AnEconomics-driven Approach for Automated SLA Negotiation for Cloud Services Adoption: *Aspoc*^{2}

Samia Nisar
School of Computer Science
University of Birmingham
Birmingham, UK
snab88@gmail.com

Rami Bahsoon
School of Computer Science
University of Birmingham
Birmingham, UK
R.bahsoon@gmail.com

Abstract — **With the increasing number of clouds providing software-, infrastructure, data storage, and/or information as services the process of adopting cloud services involve intensive negotiation to mediate users' requirements with that of Service Level Agreement (SLA) promises. This increases the need for an effective, dynamic, and flexible automated approaches for negotiation for resolving conflicts and mediating users' requirements relative to SLAs. The novel contribution of this paper is an economics-driven approach, which Pareto optimality and Bayesian learning for automated negotiation. We motivate and describe the approach and propose an architectures, which realizes the economics-driven bilateral and multiple attribute negotiations. We report on the implementation of the approach and its evaluation by looking at some scenarios.**

Keywords-Automated negotiation, Cloud SLA, Pareto Optimal, Bayesian Updating, Negotiation Protocols.

I. INTRODUCTION

With the ubiquitous availability of Internet, businesses are gradually outsourcing their IT services and moving towards the cloud making Service Level Agreements (SLA) the de facto in any business relation. As businesses endeavor to reduce costs, an SLA should define a service in such a manner that can be measured and managed whilst guaranteeing the maximum value for cost. Standard definition of service level agreement presents it as a legal and *manually* created document, which acts as a contract between the service provider and the consumer stating the service level to be delivered, the liabilities on the part of a service provider and the consumer and actions to be taken under specific circumstances [25], [14].

Studies of current SLAs highlights that cloud providers when defining the terms and metrics overlook the dynamic nature of the services offered resulting in insubstantial negotiation outcomes. These outcomes include overambitious SLAs with heavy penalties increasing the cost to a level unacceptable to the consumer. As highlighted by [28] an overly large penalty may cause financial difficulties in the provider and cause a loss of service, which may be particularly problematic if the service is not available from another provider.

The reason for the aforementioned problems is likely to be attributable to poor negotiating strategies in use. With the emerging use of cloud, there is still a general lack of systematic approaches and methods for automated negotiation for SLAs. A negotiation protocol defines how agents communicate through an ordered interchange of structured messages [32]. There are varieties of negotiation protocols that can be applied when negotiating for web services but not all protocols befits the cloud environment. To identify a suitable negotiation protocol for cloud environment it is important to understand the significance of 'negotiation' in cloud.

A. Negotiations: the 'what' and the 'why'

Literature provides multiple definitions of negotiations. According to [36] negotiation consists of two parts: a payload part that contains the content of a proposal and a type field, which may be "proposal", "Accept" or "Reject". [27] described negotiation and bargaining as a same process and defined bargaining as a means through which two or more purposive arrive at a specific settlements or outcomes under conditions of strategic interaction or interdependent decision making [27].

A more precise definition related to cloud SLA is provided by [32] who reviewed negotiation as a difficult problem for human and defined it as a route that is taken to achieve mutual benefits via bargaining on the goals, searching for alternatives, and *strategically* sharing information.

The diversity of definitions of SLA parameters among the cloud providers and the consumers such as availability, bandwidth, response time, repair time etc., makes it crucial that the provider and consumer speculate to a mutual understanding of the parameters involved. Negotiation plays a vital role in achieving this mutual understanding. But in cloud SLA domain its importance is not fully understood and is often underestimated, resulting in the poorly constructed SLAs. A good negotiation protocol can significantly improve the reliability of the SLAs as the violation of SLAs is often induced by poorly negotiated service metrics and consumers are left incomprehensive.

B. *Aim of the paper*

Different approaches in negotiation leads to different results but the overall goal of negotiations remain the same i.e. to maximize benefit and efficiency. Much effort has been spent on proposing automated negotiating systems for the e-commerce industry but none has been implemented in cloud environments, where negotiations are an integral part when it comes to defining SLA. This paper proposes an economics-driven approach for automated SLA negotiation for the problem of outsourcing services in the cloud, referred to as (*Aspoc*^{2}). The automated negotiator aims at attaining maximum benefits for the cloud service user and the provider in a restricted time frame.

The automated negotiation approach is composed of three major phases: the first phase is that of searching for cloud services and providers, where a publish-subscribe model is adopted. That is, cloud services are published with their SLAs and users can subscribe to search and screen for offerings. The second phase is that of negotiation the services with the cloud provider using economics-inspired approach. If the negotiation is successful, the outcome is stored as a final SLA that can be used for the process of monitoring. The third phase provides a communication medium for the service provider and service consumer to exchange offers during the negotiation. The focus of this paper is the second phase, an important phase in the SLA lifecycle, where the defined metrics for monitoring the SLA are the outcome of this phase.

The remaining of the paper is structured as follows: Section 2 motivates the approach and highlights the major problems and requirements for negotiating SLA. Section 3 provides a brief overview of automation levels while section 4 argues on the benefits of automated negotiation over manual negotiation. Section 5 explains in detail the negotiation model and provides implementation algorithms of the model. Section 6 shows the implementation of the concept and sections 7 explains the example setup for evaluation section 8 shows the experimental results obtained given various scenarios after the implementation of the system. Section 9 provides the similar approaches taken to solve the problem mentioned in previous section and compare the solutions with the suggested solution this paper. We conclude by discussing the results achieved and highlighting the future work in this field.

II. MOTIVATION

A. *The negotiation problem*

According to [9] the major issue in negotiating an SLA lies in the definition of the document as current SLA schemes are defined in a manner that the committed agreement cannot be modified during service provision and is effective until all the activities pertaining to it are finished or until one party decides to terminate it. Based on this it can be concluded that re-negotiating an SLA during service runtime is not in practice.

As cloud services are offered over the Internet, the bandwidth of the connection shows another issue to consider during a negotiation. The bandwidth in a wide area network is constantly fluctuating resulting in inconsistent performance level of the service. Therefore, it can be suggested that performance metrics of cloud services should be discussed based on their level of consistency. This can be achieved by classifying the metrics according to peak and off-peak times in an SLA. This could result in improved liability of the terms defined in the document.

B. *Requirements for negotiation system*

In their discussion about computer support systems for negotiation [27] pointed out three basic requirements: 1) the system should support requirements analysis. 2) It should support strategic analysis, which includes analysis of other party's need as well as the team's joint position with respect to the efficiency frontier (refers to the locus of achievable joint evaluation from which no joint gains are possible). 3) The system should provide a communication channel through which the negotiators are able to refer to some common referents for facilitating the interaction.

