A Framework for Web Service Negotiation

SHAMIMABI PAUROBALLY
University of Westminster
VALENTINA TAMMA and MICHAEL WOOLDRDIGE
University of Liverpool

In a survey on the theory and practice of agent system deployment, conducted by the AgentLink workgroup on networked agents, it was found that there are an increasing number of initiatives for the migration of agents research towards new Internet technologies such as the semantic web, Grid, and Web services. In fact, Grid computing and multi-agent systems research have similar objectives. They both aim to achieve "large-scale open distributed systems, capable of being able to effectively and dynamically deploy and redeploy computational (and other) resources as required, to solve computationally complex problems" [Foster and Kesselman 2003]. On the one hand, service-oriented Grid architectures need to support dynamic cooperation, negotiation, and adaptive interactions between Web services controlling Grid resources for efficient resource and task allocation and execution. On the other hand, the Grid can facilitate agent communication, life-cycle management, and access to resources for agents. Although the relevance of Grid for agent research and vice versa has been identified in several forums, actual collaborative applications are still in their infancy. In this article, we discuss our recent work on deploying multi-agent negotiation techniques to facilitate dynamic negotiation for Grid resources as a step closer to an adaptive and autonomous Grid. In particular, we describe a Web service development of the Contract Net Protocol for negotiation between insurance companies and repair companies. We evaluate our approach to show the added value of negotiable interactions between Web services as opposed to inflexible single-shot interactions that are currently the state of the art.

Categories and Subject Descriptors: I.2.II [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Grid, negotiation, Web services, insurance

ACM Reference Format:

Paurobally, S., Tamma, V., and Wooldrdige, M. 2007. A framework for Web service negotiation. ACM Trans. Autonom. Adapt. Syst. 2, 4, Article 14 (November 2007), 23 pages. DOI = 10.1145/1293731.1293734 http://doi.acm.org/ 10.1145/1293731.1293734

The work presented in this article was supported by the EU FP6 Project Ontogrid (IST-FP6-511513). This publication reflects only the authors' views. The EU is not liable for any use that may be made of the information contained herein.

Author's address: S. Paurobally, Department of Information Systems and Computing, University of Westminster, U.K.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2007 ACM 1556-4665/2007/11-ART14 $5.00 DOI 10.1145/1293731.1293734 http://doi.acm.org/ 10.1145/1293731.1293734

1. INTRODUCTION

The long-term Grid vision involves the development of "large-scale open distributed systems, capable of effectively and dynamically deploying and redeploying computational (and other) resources as required, to solve computationally complex problems" [Foster and Kesselman 2003]. But the Grid community is by no means the only research community with such a set of goals: research in multi-agent systems addresses a very similar set of issues [Wooldridge 2002]: "The Grid and agent communities are both pursuing the development of such open distributed systems, albeit from different perspectives. The Grid community has historically focused on [. . .] "brawn": interoperable infrastructure and tools for secure and reliable resource sharing within dynamic and geographically distributed virtual organizations (VOs) [Foster et al. 2001], and applications of the same to various resource federation scenarios. In contrast, those working on agents have focused on "brains," that is, on the development of concepts, methodologies, and algorithms for autonomous problem solvers that can act flexibly in uncertain and dynamic environments in order to achieve their objectives" [Foster et al. 2004]. Thus, research in the architecture of the Grid has focused largely on the development of a software middleware with which complex distributed systems, (often characterized by large datasets and heavy processing requirements) can be engineered. On the other hand, research in multi-agent systems has addressed the issue of how independently acting (often called "autonomous") computing elements (termed agents) can work together (cooperate) to solve complex problems that are beyond the capabilities of any of the individual computing elements. As such there is not one Grid, but several Grid systems, proposed by several industrial and research initiatives. Web services are a representation of Grid services through the specification of a WSDL (Web Service Description Language) [Curbera et al. 2002] interface.

Grid and Agent research are therefore related, and in more than one way. The AgentLink forums recognize the evolution of Grid research towards autonomous and adaptive Grids and of agent research towards large-scale robust and realistic applications. More specifically, three AgentLink forums have investigated this relation between agent and Grid systems, namely the forums on networked agents, on semantic web agents, and on multiagent resource allocation. One of the main findings of the networked agents working group is the emergence of communication, negotiation and coordination techniques in e-business, Grid, and semantic web applications. Moreover, more than half of the survey's respondents indicate their interest in integrating agent technology in Web and Grid services systems and are interested in Web services standards. On the other hand, current Grid systems also require automated and adaptive interactions between parties with heterogeneous information needs. In Grid systems, resources are not always available, with the need for on-demand provision of resources according to dynamic requirements. The effective allocation of limited resources in an open environment to satisfy customer requirements can be complex. Thus, resource management systems are changing from localized resources and services towards virtual organizations (VOs) sharing millions of

heterogeneous resources across multiple organisations and domains. The virtual organizations and usage models include a variety of owners and consumers with different usage, access policies, cost models, varying loads, requirements, and availability, requiring the emergence of an autonomous and adaptive Grid.

More specifically, the existence of multiple parties with different requirements implies that interaction and cooperation between them are crucial for sharing and trading resources. However, online trading between Web services for resources remains very crude in current Grid systems. Service provisioning may also include terms about performance levels and penalties. In current Grid applications, heterogeneity and dynamic provisioning are limited, and dynamic virtual organizations are restricted to those parties with a priori agreements to common policies and practice. Given that multi-agent systems can interact and adapt in dynamic environments through cooperation, coordination, and negotiation to satisfy their individual or common goals, Grid and agent technology can complement each other for efficient provisioning and management of services. The AgentLink networked agents forum identified this opportunity to adapt agent theories and experiences into the domains of Web services, the semantic web, Grid, systems and e-Business and investigate large-scale open environments populated by agents. The Ontogrid project [Ontogrid Project 2005] is also investigating how to share and deploy knowledge in Grid computing and the semantic Grid architecture that results from this project would stand as a methodology for developing Grid systems that optimize cross-process, cross-company, and cross-industry collaboration. In this article, we focus on the negotiation component of the Ontogrid project, which facilitates an iterative Contract Net Protocol (CNP) [Smith 1981] negotiation between autonomous, adaptive and rational entities, represented as Web services, with possibly conflicting goals. Negotiation mechanisms between Web services allow the latter to reach agreements such as for service provision and can thus bring a degree of autonomy and adaptability to Grid systems where interactions are currently inflexible. Thus our mechanisms address the current limitation of Grid applications for supporting adaptive negotiation strategies for task and resource allocation by starting in this article with an iterative Contract Net Protocol deployment between Web services, and evolving towards computational auctions in future work. Our aim is to deploy negotiation and interaction mechanisms between Grid services in the same way that autonomous agents negotiate in a collaborative or competitive distributed system.

The remainder of this article is structured as follows. Section 2 gives an overview of the relation between multi-agent systems and Web services. Section 3 describes an insurance scenario in which we are deploying our negotiation service. Section 4 presents a Contract Net Protocol (CNP), including an iterative CNP, between Web services for negotiation. Section 5 specifies our representation of the preferences and decision making used in the Contract Net Protocol. Section 6 details the evaluation of the Contract Net Protocol service (CNP service). Section 7 discusses related work. Section 8 concludes and gives an overview of our future work.

