On the Flexibility of WS-Agreement for Job Submission

Rizos Sakellariou
rizos.sakellariou@manchester.ac.uk

Viktor Yarmolenko
viktor.yarmolenko@manchester.ac.uk

School of Computer Science, The University of Manchester
Kilburn Building, Oxford Road, Manchester M13 9PL
United Kingdom

ABSTRACT

This paper considers extensions to the WS-Agreement specification, namely the Guarantee Terms of WS-Agreement [1]. Experiences and conclusions drawn are in the context of Agreement-based job management systems. A key idea of these extensions is the use of functions for the Guarantee Terms of the Agreement rather than constant values or ranges. Functions may contain variables defined in a particular agreement or be drawn from the known set of reference variables, such as wall-clock time, job start time, *etc.* We show that such an approach can potentially reduce the negotiation overheads associated with job renegotiation and/or reduce the number of failed agreements.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Client/server; C.4 [**Performance of Systems**]: Reliability, availability, serviceability; D.4.7 [**Organization and Design**]: Distributed systems

General Terms

Design, Standardization, Measurement, Performance

Keywords

Service Level Agreement, WS-Agreement, Job Submission

1. INTRODUCTION

Developments in the area of service-oriented architectures will naturally attract a range of commercial applications. An important part of this development is the increasing use of the notion of a *Service Level Agreement* (SLA) (for example, WS-Agreement (WS-A) [1]), which is essentially a contract outlining service qualities and guarantees. Although the WS-A specification covers an extensive set of scenarios in general, some weaknesses have been noticed in the area of SLA renegotiation [2, 6].

In the domain of SLA-based resource management an efficient system relies heavily on the possibility of renegotiation [7, 5, 4] of the whole or part of the agreement. Renegotiation includes reconsideration, by all participants of an agreement, of the quality and the level of service, such as, time bookings, bandwidth, processor or memory requirements, and many more. A significant proportion of agreed SLAs may have to be renegotiated in order to keep an SLA [17]. This requires the user's participation at some stage between the time when the initial agreement is made and when it is completed. This requirement might contrast with autonomous systems [16], an area in which there is significant interest for commercial development [11], where the aim is to reduce the amount of interaction with the end-user. Another related branch — value added services, that are composed of a number of single services — are based upon the assumption that a certain level of automation exists.

In this paper, we consider how extensions to the Guarantee Terms of WS-A can be used to minimise the impact of renegotiation overheads. The scope of this paper does not include negotiation protocols or what XML schema might be used, nor does it consider scenarios other than job submission and management. However, its conclusions directly affect renegotiation for SLA-based resource management and could be applied to other aspects of service-driven applications on Grid. Thus, the paper aims to address the following issues: how the network traffic associated with the job renegotiation and user involvement can be reduced; and how all this can fit within the existing WS-A framework.

The paper is structured in the following way. Starting with the motivation and related work in Section 2, it then proceeds with a description of extensions to Guarantee Terms in WS-A (Section 3). This includes a few specific examples (Sections 3.1-3.5). To support one of the examples a simple model was constructed, which is discussed in Section 4 along with simulation results obtained from the model. Concluding remarks are given in Section 5.

2. MOTIVATION AND RELATED WORK

Earlier work [17] indicated that, at high resource utilisation, renegotiation is inevitable as a means to maximise the success rate of SLAs agreed. In [8], the authors evaluate several SLA-based job management scenarios all of which use re-planning on local resources. This method seems to be widespread across the entire job management domain.

Re-planning involves changes in queue order, nodes used, booked time, *etc.* and is an essential part of job management with or without renegotiation. Renegotiation allows to

MGC'05, November 28-December 2, 2005 Grenoble, France Copyright 2005 ACM 1-59593-269-0/05/11...$5.00

change SLA constraints whilst keeping the agreement. This provides the necessary flexibility and achieves a higher resource utilisation, although the overall performance suffers as a result. In fact, the performance of the job submission system might be measured using a function inversely proportional to the number of renegotiations required to satisfy SLAs. In addition, some optimisation algorithms for scheduling perform better with *soft* constraints [14, 9, 12], meaning a large number of renegotiations is required or such algorithms cannot be used in a Grid environment.

