A Generic Platform for Conducting SLA Negotiations

Edwin Yaqub, Philipp Wieder, Constantinos Kotsokalis, Valentina Mazza, Liliana Pasquale, Juan Lambea Rueda, Sergio García Gómez, and Augustín Escámez Chimeno

Abstract In service-oriented systems, negotiating service level agreements (SLAs) occupies a central role in the service usage cycle. It is during negotiations that parties are brought together in an interactive mechanism determined by the negotiation protocols. The choice and description of negotiation protocol determines the scope of information flow which in turn influences convergence upon an agreement. In this chapter, we observe the state of the art on negotiations and introduce the generic negotiation platform developed for the SLA@SOI framework. We strive for a generic approach for protocol description and execution that also caters for domain-based rationality and ease of adoption.

1 Introduction

Procuring software as a negotiated service is gaining popularity for various business and technological reasons [7, 9]. Various offshoots of this paradigm include Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). These emerging business models have an inherent potential to reduce the Total Cost of Ownership (TCO) and improve Return on Investment (ROI). Traditionally, software is purchased under a license and used accordingly. Under the

Edwin Yaqub, Philipp Wieder, Constantinos Kotsokalis
TU Dortmund University, August-Schmidt-Strasse 12, 44227 Dortmund, Germany, e-mail: {edwin.yaqub,philipp.wieder,constantinos.kotsokalis}@tu-dortmund.de

Valentina Mazza, Liliana Pasquale
Politecnico di Milano, piazza L. Da Vinci, 32, 20133 Milano, Italy,
e-mail: {vmazza,pasquale}@elet.polimi.it

Juan Lambea Rueda, Sergio García Gómez, Augustín Escámez Chimeno
Telefónica Investigación y Desarrollo, Madrid, Spain,
e-mail: {juanlr,sergg,escamez}@tid.es

P. Wieder et al. (eds.), *Service Level Agreements for Cloud Computing*,
DOI 10.1007/978-1-4614-1614-2_12, © Springer Science+Business Media, LLC 2011

new ”aaS” models, the user subscribes to a software [17]. This is determined by a process of negotiation expected to converge on an agreement between the service customer and the service provider. During negotiation, the software is tailored to a consumer's needs and provider's capabilities.

Hence, the ”as a service” paradigm injects flexibility to the notion of software usage. In an open market, as envisaged by service-oriented computing, this flexibility becomes a necessity, as usage aspects like cost and quality cannot be fixed beforehand; rather, they depend on the current situation of supply and demand in the market [12]. Analogies to the stock exchange market are often made to explain this paradigm.

Negotiating parties are brought together through negotiation protocols, which determine the rules of engagement. Various styles have been observed, varying from simple take-it-or-leave-it to multi-round negotiations and even more complex auction-like interactions. These have been studied extensively in the literature under the context of automated negotiations, as we present in Section 2.

A negotiation protocol determines the cardinality of parties involved, their roles, the visibility of the offers exchanged, session management, bounds for negotiation rounds, and so on. Usually, a dedicated software machinery is required to execute negotiation protocols, so that the negotiating agent may perform its domain-specific functionality either as a client or provider of the service under negotiation. In SLA@SOI, this software machinery is developed as a generic negotiation platform. The platform is designed to abstract from the lower-level functionality, which tends to get domain-specific, by allowing a loose coupling with the planning and optimisation component (see Chapter ‘GSLAM – The Anatomy of the Generic SLA Manager’), such that a strategy is used to evaluate each proposal to maximise a utility function.

To enter the market, providers often advertise their services using publishing templates. Templates express the functional and non-functional properties of a service, along with the necessary constraints to tailor it as a concrete offer. A single service may be advertised through multiple templates. Customers shortlist providers based on templates of interest and use template(s) to initiate a particular negotiation. Template-based negotiations have been implemented by the Web Service Agreement (WSAG) framework (lately as Web Service Agreement Negotiations) [1, 20] and IBM's WSLA framework [10]. The SLA@SOI framework has also adopted template-based negotiations. Using these templates, offers and counter-offers are exchanged between the negotiating parties in a sequence determined by the negotiation protocol. In the best case, an agreement is reached and documented as a Service Level Agreement (SLA). The provider provisions the agreed upon resources and the customer starts to use the service from the time SLA comes into effect. The customer abides by the agreed usage levels and the provider maintains the agreed quality of service levels. In case of violations, penalties are enforced. An SLA may be need to be renegotiated if customers experience a change in service's demand, or the provider needs to readjust its resources.

2 State of the Art

Negotiation has gained a lot of interest in research. Various concepts from economics, artificial intelligence and game theory have been combined to address negotiation-related concerns through interdisciplinary approaches.

