IST-2001-32133
GridLab – A Grid Application Toolkit and Tesbted

D9.1 “Requirements Analysis and Definition for the Grid Resource Management System”

Author(s):	Krzysztof Kurowski[1], Bogdan Ludwiczak[1], Jarek Nabrzyski[1], Juliusz Pukacki[1], Varvarigos Manos[2], Dolkas Konstantinos[2], Tsasakou Sofia[2]
Title:	Requirements Analysis and Definition for the Grid Resource Management System
Subtitle:	none
Work Package:	9
Lead Partner:	PSNC
Partners:	PSNC, NTUA/ICCS, ISUFI, MU, VU, SUN,
Filename:	GridLab-9-D.1-0001-GRM_Analysis
Version:	1.0
Config ID:	GridLab-9-D.1-0001-1.0
Classification:	INT

Abstract: In this document a requirements analysis is performed to evaluate the needs of all classes of end-users and their applications regarding the resource management and furthermore to fulfill these requirements by providing the GridLab Resource Management System (GRMS). This requirements document will further lead to the architecture

[1] Poznan Supercomputing and Networking Center
[2] NTUA/ICCS

and design specification for GRMS. In the latter one the interfaces to other sub-systems will be defined. Based on these requirements the design and architecture document and architectural specification of the components (and their relationships) necessary to meet the WP objectives will be established in the next design document. Boundary conditions and interfaces with other Grid components will be specified and where appropriate API's will be defined.

1. WP9 Goals – based on the Annex1

It is clear that sophisticated, fault tolerant scheduling mechanisms based upon reservation facilities and performance prediction are required in order to efficiently utilize Grid environments. But much more than this, in a dynamic environment, the needs of the processes running on the Grid will change, resources will change, and a mechanism must exist to monitor the present needs, the expected performance, and the actual performance, and to adapt accordingly. Such capabilities do not exist today, but will be critical for the types of applications and Grids we envision.
WP9 will develop example components needed to manage Grid resources of various kinds, and also the API's needed to call the resource managers from applications, from portals, etc. The APIs will be flexible enough to incorporate resource managers developed by other projects as well. Two specific objectives for WP9 are: (i) development of an architecture for distributed scheduling and resource management on a Grid, building a Resource Broker (RB) based on user's preferences and resource local policies, and (ii) development of generic APIs for this RB and other RBs developed elsewhere (e.g. Sun Grid Engine components, GrADS RBs) to be used in the system. The APIs will form part of the GAT, and will be callable from the various application GATs (CGAT, TGAT) as well as from the portal. Developments of GridLab, and those elsewhere in the Grid community, will be discussed at GGF meetings for wide applicability and interoperability of different resource management components.
The specific RB developed by the WP9 will incorporate prediction-based scheduling, including predictions for job queue wait time and job completion time as well as data access/communication time.
Various criteria for selecting resources will be used and evaluated. The work will be coordinated also with WP7 (Adaptive Grid Components), and will be implemented and tested through WP5 (Testbeds).

This document is a deliverable associated with the task 9.1 which states the following: T9.1: Dynamic Grid Scheduling requirement analysis, functional and technical specification (Month 1-36). The first deliverable of WP9 will be the requirement specification in month 3. This specification will include the application specific demands to dynamic grid resource management services and it will conclude with the system architecture. The document will be followed by a technical specification document in month 6, which outlines the complete resource management infrastructure, and especially defines the interfaces and the demands to the underlying low level grid resource management middleware (Globus, Sun Grid Engine and Condor).
These two documents will be updated continuously, and will also contain specifications for interfaces to a) other GridLab work packages, b) grid middleware layers as defined by the GGF and associated research groups.