Considering the dilemma between efficient scheduling on one hand and the renegotiation overhead on the other, we propose to extend the Guarantee Terms in WS-A aiming to reduce the need for renegotiation of a service and to provide the extra flexibility needed for a wider spectrum of resource management applications. In the draft WS-A specification [1], terms of an agreement that were negotiable were labelled as such, whilst the negotiable range remained unknown. Further extensions, addressing this issue [2, 6], were proposed in which a set of negotiable templates can be added to WS-A. These increase the overall flexibility of the job negotiation, but they still require some renegotiation to be performed. In [13], an SLA language specification was defined (WSLA), which, in our opinion, addresses some important issues mentioned in this paper; however, the future of this work is somewhat unclear.

In this paper, we elaborate on extensions to the guarantee terms of WS-A, namely the *Service Level Objective* (SLO) and the *Business Value List* (BVL). These can be applied to distributed computing on the Grid, and map onto the general WS-Agreement specification [1]. Our initial motivation was to address the problem of resource management and scheduling in an SLA aware environment. It soon became clear that the efficiency of the service could be increased by using techniques [10], which unfortunately could not be accomodated by the current state of Grid and Web Service Architectures. We believe that the suggested extensions to WS-A can facilitate the development of new services, for example, scheduling based on optimisation methods, which have succeeded in other areas.

3. EXTENSIONS TO THE TERMS

In most cases of SLA-based resource management applications that work within the WS-A framework [3, 15], the set of guarantee terms is rigidly defined, however there is no evidence that such constraints are really essential in the WS-A. Speaking in terms of WS-Agreement specification [1], the Service Level Indicators contain constant values and thus define a Service Level Objective (SLO) in a limited way.

Within such parameters, it is known in advance, for example, what the exact financial gain will be when an SLA is audited. The obvious benefit of such an arrangement is a relatively compact SLA that is simple and has small overheads. However, we find that this arrangement carries too little information for other participants of this SLA to take the initiative for some deviation without the need for renegotiation. The flexibility of an agreement can be improved by adding more terms, but there is a limit on how much overhead is practical, whereas most of the terms can easily be described analytically.

We believe, therefore, that in most cases a slightly increased overhead in the SLA can be justified if the number of renegotiations or failed SLAs can be reduced. The latter can be achieved by providing an infinitely large set of term configurations in SLA. The extra SLA feeds can provide complex relationships between guarantee terms (SLO, BVL, *etc.*) that could potentially be used by autonomous applications in providing qualitatively new levels of service. In other words, we propose to extend the SLA specification, which describes the service, with higher degrees of description. These can be described as follows:

(1) A list of *universal* variables is introduced. These can be either constants or static functions, i.e. invariant to the environment. They are predefined elsewhere and simply referred to in the SLA. Among such variables/functions are: current Wall Clock Time, current network bandwidth, *etc.* (the relevance of these will be explained shortly).

(2) A list of predefined common functions is introduced. For example an average value of a list, a Gaussian or other function that defines min/max bounds, *etc.*

(3) Service Level Indicators (SLIs) are described not as constants but as functions of *universal* variables, or other SLIs. These functions can be defined within an SLA or can be one of the predefined common functions.

Therefore, the Guarantee Terms of WS-A are no longer described as a single point or flat limits (formed by the SLI range-values independent of each other) in SLA space, but become non-trivial multidimensional volumes defined by a system of functions. Such an agreement has an infinitely large number of outcomes that belong to the continuum defined by the system of functions. When such a system of functions, which describes the entire guarantee terms of an SLA, is chosen correctly it can potentially provide a far richer term set for job management applications and hence reduce the need for renegotiation. Next, we give several examples of the extensions discussed in this section.

3.1 Universal Variables and Functions

Earlier, we expressed the need for *universal* variables, the actual values of which may not be known at the time when an SLA is formed. Below is an example of such *universal* variables, which an extended WS-A could include:

(1) Current Time - returns the global wall clock time.

(2) Current Resource Load - returns a value {0...1} that indicates the current resource load at the time of usage. In the simple case, this value is formed as the ratio of non-idle CPU nodes of the resource to its total number.

(3) Current Actual Bandwidth - returns the actual bandwidth capability at the time of usage that was allocated to satisfy a specific SLA. This may depend on the time of the day, and other factors.

