Software Engineering Project (2IP40)

Project Group 1

Software Requirements Document

version 1.0.1 (Externally Accepted), 28 March 2006

Project Team:	Sven Bego	0550191
	Roel Coset	0548132
	Robert Leeuwestein	0546746
	Maarten Leijten	0547649
	Ivo van der Linden	0547632
	Joery Mens	0547515
	Marcel Moreaux	0499480
	Tim Muller	0547961
Project Manager:	Tom Kleijkers	0515015
Senior Manager:	L. Somers	TU/e HG 7.83
Advisor:	Y.Usenko	TU/e HG 5.71
Customer:	M. ter Linden	Dutch Space
	H. de Wolf	Dutch Space

Technische Informatica, Eindhoven University of Technology, Eindhoven

Abstract

This document contains the software requirements for the SPINGRID system. This project is one of seven assignments for the course 2IP40 at Eindhoven University of Technology.

These software requirements were established according to requests formulated by Mark ter Linden and Hans de Wolf of Dutch Space. The document complies with the Software Requirements Document (SRD) from the Software Engineering Standard, as set by the European Space Agency [ESA].

Contents

Document Status Sheet

Document Title	Software Requirements Document
Document Identification	SPINGRID/Documents/Product/SRD/1.0.1
Author(s)	R. Leeuwestein, S. Bego, M. Moreaux, R. Coset, I. v.d. Linden
Version	1.0.1
Document Status	draft / internally accepted / conditionally approved / approved

Version	**Date**	**Author(s)**	**Summary**
0.0.1	20-01-2006	R. Leeuwestein	Document creation
0.0.2	24-01-2006	R. Leeuwestein, S. Bego	Added text to chapter 2, started with chapter 3
0.0.3	09-02-2006	R. Leeuwestein, M. Moreaux, S. Bego, R. Coset	First draft version for customer
0.0.4	22-02-2006	R. Leeuwestein, M. Moreaux, S. Bego, R. Coset, I. v.d. Linden	First version for internal review
0.0.5	26-02-2006	R. Leeuwestein, M. Moreaux, S. Bego, R. Coset, I. v.d. Linden	Version for the second internal review
0.1.0	26-02-2006	R. Leeuwestein, M. Moreaux, S. Bego, R. Coset, I. v.d. Linden	Internally accepted
0.1.1	09-03-2006	R. Leeuwestein, M. Moreaux, S. Bego, R. Coset, I. v.d. Linden	Version for the third internal review
0.2.0	09-03-2006	R. Leeuwestein, M. Moreaux, S. Bego, R. Coset, I. v.d. Linden	Internally accepted
0.2.1	22-03-2006	R. Leeuwestein, S. Bego, I. v.d. Linden	Internally accepted
1.0.0	28-03-2006	S. Bego, I. v.d. Linden	Externally accepted
1.0.1	28-03-2006	R. Leeuwestein	Fixed the document status

Document Change Report

Document Title	Software Requirements Document
Document Identification	SPINGRID/Documents/Product/SRD/1.0.1
Date of Changes	N/A

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to specify the software requirements and a logical model of the SPINGRID system in a clear and consistent manner.

1.2 Scope

The software will be used to implement a computational grid. This grid is able to execute specific jobs when it receives an executable accompanied by a set of data files. It completely hides the complexity of grid technology which makes the system easy to use. Usability is also increased by offering a web-based front-end for users to access the system.

1.3 List of definitions and abbreviations

This section contains the definitions of all used terms, acronyms and abbreviations in this document.

1.3.1 Definitions

2IP40	The Software Engineering Course ID at Eindhoven University of Technology
Agent	Program that is used by a resource provider to retrieve and execute jobs.
Application	A combination of a set of shell scripts and a set of (URLs to) platform dependent data.

Application Provider	An application provider can offer a set of applications to the SPINGRID system. They can restrict access for projects and for resource providers to their applications.
Client	Program that is used by all the users except those in the role of resource provider (who use the agent).
Computational Grid	A hardware and software infrastructure that enables coordinated resource sharing within dynamic organizations consisting of individuals, institutions and resources.
Customer	Dutch Space B.V.
Data	Either platform dependent data or platform independent data.
Data Provider	A data provider can offer a set of platform independent data to the SPINGRID system. They can restrict access for projects and for resource providers to their data.
Dispatcher	A dispatcher acts like a server and manages the distribution of jobs over the computational grid.
Job	Specification of application, platform independent data and auxiliary information to compute the job.
Job Provider	Job providers are users that offer jobs to projects. They have to be a member of those particular projects.
Platform Independent Data	A set of platform independent files that is used (input, uncompiled executable, etc.) to compute a job.
Platform Dependent Data	A set of platform dependent files that is required to initialize a job.
Project	A collection of jobs with specified access rights to which users (project members) can be assigned.
Project Administrator	The project administrators administrate projects and can assign and remove job providers, configure a project and restrict access for resource providers.
Resource Provider	Resource providers are users that offer time on their computers to the SPINGRID system. They can restrict access to their computer for application providers and projects.
Role	The actions and activities assigned to or required or expected of a user.
Shell Script	One or more (platform dependent) shell-commands.
SPINGRID	A computational grid using SPINGRID software.
SPINGRID Software	Software developed by Dutch Space and TU/e to build computational grids for distributed data processing.
SPINGRID System	The full name of the entire system using SPINGRIDSoftware.
System Administrator	The system administrator oversees the entire SPINGRID system and has the right to configure the system, to create and remove projects and assign and remove project administrators.
User	Either a job provider, a data provider, an application provider, a resource provider, a project administrator or a system administrator.

1.3.2 Abbreviations

ADD	Architectural Design Document
DDD	Detailed Design Document
ESA	European Space Agency
JSDL	Job Submission Description Language
SR	Software Requirements
SRD	Sofware Requirements Document
TU/e	Eindhoven University of Technology
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	eXtensible Markup Language

1.4 Documents

1.4.1 Reference Documents

[ESA]	*ESA Software Engineering Standards (ESA PSS-05-0 Issue 2)*, ESA Board for Software Standardization and Control (BSSC), 1991
[JSDL]	*Job Submission Description Language (JSDL) Specification*, Version 1.0, November 2005
[ADD]	*Architectural Design Document*, SPINGRID team, TU/e, not yet available
[DDD]	*Detailed Design Document*, SPINGRID team, TU/e, not yet available

1.4.2 Applicable Documents

[URD]	*User Requirements Document*, SPINGRID team, TU/e, version 1.0.0, February 2006

1.5 Overview

This document is structured according to the standard described in [ESA]. In chapter two a description of the system and its environment is given, along with a model that describes the software. Chapter three contains the software requirements. The fourth chapter gives the traceability tables, which link the user requirements to the software requirements, and vice-versa. Finally in Appendix A, state diagrams are presented to illustrate the flow of the SPINGRID system.

Chapter 2

General Description

2.1 Relation to current projects

There are a lot of current projects which are related to the SPINGRID project. The following subsections will give a general overview of the most relevant ones. These summaries are taken from the corresponding websites.

2.1.1 GridAssist

GridAssist[1] is a software framework that provides benefits of computing in a Computational Grid environment to programs that are not inherently Grid-aware, for users who are not experts on Grid technology. GridAssist provides a portal for access to applications, resources and data using high-speed networks, a scenario builder that can be used to construct scenarios consisting of chains of data and applications, and a controller that schedules the jobs on a Computational Grid. Unfortunately the system can not work effectively when behind a firewall. This is the main reason we do not adapt GridAssist.

2.1.2 BOINC

BOINC[2] is a software platform for distributed computing using volunteered computer resources. The most important feature of BOINC is resource sharing among independent projects, therefore many different projects can use BOINC. Projects are independent; each one operates its own servers and databases. Participants can participate in multiple projects; they control which projects they participate in, and how their resources are divided among these projects. When a project is down or has no work, the resources of its participants are divided among other projects.

[1] http://tphon.dutchspace.nl/grease/
[2] http://boinc.berkeley.edu/

2.1.3 Globus Toolkit

The open source Globus Toolkit[3] is a fundamental enabling technology for the grid, letting people share computing power, databases, and other tools securely online across corporate, institutional, and geographic boundaries without sacrificing local autonomy. The toolkit includes software services and libraries for resource monitoring, discovery, and management, plus security and file management. In addition to being a central part of science and engineering projects that total nearly a half-billion dollars internationally, the Globus Toolkit is a substrate on which leading IT companies are building significant commercial Grid products.

2.2 Relation to predecessor and successor projects

There are no real predecessors of the SPINGRID project, though Dutch Space has experimented with a project called GridAssist which is related to this project. But GridAssist is built on the Globus Toolkit and is focused on workflow computing, whereas the SPINGRID system is not using an existing toolkit to provide the grid and does not support workflow-management. So calling GridAssist a predecessor of the SPINGRID project is not correct but the two definitely have a relation with each other.

Dutch Space is convinced that using grid technology is useful to compute, because they already have been researching it for a couple of years. The goal of this project is to research a new technical view to grid computing, namely agent polling. A comparison between GridAssist, BOINC, Globus Toolkit and SPINGRID can be seen in the following figure:

	GridAssist	BOINC	Globus Toolkit	SPINGRID
Easy Installation	X	X		X
Multi Platform	X			X
Firewall Independent				X

2.3 Function and purpose

This document translates all user requirements found in [URD] into software requirements. This was done by reasoning about all user requirements and identifying where they would be used in the product. Next a logical model was made. The logical model gives a simplified description of the product and contains the functionality that was found in the user requirements. The model specifies the relations between different entities in the product. In this stage, implementation terminology must be avoided to ensure that all options of implementations are possible. The implementation that will be chosen for the project will be discussed at a high level in the Architectural Design phase. For more information, refer the Architectural Design Document [ADD] and the Detailed Design Document [DDD].

[3] http://www.globus.org/toolkit/

2.4 Environment

The environment in which the system runs is generally described in [URD], §2.5.

2.4.1 System requirements

The software programs - client, agent and dispatcher - will be further described in section 2.7, but have the following minimal requirements:

General Requirements:

- Windows XP, Mac OS X or Linux 2.4 (or higher)
- Sun JRE 1.4.2 or 1.5

Dispatcher:

- Intel Pentium IV 2 GHz or equivalent
- 2 GB RAM
- 2 MBit/s network
- work on Linux
- dedicated server

Furthermore, the dispatcher is able to simultaneously:

- handle 40 agents
- handle 10 clients
- monitor 2 times/minute

Agent:

- Intel Pentium II 300 MHz or equivalent, G4 700 MHz or equivalent
- 128 MB RAM
- 256 MB Available Harddisk Space
- handle 3 dispatchers
- poll 1 time/minute

A client does not have any specific requirements besides the general requirements.

