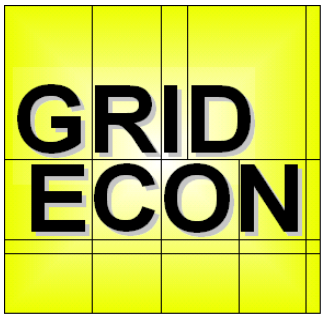

Project Number:	**FP6 – 2005 - IST5 – 033634**
Project Title:	Grid Economics and Business Models
Deliverable Number and Title:	**D4.1: Validation Scenario Report**
Contractual Date of Delivery to EC:	M12 2007
Actual Date of Delivery to EC:	M13 2007
WP contributing to the Deliverable:	WP4
Nature of the Deliverable:	R
Deliverable Security Class:	PU
Editor:	ATC
Current Version:	v1.0

Abstract: The main objective of this deliverable is to sketch the validation scenarios that will be performed once the GridEcon components have been implemented and integrated. These scenarios are based on the initial scenario report (D1.1) and business model report (D1.3). While the validation scenarios do not implement the precise description found in the documents coming from WP1, they aim to address the main issues identified in the WP1 scenarios and quantify the extend to which the newly developed components of WP4 better the performance and utilization of existing Grid platforms, from an economic point of view.

Quality assurors and contributors

Internal reviewers	Jörn Altmann (IU), Costas Courcoubetis (AUEB), Dimitris Sotiriou (ATC)
Contributor(s)	ATC, RTEL, ICL, LCMG, GISP, AUEB, EY

Version history

Version number	Description
0.1	Structure Outline
0.2	Use Cases
0.3	New Structure Outline
0.4	New Structure Outline
0.5	New Structure Outline
0.51	Revised Section 3 and 4
0.6	Section 3.8 and 4.5
0.61	Section 3.7
0.62	Added Executive Summary, Introduction, Conclusion and Abbreviations
0.63	Review of existing sections and completion of missing contributions
0.64	Section 2.6 and Internal Comments
0.65	Addressed Comments
0.66	Added Reference Implementation chapter
0.7	First full version ready for comments
0.8	Revision after comments
1.0	Updated formatting to reflect GridEcon standards, complete version for approval

Executive Summary

This main objective of this deliverable is to sketch the validation scenarios that will be performed once the GridEcon components have been implemented and integrated. These scenarios are based on the initial scenario report (D1.1), the business model report (D1.3), and the deliverables of WP3. The validation scenarios aim to address the main issues identified in the WP1 scenarios and quantify the extend to which the newly developed components of WP3 improve the performance and utilization of existing Grid platforms, from an economic point of view.

In order to assist in the development and integration of the GridEcon reference architecture, we provide in this deliverable an implementation plan describing the methodology that we are going to follow for achieving our goal. In addition, we look into three distinct validation scenarios, providing details of what we want to showcase through them and how we are going to implement them. We then sketch a set of metrics for each validation scenario in order to be able at the end of the project to assess our work and, if required, revise either our architecture or our implemented components in order to meet our initial objectives. Finally, we present the first reference implementation of the GridEcon system and what will be the first "sprint" for accomplishing this.

List of Abbreviations

Abbreviation	Explanation
HPC	High Performance Computing
PC	Personal Computer
QoS	Quality of Service
SLA	Service Level Agreement
SME	Small and Medium-Sized Businesses
FLOPS	Floating Point Operations Per Second
WSDL	Web Services Definition Language
VCG	Vickrey-Clarke-Groves
UoT	Units of Trade

Table of Contents

Table of Figures

1 Introduction

The implementation of the GridEcon architecture is the objective of WP4 and one of the key milestones of the GridEcon project, as it will allow us to demonstrate the applicability and validity of all the work and research conducted in the other WPs. This will require a well-planned set of actions and the collaboration of all the partners in materializing all the knowledge, from business aspects to economic modeling, into a set of components/services.

In this deliverable we will try to describe at foremost the approach that we will follow in order to succeed our goal of providing economic-aware Grid components. In order to do this we will provide a description of the development methodology that we are going to follow, along with details regarding the technical and programming choices that we have made.

Along with the above information we will also describe in this document three distinct scenarios that are going to be validated at the end of the project. These validation scenarios are based on the input of WP1 and WP3 and provide an indicative set of functionalities that the GridEcon architecture is able to provide.

We then provide a set of metrics, based on which we will evaluate the results of our work. These metrics will allow us to assess the degree to which the GridEcon implementation meets our initial objectives and provide means to further improve and fine-tune the GridEcon components/services.

Finally, we give a reference implementation of the architecture described in WP3. This implementation provides a more technical approach into how we will accomplish to materialize the research and economic aspects investigated in WP2 combined with the open architecture designed in WP3. In addition, we give an overview of our first "sprint" (according to the SCRUM methodology) for achieving and implementing the validation scenarios. This reference implementation is based on the architecture of WP3 and is open enough to showcase all three validation scenarios described in Section 2 of this deliverable. By incrementally building all the services and components of WP3 into this reference implementation, we will be able at the end of the project to provide the full version of the GridEcon system and validate all three scenarios.

2 Implementation Strategy

2.1 *Introduction*

This section of the document describes how GridEcon plans to build and validate an implementation of the reference architecture. We begin by discussing the rational behind our approach, and then discuss practical design and implementation issues such as the development and execution environment, reference architecture, and source versioning. The remainder of the section describes the implementation of the three chosen scenarios.

2.2 *Approach*

The GridEcon reference architecture has many different components and players, each of which may communicate with many other components, and may indeed modify their behavior dynamically to match the requirements for different customers and to take into account the market forces in place at the time. It is not possible for GridEcon to build a complete implementation to demonstrate the market - the number of permutations and combinations is just too large. Instead we will focus on a limited number of validation scenarios and build the components necessary to demonstrate these scenarios. Another way of looking at this is to say that a validation scenario can be thought of as a problem to be solved or a goal to be reached, with many different potential solutions or paths available, and that GridEcon has chosen one fixed path through the reference architecture for each scenario.

We have made some assumptions in the design and development process. These are:

- All components are Web services
- All published interfaces are defined using the Web Services Definition Language (WSDL) [1]
- All components can be built, tested and deployed in the GridEcon development and execution environment described below

2.3 *Development and Execution Environment*

The GridEcon development and execution environment uses popular open source tools and packages to minimize the cost for developers. The initial environment comprises:

- Development language – English
- Source code repository – Subversion running on the GridEcon Web server
- Programming language / development kit - Java SE SDK version 6.1
- Web server - Apache HTTP Server version 2.2.4
- J2EE container - Apache Tomcat 6
- Build Scripting - Ant version 1.7.0
- Database - MySql 5.0 Community Server Edition
- Unit testing – Junit
- Documentation – JavaDoc

2.4 *Building and Deploying Technologies*

GridEcon will develop Ant scripts to perform the following operations:

- Automatically pull latest source files from Subversion
- Build all components, including Web service stubs from WSDL documents
- Build test database(s)
- Run regression tests using JUnit
- Compile documentation using JavaDoc
- Deploy components to local Web server/J2EE container

2.5 *Source Versioning*

GridEcon uses Subversion as a source code repository. This is configured as follows:

- Remote access – remote access is available to GridEcon project members via HTTPS
- Main Development Branch – contains the latest software. It does compile but has not necessarily been fully integration tested
- Release Branch(es) – contains a snapshot of the reference architecture and any test/demo applications, fully tested and signed off by the WP4 leader (or appointed designate)
- Restricted Access – to properly manage access, all members of the GridEcon project have read-only access to Subversion, and a sub-set of representatives from each partner have write access

2.6 *Auction-R/G/B Service Area*

2.6.1 Validation Scenario Description

This scenario involves conducting an auction for trading compute resources in grids. In a Grid, products correspond to computational services being offered on top of hardware that users buy (rent) over a period of time. The compute resources in the grid are not homogeneous, a machine (or set of machines) may be faster, have more memory or disk storage, access to greater bandwidth, or other features that distinguish it from other machines (or sets).

GridEcon will represent computational resources as Cybernodes[1]. Each Cybernode represents a specific compute resource with qualitative and quantitative capabilities. The aggregation of a Cybernode's capabilities will represent a compute resource as a commodity that can be bid on.

We assume that there are two types of Cybernodes with guaranteed performance, and one type of Cybernode that offers best effort. Namely, there are Cybernodes offering a higher level of QoS (H) level, and Cybernodes offering a lower level of QoS (L). Cybernodes belonging to the same type are assumed identical in terms of the associated performance guarantees, while the Cybernodes of the high level are superior to those of the low level. We also assume that there are a number of

[1] A Cybernode is a term used by the Rio project, and is used to represent a compute resource as a service available through the network

Cybernodes in general do not suffice to satisfy the demand for job execution. Therefore, we take that the excess demand is served by a best-effort physical machine, which can be highly congested, at zero or at a fixed low price per task.

This scenario will produce a mechanism for simultaneously auctioning multiple objects, namely, a multi-object sealed-bid one-round Vickrey-Clarke-Groves (VCG) auction. Specifically, the auction addresses a single-unit demand with two guaranteed types of Cybernodes (outlined above). The auction consists of two phases, namely the resource allocation (winner determination) phase and the payment (service charge) computation phase.