The exchange of offers introduces new data into a negotiation with every counter proposal. It is important to use this data by updating the current knowledge frame of the negotiation to achieve better outcomes. It has been well established that the ability to update the data flow whenever new information comes in, irrespective of the source is highly desirable. The ability to learn becomes compulsory in a dynamic environment such as Cloud and it appears its implementation in automated negotiating systems has been much awaited.

Another important consideration when negotiating for a service is to consider the historical data such as negotiation history, monitoring history and violation history. This helps defining the boundaries of the negotiation and provides an overview of what can be expected from the service of a similar nature. Referring to the historical data of a service not only assist the negotiation but also furnishes comprehensive view of the cloud performance and facilitate the grounds for negotiation.

III. LEVELS OF AUTOMATION IN NEGOTIATION

The software designed for negotiation support and automation can be classified according to their level autonomy and support. These systems have evolved during time substituting one for an other however, the all have the same motive which is to achieve a decision automatically that is consensual. Numerous researches have defined these systems, [22] summarized the classifications as follows:

- A negotiation support system (NSS) is a software which implements models and procedures, has communication and coordination facilities, and is designed to support two or more parties in their negotiation activities.

- An e-negotiation system (ENS) is a software that employs Internet technologies and it is deployed on the web for the purpose of facilitating, organizing, supporting and/or automating activities undertaken by the negotiators and/or a third party.
- An e-negotiation table (ENT) is a software that provides negotiators with a virtual space (bargaining table) and tools, which they use in order to undertake negotiation activities.
- A negotiation software agent (NSA) is a software that is actively involved in a significant part of negotiations and makes decisions on behalf of its human or artificial principal.
- A negotiation agents-assistant (NAA) is a software agent that provides a human negotiator and/or a third party with timely and context-specific advice, critiques, and support.

IV. Why Automation over Manual?

The major reason for the appraisal of automated negotiation could be due to the recognition of e-commerce. It is believed that automated systems are more efficient in solving complex problems that involve business related decisions and can achieve better result than humans. This assumption is grounded in the subsequent lack of experience and capability in humans when compared to a software agent.

Automated negotiation is expected to reduce the cost incurred by the third parties hired for the assistance of the manual negotiation process. [28] promoted the theory expressing that in a future environment, it will be highly desirable that negotiation of the SLA parameters be automated and that the resulting SLA used as input to an automated resource allocation system as means of meeting required level of performance.

According to [7] changing to automated negotiation will have three fundamental effects on the negotiating: Firstly, negotiations will happen more quickly. Secondly, there will be a increase in channel richness during the negotiation process and finally, a decrease in the number of problem dimensions that can be ‘discussed’, as the absence of face-to-face interaction inevitably decrease the opportunities for all sides to discover new issues from which they both gain. Supporting automated negotiation [27] also stated that the use of computer support will have much to offer in terms of compensating the negotiator with what they lack in conducting rational negotiations, i.e., higher information processing capabilities and capacities.

V. Negotiation Model

Aspoc^2 takes an economics-driven approach in performing automated negotiation. The major benefit of using economics approaches in automated negotiations has been discussed by [27]; economics models are often dynamic models that not only focus on achieving better results, but also on the actual process that leads to better results.

This paper introduces the concept of Pareto optimality in the context of cloud SLA negotiation as the underlying mechanism that complements the work of [10]. Pareto optimality is mirco-economics concept that suggests the use of available resources efficiently that will result in most beneficial outcome. The outcome is referred to as a Pareto efficient point. If any system is not Pareto efficient, there is a potential for a Pareto improvement i.e. when there is a possibility to increase the Pareto efficiency by the reallocation of resources to improve the outcome of at least one participant without reducing the efficiency of other participants’ involved [13].

The second economic theory used is the Bayesian theory. This theory provides a framework for automated learning agent that will update the information of the service provider’s negotiation behavior and apply in the next rounds of negotiation using Pareto algorithm to incrementally improve the negotiation results. [36] presented this technique as *Bazaar*: an experimental system for updating negotiation offers between intelligent agents during a bilateral negotiation. The paper models the negotiation as sequential decision-making task and uses Bayesian probability as the underlying mechanism.

A. *Negotiation Phases of Aspoc*^{2}

Phase 1. Attribute selection for negotiation

The procedure followed in the attribute selection used in this paper is ‘ask n return’. In this method, the user is asked to select a pair of attributes, namely issue x_1 and x_2. The paper will follow only a single pair of issues that are selected based on the level of importance in the negotiation process. In a real time environment multiple pairs can be selected to negotiate simultaneously but this approach is likely to raise the costs. The discussion of most critical issues first will determine the effectiveness of the negotiation. If the results for the critical attributes are not satisfactory, the negotiation can be terminated saving the computational effort used to negotiate all attributes at once.

The information collected from the user for the issues is the preferred value of the selected attribute and relaxation parameter. Relaxation parameter is the upper and lower limit threshold, the value beyond which the user will not allow any changes and could terminate the negotiation process. This defines the range of values the negotiator is willing to settle.

After the selection of the user preferred attributes has taken place, they are translated into parameters and stored in a vectors such as UP= $\{up_1, up_2, ...up_n\}$ [16]. These vectors are passed to step 2 of the negotiation process described in the next section. After the Pareto solution is selected for the first pair of issues, system goes through the same process for the second pair of issues until there are no more issues left to negotiate.

```
<wsp: TemplateSLA>
  <nb: Cloudname> CloudA </Cloudname negotiable='false'>
  <nb: Description> This is a sample cloud </Description negotiable='false'>
  <nb: CloudType> IaaS </CloudType negotiable='false'>
  <nb: ServiceHighlights>
        <nb: Instances> Standard, Micro, Cluster </Instances negotiable='false'>
        <nb: OperatingSystem> Windows, Linux, OSX</OperatingSystem
        negotiable='false'>
        <nb: Software> my SQL, IBM</Software negotiable='false'>
        <nb: Pricing> $94 per hour for all the services</Pricing negotiable='true'>
        <nb: Monitoringservice>Cloud Watch</Monitoringservice
        negotiable='false'>
           <nb: MonitoringPrice> $30</ MonitoringPrice negotiable='true'>
  </ServiceHighlights negotiable='false'>
  <nb: Servicemetrics>
     <nb: Availability> 99.5 </Availaibility negotiable='true'>
     <nb: penaltycharges> $15 per fine</Penalty negotiable='true'>
     <nb: paymentprocedure> creditcard</paymentprocedure negotiable='true'>
     <nb: security> SSL connection </security negotiable='false'>
        <nb: privacy> current set 2, amount of information display from 1 to 5.5
        being the highest level of privacy</privacy negotiable='true'>
  </Servicemetrics>
</TemplateSLA>
```

FIGURE 1. XML TEMPLATE SLA

The above-described process is used to define the service consumer's preferences. In a bilateral negotiation, there are two sets of preferences, which are collected from both parties. In this system service provider's values and attributes are selected from the template SLA of service provider stored in the system's database. The attributes are only available for negotiation if the provider has specified the values as negotiable, shown in Figure 1. The service consumer by default will only be allowed to specify the preferences of the negotiable attributes. These attributes are processed in the same way as the service consumer's preferences.

