem to be supporting any explicit user collaboration.

Application Development Support A wide range of tools could be used to support application development for PSE. The choice of a tool depends upon the requirements the PSE in question is addressing. These tools could be WfMS, RAD tools, Paramsweep applications, compilers, Legacy hardware and software toolkits and libraries. Essentially, they makes up the core components of a given PSE. LabGrid [8] is one such example wherein a range of "Application Packages" combine to form a PSE infrastructure. Triana [8] integrates various tools and technologies in order to achieve a range of functionality under a single user interface. Notable components include java tools, Legacy applications Grid and P2P toolkits. Ecce [9] provides support for python and TCL. It includes various domain specific tools like calculators for chemistry, visualisation tools and a molecule builder application. Both Triana and ecce are extensible, in that additional components could be added to extend their functionality.

Execution Model A PSE could often have more then one execution modalities depending upon the problem at hand. On one hand a simple problem involving a single job could be handled through a command-line job-submission method. On the other hand a complicated problem might need a team of users composing a complex workflow using GUI-based tools for the deployment. Certain problems with intermediate complexity could be solved by a technical expert using an xml-like "meta-language" to compose an execution sequence. Triana [7] can be used as a dataflow or distributed workflow system depending upon the application and user requirements. It has a rich GUI-based execution model. A user can graphically compose the execution scheme using the "box-and-arrow-modality". A box is a representation for processing and an arrow represents flow of data and/or control. LabGrid [8] has no specific execution model. Jobs could be submitted through a web-browser based console. Every job is specified by its associated properties including application package, output filename, portnumber and job specific parameters. The results of this job are written on the specified output file. Ecce uses a combination of globus [10] based globusrun and Resource Specification Language(RSL) [] as its execution specification. For the cases wherein RSL is not suitable, ecce relies upon the

HPC Usage The amount of computation power needed and the data produced is increasing rapidly in most scientific application domains, including the Medical Image Analysis [2]. Acquiring, processing and archiving of large amount of data requires the usage of sophisticated devices and supporting software. Further, the development and testing of application programs in such feilds takes considerable time which could be minimized through HPC. Triana [7] uses the underlying grid and p2p resources through Grid Application Toolkit(GAT) [11] and GAP Jxta [1]. LabGrid [8] uses an existing computing infrastructure at the university including personal computers, routers and other network instruments. This provides sufficiently high distributed computing power to perform the scientific computing. A special job control and monitoring component is responsible for launching and migrating jobs dynamically among the resources. Job submission and scheduling is performed using a number of existing tools including Globus [10], Legion and NetSolve. Ecce includes an advance reservation-based metascheduler called Silver.

[1] http://www.jxta.org

Silver manages a queue of jobs and dispatches the jobs based on a job and target resources' constraints match.

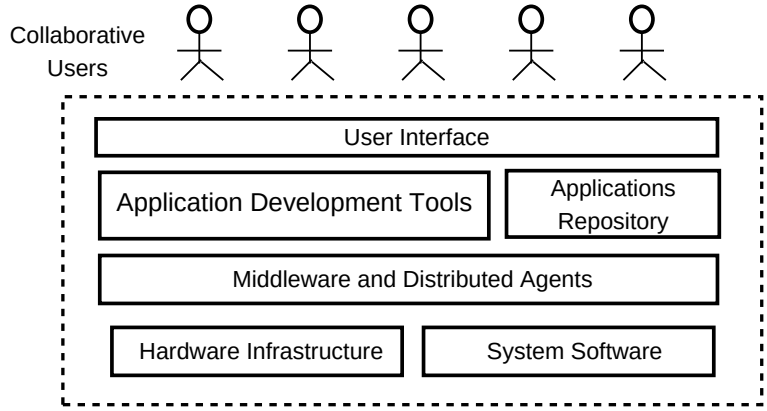


Figure 2.2: A general PSE model

2.2.1 Workflow Management Systems

Architecture The Workflow Management Coalition [3] defines a WfMS as

> The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

The OGSA-WG (Open Grid Services Architecture-Working Group) glossary [12] describes the term work°ow as closely associated with choreographyand orchestration and defines it as

> Workflow is a pattern of business process interaction, not necessarily corresponding to a fixed set of business processes. All such interactions may be between services residing within a single data center or across a range of different platforms and implementations anywhere.

In simplest terms, a workflow could be treated as a "program" at higher level of abstraction intended to run on a specific execution-enabled environment [13]. A workflow should be able to support common "patterns" [14], including sequential, repetitive and conditional execution. Figure 2.3 shows an architectural model of a generic WfMS as proposed by Workflow Management Coalition (WfMC). This acts as a common reference model for WfMS products. A set of workflow engines lie at core of this architecture. These engines communicate to the external world including other workflow systems through several dedicated interfaces. The grid-based WfMS architecture proposed by Yu and Buyya [15] as shown in Figure 2.4 is based on the WfMS model. An important feature of this model is the classification of functions into build-time and run-time functions. The build-time functions are concerned with describing tasks and dependencies while the run-time functions are concerned with executing and monitoring of these tasks. The model replaces the normal computing resources with grid resources and a grid middleware layer to manage these resources. Grid-flow is a petri-net based WfMS. The architectural model is depicted in Figure. The Grid-Flow Description language acts as an interface between the graphics based petr-net interface and the grid-flow engine. The data and program integration layer manages the

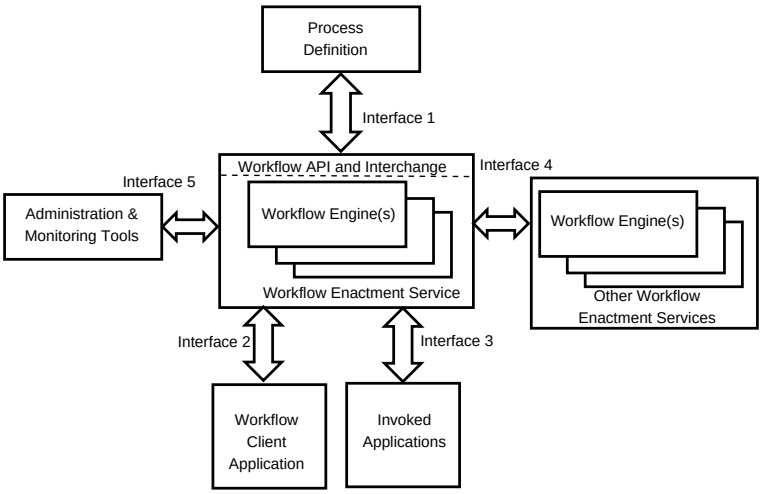


Figure 2.3: A general WfMS model [3]

underlying distributed resources. The program integration layer works on top of the WebRun system, a unified platform that invokes remote programs using grid computing modality. A grid-flow repository helps in storing a number of compiled programs, meta-data, system property info and other necessary context information in order to run the applications.

Krishnan et. al. [16] proposes an OGSA compliant grid-based workflow framework called Grid-Services Flow Language or GSFL. GSFL is based on a set of grid services. It has the following important features:

- Service Providers is a registry of the services taking part in the workflow.
- Activity Model is a list of all operations belonging to the individual service providers.
- Composition Model describes the composition of grid services into a new service.
- Lifecycle Model is concerned with the runtime activities of the services.

