XV Jornadas de Ingeniería del Software y Bases de Datos
JISBD 2006
José Riquelme - Pere Botella (Eds)

ENGINEERING AUTOMATED NEGOTIATIONS

Manuel Resinas[1*], Pablo Fernandez[1] and Rafael Corchuelo[1]

1: Dpto. Lenguajes y Sistemas Informaticos
ETS Ingenieria Informatica, Universidad de Sevilla, Avda. Reina Mercedes, s/n, 41012 Sevilla
e-mail: resinas@us.es, web: http://www.tdg-seville.info

Palabras clave: software architectures, web engineering, SLAs, negotiation

Resumen. *To make automated negotiation widely used in real scenarios, it is necessary to develop advanced software systems that are able to carry out such negotiation in those scenarios. However, although much work has been done in automated negotiation, most of these efforts have been focused on the development of negotiation protocols and strategies and only a few are centred on software frameworks for automated negotiation. We are concerned with the engineering aspect of automated negotiation. Specifically, we detail a software framework to develop automated negotiations of service level agreements. Unlike other proposals, we provide a detailed description of elements for full decision-making support including evaluating offers, strategy selection, building a market and opponents model, and creating counterproposals.*

1 INTRODUCTION

An automated negotiation process can be understood as a search, in the space of possible agreements, of a mutually acceptable agreement by the parties that are carrying out the negotiation. In this work, we focus on SLA negotiations[1]. The goal of this kind of negotiation is to reach an agreement between a service provider and a service consumer about the terms and guarantees of the service consumption. In the last years, much work have been done on the development of new decision-making algorithms or the construction of new protocols that present certain desirable characteristics for automated negotiations [5]. However, much less attention has been paid to the software artefacts that are necessary to carry out this automated negotiation.

In this paper, we are concerned with the software engineering aspect of automated negotiation. Namely, we report on a software framework to build automated negotiation systems. The novelty of our proposal lies in two characteristics. First, we use a service-oriented approach to precisely define the elements (services and data) of the framework. Second, unlike other proposals, we cover all required elements that are necessary to create

[1]Also known as service-oriented negotiations [10]

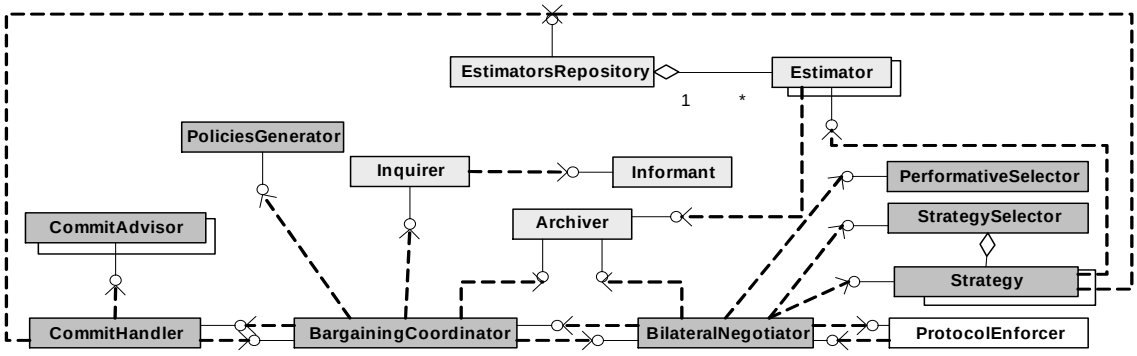

Figura 1: Services overview

a complex automated negotiation system including world modelling, decision-making, preferences, negotiation protocols and negotiation object [8].

The structure of this paper is the following. First, in Section 2 we detail the software framework. Next, we report on related work in Section 3, and we conclude in Section 4.

