Automated negotiation for service contracts

Russell Lock
Computing department
Lancaster University
r.lock@comp.lancs.ac.uk

Abstract

Automated negotiation draws upon research from a number of different computing disciplines, predominantly those of Game Theory, AI, Requirement specification & Authorisation research. Automated negotiation allows clients / services to come to agreements regarding service utilisation. A number of problems exist within the area, primarily those of requirement elicitation and trust. These problems can be minimised through standardisation and careful design; however, human participation in the process cannot be completely removed. This paper examines a possible format, architecture and implementation (TRANSACT) to aid in the automated negotiation of service contracts based on exogenously stated requirements / capabilities. In doing so it explores the issues and areas in which further developments are required to support future service developments.

Keywords
Negotiation, Contracts, Authorisation, Quality Management, SOA, eCommerce, Security

1. Introduction

The ongoing adoption and commercialisation of web and grid Service Oriented Architectures (SOA) has increased the pressure to develop new high level service support. With more business cases for usage being developed it is becoming clear that it is the higher level functionality that now needs addressing.

Whilst rudimentary mechanisms exist for dynamic service discovery in the form of technologies including UDDI, it is clear that service negotiation and monitoring have been relatively neglected. It is also clear that the service vision cannot be implemented satisfactorily if negotiation for use of a given resource has to occur "out of bounds". Given service negotiation, and agreement on a given contract/SLA it is equally important that some way of monitoring the adherence to a given contract/SLA is possible, though this step is outside the purview of this paper. Research in this area includes that done by the SECSE consortium[(1)] who favour a faceted approach to the service domain.

Previous attempts to translate heavily worded legal documents into a workable electronic format capable of adaptation to specific circumstances have met with limited success. The ALDUS project (1990) ALDUS[(2)] was enacted to examine possible areas of automation specifically with mind to the creation of sales contracts. It examined the issues present in three key areas; identification of stakeholders and budgets; the functionality possible with current and envisaged tools; and the need of users with regard to contracts. At the conclusion of the project, however, it was decided that there was no viable economically feasible products upon which such tools could be built at the time. Fortunately this situation is changing and a number of new projects are starting to address the problem in depth.

The remainder of this paper is as follows; section two examines service negotiation issues and existing research in the area. Section three outlines the design and implementation of a possible solution to the negotiation problem. Section four provides information regarding the evaluation of the approach. Finally, sections five and six provide information on further work and conclusions respectively.

2. Service Negotiation

The levels of service development / integration seen presently generalise to two different scenarios:

Not for profit Services – Services available without a cost specific to their invocation. These are more commonly seen in scientific collaborations and in the development and evaluation of new products. This model is suitable in business situations only where client membership remains within a single administrative and organisational boundary; for example, as an internal service to employees. Quality of Service guarantees for free services are uncommon, and if they do appear are unlikely to be backed by financial obligation. Generally these services are characterised by a best effort level of service provision, which could prove insufficient for many envisaged VO collaborations [(3)].

Economically services – A more feasible business model but brings with it additional concerns. An economic model requires attention to the following infrastructure considerations:

- Service discovery
  - The discovery and active differentiation of services
- Service negotiation
  - The requirement specification and capability specification respectively. The determination of a compromise situation between the two parties.
- Service Agreement
  - The signing of documents to guarantee service level attributes. This could involve a contract / SLA
- Service mediation
  - The specification of standardised complaint and renegotiation policy
- Service monitoring
  - The monitoring of service use, based on data provided by a combination of client, service and possibly trusted 3rd parties [(4)].

Traditionally service contracts are encountered in two forms, mainly dependent on the size and cost of a given contract.

- Standardised contracts based on a service classification. For example, Gold, Silver, Bronze etc. In these situations no customisation occurs, the possibilities for service requirement-capability matching is severely limited.
- Manual service negotiation, through meetings between client, service and legal aid. This model is most applicable to the VO vision[(5)] as the negotiation can dynamically determine the most accurate compromise position possible. However, such processes clash with the need to maintain agility in the business process. It is this type of service negotiation that TRANSACT aims to emulate.

