European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies
A Network of Excellence funded by the European Commission

A Comparison of SLA Use in Six of the European Commissions FP6 Projects

Michael Parkin, Rosa M. Badia
{michael.parkin,rosa.m.badia}@bsc.es
Universitat Politècnica de Catalunya
Nexus II, Jordi Girona 29
08034 Barcelona, Spain

Josep Martrat
josep.martrat@bsc.es
Atos Origin Research & Innovation
Avinguda Diagonal 210–218
08018 Barcelona, Spain

CoreGRID Technical Report
Number TR-0129
April 14, 2008

Institute on Resource Management and Scheduling

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

A Comparison of SLA Use in Six of the European Commissions FP6 Projects

Michael Parkin, Rosa M. Badia
{michael.parkin,rosa.m.badia}@bsc.es
Universitat Politècnica de Catalunya
Nexus II, Jordi Girona 29
08034 Barcelona, Spain

Josep Martrat
josep.martrat@bsc.es
Atos Origin Research & Innovation
Avinguda Diagonal 210–218
08018 Barcelona, Spain

CoreGRID TR-0129
April 14, 2008

Abstract

This technical report compares the approaches, techniques and technologies used to implement support for Service Level Agreements (SLAs) in six of the European Commission's Framework Programme 6 (FP6) funded projects. SLAs can be used to guarantee Qualities of Service (QoS) in Grid computing and provide a mechanism for the service provider to become sustainable through charging a customer for meeting the QoS they agreed. Thus, comparing the current approaches in these projects, assessing how they differ and where they are similar is necessary to understand possible gaps and potential problems with the solutions proposed by each project we assess. As the results of our comparison show, there is a great deal of common ground between the work of each project, but also some substantial variations.

1 Introduction & Motivation

This CoreGRID technical report compares the approaches, techniques and technology used to provide Service Level Agreements (SLAs) in six of the European Commission's (EC's) Framework Programme 6 (FP6) funded projects. An SLA is defined as "a contract between a provider and a user that specifies the level of service that is expected during the term of the contract" [1] and the distributed computing community's interest in them comes from the need to guarantee qualities of service (QoS) for distributed services and composite applications built from those services. These QoS guarantees may be on any aspect of the service being provided, e.g. the level of availability, amount of throughput, persistence of data or concurrent users supported. The desired goal is for the users of those Grids and applications supporting SLAs to have confidence that the guarantees the SLA provides are assured and enforceable and for the provider of those services to plan ahead and possibly charge a fee for meeting those guarantees.

The FP6 projects reviewed in this report with respect to their use of SLAs are Akogrimo, AssessGRID, BEinGRID, BREIN, NextGRID and TrustCOM. These projects have been chosen from the FP6 projects that provide solutions to agree, monitor and evaluate SLAs as ATOS Origin Research and Barcelona Supercomputing Centre are working in or closely involved with these projects and are therefore most relevant to us.

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

This evaluation was carried out because each of the projects listed above is focussed on particular problem domain and producing solutions to meet specific requirements. As a result, there may be differences in approach and functions provided in the provisioning of SLAs by each project. Thus, the motivation for this comparison is to take an overview of their work to asses where there is common ground between them, where there are differences, possible gaps in the solutions provided and potential problems with the solutions proposed.

1.1 Importance of SLAs

Within the Grid and distributed computing field, there is momentum behind the adoption of SLAs for many reasons including, but not limited to:

1. Grid Computing can be more than 'Best Efforts'.

   Previously, most distributed computing provision has been on a 'best efforts' basis, i.e. a service provider will attempt to provide a consistent, available service but makes little or no guarantees on that provision. An example of this type of provision are high-throughput Grids which may consist of several distributed sites providing a minimum hardware capability. Often, such as in a deployment of the Condor workload management system, the Grid user is able to specify the minimum level of hardware required to run their job. These requirements are matched by the Grids resource management system (RMS) to available resources or queued until a resource with the specified requirement becomes free. However, within this type of service provision users may need extra guarantees apart from those regarding the capabilities of the resources the Grid offers. For example, users may require urgent tasks be completed by a certain point in time, or be provided with a guarantee that a certain class of resources are pre-allocated (which can be the case when computational tasks are run interactively).

   At the moment, most Grid deployments agree and enforce these guarantees manually, often using telephone, email or web forms to contact system administrators who then perform the steps necessary to reserve Grid resources. Two examples of this are the UK National Grid Service (NGS) and the US TeraGrid. The NGS provides a web form for its users to complete which is then forwarded to system administrators[1]. On the US TeraGrid advance reservation is performed through a manual, telephoned or emailed agreement to system administrators who manually create a dedicated job submission queue that is used during the period the resources have been reserved.

   Obviously, manual solutions such as these are neither efficient nor do they allow for dynamic resource provision. Infrastructure like those proposed in in the FP6 projects reviewed in this report can help meet the requirements of customers and providers through allowing the customer to express their requirements in a regular format in order that a provider can automate the provisioning of resources. The use of SLAs can allow the provider to reduce costs and allow them to make more efficient use of their resources by being able to plan ahead and, if necessary, commission or decommission new resources when there are spikes or troughs in demand.

2. Move to Service-Oriented Computing.

   Recently, a move towards service-oriented computing within a service-oriented architecture (SOA) has allowed individual units of functionality are encapsulated within a well-defined service boundary. The vision of SOA architects is to allow these loosely-coupled, shared services to be assembled into new applications that provide a value greater than the sum of its parts. However, when composing these new applications, particularly across administrative domains where there is no previous experience of the service or of the provider, how can the 'best' service be chosen? How can the end-user be assured that the service will do what it advertises? This requires information about the services qualities (in order for the customer to make their decision) and then an enforceable agreement of the QoS agreed between the customer and the provider, which may be on aspects of the system not just associated with its performance. The advertising and guarantee of those qualities through the formation of an enforceable contract in the form of an SLA allows the end user to discover the quality and standard of service provision and then have confidence in the delivery of that service.

3. Advent of Business-Oriented Grids.

   Business-Oriented Grids (BOGs) such as those proposed by projects such as Akogrimo and BREIN (introduced in Sections 4 and 7 respectively) require SLAs for many reasons. For example, SLAs agreed ahead of resource

[1] Although it should be noted that the NGS core sites offer the automated advance reservation of resources through deployments of the HARC software [2].

provision allow the resource provider to plan its resource allocation and make the most efficient use of those resources. This can be achieved by the provider migrating less important computational tasks to cheaper or less powerful resources, for example. This in turn can help the service provider minimise its costs thereby increasing their profits. Other examples of the benefits in using SLAs in BOGs are that by providing a guaranteed service, a service provider then can introduce different charges depending on the different levels of service guaranteed.

However, a key component to supporting this type of automated, dynamic service provision is the ability to negotiate, monitor and conclude a contract for service usage in the form of an SLA. This is required in order that the obligations and expectations of the service customer and provider are clearly set out and can be monitored and assessed during the contracts lifetime and/or after the contract has completed. Contracts are one of the foundations of commerce as they form a legally binding exchange of promises between parties that can be enforced using legal recourse. Therefore, it is consonant to base business-oriented Grids on the same concepts. This allows new business models to leverage the existing knowledge and infrastructure that has gone into contract agreement and enforcement without requiring them to re-invent existing processes.

Thus, the ability to form and manage and assess SLAs are crucial to many aspects of future Grid development and provide a link to forming new business models for distributed computing and, ultimately, allowing Grids to charge for service and resource usage and become self-sustaining through those charges.

1.2 The Link with CoreGRID

CoreGRID is the EC's FP6 Network of Excellence for foundations, software infrastructures and applications for large scale distributed, GRID and peer-to-peer technologies. Within CoreGRID, the Resource Management and Scheduling (RMS) institute has an objective, amongst others, of providing technology to support future business models for Grid computing – critical if Grids are to be sustainable in the long term. As described above, fundamental to supporting this objective is the formation and management of SLAs for Grid resource usage in order that business-level relationships can be formed. This work contributes directly to RMS institute Task 6.8 (Service Level Agreements) and deliverables D.RMS.07, a review of SLA use for resource management and scheduling (M30), and D.RMS.09, a proposal of SLA negotiation approach (M38).

1.3 Structure of this Report

The previous section has introduced the motivation for this report and the importance of SLAs in distributed computing. The remainder of this report is structured as follows: Section 2 introduces SLAs with Section 2.2 describing the life-cycle of an SLA and Section 2.3 explaining how the ability to provide SLAs by a Grid depends on the availability of other Grid subsystems; Section 3 describes the methodology used to evaluate each of the FP6 projects assessed in this report; Sections 4 to 7 details the SLA infrastructure and related subsystems of each of the projects evaluated; Section 10 summarises the evaluation; Section 11 discusses open issues and presents our conclusions.

2 SLAs in More Detail

This section describes in more detail how an SLA is structured, the phases of its life-cycle and how the provisioning of SLAs depends on several other Grid subsystems being available.

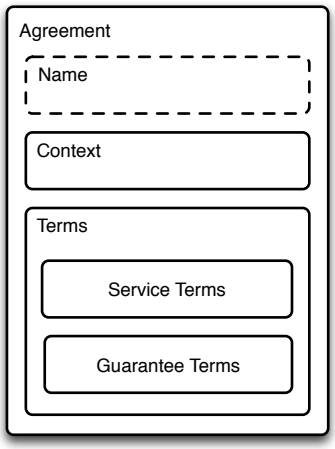

Figure 1: The General Structure of an SLA

2.1 SLA Structure

In its most basic description an SLA in Grid computing is a document containing agreed terms for service provision. The contents of this document are usually agreed between a service provider (the entity providing computational resources) and its customer (the consumer of those services), though there are situations where an SLA is agreed between the customer and a third-party broker (an entity arranging a service for the customer and not providing any service itself) and/or between a broker and the provider.

Figure 1 shows the generally-accepted structure of an SLA taken from the WS-Agreement specification [4, Sec. 4]. The optional *name* section may be used to provide a human-readable name to an agreement, whilst the mandatory the *context* section contains information about the agreement "such as who the involved parties are, what the services is that is being agree to [sic], and the duration of the agreement" [4, Sec. 4].

The terms for service provision within the SLA are known as *QoS Goals* [3] or *Guarantee Terms* [4, Sec. 4.2.6] and an individual term may be relevant to either the customer or the service provider. There are, therefore, *Provider Terms* and *Customer Terms* (though in current technology and SLA representations, this distinction is not made). For example, payment terms will be relevant to the customer, whilst compensation or penalty terms (for the failure to meet the QoS goals) are relevant only to the provider. Each term can be given an *importance* [3] or *business value* [4, Sec. 4.2.6.4] that allows trade-offs of QoS Goals in the agreement.

2.2 SLA Life-cycle

A well-used model to describe the phases of an SLA's life-cycle is that described by the Telemanagement Forum [5] and shown in Figure 2. However, when looked at in detail it appears that this model mixes the life-cycle of the SLA with the life-cycle of the service it applies to. For example, the stages of 'service development' and 'decommission' apply to the service, not the SLA. Secondly, in the opinion of this report, there are some important aspects of an SLAs life-cycle that are missing from the model, such as the points where settlement of the SLA takes place.

A better model for the life-cycle of an SLA could be that shown in Figure 3, where each of the phases apply to the SLA, not the service. Taking each of the stages in turn; the first stages of an SLAs life-cycle are when an general SLA template is formed then advertised by the provider. An individual SLA is negotiated then agreed on the basis of this template. The negotiation phase is separate to the agreement phase because a negotiation may not always lead to an agreement. Then, because the service described in the SLA may start some time after the SLA was agreed (for example in the case of advance reservation) the agreement and execution phases are also separated.

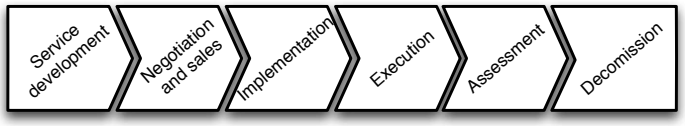

Figure 2: Telemanagement Forum SLA Life-cycle (From [5])

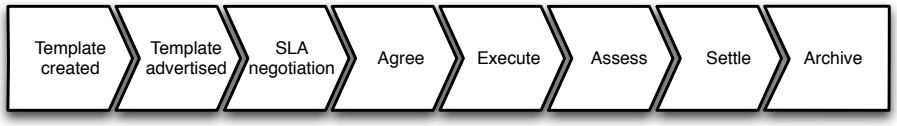

Figure 3: Suggested SLA Life-cycle

The execution phase is where resources are commissioned and deployed to complete the agreed service. Whilst the service is executing (or active) the SLA is in the assessment stage where, periodically, the performance of the service is reviewed against what was agreed in the SLA. Finally, when the service has completed, the final settlement phase takes place where it is determined if the SLA was met and what the costs to the customer are and what penalties may apply to the provider for breaching the terms of the SLA. According to each of the parties policies after this stage is completed the SLA may (or may not) be archived though, as explained in [6], there is usually a statutory period (known as the 'limitation period') where the SLA record must be kept because it is a legal document describing how services were provided. The archive phase also provides a mechanism for providing an audit trail for later reference.

