An Abstract Architecture for Service Trading.  
On Automated Creation of Agreements  
Manuel Resinas, 52319291-H  
resman@tdg.lsi.us.es

Supervised by Prof. Dr. Rafael Corchuelo

Research Report submitted to the Department of Computer Languages and Systems of the University of Sevilla in partial fulfilment of the requirements for the degree of Ph.D. in Computer Engineering.  
(Research Period)
Contents

1 Introduction 1
   1.1 Research context .......................... 1
   1.2 Motivation and goals ....................... 4
   1.3 Structure of this report ................. 5

2 A conceptual framework for automated negotiations 7
   2.1 Negotiation object ........................ 7
   2.2 Preferences ................................ 8
   2.3 Information ................................ 10
   2.4 Protocol .................................. 12
      2.4.1 Performatives ......................... 13
      2.4.2 Rules ................................. 14
      2.4.3 Information exchanged .............. 14
   2.5 Decision-making ............................ 16
      2.5.1 Binding decision ...................... 16
      2.5.2 Response generation ................. 18
      2.5.3 Decommit decision ................... 19

3 A reference architecture for automated negotiation of SLAs 20
   3.1 Introduction and motivation ................. 20
      3.1.1 Modeling elements ..................... 22
      3.1.2 Phases in the negotiation process ...... 24
   3.2 Data structures ............................ 25
   3.3 Information ................................. 27
      3.3.1 Roles ................................ 28
      3.3.2 Collaborations ......................... 32
   3.4 Agreement Making ............................ 34
      3.4.1 Roles ................................ 35
      3.4.2 Collaborations ......................... 38
   3.5 Binding .................................. 39
      3.5.1 Roles ................................ 40
      3.5.2 Collaborations ......................... 42
   3.6 Related standards and technologies ...... 43
4 Analysis of architectures for automated negotiation of SLAs 45
  4.1 Properties of service trading architectures 45
      4.1.1 Functional properties 45
      4.1.2 Non-functional properties 47
  4.2 Analysis of our reference architecture 48
  4.3 Comparison with other proposals 49

5 Conclusions 51
  5.1 Analysis of previously set goals 51
  5.2 Future work 52
List of Figures

1.1 Contracting process and service trading process ............... 3
2.1 Conceptual framework - Negotiation object .................. 8
2.2 Conceptual framework - Preferences ........................... 9
2.3 Conceptual framework - Information ............................ 11
2.4 Conceptual framework - Protocol .............................. 13
2.5 Conceptual framework - Decision making ...................... 16
3.1 Reference architecture ............................................. 23
3.2 Phases of the negotiation process ............................... 25
3.3 Data structures ...................................................... 26
3.4 Information organisation - Roles and data dependencies .... 29
3.5 Information organisation - Collaborations diagram .......... 32
3.6 Agreement making organisation - Roles and data dependencies 36
3.7 Agreement making organisation - Collaborations diagram .... 38
3.8 Binding organisation - Roles and data dependencies ....... 41
3.9 Binding organisation - Collaborations diagram ............... 43
# List of Tables

3.1 Related standards ........................................ 44

4.1 Comparison of proposals ................................. 49
Acknowledgements

I want to thank to all the people of the LSI Department for their support and motivation. In a special place, I would like to mention my colleague and friend Pablo that is partially responsible of this document. Finally, I would like to thank my supervisor Rafael for his dedicated efforts.
Resumen

En los últimos años, Internet y los servicios web han abierto nuevas formas de subcontratación en la industria del software a través del uso de servicios proporcionados por una compañía como parte del proceso de negocio de otra. Para ofrecer una garantía en la calidad del servicio, se pueden establecer acuerdos de nivel de servicio (SLAs) que fijen los requisitos funcionales y no funcionales que deben ser contemplados por el proveedor y el consumidor del servicio durante la ejecución del mismo. Automatizar la creación de acuerdos, reduciendo al mínimo la participación humana en el mismo, propone las siguientes ventajas: recorta el coste de alcanzar un acuerdo, incrementa la velocidad del proceso de contratación y permite un uso más racional de los recursos, realizando el aprovisionamiento en función de los acuerdos alcanzados.

En este informe, nos centramos en la negociación automática de SLAs. Para lograr que la negociación automática sea usada ampliamente en escenarios reales, es necesario desarrollar sistemas software avanzados que permitan llevarla a cabo en dichos escenarios. Sin embargo, aunque se ha hecho mucho trabajo en negociación automática, la mayoría de esos esfuerzos iban encaminados al desarrollo de protocolos y estrategias de negociación. Esto es, al aspecto algorítmico de la negociación automática.

Sin embargo, hay menos trabajos que describen cómo desarrollar estos sistemas software (es decir, el aspecto de ingeniería del software de la negociación automática). En este trabajo intentamos solucionar esta carencia presentando una arquitectura de referencia para negociación automática que identifica y describe los elementos (subsistemas, componentes, interfaces, tipos de datos y colaboraciones) que deben implementarse en ellos y da guías acerca de cómo desarrollarlos. Esta arquitectura de referencia puede utilizarse luego para definir arquitecturas concretas de negociación automática en dominios específicos y usando distintas tecnologías.

Para desarrollar esta arquitectura de referencia, tomamos un framework conceptual para negociaciones automáticas de SLAs como punto de partida. Al contrario que otras propuestas, este framework conceptual está orientado al desarrollo de sistemas software de negociación automática, en lugar de a algoritmos. A continuación, categorizamos las propuestas más significativas basándonos en el framework conceptual. Luego, especificamos la arquitectura de referencia siguiendo una aproximación basada en roles. Esto significa que está descrita por medio de los roles y colaboraciones que son necesarias para desarrollar un proceso de negociación automática. Finalmente, también enumeramos un conjunto de propiedades para sistemas de negociación automática y las usamos para comparar nuestra propuesta con otras arquitecturas existentes en la literatura.

En resumen, la contribución de este trabajo es cuádruple. Primero, describimos un novedoso framework conceptual para sistemas de negociación automática. Segundo, basándonos en este framework, categorizamos y clasificamos las propuestas más importantes de negociación automática de SLAs. Tercero,
desarrollamos una arquitectura de referencia para negociación automática de SLAs que fija las bases para el desarrollo de arquitecturas más concretas. Y cuarto, definimos un conjunto de propiedades para sistemas de negociación automática.
Abstract

In recent years, Internet and web services have opened new ways of subcontracting in the software industry through the use of services provided by a company in the business process of another one. To offer a guarantee on the quality of the service, service level agreements (SLAs) may be settled to establish both functional and non-functional requirements that must be observed by the service provider during the service execution. Automating the creation of agreements, so that the human participation in the process was reduced to the minimum possible shall bring the following advantages: cutting the cost of reaching an agreement, increasing the speed in the contracting process, and allowing the establishment of new business relationships in a more flexible way.

In this report, we focus on the automated negotiation of SLAs to reach mutually acceptable agreements. 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 centred in the development of negotiation protocols and strategies. That is, the algorithmic aspect of automated negotiation.

Nevertheless, there is no work that describes how to develop those software systems (i.e. the software engineering aspect of automated negotiation). In this work, we fill this gap by presenting a reference architecture for automated negotiation that identifies and describes the elements (subsystems, components, interfaces, data types, or collaborations) that must be implemented in those systems and gives guidelines on how to develop them. This reference architecture may be later used to define concrete automated negotiation architectures in concrete domains and using a variety of technologies.

To develop this reference architecture, we take a conceptual framework for automated negotiations of SLAs as a starting point. Unlike other proposals, this conceptual framework is oriented to the development of automated negotiation software systems, instead of algorithms. Next, we categorise the most significant proposals on automated negotiation based on the conceptual framework. Then, we specify the reference architecture following a role-based approach. This means that the reference architecture is described through the roles and collaborations that are necessary to develop an automated negotiation process. Finally, we report on a set of properties for automated negotiation systems and we use them to compare our proposal with other architectures proposed in the literature.

In summary, the contribution of this work is fourfold. First, we describe a novel conceptual framework for automated negotiation systems. Second, on the basis of this conceptual framework, we categorise and classify the most significant proposals of automated negotiation of SLAs. Third, we develop a reference architecture for automated negotiation of SLAs that settles the basis
for developing concrete architectures for automated negotiation. And fourth, we define a set of properties for automated negotiation systems.
Chapter 1

Introduction

In this chapter, we introduce this research report. In the first section, we establish the context of our research. In Section 1.2, we outline some concepts on automated negotiation, motivate our research work and detail the goals that we want to reach with it. Finally, in Section 1.3, we describe the structure of this document.

1.1 Research context

Service Oriented Architecture (SOA) is evolving into a mature concept used widespread in software industry. This architecture is the result of a widely accepted web service framework [9] composed of an array of layered standards that are continuously extended.

SOA is useful in both, intra-organisational and cross-organisational scenarios [7]: (i) On the one hand SOA provides a flexible way to integrate different systems in an enterprise; regarding this scenario it has been proposed [32] the idea of an enterprise service bus [37] as the next generation of integration middleware that would act as a central infrastructure of the whole organisation. (ii) On the other hand, interoperability amongst companies has been the holy grail different technologies have pursued for decades. Web services initiatives can be seen as a step forward in this context since they provide an array of open standards that make it possible loosely coupled systems to interact.

Service level agreements (SLAs) are used by many different service industries to grant guarantees about how a service will be provided or consumed by establishing both functional and non-functional requirements that must be fulfilled by both parties during the service development. Therefore, in a SOA, the service level agreements becomes a key point. Furthermore, although traditionally, SLAs have been associated with cross-organisational transactions where a company must enforce a certain level of service to their partners, as SOA is making its way into the integration of complex organisations, SLAs are starting to be an important issue to be addressed amongst the subsystems involved. One
of the main reasons to adopt SOA in the integration is to rationalise the usage of resources inside an organisation which argues for SLAs to be managed as automatically as possible.

Additionally, in recent years, Internet and web services have opened new ways of subcontracting in the software industry through the use of services provided by a company in the business process of another one. To offer a guarantee on the quality of the service, SLAs must be settled to establish both functional and non-functional requirements that must be observed by the service provider during the service execution.