2.1 Automated Negotiation Types

Automated negotiation for the purposes of this paper refers to an aid to decision making, rather than entirely autonomic service utilisation. Automation in this domain is generally split into two types:

EBA (Electronic Bargaining Agents)

Agents which attempt to develop compromise positions for services and clients autonomously.

NSS (Negotiation Support Systems)

Support systems which provide information regarding the negotiation process, but which do not act in an autonomous manner on behalf of a client.

TRANSACT aims to provide more than an NSS by automating the majority of the negotiating process; the client however, retains the final decision on service choice. Compromise seeking programs come in two flavours:

- Distributive, where values are negotiated within a fixed cost boundary;
- Integrative where the price can also expand or contract throughout the negotiation process.

TRANSACT follows an integrative path, which is more difficult to control, but is considerably more flexible as it allows the two parties to explore a larger proportion of the problem space whilst negotiating.

2.2 External Issues

The scope of this particular project cannot hope to provide solutions to all steps of the service cycle outlined in section 2. Fortunately, research in certain areas by others is already well advanced. Dynamic binding to services in a VO (Virtual Organisation) context requires careful consideration of service monitoring. Beyond simple provisioning of service monitoring using either centralised third party monitors or decentralised p2p technologies, a number of more complex monitoring issues remain. For example, services cannot be assumed atomic, and the assumption that composite services may make use of sub-services, not necessarily within the same geographic or organisational boundary, entails a number of problems and issues relating to trust management and monitoring. This piece of research however, has put aside this particular issue in order to make more progress in the core areas of service negotiation.

There is also an ongoing issue relating to the standardisation of terms for use in contracts, and web services in general. Contributions toward greater levels of standardisation with regard to service orientated architectures can be seen in the works of a number of organisations and standards bodies including Oasis, the IETF, RosettaNet & UN/CEFACT[(6)]. Ontologies provide one possible underlying structure for standardisation, as they provide the means to both classify and infer about data held, leading to the creation of structures capable of dealing with different formats and unit types. However, the role of ontologies in general can be overestimated, and it is possible that simpler data structures, closer in stature to standard taxonomies may prove highly beneficial to the standardisation process. The methods by which standardisation could occur, is outside the scope

of this document, however work in this area has been completed by a number of projects, including DIRC[7].

2.3 Existing Negotiation Projects / Products

A number of projects have attempted to address the problems associated with this area of research; the following, looks at one successful product in the area, followed by an existing research project examining part of the problem TRANSACT is addressing.

E-mediator

One of the more notable negotiation products is eMediator[8], which allows multiple constraints to be specified exogenously. However, the model it uses to specify requirements is based purely on the specification of options for a given price (distributive); rather than a more flexible model where different values for given attributes could affect each other in different ways. As with many other projects of its type, it relies heavily on reasonable behaviour of opposing parties, which can never be guaranteed.

SNAP

SNAP[9] is a protocol under development at Argonne Laboratories, the home of the Globus toolkit, to address some of the issues left by the ongoing development of their CAS (Community Access Server). It primarily addresses the lack of a standardised agreement and structure for contracting of agreements. SNAP uses XML to create SLA's between users etc. The developers recognise the need to negotiate SLA's at multiple points in the process, but stop short of providing an actual negotiation tool; instead concentrating on the standardised nature of the agreement specification. In doing so SNAP takes a hierarchical view of the agreement structure, allowing the linking of lower level agreements to form part of higher level agreements through the aggregation of SLA's. This approach allows flexibility in some respects but also raises issues of verbosity.