2.3 Relationship to Other Grid Subsystems

Having described an SLA and its lifecycle, a service provider can only provide SLAs to its customers if there are supporting components and subsystems deployed and available within its infrastructure. This section introduces those subsystems and describes the relationships between them as they form the basis for some of the evaluation criteria introduced in Section 3 of this report. Thus, the subsystems and their relationships are shown in Figure 4.

In the second stage of an SLAs life-cycle (shown in Figure 3) there must be a mechanism by which the provider can advertise its capabilities to potential customers. This requires advertising subsystem that must also interact with the scheduling subsystem (relationship 1) in order find what resources are available and which can be publicised. The advertising subsystem should also link into accounting subsystem (relationship 2) so that targeted notices about resource provision can be made to customers who may have used a certain set of services in the past.

The next phases of an SLAs life-cycle are when it is negotiated and agreed. Therefore, a mechanism to negotiate and agree the future availability of computational resources must be available through a negotiation and agreement subsystem. This capability therefore requires interaction with the scheduling subsystem (relationship 3), to determine what resources the provider has and when they are available, the accounting and auditing subsystem (relationship 4) and a settlement subsystem (relationship 5, described below) to ensure that the customer negotiating for resources has not used their allocated quotas, is not a bad customer (i.e. is not in significant debt to the service provider) or untrustworthy (e.g. has previously carried out unsafe actions).

Once an SLA is in the 'active' phase of its life-cycle, the monitoring subsystem provides information about what is happening (or what has happened) with the service providers resources. This may be a system that aggregates information from low-level resources (such as individual computers or network routers) or a set of smaller services that each produce information. The monitoring subsystem forwards information to the accounting and auditing subsystem (relationship 6) which gives the provider a method of being able to assess how the SLA is performing in real-time. It may also be the case that the monitoring system interacts with the accounting system to determine if the user has exhausted their usage quotas/limits, which in turn may lead to the job and their account being suspended.

Once the job has completed and the SLA is in the settlement phase, the accounting and auditing subsystem can be used by the settlement subsystem (relationship 7) to retrospectively determine what actions happened when, who caused those actions to happen, to evaluate if an SLA was met (i.e. if what was agreed to was supplied) and the total charge to the customer. The settlement subsystem also ensures that accounts are settled and resources are paid for.

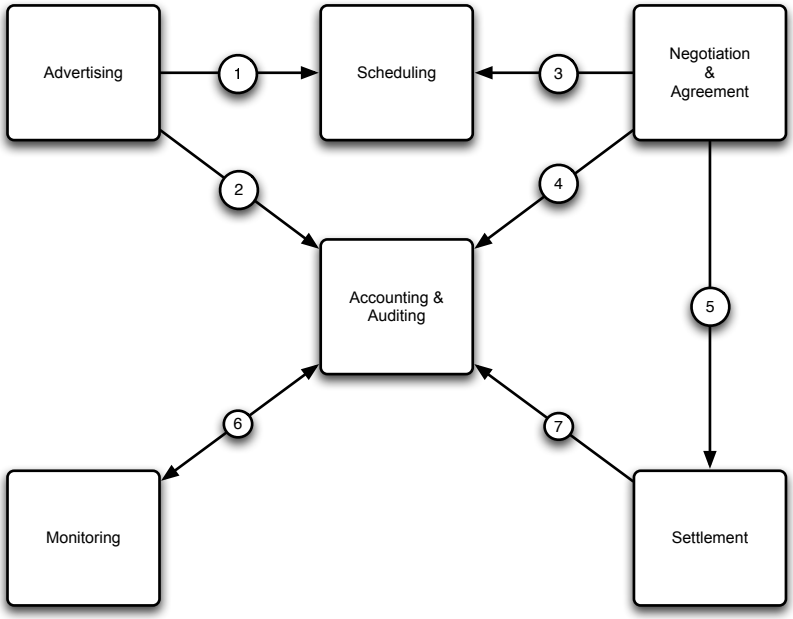

Figure 4: Relationship between Subsystems Required for the Provisioning of SLAs

In summary, we feel that these 6 subsystems form the minimum infrastructure to support the provision of SLAs.

3 Review Methodology

The methodology this report uses is that of a comparative evaluation to contrast the approaches to SLAs of the FP6 projects being evaluated. The information used for the evaluation about each project comes from papers published in conference proceedings, personal correspondence or documents published directly by each project, such as project deliverables, whitepapers and factsheets. Where possible we have provided a detailed citation to where each piece of information can be found within each document (as some of the documents are over 100 pages long) and a cross-references to confirm information.

This report uses four criteria for evaluating the information about each project; these are taken from and based upon those identified by the Gridipedia web-site as necessary for SLA provisioning[2]. Gridipedia actually lists eight catagories but this report does not use the 'provider resource optimisation' (i.e. how resource providers allocate, re-allocate and make the most efficient use of their resources) criteria in the evaluation as how this is achieved is considered to be an internal service provider decision. This report also considers the 'negotiation' and 're-negotiation' requirements and the 'monitoring', 'evaluation' and 'accounting' requirements to be too closely related to be evaluated differently and these five aspects have been combined into two evaluation criteria. Thus, the four areas of comparison are:

1. SLA Publication & Discovery of SLAs.
2. Protocols for SLA Formation.
3. SLA Representation.
4. SLA Monitoring, Evaluation & Accounting.

Each of these criteria is now discussed in more detail.

3.1 SLA Publication & Discovery of SLAs

Within a distributed environment, customers need to find what services are being offered by service providers and what their qualities of their services are. There must exist, therefore, a common mechanism for the service provider to publish the capabilities of their services and resources. As in the real world, these advertisements may be published to all possible users or targeted to a certain set of customers, such as those who are prompt at settling accounts or those that have used specialised resources in the past. This evaluation criteria will document and assess these mechanisms.

3.2 Protocols for SLA Formation

How SLAs are negotiated between the parties involved in the provision of goods and services are of interest because, as described in [3] and [7], SLAs are part of a formal contract for service provision. However, this formal contract is only as legally enforceable as the protocol used to form that contract.

The protocols used to form the SLAs in the projects reviewed will be assessed with respect to how they take into account these legal considerations of contract formation (as discussed in [7]), how the protocols promote interoperability between customers and service providers, how they protect the service provider from denial-of-service (DoS) attacks (how this occurs in agreement protocols is discussed in [7]) and whether true 'negotiation' (defined as a multi-round discussion aimed at reaching an agreement) can be achieved with the protocol.

As mentioned in the introduction to this section this report considers the process of negotiation and re-negotiation to be the same as re-negotiation is a negotiation within the context of an existing agreement. Thus, the advertising and agreement processes used in the negotiation process can be used to re-form contracts and the capability to re-negotiate a contract is also evaluated under this criteria.

3.3 SLA Representation

Like the protocol used to form SLAs, the choice of SLA representation also effects the interoperability between a service provider and its potential customers. For example, if the service provider chooses a proprietary format to

[2] These can be found at `http://www.gridipedia.eu/index.php?id=121`.

represent their SLAs, it must expect the customer to understand the content and its meaning before the customer can commit to a contract based on that content. Likewise, if the customer insists on using its own format the service provider will not be able to interpret it. Either situation leads to the agreement process breaking down before it has begun because neither party can understand the other.

Thus, this evaluation criteria is concerned with how SLAs are represented and, more specifically, how the QoS terms, guarantees, payment and penalty terms are represented within the SLA. One of the factors that will be looked at when evaluating the SLA representation of each project is how each of them uses a common language to represent those terms as, again, commonly agreed terms will directly effect the interoperability of the solutions reviewed.

3.4 SLA Monitoring, Evaluation & Accounting

Once an agreement is formed between a service provider and a customer the resources provided must be monitored and their capabilities and/or performance constantly evaluated to ensure that they are meeting the QoS guarantees agreed in the SLA. As described in Section 2.3, the ability to account for and audit resource use is closely linked to the monitoring function — indeed, these could be said to be the same as accounting and evaluation must take place together to determine what is currently happening in the system as a whole.

As a result, this evaluation criteria will evaluate how each project monitors the QoS parameters agreed with the customer, how and when these are evaluated and how the final accounting for resource usage takes place.

4 Akogrimo

Akogrimo ('Access to Knowledge through the Grid in a mobile World', IST-2002-004293) is an FP6 project that ran from July 2004 until September 2007 with the aim of defining a 'mobile Grid architecture'. This was then realised through the development of several prototype implementations. The Akogrimo project envisages a world where pervasive Grid services meet the needs of fixed, nomadic and mobile citizens in the 'anywhere at any time in any context' paradigm[3]. These services, including services providing personalised knowledge and common semantics, allow for the ad-hoc and dynamic formation of problem solving environments for business, science and everyday life. As a result, network and service operators should be able to develop new business activities based on Grid and mobile communications concepts by providing services in such an integrated world.

Within the Akogrimo model services are provided by businesses to their customers. Thus, for the reasons given in Section 1, QoS need to be assured in order that the business relationship can be maintained by setting expectations on the delivery of those services. Akogrimo use SLAs both to define the service provider's external relationships with their customers and also internally within the service provider to plan for anticipated resource usage.

Two examples of SLA use within Akogrimo are shown in their eLearning and eHealth scenarios (described in [8] and [9]). In the eLearning scenario, students are equipped with network-enabled PDAs to use in a field trip. During their trip they need to collaborate to produce a 3D rendering of the site they are visiting. This requires an agreement with a 3D rendering service to make their images. Thus, QoS such as the minimum amount of concurrent users and availability of the service need to be guaranteed for the students in an SLA so that they can complete their work.

The Akogrimo eHealth scenario is similar to the eLearning scenario in that mobile devices, this time heart monitoring devices, require interaction with another service, a Heart Monitoring and Emergency Service (HMES). The HMES requests the heart monitoring data, which in turn is analysed by a Electrocardiogram (ECG) analysis service. The ECG analysis service checks the data according to appropriate criteria set by a doctor. When a problem arises, an emergency handling service determines the location of the patient, identifies the responsible emergency dispatch centre and a locally available emergency ambulance. As this scenario involves several independent parties interacting, they need to be bound by contracts that govern aspects of their quality of service such as the HMES service availability, ambulance response time, and access to the patients data.

4.1 SLA Subsystem Design

The Akogrimo SLA subsystem design is described in [10], [11], and [12]. Figure 5 on the next page shows the Akogrimo SLA service provider components and their interactions. These components are:

- The *execution management services* (EMS) are a set of co-ordinating services that plan and execute the services the provider has agreed to supply to the customer. They carries out several functions, including the advance reservation of the most appropriate services and/or resources for completing the SLA and the initialisation and termination of the monitoring services for those services and resources. These steps are prefixed with a '1' in Figure 5.

- The *metering service* is created by the EMS to measure the amount of low-level resources (such as CPU, memory and disk utilisation) consumed during the execution of a service. Upon receiving information it aggregates and reformats that data for use by other services. When a change in any monitored parameter takes place it notifies the monitoring service.

- The *monitoring service* is a service that receives data from the metering service(s) and network monitoring services being metered as part of fulfilling the SLA. When it receives a notification that a change in performance has occurred (step 3 in Figure 5), it aggregates all the necessary information and notifies the SLA Controller.

- The *SLA enforcement services* contains the *SLA controller* and the *SLA decisor service*. The former is a transient service created by the EMS when the SLA is agreed [13, p.61]. During the lifetime of the controller it receives notifications of QoS parameter changes from the monitoring service (step 4 in Figure 5). Upon receiving these notifications the SLA controller service calculates if the QoS measurements are inside or outside the thresholds set in the SLA. If the measurements are inside the allowed thresholds, then no action is taken. If the measurements are outside the thresholds the persistent SLA decisor service is notified (shown as step 5) of a breach of

[3] From `http://www.mobilegrids.org`.

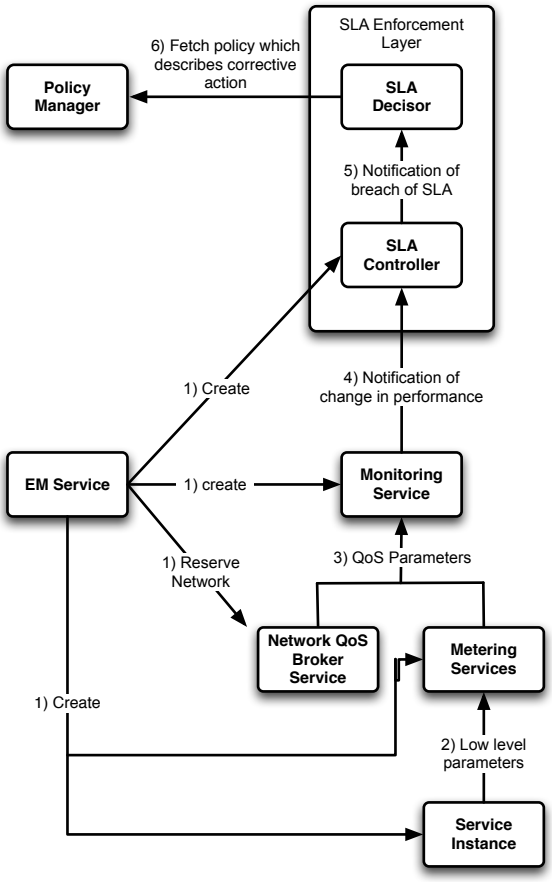

Figure 5: The Components in the Akogrimo SLA Subsystem

the SLA. Depending on the policy and configuration of the SLA decisor service the customer's service may be given more resources to complete its task, moved to another resource or, if the SLA breach is serious and cannot be recovered, the service may be terminated.

