An Abstract Architecture for Service Trading.

On the Design of the Framework

Pablo Fernandez, 28782044-M
pablo@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 Motivation and Goals of the report .......................... 5  
   1.2 Report Structure ............................................ 6  

2 Background information  
   2.1 Scenarios ...................................................... 7  
   2.2 Properties of service trading architectures ................ 10  
      2.2.1 Functional properties ................................. 10  
      2.2.2 Non-functional properties ............................. 13  
   2.3 Analysis of service trading architectures .................. 14  

3 An abstract architecture for service trading  
   3.1 Introduction and motivation ................................. 19  
      3.1.1 Modelling elements .................................... 20  
      3.1.2 Organisations behaviour overview ..................... 22  
   3.2 Data structures .............................................. 24  
   3.3 Trading ...................................................... 29  
      3.3.1 Roles .................................................... 30  
      3.3.2 Collaborations .......................................... 32  
   3.4 Discovery .................................................... 33  
      3.4.1 Roles .................................................... 33  
      3.4.2 Collaborations .......................................... 36  
   3.5 Information .................................................. 39  
      3.5.1 Roles .................................................... 40  
      3.5.2 Collaborations .......................................... 43  
   3.6 Selection .................................................... 46  
      3.6.1 Roles .................................................... 47  
      3.6.2 Collaborations .......................................... 50  
   3.7 Agreement Making ............................................ 54  
      3.7.1 Roles .................................................... 55  
      3.7.2 Collaborations .......................................... 58  
   3.8 Binding ...................................................... 60  
      3.8.1 Roles .................................................... 62  
      3.8.2 Collaborations .......................................... 64
3.9 Related standards and technologies .............................................. 65

4 Conclusions .............................................................................. 68
  4.1 Analysis of the abstract architecture ................................. 68
  4.2 Analysis of previously set goals ............................................. 70
  4.3 Future work ............................................................................ 72

A Curriculum vitae ............................................................... 74
List of Figures

1.1 Contracting process and service trading process ............ 4
3.1 Abstract architecture ........................................ 21
3.2 Abstract architecture - Collaborations view ................. 23
3.3 Data structures ............................................... 24
3.4 Trading organisation - Roles and data dependencies ........ 30
3.5 Trading organisation - Collaborations diagram .............. 32
3.6 Discovery organisation - Roles and data dependencies .... 34
3.7 Discovery organisation - Collaborations diagram .......... 37
3.8 Information organisation - Roles and data dependencies ... 41
3.9 Information organisation - Collaborations diagram ......... 44
3.10 Selection organisation - Roles and data dependencies .... 47
3.11 Selection organisation - Collaborations diagram .......... 51
3.12 Agreement making organisation - Roles and data dependencies ... 55
3.13 Agreement making organisation - Collaborations diagram .. 59
3.14 Binding organisation - Roles and data dependencies ...... 62
3.15 Binding organisation - Collaborations diagram .......... 64
List of Tables

2.1 Comparison of abstract architectures........................................ 16
3.1 Related standards..................................................................... 66
4.1 Comparison of our abstract architecture................................. 71
Acknowledgements

I would like to express my gratitude to all the people of LSI Department for their support, motivation, and courage. In a special place, I would like to mention my colleague and friend Manolo that is partially responsible of this document. Last but not least, I would like to thank my supervisor Rafael, this report could not have been written without his dedicated efforts.
Resumen

En el contexto de SOA (Arquitecturas orientadas a servicios), los acuerdos de nivel de servicio (SLAs) están adquiriendo una gran importancia tanto en escenarios interorganizacionales como intraorganizacionales por las ventajas que aportan. Por un lado, los SLAs ofrecen a los consumidores garantías acerca de cómo le será suministrado un servicio. Por otro lado, los SLAs permiten a los proveedores llevar a cabo un aprovisionamiento automatizado de los servicios basado en los SLAs que han acordado con sus clientes. Los acuerdos de nivel de servicio son un importante área de investigación en la actualidad tanto en la industria como en el ámbito académico. En esta memoria, nos centramos en el proceso de trading de servicios, que es el proceso de localizar, seleccionar, negociar y crear SLAs. Este proceso se puede aplicar a multitud de escenarios distintos y, por tanto, sus requisitos son también muy distintos.

Pese a que se han propuesto algunas arquitecturas para el trading de servicios, éstas no son apropiadas para cualquier escenario. En particular, sus puntos más débiles son que no cubren completamente todos los aspectos de los escenarios más complejos de trading de servicios y que carecen de soporte para la implementación de mecanismos avanzados de toma de decisión automática. Para resolver estos inconvenientes, hemos desarrollado una arquitectura abstracta de trading de servicios. Para nosotros, una arquitectura abstracta es una especificación de un conjunto de elementos (subsistemas, componentes, interfaces, tipos de datos o colaboraciones) y que pueden ser aplicada a distintos dominios e implementada con tecnologías y estándares diversos.

Los objetivos de la arquitectura abstracta son dos: debe ser capaz abordar escenarios complejos de trading de servicios y debe dar soporte completo para implementar mecanismos avanzados de toma de decisiones automática. Para alcanzar este objetivo, tomamos como punto de partida un conjunto de propiedades de arquitecturas abstractas de trading de servicios obtenidas como resultado de un análisis de diversos escenarios prácticos. A continuación, especificamos la arquitectura abstracta por medio de los roles y colaboraciones necesarios para llevar a cabo este trading de servicios automatizado. Además, definimos los elementos necesarios para dar soporte a una toma de decisión automática avanzada. Por último, también motivamos y introducimos la idea de protocolo de trading, que es una especificación de las restricciones temporales impuestas en el proceso de trading y las coreografías utilizadas en cada fase del proceso.

En resumen, la aportación de este trabajo es triple. En primer lugar, identificamos las principales características del proceso de trading de servicios. En segundo lugar, analizamos y comparamos las arquitecturas abstractas de trading de servicios más significativas. Finalmente, desarrollamos una arquitectura abstracta para el trading de servicios que aborda escenarios complejos y que soporta una toma de decisiones automática avanzada. Además, introducimos y motivamos el concepto de protocolo de trading.
Abstract

Many software companies are using SOA (service-oriented architectures) as the cornerstone for their business activities. In this context, service level agreements (SLAs) are gaining importance in both cross-organizational and intra-organizational scenarios because they provide important benefits to both the service consumer and the service provider. On the one hand, SLAs grant consumers guarantees about how a service will be provided. On the other hand, SLAs allow the providers to deploy an automated provision of services based on the SLAs on which they have agreed with their customers. SLAs are a prominent research field in both the academy and the industry. In this report, we focus on the service trading process, which is the process of locating, selecting, negotiating, and creating SLAs. This process can be applied to a variety of scenarios and, hence, their requirements are also very different.

Despite some service trading architectures have been proposed, they are not suitable for every single service trading scenario. Particularly, they do not cover well all aspects of complex service trading scenarios and they lack support for the implementation of advanced automated decision-making. To solve these shortcomings, we have developed an abstract architecture for service trading. We understand an abstract architecture as a specification that defines a set of elements (subsystems, components, interfaces, data types, or collaborations) and that may be applied to different domains and implemented with a variety of technologies.

The goals of the abstract architecture are that it must tackle complex service trading scenarios and it must give full support to implement advanced automated decision-making mechanisms. To reach these goals, we take as a starting point a set of properties for abstract service trading architectures obtained as a result of an analysis of several practical scenarios. Next, we specify the abstract architecture following a role-based approach. Therefore, the abstract architecture is described by means of the roles and collaborations that are necessary to develop this automated service trading. Furthermore, we define the elements that are necessary to support an advanced automated decision-making in service trading. In addition, we motivate and introduce the idea of trading protocols. A trading protocol specifies the temporal constraints imposed in the trading process and the choreographies used in each stage of the process.

In summary, the contribution of this work is threefold. First, we identify the main characteristics of the service trading process. Second, we analyse and compare the most significant abstract architectures for service trading. And third, we develop an abstract architecture for service trading that deals with complex scenarios and supports advanced automated decision-making. Additionally, we introduce and motivate the concept of trading protocol.
Chapter 1

Introduction

SOA [20] has irrupted into software development world as a new approach to deal with complexity of systems. This architecture is the result of a widely accepted web service framework [5] composed of a set of standards that is continuously extended forming an integrated stack of technologies.

Within SOA, complex systems can be seen as a collaborative environment of services. Each of those services interacts with each others through a standard-base expression of its capabilities. Furthermore, the aim of this infrastructure would make possible not only a discovery and use of services but also composition (workflow of services) and quality of service assurance mechanisms.

There are two main scenarios where SOA shows its benefits, Intra Organisational scenarios (also known as EAI: Enterprise Application Integration) and Cross Organisational scenarios (or B2B: Business to Business) [4]: (i) On the one hand SOA provides a flexible way to integrate different systems in the enterprise; regarding this scenario it has been proposed [20] the idea of an enterprise service bus as the next generation of integration middleware that would act as a central infrastructure of the whole organization (ii) On the other hand, interoperability among companies has been the holy grail pursuit by different technologies that have appeared in the last decades. At this point, web services initiatives can be seen as a step forward in that direction with a set of open standards that make possible a high degree of interaction between loosely coupled systems.

In this context, an example of the deep adoption of the service-orientation in the industry can be found in the Grid Services [11] initiative that aim to make use of the service-oriented computing paradigm [5] in the traditional Grid environment. This approach opens new promising possibilities of hybrid scenarios cross/intra-organisational by virtualising the resources of the grid as services.

Business to business Concerning the B2B, though service orientation has appeared as a new concept in the software, this approach has been widely

\footnote{More info in W. Chappel, Enterprise Service Bus, O’reilly 2004}
adopted in other industries. Among other consequences, service orientation boosts an special kind of collaboration between corporations: the so called, outsourcing [23] of services. Through this mechanism, a given enterprise would delegate some part of its business process to an external one. In order to understand the motivation and benefits of outsourcing, we can briefly analyse a classical industry: the supply chain management.

A supply chain is a network of facilities and distribution options that performs the functions of procurement of materials; transformation of this material into intermediate and finished products; and distribution of these finished products to customers. In this kind of scenario, enterprises based its business in the outsourcing of different parts of the chain. From a customer point of view, these companies appear as a black box that serves products; however, the main source of profit comes from the integration and management of the whole chain instead of the product building itself.

In the case of software industry, the idea of outsourcing has been adopted in last decades. There have been some specific studies of these techniques in IT companies and major benefits have been identified [31]: Reduction of risks, quick adoption of new technologies, optimisation of resource exploitation and cost saving. Traditionally, this outsourcing has been related to long term relationship between corporations where a given enterprise subcontracts a part of the system for a certain period of time (and , in some cases, forever).

