CoreGrid Research Exchange Programme Report.

REP Title	The Applicability of Dynamic SLAs for Grid Resource Management
Applicant	Viktor Yarmolenko
Hosting Institute	Cardiff University, Prof. Omer Rana
REP ID	CR32 UOM - CR25 UWC
Duration	11 weeks, starting 2nd June - 17 Aug 2008
Linked to	Institute on Resource Management and Scheduling (RMS)

Contents

1. Overview of the Work and Introduction.

The key objective of this project was to investigate the possibilities and the challenges of automatic (negotiation and) creation of Service Level Agreements (SLAs) for individual services and eventually applying the methodology to the automatic creation of SLAs for complex workflows, based on SLAs of the former. The plan was to build on our existing work [1] in the area of dynamic SLAs, developing it further in the context of usage and application of real workflows (local, remote, Web Services [2], *etc.*) using such workflow building environments like Triana [3], developed by the researchers at Cardiff University.

SLAs play a vital role in scientific Grid computing, and its wider application in a commercial world as a Service Oriented Architecture (SOA) marketplace. SLA is essentially a contract between (usually) two parties. Such a contract outlines the level of quality of the provided service, states guarantees of each quality of service (QoS) in a measurable form. The workflow builders like Tirana allow composition of several Web Services (WSs) in a single workflow. These WSs are hosted by third party providers and are spanning many administrative domains. The composed workflow can itself be offered as a WS. An interesting challenge is to perform the mentioned process in an economic environment, where the services offered should be accompanied by the relevant SLAs and, in case of Triana, these SLAs must be formed automatically during the negotiation process.

In [1,4] we already showed the advantages of the more expressive SLAs in terms of resource utilisation and performance. What was not clear however, is how such an expressive SLA can be constructed and what it would look like? How the data and the rules, constraints and guarantees of SLA can be materialised? What mechanisms and processes have to be deployed and what system design is appropriate. Can the current standards and state of the art SLA specifications accommodate the process of automatic population and composition of SLAs? It is these questions and respective challenges that I attempt to answer in this report, which is a result of 11 weeks of my research exchange programme (REP).

A focus of the experimental part of the research is the feasibility of using a historical information (or observation) about the execution of offered WSs for automatic (negotiation and) creation of SLAs. The created SLAs are sought to describe a set of important guarantee terms (such as response by a specific deadline, accuracy of result, price of the service, penalty, *etc.*) derived from this historic data, by analysing a set of relevant metrics such as probabilities and risks of WSs executions, input and output data, profit margins and other business metrics.

This REP links with research activities in the Institute on Resource Management and Scheduling (RMS) and is part of the following RMS Institute tasks:

- Task 6.7: Identification of requirements and strategies for supporting Grid business models within the scheduling and resource management process.
- Task 6.8: Service Level Agreements.

The brief description of the original work carried out during this REP follows:

- Performed a survey of SLA specification approaches;
- Investigated a use of historical data (or observations) to analyse and build dynamic SLAs;
- Investigated the applicability of building compositions of SLAs for third party services;
- Investigated compliance of an SLA specification standard, WS-Agreement, and identified the need for an extension, which includes references to other SLAs;
- Identified additional SLA terms required for SLA composition and for an effective SLA in general in the context of composition of third party services;

- Investigated mechanisms for obtaining and/or generating these additional SLA terms automatically;
- Performed a series of experiments to support my claims and findings from the mentioned above investigations.

The above list of original work is presented in this report in the following order. The identification of challenges of the outlined research programme is given in Section 2. The section is logically split in general introduction and the three main parts. These parts discuss various approaches to SLA specification (Section 2.1) and the choice of the appropriate specification in the view of the challenges identified in the introduction of Section 2; the design of the underlying system that would support such SLA specification in the context of new challenges (Section 2.2); and the relationships between the two addressing the challenge of SLA composition including related issues (Section 2.3).

The design of the prototype and the experiment setup (to support the claims outlined in Section 2) is followed in Section 3. The section is divided in three parts: the focus of experiments (Section 3.1), the design aspects of the prototype to be used in the experiments (Section 3.2) and the experimental setup (Section 3.3). The results and discussion are presented in Section 4, which are followed by the outline of future work in Section 5.

2. Identification of Challenges.

The first challenge is to identify the appropriate SLA specification approach in general as well as specific terms, additional to the traditional SLA, in the context of Grid. When examining various workflows and individual web services (that may consist of workflows) the following general types of SLA terms were identified, which were essential for the automatic creation, composition and analysis of SLAs:

- **service related guarantee terms** and conditions such as reserved service time, deadlines, size (processing power or storage) of the resource being reserved, bandwidth required, data traffic to be accommodated, contentions, availability, robustness or guarantees against failures (resulting in price and penalty or discount terms) and so on.

- **data related guarantee terms** and conditions (restrictions on the input data such as file sizes, or initial parameters as well as restrictions and/or guarantees with respect to the output data). For example, it may important to note in the SLA for what kind of input parameters or other input data the client is guaranteed a result. In other cases the guarantees for the output (or result) properties are more appropriate than input; for example, the output file is guaranteed to be in certain format or of certain size, the result is guaranteed to fall in a certain value range, *etc.*

- **unambiguous but descriptive relationships** between the above terms. Often, during the negotiation of an SLA, either client or provider may not be able to secify certain information about one or more important terms. Yet omitting such terms altogether is not acceptable. This is where a function-based SLA specification [4] is required. The both parties could agree on the (continuous) set of guarantee terms that put obligations on each and describe the successful outcome for all the possible turns of events. For example, for a given combination of input parameters provider guarantees a maximum completion time or minimum/maximum output size and so on. Only with the function based approach the potentially infinite number of outcomes can be neatly described in a brief description (more on this in Section 2.1).

- **terms referring to dependent SLAs** are particularly important for the SLA composition process. The terms of dependent SLAs (SLAs that have been co-allocated to compose a complex workflow out of individual third party WSs) may need to be renegotiated (*e.g.* in case of failures of one task of the workflow) or relationships between the terms of dependent SLAs need to be estimated. In the latter case, unambiguous descriptive relationships (mentioned in the previous point) may be not only between the terms of the same SLA, but also between the terms of two inter-dependent SLAs. Note, that such referencing is aimed to achieve extra flexibility in the management of services, avoiding potential SLA locking (more on this in Section 2.3)

The second challenge is to design a system and identify the mechanisms that would be able to automatically generate the traditional SLA terms together with the new terms discussed above. As stated elsewhere in this report, the need for such a descriptive and automatically generated SLA lies in the growing demand on service providers to supply better QoS descriptions as Grid enters realms governed by economic rules. More descriptive and flexible SLAs also facilitate the creation of value added services and their SLAs via composition of third party services/resources in an automatic manner.

In the next subsections these two challenges are addressed in more detail: by presenting a general survey of state of the art of SLA specifications (in Section 2.1.) and making the case for a function based approach; by describing in detail the system of choice (in Section 2.2.) for the dynamic environments, mentioned earlier, in which such descriptive SLAs will be used; and by looking at the current WS-Agreement standard, as a representative of traditional SLAs, and its weaknesses related to the support for SLA composition (in Section 2.3.)

2.1. Survey of Service Level Agreements Specification Approaches.

The essence of SLA in the context of SOA and its commercial exploitation is a legally binding contract. This contract usually outlines various QoS promised, constraints imposed and the associated business value defined for each QoS and its violation. In other words, SLA is a description of a successful business transaction. The purpose of any action or transaction is for it to be successful. Thus the goal of an SLA should be not only to specify a set of terms and conditions that constitute a successful transaction, but also to specify them in such a way that it enhances a chance of successful transaction.

