8158

©Saxe-Coburg Publications, 2011.
Trends in Parallel, Distributed, Grid and
Cloud Computing for Engineering
P. Iványi and B.H.V. Topping, (Editors)
Saxe-Coburg Publications, Stirlingshire, Scotland, ~~888-888~~.

Chapter ~~X~~

Negotiations in an Agent-based Grid Resource Brokering System

K. Wasilewska[1], M. Ganzha[1,2], M. Paprzycki[1,3], M. Drozdowicz[1], D. Petcu[4], ~~C.~~ C. Badica[5], N. Attaoui[6], I. Lirkov[7] and R. Olejnik[8]

XX

[1]Systems Research Institute Polish Academy of Sciences, Warsaw, Poland
[2]University of Gdansk, Poland, [3]Warsaw Management Academy, Poland
[4]Western University of Timisoara, Romania, [5]University of Craiova, Romania
[6]University of Tetouan, Morocco
[7]Bulgarian Academy of Sciences, Sofia, Bulgaria, [8]CNRS, Lille, France

Abstract

The aim of this chapter is to present an overview of issues concerning negotiations, considered as a part of resource allocation process in the grid. In this context, autonomous negotiations constitute one of the crucial phases of establishing the *Service Level Agreement* (*SLA*) between a client and a service provider. Hereafter, European research projects that represent different approaches to the problem of integration of business and the grid architecture are discussed, as well as various methods of establishing the *SLA* between grid users. The experience gained in these projects is used as a foundation of conceptualization of the *SLA* negotiations in the *Agents in Grid* (*AiG*) project. The resulting approach to the *AiG* negotiations is summarized, with particular attention paid to the use of ontologically demarcated information based on the *AiG Ontology* (introduced in [1]).

Keywords: software agents, grid, resource brokering and management, negotiations, ontology.

1 Introduction

1.1 Resources in the grid

Nowadays, grids are gaining popularity as an interesting approach to the creation of highly available computing infrastructures. Note that while there are examples of attempts to utilize grids as providers of non-computational resources (*e.g.* data grids [2]), it is the use of grid as a source of "computational power" that became the most prevalent, and thus is the focus of our work and this chapter. Therefore, the infrastructure of our interest includes heterogeneous, geographically distributed, multi-domain computer resources. For such an infrastructure it was always assumed

PROOF

(see, [3]) that it will become a source of income to its owners (*e.g.* in a manner similar to the Sun Grid by Oracle [4]).

When we consider integration of business into the grid infrastructure, users (physical persons, or institutions) can be divided into two groups, according to the role that they play: (i) owners (service providers) who make their resources accessible, and (ii) clients who search for resources to commission execution of a job. Note that these roles are interchangeable, as providers of one type of services may become consumers of another. Separately, it should be pointed out that "business" computations are usually conducted in a predefined order, and within a fixed time frame (contrary to volunteer computing projects, *e.g.* the ABC@home [5], where neither of these two factors plays an important role). In order to assure availability of resources and timeliness of job execution, it is necessary to formalize the contract in the form of a *Service Level Agreement* (*SLA*), followed by job monitoring, as well as assessing the quality of service. Therefore, grid systems that aspire to provide business solutions should also consider the *SLA* management as part of their functionality. Here, the *SLA* is defined as a contract between a user of a service and a service provider, specifying the scope of the service and co-operation conditions offered during the period of contract validity, to ensure the *Quality of Service* (*QoS*) (see, also [6]). The *SLA* can be considered as a means to provide more flexibility, scalability and optimizing resource utilization, and, consequently, increase the reliability of the grid.

With this background let us proceed as follows. First, in Section 1.2 we outline the *Agents in Grid* (*AiG*) project, which utilizes software agents and semantic data processing for meta-level grid resource management. This project will use autonomous *SLA* negotiations, facilitated by agents representing grid resource providers and consumers. The general concepts needed to discuss *SLA*s and *SLA* formation are described in Section 1.3. The proposed approach to *SLA* negotiations will be based on experiences gained from six European research projects summarized in Section 2. Specifics of the proposed solution will be summarized in Section 3.

PROOF

1.2 The *Agents in Grid* project

To start with, let us outline the main features of our approach to resource management in the grid, proposed in the *Agents in Grid* (*AiG*) project. As stated in [7], one of the possible solutions to the problem of over-complicated grid middleware, is to utilize software agents to infuse intelligence into the infrastructure, and ontology for a knowledge representation. This approach, was used in an initial implementation, described in [8] that serves as a prototype for the *AiG* project. The *AiG* project treats the grid as an open infrastructure that can be joined easily by users acting as clients who want to utilize resources available there (and pay a fair price for their use), or acting as workers who contribute their resources and are paid for their use. It should be underlined that in the case of individual, privately-owned, computers it is difficult to guarantee reliability and ensure that the terms of the *SLA* are met. Therefore, the solution proposed in the *AiG* project is based on utilization of agent teams [8,9]. Other general assumptions that have been made to facilitate the needs of workers and clients are:

- each team has a single leader (the *LMaster* agent),
- an incoming worker (*LAgent* agent) joins teams based on its individual set of criteria,
- decisions about joining a team, and accepting new workers, involves multi-criteria analysis,
- each worker (*LAgent* agent), if needed, can play a role of the *LMaster*,
- each *LMaster* has a "back-up" agent – the *LMirror* agent who can take over its job in case the *LMaster* goes down (a way to add resilience to the team),
- matchmaking is facilitated by a *Client Information Center* (*CIC*) infrastructure, represented by a *CIC* agent. The *CIC* acts as a repository for "offers" and, among others, stores: (i) meta-data describing the team itself, (ii) information about offered services (resources), and (iii) requirements for potential workers. Furthermore, each agent in the system has to register with the *CIC* agent (a way to add security to the system).

Combining these assumptions, resulted in a system presented in Figure 1, as a use case diagram.

Let us now briefly focus our attention on interactions between components of the system (more details will be provided in section 3.2). Two main interaction scenarios are: (i) user, represented by their *LAgent*, is looking for a team to execute their jobs, and (ii) user (also represented by their *LAgent*) who wants to be paid for use of their resources is looking for a team to join. Figure 2 represents the process taking place when a worker is attempting to join a team. After the user formulates conditions for joining (and, if not done already, the *LAgent* registers with the *CIC*), the *LAgent* requests from the *CIC* a list of teams that satisfy the user-specified criteria. As a result the *CIC* agent responds with a list of *LMaster* agents representing suitable teams. After that the *LAgent* negotiates directly with the *LMaster* agents (recall that each team has its own criteria for a candidate worker). When negotiations succeed, the *LAgent* joins the selected team. In the case when no team satisfying the criteria was found, or when negotiations failed, the user has the following options: to quit, to redefine the criteria for team selection, or to register a new team with his *LAgent* being the only member (and thus its *LMaster*). The latter solution may be necessary if multiple attempts to join multiple teams (exploring different joining criteria) failed.