2.6.2 Objectives

The objectives of this validation scenario are:

- Provide periodic auctions can be used for resource allocation; that is, allocate Cybernodes to those users that value them the most
- Create a dynamic market for grid compute resources

2.6.3 Use Case diagram

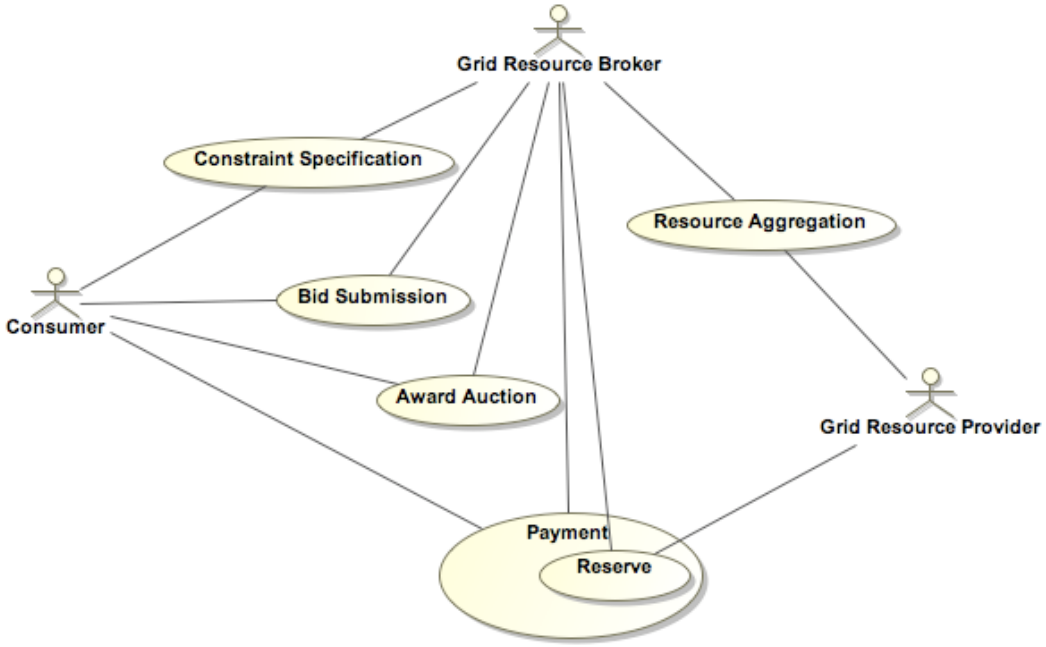

Figure 1 Use case for R/G/B scenario

2.6.4 Actors Involved

Actor	Role
Consumer	The Consumer bids on compute resources with specific QoS constraints
Grid Resource Broker	The Grid Resource Broker conducts the auction. The Grid Resource Broker accepts set of requirements from the user, aggregates available compute resources from Grid Resource

	Suppliers, and brokers the reservation and payment between consumers and suppliers
Grid Resource Supplier	The Grid Resource Provider advertises available compute resources, reserves them for users that wish to pay for them

2.6.5 Services Involved

This scenario will include a subset of the services from the GridEcon reference architecture. The services listed below are described as Level 1 Basic Resource Services, they are:

- Resource Broker
- Resource Scheduler
- Resource Monitoring
- Resource Metering, Accounting, and Charging
- Resource Allocation and Management
- Utility Computing Management System
- Virtualization (including metering)

2.6.6 Application Details

Application execution will involve a distributed system composed of software components that implement the services described above. A user interface will be created allowing consumers to provide the constraints to brokers specifying the type(s) and criterion of compute resources they wish to purchase. Grid compute resources will be represented by software components, and discovered through the network, on a suitable and representative grid of compute assets.

The executable system will be primarily Java technology based, and utilize Internet and open standards computing technologies to deliver the scenario as a working system.

2.6.7 Sequence diagram

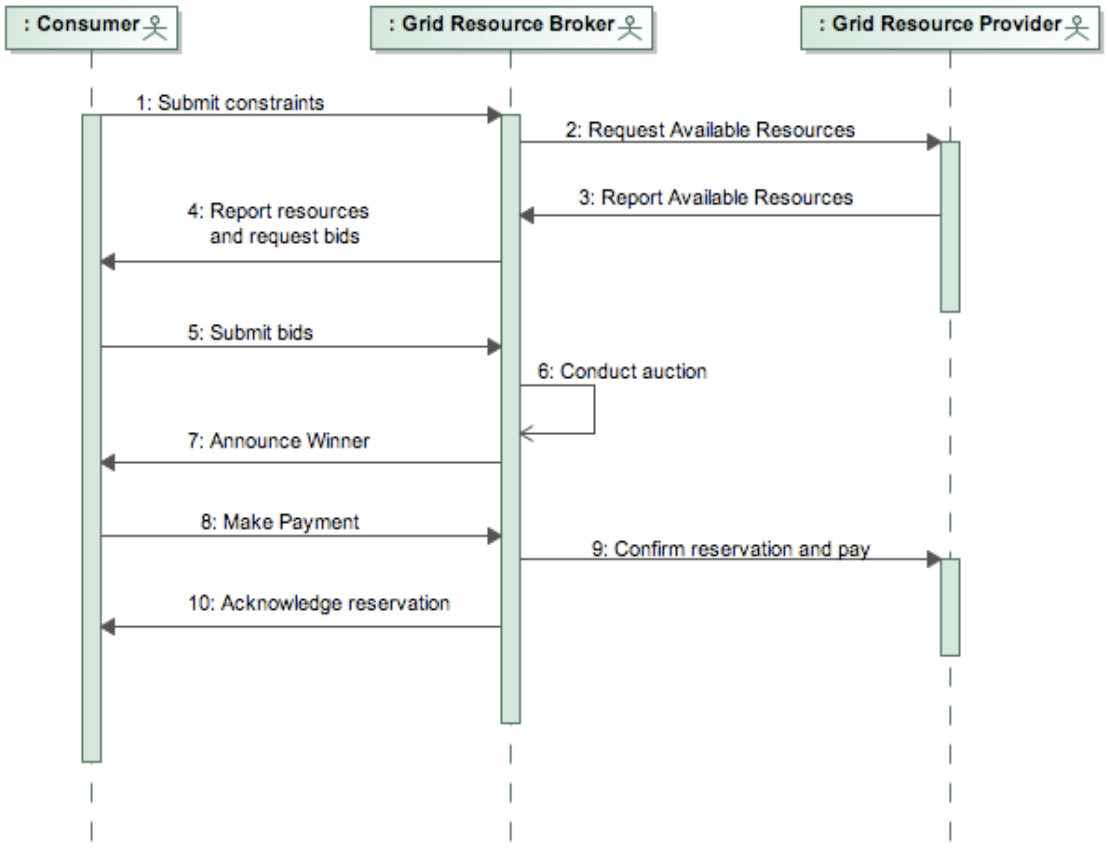

Figure 2 Sequence diagram for R/G/B scenario

2.7 *Reservations-Future Market Area*

2.7.1 Validation Scenario Description

This validation scenario concerns three areas of Utility computing that are interlinked:

- Spot reservations
- Forward reservations
- Options (where you buy the right to buy (or sell) resources)

The latter is a sign of a mature (and liquid) market. The interconnection becomes clear when you start thinking about it. Anything beyond spot reservations becomes forward reservations or even options on reservations. Each of these types has its own questions as you can see in the table below:

Type of reservation	Time	Remarks
Spot	Within x minutes, the resources will be allocated for a certain time period.	There are a number of issues like pricing, wake up time, deployment time etc
Forward	Any thing beyond a Spot	How do you price forward

	reservation with no theoretical time limit	reservations? Time is an important factor from the premise that capacity delivered in the future are priced differently than capacity delivered now
Options	Anything beyond spot reservations with again no theoretical time limit	How do you price an option where the user has the right to buy (or sell) but not the obligation?

These three areas are strongly connected with each other. The number of forward reservations influences the available spot capacity. The number of options influences forward reservations and spot capacity (when the option matures and capacity is not needed).

This validation scenario relies heavily on other scenarios and components and, in our opinion, solely focuses on the pricing of forward reservation and options. Spot markets can use any auction mechanism currently being looked at by other work packages (and analyzed in the third validation scenario). What is a challenge is the wake-up and sleep mode of resources vs. the cost and economic value of these resources. In other words, when do you turn off resources (potentially losing revenue but incur less cost than when running (and how much cost)) and when do you wake them up (i.e. when is demand reaching a threshold level (and how high should it be) to wake systems).

Especially in the area for options, there are a number of challenges. Since options rely on a mature market, where the underlying values (i.e. the capacity) are available, we will probably have to simulate the market. Apart from pricing, optimization of reservations against options (and the likelihood of delivery at maturating date) is another challenge that requires an algorithm than can maximize revenue.