Kepler [5] is a scientific workflow management system built on top of the PtolemyII (http://ptolemy.eecs.berkeley.edu/ptolemyII) system. Webservice extensibility is one of the highlights of Kepler. The "actor" modality of Kepler allows to create actors representing webservices/grid services hence facilitating an extension to the existing functionalities. "Box and pipe" model of user interface makes it easy to use for scientific workflows. Planning for Execution in Grids or Pegasus [17, 18, 19] is a scientific workflow management system that maps complex and abstract scientific workflow specification onto the grid resources. A user can create an abstract workflow using Chimera [20] and Directed Acyclic Graph (DAG) tools. Chimera is a "virtual data system" that could be used for the generation and management of derived data in scientific experiments. This abstract workflow is submitted to the Pegasus engine. Pegasus, in turn cosults the grid based resource brokers to locate apropriate resources, data and applications to execute the workflow tasks.

Taverna [21] from the $^{my}Grid$[2] project is webservice based tool to create and execute workflows in the field of lifesciences. A workflow is formed of tasks or "processors" connected with each other in a flow represented by arrows. A processor can consume multiple input data and produce

[2]http://www.mygrid.org.uk

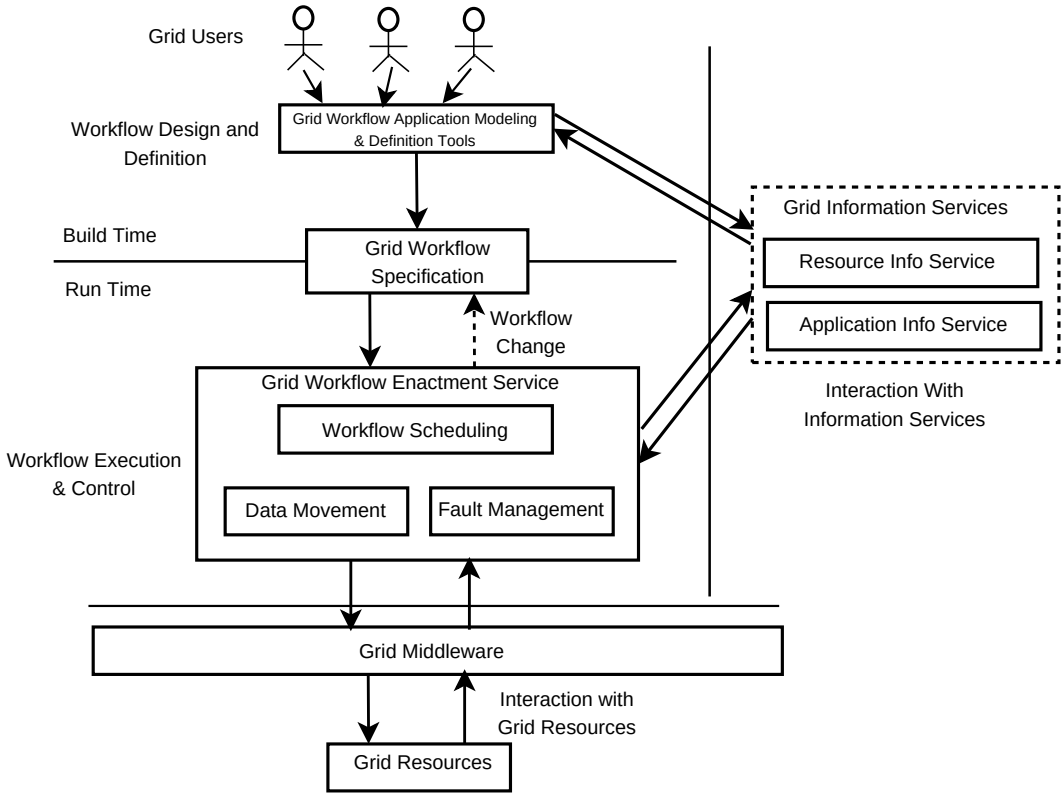


Figure 2.4: A grid based WfMS model [15]

multiple output data. The workflows are written in a new, xml-based conceptual language called Simple conceptual unified flow language (Scufl).

Karajan [22] is a derivative of gridant [23]. It is an engine as well as workflow language that supports a number of workflow patterns [14]. Karajan is scalable, parallel and extensible language that can be used to manage a large number of tasks as a workflow.

User Collaboration Modern science is an era of collaborative work. With the emergence of sophisticated computing infrastructure, more and more science is done across organisations in a collaborative fashion. Workflows becomes pivotal in sharing resources, data, applications and knowledge, hence fuelling collaborative environment [24]. A workflow may be considered as a societal entity supporting work coordination. One of the important component for a workflow enactment engine is a human control interface [13]. Its functionality include supervision of the overall execution, distribution of the results and manage collaborative activities. Collaboration in WfMS is about providing the following:

- Co-editing of the workflow specification.
- Simultaneous reciept of the results of execution.

Currently, no WfMS seems to provide a fully collaborative environment. However, several WfMS provide a semi-collaborative environment. Most WfMS provide an automated offline-sharing and reusability mechanisms at the workflow as well as resource level. Taverna enables the sharing

of resources through webservices. These shareable services are used by the bioinformatics community. Non-sharable scripts could also be used for individual purposes using Taverna. GSFL incorporates a serviceprovider modality implemented as a list of services identified by a unique name. These services are used as working units in GSFL. Users can use control,data and notification models as described in section 2.2.1, in order to communicate with processes as well as other users. Communication in GSFL includes control messages as well as large amount of data. The Grid Workflow Definition Language(GFDL) proposed in [24] is a translatable language by design with most of the scripting workflow languages. This enables construction of a translator an easy and mechanical task. This makes an integration and collaboration with other workflow systems and hence users can become potentially easy. GridANT and Karajan doesnt provide any specific features for collaboration per se, however they could be used collaboratively through their underlying service based modality. Users can use shareable grid services in the form of workflow tasks.

Application Development Support Most WfMS provides a minimal set of auxiliary or integrated tools for application development. Depending upon the problem domain, they might or might not be sufficient. This calls for an external development environment suitable to the user needs along with the workflow system. Several systems provide mechanisms for saving and reuse of previously created workflow and legacy applications. These repositories, often called "application factories" [25] acts as a collection of invocable applications through a registry. User may compose new workflows using a combination of these applications with appropriate parameters. Kepler [5] provides some support through [write more ...]. Other workflow systems does not seem to provide any significant application development support with an exception of gridAnt, which provides an application factory construct.

Execution Model Workflow definitions for large-scale problems may get reasonably complex [26]. Such complexity calls for a sufficiently usable execution model. A workflow execution model can be as sophisticated as an intuitive box-and-pipe network or can be as raw as a set of customised scripting statements. At times the execution model might be difficult to learn and apply. Thus requiring user-training for more then simplest of problems. XML and XML-based vocabularies are most popular execution specification models for workflow systems. Business Process Execution Language is increasingly becoming popular for the representation of scientific workflows[26]. BPEL is Based on Service Oriented Architecture (SOA) wherein, every process acts as a service. The execution specification is based on the XML schema. This schema is responsible for the typed variable system and name resolution through namespace management. It provides constructs for activities such as variable declaration, service invocation and SOAP message reciept, dataflow and controlflow. BPEL can maintain and communicate temporary data structures to other services during execution.

