IBM Research Report

Architectural Issues for Metering and Accounting of Grid Services

Vikas Agarwal, Neeran Karnik, Arun Kumar

IBM Research Division
IBM India Research Lab
Block I, I.I.T. Campus, Hauz Khas
New Delhi - 110016. India.

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

Architectural Issues for Metering and Accounting of Grid Services

Vikas Agarwal, Neeran Karnik, Arun Kumar*
IBM India Research Laboratory,
Block-1, IIT, Hauz Khas,
New Delhi-110016, INDIA

Abstract

Grid Computing has been explored over the years in the academia and has recently gained the interest of the industry. Applying grid concepts to the commercial world introduces several issues that have not yet been considered seriously. An infrastructure for enabling metering and accounting of grid resource usage is one such essential requirement to make commercial grid computing a success. This paper explores various issues and describes alternative architectures that can be applied in the context of recently proposed Open Grid Services Architecture (OGSA). The proposed architectures allow grid services to retain the control over the accounting function and the business models used. Enforcement of accounting-based authorization, taking into account certain business models, are also discussed. This paper does not address the issues related to pricing schemes that the grid services may adopt for charging their customers.

1 Introduction

This paper explores alternative metering and accounting architectures that could be used in the context of commercial computational Grids especially for the recently introduced Open Grid Services Architecture [5].

Existing literature has identified the need to have an economic model that drives the allocation of resources in computational Grids [1, 2]. Authors in [8] explain that the term *utility* implies not only shared transportation and plug−n−play user experience but also the fact that it is a traded commodity [8] and therefore, markets not policies set commercial prices. For utilities such as a compute utility, an approach based upon an economic model is favored over a simple 'passive' logging like model [1, 2, 8]. An economic model requires for its application, a monetary or credit unit [1]. This paper studies architectural issues for enabling metering and accounting functionality independent of the actual pricing models used.

- An *elementary grid service* makes use of hosts' basic resources alone, such as storage, network access, database access, plotting, printing etc.
- A *non-elementary grid service* is one that makes use of other grid services. Thus, it could also be called as a *composite grid service*.

A non-elementary grid service therefore represents a tree of grid services, whose nodes are non-elementary services and the leaves are elementary services. (Even if it is a graph, it can still be treated as a tree if each invocation can be considered independent of the previous invocation, which is probably the case here).

From accounting point of view, elementary grid services differ from composite grid services in terms of monitoring and metering. An elementary grid service uses basic system resources. Therefore, a monitor for an elementary grid service would measure the usage of those basic resources. The corresponding meter for the service would use this monitoring information alone. On the other hand, a composite service is metered in terms of the underlying service(s) that it uses, the value-add

*All correspondence should be addressed to this author at kkarun@in.ibm.com

that it provides and the basic resources that it itself consumes (this last element may actually be merged into the cost of the value-add that a composite service provides, but the basic resource usage may have to be monitored anyway to keep track of available capacity). Each component grid service of a composite service treats its invoker, instead of the end-user, as its customer. It is this *customer grid service* who is liable to pay the underlying service for the usage. The top level grid service of a composite tree is the one who accumulates the total consumption and charges the end-user.

Each service decides for itself, the price/charge for the service that it provides. For instance, the pricing may be demand driven or follow a fixed-price model.

The literature cited above presents different approaches to enable commoditization of Grid resources. This document aims at defining the architecture that would enable the realization of such mechanisms. Section 2 discusses some design requirements of metering and accounting infrastructure in the context of grid services. Section 3 considers alternative models for the accounting infrastructure and talks about the metering and accounting mechanisms in the context of those models, Section 4 discusses about enforcement of accounting policies based upon the business model used. Finally, we conclude in Section 5 and mention our future course of action.

2 Design Requirements

This section briefly talks about some of the design considerations specific to the Grid environment, that need to be kept in mind while designing a metering and accounting architecture.