2.7.2 Objectives

The objectives of this validation scenario are:

- Develop a pricing mechanism for all three areas (premise is that spot market use auction mechanisms and will be mainly validate in the third scenario)
- Test the validity of the pricing mechanism by simulation
- Determine the optimization of reservations vs. options, wake up vs. sleep time and others

2.7.3 Use Case diagram

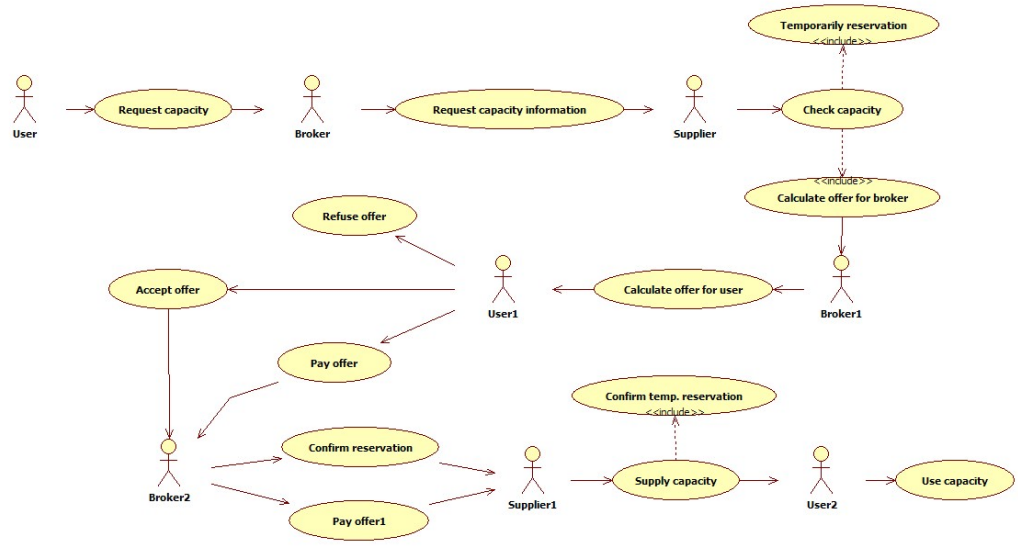

Figure 3 Use case for Spot market

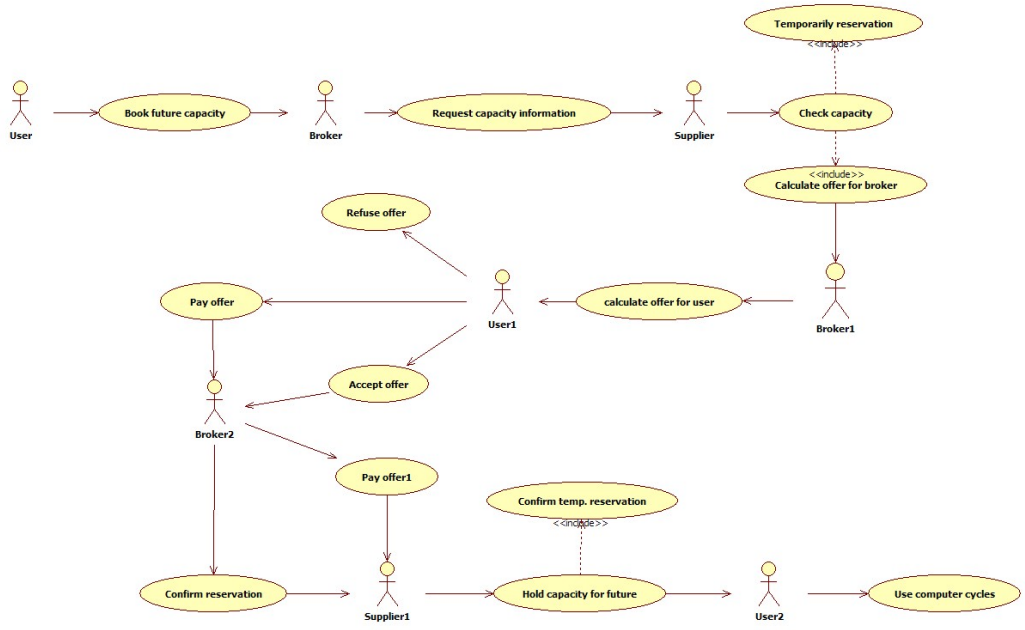

Figure 4 Use case for forward market

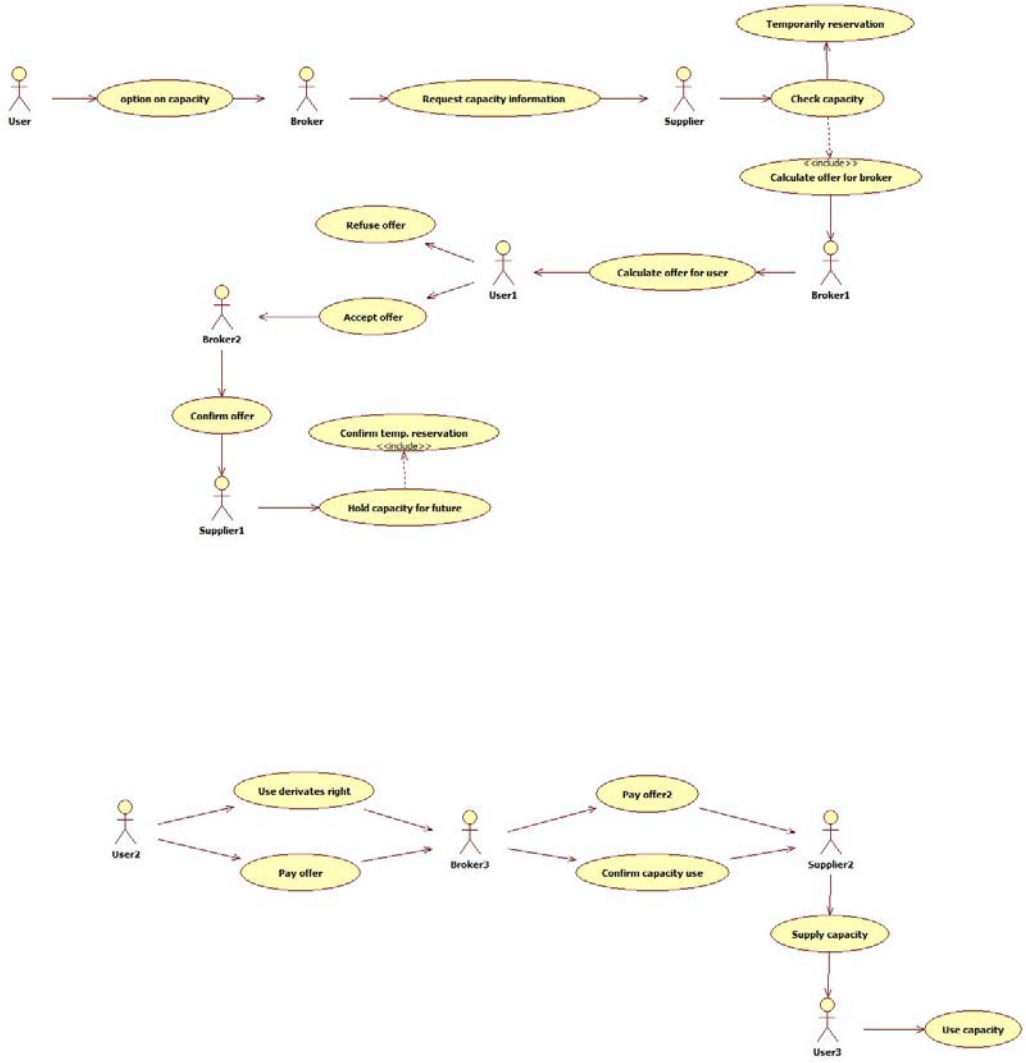

Figure 5 Use case model for derivatives

2.7.4 Actors Involved

The actors involved are:

- User(s) - provide a set of requirements to the Grid Resource Broker, obtain prices dependent on the market conditions and get their jobs executed
- Grid Resource Broker – provides a service which matches user requirements to available Grid resources, returns a price to the User and charges the user for this service
- Grid Resource Providers – provides the underlying Grid resource, returns a price to the Grid Resource Broker, and charges the broker for the resources used

For the purposes of this scenario, we have assumed that the following services are provided by the Grid Resource Broker (and are not separately charged):

- Resource Scheduler
- Resource Monitoring

- Resource Metering, Accounting, and Charging
- Resource Allocation and Management

2.7.5 Services Involved

Figure 9 shows the subset of components from the GridEcon reference architecture required to implement the scenario (highlighted in yellow in the diagram). These are:

- Resource Broker
- Resource Scheduler
- Resource Monitoring
- Resource Metering, Accounting, and Charging
- Resource Allocation and Management
- Utility Computing Management System
- Virtualisation (including metering)

Separately we rely on a simulation environment to simulate the market movements.

2.7.6 Application Details

The services that will be executed within this validation scenario are:

- **Spot market**
  Use of an auction model from other work packages, also considering the possibility of turning off resources and waking them up on demand (pending an algorithm). While this market will be mainly investigated in the third validation scenario, it is a pre-requirement for the forward and option market and as thus included in this scenario
- **Forward market**
  Calculation of pricing for forward reservations (including overrun and pricing of subsequent timeslots)
- **Option market**
  Calculation of option prices, optimization of the options to reservations ration and other services

2.7.7 Sequence diagram

The following two figures provide a general sequence view of the previously described use cases. The first one illustrates the case of a rejected offer while the second one that of an accepted.