2. GridLab Resource Management System –general requirements

The GridLab Resource Management System's main goal is mapping the job resource requests to the resources in a way that will satisfy both the application users (Job Owners) and Resource Administrators (Resource Owners). A resource here can be a computing element or any other device which is required by a job and further it will be annotated as RES. Moreover within WP9 workpackage the APIs of the GRMS will be defined and made available to the grid application developer as a part of GAT. The APIs will be defined as soon as the required functionalities of GRMS are defined and designed.
The following WP1 requirements must be fulfilled by the GRMS:

- enable efficient and effective use of resources,
- run and control jobs in a trustable, secure, environment (access to machines, data transfer, software),
- provide slow startup functionality (the job starts at the disconnected from the grid resource or set of resources, without any grid middleware and, when the resources get connected to the Grid, the GRMS allows the application to migrate to better resources),
- provide flexible, easy-to-use, simple interfaces for applications,
- provide support for mobile users' jobs,
- hide the complexity of the grid,
- support the collaborative environments,
- guidelines for making effective use of grid environment (adding e.g. checkpointing, portability, fault tolerance capabilities to application)

As soon as the detailed requirements from all workpackages will be known, we will extend this list.

The following assumptions are made for designing the architecture of the GRMS system:
The job requests will be directed to the Job Request Queue (JRQ). There will be a possibility to plug-in various algorithms of JRQ scheduling in the system. This will be done for research and evaluation of various scheduling strategies for grids. So, the JRQ will be served starting from the priority based FIFO algorithm, which in principle serves the jobs in order of their appearance, unless the priorities of newly coming job are higher that the jobs already place in JRQ, up to very complicated algorithms like reservation based ones or various metaheuristic multicriteria scheduling algorithms.
Various scheduling approaches will also be possible. For example a choice of the best RES could be performed by taking into account only the first (only one) job resource request in the JRQ and the other, waiting requests are not taken into consideration. The other option is to perform scheduling of many RESes at one time.

For each request the best RES (or "best compromise" in a case of multicriteria-based request) is always used, taking into account user preferences.
In general processing of the job on the target resource may be delayed due to unavailability of the resource. Resource reservation will be made possible where such a possibility exists.
While making the decision on the job to resource mapping the performance indexes are optimized. It means that such objectives like overall throughput, response time, load balance, data transfer, communication time etc. are addressed in the optimization process.

GRMS will also deal with all the dynamic variations of the grid environment, allowing a job to be rescheduled, preempted (in the local resource management system) and migrated.
Various scheduling algorithms will be tested and used in the GRMS, including min CT, backfilling, as well as some other heuristic algorithms. To perform these algorithms some prediction information about job time characteristics will be needed. This will be provided by both WP9 and WP7. These prediction mechanisms must be able to evaluate the expected times for each job and for each RES.
The pilot version of the GRMS will be used to perform some research on various prediction and scheduling mechanisms to tune the scheduling algorithms towards the GridLab applications and users. As the environment will finally migrate to other grid communities some learning techniques will be embedded into the scheduling decision modules.

To specify interfaces of GRMS full knowledge of applications requirements (WP1, WP2, WP3 as well as WP4) is needed. Since no all the requirements coming from those WPs are known at the moment we start with proposing the some basic "need-to-have" requirements, based on common user/application work scenarios. The GRMS will support the following operations:

- Submit Job – this is the most important functionality of GRMS; application describes job using some universal description language and passes that description to GRMS; GRMS returns job identifier which than can be used by application in interaction with other services (e.g. job monitoring)
- Job control – resume, cancel, stop, preempt…jobs in progress
- Migrate Job – from GRMS point of view migration is similar to job submission, but it requires some additional mechanisms to transfer the application to the destination node(s).
- Spawn Job – this can only be done via Submit Job.
- Resource Query – just query what resources are available

- Job Predict - predict my job future (When will it start? When will it finish? How long will it run on particular resource?) – the requirements towards WP7 will be specified here at the WP9 technical specification stage, but, in general, the WP7 will deliver the mechanisms for short time future predictions of resources and networks. Within WP9 some additional mechanisms, history based predictions, will be developed.
- Estimate Resources - Estimate the resource needs for my job (What resources do I need to finish my job before deadline?).
- QoS negotiation and contract specification - negotiate the service level agreement between the GRMS (resources) and the user/job.
- Resource reservation – make a resource reservation.