On theoretic lines, one of the first formal analyses of the negotiation process was carried out by John Nash [11] in his work on one-to-one bargaining and later on non-cooperative games. This popularised game theory, and later led to its inception in computer science, especially among distributed intelligent agents [26]. Several phenomena have been analysed when agents negotiate pursuing individual strategies. Some of these are summarised here:

- Pareto efficiency: If no agreement improving the utility of one of the negotiating parties can be found, the negotiation is considered to be Pareto efficient.
- Stability (Nash equilibrium): Two strategies are said to be in Nash equilibrium if they are the best for each of the parties involved in the negotiation. There might be multiple equilibria or none at all.
- Cooperative/non-cooperative: If the aim of a certain negotiation is to maximise the utility functions of each of the partners involved in the negotiation, the negotiation is said to be cooperative. On the other hand, when parties only take care of their own interests, the negotiation is non-cooperative.

Several frameworks and Negotiation Support Systems (NSS) have been proposed in the literature. OPELIX [16] is a European project that permits a customer and a provider to have fully automated bilateral negotiations. The OPELIX architecture implements all the fundamental phases of a business transaction: product offers and discovery, a negotiation process, payment activities, and the delivery of the product to the customer. However it does not support sophisticated negotiation protocols, rather it is restricted to bilateral negotiations.

Inspire [15], Aspire [14] and e-Agora [6] are related projects developed by Concordia University (Montreal) in conjunction with Carleton University (Ottawa). Inspire [15] supports human operators in managing bilateral negotiations, managing offers and counter-offers made by the participants. Functions guiding the decision of each participant are kept confidential.

Aspire [14] improves upon Inspire by providing negotiation support through intelligent agents that make suggestions to users regarding what operations to perform. Note that agents do not completely automate the negotiation process, but only provide support in taking decisions; they are completely aware of the status of the negotiation sessions, and implement a specific negotiation strategy defined in terms of weights on negotiation variables and objective functions.

The e-Agora [6] project provides a complex marketplace in which users interact through autonomous intelligent agents. The system provides a process model and a set of supported protocols. The process is defined as a series of activities and phases; protocols are defined by means of rules and restrictions on negotiation activities.

Kasbah [2] allows potential buyers and sellers to create their own agents, assign them some strategic directions, and send them to a centralised marketplace for ne-

gotiations. Support is limited to bilateral negotiations. The only valid action in the distributive negotiation protocol is for buying agents to offer a bid to sellers. Selling agents respond with a binding ”yes” or ”no”. Given this protocol, Kasbah provides buyers with several negotiation strategies that determine the function for increasing bids for a product over time.

AuctionBot [21] offers a versatile online auction server. Software agents are provided that conduct auctions on the basis of particular parameters: participation (i.e., number of participants), discrete goods (bids are allowed only for integer quantities) and bidding rules that determine acceptability and improvement of offers and closing conditions.

ASAPM [18] is multi-agent system that allows automated negotiations using the FIPA Iterated Contract Net Protocol (ICNP). Agents negotiate over quality of service (QoS) terms and the ICNP accommodates this by allowing multiple rounds of negotiation.

BREin [13] provides a broker-based framework for conducting SLA negotiations. A multi-tier negotiation protocol is used that is based on the FIPA Contract Net Protocol. The protocol scope is extended to allow for negotiation interactions among different service chains.

CAAT [19] is another framework that can be used to design multi-agent systems for automatic bilateral and trilateral negotiations. The negotiation protocol allows valid sequences of interactions using messages built upon the FIPA Agent Communication Language (ACL). An ontology defining communication semantics is developed and used in messages to convey a certain action.

The approaches presented above make interesting advances towards automated negotiation, yet they are not flexible enough to design custom interaction behaviours or to easily customise negotiations for individual application domains.

To this end, SECSE [8] provides a flexible infrastructure that can be tailored in terms of multiplicity, workflow, protocol and decision model to fit a specific application domain. The architecture of the negotiation framework is composed of a marketplace that harbours multiple agents. Each agent is associated with a specific negotiating participant and a negotiator component. Negotiators interface human participants with the negotiation framework through GUIs that allow them to place offers and counter-offers. Additionally, a built-in decision model or user-defined decision model can be encapsulated to execute automatic negotiations. SECSE supports hybrid negotiations, where some participants are automated agents while others are human beings. A participant may exploit a negotiation coordinator, which is responsible for coordinating the actions taken by its various negotiators. The marketplace acts as an intermediary in all interactions between the participants, providing validity checks for the offers exchanged. These checks are based on the structure and current state of the negotiation workflow. To make the search for agreements more efficient, the marketplace provides a mediator component, which guides the generation of offers towards a convergence of the individual objectives. This, however, requires that participants share their objectives with the mediator. The negotiation framework allows designers to define their negotiation workflow as a state chart

using ArgoUML, and their negotiation protocol as a set of rules in the JBoss rule syntax.

WSAG [1] is a standardising effort from the Open Grid Forum (OGF) delivers a specification for web-service-based agreements. A language is developed that can be used to specify an agreement template and standard operations for managing the life cycle of the service. In addition, it provides a negotiation protocol that allows for take-it-or-leave-it styled bilateral negotiations. More recently, work on Web Service Agreement Negotiation (WSAG-N) [20] has addressed broadening its scope to specify custom interaction behaviours and thus support a host of negotiation protocols written as per given specifications.