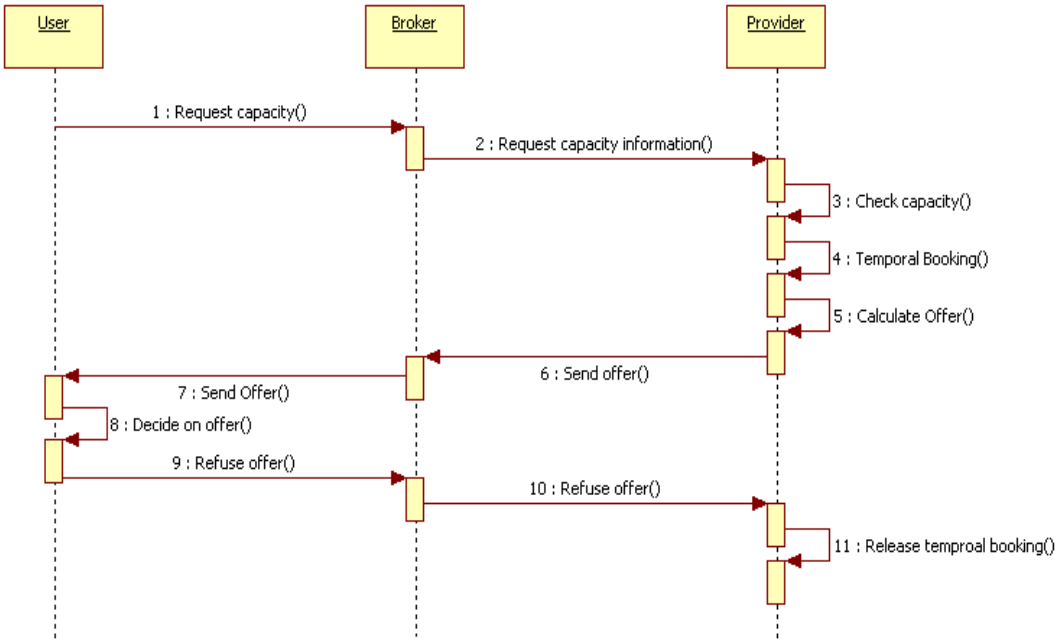

Figure 6 Sequence diagram for rejected offer

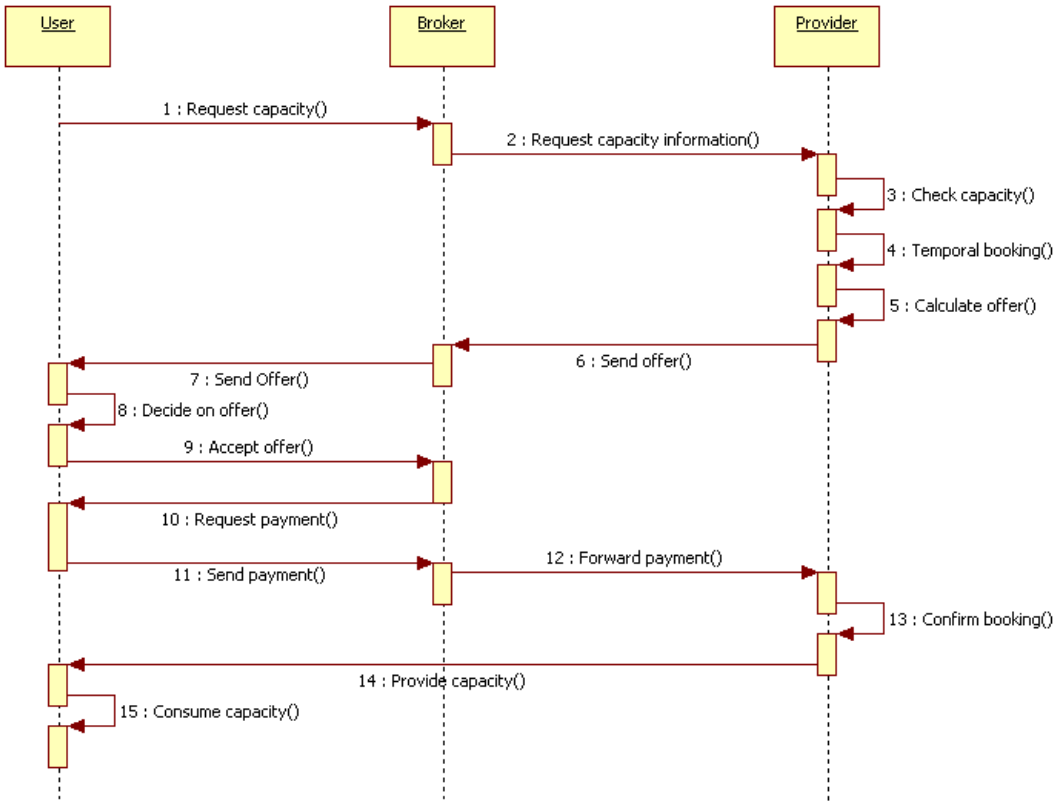

Figure 7 Sequence diagram for accepted offer

2.8 *Utility Computing-Interconnection-Resource Mapping Area*

2.8.1 Validation Scenario Description

Utility Computing is the provision of remote execution within compute or storage environments, by third party vendors. The provisioning of such services is normally realized by large localized federations of servers or clusters, in order to support the computational requirements of customers on a use-per-demand/pay-per-use basis. Utility Computing dramatically lowers the cost of ownership (e.g. for HPC users) and has many other benefits accruing from specialization, centralized management and economies of scale. The Sun Microsystems N1 Grid Engine [2] Pay-per-Cycles facility and the Amazon Elastic Compute Cloud (EC) [3] use Grid computing technology to provide Internet-available execution and storage platforms that can be acquired and used on a pay-per-use basis. Furthermore, the development of transparent application mapping and job submission systems (e.g. GridSAM [4]) means that such execution platforms become interchangeable or substitutable as applications can be mapped or adapted to alternative platforms. This in turn makes the execution of applications a tradable good/commodity, opening up the potential for markets in compute cycles or storage.

In Utility Computing environments, it is becoming increasingly important to have a system/vendor independent architecture for the provisioning of resources. Users and software providers will only need to specify QoS constraints on the application to the resource broker which will then determine resource assignments based on market conditions, user constraints etc. The pricing, metering, charging and accounting services need to be efficient and scalable. This allows the utility computing provider to bill the resource broker/users/software provider for resource usage.

This validation scenario describes the situation where the GridEcon reference architecture is used to implement a market for Grid resources, where several users require allocating appropriate Grid resources based on differing sets of requirements, and several resource providers exist with differing resource profiles and charges. The idea is to show that market conditions prevail when:

- With several users with different requirements and a market consisting of a finite number of resources, the price differs in proportion to the availability of the resources at the time a new user is introduced
- With several users with differing requirements and a market consisting of an infinite number of resources, the price differs in proportion to the user requirements
- With several users with differing requirements, and a market consisting of a finite number of resources, the price differs in proportion to both the user requirements and the resources available at the time a new user is introduced
- With several users with differing requirements, and a market consisting of a number of providers with differing cost (and therefore charge) profiles, the price at any one time is dependent on the requirements of a new user, the level of resource usage at the time, and the pricing profiles of the providers

- With several users with differing requirements, and a market consisting of a number of providers, some providing guaranteed SLA, some not, the price at any one point in time depends on a combination of the user requirements and the level of resource usage at the time. When new users are added and all resources are currently in use, we should find that the new users have to pay a premium in order to obtain the resources, and that they displace existing based on the value of its user (willingness to pay).

2.8.2 Objectives

The objective is to demonstrate that the GridEcon reference architecture can:

- Map available resources to the requirements
- Dynamically provision and schedule resources to meet changing demand
- Integrate with existing utility computing providers
- Be used to implement a market place for Grid resources
- Transparently carry out job execution using job submission systems
- Be applied in scenarios such as University Utility Computing, Accessible Supercomputing for SME, Brokerage of Enterprise Computing and Storage Capacity
- Validate the pricing, accounting and charging mechanisms required for utility computing environments