The above mentioned functionality will be further presented and described by definition of the use cases.

3. User-level use cases

There are two kinds of direct users of the GRMS: application/portal user and the application developer who uses the GRMS via GAT API. The following are the application/portal user level use cases.

Use case:	**Define/Negotiate QoS**
Actors:	Application, Resource Need Estimator, Authorization System, Information System
Type:	Primary
Description:	Use case begins when Application calls a specific function of QoS Definition/Negotiation. At first the operation should be authorized in the Authorization System. The system (GRMS) takes - provided by Application - job description and according to it submits a report of possible quality of services for that job. For defining the different quality of services a Resource Need Estimator (part of WP9) and the Grid Information System (provided by WP10) are used. The system returns the possible quality of services and the user has the ability to choose the one that best fits his/her needs. It is also possible that some changes are requested and a special request on some parameters is made. In this case the new data are resubmitted and an acceptance is replied or new choices are returned to the user.

Use case:	**Resource Reservation**
Actors:	Application, Authorization System
Type:	Primary
Description:	Use case begins when Application calls a specific function of resource reservation. At first the operation should be authorized in Authorization System. GRMS takes - provided by the Application - the characteristics of the resources needed and according to them searches for possible available resources. According to the results of the search the Resource utilization profile(s) of the suitable resource(s) is/are processed and some time of those resources and CPU percentage is reserved for a specific period in the future. The user is informed and after confirming the action the resource utilization profile of the target resource(s) is/are updated.

REMARK: The two above use cases are very much research oriented and will be designed and implemented only for the (local resource management) systems which allow making resource reservations. The UML diagrams of those use cases will be provided later.

Use case:	**Submit Job**
Actors:	Application, File Transfer System, Authorization System, Information System
Type:	Primary
Description:	Use case begins when Application calls a specific function of job submission. At first the operation should be authorized in the Authorization System. The system (GRMS) takes - provided by Application - job description and according to it submits a job. For finding resources the Grid Information System (provided by WP10) is used. Depending on a job description the system can interact with File Transfer System (WP8) for copying files. The system returns a job identifier to the Application and use case ends.

Use case:	**Migrate Job**
Actors:	Application, File Transfer System, Authorization System
Type:	Primary
Description:	Use case begins when Application calls a specific function of job migration. At first the operation should be authorized in Authorization System. GRMS takes - provided by the Application - job migration description and according to it submits job. The File

Transfer System is used to copy files to new location.