- **Site/Service Autonomy :** Generation, collection and reporting of metering information is controlled by the respective grid service. In other words, either the grid service itself generates the metering information or delegates the task to some external component from which it may collect that information. The authorized users of that information are also specified by the grid service. Similarly, the grid service has control over the pricing policy of the service that it offers.

- **Standardized metering :** The metering component of all grid services have to have standard query and reporting interfaces. Therefore, it is imperative that the metrics, reported by the grid service *meters*, are common across all services. This implies that service-specific usage metering has to be reported in terms of standardized units.

- **Composite services :** Each grid-service may be composed of other grid services. The composition tree of the grid-service may not be visible to the user of the grid service.

- **Job prediction :** Each grid service may use multiple resources and the usage of those resources may vary from request to request. This implies that the usage requirement of a user-request may not be predictable.

- **Pricing :** The price of the service may be static or dynamic. The pricing policy should also be replaceable, by the service, at runtime.

- **Scalability :** The components of the metering and accounting infrastructure should be scalable with the size of the grid. Moreover, since accounting related information would be needed by the authorization module to allow/deny access to the service, bottlenecks in the accounting infrastructure may affect the QoS of the grid service.

- **Security :** The metering/monitoring/accounting data will have to be stored, communicated and used in a secure way to avoid tampering from malicious users.

- **Efficiency :** The presence of the metering and accounting infrastructure should not introduce significant, unnecessary overheads on usage of basic infrastructural resources. In other words, it should not significantly degrade the performance of the grid-service itself.

- **Extensibility :** The metering and accounting components should not be specific to business models used. They should be extensible enough to incorporate newer business models.

- **Distributed enforcement :** For enforcement of accounting policies, each grid service should have the mechanism to allow/deny access on the basis of requester's accounting information. Since a grid service may be a composite one, the accounting infrastructure would need to be able to enforce the accounting policies across all component grid services.

3 Architectural Considerations

In our view, the accounting infrastructure for grid services could be visualized as a composition of three essential elements:-

- **Monitoring :** The monitoring component is instrumental in collecting and reporting the raw information related to usage. The entity being used could be a basic resource (such as storage, memory etc.) or another service or an entity like time-spent. The monitoring function could either be embedded into each service or it could be accomplished through the use of external monitoring agents. The former approach fixes the scope and definition of monitoring once the service has been defined and implemented (assuming that the service is non-modifiable at runtime). The latter approach enables the choice of monitoring metrics to be independent of the service itself though they may be derived from the information made available by the service. The advantage of external monitoring is that the monitoring functionality is not fixed and can be improved over time. The other approach, however, has the benefit of doing service specific monitoring but the onus of providing monitoring functionality lies with the service in this case.

  Monitoring infrastructure of any system may be used for multiple purposes, i.e. for resource management, fault-tolerance, usage metering etc. Any monitoring system complying with the monitoring interfaces of our architecture can be used.

- **Metering :** The metering component aggregates the raw monitoring information for the currently active user-requests and converts it into units of one or more accounting metrics. It is imperative that all the services use the same accounting metric(s) which could be *grid-unit* (a monetary unit used to trade resources within the grid). This transformation may involve an accounting function that takes the basic resource consumption, the value-add of the service and the consumption of underlying services, as input and returns value(s) of accounting metric(s) as output. The raw monitoring information is supplied by the monitoring component.

- **Accounting :** The accounting component maintains the business relationship between the grid service and its users. The business relationship describes the business model used and the quota allocated to the user. The metering module reports the accounting metrics for each service request for different users and services. The accounting component then accumulates these metrics and uses it to reduce the quota, of the respective user, by an appropriate amount. The accounting component also maintains a log of the metering information for later use such as auditing purposes. Since a grid service can be a user of another grid service, we assume that the accounting component maintains a business relationship for each such interaction, i.e. between an end-user and a grid service or between two grid services.