Phase 2. Applying negotiation using Pareto optimal theory

There are three major components of Pareto Optimality method used in the system these include *Indifference curve, Budget constraint,* and *Utility Value*.

- *Indifferent curve* is defined as range of values for x_1 and x_2 on which the Utility value of the issues remain constant. These indifference curves are developed using the relaxation parameter values for x_1 and x_2. The range of values between upper and lower thresholds are paired for each issue and used to determine indifferent curve.
- *Utility value* of a pair of issues x_1 and x_2 is defined as $x_1 * x_2$.
- *Budget Constraint* is defined as average sum of service provider's and service consumer's Utility values for all the issues under discussion such that $P_x x+P_y y=$ Budget.

The problem *Aspoc²* tries to solve is to find the Pareto frontier using indifference curves and Budget constraint for an issue under negotiation. The Pareto frontier is made up of Pareto optimal points which represent the set of feasible choices that interested parties can take to achieve the best outcome. For a given budget constraint, the system's role is to find a joint tangent between both parties' indifference curves. The system can search for a joint tangent by optimizing the values of x_1 and x_2 under the same constraint. If the parties' indifference curve under the given constraint coincides, a joint tangent has been [10]. Otherwise, the values of x_1 and x_2 are shuffled to develop new indifference curves as explained later in this section.

To develop a Pareto frontier there has to be several different Pareto optimal points, for this purpose the Budget constraint has to vary in a appropriate way so that it results in a different Pareto optimal point than its predecessor. The next reference point is selected by using Sliding Reference point technique which involves sliding the reference point along the normal of the most recent joint tangent curve presented in [10] as:

$$\mathbf{r}^{l+1} = \mathbf{r}^{l} + \eta, \qquad l = 1,2, \ldots \quad (1)$$

Where the parameter η is a suitable step size. The smaller the step size η is, the closer the new solution is to the previous solution.

a) Negotiation protocol

The negotiation protocol is based on a bilateral negotiation process where a consumer is interested in some services of the service provider and initiates a negotiation procedure through *Aspoc²*. The consumer has provided the system with his/her preferences for each attribute and these preferences are stored in an array CP = [cp_1, cp_2, ... cp_n], where n= maximum number of consumer preferences from cp_1 being the most preferred value. The service provider's preferences are uploaded in the array SP =[sp_1, sp_2, … sp_n]. The data in this array is added from the historic database that stores the previously used values for same provider. If there is no data found in the database, the values are uploaded from the Template SLA of the provider as described in *Phase 1*. Figure 2 illustrates the process of Pareto point formation. The negotiation protocol is described in following steps:

Step 1. The user is asked to select a pair of issues from the negotiable attributes.

Step 2. The values of the selected attributes are loaded as explained in previous section.

Step 3. The loaded values are used to determine indifferent curves of each negotiating party involved.

Step 4. If the indifferent curves coincide with each other at the reference point specified, an approximation of Pareto optimal point has been found. If the values do not coincide, the system adjusts the reference point to a specific value using the aforementioned algorithm and tries again.

Step 5. If additional Pareto optimal points are required to obtain a Pareto frontier, the system shuffles the preferences array and repeats step 3 again and stores all the Pareto optimal points in other array called PF =[pf_1, pf_2, ... pfn].

Step 6. The Pareto frontier is presented to the consumer in the form of offers from which the consumer can select and decide to send to service provider as a proposal.

Step 7. If the service provider replies with a counter offer, the system updates the service provider's preferences using Bayesian updating rule and repeats step 3. Or if the service provider requests to re-negotiate the offer, the second offer in the array PF [] is presented as the offer. This process is repeatedly carried out until the offer has been accepted or PF [] is empty in which case the system terminates.

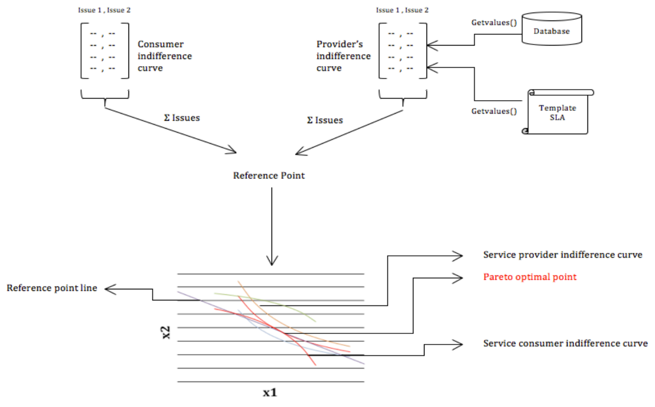


FIGURE 2. THE NEGOTIATION MODEL

b) Algorithm for Pareto Optimal with Bayesian Updating

Input:

consU= consumer utility value
provdU= provider utility value
List= result of Bayesian updating value
cx1= [ ], consumer upper and lower limit value for issue 1
cx2= [ ], consumer upper and lower limit value for issue 2
sx1= [ ], service provider's upper and lower limit value for issue 1
sx2= [ ], service provider's upper and lower limit value for issue 2
newdata [ ], updated values of service provider's issues, nx1[ ] and nx2 [ ]