2.2.2 Rapid Application Development Environment

Architecture Scientists that develop applications need a very high level of abstraction. This motivates a need for rapid prototyping and the reuse of existing components in an easy fashion. Java CoG kit [27, 28] is a Rapid Application Development environment for Computational Grids. Talisman [29] is a Rapid Application Development component of the $^{my}Grid$ project.

2.2.3 Parameter Sweep Environment

Architecture AppLeS or the Application Level Scedular is a parameter sweep system that takes into account the system and application requirements in order to adaptively schedule the application jobs. It attempts to exploit the embarrasingly parallel constructs within the application and deploys them on the grid.

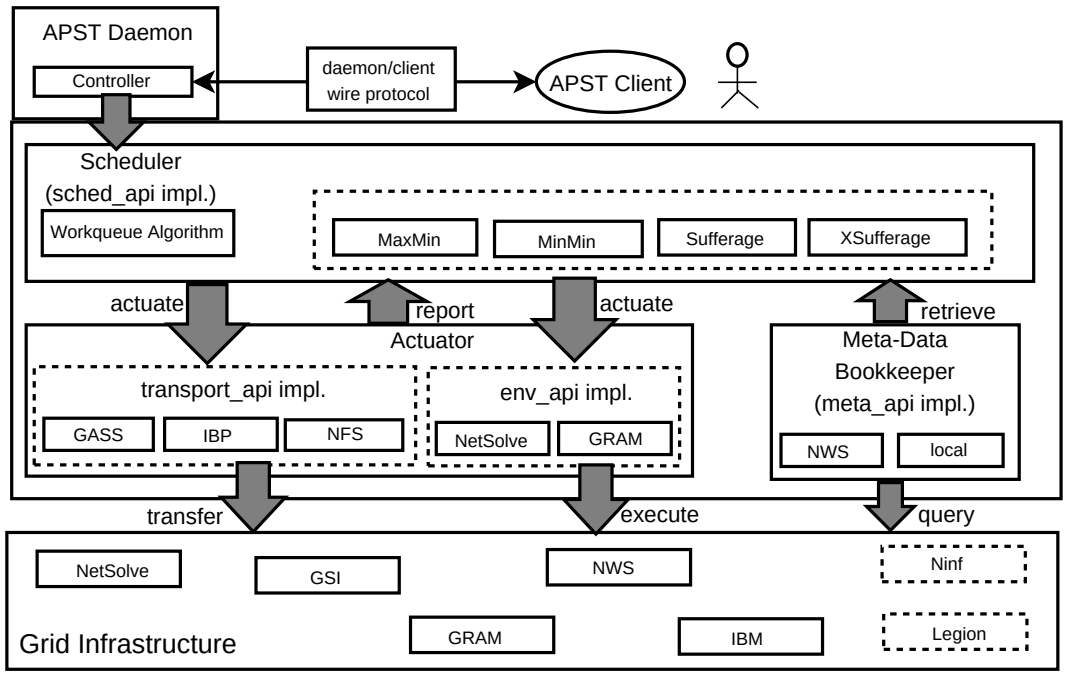


Figure 2.5: The AppLeS parameter sweep model [30]

Application Development Support AppLeS supports the development of large scale parameter sweep applications through the what is called the AppLeS Parameter Sweep Template or APST.

HPC Usage AppLeS uses the underlying grid resources in an optimal way by enabling colocation of data by parallel jobs.

CHAPTER 3

Available Infrastructure

3.1 Introduction

As a part of this research, it is desired to leverage the available infrastructure within the Virtual Laboratory for e-science (VL-e project). This chapter describes the available infrastructure and their working in brief.

3.2 Resources Infrastructure

The Storage Resource Broker (SRB) The Storage Resource Broker (SRB) [31] is the large-scale storage facility hosted by the SARA SARA Computing and Networking Services and the Dutch Institute for Nuclear Physics and High Energy Physics (NIKHEF) at Amsterdam.

The VL-e Proof-of-Concept Environment(VL-e PoC) The Vl-e Proof of Concept (VL-e PoC) environment is a common, shared hardware and software support platform within the VL-e project. Various tools are available to scientists to run scientific applications.

The VL-e Toolkit (vlet) The VL-e Toolkit is a software utility developed within the VL-e project in order to ease the usage of virtual resources available in a transparent and user-friendly manner by aggregating the resources and presenting them to the users via a single graphical interface.

The Adaptive Information Disclosure (AID) Services AID services can be used to support provenance. The annotation tool is probably the most relevant service to your problem. It currently works as a firefox plugin that let's you 'annotate' the URL being viewed with either established jargon (e.g. imported as OWL into the repository) or with ad hoc tags. AID is a generic toolkit developed by the AID group, aimed at groups of knowledge workers that cooperatively search, annotate, interpret, and enrich large collections of heterogeneous documents from disparate locations. It consists of a set of services for metadata, learning, and storage and retrieval. A few example clients are provided such as an annotation client, query construction client, and a named-entity recognition client that show how the services can be applied in practical applications. The services perform the following tasks: text indexing, text statistics, meta-data storage and querying, thesaurus reasoning, annotation, text retrieval, spelling correction, synonym detection, and model learning. The AID toolkit is based on web services and W3C standards such

as OWL, RDF(S), and SKOS, and as well as other (mostly) open source technologies such as SRB, Lucene, Sesame, and Jena. The AID components are available as web services in the PoC environment.

3.3 Legacy Systems

This section describes the legacy systems used for this work: DeVIDE for the development, Nimrod for optimization (parameter sweeps) and evaluation (image sweeps), and the AMC-DWMS for clinical deployment.

3.3.1 DeVIDE

DeVIDE or the Delft Visualisation and Image Processing Development Environment is a platform independent software system that facilitates for the rapid creation, testing and application of modular image processing and visualisation algorithm implementations[32]. DeVIDE provides, what is called a modular development environment, wherein, the modules provide a visual input/output and connectivity modality, through which, a pipeline or 'network' could be developed. This network accepts an image and optionally several parameters and performs the series of operations as specified by the network modules. Every module in DeVIDE has its own 'properties' that control its runtime behavior. DeVIDE facilitates to interactively alter the values of module properties. The network hence developed could also be used as a new third-party module by another application. A GUI for visualising and interacting with the processed image data makes the experimentation with alternative algorithms faster and effective.

VTK and ITK Support The Visualization ToolKit (VTK)[33] is an open source, freely available software system for 3D computer graphics, image processing, and visualization. VTK consists of a C++ class library, and several interpreted interface layers including Tcl/Tk, Java, and Python. The National Library of Medicine Insight Segmentation and Registration Toolkit (ITK)[34] is an open-source software system for medical image processing in two, three and more dimensions. Both VTK and ITK are application libraries, and provide a potentially powerfule set of functionality through wrapped usage from the Python environment. DeVIDE integrates all functionality in both of these libraries and so makes all this functionality available through a data-flow application builder architectural model. It is possible to experiment with any combination of VTK and ITK elements at runtime. Effectively, one can experiment not only with parameters and input data, but also with different code paths.

DeVIDE Environment Figure 3.1 shows a snapshot of the DeVIDE environment. Shown in the figure is a visual interface to a simple DeVIDE network consisting of three modules. A DicomRDR module provides the image slices from the Dicom data files it reads as input. DicomRDR sends the output to the resampleImage module where the sampling rate could be adjusted by modifying the spatial parameters. Finally, the resampled output data goes as input to the slice3DViewer module. A user can activate/execute the network and visualise the image through slice2DViewer. Figure 3.1 shows a snapshot of the image as visualised through the slice3DViewer module. The interface also provides an interactive facility using which a user can navigate through the image in four dimansions using standard I/O devices such as a mice and keyboard.

