PANDA: Specifying Policies for Automated Negotiations of Service Contracts

Henner Gimpel[1], Heiko Ludwig[2], Asit Dan[2], and Bob Kearney[2]

[1] Universität Fridericina Karlsruhe (TH), Englerstrasse 14, 76131 Karlsruhe, Germany
gimpel@iw.uni-karlsruhe.de
[2] IBM T.J. Watson Research Center, 19, Skyline Drive, Hawthorne, NY, 10025, USA
{hludwig, asit, firefly}@us.ibm.com

Abstract. The *Web and Grid services frameworks provide a promising infrastructure for cross-organizational use of online services. The use of services in large-scale and cross-organizational environments requires the negotiation of agreements that define these services. Buying at a fine granularity just when a need arises is only feasible if the costs of establishing new agreements are low. Today, negotiation is often a manual process yet many simple online services would allow full or partial automation. The PANDA approach automates decision-making and proposes to specify a negotiation policy, expressing a party's private negotiation strategy, by combining rules and utility functions. In addition, the decision-making problem can be decomposed into different aspects that can be executed by different interacting decision-makers. Using PANDA for policy specification and negotiation decision-making reduces the costs of setting up new services and contracts. Hence, the use of fine-grained on-demand services becomes feasible.*

1 Introduction

Web and Grid services facilitate on-demand use of services accessed over a network – potentially across organizational boundaries. This may lead to an environment in which services are bought at a fine granularity from a number of competing providers. Given a marketplace of providers of similar or comparable services, it also enables organizations to dynamically buy resources on the spot when additional business requires access to more capacity than available in-house. Negotiating agreements and buying services ad-hoc enables business partners to tailor their contractual bindings in regard to time varying needs and constraints. This reduces the risk of being stuck in long-term contracts that are no longer profitable.

PANDA (**P**olicy-driven **A**utomated **N**egotiation **D**ecision-making **A**pproach) facilitates automated decision-making within negotiations. It allows decomposition of intended negotiation behavior, integrates different formalisms for policy, and structures the strategic reasoning of a negotiator.

M.E. Orlowska et al. (Eds.): ICSOC 2003, LNCS 2910, pp. 287–302, 2003.

1.1 Agreements in the Web Services Domain

Relationships between organizations are defined by agreements between them. Those agreements may be implicit by accepting some fixed terms that are published by a service provider or they are made explicit in the form of a *contract* that is specifically negotiated.

Most description formats of the Web services stack are unilateral in their nature, i.e., a service provider describes properties and usage conditions of a service. This is the case, for example, for WSDL [5] and WS-Policy [4]. Unilateral descriptions are limiting because they cannot represent consensus and reciprocity. Some description formats in the context of Web services, however, represent aspects of agreements among two or more parties, such as the Business Process Execution Language (BPEL) [1] or proposed languages for service level agreements such as the Web Service Level Agreement (WSLA) language [15] and the Web Service Management Language (WSML) [19]. A draft for "Agreement-based Grid Service Management (OGSI-Agreement)" has been submitted to the Global Grid Forum [6]. As relationships between service providers and clients become more complex and more non-functional requirements such as response time guarantees must be considered, languages will be defined to express agreed relationships in contracts. In dynamic environments, those contracts will vary for each relationship and can be negotiated ad hoc.

1.2 Negotiating Agreements

Negotiations are mechanisms that increase the flexibility of possible service contracts. We use the term negotiations as comprising all exchanges of messages, such as offers and acceptance messages, between two or more parties intended to reach an agreement.

In the context of dynamically setting up service relationships, it is important to use an efficient decision-making process that reduces cost and time of the setup. Human-based negotiations are time-consuming and expensive. Hence there have been many approaches to support human decision-makers with decision support systems and negotiation support systems [14]. In addition, in some cases requiring simple decision-making, it is desirable to automate all or a part of the negotiation task of a participating party and hence to drive down the costs, and particularly time of establishing a new agreement. However, engineering negotiation applications is complex, time-consuming, and expensive. In many cases, it is not worth to implement a new negotiation application for each new negotiation task or situation.

Ideally, an employee of an organization can express the negotiation preferences as a negotiation policy that can be interpreted by a negotiation engine. A policy in the context of this paper is an explicit representation of intended behavior to be interpreted by an engine that implements the behavior. In the case of negotiations, a negotiation engine responds to offers and other communication in accordance to this specified policy. PANDA provides the expression of policies that combine utility functions and business rules.

1.3 Objective and Structure

To facilitate the automation of decision-making based on negotiation policy, the objective of this paper is to propose a mechanism an organization can use to define its preferences – PANDA. Using a combination of utility functions and rules provides a user a suitable formalism to represent preferences in a way that can be easily expressed and managed.