2 A SOFTWARE FRAMEWORK FOR AUTOMATED NEGOTIATION OF SLAs

In this work, we focus on the particular case of concurrent proposal-based bilateral negotiations of SLAs. That is, there are two participants and the information exchanged consists exclusively in agreement proposals. The rationale is: (i) proposal-based bilateral negotiations are a very well studied field and mature enough to start developing software systems that work in a real scenario; (ii) it is aligned with recent efforts of the industry in this area, such as WS-Agreement [1], and (iii) other approaches are not well suited to deal with complex SLAs with multiple terms (e.g. auctions) or may lead to complex systems with less techniques developed for them (e.g. argumentation-based approaches).

The goal of the proposed software framework is to provide engineering support for software developers to build automated negotiation systems that implement the conceptual framework for automated negotiations described in [8], including support for negotiation object, preferences, world modelling, negotiation protocol and decision-making. The software framework developed uses a service-oriented approach. Figure 1 depicts an overview of the services[2] that compose the framework and their relationships. The boxes represent the services of the framework; the circles, the interfaces, and the arrows represent a relationship between the services. The framework provides a specification for each service and a reference implementation for two of them: *bilateral negotiator* and *bargaining coordinator*. The remaining services are intended to be implemented following the algorithms already developed in the automated negotiation literature.

[2]Note that not all relations are depicted in the diagram.

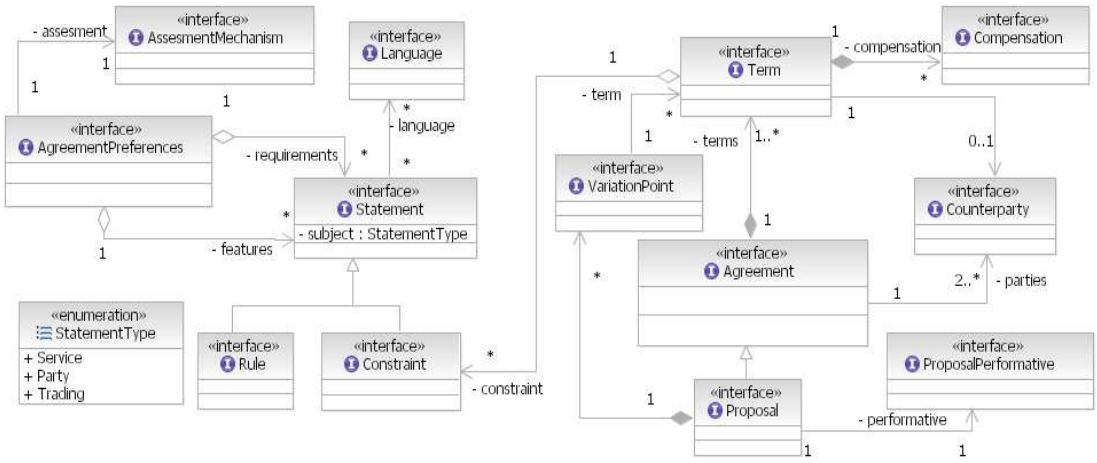

Figura 2: Data structures

2.1 Negotiation object and preferences

Figure 2 depicts the data structures implemented in the framework to support the negotiation object and the preferences of the user. The negotiation object is an *agreement* that it is composed of the *parties* involved in it and a set of terms that define its content. A *term* includes the *counterparty* to whom it is applied, constraints to specify functional or non-functional descriptions or guarantees of the service, and, optionally, compensations that must be paid if the term is not fulfilled.

The *agreement preferences* are composed of a set of statements that expresses features and requirements about the service and the desired characteristics of the negotiation process, and an assessment mechanism that is used to evaluate the proposals received. A statement is an expression, usually rules or constraints, that can be applied over a service, a party or the negotiation process itself, and that it is linked to a set of languages that give semantics to the vocabulary used in it.

2.2 Protocol management

The software framework gives support for the negotiation protocol as follows: on the one hand, the information exchanged is expressed by the proposal data structure (see Figure 2). A *proposal* extends the agreement data structure by including a negotiation *performative* and allowing for the *terms* of the proposal to be left opened in order to be refined later on. A negotiation performative is the expression of the intention of the sender of a message about it. The minimum subset of performatives is *accept*, *reject proposal*, *reject negotiation*, and *propose*. On the other hand, the *protocol enforcer* (the white box in Figure 1) service acts as a proxy between the *bilateral negotiator* service and the counterparty, it adapts the concrete protocol messages into the proposal data structure

and vice versa, it provides the performatives that can be used in a certain moment of the negotiation, and ensures that the protocol rules [2] are obeyed.