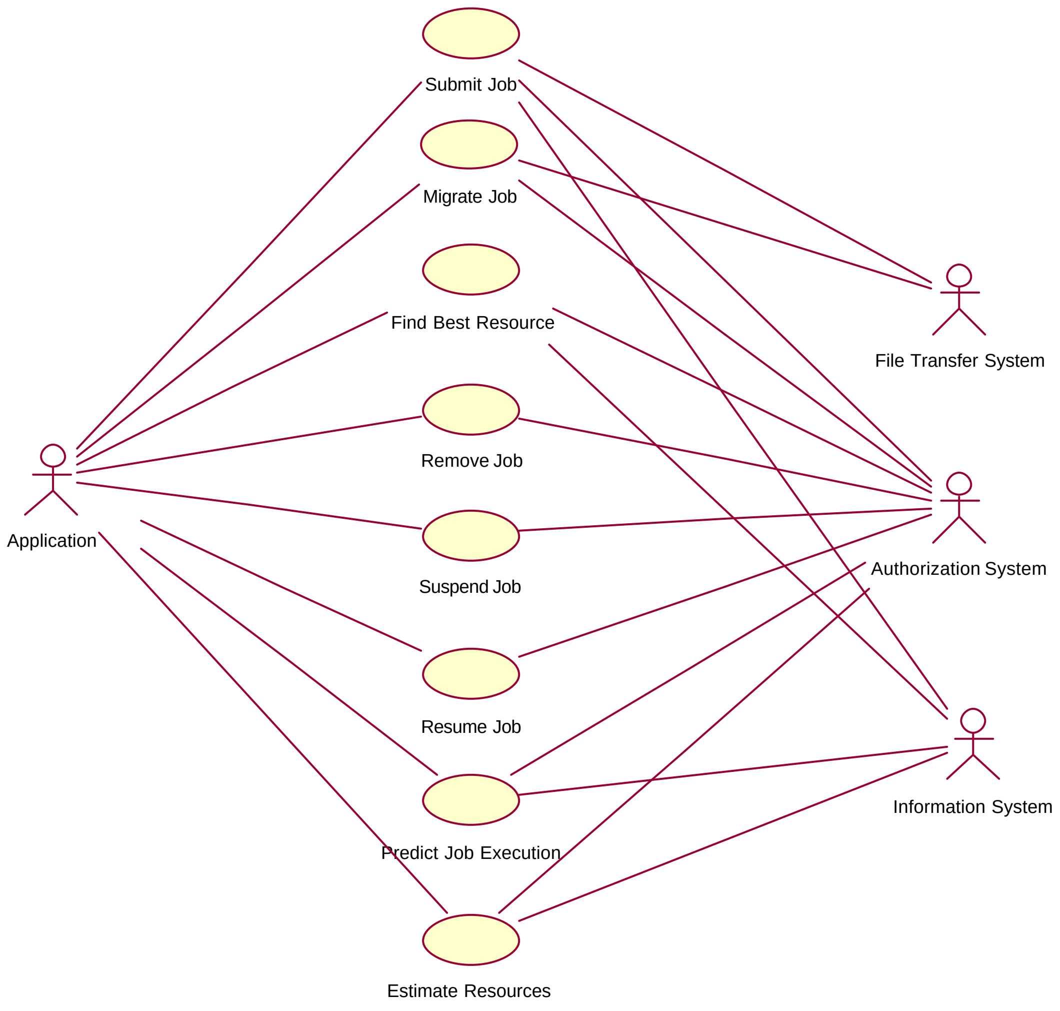
Use case:	**Find Best Resource**
Actors:	Application, Information System, Authorization System
Type:	Primary
Description:	Use case begins when Application calls a specific function of finding the best resource. At first this operation should be authorized in Authorization System. Then the GRMS evaluates resources which are provided by Information System. The system returns a chosen resource and use case ends.

Use case:	**Remove Job**
Actors:	Application, Authorization System
Type:	Primary
Description:	Use case begins when Application calls a specific function of removing a job. Operation should be authorized first. Then the system removes a job of *id* provided by the Application.

Use case:	**Resume Job**
Actors:	Application, Authorization System
Type:	Primary
Description:	Use case begins when Application calls a specific function of resuming a job. Operation should be authorized first. Then the system resumes a job of *id* provided by the Application.

Use case:	**Predict Job Execution**
Actors:	Application
Type:	Primary
Description:	Use case begins when Application calls a specific function of predicting job execution. According to a job description given by the Application the system predicts job execution (start time, end time, run time, queue wait time etc.) on specific resource

Use case:	**Estimate Resources**
Actors:	Application
Type:	Primary
Description:	Use case begins when Application calls a specific function of estimating resources. The system estimates the resource needs for job (e.g. what resources are needed to finish job before deadline)


Submit Job
Migrate Job
Find Best Resource
Remove Job
Suspend Job
Resume Job
Predict Job Execution
Estimate Resources
Application
File Transfer System
Authorization System
Information System

4. Developer-level use cases

The following are the GRMS system developer use cases:

Use case:	**Submit Job**
Actors:	Application (initiator), File Transfer System, Authorization System
Purpose:	Submit Job to the Grid Environment
Overview:	Application calls function responsible for job submission. The system submits a job to Grid Environment according to job description given by application.
Type:	Primary

Typical Course of Event:

Actor Action	System's Response
1. Use case begins when Application calls "submit job" function with job description as argument.	2. Authorizes operation for user who runs the Application, using the Authorization System
	3. Parses job description to gather information needed for submission
	4. Finds potential resource for job execution according to job description (for finding potential resources an Information System is used)
	5. Evaluates resources to find the best one (or set of resources) for current job
	6. If it is needed (according to job description) it finds an appropriate executable for the chosen resource
	7. If it is needed the system transfers files (data or/and executable) using File Transfer System
	8. If a job submission is successful the job *id* is registered and returned to the Application