Finally, the use of the service-oriented computing paradigm in Grid environments has boosted the idea of Grid Services by virtualising the resources of the grid as services. In this context, SLAs may be used to represent the requirements and obligations for completing one job in a submission scenario. This use of SLAs is essential for the next generation of Grid Computing technologies where the orchestration of complex connected problems on a distributed environment requires a number of autonomic properties. The autonomic properties of a system are: self-configuring (ability to adapt to changes in the system), self-optimising (ability to improve performance), self-healing (ability to recover from errors), and self-protecting (ability to anticipate and cure intrusions). In our case, we are interested in the self-configuring component of an autonomic computing system. A self-configuring system must be able to change itself to adapt to diverse changes produced in the environment or the system itself. In a distributed environment, this self-configuring may entail the autonomic allocation or reallocation of tasks or resources in the system. That is why the use of SLA is essential to build such systems.

The application of Internet-based technologies for electronic commerce to establish and manage these SLAs offers significant advantages to the traditional use of SLAs. Specifically, automating the creation and management of SLAs, so that the human participation in the process is reduced to the minimum, brings benefits such as cutting down the cost of reaching an agreement, increasing the speed in the contracting process and allowing providers to deploy an automated provision of services based on the SLAs agreed with their customers. Furthermore, automating the establishment and management of SLAs enables an automated provision of services based on the agreed SLAs which allows a better rationalisation of the usage of resources in the organisation.

Following this line, a contracting process based on the creation and management of SLAs has been defined. This process is divided into four phases: information, negotiation, deployment and fulfilment. The goal of the information phase is to match service providers with potential consumers and vice versa. In the negotiation phase, an agreement on the execution of a service is created between service provider and consumer. The goal of the deployment phase is to set up a deployment plan to make it possible to follow all terms established in the settled agreement. Finally, the fulfilment phase involves the fulfilment of the obligations established in the agreement and in monitoring the whole process in order to ensure that both parties observe the agreement correctly.

However, some areas related to the automated management of this contract-
In our work, we focus on automating the service trading process. We define service trading process as the process of locating, selecting, negotiating, and creating SLAs. Therefore, the service trading process is the subprocess that covers the information and negotiation phases of the contracting process [27]. The characteristics of the service trading process depend on the particular scenario where it is developed. These scenarios are very diverse and can range from a traditional supply chain to dynamically selecting the best VoIP (Voice over IP) provider or contracting or renegotiating a contract with an ISP. As the scenarios are very different, the requirements for each of them are also diverse. Therefore, we argue that there is no one unique solution for service trading but we must choose the most appropriate option for each situation. Note that, despite its name, service trading is also adequate for integration proposals in which complex organisations are divided into sub-organisations and services are provided and consumed amongst them, chiefly if a SLA must agreed upon and adhered
An important part of service trading is the negotiation of SLAs. Providers’ preferences are usually opposed to consumers’ ones. Providers want to reach an agreement about the use of a service in the most profitable terms for them, which, generally, are not the most beneficial to the consumer. Therefore, both parties must negotiate in order to reach a mutually acceptable agreement. Here is where automated negotiation techniques play their main role that we analyse in the following section.

1.2 Motivation and goals

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. Different categories of automated negotiation can be established based on the characteristics of the parties involved in the negotiation, their goals and their relationship.

In this work, we focus on SLA negotiations. SLA negotiations are equivalent to the so-called service-oriented negotiations [38]. However, we prefer to avoid the term service-oriented because today it may lead to misunderstandings. 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. This scenario defines the characteristics of the negotiation that shall be carried out. It is a non-cooperative negotiation because the parties involved in it are selfish and, therefore, they just try to maximise their own benefit. It is also a negotiation with partial information because if a party has access to the private information of other parties, it may use it in its own benefit to get a more profitable agreement. Finally, a service-oriented automated negotiation has hard computational constraints, because we are interested in negotiating in a real-world scenario. From now on, when we use the term automated negotiation, we are referring exclusively to SLA automated negotiations.

In the last years, much work have been done on automated negotiation. This work has been mainly centred in three aspects of automated negotiation [20]: the negotiation object (what is being negotiated and how it is expressed), the negotiation protocol (which are the rules that govern the negotiation process and how the parties communicate each other), and the decision-making model (how to evaluate proposals, create counterproposals and build opponent models). In those works the focus is on the development of new decision-making algorithms or the construction of new protocols that presents certain desirable characteristics for automated negotiations. However, much less attention has been paid to the software artefacts that are necessary to carry out this automated negotiation.

We believe that to build an automated negotiation system that is able to successfully negotiate in real-world scenarios, it is necessary to start working in how to assemble all these different parts of an automated negotiation system.
Therefore, the main motivation of this report is to take a software engineering approach to automated negotiation by focusing on the elements that are necessary to develop an automated negotiation system and their relationships, instead of centring on how to implement those elements. To do so, we divide our work into three goals that we want to accomplish with this report:

1. **Analysing the problem of automated negotiation of service level agreements from a software engineering perspective.** Today, most works on automated negotiation are centred on the algorithms instead of the software artefacts that are necessary to carry it out. With this goal, we want to identify the elements that are required to build a system that develops an automated negotiation of SLAs.

2. **Categorising and classifying the most significant automated negotiation proposals.** This work allows us to compare and identify which proposals are useful for negotiating service level agreements.

3. **Developing a reference architecture for SLA automated negotiation.** Today, there is a lack of a reference architecture for automated negotiation of SLAs that identifies and describes the elements that must be implemented in those systems and gives guidelines on how to develop them. This reference architecture may be later used to define concrete automated negotiation architectures.

In addition, we must bear in mind that the agreement creation is part of a more general process called service trading. Therefore, the results obtained here must be aligned with a general service trading abstract architecture that covers the remaining parts of the service trading process and that it is also being developed as part of the same research project.

### 1.3 Structure of this report

This research report is organised as follows:

**Chapter 1: Introduction.** This very chapter. It contextualise the research work and outlines some concepts of automated negotiation.

**Chapter 2: Conceptual framework.** In this chapter, we present a novel conceptual framework of SLA automated negotiations. Unlike other similar works, we take a software engineering perspective and centre on the elements that are required to build an automated negotiation system. This conceptual framework settles the bases for a later analysis of the different negotiation proposals and defines the common vocabulary that is used in this report.

**Chapter 3: Reference architecture.** The goal of this chapter is to present a reference architecture for proposal-based automated negotiations. First,
we start motivating our research as a conclusion of the analysis developed in the previous chapter. Then, we describe the phases that compose the negotiation process. Finally, we detail the elements of our proposed reference architecture. This proposal addresses the presented negotiation process by defining a set of abstract elements and architectural guidelines that can be used to build more concrete architectures to deal with it.

Chapter 4: Analysis of architectures for automated negotiation. In this section, we obtain a set of properties that present automated negotiation systems. These properties are obtained as a consequence of the conceptual framework in Chapter 2. Then, we use these properties to evaluate our reference architecture and to compare it with other architectures for automated negotiation of SLAs.

Chapter 5: Conclusions. In this chapter, we present our conclusions and we outline future research directions.
Chapter 2

A conceptual framework for automated negotiations

Traditionally, an automated negotiation system has been characterised by three elements: protocol, negotiation object and decision-making model. In our conceptual framework we propose to extend this characterisation with two additional elements: information and preferences. Furthermore, we detail more precisely the decision-making and protocol elements. Additionally, we also provide a concrete description of the negotiation object given that we focus exclusively on SLA automated negotiations.

2.1 Negotiation object

In an SLA negotiation, the object that is being negotiated is an agreement between parties. An agreement defines a dynamically-established and dynamically-managed relationship between parties [2]. The goal of the agreement is to establish the guarantees that must be observed during the execution of a service. An agreement is composed at least by the following:

- A specification of the parties involved in them. In principle, the number of parties involved in an agreement is not constrained. However, the most common case is two-party agreements.
- A collection of terms that describes both functional descriptions and non-functional guarantees of the service. Additionally a term can also express other aspects of an agreement such as termination clauses. These terms conform the main part of the agreement and they regulate how the later execution of the service must be carried out in the context of the agreement. The terms used in an agreement must be fully specified and ambiguities must be avoided in order to prevent future problems. A term is composed of three parts:
The *counterparty* whom the term is applied to. Each term is to be applied to one of the parties involved in the agreement and the party is obligated to fulfil what it is specified in it. Obviously, the *counterparty* must be one of those that have been designated in the agreement as one of the parties that are involved in it.

- A set of *constraints* to specify functional or non-functional descriptions or guarantees of the service. It is expected that the content of these constraints will be very broad and domain-specific. Some examples of functional aspects of the service that can be expressed by using these constraints are the service interface by referencing a WSDL document, for instance, or the endpoint where the service is located. Regarding the non-functional guarantees, some examples are the response time will be less than 5 ms.

- A set of *compensations* that will be applied in case the party does not fulfil the constraints specified in the term. This element is optional and it is not supported by the majority of the negotiation strategies.

### 2.2 Preferences

The agreement preferences express the data that is used to assure that the user needs are correctly dealt among the negotiation process. The way the user defines their preferences about the desired agreements is very important to the negotiation process because different negotiation strategies and protocols requires the agreement preferences to be expressed in a certain way. There-
fore, the form of the agreement preferences shall affect the way a negotiation is performed.

There is another important aspect regarding the agreement preferences: the elicitation of these user preferences. Obtaining an explicit representation of the preferences from the user so that the resulting agreement after the negotiation meets his requirements is not easy [20]. Two different approaches have been followed to overcome these difficulties: using formalisms that are natural to the user such as rules [17] and eliciting the user preferences indirectly as in [6].

These preferences comprise the initial data provided by the user (such as the requirements/features of the service demanded/offered) as well as characteristics about the negotiation process itself such as the deadline to finish the negotiation or the eagerness of the user to obtain an agreement:

- A set of statements expressing the features, that is, the capabilities and/or characteristics presented by the user. In the case of a service provider, this set of features also comprises the service description in both functional and non-functional way.

- A set of statements expressing the requirements of the user in terms of the wished features of the counter-parties. These requirements can be imposed on either the service or the party itself.

- An Assessment Mechanism to evaluate and compare potential agreement proposals. The most common way of evaluating proposals is through the definition of utility functions for the features expressed in the terms of a given proposal. Depending on the value of the term, it has a certain utility to the party [11]. The total utility of the agreement is usually calculated as the weighted sum of the utilities of each term of the agreement.
A set of statements expressing the desired characteristics of the negotiation process that will take place to reach an agreement.

An statement groups different types of expressions over a certain party or the negotiation process itself. Statements can be classified depending on the domain in which they are applied:

- **Service statements.** These statements are applied to the service offered (or demanded) itself. In this context, they can refer to either functional or non-functional characteristics of the service such as the service interface or the service cost.

- **Party statements.** In this case, an expression about the party is stated. This statements can express either features or requirement over a given party. Examples of this can be: *Party Z is located in Iran* or *Party X has a low reputation on service Y.*

- **Negotiation statements.** They specify features about the negotiation process itself, such as the aforementioned eagerness or negotiation deadline.