Analysing the architecture and design approaches proposed for the NSS, different patterns are observed: 1) broker-based architectures, where a broker component manages one-to-one negotiations on behalf of involved parties; 2) marketplace-based architectures, where the parties involved in M-to-N negotiations are managed by an intermediate marketplace (approaches 1) and 2) require negotiation participants to expose their preferences to the negotiation framework); and 3) independent agents negotiate with each other without mediation. These patterns freely compete or cooperate based on individual rationality. From the protocol description perspective, we observe rule-based approaches where business rules regulate the negotiation process, use of ontologies and schemas represent message content and semantics, and negotiation protocols have parameter-based configurations.

3 Protocol Engine

In SLA@SOI, agents modelled as SLA managers (see also Chapter 'GSLAM – The Anatomy of the Generic SLA Manager') conduct automated negotiations using a generic platform called the Protocol Engine. The Protocol Engine is an integral component of the Generic SLA Manager and is therefore available to all concrete implementations of GSLAM. The Protocol Engine establishes communication between negotiating parties by using a negotiation protocol. The negotiation protocol in this text does not refer to a low-level communication protocol like TCP or routing protocols like IP. In fact, it refers to a higher-level interaction mechanism that is employed by the negotiating parties under a unique context. This context is called the negotiation session and is managed by the Protocol Engine at each negotiating end. In SLA@SOI, a flexible approach to encoding negotiation protocols has been developed. The basic idea circulates around modelling interaction behaviour as a state machine. This approach is further discussed in Section 4. In addition to interaction behaviours, the negotiation protocols may consider domain-sensitive content that may affect negotiations, keeping in sight past negotiation experience and current business policy. The Protocol Engine, however, is designed to operate at a higher level of abstraction than the negotiation protocols, and is therefore able to execute them without tight coupling to the domain or universe of discourse served by its SLA manager. Domain agnosticity of the Protocol Engine, combined with domain

sensitivity of the negotiation protocols, allow SLA@SOI to achieve a generic mechanism for conducting automatic negotiations among various SLA managers.

3.1 Design

The functionality provided by the Protocol Engine is broken up into three tiers, as shown in Figure 1.

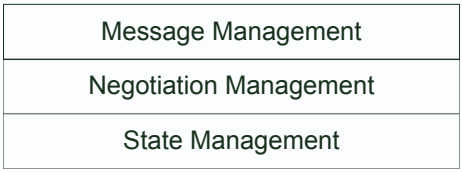

Fig. 1 Tiers of the Protocol Engine

- Message management: This tier acts as the Protocol Engine's faćade to the outside world. All negotiation requests and responses are handled here by a singleton message handler component. The message handler implements an INegotiation interface, as seen in Figure 2, which provides all operations needed to conduct negotiations. This interface is exposed as a web-service for remote access. A client program offered by the SyntaxConvertor component (as described in Chapter 'GSLAM – The Anatomy of the Generic SLA Manager') of the GSLAM is used to invoke operations of the web-service. The message handler passes incoming requests to the negotiation management tier. Additionally, it posts requests from the negotiation management tier to the negotiating parties, as in the case of the *initiateNegotiation* and *negotiate* operations.
- Negotiation management: This tier allocates a negotiation manager for each negotiation. The negotiation manager maintains the negotiation session identifiable by a unique identifier. This identifier is used by the negotiating parties in subsequent operations. The session is initialised using two artifacts: a) the negotiation protocol and b) the template(s) of the service under negotiation. The session also stores information such as the involved parties, offers received, counter-offers sent, reasons for cancellation or termination of SLAs (when applicable) and protocol parameters. This tier further collaborates with the state management tier to ensure that the protocol rules are abided by before control is handed over to the planning and optimisation (POC) component (see Chapter 'GSLAM – The Anatomy of the Generic SLA Manager').
- State management: This tier implements a state engine that maintains the states of the negotiation based on the execution of the state machine as defined in the negotiation protocol. In SLA@SOI, the protocol is encoded using rules. The state

management tier therefore acts as a wrapper over a rule engine. It implements a feedback control loop by passing events to the rule engine corresponding to the invoked operations, and receiving the processed results. Inside the state engine, protocol-specific events are converted to rule-engine-specific commands and *vice versa*.

Figure 2 shows an architectural view of the Protocol Engine.