4.2 SLA Agreement Process

The process by which SLAs are agreed in Akogrimo is described in [11] and [13]. Negotiation between customers and service providers is carried out by the EMS introduced above. As described in [13, p. 19], the EMS is a set of services each of which provides a specific function in the advance reservation, execution and management of client tasks and other Akogrimo services. One of the functions they provide is that of negotiating SLAs with the customer, which is done through the *negotiation service*.

The internals of the negotiation service [13, p. 19] are not considered in detail here, but it interfaces with a *discovery service* to provide the customer with information about free resources and a *reservation service* to secure resources for a customer. To summarise [13, Sec. 2.1.1], the advertising, agreement, reservation and execution process follows the following stages:

1. The service provider advertises its business services.
2. The customer discovers services and network availability (the 'negotiation phase') which match its requirements for the task to be carried out.
3. The business service and network are co-allocated in the 'reservation phase'.
4. The business service is executed and monitored.

Each of these stages is now discussed.

4.2.1 Advertising of Business Services

Before business services are published and made available to end users, the service provider must become a member of an Akogrimo Base Virtual Organisation (BVO), which is "an enabler VO that offers services useful to create new VOs (OpVOs)" [14, Sec. 2.1.1] or a "enterprise network" of services [15, Sec. 5.1]. After the service provider registers with the BVO, which is "open" [14, Sec. 2.1.1] in order that anyone can join, they can then publish their business services to the Akogrimo GrSDS (Grid Services Discovery System), a service that exists in the BVO to "a bridge between the Service Requests and the actual Service Discovery Mechanisms" [16, Sec. 5.1] of the underlying Grid and network resources. According to [17, Sec. 4.1.3], the service provider adds the following information about their service into the GrSDS:

- Information about the service provider.
- "Functionalities of the service".
- The services interface, i.e. the services WSDL document. Publishing "a semantic description of method logic is [also] recommended".
- What QoS conditions the service supports, i.e. the SLA template.
- The address of a service that will perform a negotiation for the advertised service.

With this in place, the potential customer of a service then form an Operative Virtual Organisation (OpVO), "the environment that will allow executing and invoking [an] application" [18, Sec. 2.2.2]. This is achieved through the execution of a single process that carries out the initial OpVO setup, then searches and negotiates for services, deploys the 'workflow' (the user's job) to the services and performs final configuration tasks. This procedure is described in the following section.

4.2.2 Discovery, Negotiation & Co-allocation Phases

The process by which SLAs are agreed is based on the WS-Agreement specification [13, Sec. 2.4.2], therefore it is assumed that 'negotiation' is a standard WS-Agreement exchange, where there the provider can either accept or reject a proposal put to it by the consumer. The consequences of this are now noted.

First, there is no real negotiation occurring within the 'negotiation and discovery phase'. According to [13, Fig. 6], the interaction between the customer and the negotiation service (representing the service provider) is a standard WS-Agreement interaction where the customer submits its requirements and, if the requirements can be met, the negotiation service returns the EPR to a resource or *null* if they cannot[4]. Within the service provider no negotiation between the negotiation service and the discovery service or QoS broker takes place either as the negotiation service simply selects the first service that meets the customers requirements with the required network availability. Really, what is happening here is a discovery of a providers capability to perform a specified task.

Secondly, as indicated in [13, Figs. 6 & 7], there is a two-step approach to discovering and then co-allocating (i.e. reserving) the business service and network resources by the negotiation service. Performing co-allocation like this runs the risk of reserving only one or none of the two required resources because of the possibility that one or both of the two resources will be 'booked' by another reservation process. Therefore, what is also required is some method of cancelling the business service if the network cannot be allocated and vice-versa. From the Akogrimo deliverables reviewed no mention of this situation is noted nor is there a process for resolving this situation.

4.2.3 Service Execution & Monitoring Phase

The Akogrimo service execution and monitoring phase is described in [20, p. ?] and [13, p. 26]. The business service being monitored is created in the previous co-allocation phase [20, Fig. 11][13, Fig. 7] with implication that the business service could be created for some time before the service is actually required for use, potentially consuming the service provider's resources such as software licenses.

The business service is 'executed' in step 7 of [20, Fig. 12] (this is also shown in step 8 of [13, Fig. 8]) and passes its monitoring information to the metering service (as shown in Figure 5 on page 10 of this document). This service then passes information to the monitoring service[5].

One of the requirements of the metering service is that it should be scalable in order that it can deal with a large amount of services producing a large amount of data [10, Sec. 3.4.1.3]. However, the approach of creating a metering *and* monitoring service for each business service could lead to possible scalability problems because in order for it to be available the state of these services must be replicated and the consistency maintained across more than one network endpoint. This is an overhead that should be avoided and a better approach would be to use stateless (i.e. web-style) services which could then be replicated without having to replicate their state too.

Another criticism that could be made of the execution and monitoring phase of the Akogrimo architecture is that there are lot of interacting services involved (e.g. as shown in [13, Figs. 2 & 7] or [20, Fig. 17]). If a service fails or the network between two of the services is unavailable, both of which are likely and should be assumed, what happens? No mention of fault-tolerance is made in the Akogrimo documents reviewed.

Moving to the SLA enforcement aspect of the execution and monitoring phase, this is described in [10, Sec. 3.5]. As mentioned above, the SLA controller is responsible for notifying the SLA decisor of any failures to meet QoS thresholds agreed in the SLA. The SLA decisor, with the support of the *policy manager* [10, Sec. 3.7], then takes the appropriate action to rectify the problem(s).

Policies are "a mean [sic] of specifying and influencing management behaviour" [10, Sec. 3.7.1], i.e. they determine what action within the Akogrimo infrastructure should be taken when an SLA is violated. The policy manager allows the creation, storage and editing of policies for the Akogrimo VO [10, Sec. 3.7.2]. Stored policies are supplied as information to other components which may have to take a decision based on their contents. With respect to SLA enforcement, the policy manager will be asked by the SLA decisor for "the policy containing metrics mapping parameters to guarantee QoS of the service" [10, Sec. 3.7.1.5]. The SLA decisor then uses this information to determine what action it should take when an SLA is violated.

[4] The lack of a negotiation capability in WS-Agreement has been widely noted (e.g. [19]) and the motivation for a possible WS-AgreementNegotiation standard.

[5] Note that this is inconsistent with the description in [20, p. 43] where, during the execution of the business resource, "each time the state of execution changes a notification message is generated... and sent to the EMS". [20, Fig. 12] shows the business service sending status notifications directly to the EMS and not through the monitoring service.

4.3 SLA Representation

The WS-Agreement XML schema is used to represent SLAs in Akogrimo [17, Secs. 3.7 & 4.7]. But, in the Akogrimo deliverables reviewed, no specific example of an SLA is given. Neither are there and explicit details about the QoS goals or guarantee terms that may be contained in the Akogrimo SLAs. But, in the sections about SLA enforcement service implementation and the technologies involved [10, Sec. 3.5.2.4], [21, Sec. 3.4], there are several mentions to the Web Service Level Agreement (WSLA) language specification being used. However, there are no other details about this than, for example, "WSLA has been taken into account to define the contract template" [13, Sec. 2.4.2.1.].

However, [20, Sec. 3.6.2.2] describes some of the QoS parameters used internally by the monitoring service to determine if the SLA has been broken. These QoS parameters are "cpu usage, memory usage, disk usage and wall clock time (time elapsed while the business service was running)" and it is assumed that these parameters are used in the WS-Agreement templates used to form the SLA. One problem with these parameters is that it is unclear what they mean; for example, do the QoS parameters referring to "usage" mean maximum, average or total used?

5 AssessGrid

AssessGrid ('Advanced Risk Assessment & Management for Trustable Grids', IST-2005-031772) has the main objective of addressing "obstacles of a wide adoption of Grid technologies by bringing risk management and assessment to this field, enabling use of Grid computing in business and society"[6]. The project was started in April 2006 and is scheduled to be active until January 2009. Thus, a key goal of AssessGrid is to improve the attractiveness and commercial uptake of Grid services through providing transparency, risk-awareness and decision support for end-users in negotiating for use of Grid services. In doing so it is hoped that the customer is aware of the risk of entering into an SLA for service provision and the likelihood that the SLA will be met or broken.

The AssessGrid service provision model contains three domain-independent roles; the service provider, broker, and customer (i.e. the entity requesting and possibly entering into a contract for the service) [22, Sec. 2.2]. To demonstrate the possible interactions between these parties AssessGrid has developed three motivating "showcase scenarios" [23, Sec. 3.6]:

- Single-task scenario. In this scenario the customer can submit a single job to a service provider directly or through a broker. If submitting through a broker, the broker will return a ranked list of providers it knows of which can meet the requirements of the task submitted together with a risk assessment. The customer chooses a provider based on these criteria and forms the SLA directly with the provider.
- Workflow scenario with broker acting as mediator. This is where a customer has a workflow (i.e. a job composed of a set of tasks) they whish to execute and ask the broker to search for a provider capable of executing each of the tasks. The customer then creates an SLA with each provider it wants to execute each task. This is quite a complicated arrangement as the customer has to take an active role in managing the set of contracts for the workflow and take action if a provider fails or is not meeting one of the SLAs.
- Workflow scenario with runtime-responsible broker. Here, the customer agrees one SLA for the entire set of tasks in the workflow with the broker who chooses the service provider for each task. Thus, in this scenario, the broker is not only mediating between the customer and service provider but also providing a higher level of service by selecting the service provider to carry out each task and managing the relationship with it.

Thus, the AssessGrid motivating scenarios allow for most (if not all) patterns of interaction between a service provider, broker and customer. The remainder of this section describes how AssessGrid achieves these patterns of interaction.

5.1 SLA Subsystem Design

As stated above, there are three roles within the AssessGrid design: the service provider, the broker and the customer. The interactions between them are described in [22]: within the service provider the *SLA Negotiation Process* is responsible for all SLA interactions with a broker or a customer; the *SLA Contracting Process* is responsible for all interactions by the broker with a service provider or customer; and the customer uses the *AssessGrid Client* to connect to either the provider or broker. All SLA-based interactions between the three roles are performed using the WS-Agreement interaction patterns, with the restriction that the customer is always the initiating party in the WS-Agreement protocol [22, Sec. 3.2]. These components and interactions are show in Figure 6.

5.1.1 Service Provider Components

As well as the SLA negotiation process introduced above, Figure 7 shows other components present in the service provider. Each of these is now introduced.

- The negotiation service is responsible for interacting with the the party wishing to use the service providers resources (the contractor). If the service provider thinks it can meet the requirements of the contractor it supplies them with an SLA for their job including an value indicating the risk of non-completion of the job. How an SLA is agreed is describe in more detail in Section 5.3, below.
- The scheduler is responsible for finding available time on resources to run jobs so that SLAs can be made by the negotiation service. When an SLA is agreed, the scheduler also starts the jobs on the allocated resources. In the event of a Grid resource failing the scheduler is also responsible for re-starting the job on other resources.

[6] From `http://www.assessgrid.eu/`.

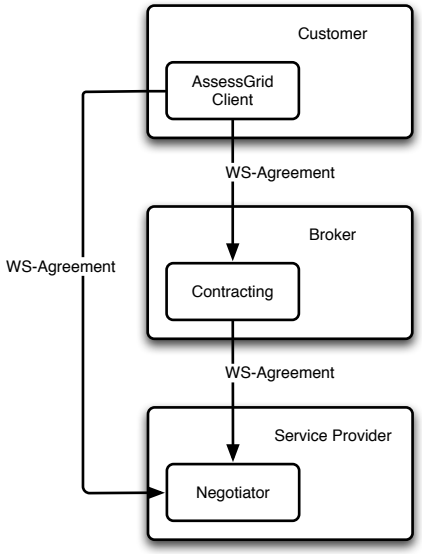

Figure 6: Interactions between AssessGrid Roles

- The risk assessment service is responsible for quantifying the risk of failure of jobs described by a specific SLA or the risk failure of infrastructure components. It also provides dynamic assessment of SLA violations during job executions.
- "The consultant service supports the risk assessor with statistics in order to estimate the failure risk of fulfilling an SLA" [24, Sec. 6.2.2]. These statistics are generated from the mining of historical information collected by the monitoring service.
- The monitoring service carries out the monitoring of resources against agreed SLA criteria. This component and its interactions is discussed in more detail in the following section.

What is confusing about the description of this architecture and its components in [23] and [24] is that the role definitions and the interactions between them are not consistent. For instance, [23, Fig. 4][7] says that the "negotiation manager uses the consultant service to find a risk assessment" and a direct relationship between the negotiation manager and the consultant service. This relationship is also shown in [23, Fig. 13]. However, in the same document it says that "the scheduler informs the negotiator about the reservation and the associated risk" [23, Sec. 6.2.1] and, in [23, Fig. 13], the risk assessor is shown providing the risk assessment to the scheduler. This is confusing as, in our opinion, the negotiation manager should receive the risk assessment from the risk assessor or scheduler, not both. Therefore, Figure 7 shows our interpretation of the AssessGrid architecture, which may or may not be correct because of ambiguities in the documents reviewed.