Each statement is linked to a set of languages that give semantics to the vocabulary used within the statement. Ontologies can be seen as an example of languages describing the relationship amongst concepts of a given semantic domain.

Usually, the statements are expressed using two different formalisms: rules and constraints. However, other formalisms could be used.

- **Rule.** A rule describes a set of actions or effects triggered by a condition. In this context a rule is used to express conditional behavioural preferences of a certain party. An example of rule could be *If the consumer buy more than 100 executions of the service, price is reduced 10 per cent.*

- **Constraint.** Constraints express a restriction of an attribute over a given domain. This structure is used to state assertions expressing requirements, features, terms or trading protocols. Examples of constraints can be *Location of partner must not be Iraq nor Iran* or *Speed of service varies from 1 to 10 sec.*

### 2.3 Information

An SLA automated negotiation is a non-cooperative negotiation with partial information as we stated before. On the other hand, the more information we have about other parties, the better our performance is in the negotiation process [47]. Furthermore, it has been shown that taking the conditions of the market into account does improve the results of negotiations with several simultaneous opponents [40]. Therefore, it is essential for an automated negotiation system to manage information about the counterparties and the market.
The kind of information managed by an automated negotiation system can be classified attending to three different criteria: the subject the information is about, the subjectivity of the information and the mechanisms used to obtain it.

The information used by an automated negotiation system may be about three different subjects:

- Information about the counterparties, including the characteristics of the service demanded/offered, the negotiation process followed by the counterparty and the counterparty itself (e.g. reputation or geographical location).

- Information about the market, including information such as the market reservation price, or the probability of appearing outside options during the negotiation [26].

- Information about the service domain such as knowledge about the vocabulary used in the terms of the agreement.

Regarding the subjectivity of the information, it may be either objective or subjective:

- On the one hand, an automated negotiation system may handle objective information about the service and the negotiation process followed by the potential counterparty. The amount and type of information collected from each candidate may be different, but it typically includes the public
features about the service demanded/supplied. Additionally, although uncommon, an automated negotiation system may also handle objective information about the counterparties themselves (e.g. their geographical location) and the market (e.g. the number of known buyers interested in the same service).

- On the other hand, an automated negotiation system also manages subjective information such as the reputation of a counterparty or the market price of a certain service. Unlike the objective information, this one is usually obtained either by querying external sources (e.g., reputation servers about companies) or by analysing the results of previous interactions with potential counterparties. However, it is also possible to gather information related to one potential counterparty by querying it directly.

Attending the procedures employed to obtain information, it can be classified into three categories:

- Directly polling the potential counterparty. In this case, the system must implement a compatible specification of a format to express functional and non-functional features of services and a procedure to query and to inspect services.

- Querying a third party entity to obtain information related to a specific counterparty. For instance, to obtain information about its reputation or its geographical location. In this case, a protocol to carry out this query as well as a shared taxonomy of terms must be implemented in the organisation.

- Analysing previous interactions with a potential counterparty. The results of the analysis may be stored in order to be used later on, while making decision about proposals related to the potential counterparty. This approach has been followed in works such as [47] and [8] where Bayesian learning and kernel density estimation techniques have been respectively used to build a model of the preferences of other participants in the negotiation enabling us to build better counterproposals.

We envision that the first procedure shall be commonly used in gathering objective information about the counterparties, while the second and third procedures shall be more common in obtaining subjective information about the market and the potential counterparty itself.

2.4 Protocol

The negotiation protocol establishes the rules that govern the negotiation and the way the communication between the different parties involved in the negotiation is carried out as well as the information exchanged between the parties. We distinguish three different, although strongly related, aspects in a negotiation protocol: rules, performatives and information exchanged.
2.4.1 Performatives

In a negotiation protocol, information is not exchanged as is, but there are semantics linked to each message describing its intention. For instance, we can exchange always the same type of information (e.g. an agreement proposal) but we may mean many different things such as *I am proposing this agreement*, *I am accepting this agreement*, or *I am committing to this agreement*. That is why we need what it is called *performatives*. A performative is the expression of the intention of the sender of a message about it. The term performative is borrowed from the FIPA terminology and the speech act theory. Thus, the sender’s intentions that can be expressed as a performative such as *propose*, *accept*, or *commit*.

The set of performatives used in a negotiation protocol may differ significantly. However, there are a minimum subset of performatives that are common to the majority of negotiation protocols. Namely, *accept* (accept a proposal), *reject proposal* (reject the proposal but the negotiation goes on), *reject negotiation* (cancel the whole negotiation process), and *commit*\(^1\) (commit to a given proposal). Depending on the specific protocol that is being used, other performatives may be necessary. For instance, in auctions, it is common to use the

\(^{1}\)In the literature, this performative is usually called *propose* meaning making a binding proposal. However, we prefer to leave the term *propose* to non-binding proposals and to use *commit* for binding proposals.
inform performative to notify events occurred during the negotiation such as that a new bid has been done. In protocols that differentiates between binding and non-binding proposals, the propose (make a non-binding proposal) performative may be introduced. Other protocols use a vote system to decide which is the preferred offer [23]. In such protocols, a vote performative may be used. Finally, in negotiation protocols that use argumentation [33], other performatives to introduce arguments supporting our proposal are used such as argue and challenge [1].

2.4.2 Rules

In a negotiation protocol, it is usual that there are some restrictions regarding a variety of aspects of the negotiation such as how the proposals must be built, when a participant may post a proposal, which performative may be used in each moment, when a participant may join to the negotiation, or when the negotiation may finish. For instance, in an English auction, only higher proposals than the current one are accepted, and in some bargaining protocols the terms of the proposal are negotiated one-by-one following an established agenda [12], or restricted to be narrower in the counterproposal than in the proposal [21].

Usually, these rules are implicit in the description of a negotiation protocol. However, in [5] it is argued that to build automated negotiation systems that supports a variety of negotiation protocols, it is necessary to explicitly define the rules of the negotiation protocol. In that work, a taxonomy of rules for negotiation is presented: rules for admission of participants, rules for proposal validity, rules for protocol enforcement (determines when a proposal may be posted, which proposals may be posted and when a proposal may be withdrawn), rules for updating status and informing participants, rules for agreement formation, and rules for lifecycle of negotiation.

Although it is a thorough taxonomy and the majority of the aspects of automated negotiation are covered, we believe that it should be extended with one additional set of rules. Namely, rules for decommitting from previously created agreements. It has been shown that allowing the decommitment from agreements may significantly improve the outcome of the negotiation in certain conditions [36]. Furthermore, decommitment is a very useful mechanism to enable bilateral negotiations with n simultaneous opponents [29]. Therefore, we argue that there must be rules that explicitly specify whether a decommitment may take place, and, if so, when it may occur and which are the penalties to be paid as a compensation.

2.4.3 Information exchanged

The third relevant aspect in a negotiation protocol is the type of information exchanged amongst the participants in the negotiation. A variety of approaches has been presented in the literature. Those approaches may be classified into three broad groups:
• The information exchanged explicitly states the parts of the agreement that are disliked by the party as well as the proposed changes.

• The information exchanged consists only of proposals. In other words, the negotiation protocol is proposal-based. In this case, the information about the disliked parts of the agreement is implicit in the counterproposal and it is up to the parties to infer it. The advantage of this approach is that it unveils less information to the other parties. The disadvantage is that the lack of explicit information implies a blind search of a mutually acceptable agreement that may lead to longer negotiations and even to not to find any agreement at all.

• The information exchanged includes proposals, as in proposal-based protocols, and statements that are used to persuade or convince the opponent to accept our proposal [34]. This approach is called argumentation and it is a promising field that may eventually overcome the drawbacks of the proposal-based negotiation [33]. However, the negotiators that supports argumentation tend to be very complex and no argumentation approach has been applied to a real scenario today.

Nevertheless, some proposals use a mixed approach. For instance, the information exchanged may consist of proposals but they may be accompanied by a collection of variation points. They can be used as guidelines to facilitate the process of finding an agreement. For instance, these variation points can be used to relax constraints specified in those terms or to express possible trade-offs amongst terms. For instance, I accept a response time higher than 20 ms but only if the price is lower than 50 cents per execution. A variation point provides guidelines to facilitate the process of finding an agreement through the exchange of proposals. Each variation point is applied to a collection of terms specified in the proposal. They can be used, for instance, to relax constraints specified in those terms, to express possible trade-offs amongst terms (e.g. I accept a response time higher than 20 ms but only if the price is lower than 50 cents per execution), to indicate whether a term is negotiable, or to provide the other party with partial information about our utility function.

Another way of classifying the information exchanged is depending on the formalism that is used to express them. A variety of ways of expressing the information exchanged have been proposed. For instance, in WS-Agreement [2] the proposals use a vocabulary defined in XML and the terms of the agreement are defined as restrictions over the elements of the vocabulary. In QRL [35] and [25], the information exchanged is understood as mathematical constraints and, hence, the negotiation is viewed as a constraint satisfaction problem. In [41], an ontology based approach using OWL is followed to give semantics to the terms of the agreements and, thus, improving the interoperability amongst participants in the negotiation. In this line, there are some efforts\(^2\) to create a semantic-enabled form of WS-Agreement. Finally, other logic-based languages have been

\(^2\)See http://lsdis.cs.uga.edu/projects/meteor-s/swaps/
studied for negotiation [45]. This is particularly common in argumentation-based approaches because they require an explicit knowledge based that may be modified by statements coming from the other participants in the negotiation.

### 2.5 Decision-making

The decision-making model determines the way a certain party behaves while involved in a negotiation process. Three elements form part of the decision-making model of an automated negotiation system: the decision of what is considered an acceptable agreement and whether to commit to it, the construction of responses to this information, and the decision to decommit from a previously established agreement if possible.

#### 2.5.1 Binding decision

The most important decision that has to be made in an automated negotiation system is whether we commit to an agreement. This includes determining when a binding proposal must be submitted and whether a binding proposal that has been received should be accepted. In addition, it is necessary to establish when these decisions are going to be made. For example, one option is to make the decision as the proposals are received; another possibility is to make the decision at some specific points in time that has been previously set. Therefore, an automated negotiation system has to decide not only whether a binding proposal must be accepted or submitted but also when these decisions shall be made.