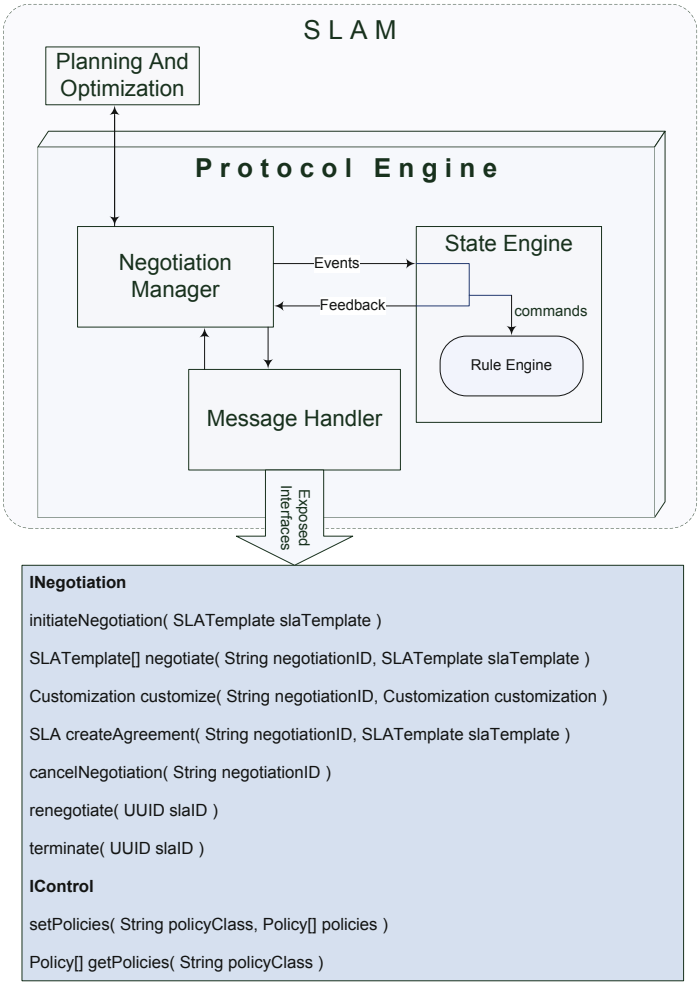

Fig. 2 Architecture of the Protocol Engine

4 Protocol Description

Negotiation protocols have been widely studied in the literature. Before we present the objectives set for our work, and the resulting approach we developed, it is worthwhile to have a walk through the closely related work.

4.1 Related Work

Negotiations are best described as a process with multiple aspects. Negotiation protocol determines how negotiating parties are brought together and the interaction behaviour that follows (e.g., which sequence of messaging is allowed and how the negotiation is concluded or terminated). As seen in Section 2, most NSS support restricted interaction behaviour for conducting negotiations. Further on, although not directly controlled by the protocol, the messaging mechanism (synchronous or asynchronous) upon which an interaction mechanism is based is also an important factor for agents. In attempts to support a host of protocols as required, research efforts have been made to generalise these mechanisms and related characteristics to abstract from any single protocol. It is in this context that we observe a trend towards employing rule-based approaches to capture protocol semantics that are understood unambiguously by the negotiating agents. Despite this commonality, each solution differs in its scope, objectives and design approach.

In [22], negotiation rules are studied under the context of auctions. Three activities are extracted as applicable to all auction protocols: handling of requests, computing exchanges, and sharing of intermediate information helpful to reach a conclusion. The activities are complemented with a set of standard parameterised rules that impose restrictions: rules related to bids, computing exchange (counter-offers), and the visibility of bids among participants, for example. Although acknowledged, the structuring of these activities and rules to model a custom interaction behaviour is left to the protocol designer. On somewhat similar lines, Jennings et al. [3] have developed a negotiation framework that can be used to model a variety of negotiations. They provide a taxonomy of predefined rules and a simple interaction protocol that uses these rules to realise a certain negotiation mechanism based on an asynchronous mode of communication as specified by FIPA ACL messaging. In addition, an OWL-Lite-based ontology language has been developed to represent service templates and offers. Both [22] and [3] target price-centric negotiations that try to build upon well-engineered rule sets. Although befitting controlled traditional auction settings, these approaches become restrictive when it comes to SLA negotiations taking place in open world service-oriented markets.

SLA negotiations are usually based on service templates that the service providers make publicly available for negotiation. The templates contain a set of properties, with price being just one of them. Most of these properties concern the quality of service (QoS) that the customer and the provider negotiate to agree upon. Each QoS property contains a set or range of values that the customer may choose

from. This is a fundamental shift from a single attribute price-centric model towards a multi-attribute model. Needless to say, a single template may also be used to conduct multi-unit negotiations, which usually require special considerations as in multi-commodity auctions. The problem is further complicated by the fact that in service-oriented markets, most agents are self-interested and would not like to share information related to their business objectives or utility-maximising functions. This introduces challenges for the above-mentioned approaches, which for instance try to deliver a standard rule for judging improvement in offers received in subsequent negotiation rounds. Among self-interested agents conducting SLA negotiations, complicated correlations among the negotiable properties are kept private.

A generic approach for conducting SLA negotiations therefore requires more flexibility and loose coupling between the domain-specific and generic aspects. An attempt to draw this fine line has been made in [25], where a set of generally applicable negotiation parameters have been identified and implemented as an XML language. A meta-negotiation phase allows the negotiating agents to fix the values of the negotiating parameters that serve as a concrete negotiation protocol. Some parameters include party roles, permissions, cardinality, admission credentials, starting and termination criteria. Rule-based restrictions can be appended to parameters in external rule languages without limiting choice. In addition to multilateral negotiations, bilateral negotiations are also given due consideration. The language inherits its service description and guarantee term constructs from WS-Agreement.