5.2 SLA Advertising

In AssessGrid there is no explicit SLA advertising function. To determine if a service provider is capable of carrying out a requested task SLA 'templates' for jobs are returned by the service provider during the agreement process (described below). If the provider cannot carry out the task being asked for then a fault is returned.

5.3 SLA Agreement Process

As described in above, there are three agreement scenarios considered in AssessGrid. However from the point-of-view of the service provider these are all similar as "it makes no difference if the SLA has been negotiated with an end-user

[7] Note that this figure, "the AssessGrid system architecture specification", does not contain the risk assessment service.

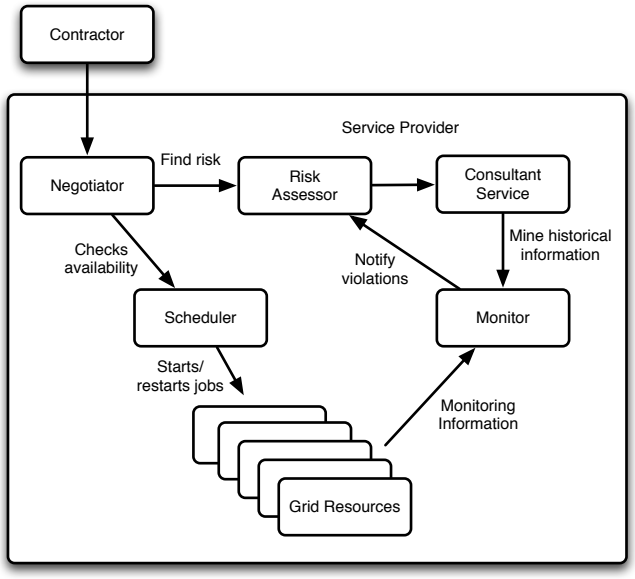

Figure 7: AssessGrid Service Provider Components

or a broker at Grid middleware" [23, Sec. 3.6.1.2]. Therefore, this section will concentrate on case where an agreement with the provider is made from the generic role of 'contractor', which may be either the customer or the broker.

AssessGrid have recognised that WS-Agreement is not capable of negotiating SLAs [24, Sec. 6.2.2] and have therefore extended and modified WS-Agreement to meet this requirement. The interactions between the contractor and service provider are now [24, Sec. 6]:

1. The Contractor retrieves a SLA template from the Negotiation Manager Agreement Factory.
2. The Contractor completes and returns the SLA template to the Agreement Factory. The Agreement Factory creates a WS-Resource representing a Negotiation Manager Agreement and the EPR of this resource is returned to the Contractor.
3. The Contractor queries the resource properties to determine if they meet their requirements.
4. If the properties of the WS-Resource are acceptable to the Contractor they ask that the agreement is created through calling the *commit* operation of the WS-Resource.

First, it is clear that there is no negotiation occurring in this sequence. Secondly, in step 2 the operation to create a WS-Resource representing the agreement has its semantics changed from those defined in the WS-Agreement specification. In the WS-Agreement standard step 2 creates a new agreement in the 'observed' state indicating that the agreement has been made. In AssessGrid the meaning of this operation has been changed to create an agreement in the 'pending' state and the agreement is only moved to the 'observed' state through the extra AssessGrid *commit* operation. Obviously, not adhering to the WS-Agreement standard means that clients built for the AssessGrid platform cannot be used with other implementations of WS-Agreement. Finally, because the 'contractor' plays the role of the offeree in the protocol, the service provider is open to denial-of-service attacks because it is always dependent on the contractor calling the *commit* operation to create the SLA, which it may never do.

5.4 SLA Monitoring & Accounting

In AssessGrid, the monitoring of a providers resources is carried out through a monitoring service which receives information from Nagios daemons running on the Grid resources [23, Sec. 6.2.4]. Nagios[8] is an open-source system and network monitoring application. When the monitoring information is received by the monitoring service, if a

[8]Homepage: `http://www.nagios.org/`.

parameter's value is higher than a defined threshold the risk assessor is alerted by the which calculates the cost of the SLA violation.

The monitoring service itself is based upon the AssessGrid-developed Unified Monitoring Framework (UMF) [25]. Part of the UMF, the Unified Management Server (UMS) distributes the distributes the configuration to the Nagios daemons running on the Grid resources. The UMS is supplied the parameters required to configure the Nagios daemons from the consultant service.

The AssessGrid monitoring system only provides 'passive' monitoring of SLA data; i.e. only violations of SLAs are recorded, and when the monitored parameters of an SLA are broken, the AssessGrid framework does nothing to limit the SLA breaches. The monitoring information is used only to calculate the SLA penalties applicable. Accounting is carried out therefore when the job is completed and the total cost and penalties can be assessed.

5.5 SLA Representation

The structure of the AssessGrid SLA is described in [23, Annex C]. This template is based on the standard WS-Agreement structure shown in Figure 1 of this document with a context and terms section. In AssessGrid service description terms are expressed using the JSDL HPC profile [26]. Example terms are given in [23, Fig. 55], and an example SLA sent to us by the AssessGrid project corroborates that this template is followed. Example SLA terms used by AssessGrid are operating system, CPU architecture, individual CPU count, physical memory, virtual memory, total CPU count and total resource count. As a result of the SLA terms being based on the JSDL standard [27], these terms are well-defined and understood.

6 BEinGRID

BEinGRID (Business Experiments in GRID, IST-2005-034702) is a three-and-a-half year project which started in June 2006 to provide a set of 'experiments' to demonstrate the business benefits of Grid technology. In providing these experiments, BEinGRID is illustrating that Grid technology can be exploited commercially and that this exploitation will help strengthen the EU's leadership in this area of distributed computing and in business competitiveness generally. Thus, a central aim of BEinGRID is to establish routes through which Grid technologies can be adopted by business and BEinGRID does this through involving many potential and existing end-users of Grid technology to establish exactly what the business requirements for Grids are and by developing a toolset repository for the end-users based on popular Grid middleware to meet those requirements.

Currently, there are 18 business experiments in BEinGRID, ranging from 3D visualisation to sales management systems to the production of advanced textiles, across 12 different business sectors (e.g. retail, architecture, finance, etc.). These experiments have common areas of interest which are grouped into technical and business cross-activities, such as trust and security, interoperability and organisation management. Some of the cross-activities are sub-divided into more specialist clusters. Within the service management cross-activity of BEinGRID the SLA cluster is of direct relevance to this work, and it is the work of this cluster which will be reviewed in this report.

BEinGRID's SLA cluster considers the typical life-cycle of an SLA as shown in Figure 2 of this report (page 5) and seeks to provide design patterns, common technical requirements and generic components to manage all stages of this life-cycle. At the time of writing, the SLA cluster has determined their requirements (which are similar to those given in Section 3 on page 7 of this report and described briefly below) and produced abstract design patterns for the provisioning of SLAs.

The business experiments in BEinGRID[9] that are particularly interested in SLAs are: Experiment 6, which is concerned with guaranteeing the availability of HPC resources for compute-intensive groundwater flow modelling; Experiment 8 requires an SLA solution focussed on the integration of engineering and business processes in metal forming; Experiment 10 which deals with end-to-end pharmaceutical supply chain management; and Experiment 16, which seeks to bring SLAs to the shipbuilding design process.

6.1 SLA Subsystem & Components

BEinGRID has taken a different approach to the other FP6 projects reviewed because, as mentioned above, they have provided in their deliverables abstract, platform and implementation-independent design patterns to provide solutions which can be implemented using a variety of technologies. This is a bold step by BEinGRID as it recognises that there is not and will not be a Web Services hegemony and that many other implementations other than ones based on Web Services are possible. In the case of the BEinGRID SLA cluster the following design patterns have been produced to meet the common technical requirements of the business experiments interested in SLA technology:

- Monitoring & Evaluation. These require that the resource provider has some kind of low-level monitoring in place on its resources and networks and that it implements the BEinGRID monitoring service containing the SLA runtime monitor. 'Reaction modules' are also required to take action when an SLA violation occurs.
- SLA Negotiation. To enable this capability the provider relies on having a SLA template store, a set of business rules to guide the negotiation and an interface to the local scheduler to determine resource availability.
- SLA Accounting/Charging. An SLA accounting module contains the usage and penalty charging calculators and accounting manager.

Note that persistent SLA storage is required by all three design patterns. Also note that a resource optimisation design pattern is also included by the SLA cluster but, for the reasons given in Section 3, this function is not evaluated in this report.

6.2 SLA Advertising

The mechanisms by which SLAs are publicised by providers and discovered by clients is described in [28, Sec. D.4.4]. Briefly, BEinGRID envisions a situation where service providers capabilities are advertised in a service marketplace.

[9] A full list of business experiments can be found at `http://www.beingrid.com/be.html`.

Customers discover providers through inspecting the marketplace or through assigning a resource broker to that inspects the marketplace for the customer.

Several requirements for the service marketplace are given in the same section. These include that marketplaces can be recorded in a common registry, that service providers, resource brokers and service consumers register themselves in the marketplace and that service providers and publish their "service prices" in the marketplace. As also noted in [28, Sec. D.4.4], the success of such a marketplace is dependent on there being a common SLA representation in order that customers can interact with the marketplace and, most importantly, understand what is being provided by the service providers using the marketplace.

Finally, the following statement is made in [28, p. 151]: "Grid services may be persistent or transient. The two types of services need different discovery technologies". Questions that arise from this statement are what are those differences and why do they need different discovery technologies? No further information is given.

6.3 SLA Representation

As BEinGRID have developed a series of abstract design patterns, each implementation may use any SLA representation it requires. So far, implementations in BEinGRID have used WS-Agreement and GRIA[10] representations.

6.4 SLA Agreement Process

The process and protocol by which SLAs are agreed is documented in [28, D.4.3]. The BEinGRID agreement protocol message sequence is the following [28, Figs. 40 & 49] :

1. An SLA template is retrieved from the service provider by the potential customer.
2. The potential customer completes the template and returns it back to the provider. The completed template is considered as a non-binding *quote request*.
3. The provider returns a non-binding *quote* to the customer describing the providers response to the quote request.
4. If the customer does not find the content of the quote acceptable then they can return another quote request to the provider and the process returns to step 2, above.
5. If the customer finds the contents of the quote acceptable it can return a *proposal* to the provider. This message, if accepted is binding — i.e. the customer cannot back out of the proposal once it has been sent to the provider.
6. The provider can then "accept or reject" the proposal. If the provider accepts the proposal the protocol terminates.

From the sequence above, steps 5 and 6 represent a simple agreement protocol with a pre-agreement phase (steps 2–5). Although some discussion occurs in this protocol, it is debatable whether this sequence of messages is full negotiation. For instance, if between step 3 (the customer receiving the quote) and the step 5 (the customer sending the proposal) the resource provider agrees to another SLA that means it cannot fulfil the proposal from the customer (based on a quote from the provider) the agreement phase will fail and the protocol will have to start again. An addition to this protocol could be to have the provider return a 'quote' in step 6 if they cannot agree the proposal or a 'reject' message to indicate that the provider does not want to continue with the negotiation.

There is also another problem with this protocol — it is incomplete. For example, what is returned to the customer to indicate that an agreement has not been made in step 6? This could be solved through the inclusion of the reject message mentioned above sent from the provider to the customer. However, given that the reject (and the signed SLA returned to indicate a successful agreement) can be lost in transit (or delayed for an infinite amount of time) there also needs to be some mechanism for the customer to query the provider to determine the current state of the SLA. As a solution it is suggested that the proposal message is allowed to be sent again to the provider and the response that the provider originally sent returned to the customer.

Another of the evaluation criteria for the SLA agreement process is how it meets the requirements of contract law. BEinGRID recognises the importance of this as in [28, D.4.5.1] they state (although about the SLA template) "as BEinGRID concentrates on business requirements the use of SLAs has always to be in line with legal aspects; this

[10] Homepage: `http://www.gria.org/`.

is critical to be usable in the business domain". In the BEinGRID's protocol the quote message can be mapped to contract law's 'invite to treat', the proposal message can be mapped to an 'offer' and the signed SLA returned to the customer as an 'accept' message. However, proposals/offers are not acknowledged as required in the EU e-Commerce directive, as described in [7].

This report also assesses the re-negotiation of agreements. BEinGRID do not provide a re-negotiation process as they state that re-negotiation "is a complicated protocol to follow for both parties" [28, p. 143]. However, the same deliverable also reports that the re-negotiation of an SLA could be achieved using "the same quote framework" [28, p. 149], i.e. using the same protocol. BEinGRID seem to have recognised that re-negotiation is simply negotiation within the context of an existing SLA, and it is disappointing that this was not pursued further.

Finally, in the BEinGRID protocol the provider only ever issues non-binding messages in the agreement process. This protects the provider from a denial-of-service attack as they have not 'locked' their resources and are free to offer the same SLA to another customer before a proposal message is returned by the original customer.