The binding decision depends on several factors that may vary depending on whether it is a service consumer who is making the decision or it is a service provider. Nevertheless, we can divide these factors into three broad groups:

- First, preferences of the user. These preferences may be related to the
contents of the agreement (e.g. constraints on the values of the terms of the agreement or an utility function indicating the importance of these terms to the user), the party we are negotiating the agreement (e.g. we may not want to make an agreement with a company that competes with us with another product or with a company located in a country that we are not allowed to sell services to), and the negotiation process (e.g. the deadline and the eagerness to reach an agreement).

- Second, the information the system have about the status of the market and other possible concurrent negotiations. For instance, we may be more reluctant to accept a proposal if we know it is very likely that in a short amount of time we will receive a better proposal than the current one [26], or if the number of trading partners is high and the spread between us and the participants of other concurrent negotiations is small [39].

- Third, external factors that may prevent a party to commit to an agreement. For instance, the provider’s capability to accept new agreements or the existence of dependencies amongst the agreements a service consumer wants to reach.

The complexity of the binding decision may vary significantly depending on the negotiation protocol and the number of concurrent negotiations that are being carried out. In many negotiation strategies, only the preferences of the user are considered in the binding decision. Specifically, a proposal is accepted if its utility, which is calculated by applying the user preferences, is higher than a certain threshold that may change during the negotiation [11]. This is because in these models no concurrent negotiations are considered and all proposals are binding ones. However, when multiple agreement negotiations are being carried out simultaneously, the binding decision becomes more complex because it must coordinate the sending or acceptance of binding proposals to the other parties. In so doing, it is possible to avoid undesirable behaviour such as a service consumer committing simultaneously to two agreements to cover its needs when only one is necessary.

Moreover, we envision that most of the proposals exchanged between the parties will be non-binding proposals, while binding proposals will be only sent when it is really appealing to us and likely to be accepted by the other party. In so doing, negotiations can evolve more independently and quickly than sending binding proposals continuously because the coordination amongst them only occurs when a binding proposal is decided to be sent.

Together with the decision of making or accepting a binding proposal, it has to be decided when this decision take place. Usually, the decision is made as the proposals are received like in [30]. However, other approaches may be followed such as making the decision in some certain points in time previously defined. In this way, the window of proposals taken into account in a decision point is broader and, hence, it is less likely to reach a suboptimal agreement. These decision points may be static and established before the negotiation process starts,
or they may change dynamically depending on the status of the negotiations or the preferences of the user.

2.5.2 Response generation

Other important task in an automated negotiator is to decide which response must be sent to the other participants in the negotiation. On the one hand, this response is subordinated to the binding decision. That is, if it is decided not to commit to one proposal, the response generated must not imply a commitment to the proposal. Alternatively, if it is decided to commit to a proposal, the response must be a commitment to it. On the other hand, the response generated must obey the rules imposed by the negotiation protocol.

The process followed to generate the responses varies significantly depending on the negotiation protocol. However, in general, this complexity depends on the performatives of the negotiation protocol and the expressivity of the information exchanged during the negotiation:

- In auctions, the unique possible response is a bid together with the bidding price. Therefore, in this case the problem is centred on deciding in which auction must be placed the bid [4].

- In bilateral proposal-based protocols, a counterproposal must be generated. A wide variety of techniques have been developed to generate them. The most significant are: those that use time-dependant functions, resource-dependant functions, etcetera to obtain the counterproposal by modifying the values of the terms of the offer [11]; those that try to make the counterproposal more appealing to the opponent by sending the counteroffer with the highest similarity to the received offer [10]; those that use constraint resolution techniques [25] or that are based on fuzzy constraints [24], and those that interpret the negotiation as if it were a game and use techniques similar to those used in chess games [22]. Genetic algorithms have also been used to calculate offline which is the best strategy to use depending on the conditions of the negotiation in a certain instant [13].

- In negotiation protocols that supports argumentation, the response generation is a much more complex process because there are many performatives that can be used (reward, information, threat...) and the content of the message may be very diverse (there are potentially no limitations on it). In this case, the response generation includes two problems [33]. First, the different arguments are generated and then, the best argument from the point of view of the speaker is selected. However, it must be noted that both the generation of arguments and the selection of the best one may occur at the same time. Nevertheless, whether or not this is possible depends on the specific argumentation framework that is being used.

Note that the binding decision may take place in conjunction with the response generation. It is not necessary that an automated negotiator consider
them as two independent processes. However, in some cases, it is convenient to consider them separately. For instance, in simultaneous bilateral proposal-based protocols with explicit commit, a decoupled approach may bring more benefits because it allows to explore the space of possible agreements without the overhead of deciding whether to commit to a proposal. Additionally, a decoupled approach is also convenient when the binding decision has a high computation cost. This is the case if the binding decision heavily depends on external factors that change dynamically such as the available resources in the service provider to provision the service, and a capacity planner is required to analyse whether or not the service provider is capable of accepting the new agreement.

2.5.3 Decommit decision

Recently, it has been analysed the benefits of allowing the decommitment of established agreements by paying a certain decommitment fee that has been agreed previously. The first work in this line was presented by Sandholm and Lesser [36]. In that article, a formal analysis of the advantages of decommitting from established agreements and it is shown how it may increase the payoff obtained in several negotiations. However, they just covered a two person game with fixed penalties for decommitting.

Later, Nguyen and Jennings [31] have used decommitment to build a system that allows multiple concurrent negotiations and support varying decommitment fees depending on the stage of the process at which the commitment is broken. In this work, the decommitment is handled by a commitment manager. The commit reasoner decides when it is worth it to decommit from an already established agreement in order to commit to another one by taking into account the utility of both agreements, the decommitment fee and other parameters that characterise the behaviour of the negotiator.

Therefore, it has been proved that decommitment is a valuable resource when dealing with multiple simultaneous negotiations and, hence, it is another decision element that must be included in an automated negotiation system. The decommit decision is highly related to the binding decision and depends on the same elements: user preferences, information about the market and other opponents, and external factors such as the availability of resources. That is the reason why in many cases both the binding and decommit decisions are made by the same element [30].
Chapter 3

A reference architecture for automated negotiation of SLAs

3.1 Introduction and motivation

The analysis of the most significant automated negotiation proposals carried out in the previous chapter leads to two conclusions that motivate the necessity of developing a software architecture for automated negotiation of SLAs:

- The first conclusion is that there exist a variety of negotiation protocol. Therefore, for our system to work in an open environment such as the Internet, it is necessary to give support to a significant amount of them. Additionally, the system must be extensible in order to add new protocols in the future.

- The second conclusion is that the goodness of the different negotiation strategies depends on the negotiation context. Here, we understand negotiation context as aspects such as the negotiation object, temporal constraints imposed by the parties or the number of participants in the negotiation process. For instance, if the negotiation deadline is near, the negotiator must concede more than in the beginning of the negotiation. Therefore, the architecture must support a variety of negotiation techniques and must be able to select the most appropriate in each moment.

Therefore, it is necessary to develop advanced software systems that are able to carry out automated negotiation of SLAs in real-world scenarios. However, although much work has been done in automated negotiation, most of these efforts have been centred in the development of negotiation protocols and strategies. That is, the algorithmic aspect of automated negotiation.
In this research work, we take a software engineering approach to automated negotiation by focusing on the elements that are necessary to develop an automated negotiation system and their relationships, instead of centring on how to implement those elements. Namely, we fill this gap by reporting on a reference architecture for automated negotiation that identifies and describes the elements that must be implemented in those systems and gives guidelines on how to develop them. This reference architecture may be later used to define concrete automated negotiation architectures. In this context, we understand a reference architecture as a precise specification that defines a set of elements for automated negotiation of service level agreements. A reference architecture is not intended to be directly implemented, but it has to be further refined to a concrete architecture before its implementation.

In Chapter 2, we give many different implementation alternatives for developing all five basic concepts of every automated negotiation system. These alternatives are specially diverse for protocols and decision-making. However, if our goal is to build an useful reference architecture for automated negotiation systems, we must restrict this number of alternatives. The reason behind this is that, as the alternatives are very heterogeneous, an architecture that covers them all is too general. The problem here is that if the reference architecture is too general, there is much work to be done by the developers and, hence, its usefulness decrease.

To avoid this problem, we have decided to focus on the particular case of proposal-based negotiations of SLAs. The rationale behind this decision is:

- Proposal-based negotiations are a very well studied field. During the last ten years, much work have been done in this area and significative advances have been done in the algorithms and protocols to carry out the automated negotiation. Therefore, it is a field mature enough to start developing software system with the intent to use them in a real scenario.

- It is aligned with recent efforts of the industry in this area. For instance, new emergent standards form the industry such as WS-Agreement [2] and WS-AgreementNegotiation [3] are based on an exchange of proposals to carry out the agreement creation process.

- Other approaches such as argumentation are too hard to put in practise. Argumentation-based negotiation overcomes many limitations of proposal-based negotiations but the price to pay is a extremely complex system. Moreover, argumentation-based negotiation is a more recent field of study and there are less techniques developed for it.

The reference architecture has been developed taking the conceptual framework described in Chapter 2 as a starting point. The input to the architecture is twofold:

- The agreement preferences, which is information (requirements or features) gathered during the preparation phase. The actor that provides the agreement preferences to the architecture is called the user.
• A set of counterparties with whom our system is going to negotiate. To obtain these counterparties, a previous discovery and preselection may have been carried out. Additionally, information about the preferences of the counterparties may be provided as well.

The output of the reference architecture is a set of SLAs established with a variety of parties. This SLA can be used to drive the further deployment and fulfilment phases.

It is important to remark that this reference architecture is independent of the nature of the stakeholder, i.e. whether it acts as the service consumer or the service provider. Therefore, our architecture is symmetric in the sense that both service provider and service consumer can simultaneously be negotiating proposals with other parties that are interested to create an agreement with us. Another consequence of this symmetry lies in that the data structures used to express the agreement preferences are the same because they just express requirements on the other party and features of our party despite being a service consumer or provider.

3.1.1 Modeling elements

Before detailing more accurately the different parts of the architecture, it is convenient to define and clarify the modeling elements that are used all along the document as well as its graphical representation in Figure 3.1.

Organisations  In order to deal with the complexity of the service trading problem, we use the organisational metaphor proposed in GAIA [46], where organisations are outlined developing a general architectural characterisation of the problem. Each organisation must have a concrete and well defined goal and the number of communication lines amongst organisations must be low. Besides, each organisation is composed of several roles that work together to achieve the goal of the organisation.

In Figure 3.1 the organisations are depicted as big blue boxes that bring together all roles that comprise the organisation. The name of the organisation is represented in the top-right part of the box.

In so doing, our reference architecture is composed of three organisations: information, agreement making and binding, that interact to accomplish the negotiation process.