2.5 Relation to other systems

There are no external systems directly connected to the SPINGRID system.

2.6 General constraints

The general constraints of the system are described in [URD], §2.3.

2.7 Model description

The logical model is constructed in an object-oriented way, with the Unified Modeling Language (UML). The logical model is directly derived from the user requirements in [URD]. The *high level model* can be found in section 2.7.1 which gives an overview how the different user groups are divided in the SPINGRID system. The 'normal' operation of the system is described in section 2.7.2. The classes are described in section 2.7.3, 2.7.4 and 2.7.5. These models specify the public interfaces, hence private methods are left out. Note that the displayed relations between classes are not conclusive for the implementation if better solutions are found. The logical model is a tool that makes it possible to reason about different solutions.

2.7.1 High level model

There are three programs in the SPINGRID system which can be seen in figure 2.1: the agent, the client and the dispatcher.

The agent is used by the resource provider. All resource provider requirements, as described in the [URD] §4.4, are fulfilled by the agent. An agent is connected to one or more independent dispatchers.

The client is used by the application, data and job provider. All application, data and job providers requirements, as described in [URD] §4.6, §4.7 and §4.8, are fulfilled by the client. The system admin and project admin also use the client and these requirements ([URD], §4.3 and §4.5) are also fulfilled by the client. A client is connected to one or more independent dispatchers. The main difference and therefore the reason to make an agent and a client, is that the agent is communicating with the dispatcher on a regular basis, which is not the case with the client. Communication between the client and dispatcher only exists when a command is given by the user, for example submitting a job.

The dispatcher acts like a server and manages the distribution of jobs over the computational grid. A dispatcher is connected to zero or more agents and to zero or more clients. Dispatchers operate independently.

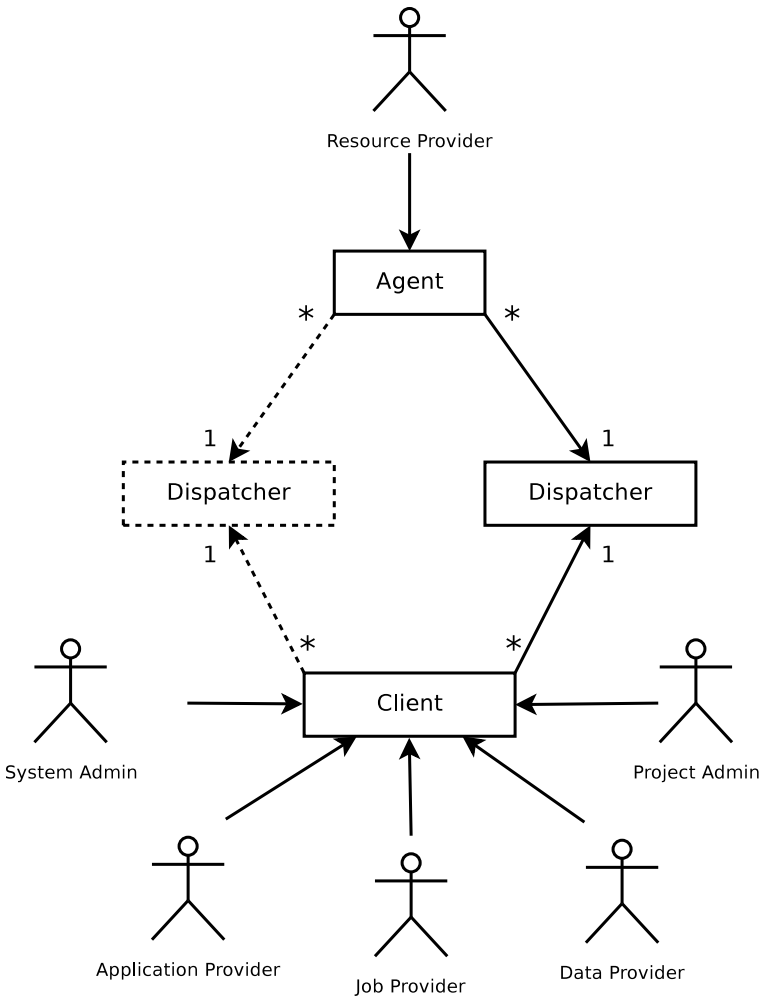

Figure 2.1: high level model

2.7.2 System Operations

Figure 2.2 displays the 'normal' operation of the system. The explanation below describes which actions need to take place in order for a job to reach an agent so it can be computed.

A (Application Provider → Dispatcher): An application provider provides an application to the dispatcher. An application is a combination of a set of shell scripts (⟶ 1) and a set of (URLs to) platform dependent data (→ 2).

B (Data Provider → Dispatcher): A data provider provides URLs to platform independent data (→ 3) to the dispatcher.

C (Dispatcher → Job Provider): A job provider requests a list of applications, which is a combination of shell scripts (→ 5) and platform dependent data (→ 6), and a list of platform independent data (→ 4). A job provider can then make a selection in both of these lists. This selection will be used to describe a job.

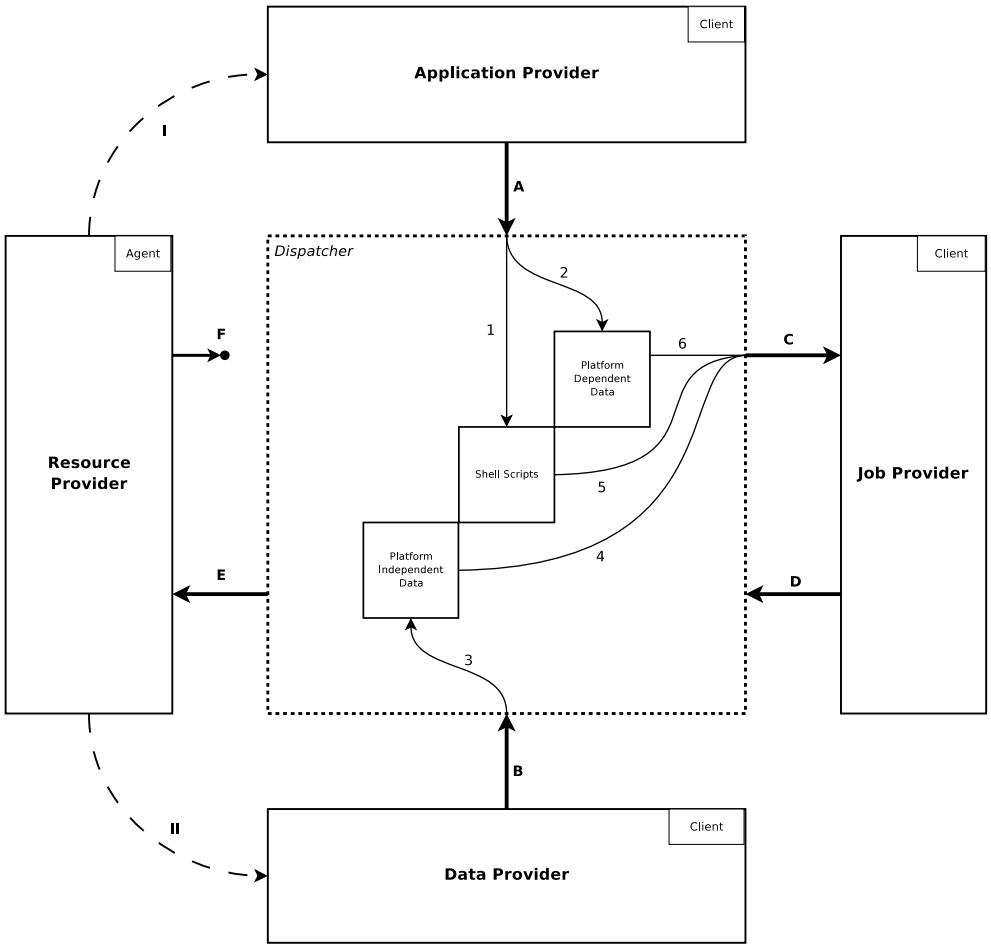

Figure 2.2: System Operations

D (Job Provider → Dispatcher): The job provider provides a job (with a proper description) to the dispatcher.

E (Dispatcher → Agent): The dispatcher found a suitable agent to compute the job on. The agent downloads the necessary files (platform independent and dependent data) and runs the shell script. The agent needs to download the platform dependent data from the application provider (→ I) and the platform independent data from the data provider (→ II). A security system could be implemented here so that the agent can only download the data if he is authorized.

F (Agent → Output): The agent finished the computation of the job and sends the results (like URL's) to the specified (which is described in the job by the Job Provider) output destination.

Monitoring from the different users and changing settings is left out of figure 2.2 because it makes the figure unnecessary complicated.

2.7.3 Classes from the dispatcher perspective

The UML model which can be seen in figure 2.3 is derived from the user requirements. All the classes and their relations are described here. Note that figure 2.3 illustrates the system from the perspective of the dispatcher. The UML model from the other perspective, namely the user perspective, can be found in section 2.7.4.

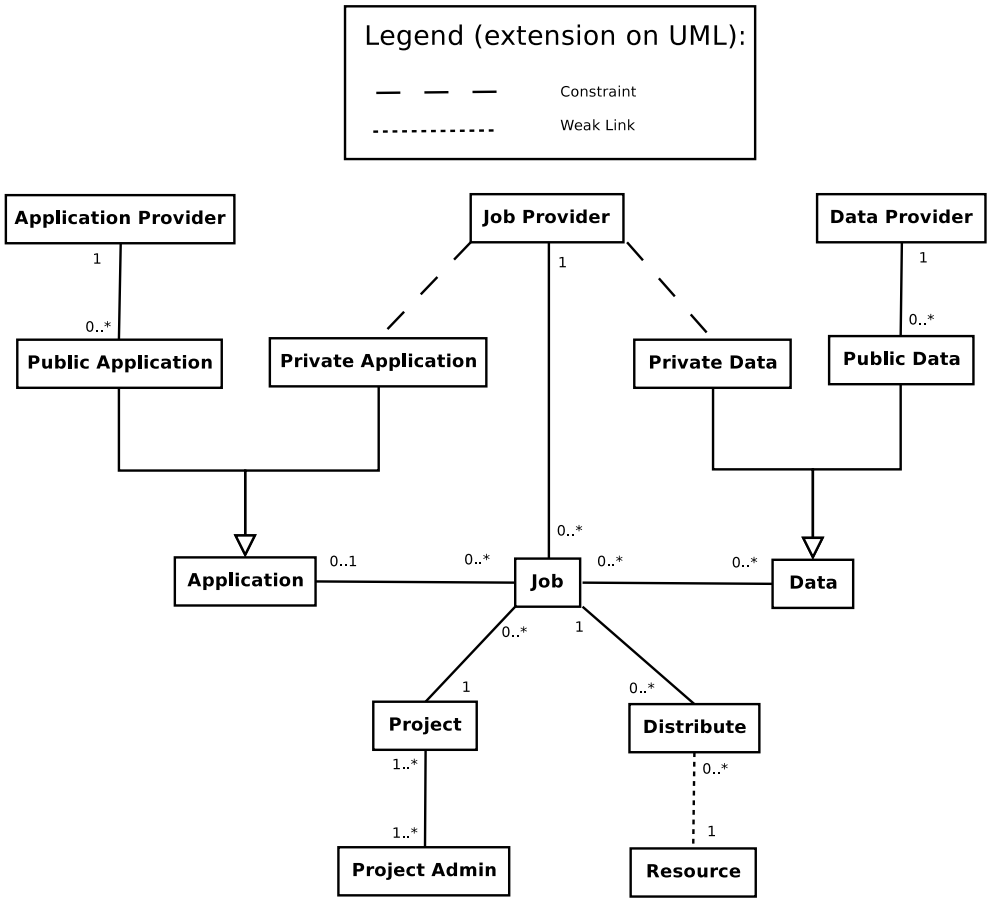

Figure 2.3: UML model from the perspective of the dispatcher

Application Provider An application provider can offer a set of applications to the SPINGRID. They can restrict access for projects and for resource providers to their applications. An application provider can provide zero or more public applications.