Output:

```
1     Initialization
2     procedure budgetConstraint (consU, provdU)
3     budget=(consU*provdU)/2
4       end procedure

5     procedure budgetMapping (budget)
6     map budget to x and y.
7     end procedure

8     procedure obtainValueHistoricDB( ) //values for provider curve
9         start jdbc connection
10        sql= execute query ( select issue1 from history)
11            while (sql.next)
12                sql.fill(sx1[ ] )
13                sql2= execute query ( select issue1 from history)
14            end while
15            while (sql.next)
16                sql2.fill(sx2[ ])
17            end while
18    end procedure

19    procedure consumerIndifferenceCurve( cx1[ ], cx2[ ] )
20        for each I in cx1 to length ( j ) in cx2 do
21            for (int i=0; i<cx1.length; i++ )
22                for (int j=0; j<cx2.length; j++ )
23                    while ( cx1.length=cx2.length)
24                        sum=cx1[ ] + cx2[ ]
25                        if ( sum==cx1[ i ] + cx2[ j ] )
26                            set consindiffcurve[ ] [ ]=0
27                            add cx1[ ] and cx2 [ ] to consindiffcurve [ ] [ ]
28                        else if ( sum ≠ cx1[ i ] + cx2[ j ] )
29                            collection.shuffle( array. cx1[ ] )
30                            collection.shuffle( array. cx2[ ] )
31                            go back to step 31
32                        else
33                            sum[ ].next
34                        end if
35                    end while
36            return consindiffcurve [ ] [ ]
37    end procedure

38    procedure providerIndifferenceCurve( sx1[ ], sx2[ ] ) //values collected from step 15
39        same as procedure consumerIndifference curve
40    end procedure
```

```
41    procedure bayesianIndifferenceCurve( nx1[ ], nx2[ ] )
42        for each i in nx1 to length ( j ) in nx2 do
43            for (int i=0; i<nx1.length; i++ )
44                for (int j=0; j<nx2.length; j++ )
45                    while ( nx1.length=nx2.length)
46                        sum=nx1[ ] + nx2[ ]
47                        if ( sum==nx1[ i ] + nx2[ j ] )
48                            set updateindiffcurve[ ] [ ]=0
49                                    add nx1[ ] and nx2 [ ] to updateindiffcurve [ ] [ ]
50                                else if ( sum ≠ nx1[ i ] + nx2[ j ] )
51                                    collection.shuffle( array. nx1[ ] )
52                                    collection.shuffle( array. nx2[ ] )
53                                else
54                                    sum[ ].next
55                                end if
56                            end while
57                    return updateinindiffcurve [ ] [ ]
58                    end procedure

59                    procedure paretoOptimalwithBayesian( updateindiffcurve[][], consindiffcurve[][] )
60                        int count=1
61                        while ( count > 1)
62                            if ( cid[][] == uid[][] && cid[][] == B[][] && uid[][] == B[][]) //matching
63                                add cid[ ][ ] to paretosolution[ ][ ]
64                                return paretosolution[ ] [ ]
65            else
66                set new= budget + 10
67                call budgetMapping(new)
68            else
69            return no match found
70        end procedure
```

Phase 3. Updating negotiation data using Bayesian Updating

The importance of learning in negotiation has recently been recognized in the game research community as fundamental for understanding human behavior as well as developing new solution concepts [36]. In any negotiation, the final result depends on various factors out of which the most important is the opponent's preferences and the situational updates. For a successful negotiation, it is critical that a system must understand these updates and apply to it in the upcoming levels of the negotiation at hand.

According to [16] updating is more effective for online learning in the course of a sequence of interactions with its opponent in a negotiation. In a negotiation process, an agent does not know about the opponent at the beginning, however it learns gradually through the series of interaction to make better decision in the course of negotiation. As negotiation is an iterative process, an agent can get feedback from its opponent through a series of events E and can update its beliefs which led to the initial hypothesis to result in probabilities that are more precise. This concludes that negotiating agent can make better decision in each negotiation round as more is learnt about the opponent.

This paper applies Bayesian updating technique for the purpose of updating the information learned after the first round of the negotiation. [36] proposed the technique for the first time in a non-automated negotiation model. They applied Bayesian updating as sequential decision making model and characterized the following: First, there is a sequence of decision making points which are dependent on each other. Secondly, the decision maker has a chance to update his knowledge after implementing the decision made at a certain stage and receiving feedback so that he can make a more informed decision can be made at the next stage.

a) Bayesian Updating in *Aspoc*^{2}

Bayesian updating is an input and output process. It requires data flow of at least one negotiation round to successfully update the new criteria. The following are the pre-requisites of the technique for *Aspoc*^{2}:

- There should be some relevant information available about the supplier and known to the negotiation system.
- The negotiation should be a multi-rounded procedure where the proposals are exchanged more than once.
- The negotiating system must receive any form of feedback about the topic under discussion such as price, availability etc. from any of the parties involved.
- The negotiating system should update itself at any new information received about the issue irrespective of its source and should utilize the newly acquired data in the next round of negotiation if applicable.

The system considers the negotiation from view of consumer. At its current state the information about the supplier and its preferences are fed through the Template SLA and the historical database that contains the past negotiations of the provider. The working of Bayesian Updating in *Aspoc*^{2} follows the work of [36] and utilizes the following equation:

$$P(H_i|E) = \frac{P(H_i)P(E|H_i)}{\sum_{k=1}^{n} P(E|H_k)P(H_k)} \qquad (2)$$

b) Algorithm for Bayesian Updating

Input:

Hk = Historic data Information.
IOs = First offer presented by service provider disclosing his preferences.
Hy = Probability of each Hypothesis coming true for an attribute under negotiation from historic data.

Output:

```
1    Initialization
2    Hk[ (hk, cloud_id) ]:=1
3    negoround[ (session_id, username) ]:=1
4    IOs:=1

5    procedure detectNewData( Hk[i], IOs[j], newData[k])
6      for each i in Hk[] to length ( j ) do
7          if ( Hk[i] ≠ IOs[j])
8              initialize newData[k]
9              while( i≠j )
10               add to newdata[k]
11               return newdata[]
12             end while
13         else return 0
14         end if
15     end for
16   end procedure

17   procedure bayesianMeasuring( Hy[ i ], newData[ k ])
18       probabilty=0
19       Hy={h_1,h_2…..h_n}
20       newData={n_1,n_2….n_n}
21       for each i in Hy[ ] to length newData[ ] do
22           probability[ j ]={n_1,n_2….n_n} divide by {h_1,h_2…..h_n}
23               for each j in probability[ ] to length Hy[ ] do
24                  calculation of $P(H_i)P(E|H_i)$
25                  calculation of $\sum_{k=1}^{n} P(E|H_k)P(H_k)$
26                   set countp:=0
27                   while a[ ] > 0
28                       set counta=counta + 1
29                       set suma= suma+ counta
30                       return suma
31                   end while
32                   set sum:=0
33                   set countp:=0
34                   while probability[ ] > 0
35                     28, 29, 30
36                   end while
37       end for
38       List1=Array.list( probability[ ] )
39           for listcount=1 then list divide by a[]
40               return list
41           end for
42           if ( list[ i ]>Hy[ i ] )
43              return newdata [ ] //values for updated indifference curve
44   end procedure
```

Phase 4. Calculating efficiency using Utility functions.