Alternative Courses:

- Line 2: Permission denied to carry out operation. Return error to the Application
- Line 3: Invalid job description. Return error to the Application
- Line 4: Failure in calling Information Service. Return error to the Application
- Line 6: Fail to get proper executable. Return error to the Application
- Line 7: Fail to call File Transfer System. Return error to the Application

Use case:	**Migrate Job**
Actors:	Application (initiator), File Transfer System, Authorization System
Purpose:	Migrates job from one resource to another
Overview:	Application calls a function responsible for job migration. The system submits a job to a new resource according to a job description given by application.
Type:	Primary

Typical Course of Event:

Actor Action	System's Response
1. Use case begins when the Application calls "migrate job" function with migration description as argument.	2. Authorizes operation for a user who runs the Application, using the Authorization System
	3. Parses migration description to gather information needed for submission to new resource
	4. Finds potential resource for job execution according to job description (for finding potential resources Information System is used)
	5. Evaluate resources to find the best for current job
	6. If it is needed (according migration description) finds appropriate executable for chosen resource
	7. Transfers files (data and executable) using File Transfer System
	8. If job submission is successful, changes in job ids are registered

Alternative Courses:

- Line 2: Permission denied to carry out operation. Return error to Application
- Line 3: Invalid migration description. Return error to Application
- Line 4: Failure in calling Information Service. Return error to Application
- Line 6: Fail to get proper executable. Return error to Application
- Line 7: Fail to call File Transfer System. Return error to Application

Use case:	**Find Best Resource**
Actors:	Application (initiator), Information System, Authorization System
Purpose:	Finds the best resource according to given description
Overview:	Application calls function responsible for finding best resource. The system finds the best resource that fulfill Application's requirements
Type:	primary

Typical Course of Event:

Actor Action	System Response
1. Use case begins whe n Application calls "find best resource" function with resource description as argument.	2. Authorizes operation for user who runs Application, using Authorization System
	3. Parses resource description
	4. Using Information System to find potential resources
	5. Evaluate resources and chooses the best
	6. Return result to Application

Alternative Courses:

- Line 2: Permission denied to carry out operation. Return error to Application
- Line 3: Invalid resource description. Return error to Application
- Line 4: Failure in calling Information Service. Return error to Application

Use case: **Remove Job**

Actors:	Application (initiator), Authorization System
Purpose:	Destroying a job submitted to Grid Environment
Overview:	Application calls function responsible for killing a job. The system finds the resource where the job is running and destroys it
Type:	primary

Typical Course of Event:

Actor Action	System Response
1. Use case begins when Application calls "remove job" function with job identifier as argument.	2. Authorizes operation for user who runs Application, using Authorization System
	3. Finds job id on resource where the job was submitted, according to id provided by Application
	4. Destroys the job

Alternative Courses:

- Line 2: Permission denied to carry out operation. Return error to Application
- Line 3: Invalid job id. Return error to Application

Use case: **Suspend Job**

Actors:	Application (initiator), Authorization System
Purpose:	Suspending a job submitted to Grid Environment
Overview:	Application calls function responsible for suspending a job. The system finds the resource where the job is running and suspends it
Type:	primary

Typical Course of Event:

Actor Action	System Response

1. Use case begins when Application calls "suspend job" function with job identifier as argument.	2. Authorizes operation for user who runs Application, using Authorization System
	3. Finds job id on resource where the job was submitted, according to id provided by Application
	4. Suspends the job

Alternative Courses:

- Line 2: Permission denied to carry out operation. Return error to Application
- Line 3: Invalid job id. Return error to Application

Use case:	**Resume Job**
Actors:	Application (initiator), Authorization System
Purpose:	Resuming a suspended job
Overview:	Application calls function responsible for resuming a job. The system finds the resource where the suspended job is running and resumes it
Type:	primary