To this end, the remainder of the paper is structured as follows: Section 2 introduces a motivating example and discusses the specifics of negotiating services. Subsequently, expression of intended behavior is analyzed in Section 3. In Section 4 we introduce the PANDA approach that defines a model of decision-making. Section 5 illustrates the use of the presented approach based on the example. Finally, Section 6 summarizes the results, compares the approach to related work, and gives an outlook on future work.

2 Negotiation of Service Agreements

The negotiation of service agreements takes an important role in the life cycle of agreements, facilitating the creation of complex agreements in lieu of simple binding to services that are described. Parties in a potential service relationship use advertising and search functions to find each other, either directly or using an intermediary. Negotiation messages such as offers and acceptance notifications are exchanged. This may finally lead to a contract. Upon successful negotiation, each organization creates a contract implementation plan that defines how to implement a particular contract [16]. However, beyond general issues of negotiating contracts, online and Web services have some specific properties that require further analysis.

2.1 Example

To illustrate the further discussion, we use the example of a stock quote service that is offered by a service provider FastQuote that negotiates some attributes of the service for particular customers. The service is offered at an interface defined by FastQuote in a WSDL file. It exposes an operation getQuote in a binding specifying SOAP over HTTP as transport. The service is offered by FastQuote at different levels of delays of the quotes from the trading floor, 20 minutes, 5 minutes, or real time. Furthermore, on an IT level, FastQuote negotiates different levels of service regarding availability and response time at requested levels of throughput. The throughput is measured in invocations per minute. Finally, the price is open to negotiation.

FastQuote's preferences over the variety of possible contracts include a reasonable coherency between price and delay; delivering stock quotes with a shorter delay is a more valuable service and should yield higher earnings. For building up new business relationships FastQuote is willing to be more acquiescent in negotiations with first time customers.

Once an agreement is reached, FastQuote plans the deployment of the new agreement. Depending on the quality of service parameters agreed upon it plans the allocation of capacity on existing hosts or the provisioning of new hosts. The host capacity relates to the amount of memory and the number of CPU seconds required. This plan

is called the contract implementation plan. The company has an algorithm that returns capacity requirements for given throughput rates and response times at a requested availability.

2.2 Service Characteristics

Services have a number of characteristics, which impose special requirements for a decision-making in the course of a negotiation. The most important are:

Non-storability denotes the fact that resources not used yesterday are worthless today [13]. The implication for marketing one's resources by providing services is that time and current resource workloads are crucial factors in decision-making. If time runs out and capacity is going to be wasted, providers will make stronger concessions.

Complexity is another service characteristic. Service contracts are usually complex due to the fact that they have many defining parameters. The complexity issue can be addressed by means of templates, utility functions, and sophisticated tactics for offer creation.

Intangibility raises problems in determining the value of a service contract because the good sold to a consumer does not equal the operating expense dedicated by the service provider. A provider has resources and uses them to yield a return on his investment. A consumer has needs and satisfies them by buying a service. The service agreement bridges the gap between provider resources and consumer needs; neither side has to be acquainted with the precise nature of the other side's concerns. The intangibility of services leads to the need of an internal transformation from a service contract to a deployment plan.

Provisioning a service instead of settlement is a feature distinguishing services from, e.g., financial products and hard goods. Many commodity trades have a settlement time, for exchanging money and goods. Services are not settled, but provisioned and consumed, which calls for a contract implementation plan accounting for the whole time span.

2.3 Negotiation Issues

A common way of analyzing negotiations is differentiating the *negotiation protocol*, comprising the rules of the encounter, the *negotiation object*, and the *decision-making model* [11]. A number of simple negotiation protocols are used for match-making and reservations without considering economic aspects. For example, SNAP (Service Negotiation and Acquisition Protocol) has been proposed for resource reservation and use in the context of the Grid [8].

The remainder of the paper focuses on direct bilateral negotiations, not involving third parties like regulators, facilitators or mediators enforcing special rules of interaction, although PANDA can be applied in that situation.

A common problem in negotiations is the *ontology problem of electronic negotiations* [20]. It deals with the common understanding of the issues among negotiating parties. One approach of solving the ontology problem is the use of templates. Tem-

plates are partially completed contracts whose attributes are filled out in the course of the negotiation. Template-based negotiations facilitate structuring of the negotiation process and understanding of resulting service contracts [18].

2.4 Requirements

The discussion in this section leads to a set of requirements to be addressed by a negotiation decision-making approach:

- The approach should impose few restrictions on the **negotiation protocol and object**.
- The policy representation should **trade off** expressiveness and ease of specification. It must allow a policy specifier to structure the policy along his or her thinking and to understand and manage real-life policies.
- The **practicality** of a negotiation system entails that a regular user should not be required to have programming skills. A negotiation is not an end in itself, but means to an end.
- **User interaction** should be possible but not mandatory. Automated decision-making is capable of speeding up the negotiation process and reducing its costs. Nevertheless, user interaction may be necessary.
- The approach must consider the **specific properties of services**, which are: non-storability, complexity, intangibility and need for provisioning.