Figure 3 presents the sequence of actions for the case when a user, represented by the *LAgent*, wants to commission job execution. As in the team joining scenario,

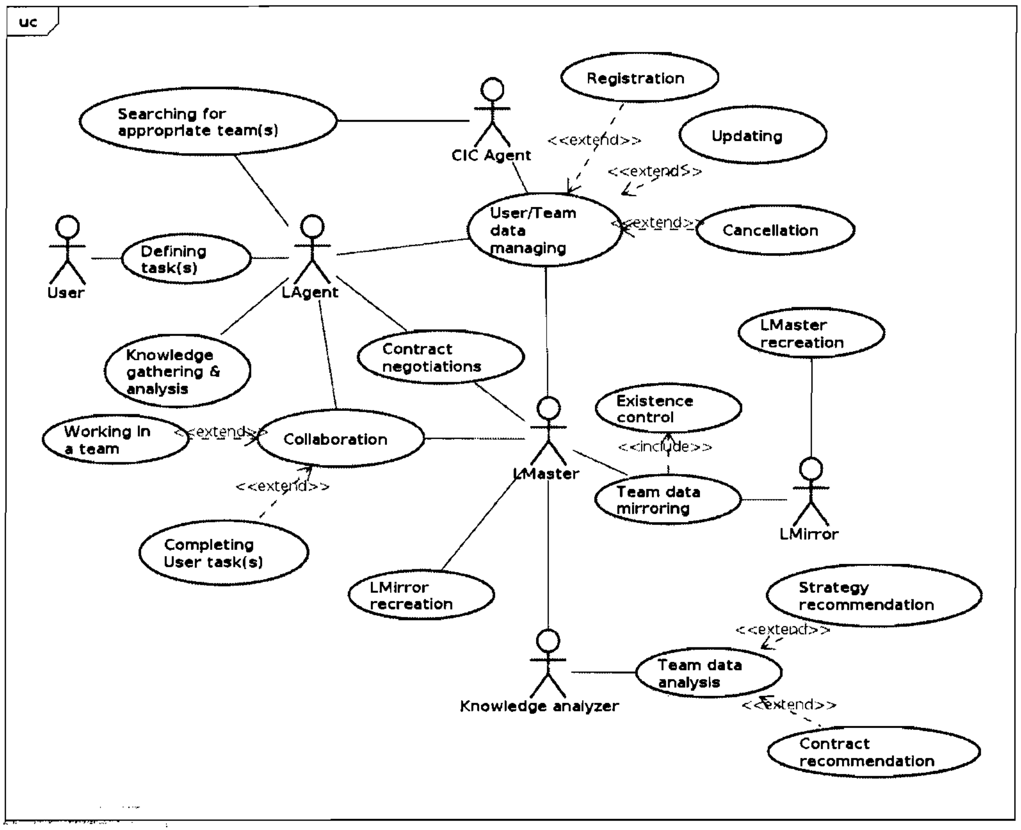


Figure 1: Use case diagram of *AiG* system

the first step for the *LAgent* is to register with the *CIC* (note that this step is needed only once). Afterwards, the *LAgent* requests from the *CIC* a list of teams that satisfy criteria provided by the user. As a result it receives a list of *LMaster* agents that meet the criteria, and negotiates with them directly to choose the one that will execute the job. One should note also here that multi-criteria analysis is used by both, *LAgent* and *LMaster* sides. Multi-criteria analysis is necessary as soon as the team selection is not based on a single parameter (*e.g.* the price). It provides a means to consider multiple parameters and their priorities as would take place in real-life scenarios. In the following sections, a more detailed description of possible negotiation parameters will be given.

In both scenarios the *CIC* is used to narrow down the number of contacted *LMaster* agents (based on the criteria obtained from the *LAgent*). Consequently, the ensuing negotiation phase includes only a limited number of teams represented by their *LMaster* agents.

In summary, from the above discussion, it can be concluded that autonomous negotiations, and *SLA* management, are a key part of the *AiG* system.

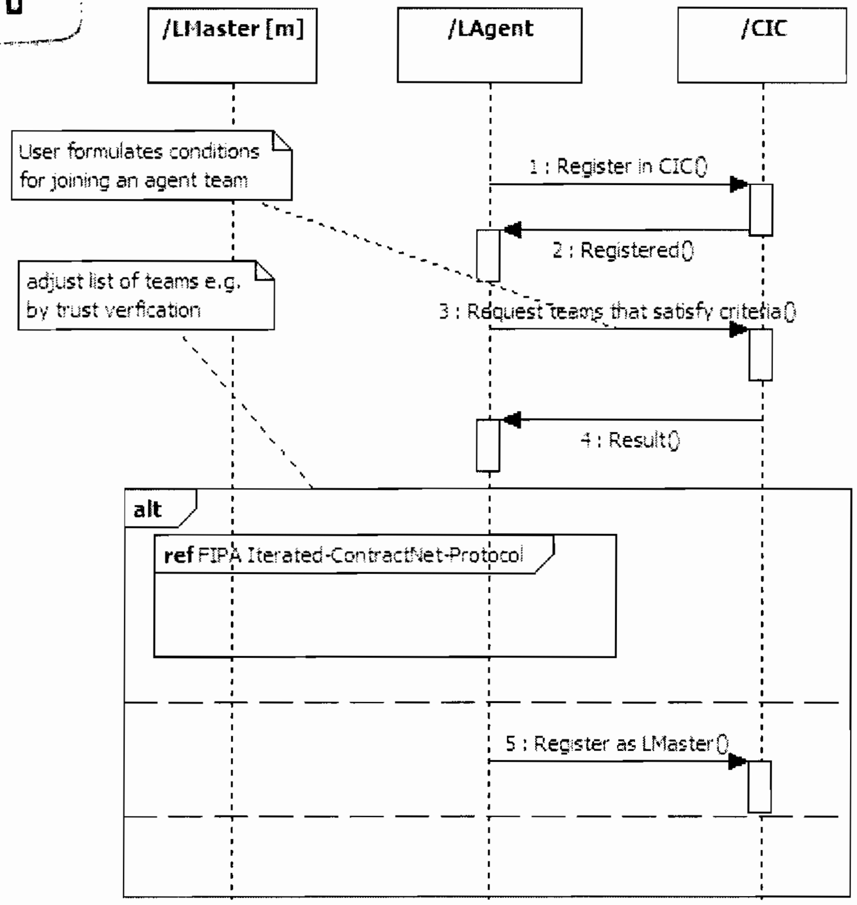

/LMaster [m]
/LAgent
/CIC
User formulates conditions for joining an agent team
adjust list of teams e.g. by trust verfication
1 : Register in CIC()
2 : Registered()
3 : Request teams that satisfy criteria()
4 : Result()
alt
ref FIPA Iterated-ContractNet-Protocol
5 : Register as LMaster()

Figure 2: Sequence diagram for joining a team scenario

1.3 The structure of the *SLA*

Considering the importance of establishing the *SLA* between parties in the resource brokering in the grid, let us begin with an overview of the main issues concerning the *Service Level Agreement* (*SLA*) formation. The *SLA* can be described as a document defining dynamically-established and dynamically-managed relationship between parties that is a result of a negotiation process. One of the most popular conceptualizations of the structure of the *SLA* was proposed in a WS-Agreement protocol specification [6]. There, an agreement is conceptually composed of the following parts:

- *Context:* required element that provides information about the agreement that are not included in the *Terms* section, *e.g.* who the parties involved are, what services they agreed to, and the duration of an agreement.