CNP (Contract Net Protocol)

CNP was first deployed in 1980 as part of a Distributed Acoustic Sensor Network simulation. It uses two types of agent, a participant and an initiator to find and supply user requirements. It follows a standard RFQ style process, with an initial call for proposals (CFP). This sends the request information to a number of potential services. Those services then bid on those options. The user then selects a supplier, rejecting all other offers. Research on CNP led to a number of other more advanced systems including TRACONET[10]. The simple method of specifying standardised requests, and weighing responses for the best solution has allowed CNP to be used in a number of interesting places. Indeed, TRACONET (TRAnsportation COoperation NET) was designed to route delivery trucks through road networks more efficiently. In conclusion CNP is a popular style of agent negotiation tool, hampered to some extents by the need for honest users, and the lack of a synchronisation architecture which is essential for all time based requests. It does however provide a simple integrative protocol. It is not a adequate solution to the problems this project aims to solve however, as it relies on a one shot approach to negotiation, without the intensive bartering style required to choose between multiple options from a single service.

2.4 Securing Contracts using PKI

The negotiation process needs to produce secure contracts in order to ensure non-repudiation, without which negotiation is meaningless.

PKIs[11] secure information based on mathematical formula that allow easy conversion into a form which can only be converted back using a different key, and vice versa. This allows a person to encrypt documents using a person's public key knowing it can only be decrypted using that person's private key. Reverse this approach and you can prove a person encrypted a file using the private key by decrypting using the public one. Presuming the private key is kept private this can be used to prove persons credentials. The basic structure of this is illustrated below in Fig 1.

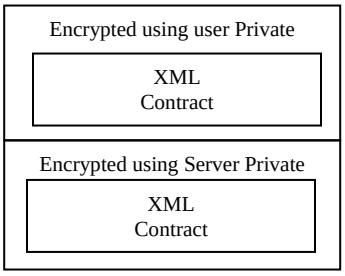

Fig 1: Encryption utilising PKI

If both contracts match when decoded, they both agreed to the same contract. These can be authenticated by anyone with the public keys of both parties.

This model of authorisation requires a considerably more lightweight server end authorisation mechanism to process incoming requests due to its reliance on the actual information received (the contract) rather than a database of individual permissions stored locally. This allows for easier replication of authorising mechanisms without the need to control multiple copies of databases. The provision of secure communication is essential to underpin the development of higher level functionality including that of automating the negotiation cycle.

3 Automating the Negotiation process

The negotiation process is based around the XML encrypted contract instances outlined above. The contract model operates on a request / reply basis with the client making demands, and the service counter offers. The received demands / counter demands are then taken into account for the next round. Fig 2 illustrates this model:

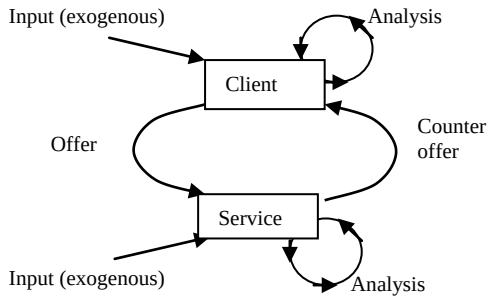

Fig 2: Negotiation model

Individual contract iterations are split into two sections, a header and main section. The header contains compulsory information regarding the basic needs of a contract. For example, the names of the parties, the validity and expiration date of the agreement etc. The main section contains the contract clauses that are being negotiated upon, for example Auto_Resubmission (on failure). These clauses can contain either a numerical or textual description of the values being negotiated. The user can extend the main section to include any contract clauses they wish, providing they are supported by the service they wish to negotiate with. The provided clauses a given service provides can be retrieved from within the negotiation tool environment either from the service directly or via a standard UDDI registry.

Each clause has a name, value, description and definition component. The latter two are used to ensure the clause can be linked to a textual description of its meaning and a statement of where that description was defined respectively. These could be obtained through the use of a QoS ontology[12], however this is currently classed as an area of future work. The description helps reduce the ambiguity factor in human understanding, whereas the definition helps avoid clashes of clause names between institutions. An example of this is shown below:

Contract Clause:
Name: Payment
Value: 50
Description: In £. "Payment" means the price for the goods excluding carriage, packing, insurance and VAT.
Definition: Standard_Contracting_Ontology.OWL

TRANSACT includes the description itself rather than merely relying on the use of a URI etc for clause location & definition, in order to maintain the self-contained, human readable nature of the contracts themselves. This also ensures that contracts can be pinned to a specific version of the QoS definitions.