2.8.3 Use Case diagram

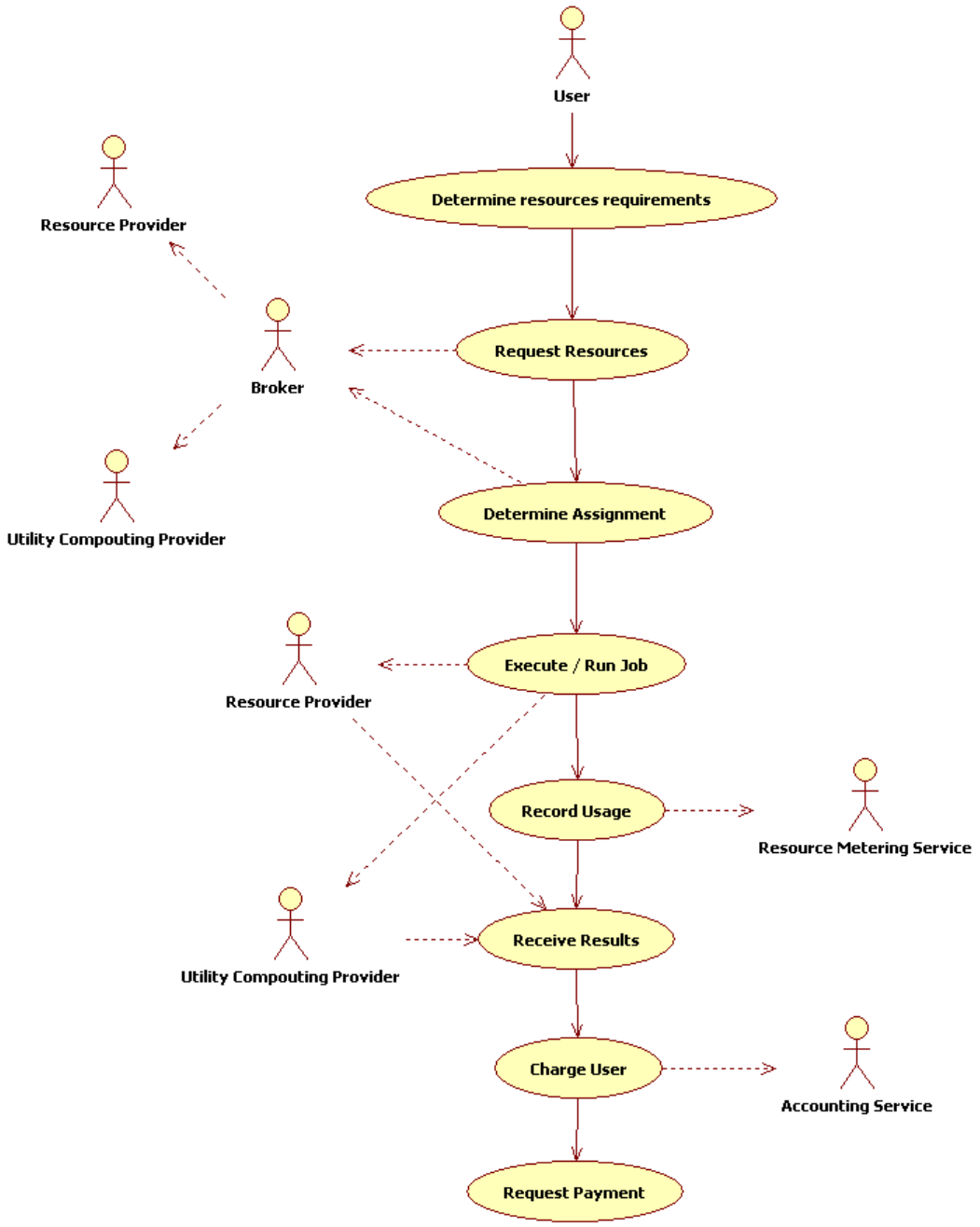

Figure 8 Use Case for Utility Computing

2.8.4 Actors Involved

The actors involved are:

- User(s) - provide a set of requirements to the Grid Resource Broker, obtain prices dependent on the market conditions and get their jobs executed

- Grid Resource Broker – provides a service which matches user requirements to available Grid resources, returns a price to the User and charges the user for this service
- Grid Resource Providers – provides the underlying grid resource, returns a price to the Grid Resource Broker, and charges the broker for the resources used

For the purposes of this scenario, we have assumed that the following services are provided by the Grid Resource Broker (and are not separately charged):

- Resource Scheduler
- Resource Monitoring
- Resource Metering, Accounting, and Charging
- Resource Allocation and Management

2.8.5 Services Involved

Figure 9 shows the subset of components from the GridEcon reference architecture required to implement the scenario (highlighted in yellow in the diagram). These are:

- Resource Broker
- Resource Scheduler
- Resource Monitoring
- Resource Metering, Accounting, and Charging
- Resource Allocation and Management
- Utility Computing Management System
- Virtualisation (including metering)

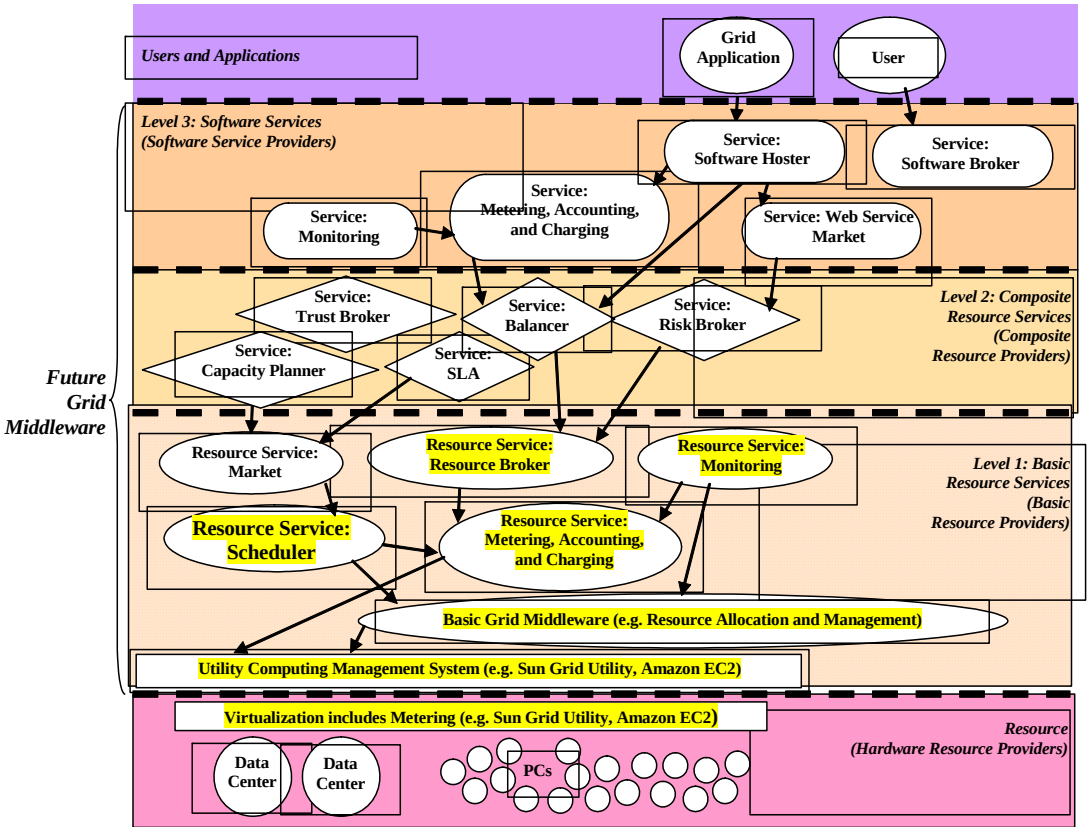

Figure 9. Utility Computing – Resource Mapping Components

2.8.6 Application Details

We plan to develop each of the identified GridEcon components sufficiently to allow the scenario to function, and in addition to develop the following test client applications:

- **Resource Broker monitoring client application**
  This client application will interface to the Resource Broker service and will allow the entry and display of all methods and attributes exposed. This will give information regarding available resources, statistics on the usage and the price of each resource, the market (spot, reservation, option) they were purchased etc. All this information will allow us to observe how resources are consumed in an open market environment
- **Sample application**
  We require an application that's simple to use, is repeatable, and can be accurately timed. We plan to use the Povray scene rendering application as it meets all these criteria

We plan to use RIO [5] as the underlying resource management system and to use the existing RIO monitoring tools to demonstrate the use of resource at the utility computing level.