Application A combination of a set of shell scripts and a set of (URLs to) platform dependent data. An application can be used by zero or more jobs.

[SR_5060]: An application has an attribute called *Characteristics* which contains the characteristics that the application requires.

Priority: 2

Public Application This class is a subclass of application. A public application is provided by precisely one application provider. When a job provider creates a job it will be possible for him to select an application from all public applications (provided he has been authenticated).

Private Application This class is a subclass of application. A private application is provided by precisely one job provider. Note that this does not follow automatically from the UML model, therefore a constraint is introduced:

[SR_7110]: $(\forall a : a \in PrivateApplication : (\exists j : j \in Job : (j.Application = a)) \ \wedge \ \neg(\exists j_1, j_2 : j_1, j_2 \in Job \wedge (j_1.Application = a) \wedge (j_2.Application = a) : (j_1.JobProvider \neq j_2.JobProvider)))$
Priority: 1

This also implies that a job provider cannot make a new job with a private application provided by a different job provider. Even better, a job provider should never be able to see a private application provided by a different job provider.

Data Provider A data provider can offer a set of data to the SPINGRID. They can restrict access for projects and for resource providers to their data. A data provider can provide zero or more sets of data.

Data Data should be read as platform independent data and is provided by one data provider. The data can be used by zero or more jobs and is used as input for an application.

Public Data This class is a sub class of data. A public data object is provided by precisely one data provider. When a job provider creates a job it will be possible for him to select a set of data from all public data sets (provided he has been authenticated).

Private Data This class is a subclass of data. A private data object is provided by precisely one job provider. Note that this does not follow automatically from the UML model, therefore a constraint is introduced:

[SR_7120]: $(\forall d : d \in PrivateData : (\exists j : j \in Job : (j.Data = d)) \ \wedge \ \neg(\exists j_1, j_2 : j_1, j_2 \in Job \ \wedge \ (j_1.Data = d) \ \wedge \ (j_2.Data = d) : (j_1.JobProvider \neq j_2.JobProvider)))$
Priority: 1

Job Provider Job providers are users that offer a job to a project. They have to be a member of that particular project. A job provider can provide zero or more jobs.

Job Specification of application, platform independent data and auxiliary information to compute the job. A job cannot use more than one application and uses zero or more data sets. The zero in "0..1" is inherited from JSDL, in which it is possible that job does not have an application. A job is also distributed zero or more times and is always provided by one job provider.

[SR_7070]: A job has an attribute called *JobDefinition* which contains a full description of the job.
Priority: 1

Distribute A job needs to be distributed to a resource in order to complete the job. Note that the link between a distribute- and resource-object is only a weak link because a resource can disappear from the grid at any time.

Resource A resource is a computer which can be used by the grid to execute jobs. A resource can have zero or more distributions.

[SR_3011]: A resource has an attribute called *Characteristics* which contains the characteristics of the resource.
Priority: 2

Project A project is a collection of jobs with specified access rights to which users (project members) can be assigned:

[SR_7130]: A project has at least one project admin.
Priority: 1

Project Admin The project admin administrates projects and can assign and remove job providers, configure a project and restrict access for resource providers. A project admin administrates at least one project.

[SR_7131]: A project admin administrates at least one project.
Priority: 1

System Admin The system admin is the administrator of the SPINGRID. He can authorize and unauthorize users as application provider, data provider, resource provider and project admin. It is also possible for the system admin to add and remove projects and to change global settings. There is only one instance of the system admin and therefore is left out of the UML model.

[SR_7150]: The SPINGRID shall have one system admin.
Priority: 1

2.7.4 Classes from the user perspective

Figure 2.4 illustrates the commands that the client program can send to the dispatcher. The commands are categorized by type of user. Obviously, there are some commands that are available to all the users. Each command will have one or more software requirements, which will be described in section 3.1.1.

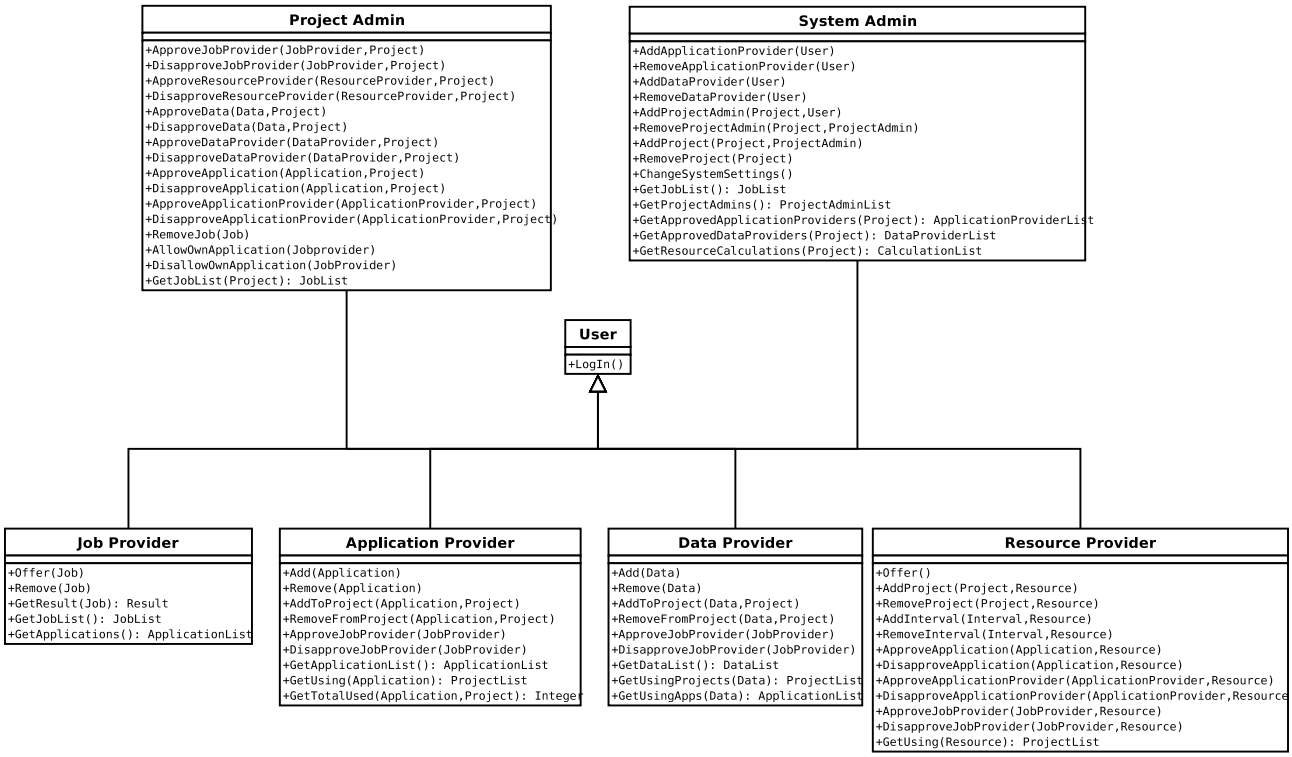

Figure 2.4: UML model from the perspective of the user

2.7.5 JSDL class

In the SPINGRID system, it is necessary to describe a job. A standardized language is a good solution because it saves time and it is easy to reuse. JSDL has been chosen because this option was suggested by the customer and it was considered adequate for the SPINGRID

system.

"The Job Submission Description Language (JSDL) is a language for describing the requirements of computational jobs for submission to resources, particularly in Grid environments, though not restricted to the latter. The JSDL language contains a vocabulary and normative XML Schema that facilitate the expression of those requirements as a set of XML elements."

JSDL is illustrated in figure 2.5 and a description can been found in the text below. The requirements of JSDL can been found in section 3.1.2. Note that a lot of requirements have a low priority and thus a lot of attributes do not need to be implemented. JSDL fully supports this because not implemented attributes are left undefined. More detailed information of JSDL can be found in [JSDL].

[SR_7140]: JSDL is partly implemented as the job description language.
Priority: 1

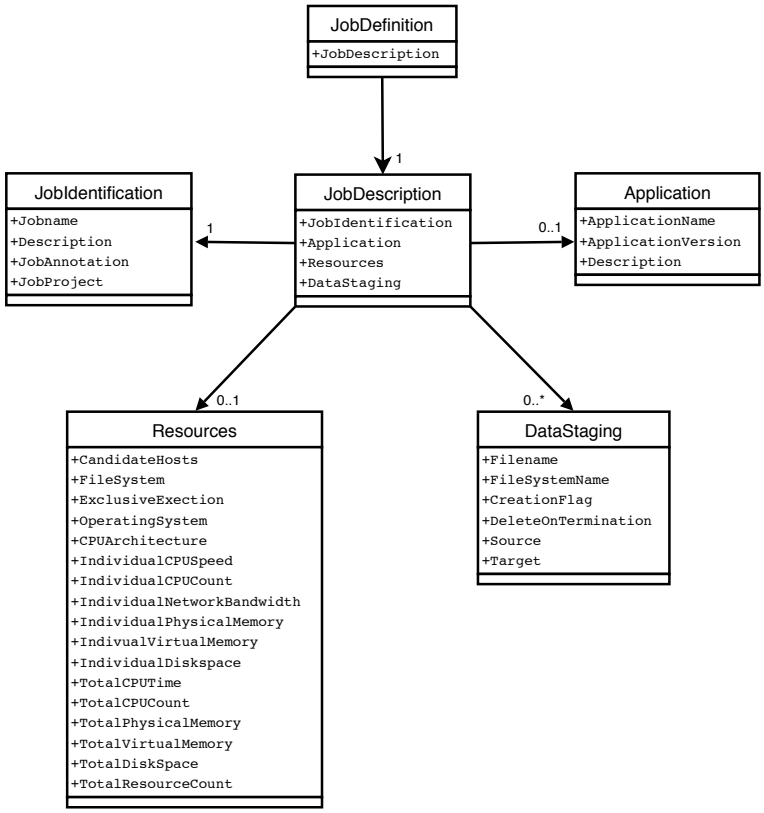

Figure 2.5: JSDL

JobDefinition This element describes the job and its requirements. It contains a JobDescription section. It is the root element of the JSDL document.