(4) Actual Data Traffic - returns the amount of data that was transferred as a result of the client's job.

(5) Actual Disk Usage - returns the amount of disk space used by the client's job.

(6) Actual Maximum Memory Used - returns the highest level of memory occupied by the client's job.

(7) Actual Execution Time - returns the time it took for the client's job to be executed.

(8) Actual Job Start time - returns actual global wall clock time at which the client's job begun its execution.

Below we give a list of common functions. The criteria that determine which functions should be included in the list must be based on (1) how elaborate the function is in its description, and (2) how common this function is.

$f_{norm}(x, low, high)$ - a binary function which returns 1 in the region $low < x < high$ and zero for other values of x.

$f_{inv}(x, low, high)$ - a binary function which returns 0 in the region $low < x < high$ and 1 for other values of x.

$f_{tr}(x, low, \alpha, high, \beta)$ - a function described in Figure 1(a).

Note that lists for *universal* variables and common functions are not limited to those elements suggested in this paper. Also, it is worth re-emphasising that the *universal* variables and common functions are predefined and as such are available offline, thus there is no need for their descriptions to appear in SLA, but only their references. This constitutes the first part of the extensions suggested to WS-A.

3.2 Guarantee Terms as Functions

In the previous section, we described some of the *universal* variables and functions as WS-A extensions that can be used when defining a guarantee term. Guarantee terms, in turn, may be used in other parts of the agreement. In this paper, we only concentrate on guarantee terms, such as, SLOs and BVLs.

Furthermore, we present three specific examples of how and why guarantee terms such as number of CPU nodes, memory or time slot required for the successful execution of a job, can and should be able to be expressed as functions and not as constants, renegotiable constants, or ranges. Similarly, BVL terms such as price, penalty for the service or importance of an SLO can and should also be able to be expressed as functions of any guarantee term(s) as well as *universal* variables and common functions. This constitutes the second and main part of the extensions suggested to WS-A.

3.3 Example with Reward Term as a Function

Let us now consider a simple example of how a term from BVL (call it Integrated Reward) can be described as a function of SLOs, other BVLs and *universal* variables, *etc.* First we define a few guarantee terms for the WS-A:

SLO: Earliest job start time, T_S
SLO: Latest job finish time, T_F
SLO: Number of CPU Nodes required, N_{CPU}
SLO: Reserved time for job execution, t_D
BVL: Penalty limit for non-compliance, V_{pn}^{max}
BVL: Price limit for compliance, V_{pr}^{max}
BVL: Integrated Reward, V_{tot}

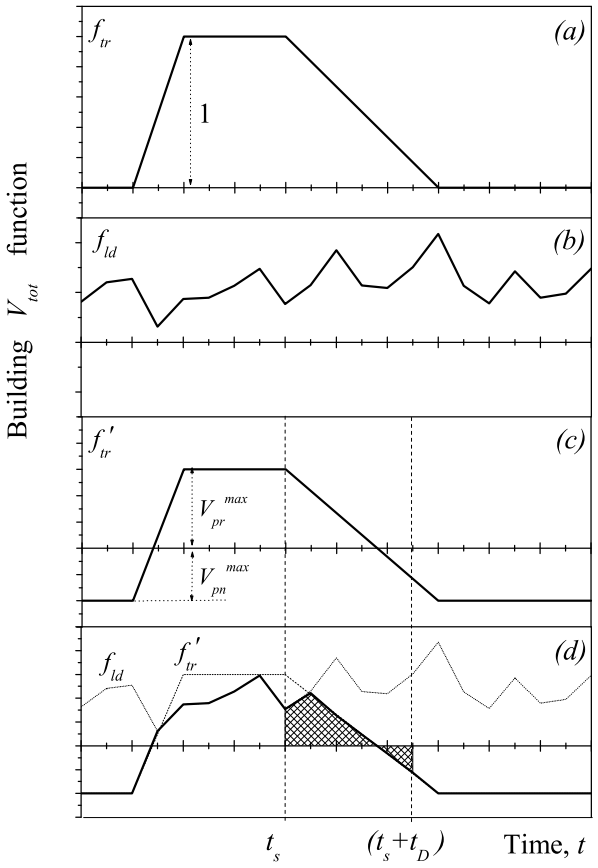

Figure 1: An example of Integrated Reward, V_{tot}, expressed as a function (Equation 2). (a) - f_{tr} function (Section 3.1); (b) the Resource Load function (Section 3.1(2)); (c) - a version of (a) that defines values for price/penalty (Equation (1)); (d) - same as (c) but takes into account the price variation depending on (b).