Roles  We call role to a precisely defined task that must be carried out in the context of an organisation by one or more software artefacts1 in the running system. A role must belong to one organisation and they are described in terms of their functionalities, activities, and responsibilities, as well as in terms of their interaction protocols and patterns.

---

1In MAS, these software artefacts are usually agents
Figure 3.1: Reference architecture
In Figure 3.1 the roles are depicted as small black boxes. Their name is represented inside the box. The arrows that connect two different roles indicates the existence of a communication between them. The arrow point represents an input interface in the role, while the beginning of the arrow represents an output interface in that role.

Environment The reference architecture is immersed in an environment that determines the entities and resources that the roles can exploit, control or consume when it is working towards the achievement of the organisational goal [46]. The main characteristic of a resource in the environment lies in that it is available to all the roles. For a resource in the environment, it is necessary to specify its name and description together with the type of actions that the roles can perform on it.

In the abstract architecture, the resources of the environment are the agreement preferences repository, the agreement repository, and the world model. They are all listed inside a green box located in the top-right part of the Figure 3.1.

Collaboration A collaboration is the interaction of two or more roles in order to achieve a concrete subgoal of the organisation. Therefore, a collaboration belongs to one unique organisation and develops a subgoal of the goal of the organisation. A collaboration is characterised by its goal, the roles that participate in it, the sequence the roles participate in the collaboration, and the dependencies of the collaboration with others. In this context, a dependency between collaborations means that the depended collaboration must have been carried out at least once for the dependent collaboration to start.

The main utility of the collaborations in our model is to abstract out the complex interactions between the roles of the architecture.

3.1.2 Phases in the negotiation process

The negotiation process can be divided into four phases (see Figure 3.2): information, pre-negotiation, negotiation and commitment. The first two phases are optional but they should be present in an automated negotiation system that works in open environments such as the Internet. The process starts when a new party is given to the system in order to negotiate to it.

1. Information. The goal of this phase is to obtain information about the party. This information may be obtained by directly querying the party and asking it for its preferences or by querying third parties about it.

2. Pre-negotiation. Before starting a negotiation, the negotiation protocol must be agreed between the participants in it. Note that the negotiation protocol have a great influence on the decision-making procedure that shall be used. For instance, it is different to negotiate in a protocol with non-binding proposal than in a protocol where all proposals are binding.
Additionally, in some cases, the agenda of the negotiation has to be decided [12]. The agenda of the negotiation is the order in which the terms are being negotiated. This agenda may be endogenous or exogenous. An agenda is exogenous if it is externally imposed. Conversely, it is endogenous if it is determined as part of the bargaining equilibrium.

3. **Negotiation.** This is the phase where the actual negotiation take place. It involves exchanging proposals in order to reach an agreement. This phase includes three steps. First, the system receives the proposal and updates its world model based on it. Then, the system selects the strategy\(^2\) that will be used to generate the counterproposal. Finally, the counterproposal is built and send back to the counterparty.

4. **Commitment.** When the received proposal is decided to be acceptable, the commitment phase take place. In this phase both parties commit to an agreement and it concludes the negotiation process. If the commitment failed (e.g. the other party refuses our proposal), the negotiation phase may continue until either a new commitment phase take place or some party decides to finish the negotiation process.

The remainder of this chapter is organised as follows. First, in Section 3.2, we detail the data structures that have been identified in the architecture. Then, from Section 3.3 to 3.5, we deeply describe all organisations that are part of the abstract architecture as well as the collaborations and roles that compose them.

### 3.2 Data structures

The data structures used by the reference architecture (see Figure 3.3) are heavily based on the concepts of negotiation object and preferences defined in Section 2.1 and Section 2.2, respectively. The new data structure introduced by the reference architecture is the *proposal*.

\(^2\)In this context, a strategy is the algorithm that is used to build a counterproposal.
An agreement is composed of the parties involved in them and a set of terms that define its content. The terms comprise the counterparty to whom the term is applied, constraints to specify functional or non-functional descriptions or guarantees of the service, and, optionally, the compensations that must be paid if the term is not fulfilled.

The agreement preferences are composed of a set of features and requirements about the service and an assessment mechanism that is used to evaluate the received proposals.

Proposal A proposal is an offer for an agreement made by one party to its counterparty. We can understand a proposal as a type of agreement where some terms are left open in order to be refined in later interactions amongst the parties. In our reference architecture, a proposal is composed of:

- The parties the proposal is about.\(^3\)
- A set of terms, similar to those specified in the agreement, to give a functional description, non-functional guarantees and other agreement-related

\(^3\)This is equivalent to the parties in the agreement
aspects such as termination clauses. However, unlike in an agreement, the terms of a proposal can be left opened, that is, underspecified, in order to be refined later on.

- A collection of variation points referring to some terms specified in the proposal. These variation terms can be used as guidelines to facilitate the process of finding an agreement through the exchange of proposals.

- A proposal performative to express the intention of the sender about that proposal. That is to say, if the proposal is a binding one or if it represent the acceptance or rejection of a previously made proposal.

Variation Point A variation point is an expression to indicate ranges of acceptable variations of the terms specified in a proposal. These variation points can be used to relax constraints specified in those terms or to express possible trade-offs amongst terms. For instance, I accept a response time higher than 20 ms but only if the price is lower than 50 cents per execution. A variation point provides guidelines to facilitate the process of finding an agreement through the exchange of proposals. Each variation point is applied to a collection of terms specified in the proposal. They can be used, for instance, to relax constraints specified in those terms, to express possible trade-offs amongst terms (e.g. I accept a response time higher than 20 ms but only if the price is lower than 50 cents per execution), to indicate whether a term is negotiable, or to provide the other party with partial information about our utility function.

Proposal Performative A proposal performative is the expression of the intention of the sender of a proposal about it. The term performative is borrowed from the FIPA terminology and the speech act theory. In this context, some examples of a sender's intentions that can be expressed as a proposal performative are make a binding proposal, make a non-binding proposal, reject a previous proposal, or accept a proposal.

3.3 Information

Goal The goal of this organisation is to manage information about the counterparties and the market. These counterparties may have been found following a discovery process or may have been got known due to the reception of a proposal from them.

The information managed by this organisation is twofold:

- On the one hand, it handles objective information about the service and the trading process followed by the potential counterparty. The amount and type of information collected from each candidate may be different; however, at a conceptual level the information should include, at least, the public features about the service demanded/supplied. Typically, this kind of information is collected by directly querying the potential counterparty.
On the other hand, it manages information related to the potential counterparty itself. Unlike the information about the service and the trading process, this one is usually obtained either by querying external sources (e.g., reputation servers about companies) or by analysing the results of previous interactions with the potential counterparties. However, it is also possible to gather information related to the potential counterparty by querying it directly. This information is expected to be more subjective in nature than the information about the service and the trading process. Nevertheless, it may also include elements of objective information such as the geographical location of the potential counterparty.

Requirements

There are three procedures to obtain the necessary information to fulfil the goals of the organisation:

- Directly polling the potential counterparty. In this case, the organisation must implement a compatible specification of a format to express functional and non-functional features of services and a procedure to query and to inspect services.

- Querying a third party entity to obtain information related to a specific counterparty. For instance, to obtain information about its reputation or its geographical location. In this case, a protocol to carry out this query as well as a shared taxonomy of terms must be implemented in the organisation.

- Analysing the information supplied by other roles of the architecture in regard to interactions carried out with a potential counterparty. The organisation must implement mechanisms to carry out this analysis. Moreover, the results of the analysis may be stored in order to be used later on, while making decision about proposals related to the potential counterparty.

We envision that the first procedure shall be commonly used in gathering service and trading process information, while the second and third procedure shall be more common in obtaining information about the potential counterparty itself.

We must note that while the information about the service and the trading process followed by the potential counterparty is strictly necessary to carry out the remaining service trading process, the information related to the potential counterparty itself is only necessary to improve the decision-making process. Therefore, the second and third procedures are just best-effort ones, and not mandatory to implement a simple service trading process.

3.3.1 Roles

The organisation is composed by four roles. Two of them (Inquirer and Informant) manage the service and trading process information, while the others (World Modeller and Third Party Informant) handle the information about the potential counterparties themselves.
Figure 3.4: Information organisation - Roles and data dependencies
**Inquirer**

**Goal** The *Inquirer* is the role in charge of obtaining service and trading process information by polling the *Informants* of the counterparties.

**Data input** The role receives a collection of counterparties to whom query. Additionally, the role reads the *agreement preferences* environmental resource in order to obtain guidelines about which specific information to ask the counterparty for.

**Data output** The output is *Information* related to each of the counterparties received as input.

**Behaviour** After receiving the collection of potential counterparties, the *Inquirer* contacts with the *Informant* role of each counterparty and queries it for the information it considers relevant. To decide which information is relevant, the *Inquirer* may use the *agreement preferences*. In the process of polling the *informants*, the role can select different strategies of querying, depending on the interaction standard and the type of information needed to match agreement preferences.

**Informant**

**Goal** The *Informant* role is responsible for publishing all public agreement preferences that can be useful to other parties in order to evaluate the possibilities to make an agreement with it.

**Data input** The input is the query made by an external element and the *agreement preferences* environmental resource.

**Data output** The output is *information* responding the query requested, typically extracted from the *agreement preferences*.

**Behaviour** The *informant* extracts a set of information from the *agreement preferences* to respond the queries that have been requested. Note that not all the information of the *agreement preferences* is intended to be public, therefore a process of determining which information is going to be sent is required. This process can be very simple (e.g. it is annotated in the *agreement preferences* whether some statement is public) or more complex (e.g. the decision of which information is public is determined by previous experiences). Then, after receiving a request, the *informant* responds it with the information it thinks convenient, for instance, depending on the nature of the requester, the *informant* can send back more or less information.
World Modeler

Goal The goal of the world modeler is to build up a model of the potential counterparties to whom agreements can be made (i.e. model of the world). This model can be based on information supplied by external third parties or by other roles of the abstract architecture. To build the model both objective and subjective information can be used.

Data input The input to the roles is twofold:

- Queries made to external third parties about the potential counterparties involved.
- Information supplied other roles of the abstract architecture about these potential counterparties. This information can range from a simple reference to the counterparty to the sequence of proposals exchanged during a bilateral bargaining.

Data output The result is a modification of the world model environmental resource, where all changes coming from the new information provided to the world modeler have been applied.

Behaviour The world modeler shows two different and mostly independent behaviours.