6.5 SLA Monitoring

The requirements for runtime monitoring and evaluation in BEinGRID are introduced in [28, D.4.2] and the abstract design pattern for achieving those requirements is described in [28, D5.1]. The requirements for SLA monitoring are straightforward: monitoring information is compared against the agreed SLA and if a violation occurs recovery procedures are started to rectify the problem or minimise the consequences of the violation. As shown in the figure on [28, p. 156] the abstract monitoring evaluation pattern works through a publish/subscribe pattern. The SLA runtime monitor in this component subscribes to monitoring information from the resource and the violation evaluator determines if this information indicates that the SLA is being broken.

The sequence of events from information entering the runtime monitor to violation notification is shown in the figure on [28, p. 158] and it appears from the text of the document that this sequence must be followed each time monitoring information is received. An open question remains about this design: if a high volume of monitoring information is being received or monitoring information is being received very frequently/many times per second, can this sequence, where the SLA is retrieved and assessed each time monitoring information is received, cope?

6.6 SLA Accounting

SLA accounting in BEinGRID is described as covering "the procedure of billing consumers for their usage of the provider's services according to the pricing terms agreed…in the SLA" [28, D.4.6]. The accounting component should, according to the same section, collect information periodically from the runtime monitoring component (described above) and this is shown in [28, Fig. 47].

The customer is allowed to find the cost of the SLA to them (the difference between the cost of the resources used and penalties accrued by the provider for SLA violations) periodically through querying the account manager component [28, D.4.6.1].

7 BREIN

BREIN (Business objective driven REliable and Intelligent grids for real busiNess, IST-2005-034556) is a business-focused FP6 project concerned with increasing the responsiveness of organisations and their interactions within multi-partner collaborations though intelligent adaptation of networks of supply chains. These supply chains may be within a single organisation or between organisations. The BREIN partners have recognised that current Grid middleware and tools do not exhibit the business-centric properties necessary to achieve the required flexibility in modern supply chain management and their goal is to enhance organisations with technologies and methods developed by the artificial intelligence, intelligent systems and semantic web communities.

The motivating scenarios for BREIN are that of virtual engineering and airport management, which are described in full in [29]. At a high-level, both of these scenarios are concerned with creating, managing and assessing business-relationships between a provider of services and their customers. Thus, BREINs interest in SLAs comes from this business-oriented approach to the provision of distributed services in order that qualities of service can be guaranteed by businesses to their customers and expectations can be set on that business relationship.

BREINs approach to providing tools and Grid middleware components is also innovative. Instead of developing their own solutions they are taking the best work of previous projects in the areas of Grid, semantic web, business workflow processing and security to develop solutions that meet the requirements of their motivating scenarios.

BREIN is a relatively young project and, at the time of writing, many decisions regarding its approach to a final design and the deliverable components are still to be made. However, what BREIN have done is to produce a set of 'rapid-prototypes' to gain experience in several key areas of technology relevant to their motivating scenarios. Some of of these prototypes connected to SLA creation and management, but the one of most interest to use manages a Virtual Organisation (VO) formed through an SLA negotiated between a service provider and a customer. This is known as the 'Virtual Organisation' prototype and it will be evaluated in the following sections. Thus, the experiences BREIN gain from their prototypes using SLAs will be fed back into their final SLA subsystem design.

7.1 SLA Subsystem Design, Monitoring, Accounting & Evaluation

As part of BREINs approach is to re-use existing work where possible they have decided to use the SLA management system developed as part of the TrustCoM project [30, A.9], [31, Sec. 3.3.2]. TrustCoMs SLA subsystem is described in Section 9.1 on page 27 of this report.

7.2 SLA Representation

As described above, BRIENs approach has been to re-use components from the TrustCoM project. In TrustCoM, the SLA representation used is based on WS-Agreement with WSLA terms, as described in Section 9.3 on page 28 of this document. Thus, the BREIN project also uses a standard WS-Agreement with WSLA terms for its SLA representation.

7.3 SLA Agreement Process

As the BREIN project is ongoing, the final SLA agreement process has yet to be finalised by the project members. However, as described above, as part of BREIN a series of rapid prototypes have been produced to demonstrate and evaluate the effectiveness of BREINs general approach. The Virtual Organisation prototype includes 'intelligent SLA negotiation' [31, Sec. 3.3] and it is this design that will be reviewed.

The agreement process described in [31, Sec. 3.3] is based on the FIPA Contract-Net Interaction Protocol [33] implemented using the JADE (Java Agent DEvelopment) framework. Contract Net protocol describes a flow of messages between an initiator and one or more participants in an agreement process, with these messages sent and received as shown in Figure 8. Although advertised as incorporating negotiation, the Contract Net specification does not allow for any form of negotiation to take place within the process; a proposal is presented in a 'take-it-or-leave-it' manner. Thus, negotiation is achieved in the Contract Net protocol through multiple instances of the protocol; if one proposal is rejected a new instance of the protocol is initiated. This behaviour of having to start again each time a call for proposals is rejected does not allow a long-running multi-round conversation to take place between the resource provider and requester. However, BREIN claim to have used Contract Net for multi-round negotiation [31, Sec. 3.3], but it is unclear how this has been achieved and how a solution to keeping the negotiation context across multiple instances of the protocol has been solved.

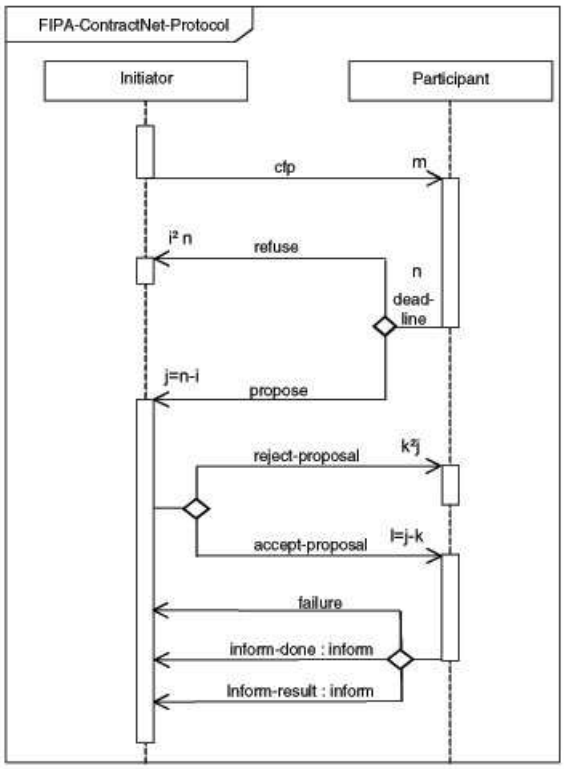

Figure 8: The FIPA Contract Net Interaction Protocol (From [32])

The Contract Net protocol has been standardised by the Foundation for Intelligent Physical Agents (FIPA). However, their specification [32] gives no consideration to the possibility of denial-of-service attack on the provider; for example if the provider is in the role of Contract Net 'participant'[11] it leaves the acceptance of the proposal up to the customer (in the role of 'initiator'). As we describe in [7] this leaves the providers resources 'locked' and unable to be offered to anyone else — a possible route to a denial-of-service attack. Alternatively, if the provider takes the role of initiator there is a problem too as the completion of the Contract Net protocol relies on an acknowledgement message from the participant to terminate. If the customer was being malicious it may never send this acknowledgement and leave the provider in a 'limbo' state.

The FPIA specification is described in a single Agent UML (AUML, which is essentially a UML sequence diagram) template (Figure 8) which shows a simple, ideal scenario and does not specify how all possible cases of message receipt and response should be dealt with. This is recognised in [34], which describes Figure 8 as "suffering from ambiguities and incompleteness". [35] also notes that the specification is "vague and containing errors" and that "even simple FIPA protocols, with only a few exchanged messages, contain a number of errors and imprecision. . . these errors are inherent to the AUML notation and are present in all AUML protocols".

Thus, the conclusion of the BREIN project that their implementation of Contract Net "offers [a] good protocol for SLA negotiation" [31, p. 33] is debatable.

[11] BREIN does not describe which Contract Net role the provider and customer takes in the Virtual Organisation prototype.

8 NextGRID

NextGRID ('Architecture for Next Generation Grids', IST-2004-511563) started in September 2004 and ran for 36 months until September 2007. "The overall vision of NextGRID is one of creating an infrastructure to enable new business" [36, p. 8]. NextGRID attempted to realise this vision through considering the requirements of stake-holders, such as users, providers and research organisations, and shaping the next generation Grids to meet their needs and to create new markets for Grid services. NextGRID attempts to lay the foundations through "architectural solutions that streamline all aspects of Grid operation" [37] from installation and maintenance of the infrastructure, development and deployment of Grid applications to the user orchestration of the resulting resources.

The NextGRID design and generalised specifications were produced by examining the "key Grid applications" [36, p. 6] in the areas of finance, digital media, supply chain management and electronic record processing. These design and specifications were based on the three architectural principles of [36, Sec. 5]:

- A dynamic Grid infrastructure that allows relationships and federations to be formed. This is necessary because, as in business or the commercial world, working relationships (in this case the sharing of computational or service-based resources) need to be created and destroyed rapidly and with the minimum of overheads.
- A minimal Grid infrastructure through a set of common capabilities shared by all participants is a key architectural principle of NextGRID to provide ease of installation and maintenance and therefore lower running costs.
- Service level agreements to "define the nature and consequences of an interaction between a service provider and customer" [36, p. 24].

From the list of architectural principles, SLAs between service providers and customers are central to the conceptual model of NextGRID because they form the foundation for the other two architectural principles. For example, the dynamic Grid infrastructure NextGRID envisages can only be realised through dynamically formed relationships defined though an SLA. Secondly, users of the NextGRID infrastructure can only determine if the capabilities they require if they are advertised through an SLA, which is "the obvious location to contain information that can be used…to ensure the correct environment…can be created by the service provider" [36, p. 25]. Thus, within NextGRID's output, this report is interested in the deliverables of NextGRID's work package 4, which describes how NextGRID performs service management through a framework which uses SLAs and guaranteed qualities-of-service.

8.1 SLA Advertising & Discovery

[41, Sec. 6.1] describes the mechanism by which SLAs are advertised and discovered by potential customers of the service provider. [41, Fig. 4] shows how the customer uses a special NextGRID 'Customer GUI' to access a discovery service to ask for a set of SLA templates from providers who may be candidates for negotiation. The discovery service interacts with two further services; a registry service, with whom the service providers have registered and which contains information on services provided by providers, and a SLA template repository. The discovery service first queries the registry to find the providers capable of meeting the requirements, then finds the appropriate SLA templates to return to the client.

8.2 SLA Representation

NextGRID have stated that they have chosen not to use either the WS-Agreement representation or terms from WSLA for their SLA because they "are too focussed on the technical aspects of the service and do not attempt to cover the service's business aspects" [43]. Instead, NextGRID have used WS-Agreement and WSLA as "inspiration" for their own schema [44, Slide 25]. The detailed NextGRID SLA schema has been made confidential by the NextGRID consortium so it cannot be reviewed in full in this report, though NextGRID have communicated that it may be released in the near future [45] (early-to-mid 2008). There is however a high-level explanation of their SLA schema in [46]. This shows how a NextGRID SLA is split into three parts[12] containing information on *parties*, *negotiable terms* and *static terms*. The following description of each of these parts is taken from [46].

[12] WS-Agreement, shown in Figure 1 on page 4, has only two parts.

- The *parties* section of the NextGRID SLA is similar to the *context* section of a WS-Agreement as it contains information regarding the parties taking part in the SLA. Unlike the *context* section of a WS-Agreement, this section "allows for the declaration of one or more supporting parties", e.g. trusted-third-parties who will be evaluating the SLA or arbitrating over disputes.
- The second section of the NextGRID schema details the "dynamic" or negotiable terms in the SLA, or "what is to be measured, how it is to be measured, the threshold for violation and. . . the method to determine if violation has occurred".
- The *static terms* elements of the NExtGRID SLA schema contains "elements [NextGRID] consider to be key in order to use SLAs in a commercial world". An example of one of these terms given in [46] is the ISO9001 compliance of a service provider. Also expressed in this section are terms which describe "the overall environment the application will run" and this is assumed to mean not only the commercial environment but also the technical environment too.

Given the available information about the NextGRID SLA schema, what is striking is that NextGRID have felt the need to create their own SLA document structure rather than extending the existing WS-Agreement standard. For example, SLA elements NextGRID feel necessary to be placed in the *static terms* element could have been placed inside an extended WS-Agreement *context* which defines "static metadata about the agreement" [4, Sec. 4.1]. Generally, creating your own custom schema like this is reduces interoperability and "the cost of rolling your own [schema] is a lot higher than you think, and you should really try to avoid doing that if you possibly can" [47].

Secondly, NextGRID differentiate between dynamic and static terms for completing the job inside the SLA. This approach assumes that terms that are static will always remain invariant. A different approach that could have been considered for this would have been to designate a general 'term' element and then mark it with a boolean 'negotiable' or 'non-negotiable' flag. Thus, if an element becomes negotiable (or dynamic) it can be marked as such without altering the structure of the document. A use-case for marking SLA elements like this comes from the OGF's GRAAP-WG requirement to re-negotiate of exiting SLAs [48] and the possibility that an element which could have originally been not-negotiable in the original SLA can be made re-negotiable if the circumstances of a party in the SLA changes. An example could be the latest possible time a customer requires a computational job to complete for; a hard, non-negotiable deadline the customer originally required, and marked 'non-negotiable', may change if the customers circumstances change (e.g. they require the job complete earlier). In this case a method of redefining a term as negotiable is required. This approach, in the opinion of this report, then allows SLAs to be as flexible as possible.