  Also, since the accounting component maintains many-to-many relationships (user to service business relationships), it is reasonable to model it as a service. Multiple accounting services can co-exist in the grid and each grid service can subscribe to any one of them. Metering component of a grid service would need to be aware of the corresponding accounting service. For a commercial grid most service invocations would consult the accounting service as the requests would need to be authorized from accounting point of view.

  Though the accounting service maintains the business relationships for specific user-service interactions, there needs to be a place to keep track of the global accounts of various users and services. Therefore, a *grid-bank* is required to maintain these grid-wide accounts. Establishment and/or termination of a business relationship results in updation (debit/credit) of the respective user and grid-service accounts.

The monitoring of usage and collection of raw monitoring information is, in most cases, service specific and each service may choose to adopt the mechanism best suited for it (i.e. external monitoring agents or internal service monitoring). Basic accounting functionality on the other hand is

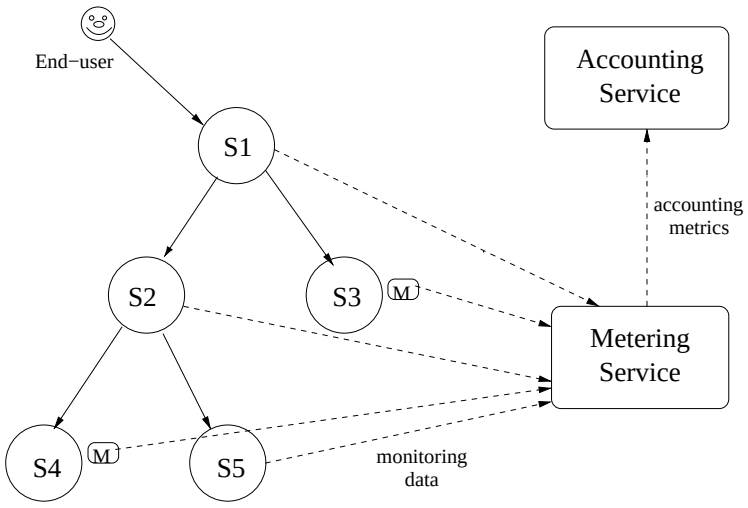

Figure 1: Model 1

a system-wide, service independent concept and can be modeled irrespective of the service architecture. Metering is the link between the two that transforms service specific usage information to service independent accounting metrics. This component is generally influenced by the service architecture. Therefore, metering component has to be standardized across all services and thus should be modeled keeping that in mind.

For Open Grid Services Architecture, two different models can be considered for the metering component:-

3.1 Model 1: External, Centralized Metering Service

In this model (refer fig 1), a metering service which could possibly be another grid service or a component of the grid middleware, provides the functionality of a meter. Each grid service registers with the metering service at deployment time. For every request, the grid service sends an initiation request to the metering service. This request includes a service generated correlation id, and other contextual information. The functions of such a metering service could include some or all of the following tasks:

- Collecting and storing raw monitoring information. The raw information to be maintained (and probably the format) is dictated by the service.
- Maintaining these values on a per service per user basis to be able to provide per user usage. For a composite service, this means that the metering service has to collect the usage of all the underlying services in the composite tree and maintain a correlation amongst service invocations that have resulted from the same end-user request. This would be necessary to aggregate the total usage of the composite service.
- Obtaining the Usage model (i.e., in what metrics is the usage of resources reported) and Accounting model (i.e., the function that determines how the service usage is computed, from the reported resource usage metrics, in terms of accounting metrics such as grid-units) of each of the component grid services in the composition tree for use in computing the final usage.
- Periodically or after completion of every request, compute the corresponding accounting metrics such as grid-units spent.
- Reporting the computed accounting metrics to the respective accounting services of the services involved in the composition tree.