3 Capturing Intended Behavior

Utility functions and rule-based systems are two standard methods for externalizing preferences and intended behavior. However, both approaches pose difficulties for users to express complex strategies for negotiating service agreements.

3.1 Examples of Decision Relevant Considerations

Deciding whether or not to accept an offer or to create a counter-offer may involve evaluations of many different aspects of the contract and checking many decision criteria. Below we illustrate a typical set of evaluation criteria for service offers addressed in negotiation policies:

- *Can this contract be supported?* Before accepting a new contract a provider needs to make sure it can be supported given the set of existing contracts and available resources. Answering this question may involve a detailed model of the system [6] and evaluating the expected violations with this new contract.
- *How desirable is this contract?* A highly profitable contract may be desired over an existing one even if not enough resources are available. The business may have decision criteria on terminating an existing contract not just based on profitability but many other aspects such as business reputation and customer satisfaction.

- *Will the counterparty lose interest?* A counter-offer selection can not be guided simply by maximization of profit. The business may have decision criteria on how far to deviate from a client offer or how much to concede.

For some of the above considerations the attributes of the contract template are insufficient for the decision-making process. The evaluation involves a complex estimation of one or more additional decision parameters, such as probabilistic measure of risk, resource costs and desirability of a new contract. This includes the attributes of the contract implementation plan. We refer to program components that derive these additional parameters as *estimation programs*. Both, attributes of the template and additional decision parameters can be used in utility functions and rules.

3.2 Utility Functions

Economists have been using utility functions as representation for preferences since the 19th century. A utility function maps properties related to an offer onto a single dimensional abstract utility value for an individual or an organization. The utility value is then used as a representation for the individual's preferences: alternative A is preferred to alternative B if, and only if, A's utility value is greater than B's.

In automated negotiations maximization of externalized utility functions can be used for decision-making. Hence, utility functions have to embrace four preferential aspects:

1. *Risk*[1] is a part of service negotiations mainly because of the non-storability characteristic resources might possess.
2. Complex service agreements often define multiple issues or attributes. *Multi-attribute* considerations are a well-known component of decision and negotiation analysis.
3. *Time* influences negotiations in two ways: In the short run there might be deadlines for reaching an agreement. A consumer may have a fixed time line and a provider may want to have his resource booked in advance to avoid risk. In the long run the continuity of business relations might become important.
4. *Inter-personal* elements deal with bounded self-interest, i.e. an individual's utility may depend not only on his own situation, but as well on the situation others face, examples being altruism and positional goods.

It is a well-known problem that even if an individual's preferences match a specific utility function, it is hard to externalize this function: An individual usually does not know his or her personal utility function. It has to be elicited [10, 17]. Service providers can build their utility functions by analyzing a services business model and the cost structure obtained from the IT controlling. Typical fixed costs are, e.g., the depreciation of servers. Typical variable costs are ISP bandwidth, electricity, and personnel. Consumers can derive a bigger part of their utility functions by estimating the costs of in-house provisioning and considering the benefits from service usage.

---

[1] In decision theory there is a difference between *risk* and *uncertainty*. Both terms refer to random outcomes. However, risk implies a mathematical probability measure of the outcomes, uncertainty does not. Within this paper we shall use the term *risk* to refer to either situation.

Considering the preferential elements, it is easy to imagine that one single function capturing all aspects might become complex.

FastQuote's desired price/delay coherency is an example for multi-attribute aspects within a negotiation. It can be modeled by means of a utility function, e.g., the one presented in figure 1. A higher price is better for FastQuote; consequently the function is strictly monotonic increasing in the price. The delay however determines the curvature of the function. The higher the delay, the more concave the function. This implies that for a given price the utility is higher when the delay increases.

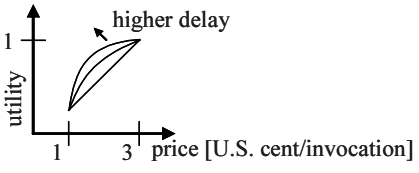

Fig. 1. Multi-attribute utility function representing FastQuote's price and delay preferences.

Integrating FastQuote's first time customer cordiality in a utility function is not straightforward. It could be done by increasing the functions dimensionality, but definition, calculation, and optimization would become far more demanding.

3.3 Rule-Driven Decision-Making

Directions for decision-making can be also represented in form of rules. There are numerous approaches for expressing business rules but no absolutely best way because one has to make a trade-off between expressive power and ease of use while ensuring automated execution. Negotiation rules have to express knowledge in a high-level machine executable language on a high degree of abstraction, which is closer to the understanding of business domain experts, who are often not educated in programming languages, than hard-coded if-then constructs. The rule corpus should be simple to modify by adding or removing single rules and execution of procedural attachments should be possible. Procedures are necessary for informational input to the rules, i.e. as sensors, and for conducting desired actions, i.e. as effectors. Some procedures might perform optimization tasks, such as implementing a trade-off heuristic for computing an adequate counteroffer.