Note that we omit other fields of the WS-A template, such as Context, Name and all the tags associated with them and only concentrate on a few guarantee terms for simplicity.

The Integrated Reward, V_{tot}, may be unknown at the time when an agreement was formed and is calculated after the WS-A is finished. What is known, however, is the limits within which this business value can fluctuate and how.

Let us, thus, build this function from the elements mentioned in Section 3.1 starting with softening the time constraints (T_S and T_F) with f_{tr}, shown in Figure 1(a). Now we introduce the maximum price and penalty factors, $V_{pr}^{max} = (c_1 - c_2)$ and $V_{pn}^{max} = c_2$ respectively, into the equation, so that:

$$f'_{tr} = \left(f_{tr}(t, T_S, \alpha, T_F, \beta) \cdot c_1 - c_2\right) \qquad (1)$$

where c_1 and c_2 are constants in mutually agreed reward units per time unit (e.g. \$, £, pigs per hour, *etc.*) and α, β are the parameters of f_{tr} that determine the angles of the trapezium (Figure 1(c)). The area of the function below zero (as low as V_{pn}^{max}) will contribute to the penalty whereas that above zero (as high as V_{pr}^{max}) contributes to the price.

At this stage, we would like to point out the time limits of V_{tot}, namely the start time of the job, t_s and its duration t_D (Figure 1). For simplicity, we assume that the time reserved for the job, t_D is equal to the time it took for the job to complete, however one could introduce a relationship between these SLOs, which might influence the final outcome.

All mentioned SLA terms, except t_s and t_D, are SLOs that are usually defined by the client. Let us impose a constraint on V_{tot}, which is defined by the service. For example, a function that represents the resource load, $f_{ld}(t)$ (Section 3.1(2) and Figure 1(b)). The provider may want to vary the amount of the reward generated from the service depending on the demand of this service. Naturally, $f_{ld}(t)$ is unknown when the agreement is created.

We build V_{tot} so that only the area under both functions, $f_{ld}(t)$ and f'_{tr} contributes to the outcome (Figure 1(d)):

$$V_{tot} = \begin{cases} \int_{t_s}^{t_s+t_D} f'_{tr}(t, T_S, \alpha, T_F, \beta, V_{pn}^{max}, V_{pr}^{max})\ \mathrm{d}t \\ \int_{t_s}^{t_s+t_D} c_3 \cdot f_{ld}(t)\ \mathrm{d}t \end{cases} \quad (2)$$

The shaded area represents total reward, V_{tot}, that a client has agreed to pay. In the case when the job started later than t_s on the figure, this value decreases up to the point where the negative term outweighs the positive term under the integral and the service provider ends up paying some penalty, but not higher than that limited by V_{pn}^{max}.

Thus, we defined a guarantee term in the agreement which is based on functions and *universal* variables whose values could not be available at the time when SLA was formed. Also, in this example, we were able to agree on an infinitely large number of outcomes for V_{tot}. Now the service provider can re-plan its activities without the need for renegotiation.

Similarly, other business values and SLOs can be described as functions or a system of functions. For example a client's application can start with different memory or CPU requirements, which may affect the execution time, with the relationship defined in SLO:t_D. Alternatively, a service provider may want to encourage clients to use wider time bounds, so that business values depend on time window $(T_F - T_S)$.