4.2 Design

In the SLA@SOI project, a broadly scoped meta-model called SLA* has been developed to describe a service template that includes negotiable QoS properties, provider information and more[1] . Negotiations take the service template into account. This serves to clearly differentiate the subject of negotiation (i.e., the QoS terms of the service) from the aspects that govern the negotiation process. In this section, we present our methodology in representing the negotiation process. We abstract this process as a set of phases that can be structured together as a general purpose state machine (GPSM). This representation is highly generic and is termed a generic protocol that basically serves to develop an interaction behaviour. We further employ a customisable parameter-oriented approach to transform the generic protocol into a concrete negotiation protocol that would govern negotiation behaviour. In the following section, we first describe the GPSM and how the approach is modeled, making it easy to plug protocols into our execution platform: the Protocol Engine. Although not related to the field of negotiation, we do draw some design principles from past experience in encoding and executing medical protocols intended for the personal health records domain [28].

[1] The model is described in full detail in Chapter 'The SLA Model'

We adopted event-driven design in liaison with modestly engineered rules to reach our technical objective: specialisation of generic negotiation protocols still able to be executed in a standard manner. The generic negotiation protocol is structured as a GPSM that abstracts upon different phases of negotiation, as shown in Figure 3. The generic protocol provides a reference interaction behaviour; however, the approach does not restrict the protocol designer to a given state set, or to any particular structure, thereby allowing the design of custom interactions. The GPSM comprises five states: initialise, customise, negotiate, decide and end. At any point in time, the negotiation process resides in a single state. Each state determines what operations are allowed or disallowed by entertaining the trigger events in a certain manner. This also determines the next state to which the machine will transit. The protocol is encoded using rules that are divided into two categories: The first category comprises *generic rules* that encode the state machine; a reference rule set is provided for GPSM. The second category comprises *domain-sensitive rules* that take into account an agent's local considerations when conducting the negotiation. Before addressing the encoding details for the rules, we briefly describe the semantics of the five GPSM states.

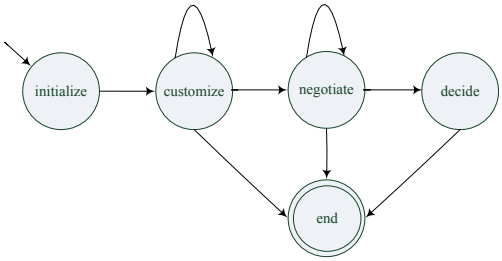

Fig. 3 A General Purpose State Machine

- *initialise*: This state represents establishment of a negotiation session between negotiating parties. A unique identifier is assigned and used by involved participants for conducting subsequent operations. This state is mandatory and is influenced by the arrival of an initialisation event.
- *customise*: This optional—but important—state follows the initialise state and constitutes a customisation mechanism where negotiating parties attempt to customise generic protocol parameters. An output is a concrete protocol for governing subsequent behaviour. This state is influenced by the arrival of a customise event. More details on the customisation mechanism are provided in Section 5.
- *negotiate*: In this state, parties negotiate with each other by submitting offers and counter-offers to reach an agreement (SLA). Usually multiple rounds would be required to conclude this state. This state is influenced by proposal-related events.

- *decide*: This state determines if negotiation can be gracefully concluded. It is reached if an agreement creation is requested, negotiation rounds are consumed, or a timeout occurs. Several events may trigger this state.
- *end*: This state marks the end of negotiation, which could possibly result in the creation of an SLA.

A lightweight data model has been developed that provides classes for concepts like events, protocol parameters, negotiation sessions, and states. Generic categories of rules employ states to encode the state design pattern. A rule represents preconditions specified in the IF part, that when met execute the post-conditions specified in the THEN part. Rule qualification is driven by arrival or departure of events; therefore rules are coupled with events to regulate success or failure scenarios. In the case of the latter, appropriate exceptions are generated with a specialised message as provided in the rule. A taxonomy of events has been realised as a result. Following this simple rule-encoding scheme allows a loose coupling with the Protocol Engine component that is responsible for generating, passing and receiving processed events from the rule engine that executes the negotiation protocol. Keeping behavioural logic in rules inherits additive benefits of the rule-based approach: for instance, the protocol remains maintainable over time, as it is humanly readable and machine executable. Further, rules can be externally configured without requiring code recompilation or deployment. Two reference rules representing a success and failure scenario are shown in Figure 4.

Negotiation protocols are divided into two basic categories: bilateral and multilateral negotiations. For proper demarcation, these are kept under distinct negotiation interfaces and are to be provided over different ports. The interfaces provide actual operations that the client programs may use to conduct negotiations. Our protocol description approach proves beneficial when negotiation interfaces are conjoined with behaviour-regulating rules while maintaining a plethora of information in the session associated to the ongoing negotiation.

Session management is important for the seamless functioning of other components involved in performing negotiation. In this regard, the planning and optimisation component (POC) plays a special role. It acts as the local executive controller of the SLA manager. The POC implements domain- and use-case-specific strategies that drive negotiation from the back seat. A strategy implements some decision-making logic to process an incoming offer and generate counter-offer(s) by considering the current state of available resources as well as business objectives. Further on, the POC resolves service dependencies (if any) and decides when to outsource incoming requests to third parties by conducting nested negotiations. For reasons of convergence, POCs benefit from information available in the negotiation session by analysing the offers exchanged with the negotiating party and its profile (Section 5.1). This analysis provides a possibility of cooperation even among self-interested agents by understanding their partner's sphere of interest. In Section 5.2, we outline how an optional critique may be provided by the POC to encourage a negotiating partner to move future offers in a particular direction of interest, or to pull him to a middle ground.