3.1 User Controls

Two specific models of interaction are defined: A visual environment to allow users to directly negotiate contracts, and an API to enable negotiation to act as part of a wider workflow control process. For brevity this paper will concentrate on the GUI side of the development.

The negotiation process operates under the premise that users should be removed from the negotiation itself as far as possible. For the client this entails requirement specification and to accept or decline a contract at the end of the process. For the service side this entails capability specification, and the honouring of contracts agreed by the negotiating clients with its negotiation service.

The following sub sections are split into the different types of contract manipulation mechanisms, complete with explanations on use.

Bound Specification for numerical clauses

A numerical type clause must range between an upper and a lower bound.

Option Specification for Textual clauses

The textual clauses operate on a simple pop up list from which a user can select a given option.

Bias

Bias can be entered into the system by a series of interlinked slider bars attached to each clause. The values on the sliders individually add up to 100%, giving each of the sliders a percentage of importance in calculations. The sliders can be locked in place to avoid movement allowing a user to easily manipulate the levels of importance ascribed to the individual clauses.

The primary purpose of the sliders is to allow the negotiator to rank clauses in relation to the need for change in the next iteration cycle. The way in which this information is taken into account is covered in section 3.3.

Starting conditions

The client / service are also given control over the starting stance taken for negotiation. For example, does

the client start with a request for everything it requires, or work towards this, analysing the cost implications from a less demanding starting position. In addition, control over the aggressiveness of the negotiating client with regard to the negotiating itself is provided.

An indication of the importance of clauses in relation to each other is a powerful tool, sufficient for simple negotiations. However, further user input is required to indicate the relative importance of individual values within the clauses of a negotiation. Take, for example, the following simplistic scenario:

Clause required between 0 – 100

A priority slider bar can indicate the importance of this clause to the negotiating client; however, it cannot give an indication of whether the user treats all possibilities within that clause range with equal acceptance. Is a 0 better than a 100, or vice versa etc. This type of information could be implemented by users merely indicating their range boundaries from least to most acceptable at all times; thereby giving coarse grained information regarding the acceptability flow. However this simplistic solution still does not address the issue of acceptability of values within a given range. In many real world situations the levels of acceptability would not necessarily follow a linear flow, but perhaps a curved one. Below are examples of curves that could represent a user's wishes; to the left a situation where acceptability increases exponentially across a range; to the right, the opposite:

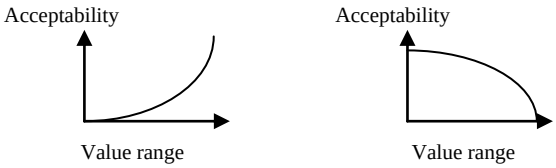

Fig 3: Curved acceptability ranges

There are effectively an infinite number of basic curves that may represent user's preferences. It is also possible users preferences will follow a normal curve like the one shown below in fig 4:

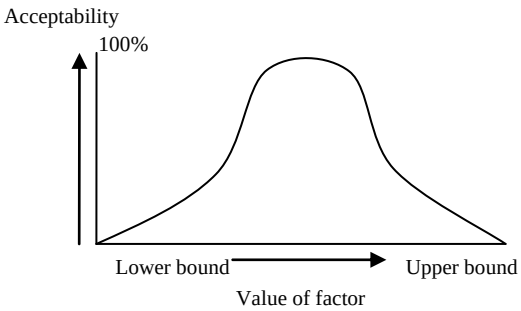

Fig 4: Normal acceptability ranges

A normal curve would represent a common situation in the negotiation of a product. The user has a preference for a specific subset of values within a wider range of possibilities, in line with the economic theory of satiation.

Preset acceptability curves that the user can select, are a useful facility to enable the user to make a quick decision. A number of basic types can be defined, borrowing from the linear, quadratic curve and normal distributions. However, it is also important that any given acceptability distribution be inputted if that were the users wish. This could be achieved by allowing the user four basic controls in the creation of their own customisable acceptability curve.