2. AUTONOMOUS AND ADAPTIVE SYSTEMS FOR WEB SERVICES

In this section, we draw out the relation between Grid services and networked agents, to highlight the niche for autonomy and adaptability in Grid systems. This relation is the creation of virtual organisations and cooperation within these virtual organizations. In the case of the Grid, traditional centralised methods needing complete information for system-wide optimization of performance are not enough [Foster and Kesselman 2003]. Nowadays, there is a drive towards service-oriented and virtual organizations architectures that can support a broad range of commercial applications [Firth 2003; Kelly et al. 2005].

2.1 Agents and the Grid

The AgentLink technical forum on networked agents conducted a survey on the theory and practice of agent system deployment in organizations engaged in agent research and deployment. The survey consists of agent technology providers' and consumers' points of view. We analyze the survey for indications of adoption of Grid and Web services technology by the agent community and vice versa. Overall, 43% of respondents indicated that online services are relevant to agent technologies. Agent technology brings highest benefits to highly distributed, multiorganizational nature, Web and complex information exchange applications.

One of the findings is that 60–70% of providers and consumers are involved in the combination of agents with other technologies and in runtime/platform tools. Specifically, Web services, XML and SOAP (Simple Object Access Protocol) [Curbera et al. 2002] tools are listed as those that are already being used for significant deployment in production environments. The tools that developers work with include Web services development tools such as Axis, Tomcat, Globus toolkit [Sotomayor and Childer 2006], IBM Websphere, SOAP, and WSDL. As for the application of nonagent systems for agent purposes, the nonagent technologies listed include Web services tools, semantic web tools and Globus toolkit. In fact, 53% of all respondents are investigating the extension of existing nonagent systems with agent functionality, of which 60% are concerned with Web services and 47% with the semantic Web. In the 12 months following the survey, the most likely investigation in the combination of nonagent technologies with agent technology lies 57% in the Web services sector, 53% in the semantic web and 40% in grid technologies. These figures show the high interest in integrating the work from the agent and the Web and Grid services communities.

Moreover, larger companies were more involved in Internet, Web services, and IT technologies than in Artificial Intelligence. Although the participants identified simulations on the Grid and advance Web services as areas with strong potential for application of agent technology, they also indicated that agent technology is not yet ready for deployment in open systems, where security, scalability, and management are still a challenge. The barriers to applying agent technologies include immaturity of basic agent tools, lack of robustness, scalability, large-scale distributed deployment, and inability to interoperate with current tools. The Web and Grid community can help overcome these

barriers by virtue of their large-scale, robust and mature tools and applications. It was also noted that communication, negotiation, and coordination are main techniques for agent deployment in Web agents and VOs. Technology providers responded that compliance to standards is an important factor—30% indicated their highest priority being Web services standards and 17% for Web services orchestration standards. Finally, the sectors in which agent technology will see the highest take up include the insurance industry and Internet applications, and this relates to our insurance use case in this article.

The "multi-agent resource allocation (MARA)" AgentLink forum also investigated Grid computing as one of its four major application areas for multi-agent resource allocation. This forum's report [Chevaleyre et al. 2006] states that the Grid domain is one of the most pressing applications for agent resource allocation techniques. Research in MARA is investigating the evolution of the Grid from centralized systems seeking optimal allocations towards the Grid as a commodity, where resource allocation methods are integrated with payment systems. This group suggests combinatorial auctions for centralized systems and bilateral negotiations between service consumers and providers in distributed systems. Since both types of negotiations have drawbacks such as computation costs and uncertainty in evaluation, a market-based allocation is proposed as a possible resource allocation strategy to suit the heterogeneous nature of the Grid.

2.2 Agents and Grid Cooperation Life Cycle

The agent's cooperation life cycle [Wooldridge 2002] in Figure 1 aims to characterize the different types of cooperation that may occur in problem-solving systems, and the way in which these types of cooperation and agreement relate to one another. There is a striking similarity between the cooperation life cycle and the operation of systems in the Grid. The premise is that there exist agent and Grid systems composed of multiple agents or Grid processes respectively—agent behavior is dictated by resources, capabilities, and goals, whereas for a Grid process, we speak in terms of resources, services, and constraints.

The first stage of the cooperation life cycle, shown in Figure 1, involves coalition and virtual organization formation. Coalition formation is the process of agents deciding who they want to work with. The output of the coalition formation process will be a coalition structure, that is, a partition of the set of all agents into sets of agents that have some commitment to work together [Sandholm 1999]. Similarly, virtual organizations are dynamic collections of individuals, institutions, and resources for flexible, secure, coordinated resource sharing. VOs enable disparate groups of organizations and/or individuals to share resources in a controlled fashion, so that members may collaborate to achieve a shared goal [Foster et al. 2001].

The second stage of the cooperation life cycle is team formation. At this stage, an unstructured set of agents is transformed into a team by agreeing on who in the coalition does which task with which resources. For example, in multi-agent planning, the agents develop detailed plans of activity to be executed subsequently. In the Grid context, when the VO receives a requestor's call, a

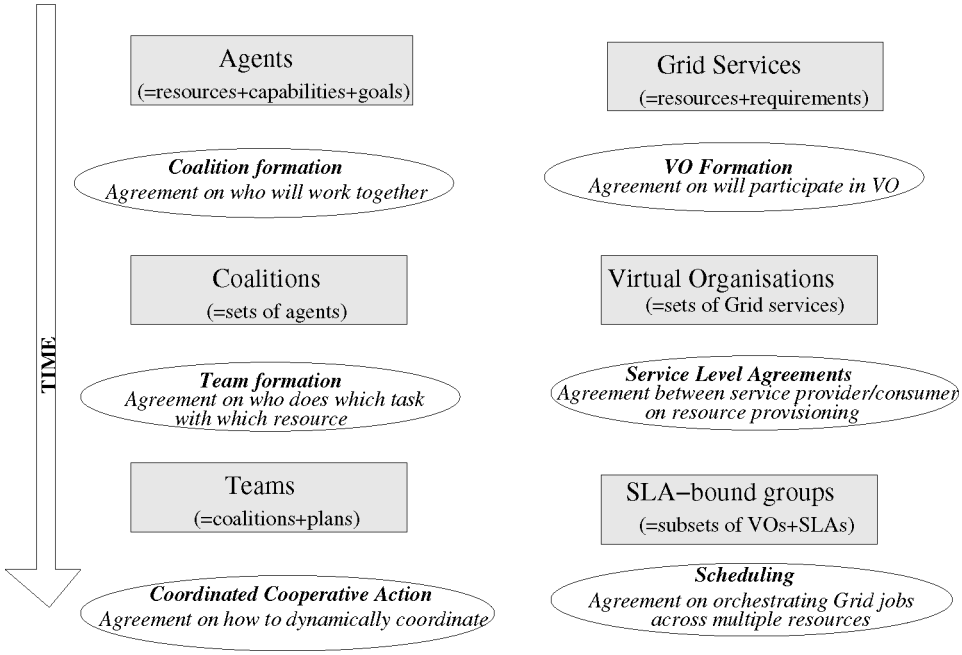

Fig. 1. Parallels between types of agreement in the agent and Grid cooperation life cycle.

subset of that VO is selected to satisfy the request and there is a service level agreement (SLA) with selected resource providers in the VO for resource provisioning. SLAs state the terms of the agreements between the consumer and the provider as a contract for the provider to perform a task or to provide agreed resources. The third stage is coordinated cooperative action and scheduling. This involves expediting the various tasks that agents and the VO members have agreed to carry out, ensuring that their activities are correctly coordinated. The coordination problem is that of managing interdependencies between the activities of agents and dynamic resource sharing between Grid processes such as those used in scheduling techniques. The goal is to efficiently expedite the tasks with the right resources.