2.8.7 Sequence diagram

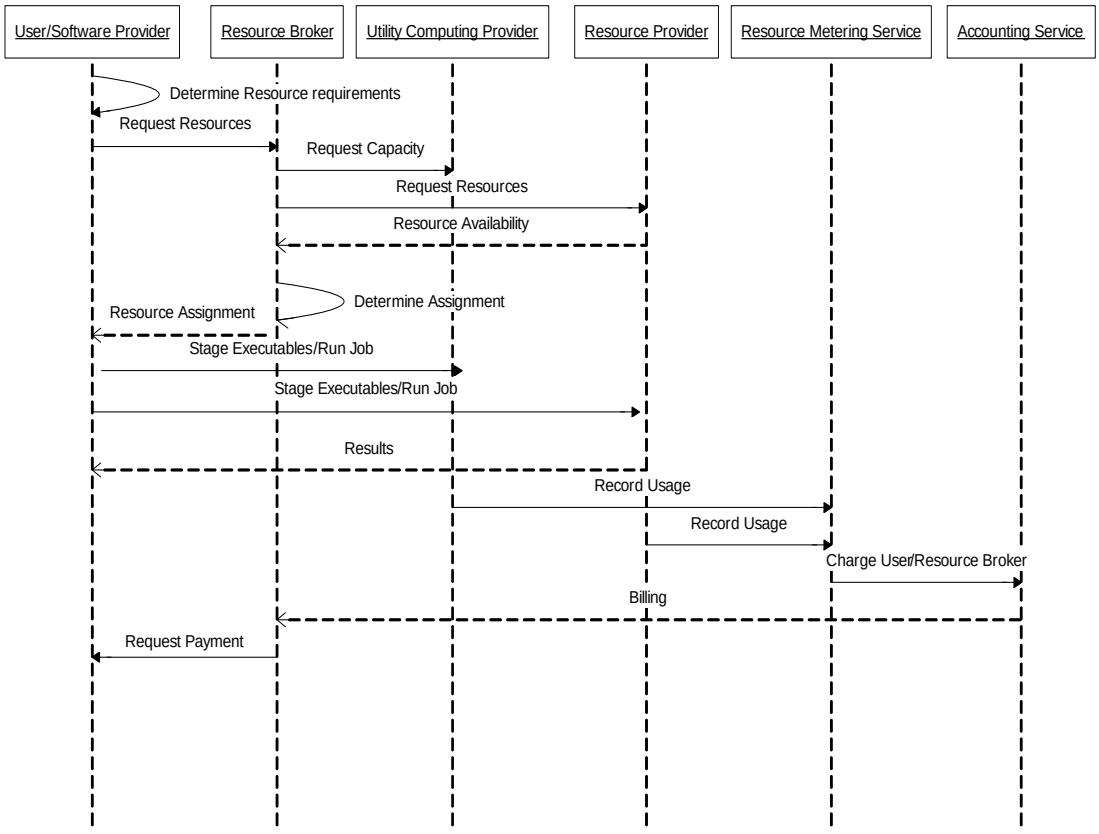

Figure 10 Sequence diagram for utility computing

2.9 *Plan*

Realizing the scenarios and use-cases described previously within the scope of a demonstrable system will help visualize and refine the semantics that the GridEcon project is pursuing. The demonstrable model will further communicate (through a working system) the core thesis of the project.

The approach taken with this effort will be to form a cross-functional team to deliver on a series of iterations that represent potentially shippable results, culminating in the final deliverable. Each of these iterations are called “Sprints”.

Each Sprint be composed of activities, some dependant on others. As much as possible the ability to run short sprints is desirable, accomplishing meaningful results that can be built on with subsequent sprints.

The approach is widely known as Scrum [6], and generally has the following characteristics:

- Users become a part of the development team
- Frequent intermediate deliveries with working functionality
- Incremental development and releases – provide the opportunity to validate and verify at shorter intervals rather than only at the end; thereby, providing time to fix, and reduce the cost to fix
- Frequent risk and mitigation plans developed by the development team itself – Risk Mitigation, Monitoring and Management (risk analysis) at every stage and with genuinity make it a live and continuous activity
- Status discussion with the team – Standup meetings (accomplishments, to be accomplished, issues / concerns / risks)
- Transparency in planning and module development – Let everyone know, who is accountable for what and by when
- No problems are swept under the carpet. No one is penalized for recognizing or describing any unforeseen problem

The Scrum methodology is illustrated in

Figure 11 [7]. As we can see, it’s basically split into three phases, the pregame, the game and the postgame. The first phase includes the initial planning and system architecture, which has been performed in GridEcon mainly by WP3 during the first 9 months of the project. The second phase is the main part of the development process and consists of a number of sprints, which include the development, wrap-up, review and adjustments of all the code developed. These sprints will be performed by WP4 and will be our main focus during the rest of the project, up to month 25. Finally, the postgame phase consists of the closing phase, which will include the validation reports and the final report delivered by WP4 at the end of the project.

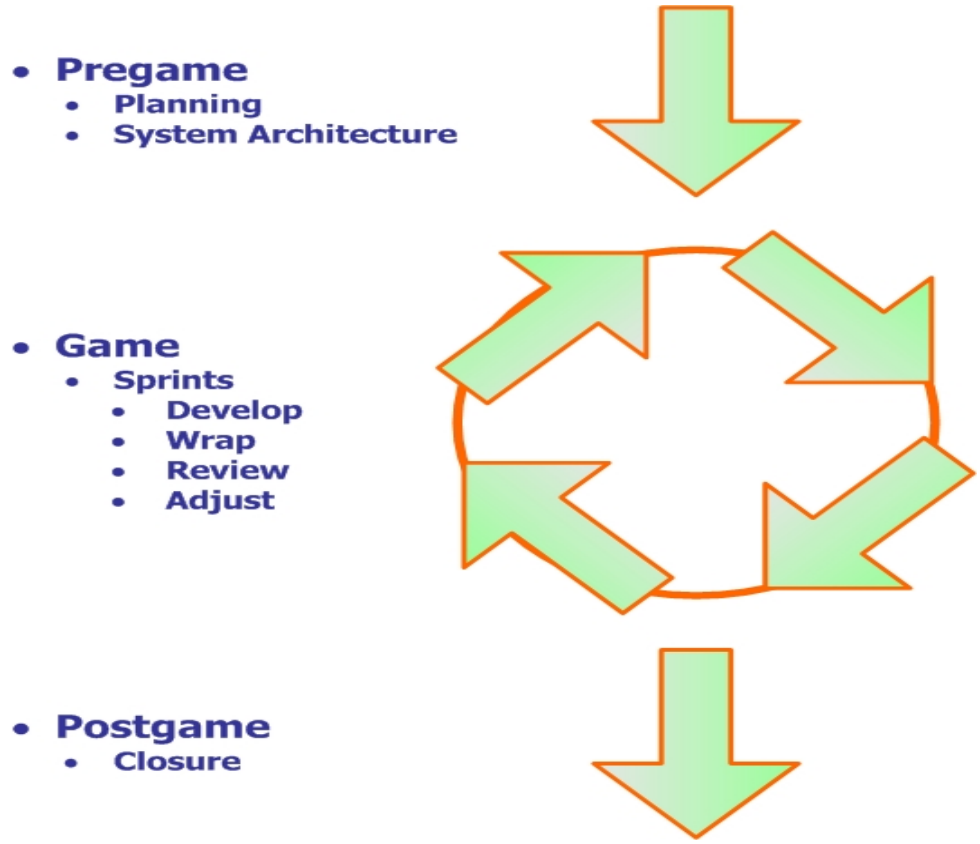

Figure 11 Scrum methodology

3 Performance Metrics

3.1 *Introduction*

This section begins by describing our approach of validating (testing) the various validation scenarios, and then describes the actual tests identified for each scenario.

3.2 *Approach to Validation*

In order to validate a validation scenario we will define a set of curves which compare predicted resource price against time with actual price against time. The predicted price can be calculated using the economic market models introduced in WP2 and is the price that is expected based on the theoretical/analytical analysis of the market. The actual price is the price observed during our simulations of the GridEcon system. The accuracy of the GridEcon model can thus be defined as a single figure equating to the Pearson Correlation Coefficient [8] where a figure close to 1 indicates a high correlation between the predicted and actual behavior.

The aim is to obtain a high correlation between predicted and actual behavior for a particular scenario. If we find that there is a low correlation then time permitting we will take corrective action and rerun the scenario. We anticipate that this will be an iterative process, repeating until the correlation is satisfactory.

The Pearson Correlation Coefficient can be calculated using the following formula:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\,\sqrt{n\sum y_i^2 - (\sum y_i)^2}}.$$

, where:
r_{xy} denotes the Pearson Correlation Coefficient
x is the predicted price
y is the actual price

Using this method, there are two possible results:

- High Correlation – the behavior predicted by the GridEcon market model closely relates to the actual behavior exhibited by the reference architecture. No further work is required
- Low Correlation – the predicted and actual behavior differ significantly. Remedial work is required.

A low correlation may be caused by one or more of the following factors:

- The economic model is inaccurate – in this case we will revisit the model and revise
- The reference architecture is incomplete – perhaps missing parameters or connections required to implement the economic model. We will revisit the reference architecture and correct as required

- The validation scenario is incomplete – the scenario doesn't fully represent the economic model (too simple perhaps) in which case we will revise the scenario

3.3 *Auction-R/G/B Service Tests*

This scenario will validate the auctioning mechanism (VCG). The validation will evaluate different bidding and auctioning mechanisms for selling resources of different quality to users in an open market. More specifically, we will validate the following:

1. During the auctions we will simulate a number of users with different demands for units of resources of different classes (quality) and investigate how the demand and supply change the price of each unit based on the different auctioning mechanisms

2. Once the auctioning has been completed and all resources allocated to users, we will investigate how past auctions affect future auctions (and the way users bid)

3. We will then investigate how the system (Rio) guarantees that all users get their resources as won in the auction and if the billing mechanism for the resources works properly

4. Finally, we will investigate how providers may alter their resource allocation into different classes (quality) of resources in order to maximize their profit based on user demand for these different classes

3.4 *Reservations-Future Market Tests*

In order to evaluate the success of this validation scenario we need to check how close the behaviour of the GridEcon system is to the expected behaviour of the model. For this, we have split the metrics into the three distinct market scenarios defined:

1. In the forward market we will evaluate how the GridEcon infrastructure permits users to book resources for future use. We will also investigate pricing mechanisms and how forward reservations affect the pricing of the spot market.

2. In our last sub scenario we will investigate how options affect the market, calculating the option prices, optimizing resources by allowing all three market options to work concurrent (spot, forward, options).