8.3 SLA Monitoring, Evaluation & Accounting

The monitoring of resources in NextGRID is performed by software on the monitored resource known as 'the monitoring component' [38, Sec. 6.3]. This component collects information it has been configured to collect and forwards it to the evaluation infrastructure[13].

Within the evaluation infrastructure, the *management information and distribution* (MID) service receives the information collected by the monitoring components [40, Fig. 6]. "The MID is an event relay that forwards all messages that it receives according to the management policies that it has been configured with. This allows a decoupling of the application and the evaluator component that is managing the application" [40, Sec. 4.2.1]. The *evaluator* receives and then processes information from the MID to determine if any corrective action should be taken, such as increasing the amount of resources available to the service, on the monitored resource.

However, before all this takes place the evaluator, MID and monitoring component are created and configured by the *supervisor* that " executes all the necessary steps to enact service management according to a specific a SLA within the operational management framework. It also executes all the necessary steps to decommission the management framework for this SLA once its lifetime comes to an end" [40, Sec. 4.2.1].

This approach with a single co-ordination component is similar to that taken by Akogrimo (shown in Figure 5 on page 5 of this document). However, the difference between the Akogrimo and NextGRID designs are that in NextGRID there is a single monitoring component for the whole system and an evaluation component for each SLA [40, p. 8] whilst in Akogrimo there is a monitoring service and evaluation component for each SLA.

[13] However, [39, Fig. 4] shows a 'metric poll service' polling for rather than being sent monitoring information.

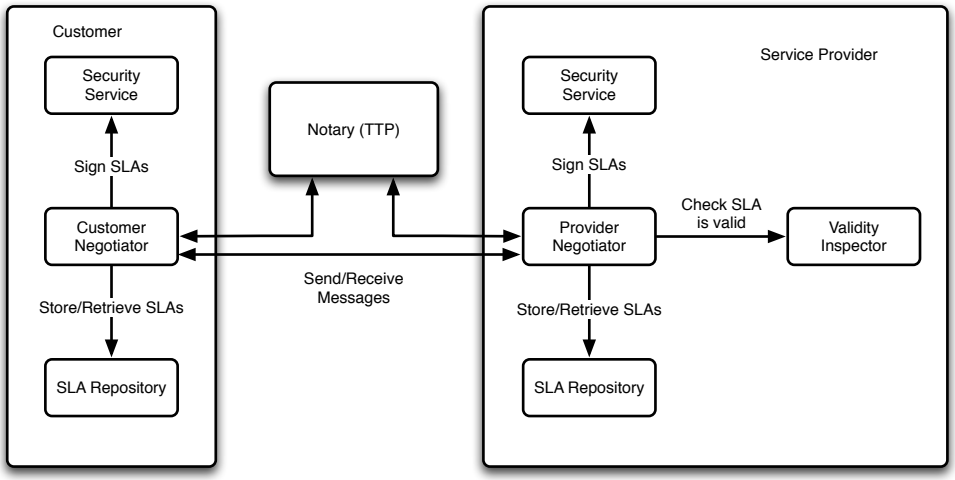

Figure 9: Components involved in the NextGRID Agreement Process

Accounting for resources used during the execution of a NextGRID service is carried out through the MID sending billing events to an accounting/billing component, as shown in [40, Fig. 13]. The accounting/billing component performs two functions: it collates the events that correspond to SLAs to form billing information and it also keeps "track of usage limits within accounts/SLA". It is therefore assumed that if the service customer has used their account/SLA usage limit the evaluator component will be informed of this and the service halted or the customer informed. However, in [40] no mention of this can be found and the accounting/billing component is seen as an endpoint with no capability to react to events it is passed.

8.4 SLA Agreement Process

The SLA agreement-process in NextGRID is described in [38, Sec. 6.1], which describes the message flow between the participants (the customer and the service provider). This sequence is as follows (all direct quotations are taken from [38, Sec. 6.1]):

1. During the discovery phase (described in Section 8.1) the customer retrieves an SLA template.
2. The customer completes and sends the template (known as an 'SLA offer') to the service provider.
3. The service provider compares the SLA templates it has stored to the offer received from the customer. "Once a matching template has been found, i.e. a template that covers all terms he wants to negotiate with matching values, the service provider will use the information in it to create a SLA proposal".
4. "If the service provider agrees to provide his service to the customer under the negotiated conditions, the service provider negotiator sends this SLA Proposal back to the customer".
5. If the customer finds the SLA proposal acceptable it "will inform the service provider of his acceptance (or declination)".
6. If the service provider receives an acceptance or declination they send a confirmation to the customer.
7. "After that the Customer and Service Provider sign the SLA".

The implementation of this protocol is described in [38, Sec. 7.2], where the components shown in Figure 9 are described. Each of the components shown in Figure 9 is involved in the protocol as follows:

- The customer negotiator is a service acting on behalf of the customer that requests an SLA proposal for a specific service from the Service Provider by sending them an SLA offer, as described above.

- The provider negotiator is a service acting on behalf of the service provider and is where SLA offers are received from a customer. The service negotiator queries the SLA repository, triggers validation of the SLA offer and performs the advance reservation for the offered service.
- Both the customer and service provider have a security service that sign and/or encrypt messages exchanged in the agreement process[14].
- The customer and service provider also have an SLA repository that contain SLA related information. The customer stores SLA templates containing their SLA goals in the repository so that they are able to determine if an SLA proposal received from a provider meets their requirements. The service provider uses the SLA repository for storing SLA templates that can be returned to a customer on request.
- The service provider's validity inspector takes the SLA offer from the customer and validates it against what is currently available [38, Sec. 7.2.4] so that the service negotiator can return an SLA proposal if what the customer is asking for is possible.
- The notary (a trusted-third party or TTP) may or may not be used in the SLA agreement process, it depends on how much each party trusts the other. If there is a lack of trust between the customer and service provider then the notary can be used to "compare the two SLAs [from the customer and service provider] by authenticating their origin and by verifying that there is no mismatch between the contents".

First, the requirement for the 'notary' is unclear. NextGRID are implementing the notary using PKI-generated asymmetric keys to signing the SLAs. In this case there is no reason for the notary to "authenticate the origin" of the signed SLA as it can have only come from the entity holding the private key and the receiver of a signed message can use standard PKI technology to determine this and therefore render the notary redundant. The other function of the notary is to "verify that there is no mismatch between the [SLAs] contents". Again, using a notary for this is not necessary as, when asymmetric keys are used, a signed hash of the SLAs contents can be used to confirm that a messages contents have not been altered. Each party can verify this hash independently of the notary.

From the message sequence on page 25, negotiation cannot take place in the protocol described; i.e. the customer completes a template and this is accepted or declined by the provider. This is not negotiation as defined in Section 3.2 of this report. NextGRID confirm this by stating "NextGRID advocates the "discrete offer" protocol" [36], i.e. a simple offer-accept model. This is corroborated in [42, Sec. 2.1.3], which states that "there is no scope for negotiation [in the discrete offer protocol] as the parameters of the offered services are fixed".

The protocol NextGRID describe is also incomplete. When the completed template is sent to the provider in step 2, what if the provider does not find a matching template or finds the SLA offer unacceptable in step 3? No instruction as to what the provider should do or what the client can expect is given.

With respect to the other evaluation criteria for the agreement process, this protocol does not meet the requirement of being legally-compliant because it is unclear as to when the SLA is actually formed, i.e. it is unclear which is the 'accept' message in the protocol: although the customer notionally accepts the SLA proposal in step 5, both parties only sign the SLA in the final phase (step 7). What if the customer does not sign the SLA in step 5? What if the message sent in step 6 never reaches the customer? Has the SLA been formed but not signed?

Finally, the protocol as described also leaves the provider open to the possibility of a denial-of-service attack as, from the protocol description, it seems that the customer is the party in the role of the offeree (i.e. it accepts the offer in step 5). As described in [7], this means the provider (in the role of the offeror) has locked their resources until the customer responds with their message accepting or declining the service providers SLA proposal, which they may never do.

[14] In [38, Sec. 7.2.5] the actions 'sign' and 'encrypt' are used interchangeably, but it should be noted that these do not mean the same thing.

9 TrustCoM

The TrustCoM project (IST-2003-01945) started in February 2004 and ran for 3 years. The goal of TrustCoM was to develop a framework for trust, security and contract management in dynamically evolving virtual organisations (VOs). Three motivating scenarios were examined to produce the TrustCoM framework and meet this goal: collaborative engineering, ad-hoc aggregated services and the construction of 'virtual communities' [49, Sec 1.1]. Each of these scenarios is introduced with respect to their interest in SLAs:

- The collaborative engineering scenarios come from the requirements of B2B interactions in the aerospace industry. Here, because of the scale of aerospace projects, companies work together in contractor/sub-contractor relationships or form supply chains for systems and components. Thus, in this scenario, there is a requirement to automate the dependencies "between contractual terms, policies, and enforcement and monitoring" [49, Sec. 1.2]. SLAs play a part in this as they encapsulate the contractual terms, duties and responsibilities of each party in the business relationships formed as part of the project.
- An 'aggregated service' is a domain-independent term used in TrustCoM to describe an application composed of other services. TrustCoM envisage a standard SOA-style architecture where individual services are offered by service providers on a commercial basis, i.e. "in exchange for payment or as part of a reciprocal arrangement" [49, Sec. 1.3]. Aggregate services are offered by VOs (which are either dynamically assembled or pre-existing) to customers with guaranteed QoS, encapsulated in an SLA.
- In TrustCoM, a virtual community is formed through members of a community coming together to form a VO. As a specific example, TrustCoM used the formation and operation of a credit union, where members can join and leave and deposit and withdraw funds according to agreed rules [49, Sec. 1.3]. These rules are encapsulated in a contract, or SLA.

TrustCoM form these VOs from what they call 'enterprise networks'. An enterprise network is pool of organisations who are willing and capable of forming VOs [50, Sec. I.3]. Therefore, an organisation must register with the enterprise network as a pre-requisite to being a member of a VO.

The Web Services-based TrustCoM framework consists of the following: a VO and enterprise network management infrastructure, a business process enactment and orchestration subsystem, a set of trust and security services and policy control and SLA management services [51, Sec. 1.2]. The final set of services is of most relevance to this report and will be reviewed here.

Finally, what was interesting about the TrustCoM project is that the consortium contained experts from the legal and economics communities to determine, document and assess the socio-economic and legal issues associated with forming and operating relationships between individuals and businesses. Cross-disciplinary involvement like this is to be encouraged as other subject areas often have produced solutions to problems the Grid community are attempting to solve. For example, the mechanisms of agreement formation are widely understood by the legal community and have been for many years.

9.1 SLA Subsystem

The TrustCoM SLA subsystem design is presented in [51, Sec. 5.1.1] and [50, Sec. III] and has been designed to negotiate SLAs, to offer monitoring services on a service, and to compare the performance to levels agreed when the SLA was signed [50, p. 41]. TrustCoM make extensive use of notification and publish/subscribe patterns to propagate information through the SLA subsystem.

9.2 SLA Advertising & Discovery

The TrustCoM project identify the need for the publishing and discovery of service capabilities in [52, Sec. 2.6.1.1]. These actions occur in the 'preparatory' and 'identification' phases of the TrustCoM VO life-cycle [50, Fig. 4].

In the preparatory phase organisations register their services capabilities with the enterprise network using the service registry and publish the SLA template for each service with an SLA template repository. These are shown as events with the prefix '1' in Figure 10. Then, when the VO is being formed, the VO manager queries a discovery service, which in turn queries the service registry to determine which services meet the goals of the VO being

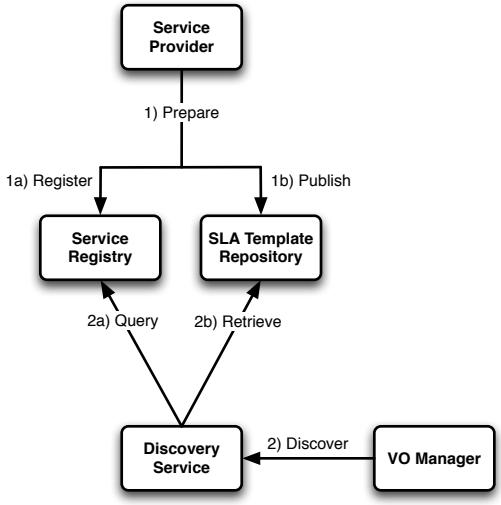

Figure 10: The TrustCoM SLA Advertising & Discovery Framework

formed [50, Fig. 32] (events labelled '2' in Figure 10). The discovery service returns a list of services that are capable of meeting the VO managers criteria together with the service's SLA templates from the repository and the address of the negotiation service which will make the agreement on behalf of the service. The VO manager can then use this information to agree and SLA with a negotiation service using the SLA agreement process described below.