JobDescription This element describes the job and its requirements. It contains JobIdentification, Application, Resources, and DataStaging elements.

JobIdentification This element contains all elements that identify the job: JobName, Description, JobAnnotation, and JobProject. If this element is not present then its value, including all of its sub-elements, is undefined.

Application This element describes the application and its requirements. It contains ApplicationName, ApplicationVersion and Description elements. It serves as a high level generic container that is intended to hold more specific application definitions. One such definition is that of the POSIX compliant normative extension given in [JSDL], §8.1.1. Used without any extension, it uniformly describes an application by its name and version number. If this is not present then this job definition does not define an application to execute. The JSDL document could be defining a data staging job, or a null job. See also [JSDL], §6.3.2 and §8.1.2.

Resources This element contains the resource requirements of the job. If this element is not present then the consuming system may choose any set of resources to execute the job.

Any combination of the listed resources sub-elements may be present in the resources element of a JSDL document. In particular, any combination of "Individual", and "Total" elements of the same or different types may be present in a resources element. But note that all elements present in a JSDL document must be satisfied for the entire document to be satisfied. (See also [JSDL], §4)

DataStaging Data staging defines the files that should be moved to the execution host (stage in) and the files that should be moved from the execution host (stage out). Files are staged in before the job starts executing. Files are staged out after the job terminates.

If a directory is specified in the FileName element or Source element then a recursive copy will be performed. If the execution environment does not support recursive copying an error should be reported. The specification of this error, including how or when it is raised, is beyond the scope of the JSDL specification.

It is possible to stage out the same file more than once by specifying the same FileName (on the same FileSystem) in multiple stage out DataStaging elements.

It is also possible, but deprecated, to use the same FileName in separate DataStaging elements to stage in to the same local file. The result is unspecified.

The CreationFlag determines whether the staged file should append or overwrite an existing file. This element must be present in a DataStaging element.

The DeleteOnTermination element may be used to delete a file after the job terminates. If the file is to be staged out the deletion is done after the stage out completes. A file may be deleted even if it was not created by the job. For example, suppose that a file exists on the execution host and the CreationFlag is set to dontOverwrite. This file will still be deleted if DeleteOnTermination is true provided the job has the appropriate rights.

The ordering of the DataStaging elements in the JSDL document is not significant. That is, the order of the DataStaging elements in the document does not imply any ordering, besides the ordering already mentioned concerning job execution and carrying out the different stage in (or stage out) operations.

More complex file transfers, for example, conditional transfers based on job termination status or pre-warming of grid-enabled file-systems are out of scope. Permission and access control for the staged files should be handled by the implementation and is out of scope of the JSDL specification.

More complicated deployment scenarios than the file staging described here, for example, deployment and configuration of the execution environment itself, are out of scope of the JSDL specification.

Chapter 3

Specific requirements

In this chapter all requirements and constraints of the product to be developed are given except a few requirements which can be found in chapter 2. The product will adhere to these requirements. Each of the requirements has a unique identifier so it can be traced throughout the entire project. Every requirement is accompanied by a priority level. This priority level is mapped on integers from 1 to 5, where 1 stands for the highest priority and 5 stands for the lowest priority. Software requirements marked with priority level 1 and 2 will be implemented regardless of resources. Software requirements marked with priority level 3, 4 and 5 will only be implemented in priority order if time allows.

3.1 Functional requirements

3.1.1 User class requirements

User

Methods

- LogIn()
  [SR_8000] The user can log in after which he can preform his actions accordingly to his role(s).
  Priority: 1

Job Provider

Methods

- Offer(Job)
  [SR_7010] Offers a job to the system.
  Priority: 1
  [SR_7012] A job provider can provide a private application.

Priority: 3
[SR_7013] A job provider can provide private data.
Priority: 2

- Remove(Job)
  [SR_7011] Removes a job from the system.
  Priority: 2

- GetResult(Job):Result
  [SR_7030] Returns the results of a completed job.
  Priority: 3

- GetJobList():JobList
  [SR_7040] Returns a list of the provider's jobs.
  Priority: 2

- GetApplications():ApplicationList
  [SR_7020] Returns a list of applications available to the provider.
  Priority: 2

Application Provider

Methods

- Add(Application)
  [SR_5010] Provides an application to the system.
  Priority: 1

- Remove(Application)
  [SR_5011] Removes an application from the system.
  Priority: 2

- AddToProject(Application, Project)
  [SR_5020] Adds a project on which the application may be used.
  Priority: 2

- RemoveFromProject(Application, Project)
  [SR_5021] Removes a project on which the application may be used.
  Priority: 2

- ApproveJobProvider(JobProvider)
  [SR_5070] Allows a job provider to use the applications of the application provider.
  Priority: 2

- DisapproveJobProvider(JobProvider)
  [SR_5071] Disallows a job provider to use the applications of the application provider.
  Priority: 2

- GetApplicationList():ApplicationList
  [SR_5030] Returns a list of the provider's applications.
  Priority: 3

- GetUsing(Application):ProjectList
  [SR_5050] Returns a list of projects where the application is used.
  Priority: 3

- GetTotalUsed(Application, Project):Integer
  [SR_5040] Returns the total number of times the application has been used in the project.
  Priority: 3

Data Provider

Methods

- Add(Data)
  [SR_6010] Provides platform independent data to the system.
  Priority: 1

- Remove(Data)
  [SR_6011] Removes platform independent data from the system.
  Priority: 1

- AddToProject(Data, Project)
  [SR_6020] Adds a project on which the platform independent data may be used.
  Priority: 2

- RemoveFromProject(Data, Project)
  [SR_6021] Removes a project on which the platform independent data may be used.
  Priority: 2

- ApproveJobProvider(JobProvider)
  [SR_6060] Allows a job provider to use the platform independent data of the data provider.
  Priority: 2

- DisapproveJobProvider(JobProvider)
  [SR_6061] Disallows a job provider to use the platform independent data of the data provider.
  Priority: 2

- GetDataList():DataList
  [SR_6040] Returns a list of provider's platform independent data.
  Priority: 3

- GetUsingProjects(Data):ProjectList
  [SR_6030] Returns a list of projects which use the platform independent data.
  Priority: 3

- GetUsingApps(Data):ApplicationList
  [SR_6050] Returns a list of applications which use the platform independent data.
  Priority: 4

Resource Provider

Methods

- Offer()
  [SR_3010] Offers and identifies the resource to the system.
  Priority: 1

- AddProject(Project,Resource)
  [SR_3020] Adds a project on which the resource may be used.
  Priority: 2

- RemoveProject(Project,Resource)
  [SR_3021] Removes a project on which the resource may be used.
  Priority: 2

- AddInterval(Interval,Resource)
  [SR_3030] Adds an interval when the resource can be used.
  Priority: 5

- RemoveInterval(Interval,Resource)
  [SR_3031] Removes an interval when the resource can be used.
  Priority: 5

- ApproveApplication(Application,Resource)
  [SR_3050] Allows an application to be executed on the resource.
  Priority: 2

- DisapproveApplication(Application,Resource)
  [SR_3051] Disallows an application to be executed on the resource.
  Priority: 2

- ApproveApplicationProvider(ApplicationProvider,Resource)
  [SR_3060] Allows the applications of an application provider to be executed on the resource.
  Priority: 2

- DisapproveApplicationProvider(ApplicationProvider,Resource)
  [SR_3061] Disallows the applications of an application provider to be executed on the resource.
  Priority: 2

- ApproveJobProvider(JobProvider,Resource)
  [SR_3070] Allows a job provider to provide his jobs to the resource.
  Priority: 2

- DisapproveJobProvider(JobProvider,Resource)
  [SR_3071] Disallows a job provider to provide his jobs to the resource.
  Priority: 2

- GetUsing(Resource):ProjectList
  [SR_3040] Returns a list of projects where the resource is used.
  Priority: 4

Project Admin

Methods

- ApproveJobProvider(User,Project)
  [SR_4010] Allows the user to provide jobs for the project. If the user isn't already a job provider then he will get the role of job provider.

Priority: 1

- DisapproveJobProvider(JobProvider,Project)
  [SR_4011] Disallows the job provider to provide jobs for the project. If the job provider was only allowed to provide jobs for the project then he will loose the role of job provider.
  Priority: 1

- ApproveResourceProvider(ResourceProvider,Project)
  [SR_4020] Allows a resource provider to process jobs of the specific project.
  Priority: 2

- DisapproveResourceProvider(ResourceProvider,Project)
  [SR_4021] Disallows a resource provider to process the specific project.
  Priority: 2

- ApproveData(Data,Project)
  [SR_4030] Allows platform independent data to be used for a specific project.
  Priority: 2

- DisapproveData(Data,Project)
  [SR_4031] Disallows platform independent data to be used for a specific project.
  Priority: 2

- ApproveDataProvider(DataProvider,Project)
  [SR_4032] Allows a data provider to provide data for a specific project.
  Priority: 2

- DisapproveDataProvider(DataProvider,Project)
  [SR_4033] Disallows a data provider to provide data for a specific project.
  Priority: 2

- ApproveApplication(Application,Project)
  [SR_4040] Allows an application to be used for a specific project.
  Priority: 2

- DisapproveApplication(Application,Project)
  [SR_4041] Disallows an application to be used for a specific project.
  Priority: 2

- ApproveApplicationProvider(ApplicationProvider,Project)
  [SR_4042] Allows an application provider to provide applications for a specific project.
  Priority: 2

- DisapproveApplicationProvider(ApplicationProvider,Project)
  [SR_4043] Disallows an application provider to provide applications for a specific project.
  Priority: 2

- RemoveJob(Job,Project)
  [SR_4060] Removes a job from a specific project.
  Priority: 2

- AllowOwnApplication(JobProvider)
  [SR_4050] Allows a job provider to provide private applications.
  Priority: 2

- DisallowOwnApplication(JobProvider)
  [SR_4051] Disallows a job provider to provide private applications.
  Priority: 2

- GetJobList(Project):JobList
  [SR_4070] Returns a list of all jobs in the project.
  Priority: 2