2.3 Automated Negotiation for an Adaptive Grid

Owning and using Grid resources may be expensive, and resource owners would prefer to charge for their resources instead of the free sharing and provision of resources. After service discovery, there are various ways that a requester entity might engage and use a Web service and there is typically then an exchange of messages with the provider agent. However current Web services are inflexible and do not allow a party to adapt to other stakeholders' preferences. In reality, consumers and providers often have differing preferences and one-shot interactions, as current in the Grid, mostly result in rejections, and if repeated are costly and time-consuming. Thus, negotiation is necessary if the different stakeholders are to reach an agreement within their deadlines. For instance, the premises of a negotiation for each party could be as follows:

—A consumer has requirements for trading for computing or storage resources and chooses the best resource provider.

—A consumer can affect resource behaviour by changing its requirements and private preferences.

—A consumer often requires assurances or guarantees concerning the level and type of service being provided.

—In constrast, a service provider has advertised its service and is ready to meet requests that comply with the service interface. The service provider controls how much service information is exposed to the client, has to deal with multiple requests and reserve its resources optimally.

—A service provider's resources may not be readily available and the provider may negotiate about service provision for a personalised quote to the requester.

—A service provider wants to maintain local control over the use of its resources.

Reconciling resource consumer and provider preferences and constraints requires both parties to automatically adapt their preferences to resource availability. Negotiating SLAs can achieve such autonomy and flexibility. Agreements on resource provisioning may include not only the provider's commitment to execute a task or provide the resources, but also include terms about performance levels and penalties. Thus, negotiation may take place and may require several rounds of interaction between a consumer and a provider to determine whether and how the service provider can fulfill the request, to allow the consumer to change its requirements and finally for both to agree on a SLA. Negotiation in many Grid systems must be fast and the provider may need to satisfy thousands of requests simultaneously. Thus negotiations in the Grid have to be automated and adaptive to cope with heterogeneous environments and dynamic preferences and resource availability. Thus, we believe the ability to engage in automated negotiation could help Grid processes to be autonomous and adaptive, and to better manage their activities and resources at runtime, with a reduced requirement for design time coordination and allocation regimes. Below, we list some well-known agent negotiation mechanisms which can be used for cooperation and agreement between Grid entities.

—*Brokering algorithms for dynamic coalition formation*. For example, a VO manager acting for its VO members. [Decker et al. 1997; Czajkowski et al. 2001].

—*English auction with timeouts for resource allocation*. Computational auctions may facilitate the efficient allocation of scarce resources to Grid entities, based on economic principles [Vulkan and Jennings 2000; Sandholm and Vulkan 1999].

—*Contract net protocol for task allocation* [Sandholm 1993].

—*Bilateral alternating offers protocol for bargaining* [Kraus 2001; Rosenschein and Zlotkin 1994].

In this article, we implement an iterative form of Contract Net Protocol (CNP) between Web services for task allocation (Section 4). In the CNP, there are

two types of roles—manager and contractor. The manager has a task to be achieved, and aims to outsource this task to contractors. The manager initiates the negotiation process through a call for proposals (cfp) announcing the task specification to the contractors. A contractor receiving the cfp evaluates it and decides whether to answer with a refusal or a proposal to execute the task. The manager receives the contractors' proposals and in turn decide which proposals to accept and which proposals to reject. Rejected contractors consider that the negotiation has terminated, while accepted contractors must expedite the task and send back the results of their work to the manager.

In the *iterative CNP*, the process of a manager invoking a cfp and a contractor submitting a proposal is repeated several times until either the manager decides to accept a proposal or the manager's deadline is reached. Thus there are several rounds of proposals in an iterative CNP, and the contractors aim to improve on their earlier proposals to be accepted by the manager. We adapt and deploy the Contract Net Protocol in its iterative form in a Web services domain, and more specifically in the insurance scenario described in the next section.

3. INSURANCE SCENARIO

The insurance field largely relies on traditional methods of claims handling. Every aspect of a claim will often be dealt with by a different specialized person working in a different department of an insurance company. Therefore, the process is very costly. The insurance market is, therefore, increasingly looking for ways to reduce the costs of handling claims. In Ontogrid, an *InsuranceGrid* (see `http://www.insurancegrid.org`) has been implemented for an imaginary company called DamageSecure, which is intended to demonstrate the value of Grid and agent technologies in this domain.

DamageSecure manages businesses involved in car damage claims on behalf of insurance companies. The goal of DamageSecure is to enhance the quality and efficiency of the total damage claims handling process among consumers, damage repair companies, and insurance companies. Every year around 100,000 damages are reported to DamageSecure, of which 40% are repairs and 60% replacements. If an InsuranceGrid can work without human intervention, it could potentially save 172M euros per annum [Smulders et al. 2005]. The various parties in the car repair claims handling have different preferences. When a car is damaged, a customer wants to get the car repaired as soon as possible. An insurance company needs to pay for the damages (claim settlement) under the best circumstances (lowest prices, highest quality, as soon as possible, as close as possible, etc). Our aim is have repair services representing repair companies negotiate for repairs. This has two advantages: (1) it is more efficient than the current claim settlement process, which is done by hand, and (2) repair prices from repair companies will drop, and the quality of repairs will increase.

We illustrate the interactions when handling claims for car repairs. In Figure 2, a customer is insured at a company offering insurance services. Car repair services are carried out by the damage repair company, and a surveyor may provide expert advice on the quality and cost of repairs. DamageSecure provides services for liaising between insurance settlement services. Before

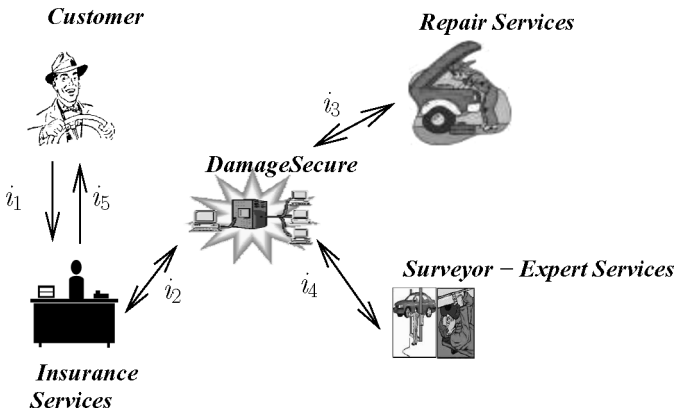

Fig. 2. Members and interactions in the InsuranceGrid VO.

any repair claims are received, the repair service can negotiate a contract with DamageSecure and insurance companies for bidding on repair jobs from insured customers. Customers buy services from insurance companies, which negotiate with the DamageSecure service on the best price and quality for their insured customers. After an accident, the following interactions to handle a claim occur between the insurance settlement services:

—i_1: A customer files a damage report with an insurance company.

—i_2: Based on a description of the damage, the insurance service requests DamageSecure to select the most appropriate repair service representing a repair company that will physically repair the car.

—i_3: Before any negotiation is carried out, DamageSecure selects a number of repair services based on the competences of the repair companies they represent. DamageSecure performs a CNP negotiation, possibly iterative, with selected repair services to find the best repair service to satisfy that damage report.

—i_4: DamageSecure can employ the services of an expert surveyor to analyze the quality of repairs on the damages and the charged costs.