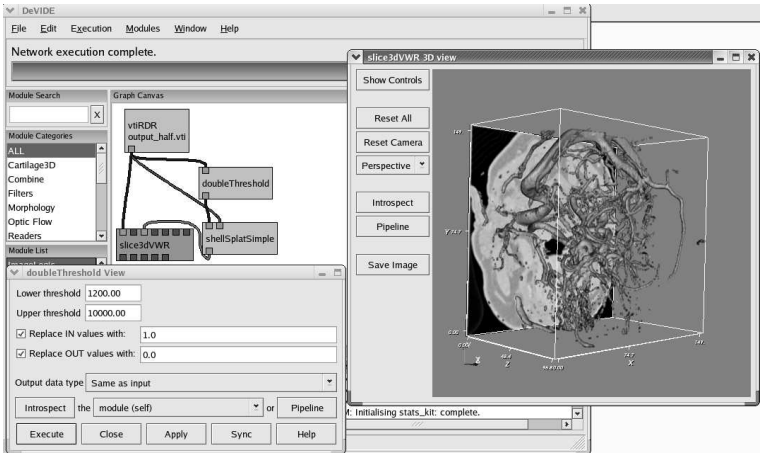


Figure 3.1: DeVIDE network in interactive mode during the development phase.

3.3.2 AMC-DWMS

The AMC-DWMS [35] or the Acedamisch Medical Centrum Distributed Workflow Management System is a software framework developed at AMC for the deployment of the custom MIA applications in a clinical setting. The system provides logistics facilities to support various clinical activities. Based on a Service Oriented Architecture(SOA), AMC-DWMS uses loosely coupled modules specialized to perform a specific task. A workflow is specified using a specialized XML vocabulory as shown in figure 3.2. These 'templates' are composed of logical units that correspond to various activities within the workflow. The system supports image import-export, caching, analysis and notification services among others. A relational Database is used to store and retrieve the information related to workflow actions. A semi-graphical interface, called 'control-center' is used to install, configure, update, synchronize and monitor module activities. The system is specialized in handling DICOM server data, a proprietary image format and protocol.

3.3.3 Nimrod

Nimrod/G is a High Throughput Computing environment that performs job management and resource scheduling based on a cost and time optimization strategy. It provides a queue management mechanism; a job submission system to communicate with other resources, and an integration facility with resource management entity. In this way the resources are chosen from a resource pool for individual jobs hence the jobfarming. The job is wrapped in a software container called nimrod agent that contains job scheduling and dispatch specific information. Hence one agent per node per job makes all job independent of each other. Nimrod/G performs these activities with the help of a batch-like plan file submitted to its parametric engine(see figure 3.3). It sends out a number of parameter sweep calculations at the start of the execution of the tasks. Next, these calculations are executed in parallel on available resources. When finished, the calculation results are collected back from the resources and sent back to the client. During the execution of these tasks, the client computer only needs to dispatch and collect the jobs to and from alternative nodes in accordance with the commands on a plan file. A sample plan file is shown in figure 3.3. Here a python script called 'tangent.py' is executed across a range of 1 to 3 for four parameters hence forming 81 independent jobs. In job farming, each node and,

```
<unit xsi:type="xwf:DicomSeries" description="CTA Blanco">
<dicomTagCondition>
<tag tagId="0" description="Acquisition Protocol">
<group>24</group><element>4144</element>
<regEx>MMBE j CTA j CAROTIDEN j WILLIS</regEx>
</tag>
<tag tagId="1" description="Manufacturer Model">
<group>8</group><element>4240</element>
<regEx>MX8000</regEx>
</tag>
<tag tagId="2" description="Contrast Bolus Agent">
<group>24</group><element>16</element>
<regEx>CONTRAST</regEx>
</tag>
<boolEx>tag(0) & tag(1) & !tag(2)</boolEx>
</dicomTagCondition>
</unit>
```

Figure 3.2: Example of an XML used as workflow specification with DWMS.

therefore, each analysis runs independent of all other nodes. This method is particularly suited to examining large parameter spaces.

```
parameter THRESH integer range \
from 1200 to 1600 step 1;
task main                                          stage-in
   copy ./rundevide.sh node:.
   copy ./dev_nw.dvn node:.
   copy ./img_in.vti node:.

   node: execute ./rundevide.sh $THRESH img_in.vti \
   ./test_network.dvn img_out.vti
                                                   stage-out
   copy node: stdout stdout.${THRESH}
   copy node: stderr stderr.${THRESH}
   copy node: dev_nw.dvn dev_nw${THRESH}.dvn
   copy node: img_out.vti img_out${THRESH}.vti

  endtask
```

Figure 3.3: Example of a nimrod plan file.

CHAPTER 4

Proposed Architecture

4.1 Introduction

Architectural framework that support the MIA lifecycle must provide a way so as to make a diverse set of WfMS and PSE work together. In order to support new and unforeseen tools the architecture must be extensible. Further, it should be able to make use of the underlying grid resources.

The proposed architecture is a component based approach where-in customized and glued interfaces of existing systems talk to each other in a predefined way.

4.2 System Architecture

The proposed architecture is illustrated in 4.1. It supports the information flow among different application lifecycle phases via data provenance mechanisms [36]. Three layers (User Interface, Web-Services and Distributed Resource) along with legacy systems (AMC-DWMS, Nimrod and DeVIDE) form a framework for managing and organizing the information associated with a MIA application.

User Interface Layer The user interface layer provides command-line and web-browser based interaction with the system.

Grid-services Layer The grid-services layer consists of Application Description, Data Provenance, Project and Application Execution management services. These services are invoked directly by the user or through the stubs extended from the legacy systems. Application Description services provide a mechanism to describe, store, access and retrieve MIA applications and their versions. Application Execution services are used to store information that needs to be tracked from phase to phase, including applications, parameters, image data, and standard logs. Project management services facilitate the creation, access, subscription and cleaning up of projects. Using the Data Provenance services, the user can retrieve provenance data to navigate back during the MIA lifecycle. The complete settings used to generate a given result can be restored and used to run the application interactively for diagnosis and backtracking purposes. After sufficient testing, The application is saved in the Application Registry using the Application Description services.

Resources Interface The Virtual File System (VFS) layer abstracts the database. Distributed resources are managed through vlet API. The postgreSQL database is accessed through the Java Database Connectivity API through a database connection pool.

Resource Layer Distributed storage resources available within the VL-e are used to store persistence data. The data and references are controlled through PostgreSQL database. SRB, gridFTP and local file systems form the data storage. These are transparent to the user and accessed through URLs.

Legacy Systems The legacy systems perform application execution operations. These executions are recorded in the system through service calls placed in the service-call stubs. These stubs are service call programs inserted into the legacy systems.