Rule-driven strategies are integrated in some negotiation infrastructures like the ones from Su et al. [21] and Benyoucef et al. [2]. For some kinds of decisions, rules can be seen as a more human-like way of thinking than the maximization of utility, hence they are easier to elicit.

FastQuote's first time customer cordiality can be easily expressed with an if-then construct, in prose this might be: "If the customer's offer is close to FastQuote's last offer and the customer is a new customer then the offer is accepted." The price/delay coherency would be much more difficult as a rule expression, than it was in form of a utility function. The curves could be approximated by step functions defined by a sequence of rules, but a continuous function is easier to define, optimize and manage.

3.4 Mixed Policies: Externalized Negotiation Behavior

Given the advantages and drawbacks of rules and utility functions in terms of expressiveness, manageability and ease of elicitation, we propose to combine both approaches for the representation of negotiation policy. The mixed policy approach requires a model of decision-making that defines how rules relate to utility values. This includes:

- Definition of points of decision-making,
- The association of utility functions with those points, and
- The definition of objects that can be subject to rule expressions.

The PANDA approach addresses exactly these design issues and consequently helps building negotiation applications.

4 PANDA Framework

The model of decision-making is designed as an object-oriented framework. The framework decomposes the decision-making task and identifies points of decision-making, assigns different utility functions to these points, and specifies the objects that can be accessed for rule-based reasoning.

4.1 Negotiation Object and Protocol

Automated decision-making depends on the negotiation object and the protocol. In the proposed approach, the negotiation object is a contract template [16, 18], where a template comprises fixed and variable parts. The fixed ones are usually the general terms and conditions, the variable parts may either be negotiable, such as quality-of-service attributes, or non-negotiable such as the names of the parties. Variable parts may contain single values and may be restricted by ranges, enumerations of values, simple logical constraints, and inter-attribute constraints. In template-based negotiations the meaning of negotiable issues is clearly defined as parts of the template.

A negotiation protocol is a set of rules governing the interaction. The assumptions on the protocol made by the framework are fairly weak for keeping it applicable to many negotiation scenarios. The basic structure is a bilateral message exchange. The integration of an intermediary does not impose changes in the PANDA framework. Either party can start a negotiation with a *request for negotiation* indicating the template to use. Follow-up messages are of the types *accept*, *reject*, *offer*, *withdraw*, or *terminate*. The parties are neither required to alternate with sending messages nor to agree on the utilization of the decision-making framework. *Accept* leads to a contract based on the other party's last offer, *reject* to the rejection of the last offer. Offer indicates that a (partially) filled template is sent as proposed contract, *withdraw* annuls the last offer, and *terminate* ends the entire negotiation process immediately.

4.2 Decision-Maker Components

The PANDA framework's architecture is built around *decision-maker* (DM) components. The internal structure of a single DM is illustrated in figure 2. Its primary task is to combine a set of *utility functions* and processing a rule set, stored in a *XML Repository*. Rules and utility functions both have access to an *object pool*, containing data items and functions that the utility functions can evaluate and the *rule interpreter* can reason on. The object pool can, e.g., contain *estimation programs* (see 3.1), the *negotiation history* and other objects.

Rules are expressed in a high-level language specified by an XML schema. The primary goal of this language is to allow a business domain expert to specify negotiation strategies without having to deal with the programmatic implementation of the decision-making system. The implementation details of the object pool members are abstracted by means of sensors and effectors, which can then be used within rules.

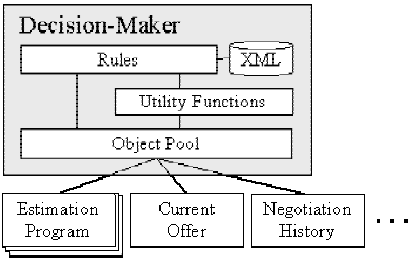

Fig. 2. Building blocks of a decision-maker component

The basic building block of a strategy is a single rule, consisting of a condition part and an action to perform, if the condition is met. A condition is a Boolean expression, composed of Boolean- and mathematical operators, constants, and so called data sources. An action is simply a series of data sources. In conditions and actions likewise a data source is build from a sensor or an effector and a list of parameters. Sensors and effectors are defined separately and map their name to the call of a procedure belonging to one of the object pool's elements. Parameters can either be constants, data sources, or objects. Following is a rudimentary rule example, including the sensor *LEVEL_OF_DISSENT* and the effector *ACCEPT_OFFER*, the definition of both is omitted.