—i_5: The insurance service pays the repair company, determines the liability, and processes the claim accordingly.

On analyzing the above scenario, it can be seen that negotiation arises at various points between the services. Before any specific accident and claim is handled, there are negotiations to draw up long-term contracts between the insurance company and DamageSecure and between DamageSecure and repair services. During the claims process itself, there is a negotiation between DamageSecure and repair services.

4. CONTRACT NET PROTOCOL BETWEEN WEB SERVICES

We refer to our deployment of the iterative CNP between Web services as the Contract Net Protocol service (CNP service). Although we describe the CNP service in the context of the insurance scenario, our CNP service is generic to

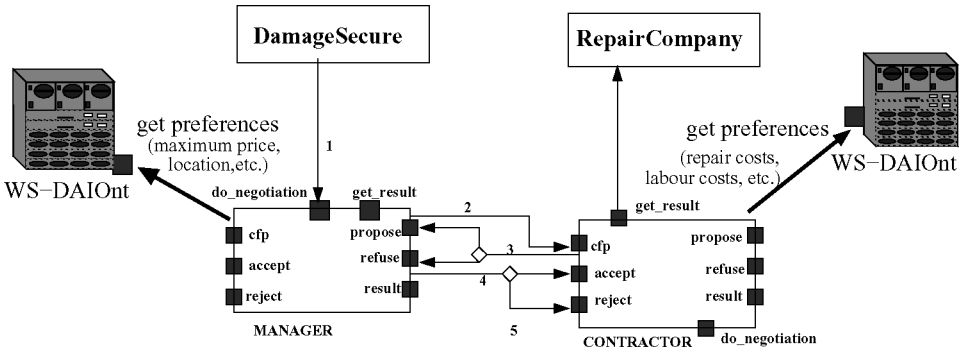

Fig. 3. Relationship of CNP services with the WS-DAIOnt Service.

any resource and task allocation negotiations between Web services. Thus, the negotiation illustrated in Figure 2 between DamageSecure (the manager) and repair companies (contractors) follows a Contract Net Protocol that could be an iterative CNP.

4.1 Contract Net Protocol API

We implement the protocol via method invocations, rather than by asynchronous message passing. The contractor and manager are services, and each exposes its interface. What in an agent context would be a message from a sender to a receiver, is implemented as a method provided by a receiver and invoked by a sender. For example, in an agent context, a manager sends a `cfp` to a contractor. In our Web services implementation, a contractor service publishes a `cfp` method as part of its interface and a manager service invokes the contractor's `cfp` method. The contractor's implementation of the `cfp` method will evaluate and choose the appropriate response to an invocation of its `cfp` method. Thus, the interfaces that manager and contractor services expose have to support methods analogous to messages sent and received in the CNP. The contractor service would expose and implement the methods: `cfp, accept, reject`. The manager service should support the methods: `propose, refuse, result`.

Figure 3 illustrates the methods implemented on the CNP services. The DamageSecure and Repair companies each implements a CNP service, and are shown in the figure as separate entities. The dark boxes on the CNP services represent the methods published by that CNP service and stands as the service's API. The back end of a CNP service implements the decision strategies. The arrows between DamageSecure, a repair company, and their CNP services show method invocations between them, and the number on the arrows show the sequence of the method invocations. DamageSecure as a client first invokes `do_negotiation` on its CNP service, which then invokes the `cfp` method on a repair company's CNP service. Thus the CNP protocol is performed according to the message sequence specified by the arrow numbers. In the iterative CNP, method invocations labeled 2 and 3 in Figure 3 are executed iteratively. Furthermore, the preferences described in Section 4.3 are retrieved from backend

databases through an ontology service interface called WS-DAIOnt (see Section 4.4). Given that an EPR (end point reference) contains the URI of a service, the API of a CNP service, specified as a WSDL file, exposes the following methods:

(1) `do_Negotiation(String context_Job, EPR[] contractor_list)`. A client wishing to launch a CNP between two Web services invokes this method on the Web service acting as the manager of the CNP. In the insurance scenario, the `do_Negotiation` method of DamageSecure service is invoked, thereby assigning it a manager's role. The input parameters is a URI to the damage report, which specifies the damages that need to be repaired. The second parameter is a list of URIs identifying repair services acting as contractors.

(2) `ResultNegBean get_result(String contextJob)`. This method is invoked to query the result of a negotiation with respect to that particular repair job—whether the CNP is still ongoing, and if terminated whether the result is an agreement, a rejection, or a refusal.

(3) `cfp(EPR m, CNP_negotiation_subject cfp_1)`. This method is invoked by a manager on a contractor and the manager passes the contractor 1) its URI m and, 2) the values in the `cfp` of type `CNP_negotiation_subject`. Negotiation subjects are described in more detail in Section 4.2.

(4) `propose(EPR c, CNP_negotiation_subject proposal)`. A contractor invokes a manager's `propose` method with parameters the contractor's URI c and the contractor's proposal values.

(5) `refuse(EPR c, CNP_negotiation_subject cfp_1)`. Instead of a proposal, a contractor may choose to send a refusal to a cfp with the original values it received from the manager in `cfp_1`. c is the address of the contractor.

(6) `accept(EPR m, CNP_negotiation_subject proposal)`. On receiving a number of proposals, the manager with EPR m chooses the which proposal to accept and reject.

(7) `reject(EPR m, CNP_negotiation_subject proposal)`. Contractors that are not chosen for acceptance have their `reject` method invoked by the manager.

(8) `public void acknowledge(EPR c, CNP_negotiation_subject agreement)`. Finally, an accepted contractor invokes the `acknowledge` method on the manager with the accepted proposal as an agreement.

Figure 4 gives an overview of the port-type of the CNP service for the `cfp`, `do_negotiation`, `propose` and `accept` methods. It is defined as a collection of operations exposed by the CNP service and abstractly represents the method definitions. Each operation specifies the types of messages that the CNP service can send or receive as part of that operation.

4.2 Negotiation Subject

The nature of agreements is dependent on type of resource being provided. We define a Java Bean called `CNP_negotiation_subject` to describe the properties of the resources being negotiated about in the insurance scenario. This Java Bean

```
<wsdl:portType name="CNP">
  <wsdl:operation name="do_Negotiation"
       parameterOrder="Context_Job contractor_list">
    <wsdl:input message="impl:do_NegotiationRequest"
       name="do_NegotiationRequest"/>
    <wsdl:output message="impl:do_NegotiationResponse"
       name="do_NegotiationResponse"/>
  </wsdl:operation>

  <wsdl:operation name="cfp" parameterOrder="M cfp_1">
    <wsdl:input message="impl:cfpIn" name="cfpIn"/>
    <wsdl:output message="impl:cfpOut" name="cfpOut"/>
  </wsdl:operation>

  <wsdl:operation name="propose" parameterOrder="C proposal">
    <wsdl:input message="impl:proposeIn" name="proposeIn"/>
    <wsdl:output message="impl:proposeOut" name="proposeOut"/>
  </wsdl:operation>

  <wsdl:operation name="accept" parameterOrder="M proposal">
    <wsdl:input message="impl:acceptIn" name="acceptIn"/>
    <wsdl:output message="impl:acceptOut" name="acceptOut"/>
  </wsdl:operation>
  <...>
</wsdl:portType>
```

Fig. 4. CNP service port-type.

holds the values of the various negotiable properties in the damage report. The properties that we consider for negotiation in the CNP are:

—*Price*. This is a function of the labor costs, repair costs, and profit margin calculated on the repair service side. On DamageSecure's side, price is what it is willing to pay for the repair job.
—*Location*. This is the distance of the car repairer from a particular insured client.
—*Type of material* (original or not). This represents the option of replacing car parts with original or second-hand parts.
—*Speed of repair*. This is how soon can the repair company perform the repairs.
—*Appointment date*. This is the date to leave or collect the car.
—*Method of repair* (repair or replace). After analyzing the damages, a repair service may propose to either repair or replace certain car parts.
—*Type of paint* (metallic or not). For some cars, metallic paint may be needed, and this would cost more than the normal car painting.
—*Quality of repair*. This is a qualititative attribute and can be defined by comparing with previous repairs or using expert services on car repairing.
—*Color*. This is the color of the car and is normally nonnegotiable.