System Admin

Methods

- AddApplicationProvider(User)
  [SR_2010] Authorizes the user to be an application provider.
  Priority: 2

- RemoveApplicationProvider(ApplicationProvider)
  [SR_2011] Unauthorizes the user to be an application provider.
  Priority: 2

- AddDataProvider(User)
  [SR_2020] Authorizes the user to be a data provider.
  Priority: 3

- RemoveDataProvider(DataProvider)
  [SR_2021] Unauthorizes the user to be a data provider.
  Priority: 3

- AddProjectAdmin(Project,User)
  [SR_2030] Authorizes the user to be a project admin of a specific project.
  Priority: 1

- RemoveProjectAdmin(Project,ProjectAdmin)
  [SR_2031] Unauthorizes the user to be a project admin of a specific project if at least one other project admin is authorized to the project.
  Priority: 2

- AddProject(User)
  [SR_2040] Adds a project to the system and makes the user project admin of the project. If the user isn't already a project admin then he will get the role of project admin.
  Priority: 1

- RemoveProject(Project)
  [SR_2041] Removes a project from the system. If there are job providers who where only allowed to provide jobs for the project then they will loose the role of job provider. If the project admin was only project admin of the project then he will loose the role of project admin.
  Priority: 1

- ChangeSystemSettings()
  [SR_2050] Changes the settings of the system.
  Priority: 1

- GetJobList():JobList
  [SR_2060] Returns a list of all jobs in the system.
  Priority: 2

- GetProjectAdmins(Project):ProjectAdminList
  [SR_2070] Returns a list of all project admins authorized to a specific project.
  Priority: 2

- GetApprovedApplicationProviders(Project):ApplicationProviderList
  [SR_2080] Returns a list of application providers authorized to a specific project.
  Priority: 2

- GetApprovedDataProviders(Project):DataProviderList
  [SR_2090] Returns a list of data providers authorized to a specific project.
  Priority: 3

- GetResourceCalculations(Project):CalculationList
  [SR_2100] Returns a list of resources and which jobs they have calculated in a specific project.
  Priority: 3

3.1.2 JSDL class requirements

JobIdentification

Attributes

- JobName
  [SR_1210] This element is a string that may be specified by a user to name the job specified in the JSDL document. It may not be unique to a particular JSDL document, which means that a user may specify the same JobName for multiple JSDL documents. If this element is not present then it is not defined.
  Priority: 1

- Description
  [SR_1220] This element provides descriptive, human readable, information about its containing complex element. It may be present as a sub-element of a number of other JSDL elements: JobIdentification, Application, FileSystem, etc. If this element is not present as a sub-element then no description is defined.
  Priority: 3

- JobAnnotation
  [SR_1230] This element is a string that may be specified by a user to annotate the job. If this element is not present then it is not defined. In contrast to the Description element, JobAnnotation may contain information that is intended for use by the consuming system.
  Priority: 3

- JobProject
  [SR_1240] This element is a string specifying the project to which the job belongs. The project could be used by accounting systems or access control systems. The interpretation of the JobProject elements is left to the implementation of the consuming system. If this element is not present then it is not defined.
  Priority: 1

Application

Attributes

- ApplicationName
  [SR_1310] This element is a string that defines the name of the application and is used to identify the application independent of the location of its executable on a host or system. If this is not present then it is not defined and a null job is assumed unless there is an application extension element present that defines the application to execute.
  Priority: 1

- ApplicationVersion
  [SR_1320] This element is a string that defines the version of the application to execute. The consuming system must use exact textual match to select the version of the application. If this element is not present then it is not defined and any version of the application may be executed.
  Priority: 1

- Description
  [SR_1330] This element provides descriptive, human readable, information about its containing complex element. It may be present as a sub-element of a number of other JSDL elements: JobIdentification, Application, FileSystem, etc. If this element is not present as a sub-element then no description is defined.
  Priority: 3

Resources

Attributes

- CandidateHosts
  [SR_1410] This element is a complex type specifying the set of named hosts which may be selected for running the job. If this element is present then one or more hosts from the set must be chosen to run the job. If this is not present then it is not defined. A named host may be a single host (e.g., a machine name), a logical group of hosts (e.g., a named logical group or cluster), a virtual machine, and so on.
  Priority: 3

- FileSystem
  [SR_1411] This element describes a filesystem that is required by the job. It is a complex type that may contain the location where the filesystem should be made available, the required amount of disk space and the type of the filesystem. The filesystem may be local to the resource (e.g., on a local disk), or may be remote (e.g., an NFS mount).
  Priority: 3

- ExclusiveExecution
  [SR_1412] This is a boolean that designates whether the job must have exclusive access to the resources allocated to it by the consuming system. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 3

- OperatingSystem
  [SR_1413] This is a complex type that defines the operating system required by the job. It may contain Description, OperatingSystemVersion, and OperatingSystemType elements. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 2

- CPUArchitecture
  [SR_1414] This element is a string specifying the CPU architecture required by the job in the execution environment. If this is not present then it is not defined and the consuming system may choose any value. Values not defined by the JSDL ProcessorArchitectureEnumeration (see [JSDL], §5.2.1) may be used by specifying the special token "other" and including the value as an extension (see [JSDL], §7.3). See also examples below.
  Priority: 2

- IndividualCPUSpeed
  [SR_1415] This element is a range value specifying the speed of each CPU required by the job in the execution environment. The IndividualCPUSpeed is given in multiples of hertz. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 2

- IndividualCPUTime
  [SR_1416] This element is a range value specifying the total number of CPU seconds required on each resource to execute the job. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 3

- IndividualCPUCount
  [SR_1417] This element is a range value specifying the number of CPUs for each of the resources to be allocated to the job submission. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 2

- IndividualNetworkBandwidth
  [SR_1418] This element is a range value specifying the bandwidth requirements of each individual resource. The amount is specified as multiple of bits per second. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 2

- IndividualPhysicalMemory
  [SR_1419] This element is a range value specifying the amount of physical memory required on each individual resource. The amount is given in bytes. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 2

- IndividualVirtualMemory
  [SR_1420] This element is a range value specifying the required amount of virtual memory for each of the resources to be allocated for this job submission. The amount is

given in bytes. If this is not present then it is not defined and the consuming system may choose any value.
Priority: 3

- IndividualDiskSpace
  [SR_1421] This is a range value that describes the required amount of disk space for each resource allocated to the job. The amount of disk space is given in bytes. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 2

- TotalCPUTime
  [SR_1422] This element is a range value specifying total number of CPU seconds required, across all CPUs used to execute the job. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 3

- TotalCPUCount
  [SR_1423] This element is a range value specifying the total number of CPUs required for this job submission. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 3

- TotalPhysicalMemory
  [SR_1424] This element is a range value specifying the required amount of physical memory for the entire job across all resources. The amount is given in bytes. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 3

- TotalVirtualMemory
  [SR_1425] This element is a range value specifying the required total amount of virtual memory for the job submission. The amount is given in bytes. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 3

- TotalDiskSpace
  [SR_1426] This is a range value that describes the required total amount of disk space that should be allocated to the job. The amount of disk space is given in bytes. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 3

- TotalResourceCount
  [SR_1427] This element is a range value specifying the total number of resources required by the job. If this is not present then it is not defined and the consuming system may choose any value.
  Priority: 3

DataStaging

Attributes

- FileName
  [SR_1510] This element is a string specifying the local name of the file (or directory) on the execution host. The FileName must be a relative path (see [JSDL], §6.5.3) and the only delimiter allowed must be "/". In other words the FileName must NOT start with an initial /. The FileName may be a hierarchical directory path of the form $< directory > / < directory > /.../ < name >$. The "$< name >$" may be either a directory or a file.
  Priority: 1

- FileSystemName
  [SR_1520] If the FileSystemName is specified then the FileName is relative to the specified FileSystem declaration referenced by the name. In this case there must also be a FileSystem element with the same name. If the FileSystemName element is not present then it is not defined. If the FileSystemName is not defined then the FileName is relative to the working job directory as determined by the consuming system. Note that JSDL extensions may allow defining a value for the working job directory. See, for example, the WorkingDirectory element definition ([JSDL], §8.1.7) of the "Executables on POSIX Conformant Hosts" extension.
  Priority: 3

- CreationFlag
  [SR_1530] This element determines whether the file created on the local execution system can overwrite or append to an existing file. A typical value for this element, expected to be commonly supported, is "overwrite".
  Priority: 3

- DeleteOnTermination
  [SR_1540] This is a boolean that determines whether the file should be deleted after the job terminates. If true the file is deleted after the job terminates or after the file has been staged out. Otherwise the file remains, subject to the persistency of the FileSystem it is on. If not present, behavior is unspecified and depends on the consuming system.
  Priority: 3

- Source
  [SR_1550] A Source element contains the location of the file or directory on the remote system. This file or directory must be staged in from the location specified by the (optional) URI before the job has started. If this element is not present then the file does not have to be staged in.
  Priority: 1

- Target
  [SR_1560] A Target element contains the location of the file or directory on the remote system. This file or directory must be staged out to the location specified by the (optional) URI after the job has terminated. If this element is not present then the file or directory does not have to be staged out.
  Priority: 1

3.2 Non-functional Requirements

[SR_9000]: The SPINGRID system shall implement a computational grid.
Priority: 1

[SR_9010] The system requirements are as described in section 2.4.1.
Priority: 2

[SR_9020]: The language used in the product will be English.
Priority: 1

[SR_9030]: Interaction with the system will be provided by a command-line interface.
Priority: 1

[SR_9031]: Interaction with the system will be provided by a web-based user interface.
Priority: 4

[SR_9040]: The SPINGRID system will be able to process at least 40 executing jobs at a time.
Priority: 1

[SR_9050]: The SPINGRID system will select the resource that will be used for processing a job.
Priority: 1

[SR_9060]: The SPINGRID system shall only send jobs to resources if their characteristics at least match the characteristics required by the job and application used in the job.
Priority: 2

[SR_9070]: The SPINGRID system shall provide a trust model as described in appendix A of [URD].
Priority: 4

[SR_9080]: The (un-)installation of the SPINGRID system will not require a computer expert.
Priority: 1

[SR_9090]: When there are at least 2 dispatchers in the system and one of them disappears, the system will continue without malfunction.
Priority: 5

[SR_9100]: When all the dispatchers in the system are down and one of them is restarted, the system will continue without malfunction.