This model has certain advantages such as an external monitor supplied by a third party or the grid middleware can take the responsibility of reporting the metrics to the metering service. Moreover, the service need not concern itself with accounting related functions. Since, the metering service needs the usage and accounting models of the services involved, these may actually be exported as part of the service description. Also, since all the monitoring information comes to the metering service, it is in a position to log all that usage information for later use such as reporting or for statistical purposes. Further, an existing service need not be changed to adapt to this model. This model is generic enough to be used with different grid architectures, as it does not assume anything about the grid middleware.

Architectural Issues

- For a composite grid service, the metering service has to correlate the metering information, corresponding to an end user request, for all the underlying services. Moreover, this correlation has to be done for each grid service, so that an underlying service of a composite tree can charge its calling service for its usage. Correlating the metering information would require the use of a correlation id. Each request results in generation of a new correlation id by the grid service. For composite grid services, where a user request may spawn into multiple requests to underlying services, through iterations, several new ids may get generated. Generating, maintaining and communicating so many ids to the metering service is an unwanted overhead and adds complexity.

- The metering service may not know when the underlying service has completed its work. In other words, it does not know how long to wait for receiving the monitoring information from component services of a composed service. One mechanism to do this is to have methods to indicate start and end of metering a request. Then each component service of a composite tree would send metering completion message once its local job is done. These metering completion messages would help in determining when the underlying services have finished their tasks and the aggregate of the request can be computed.

- All the contextual information related to a particular request has to be communicated to the the metering service. Such contextual information could include an upper limit on the resource usage of that particular request. For long running jobs, an interval at which to report the intermediate usage would need to be specified in order to ensure that the user's quota is not getting exceeded. This also implicitly assumes that the metering service implements a condition evaluation engine and a notification module to inform the corresponding grid service if the request exceeds its upper limit.

- The current model assumes that there is a centralized metering service which may not be desirable from scalability point of view. Therefore, a variant of the current model may allow multiple metering services. Each individual grid service would have the choice to register with any metering service, i.e. one or more grid services may share a common metering service. But, this model brings up additional issues related to aggregation of the accounting information for a composite request. The calling grid service would have to inform its metering service about the metering services of the underlying grid services. Since only authorized users are allowed to access the metering information, the calling grid service would have to send its metering service's authenticaton information to its underlying service. This information would have to be sent as the meta information of the request.

- Since monitoring information of each service is reported to the metering service there may be performance issues arising out of numerous messages being generated.

3.2 Model 2: Internal Metering

In this model (refer fig. 2), the collection of monitoring information, metering and reporting of the accounting values is built into the grid-service itself. The metering component can be implemented either as part of the grid service or as a separate component plugged into the service. The accounting information flows from a service to its user (which could be the end-user or another service). In

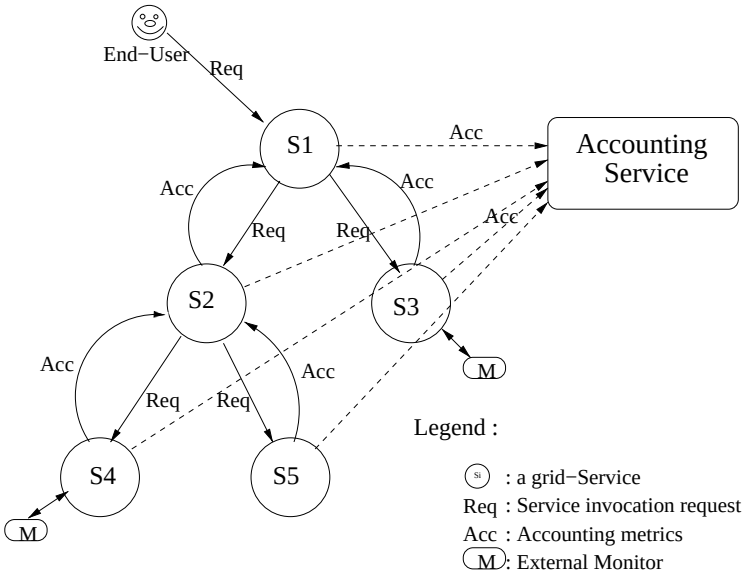

Figure 2: Model 2

addition, this information is also sent to the accounting service so that the calling service can be appropriately charged.