4.3 Negotiation Preferences

Preferences are private to a negotiating party and are used by that party to calculate what values in the negotiation subjects it accepts or responds with. Normally preferences are parameterized resource attribute metrics describing particular range of acceptable values. We define a special class called `negotiation_preferences` to encapsulate the preferences of DamageSecure and repair services.

First we define a class `pref_terms` to have private attributes `preferred_value`, `reserve_value`, `utility`, and `weight`. The negotiation preferences also contains each property of the negotiation subject, but the properties are declared as type `pref_terms`. For example, the negotiation preferences can include as a private attribute (or issue) `price`, which is of type `pref_terms`, and thus price itself has attributes `preferred_value`, `reserve_value`, `utility`, and `weight`. An issue's `preferred_value` is the preferred (ideal) value of that issue for a contractor or manager. For example, a contractor would ideally like to receive £40 as the price for carrying out a task, but the actual value of the price that is being passed in the negotiation subject is £30. The `reserve_value` of an issue is the limit to which a participant is willing to concede. For example, if the reserve value of a contractor for price is £20, it means that he/she is not willing to receive less than £20 for performing a job. There is usually a relationship between the actual value, preferred value and reserve value, depending on whether that participant wants more or less of that issue. For a contractor preferring a high value for price, then the relation between its preferences is:

$$\textit{reserve value of price} < \textit{actual value of price} < \textit{preferred value of price}.$$

The utility of an issue is how much is it worth to a participant. A higher utility means a higher worth. The weight of an issue is its importance relative to the other attributes. For example if price has a weight of 0.5 and location has a weight of 0.2 means that price is a more important property than location.

4.4 Storing Contracts and Preferences in WS-DAIOnt

The above preferences are stored in the private databases of the various parties in the insurance scenario. There are a number of databases in the insurance scenario and these are queried at various points before and during the CNP negotiation to appropriately evaluate and generate cfps, proposals, acceptances, refusals, and rejections. Insurance companies have private databases containing insured customers and damage reports that still need to be processed. DamageSecure has a private database for storing previously drawn-up contracts between insurance companies and registered repair companies. A car repairer itself holds databases that contain the contracts between itself and insurance companies, and that contain its preferences regarding car repairs. An insurance company selects a damage report from its internal databases and delegates the repair to DamageSecure. Given a damage report, based on the type of damages and contracts between repair and insurance companies, DamageSecure will select a number of repair companies from the insurance companies database. Selected repair companies already have a contract with

the insurance company. During negotiation, a repair service will analyze the damages in detail and make a calculation of the repairs or replacements costs by using their "parts" internal databases.

As shown in Figure 3, these databases are stored and queried using the WS-DAIOnt service [Esteban Gutierrez et al. 2006]. The WS-DAIOnt service defines a framework for creating ontology access services in a Grid environment. WS-DAIOnt uses the standard Grid data access vocabulary, and extends the data access mechanisms with the patterns, properties, and behaviors needed for providing ontology access. The WS-DAIOnt specification and the accompanying realizations define the data access services needed for dealing with ontologies in Grid environments. Figure 3 illustrates the relationship between WS-DAIOnt and CNP services. The WS-DAIOnt service is queried about the preferences of DamageSecure and repair services during evaluation and generation of the values in a negotiation subject.

5. STRATEGIES FOR THE CNP SERVICE

There are a number of properties that negotiation strategies ideally seek to satisfy so that the parties interact productively and fairly [Kraus 2001]. Here we present strategies suitable for use in the CNP between Web services. In this protocol, there are several stages where either the contractor or the manager have to make a decision according to the negotiation subject and their own preferences. The evaluation of a cfp and a proposal, and the generation of a cfp and an acceptance are calculated according to the preferences of the manager or the contractor as applicable, towards the goal of reaching agreements. The utility functions guide the services in distinguishing favorable outcomes from unfavorable ones.

5.1 Evaluation and Generation of CFPs and Proposals

In this section, we briefly describe our decision making algorithms for the evaluation (evaluating a cfp and a proposal) and for the generation (generating a cfp, a proposal and an acceptance) of the negotiation subject. In all of these cases, the functions are calculated according to the preferences of the manager or the contractor as applicable. For example, the values of price, quantity, and delivery date passed in the `cfp` are a function of DamageSecure's preferred value for these issues, and within its reserve limits and of the deadline of the negotiation. There are three specific strategies used for generating a cfp, as described in Section 5.2.

For evaluating a `cfp`, a repair service's evaluation function is passed the values in the `cfp` and the repair service's preferences with respect to that cfp. These preferences are retrieved from queries made to the WS-DAIOnt service on the repair service's relevant databases. The evaluation function will return `true` if it is calculated that the cfp lies within a margin of the repair service's preferences, including its utilities. For a repair service i, the utility of the overall cfp, U^i_{cfp}, is calculated from equation 1 as the normalised weighted summation

of the utility of the individual issues V_j^i for that repair service.

$$U_{cfp}^i = \sum_{1 \le j \le n} \omega_j^i V_j^i(cfp[j]). \quad (1)$$

U_{cfp}^i must lie within the repair service's utility limit, which is itself calculated as a normalized summation of the reserve values of each issue for that service. Thus, even if the cfp is outside the preferences but within a said margin, subsequent negotiation can bring the proposal within the preferences.

For generating a proposal by a contractor repair service, each issue in the negotiation subject is possibly given a new value in the generated proposal so that the utility of the overall proposal is more than the utility of the overall cfp for a repair service. The generated value of an issue in a proposal is also a function of the value in the cfp and of the `preferred_value` in a repair service's preferences. It may be that the generated value lies outside the repair service's `reserve_value` because the value in the cfp lies significantly outside the `reserve_value`. In this case, the repair service proposes a value close to its `reserve_value` for that issue. We use the three strategies in Section 5.2 to generate a proposal.

DamageSecure evaluates whether to accept or reject the proposal by 1) choosing those proposals falling within DamageSecure's preferences because there are no further rounds of counter-proposals, and 2) choosing the proposal with the highest utility to DamageSecure according to Equation (1). DamageSecure can reiterate the cfp according to the best received proposal.