Rolling out the above statement further, it is therefore appropriate that the metrics used to evaluate different SLA approaches (or SLA documents for that matter) should measure the success rate of the transaction. Therefore, the approaches discussed further can be generalised as a technique that express these satisfying conditions (or constraints, dependencies, guarantees – as they are also called). In other words, a good SLA (as any contract) should provide not only an unambiguous set of qualifying conditions, but also a complete coverage of all possible outcomes of the transaction.

	Parameter	**Constraints**	**Function**	**Logic**
Usage	Parameter-based usage SLA	Constraints -based usage SLA	Function -based usage SLA	Logic -based usage SLA
Instance	Parameter-based (instance) SLA	Constraints -based (instance) SLA	Function -based (instance) SLA	Logic -based (instance) SLA

The table above shows current approaches or specification techniques to record SLA, as observed in the literature and discussed with experts in the field. As it is shown in the table (as rows), there are two main usages of the term SLA in the literature. The *usage SLA* can be described as a long term contract with general outlines, similar to the *Terms & Conditions* disclosure, which a client is given

by a telephone or television provider, for example. This terminology can be seen in works from Globus Aliance [5] and researches that work closely with the former [6]. However, this is not the view of SLA on Open Grid Forum [7], namely in GRAAP Working Group [8], the authors of WS-Agreement specifications [9]. Their view of SLA is that of an *instance SLA* - that is an SLA which is specific to each individual transaction. Every job submission, database query or remote service invocation is accompanied by its own specific SLA. Note, that the notion of a template in WS-Agreement specification document is not the same as the *usage SLA*. The research outlined in this report is only focused on the *instance SLA* and is henceforth referred to as SLA.

There are four main approaches (shown in columns in the table) to record SLA, which vary in the way they view the satisfying conditions. These approaches are not mutually exclusive and can be used in conjunction with each other in SLA document.

Parameter-based specification is arguably the simplest and the most straightforward approach. The specification of an SLA is performed in a form of a record that consists of the set of parameters (as constants) - terms of an agreement. For example:

Type of service: Distributed Job
*Time duration: **5** hours*
*Size required: **8** CPU nodes*
*How much: **5** candies paid by client*
*Failed Service: **2** candies compensation paid by provider*

Such specifications are used in trade of simple or commodity services, such as CPU time, online storage. The obvious benefits of this approach are the simplicity of negotiation and automatic creation of SLAs, automatic monitoring and enforcement of the specified SLA constraints, small size of SLA. The obvious drawback of such approach is lack of flexibility and adaptability to rapidly changing conditions in dynamic environments such as Grid. These are outlined in more detail in [4].

Constraints-based specification is a natural step up from the parameter-based specification and views the terms of the agreement in format of acceptable ranges, such as: upper/lower bounds, intervals (inequalities), intervals with strides and so on. They are in fact numerical constraints and as such they capture not one but all qualifying conditions in range. Following the previous example for the parameter-based specification the following addition to an SLA would be a constraints-based description of an SLA term:

*Acceptable execution time: between **12AM** and **12PM***

Function-based specification is qualitatively more expressive than its constraints-based counterpart. Unlike the constraint-based approach, it does not only encapsulate all the possible outcomes for a specific SLA term but also allows interdependencies between the SLA terms. So the parameter or a range that define a term is not constant but can be dependant on another parameter or a range. In other words it is a function of another SLA term which may also be a function of another term. Following the previous example for parameter-based specification the alteration of the two SLA terms below would be an example of function-based SLA specification:

*Time duration: **40** hours **divided** by **Size required***
*Size required: between **2** and **8** CPU nodes*

Logic-based specification may have many guises such as first- or second- order logic or description logic and the family of knowledge representation including semantics, *etc.* This specification methodology could be very powerful but is most difficult to analyse, interpret and verify, among the four approaches presented. Compared to the function-based approach, the description logic does not

have the background information about qualifiers (e.g. CPU speed, memory size, CPU size or time) *hardwired* into the recorded SLA specification. This means that this approach is more geared toward abstraction of services, specifics of which can be easily changed without need to change any rules or relationships between the qualifiers. This approach might have a strong case in truly heterogeneous hybrid systems but currently is overkill for the types of services present in Grid. It is difficult to provide a compelling but a simple case for the logic based approach, partly because the case for using logic for representing SLA is stronger when it's least simple, and partly because there are no real use cases for it. A crude example, however, may look like this:

$$CPU\ Required \subseteq (CPU \cap \exists \mathbf{hasSpeed} > \mathbf{500}) \cup (CPU \cap \exists \mathbf{hasSpeed} > \mathbf{100} \cap \exists\ \mathbf{hasGPU})$$

Here, the constraints and requirements are joined by the simple logic which says that required service needs to have a CPU with a *speed* of at least 500 or to have a Graphics processor (GPU) but the qualifiers such as *hasSpeed* can be reinterpreted without changing the formula.

My conclusion is that each approach has its application, but there is no strong use case for the logic-based approach on the horizon. In fact, I had difficulty convincing some of the participants of OGF [10] and CoreGRID [11] meetings for the case of function-based approach. On the other hand, the parameter- and constraints-based approaches are simply not descriptive and flexible enough for the challenges investigated in this project. Therefore, based on the previous experiments [4,12], the function-based approach to SLA specification was chosen, on the basis that it is powerful and flexible enough for the job without additional baggage of having to build and describe a framework which interprets external factors and historical data of the services and automatically creates an ontology-based agreement. Nonetheless, the high level design and the use case environments, presented in this report do not limit the use of more descriptive approaches such as description logic.

2.2. The System of Choice: a Big Picture.

The challenges discussed earlier present the scope of consequent issues and challenges to be addressed significantly wider than the scope of this research programme. Therefore this report is focused on a specific use case where the service provided consists of a specific workflow with fixed functionality (not an arbitrary code), the behaviour of which only changes with the input parameters from the client's request. Thus, the service is treated as a single task or a unit which is reflected in a single SLA that describes the service. Examples of such workflows may range from a database query to number crunching calculations of weather prediction algorithms. Owing to a large combination of possible parameters that can be applied to the workflow it is unlikely that the collected data about, say, its execution would yield an exact match. Moreover, there's bound to be a deviation in the workflow execution behaviour (such as already mentioned execution time) even for the same input parameters (for example due to the background workload of the resource). To stay competitive, the service provider has to offer a better service (with more attractive QoS guarantees or SLA) at the best possible price. This can be achieved by the ability of the provider to anticipate the behaviour of its SLA workload.

The overall system design is presented in Figure 1. All the elements of the system will be briefly discussed in this section, followed by the detailed description of the few elements that comprise the focus of this study. Arrows in the diagram represent the direction of the information and instructions flow. The reservation and negotiation of the resource by the client is achieved via *Negotiation* module. This is where SLA is formed. The guarantee and the business terms are agreed and recorded during this process. All of the SLA terms are provided from *Analyzer* (more on this in Section 3.2). Once an SLA is formed, its copy is stored in the *SLA Bank* of the service provider, which is used by

the *SLA Scheduler* to allocate resources for all of the jobs stored in *SLA Bank* according to the constraints recorded in each SLA.