However, with the arrival of SOA, we can foresee a new approach to this idea: the dynamic outsourcing.

**Automated outsourcing** Service orientation in software development provides the perfect scenario for a short term outsourcing that can be dynamically created to fit the business needs of each moment. This would mean, a next step to an automatic procurement of services that will allow the customer to choose the best provider according to its business rules at run time.

Nevertheless, in order to achieve a high level of dynamism in the outsourcing phase, we should move toward an automated process that can reduce the cost and time of creating the business partnership among organisations involved. In fact, in the heart of the outsourcing process we can find the service level agreement (SLA) concept that would regulate the responsibilities and guarantees of every party in the business relationship.

**Enterprise Application Integration** Although SOA has been traditionally boosted as a promising field in the cross-organizational transactions, we can see that SOA is also making its way into the integration of complex organizations.

In this context, it is important to remark that the majority of software development companies, are proposing different approaches for the integration of IT systems based on SOA. However, in spite of the differences about proposed infrastructures, in general, the classical concept of integration middleware is evolving into an idea of service bus that would drive the interoperability of organizational subsystems with a SOA focus.
In particular, the so called ESB (Enterprise Service Bus) generally provides an abstraction layer on top of a messaging infrastructure which allows a higher level integration that exploit the value of messaging maximising the composition of existing services without ad hoc implementation. This integration approach differ from classical EAI brokers which are usually implemented as a fixed monolithic stack, the foundation of an enterprise service bus is built of base standardized functions broken up into parts, with distributed deployment where needed, working in harmony as necessary.

Concerning the QoS, the addressing of SLAs are starting to be an important issue amongst the integration of subsystems. In so doing, an extension of ESB capabilities to deal with this issues appear as an interesting aim. In particular, amongst other benefits, an automation of SLA management could result in a better rationalisation in the usage of resources inside the integrated organisation.

**Service level agreements** As we have motivated, the idea of an automated management of SLAs (Service Level Agreements) is a key point in both B2B and EAI scenarios. In general, this agreements are means 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 parties during the service execution.

In the field of SLA management, several research efforts are facing open problems effectively but on the other hand, some areas have not been addressed. As an instance there is enough infrastructure to provision SLAs and services [18] that agree with them automatically, but there is little support to create the agreements themselves, which is still a human-centric process. In this report we face this unresolved problems that keep systems from an effective automated management of the SLAs.

**Contracting process** In order to address an automated management of SLA, it is necessary to identify the different phases that should be followed to create and make use of SLA; this has been called in the literature as the *contracting process*[16] (as it is shown in the Figure 1.1). Outside of the contracting process, there is a special stage called as preparation phase that involves the creation of the offer by the provider of the service and the analysis of its functional and non-functional requirements by the consumer. In the contracting process itself, first step is defined as information phase whose goal is to match service providers with potential consumers and vice versa. In the next phase, they may start a negotiation with those consumers or providers to find a mutually acceptable agreement. At the end of this phase the result is the creation of an agreement on the execution of a service between a provider and a consumer. In the fourth phase, both service provider and service consumer set up a deployment plan to make it possible to follow all terms established in the agreement settled in the previous phase. The last phase in the contracting process is the fulfilment phase. This phase involves the fulfilment of the obligations established in the agreement and in the monitoring of the whole process in order to ensure that...
both parties observe the agreement correctly.

**Service trading process** Building on the top of the contracting process, we focus in a subprocess derived from it. Concretely, we address the so called service trading process defined as the process of locating, selecting, negotiating, and creating SLAs. Therefore the service trading process is a subprocess that covers the information and negotiation phases of the more general contracting process.

The input to this process consists in the *agreement preferences*, which is a set of information (requirements or features) gathered during the preparation phase (of the general contracting process). In order to formalise the nomenclature, the actor that provides the *agreement preferences* is called *the user* in the rest of the report. The output of the service trading process is an established SLA that can be used to drive the further deployment and fulfilment phases of the contracting process.

We argue that automating this service trading process is very appealing in the point of view of both a service provider and a service consumer. For the service provider, this automation enables an automated provision [18] of services based on the agreed SLAs which allows a better optimisation of the available resources in the systems. For the service consumer, the automation enables a rapid adaptation to the changes in the market.
1.1 Motivation and Goals of the report

The main goal of this research report is fourfold:

- To analyse the problem of the service trading by studying several scenarios. This analysis is intended to identify a set of abstract properties that leads to an effective characterisation of the problem of automated service trading.

- Based on the identified properties, to carry out an analysis of abstract architectures that deal, in some way, with the service trading process. The result of this study is intended to be used as the basis for the construction of a better abstract architecture.

- A development of a new abstract architecture that could cover complex scenarios by making a flexible definition of elements and identifying the decision-making components needed for the automation of management.

- An study of the different current technologies in order to establish a classification table that represents the implementation choices that could be used to develop more concrete architectures upon the proposed abstract one.

In this way, the main motivation of this report is the achievement of an effective abstract architecture that could deal with the automation of the trading process. In so doing, it is important to remark the design principles that guide our abstract architectural proposal:

- First, maximum flexibility. Our proposal is divided into several conceptual organizations and roles that enable several levels of variability since new trading and negotiation protocols may be integrated if necessary, as well as new technologies and standards.

- Second, maximum adaptability to new scenarios from an integration to a cross-organizational one.

- Third, maximum business opportunities: First, our intend is to establish the figure of a potential Trader of services, that would boost a market of services as a network of traders that act as domain-specific intermediaries. Second, concerning the deployment of an specific Trader, we put forward the idea of a development based on the composition of different business entities specialised in scopes such as negotiation, service matching and so forth.

- Lastly, the symmetry of the proposal is a key factor that has been addressed. In this way our abstract architecture is independent of the nature of the stakeholder, i.e. whether it acts as the service consumer or the service provider.
1.2 Report Structure

The report structure is divided in four main blocks as following: Introduction (Chapter 1), Background Information (Chapter 2), Abstract Architecture (Chapter 3) and Conclusions (Chapter 4). The fist two chapters provide the conceptual grounding for our proposal.

In particular, in the next chapter (Chapter 2), we develop an extensive analysis of the background of the related concepts and works in order to establish a strong basis for the further development of a new abstract architecture. Concretely, we focus on the abstract architectures that define a set of elements for service trading process. Our main goal in this chapter is to define a set of properties for these abstract service trading architectures based on an analysis of several service trading scenarios. These properties enable the analysis and comparison of those abstract architectures, which is an effective grounding for discovering the weaknesses and strengths of each one of them and give an important step towards a development of the new architecture.

In chapter 3, based on the taxonomy of properties developed in chapter 2, a novel abstract architecture is proposed. This proposal addresses, the whole service trading process by defining a set of abstract elements and architectural guidelines that can be used to build more concrete architectures that deal with the service trading process. In the last part of the chapter, an analysis of the current standards and technologies is shown.

Finally in the last chapter (Chapter 4) we presents a set of conclusions. In particular, we analyse the set of properties identified in Chapter 4 over our architectural proposal of Chapter 3.
Chapter 2

Background information

The service trading process is the process of locating, selecting, negotiating, and creating SLAs. Therefore, the service trading process is a subprocess of the more general contracting process [16] that covers the information and negotiation phases. The characteristics of the service trading process depend on the particular scenario where it is developed. There are a variety of scenarios that 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 (see Section 2.1 for more information). 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.

We focus on abstract architectures for service trading, which are specifications that define a set of elements for service trading. These abstract architectures can be later implemented by using different technologies and applied for different problem domains. Our goal is to define a set of properties for these abstract service trading architectures based on an analysis of several service trading scenarios. These properties enable the analysis and comparison of those abstract architectures and help in the process of developing new abstract architectures.

This chapter is structured as follows. First, in Section 2.1, we present four service trading scenarios that serve as the basis for the set of properties for abstract service trading architectures presented in Section 2.2. In Section 2.3, these properties are used to analyse and compare the most relevant abstract architectures for service trading.

2.1 Scenarios

In this section, we describe a set of scenarios that correspond to different cases of service trading that have been selected based on the diversity of features, stakeholders and domains: e.g. this set cover from service consumers that actively
search for providers to the more passive ones that wait for providers to make an offer. Additionally, it is worth pointing out that despite the services described in these environments are mostly electronically delivered, our vision also tries to cover the idea of classical services that can be managed automatically (i.e. trading process management) with similar architectures.

Scenario 1. Service consumer looking for ISPs

This case relate to a mid-large size company that looks for an ISP (Internet Service Provider). In this scenario, a company publishes its demands and wait for the ISPs to make their offers. In so doing, the company has a passive role while the providers act as active organisations searching for customers.

In this domain and from the point of view of the company, it is appealing to have a periodic renegotiation of the service. Furthermore, a high level of automation in the service trading enables that every renegotiation is open to different ISPs in order to select the best possible in each case; in this way, it is boosted a dynamic market where each provider look forward competitive offers adjusted at their capabilities in each moment. An additional issue is the strict temporal sequencing of the service trading process. The trading process should coherently encompass the stages to fit the temporal constraints for the company to avoid problems such as a lack of the service due to the change of ISP.

In this scenario, the QoS terms during the SLA establishment are a key factor. In this way, an interesting feature is to be able to automatically negotiate such features; moreover, temporal intervals for the service are aspects to be taken into account in that process. Concerning the decision-making process, the information known about providers is the most important element; i.e. the reputation of provider or the historic knowledge based on previous trading process. Lastly, one relevant issue is the non-repudiation mechanism for established SLAs. Participant organisations should have the guarantee that the agreements reached by the system are legitimate.

Scenario 2. Computing services provider

In this case, a company offers computing services to other organisations. In particular, this case is becoming increasingly popular in research fields with intensive computational needs such as bioinformatics. In a concrete manner, the company in this scenario can be described as a computing service provider that receives demands from other organisations in terms of computing jobs to be developed.

In this scenario it should be allowed for the company to specify offers that optimise the usage of its resources. Specifically, in a computing company, unused (or low used) resources means a decrease in the recovery of the initial investment. In so doing, offers should vary based on the resource usage and the set of SLAs the company has reached with its customers. Closely related with those ideas, from the perspective of the customer, a negotiation of the terms of the SLA is an interesting issue to be addressed. Moreover, this negotiation process can be used by the provider to slightly adapt the final SLA and make concessions or restrictions in order to optimise the current usage level of its resources.
Additionally, as those offers are tightly adjusted to the resource status in each moment, the decision making infrastructure should also take into account this information as a first level element before establishing new commitments with a customer. Finally, it is interesting to remark that, unlike the previous case, the reputation information is not an important issue from the perspective of the computing provider.