We also implement a second evaluation strategy called the cost endowment strategy to evaluate a cfp and a proposal. In the cost endowment strategy, a user's endowment denotes the financial resources available for that user in the system. Assume at time t, a contractor i is currently scheduled to carry out the set of tasks τ_i^t, and that it has an *endowment* of e_i. The negotiation starts when the cfp method with speech act $\tau(\mathtt{ts})$ is invoked on i. Let $c_i^t(\tau)$ denote the *cost* to contractor i of carrying out the set of tasks denoted by τ at time t. The *marginal cost* of carrying out tasks τ at time t for contractor i is denoted by $MC_i(\tau(\mathtt{ts}) \mid \tau_i^t)$:

$$MC_i\big(\tau(\mathtt{ts}) \mid \tau_i^t\big) = c_i\big(\tau(\mathtt{ts}) \cup \tau_i^t\big) - c_i\big(\tau_i^t\big).$$

The marginal cost for contractor i of agreeing to carry out the task is the difference between the cost of carrying out its newly allocated tasks $c_i(\tau(\mathtt{ts}) \cup \tau_i^t)$, and the cost of only carrying out its previously agreed tasks. If $MC_i(\tau(\mathtt{ts}) \mid \tau_i^t) = 0$, then the marginal (extra) cost of carrying out $\tau(\mathtt{ts})$ is zero: they can be done for "free." If $MC_i(\tau(\mathtt{ts}) \mid \tau_i^t) < e_i$, then the contractor can make a proposal; otherwise, it will not propose. If the payment for carrying out these tasks is $e(\mathtt{ts})$, then the decision as to whether to propose is whether $MC_i(\tau(\mathtt{ts}) \mid \tau_i^t) < (e(\mathtt{ts}) + e_i)$: if it is, then the contractor proposes, otherwise, it rejects the cfp.

5.2 Strategies for Generating a Negotiation Subject

We implement three strategies in the Contract Net Protocol for generating a cfp and a proposal. These are the truth telling, decrement, and time dependent strategies.

In the truth telling strategy, both the manager and contractors reveal their true preferences. Thus, DamageSecure constructs the cfp with its preferred value for each issue. A repair service replies with a proposal where each issue is given its own preferred value for each issue. If the cfp lies outside the reserve values for negotiable issues, then the repair service's proposal is grounded with the repair service's reserve values.

In the decrement strategy, the participants have evaluation and generation margins, against which they evaluate a cfp and generate a proposal above or below their reserve values. Thus the parties have chance to converge to an agreement during the negotiation process instead of rejecting immediately.

In the CNP, there is a deadline for receiving proposals from contractors. In the time dependent strategy, a proposal is computed as a function of the proportion of the remaining time over the total time allocated to the CNP. A contractor also has its personal deadline and knows the manager's deadline for receiving proposal. As the time left decreases, the concession rate of a participant increases. The proposal $proposal_r^t[j]$ of a repair service r at time t with deadline t_{max} is calculated in Equations (2) and (3) for each issue j. Let $pref_j$ and res_j respectively denote the preferred and reserve value for issue j in the service's preferences.

$$proposal_r^t[j] = res_j + \frac{min(t, t_{max})}{t_{max}}(pref_j - res_j) \text{ if } r \text{ prefers high } j \tag{2}$$

$$proposal_r^t[j] = pref_j + (1 - \frac{min(t, t_{max})}{t_{max}}(res_j - pref_j)) \text{ if } r \text{ prefers low } j. \tag{3}$$

6. EVALUATION OF THE CNP SERVICE

We deployed the CNP service, which implements the iterative CNP, over two frameworks: 1) Apache Axis and Tomcat [Brittain and Darwin 2003], and 2) Globus Toolkit 4 (GT4) [Sotomayor and Childer 2006]. Both platforms are popular for service deployments and are free. We conduct our evaluation over the GT4 platform. We evaluate the iterative CNP service by varying the strategies, number of repair services, and deadlines. We then measure the time taken to finish the CNP and the utility of the agreements, which is calculated from Equation (1). The iterative CNP negotiation is performed between DamageSecure and a number of repair services. We compare the performance of the CNP service with the case of no negotiations between Web services.

Figures 5 and 6 show the utility of the proposals for DamageSecure, and Figures 7 and 8 show the utility of the proposals for an accepted repair service, with respect to time in milliseconds. For both parties, using negotiation yields a higher utility for the end agreement than when there is no negotiation. As the negotiation progresses over time, the quality of proposals are of better utility than when there is no negotiation. A repair service stands to lose more if there is no negotiation since it has to accept DamageSecure's advertised cfp. As time elapses, the utility of proposals received by DamageSecure increases according to DamageSecure preferences, while the utility of proposals sent by repair services decreases. This is because repair services have to concede and offer better

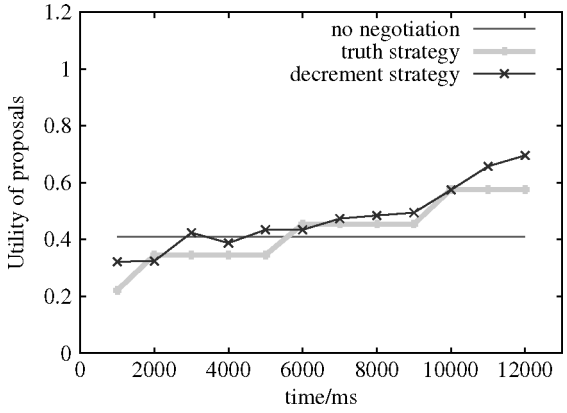

Fig. 5. Utility of messages from a DamageSecure's point of view for truth and cost decrement strategies.

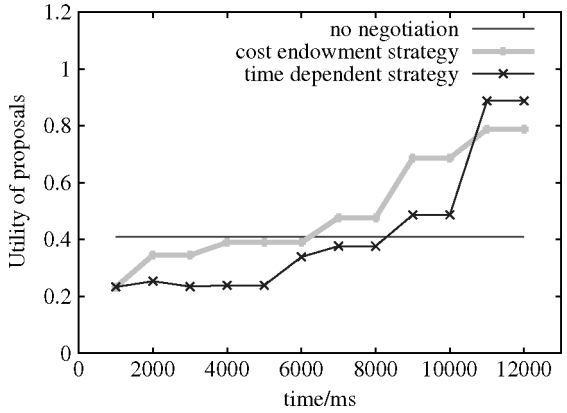

Fig. 6. Utility of messages from a DamageSecure's point of view for time and cost endowment strategies.

proposals as the negotiation progresses. DamageSecure also has to concede in its cfp such that there is a convergence between the cfps and the proposals, as shown in Figures 9 and 10.

For both DamageSecure and a repair service, the order of performance of the strategies from least to the best performing strategy is as follows:

$$\textit{no negotiation} < \textit{truth telling} < \textit{decrement} < \textit{time dependent}$$

Furthermore, in the time-dependent strategy, the concession rate increases as the deadline approaches such that there is a abrupt concession by a repair service just before the CNP's deadline.

Figures 11 and 12 show the utility of the agreements reached with varying numbers of contractors. Again, the time dependent and cost endowment strategies perform the best and the least utility is obtained when there is no negotiation. A greater number of repair services implies more competition between them and thus more competitive proposals and a better deal for DamageSecure.

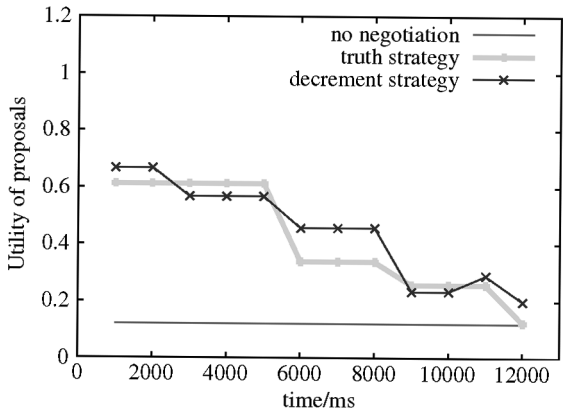

Fig. 7. Utility of messages from the accepted repair service's point of view for truth and decrement strategies.