An example rule, in prose, is: *If the level of dissent is less than 0.05 accept the counterparty's offer. The level of dissent is defined as the utility difference between the party's last offer and the counterparty's last offer.*

The same rule in the proposed XML representation:

```
<Rule>
  <Condition>
    <BooleanExp>
      <RelationExp>
        <Operator>LESS</Operator>
        <Sensor>
          <Name>LEVEL_OF_DISSENT</Name>
        </Sensor>
```

```
          <Constant>0.05</Constant>
        </RelationExp>
      </BooleanExp>
    </Condition>
    <Action>
      <Effector>
        <Name>ACCEPT_OFFER</Name>
      </Effector>
    </Action>
  </Rule>
```

Rules are assembled to rule sets. A strategy can contain an arbitrary number of rule sets. The clustering of rules allows inducing control flow within the strategy interpretation. Some of the control-flow aspects are iterations over a rule set (i.e. a *while* loop), stopping a rule set's processing after a certain rule within it was fired (i.e. a *break* statement), and the unconditional processing of a rule set at the end of the strategy interpretation (i.e. a *finally* statement).

4.3 Combining Multiple Decision-Makers

The PANDA framework decomposes behavior externalization by delegating different aspects to different decision-maker components. Different DMs can then be assembled for getting the overall intended behavior. Figure 3 exemplifies FastQuote's negotiation system architecture with three decision-makers: the *negotiating agent* (NA), the *negotiation coordinator* (NC), and the *utility update* (UU).

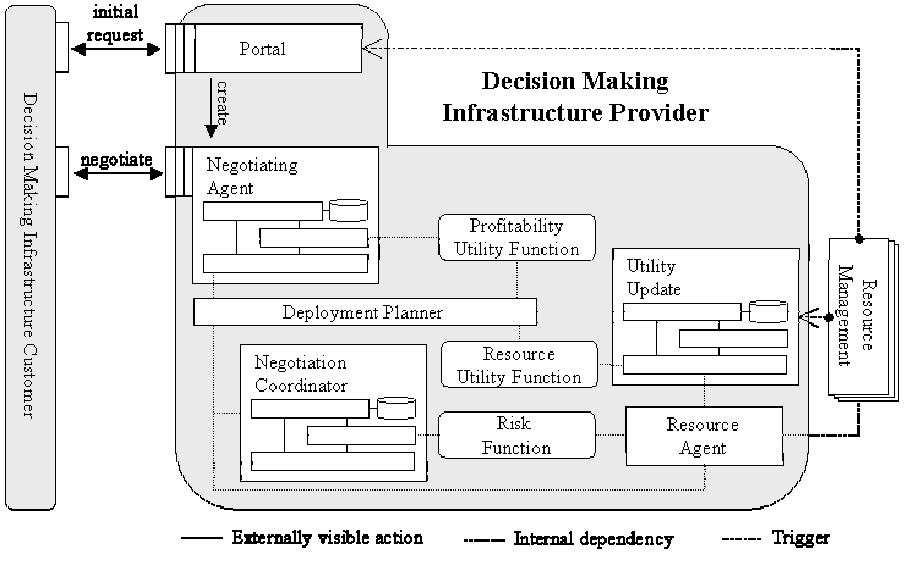

Fig. 3. Combination of decision-maker components in a single negotiation system

This configuration is appropriate for FastQuote's needs, but not mandatory within the framework. PANDA allows the combination of an arbitrary number of DMs. The UU

could be needless in some scenarios; others might require additional components like, e.g., the integration of a customer relationship component. DMs can either be objects directly accessible from another DM, or they can be detached and indirectly affect others by changing their objects.

A negotiation is initiated through the *portal*, either by handling an incoming request or triggered by changes in the resource management. The portal creates a NA instance for each new interaction and endows it with information on the counterparty, the template to use, and the relevant negotiation policy and objects. In figure 3 the NA, the *deployment planner* (DP) and the *profitability utility function* (PUF) are individually instantiated for each negotiation. The other components are singletons. After creation the new NA is registered at the NC and takes control of the interaction with the counterparty.

- **Negotiating Agent (NA):** The NA is the focal point of negotiation handling and all other components within the decision-making infrastructure, except the Portal, support it by providing information on the current situation and environmental conditions. Upon reception of a message, the NA performs some administrative tasks such as checking the validity of the incoming message before proceeding to more sophisticated message handling depending on the message type. Upon reception of a termination or acceptance the procedure is straightforward: cleaning up all negotiation dependent objects and possibly canceling resource reservations, or passing the final contract on for deployment. Otherwise, the NA processes its rule corpus for producing a response message to send or for deciding to wait.