Each request's context contains metering related information like request id, maximum permissible limit of resource usage, etc. The context is either sent as part of the meta information of the request or as a parameter in the method call. Moreover, the grid service may also implement an interface for allowing querying of metering information by authorized users. The accounting metrics are supplied to the calling service either through notification or by piggybacking it on the reponse.

The advantages of this model are that the service need not expose its usage and accounting model. The accounting model (which may incorporate pricing in terms of grid-units) is contained in the service itself and there is no need to trust an external entity for applying it. This model also gets rid of the overhead of high message exchange rate and the correlation problem present in the first model. Moreover, there is no complexity of maintaining the correlation for a composite request.

Architectural Issues

- In this model, the reporting of accounting information is integrated into the service and therefore the model is specific to a service oriented grid middleware such as OGSA [5].
- Each grid service, in this model, is overloaded with accounting functionality in addition to its basic business function.
- Adapting an existing service to this model requires changes to the service.
- Since, the accounting metrics may not reflect the usage in terms of the actual resources, we might be losing valuable information regarding actual usage of the underlying resources. These may be needed for reporting or for statistical purposes. To collect such information in this model, each individual service will have to do the logging.

3.3 Model 3: Exclusive metering component for each grid-service

In this model, each grid-service has an exclusive metering component having its own interface. The service invocation and the flow of metering information follow separate paths. Each service sends a unique id as part of the request while invoking an underlying grid-service. It also supplies this id and other metering specific contextual information of the request to its metering component. The called service instructs its metering component to send the accounting metrics of this request to the calling service's metering component.

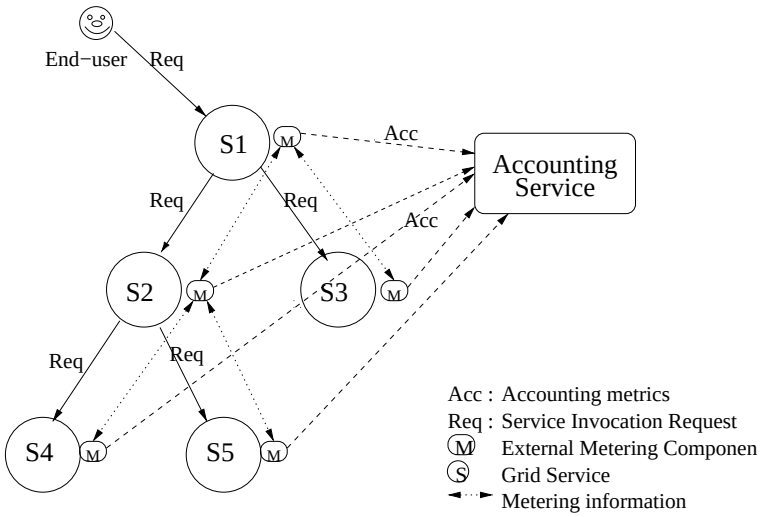

Figure 3: Model 3

This model is different from the distributed version of model 1 (refer section 3.1) in the sense that each grid-service has a dedicated metering component compared to the shared metering service in model 1. This simplifies the task of the metering component because it does not need to maintain correlation for different services. This model retains the benefit of model 1 that each grid service only needs to implement its business function and not the metering functionality. Compared to model 2 this is more scalable since the metering information follows a different path and the metering component itself can be a separate entity.

Architectural Issues

- The metering component of the calling service has to authenticate itself to the metering component of the called service in order to get the metering metrics. In order to facilitate this the calling service will have to give some information about its metering component to the called service.
- Since the metering requests and responses are sent separately and are not a part of the actual request, the message overhead increases.