3.4 Example with variable number of CPUs

Another simple example of usage of the extended SLA would be useful for applications that can vary the number of CPU nodes required at a start of the computation (e.g., MPI based parallel applications). The extended SLA would look similar to the standard SLA except two of its SLOs:

SLO: CPU Nodes required, $N_{CPU} = \{2, 3, 4, ..., N_{CPU}^{max}\}$,
SLO: Reserved time for job execution, $t_D = \frac{t_D^o}{N_{CPU}}$,

where t_D^o is some constant and N_{CPU}^{max} is limited by the capacity of the Resource. The relation can be more complex than that. Compared to the previous example, this description of SLOs can only produce a limited number of options. However, even in cases like this, a relatively small function could describe something in a more laconic fashion than a list of paired values (N_{CPU} and t_D), which, in the simplest case, can reach the list of $(N_{CPU}^{max} - 1)$ items in size (option with $N_{CPU} = 1$ is not considered). The number of possible renegotiations even for such a limited set is unacceptably high. If the Resource's scheduling algorithm needs to renegotiate every single $\{N_{CPU}, t_D\}$-paired option for each job submitted in order to increase its utilisation then, potentially, $(N_{CPU}^{max} - 1)^{N_{Job}}$ renegotiations would be required (assuming that responses are cached by the Resource).

Because of its relevance and simplicity, we ran an experiment (Section 4), which quantitatively measured benefits of the extended SLA in which N_{CPU} and t_D correlate. Results show over 10% increase in resource utilisation at about 95% resource load. Summarising this example it is worth adding that a user may want to add variable importance as to what number of CPU Nodes to use, adding extra correlation to the Business Value List and combining therefore this example and the example described in Section 3.3.

3.5 Example with variable Bandwidth

The next example considers the communication speed between the working nodes. For example, for message passing intense applications a 32-processor machine is likely to perform better than a Grid cluster, which aggregates 32 individual nodes of comparable power spread across the world. As in the previous examples, let us define the relevant SLOs:

Universal Value: Bandwidth between the nodes, B,
SLO: Reserved time for job execution, $t_D = \frac{t_D^o}{B \cdot N_{CPU}}$.

Again, the relation for t_D can be more complex than that. The more detailed the description, the better service, in general, can be achieved. Combining this example with the examples described in 3.4 and 3.3 increases the vector of options available to the Resource, making value added autonomous services more feasible.

4. EXPERIMENTS AND RESULTS

This experiment refers to the example described in Section 3.4. We choose two metrics: the average job rejection rate and the average number of renegotiations per job request. These metrics will be used to evaluate the negotiation efficiency of the system that uses extended WS-A compared to that of normal WS-A, both with respect to the job load. The model description and relevant details follow.

Two parties are involved in the SLA, the User and the Resource. The Resource schedules jobs using a single iteration Earliest T_F-Deadline First algorithm. Its availability is 64 working nodes over 147 virtual hours. The User always generates 340 job requests. At least one configuration for the set of all 340 jobs exists, by which jobs can be scheduled on the Resource with 100% utilisation, so that 100% of SLAs are satisfied with zero idle resources during the availability period.

When the submission is successful, an SLA is formed with the SLOs: T_S, T_F, N_{CPU} and t_D. Other guarantee terms (including BVL) are omitted in this experiment for simplicity. Upon its creation, each SLA describes t_D as a function of N_{CPU}, so that the product of these two, A, is always constant, $t_D = \frac{A}{N_{CPU}}$. A, varies between SLAs with an average value of 21.85 for the set of 340 job requests. Only the cases

with the whole values of N_{CPU} and t_D are considered. For example, for $A = 24$ the relation can produce only 7 paired options (option: $N_{CPU} = 1$ is not considered). Depending on the value of A, the number of options vary from SLA to SLA with the average number of options being below 5.

In the first scenario, the User attempts to submit a job, constrained by the criteria mentioned earlier, one by one to the Resource. If the Resource is able to fit the job, then an SLA is formed, in which all SLOs are constants. N_{CPU} is always chosen to be the highest possible for the particular job, but not higher than the Resource size (64). If the Resource refuses to accept the job, it is completely discarded. This scenario represents job submission using a normal SLA.

The second scenario differs from the former in that it uses a function to describe t_D, in which only whole multiplicands are allowed with limitations on $N_{CPU} = \{2, 3, ...64\}$. The initial configuration in this case also starts with the highest possible value of N_{CPU} for the job. However, if the Resource cannot accept the job with these parameters it simply tries another configuration from the SLA. This scenario represents a job submission using an SLA with SLOs defined as functions, but in this specific example there is only one SLO (t_D) that is described in such a way, and the number of possible options generated from this is very limited. In real life, t_D is more fine-grained than that of the simulation, resulting in more options for the relevant SLO (see Section 3.4).