Typical Course of Event:

Actor Action	System Response
1. Use case begins when Application calls "resume job" function with job identifier as argument.	2. Authorizes operation for user who runs Application, using Authorization System
	3. Finds job id on resource where the job was submitted, according to id provided by Application
	4. Resumes the job

Alternative Courses:

- Line 2: Permission denied to carry out operation. Return error to Application
- Line 3: Invalid job id. Return error to Application

Use case:	**Predict Job Execution**
Actors:	Application (initiator), Information System, Authorization System
Purpose:	Predicting job execution on specific resource
Overview:	Application calls function responsible for predicting job execution. The system according to job description given by Application the system predicts job execution (start time, end time, run time) on specific resource
Type:	primary

Typical Course of Event:

Actor Action	System Response
1. Use case begins when Application calls "predict job execution" function with job identifier as argument.	2. Authorizes operation for user who runs Application, using Authorization System
	3. Parses job description to gather information needed for prediction
	4. Finds resources using Information System, according job description
	5. Calculate prediction parameters for described job and resources
	6. Return information to Application

Alternative Courses:

- Line 2: Permission denied to carry out operation. Return error to Application
- Line 3: Invalid resource description. Return error to Application
- Line 4: Failure in calling Information Service. Return error to Application

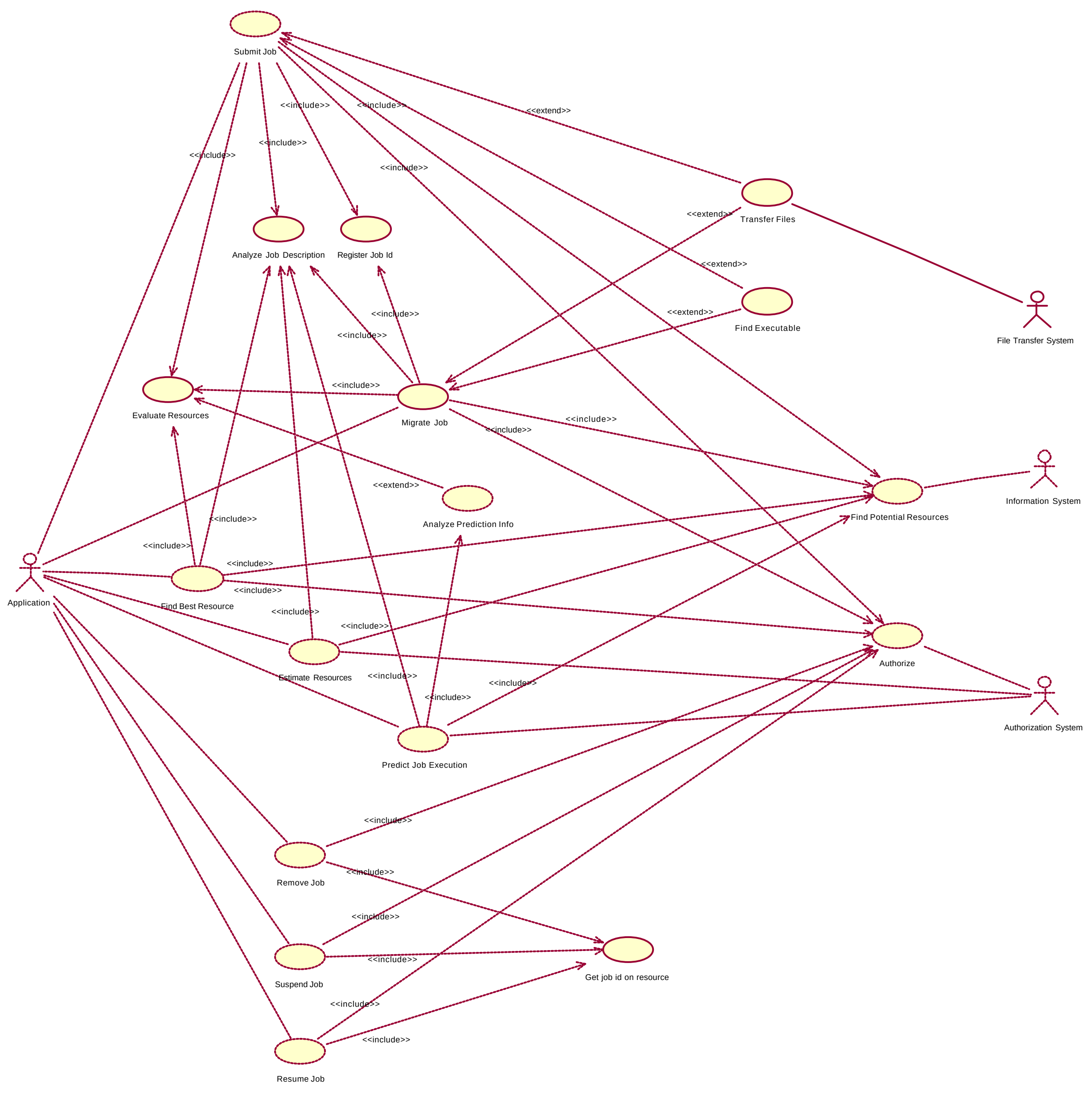
Use case:	**Estimate Resources**
Actors:	Application (initiator), Information System, Authorization System
Purpose:	Providing resource estimation information
Overview:	Use case begins when Application calls a specific function of estimating resources. The system estimates the resource needs for job (e.g. what resources are needed to finish job before deadline)
Type:	Primary