2.3 World modelling services

The goal of these services is to build a world model that is necessary for the decision-making services to carry out their tasks. There are five types of world modelling services (the light grey boxes in Figure 1). The *inquirer* service directly polls the *informant* service of other counterparty to obtain objective information about it. The *estimator* services build models based on subjective information about other participants [13], the market [11] and the service domain. Therefore, an automated negotiation system may use a reputation estimator service, an opponent behaviour estimator service, or a proposals similarity estimator service [3]. These estimators may be internal to our organisation or implemented by third parties. Estimator services must implement a query interface and may implement an update interface to upgrade their information about the world. Alternatively, they may use other sources of information such as querying the *archiver* service. The *archiver* service serves as a storage of the history of all negotiations that have been developed by the system. Finally, the *estimators repository* service decouples the services of the framework from a certain estimator service by acting as an intelligent registry of estimator services.

2.4 Decision-making services

The decision-making services (the dark grey boxes in Figure 1)defined by the software framework provide mechanisms to support the binding decision and response generation.

Response generation The services for response generation can be divided into two groups: negotiation services and coordination services. There are four negotiation services: the *bilateral negotiator* service coordinates all other negotiation services in order to carry out a proposal-based bilateral negotiation with one counterparty. Specifically, it firstly uses a *performative selector service* to choose which response shall be sent to the counterparty amongst all possible performatives. Then, if a counterproposal has to be built, it uses a *strategy service*. This service builds a counterproposal given the last proposal submitted by the counterparty and the models provided by world modelling services. In addition, a *strategy selector service* may be previously used to choose the most appropriate strategy service. Therefore, this service can be seen as an intelligent registry of strategy services.

There are a number of approaches to coordinate several negotiations. In [7], the coordinator indicates which strategy must be used in each negotiator and it is informed of each proposal received from the counterparty. Instead, here we propose a more decoupled approach. The *bargaining coordinator service* coordinates several *bilateral negotiators* and acts as a bridge between them and the binding services. It periodically receives informa-

tion about the status of the negotiations and it provides them with policies that guides their behaviour (e.g. "*be more agressive*"). To obtain these policies, it uses the *policies generator services*.

Binding decision Two types of services provide support for the binding decision. The *commit handler* service has the final decision on when a binding proposal must be submitted and whether a binding proposal that has been received should be accepted. In addition, it must establish when these decisions are going to be made. The *commit handler* service may use *commit advisor* services (e.g. a capacity planner) in order to obtain advise about the convenience of committing to an agreement proposal.

3 RELATED WORK

While much work has been done in the development of negotiation protocols and strategies, the work on software frameworks that supports automated negotiation has been much more limited. Moreover, most of them only deal with negotiation protocols and do not give support for world modeling and decision-making. This is the case of [2, 9]. In [2], Bartolini et al. present a rules taxonomy for negotiation protocols and a software framework that uses the rules to deal with them. In [9], a service-oriented architecture is used to manage several negotiation protocols. Therefore, these works can be seen as possible implementations of our *protocol enforcer service*.

There have been other proposals [12, 4, 6] of software frameworks for automated negotiation that do provide some limited support to decision-making and world modeling. In [6], a framework for automated negotiation of service level agreements in service grids is presented. This framework builds on WS-Agreement and provides a protocol service provider and a decision making service provider to deal with the negotiation process. However, it lacks world modelling support and all decision-making has to be implemented in one unique service decreasing the reusability.

PANDA [4] is a framework that mixes utility functions and rules to carry out the decision-making process. The decision-making component is composed of rules, utility functions and an object pool with several estimation libraries, the negotiation history and the current offer. However, elements like the object pool are vaguely specified. Moreover, it does not support querying other parties to get information either. Another drawback is that information related to the negotiation process is not managed.