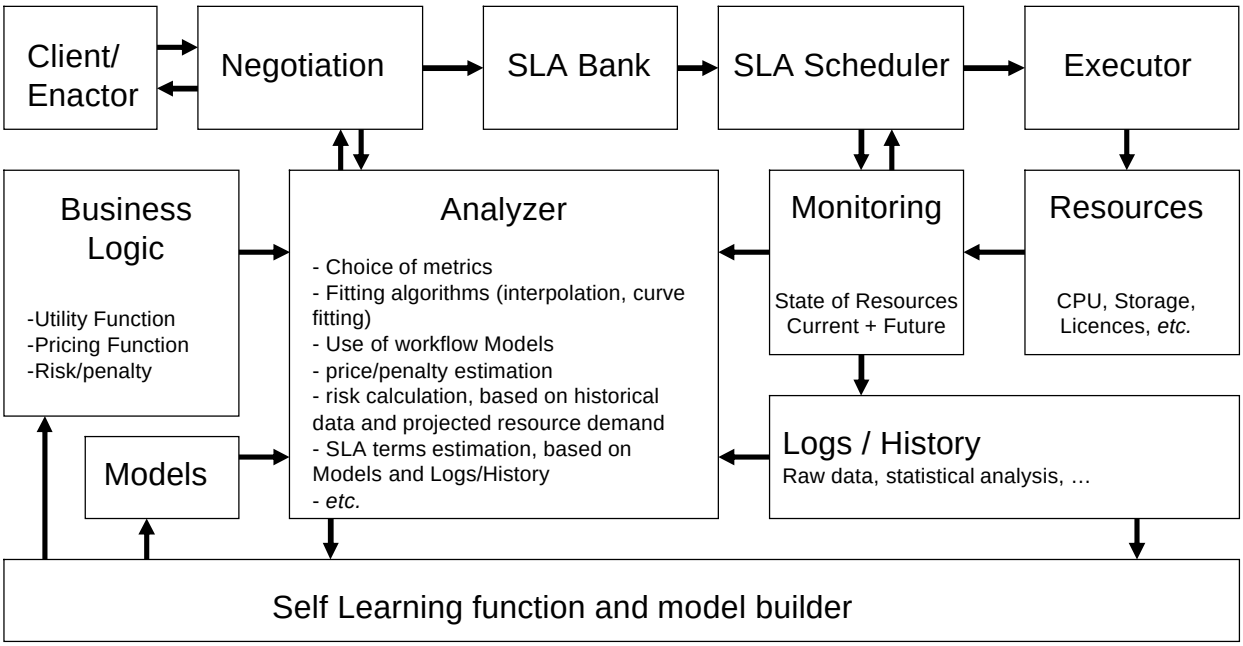

Figure 1. System design on the level of the local resource.

Executor simply performs operations such as starting and stopping of the job, acquire or release of necessary resources, as prescribed by *SLA Scheduler*. The information about job executions and related data from the *Resources* and *SLA Scheduler* is then collected by the *Monitoring* module. This module records information about execution of each service (workflows and individual jobs that comprise it) and related metrics, such as execution times, failures and crashes, input parameters and so on. All this data is organised in a structured form and is passed to *Analyzer* and *Logs & History* modules, where the data is stored and statistics about service executions (resource load, failures, utilisation, *etc.*) are compiled. The information from *Logs & History* is also fed into *Analyzer*.

There are two more modules from which *Analyzer* draws information. These are *Business Logic* and *Models*. The former provides information about utility functions, pricing strategy, risk functions and other business data, whereas latter provides analytical information about function of each service and as such complements the information from *Logs & History*. Both modules are static source of information (rules, constraints and functions), unlike the rest in the system presented where data is generated dynamically.

A simple example of *Models* in action is a provision of the information about the dependence of the execution time of the task as a function of one or more input parameters (recorded in SLA). Partial information on the workflow metrics is likely to be available from the developer (software spec). One of such common metrics - the execution time of a process (say, a tree search routine) - may depend exponentially or as a square on one of the input parameters. This information, in conjunction with the statistical data from *Logs & History*, provides a basis for the estimation of the resource demand for each future service. The accuracy of this estimate affects the cost and the risk terms of the negotiated SLA. This estimate and prediction is performed by *Analyzer*.

The final module, located at the very bottom of the diagram (Figure 1), will be briefly discussed in the future work and is outside the scope of this investigation. The potential future investigation depicted by this module can be summarised in the applicability of self-learning algorithms and/or other techniques to the automatic building of analytical workflow models and business functions based on the historical data of these workflows.

Section 3 will focus on the specifics of *Analyzer* as the most interesting part of the entire system design to be chosen due to the time limitation of the current research programme, whereas the rest of Section 2 discusses the role of the mentioned system design (Figure 1) in a bigger picture, namely the orchestration of complex workflows from the individual third party services and the challenges associated with such orchestration. A generic view of such prospective is presented in Figure 2.

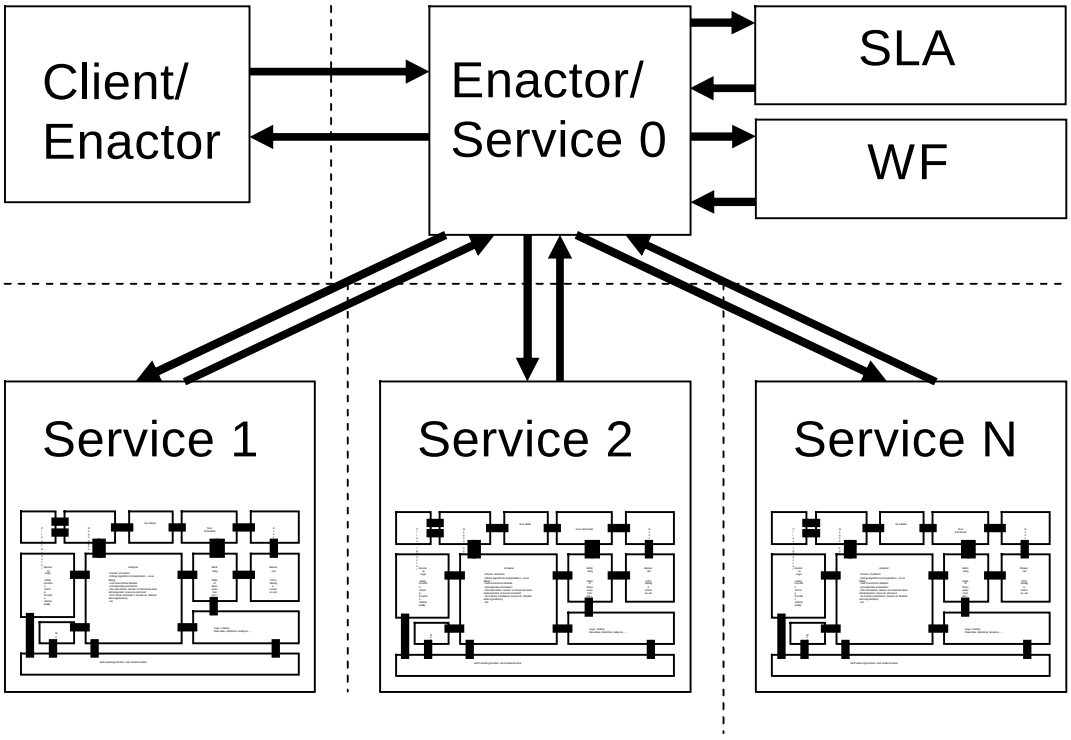

Figure 2. System design, one level up from the local resource.

With the expressive SLAs such as outlined in earlier sections an enactor (service zero) could assemble complex workflows from the services provided by our system of choice, orchestrate and co-allocate the necessary resources and form SLAs with each of the services involved, analyse the terms of these SLAs and produce a single SLA between itself and the client automatically. The dashed line in the diagram represents the boundaries of administrative domains. The argument is that the more expressive are the SLAs the more likely the above diagram to be implemented in the real world and the essential ingredient for this to happen is the appropriate SLA specification and the mechanisms or a framework that enable automatic generation of such expressive SLAs. Another, as important, characteristic of the successful SLA specification is the ability to refer to other SLAs, which is described in detail in the next section.