- On the one hand, it queries external third parties informants to obtain information such as the reputation or the geographical location of a specific potential counterparty. Then, it adds this information to its world model, possibly after weighing up the information depending on our trust in the third party informant.
- On the other hand, it receives information supplied by other roles of the abstract architecture. Then, this information is analysed and the results are stored in the world model environmental resource.

Third Party Informant

Goal The Third Party Informant provides specific information regarding to potential counterparties such as the reputation.

Data input The input is the query asking for information related to a specific counterparty.

Data output The output is information answering the query requested.

Behaviour Third party informants are envisioned to be specialised parties that provides relevant information of a certain company such as its reputation or the geographic location.
3.3.2 Collaborations

The organisation can be decomposed in two different collaborations, each one in charge of one of the subgoals described above: the ServiceInformation collaboration, whose goal is to query the potential counterparty to gather information about them; and the WorldInformation collaboration, which is in charge of supplying to the world modeler information coming from other roles of the abstract architecture or third party informants.

Next, we describe the ServiceInformation and WorldInformation collaboration as well as other relevant subcollaborations.

ServiceInformation

Goal The goal of the collaboration is to obtain a list of potential counterparties and to query them in order to gather information about them. This information can be related to the service, the trading process or the potential counterparty itself.

Roles Inquirer, Informant, and Agreement Maker.

Dependencies The collaboration can be decomposed into two more collaborations, namely, the CounterpartyInformationRequest (that sends a list of potential counterparties that have submitted us a proposal in order to query them and get more information), and the InformationExchange (that queries the counterparty to obtain information about it).

Therefore, there exists a dependency between the InformationExchange and the other two collaborations because it is necessary to know which are the counterparties, that we want to obtain information from, before interacting with them through the InformationExchange collaboration.
InformationExchange

Goal  The aim of the collaboration is to query a potential counterparty to obtain information about it.

Roles  Inquirer and Informant

Data exchanged  The data used in this collaboration is Information, that is to say, a collection of Statements about the service, the trading process supported by the counterparty, and the potential counterparty itself.

WorldInformation

Goal  The goal of the WorldInformation collaboration is to provide the world modeller with information coming from other roles of the abstract architecture or third party informants.

Roles  World modeller, Third party informant, Inquirer, and Agreement maker (from Agreement Making).

Dependencies  The collaboration is composed of three collaborations. These collaborations represent the interaction between the world modeller and each one of the other three roles that form part of the WorldInformation collaboration. The goals of these subcollaborations is just to supply the results of the roles’ interactions with potential counterparties to the world modeller. Therefore, strictly speaking, there are not dependencies amongst all these subcollaborations. Instead, they can happen whenever a specific role considers it is appropriate to send the results to the world modeller. However, there exist a light dependence between the collaboration where world modeller and third party informant interact (ExternalInformationSupply) and the others, because we must know first which are the potential counterparties we are interested in, before querying a third party informant about them.

ExternalInformationSupply

Goal  The goal of the collaboration is to query information about potential counterparties to a third party informant.

Roles  World modeller and Third party informant.

Data exchanged  The data used in this collaboration is expected to be dependent on the third party informant that is being queried, and, therefore, domain-specific.
**InformationSupply**

**Goal** The goal of these collaborations is to provide the *world modeler* with the results of the interaction of roles of the abstract architecture with potential counterparties.

**Roles**
- *InquirerInformationSupply*: World modeler and Inquirer
- *AgreementMakerInformationSupply*: World modeler and Agreement maker

**Data exchanged** The data used in these collaborations depend on the type of interaction the role has with the potential counterparties. Therefore, in the *AgreementMakerInformationSupply* the data used is a *proposal*, and in the *InquirerInformationSupply*, the *information* obtained directly from the potential counterparty.

### 3.4 Agreement Making

**Goal** The goal of the agreement making organisation is to provide a mechanism to create agreements, possibly through an automated negotiation process, that are acceptable to all the parties involved in them. Therefore, the result of this organisation is an agreement that specifies the terms under which the service shall be executed. This may include both functional and non-functional terms.

Although the agreement is created and signed in this organisation, the actual decision of sending a binding proposal or accepting a proposed agreement is not made in it but delegated to the binding organisation. Therefore, the binding organisation must be asked for permission before creating an agreement.

**Requirements** Consequently, the requirements of the agreement making organisation are:

- It must support an agreement format understood by both parties and that allows them to identify the terms of the agreement.
- It must implement at least one protocol to create agreements and, optionally, to negotiate them.
- It must provide decision making mechanisms to evaluate the proposals received and to generate their own bids or counterproposals if necessary.
- It must offer a way to create reliable and non-repudiable agreements. In this context we say an agreement is reliable if both parties are signing and accepting the same previously agreed document.
Estos mecanismos lo mismo irían bien en el Agreement Maker. The decision-making mechanisms determine the way the parties involved in the negotiation process behave. There are four procedures that are usually used during the decision-making in a negotiation process:

- a proposal evaluation, usually carried out through the definition of utility functions to each term of the agreement
- a model of the world and of our potential counterparties in order to improve our negotiation capabilities [47]
- a decision on which response shall be sent to the counterparty
- a construction of a counterproposal if necessary [11]

The two first procedures are common to other parts of the abstract architecture and are addressed in other organisations of the architecture. Specifically, the proposal evaluation is carried out by using the assessment mechanism defined in the agreement preferences, and the model of the world and the opponents is built in the information organisation.

Nevertheless, the other two procedures are specific to this organisation and, therefore, they must be implemented by some role of the organisation. However, as the assessment mechanism and the world model are generic and common to all parts of the abstract architecture, we envision that more specific and negotiation-oriented procedures can be implemented in this organisation to refine and complement the common ones.

3.4.1 Roles

The organisation is composed by three roles. The agreement maker and the counterparty’s agreement maker exchange proposals to create an agreement and the notary guarantees the reliability and non-repudiability of the created agreements.

Agreement Maker

Goal The goal of the agreement maker is to come up with created and signed agreements. To create those agreements, a negotiation process may be followed. Therefore, this is the role that implements our agreement creation mechanism and it must understand an agreement format and support at least one protocol to create agreements.

Data input The input to this role is a proposal or collection of proposals. Besides, the role makes use of the agreement preferences and the world model environmental resources.

Data output The output is zero or more (ideally one or more) agreements created and signed with one or more counterparties. These agreements
are then stored in the agreements repository environmental resource for later query and access.

**Behaviour** First, the *agreement maker* receives a collection of initial *proposals*. Then, it starts the pre-negotiation phase. During this phase, it decides with the other party which is the concrete negotiation protocol that is going to be used during the negotiation. Additionally, if it is necessary, the negotiation agenda may be established. Once the negotiation protocol has been agreed, the negotiation starts.

After the first submission of the initial proposal, the *agreement maker* and the *counterparty’s agreement maker* may start an exchange of proposals or bids (i.e. a negotiation process). There are no restrictions on the number of *counterparty’s Agreement Makers* that the *agreement maker* can be negotiating with simultaneously.

To carry out the negotiation process effectively, the role must include, at least, decision-making mechanisms to decide which response shall be sent to the counterparty and when and to construct the counterproposals or bids. Besides, it must make use of the *assessment mechanisms* of the *agreement preferences* and the *world model* to know characteristics about the counterparty we want to make an agreement with.

Before sending a binding proposal or accepting a proposed agreement, the binding organisation must be asked for permission. Therefore, when the *agreement maker* considers that a proposal must be accepted or that a binding proposal should be submitted, it sends that proposal to the *commit handler* in the binding organisation. We envision that most of the proposals exchanged between the parties will be non-binding proposals and binding proposals will be only sent when the proposal is appealing to us and the probability of being accepted is relatively high.

After the *commit handler* gives permission to commit to a proposal,
the agreement creation process starts. This process may involve the in-
teraction with a notary in order to guarantee the reliability and non-
repudiability of the created agreement. Finally, the agreement maker
stores the new agreement in the agreement repository environment
resource.

Counterparty’s Agreement Maker

Goal The counterparty’s agreement maker role represents the counterparty
that we are trying to reach an agreement with. This role is the antag-
onist of the agreement maker and, therefore, its goals are the same than
the goals of the agreement maker, i.e. to create and sign agreements pos-
sibly after a negotiation process.

Data input The input to the role is a proposal submitted by the agreement
maker.

Data output The output is the response to the proposal that may typically
be another proposal.

Behaviour The behaviour of the role depends on the communication protocol
established with the agreement maker (e.g. a take-it-or-leave-it protocol, a
bargaining protocol, or an auction protocol) together with the preferences
of the counterparty, obviously. As the counterparty’s agreement maker and
the agreement maker are exchanging offers in order to reach an agreement,
this role must implement the same communication protocol and agreement
format than the agreement maker.

Notary

Goal The Notary role must guarantee that the agreement created between the
two parties is reliable and non-repudiable.

Data input The input of the roles is the agreement document established be-
tween the agreement maker and the counterparty’s agreement maker.

Data output The output is the assurance of the reliability and non-repudiability
of the created agreement. Additionally, the notary can also maintain a
repository of all created agreements that it has verified to resolve possible
later disputes.

Behaviour The behaviour of the role depends on the protocol used to guaran-
tee the reliability and non-repudiability of the process. However, it must
include, at least, the submission of the signed agreement by both the agree-
ment maker and the counterparty’s agreement maker and a notification
that the agreement has been verified by the notary.
3.4.2 Collaborations

The organisation is composed of two collaborations: AgreementNegotiation (negotiate the proposals with the counterparty’s agreement makers) and AgreementCreation (create and sign the actual agreements).

There is also a dependency between the AgreementCreation and the AgreementNegotiation because to negotiate the proposals it is necessary to know which are the proposals that are going to be negotiated, and to create an agreement, it has to have been previously proposed to the counterparty during the AgreementNegotiation collaboration.

Nevertheless, several AgreementNegotiation collaborations with different counterparties can be carried out simultaneously. Moreover, these collaborations may be developed in parallel with other AgreementCreation collaborations, where agreements with other different counterparties are being created.

Finally, the AgreementNegotiation collaboration also depends on the ApprovalRequest collaboration from the binding organisation because to make a commit during the negotiation it is necessary to ask first for an ApprovalRequest.

**AgreementNegotiation**

**Goal** The goal of the collaboration is to reach an agreement between the involved parties by exchanging proposals\(^4\).

**Roles** Agreement maker and counterparty’s agreement maker

---

\(^4\)Note that a take-it-or-leave-it protocol also implies the exchange of proposals, specifically, the exchange of one proposal, and that an auction can be seen as an exchange of proposals between the bidders and the auctioneer.
Data exchanged The data used in this collaboration is a proposal.