DynamiCS is an actor-based framework developed by Tu et al. [12]. This framework make a neat distinction between negotiation protocol and decision making model and they use a plug-in mechanism to support new protocols and strategies. Nevertheless, it does not take into account external factors in the binding decision (e.g. resources available in the provider), and its world modeling capabilities are very limited: no subjective information is managed, querying information to other parties is not supported, no market models are built and it does not create models based on the analysis of previous interactions.

4 CONCLUSIONS

The work in automated negotiation in the last ten years has lead to a variety of decision-making algorithms and protocols. From these works, we extract two conclusions: (i) for an automated negotiation system to work in an open environment such as the Internet, it is necessary to give support to a number of negotiation protocols; (ii) the most appropriate negotiation strategy depends on the context and, hence, the system must support a variety of negotiation techniques and select the best one in each moment. Therefore, it is convenient to have a software framework that provides engineering support for software developers to build automated negotiation systems. In this paper, we present a service-oriented software framework for concurrent proposal-based bilateral negotiations of SLAs. Unlike other proposals, our framework offers full support for world modelling and advanced decision-making, including changing dynamically of negotiation strategy depending on the negotiation context and support for an advanced binding decision.

REFERENCIAS

[1] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. WS-Agreement Specification, 2004.

[2] C. Bartolini, C. Preist, and N. R. Jennings. A Software Framework For Automated Negotiation. In *Software Engineering For MAS III: Research Issues and Practical Applications*, pages 213–235. Springer Verlag, 2005.

[3] P. Faratin, C. Sierra, and N. R. Jennings. Using Similarity Criteria to Make Trade-Offs in Automated Negotiations. *Artificial Intelligence*, 142:205–237, 12 2002.

[4] H. Gimpel, H. Ludwig, A. Dan, and B. Kearney. PANDA: Specifying Policies For Automated Negotiations of Service Contracts. In *ICSOC 2003*, Lecture Notes in Computer Science, pages 287–302. Springer-Verlag, 2003.

[5] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, M. Wooldridge, and C. Sierra. Automated Negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation*, 10:199–215, 3 2001.

[6] A. Ludwig, P. Braun, R. Kowalczyk, and B. Franczyk. A framework for automated negotiation of service level agreements in services grids. In *Business Process Management Workshops: BPM 2005*, volume 3812 of *Lecture Notes in Computer Science*, pages 89–101. Springer, Feb. 2006.

[7] T. Nguyen and N. Jennings. Managing commitments in multiple concurrent negotiations. *Electronic Commerce Research and Applications*, 4:362–376, 2005.

[8] M. Resinas, P. Fernandez, and R. Corchuelo. A conceptual framework for automated negotiation systems. In *Proceedings of IDEAL 06*, Lecture Notes in Computer Science, page To appear. Springer, 2006.

[9] S. Rinderle and M. Benyoucef. Towards the automation of e-negotiation processes based on web services - a modeling approach. In *Web Information Systems Engineering - WISE 2005*, volume 3806 of *Lecture Notes in Computer Science*, pages 443–453. Springer, 2005.

[10] C. Sierra, P. Faratin, and N. R. Jennings. A Service-Oriented Negotiation Model Between Autonomous Agents. In *Proc. of the 8th Europ. Workshop On Modelling Auton. Agents in a Multi-Agent World*, pages 17–35. Springer-Verlag, 1997.

[11] K. M. Sim and E. Wong. Toward market-driven agents for electronic auction. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 31(6):474–484, 2001.

[12] M. Tu, C. Seebode, F. Griffel, and W. Lamersdorf. Dynamics: An actor-based framework for negotiating mobile agents. *Electronic Commerce Research*, 1(1 - 2):101–117, Feb. 2001.

[13] D. Zeng and K. Sycara. Bayesian Learning in Negotiation. *Int. J. Hum.-Comput. Stud.*, 48(1):125–141, 1998.