Typical Course of Event:

Actor Action	System Response
1. Use case begins when Application calls "estimate resources" function with job identifier as argument.	2. Authorizes operation for user who runs Application, using Authorization System
	3. Parses resource description
	4. Using Information System to find potential resources
	5. Evaluate resources and chooses the best
	6. Return result to Application

Alternative Courses:

- Line 2: Permission denied to carry out operation. Return error to Application
- Line 3: Invalid resource description. Return error to Application
- Line 4: Failure in calling Information Service. Return error to Application


Submit Job
Analyze Job Description
Register Job Id
Transfer Files
Find Executable
File Transfer System
Evaluate Resources
Migrate Job
Analyze Prediction Info
Find Potential Resources
Information System
Application
Find Best Resource
Estimate Resources
Authorize
Authorization System
Predict Job Execution
Remove Job
Suspend Job
Get job id on resource
Resume Job
<<include>>
<<extend>>

5. Some other requirements

The GridLab application and usage scenarios requirement's analysis, the GAT requirement analysis as well as a number of requirements identified by the Scheduling Working Group of the Global Grid Forum force us to design a GridLab Resource Management System which will have the following features:

- 10 step superscheduling: The system will support the 10 step superscheduling proposed by the GGF (Jenny Shopf), with some modifications allowing to support the apps requirements for dynamic grid computing,
- Flexible Control: System is able to allow its clients (components of this system as well as the applications, users, scripts, portals) to control their consumption of potentially remote Grid resources. In addition to individual resource access, this control must support coordinated use of resources in different administrative domains as well as different virtual organizations.
- Extensibility: The system must provide dynamic access to new resource types and/or access modes not necessarily recognized by all clients or managers. This will be done be introducing the standard Grid Resource Management Protocol.
- Notification: The ability to monitor the status of resources, user jobs and resource acquisition attempts as this is essential to the use of resources in dynamic environments. As such, the notification system should be asynchronous and it should be able to allow the system administrator to differentiate the importance of the status changes and further, based on this different status changes, construct the notification rules and events. This will be done together with WP11.
- Reliability: The system must provide a reliable semantics to satisfy clients ranging from interactive and collaborating users and their applications, to fully-automated resource brokers.
- Negotiation mechanisms: the system must provide some language and protocol mechanisms to allow any user and/or agent working on his behalf to negotiate various things, e.g. software versions, required resource features, important criteria from the user's and administrator's perspectives, QoS, reservations.
- Hierarchy: Many resource requests (acquisitions) are managed hierarchically, e.g. a single reservation may be subdivided and applied to multiple simultaneous or sequential resource allocations. The system must provide uniform access and control for such hierarchical management scenarios.