AgreementCreation

Goal The aim of this collaboration is to actually create and sign an agreement and to guarantee that the created agreement is reliable and non-repudiable. The agreement has already been reached in the AgreementNegotiation collaboration.

Roles Agreement maker, counterparty’s agreement maker, and notary.

Data exchanged The data used is the agreement reached at the end of the AgreementNegotiation collaboration. This is the agreement that has to be signed and formally accepted by the parties involved in it.

Dependencies The AgreementCreation collaboration starts when both parties decide to create an agreement at the end of the AgreementNegotiation collaboration. Therefore the starting point of the AgreementCreation collaboration is the agreement reached during the negotiation. To be valid, the agreement must be formally approved and signed by both parties involved in it.

It is common to use a trusted third party to guarantee that the agreement that is being signed is reliable and non-repudiable. This third party is represented by the notary. We envision that the process of validation may involve the submission by both parties of the agreement to the notary, and the reception of an acknowledge from the notary indicating that the agreement is valid. However this process can be much more complex.

3.5 Binding

Goal The goal of the binding organisation is to determine when a binding proposal must be submitted and whether a binding proposal that has been received should be accepted. In addition, this organisation must establish when these decisions are going to be made. For example, one option is to make the decision as the proposals are received; another possibility is to make the decisions at some points in time that has been previously set. Therefore, the responsibilities of this organisation are not only to determine whether a binding proposal must be accepted or submitted but to establish when these decisions shall be made as well.

The rationale behind the binding organisation is that most of the time there will be several agreement negotiations being carried out simultaneously with different parties. Hence, there is a need for an element that coordinates the sending or acceptance of binding proposals to the other parties. In so doing, it is possible to avoid undesirable behaviour such as a service consumer committing simultaneously to two agreements to cover its needs when only one is necessary.
Moreover, we envision that most of the proposals exchanged between the parties will be non-binding proposals, while binding proposals will be only sent when it is really appealing to us and likely to be accepted by the other party. In so doing, negotiations can evolve more independently and quickly than sending binding proposals continuously because the coordination amongst them only occurs when a binding proposal is decided to be sent.

Requirements

The requirements for this organisation are:

- A mean to express the preferences about the contents of the agreement, the party, and the trading process.
- Elements that evaluate the external factors that may prevent a party to commit to an agreement and give advise based on that. These elements are expected to be mainly domain-specific.
- Mechanisms to make decisions about whether to commit to an agreement and about when this commit is to be made. The most important decisions of the whole system take place in this organisation because it is where the final decision about the acceptance of an agreement is made.
- A protocol to communicate with the agreement making organisation in order to coordinate when to commit to an agreement.

3.5.1 Roles

This organisation is composed of two roles, the commit handler and the capacity planner. Unlike the other roles of the abstract architecture, the capacity planner is a service provider-specific role that analyses the provider’s capability to provision an agreement. At this moment, we have identified this role in the provider side, however, we envision that more roles advising the commit handler whether to commit or not to an agreement may appear in both consumer and provider side.

Commit handler

Goal The commit handler role has the final decision on whether to bind to a proposal or not and it is also in charge of determining when these decision are made. To make these decisions it takes into account the user preferences about the contents of the agreement and the agreement process and it queries other roles about the feasibility of committing to an agreement.

Data input The proposal that it is wanted to be committed to. Additionally, the commit handler reads the agreement preferences and world model environmental resources to get information in order to make the decision.

Data output The output of the role is the permission or rejection to commit to a proposal.
Behaviour  When the agreement maker decides a proposal is interesting enough to commit to it and that it is likely to be accepted by the other party, it sends the proposal to the commit handler for approval. Then, the commit handler queries the commit advisers about the feasibility of committing to the given proposal. For instance, in the case of the service provider, the commit handler may inquire to the capacity planner about the provider’s capability to provision the proposal.

The recommendations of the commit advisers together with the information in the world model and agreement preferences environmental resources is used by the commit handler to decide whether to approve the proposal to be committed.

The other aim of the commit handler is to determine when these approval decisions are made. There are several approaches to it, the two options more relevant are:

- Making the decision as the proposals are received. This option is the easier to implement and decisions are very quickly made. However, if we just want to reach a limited number of agreements, we may miss some very good ones only because we previously accepted others that were not as good as them.
- Make the decisions at some points in time that have been previously set. These points may be dynamically selected, depending on changing conditions of the environment such as the frequency of arrival of proposals, or statically determined based on temporal constraints imposed by the trading protocol, or a combination of them both.
Therefore, there is no best option but it depends on the agreement preferences (e.g. the number of agreements we are willing to make, or the eagerness to reach an agreement) and the characteristics of the market and the other parties.

Capacity planner

Goal The Capacity Planner role analyses the provider’s capability to provision a certain agreement and recommends the Commit Handler to commit or not to that agreement. This role is specific to the concrete deployment of the service provider.

Data input A proposal requested for approval.

Data output A recommendation about whether to approve the commitment to the proposal.

Behaviour As this role is specific to the deployment of the service provider, there is no common behaviour to them. The unique behaviour that must be shared by all capacity planners is that when it receives a proposal, it has to respond to it with a recommendation about whether to approve the commitment to the proposal.

3.5.2 Collaborations

The organisation is composed of two collaborations, namely the ApprovalRequest (ask for approval on making a binding proposal), and CapacityQuery (ask for recommendation about the approval of a specific proposal). Therefore, there is a dependency between the CapacityQuery and the ApprovalRequest because every CapacityQuery is referred to an ApprovalRequest that has to be responded.

On the other hand, the ApprovalRequest collaboration depends on the Agreement Negotiation collaboration because the proposals that have to be approved by the ApprovalRequest are constructed during the negotiation process carried out in the Agreement Negotiation collaboration.

ApprovalRequest

Goal The goal of this collaboration is to get approval for the agreement maker to submit a binding proposal or accept a binding proposal from the commit handler.

Roles Commit handler and agreement maker

Data exchanged The data used in the collaboration is the proposal asked for approval.
Dependencies This interaction can be very simple or more complex depending on the coupling between the agreement maker and the commit handler and the way the commit handler determines when to make an approval decision. If the commit handler and the agreement maker are decoupled and the commit handler makes decisions about the approval as it receives them, then the interaction may be just one single message to ask for approval and the response.

However, if the commit handler uses information about the current status of the negotiations from the agreement makers or has to inform them when the following approval decision is going to be made, then the interaction may be very complex, with several points of synchronisation.

CapacityQuery

Goal The aim of this collaboration is to get specialised advise about the approval of a specific proposal.

Roles CapacityPlanner and CommitHandler.

Data exchanged The data used in this collaboration is the proposal and a recommendation about the feasibility of its approval.

3.6 Related standards and technologies

Several standards have emerged to enrich the basic web service stack that can be used in the implementation of the organisation that compose the reference architecture. Table 3.1 shows a distribution of standards over the organisations that have been identified.
Information There are a variety of standards that deal with the exchange of service descriptions, from both a functional and a non-functional point of view and they can be used in the implementation of the information organisation. For instance, WS-MetadataExchange [44] and WS-InspectionLanguage. Alternatively, WS-Agreement [2] uses a template-driven procedure, and those templates can be seen as a mean of expressing the preferences of a given party.

Agreement making The most significant specification that covers many aspects included in this organisation is WS-Agreement [2]. WS-Agreement is being developed by the GRAAP 5 Workgroup of the Global Grid Forum (GGF). On the one hand, WS-Agreement specifies the structure of an agreement document, so that it must be used together with one or several domain-specific vocabularies to give the proper semantic to the terms of the agreement. On the other hand, WS-Agreement defines a protocol and a web service-based interface to create, represent and allow the monitoring of agreements. Therefore, WS-Agreement may be used as the format to express the proposals exchanged as well as the communication protocol used to carry out this exchange of proposals.

However, the protocol defined by WS-Agreement is just a take-it-or-leave-it protocol, if we want more complex forms of negotiation other protocols must be used in this collaboration. WS-AgreementNegotiation is one of such negotiation protocols that is built on WS-Agreement and specifies a bilateral negotiation protocol. Nevertheless, its problem is that it is still in a very early stage of development. There are other negotiation protocol specifications that can be implemented by this collaboration, for instance, the negotiation protocols defined by FIPA, such as the FIPA Iterated Contract Net [16], the FIPA English Auction Interaction Protocol Specification [15] or the FIPA Dutch Auction Interaction Protocol Specification [14].

---

5Grid Resource Allocation Agreement Protocol
Chapter 4

Analysis of architectures for automated negotiation of SLAs

4.1 Properties of service trading architectures

In this section, we present a set of properties for automated negotiation systems of SLAs. These properties are obtained by analysing the conceptual framework of automated negotiations in Chapter 2. We must remark that these properties are conceived to be general, hence, they are just centred on high-level details of service trading architectures and do not cover lower-level elements such as concrete technologies, protocols or algorithms.

We distinguish two kinds of properties: functional and non-functional: (i) The functional properties are related to the features of the automated negotiation system. For instance, the kind of information it is prepared to manage, the ability of developing different types of negotiation or the use of elements to facilitate the decision-making such as a provider’s capacity estimator. (ii) The non-functional properties refer to high-level features that comprehend cross cutting concerns among all the automated negotiation system. In this way, these properties cover an heterogeneous range of topics such us flexibility or expressiveness.

4.1.1 Functional properties

1. **Information query** An information query is an inquiry made by one party to another to obtain more detailed information about it or about the service it provides or demands. Therefore, for an automated negotiation system to support information queries, it must have mechanisms to query services or to respond to those queries.
2. **World model** An automated negotiation system builds a world model if it analyses previous interactions with the elements external to the architecture, such as other parties or the discovery services, and uses the results to make better decisions [20] during the service trading process.

3. **Third party information** An automated negotiation system uses third party information if it explicitly queries a third party entity to obtain information related to a specific party. For instance, to obtain information about its reputation or its geographical location. In this case, a protocol to carry out this query as well as a shared taxonomy of terms must be supported by the architecture.

4. **Information managed about the parties** There are three types of information that can be managed about the parties:
   - Service information: Information about the functional and non-functional characteristics [35] of the service that a party demands or provides.
   - Trading information: Information about the features of the trading process followed by the party. For instance, its temporal constraints or the protocols that it supports.
   - Party information: Information about the party that provides or demands a service, such as its reputation or its geographical situation.

5. **Multiple negotiation protocols supported** An automated negotiation system supports multiple negotiation protocols if it is able to use a number of negotiation protocols such as different bilateral negotiation protocols or several auction protocols [20].