TRANSACT has implemented a user definable graphical representation of acceptability over a range allowing the reproduction of any single oscillatory acceptability graph through the manipulation of four controls (Left edge, Right edge, Protrusion vertical height and Protrusion horizontal orientation). Fig 5 shows a screen shot of the tool:

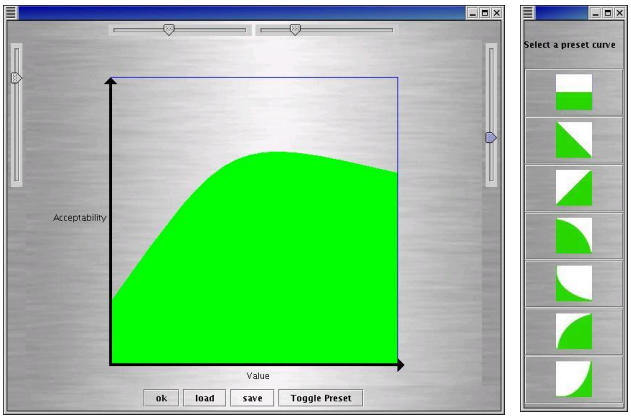

Fig 5: Acceptability rating manipulation

3.2 Trade-off support

The input mechanisms above are adequate for the inclusion of basic user requirements. However, they cannot capture more complex domain specific information, which the user may wish to express. This input can be provided through the implementation of logic rules / production rules.

Tradeoff rule input is used to enter any specific interdependencies into the system prior to negotiation. The format uses keywords, values and numbers, structured to avoid ambiguity in the processing of user instructions. TRANSACT makes use of the rules to dynamically change the base values for clauses, importance etc, during negotiation. The list of currently supported keywords is listed below.

IF	Test whether something has occurred
THEN	Do something specific based on IF
>	To specify something should be greater than a given value
<	To specify something should be less than a given value
=	To specify something should be a certain value
AND	Link between two arguments
UpperBound	By changing the upper bound of a numerical clause the negotiation strategy can be changed dynamically
LowerBound	See above.
Weight	Used to raise and lower the importance of the given clause in calculations

In many production rule systems brackets are used to avoid ambiguity caused by statements like:

IF A > B OR B < C AND E > F THEN…
Thus:
IF (A > B OR B < C) AND E > F THEN …

However by restricting the user to AND statements only, ambiguity can be avoided. It is possible to approximate most rules using AND / OR with one of more rules using AND. This also has the advantage of making the format of the logic rules considerably easier to learn. The number of different permutations for bracketing also makes the creation of a recursive parser highly complex. To ensure rules can still be implemented without the OR keyword it is necessary to make sure that all rules can only be applied once. For example:

If A > B THEN A**.**Weight +40%

The above example appears relatively innocuous, however, if applied iteratively, for example several times during a single negotiation, problems would emerge. Rather than using a relative increase as the basis for a rule of this type, the rule can be rewritten to restrict it to only apply a single time; see below:

IF A > B THEN A**.**Weight = 40%

No matter how many times this rule is executed its effect is predictable and constrained. Though it can be argued that the rule is less powerful in this form, it is still likely to be sufficient for the types of rules input by the user. The use of rules that are unaffected by multiple implementations also allow the elimination of the OR statement thus:

IF (A>B OR B<C) AND E > F THEN A**.**Weight= 40%
Thus:
IF A > B AND E > F THEN A**.**Weight = 40%
IF B > C AND E > F THEN A**.**Weight = 40%

The down side of this type of rule creation is the use of multiple rule lines, but in doing so makes them considerably easier to learn, validate for acceptability, and parse internally.

A further example of rule construction can be seen below:

IF Downtime > 10 AND User_Notification_Fail = “No” THEN Cost UpperBound = 55

If the downtime is greater than 10% of the contract time, and the user is not informed directly of system failure, then the upper boundary for the contract cost could be reduced to £55, presumably from some greater figure. Notice that denominations (£, $ etc) are not specifically used within the rules to avoid processing problems, but are instead made clear in the clause description.