4 Business models for accounting

Authorization module of a grid service includes an accounting policy to ensure that the requesting user has not used up his/her account. The business model followed by a grid service defines the accounting policy to be used. Three possible business models [4, 6] are considered below:

- The lease/license model
- The membership/subscription model
- The transactional model

A grid-service may support more than one business model. In the lease/license model, the service is available at a fixed price for use with an upper limit on its instantaneous usage and a predetermined lease/license period. The limit on usage at an instant of time could be specified in terms of volume of transactions, volume of simultaneous access [6] or it could even be in terms of amount of resources used.

The membership/subscription model refers to the revenue model that pertains to an established user account with specific terms for usage [6]. The terms of usage may allow the user a cumulative

total usage over a period of time (usage based on period and quantity) or usage amounting to a pre-determined cumulative upper limit (usage based on quantity alone).

The transactional model, utilizes the notion of a charge per transaction. There can be two ways for enabling it. One is *pay-as-you-go/pay-per-use* approach in which the user's request uses the service provider's resources as needed and the user is billed for the total usage or for the period of service usage. This is different from the membership/subscription model because there is no prior arrangement, no validity period and no commitment on behalf of the user in terms of minimum service use. The membership/subscription model is still useful as the service under that model would typically be offered at lower rates due to guaranteed minimum subscription charges. Also, the user may not have any idea regarding how much the request would finally cost while using the pay-as-you-go model. Due to this reason it may be desirable that the service exports its charging procedure which may include charges for its own resources (may be time based and/or quantity based), the charge for the value-add that it provides and the charges that it may incur with each invocation of its underlying grid-services. If the underlying services themselves allow only pay-as-you-go model then they also should export their charging procedure to help the user estimate the total cost of service. The other approach is *pay-per-click/pay-per-request* approach in which the the service has a good estimate of how much usage would result in a particular invocation and can specify the charges upfront (the service may take into account the input values contained in the user's request to arrive at the estimate). Therefore, each invocation has a pre-determined price attached to it.

4.1 Enforcement

- In a lease/license based business model, the accounting related authorization check that needs to be made is that the instantaneous usage of the requester is below the upper limit.
- In a subscription based model, following approaches could be adopted:
  - *Optimistic Approach :* In this approach, the request is accepted if the user has some (may be insufficient) quota left in his/her account. However, if during the execution, the quota is consumed then the request may be aborted half-way through and the user charged for the partial execution. Enforcing this abortion would be non-trivial for composite services. It may be safe to avoid following the optimistic approach, if denial-of-service or other malicious attacks have to be prevented.
  - *Pessimistic Approach :* In this approach, usage requirements of the submitted request are estimated and is compared against the remaining user quota. Only if there is sufficient quota left then the request is accepted. Estimating the usage requirements is the key here. If the service does not export a pay-per-request interface then it may not be possible to estimate the usage requirements. It becomes even more difficult in the case of composite grid services where the top level service may have no clue about the amount of underlying services that the incoming request may need.
- In transactional model enforcement is not required. For the pay-per-request scenario, once accepted a request has to be serviced and for the pay-per-use scenario, the user would be charged as much as he/she uses.

It is implicit that the pricing/usage information exported by individual services (in some directory service or as part of their interface) is in conformance with the business model(s) that it supports. For instance, if a service exports an interface for the pay-per-request transactional model then it is assumed that it can estimate the usage requirements from the request and the reports the service charges accordingly. In other words, once a request has been accepted under pay-per-request model, it has to be served even if the service consumes above estimated usage limits and suffers a loss.