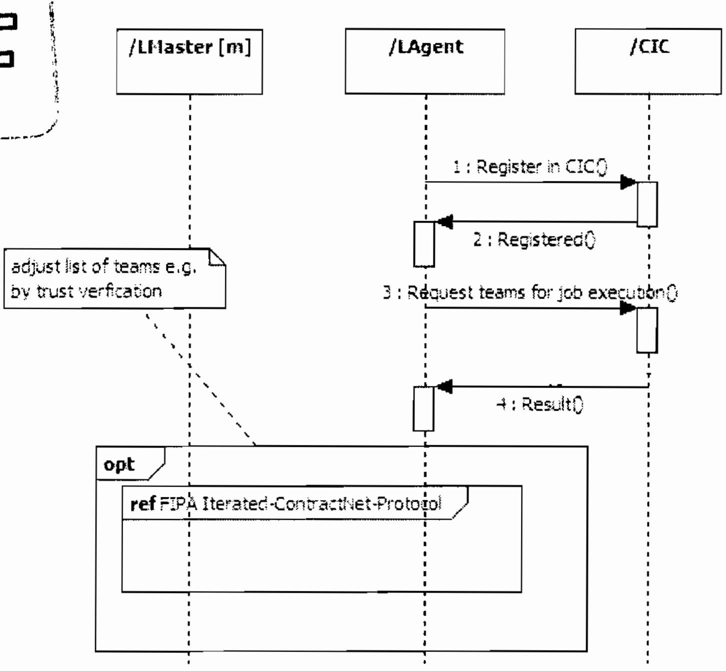

/LMaster [m]
/LAgent
/CIC
1 : Register in CIC()
2 : Registered()
adjust list of teams e.g. by trust verfication
3 : Request teams for job execution()
4 : Result()
opt
ref FIPA Iterated-ContractNet-Protocol

Figure 3: Sequence diagram for contracting a job execution

- *Name:* this is an optional name that can be given to an agreement (usually in a human-understandable form),

- *Terms:* element containing terms of an agreement that consist of:
  - *Service Terms:* provides information that are needed to identify and instantiate a service, to which this agreement pertains,
  - *Guarantee Terms:* specify service levels that the parties agreed that can be used to monitor the service and enforce the agreement.

Apart from the WS-Agreement standard for representation of a contract between parties, there also exists a WSLA (Web Service Level Agreement) [10], which is a framework developed by the IBM, for specifying and monitoring the *SLA* compliance at the runtime. It is based on a WSLA language and structured as an XML document.

The key difference between those representations is that the WS-Agreement *SLA* document is highly extensible *i.e.* it is suggested that customized templates may be created based on domain-specific terms. Another difference is that the WS-Agreement does not explicitly specify metrics associated with individual agreement parameters, as is done in the WSLA. On the other hand, the WSLA is considered an stimulus for the WS-Agreement standard that eventually superseded it [11].

Note that economic models and their execution are one of the important missing components to the grid uptake. Here, the *SLAs* form a basis for moving grids from the research field to business. They constitute a mechanism by which the grid can be used to provide services in a competitive and profit-driven market. The WS-Agreement (and also the WSLA) is very popular as, it has an XML representation and is intended for web services. However, as was stated in the previous section (and in [1]), in the case of the *AiG* project, it was decided that only semantic knowledge representation will be used. Therefore, the WS-Agreement cannot be utilized directly. Nevertheless, the structure and scope of information that is transfers, should be taken into account when developing the semantic contract representation.

2 The *SLA* in other projects

As far as applying business solutions to the grid is concerned, several projects have been recently completed. They deal, in different ways, with establishment, execution and monitoring of execution of the *SLA*. Projects that will be discussed in this section include: NextGRID [12], Akogrimo [13], BEinGRID [14], ARGUGRID [15] and BREIN [16]. They represent very diverse approaches, solutions and technologies, which stemmed from specific requirements for each one of them. It should be mentioned that the authors of [17] prepared an analysis of most of these approaches, and pointed out possible general problems in the area of *SLA* negotiations.

The most crucial conclusions from their work were:

- There is a tendency to represent the *SLA* with a structure proposed in the WS-Agreement specification. However, there is no consistency in the terminology used and a placement of information within a document, since the WS-Agreement does not cover the vocabulary terms for the *SLA*.
- The negotiation process is usually conducted as a simple single-round protocol, *e.g.* as an offer-accept model. The majority of solutions do not consider multi-round negotiations, and/or a mechanism for re-negotiating the *SLA*.

However, it has to be noticed that, when the analysis presented in [17] was prepared, the BEinGRID and BREIN projects were still in progress (and the BREIN project was definitely far from being completed), while (for some unknown reason) the ARGUGRID project was completely omitted. Therefore, the following analysis should be viewed, first, as an extension and completion of the endeavor undertaken in [17]. Second, we will focus on establishing the *SLA*, seen in the context of the *AiG* project. Moreover, extending the results summarized in [17], this chapter also considers accounting and billing approaches used in the five projects. Overall, the results of our analysis; in particular, the best practices used in the projects studied, became an inspiration for the specific *AiG* approach.

Finally, let us note that, in all the projects known to us, the focus of the negotiation was to establish the *SLA* for the case when a client is to contract resources from the grid. However, in the *AiG* project we deal not only with this scenario (note that the *LMaster* represents a team of workers, not a specific grid resource). Separately, we need mechanisms for establishing the *SLA* when the resource-representing agent negotiates joining a team. Here, only general principles of contract negotiations, originating from the projects considered, can be taken into account. However, specific rules for contract negotiations have to be developed to match specific scenarios.

2.1 NextGRID

Since our presentation of completed EU-funded projects pertinent to our work will be chronological, let us start from the NextGRID (Architecture for Next Generation Grids) project, which was undertaken from September 2004 to September 2007. Its aim was to develop an architecture that would enable grids to be commonly used by research and industry, thus creating a dynamic marketplace for services and products [18].