Scenario 3. Company delegating to a trader specialized in VoIP
This case describes a company that delegates its telephony needs to a trader that handle its requests and locate the best possible VoIP (Telephony through Internet) provider for each call. This trader represents an organisation that makes profit acting as a facilitator between end-user companies and VoIP providers. These providers offers different services characteristics (e.g. Bandwidth guaranteed for the call) or restrictions (e.g. Some of them could only operate between certain countries). In so doing, this trader offers a service of calls management by creating concrete agreements for each call with the telephony provider taking into account the preferences of the company: e.g. minimising cost.

The information about the telephony service providers can be divided into two sets: First, a set of information describing the capabilities of the provider in general terms such as the operational countries or time slots classification (Peak hours, reduced cost hours, etc.); this set can be used to create a selection of potential providers. Second, in each moment, when handling an specific call, the trader can ask about last minute offers from the providers; this offers would be based on the resource status of provider (as in the previous scenario). In so doing, based on the information harvested, the trader can construct the specific SLA proposal the most appealing provider and, finally, if it agrees, the final SLA is established and the call can be carried out.

Scenario 4. A generic service trader in a supply-chain One of the scenarios where service trading fits better is supply-chains, where each organisation create added value by composing services from different providers. In this case, a trader of services represents organisations that create high-level services based on the composition of lower-level services. Many examples can be found in the literature from telecommunications domain [14] to the transport domain [11]. This idea of supply-chain can be isolated from a specific domain and, hence, the elements and requirements expressed in this scenario are mostly valid for the majority of supply-chains independently of the nature of the services supplied.

In this scenario, the key point to be addressed is an efficient composition of services that adds value to the customer whose the trader sells its services. To achieve its goals, an aspect to be addressed is the adaptability to different markets. To a service trader the ability to understand different ontologies is important because it allows him to communicate with different markets or providers. Closely related with the previous ideas, information harvested about different providers is necessary for an efficient decision making process of selection of services. For each potential service provider, the trader should ask
to several sources: the provider itself for information about the capabilities it claims to have, and third parties that can also supply important information about the reputation of a certain provider.

In order to construct a composed service, the trader should agree several SLAs with providers for each of the services that will be composed. In this way, during the establishment of every SLA, three processes are relevant: (i) a classification of the proposals (coming from the providers); (ii) a selection of the most promising proposals; (iii) a decision about the handling process for each of the selected proposals: e.g. whether we negotiate them or not.

2.2 Properties of service trading architectures

In this section, we present a set of properties describing features of abstract architectures for service trading. These properties are derived from the four scenarios described in Section 2.1. For each property a reference to the related scenario is supplied; concretely, this references are based in the special importance of the property for the specified scenario. We must remark that these properties are conceived to be applied to abstract architectures and, 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 abstract architecture. 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 abstract architecture. In this way, these properties cover an heterogeneous range of topics such us flexibility or expressiveness.

2.2.1 Functional properties

1. External discovery (S.1, S.2, S.3 and S.4): We say an abstract architecture has an external discovery process if the architecture uses an external infrastructure to obtain the list of potential parties that demand (or supply) a service that other party provides (or needs), such as an external registry of services. Alternatively, the discovery process is internal if no external infrastructure is used, for example, if the list of potential parties is directly provided by the user. If the discovery process is external, it can be either centralised or distributed. It is centralised if the architecture specify a common registries of services shared by all stakeholders. It is distributed if the architecture obtain the parties following a progressive discovery amongst different sources following some shared protocol such as an event-driven protocol [12] between previously-known elements or a more distributed P2P (Peer to peer) protocol [28].
2. **Knowledge adaptation (S.4)**: In this context, an abstract architecture has knowledge adaptation [15] [10] if it provides elements to adapt the local knowledge model to the appropriate discovery infrastructure, making independent the characteristics of the market modelled by the discovery service to the rest of architecture.

3. **Market observation (S.3 and S.4)**: An abstract architecture with market observation monitors the changes in the market through observation of the information provided by external discovery infrastructures and informs about these changes to the elements of the architecture.

4. **Symmetric architecture for providers and consumers (S.4)**: An abstract architecture is symmetric if both service provider and consumer can start the service trading process and there is no commitment as to which party advertises and which party queries to the discovery service. Alternatively, an abstract architecture is asymmetric if only one consumer or provider can start the service trading process.

5. **Information query (S.3 and S.4)**: 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 abstract architecture to support information queries, it must have mechanisms to query services or to respond to those queries.

6. **World model (S.1, S.3 and S.4)**: An abstract architecture 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 [13] during the service trading process.

7. **Third party information (S.1, S.3 and S.4)**: An abstract architecture 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.

8. **Information managed about the parties (S.1, S.2, S.3 and S.4)**: There are three types of information that can be managed about the parties:

   - **Service information**: Information about the *functional* and *non-functional* characteristics [24] 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.
9. **Proposals preselection (S.3 and S.4)** An abstract architecture has a proposals preselection process if, before starting an agreement creation process or a negotiation, it ranks and/or filters the proposals that it has received or built based on criteria previously specified.

10. **Agreement creation mechanisms (S.1, S.2 and S.4)** An abstract architecture has multiple agreement creation mechanisms if it supports different protocols to reach to an agreement. These mechanisms can range from a take-it-or-leave-it protocol [1] to a bilateral negotiation protocol or an auction protocol [13].

11. **Notary (S.1 and S.4)** An abstract architecture has this property if it provides any mechanism to guarantee that the agreement created between the two parties is reliable and non-repudiable [19]. We say an agreement is reliable if both parties are signing and accepting the same previously agreed document.

12. **Decommitment from previously established agreements (S.1 and S.4)** An abstract architecture supports the decommitment [25] 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.

13. **Capacity estimator (S.2 and S.3):** An abstract architecture 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 [18].

14. **Trading protocols (S.1, S.2, S.3 and S.4):** A trading protocol is a set of stages (e.g. advertisement, proposal submission, negotiation, resolution, etcetera) cross-linked in accordance to some temporal constraints and bounded to some choreographies [21]. The temporal restrictions specify a set of constraints about the life-cycle of the trading process. These restrictions can vary from simple fixed temporal points (e.g. End by 14:00 of 14th, March) to complex relationships amongst the durations of some stages (e.g. Information stage starts in the middle of the discovery stage). Therefore, an abstract architecture that supports different trading protocols must be able to deal with different temporal constraints on the stages of the service trading process.

15. **Creation of agreements for composed services (S.4):** A composed service [3] is a service whose implementation is based on the execution of other services that may be provided by external entities and, hence, there may exist agreements regulating that execution. The support for creating agreements for composed services can vary significantly. In its simplest form it just ensures that dependencies between the services such as “I
want either an agreement on all different services or no agreement at all" are fulfilled. The most complex form takes into account the service level properties desired for

16. **Cooperative or non-cooperative agreement creation (S.1, S.2, S.3 and S.4):** An abstract architecture supports non-cooperative agreement creation when it acts as a self-interested party reaching agreements with other self-interested parties. Alternatively, an abstract architecture supports cooperative agreement creation when it can reach agreements with other parties trying to maximise only the social welfare.

17. **Consumer or provider orientation (S.1, S.3 and S.4):** An abstract 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 an abstract architecture may be both consumer and provider-oriented if it describes both behaviours.

18. **Deployment options:** An abstract architecture may present several deployment options depending on their characteristics. Some examples of deployment are to integrate the architecture in the service provider or to implement an independent trader of services offering its trading services to several service providers or consumers.

19. **Assessment mechanisms (S.1, S.3 and S.4):** The assessment mechanisms of an abstract architecture 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 [29]. For instance, the most usual assessment mechanism in service trading is utility functions.

20. **Forms of expressing information and preferences (S.1, S.2, S.3 and S.4):** The preferences and the information managed about the service and the parties can be expressed in different ways. Each abstract architecture may have their own way to express them, however, the most commonly used are to express them as constraints or as rules.

### 2.2.2 Non-functional properties

The non-functional properties are a consequence of the functional properties and the design decisions made in the abstract architecture. Unlike the functional properties, in most cases it tends to be 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 2.2.1.

- **Expressiveness** The expressiveness of an abstract architecture is related to the things that can be stated in the agreements, proposals or agreement preferences (assessment mechanisms, user preferences), managed by
the architecture. The more expressive an architecture 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 abstract architecture measures the ability of the architecture to be used in a complex real-world scenario. Therefore, an abstract architecture with a high industry-orientation must have elements to create reliable agreements, support non-cooperative agreement creation, must have different agreement creation mechanisms and support different trading protocols 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 abstract architecture 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 trading protocols and agreement creation mechanisms.

- **Interoperability** The interoperability of an abstract architecture is related to the ability of the architecture to operate in a heterogeneous environment interacting with other architectures that may be implemented using different technologies and languages to express the terms of the agreements and preferences. The interoperability property is related to functional properties such as different trading protocols, agreement creation mechanisms and knowledge adaptation.

### 2.3 Analysis of service trading architectures

Our goal is to apply the set of properties defined in the previous section to the most relevant abstract architectures. In this context, we understand an abstract architecture as a specification that defines a set of elements (subsystems, components, interfaces, data types, or collaborations) for service trading and that can be applied for different domains and implemented with different technologies. Therefore, it is not the goal of this paper to analyse concrete architectures such as CREMONA [17].

The four abstract architectures analysed are the Open Grid Services Architecture [11] (OGSA), the Semantic Web Services Architecture [2] (SWSA), the Web Services Modelling Ontology Full [22] (WSMO-Full) and the Adaptive Services Grid reference architecture [14] (ASG). Following, for each architecture,
each property is commented and, as a summary, table 2.1 shows the comparative conclusions raised up from the analysis.

Open Grid Services Architecture is an abstract architecture for Grid systems and applications. The analysis of OGSA is based in [11] and other specifications developed by GGF (Global Grid Forum), which detail some aspects not fully described in that document, such as WS-Agreement [1]. The discovery employed is external and it is carried out by the so-called information services. The architecture is symmetric for service consumer and provider as both of them may act as agreement initiators in WS-Agreement. There is no specific element to deal with knowledge adaptation during discovery, although the use of semantic-enabled discovery services could solve that problem. Concerning the market observation, it is achieved by using a subscription mechanism specified in the Grid Monitoring Architecture. Like discovery, both the information query and the third party information is developed by using the information services. The agreement creation mechanism employed in OGSA is the WS-Agreement protocol, although a negotiation protocol is also being developed and agreements for composed services can be created by using the Execution Planning Services. Concerning the deployment, OGSA is conceived to be deployed as independent services that are later used by higher-level applications. Finally, elements that support the decision-making such as the creation of a world model and the types of information managed about the parties together with the assessment mechanisms and the forms of expressing information and preferences are not in the scope of the architecture. This is also the case of the proposals preselection, although Candidate Set Generator could develop that function.