4.2 Policing

Apart from the enforcement of accounting policy by means of checks during the access control, active policing needs to be done to enforce the policy after the request has been accepted. Scenarios such as the Optimistic approach (described above) require that not only the service usage of a user be

monitored and metered periodically but also action should be taken if it violates the preset upper limits. Therefore, mechanisms are needed to:

- **Enforce limits on the service usage :** The basic idea, as seen in Section 3, is that to deal with service usage, three things need to be accounted for. These are the usage of basic resources by the service itself, its value-add and the usage of underlying services. Therefore, all three of these need to be controllable in order to enforce usage limits. Controlling of basic resources is discussed below. Regarding the value-add, it would be determined by the pricing model of the grid service. The key factor is to be able to determine the cumulative usage (i.e., a function of these three parameters for a particular service which includes the usage of the underlying services in the composition tree) and enforce the limit on it. It is a non-trivial task since each service may know how many grid-units it has consumed in terms of the basic resource usage and in terms of value-add but it may not know how many grid-units its underlying services have consumed. Approaches to control the overall usage need to be explored.
- **Enforce limits on usage of basic resources :** Various tools exist that allow basic resources such as storage, bandwidth, <cpu, memory> to be monitored and metered. In addition, generic mechanisms need to be explored that would allow preemption of active requests if the total service usage has exceeded some limit. Systems such as [3] have addressed these issues and may be useful.

5 Conclusion and future work

From the discussion in this paper we think that model 2 should serve as a reasonable architecture to pursue. There are some issues that we are currently addressing. One such issue is the preemption of executing request when the user exceeds his/her quota. Secondly, various interfaces and protocols have to be finalized. Once the interfaces and the protocols have been decided then we need to build a prototype to demonstrate its applicability.

Acknowledgements

We thank Ashish Kundu for participating in various discussions and providing useful comments during the course of this work.

References

[1] C. Anglano, S. Barale, L. Gaido, A. Guarise, S. Lusso, and A. Werbrouck. An Accounting System for the DataGrid Project. Preliminary Proposal v 3.0, http://www.to.infn.it/grid/accounting/DataGrid-01-TED-0115-3_0-acct_prop.pdf, February 2002.

[2] Rajkumar Buyya, David Abrahamson, and Jonathan Giddy. An Economy Grid Architecture for Service Oriented Computing. Working Draft Document, Accounting Models Research Group, Global Grid Forum. http://www.nas.nasa.gov/~thigpen/accounts-wg/Documents/ecogrid.pdf.

[3] F. Chang, A. Itzkovitz, and V. Karamcheti. Secure, User-level Resource-constrained Sandboxing. Technical Report TR1999-795, Department of Computer Science, New York University, Dec 1999.

[4] W. Eibach and D. Kuebler. Metering and Accounting for Web Services. http://www-106.ibm.com/developerworks/webservices/library/ws-maws/?dwzone=webservices.

[5] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers.html, January 2002.

[6] Dan Gisolfi. Web Services Architect Part 2 : Models for dynamic e-business. http://www-106.ibm.com/developerworks/webservices/library/ws-arc2.html, April 2001.

[7] Thomas J Hacker and Brian D. Athley. Account Allocations on the Grid. Working Draft Document, Accounting Models Research Group, Global Grid Forum, http://www.nas.nasa.gov/~thigpen/accounts-wg/Documents/accounttemplates.pdf.

[8] Chris Kenyon and Giorgos Cheliotis. Architecture Requirements for Commercializing Grid Resources. IBM Research Report No. RZ 3408. http://domino.watson.ibm.com/library/cyberdig.nsf/Home, March 2002.

[9] Laura F. McGinnis. Resource Accounting - Current Practices. Working Draft Document, Accounting Models Research Group, Global Grid Forum,http://www.nas.nasa.gov/~thigpen/accounts-wg/Documents/Current Practices.pdf, February 2001.

[10] IPG Accounting Project. Market based Grid Computing - Survey Results. http://www.nas.nasa.gov/Groups/Database/distacct.pdf.

[11] Bill Thigpen and Tom Hacker. Distributed Accounting on the Grid. Working Draft Document, Accounting Models Research Group, Global Grid Forum, http://www.nas.nasa.gov/~thigpen/accountswg/Documents/distributed accounting.pdf.

[12] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, and C. Kesselman. Grid Service Specification. http://www.globus.org/research/papers.html, February 2002.