2.3. Challenges with SLA Composition.

The interest in SLAs in the context of the workflows on Grid was steadily growing since early 2000s. This interest picked up the pace between the time, when the researcher of the current REP chaired a meeting on Service Level Agreements (SLA) for the workflows (as a part of a CoreGRID WP6 meeting [13] in Manchester), and the present day. This development of the interest is natural for two reasons: (a) SLAa are essential in commercial Grid environment and (b) the substantial bulk of benefits that Grid offers lie in aggregation and composition of distributed resources into a big workflow.

Despite the growing interest in these two areas, the literature on SLAs as well as work on SLA standards (at OGF for example) largely ignore the support for workflows; for example, WS-Agreement standard does not touch on workflow related aspects of Service Level Agreements. The reason for such lack of progress on the topic (at least in part) is due to the number of serious challenges that need to be addressed before the SLAs and the workflows can be reconciled on the Grid arena. Apart from the ancient issue of trust between the parties, which the presence of SLA in the transaction should be able to resolve, the problem can be summarised as the lack of control that resource providers have over their own and subcontracted resources. This problem encapsulates a larger bouquet of inter-related problems that deserve their own research each, to name but few:

- The lack of up-to-date information about the resources and the processes executed on these resources;
- Inability to respond swiftly and efficiently to any changes to the SLA document (*e.g.* as a result of re-negotiation) or those that happen on the resource (*e.g.* as a result of resource failures, process failures, other external factors);
- Inability to reorganise workload automatically and efficiently on request whilst satisfying agreed SLAs;
- Inability to anticipate or plan SLA bound workload at the complexity level required as a result of a larger scale SLA composition.

The list could go on. The lack of control, knowledge and flexibility as well as effort involved to achieve the sufficient levels of the former, forces resource providers to be very conservative in their collaboration. The issues addressed in Sections 2.1 and 2.2 are essential steps toward resolving the problem discussed in this section (unwillingness to collaborate) and building more powerful Grid services. To summarise main points of Sections 2.1 and 2.2, we need more flexible and expressive SLAs which encapsulates much richer context in a form that is easy to generate and interpret, as well as system designed to be sensitive and intelligent to the workload and related data, SLAs and related SLAs, economic factors, other factors and their effect on and interdependencies with each other.

More intelligent systems (such as that suggested in Figure 1.) will be able to create dynamic SLAs automatically that give more flexibility to the higher level brokers/enactors/orchestrators. These high level enactors can combine third party resources in complex workflows, offering the latter as a new service to their clients (*e.g.* other brokers, enactors, orchestrators). Such new services would be backed by a new SLA, which is generated automatically based on dynamic SLAs already agreed with each third party resource (Figure 2). Are the current SLA specification approaches up to the task? The survey in Section 2.1, which reviewed different approaches to SLA specification and structure in general, does not suggest that any of them explicitly support the composition of SLAs by design. However, the SLA specification of choice (a function-based approach) does contain properties required for SLA composition. What is still required however, is the framework for linking dependent SLAs and the protocol to support it.

In view of this situation the REP researcher arranged several meetings with CoreGRID members as well as commercial companies that are particularly active in the area of SLA and are interested in its practical applications to the workflows on Gird (the list of participants is found at the end of Section 5). The agenda was to bring up the concerns, outlined in the beginning of Section 2, regarding the lack of support for workflows in SLAs using WS-Agreement specification as a common ground. In particular the following points have been raised:

Point 1. The advantages of dynamic SLAs [4] and their potential to address the challenge of supporting the workflows that consist of a third party services.

Point 2. The ability of WS-Agreement documents, as the standard currently stands, to refer to other WS-Agreement documents and how important and or necessary this is for the support workflows that consist of a third party services.

Point 3. The ability of other parties (besides initiator and receptor of WS-Agreement) to renegotiate parts of the agreement (possibly with limited access to the SLA document).

Point 4. What are the other standards that may be affected by and involved in the support provided for workflow management on Grid.

The scenario to consider was that described by Figure 2 (of Section 2.2.). The workflow allocation and execution cycle on such a Grid system is depicted in Figure 3. The client submits a workflow to the workflow execution service (FW Exec) or enactor, which schedules parts of the workflow to the local resource managers (LM) or other FW Exec (see the diagram in Figure 3). Each arrow in the diagram would require a separate SLA to be negotiated and created between the two relevant parties for every segment of the entire workflow submitted on a distributed resource. In addition, each arrow may be crossing an interface between different administrative domains.

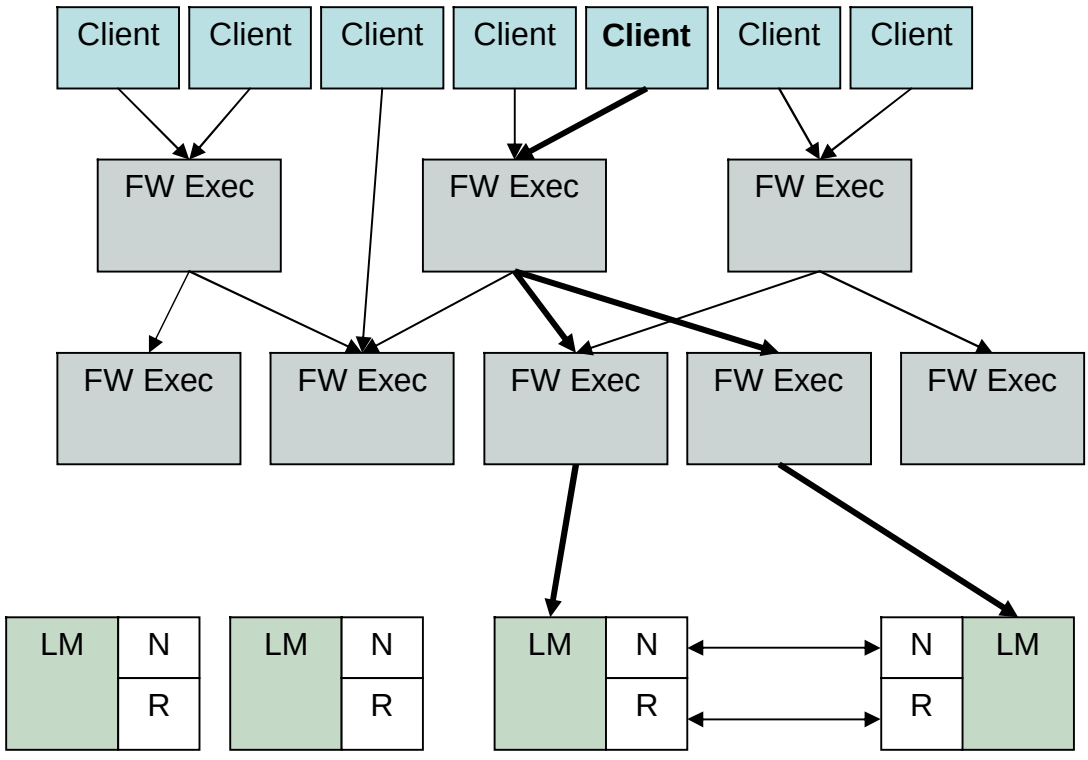

Figure 3. The workflow allocation, execution and the exchange of instructions.

The efficient scheduling of a workflow or even general robustness of the system decreases dramatically with the scale of the system. In such scenario, a flexible or dynamic SLA is required and the need for intensive renegotiation of SLAs is very likely [4]. To make the system scalable, WS-Agreement (and possibly other standards, such as WS-Notification) would have to be extended in order to allow LMs to renegotiate parts of the agreement which does not belong to them (i.e. SLAs, in creation of which LMs were not involved) but which indirectly relates to that SLA which does belong to them. For example two LMs can have separate agreements with the same or different FW Exec, each concerning a separate portion of the same workflow. In this case, the changes and failures of one SLA severely affect the success of another SLA.