Semantic Web Service Architecture describes an abstract reference architecture for semantic web service interoperability. In this architecture, the discovery issues are addressed from the perspective of semantic registries (Matchmakers). The knowledge management is a key point in this architecture expressed in the specification of different ontologies. In this context, despite the idea of market can be induced from this architecture, there is not an explicit element that actively reacts to different changes in the market (Market observation). This architecture is not symmetric due it is highly focused in the organisation that acts as service consumer and leaves the service provider as a comparatively simple systems that remain passive during the service trading process. The information query mechanism can be developed during the engagement phase in the contract preliminaries interaction. However, there is not specified an interaction with third parties. The interaction mechanisms related with agreement creation are based on abstract protocols described in FIPA Conversational Language. There is an explicit requirement for non-repudiation mechanisms during the enactment phase. Finally, though it is not specifically stated, this architecture is oriented toward non-cooperative.
<table>
<thead>
<tr>
<th></th>
<th>OGSA</th>
<th>SWSA</th>
<th>WSMO-Full</th>
<th>ASG</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Yes (distributed)</td>
<td>No</td>
<td>Yes (ontologies)</td>
<td>No</td>
</tr>
<tr>
<td>(2)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(3)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(4)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(5)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(6)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(7)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(8)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(9)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(10)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(11)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(12)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(13)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(14)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(15)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(16)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(17)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(18)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(19)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(20)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison of abstract architectures
Web Services Modelling Ontology Full presents an abstract conceptual architecture for semantic web services and it is oriented to cross-organisational scenarios. It uses an external discovery based in the Web Service Architecture and it is symmetric for consumer and provider. It also supports knowledge adaptation by using semantic-based service descriptions. However, there is not explicit information about how to carry out a market observation, the information query nor third party information. Regarding the information managed about the parties, it uses service and trading information (e.g. supported choreographies) but it is not stated whether it can use information related to the parties. Like OGSA, the mechanisms to support decision-making are out of the scope of WSMO-Full. Therefore, neither the world model, the capacity estimator nor the assessment mechanisms are covered. The agreement creation mechanisms supported are specified through the so called contract agreement choreographies. WSMO-Full includes partial support for decommitment in the post-agreement choreography but the mechanism is not fully defined. It also partially supports trading protocols through contract agreement and post-agreement choreographies but it does not consider the specification of temporal constraints on them. However, WSMO-Full does not include any support for complex service trading elements such as a notary or the creation of agreements for composed services. The architecture seems conceived to operate in a non-cooperative agreement creation, although there is no explicit limitation in using it in a cooperative environment. Finally, as it is a conceptual architecture, it does not consider any deployment options.

Adaptive Service Grid This reference architecture has been developed as an intent to create service providers that quickly adapt to business changes. In particular, the main goal is to achieve an efficient way of composing services to create more complex services with an added value. In the case of symmetric property, the elements that implement the provider-part and consumer-part of the system in ASG are not symmetric. The discovery is handled by the so called DDBQuery in a centred way through a semantic registry (with reasoning capabilities). In this way, though different ontologies handling are considered as part of the registry there is not an explicit market that is observed. Concerning the world model, this architecture specifies an element called ServiceProfiling that stores information about historic interactions with providers creating a relative model of the provider that is taken into account for the optimisation of the negotiation and selection of services. The information managed about parties is highly oriented to service; neither provider nor trading information are described in any of the processes. This approach leaves open the specific negotiation protocol used to establish the SLA for each service composed. However, WS-Agreement standard is specified as an implementation option. ASG can be applied to either cooperative or non-cooperative scenarios. Despite ASG describes the architecture of a composed services provider, from an architectural point of view this case is chiefly service consumer oriented because it just looks for atomic service providers to be composed. In ASG, the deployment
possibilities are specified in terms of different development technologies and by identifying subsets of elements that are mandatory and other that can be optional.
Chapter 3

An abstract architecture for service trading

3.1 Introduction and motivation

The service trading process is the process of locating, selecting, negotiating, and creating SLAs. Several abstract architectures for service trading have been proposed such as the Open Grid Services Architecture [11] (OGSA), the Semantic Web Services Architecture [2] (SWSA), the Web Services Modelling Ontology Full [22] and the Adaptive Services Grid reference architecture [14]. In this context, we understand an abstract architecture as a specification that defines a set of elements (subsystems, components, interfaces, data types, or collaborations) and that can be applied for different domains and implemented with different technologies. In Section 2.3 a detailed analysis of these abstract architectures is developed. From this analysis, several conclusions can be extracted:

1. The discovery process is well supported and most abstract architectures provide knowledge adaptation.

2. Most abstract architectures do not cover elements to support the decision-making. For instance, both the world model and the assessment mechanisms are out of scope of the majority of the abstract architectures analysed.

3. There is little support for the most advanced features of service trading such as the notary, the decommitment from established agreements and the trading protocols.

Due to these lacks, some complex service trading scenarios cannot be completely achieved and the automation of the process is somewhat limited because of the lack of elements to support the decision-making. Therefore, it is convenient to develop a new abstract architecture that solves these lacks enabling these complex service trading scenarios to be addressed successfully.
The abstract architecture that have been developed takes the properties described in Section 2.2 as a starting point and defines the elements that are necessary to accomplish them. The input to the abstract architecture (see Figure 3.1) are 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*. The output of the abstract 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 abstract 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 actively finding new parties to make agreements with them and receiving and 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 Modelling elements

Before detailing more accurately the different parts of the architecture, it is convenient to define and clarify the modelling 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 [32], where organisations are outlined developing a general architectural characterization 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 abstract architecture is composed of six organisations: trading, discovery, information, selection, agreement making and binding, that interact to accomplish the service trading 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 artefacts 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.

---

1 In MAS, these software artefacts are usually agents
Figure 3.1: Abstract 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 abstract 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 [32]. 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 Organisations behaviour overview

As an introduction to the abstract architecture, we can sketch out the global behaviour of organisations as following:

1. The main goal of the trading organisation is to specify a choreography that will regulate how the whole process is carried out, that is, it cares of setting the timing for when to start the search for parties, when to submit offers, when to wait for responses, when a negotiation must start or when a binding offer can be sent. Additionally, the trading organisation monitors the market (making use of the discovery organisation) in order to decide when the agreement search should be started.

2. The discovery organisation performs the process of locating a set of potential providers or consumers according to a number of functional and non functional requirements;
3. candidates discovered, are then passed to the information organisation in order to gather detailed information about the characteristics and preferences of each potential party.

4. This information is subsequently used by the selection organisation to create and select a set of promising agreement proposals with other parties. The selected proposals are also analysed to decide whether we would start a negotiation process with other parties or produce a take-it-or-leave-it offer.

5. These instructions are delegated to the agreement making organisation responsible to actually negotiate or propose the agreement to other party and to create and sign the final agreement.

6. During this procedure, the agreement making organisation interacts with the binding organisation by asking for approval to make or accept binding offers. In so doing, the responsibility of the binding organisation is to determine when an offer may be accepted.

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 3.4 to 3.3, we deeply describe all six organisations that are part of the abstract architecture as well as the collaborations and roles that compose them.
3.2 Data structures

Agreement  An agreement defines a dynamically-established and dynamically-managed relationship between parties [1]. 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, in our abstract architecture, we only deal with two-party agreements.

- A collection of terms that must describe both functional descriptions and non-functional guarantees of the service. 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.

Term  Terms define the content of an agreement. The terms can be applied to both the service consumer and the service provider. A term mainly specify some functional or non-functional guarantee of the service that must be observed.
by the parties during its execution. For instance, a term can express that the service provider guarantees that the response time will be less than 20 ms. and another term can express that the service consumer will pay 1 euro for each service execution and that will not execute the service more than 10 times per minute. However, a term can also express other aspects of an agreement such as termination clauses. 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 observe the constraints specified in the term. This element is optional.

**Compensation** A compensation is part of a term of an agreement and represents the penalty that a party suffers if it does not fulfil the constraints expressed in that term.

**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 abstract architecture, a proposal is composed of:

- The *parties* the proposal is about.

- A set of *terms*, similar to those specified in the *agreement*, to give a functional description, non-functional guarantees and other agreement-related 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.

---

2 This is equivalent to the parties in the *agreement*
• A collection of *variation points* referring to some *terms* specified in the proposal. These variation points can be used to indicate acceptable variations or information about the negotiability of the terms specified in a *proposal*.

• 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*.

**Agreement preferences** In the agreement preferences are expressed the set of data that it is used to assure that user needs are correctly dealt among the trading process.

These preferences comprise the initial data provided by the user (such as the requirements/features of the service demanded/offered) as well as the dynamic information created to guide the process (i.e. the *trading orchestration*).

This structure comprises:

• A set of *features* that express the capabilities and/or characteristics of 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 *requirements* that describes the needs of the user in terms of the wished features of counter-parties to be searched. This requirements, can be set to the service or the party itself.

• A mean to evaluate and compare potential agreement proposals: an *Assessment Mechanism*.
• A trading orchestration that will coordinate the temporal behaviour of the different roles to correctly achieve agreement goals. In this context, this element reference to a trading protocol statement.

Assessment mechanism This structure is used when a given element needs to compare two or more proposals from other parties; this mechanism, should provide the unique way of proposal evaluation that allow a common coherent ranking amongst elements of the abstract architecture when applied to the same group of proposals.

An example of this assessment mechanism would be a set of utility functions to be applied over the features expressed in the terms of a given proposal.

Statement This structure groups different types of expressions over a certain party or the trading process itself:

• On the one hand, statements are the basic building block used to construct the set of features and requirements of the party; in this way, statements would be rules or constraints.

• On the other hand, statements can describe the characteristics of a particular trading process creating what is called as trading protocol.

Additionally, statements can be group depending on the domain in which it is applied; the attribute subject specify statement domain, being possible the following major areas:

• 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.

• Trading Statement. They specify features about an specific trading process. In this context, it is worth pointing out that trading protocols (as well as its components) are kinds of statements that belong to this domain.

Each statement is linked to a set of languages that give semantics to the vocabulary used within the statement.