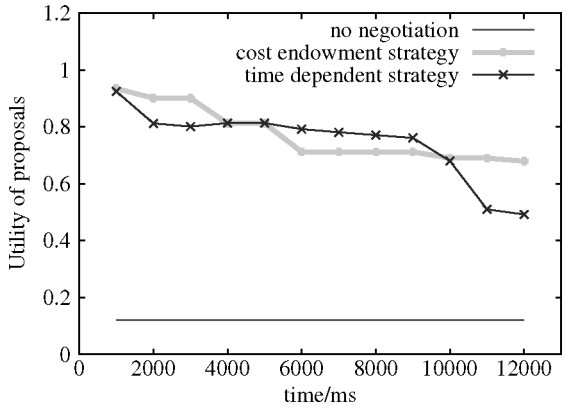

Fig. 8. Utility of messages from the accepted repair service's point of view for time and cost strategies.

Thus a low number of repair services means that those repair services form a monopoly and the no negotiation policy performs better than the truth telling and decrement strategy for DamageSecure. However, if the number of repair services is greater than 2, then DamageSecure obtains a better deal with negotiation and the utility of this deal increases with the increase in competition between repair services.

7. RELATED WORK

In this section, we discuss existing work on deploying negotiation between Web services. Traditional negotiation approaches in agent mediated electronic commerce [Rosenschein and Zlotkin 1994; Kraus 2001; Sandholm 1999] usually involve predetermined negotiating agents, and the negotiation protocols are hardcoded within the agents. The negotiation protocols are tailored for particular agent interactions and their accompanying strategies. As the AgentLink networked agent survey indicates, agent technologies are moving

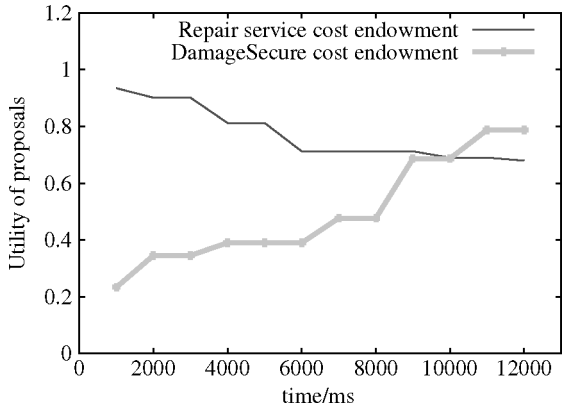

Fig. 9. Convergence towards an agreement for the cost endowment strategy.

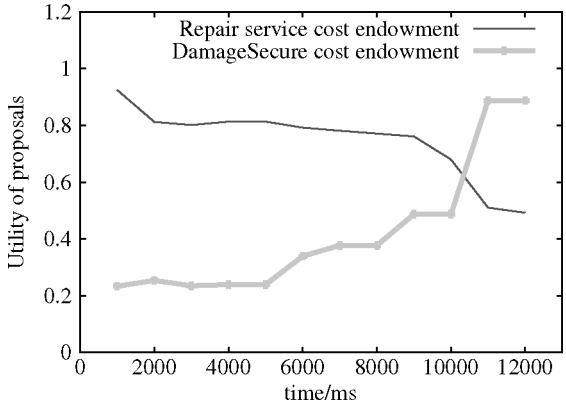

Fig. 10. Convergence towards an agreement for the time dependent strategy.

towards open negotiation environments where agents can freely enter and leave multi-agent systems, teams or virtual organizations. We introduce a degree of interoperability in this article through Web services advertising their negotiation interfaces in WSDL and through the WS-DAIOnt service to obtain the negotiation preferences.

GRAAP WS-Agreement [Andrieux et al. 2004] specifies an XML-based language for creating contracts, agreements, and guarantees from offers between a service provider and a client. An agreement may involve multiple services and includes fields for the parties, references to prior agreements, service definitions, and guarantee terms. However, WS-Agreement does not support negotiation and considers negotiation outside its scope of work. Moreover, WS-Agreement messages are limited to two types—`offer` and `agree`—and are constrained according to a template a service provider publishes. The WS-Agreement specification is used only at the last stage in a transaction where the parties close their interaction with a contract specified as a WS-Agreement. Even with a more varied set of messages, WS-Agreement still suffers from the lack of an interaction protocol specified between parties. Without an adequate

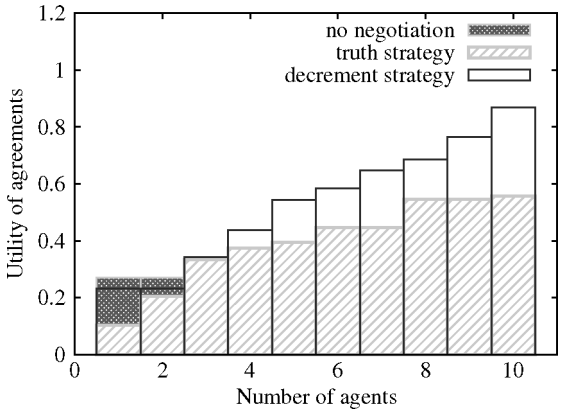

Fig. 11. Utility of messages from a DamageSecure's point of view for truth and decrement strategies with varying number of repair services.

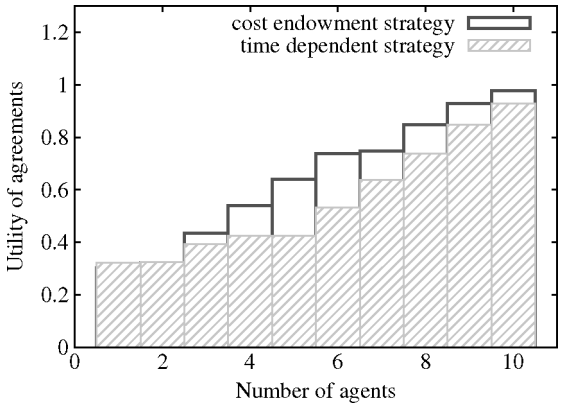

Fig. 12. Utility of messages from a DamageSecure's point of view for time and cost endowment strategies with varying number of repair services.

set of speech acts and specification of how to construct interaction protocols, the usefulness of a WS-Agreement exchange is limited to cases such as buying from catalogues, with take-it-or-leave-it offers from the seller or buyer. Aiello et al. [2005] propose to group the WS-Agreement states into tuples and specify state transitions. However, proposed combinations of these states are incomplete and inconsistent. For example, the overall state of an agreement is unclear when n terms in the agreement are fulfilled and m terms are violated. In their approach, the lifecycle of an agreement contains endless loops because whenever an agreement is fulfilled it is checked again.

Paurobally and Jennings [2005] propose an extension to the WS-Agreement and WS-Conversation languages (WSCL) with XML-based structures defining standard speech act–like messages and transitions as in interaction protocols. Essentially, this extension models FIPA-like speech acts as WS-Agreement schemas and interaction protocols in WSCL. The Contract Net Protocol is expressed in this WS-Agreement and WSCL extension and in statecharts as a

visual tool. The work in this article is different from the work in Paurobally et al. [2005] and Aiello et al. [2005] in that these latter works remain at the level of communication languages and interaction protocols without dealing with the negotiation subject and strategies, and their implementation. Their work is not concerned with the negotiation subject, issues and preferences, whilst this is one of the main concepts in this article. Our article here has no relation with WS-Agreement, and we are not proposing a standardisation of agent communication languages or interaction protocols for Web services. Instead we are focusing on implementing actual Web services negotiations and their strategies for Grid services to adapt to varying stakeholders' preferences.