The *SLA*s between service providers and users were one of the main objectives of the project, as they become a foundation for other objectives and a central point in the conceptual model. In the NextGRID, the SLA was represented as a XML document based on the WS-Agreement and the WSLA standards. Interestingly enough, according to [17, 19], NextGRID developers decided not to extend the WS-Agreement but to create their own representation; based on both the WS-Agreement and the WSLA standards, used as an "inspiration." In [19], it is stated that such a decision was made because the WSLA standard was superseded by the WS-Agreement, and the WS-Agreement did not consider the business context (environment description) represented with static terms (*e.g.* state of the system or network) in the *SLA* document (but it was good as a starting point). Obviously, such a decision can be seen as somewhat controversial, by not advancing standardization in the SLA formation area.

The NextGRID *SLA* is split into three parts: (a) the *Parties* section, which is similar to the *Context* section in the WS-Agreement specification and contains information about parties to the contract (*e.g.* contact, finance information, security keys in case of sensitive data), (b) the *Dynamic Terms* section, which contains negotiable terms with information about how to measure those terms and their possible violations (e.g. according to [20], a package of offerings with the option to choose a pre-quantised quality level was introduced to avoid the complexity, cost and time of multi-parameter negotiations), (c) the *Static Terms* section, which includes non-negotiable terms describing the overall environment [20] (*e.g.* if the provider is compliant with a specific ISO norm). This last section is an extension to the WS-Agreement standard.

In the context of the *AiG* project, the idea of dividing parameters into negotiable and non-negotiable sections seems to be worth considering. Note that we can distinguish terms that describe the infrastructure of a worker/team hardware resources that are constant in time, as well as terms that depend on the current state of the system e.g. availability or workload. However, mechanisms to change the classification of terms, which would enable re-negotiation of an existing *SLA* are missing. For instance, let us consider a contract between a client and an agent team to execute a job. A job completion time may be initially treated as a non-negotiable parameter placed in the *static Terms* section. However, during the negotiation process, the client's requirements may change and the job completion deadline may also undergo negotiations. If there was a possibility to mark a given parameter as negotiable then, the negotiation protocol could proceed without reinitialization. Unfortunately, such parameter flexibility is not available in the negotiation model introduced by the NextGRID project.

NextGrid takes care of accounting and billing by means of a dedicated component residing at the service provider, which monitors and reports resource usage for the particular metrics defined in the *SLA*. A monetary charge can be treated as one of the metrics. The user is able to query the resource usage at any time, to determine the amount of resource used over time for metrics of interest. Here querying for resource usage with the metric being a monetary charge is equivalent to getting a bill for job execution.

An interesting comment, based on the NextGRID project experience, was given by the authors of [21]. Here, it was suggested that in many cases it is more efficient to utilize a third party negotiation module than to have it built in to the client and service provider software, *i.e.* instead of putting the full burden of negotiation on all parties involved, parties can profit from utilizing other provider's expertise (ones providing third-party brokers). Information about the third party modules may be included into the parties section of the NextGRID *SLA*. This claim is interesting in its own right and in some way matches considerations presented in [22]. However, the *AiG* system considers the negotiation module(s) as an integral part of the logic of the *LAgent* and the *LMaster*. This approach is in line with the advances in agent technology, and understanding of autonomous contract negotiations. These advances can be seen not only in projects described later in this section, but also in the ongoing work realized within the framework of the EU COST action IC0801 "Agreement Technology" [23].

Finally, according to [24], the NextGRID uses a Discrete-Offer protocol, which is an example of a one-phase negotiation. This protocol assumes that the client may choose only from a set of offers received from the service providers. Offers cannot be modified by the client, they can be only accepted or rejected, and accepted offers lead to the *SLA* establishment. In the *AiG* project a more complex approach is planned to be applied, *i.e.* to use multi-round (iterative) negotiations with a possibility to re-negotiate the existing *SLA*.

2.2 Akogrimo

The second project combining grid and business solutions was Akogrimo (Access to Knowledge through the Grid in a Mobile World). It was undertaken from June 2004 to September 2007, and driven by an idea that the next generation grids should be integrated with the next generation networks. As a result it assumed a pure IP-based underlying infrastructure (ALL-IP) for the grid [13]. According to [25], the utilized *SLA* representation was based on an XML document and used the WS-Agreement and the WSLA standards. Evaluation of the prototype led to the conclusions similar to those from the NextGRID project, *i.e.* both static (non-negotiable) and dynamic (negotiable) terms need to be present in the *SLA*. Therefore, in the Akogrimo, a mix of both standards was used to obtain the best *SLA* template. The *SLA* document specification was derived from the WS-Agreement (structure of the document), however the contract template definition was based on an analysis of approaches from the WSLA specification. As is easy to see the concepts inherent in the *SLA* representation the in Akogrimo (as in the NextGRID) cannot be directly applied in the *AiG* project, which shall use only semantic knowledge representation. Interesting solution that was applied was to use high-level and low-level parameters. High-level parameters are used when a user specifies her/his requirements. In the next step, high-level parameters are mapped on to low-level parameters that are transparent to the user. Policy metrics related to the mapping are defined by the service provider. Consequently, two equivalent *SLA*s are established. The one with high-level parameters is user-friendly, while the other, with low-level parameters (transparent to the user), is used internally, *e.g.* by the monitoring component. This is an interesting approach, however in the *AiG* project it is the user who specifies her/his requirements by creating constraints on low-level parameters using a user-friendly GUI. Furthermore, as stated in [26], while the Akogrimo uses an extended WS-Agreement protocol, it is still based on a single accept/reject model that we consider to be too simple for realistic negotiation scenarios (see, also the discussion in the previous section).

When we consider accounting and billing processes in the Akogrimo ([25]), the following assumptions were made: (i) every service has parameters that it wishes to account for and implements meters for them (*e.g.* some jobs can be CPU consuming, while others are I/O intensive, or both), (ii) there is a charging component that receives the accounting data, (iii) according to a charging schema, charges are calculated and added to the user's aggregated monthly bill. Here, the charging schema may consider that job executed will be CPU intensive and charge only for the CPU time, while I/O operations will be offered free of charge. Moreover, in the Akogrimo, the metering of chargeable parameters can be applied on the job (measurements for a specific job are reported) or on the user level (measurements for a specific user are reported).

2.3 BEinGRID

The BEinGRID (Business Experiments in Grid) project started in June 2006 and ended in November 2009. One of its objectives was to develop and deploy numerous grid-enabled projects and, thus, to understand the requirements for grid use in the commercial market. Furthermore, it has build a toolset repository with components and solutions based on the main Grid software distributions *e.g.* Globus Toolkit, gLite, Unicore, GRIA [27].

One of the components developed within the BEinGRID project, and described in [28], is an *SLA Negotiator*, which can be used to negotiate *SLAs* and provides implementation of the WS-Agreement protocol. The *SLA* representation is not limited to the WS-Agreement specification but may differ depending on the requirements of the prototype application, e.g. the GRIA *SLA* specification was also considered. According to [17], the message sequence in the agreement protocol can be divided into two phases (1) the pre-agreement service search, and (2) the agreement phase. Note a potential problem that can arise in that approach, since resources are not locked during the negotiation process. This may lead to a situation where resources may become unavailable, while the negotiations concerning them are still "in progress." Authors of [17] also point out that the *SLA* negotiation protocol is incomplete, and there are additional flows of messages that should be considered. Nevertheless, the BEinGRID negotiation protocol is closest to multi-round negotiations among the projects that have been presented thus far.