9.3 SLA Representation

TrustCOM are unambiguous about what SLA representation they use: "the SLA document description is based on a combination of two standards: WS-Agreement, for the general document structure, and WSLA, for the description of QoS requirements" [53, Sec. 4.3]. That is, WSLA is used "for the specification of Service Level Objectives (SLO)" [53, Sec. 4.3.5]. Example and SLOs, such as available storage, response time and invocation count are given in [51, Sec. 5.1.2.3] and [54, Sec. I.2.f]. These terms/SLOs are well-defined and unambiguous.

9.4 SLA Agreement

The process by which SLAs are agreed in TrustCoM is described in [55, Sec. III.2.b], which shows the protocol consisting of a simple offer-accept exchange between the ServiceCustomerNegotiator component (acting on behalf of the 'customer', in this case the VO manager) and the ServiceProviderNegotiator. There then follows a 'signing protocol' so that each party can sign the agreed SLA [55, Sec. III.2.c] and have a trusted third-party (TTP) validate and verify the SLA returned.

The TrustCoM protocol is similar to (though less sophisticated than) the NextGRID protocol described in Section 8.4 and similar criticisms can be made of it. For example, no negotiation is possible and it is unclear as to what happens between the offer-accept protocol and the signing protocol if the customer decides not to return a signed SLA — has the SLA been formed or is it only formed after it has been signed? Further, how long does the provider wait for a signed SLA until it 'gives up'? During this period the provider has effectively denied any other party from booking the resources offered in the agreed SLA — route to a possible denial-of-service attack.

Also like NextGRID, TrustCoM have introduced a TTP to validate signed SLAs. As was mentioned when the NextGRID protocol was analysed, it is felt that the introduction of a TTP is unnecessary to verify digital signatures.

Finally, it should be noted that the protocol description given in [55, Sec. III.2.b] is only a suggestion and "the implementation of the particular negotiation protocol is left in the hands of the SLANegotiator components of both the service provider and consumer"[55, Sec. III.2.b].

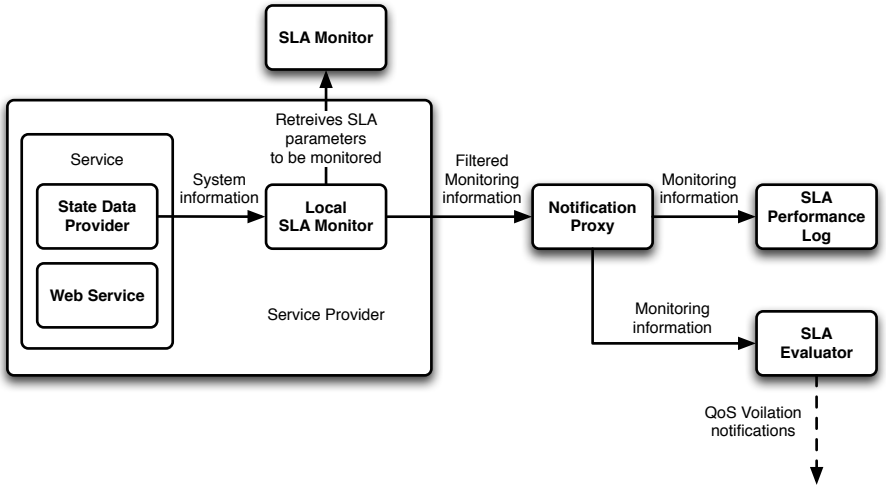

Figure 11: The TrustCoM SLA Monitoring Framework

9.5 SLA Monitoring & Evaluation

SLA Management in TrustCoM is described in [55, Sec. III] and shown in Figure 11. Once the SLA is active and the local SLA monitor has retrieved details about what parameters are to be monitored these parameters are sent via the notification infrastructure to the SLA evaluator and SLA performance log, which "accumulates historical data on the performance of SLAs for future evaluation and use (e.g. for accountability purposes)". These two components, are hosted outside the service provider by an external trusted third-party (TTP) [55, Fig. 12]. If violations of agreed QoS are detected by the SLA evaluator then it publishes notifications to interested parties that can take corrective action to make sure the SLA is met.

As with the use of TTPs in the agreement process, it is also difficult to see what TTPs add to the monitoring process. TrustCoM have used a TTP SLA evaluator because they "are trusted by the signatories of the agreement to be impartial in their evaluations" [55, Sec. III.1.i]. This may be the case, but their impartial assessment depends on data that is being sent from the service provider. There is nothing to stop the service provider altering the data before it is sent to the SLA evaluator rendering its impartial assessment meaningless. If the TTP model like the one TrustCoM have used is compared to the 'real world' and an e-Commerce platform such as Amazon, there is no external TTP evaluating if Amazon have fulfilled their duties in a timely fashion. This is because it is in the best interests of Amazon not to break their agreement as they recognise that misinforming or misleading a customer will not lead to repeat business and, according to a New York Times article about them, Amazon recognise that "taking care of customers can be the best way to build a lasting business" [56].

10 Summary

Table 1 on page 31 sums up the properties of the SLA subsystems for the projects reviewed in this report. The following sections describe a comparison of the FP6 projects by evaluation criteria.

10.1 SLA Publication & Discovery

From Table 1, it can be seen that there is no common mechanism for the advertising and discovery of SLAs although four of the projects (Akogrimo, BREIN, NextGRID and TrustCoM) have very similar designs, where a service provider registers with a third-party infrastructure though which the customer discovers the provider's capabilities.

10.2 SLA Representation

There are signs of a gathering consensus around using WS-Agreement as a container for QoS goals. However, there is a lack of consensus regarding a common vocabulary about how these goals should be expressed within a WS-Agreement document, although JSDL or WSLA terms appear to be popular, as shown in Akogrimo, AssessGrid and TrustCoM.

10.3 SLA Agreement Protocol

From the evaluation, only one of the agreement protocols proposed (from BEinGRID) allows some form of multi-round exchange which does not put the service provider at risk of a DoS attack. However, BEinGRID's protocol is incomplete and loosely-specified. It may be that BREIN's prototype protocol also allows multi-round negotiation, but not enough details were available to make a complete assessment.

10.4 SLA Monitoring, Evaluation & Accounting

A direct comparison of the subsystems used for SLA monitoring, evaluation and accounting is difficult as each project has designed their system to achieve a certain goal and, as a result, they all vary. However, from the evaluation summary, it can be seen that some projects may have scalability issues.

Evaluation Criteria	Akogrimo	AssessGrid	BEinGRID	BREIN	NextGRID	TrustCoM
Publication & Discovery	Services published and discovered through a ‘base virtual organisation’	Customer discovery directly to provider	Abstract marketplace model	As TrustCoM	Customer discovery through discovery infrastructure	Services published and discovered through an ‘enterprise network’
SLA Representation	WS-Agreement with WSLA terms	WS-Agreement with JSDL terms	None specified: upto the individual implementation	As TrustCoM	Proprietary	WS-Agreement with WSLA terms
Agreement Protocol	WS-Agreement	Offer-accept based on modified WS-Agreement	Hybrid: initial negotiation then offer-accept	Prototype is based on a modified ContractNet implementation	Offer-accept then signing phase	Offer-accept then signing phase
Monitoring, Evaluation & Accounting	Possible reliability problems because of number of components?	Passive: does not respond to breaches of SLA. Monitoring information used to calculate SLA penalty only	Possible problems if large volumes of data received?	As TrustCoM	Monitoring of usage quotas, but no action taken if breached?	TTP Model

Table 1: Summary of Evaluation

11 Conclusions & Future Work

This section provides our conclusions and suggestions for future work based on the findings of our evaluation and comparison.

11.1 Infrastructure vs. Components

From our evaluation, most of the projects reviewed have has produced a unique, complete infrastructure to provide a solution to a certain problem. Therefore, because each infrastructure is designed to be used in totality it is difficult to re-use the individual components of the platform delivered by each project. This has two effects; first, it is difficult to re-use components, such as an SLA subsystem, developed by a project without detailed knowledge of their infrastructure. Secondly, the interoperability between each project's infrastructure is limited, as shown by there being no common mechanism for SLA advertising and discovery. Interoperability is fundamental to creating open and competitive e-commerce marketplaces as it enables software systems built, deployed and managed independently to exchange and process each others information.

The approach taken by BEinGRID and NextGRID may be an answer to some of these problems since they remove some of the obstacles to participating in a BoG by providing a set of independent solutions to common problems like those mentioned in Section 3. This bottom-up rather than top-down style has been shown to work well in software design as it promotes code re-use and flexibility. A driver for this approach was the number of different business domains covered in the BEinGRID and NextGRID projects, therefore flexible and re-usable components were necessary to meet the requirements they found.

11.2 Advertising & Discovery

As with agreement protocols, each project we have reviewed has a different method to allow service providers to advertise their services and customers to discover those services. In addition, all of the projects require the customer to install special software in order to take part in each projects proposed solution to SLA advertising and discovery. This raises the barrier to entering into an SLA-enabled Grid and should be compared with discovering and using services on the web, such as on-line bookstores, which require very little use — often no more than a web browser, a valid email address and credit card. Secondly, also like the agreement protocols, each projects solution for advertising and discovery is not interoperable with another.

Our suggestion to meet the interoperable advertising and discovery requirement that has a low barrier to entry is to use an existing, proven, standard and interoperable method of advertising, publishing and discovering data: the Atom Publishing Protocol [57] and Atom Syndication Format [58].

The Atom Syndication Format is an a standard XML-based Web content and metadata syndication format that allows data to be made available within an XML Atom feed. An Atom feed can contain many individual Atom entries which are XML containers for any other type of document, e.g. XML, plain text or binary files such as music or pictures. There is no reason that an Atom entry cannot be an SLA template, such as a WS-Agreement document. A customer could therefore subscribe to an Atom feed from a service provider to receive the latest pricing, capabilities, QoS and availability of their services.

The Atom publishing protocol is a standard method of publishing content to an Atom feed and, as we have described, could be used by a provider to publish information about the services they offer to customers subscribing to their feed. Atom Feeds can also be aggregated into other Atom feeds, as demonstrated by the popular GoogleReader application[15] that can aggregate many Atom feeds and presented them in an intuitive manner. How this could be used an SLA advertising and discovery context could be, for example, where a third-party or broker aggregates and filter feeds from several providers to create a custom feed that meets a particular customers requirements.

In summary, the benefit of using these two standards is that a great deal of libraries, tools and end-user applications, many of them free and open-source, already exist that support and use the Atom Syndication Format and Publishing Protocol. For example, Microsoft recently confirmed that they saw the "Atom Publishing Protocol as the future direction for Web APIs" [59] and specifically for content publishing in a standard XML format. As we have argued, the same principles can be applied to the publishing of SLA templates, and we feel that the Grid community should take note of developments like this from the web world as they can often meet the the requirements Grid computing presents, simply and cheaply and in an interoperable manner.

[15] `http://www.google.com/reader/`.

11.3 Agreement Protocols

All of the projects reviewed allow an SLA to be agreed. However, all perform this agreement using different protocols and only one of the projects (NextGRID) allows some form of multi-round negotiation to take place. This difference in agreement protocols has come about because the current standard protocol to reach an agreement, WS-Agreement, is limited to a simple offer-accept interaction. This limitation has forced each project to modify or consider alternative protocols because the standard WS-Agreement protocol cannot meet their needs. The implication of this is that customers using the infrastructure based on one project cannot interoperate with any of the other infrastructures because each supports a different agreement protocol.

Secondly, some of the projects (e.g. BEinGRID, NextGRID) assume that an agreement *will* be reached in their agreement process and there is no provision made for the failure of an attempted agreement. It is our opinion that failure to agree will be the normal mode of operation and that agreement may only occur in a minority of attempted negotiations and an agreement process should be designed from this starting point.

11.4 Trusted-Third Parties

An issue that runs throughout the work reviewed as part of this report is that of shared infrastructure and trusted-third parties. For example, in Akogrimo the BVO infrastructure must exist in order that a service provider can join it to form an OpVO with customers; in AssessGrid, third-party brokers can act as a middleman between the provider and customer; BEinGRID envisions a marketplace where services are bought and sold; TrustCoM and NextGrid use 'notaries' to validate to perform verification and validation tasks in the SLA agreement and evaluation phases. There are many other examples we could have cited.

We feel that a great deal of this infrastructure and components are unnecessary. As we have noted in the evaluation, adding TTPs to an infrastructure is not a panacea to the problems of establishing trust and reputation and sometimes, such as in the case of digital signature verification, they are not needed. Also, if approaches like we have outlined for the advertising and discovery of SLAs are followed, this will remove the need supporting infrastructure such as service registries, SLA template repositories and discovery services.

Secondly, the Wikipedia definition of a TTP is "an entity which facilitates interactions between two parties who both trust the third-party". This leads to a fundamental question that can lead to the failure of the TTP model before it is ever used: what if either party does not trust the third-party and they cannot agree on which TTP to use? Presumably, introducing a TTP into, for example, an agreement process means that there must also be a pre-agreement to agree the TTP to be used in the actual agreement process.

Finally, another question about TTPs that remains to be answered by any of the projects is who actually pays for the running and maintenance of this infrastructure. The infrastructure must be hosted and operated from somewhere outside of either the customers or service providers administrative domain in order for it to remain independent and a 'third-party'. However, no information about how a TTP is to be made sustainable has been provided by any of the projects.