Ayienga et al. [2004] use agent-mediated electronic commerce techniques for the provision of QoS at the network level in a Grid environment. They address the problem faced by the Internet for supporting Grid applications requiring high bandwidth and low latency. The negotiation component has not been implemented yet, and it would be interesting to see how negotiation decision functions are actually specified and implemented in this dynamic environments.

8. CONCLUSIONS

The aim of our work in the context of the Ontogrid project is to develop semantic web services that can cooperate, reach agreements and self-organize in order to effectively solve overall problems. Having analyzed existing standards for enabling negotiation between Web services, we found that there is a need for Web services frameworks that facilitate trading and negotiation between Web and Grid services. In this article, we have discussed this need, we have drawn out the relation between Grid services and agent systems, and we have identified the opportunity to reuse agent interaction and negotiation techniques in the Web services domain. The survey conducted by the AgentLink networked agents working group also identifies this opportunity for the combination of agents with other technologies. More specifically 60-70% of agent technlogy providers and consumers have high interests in Web services, XML and SOAP tools.

Thus, our work in this article lies in view of this growing integration of agent and Web services technologies. More specifically, we have described our deployment of the Contract Net Protocol and its associated strategies for negotiation between Web services. We have applied and evaluated our approach in the context of a real-life insurance application. It is salient to note that our proposed Contract Net Protocol API and decision strategies are generic and are not bound to the insurance application. Thus, our approach can also be applied to a variety of other Grid services applications. Future work includes deploying other negotiation protocols such as English auctions and the bilateral protocol.

REFERENCES

AIELLO, M., FRANKOVA, G., AND MALFATTI, D. 2005. What's in an agreement? An analysis and an extension of WS-Agreement. In *Proceedings of International Conference on Service-Oriented Computing (ICSOC)*. Lecture Notes in Computer Science, vol. 3826, Springer, Berlin, Germany, 424–436.

ANDRIEUX, A., CZAJKOWSKI, K., AND DAN, A. 2004. Web Services Agreement Specification (WS-Agreement). World-Wide-Web Consortium (W3C), http://www.w3.org/ Web document http://www.gridforum.org/Meetings/GGF11/Documents/draft-ggf-graap-agreement.pdf.

AYIENGA, E., MANDERICK, B., OKELLO, O., AND NOWE, A. 2004. Multi-agent systems for efficient quality of service routing in grids. *ERCIM News 59*, 39–42.

BRITTAIN, J. AND DARWIN, I. 2003. *Tomcat: The Definitive Guide*. O'Reilly, Cambridge, MA.

CHEVALEYRE, Y., DUNNE, P. E., ENDRISS, U., LANG, J., LEMAOTRE, M., MAUDET, N., PADGET, J., PHELPS, S., RODRIGUEZ-AGUILAR, J. A., AND SOUSA, P. 2006. Issues in multiagent resource allocation. *Informatica 30,* 1, 3–31.

CURBERA, F., DUFTLER, M., KHALAF, R., NAGY, W., MUKHI, N., AND WEERAWARANA, S. 2002. Unraveling the Web services Web: An introduction to SOAP, WSDL and UDDI. *IEEE Inter. Comput. 6*, 86–93.

CZAJKOWSKI, K., FITZGERALD, S., FOSTER, I., AND KESSELMAN, C. 2001. Grid information services for distributed resource sharing. In *Proceedings 10th IEEE Symposium on High Performance Distributed Computing.* IEEE Computer Society, Washington, DC, 181–190.

DECKER, K., SYCARA, K., AND WILLIAMSON, M. 1997. Middle-agents for the Internet. In *15th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, Burlington, VL, 578–683.

ESTEBAN GUTIERREZ, M., GOMEZ-PEREZ, A., AND MUNOZ GARCIA, O. 2006. ontology access in grids with WS-DAIOnt and the RDF(S) realization (poster). In *3rd European Semantic Web Conference (ESWC06)*. Lecture Notes in Computer Science, vol. 4011, Springer, Berlin, Germany.

FIRTH, S. 2003. The future is grid. Hewlett-Packard Labs, `http://www.hpl.hp.com/news/2003/oct_dec/grid.html`.

FOSTER, I., JENNINGS, N., AND KESSELMAN, C. 2004. Brain meets brawn: Why grid and agents need each other. In *3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems.* IEEE Computer Society Press, 8–15.

FOSTER, I. AND KESSELMAN, C. 2003. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, Burlington, VT.

FOSTER, I., KESSELMAN, C., AND TUECKE, S. 2001. The anatomy of the grid: Enabling scalable virtual organizations. *Inter. J. Supercomput. Appl. 15,* 3, 200–222.

KELLY, N., JITHESH, P., DONACHY, P., HARMER, T., PERROTT, R. MCCURLEY, M., TOWNSLEY, M., JOHNSTON, J., AND MCKEE, S. 2005. Genegrid: A commercial grid service oriented virtual bioinformatics laboratory. In *Proceedings of the IEEE Conference on Services Computing*. IEEE Computer Society, Washington, DC, 43–50.

KRAUS, S. 2001. *Strategic Negotiation in Multiaagent Environments*. MIT Press, Cambridge, MA.

ONTOGRID PROJECT. 2005. Paving the way for knowledgeable grid services and systems. EU FP6 project (FP6-511513), http://www.ontogrid.net/.

PAUROBALLY, S. AND JENNINGS, N. 2005. Protocol engineering for Web service conversations. *Engine. Appl. Art. Intell.* (Special Issue on Agent-Oriented Software Development) *18,* 2, 237–254.

ROSENSCHEIN, J. AND ZLOTKIN, G. 1994. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press, Cambridge, MA.

SANDHOLM, T. 1993. An implementation of the Contract Net Protocol based on marginal cost calculations. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*. The AAAI Press/The MIT Press, Cambridge, MA, 295–308.

SANDHOLM, T. 1999. Distributed rational decision making. In *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence*, G. Weiss, Ed. MIT Press, Cambridge, MA, 201–258.

SANDHOLM, T. AND VULKAN, N. 1999. Bargaining with deadlines. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*. AAAI Press/The MIT Press, Cambridge, MA, 44–51.

SMITH, R. G. 1981. The Contract Net Protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput. C-29,* 12, 1104–1113.

SMULDERS, J., VAN AART, C., VAN HAPERT, P., FINTELMAN, V., AND STORMS, P. 2005. Ontogrid, Deliverable 9.1, Businesss Cases and User Requirement Analysis. EU FP6 project (FP6-511513), http://www.ontogrid.net/.

SOTOMAYOR, B. AND CHILDER, L. 2006. *Globus Toolkit: Programming Java Services*. Morgan Kaufmann, Burlington, VT.

VULKAN, N. AND JENNINGS, N. R. 2000. Efficient mechanisms for the supply of services in multi-agent environments. *Int. J. Decis. Sup. Syst. 28,* 1-2, 5–19.
WOOLDRIDGE, M. 2002. *An Introduction to Multiagent Systems*. John Wiley and Sons Ltd.

Received August 2006; revised May 2007; accepted August 2007