The preliminary investigation of this scenario suggests the involvement of notification protocols (relates to N in Figure 3) between various parties that share the responsibility for execution of the entire workflow (point 4) and direct renegotiations (relates to R in the figure) of SLAs between the parties that did not participate in the creation of this SLA initially (points 2 and 3). The stress on LMs and WF Execs can be significantly reduced by carefully designed dynamic agreements (point 1), the terms of which may refer to the terms of other agreements (point 2). This means that WS-Agreement, WS-Notification and possibly other standards (for example involving authorisation) need to be reviewed. Following the examples in Section 2.1, the possible view of such referrals in SLA terms is presented below:

*Acceptable execution time: between **12AM** and **12PM*** (re-negotiable, see ***Dependencies***)
Dependencies: *Execution time of SLA#URI=uniqueHandle:12345*
Price of SLA#URI=uniqueHandle:12346
Bandwidth reserved of SLA#URI=uniqueHandle:12347

Such SLA specification allows the limited list of third parties, which did not participate in the creation of the original SLA, to re-negotiate directly a specific SLA term only as much as they are allowed by the original creators of this SLA. The actual SLA would also contain other information in the example given, such as the relationship between the terms of different SLAs. The simple example being to ensure that the reserved time of the service is increased in case the bandwidth is to be decreased, as well as the limits on such alterations of the terms.

3. Experiment Design and Objectives.

The discussion in Section 2 clearly shows the big scope of the problem chosen to be investigated. This is evident from the challenges (Sections 2.1-2.3) that themselves stemmed from the problem space, which was singled out from the original problem space, initially identified. During the identification of the mentioned challenges, a new set of issues to consider has emerged. This forces the current REP (owing to the time limitations) to narrow the focus of the experimental part of the study. As we study diagrams from Section 2 carefully, it becomes obvious that the most unexplored and challenging part of the chosen system design on the level of the local resource is represented by *Analyzer*. On the higher level, the most challenging problem is the composition of several SLAs and it has been explored to some degree in Section 2.3. Therefore, the rest of this section concentrates on the practical aspects associated with *Analyzer* from the diagram in Figure 1.

3.1. Focus of the Experimental Research.

The negotiation protocols and strategies were widely discussed in [14,15], whilst the structure and formation of SLA was discussed in [9,16]. The SLA aware scheduling of interdependent concurrent

tasks is relatively unexplored with interesting results in [4,12,17]. Some of the popular industrial execution engines [18,19,20] are not SLA aware systems and the work on integration of SLA aware schedulers with existing solutions is practically non-existent, but not the most challenging problem here. Monitoring and logging was widely discussed in [21,22]. The work on business aspects of the service provision on Grid is not as well covered as monitoring or negotiation aspects, nonetheless some interesting work was done in [23,24].

The use of historical data, from the workflow executions and other related information, for predicting execution times of future executions is not completely new [25,26], but there was no evidence of work found which applies such techniques in SLA aware environments for the purpose of generating guarantee and business terms of the agreement. Thus, this is the most sensible choice of exploration with a potential to be a source of many intriguing and fascinating challenges.

3.2. Analyzer: The Prototype and Techniques

The experiment will employ a few simplified interpolation and statistical techniques that use historical task execution data to generate the guarantee terms and predict the task runtimes and properties of the output data and establish how feasible the use of these predictions is in the economic environment. The use of and the case for dynamic SLA in the given system design will also be presented, using a functional approach to SLA terms specification (as argued in Section 2.1). It is anticipated that the mechanisms and techniques used to estimate runtimes and other data for each service (see following sections) could be used to derive analytical functions, which in turn are used to form SLA terms, and thus provide more expressive SLA which covers a large set of outcomes. Note that the main stress of the exercise is in the context of dynamic SLA specification and not interpolation techniques.

For example, we know that a specific database query may produce between 10 and 1000 entries in its output with no way of telling how many exactly. Knowing the previous input parameters for the query and correlating them with the potential size of the output, the SLA guarantee term (which describes the size of the output as a function of the input parameters) is then constructed. This way enactor knows exactly what to expect from the database query and is able, in the first place, to limit the size of the output when the execution time reaches an agreed value.

The examples chosen in the experiments are slightly different, but are seen in the majority of the real cases of WS on offer and as such serve as an adequate proof of concept. The environment chosen for building and testing of the proof of concept prototype is a set of services acquired and enacted using the workflow management package Triana [3], built at Cardiff University. Each chosen service consists of a known workflow or a remote WS (which may or may not be a workflow itself).

The metrics chosen in the experiments are the execution time of the entire workflow (or agreed service) and/or the size of the output. The values of these metrics may change with the change in input parameters in undetermined way. For this prototype we choose a simple method of aggregation of input parameters and building the metrics map - an N-dimensional system of coordinates (hyperspace) is built; with input parameters each represented by an orthogonal axis. The measured metric of interest (in this case execution time and data output size of the service) from each requested service is represented by a point in the hyperspace. This cloud of points is then used as a frame reference for estimating future behaviour of the requested service. Below is discussed a set of different techniques applicable for different circumstances.

3.2.1. No redundancy in the input parameters.

In cases when the combination of the input parameters to the workflow are seldom (unlikely to be) repeated, the historical data is represented in the following way. The unique points in the hyperspace are used to build a continuous surface which will have the solution (for the metric of interest) for any combination of the input parameters. The surface can be built using a number of mathematical techniques:

- **A simple construction of polygons** that join the data points to provide a crude surface in semi-analytical form – a function based on 3D mesh. From this function any combination of input parameters returns, say, the projected execution time of the service.
- **A polynomial interpolation** involves fitting the surface defined by the N-degree polynomial so that there is no break or discontinuity on this surface present (unlike the previous method)
- **Adjacent averaging** does not require for the surface to pass through the collected data points exactly but builds a smoothed version of the original surface defined by these data points.
- **Fourier Transform filtering** removes the high frequency ripples of the original surface, also resulting in a smoothed version of the surface.
- Other curve fitting and interpolation techniques could be explored

When the input data for the new service falls outside the historical record, a number of extrapolation techniques (for example using Lagrange polynomial) could be applied possibly in conjunction with the model of the task shown as *Models* component in Figure 1 (if the model for the exists). Most of the methods discussed yield error estimates which can be used to assess the confidence of the prediction and the risks involved. Also, by coarse graining the space or using clustering techniques, a redundancy in the data points of the hyperspace can be artificially created, whereby each coordinate (set of input parameters) may contain more than one value of the execution time. In the latter case there are more options available or the estimation of the required metrics of the service behaviour (*e.g.* execution time, output data size).

3.2.2. Parameters with data redundancy.

Parameters with data redundancy can be viewed (in this use case) as statistical data. Therefore, there are more options available for interpreting this data and constructing the surface or a map of execution times, mentioned previously. The required (*e.g.* mean or average) value can be calculated from sum of squares, as an average or the peak of the statistical distribution. Then the surface is built using the methods described earlier, based on the new single value for each known point in the hyperspace. Alternatively, more complex relationships could be employed, for example setting a weight value for each point and build a smooth surface whish is closer to the points with more weight. The number of methods available is large indeed.

3.3. Running Experiments.

To test the prototype and the concept described in this document we choose 2 specific tasks. One is an image processing workflow assembled in Triana and executed locally and another is a text based web service.

The image processing workflow is created from the typical image processing tasks (colour split, blur, 2 image diff, *etc.*) and as such is representative of the behaviour of such workflows created and executed either locally or offered as a service. The workflow is executed many times using different images of the same resolution for inputs. The images of different resolutions, and different files sizes are also used.