To deal with accounting and billing, an *SLA* accounting component (following the *SLA* Accounting Design Pattern) was developed. It is responsible for calculating the total price and sending it to the company's billing department [29]. The price is evaluated on the basis of notifications, sent by the evaluation component, where violations and threats are also taken into account.

The BEinGRID project provided a set of prototype applications to demonstrate the benefits of business in the grid, and additionally developed a set of design patterns to manage all stages of the *SLA* lifecycle. Overall, this project was not bound to one technology, but conceptual solutions that were presented can be developed in different technologies. In the *AiG* project, an accounting module is to be developed as well, and the *SLA* Accounting Design Pattern provides a good formal description of this service. However, apart from sending a final bill to a billing department that forwards it to the client, in our case it would be better to have the possibility to send it directly to the client. The *AiG* treats the grid as an open environment, and is not dedicated only to companies but also to private persons who want to earn commission on providing resources to the grid (their team). Additionally, it would be beneficial if a client had the opportunity to query resource usage and the current bill during job execution (as in the NextGRID).

2.4 ARGUGRID

A research project that utilizes software agents and processes data semantically is ARGUGRID. In this project a two-phase approach to negotiations, based on argumentation, has been proposed. According to [15], negotiation utilizes the minimal-concession protocol that proceeds in rounds and makes alterations to offers already submitted possible (the reciprocity principle). The minimal concession strategy starts the bargain with the "best" offer (made by all sides) and then participants concede minimally in order to come to an agreement. Agents take turns in making offers. For example, let us consider only one parameter *i.e.* the price of the service. A service provider starts with the best offer from him *e.g.* 60 (in any monetary value), and the client responds with 40. In the third step the services provider concedes minimally, offering 50. Since the opponent conceded in the previous step, the client also concedes and responds with a higher sum. If it is 50, then both sides reached agreement, otherwise the process continues. Moreover, in the ARGUGRID, a protocol called reward-based minimal concession protocol was developed (for more details, see [15]). It enables making offers with additional services during the negotiation process and basing them on the needs of the other party. This is possible because mutual preferences are known to the negotiating parties, as well as the importance of specific (individual) negotiation terms. For example, at the beginning of the negotiation process, the service provider knows that the client needs storage space, so the initial offer is based on this requirement. However, during the negotiation proceeds, the services provider may include additional services that he assumes client may need *e.g.* administration and data backups. Such an assumption can be debatable since not every user wants to make his preferences public. Furthermore, preferences may change on the basis of offers obtained. For instance, a user may not be particularly interested in a machine with large memory, but if such a machine suddenly becomes available (one of the teams has such machine free and offers it cheaply, rather than have it running empty), the user may be willing to take it (which was not represented at all in the original set of preferences).

As far as the *SLA* representation is concerned, in the ARGUGRID its representation is based on a module provided by the GRIA (grid middleware) *SLA* Management service (with its own *SLA* representation). Each service is associated with a specific *SLA* template. It includes service description, endpoint reference, non-functional properties (*e.g.* price, availability) and and the *SLA* template that refers to a specific service.

As far as use of ontologies is concerned, agents and objects are described as using a standard WSMO Ontology (an extension of the OWL-S, [30]). In the ARGUGRID, the general *Entity* is described by its *Name*, its *Birthplace*, and by the set of messages to interact with it. *Agents* and *ObjectNodes* are represented as sub-concepts (specializations) of the WSMO *Entity* concept, *e.g.* attributes of an agent include additional role, ontology, language and protocol. This being the case, it is important to notice that there is no direct use of semantic data representation in the *SLA* template.

Finally, in the ARGUGRID, billing terms are included in the *SLA*. Billing and accounting seem to be managed by the middleware that supports the service oriented architecture.

2.5 BREIN

The last project discussed in this chapter is BREIN (Business Objective Driven Reliable and Intelligent Grids for Real Business). It started in September 2006 and ended in January 2010. Its aim was to enable easy usage of grid technologies for the needs of business participants, by leveraging agent systems and the semantic web. Similarly to other projects, the *SLA* representation is based on the WS-Agreement and the WSLA specifications, and constitutes a non-semantic description of the *SLA* template (XML is used). Interestingly enough, this project, as with the ARGUGRID, utilizes software agents; but still uses non-semantic *SLA* representation.

An innovative solution applied in the BREIN project was to introduce semantic annotations in order to provide compatibility with other standards but at the same time to utilize semantics, as well as to translate high-level requirements *e.g.* the time to complete the job, into low-level infrastructure terms, *e.g.* four CPUs [31]. Term translation allows the user to define requirements using their terminology and the system is able to understand it and discover appropriate services. Furthermore, a mapping layer assures coherent understanding of terms included in the SLA templates. The semantic layer, obtained by mapping the *SLA* terms on to the semantic terms, may use RDFS, OWL or WSMO platforms (however the BREIN's knowledge base uses OWL). One of the innovations, claimed in the BREIN, is the optimization of terms and conditions according to business party policies, *e.g.* finding a service that is as cheap as possible. This means that there is a possibility to assign priorities to negotiation parameters.

BREIN (similarly to the ARGUGRID) can be considered an intermediate step between projects using XML for the *SLA* representation, and a fully semantic approach that characterizes the *AiG*. It still transmits the *SLA* as the XML (based on the WS-Agreement and the WSLA), but then the proposal is translated into the ontological representation, for further evaluation. However, the response is translated into XML before sending. The BREIN ontology allows the *SLA* terms to be defined along with respective metrics and units that lead to unambiguous representation of these terms. According to [16], the BREIN has a mechanism for conducting multi-round negotiations based on the FIPA Contract Net Protocol. However, from [31] it can be suspected that the negotiation process actually is not iterative, with offer modifications but is based on an initiation of multiple protocol instances *i.e.* if a proposal is rejected a new protocol instance is initiated. Moreover, a multi-round approach seems to be achieved by extending the FIPA Contract Net to support combinatorial multi-attribute auctions (for more on combinatorial auctions, see [32]) over multiple supply chain levels (tiers), which, among others, allows subcontracting. Furthermore, as stated in [16], negotiation is performable using agents and web services or only web services. Accounting in BREIN is conducted by the billing calculator component that records resource usage and issues bill at the end of a collaboration.

3 The proposed *AiG* approach

Let us now re-focus on the *AiG* project, and its approach to the problem of *SLA* negotiations. It should be stressed that, as described in Section 2, contrary to the aforementioned projects, in the *AiG* there are two possible negotiation scenarios:

- prospective worker (*LAgent*) negotiates (*LMaster*) conditions for team joining with the team leader,
- client (*LAgent*) negotiates (*LMaster*) the *SLA* regarding job execution with the team leader.