3.5 *Utility Computing-Interconnection-Resource Mapping Tests*

In this scenario we plan to run the following tests, in each case comparing the actual behaviour (measured from the results of our system validation) with the behaviour predicted by the GridEcon economic models. The output of each test will be the correlation coefficient for the test run:

1. Demand outstrips supply with one provider: graph of price against available resources for each user with 1 provider and a number of users all with the different requirements are introduced over time. We should see a flat line for price until the resources are all in use and then the price should increase as the demand rises (variable demand) as the provider is able to charge more for resources.

2. Demand outstrips supply with two providers: graph of price against cost of execution for each user with two providers and a number of users all with the same requirements are introduced over time. The two providers have different pricing strategies. To begin with we should see a flat line for price until the resources from the cheaper provider are all in use, then a stepped increase in price as resources provided by the more expensive provider are used up, and finally the price should ramp up and possibly oscillate as the two providers compete with each other.

3. Introduce resources that vary in performance – i.e. provider 1 charges X for a resource with 1 MFLOPS and provider 2 charges Y for a resource with 2 MFLOPS. So users can choose between paying less for a basic service or paying more for a premium service. So if we have a number of users with varying requirements we should see that the resources from both providers are utilized.

4. Tests will be conducted to verify the accuracy of demand predictions and capacity planning. Issues that can be investigated include burstiness in demand and resource capabilities. Furthermore, the market may experience fluctuations in demands, displaying seasonality and trend. Seasonality refers to the periodicity of the request patterns. Seasonal traffic is most often represented by the regular daily activities of the users. For example, traffic in some e-trading systems has consistent peaks and valleys each day when the market opens and closes. Seasonal traffic is also observed in monthly intervals, for example, running algorithms to collate experimental data or scheduled accounting tasks, and during designated periods, for example, the holiday season, week nights or weekends.

   Seasonal requests can degrade the performance of the Grid because, for the peak duration, large batches of requests occur around the same time. The central questions are how high the peak is and how long is the peak duration. The answers to these two questions can have a significant impact on how much capacity needs to be provisioned in order to handle the workload while satisfying performance constraints. We aim to investigate resource provisioning requirements that satisfy demand constraints.

5. Introduce resources that can fail. Recovery strategies in the event of resource failure will be investigated. These include regular check pointing, and

restarting. Market mechanisms and their effect on re-allocation of resources, and resource prices will be studied.

6. The previous point raises the question of compensation costs. In the event of resource failure, where the demand was not satisfied and the QoS constraints were violated the users will expect some form of compensation. The determination of these costs and payments mechanisms will be formulated and tested.

4 Reference Implementation

In order to prepare the next steps (sprints) of the implementation of the GridEcon system, we have made a plan describing the reference implementation and also the first sprint. Following is a detailed view of this plan.

The following diagram showcases the main architecture that we are going to implement. This architecture retains the original 4 levels as described in D3.1 & D3.2 (Resources, Basic Resource Services, Composite Resource Services and Users and Applications). However, this diagram we believe gives more concrete information regarding how these services can be implemented.

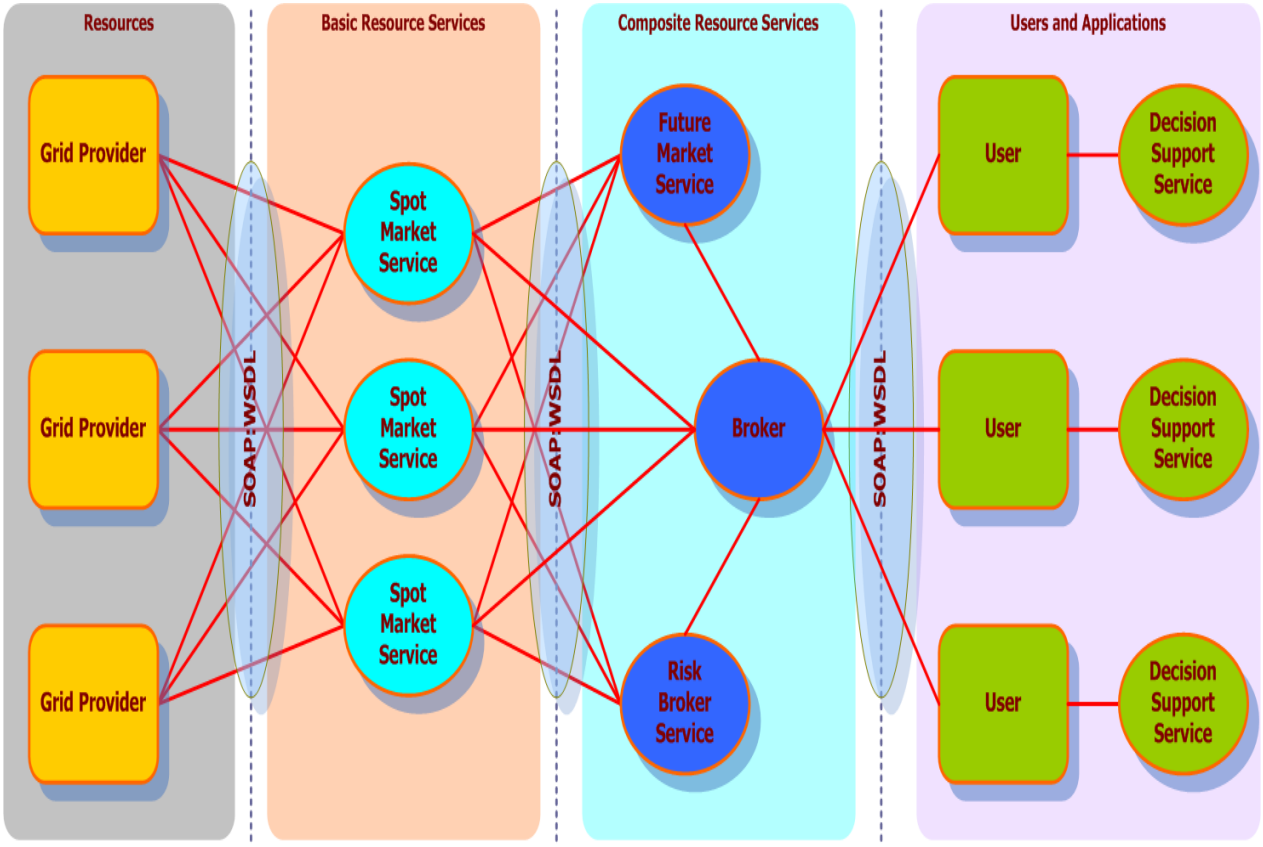

Figure 12 Reference Implementation

In more detail, the following components are illustrated above:

- **Grid Provider**
  Provider of resources that makes them available through standard interfaces
- **Spot Market Service**

The basic market mechanism for auctions in small intervals of time (i.e. each 5 min). Different spot markets will offer different resources (spot market for CPUs, spot market for VMs, spot market for different classes of resources etc.)

- **Future Market Service**
  Based on the spot markets different business entities may offer the opportunity to buy resources for future use. This service provides the environment for allowing such purchases.
- **Risk Broker Service**
  This service can be used for purchasing resources at fixed price, incooporating the risk since the spot market prices may flunctuate.
- **Broker**
  This component will assist (brokerage) the user to contact all the available services in order to buy resources (either in the spot market or in future market etc.).
- **User**
  The user interface for submitting/executing jobs. The user will make bids (or enter preference on prices) and also his requirements/preferences along with application requirements
- **Decision Support Service**
  This will assist the user in making the right bids. This could be advice on how many resources he should buy, at what prices, how to bid etc.

We foresee that in future version of the implementation more services/components will be added (such as software providers etc.). The architecture above will allow for easy integration of all entities. We will be using WSDL and SOAP for the communication.

For the first implementation we will only create a subset of the above components/services. Our scenario will demonstrate a spot market, i.e. the case of multiple users (N) biding for multiple units of two different classes (CA, CB). Thus each user i will bid for CA_i units of CA and CB_i units of CB. For each user i, we will simulate the number of resources he will be requesting with the exception of one user that will showcase the user interface. In each interval t, where t is a sufficient time that we will set (i.e. between 1 and 5 minutes), the users (through the broker) will bid for the resources. The allocation (winners and quantities for each type) will be determined by the algorithm for bidding/auctioning.

The algorithm below covers the case of multi-unit demand with two guaranteed types of units of trade (UoT) (e.g. virtual server). The objective of the auction algorithm is to allocate UoT as efficiently as possible, i.e. to those bidders that value them the most. Each bidder can be granted UoT of both types. Bidders that do not gain any UoT of the guaranteed types are served by best-effort UoT, which are given at a nominal (or zero) price. The auction algorithm is outlined as follows:

1. Bidders place their bids for the UoT of the *CA* type. Each bidder may bid for multiple nodes, possibly offering a different price for each of them.

2. A VCG auction is held to determine the winners of the *CA* UoT together with the payment per winner, which equals the social opportunity cost of its participation. In particular, if CA_k UoT are auctioned, then the top CA_k bids win. Moreover, a bidder winning *k* UoT pays the sum of the top *k* loosing bids.