The text based web service chosen is Morse code converter [27]. This is a third party service where the client or enactor has no control over it. The service is chosen for its availability on the web, the process behind the operation of this service is understood, which is important for drawing conclusions on the results from the experiment. The web service was enacted many times from Triana [3] package with different input data, namely a text from the William Shakespeare's play "As you like it" with different size and the text sequence itself.

In both cases the output size and the execution time of the service were collected and used to generate SLA. The relevant historic data was represented as a cloud of points in a 2D graph (as described in Section 3.2) and the space is interpolated by the polynomial fit of the form:

$$f_{fit} = A + Bx + Cx^2 + \ldots \quad (1)$$

However in current experiments only terms as far as second degree of the polynomial were used. The resulting fit function is then used to generate a following SLA:

Input Value, I = {1…100} (user specified)
Max Execution Time, $T_{exe} = f_{fit}(I) + 0.1\, f_{fit}(I)$
Max Output Size, $D_{out} = f'_{fit}(I) + 0.2\, f'_{fit}(I)$
Price, $P = 10\, T_{exe} + 1\, D_{out}$

Note that this simplified representation of an SLA is a typical example and the ranges in the *Input Value* term as well as coefficients in the *Price* term and other SLA terms are arbitrary and depend on the specific and individual service being agreed. The main point to note is that the price is agreed on the outcome of the execution time of the service and the data generated, which in turn depends on the input data. This way both parties involved in the SLA understand the implications of the service agreed and are able to control (via specifying various terms in SLA) the outcome of this agreement and specify in more detail what constitutes a successful outcome. In this particular illustration, one of the parties uses the function obtained from the historical information in equation (1) and constructs the execution time term as a new function (in this case the new function is very simple, it is whatever is the prediction plus 10%, but the function in SLA may bare no resemblance to the historical data prediction), similarly other SLA terms agreed.

In the next section, results of these experiments will be discussed. The main questions to be answered by the experiments are:

1. Are the typical services on the Grid likely to be predictable enough to take advantage of the proposed system design? Note that even the information, which states how unpredictable is the service, is a valuable information in itself when negotiating the SLA, but the aim of this exercise is to see whether there is more opportunities here for rich and dependable SLAs.

2. The benefits to the client or enactor of such expressive SLAs are clear, more accurate description of SLA terms. But does the use of this historical data provide additional benefits, for example, in the utilisation efficiency of the resource in the context of management and scheduling of services in SLA constraint environment?

4. Results.

The experimental results from the image processing workflow example are presented in the table below:

Image	Image 1	Image 2	Image 3	Image 4	Image 5
Pixel size	63x73	63x73	63x73	63x73	63x73
File size	7Kb	11Kb	36Kb	37Kb	36Kb
Image scaled x 1	**11**	**10.4**	**10.8**	**9.8**	**10.3**
Image scaled x 2	**18.4**	**19.2**	**18.8**	**19**	**19.3**
Image scaled x 3	**32.2**	**31.8**	**32.9**	**32.9**	**32.1**
Image scaled x 5	**87**	**86**	**86**	**85**	**83**

Five image files of equal resolution but different value (both in terms of the image contained and file size used) were submitted to the workflow and the execution times in each case is measured. The experiment was repeated with these 5 images but double the resolution of the original size, triple the resolution of the original images, then four and five times the resolution of the original images. The record of the execution times (in seconds) in the table reflects the results of the experiment. These data points are shown on the graph in the Figure 4.

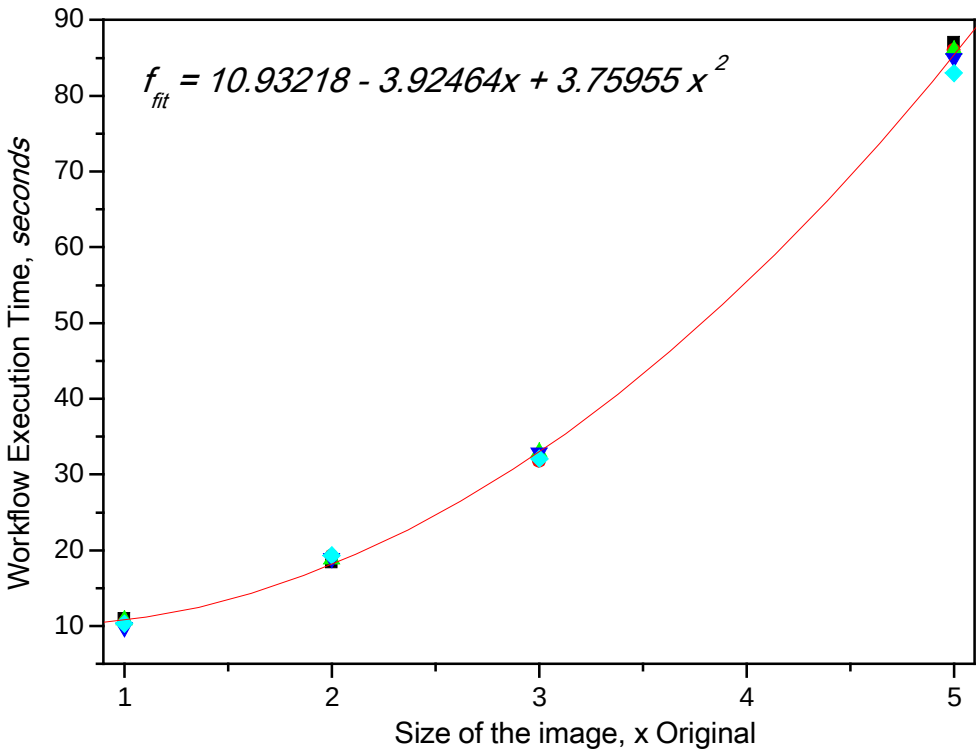

Figure 4. Execution times of the image processing workflow.

The behaviour of the workflow is found to be very predictable in the context of the dependence of its execution time on the image resolution. There is no evident correlation between the input file size and the execution time. Equally, the output size was identical (due to the specifics of the output format, which is an interesting observation when constructing SLAs). The polynomial fitting (shown as a red curve in Figure 4) produced the function shown at the top of the graph. This function was

then used in SLA as described in Section 3.3. Enacting the workflow using the images of different resolutions returned execution times close to those specified in the consequent SLAs as defined by the function. Later in this section preliminary results on the effects of more accurate predictions in SLAs, given by this experiment, will be discussed. Let us now turn to the results of the next experiment.

The Morse code converter service was enacted in a similar way, except a text of different size was used as an input. The execution times were about 3 seconds with variations comparable to the execution times. This is the example of the service where execution times are not the adequate metrics to use when describing SLA terms in complex relationships with other terms of SLA. Thus the execution time SLA term in this case was set a constant equal to the largest value found in the historical log. On the other hand the correlation between the sizes of the input and output data was very distinct as it can be seen in Figure 5.