In the following sections, the communication protocol and the ontology for those two scenarios will be discussed.

3.1 Ontology for negotiations

In [33], available languages and ontologies for resource description in the grid were analyzed from the point of view of their applicability in the *AiG*. As a result a decision was made to modify and extend the ontology developed for the CoreGrid. One of the important areas that needed extension was related to *SLA* negotiations (*SLA* management was not considered in the CoreGRID project). Therefore, additional ontologies were developed that model messages exchanged in the system during contract negotiations (*AiG Messages Ontology*), and provide a definition of terms used in the *SLA* (*AiG Conditions Ontology*). For a complete description of both ontologies, as well as other modifications to the CoreGRID ontology, see [1].

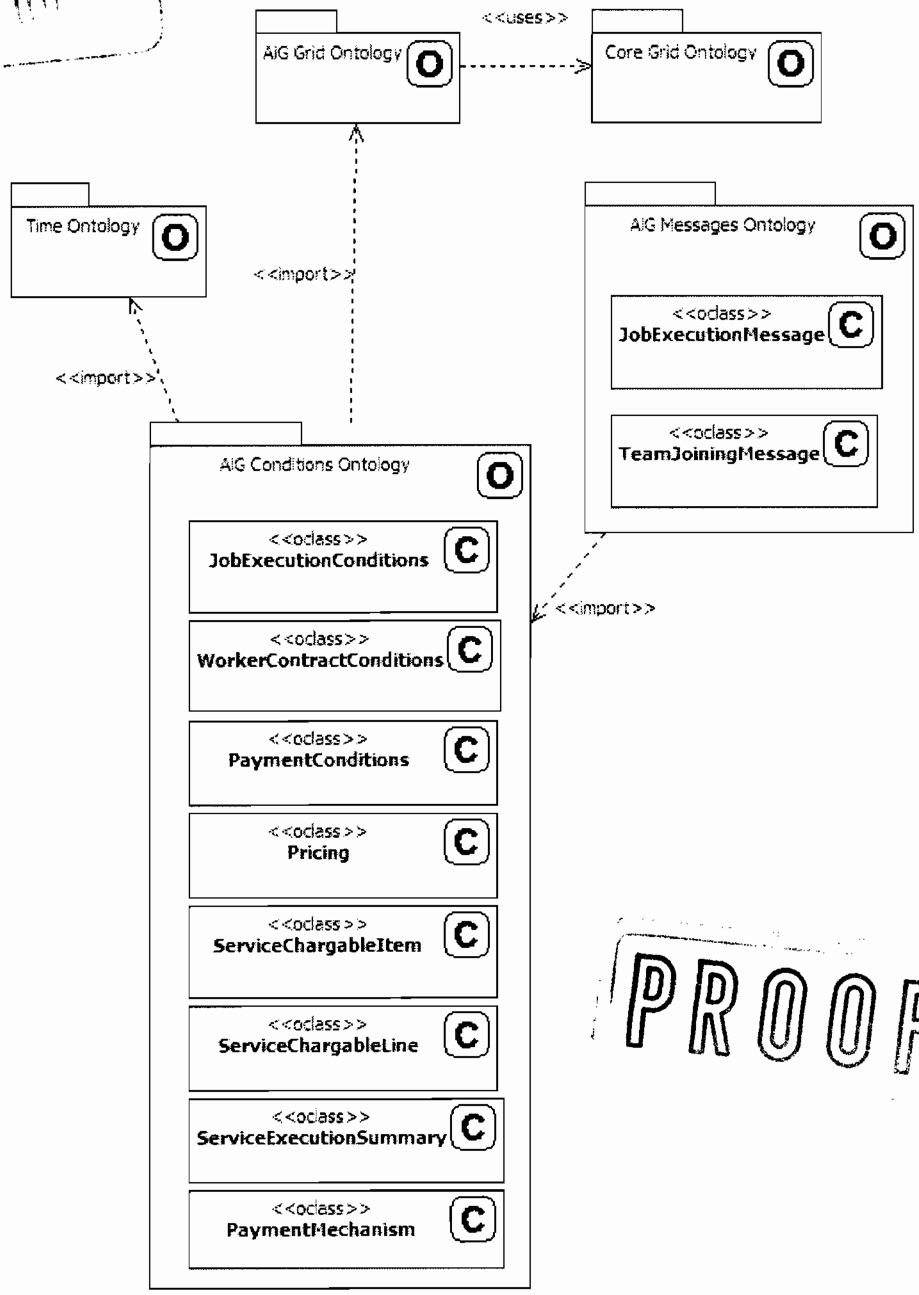
The resulting ontology supports single-round as well as multi-round negotiations with different negotiation strategies. In addition, in order to assess offers from negotiation participants, multi-criteria analysis is to be used. Therefore, each term that is negotiable should have its representation in the *AiG Conditions Ontology*. Similarly to the NextGrid project, parameters can be divided into static and negotiable. In a scenario when a user wants to commission job execution, the following criteria are considered: price, job execution start and end time, penalty for delay of job execution. In a scenario when a worker searches for a team to join, the criteria considered include *e.g.* revenue (per availability or per utilization), contract duration, possibility to extend contract and guaranteed utilization. Besides the negotiable terms mentioned, the offer contains requirements for hardware and installed software that are necessary for job execution, or that of a worker who wants to join a team.

AiG Messages Ontology defines messages that are exchanged in the *AiG* system, with content defined in terms of the aforementioned ontologies. Messages, exchanged during the negotiation process, should pass parameter values or constraints imposed on them, *i.e.* minimum, maximum (for price of job execution, or revenue of a worker), set possible values for discrete parameters (*e.g.* acceptable CPUs) or ranges for continuous parameters. Additionally, there should be a possibility to assign priorities to terms by both negotiating parties, in order to place them in a hierarchy of importance, *e.g.* a worker in a team can decide that guaranteed utilization is more important than revenue. This functionality resembles that of an optimization in a negotiation process in the BREIN project.

3.2 Interaction protocol for negotiations

Let us now focus on the negotiation process, for the two scenarios that are considered in the *AiG*. First of all, by a negotiation process we understand a flow of messages between parties, which in this case are users represented by *LAgents*, and teams represented by *LMaster* agents. After considering the projects described above, it was

PROOF



PROOF

Figure 4: Ontologies used in *AiG* project.

decided that in the *AiG*, the communication is based on the FIPA Contact Net Protocol (one-round [34] or the Iterated FIPA Contract-Net Protocol [35]). The Minimal Concession Protocol used in the ARGUGRID is not to be used in *AiG* project, since there is no assumption that negotiation sites' preferences will be public. Therefore, an offer cannot be modified based on the opponent's preferences. In the Iterated FIPA Contract-Net Protocol, when no team presented an acceptable offer, the user has to modify the offer basing the modification only on its preferences, and issue the *Call-For-Proposal* message once more. This approach is similar to that in the BREIN project, however each iteration will use a modified offer from the previous iteration, instead of re-initializing the whole protocol. An interesting problem, still to be considered, is to provide a mechanism for re-negotiation of existing *SLAs*. Here we seek a way to apply the Minimal-Concession Protocol of the ARGUGRID project, where the existing *SLA* is the starting point of one side, while the desired *SLA* is the starting point of the other.