According to [16] the main objective of a negotiation is to maximize the utility function with a certain probability of completing a deal. Utility function is an integral part of a negotiation process as it calculates the satisfaction level of the offer by comparing it to the initial utility value provided by the consumer. The utility value is calculated as:

$$U = \frac{\Sigma i \times V^i}{\Sigma i} \qquad (3)$$

Where U refers to the total utility of the consumer and V^i to the utility of the issue *i*. Using the utility value of the offer presented the satisfaction of the negotiation can be determined. The higher the difference between the consumer's utility value and the proposed offer's value, negotiation results will be less satisfactory and vice versa.

Phase 5. Termination of Negotiation.

When negotiation is required to be of multiple rounds then a specific bound is needed to terminate the process and not let it go on for a limitless time. *Aspoc*^{2} follows the following cases to terminate the negotiation procedure:

- In case where the buyer has rejected the proposal and asked to terminate the process.
- In case where the negotiation system has reached the end of the proposal array.
- In case where the time limit allocated for the specific negotiation has expired.

The latter is introduced so that the negotiation module will have a way to exit the negotiation loop and end the session for the current user, as each user will be provided with individual sessions to negotiate the SLA agreements. The negotiating parties can arrange the time limit after discussing among themselves. *Aspoc*^{2} automatically presents a final proposal at the time threshold limit and the user can either accept the proposal or terminate the system.

VI. Evaluation

The evaluation was performed using an experimental technique to judge the meaningfulness of the approach. The conceptual system was implemented in Java framework as a proof of concept and was evaluated. The evaluation involved testing the system during the negotiation between two parties with support of different scenarios.

A. *System Architecture*

Based on the literature review it can be concluded that economic theories have been mostly overlooked in negotiating strategies for web-based services despite their effectiveness. This paper utilizes economic theories to achieve an automated negotiation with efficient results and to provide new dimension in the field of automated negotiation in cloud SLA.

1) *Component view of Apoc^2*

Figure 3 show the components of the system and display the data flow between components. Each component is described below:

a) *Web service API*

This component deals with the searches and request building for the purpose of negotiation. The user of the service is requested to select a pair of issues according to their importance. The user is then asked to input the preferences of the selected issues, which is a specific range of values that will lead to the formation of indifference curves. The purpose of determining these values is to produce the most satisfying offers in relation to the issues. These preferences are then passed on to the next component.

b) *Negotiation offer builder*

This component focuses on building the offers from the provided data using two major economic theories namely Pareto optimality and Bayesian updating. The working of the algorithms will be described in the next section. This component contains various interfaces to handle the data and produce the desirable results.

i) Input handler is an interface that filters the data and arranges it according to the required specification of next layers.

ii) Update Controller checks for any new data that has been updated and chooses the negotiation algorithm accordingly. If there is no change in the data from the one that was previously processed then the input is send to Pareto optimal generator. In case of a change in the data values, the updater directs the data to Bayesian updater

iii) Bayesian Updater updates the data accordingly and sends it to the Pareto optimal generator.

iv) Pareto optimal generator generates offers by applying certain calculations that are illustrated in the algorithm provided later in the paper.

c) *Decision handler*

This component handles the provider's response and notifies the system to act accordingly. The *offer handler* handles a list of offers called Pareto frontier and transfers them to the *communicator* one at a time and uploads the provider's response onto the *response handler*.

d) *Data Storage*

This component interacts with the database to upload and download the information and pass on to other components of the system.

e) *Timer*

This component interacts with the negotiation offer builder and decision handler to update the components about the allocated time for a negotiation session.

VII. EXAMPLE TESTING

This section illustrates the working of the *Aspoc^2* using scenario views. The implementation of the system follows the algorithms described in the sections above. This section will also demonstrate the system's behavior in the extensive scenarios designed to test efficiency of the algorithm. In a prototyping development, assumptions play a vital role as they help display the system's functional part evidently. The assumptions that were followed in the example running of *Aspoc^2* are listed in the section below.

A. *Assumptions:*

The scenarios used in this paper lead to the following assumptions:

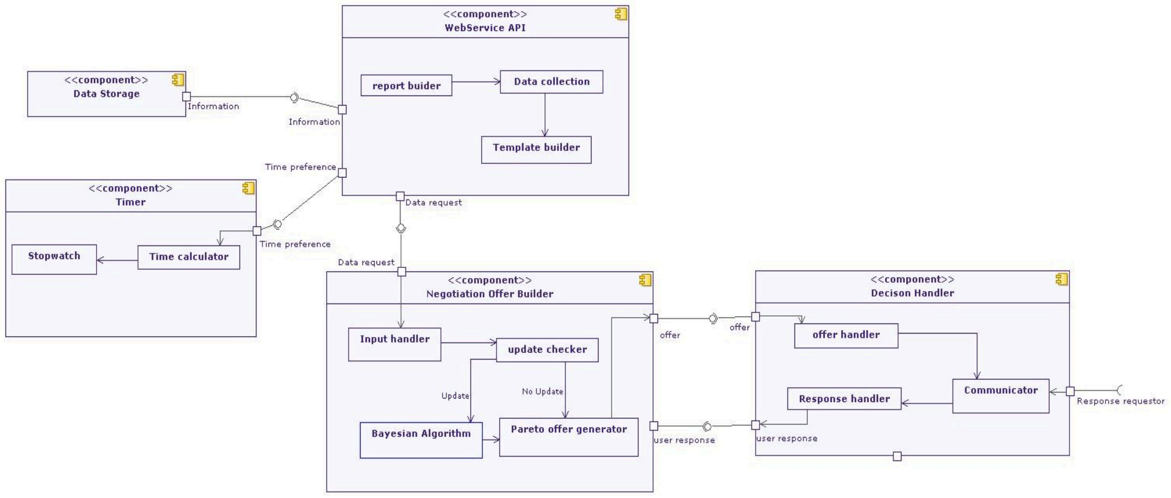


Figure 3. Components of ASPOC^2

- It is assumed that the system communicates with an agent similar to the one proposed in this paper, which handles the requests and respond to them on behalf of the service provider.
- The cloud information is communicated via a publish-subscribe model where cloud providers publish their template SLAs. These SLAs are stored in a centralised database.
- The database also holds the past transactions and negotiation history of the cloud providers for those who have used the service before for negotiations.
- The system will communicate on behalf of the consumer performing the role of an agent.

B. *Example setup*

The example setup used for this paper follows a consumer ABC organization searching for an IaaS cloud provider with the best services available but with the lowest possible prices. After successfully registering with the service, the consumer is presented with a list of cloud providers that are stored in the system (The cloud providers that are not subscribed to the service will not be displayed as there is no data available). ABC selects a cloud from the options presented. In this scenario, it is assumed that ABC selects cloudA for negotiation purposes. Template SLA of cloudA is shown in figure 1. The next stage of the system is to discover ABC's preferences for the issues that are about to be discussed. ABC is asked to select pair of issues in the order of their importance and provide their preferable value for the issues. In this case, it is assumed that ABC selects 'service price' and 'penalty pricing' from the negotiable attributes presented. The next part request ABC to enter a fixed range of values for both the issues on which they will be willing to settle. These values are called relaxation parameters (represented by x_1 and x_2 in the above algorithms).