Language Languages are means to give semantics to the vocabulary used in the statements. Main aim of languages is to facilitate the interoperability amongst parties. Ontologies can be seen as an example of languages describing the relationship amongst concepts of a given semantic domain.
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 are means to 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.*

Trading Protocol  This structure express some parts (or the whole) of the trading process. To understand what a trading protocol is, we can focus on the following real-life example: A public bidding where an institution looks for a service provider and devises a trading protocol that consists of the following stages: the announce of the bidding, a deadline for the submission of proposals, a period of resolution and, finally, the communication of results. The trading protocol also states temporal constraints for each stages.

In an abstract level, the trading protocol is defined as a set of stages (e.g. advertisement, proposal submission, negotiation, resolution, etc...) cross-linked in accordance to some temporal constraints and bounded to some choreographies. These elements, are expressed in form of constraints:

- A set of potential choreographies for each stage. In some situations, this means a list of optional choreographies but, in other cases, a unique choreography can be fixed in a given stage. An example of this type of constraint could be *In the Negotiation stage, auction or bargaining protocols are allowed* (i.e. choreographies implementing these protocols).

- Temporal restrictions specify a set of constraints about the life-cycle of the trading process. This restrictions could vary from simple fixed temporal points (e.g. *Trading process should end by 14:00 of 14th, March*) or can be complex relationship amongst the durations of some stages (e.g. *Information stage should start in the middle of the Discovery stage*).

Usage of trading protocols is twofold:

- First, it is used to guide the temporal behaviour of the different elements of the abstract architecture. In this case, some specific choreography are fixed to describe the internal interactions of elements avoiding any ambiguity.

- Second, trading protocols are used to express the public part of the trading process that should be exchanged with other parties in order to check the temporal compatibility of the trading process; it is worth pointing out that this compatibility must be assured before any posterior interactions among parties.
**Information**  This structure groups the type of data exchanged among parties. As *agreement preference*, information can be composed of two different sets of statements: requirements and/or features.

**Market event**  Market events express occurrence in the market. Each event has to be adjusted *market performative* that indicates the type of event (such as *new service provider* or *new service consumer*) that have been triggered; this performatives must be agreed amongst the different parties *market performative* that indicates the type of event (such as *new service provider* or *new service consumer*) that have been triggered; this performatives must be agreed amongst the different parties using the market in order to guarantee the interoperability.

Additionally, each event must contain a set of information about the event that allow to its classification and routing to the interested parties. This information is matched against the preferences of the party to identify the potential counterparties to start a trading process with.

Examples of market events can be: A new supplier $S$ of service $X$ has appeared with the following features: ... or $Z$ is looking for a service $W$ fulfilling, at least, following requirements: ...

In this context, it is worth pointing out that the information about requirements or features expressed within the market event could be completed with a posterior phase of information exchange.

### 3.3 Trading

**Goal**  In general, service trading is a process whose details change from scenario to scenario depending on the type of parties involved and the temporal requirements to be met. In order to deal with these issues, it is necessary an orchestration of the different stages in the trading process.

The trading organisation focuses on the global behaviour from a temporal point of view. In so doing, its goal is the coordination of the remain organisations so as to develop a trading process coherently with the *trading orchestration* of the *agreement preferences*.

In a difference to other organisations, the life-cycle of all roles in the trading organisation comprise multiple trading process.

In its management, this organisation carry out the following activities:

- To decide when to start a search.
- To manage agreement preferences.
- To Construct/Manage the trading protocol.

**Requirements**  Building on these ideas, in order to address its issues, the trading organisation should deal with the following data requirements:

- A taxonomy of *trading protocols*
a mean to fulfil the implementations for each trading protocol

- the management of the life-cycle of the elements in the system, including a mechanism for the instantiation of the actors that implement the trading protocol

- definition of the temporal parameters that will control the behaviour of other organisational. These parameters would be passed to the actors of those organisations.

### 3.3.1 Roles

**Search Planner**

**Goal** This role is in charge of interacting with the environment in order to handle agreement preferences and assure a correct trading process that optimally address these preferences. Concretely, this role is the responsible for trigger the search based on market status and the previous knowledge of the market. Once a given search is about to start, this role make use of the Trading Protocol Manager that construct the trading orchestration that will guide the whole trading process. After this process, it is responsible to make that orchestration is followed by setting the appropriate data structure (the agreement preferences) and invoke the appropriate actors that would develop the process.

**Data input** This role uses mainly the agreement preferences repository environmental resource. Additionally, it receives information about the status of the market in order to decide when to start a search.
Data output This role modifies the agreement preferences repository in order to start a particular trading process.

Behaviour This role is focused on a continuous monitoring of the market in order to decide when a specific agreement preference should be started. In order to carry out that decision, following sources are taking into account:

- Market events, received from the Market Mediator.
- The market model inside of the world model. This information relates to the historic information about the market.

Trading Protocol Manager

Goal This role analyses agreement preferences to decide which of the known trading protocols best suits to the temporal constraints specified on the preferences.

Data input This role reads from the environment the agreement preferences and receives the information about when a particular search is started from the search planner.

Data output This role construct the trading protocol that shall act as the trading orchestration.

Behaviour Process of creating a trading protocol can be decomposed in three lines:

- Basis of the trading protocol is taking from the trading constraint expressed by user in the agreement preferences. Temporal variables expressed in this constraints should be closed (i.e. relative to a concrete temporal point).
- As a complement, trading protocol manager is in charge to enrich the set of constraints to conform the trading protocol; this enrichment process is developed with temporal variables that can be open if they depend on the moment of the search start.
- When search planner announces the start of a search process, trading protocol manager close all open temporal variables to conform the final trading orchestration that will guide the search.

The process specified can be carried out in different ways depending on the implementation of the role, we can give the following two examples:

- Trading protocol manager can maintain a repository trading protocol templates and, in each case, select the appropriate one according the constraints expressed in the agreement preferences.
- The process of creating the trading protocol can be dynamically based on specific algorithms.
3.3.2 Collaborations

The organisation can be decomposed in two different collaborations, as following:

TradingProtocolCreation

<table>
<thead>
<tr>
<th>Goal</th>
<th>Roles</th>
<th>Data exchanged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal of this collaboration is setting up of the <em>trading protocol</em> that will act as the <em>trading orchestration</em> of the process. In the <em>agreement preferences</em> are included some constraints that are used by the <em>trading protocol manager</em> as a basis to the construction of the trading process. <em>search planner</em> acts in this collaboration as the role that communicates the beginning of the search to the <em>trading protocol manager</em>.</td>
<td><em>TradingProtocolManager</em> and <em>SearchPlanner</em>.</td>
<td>In this collaboration, <em>search planner</em> communicates the temporal decision about the start of the trading process.</td>
</tr>
</tbody>
</table>
MarketObservation

| Goal | The goal of this collaboration is to inform about the market events that are happening. This collaboration starts with a subscription process where the search planner express its need of information in terms of the set of market events are potentially interesting. In response to the subscription preferences, the rest of the collaboration is carried out by the market mediator as the role that provides the different market events. |
| Roles | MarketMediator and SearchPlanner. |
| Data exchanged | Data exchanged during this collaboration is related subscription and market events. |

3.4 Discovery

In the discovery organisation the main aim consist in locating potential parties demanding (or supplying) a service that other party provides (or needs). In a complementary way, this organisation would be the responsible for access the market and propagate the events generated from the internal active trading processes in the system.

In this context, the market must be seen as an abstract concept that include the set of organisations looking for bussiness relationship as well as the infrastructure that acts as a facilitator for the communication amongst them.

Requirements  In order to deal with the issues related with discovery, we can identify the following information requirements that must be implemented by the roles of the organisation:

- An infrastructure for a taxonomy of services. Each demand or provision of service should be catalogued based on the taxonomy; the classifying criteria may include both functional and non-functional features.
- A method for registering new market events. An example of such an event would be the fact that an entity is searching for service providers of a given type (as specified in the taxonomy). In so doing, it is, therefore, necessary a protocol to exchange and propagate events amongst organisations composing the market.
- A way to store access points to the different actors that generate events joint with an addressing specification to provide a mechanism to access and identify actors.

3.4.1 Roles

In this organisation, we can find four roles:
• An special abstract role that represent market infrastructure: the discovery service
• A role that adapts the local data to the one in the market acting as a facilitator of the interactions with the market: the market mediator.
• For each trading process carried out in the system, two complementary roles appear: One acting as the active part looking for parties (the tracker) and one another passive role that announces the capabilities and needs of the party: the advertiser.

Discovery Service

Goal The Discovery Service role represents an abstraction of the discovery infrastructure that should be refined in further concrete models. Different infrastructures can be selected from a wide range of models: from a centralized paradigm to a distributed one.

Different ontologies can be used in this role. (that vary from the local ones handled by the market mediator. However, ontologies used in this context, represent the ones that must be shared amongst all the participants of the market.

Behaviour This role represents the infrastructure of discovery and, therefore, it highly depends on the nature of the implementation chosen.
Market Mediator

Goal The Market Mediator role is in charge of adapting local knowledge model in a given party to the appropriate discovery infrastructure. This adaptation make independent the characteristics of market modelled by the discovery service to the rest of organisations. In so doing, this role should store information about interesting occurred events in the past in order to communicate it to the local actors in the system when demanded.

It is important to remark, that this role remains active for multiples searches due it is mandatory to perform a persistent market monitoring. Additionally, it should communicate information to the world modeller in order to supply the information harvested to analyse and create a model of the market useful for decision making process.

Data input The information that receive this role is threefold:

- A set of subscriptions from other roles. This subscriptions specify means to know which kind of events they are interested on.
- Information about a particular search that is active in the system.
- Advertisement about a trading process that is active in the system.

Data output This role supplies market events to other roles in response to its subscriptions.

Behaviour The main activity of this role is the marshalling of the market events that occur in its collaboration with the discovery service. All events observed are registered and semantically processed in order to construct appropriate events based on the local ontologies used in the system.

As the market mediator start receiving subscriptions to groups of events it sends the appropriate events that match with the ones stored in the system. Additionally, it should also save and propagate events triggered in the system, such as a new role advertising information about a particular search

Tracker

Goal In a specific search, the Tracker role is in charge of performing an active search to locate potential counterparties (i.e. interesting parties to start an agreement search process with).

Data input This role reads from the agreement preferences to extract the requirements that identify interesting possible counterparties. Additionally, this role receives market events about the existence of those counterparties

Data output This role sends the counterparties found
**Behaviour** Tracker initiates its activity with a subscription of market events to the *market mediator*. In this subscription, this role specify a subset of information that is searching based on the *agreement preferences* of the user. This process would have as a result a set of related market events that match in some way with the subset of information sent. Finally, with this market events received, the tracker develop a set of promising counterparties to start a trading process with.