A rule demonstrating a success scenario

```
IF
  { currentState : State(name == NEGOTIATE && status == RUNNING
    && currentRound <= ProtocolParameters.negotiationRounds) }
  AND
  { allowedEvent : Event(name == PROPOSAL_ARRIVED) }
THEN
  { currentState.incrementNegotiationRounds(1) }
  AND
  { update(currentState) }
  AND
  { allowedEvent.setProcessedSuccessfully(true) }
```

A rule demonstrating a failure scenario

```
IF
  { currentState : State(name == NEGOTIATE && currentRound >
    ProtocolParameters.negotiationRounds) }
  AND
  { disallowedEvent : Event(name == PROPOSAL_ARRIVED) }
THEN
  { retract(currentState) }
  AND
  { insert (new State(StateName.DECIDE && status = RUNNING)) }
  AND
  { disallowedEvent.setProcessedSuccessfully(false) }
  AND
  { disallowedEvent.setProcessingAfterMath(MAX_ROUNDS_VIOLATED) }
```

Fig. 4 Reference rules

The justification for separating strategic behaviour from the negotiation protocol is made on two levels: Firstly, strategies tend to get domain-specific, and secondly, they have high computational intensity (as in case of composite services that perform QoS-aware service composition, a known NP-Hard problem [4, 5]). These are therefore best served as black box implementations clearly separated from the generic aspects of conducting negotiations. For these reasons, the protocol rules are intentionally spared from implementing strategic behaviour: a functionality delegated to the POC during negotiations (Figure 2).

4.3 Bilateral Negotiations

One of the most widely occurring forms of negotiation among independent agents is bilateral negotiation. As bilateral negotiation serves most of the use cases considered in the SLA@SOI project, we have early adoption results for the same. In a bilateral negotiation, a customer negotiates directly with a provider. If the provider has further dependencies, nested or sub-negotiations are possible in a similar fashion. A somewhat advanced scenario would involve a customer negotiating over a product offered by a certain enterprise: The on-line business unit of this enterprise

is represented by an instance of SLAM called BSLAM. The BSLAM could depend upon external software services, offered by an agent called the Software-SLAM or SWSLAM. The SWSLAM in turn must deploy and instantiate its software services over an infrastructure capable of delivering a guaranteed QoS as required by the BSLAM. For this, the SWSLAM needs to negotiate with an infrastructure service provider, represented by its agent, called INSLAM. This scenario helps expand the negotiation scope over multiple providers and exposes a possible chain of dependencies to be resolved through negotiations. This depiction realistically sketches how SLA negotiations would be employed in service-oriented markets. Interestingly, each stage in this potentially long hierarchy of negotiations can negotiate successfully with the next agent in line in a bilateral manner, by customising the negotiation protocol with parameter values considered realistic.

Request multiplicity is taken care of by conducting multiple bilateral negotiations in parallel. The responsibility of having a unified view of currently available resources at any time considering ongoing tentative reservations is kept internal to POC. A simplified interaction is illustrated in Figure 5. Here, a customer initiates negotiation with a provider and receives a negotiation identifier. This is used in subsequent steps, first to customise the protocol, and later to negotiate offers and counter-offers. Both of these may require certain iterations. At some point, the customer requests an agreement by submitting a final offer. If accepted, the provider sends back the SLA, which is then provisioned.

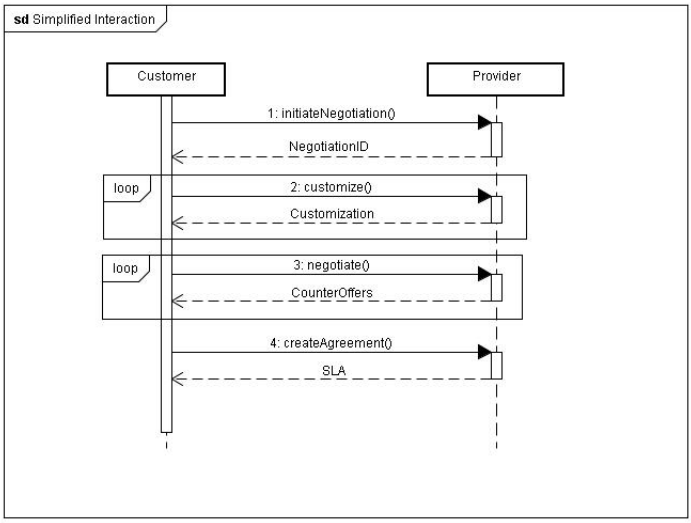

Fig. 5 Simplified interaction

As an extension of our work, multilateral negotiations are being considered under the context of auctions.