6. **Decommitment from previously established agreements** An automated negotiation system supports the decommitment [36] from previously established agreements if it can revoke previous agreements before the execution of the service, possibly by paying some compensation to the other party. This implies the implementation of any decommit protocol and the mechanisms to decide when a decommit is profitable for it.

7. **External factors in binding decisions**: An automated negotiation system may make use of a capacity estimator to determine whether the provider can provision a certain agreement before committing to an agreement. In so doing, the provider has a finer control about their resources and the implications of the agreements created [28].

8. **Cooperative or non-cooperative agreement creation**: An automated negotiation system supports non-cooperative agreement creation when it acts as a self-interested party reaching agreements with other self-interested parties. Alternatively, an automated negotiation system supports cooperative agreement creation when it can reach agreements with other parties trying to maximise only the social welfare.
9. **Consumer or provider orientation**: A reference architecture is consumer-oriented if it carefully describes the behaviour of the consumer (or the party acting on his behalf) in the service trading process. Alternatively, it is provider-oriented if it carefully describes the behaviour of the provider (or the party acting on his behalf). Note that a reference architecture may be both consumer and provider-oriented if it describes both behaviours.

10. **Assessment mechanisms**: The assessment mechanisms of an automated negotiation system is the kind of information used in the architecture to evaluate the goodness of a proposal or agreement in relation to some criteria provided by the user [43]. For instance, the most usual assessment mechanism in service trading is utility functions.

11. **Forms of expressing information and preferences**: The preferences and the information managed about the service and the parties can be expressed in different ways. Each automated negotiation system may have their own way to express them, however, the most commonly used are to express them as constraints or as rules.

### 4.1.2 Non-functional properties

The non-functional properties are a consequence of the functional properties and the design decisions made in the automated negotiation system. Unlike the functional properties, in most cases it tends to have a subjective evaluation, although some metrics could be extracted based on the analysis of functional properties. In this way, it is important to remark that we leave these properties outside the analysis of Section 4.1.1.

- **Expressiveness** The expressiveness of an automated negotiation system is related to the things that can be stated in the agreements, proposals or agreement preferences (assessment mechanisms, user preferences), managed by the system. The more expressive a system is, the more elaborated agreements can be reached and the more detailed may the agreement preferences expressed by the user be. However, the decision-making algorithms required to deal with that information are also more complex. The expressiveness property is related to functional properties such as *forms of expressing information and preferences, assessment mechanisms and types of information managed about the parties*.

- **Industry-orientation** The industry-orientation of an automated negotiation system measures the ability of the system to be used in a complex real-world scenario. Therefore, an automated negotiation system with a high industry-orientation must have elements to create reliable agreements, support non-cooperative agreement creation and must have different agreement creation mechanisms to be able to operate in a heterogeneous environment, and must be able to analyse and manage different types of information to make sure that the agreements created are profitable for the party.
• Flexibility An automated negotiation system is flexible if it can be adapted to diverse scenarios and users without making significant changes. Hence, a flexible architecture should have different deploy options, should be both provider and consumer oriented and should support different agreement creation mechanisms.

• Interoperability The interoperability of an automated negotiation system is related to the ability of the architecture to operate in a heterogeneous environment interacting with other systems that may be implemented using different technologies and languages to express the terms of the agreements and preferences.

4.2 Analysis of our reference architecture

In this section, our goal is to develop an analysis of the reference architecture proposed in Chapter 3. To achieve this goal, we check it against the properties obtained in Section 4.1.

1. **Information query** The information query is carried out by the Inquirer and the Informant roles of the information organisation (see Section 3.3.1).

2. **World model** A world model is built by the World modeller role of the information organisation. Then, this world model is used by the decision-making elements of the architecture to make decisions about the whole service trading process (see Section 3.3.1).

3. **Third party information** A mechanism to query third parties in order to get information about other parties is designed in the information organisation (see Section 3.3.2).

4. **Information managed about the parties** The abstract architecture manages all three types of information: service, trading and party information as it is stated in Section 3.2.

5. **Multiple negotiation protocols** The abstract architecture supports multiple negotiation protocols by changing the implementation of the Agreement maker role (see Section 3.4).

6. **Decommitment from previously established agreements** Currently, there is no support for decommitting from previously established agreements.

7. **External factors in binding decision** The commit handler role of the binding organisation may query a capacity estimator to determine whether the provider can provision a certain agreement before committing to an agreement (see Section 3.5).

8. **Cooperative or non-cooperative agreement creation** The abstract architecture can be applied successfully to both a cooperative and non-cooperative scenarios.
<table>
<thead>
<tr>
<th></th>
<th>Ours</th>
<th>Tu et al.</th>
<th>PANDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(2)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(3)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(4)</td>
<td>Service, party, trading</td>
<td>Service</td>
<td>Service, party</td>
</tr>
<tr>
<td>(5)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(6)</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(7)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>(8)</td>
<td>Both</td>
<td>Seems non-coop</td>
<td>Seems non-coop</td>
</tr>
<tr>
<td>(9)</td>
<td>Both</td>
<td>Both</td>
<td>Both</td>
</tr>
<tr>
<td>(10)</td>
<td>Not limited</td>
<td>Utility functions</td>
<td>Utility functions + rules</td>
</tr>
<tr>
<td>(11)</td>
<td>Not limited</td>
<td>Rules</td>
<td>Utility funcs. + rules</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of proposals

9. **Consumer or provider orientation**: Both consumer and provider behaviours are carefully described by the abstract architecture.

10. **Assessment mechanisms**: Several assessment mechanisms may be used as part of the *agreement preferences* in the abstract architecture. The only limitation is that they must allow a common coherent ranking amongst the elements of the abstract architecture when applied to the same group of proposals (see Section 3.2).

11. **Forms of expressing information and preferences**: Preferences and information about the parties are expressed by using terms. A term expresses functional or non-functional guarantees of the service that must be observed by the parties during its execution as well as other aspects of an agreement such as termination clauses (see Section 3.2).

### 4.3 Comparison with other proposals

In this section we compare our abstract architecture with two significant architectures for automated negotiation of SLAs. The work developed by Tu et al. [42] 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 is not possible to change dynamically the negotiation strategy depending on the state of the negotiation; it does not distinguish several phases in the decision making process (everything appears in the same module), and it does not define how the protocol plug-in interacts with the strategy plug-in.

In PANDA [17] a framework is proposed that mixes utility functions and rules to carry out the decision making process. A set of communication primitives is used to make independent the decision making of the negotiation protocol like in our framework. 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, this component is vaguely defined and it establishes neither the interface of the estimation libraries objects nor the process that must be followed in the decision-making. Another problem is that the strategy selection is done through rules created manually.

Table 4.1 shows the results of the comparison. As it can be seen, our proposal covers the majority of the properties (excepting decommitment). Therefore, it is well suited for complex service negotiation scenarios.
Chapter 5

Conclusions

5.1 Analysis of previously set goals

In Chapter 1, we set three goals that we want to achieve in this work. Our aim in this section is to check whether those goals have been achieved, and why they have been achieved or not. The established goals are the following:

1. Analysing the problem of automated negotiation of service level agreements from a software engineering perspective.

   We believe that the conceptual framework described in Chapter 2 fulfils this goal. In that chapter, we extend the classical elements of automated negotiation with two new ones: information and preferences and we also detail more precisely the parts that compose the other three: negotiation object, protocol and decision-making. This conceptual framework is very useful to compare the different automated negotiation proposals. Furthermore, it identifies the elements that must be taken into account when developing a system for automated negotiation of SLAs.

2. Categorising and classifying the most significant automated negotiation proposals.

   This goal is accomplished in Chapter 2. In that chapter, we use the conceptual framework to categorise the most significant proposals in automated negotiation. We distinguish between a number of approaches to negotiation protocols and decision-making models.

3. Developing a reference architecture for SLA automated negotiation.

   This goal is successfully achieved by Chapter 3. In that chapter, we propose a novel reference architecture for automated negotiation of SLAs. The reference architecture has been described following the organisational metaphor proposed in GAIA [46]. Therefore, the architecture is composed of three organisations (information, agreement making and binding) with a well defined goal and a low number of communication lines amongst
them. In that chapter, we also specify the roles and collaborations that form part of the reference architecture\(^1\). Finally, an analysis of the reference architecture is carried out in Section 4.2.

Therefore, we can conclude that all goals have been achieved successfully.

## 5.2 Future work

There are several research lines that can be derived from this work. In particular, we foresee three different lines that may open interesting fields where further research can be done. These lines are: extending the categorisation work of current automated negotiation proposals, developing a standards-based architecture for automated negotiation of SLAs taking as a starting point the reference architecture described here, elaborating a proposal for cooperative negotiation, and extending the architecture to support argumentative approaches.

### Extending the categorisation work of current automated negotiation proposals

In this work, we categorise the current proposals on automated negotiation by using the conceptual framework defined in Chapter 2. Further work in this line may be to refine this categorisation in order to identify which alternative is better in each case. In this way, an automated change of the algorithms used to develop the negotiation can be made, depending on the context of the negotiation.

### Developing an standards-based architecture for automated negotiation of SLAs

Although there exist some implementations of architectures for negotiation of SLAs such as PANDA [17], to our knowledge, no one gives full support to the most complex scenarios described here. Therefore, a concrete architecture for negotiating SLAs may be developed. This architecture must be based on the reference architecture defined in this work. In addition, it must use standards of the industry in order to better interoperate with other automated negotiation systems.

After developing the architecture, it may be deployed in several real scenarios with the different algorithms proposed in the literature in order to validate it. Additionally, this architecture may be used as a test-bed for new decision-making models and negotiation protocols.

### Elaborating a proposal for cooperative negotiation

The problem of cooperative negotiation has not been directly tackled in this work. Although, there is no limitation about using the reference architecture proposed here in a cooperative environment, no study has been made on that aspect. Therefore, one line of further work is to analyse the suitability of this approach to these scenarios. This may include some modifications of the reference architecture.

\(^1\)For a detailed description of the modeling elements that have been used, see Section 3.1.1
and also the development of new decision-making mechanisms and protocols that allows the optimisation of specific cases of cooperative negotiation.

**Extending the architecture to support argumentative approaches**  In Section 3.1, we detail why we decide to focus on proposal-based negotiations instead of other types of automated negotiation. However, in Chapter 2, we analyse the disadvantages of these proposal-based negotiations and how they are solved by argumentation approaches. Nevertheless, the problem of that approach is that the automated negotiation systems tend to be very complex. A future line of research is to extend the proposed architecture with elements that simplify the development of such automated negotiation systems.
Bibliography