Scenario 1: The provider accepts the first Pareto solution presented.

This scenario focuses on the system's response when the provider accepts the first offer presented to him. After the consumer approves the offer, it is presented to the service provider for approval. The acceptance of this offer leads to formation of an SLA that is valid for a specific period of time for e.g. 1 year. The SLA is stored in the database that could be retrieved in XML format and can be used by services like WSLA for monitoring. The monitoring of SLA is out of scope for this paper.

Scenario 2: The provider rejects the proposal offered.

In this scenario, the system will terminate and the offers that have been generated will be discarded from the system. It will also notify the consumer that the provider is no longer interested in negotiating.

Scenario 3: The provider agrees to re-negotiate the terms and presents a counter offer.

In this scenario the system will detect the counter offer, extract the data from the offer and apply Bayesian updating to update estimations for the next negotiation round. The updated data is used in Pareto optimization so that the offers generated have higher chances of getting accepted than the previous offer. The same steps are taken in case of the re-negotiation scenario.

Scenario 4: The provided time for the negotiation session has elapsed.

The time set for the negotiation is allocated by the users of the system and is activated when the negotiation starts. After the allocated time has elapsed, the users of the system can save the existing session and start another session. The new session detects the user from its Id and uploads the previous negotiation session with a new time allocation and continues with the negotiation.

VIII. RESULTS AND DISCUSSION

Aspoc^2 was evaluated in extensive experiments using the scenarios described in the previous section. This section discusses the results obtained and concludes the feasibility of such a system in a cloud environment. The results obtained were separated based on use of Pareto Optimality and use of Pareto Optimality with Bayesian Updating rule. After the start of the negotiation process, the system was observed on the following gauges:

- Number of Iterations to reach the first Pareto solution with and without Bayesian updating method.
- Use of time boundaries in the system and how it affects the algorithm performance.
- Satisfaction with the solution in comparison with the overall utility value of the issues.

The results were obtained using the values provided in the tables below. Table 1 describes the detailed input preferences of issues for each of the involved parties used that lead to the final offer presented later in the paper.

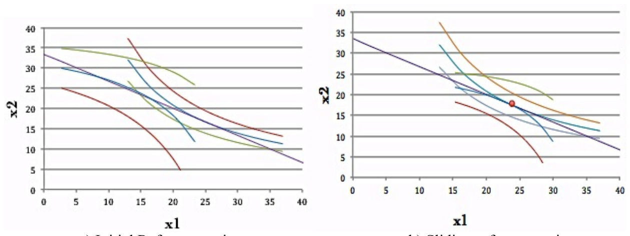
TABLE I. SERVICE CONSUMER AND PROVIDER PREFERENCES

	Issue 1 (x_1)			**Issue 2 (x_2)**		
	Service price			***Penalty price***		
	Desired value	*Relaxation parameter range*		*Desired value*	*Relaxation parameter range*	
		Upper limit	*Lower limit*		*Upper limit*	*Lower limit*
Service consumer	22	35	12	20	10	30
Service Provider[1]	Unknown	80	20	Unknown	4	35
Service Provider[2]	Unknown	40	10	Unknown	9	27

[1] Estimate based on the recorded data of the provider during previous negotiation
[2] Updated values using Bayesian updating theory after the first the round of negotiation

A. *Results achieved with Pareto Optimal*

For ease of understanding, the paper divides the negotiation process into phases. In first phase, the consumer performs the relative setting, which leads to indifference curves for the negotiation. In the second phase the actual offers are generated using the indifference curves and are exchanged. In the example used, the reference point is set initially as the average of both the parties initially to generate Pareto solutions. The time limits are not defined in this set of results.



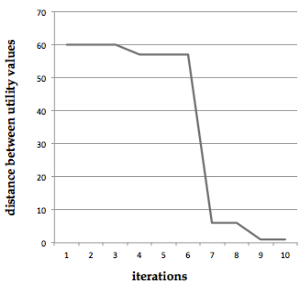
a) Initial Reference point b) Sliding reference point

FIGURE 4. GRAPHICAL VIEW OF SOLUTIONS

The indifference curves are then used to find a Pareto optimal point on a predefined reference point. In the figure 4a, it is shown that indifference curve fall on different points on the reference line. The reference point is then adjusted using slide reference point discussed in *Phase 2* of negotiation protocol. The system again enters into iterations and tries to find the Pareto optimal point shown in figure 4b. The process continues until a match is found, since there is no time boundary used for this example the system's performance decreased after 12 iterations due to excessive data handling.

B. *Results achieved with Pareto Optimal and Bayesian Updating rule*

The next test was conducted using a defined time constraint, although it did not affect the system's performance. The aim was to encourage negotiating parties to provide precise data under time restriction. The algorithm was improved by introducing Bayesian updating that lead to the solution in fewer iterations than the previous test. The figures below show the number of iterations versus the difference between the parties' utility values. When the difference between the utility values of the parties' is zero, the solution is found for the issues under discussion.



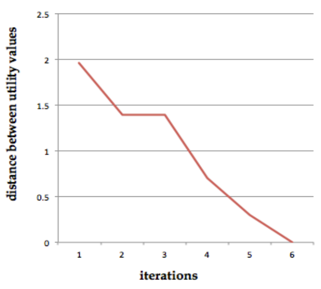


FIGURE 5A. ROUND 1 WITHOUT UPDATING

FIGURE 5B. ROUND 2 WITH UPDATING

From the figures above, it can be concluded that *Aspoc*[2] was able to reach the solution point in much less iterations when Bayesian Updating rule was applied. At the completion of the second phase, the system produced the solution point given the altered reference point in the sixth iteration shown in figure 5b. The presented solution when compared with both parties' utilities proved to be effective. The table V shows the numerical proof.

The table also shows the gains and loss in the values for both parties. The suggested solution offers the service consumer to settle for the service price higher than the desired price. On the other hand the consumer is gaining on the penalty price, as the consumer required it to be as high as possible. Discussing the results from the service provider's point of view show a different dimension when a provider is losing on the solution in term of the desired value in case of service price while there is no change in the penalty price.

TABLE II. RESULTS

	Reference point	Pareto optimal solution		Desired values		Relative difference	
		Service	*Penalty*	*Service*	*Penalty*	*Service*	*Penalty*
Service consumer	35	25	15	22	20	3	-5
Service provider[3]	35	25	15	30	15	-5	0

[3]The desired values are taken from the template SLA of the service provider to measure the system's effectiveness.