Priority: 4

[SR_9110]: If one of the resources disappears while performing a job, the system will requeue the job.
Priority: 1

[SR_9120]: A job will be declared failed after it has been requeued for a configurable number of times.
Priority: 4

[SR_9130]: The functionality of the product will not be restricted when agents or clients are behind a firewall (which does not restrict traffic over port 80) and/or NAT.
Priority: 2

[SR_9140]: The SPINGRID system will be implemented in Java according to (a tailored version of) the BSSC Java Coding Standards.
Priority: 2

[SR_9150]: The system will be able to run for at least a week without interruption.
Priority: 2

[SR_9160]: All programs in the SPINGRID system will log what they are doing.
Priority: 2

[SR_9170]: The total time in which none of the dispatchers responds will not exceed one hour a day.
Priority: 2

[SR_9180]: The SPINGRID system is able to sent a notification to the job provider when a job is completed, failed or removed.
Priority: 2

Chapter 4

Requirements traceability matrix

User Requirements	Software Requirements
UR_0010	SR_9000
UR_0020	SR_9020
UR_0030	SR_9030
UR_0040	SR_9031
UR_0050	SR_9040
UR_0060	SR_9050
UR_0070	SR_9060
UR_0080	SR_9070, SR_8000
UR_1010	SR_7140
UR_1020	SR_7010, SR_7140
UR_1030	SR_7140, SR_7070, SR_1411, SR_1412, SR_1413, SR_1414, SR_1415, SR_1416, SR_1417, SR_1418, SR_1419, SR_1420, SR_1421, SR_1422, SR_1423, SR_1424, SR_1425, SR_1426, SR_1427
UR_2010	SR_7150
UR_2020	SR_2010
UR_2030	SR_2020
UR_2040	SR_2030
UR_2050	SR_2011
UR_2060	SR_2021
UR_2070	SR_2031
UR_2080	SR_2040
UR_2090	SR_2041
UR_2100	SR_2050
UR_2110	SR_2060
UR_2114	SR_2070
UR_2120	SR_2080
UR_2130	SR_2090
UR_2140	SR_2100
UR_2150	SR_9010

UR_3010	SR_3010
UR_3020	SR_3010
UR_3030	SR_3020, SR_3021
UR_3040	SR_3030, SR_3031
UR_3050	SR_3040
UR_3060	SR_3060, SR_3061
UR_3062	SR_3050, SR_3051
UR_3070	SR_9080
UR_3072	SR_3011
UR_3074	SR_3070, SR_3071
UR_3080	SR_9010
UR_4010	SR_7130
UR_4020	SR_4010
UR_4030	SR_4011
UR_4040	SR_4020, SR_4021
UR_4042	SR_4030, SR_4031, SR_4032, SR_4033
UR_4050	SR_4040, SR_4041, SR_4042, SR_4043
UR_4060	SR_4050, SR_4051
UR_4070	SR_4060
UR_4080	SR_4070
UR_4090	SR_9010
UR_5010	SR_5010
UR_5012	SR_5011
UR_5020	SR_9010
UR_5030	SR_5020, SR_5021
UR_5040	SR_5050
UR_5042	SR_5030
UR_5050	SR_5040
UR_5052	SR_5060
UR_5054	SR_5070, SR_5071
UR_5060	SR_9010
UR_6010	SR_6010
UR_6012	SR_6011
UR_6020	SR_6020, SR_6021
UR_6030	SR_6030
UR_6032	SR_6040
UR_6040	SR_6050
UR_6042	SR_6060, SR_6061
UR_6050	SR_9010
UR_7010	SR_7010
UR_7020	SR_7020
UR_7040	SR_7012, SR_7110
UR_7050	SR_7013, SR_7120
UR_7052	SR_1410

UR_7060	SR_7030
UR_7070	SR_7011
UR_7080	SR_7040
UR_7082	SR_7140, SR_7070, SR_1411, SR_1412, SR_1413, SR_1414, SR_1415, SR_1416, SR_1417, SR_1418, SR_1419, SR_1420, SR_1421, SR_1422, SR_1423, SR_1424, SR_1425, SR_1426, SR_1427
UR_7090	SR_9180
UR_7100	SR_9180
UR_7110	SR_9180
UR_7120	SR_9010
UR_8010	SR_9090
UR_8020	SR_9100
UR_8030	SR_9110
UR_8040	SR_9120
UR_8050	SR_9130
UR_8060	SR_9140
UR_8070	SR_9150
UR_8080	SR_9160
UR_8090	SR_9170

Software Requirements	**User Requirements**
SR_1410	UR_7052
SR_1411	UR_7082, UR_1030
SR_1412	UR_7082, UR_1030
SR_1413	UR_7082, UR_1030
SR_1414	UR_7082, UR_1030
SR_1415	UR_7082, UR_1030
SR_1416	UR_7082, UR_1030
SR_1417	UR_7082, UR_1030
SR_1418	UR_7082, UR_1030
SR_1419	UR_7082, UR_1030
SR_1420	UR_7082, UR_1030
SR_1421	UR_7082, UR_1030
SR_1422	UR_7082, UR_1030
SR_1423	UR_7082, UR_1030
SR_1424	UR_7082, UR_1030
SR_1425	UR_7082, UR_1030
SR_1426	UR_7082, UR_1030
SR_1427	UR_7082, UR_1030
SR_2010	UR_2020
SR_2011	UR_2050
SR_2020	UR_2030
SR_2021	UR_2060

SR_2030	UR_2040
SR_2031	UR_2070
SR_2040	UR_2080
SR_2041	UR_2090
SR_2050	UR_2100
SR_2060	UR_2110
SR_2070	UR_2114
SR_2080	UR_2120
SR_2090	UR_2130
SR_2100	UR_2140
SR_3010	UR_3020, UR_3010
SR_3011	UR_3072
SR_3020	UR_3030
SR_3021	UR_3030
SR_3030	UR_3040
SR_3031	UR_3040
SR_3040	UR_3050
SR_3050	UR_3062
SR_3051	UR_3062
SR_3060	UR_3060
SR_3061	UR_3060
SR_3070	UR_3074
SR_3071	UR_3074
SR_4010	UR_4020
SR_4011	UR_4030
SR_4020	UR_4040
SR_4021	UR_4040
SR_4030	UR_4042
SR_4031	UR_4042
SR_4032	UR_4042
SR_4033	UR_4042
SR_4040	UR_4050
SR_4041	UR_4050
SR_4042	UR_4050
SR_4043	UR_4050
SR_4050	UR_4060
SR_4051	UR_4060
SR_4060	UR_4070
SR_4070	UR_4080
SR_5010	UR_5010
SR_5011	UR_5012
SR_5020	UR_5030
SR_5021	UR_5030
SR_5030	UR_5042

SR_5040	UR_5050
SR_5050	UR_5040
SR_5060	UR_5052
SR_5070	UR_5054
SR_5071	UR_5054
SR_6010	UR_6010
SR_6011	UR_6012
SR_6020	UR_6020
SR_6021	UR_6020
SR_6030	UR_6030
SR_6040	UR_6032
SR_6050	UR_6040
SR_6060	UR_6042
SR_6061	UR_6042
SR_7010	UR_7010, UR_1020
SR_7011	UR_7070
SR_7012	UR_7040
SR_7013	UR_7050
SR_7020	UR_7020
SR_7030	UR_7060
SR_7040	UR_7080
SR_7070	UR_7082, UR_1030
SR_7110	UR_7040
SR_7120	UR_7050
SR_7130	UR_4010
SR_7140	UR_7082, UR_1030, UR_1010, UR_1020
SR_7150	UR_2010
SR_8000	UR_0080
SR_9000	UR_0010
SR_9010	UR_3080, UR_7120, UR_2150, UR_6050, UR_5060, UR_4090, UR_5020
SR_9020	UR_0020
SR_9030	UR_0030
SR_9031	UR_0040
SR_9040	UR_0050
SR_9050	UR_0060
SR_9060	UR_0070
SR_9070	UR_0080
SR_9080	UR_3070
SR_9090	UR_8010
SR_9100	UR_8020
SR_9110	UR_8030
SR_9120	UR_8040
SR_9130	UR_8050

SR_9140	UR_8060
SR_9150	UR_8070
SR_9160	UR_8080
SR_9170	UR_8090
SR_9180	UR_7100, UR_7090, UR_7110

Appendix A

Petri-nets

A.1 Dispatcher

A.1.1 Top Level (figure A.1.1)

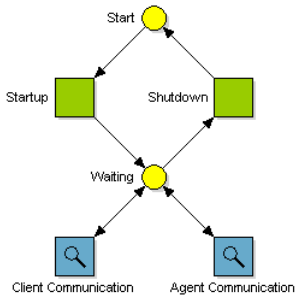

Figure A.1: Top Level

Startup:
When the dispatchers starts, it enters a state in which it waits for incoming requests from clients and agents.

Shutdown:
This processors symbolizes two methods to shutdown the dispatcher:

- The dispatcher is shutdown by a user on the machine on which the dispatcher is running.
- The dispatcher is shutdown by a remote command from the system admin. This command does not appear in the Petri-net for sake of reduction of complexity.

A.1.2 Agent Communication Subnet (figure A.1.2)

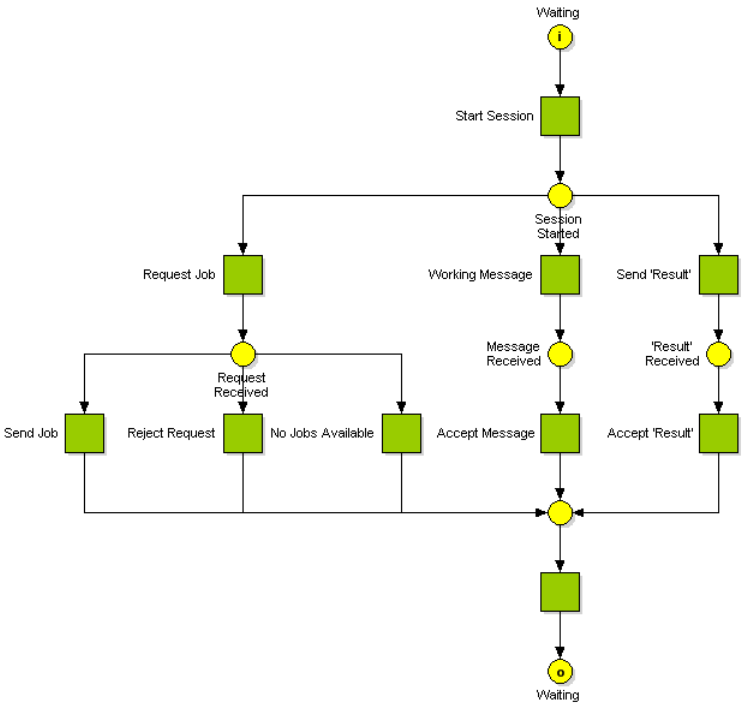

Figure A.2: Agent Communication Subnet

Start Session:
When an agent connects to the dispatcher a session is started.

Request Job:
The agent tells the dispatcher it is ready to receive a job.