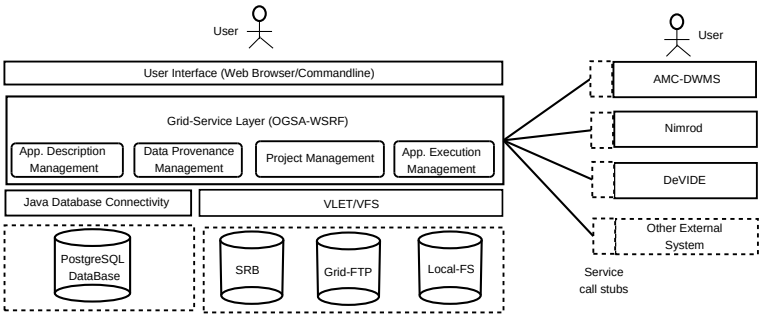


Figure 4.1: Architecture

4.2.1 Architecture Features

Passive The architecture supports "passive" information management. Its passive in the sense that the information is stored and retrieved by the external systems and users after the events happen. This enables a long term management and diagnosis of activities along the MIA lifecycle.

User-based Authentication Users associated with the system can only access and manipulate the information. This is achieved with user based authentication mechanism. The system allows only the users with valid credentials stored in a service container.

Scalable The proposed architecture is incremental systems scalable. This means that since independent calls to webservices generate a new instance any number of new systems does not affect the performance of the architecture.

4.3 User Interaction

Figure 4.2 shows a typical user's interaction with the system. The horizontal boxes show the APIs and the vertical lines show their lifetime. The horizontal arrows connecting these lines show the users interaction with these APIs. The labels on these arrows represent the function calls that the user makes to the APIs. Each vertical rectangle represents the life-time of the function call.

4.4 Modeling Concepts

The concepts as described in the requirements in chapter 1 are modeled using the Relational Database Management System (RDBMS)(figure 4.3). The database stores information in the form of values and references to the remote resources. These references are stored in the form of url links pointing to the virtual file system. The system maintains entity and referential integrity through primary and foreign-key relations respectively.

Entities and Relationships

Advantages of RDBMS Since the provenance requirements of the system are predefined and constant a relatively rigid and fast information design is best suited. RDBMS is a proven and robust approach to management of a database. Predefined associations among entities in RDBMS lends itself into intuitive and hence stable design. Retrieval of data from an RDBMS is computationally cheap, flexible and quick.