**Advertiser**

**Goal** The main aim of the *Advertiser* role is to publish certain capabilities and requirements from the *agreement preferences*. This process is carried out by using the *market mediator* as the facilitator for accessing the market through the creation of the appropriate *market events*.

**Data input** This role reads from the *agreement preferences* in order to select the most interesting information to be published in the trading process.

**Data output** As an output this role sends the information to be published; i.e. to be converted to market event and propagated amongst the market by the *market mediator*.

**Behaviour** This role has a single activity; when it is initialized, it selects the information from the *agreement preferences* and sends it to the *market mediator*.

### 3.4.2 Collaborations

Three collaborations compose this organisations, *MarketEventExchange*, *PotentialCounterpartySearch* and *Publication*. The first one is in charge of handle the link between the market and the system.

The rest, correspond with the internal collaborations amongst the roles in the information organization:

- *PotentialCounterpartySearch* collaboration consist on the interactions carried out by the *tracker* and the *market mediator* in order to actively search for potential counterparties.

- *Publication* refers to the sent of information by the *Advertiser* to the *market mediator* concerning the public information that should be advertised in the market (by the appropriate market events).
Figure 3.7: Discovery organisation - Collaborations diagram

**MarketEventExchange**

<table>
<thead>
<tr>
<th><strong>Goal</strong></th>
<th>This collaboration represents the set of interactions amongst parties that comprise the market. The main goal of this collaboration is to make known a particular event that have happened in certain system and that can be interesting for a second one.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Roles</strong></td>
<td>Roles in this interaction include: the <em>market mediator</em> and the <em>discovery service</em>.</td>
</tr>
<tr>
<td><strong>Data exchanged</strong></td>
<td>The data exchanged in this collaboration is strongly dependant on the market infrastructure used and, therefore, market-specific. From an abstract point of view, this information should communicate the news about important events happened in the market such as a new provider or consumer of a particular service has entered into the market.</td>
</tr>
</tbody>
</table>
**PotentialCounterpartySearch**

| **Goal** | Main goal of this collaboration is to perform an active search for promising parties that could complement our needs. In so doing, the result of this collaboration is a set of end-points to potential counterparties that we can start an agreement search process with. |
| **Roles** | Two roles participate in this collaboration: the tracker and the market mediator. |
| **Data exchanged** | In this collaboration the data handled correspond to the information harvested from the market (in the form of market events); concretely, from this events, some important information can be found about promising counterparties such as the its end-points or the advertisement information that they have published. In order to get this information, a subscription process is carried out by the tracker to the market mediator expressing the matching information over the market events that it handle. This matching information would consist in a subset of the agreement preferences. |
| **Examples** | Process of search can be carried out in different ways, from a single query service to a iterative search process:

- On the one hand, we can have a single interaction from de tracker to the market mediator where all data about search preferences is expressed and, as a result, a set of the related market events is transmitted.

- On the other hand, an iterative search where tracker can express progressive restrictions as part of its subscription with the market mediator. This would mean a refinement of the search by reducing the possible set of potential market events to be matched. |
Publication

| Goal | The main goal of this collaboration is to communicate the information that should be advertised in the market (through the market mediator). This information to be published would consist in the user needs (or capabilities). An example of minimal advertisement information would be the type and the functional characteristics of a service needed (or offered). |
| Roles | The market mediator and the advertiser interact in this collaboration. |
| Data exchanged | The data exchanged in this collaboration, correspond to the subset of the agreement preferences that must be published in order to make that advertisement in the market. |

3.5 Information

Goal  The goal of this organisation is to manage information about potential counterparties. These potential counterparties may have been found following a discovery process by the discovery organisation 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. In addition, it is convenient that an integration of the service features format with the taxonomy of the discovery layer was provided.

• 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.

3.5.1 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.

Inquirer

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

Data input The role receives a collection of potential 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 potential counterparty for.
Figure 3.8: Information organisation - Roles and data dependencies
**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 the responsible for publishing all the 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 other roles of the abstract architecture 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 Modeller**

**Goal** The goal of the *world modeller* 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 modeller have been applied.

**Behaviour** The world modeller 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.

### 3.5.2 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 modeller 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.
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
Tracker, Inquirer, Informant, and Proposal collector.

Dependencies
The collaboration can be decomposed in three more collaborations, namely, the PotentialCounterpartiesNotification (that obtains a list of potential counterparties to query), 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.

Finally, the PotentialCounterpartiesNotification and the CounterpartyInformationRequest depends on the PotentialCounterpartySearch (from the discovery organisation) and the ProposalReception (from the selection organisation) collaborations respectively because these collaborations its suppliers of information.
InformationExchange

<table>
<thead>
<tr>
<th>Goal</th>
<th>The aim of the collaboration is to query a potential counterparty to obtain information about it.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td>Inquirer and Informant</td>
</tr>
<tr>
<td>Data exchanged</td>
<td>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.</td>
</tr>
</tbody>
</table>

WorldInformation

<table>
<thead>
<tr>
<th>Goal</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td>World modeller, Third party informant, Inquirer, Market mediator (from Discovery), Proposal collector (from Selection), and Agreement maker (from Agreement Making).</td>
</tr>
</tbody>
</table>
| Dependencies | The collaboration is composed of five collaborations. These collaborations represent the interaction between the world modeller and each one of the other five 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**

<table>
<thead>
<tr>
<th>Goal</th>
<th>The goal of the collaboration is to query information about potential counterparties to a <em>third party informant</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td><em>World modeller</em> and <em>Third party informant</em>.</td>
</tr>
<tr>
<td>Data exchanged</td>
<td>The data used in this collaboration is expected to be dependent on the <em>third party informant</em> that is being queried, and, therefore, domain-specific.</td>
</tr>
</tbody>
</table>

**InformationSupply**

<table>
<thead>
<tr>
<th>Goal</th>
<th>The goal of these collaborations is to provide the <em>world modeller</em> with the results of the interaction of roles of the abstract architecture with potential counterparties.</th>
</tr>
</thead>
</table>
| Roles                 | • *InquirerInformationSupply*: *World modeller* and *Inquirer*  
• *CollectorInformationSupply*: *World modeller* and *Proposal collector*  
• *AgreementMakerInformationSupply*: *World modeller* and *Agreement maker*  
• *MarketMediatorInformationSupply*: *World modeller* and *Market mediator* |
| 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* and *CollectorInformationSupply* collaborations, the data used is a *proposal*; in the *MarketMediatorInformationSupply* the data is a *market event*, and in the *InquirerInformationSupply*, the information obtained directly from the potential counterparty. |

### 3.6 Selection

From an abstract point of view, the main aim of the selection organisation is to choose a set of candidate parties with whom a negotiation process can be started or to whom an agreement proposal can be submitted.

In order to deal with its goal, in this organization, a handling over proposals is carried out. For each one of them, a selection, filtering and categorization is
developed. From a behavioural point of view, main activities of this organisations are twofold:

- Create the best possible proposals to sent to other counter-parties
- Select and decide how to handle proposals in the system.

The selection starts with a set of information about potential parties coming from several sources:

- information provided by the information organisation after an active search
- agreement proposals received from other parties
- non-successful offers\(^3\) coming from the binding organisation, so that they can be processed again.

### 3.6.1 Roles

#### Proposal builder

**Goal** The main goal of this role is to create the best possible proposal for a counter-party, based on the known information about that party and the agreement preferences of the user.

**Data input** In addition to the environmental resources it access (i.e., the agreement preferences and the world model), this role receives information about counter-parties demands/offers.

\(^3\)These offers are non-successful because either they were not good enough for us or the other party rejected them.
Data output  A set of proposal is generated by this role

Behaviour  The *Proposal builder* role creates agreement proposals based on the information gathered by the information organisation. Then it sends these agreements proposals to the *Proposal collector*.

Proposal collector

Goal  This role has the responsibility of creating a unique point where all proposals are centralized for its processing. In this way, role activities are highly linked with work-load regulation. Additionally, this role should

Data input  This role receives proposals either from internal roles of the system or from external roles of other systems.

Data output  The output of this role is a set of proposals

Behaviour  The *Proposal collector* receives the agreement proposals generated by the *Proposal builder* as well as the agreement proposals coming from other parties through the *PropONENT* role and submits them to the *Proposal filter*. When an external proposal is received, this role can decide to ask for specific information to the information organization. Additionally, it can develop some work-load issues with proposals; e.g. kept them until an event occurs: they can be collected until the negotiation phase finishes.

Proponent

Goal  The *PropONENT* role represents the party that actually sends the proposal to an external organization.

Data input  This role receives a set of proposals to be sent.

Data output  As a result of its activity, one or more proposals are proposed to other organizations

Behaviour  After receiving a set of proposals to be sent to other counterparties. This role communicates with each of the *Proposal collector* in the counter-parties in order to send an agreement proposal.

Proposal filter

Goal  Main goal of this role is to process the set of potential proposals in order to filter and rank them.

Data input  This role receives as input, the set of proposals to be filtered. Additionally, it should read the *agreement preferences* and the *world model* in order to perform its operations.
Data output After the processing of the role, a set of proposals is given as output. Unlike the input set, this set is ranked and filtered ready to be dispatched.

Behaviour The Proposal filter role is in charge of filtering the agreement proposals collected by the Proposal collector and the non-successful offers coming from the binding organisation. The filter criteria are not unique but, in most cases, they depend on the preferences given by the user, the counter-party model (as part of the world model) and the status of the whole service trading process. In this process, each proposal is evaluated in order to obtain its utility for the user. In so doing, a classification is made based on this utility creating a ranking that can be used in further stages of the trading process; particularly in the process of dispatching them to the agreement makers. After the process, several proposals are rejected and the others (with its ranking) are sent to the Proposal dispatcher role.

Proposal dispatcher

Goal This role, is in charge of deciding for each of the promising proposals which Agreement Maker is most appropriate to handle it. One system may have several Agreement Makers with different characteristics and one of them may be better than the others for certain conditions. For instance one Agreement Maker can implement auction protocols, another one can implement bilateral negotiation protocols, and another one can implement just a take-it-or-leave-it protocol. The proposal dispatcher is the only role of the selection organisation that knows the features of each of the agreement maker and, therefore can assign each proposal to one of them.

Data input The information received by this role consist of different proposals with a ranking information amongst them. Additionally, this role must have a mean to know the processing capabilities of each of the agreement makers.

Data output This role send a set of proposals to each of the agreement makers in the system in order to be processed.

Behaviour After receiving, a ranking of filtered proposals, this role performs a decision The Proposal dispatcher sends the proposal to the .