5 Negotiation Rationality

Negotiation technologies have garnered a lot of interest in recent years, and are seen as a key coordination mechanism for the interaction of providers and consumers in electronic markets. Such technologies provide means to reduce the costs of efficiently managing resources in fast-changing service-oriented markets. Their importance surfaces especially in the case of multi-attribute SLA negotiations, where agents engage in sophisticated protocols to intelligently negotiate over complex services to achieve mutual gain.

Complex services are usually offered by composing or aggregating other services. Considering time and other restrictions imposed by the negotiation protocols, it becomes challenging to converge upon an agreement with the customer at one end and possibly multiple providers at the other end. The dependencies among providers require the establishment of a hierarchy of SLAs at negotiation time [27]. Therefore, a sound rationale for conducting negotiations is of fundamental importance. Optimal outcomes are obtained, as per cooperative game theory, by assuming maximum information on the objectives of the involved parties. In classical multi-attribute utility theory [23, 24], the proposed solution is the use of an independent mediator that both parties can trust to reveal their preferences. However, in the case of self-interested agents doing business in e-commerce settings, it is not possible to determine what mediators would be impartial or trustworthy for establishing the rationale.

Negotiation rationale determines the degree of feasibility of a negotiation. It also serves to rule out infeasible negotiations. Infeasible negotiations are negotiations that do not have a high chance of success. In the absence of a rationale, such blindly instigated negotiations would consume precious system resources at both the customer and provider end, where these resources could otherwise be utilised for productive outcomes. In SLA@SOI, we cater for individual rationality by gathering high-level information about the customer and the provider in the form of *profiles*. Domain-sensitive rules may then be added to the negotiation protocol to compute ranks for the negotiating partner, considering past experience and current business policy. The domain-sensitive rules are an optional but useful part of the protocol that may guide the negotiation process towards a faster convergence or conclusion. For reasons of domain dependency, however, these rules cannot be provided out of the box.

5.1 Profiles

Keeping business and negotiation requirements in view, we have modeled profiles to contain information about these characteristics of the negotiating actors. For reasons of brevity, only summarised features can be discussed here. The idea is to cross-profile the negotiating party: that is, service customers profile the service providers and *vice versa*. Additionally, the product or service being negotiated can also be profiled by its provider. The profiles add value to the negotiation process by pro-

viding negotiation- and business-related history that serves as an experience base. This high-level information about the customer and provider is inspired from the business perspective of negotiations. Domain-sensitive protocol rules raise the level of abstraction of this information to determine ranks that allow judging of the negotiating party upon negotiation setup time. In the following section, we expand this concept to shed light on how this high-level information forms the basis for customising the generic negotiation protocol as hinted earlier in Section 4.2.

Providers may classify customers as companies or end users who are the direct beneficiary of the negotiation. In case of former, the size of a company (characterised as a small, medium or large enterprise) along with other factors can be considered by using a rule to determine, for instance, the number of negotiation rounds one is willing to negotiate with this entity in the future. An abstract view of a customer's economic situation is also of importance for the provider. This is ascertained by assessing punctuality in payment of dues against already established SLAs, and the worth associated with the SLAs. Profiles also depict summarised penalty information that provides further insight into previously established SLAs. All these factors can be processed by a rule that assigns a rank to the customer using current business policy. Yet another rule may build upon this information to generate an acceptable value for maximum counter-offers exchangeable during negotiations, for example.

On the other side, customers are interested in conducting further negotiations with providers that have delivered a good quality of service for previously established SLAs, while avoiding providers who have not. Customers can also associate a rank with a provider, based on the worth of the SLA under negotiation, past services, and penalty satisfaction levels. Based on the ranks, strict business policies may be encoded in yet other rules that blacklist or whitelist the negotiating partner. If this criteria changes over time, only the related rules need modification, and this does not jeopardise the overall negotiation behaviour. Other elements of interest for doing business are aspects of location: These may restrict aspects of a business due to laws in the country of the negotiating partner, or disallow trade of a certain product in that country. This same information may allow higher prioritisation of another negotiating partner if, for example, the business policy is to increase clientele in a certain location for strategic reasons. Soft counter-offers may be generated by the POC after considering the ranks and other profile information at negotiation time. Hence, the profiles provide a degree of freedom for the POC to tune its algorithm such that generated offers and counter-offers are personalised for the party, but also in line with business objectives.

After considering the number of SLAs already made, providers may forecast the selling frequency of a certain product and establish business derivatives such as the minimum markup ratio/benefits (i.e., minimum sales required for the product/service to be profitable). These forecasts may also influence the degree of flexibility shown while negotiating over the said product. Additionally, profile information may serve to influence decisions regarding product retirement (i.e., terminating agreements with a partner due to bad service, defaulting on payments, or penalty situations). The latter may also be used to perform penalty-driven renegotiations.

Customer and provider profiles can also log vital negotiation history, including past attempts to negotiate, renegotiate, or terminate SLAs. The frequency of these attempts, along with their outcomes, can be used by a rule to assign a rank for the requested negotiation. As with the scenarios presented earlier, the negotiation rank also may influence the customisation of future negotiations. For example, if the rank is high, the customer can be considered faithful or promising, and the provider may suggest a greater number of negotiation rounds or maximum counter-offers.