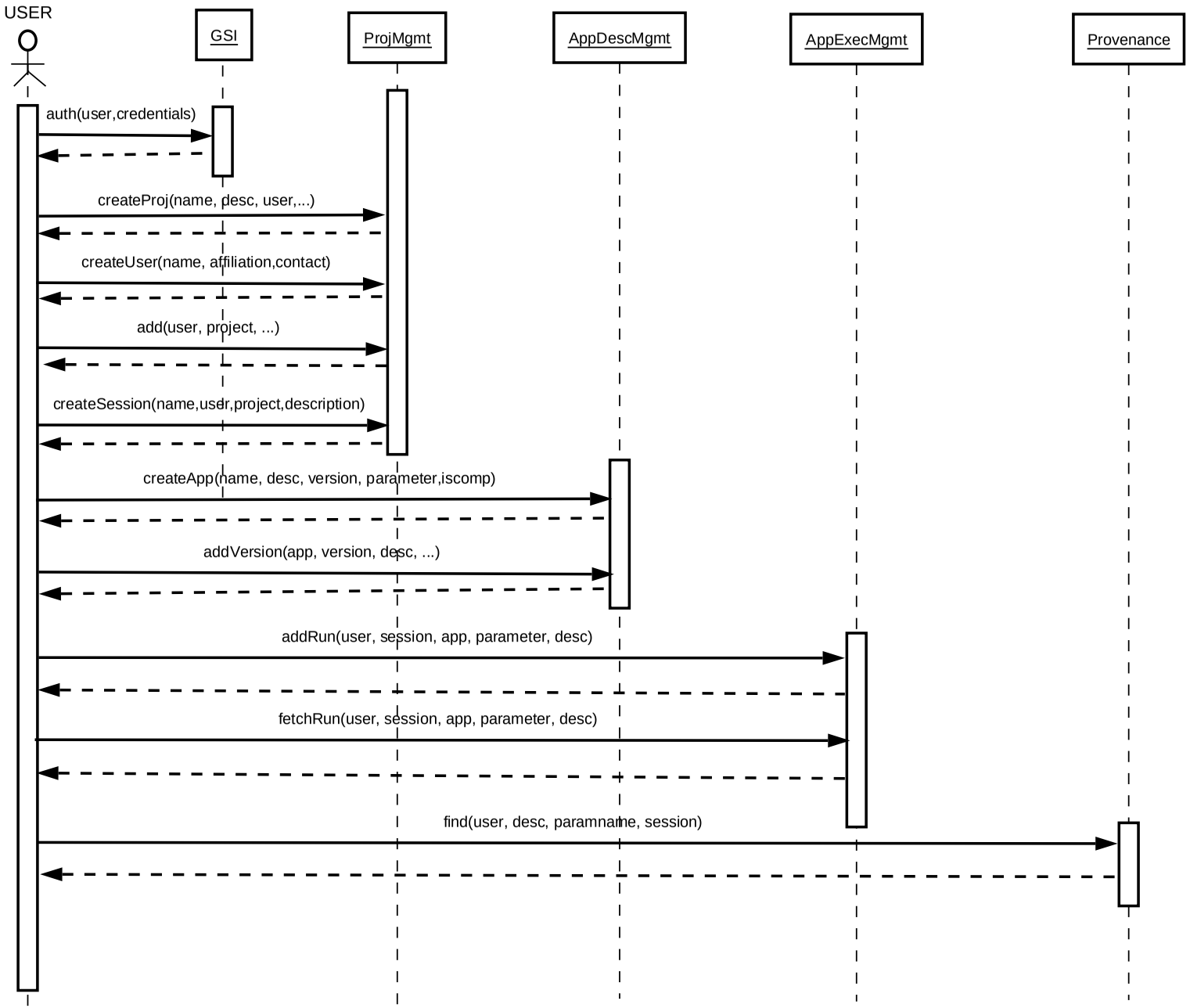


Figure 4.2: Sequence Diagram depicting the user's interaction with the system

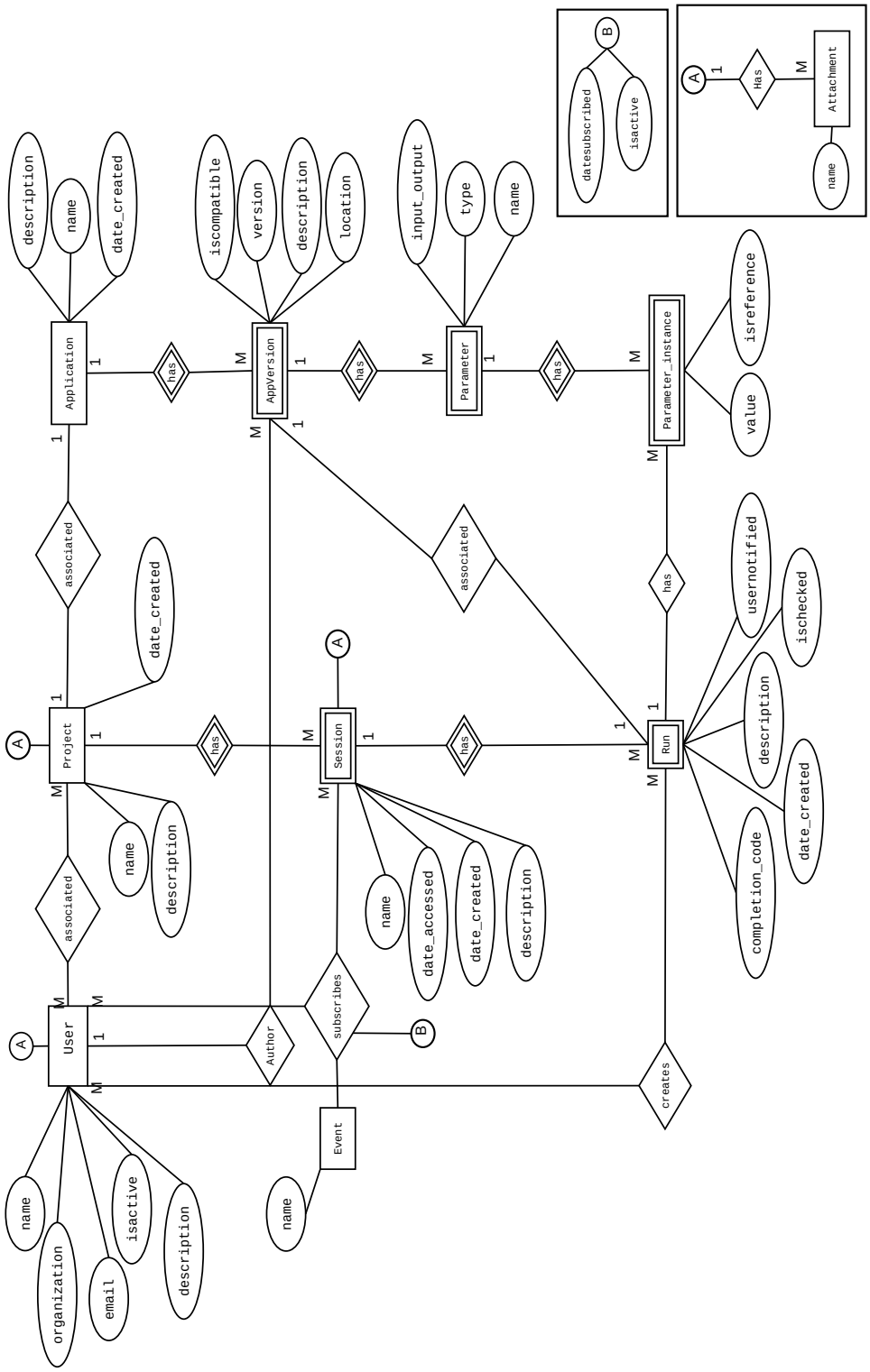


Figure 4.3: Entity Relationship Model for the MIA Information System

CHAPTER 5

Implementation and Test

5.1 Introduction

This chapter describes the implementation of the architecture proposed in the last chapter. The implementation was done with a bottom-up approach implementing the methods then wrapping them into grid-enabled grid-services. The section 5.3 describes the functions as they were implemented. The section 5.4 desribes the implementation of grid-services.

5.2 APIs and Functions

Each of the APIs as described in chapter 4 contains several functions each performing a specific task. As shown in Table api-functions.

5.3 Functional Specification

Table 5.1 shows the mapping of APIs into functions. These functions are implemented as part of APIs and wrapped into the grid-service. Several functions are overloaded with a similar signature but applying on different entities. Following is a brief description of these generic functions.

[2] **Create**: Create functions create an instance of the entity in question. This involves setting up of values of the properties of that entity and recording them into the database for future retrieval.

API	Functions
Application Description Management	Register-Application, Add-Version
Application Execution Management	AddRun, FetchRun
Data Provenance Management	Find
Project Management	create/delete [project,session,user], subscribe/unsubscribe user

Table 5.1: APIs and Functions

- **Update**: Update functions alter the values of one or more properties. Update method is either triggered by the user or triggered automatically based on events.
- **Delete**: Delete functions are used to delete an entity.
- **Get**: Get functions are used to locate the id of an entity given one or more properties associated with it. It returns more then one entities if they match with the provided parameters.
- **Find** The provenance related requirements are addressed through the find methods. Find Performs a search based on the criteria given by the user. A basic form of find performs searching of entities and an advanced form performs a search on the associations among entities.

Apart from the above mentioned generic functions, several other functions were implemented. Following list briefly describes the key functions:

- **add-run** This function performs the recording of an execution into the system. Add-Run is a batch mode function. The information recorded as part of an addrun function is the user, project and session associated with the run and parameters and user notification status for the outcome of this run.
- **fetch-run** As the name suggests a Fetch-Run action performs fetching of information related to the run to a local computer. The associated parameters are also downloaded from the remote locations. This information is fetched into a 'run-profile' file and includes information about associated user, project and session and parameters. Remotely located files associated with this run are downloaded into a designated directory of the user.
- **Register-application** This method registers the application into the system. The application could be retrieved at a later time. Further, additional versions of the application can also be registered with the application.
- **user-subscribe**This method enables a user to be subscribed to a project and session on specific events. Subscription information is used to notify users via email when the events occurs.
- **exec-query** This method provides a user friendly facility to submit a free query to the system.

5.4 Grid-Service Implementation

A factory-insance pattern based Globus 4 complaint grid-service was implementated. Figure 5.1 depicts the invocation mechanism for the service. An authenticated client invokes the factory service. The factory service in turn creates an instance of the service. The instance serves the service call and is destroyed by the container at the end of the call.

5.5 Experimental Setup

The system was tested with a representative case that demonstrates a MIA lifecycle over a time period. An experimental setup was implemented to realize all phases of a simple MIA application using the proposed PSE. Both Nimrod and AMC-DWMS run DeVIDE networks in

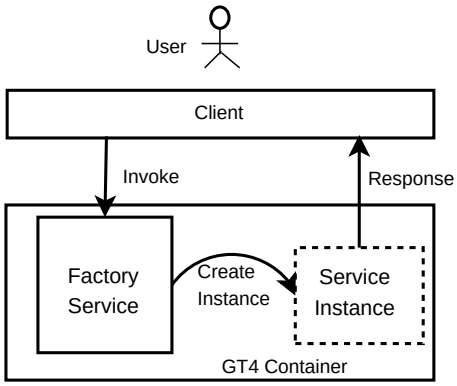


Figure 5.1: A Factory-instance pattern grid-service invocation

command-line mode. Before running the DeVIDE application, the data and DeVIDE network are staged in according to the Nimrod plan or the AMC-DWMS workflow. When the application is completed, the output results are staged back. Currently no real data provenance mechanisms are implemented; instead, all data (input/output images), standard log files (stdout,stderr) and DeVIDE networks are stored in the local file system. At a later stage, the user can manually retrieve data from the saved files and restore the interactive DeVIDE application with the same settings.

Example Application. A region-of-interest (ROI) is segmented from an image using a threshold operation. A DeVIDE network imports the image and a threshold range (TR) as parameters, and outputs a binary mask for the voxels within the chosen TR. Different ROIs can be selected by choosing a proper (optimal) TR. Here we focus on segmentation of blood vessels from contrast-enhanced CT Angiography scans of the head.

Development. A DeVIDE network was developed to implement the segmentation application using available components in DeVIDE's library. The network contains a VTK image reader, a 'DoubleThreshold' image processing operator, a custom 'Slice3DViewer' and a VTK image writer. During development, the user runs DeVIDE interactively, setting the TR on the GUI and observing the generated result in a viewer.