To compare the solution with the utility values of both parties the difference indicates that the parties will be more willing to settle on the solution since the distance between the solution and the desired values is low. The total utility values for service consumer and service provider are 440 and 450 respectively; where as the total utility value of the solution is shown as 425. The parties can evaluate the total utility value and choose to accept or decline this offer.

IX. RELATED WORK

Literature review evinces the major focus of the research in the field of cloud SLA revolves around tracking the performance of the service i.e. monitoring. There are varieties of tools that are being implemented providing efficient ways to monitor the SLA in cloud. But the previous studies have overlooked the importance of the negotiation in a cloud SLA. Numerous negotiation strategies can be spotted in the literature of economics, politics, and sociology but none of them are applied when negotiating an SLA for cloud. This is due to the lack of information and focus in the current research towards the importance of a negotiation process in the cloud. Although there have been similar approaches in the field of web services and grid computing described below.

A variety of literature can be found in the field of negotiation. Due to the complexity of the process, much effort has been put to achieve a level of sophisticated autonomy. The concept of using software to negotiate was first implanted in late 1970s. MEDIATOR [18] is among the early negotiation support systems that supported negotiators in the construction of their own decision problems and assisted a third party in the construction of the negotiator's joint decision problem. Another successful implementation of negotiation software includes Inspire, Atin and Aspire. Aspire is a combination of the two that provide the full support to its user in negotiating a task [23].

There have been many approaches for negotiating SLAs for Grid computing and Web services which includes the work of [3] who presented a proposal in the web services field that defines executable negotiation protocols in BPEL. PANDA is another similar for negotiation in Grid services presented by [12]. Table 1 provides a comparative analysis of the aforementioned systems showing the missing features that are desirable components in automated negotiating systems.

X. Conclusion and Future work

As seen in the past literature the automated negotiation protocols have been used intensely in the field of e-commerce to solve the negotiation problem. The field also gained popularity in the web services culture but the cloud environment is still faced by this problem. This paper successfully solves the automated negotiation problem in cloud SLAs using economic models. It identified the key requirements for such systems to work in a cloud environment and implemented in the prototype. The survey of the past and current literature survey suggested that none of the successful implementation of automated negotiations presented all the features in their system, shown in Table 1. This lead to the motivation of proposing *Aspoc^2* that can perform all the desired features.

The implementation of the system successfully produced the desired results showing that economic models are applicable in an automated negotiation in an environment such as Cloud. This paper also provided strong evidence of efficiency of such models noticeable through the results obtained.

The implementation of the system successfully produced the desired results showing that economic models are applicable in an automated negotiation in an environment such as Cloud. This paper also provided strong evidence of efficiency of such models noticeable through the results obtained. Algorithms used in this paper were improved based on the results of system. As there was no previous approach that implemented economic models in automated negotiation, the novelty of the research is exhibited. The component view of the system provides the reference for individual implementation in case of implementing the system with SLA monitoring systems.

Aspoc^2 opens new dimensions of research in cloud computing i.e. automated negotiation. The approach to enhance this model to cover all the scopes of cloud SLA needs to be explored in the future. Another possibility is to implement *Aspoc^2* in the real world cloud environment for the evaluation of the results. This will lead to not only the evolution of negotiation in the Cloud Computing environment but will also help the system to achieve a new potential in automated negotiations in cloud SLA.

TABLE III. COMPARATIVE ANALYSIS OF RELATED WORK

Proposal	System name	System Type	Theory applied	Field applied	Features					
					1	2	3	4	5	6
Kim and Segev, 2005	BPEL4WS	NSS	-	Web services, B2B negotiation	✓	✗	✗	✓	✗	✗
Ashri et al., 2003	ABN	NSA	Argumentation based negotiation	General	✗	✗	✗	✓	✗	✓
Gimple et al., 2003	PANDA	DSS	Utility functions	Web and Grid services	✓	✗	≈	✓	✗	✓
Schoop et al., 2003	Negoisst	NSS	Electronic auction model	E-commerce	✗	✓	✗	✓	✗	✓
Kerten and Lo, 2001	Atin	NSS, NSA	Auction model	-	✗	✗	✓	✓	≈	✓
Jarke et al., 1985	MEDIATOR	NSS	Consensus model via information exchange	E-commerce	✗	✗	✓	✓	✗	✗
Zulkernine and Martin, 2011	Negotiation Broker NB	NSA	Mathematical Policy based incl. Exponential, Sigmoid and Polynomial functions	Web services SLA	✓	✗	✓	✓	≈	✓
Samia Nisar, 2011	**Aspoc^2**	**NSS, NSA**	**Economic models: Pareto Optimal and Bayesian Updating**	**Cloud SLA**	✓	✓	✓	✓	✓	✓

Features:

1.Multiple decision-making algorithms
2.User preference settings
3.Multi user
4. Support multiple issue negotiations
5. Self-learning ability
6. Adaptability to other platform

Legend:

✓ Yes ✗ No ≈ Partially

REFERENCES

[1] ALHAMAD, M., DILLON, T. and CHANG, E., (2010) 'Conceptual SLA framework for cloud computing', *4th IEEE International Conference on Digital Ecosystems and Technologies (DEST 2010)*, pp. 606-610.

[2] ASHRI, R., RAHWAN, T., and LUCK, M., (2003) 'Architectures for negotiating agents', In *Proceedings of the 3rd Central and Eastern European conference on Multi-agent systems* (CEEMAS 2003), Springer-Verlag, Berlin, Heidelberg, pp. 136-146.

[3] BAEK KIM, J., and SEGEV, A., (2005) 'A Web Services-enabled marketplace architecture for negotiation process management', *Decision Support Systems,* **40**(1), pp. 71-87.

[4] BARTOS, O.J., (1977) 'Simple Model of Negotiation: A Sociological Point of View', *The Journal of Conflict Resolution,* **21**(4), pp. 565-579.

[5] BEIGI, M., LOBO, J., GRUENEBERG, K., CALO, S. and KARAT, J., (2010) 'A Negotiation Framework for Negotiation of Coalition Policies', *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY 2010)*, pp. 133-136.

[6] BENYOUCEF, M. and VERRONS, M., (2008) 'Configurable e-negotiation systems for large scale and transparent decision making', *Group Decision and Negotiation,* **17**(3), pp. 211-224.

[7] BLECHERMAN, B., (1999) 'Adopting automated negotiation', *Technology in Society,* **21**(2), pp. 167-174.