We feel these issues could be solved by using models that are already successful in the business world. For example, business generally operates without 'in-line' TTPs like NextGrid and TrustCoM propose to perform verification and validation tasks in the SLA agreement and evaluation phases. Businesses are required to be frequently audited by 'off-line' TTPs (e.g. accountants) and pay for that service to prove to their customers that they are operating in accordance with established rules and regulations. Secondly, if an SLA is established as a legal contract for service provision, there is always the threat to the service provider that the customer can take legal recourse if the SLA is not met to recover any losses they may have incurred and vice-versa — i.e. the TTP in this case is the legal system.

Thus, our recommendation is where possible to make use of existing bodies and methods of working that have already proven to be successful rather than re-inventing and duplicating these successful practices.

12 Acknowledgements

Michael Parkin is pleased to acknowledge that this work was carried out as part of an industrial fellowship for the CoreGRID IST project N° 004265, funded by the European Commission and partly sponsored by ATOS Origin, and for the support of Barcelona Supercomputing Centre and the Universitat Politècnica de Catalunya. This report expresses the opinions of the authors which are not necessarily those of the EC. The EC is not liable for any use that may be made of the information contained in this report.

The authors also acknowledge each of the projects for making the information used as part of this review available and particularly the NextGRID project and Francis Wray (BT) for making previously confidential deliverables available to us.

Special thanks for help in the production of this report go to (in alphabetical order) Francesco D'Andria (ATOS Origin/Akogrimo, BEinGRID), Donal Fellows (The University of Manchester/BREIN), Daniel Field (ATOS Origin/BEinGRID, BREIN), Bastian Koller (HLRS/NextGRID, BREIN), Igor Rosenberg (ATOS Origin/AssessGrid, BEinGRID), Lutz Schubert (HLRS/BREIN, TrustCoM) and Stefan Wesner (HLRS/Akogrimo).

References

[1] J. Treadwell (Editor). *Open Grid Services Architecture Glossary of Terms (GFD-I.044)*. Global Grid Forum, January 2005.

[2] J. MacLaren, M. McKeown, and S. Pickles. Co-Allocation, Fault Tolerance and Grid Computing. In *Proceedings of the UK e-Science All Hands Meeting 2006*, pages 155–162, September 2006. The HARC software is available from `http://www.cct.lsu.edu/~maclaren/HARC/`. Last accessed 30 May 2007.

[3] S. Bourbonnais, S. Malaika, V.M. Gogate, I. Narang, L.M. Haas, V. Raman, and R.W. Horman. Towards and Information Infrastructure for the Grid. *IBM Systems Journal*, 43(4):665–688, 2004.

[4] A. Andrieux *et. al*. Web Services Agreement Specification (WS-Agreement). Recommended Standard, Open Grid Forum, March 2007. Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG).

[5] Telemanagement Forum. SLA Management Handbook. Vol 2, Concepts and Principles, 2005.

[6] M. Parkin, D. Kuo, and J.M. Brooke. A Framework & Negotiation Protocol for Service Contracts. In L. O'Conner, editor, *Proceedings of the 2006 International Conference on Services Computing (SCC06)*, pages 253–256, September 2006.

[7] M. Parkin, D. Kuo, J.M. Brooke, and A. MacCulloch. Challenges in EU Grid Contracts. In P. Cunningham and M. Cunningham, editors, *Proceedings of the 4th eChallenges Conference*, pages 67–75, October 2006.

[8] G. Laria. Mobile and Nomadic User in eLearning: The Akogrimo Case. Akogrimo Whitepaper, February 2006.

[9] C. Loos. E-Health with Mobile Grids: The Akogrimo Heart Monitoring and Emergency Scenario. Akogrimo Whitepaper, February 2006.

[10] A. Litke (Lead editor). D4.3.1: Architecture of the Infrastructure Services Layer. Akogrimo WP4.3 (Grid Infrastructure Services Layer) Deliverable, July 2005.

[11] F. D'Andria, J. Martrat, P. Laria, G. Ritrovato, and S. Wesner. An Enhanced Strategy for SLA Management in the Business Context of New Mobile Dynamic VO. In P. Cunningham and M. Cunningham, editors, *Proceedings of the 4th eChallenges Conference*, 2006.

[12] J.M. Movilla, G. Gallizo, K. Konstaneli, N. Litke, F. D'Andria, S.J. Doval, N. Inacio, and R.L.A. Aguiar. An SLA Enforcement Architecture Implementation for a Mobile Grid Environment. 2007.

[13] A. Litke (Lead editor). D4.3.3: Report on the Implementation of the Infrastructure Services Layer. Akogrimo WP4.3 (Grid Infrastructure Services Layer) Deliverable, February 2007.

[14] G. Laria (Lead Editor). D4.4.1 Architecture of the Application Support Services Layer. Akogimo WP4.4 (Grid Application Support Services) Deliverable, September 2005.

[15] J. Jähnert and S. Wesner (Lead Editors). D3.1.1: Overall Architecture Definition and Layer Integration. Akogrimo WP3.1 (Overall Architecture) Deliverable, July 2005.

[16] Telenor. D4.2.1: Overall Network Middleware Requirements Report. Akogrimo WP4.2 (Overall Network Middleware Requirements) Deliverable, September 2005.

[17] A. Löhden (Lead Editor). D6.1.6: Final Dissemination Material. Akogimo WP6.1 (Dissemination) Deliverable, December 2006.

[18] G. Laria (Lead Editor). Report on the Implementation of the Application Support Services Layer. Akogimo WP4.4 (Grid Application Support Services) Deliverable, February 2007.

[19] H. Ludwig, T. Nakata, P. Wieder, and O. Wäldrich. Reliable Orchestration of Resources using WS-Agreement. CoreGRID Technical Report Number TR-0050, October 2006.

[20] A. Litke (Lead editor). ID4.3.3 Updated Grid Infrastructure Layer Architecture. Akogrimo WP4.3 (Grid Infrastructure Services Layer) Deliverable, July 2006.

[21] A. Litke (Lead editor). D4.3.2: Prototype Implementation of the Infrastructure Services Layer. Akogrimo WP4.3 (Grid Infrastructure Services Layer) Deliverable, January 2006.

[22] J.F. Molderez (Lead editor). D1.1: Requirements Analysis. AssessGrid WP 1 (Requirements and Validation) Deliverable, September 2006.

[23] J. Padget, I. Gourlay, and K. Djemame (Lead editors). D1.3: System Architecture Specification and Developed Scenarios (v0.30). AssessGrid WP 1 (Requirements and Validation) Deliverable, December 2006.

[24] AssessGrid Partners. Description of AssessGrid Prototype Implementation. AssessGrid 'Metadeliverable', October 2007.

[25] N. Lerch, H. Nitsche, K. Voß, and M. Hovestadt. First Steps of a Monitoring Framework to Empower Risk Assessment on Grids. In M. Bubak, M. Turala, and K. Wiatr, editors, *Proceedings of the 6th Karkow Grid Workshop*, pages 216–223, October 2006.

[26] M. Humphrey, C. Smith, M. Theimer, and G. Wasson. JSDL HPC Profile Application Extension, Version 1.0. Technical report, Open Grid Forum, October 2007.

[27] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D.P. Ovoca, and A. Savva (Ed.). Job Submission Description Language (JSDL) Version 1.0. Document GFD-R.056, Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG), Open Grid Forum, November 2005. http://www.gridforum.org/documents/GFD.56.pdf. Last accessed 30 May 2007.
[28] T. Dimitrakos (Editor). Design patterns for SOA and Grid. BEinGRID Meta-Deliverable AC1 (Integrating D1.1.2, D1.3.2, D1.4.2, D1.5.2, D1.6.2) v.1.0, July 2007.
[29] I. Jones (Lead editor). D3.1.1 Validation Scenario Definition. BREIN WP 3.1 (Scenarios and Conceptualisation) Deliverable, May 2007.
[30] B. Koller (Lead editor). D2.1.1 State of the Art & Reference Models. BREIN WP2.1 (Technological Background, Standards and Reference Models) Deliverable, September 2007.
[31] U. Pinsdorf (Lead editor). D4.2.1 Rapid Prototypes. BREIN WP4.2 (Rapid Prototyping) Deliverable, August 2007.
[32] FIPA TC Communication. FIPA Contract Net Interaction Protocol Specification. Specification SC00029H, Foundation for Intelligent Physical Agents (FIPA), 2002.
[33] R.G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1981.
[34] S. Paurobally, J. Cunningham, and N.R. Jennings. Verifying the Contract Net Protocol: A Case Study in Interaction Protocol and Agent Communication Language Semantics. In *Proceedings of the 2nd International Workshop on Logic and Communication in Multi-Agent Systems (LCMAS'04)*, pages 98–117, August 2004.
[35] S. Paurobally. *Rational Agents and the Processes and States of Negotiation*. PhD thesis, Department of Computing, Imperial College London, September 2002.
[36] D. Snelling, M. Fischer, A. Basermann, F. Wray, P. Wieder, and M. Surridge. Vision and Architecture White Paper V5. NextGRID Deliverable, June 2007.
[37] NextGRID Partners. NextGRID Factsheet. NextGRID Public Document. Available to download from the NextGRID website via the 'publications' link.
[38] B. Mitchell, P. Masche, B. Koller, and T. Sandholm. P4.5.3: Management Framework Reference. NextGRID WP4.5 (Advanced Deployment, Service Management and Migration) Deliverable, September 2005.
[39] B. Koller, B. Mitchell, P. Masche, and N. Quyen. P4.5.5: Validation of Management Framework Design. NextGRID WP4.5 (Advanced Deployment, Service Management and Migration) Deliverable.
[40] P. Masche, B. Mitchell, B. Koller, and M. Ahsant. P4.5.7 Revised Service Management Framework. NextGRID WP4.5 (Advanced Deployment, Service Management and Migration) Deliverable, April 2007.
[41] B. Koller, P. Masche, B. Mitchell, and H. Mizani. P4.5.6 Management Framework Reference v.3. NextGRID WP4.5 (Advanced Deployment, Service Management and Migration) Deliverable, February 2006.
[42] D. Snelling, A. Anjomshoaa, F. Wray, A. Basermann, M. Fisher, M. Surridge, and P. Wieder. NextGRID Architectural Concepts. NextGRID Public Document, March 2007. Available to download from the NextGRID website via the 'publications' link.
[43] P. Masche, P. McKee, and B. Mitchell. The Increasing Role of Service Level Agreements in B2B Systems. In J. A Moinhos Cordeiro, V. Pedrosa, B Encarnação, and J. Filipe, editors, *In Proceedings of the 2nd International Conference on Web Information Systems and Technologies (WEBIST)*, pages 123–126, April 2006.
[44] S. Davey. NextGRID Architecture. Presentation given to CoreGRID Summer School 2006 (CSS'06), July 2006. A link to this presentation can be found in the 'Training' section on the NextGRID website.
[45] P. Masche. "Re: NextGRID Representation". Personal email to M. Parkin (mparkin@bsc.es) regarding NextGRID SLA representation, July 2007.
[46] B. Mitchell and P. Mckee. SLAs a Key Commercial Tool. In *Proceedings of the 3rd eChallenges Conference*, October 2006.
[47] T. Bray. On Custom Schemas. Blog Article, June 2004. http://www.tbray.org/ongoing/When/200x/2004/06/17/
[48] T. Nakata et. al. ReNegotiationWishlists. OGF GridForge Wiki, September 2007.
[49] P. Kearney (Lead editor). D3: Case Study Scenarios. TrustCoM WP11 Public Deliverable, June 2004.
[50] M. Wilson, A. Arenas, and L. Schubert. D63: TrustCoM Framework V4. TustCoM WP27 Public Deliverable, March 2007.
[51] A. Orlov. D19: Basic TrustCoM Reference Implementation. TrustCoM WP34 Public Deliverable, September 2005.
[52] L. Schubert. D9: TrustCoM Reference Architecture. TrustCoM WP27 Public Deliverable, August 2005.
[53] J. Claessens (Editor). D71: Final Standardisation Report. TrustCOM WP13 Public Deliverable, May 2007.
[54] M. Wilson, A. Arenas, and L. Schubert. D63-A: TrustCoM Framework V4 Appendix A: Subsystem Architecture. TrustCoM WP27 Public Deliverable, November 2006.

[55] M. Wilson, A. Arenas, and L. Schubert. D63-B: TrustCoM Framework V4 Appendix B: Subsystem Architecture. TrustCoM WP27 Public Deliverable, July 2006.
[56] J. Nocera. Put Buyers First? What a Concept. New York Times Article, January 2008. http://www.nytimes.com/2008/01/05/technology/05nocera.html.
[57] J. Gregorio and B. de hOra (Editors). RFC 5023: The Atom Publishing Protocol. IETF Working Group Request for Comments, October 2007.
[58] M. Nottingham and R. Sayre (Editors). RFC4287: The Atom Syndication Format. IETF Working Group Request for Comments, December 2005.
[59] H. Wilms. Microsoft bets on Atom Publishing Protocol as the future direction for Web APIs. InfoQ Article, March 2008.