Reasons to assign a proposal to a particular agreement maker vary depending different aspects; as an example, we can describe the following:

• The case can correspond to a negotiable proposal over some terms and the counter-party has specified a concrete negotiation protocol. In this context, the proposal should be redirected over an agreement maker in our organisation that could handle that specific protocol.
Proposal is not acceptable as is but it can be interesting after a negotiation process. This proposal can be the basis of the process and should be sent to an agreement maker with capabilities to organize such a process.

Proposal has been created by our organization and is very promising; in this way, it shall be dispatched to an agreement maker that will promote it to the counter-party (through our proponent).

3.6.2 Collaborations

This organisation is composed by four principal collaborations that jointly perform the proposal selection process previous to the agreement-making. This process can be divided in three stages as following:

- The construction and reception of the proposals. The former refers to the proposal created by the system to be sent to external systems and is carried out by the ProposalConstruction. The latter correspond to the collaboration ProposalReception that involves the proposal received from a an external system. This two processes are carried out in parallel.

- After the fist stage of proposal collection, a further stage of filtering and ranking of the proposals is done based on the information and preferences. This process is developed by the ProposalSelection collaboration.

- Finally, for each of the previously selected and categorized proposals, an assignment process is fulfilled as part of the ProposalAssignment collaboration.

In this sequenced process, we can foresee a set of dependences between those collaborations, concretely:

- On the one hand, the ProposalAssignment would depend on the ProposalSelection

- On the other hand a double dependence from ProposalSelection to ProposalConstruction and ProposalReception.

Additionally, some dependences appear from an inter-organisational point of view:

- The ProposalConstruction collaboration, would strongly depend on the knowledge harvested in the information organization during the InformationExchange collaboration.
Figure 3.11: Selection organisation - Collaborations diagram
### ProposalConstruction

| **Goal** | Main goal of this collaboration is the building the most interesting proposals for a counter-party based on the actual knowledge that we could have about it and our agreement preferences |
| **Roles** | Roles in this collaboration, include: The proposal builder, the proposal collector and the inquirer |
| **Data exchanged** | In this collaboration, main data involved are the constructed proposals joint with the information needed to build it. |
| **Dependencies** | This collaboration is composed of two subcollaborations:  
- On the one hand, the PotentialCounterpartiesInformation collaboration is developed between the inquirer and the proposal builder. In this subcollaboration, the information about the service and trading of the counter-party is supplied in order to create the best possible proposal coherent with this information.  
- Depending on the previous, the builtProposalSending subcollaboration correspond to the proposal sending performed by the proposal builder to the proposal collector. |

### ProposalReception

| **Goal** | From one point of view, this collaboration involves the interactions with other counter-parties in which they communicate their proposals. In so doing, main goal of this collaboration is the reception of the different counter-parties interested in start an agreement search process with us. However, it is worth pointing out that this collaboration also can be seen from the opposite point of view where our system acts as the proponent sending proposals to external proposal collectors. |
| **Roles** | Two roles participate in this collaboration: the proponent and the proposal collector |
| **Data exchanged** | The information exchanged during this collaboration is, mainly, agreement proposals |
**ProposalSelection**

| **Goal** | This collaboration is in the core of the organization; it represent the abstract process of selection amongst all the possible proposals collected in the system. After this collaboration, the proposals shall be filtered and categorized ready to be sent to the appropriate agreement maker that will process them |
|**Roles** | Roles in this collaboration include: The proposal collector, the proposal filter, the commit handler and the proposal dispatcher |
|**Data exchanged** | Data exchanged during this collaboration are sets of proposals. |
|**Dependencies** | Three subcollaborations compose the proposal selection. The SendCollectedProposals consist in the supplier of proposals from the proposal collector to the proposal filter. The ProposalReselection subcollaboration correspond to the sent of proposals from the commit handler. Concretely, after a decision making process in which the proposals have been rejected but there is a possible further handling such as a negotiation process. Finally, the SendFilteredProposals is the subcollaboration in which the proposal filter role communicates the processed proposals (i.e. filtered and/or categorized) to the proposal dispatcher role. In this context, there is a partial dependence from SendFilteredProposals to the ProposalReselection and the SendCollectedProposals. This dependence is based on the fact that proposals sent to the proposal filter during both subcollaborations are equally handled and do not differ from the SendFilteredProposals subcollaboration point of view. |
|**Examples** | The process carried out in this collaboration can be developed in different ways; as an example, we can describe the following two: A grouped sequential procedure in which the subcollaborations SendCollectedProposals, and SendFilteredProposals are developed with sets of proposals sent in block from one element to another. Alternatively, an individual sequential processing for each proposal can be done. In this case, multiple processing of proposals could be made in parallel at the same time. Concerning the ProposalReselection subcollaboration, it should always be developed in a coherent way with the SendCollectedProposals procedure due they provide the same kind of information (potential proposals). |
Proposal Assignment

<table>
<thead>
<tr>
<th>Goal</th>
<th>This collaboration represent the set of interactions that allow a dispatch of a set of selected proposals to one agreement makers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td>In this collaboration participate the proposal dispatcher role and the agreement maker role</td>
</tr>
<tr>
<td>Data exchanged</td>
<td>In this collaboration a set of proposals is sent</td>
</tr>
</tbody>
</table>

3.7 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. Quizá se podría refinar un poco la definición

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 [33]
• a decision on which response shall be sent to the counterparty
• a construction of a counterproposal if necessary [6]

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.

3.7.1 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. There are two kinds of initial proposals and the behaviour of the agreement maker is different in each case:

- If it is a proposal that we have received from the counterparty, the behaviour of the agreement maker may change depending on the complexity of the protocol it uses to create agreements. Thus, this role can act almost as a proxy if it just implements a take-it-or-leave-it agreement creation protocol, or it can be very complex if it understand several negotiation protocols and has to create bids or counterproposals.

- If it is a proposal that has been created by our proposal builder role, the proposal is sent to the proponent role in the selection organisation to be submitted to the counterparty and wait for the counterparty’s agreement maker to contact us. Then, the behaviour of the agreement maker is the same than in the previous case.

The rationale behind this different treatment of the proposals is that we distinguish between the first proposal to one counterparty, when, for instance, the agreement creation protocol is chosen, and the later exchange of proposals between the parties that form part of the negotiation process.

If the protocol used to create agreements is complex, 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 interaction 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.

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 antagonist 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 possibly 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 between 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 guarantee the reliability and non-repudiability of the process. However, it must include, at least, the submission of the signed agreement by both the *agreement maker* and the *counterparty’s agreement maker* and a notification that the agreement has been verified by the *notary*.

### 3.7.2 Collaborations

The organisation is composed of three collaborations, namely *ProposalDelivery* (send a proposal to the counterparty via the *proponent*), *AgreementNegotiation* (negotiate the proposals with the *counterparty’s agreement makers*) and *AgreementCreation* (create and sign the actual agreements). Both the *ProposalDelivery* and the *AgreementNegotiation* collaborations depend on the *ProposalAssignment collaboration* from the selection organisation, which supplies the proposals that needs to be either negotiated or proposed.

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*.

#### ProposalDelivery

<table>
<thead>
<tr>
<th><strong>Goal</strong></th>
<th>The goal of the collaboration is to submit an initial proposal that has been created by our <em>proposal builder</em> role to the counterparty via the <em>proponent</em> role. The rationale behind this different treatment of the proposals is that we distinguish between the first proposal to one counterparty, when, for instance, the agreement creation protocol is chosen, and the later exchange of proposals between the parties that form part of the negotiation process.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Roles</strong></td>
<td><em>Proponent</em> and <em>agreement maker</em></td>
</tr>
<tr>
<td><strong>Data exchanged</strong></td>
<td>The data used is a <em>proposal</em></td>
</tr>
</tbody>
</table>

58
AgreementNegotiation

<table>
<thead>
<tr>
<th><strong>Goal</strong></th>
<th>The goal of the collaboration is to reach an agreement between the involved parties by exchanging proposals.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Roles</strong></td>
<td><em>Agreement maker</em> and <em>counterparty’s agreement maker</em></td>
</tr>
<tr>
<td><strong>Data exchanged</strong></td>
<td>The data used in this collaboration is a <em>proposal</em>.</td>
</tr>
<tr>
<td><strong>Examples</strong></td>
<td>The protocols to create agreements can range from a very simple form of communication such as the submission of a proposal by one party and its acceptance or rejection by the other one, to a more complex form based on negotiation protocols. A negotiation protocol establishes the rules that govern the negotiation and the way the communication amongst the parties involved in the negotiation is carried out. The most common negotiation protocols are based on the submission of offers and can be categorised into auctions [27] (e.g. English, Dutch or Vickrey) and bilateral negotiations. Bilateral negotiations involve the exchanging offers and counteroffers between the two parties carrying out the negotiation [26].</td>
</tr>
</tbody>
</table>

4Note 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.
## AgreementCreation

<table>
<thead>
<tr>
<th><strong>Goal</strong></th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Roles</strong></td>
<td>Agreement maker, counterparty’s agreement maker, and notary.</td>
</tr>
<tr>
<td><strong>Data exchanged</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Dependencies</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

### 3.8 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 decisions made in this organisation are based on several factors and they 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 four groups:

- **First**, preferences about the contents of the agreement. For instance, constraints on the values of the terms of the agreement or an utility function indicating the importance of these terms to the user.

- **Second**, preferences about the party we make an agreement with. For instance, 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.

- **Third**, preferences about the trading process. Some examples are the deadline and the eagerness to reach an agreement.

- **Fourth**, 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.

Therefore, 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.
3.8.1 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 decisions 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.
3.8.2 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 AgreementNegotiation collaboration because the proposals that have to be approved by the ApprovalRequest are constructed during the negotiation process carried out in the AgreementNegotiation collaboration.
ApprovalRequest

<table>
<thead>
<tr>
<th>Goal</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td>Commit handler and agreement maker</td>
</tr>
<tr>
<td>Data exchanged</td>
<td>The data used in the collaboration is the proposal asked for approval.</td>
</tr>
<tr>
<td>Dependencies</td>
<td>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.</td>
</tr>
</tbody>
</table>

CapacityQuery

<table>
<thead>
<tr>
<th>Goal</th>
<th>The aim of this collaboration is to get specialised advise about the approval of a specific proposal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td>CapacityPlanner and CommitHandler.</td>
</tr>
<tr>
<td>Data exchanged</td>
<td>The data used in this collaboration is the proposal and a recommendation about the feasibility of its approval.</td>
</tr>
</tbody>
</table>

3.9 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 abstract architecture. The Table 3.1 shows a distribution of standards over the conceptual organisations that have been identified.