Optimization/Evaluation. Nimrod was used to investigate the optimal TR to segment blood vessels. A parameter sweep on the upper threshold value was performed, while keeping the lower threshold fixed (1200 HU). somefig-left shows the Nimrod plan for varying the threshold between 1200 HU and 1600 HU. The data and executables are staged-in, DeVIDE is executed in command-line mode, providing the image, parameters and the network as arguments, and the output data is staged-out. The resulting images are visually inspected by an expert to determine the optimal threshold value to segment blood vessels (i.e. 1210 HU). In a similar fashion, an image sweep was performed with Nimrod to run the method with the optimal settings for many images.

Clinical Deployment. AMC-DWMS was used to insert this application into a simulated clinical setting. The complete workflow consists of the following tasks (see right): import the CT scan (DICOM) data from a server, convert it to the appropriate VTK data format, start the DeVIDE

application with the optimal parameters, convert output from VTK to DICOM format, and export results to a DICOM node.

Feedback in Lifecycle. During the inspection of results produced during optimization, evaluation or deployment, a medical or technical expert can detect problems (e.g., degraded output, algorithmic instability). Using the information saved in the log files, it is possible to reload the DeVIDE network in interactive mode with the input image and parameters that generated the problem. The user can then add more components to the network (e.g. other 3D viewers) to comfortably investigate the situation at hand.

5.6 Test Cases

Application Registration Application Registration action (figure 5.2) interactively performs the registering of a new application into the system. The system records the author of the application and application creation time. A new version can be added to an existing application and registered into the system. The system keeps information about the version number, compatibility and author of the version.

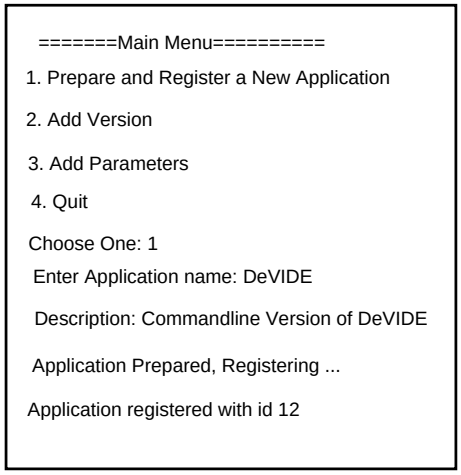


Figure 5.2: Application Registration

User Subscription A user can subscribe to a project and session (figure 5.3). A subscribed user is subject to recieve notifications via email based on the events he has been subscribed to the project and session. Common examples of events are OK, ERROR, ABORT meaning a normal execution, an error in execution and execution abnormally terminated respectively.

Add-Run User can invoke an add-run method using a commandline in batchmode. The addrun method accepts information related to the run as shown in figure 5.4 and registers the run into the system. In this case, a DeVIDE application running on a VTI image to obtain a region of

```
====Main Menu=====
1. Create Project
2. ...
.......
6. Add user to Session

Choose one: 6

Enter User name: Ketan
Enter User Description: Programmer
Enter Session name: devel
Enter Session description: Development Session
Enter event for subscription:(OK/ABORT/ERROR) ERROR
Is the user active(true/false): true
Adding user to session ... done.
```

Figure 5.3: User Subscription

interest is added by user with username “ketan” and as a part of “optimization” session. The ’ftrue’ option with the parameters specifies to save the parameters to a local folder.

```
add-run \
--user name=ketan \
--project name=ROI \
--session name=optimization \
--app name=rundevide.sh, version=1.0 \
--param threshold;1100;ffalse,infile;./image-in.vti;ftrue, \
network;./testnetwork.dvn;ftrue;outfile;./image-out.vti;ftrue \
--status OK \
--notify true \
```

Figure 5.4: Add Run

Fetch-Run A registered run can be fetched through the interactive fetch-run. As figure 5.5 shows, the fetched run is saved in a ’profile’ file marked with the id of the run fetched. The profile contains relavant information about the run downloaded and path to the associated referenced parameters downloaded. In this case the parameters associated with the run would be downloaded to the “/tmp” folder of local machine. In the case where multiple hits are found the system returns an error message.

Find The interactive find method is used to address the provenance requirements. A basic form of find performs the search on the entities. An advanced form of find performs the search on the associations of the entities. Several options for find are available. For example a user could be

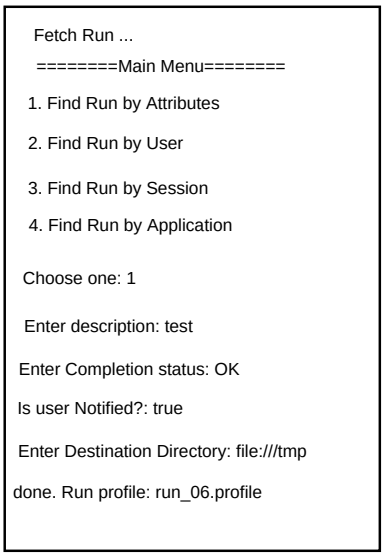
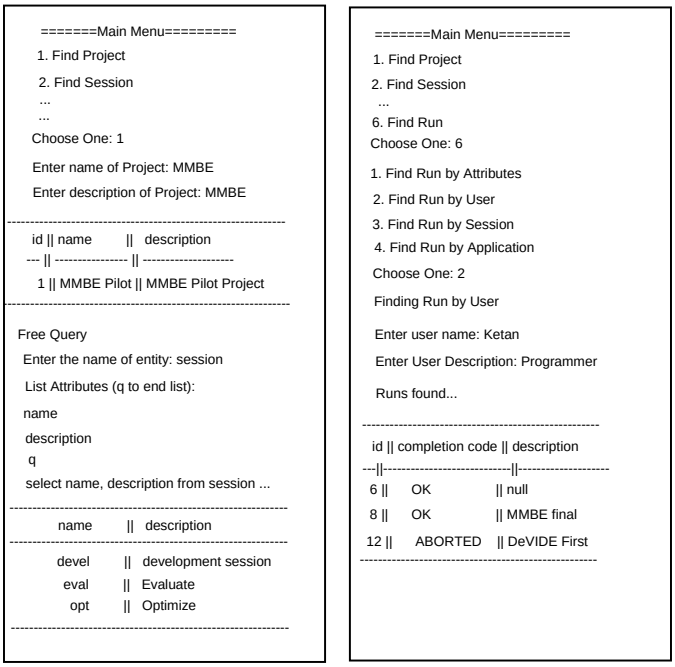


Figure 5.5: Fetch Run

found based on the runs he is responsible for or the application he has authored. As figure 5.6 demonstrates, the user finds a project with name and description followed by a free query on sessions. As shown in figure 5.6 b) one can find a run based on the users who executed it.



a) Basic Find

b) Advanced Find

Figure 5.6: Basic and Advanced form of Find

CHAPTER 6

Conclusions and Outlook

6.1 Conclusions

6.2 Outlook

We proposed a general architecture of a PSE to facilitate the information flow among systems supporting different phases of a MIA application lifecycle in the VL-e project. The PSE is composed of existing systems and a mechanism to track provenance data that can be used to navigate across phases. Information is saved to enable invoking the application in DeVIDE interactive mode at a later stage, when errors are detected during optimization, evaluation or deployment.