- Security: The system must provide the secure transfer of all user data as well as the messages between the system's components. The integrity-checked delivery of all data and message transfers over entrusted networks is an important requirement here.
- The system must embed a policy language to allow the expression and exchange of complex security requirements and policy, e.g. to represent a delegation of *agent rights* with *limited trust.*
- Resource language: The system must provide some structured language to allow a expression and exchange of complex resource requirements and configuration data.
- Standard API: The system must be based on the standard API as well as standard protocols to enable code reuse and compliance with the standard grid protocols designed by the Global Grid Forum.

- GRMS must be able to "talk" to the other middleware components, such as *monitoring and logging system, job id management service, job monitoring service.* Some of these components are an integral part of the system, but, in principal, can be used by the middleware services developed in other work packages.
- GRMS must be able to control the dynamic grid applications by providing the mechanisms for application checkpointing, migration, and spawning, taking into account the current application performance and changing grid environment.
- GRMS Will be able to run applications on all resources of (possible multiple) virtual organizations in a secure and trustable way,
- GRMS will enable efficient and effective use of resources by providing efficient scheduling decisions and algorithms, use of timed reservations and prefetching ,
- GRMS will be fault-tolerant,
- GRMS will provide the support for mobile users, allowing offline job preparation using mobile devices and then, when connected to the Grid, processing the job. Other requirements will be developed within WP12.
- GRMS will enable to get the applications running on machines with no grid infrastructure and migrate them to the fully deployed grid environments (full VOs) by means of GridLab GAT. As soon as the application connects to the GridLab testbed it must use the GRMS to fully exploit the grid capabilities. When the application connects to the Grid, it is the GRMS which controls the application run in the environments, providing to the application the resources and performance on the best effort.
- GRMS will hide the complexity of underlying grid core services.
- GRMS will provide the support for collaborative users,

- GRMS will provide an application programming model (the way of developing the applications) to fully exploit the global grid.
- GRMS will allow a user to express her/his preferences regarding various time and cost criteria based on which the schedules might be evaluated against all the other criteria, especially the VO admin or local admin criteria.
- GRMS will provide availability to test and evaluate various scheduling approaches.

6. Relations with other workpackages

The GRMS cannot be designed efficiently without collaboration of other grid middleware workpackages of the GridLab project. The most important collaborative workpackages are the following:

- **WP1, 2, 3, 4 (GAT, CGAT, TGAT, Portals) –** GRMS will follow all the requirements form these workpackages. It is assumed that all these workpackages will provide WP9 with the usage scenarios for applications and portals.
- **WP6 – Security**: GRMS will follow all the security requirements and all the operations using GRMS or performed by GRMS will go through the security infrastructure. WP9 will work closely with WP6 to design the Community Authorization System enabled services, i.e.
- **WP7 – Adaptive Components**: WP9 will use the following estimation or current information coming from the WP7 workpackage:
  - CPU speed on individual machines,
  - Network bandwidth, latency, jitter,
  - I/O bandwidth between CPUs and local disks,
  - Predictions of the future resource load estimations,
  - Timed estimations for jobs and queues.

  The technical specification of the information needed from the WP7 will be made in the next deliverable.
- **WP8 – Data Management**: WP9 will use WP8 services for transferring the files and jobs between remote machines. The technical specification of functionality needed from the WP8 will be made in the next deliverable.
- **WP10 – Information Services:** WP9 will specify the additional information providers **(IP)** required form the resource management point of view. This will be made in the next deliverable.
- **WP11 – Monitoring:** WP9 will collaborate with WP11 to define exactly what resources have to be monitored. The technical specification WP9-WP11 interrelations will be presented in the next deliverable.

- **WP12 – Mobile user support:** Although the WP12 requirement analysis is coming in month 6, WP9 will collaborate very strongly with WP9 to deliver the technical specification for mobile users support in month 6.