Discovery Concerning the discovery organisation, there are three specifications that can be used to implement its requirements:

1. UDDI can be used as a flexible repository to store the access points of elements and the taxonomies used by the discovery organisation.
2. WS-Notification can be used to subscribe and broker notification events.
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 [30] and WS-InspectionLanguage. Alternatively, WS-Agreement [1] 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 [1]. WS-Agreement is being developed by the GRAAP 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 [9], the FIPA English Auction Interaction Protocol Specification [8] or the FIPA Dutch Auction Interaction Protocol Specification [7].

Trading

Concerning the trading organisation, depending on the complexity of the trading protocol used, different approaches are possible. For complex coordinations, there are workflow standard such as BPEL [30] or choreography

<table>
<thead>
<tr>
<th>Discovery</th>
<th>Information</th>
<th>Agreement Making</th>
<th>Trading</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDDI</td>
<td>WS-MetadataExchange</td>
<td>WS-Agreement</td>
<td>WS-CDL</td>
</tr>
<tr>
<td>WS-Notification</td>
<td>WS-InspectionLanguage</td>
<td>FIPA Protocols</td>
<td>BPEL</td>
</tr>
<tr>
<td>WS-Addressing</td>
<td>WS-Agreement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Related standards

3. WS-Addressing provides an specification of the references/locations of web services by means of a standardization of the concept of endpoint references.
languages such as WS-CDL. In the case of simple cases, an alternative to implement Trading Protocols would be the specification of ad-hoc elements in the concrete architecture build upon the conceptual framework.
Chapter 4

Conclusions

4.1 Analysis of the abstract architecture

In this section, our goal is to develop an analysis of the abstract architecture proposed in Chapter 3. To achieve this goal, we check it against the service trading properties in a similar way to the analysis carried out in Section 2.3 with the other abstract architectures.

1. *External discovery*: The external discovery is carried out by the discovery organisation. Specifically, the *Discovery Service* and the *Market mediator* are in charge of developing it.

2. *Knowledge adaptation*: The knowledge adaptation is developed by the *Market mediator* (see Section 3.4.1).

3. *Market observation* The *Market mediator* is in charge of observing the market and notifying the changes to the *Search planner* (see Section 3.4.2).

4. *Symmetric architecture for providers and consumers* Since the beginning the abstract architecture has been developed to be symmetric for providers and consumers and no distinction between them has been done.

5. *Information query* The information query is carried out by the *Inquirer* and the *Informant* roles of the information organisation (see Section 3.5.1).

6. *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 3.5.1).

7. *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 3.5.2).
8. **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.

9. **Proposals preselection** The goal of the selection organisation is to make a preselection of proposals before starting an agreement creation process or a negotiation.

10. **Agreement creation mechanisms** The abstract architecture supports multiple agreement creation mechanisms by changing the implementation of the Agreement maker role. In order to support several agreement creation mechanisms simultaneously, a Proposal dispatcher has been included. (see Sections 3.6 and 3.7).

11. **Notary** An explicit Notary has been included as a role in the agreement making organisation (see Section 3.7.1).

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

13. **Capacity estimator** 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 3.8).

14. **Trading protocols** There is a full support for trading protocols in the abstract architecture. It is included in the data structures and the trading organisation is in charge of selecting and instantiating the most appropriate trading protocol in each moment.

15. **Creation of agreements for composed services** Currently, there is no support for creating agreements for composed services.

16. **Cooperative or non-cooperative agreement creation** The abstract architecture can be applied successfully to both a cooperative and non-cooperative scenarios.

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

18. **Deployment options** The abstract architecture has been specified following a role-based approach. Therefore, there are a high variety of deployment options that can be used only by changing the structural organisation of the agents that implement the roles defined.

19. **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).
20. 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).

In Table 4.1, our abstract architecture is compared with the other most significant abstract architectures for service trading. As it can be seen, our abstract architecture covers the majority of the service trading properties (excepting decommitment and composed web services). Therefore, it is well suited for complex service trading scenarios. Furthermore, we must note that, unlike other abstract architectures, the elements that are necessary to carry out an advanced automated decision-making such as the world model or the assessment mechanisms are fully covered by the architecture.

4.2 Analysis of previously set goals

In the Chapter 1, we set four 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. Identify the main characteristics of the service trading process.

   We believe that the properties of service trading defined in Section 2.2 fulfills this goal. In that section, we take a variety of service trading scenarios described in Section 2.1 as a starting point and analyse them in order to extract the properties that may appear in a service trading process. These properties can be used in two ways. First, they can be used to study the complexity of a service trading scenario by identifying the properties required by the scenario. Generally, the more properties required, the more complex is the scenario. Second, the properties can be used to compare several different approaches to implement a service trading process.

2. Analysis of abstract architectures for service trading

   We believe that this goal is accomplished in Section 2.3. In that section, we use the service trading properties identified in the previous section as a conceptual framework that enables the analysis and comparison of several different abstract architectures for service trading. We select the most significant abstract architectures for service trading and analyse their capabilities taking the service trading properties as a reference. Finally, we compare those abstract architectures.

3. Development of an abstract architecture for service trading that deals with complex scenarios and supports advanced automated decision-making.

   This goal is successfully achieved by Chapter 3. In that chapter, we propose a novel abstract architecture for service trading. The abstract architecture has been described following the organisational metaphor proposed
<table>
<thead>
<tr>
<th></th>
<th>Ours</th>
<th>OGSA</th>
<th>SWSA</th>
<th>WSMO-Full</th>
<th>ASG</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Yes</td>
<td>Yes (distributed)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>(2)</td>
<td>Yes</td>
<td>No</td>
<td>Yes (ontologies)</td>
<td>Yes (ontologies)</td>
<td>Yes (ontologies)</td>
</tr>
<tr>
<td>(3)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(4)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>(5)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(6)</td>
<td>Yes</td>
<td>Out of scope</td>
<td>Out of scope</td>
<td>Out of scope</td>
<td>Yes</td>
</tr>
<tr>
<td>(7)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(8)</td>
<td>Service, party, trading</td>
<td>Out of scope</td>
<td>Service, party</td>
<td>Service, trading</td>
<td>Service</td>
</tr>
<tr>
<td>(9)</td>
<td>Yes</td>
<td>Partial</td>
<td>Out of scope</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(10)</td>
<td>Yes</td>
<td>WS-Ag</td>
<td>Yes, FIPA-CL based</td>
<td>Yes, through chor.</td>
<td>Yes (e.g. WS-Ag)</td>
</tr>
<tr>
<td>(11)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(12)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Partial</td>
<td>No</td>
</tr>
<tr>
<td>(13)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(14)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Lacks temporal constr.</td>
<td>No</td>
</tr>
<tr>
<td>(15)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>(16)</td>
<td>Both</td>
<td>Seems coop.</td>
<td>Seems non-coop</td>
<td>Seems non-coop</td>
<td>Both</td>
</tr>
<tr>
<td>(17)</td>
<td>Both</td>
<td>Both</td>
<td>Consumer</td>
<td>Both</td>
<td>Chiefly Consumer</td>
</tr>
<tr>
<td>(18)</td>
<td>Many (role-based)</td>
<td>Independent services</td>
<td>Out of scope</td>
<td>Out of scope</td>
<td>Technologies</td>
</tr>
<tr>
<td>(19)</td>
<td>Yes</td>
<td>Out of scope</td>
<td>Out of scope</td>
<td>Out of scope</td>
<td>Out of scope</td>
</tr>
<tr>
<td>(20)</td>
<td>Not limited</td>
<td>WS-Ag</td>
<td>Semantic info</td>
<td>Semantic info</td>
<td>Semantic Info</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of our abstract architecture
in GAIA [32]. Therefore, the architecture is composed of six organisations (trading, discovery, information, selection, 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 abstract architecture\(^1\). Finally, an analysis of the abstract architecture and a comparison with the others is carried out in Section 4.1. The analysis shows how the proposed abstract architecture is well suited to tackle complex scenarios and that provides elements to develop an advanced automated decision-making.

We introduce the concept of Trading Protocol as a method for defining the temporal features and behavioural stages of trading scenarios. These protocols drive the choreography of the different elements and allow a temporal match procedure among SLA demands/offers of stakeholders.

4. Analysis of current technologies for service trading.

We believe that we successfully achieve this goal in Section 3.9. In that section, we use the abstract architecture to analyse the scope of existing approaches, in particular standards, under the point of view of creating agreements. This analysis, hence, provides useful guidance to select the most appropriate standard in an implementation of the service trading process.

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

4.3 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 working in the identification of characteristics of the service trading process, developing an architecture based on standards that use the abstract architecture described here as reference, and building a framework that facilitates and supports an advanced automated decision-making and that is chiefly focused on the negotiation of agreements.

**Identification of characteristics of the service trading process**  In this work, we describe four typical scenarios of service trading and we obtain a set of properties of the service trading process. However, these properties are conceived to be applied to abstract architectures and, 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. Therefore, the future work in this line may be:

\(^1\)For a detailed description of the modelling elements that have been used, see Section 3.1.1
• Identifying lower-level properties of non-abstract architectures so that they cover concrete technologies, protocols and algorithms. In so doing, a comparison of concrete architectures may be carried out in a similar way to what has been done with abstract architectures in Section 2.3.

• Analysing additional service trading scenarios to identify the properties that an abstract architecture for service trading must have to operate successfully in them in order to define a method to select the architecture for each of them.

Development of an standards-based architecture that use the abstract architecture described here as reference  Although there exist some implementations of architectures for service trading such as CREMONA [17], to our knowledge, no one gives full support to the most complex scenarios described here. Therefore, a concrete architecture for service trading can be developed to support those complex scenarios. That concrete architecture must be based on the abstract architecture defined in this work. Additionally, it is convenient that it uses emergent standards of the industry in order to improve the interoperability with other service trading architectures. We believe that the best approach to develop such architecture is the following:

• First, a software framework based on standards and taking the abstract architecture as reference may be developed.

• Then, this software framework may be deployed in several real scenarios in order to validate it.

Framework for automated negotiation of agreements  The negotiation of agreements is a very complex process by itself. Therefore, it is convenient to develop a specific framework for automated negotiation that supports a variety of negotiation protocols and facilitates the development of decision-making mechanisms. This mechanisms may be used to generate counterproposals during the negotiation process and to decide whether to accept or not an incoming proposal. We foresee that the development of such a framework involves a refinement of the information, agreement making and binding organisation in order to build a coherent system that gives full support to implement concrete decision-making algorithms.
Appendix A

Curriculum vitae
Bibliography