Several practical challenges had to be faced initially. Currently all systems work on compatible platforms, and a simple logging mechanism enables information flow among them. With the prototype and simple application presented in this paper, we have demonstrated that it is possible to construct a PSE that assists across all phases of MIA development for clinical applications. A proposed algorithm can be interactively developed in a rich graphical environment, then seamlessly transported through parameter optimization and evaluation phases, and finally deployed in clinical practice, all using a unified set of tools. At any stage return to a previous phase in the lifecycle is possible, with the prospect of facilitating the maintenance of MIA applications that meet the standards of health care.

The implementation described here shows a minimal setup in which the systems were adapted to run in the same platform following the requirements of the VL-e PoC. Using this experimental set-up, we now focus on the supporting data provenance mechanisms, which are essential for the design and implementation for the proposed PSE. Our future work concentrates on the investigation of extensive provenance requirements and implementation of provenance mechanism using standard practices.

APPENDIX A

Choice of Technology

java As is the programming platform used for vlet.

jpa and jakarta dbcp persistence architecture, jakarta database connection pooling

postgresql database, why this? what purpose?

javamail and jndi

globus and gridftp

vfs

Bibliography

[1] A. Dhawan, Introduction to Medical Image Analysis. Wiley Interscience, 2003.

[2] R. Kikinis, S. K. Warfield, and C.-F. Westin, "High performance computing in medical image analysis at the surgical planning laboratory," in High Performance Computing Asia'98, (Singapore), pp. 290–297, September 22-25 1998.

[3] http://www.wfmc.org/.

[4] S. Olabarriaga, J. Snel, C. Botha, and R. Belleman, "Integrated support for medical image analysis methods: from development to clinical application," IEEE Transactions on Information Technology in Biomedicine, January 2007.

[5] B. Ludscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system."

[6] E. Gallopoulos, E. Houstis, and J. R. Rice, "Computer as thinker/doer: Problem-solving environments for computational science," IEEE Comput. Sci. Eng., vol. 1, no. 2, pp. 11–23, 1994.

[7] I. Taylor, "Triana generations," e-science, vol. 0, p. 143, 2006.

[8] Y. J. Choi, T. Oroguchi, Y. Kato, M. Takeda, and Y. Tago, "Labgrid: Integrated problem solving environment system for high throughput computing," e-science, vol. 0, p. 103, 2006.

[9] K. Schuchardt, B. T. Didier, and G. Black, "Ecce - a problem-solving environment's evolution toward grid services and a web architecture.," Concurrency and Computation: Practice and Experience, vol. 14, no. 13-15, pp. 1221–1239, 2002.

[10] I. T. Foster, "Globus toolkit version 4: Software for service-oriented systems.," in NPC (H. Jin, D. A. Reed, and W. Jiang, eds.), vol. 3779 of Lecture Notes in Computer Science, pp. 2–13, Springer, 2005.

[11] G. Allen, K. Davis, T. Goodale, A. Hutanu, H. Kaiser, T. Kielmann, A. Merzky, R. V. van Nieuwpoort, A. Reinefeld, F. Schintke, T. Schott, E. Seidel, and B. Ullmer, "The grid application toolkit: Towards generic and easy application programming interfaces for the grid," in Proceedings of the IEEE, vol. 93, pp. 534–550, March 2005.

[12] https://forge.gridforum.org/sf/projects/ogsa-wg.

[13] D. C. Marinescu, "A grid workflow management architecture." White paper.

[14] W. M. P. van der Aalst, , B. Kiepuszewski, and A. P. Barros, "Workflow patterns," Distributed and Parallel Databases, vol. 14, pp. 5–51, July 2003.

[15] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," 2005.

[16] P. Wagstrom, S. Krishnan, and G. von Laszewski, "GSFL: A Workflow Framework for Grid Services," in SC'2002, (Baltimore, MD), 11-16 Nov. 2002. (Poster).

[17] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," Scienti¯c Programming, vol. 13, no. 3, pp. 219–237, 2005.

[18] E. Deelman, S. Callaghan, E. Field, H. Francoeur, R. Graves, N. Gupta, V. Gupta, T. H. Jordan, C. Kesselman, P. Maechling, J. Mehringer, G. Mehta, D. Okaya, K. Vahi, and L. Zhao, "Managing large-scale workflow execution from resource provisioning to provenance tracking: The cybershake example," e-science, vol. 0, p. 14, 2006.

[19] E. Deelman, "Mapping abstract complex workflows onto grid environments," 2003.

[20] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao, "Chimera: A virtual data system for representing, querying, and automating data derivation," 2002.

[21] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li, "Taverna: A tool for the composition and enactment of bioinformatics workflows," Bioinformatics Journal , vol. 10, no. 17, pp. 3045–3054, 2004.

[22] M. Hategan, G. von Laszewski, and K. Amin, "Karajan: A grid orchestration framework." Supercomputing 2004, Pittsburgh, 6-12 Nov. 2004. (Refereed Poster).

[23] GridAnt: a client-controllable grid work°ow system.

[24] Z. Guan, F. Hernandez, P. Bangalore, J. Gray, A. Skjellum, V. Velusamy, and Y. Liu, "Gridflow: a grid-enabled scientific workflow system with a petri-net-based interface: Research articles," Concurr. Comput. : Pract. Exper., vol. 18, no. 10, pp. 1115–1140, 2006.

[25] D. Gannon, L. Fang, G. Kandaswamy, D. Kodeboyina, S. Krishnan, B. Plale, and A. Slominski, "Building grid applications and portals: An approach based on components, web services and workflow tools.," in Euro-Par, pp. 1–8, 2004.

[26] W. Emmerich, B. Butchart, L. Chen, B. Wassermann, and S. L. Price, "Grid service orchestration using the business process execution language (bpel)," Journal of Grid Computing, vol. 3, no. 3-4, pp. 283–304, 2005.

[27] G. von Laszewski, "Java CoG Kit Workflow Concepts," accepted for publication in Journal of Grid Computing, 2006.

[28] Grid Work°ow , ch. Grid Workflow with the Java CoG Kit. 2006. in preparation.

[29] O. TM., "Talisman–rapid application development for the grid.," Bioinformatics , vol. 19, pp. 212–214, 2003.

[30] H. Casanova, G. Obertelli, F. Berman, and R. Wolski, "The apples parameter sweep template: user-level middleware for the grid," in Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM), (Washington, DC, USA), p. 60, IEEE Computer Society, 2000.

[31] C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The SDSC storage resource broker," in CASCON'98 Conference, 1998.

[32] C. Botha, "DeVIDE - The Delft Visualization and Image Processing Development Environment," tech. rep., TU Delft, May 2005. http://cpbotha.net/DeVIDE.

[33] http://www.vtk.org/.

[34] http://www.itk.org/.

[35] J. G. Snel and S. D. Olabarriaga and J. Alkemade and H. G. van Andel and A. J. Nederveen and C. B. Majoie and G. J. den Heeten and M. van Straten and R. G. Belleman, "A Distributed Workflow Management System for Automated Medical Image Analysis and Logistics," in accepted to IEEE-CMBS special track on Grids for Biomedical Informatics, 2006.

[36] P. Buneman, S. Khanna, and W.-C. Tan, "Data provenance: Some basic issues," in Foundations of Software Technology and Theoretical Computer Science, 2000.