Now let us discuss the Iterated FIPA Contract-Net protocol in more detail. Figure 5 presents a sequence diagram for the Iterated FIPA Contract Net Protocol that is to be utilized in the *AiG* project. Messages described in the protocol specification and used in both negotiation scenarios include: (i) initiating an ACL message of type CFP (*Call-For-Proposal*), that contains the user's requirements sought for the team *e.g.* required software, operating system, processor type *etc.* (in both cases the initiator of the negotiation process is the *LAgent*), (ii) an ACL message of type *Propose* that is a response to the received offer, (iii) messages of type *Accept-Proposal* and *Refuse-Proposal* that are send respectively to an *LMaster* of a chosen team, and to the *LMaster* agents representing teams that were not accepted, (iv) message of the *Inform-Done* type that is sent by the *LMaster* after receiving the *Accept-Proposal* message. The difference between the FIPA Contract-Net Protocol and its iterated version is the possibility for the initiator to iterate the process by issuing a revised *Call-For-Proposal* to a selected subset of participants, and rejecting others. The goal of this protocol is to allow the initiator to seek better offers by modifying the original CFP, and requesting new offers.

4 Concluding remarks

Studying the European projects integrating business and grid infrastructure, it is possible to point out two potential problems. First of all, most implementations consider only simple one-round negotiations based on a discrete-offer protocol. (ARGUGRID is the only project with a more advanced multi-round protocol.) To address this issue, in the *AiG* approach, multi-round negotiations will be achieved by applying the FIPA Iterated Contract Net Protocol; within which the *AiG Messages Ontology* will be utilized. Second, non-standardized terminology and vocabulary, and lack of common terms representation within the *SLA* can become a problem for wide-spread uptake of any of the solutions discussed. Here, what is particularly revealing, is use of the WS-Agreement and the WSLA standards as a basis to lead to different final solutions,

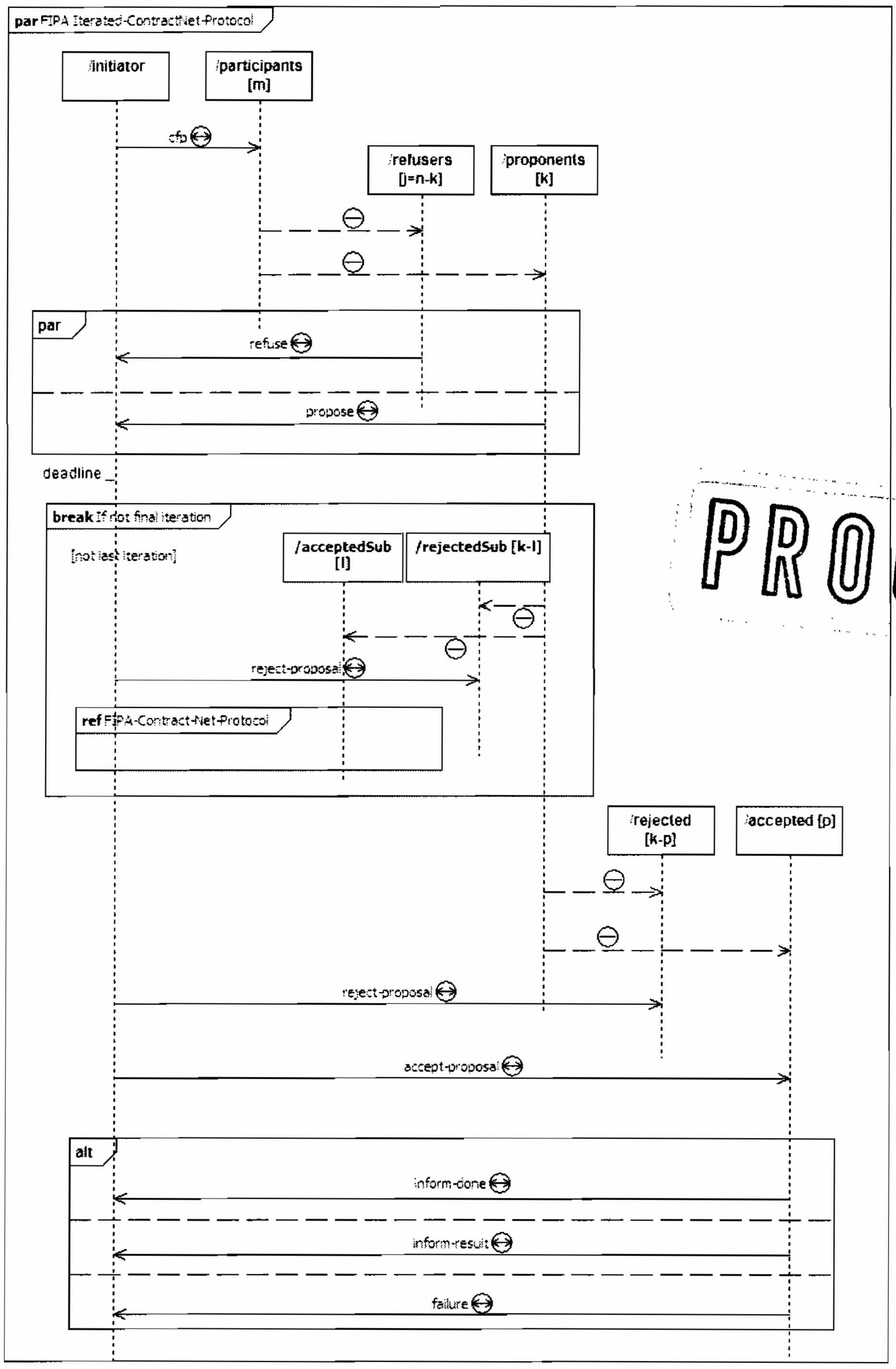


Figure 5: FIPA Iterated Contract Net Protocol

and lack of use of an actual *SLA* ontology in either project. Ontologies developed within the *AiG* project partially solve this problem by arranging terminology, defining messages, and their content. One of the important goals that needs to be realized in the *AiG* project, is to use developed ontologies to define the ontology of a complete *SLA* template.