Send Job:
The dispatcher accepts the request and sends a job to the agent.

Reject Request:
The dispatcher rejects the request for a job. The agent might have been banned.

No Jobs Available:
The dispatcher informs the agent that there are currently no jobs available.

Working Message / Accept Message:
The agent informs the dispatcher is it still working on a job.

Send 'Result' / Accept 'Result':
The agent sends the result to the dispatcher. It is possible that the result should be send somewhere else. In this case, the agent informs the dispatcher that it did.

A.1.3 Client Communication Subnet (figure A.1.3)

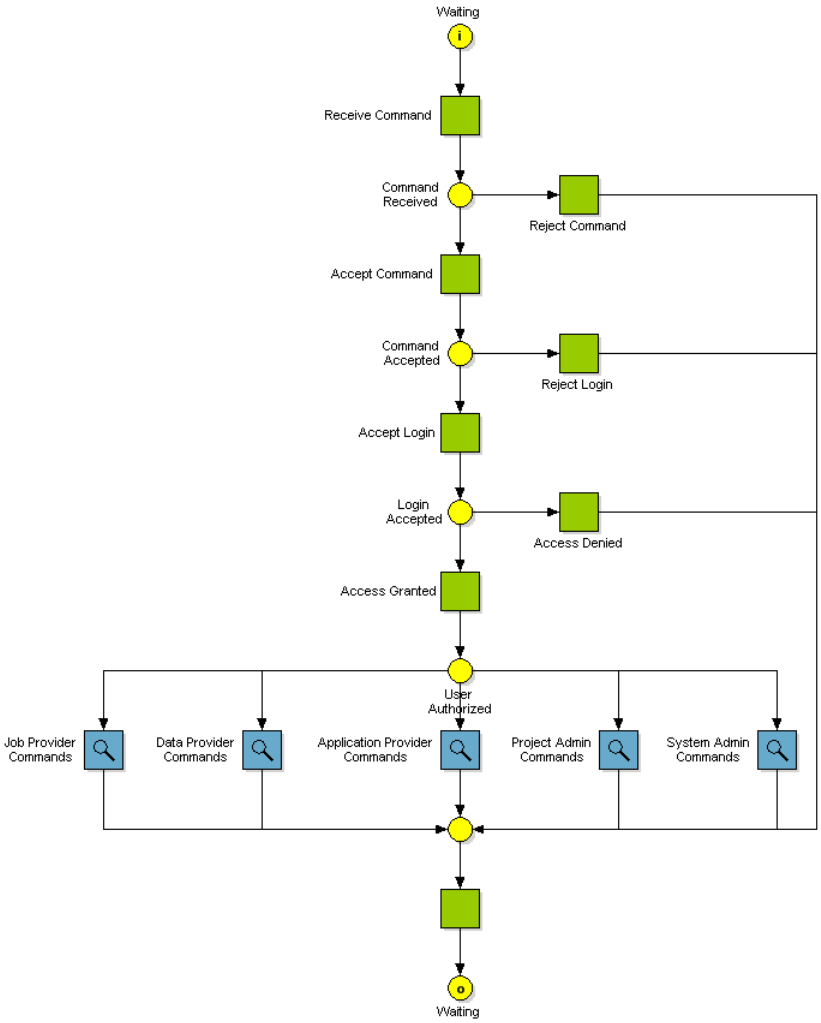

Figure A.3: Client Communication Subnet

Receive Command:
A request is received from a client.

Accept Command:
The received command is a known command.

Reject Command:
The received command is not known.

Accept Login:
The authentication-data that was sent along with the command is correct.

Reject Login:
The above-mentioned data is incorrect.

Access Granted:
The required access level to execute the command is matched to the access level of the user that sent the command. In this case, access to the command is granted to the user.

Access Denied:
The user does not have the right to execute the command that he sent.

A.1.4 Job Provider Commands Subnet (figure A.1.4)

The *Submit Job*, *List Jobs*, *Remove Job*, *List Applications* and *Get Job Result* processors represent the commands that are available to a job provider. These processors will not be listed below as their function is trivial.

The *Send Joblist*, *Send AppList*, *Accept Private App*, *Accept Private Data* processors represent the sending of the requested result to the client. These processors will also not be listed.

Accept Job:
The job that was sent with the Submit job command is in the right format (JSDL).

Reject Job:
There is an error in the job. For example, it does not match the right format (JSDL).

Accept Remove:
The job can be removed.

Reject Remove:
The job can not be removed (for example, it does not exist).

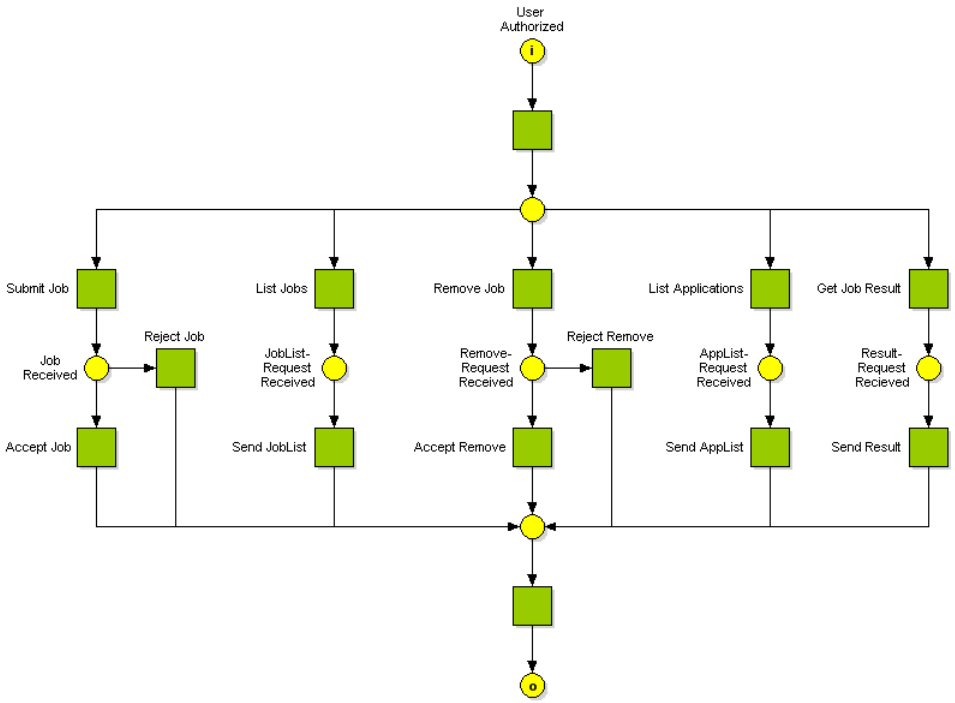

Figure A.4: Job Provider Commands Subnet

A.1.5 Data Provider Commands Subnet (figure A.1.5)

The *List Data*, *Provide Public Data*, *Remove Public Data* and *Change Data Access Rights* processors represent the commands that are available to a data provider. These processors will not be listed below as their function is trivial.

The *Send Datalist* and *Accept Public App* processors represent the sending of the requested result to the client. These processors will also not be listed.

Accept Remove:
The data can be removed.

Reject Remove:
The data can not be removed (for example, it does not exist).

Accept Changes:
The given changes are acceptable.

Reject Changes:
The given changes are not acceptable. For example, the changes are inconsistent.

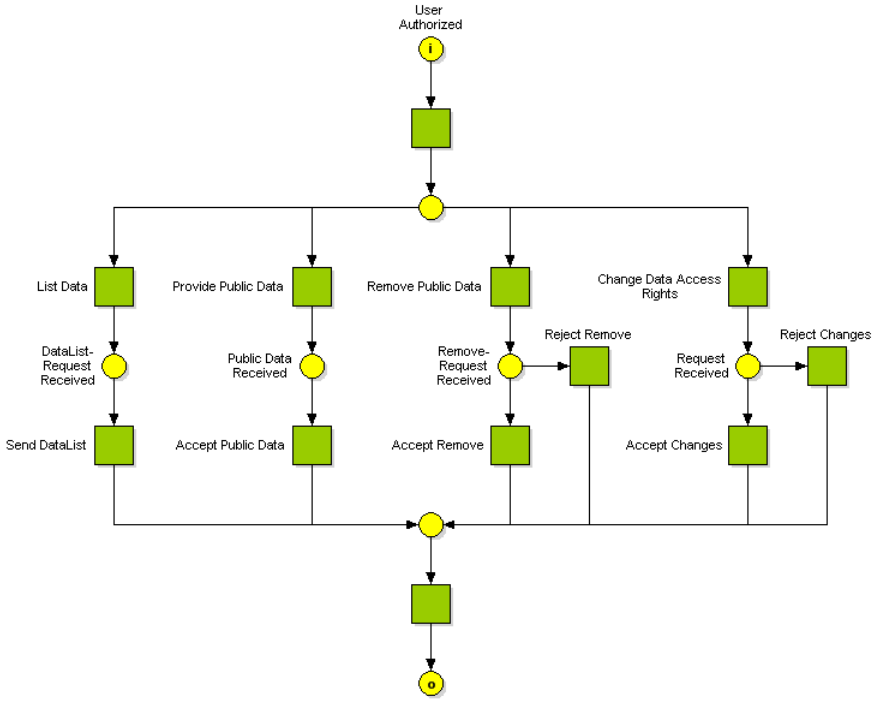

Figure A.5: Data Provider Commands Subnet

A.1.6 Application Provider Commands Subnet (figure A.1.6)

The application provider subnet is identical to the data provider subnet, except for the processor names. The processors in the application provider subnet have similar functionality to their equivalents in the data provider subnet. Therefore, the processors in the application provider subnet will not be listed.

A.1.7 Project Admin Commands Subnet (figure A.1.7)

The *List Jobs*, *Remove Job*, *Add Job Provider*, *Change Job Provider Settings*, *Remove Job Provider*, *Change Project Access Rights* and *Change Project Settings* processors represent the commands that are available to a project admin. These processors will not be listed below as their function is trivial.

The *Send JobList*, *Accept Removal* and *Accept Settings* processors represent the sending of the requested result to the client. These processors will also not be listed.