In Figure 2, we present two curves that were obtained using the scenarios mentioned. The values are averaged over 5,000 experiments. As we can see from the figure, the scheduling algorithm begins to struggle with a density of jobs higher than 90% rejecting some of the requests as a result. In the case when the SLA describes terms as fixed values, the Resource rejects a higher number of jobs, on average, than that of the extended SLA. The difference is 11.66% at its widest point of 95.88% of submitted jobs. It is anticipated that the difference in the rejection rate would increase as the number of configurations per SLO continues to increase.

Let us consider a third scenario in which the Resource renegotiates an SLA before cancelling the job. Each renegotiation produces another set of SLOs. The job set is the same as in the second scenario. The only difference is that the Resource must contact the User each time the next pair of SLOs is used.

Figure 3 shows the dependence of the average number of renegotiations per job on the percentage of submitted jobs. The number of renegotiations increases as more jobs are submitted to a highly loaded resource. This value remains at 1 at any resource load, when the extended SLA is used, because the Resource has all the information in the first instance and does not require additional renegotiations. The number of renegotiations levels off at the value of the average number of configurations in SLA, which was anticipated. As the Resource struggles more to fit a job it runs through all possible configurations before giving up. It is obvious that the number of renegotiations, limited in this simulation, may potentially reach large numbers, rendering the entire renegotiation concept useless. In real life, the number of options can progress from a limited set to infinity for each guarantee

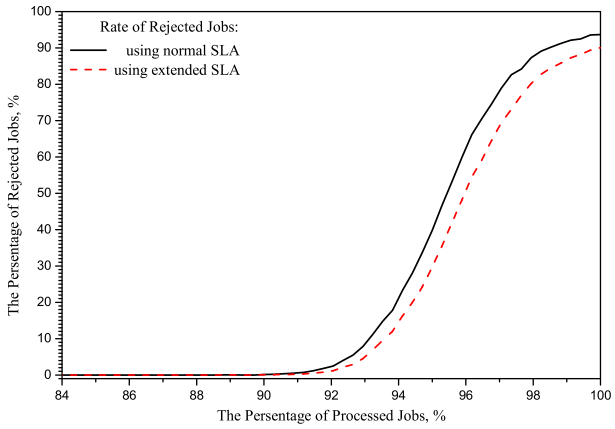

Figure 2: The rate of cancelled job requests depending on the percentage of jobs submitted. The job submission using extended SLAs (dashed line) produced less rejected jobs at high resource load, compared to the normal SLA scenario (solid line).

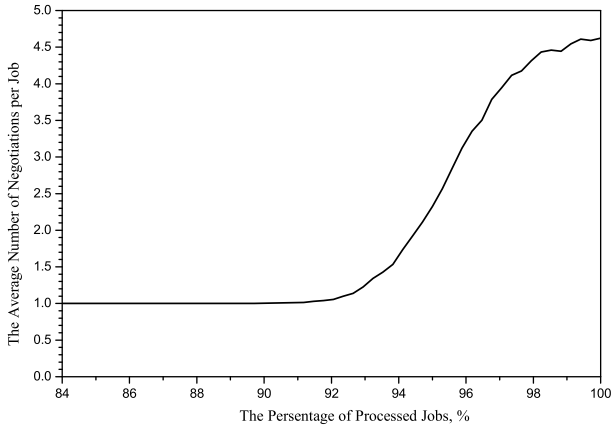

Figure 3: The rate of negotiation attempts depending on the percentage of jobs submitted.

term, as well as more guarantee terms can be represented as analytical functions of other guarantee terms. All this could potentially increase the overall efficiency if it was not for a dilemma associated with renegotiation expenses. Currently, this issue is addressed by making a compromise; in this, only a limited number of renegotiations is allowed before it becomes too expensive. This approach automatically cuts off a growing number of more sophisticated scheduling techniques which could be adopted to the problem of job management and scheduling on grids.

5. CONCLUSIONS

We argued the case in which suggested extensions to the standard WS-Agreement were discussed. These include introduction of *universal* variables and common functions and extending guarantee terms to be described as analytical functions of *universal* variables, common functions and/or other guarantee terms. We showed, using a simple experiment, that even a little flexibility in the Agreement, defined as the function of only one guarantee term, t_D, improves the over-

all system performance by over 10%. The situation is bound to be improved when the number of configuration options in the SLA approaches infinity, something that it would be possible to define with the extended WS-A proposed in this paper.