  The NA can via its object pool (partially outlined in figure 3) obtain information on all messages in the negotiation history, perform searches for utility maximizing points within offer-spaces and access counter-offer tactics such as trade-off heuristics [9] and if necessary user input. The *resource agent* (RA) establishes the interface to *resource management* and forecasting systems and is able to handle reservation and information requests. The PUF is a multi-attribute utility function, evaluating a *contract template*. Since service provisioning involves the use of resources which are often not explicitly defined in the service contract, the PUF cannot evaluate all contract attributes and has to invoke the *resource utility function* (RUF). The RUF is another multi-attribute utility function. It aggregates the resource requirements for deploying a service to a single utility value that is further processed by the PUF.
- **Negotiation Coordinator (NC):** The NC is designed to coordinate multiple NAs, as each single one of them is ignorant of its siblings and the work load risk. Each NA can invoke the NC for receiving a level of giving in, which is determined by processing the NCs rule corpus. This provides an indication on how strongly a NA should concede the counterparty. The NC can reason on status reports requested from all NAs and it has access to the *risk function* (RF). The RF is the third utility function within FastQuote's decision-making system. Unlike PUF and RUF, it doesn't consider multi-attribute aspects, but takes the work load risk into account. This enables a business domain expert to incorporate the desired level of committed resources.
- **Utility Update (UU):** The UU cannot be directly accessed from the NA, but modifies the RUF which might be volatile and depended on the resources' work loads. A sparse resource might, e.g., have more influence within the function, than an

abound one. As the individual resources' loads change over time, the RUF can adapt. The update process is triggered by a resource management system and comprises the interpretation of a utility update strategy by the UU. The strategy can use data, obtained from the RA, to change the RUF during runtime.

The **Deployment Planner (DP)** is an important component besides the three decision-makers. As figure 3 indicates, the NA consults the DP when calling the NC, RA, or indirectly the RUF. Often, the utility depends on the resources consumed, specified in the contract implementation plan, in addition to the negotiated parts of the contract. The decision-making infrastructure must map from the contract to the contract implementation plan. The DP determines resource requirements by transforming contract attributes.

The DMs and the DP described above facilitate the manageable specification of a negotiation policy comprising utility functions and rules. The PANDA framework is implemented in Java and defines the basic components as well as the control flow for decision-making in negotiations. Existing object types can be extended and additional object types can be added to decision-makers by using the mechanisms provided by Java inheritance.

5 Policy Example

The example illustrates a possible negotiation strategy of FastQuote and its potential customer NewsOnline, a personalized online newspaper. Due to a change in subscriber behavior NewsOnline has to increase its capacity in stock quotes and initiates a service negotiation with FastQuote.

5.1 FastQuote's Negotiation Policy

Besides publishing a contract template, FastQuote internally specifies a contract implementation plan and a negotiation policy: rules for the negotiating agent and the coordinator, a profitability utility function, a resource utility function, and a risk function. The following rules are not expressed via XML for space restrictions; capitalized words indicate sensors and effectors.

The rules of the NA are:

```
RuleSet:
  Rule NA1:
   if     LEVEL_OF_DISSENT < 0.05
   then   ACCEPT; break;
  Rule NA2:
   if     LEVEL_OF_DISSENT < 0.2 and NEW_CUSTOMER
   then   ACCEPT; break;
  Rule NA3:
   if     LEVEL_OF_DISSENT > 0.2
   then   FIND_TRADE_OFF_OFFER(LAST_UTILITY - 0.5*LGI);
          MAKE_OFFER; break;
  ...
End
```

The rules of the NC are:

```
RuleSet:
  ...
  Rule NC4:
   if     RISK_UTILITY < 0.65 and RISK_UTILITY > 0.5
   then   LGI = (1-RISK_UTILITY) / NUMBER_NEGS; break;
  ...
End
```

The three utility functions are jointly outlined in figure 4. RUF and PUF are both computed by evaluating the sub-functions in the leaf nodes and aggregating them by building the weighted sum. The weights are displayed at connecting edges. The RF first aggregates work load forecasts for the two resource types by taking the maximum and then maps this maximum on the interval from zero to one. 70% is the target work load that FastQuote wants to achieve.

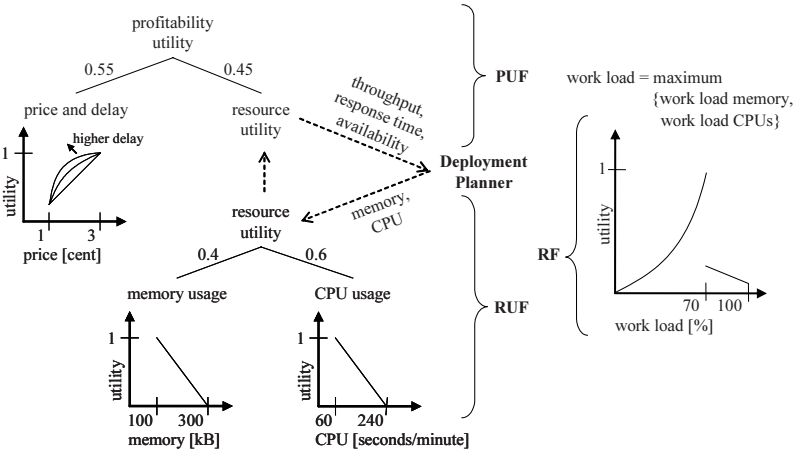

Fig. 4. Example utility function composition.