3.3 Decision making

Each contract offer consists of a number of clauses and values. For each round of the iterative negotiation cycle these clauses need to be individually analysed for acceptability, to determine the extent to which changes may be required in future iterations. A given number of them can then be changed for the next iteration. Firstly the logic rules are applied, which can have considerable effects on the negotiation strategy taken for the next iteration. The analysis of clause values then occurs, as a three stage process, described in detail below; following this is an explanation of how cost can be associated with the clause values, and finally how the compromise position evolves:

Stage 1 – Determination of a clauses current percentage

- For textual clauses there is a simple rating of either 100% if matched to what the user specified or 0% if it didn’t
- For numeric clauses, the value depends on the position of the value within the range following a simple formula for calculation of position within a given range:

$$\left(\frac{(\text{Value} - \text{Lower Range})}{(\text{Upper Range} - \text{Lower Range})} \right) * 100$$

For example:

Upper Range = 120
Lower Range = 40
Value = 80

$$\left(\frac{80 - 40}{120 - 40} \right) * 100 = 50\ \%$$

Stage 2 – Acceptability adjustment: Model

Given the acceptability graphs the user defined prior to the start of negotiation, the acceptability of a given value within a bounded range can be extracted by calculating the intersection of a value with the curve. For example:

User bounds 20 – 80
Value 50

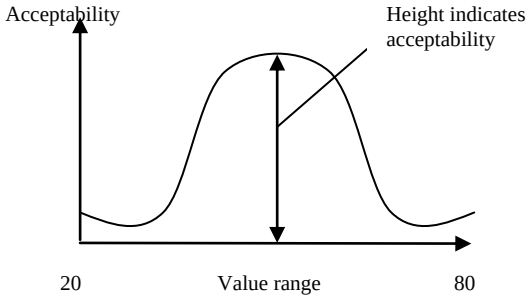

Fig 6: Acceptability calculation

In this simple case, the intersection of the value 50 and the curve would be 100% acceptability. A value like 40 would probably be around the 75% mark and so on.

Stage 3 – Prioritisation & ranking

The values determined in the previous section have now been adjusted for their formula model. At this point, it is necessary to take into account the priorities that have been attributed to the different clauses by the user (through use of the slider bars). Once this has been achieved, decisions as to the urgency for change in each clause for the next negotiation iteration can be made from the current clause rankings. The priority data is applied through the following formula:

((100 - Stage 2)* Priority) /100

The formula was developed largely through experimentation, and is designed to provide a suitable ranking, where a higher value elicits greater importance for change; effectively adjusting the acceptability value for priority. For example:

Clause A 40% stage 2 Bias 20% = 12
Clause B 40% stage 2 Bias 30% = 18
Clause C 20% stage 2 Bias 5% = 4
Clause D 80% stage 2 Bias 45% = 9

Notice that clause B is more important than clause A. Therefore, it proves more important to modify next time around. Clause D is the most important of all, but may not be changed next time around due to the very high level of acceptability in the current offer. The results therefore show that in the next iteration of negotiation changes should be made reflecting the fact that the rank of acceptability is: B-A-D-C.

Defining cost

As the reader may have gathered from the previous sections, TRANSACT is built upon an economic model of service interaction for grid resources. This means that a given contract offer is made up of a number of distinct clauses which have an associated cost. The cost clause itself is generic and could be represented in a number of different ways from euros to usage quotas. Cost is only bartered indirectly in the TRANSACT model. It is calculated from the values of the clauses in the negotiation. For example, given:

Clause A = 40 elicits 50% acceptability
Clause B = "yes" elicits 100% acceptability

Given a cost clause ranging from 0 – 100 the input above would set the price at:

$$\frac{50 + 100}{200} = 75 \text{ (Simple average calculation)}$$

Thus, the negotiator recognises the concept of value for money. The actual calculation is slightly more complex than this as it has to take into account the importance of the given clauses, and the way in which the clauses are combined to provide an associated cost.