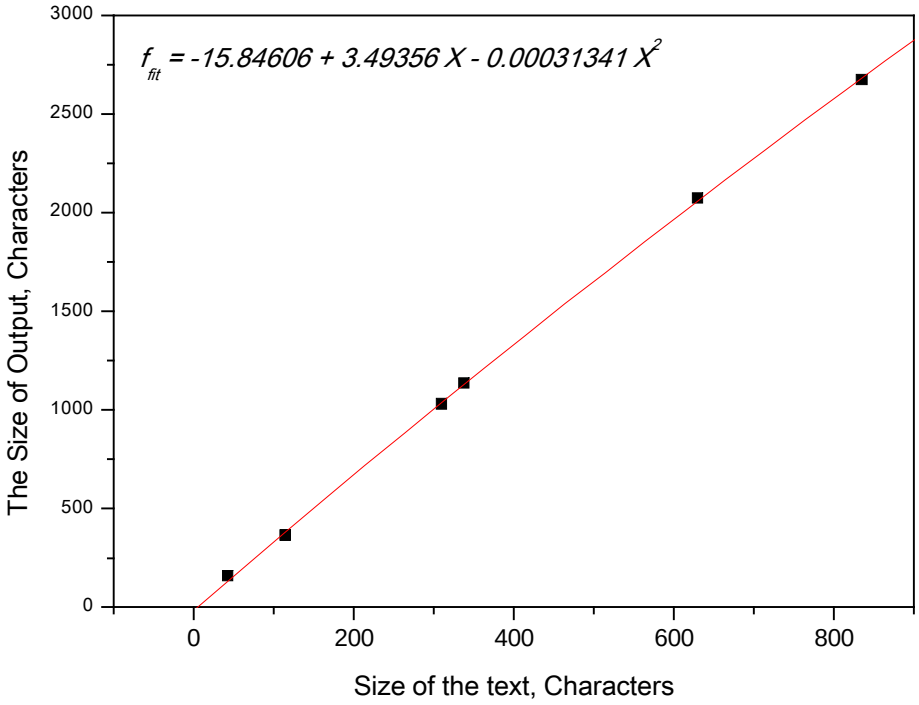

Figure 5. The measured output sizes of the text converter service.

As in the previous experiment, the SLA term *Output Size* described in Section 3.3 was defined as the function (shown at the top of the graph in Figure 5 and represented by the red curve) of another term, namely the size of the input data. The consequent calls to the service resulted in the data size being very close to that predicted by the polynomial in Figure 5. What is not seen clearly in the graph is that some data points are so close that they appear as one point. This is a very good example of SLA usage in the SLA composition scenario (Figure 2). The reason for such a close fit for the web service used lies in the two facts.

First, each single part of the input data (ASCII character limited by the alphanumeric set used in English language) is represented by a sequence of characters (Morse code's dot and dash) between 1 and 6 characters. However, on its own this fact does not guarantee the consistency observed in Figure 5, which brings us to the second point. The frequency analysis of characters used in English language reveals a histogram, which remains virtually unchanged for any English text [28] as it is

shown in the Figure 6. In fact most languages possess this feature. This means that the larger the input size the more accurate is the prediction.

The given text converter service could be a part of the larger workflow and the information about the output of former (as described in SLA) could be vital to generate SLA terms for the dependent services in this larger workflow and hence form a more successful SLA composition. Note that the data measured in the experiments above is not a result itself and still requires the full and complete execution of the agreed service, but is a framework within which the results can be expected. Similar issues were addressed in WSDL [29] and JSDL [30] specifications, only on a different level. In both description languages the input and output is expected to be of a certain type. In this document we take this practice further as well as define it in a dynamic way with help of rich SLA specification of the function based approach.

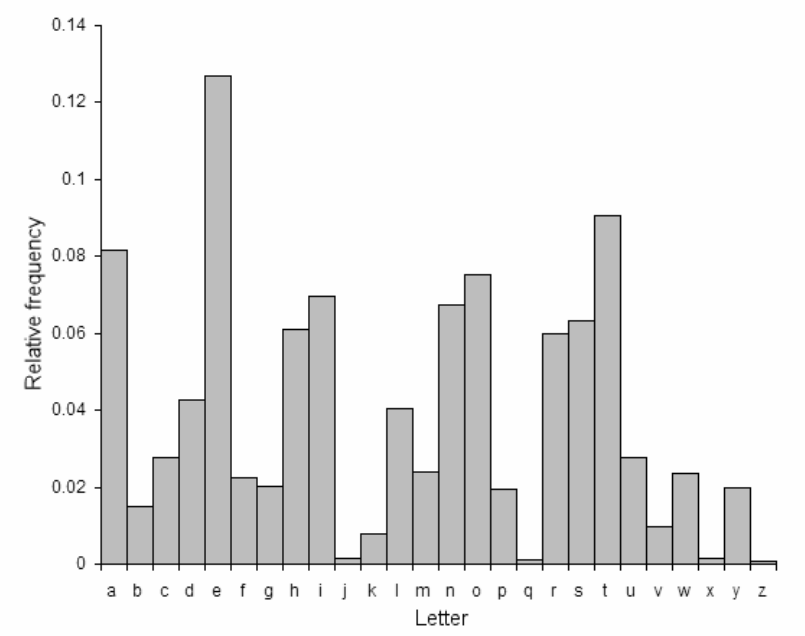

Figure 6. The histogram showing the appearances of each letter in English language.

From Figure 4 it was clear that the predictions of certain aspects of the service behaviour are not precise but a result of a statistical distribution. This distribution can be used as an error estimate or confidence in the prediction. This confidence can be used in many guises in the SLA with potential benefits to both clients and providers of the service. From the preliminary results of experiments, conducted using the SLA aware schedulers [12] and function-based SLAs together with confidence information provided by the historical data as one of the SLA terms, it was found that the utilisation of the resource (which consisted of 128 CPU nodes and available over 5 hours) could be increased by 20% when a typical SLA workload of executing services is submitted to full capacity of the resource. Without this information, the reallocation of services, bound by SLAs, could not be performed effectively, resulting in the idle gaps on the resource or failed SLAs.

The results obtained from these initial experiments show the potential of dynamic SLAs being used in conjunction with more intelligent mechanisms for SLA creation in order to build a composition of SLAs as a new value added service. The predictability of certain metrics of the service and information about other metrics along with prediction techniques could produce richer SLAs containing information necessary for successful co-allocation of a third party resources into a single workflow. Moreover, the results show the potential benefits to the service providers, giving them incentive to supply such rich information in a dynamic SLA.

5. Future work.

The investigation that was carried out during the short Research Exchange Programme with Cardiff University opened up several research directions which are both adventurous and with potential benefits to the Grid community, especially in the area of orchestration of workflows from a third party services. Most of the issues on the SLA aware workflows and their management, which are discussed in this report, are poorly covered by the community.

The discussion described in Section 2.3 initiated a debate which is still ongoing (via email and other forms of communication). It is anticipated that the more developed form of this debate along with more refined arguments for a number of extensions (mentioned in this report) will be shared at the upcoming OGF and related meetings to the wider community.

Owing to the time limitations it was impossible to device a working experiment which would build on the experiments presented in Sections 3 and 4 and investigates different aspects of SLA composition in real Grid environments. The results presented in this report show the potential for further development in this direction and is the most adventurous topic to be pursued in the future.

It is anticipated that all or at least part of the study, mentioned in the previous paragraph, will be taken further to be developed it in a full research project proposal. Certainly it is intended to submit at least one publication in collaboration with University of Cardiff (host of the REP) in the next call to IEEE International Symposium on Cluster Computing and the Grid (CCGrid) [31] or the Cloud Computing workshop, running alongside CCGrid. The work will mainly be an extended version of Section 2.1 which surveys different approaches to SLA specification.