5.2 Creating a Counteroffer

Figure 5 gives a snapshot of the offers exchanged between FastQuote and NewsOnline.

Prices are given in cents per invocation, delay is zero, 5, or 20 minutes, throughput in invocations per minute, response time in seconds, and availability is either 98%, 99%, or 99.9%.

Sender	…	FastQuote	NewsOnline	FastQuote	…
Message No.		5	6	7	
Price		2.4	1.5	2.3	
Delay		0	0	0	
Throughput		400-600	800	600	
Response time		≥ 3	≤ 3	3	
Availability		99%	99.9%	99.9%	

Fig. 5. Example of an offer sequence during a multi-attribute service negotiation.

The creation of FastQuote's offer number 7 goes as follows: FastQuote's NA receives message 6, approves it as valid offer, and the starts processing its rule corpus containing a single rule set. Rule NA1 invokes the sensor LEVEL_OF_DISSENT, which computes the utility difference of offers 5 and 6. Computing utility values involves invoking the PUF, which passes the attributes throughput, response time, and availability on to the DP. The DP maps these three attributes to resource requirements and calls the RUF. The resulting utility values might, e.g., be 0.78 and 0.54 respectively. As a result the level of dissent is 0.24. Rule NA1 does not fire, neither does NA2.

Rule NA3 fires and the effector FIND_TRADE_OFF_OFFER invokes a method in the object pool for calculating a reasonable counteroffer. The calculation comprises searching for a contract which utility value equals at least the one given as a parameter and which is close to NewsOnline's last offer. The closeness makes it likely to be acceptable by NewsOnline. The parameter, i.e. the desired utility value, is calculated by integrating the coordinator via the sensor LGI, i.e. the level of giving in.

The coordinator consults the RA, receives work load forecasts of 60% for memory and 53% for CPUs, and aggregates them by taking the maximum. The corresponding RISK_UTILITY value of 0.64 is derived from the risk function and used during rule processing. Rule NC4 is the first rule to fire and the sensor NUMBER_NEGS simply counts the number of ongoing negotiations for services affecting either memory or CPUs. Currently their might be three negotiations, including the one with NewsOnline. The level of giving in is hence set to 0.12.

LAST_UTILITY is currently the utility value of offer 5, i.e. 0.78; therefore the parameter of the FIND_TRADE_OFF_OFFER effector is 0.72. Rule NA3 sets the type of message 7 to *offer*. It is not yet send at this point, as following rules might change either the offer's attribute values, or the message type, e.g., to *accept* or *terminate*. However the break statement in NA3 exits the single rule set. The NA identifies a valid message type and filled out template and sends message 7 to NewsOnline.

6 Conclusion and Future Work

The Policy-driven Automated Negotiation Decision-making Approach (PANDA) proposes a novel mechanism for the specification of a party's negotiation policy, i.e. the private specification that guides the analysis of offers and creation of responses. In the context of fine-granular Web services that are to be bound and integrated into composite Web services across domain boundaries, organizations need to automate the negotiation process for new service usage as far as possible to make a service-based business model viable. To enable automated negotiations it is important that organizations can specify their negotiation policies in a concise and easy way.

PANDA enables the representation of a negotiation policy based on the novel combination of utility functions and rules. This policy is executed by decision-maker components within the PANDA framework. The combination of multiple decision-makers facilitates the decomposition of the policy. Using this approach, a specifier can divide the decision problem into manageable units according to his or her understanding. Different aspects of the decision problem such as profitability of an offer

and resource situation can be specified separately for different decision-makers and can refer to each other. Using this approach, PANDA allows the specification of sophisticated negotiation behavior in a manageable way.

The approach is agnostic to specific negotiation protocols, although a particular specifier must understand them. PANDA has a template-based approach to represent the negotiation object and can deal with arbitrary service contracts. The high-level rule language helps keeping programming requirements low. In addition, the framework facilitates the involvement of users in the decision-making process, if necessary. The example shows how to use PANDA in a Web services context and hence addresses the specific needs of service negotiations.

Related work has been published on various aspects of negotiations, decision-making and rules, as discussed in the paper. Particularly relevant are the following contributions: The negotiation server by Su et al. uses rules to describe how to relax constraints defining acceptable offers in the course of the negotiation [21]. Cost-benefit analysis is used to choose between multiple acceptable offers. While no suitable offers are found, this approach does not benefit from the use of utility functions to guide the negotiation process. The complexity of utility functions and contract implementation plans is addressed by Boutilier et al. [3]. This approach is used for collaborative resource allocation within an organization and does not address negotiations across organizational boundaries.