Changes for the next iteration

TRANSACT attempts to improve the compromise position by improving a number of clauses each iteration based on the urgency for change, calculated using the three stage process above. The size of change for a given field depends upon how far from optimum a given clause is, and also on the aggressiveness of the negotiating client.

4 Evaluation

Other research completed in this area has so far concentrated on the theoretical possibilities of automation in negotiating systems rather than the construction of prototypes, which has made evaluation a non-trivial problem. In order to gauge opinion, and to gain

information regarding possible improvements to the prototype, a number of real world scenarios have been constructed. These stem from real problems and have been gathered from, amongst others Epidemiological studies, through consultation with members of the statistical departments of Lancaster and Manchester universities. The results of these scenarios are too complex for inclusion in this paper; however, a number of general observations could be drawn. The evaluation showed that the sensitivity of the controls on the prototype was too high; something which has now been addressed. Also, more importantly, that further research is required to develop more advanced honing algorithms and techniques for examining the negotiation problem space more effectively in the search for pareto-optimal solutions.

5 Future work

Future work on TRANSACT will look at the constructs necessary to provide standardised term definition including the possibilities for ontology creation. Examining the possibilities for participation in future standardisation processes for QoS is an objective of the project at this time. There is also still a considerable amount of testing required to balance the various controls and formulas used to enable automated negotiation.

6 Conclusions

In conclusion this paper has provided an overview of the automated negotiation domain. In doing so it has outlined a possible design for a solution based on standardised core web technologies including XML and SOAP. In addition, preliminary overviews of the implementation are provided in order to give some flavour of the interactions seen when utilising the prototype, and in turn, the actions it takes in negotiation. The paper has also provided information regarding future work, and evaluation of the prototype.

Acknowledgements

I would like to thank the UK Engineering and Physical Sciences Research Council, grant number GR/M52786 and the Dependability Interdisciplinary Research Collaboration (DIRC).

References

[1] SECSE consortium
Project Webpage: http://secse.eng.it

[2] Aldus
ALDUS (1992). The ALDUS project: Artificial Legal Draftsman for Use in Sales, ESPRIT Commission

[3] VO development
Katzy R. Bernhard D. Design and Implementation of Virtual Organizations. http://portal.cetim.org/file/1/67/Katzy-1998-Design-and-Implementation-of-Virtual-Organizations.pdf

[4] 3rd Party Monitoring
D. Xu, K. Nahrstedt, and D. Wichadakul. Qos-aware discovery of wide-area distributed services. *Proceedings of the First IEEE/ACM CCGrid2001*, 2001. http://citeseer.ist.psu.edu/xu01qosaware.html

[5] VO Vision
Cueni T. Virtual Organizations - the next Economic Revolution. http://www.nubix.ch/vo/virtual.pdf

[6] UN/CEFACT
UN/CEFACT Applied Technology Group (ATG) http://www.disa.org/cefact-groups/atg/index.cfm

[7] DIRC
DIRC, QoSOnt development http://digs.sourceforge.net/

[8] eMediator
Sandholm T. eMediator. A Next Generation Electronic Commerce Server. *Conf proceedings, International conference on autonomous agents.* http://citeseer.ist.psu.edu/60365.html

[9] SNAP
Czajkowski K, Foster I, Kesselman C, Volker S, Tuecke S. SNAP: A protocol for negotiating service level agreements and co-ordinating resource management in distributed systems. http://www.isi.edu/~karlcz/papers/snap-lncs-25370153.pdf

[10] CNP
Sandholm T. An implementation of the Contract Net Protocol based on marginal cost calculations. *In Proceedings of the National Conference on Artificial Intelligence (AAAI), Washington, D.C., July 1993.*

[11] PKI
RFC 2459 Internet X.509 Public Key Infrastructure http://www.ietf.org/rfc/rfc2459.txt

[12] QoSOnt
EuroMicro2005 Lock R, Dobson G, Sommerville I. QoSOnt: A QoS Ontology for Service-Centric Systems. *Conference Proceedings, Porto, Portugal, 31st August - 3rd September 2005.* ISBN: 0-7695-2431-1