The collaboration with University of Cardiff as a direct result of this REP and other CoreGRID members as a by-product of this REP strengthened my working relationships with colleagues from several world class research groups, among them to name but few:

Omer Rana, Professor of Performance Engineering at University of Cardiff (CoreGRID member, contributor to OGF groups)
Philipp Weider, University of Dortmund (CoreGRID member, contributor to JSDL, GRAAP-WG, chair of GSA-RG)
Alexander Papaspyrou, University of Dortmund (CoreGRID member, co-chair of OGSA-RSS, participant of JSDL, GSA, GSM)
Michael Parkin, Barcelona Supercomputing Center (CoreGRID member, GRAAP-WG)
Ramin Yahyapour, University of Dortmund (Leader of the CoreGRID Institute on Resource Management and Scheduling [32], Coordinator of OGF Compute area, participant in GSA-RG, GRAAP-WG, OGSA-RSS-WG and JSDL-WG)
Oliver Waldrich, Fraunhofer-SCAI (CoreGRID member, Contributor to GSA-RG, GRAAP-WG)
Wolfgang Ziegler, Fraunhofer-SCAI (CoreGRID member, GRAAP-WG)
Dirk Neumann, Information Systems, Albert-Ludwigs-Universität, Freiburg
Nikolay Borissov, Universität Karlsruhe

The mentioned above are the likely members in the future joint research proposals both that are being developed from the current REP and those developed separately.

Acknowledgements

This research was made possible through the European Commission's Sixth Framework Programme (FP6) and the funding of CoreGRID Network of Excellence, whose support I am please to acknowledge. I would also like to thank Omer Rana and the host institution for guidance, research input, expertise and support during the programme. I have obtained an invaluable experience, which would not be possible without this programme and the help of Prof. Omer Rana.

I would like to thank my current supervisor, Rizos Sakellariou, who was very supportive during the entire process. His research input and critique as well as help in various administrative and logistic issues allowed me to concentrate on the programme and produce this work.

Finally I would like to acknowledge the people (list of persons is given in the earlier section) for debates and discussions results of which contributed to this report.

References.

[1] Rizos Sakellariou and Viktor Yarmolenko, "On the flexibility of WS-Agreement for job submission", *Proceedings of the 3rd International Workshop on Middleware for Grid Computing MGC '05*, Grenoble, France, vol. 117, 1-6 (November 2005)

[2] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard, "Web Services Architecture", *W3C Working Group Note*, (11 February 2004)

[3] I. Taylor, M. Shields, I. Wang, and O. Rana, "Triana Applications within Grid Computing and Peer to Peer Environments", In *Journal of Grid Computing*, 1(2):199-217. Kluwer Academic Press (2003)

[4] Viktor Yarmolenko, Rizos Sakellariou, "Towards increased expressiveness in Service Level Agreements", *Concurrency and Computation: Practice and Experience*, vol.19, 1975-1990 (2007)

[5] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *International J. Supercomputer Applications*, 15(3), (2001)

[6] I. Foster *et al*, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation," in *7th InternationalWorkshop on Quality of Service(IWQoS)*, London, UK (1999)

[7] Open Grid Forum (OGF), http://www.ogf.org

[8] Grid Resource Allocation Agreement Protocol WG (GRAAP-WG), http://www.ogf.org/gf/group_info/view.php?group=graap-wg

[9] Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, Ming Xu "Web Services Agreement Specification (WS-Agreement)", OGF Document of GRAAP WG: GFD-R-P.107, http://www.ogf.org/documents/GFD.107.pdf

[10] Viktor Yarmolenko, "Dynamic WS-Agreements", *presented at GRAAP-WG, GGF17* , Tokyo, Japan, (May 10-12 2006)

[11] Viktor Yarmolenko, "Open Issues in SLA Aware Systems", *presented at WP6 CoreGRID Meeting*, Dortmund, Germany, (March 13 - 14 2008)

[12] Viktor Yarmolenko, Rizos Sakellariou, "An Evaluation of Heuristics for SLA Based Parallel Job Scheduling", *3rd High Performance Grid Computing Workshop* (in conjunction with IPDPS 2006), Rhodes, Apr. 2006, IEEE Computer Society Press.

[13] CoreGrid EU Network of Excellence, http://www.coregrid.net

[14] Michael Parkin and Dean Kuo and John Brooke, "A Framework & Negotiation Protocol for Service Contracts", SCC '06: Proceedings of the IEEE International Conference on Services Computing, p253-256, IEEE Computer Society (2006)

[15] K. Czajkowski et al, "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems," in *Job Scheduling Strategies for Parallel Processing*, 8th International Workshop, Edinburgh, U. S. E. D.G. Feitelson, L. Rudolph, Ed., 2002.

[16] WSLA Language Specification, Version 1.0, January 2003, IBM Corporation, 2003. 15 June 2007, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf
[17] Rizos Sakellariou, Viktor Yarmolenko, "Parallel Job Scheduling for HPC: the Case for Novel Forms of Scheduling", in Book *High Performance Computing and Grids in Action*, L. Grandinetti Ed., IOS Press, (2008)
[18] Load Sharing Facility (LSF), http://www.platform.com/Products/platform-lsf-family/
[19] PBS (Portable Batch System), http://www.pbsgridworks.com/
[20] Sun Grid Engine, http://www.sun.com/software/gridware/
[21] A. Sahai *et al*, "Specifying and Monitoring Guarantees in Commercial Grids through SLA," Internet Systems and Storage Laboratory, HP Laboratories Palo Alto, Tech. Rep. HPL-2002-324 (November 2002)
[22] Akhil Sahai, Akhil Sahai, Anna Durante, Anna Durante, Vijay Machiraju, Vijay Machiraju, "Towards Automated SLA Management for Web Services", Internet Systems and Storage Laboratory, HP Laboratories Palo Alto, Tech. Rep. HPL-2001-310R1 (2002)
[23] R. Brunner, I. Chao, P. Chacin, F. Freitag, L. Navarro, O. Ardaiz, L. Joita, and O. F. Rana, "Assessing a distributed market infrastructure for economics-based service selection", International Conference on Grid computing, High-Performance and Distributed Applications (GADA'07), Vilamoura, Algarve, Portugal, Nov 29 - 30, 2007. Springer Verlag.
[24] Liviu Joita, Isaac Chao, Pablo Cachin, Omer F. Rana, Felix Freitag, Leandro Navarro, and Oscar Ardaiz, "Service Level Agreements in Catallactic Oriented Grid Markets", Usage of Service Level Agreements in Grids Workshop, alongside ACM/IEEE Grid 2007, Austin, Texas, USA, (Sep 2007), CoreGrid
[25]Stephen A. Jarvis and Daniel P. Spooner and Helene N. Lim Choi Keung and Junwei Cao and Subhash Saini and Graham R. Nudd, "Performance prediction and its use in parallel and distributed computing systems", *Future Gener. Comput. Syst.*, Elsevier Science Publishers, vol.22(7), 745-754 (2006)
[26]A. D. Parks, "The Quality of Service Satisfiability Thesis: Resource and Performance Prediction for Multi-Processor Systems and Random Schedules", Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), 130, IEEE Computer Society (2005)
[27] Morse Code Translator WebService, WSDL = http://www.regomnet.de/morse.asmx?WSDL
[28] Ibrahim A. Al-Kadi, "The origins of cryptology: The Arab contributions", *Cryptologia*, 16(2), 97–126 (April 1992)
[29] Roberto Chinnici, Hugo Haas, Amelia A. Lewis, Jean-Jacques Moreau, Sanjiva Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 2", *W3C Recommendation*, (26 June 2007)
[30] Ali Anjomshoaa, Michel Drescher, Donal Fellows,An Ly, Stephen McGough, Darren Pulsipher, Andreas Savva, "Job Submission Description Language (JSDL) Specification v1.0", *OGF Document GFD.136*, (July 2008)
[31] 9th IEEE International Symposium on Cluster Computing and the Grid, May 18-21, Shanghai Jiao Tong University, Shanghai, China. http://grid.sjtu.edu.cn/ccgrid2009/
[32] Institute on Resource Management and Scheduling, http://www.coregrid.net/mambo/content/category/3/16/30/