Future work will address the specification of utility functions in an externalized representation. Also, the framework extensions are planned to implement a library of common object pool elements such as estimation programs. Furthermore, users would benefit from a policy editor that supports the creation of specifications. Work is being conducted to connect the negotiation framework to an automated deployment function in the Grid context. Finally, an experimental evaluation of the mixed policy approach is necessary.

References

1. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana: *Business Process Execution Language for Web Services, Version 1.1*. 2003.
2. M. Benyoucef, H. Alj, K.Levy, R. Keller: A Rule-Driven Approach for Defining the Behavior of Negotiating Software Agents. *Proceedings of the Fourth International Conference on Distributed Communities on the Web.* Sydney, 2002.
3. C. Boutilier, R. Das, J.O. Kephart, G. Tesauro, W.E. Walsh: Cooperative Negotiation in Autonomic Systems using Incremental Utility Elicitation. *Proceedings of Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI 2003)*. Acapulco, 2003.
4. D. Box, F. Curbera, M. Hondo, C. Kaler, D. Langworthy, A. Nadalin, N. Nagaratnam, M. Nottingham, C. van Riegen, J. Shewchuk: *Web Services Policy Framework (WS-Policy), Version 1.1*. 2003.
5. R. Chinici, M. Gudgin, J-J. Moreau, S. Weerawarana: *Web Services Description Language (WSDL), Version 1.2, Part 1: Core Language*. W3C Working Draft, 2003.
6. C. Crawford, A. Dan: eModel: Addressing the Need for a Flexible Modeling Framework in Autonomic Computing. *IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*. Fort Worth, 2002.

7. K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, M. Xu (eds.): *Agreement-based Grid Service Management (OGSI-Agreement), Version 0*. 2003.
8. K. Czajkowski, I. Foster, C. Kesselman, V. Sander, S. Tuecke: SNAP: A Protocol for Negotiation of Service Level Agreements and Coordinated Resource Management in Distributed Systems. *Job Scheduling Strategies for Parallel Processing: 8th International Workshop (JSSPP 2002)*. Edinburgh, 2002.
9. P. Faratin: *Automated Service Negotiation between Autonomous Computational Agents.* Ph.D. Dissertation. University of London, 2000.
10. Y. Guo, J.P. Müller, C. Weinhardt: Learning User Preferences for Multi-attribute Negotiation: An Evolutionary Approach. *Multi-Agent Systems and Application III, Proceedings of the 3rd Int./Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003)*. Prague, 2003.
11. N.R. Jennings, P. Faratin, A.R. Lomuscio, S. Parsons, C. Sierra, M. Wooldridge: Automated Negotiation: Prospects, Methods and Challenges. *International Journal of Group Decision and Negotiation.* 10 (2), 2001.
12. A. Keller, H. Ludwig: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Accepted for publication in: *Journal of Network and Systems Management, Special Issue on "E-Business Management".* 11 (1), 2003.
13. C. Kenyon, G. Cheliotis: Architecture Requirements for Commercializing Grid Resources. *11th IEEE International Symposium on High Performance Distributed Computing (HPDC'02)*. Edinburgh, 2002.
14. G. Lo, G.E. Kersten: Negotiation in Electronic Commerce: Integrating Negotiation Support and Software Agent Technologies. *Proceedings of the 29th Atlantic Schools of Business Conference*. Halifax, 1999.
15. H. Ludwig, A. Keller, A. Dan, R. King: A Service Level Agreement Language for Dynamic Electronic Services. *Proceedings of WECWIS 2002*, Newport Beach, 2002.
16. H. Ludwig: A Conceptual Framework for Electronic Contract Automation. *IBM Research Report*, RC 22608. New York, 2002.
17. H. Raiffa, J. Richardson, D. Metcalfe: *Negotiation Analysis*. The Belknap Press of Harvard University Press, Cambridge, 2003.
18. D.M. Reeves, M.P. Wellman, B.N. Grosof, H.Y. Chan: Automated Negotiation from Declarative Contract Descriptions. *Computational Intelligence*, 18, 482–500, 2002.
19. A. Sahai, A. Durante, V. Machiraju: Towards Automated SLA Management for Web Services. *Hewlett-Packard Research Report HPL-2001-310 (R.1).* Palo Alto, 2002.
20. M. Ströbel: *Engineering electronic negotiations*, Kluwer Academic Publishers, New York, 2002.
21. S.Y.W. Su, C. Huang, J. Hammer: A Replicable Web-based Negotiation Server for E-Commerce. *Proceedings of the Thirty-Third Hawaii International Conference on System Sciences (HICSS-33)*. Maui, 2000.