We believe that such a form of WS-Agreement not only does it produce visible benefits to the current job management models, but it may also encourage new service models to emerge, especially in the area of value added and autonomous services. Summarising this section we outline the following benefits offered by the extended WS-Agreement proposed in the paper:

(1) Reduction in the network traffic associated with the negotiation of a service.

(2) Reduction in the user-service interaction, due to increase of the autonomy of the process, which consequently improves the quality of the service (at least for the job submission).

(3) The extended WS-Agreement, allows both new optimisation algorithms and those used for other problems to be applied in job scheduling and service management in general.

(4) The Extended WS-Agreement could support a larger variety of services, especially commercial services geared toward automation and value added approach.

(5) The extensions fit within the existing WS-A framework.

Acknowledgement:
This research has been partially funded by EPSRC (grant reference GR/S67654/01) whose support we are pleased to acknowledge.

6. REFERENCES

[1] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Agreement-based Grid Service Management (OGSI-Agreement). IBM Corporation, 2003.

[2] A. Andrieux, A. Dan, K. Keahy, H. Ludwig, and J. Rofrano. Negotiability Constraints in WS-Agreement. In *Version 0.1, Document to GRAAP-WG Meeting*, Argonne, IL, 26-27 Jan, 2004.

[3] J. Austin, M. Fletcher, and T. Jackson. Distributed Aero-Engine Condition Monitoring and Diagnosis on the GRID: DAME. In *COMADEM*, Cambridge, UK, 23-24 Aug, 2004.

[4] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu. Agreement-based Grid Service Management (OGSI-Agreement). In *Global Grid Forum, GRAAP-WG Author Contribution*, 12 Jun, 2003.

[5] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In *JSSPP*, pages 153–183, 2002.

[6] A. Dan, K. Keahy, H. Ludwig, and J. Rofrano. Guarantee Terms in WS-Agreement. In *Version 0.1, Document to GRAAP-WG Meeting*, Argonne, IL, 26-27 Jan, 2004.

[7] M. D'Arienzo, A. Pescapè, S. P. Romano, and G. Ventre. The service level agreement manager: control and management of phone channel bandwidth over Premium IP networks. In *ICCC '02: Proceedings of the 15th international conference on Computer communication*, pages 421–432, Washington, DC, USA, 2002. International Council for Computer Communication.

[8] C. Dumitrescu and I. Foster. GRUBER: A Grid Resource SLA-based Broker. In *EuroPar 2005*, Lisbon, Portugal, 2005.

[9] K. Ecker, D. W. Juedes, L. R. Welch, D. M. Chelberg, C. Bruggeman, F. Drews, D. Fleeman, D. Parrott, and B. Pfarr. An Optimization Framework for Dynamic, Distributed Real-Time Systems. In *IPDPS*, page 111, 2003.

[10] R. Horst and P. M. Pardalos. *Handbook of Global Optimization*. Kluwer, Dordrecht, 1995.

[11] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. *Computer*, 36(1):41–50, 2003.

[12] C. Lu, J. A. Stankovic, S. H. Son, and G. Tao. Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms. *Real-Time Systems*, 23(1-2):85–126, 2002.

[13] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. WSLA Language Specification. IBM Corporation, 2003.

[14] L. Markov. Two Stage Optimization of Job Scheduling and Assignment in Heterogeneous Compute Farms. In *FTDCS*, pages 119–124, 2004.

[15] D. Mobach, B. Overeinder, and F. Brazier. A Resource Negotiation Infrastructure for Self-Managing Applications. In *Proceedings of the 2nd IEEE International Conference on Autonomic Computing (ICAC 2005)*, Seatle, WA, 2005.

[16] R. Sterritt and M. G. Hinchey. Why Computer-Based Systems Should Be Autonomic. In *ECBS*, pages 406–412, 2005.

[17] V. Yarmolenko, R. Sakellariou, D. Ouelhadj, and J. M. Garibaldi. SLA Based Job Scheduling: A Case Study on Policies for Negotiation with Resources. In *AHM2005*, Nottingham, UK, 20-22 Sep, 2005.