Accept Request (Add Job Provider):

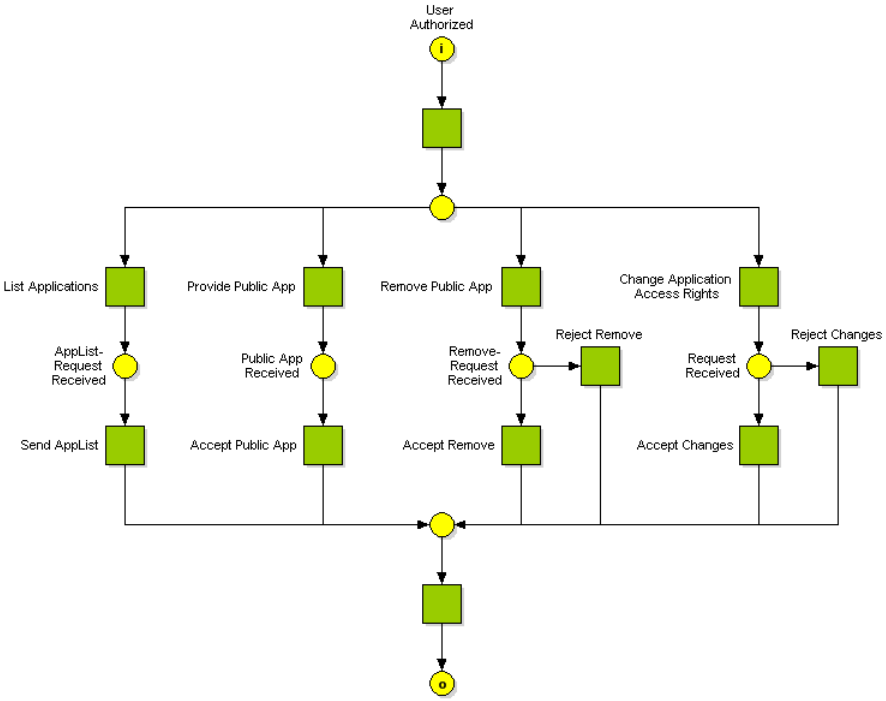

Figure A.6: Application Provider Commands Subnet

The parameters that were sent with the Add Job Provider-command are acceptable.

Reject Request (Add Job Provider):
The parameters are unacceptable. For example, the job provider is already trusted in the project.

Accept Request (Remove Job Provider):
The job provider can be removed.

Reject Request (Remove Job Provider):
The job provider can not be removed. For example, the job provider does not exist.

Accept Changes:
The proposed changes are acceptable.

Reject Changes:
The proposed changes are unacceptable. For example, they are inconsistent.

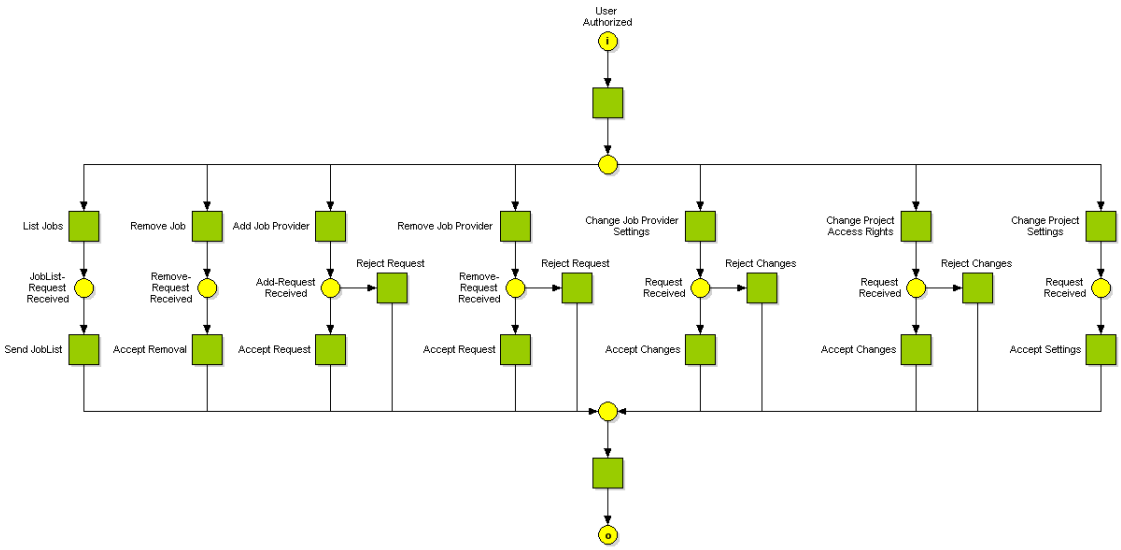

Figure A.7: Project Admin Commands Subnet

A.1.8 System Admin Commands Subnet (figure A.1.8)

The processors on the first row (*List Jobs* etc.) represent the commands that are available to the system admin. These processors will not be listed below as their function is trivial.

The processors on the last row that are not 'Accept'-processors represent the sending of the requested result to the client. These processors will also not be listed.

The 'Accept'-processors represent the negation of the corresponding 'Reject'-processors. These processors will also not be listed.

Reject Request (Add User):
The parameters of the sent command are unacceptable. For example, a user with the same username already exists.

Reject Request (Remove User):
The specified user can not be removed. For example, the user does not exist.

Reject Request (Add Role):
The specified user can not received the specified role. For example, the user already has that role.

Reject Request (Remove Role):
The role can not be removed from the specified user (parameter). For example, the user does not have that role.

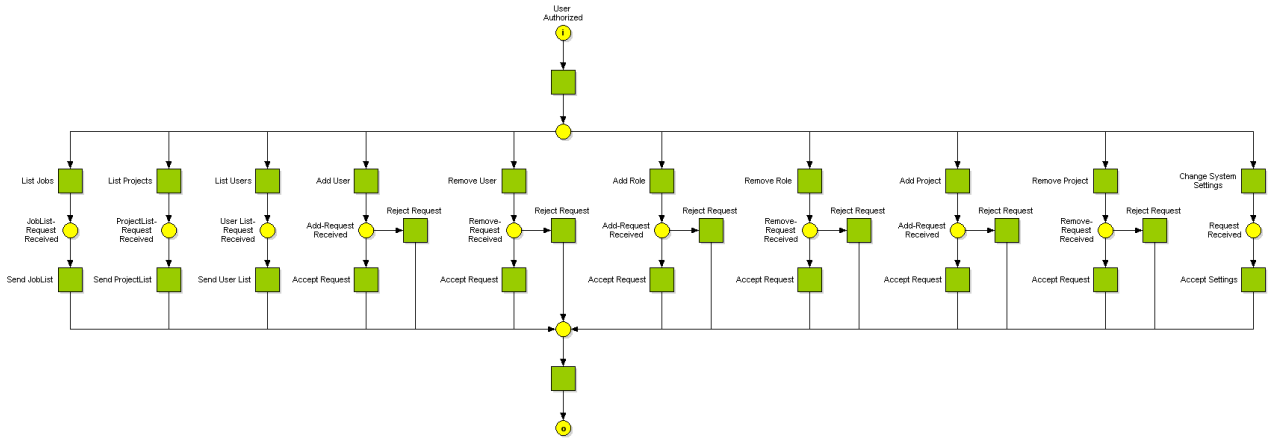

Figure A.8: System Admin Commands Subnet

Reject Request (Add Project):
It is not possible to create a project with the specified name. For example, a project with the same name already exists.

Reject Request (Remove Project):
The project can not removed. For example, it does not exist.

A.2 Agent (figure A.2)

Start up:
When the agent starts, it sends its system specification to the dispatcher and it receives a sessionID from the dispatcher. Now the agent enters a state in which it is waiting for a job to finish calculating.

Shut down:
The agent software can only be shut down from the waiting state. When a resource provider wants to shut down while a job is being calculated, the calculation will be interrupted (See the Failed processor) and the dispatcher will be notified.

Job request:
This processor notifies the dispatcher that the agent is waiting for work.

Request rejected:
When a dispatcher does not have a suitable job for the agent, the job request will be rejected. The agent will now return to the waiting state.

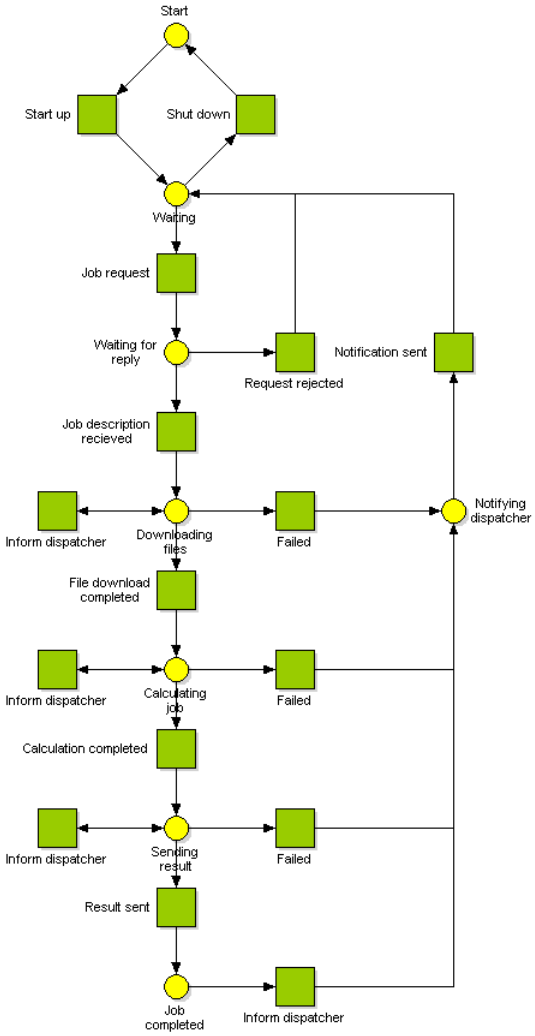

Figure A.9: Agent

Job description received:
When a dispatcher had a suitable job for the agent, the job description will be send to the agent. When the agent has received the job description, it starts downloading the required files.

Inform dispatcher:
While calculating a job, the agent informs the dispatcher, that it is still running and calculating the job.

Notification sent:
The dispatcher has been informed and the agent will return to the waiting state.

Failed:
The agent can stop the calculation of a job. Normally this will not happen, but if there is something wrong with a job this might happen. A job will also fail when a resource provider wants to shut down the agent software.

File download completed:
When all files have been downloaded, the agent starts the calculation of the job.

Calculation completed:
When the calculation is completed, the agent starts sending the result to the location specified in the job description.

Result sent:
Now is the job completed, the dispatcher will be informed that the job is completed. The agent will return to the waiting state.

A.3 Client (figure A.3)

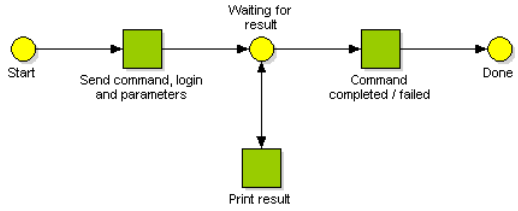

Figure A.10: Client

Send command ....:
This processor will send the input from the commandline to the dispatcher.

Print result:
This processor will print the messages, produced by the dispatcher, on the screen.

Command completed / failed:
After the command completes or fails, the program exits.