Acknowledgement

The work of Maria Ganzha and Michal Drozdowicz was supported from the "Funds for Science" of the Polish Ministry for Science and Higher Education for years 2008-2011, as a research project (contract number N516 382434). The Polish-Bulgarian collaboration was partially supported by the Parallel and Distributed Computing Practices grant. The Polish-French collaboration was partially supported by the PICS grant New Methods for Balancing Loads and Scheduling Jobs in the Grid and Dedicated Systems. The Polish-Romanian collaboration was partially supported by the Agent-Based Service Negotiation in Computational Grids; and the Agents, Grids and Heterogeneous Computing grants.

References

[1] M. Drozdowicz, K. Wasilewska, M. Ganzha, M. Paprzycki, N. Attaoui, I. Lirkov, R. Olejnik, D.Petcu, "Ontology for Contract Negotiations in an Agent-based Grid Resource Management System", in P. Iványi, B.H.V. Topping, (Editors), "Trends in Parallel, Distributed, Grid and Cloud Computing for Engineering", Saxe-Coburg Publications, Stirlingshire, UK, 2011.

[2] `http://eu-datagrid.web.cern.ch/eu-datagrid/`

[3] I. Foster, C. Kesselman, (Editors), "The Grid 2, Second Edition: Blueprint for a New Computing Infrastructure", The Elsevier Series in Grid Computing, Elsevier, 2004.

[4] "Sun Utility Computing". `http://www.sun.com/service/sungrid/`

[5] Mathematical Institute, Leiden University, "ABC@home". `http://abcathome.com/`

[6] "Web Services Agreement Specification (WS-Agreement). [Online] Open Grid Forum (OGF)", 2007. `http://www.ogf.org/documents/GFD.107.pdf`

[7] I. Foster, N.R. Jennings, C. Kesselman, "Brain Meets Brawn: Why Grid and Agents Need Each Other", International Joint Conference onAutonomous Agents and Multiagent Systems, 1, 8–15, 2004.

[8] M. Dominiak, M. Ganzha, M. Gawinecki, W. Kuranowski, M. Paprzycki, S. Margenov, I. Lirkov, "Utilizing Agent Teams in Grid Resource Brokering", International Transactions on Systems Science and Applications, 3(4), 296–306, 2008.

PROOF

[9] W. Kuranowski, M. Ganzha, M. Gawinecki, M. Paprzycki, I. Lirkov, S. Margenov, "Forming and managing agent teams acting as resource brokers in the Grid – preliminary considerations", International Journal of Computational Intelligence Research, 4(1), 9–16, 2008.

[10] "Web Service Level Agreements (WSLA) Project", `http://www.research.ibm.com/wsla/`.

[11] P. Bianco, G.A. Lewis, P. Merson, "Service Level Agreements in Service-Oriented Architecture Environments", Software Engineering Institute, 2008.

[12] B. Mitchell, P. Masche, B. Koller, T. Sandholm, "P4.5.3: Management Framework Reference. NextGRID WP4.5 (Advanced Deployment, Service Management and Migration) Deliverable", 2005.

[13] J.M. Johnert, S. Wesner, V.A. Villagr, "The Akogrimo Mobile Grid Reference Architecture - Overview", 2006. `http://www.mobilegrids.org`

[14] T. Dimitrakos, (Editor), "Design Patterns for SOA and Grid", Technical report, BEinGRID Meta-Deliverable AC1, 2007.

[15] P.M. Dung, P.M. Thang, "Deliverable D.4.1: Towards argumentation-based contract negotiation", 2006.

[16] G. Laria, "Annex D of Final BREIN Architecture: Building Blocks Detailed Design", 2009.

[17] M. Parkin, R. Badia, J. Martrat, "A Comparison of SLA Use in Six of the European Commissions FP6 Projects", 2008. `http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0129.pdf`

[18] "NextGRID Final Report", Technical report, The University of Edinburgh and members of the NextGRID Consortium, 2008.

[19] S. Davey, "NextGRID Architecture. Presentation given to CoreGRID Summer School", 2006. `http://www.nextgrid.org/training.html`

[20] B. Mitchell, P. Mckee, "SLAs A Key Commercial Tool", in P. Cunningham, M. Cunningham, "Innovation and the Knowledge Economy: Issues, Applications, Case Studies", IOS Press Amsterdam, 2005.

[21] P. Hasselmeyer, C. Qu, L. Schubert, B. Koller, P. Wieder, "Towards Autonomous Brokered SLA Negotiations", in P. Cunningham, M. Cunningham, "Exploiting the Knowledge Economy – Issues, Applications, Case Studies", IOS Press, 3, 2006.

[22] C. Badica, M. Ganzha, M. Paprzycki, "Implementing Rule-Based Automated Price Negotiation in an Agent System", Journal of Universal Computer Science, 13(2), 244–266, 2007.

[23] "COST Action: Agreement Technology". `http://www.agreement-technologies.eu/`

[24] P. Hasselmeyer, H. Mersch, B. Koller, H.N. Quyen, L. Schubert, P. Wieder, "Implementing an SLA Negotiation Framework", in "Exploiting the Knowledge Economy: Issues, Applications, Case Studies", eChallenges, The Hague, The Netherlands, 2007.

[25] A. Litke, "D4.3.1 Architecture of the Infrastructure Services Layer V1", 2005. `http://www.akogrimo.org/modulese73d.pdf?name=`

PROOF

UpDownload&req=getit&lid=37

[26] P. Cunningham, M. Cunningham, "An Enhanced Strategy for SLA Management in the Business Context of New Mobile Dynamic VO", in J. Martrat, P. Ritrovato, S. Wesner, (Editors), 2008. http://www.scientificcommons.org/43364224

[27] "BEinGRID: Business Experiments in Grid". http://www.beingrid.eu/about.html

[28] R. Heek, R. Kubert, "SLA Negotiator User Guide". https://gforge.beingrid.eu/gf/project/slanegotiator

[29] I. Rosenberg, R. Kubert, "SLA Accounting GT4 Implementation Pattern. BEinGRID. WP 1.7: Component Development", 2007.

[30] "WSMO: an WOL-S extension". http://www.w3.org/Submission/WSMO/

[31] D. Laria, "D4.1.3 Final BREIN Architecture", 2009.

[32] P. Cramton, Y. Shoham, R. Steinberg, (Editors), "Combinatorial Auctions", MIT Press, 2005.

[33] M. Drozdowicz, M. Ganzha, M. Paprzycki, R. Olejnik, I. Lirkov, P. Telegin, M. Senobari, "Ontologies, Agents and the Grid: An Overview", in B.H.V. Topping, P. Iványi, (Editors), "Parallel, Distributed and Grid Computing for Engineering", Saxe-Coburg Publications, Stirlingshire, UK, Chapter 7, 117-140, 2009. doi:10.4203/csets.21.7

[34] "FIPA Contract Net Protocol Specification". http://www.fipa.org/specs/fipa00029/SC00029H.html

[35] "FIPA Iterated Contract Net Interaction Protocol Specification". http://www.fipa.org/specs/fipa00030/PC00030D.pdf