[8] BRAUN, P., BRZOSTOWSKI, J., KERSTEN, G., KIM, J.B., KOWALCZYK, R., STRECKER, S. and VAHIDOV, R., (2006) 'e-Negotiation Systems and Software Agents: Methods, Models, and Applications', *Intelligent Decision-making Support Systems*, Ch. 15, pp. 271-300.

[9] DI MODICA, G., TOMARCHIO, O. and VITA, L., (2009) 'Dynamic SLAs management in service oriented environments', *Journal of Systems and Software,* **82**(5), pp. 759-771.

[10] EHTAMO, H., HAMALAINEN, R. P., HEISKANEN, P., TEICH, J., ZIONTS, M. V,S., (2002) 'Generating Pareto Solutions in a Two-Party Setting: Constraint Proposal Methods', *Management Science*, 45(12), pp. 1697-1709.

[11] FILZMOSER, M., (2010), 'Automated vs. Human Negotiation', *International Journal of Artificial Intelligence*, **4**(10), pp. 64-77.

[12] GIMPEL, H., LUDWIG, H., DAN, A. and KEARNEY, B., (2003) 'PANDA: Specifying Policies for Automated Negotiations of Service Contracts' In: M. ORLOWSKA, S. WEERAWARANA, M. PAPAZOGLOU and J. YANG, *Service-Oriented Computing - ICSOC 2003.* Springer Berlin / Heidelberg, pp. 287-302.

[13] HEISKANEN, P., EHTAMO, H. and HÄMÄLÄINEN, R.P., (2001) 'Constraint proposal method for computing Pareto solutions in multi-party negotiations', *European Journal of Operational Research,* **133**(1), pp. 44-61.

[14] HILES, A. N., (1994) 'Service Level Agreements: Panacea or Pain?', *The TQM Magazine*, **6**(2), pp.14 – 16.

[15] HUANG, C., LIANG, W., LAI, Y., and LIN, Y., (2010) 'The agent-based negotiation process for B2C e-commerce', *Expert Systems with Applications,* **37**(1), pp. 348-359.

[16] HUANG, R., (2003) 'Negotiation modeling and e-shopping agents', In: *Proceedings of Fifth International Conference on Computational Intelligence and Multimedia Applications* (*ICCIMA 2003*), pp. 3-10.

[17] HUDERT, S., LUDWIG, H. and WIRTZ, G., (2009) 'Negotiating SLAs-An Approach for a Generic Negotiation Framework for WS-Agreement', *Journal of Grid Computing,* **7**(2), pp. 225-246.

[18] JARKE, M., JELASSI, M.T. and SHAKUN, M.F., (1987) 'MEDIATOR: towards a negotiation support system', *European Journal of Operational Research*, **31**(3), 314-334.

[19] JENNINGS, N.R., FARATIN, P., LOMUSCIO, A.R., Parsons, S., Wooldridge, M., Sierra, C., (2001) 'Automated Negotiation: Prospects Methods and Challenges', *Group Decision and Negotiation*, 10(2), pp. 199-215.

[20] JONKER, C., ROBU, V. and TREUR, J., (2007) 'An agent architecture for multi-attribute negotiation using incomplete preference information', *Autonomous Agents and Multi-Agent Systems,* **15**(2), pp. 221-252.

[21] KELLER, A. and LUDWIG, H., (2003) 'The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services', *Journal of Network and Systems Management,* **11**(1), pp. 57-81.

[22] KERSTEN, G.E. and LAI, H., (2007) 'Negotiation Support and E-negotiation Systems' *Group Decision and Negotiation*, **16**(6), pp. 553-586.

[23] KERSTEN, G.E., AND LO, G., (2001) 'Negotiation Support Systems and Software Agents in e-Business Negotiations', In: *the First International Conference on Electronic Business Hong Kong*, pp. 19-21.

[24] LARSON, K.D., (1998) 'The role of service level agreements in IT service delivery', *Information Management & Computer Security*, **6**(3), pp.128 – 132.

[25] LEWIS, L. and RAY, P., (1999) 'Service level management definition, architecture, and research challenges', *Global Telecommunications Conference (GLOBECOM '99)*, vol. 3, pp. 1974-1978.

[26] LI, H., AHN, D., and HUNG, P.C.K., (2004) 'Algorithms for automated negotiations and their applications in information privacy', In: *Proceedings of the IEEE International Conference on E-Commerce Technology (CEC)*, pp. 255-262.

[27] LIM, L., and BENBASAT, I., (1992) 'From negotiation to negotiation support systems: a theoretical perspective', In: *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, **4**, pp. 153-163.

[28] MITCHELL, B., MCKEE, P., (2005) 'SLAs A Key Commercial Tool**'** In: Cunningham, P., and Cunningham, M., *Innovation and the Knowledge Economy: Issues, Applications, Case Studies*. IOS Press Amsterdam, pp. 30-36.

[29] PICHOT, A., WÄLDRICH, O., ZIEGLER, W. and WIEDER, P., (2009) 'Towards Dynamic Service Level Agreement Negotiation: An Approach Based on WS-Agreement', In: J. CORDEIRO, S. HAMMOUDI and J. FILIPE, *Web Information Systems and Technologies.* Springer Berlin Heidelberg, pp. 107-119.

[30] RAIFFA, H., (1985) 'Post-Settlement Settlements', *Negotiation Journal,* **1**(1), pp. 9-12.

[31] ROBINSON, N.W., VOLKOV, S., (1998) 'Supporting the Negotiation Life Cycle', *Commun. ACM,* **41**(5), pp. 95-102.

[32] SCHOOP, M., JERTILA, A. and LIST, T., (2003) 'Negoisst: a negotiation support system for electronic business-to-business negotiations in e-commerce', *Data & Knowledge Engineering,* **47**(3), pp. 371-401.

[33] SHEN, W., LI, Y., GHENNIWA, H.H., WANG, C., (2002) 'Adaptive Negotiation for Agent-Based Grid Computing', *In Proceedings of the Agentcities (AAMAS'02)*, **5**, pp. 32-36.

[34] TEICH, J.E., WALLENIUS, H., KUULA, M. and ZIONTS, S., (1995) 'A decision support approach for negotiation with an application to agricultural income policy negotiations', *European Journal of Operational Research,* **81**(1), pp. 76-87.

[35] XAIO, Z., and DONGGANG, C., (2010) 'A Policy-Based Framework for Automated SLA Negotiation for Internet-Based Virtual Computing Environment', *Proceedings of the 2010 IEEE 16th International Conference on Parallel and Distributed Systems (ICPADS 2010)*, pp. 694-699.

[36] ZENG, D. and SYCARA, K., (1998) 'Bayesian learning in negotiation', *International Journal of Human-Computer Studies,* **48**(1), pp. 125-141.