3. The outcome of the above auction is announced, and then the bidders decide on and place their bids for the *CB* UoT.

4. A VCG auction is held to determine the winners of the *CB* UoT together with the payment per winner, which again equals the social opportunity cost of its participation.

For the auction of *CA* UoT, the bids should always equal the user's actual valuation for each of the units demanded. This should be input by the user. For the auction of *CB* UoT, the bids should be determined on the basis of the number of *CA* UoT won and the corresponding payments. Thus, the user should give the number of *CB* UoT targeted as a function of the number of *CA* UoT won; e.g. *CA*+(*1/2*)**CB =10*, meaning that the user ideally needs *10 CA* UoT and each one of them can be substituted by *2 CB* ones. The user should also give the total budget, which combined with the payments for the *CA* UoT won gives the total amount that can be bided for *CB* UoT. This can then be split evenly among the *CB* UoT, or in accordance to any other function selected by the user. The decision support service will implement all this logic and help the user through the auction.

Prior to the bidding, in *t-1* time the grid provider will advertise the amount of resources (units for each class) he has available. In order to assist us in the first implementation and make things simple, we will consider a single unit (i.e. *1* unit of *CA* is a *2GHz* machine and *1* unit of *CB* is a *1GHz* machine, or if we consider Virtual Machines, then *1* unit of *CA* is a *2GHz* machine guaranteed to run only *1* VM and *1* unit of *CB* is a *2GHz* machine guaranteed to run maximum of *2* VMs).

Once the bidding has concluded the winners (and losers) are informed (regarding how many resources of each type and at what price). Once this has been completed they can then contact the Grid Provider (which has already been informed) and use their resources. The following figure shows the components that are going to be developed and used in the first implementation.

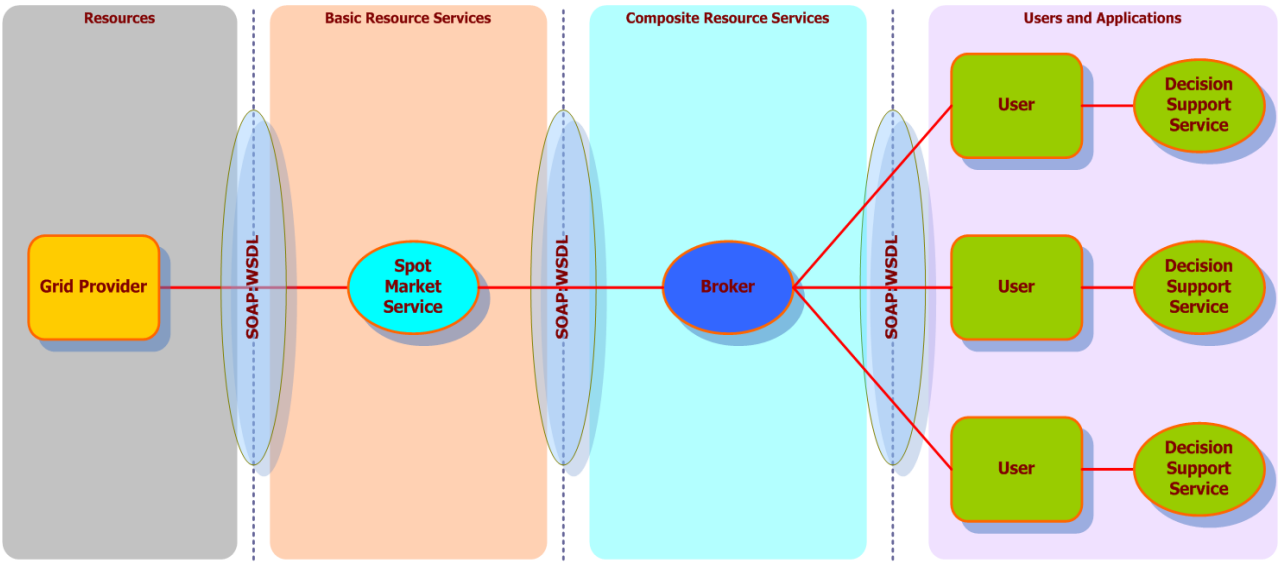

Figure 13 First "Sprint" for Implementation

The above scenario is better described in the following sequence diagram.

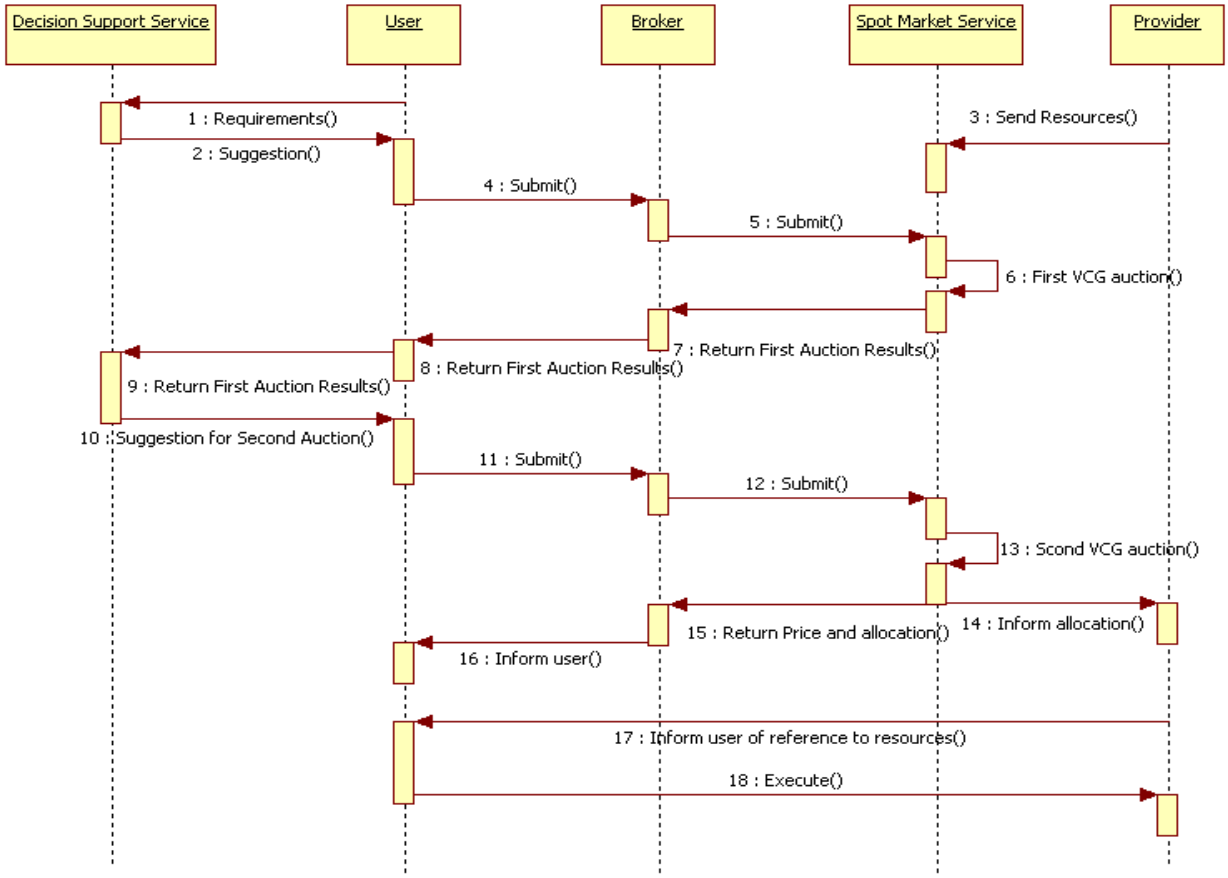

Figure 14 Use Case of first "Sprint"

5 Conclusions

During the course of this deliverable, we have looked into different implementation aspects and provided a set of validation scenarios that will allow us to validate and assess at the end of the project the quality of our work.

While this deliverable provides a high-level view of the validation scenarios and the implementation plan, in the upcoming D4.2 deliverable, more precise (technical) specifications will be provided. During this 3-month period between the two deliverables a set of sprints (according to the Scrum development methodology that we will follow) will allow us to build the basic infrastructure and services of GridEcon and gradually develop all the components specified in WP3 and implement the logic, algorithms and mechanisms provided in WP2. In the next deliverable we will provide descriptions of these next "sprints" (as we have in this deliverable for the first one).

6 References

[1] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, “Web Services Description Language (WSDL) 1.1”, W3C, 15 March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[2] Sun Grid engine, http://www.sun.com/software/gridware/

[3] Amazon Elastic Compute Cloud, http://www.amazon.com/gp/browse.html?node=201590011

[4] GridSAM - Grid Job Submission and Monitoring Web Service, http://gridsam.sourceforge.net/2.0.1/index.html

[5] Project Rio, https://rio.dev.java.net/

[6] http://en.wikipedia.org/wiki/Scrum_(development)

[7] K. Schwaber, “SCRUM Development Process”, August 2006, http://jeffsutherland.com/oopsla/schwapub.pdf

[8] J. Cohen, “Statistical power analysis for the behavioral sciences”, Hillsdale, NJ: Lawrence Erlbaum Associates, 1988