5.2 Protocol Customisation Mechanism

As mentioned in Section 4.2, the customise state allows negotiating parties to mutually agree on values for any customisable protocol parameters that govern the negotiation. This is done by exchanging customisation suggestions. Domain-sensitive rules come into action and set values to these parameters by coupling profile-based ranks with business policy and customisable values suggested by negotiating partners. This forms a pre-negotiation mechanism that may span several rounds until consensus is reached. It is important that the parties share the same protocol parameters to avoid undesired anomalous behaviour in later stages, and this is achieved through the protocol customisation mechanism. If consensus is not reached, negotiation is aborted.

The customisation mechanism is an active part of ongoing work on the SLA@SOI negotiation platform. We now summarise some of these customisable protocol parameters.

- *credentials* allow parties to verify each other if such an understanding exists. This could be an individual key under Primary Key Infrastructure (PKI)-based certification environments.
- *customizationRounds* informs the negotiating partner that there will be an attempt to reach consensus on customisable parameters in a particular number of rounds, starting with two. This is a sliding value that may be extended during customisation. Nevertheless, at any point in time, each party may respect its own value and end the customisation process as dictated by its side of the customisation rules.
- *processTimeout* determines the lifetime of the negotiation process. Negotiation is considered invalid after this timeout has occurred.
- *negotiationRounds* determine the maximum allowed number of rounds for exchanging offers. If it is set to zero, negotiation will not take place.
- *maxCounterOffers* sets a cap on the number of counter-offers allowed in response to a submitted offer.
- *optionalCritiqueOnQoS* serves as a tip to the POC to optionally annotate critiques on the QoS terms of generated counter-offers. Critiques may involve keywords like INCREASE, DECREASE, CHANGE, and so on, thus helping to convey a message to the negotiator to consider submitting values for which the chances of reaching agreement is higher. In this way agents may guide or pull each other in their direction of interest.

- *isSealed* is of interest in multilateral negotiations such as auctions. For example, it would be false for an English auction but true for First-Price-Sealed-Bid or Vickrey auction.

In addition to above parameters, which are customised in a mutual manner, there are parameters that remain non-customisable, to avoid sharing vital information. These are especially applicable to auctions (e.g., minimum and maximum bidders, auctioneer's listening time to receive bids, and the start time of an auction).

5.3 Business Take-Up of Negotiations

Business requirements drive the negotiation process for each entity involved in the negotiation. These requirements need to be met *in addition to* fulfilling customers' QoS requirements. Profiling the negotiating parties can help in adapting negotiations to a specific manner. From a business point of view, particular aspects of the negotiation need to be controlled. As seen earlier, profiles can be used to customise negotiations to better suit business goals, while assigning each negotiator a personalised negotiation field. Negotiation profiles can be applied and mapped to different ranks, which are obtained by rules-mapping current business policy to past negotiations and business information about the product, customer and provider involved. Once ranks have been determined, negotiation commences in a personalised and rational manner. This lightweight approach helps manage and drive the negotiation, while at the same time allowing it to benefit from volatile policy logic that can be easily and rapidly updated in rules.

Business Negotiation Flow:

Aligned with business-level control of the negotiation, negotiations are materialised in different adoption styles. Automatic, semi-automatic and manual negotiation are the different proposed negotiation flows.

- Automatic negotiation: Agents negotiate directly with each other and exchange offers and counter-offers that are automatically processed and generated. The agents have preset decision-making capabilities and try to maximise or minimise their own utility. Depending on the agents' decision models, agents may or may not attempt to cooperate with the other in converging upon an agreement.
- Semi-automatic negotiation: Automatic negotiation can be split in two halves: the first half involves a special subset of cases in which business personnel could be given manual control of the negotiation, while the second half includes those other cases that can be managed automatically.
- Manual negotiation: In this scenario, business personnel receive customer offers in real-time and make counter-offers or reject the offers by practicing full control of the offers being exchanged.

Each of these behaviours is adopted by different use cases that use the SLA@SOI framework. The framework provides necessary hooks and programmable interfaces

to intercept the various interactions involved in negotiations to implement a certain negotiation flow. Early adoption results show that controlling the negotiation flow becomes an important consideration for various businesses interested in the SLA@SOI framework.

6 Conclusion

In this chapter, we reviewed the state of the art available on negotiations and presented our generic negotiation platform for conducting SLA negotiations. We illustrated the flexibility of our approach, which also takes into account domain-based rationality. Early adoption results from use cases encourage us to extend our work, while also considering contemporary efforts.

Diversity in research is expected to reveal new facts regarding the process of automatic SLA creation, especially in areas such as nested dependencies, efficient and fruitful optimisation algorithms, negotiation strategies that quickly converge upon agreements, analysis of market trends, and party profiling. These areas have been established as solid research fields but have not yet been fully exhausted. As the field matures, scientific progress will be harnessed to produce tangible results that will lead towards a successful service-oriented economy.

References

[1] Andrieux A., Czajkowski K., Dan A., Keahey K., Ludwig H., Nakata T., Pruyne J., Rofrano J., Tuecke S., Xu M.: Web Services Agreement Specification (WS-Agreement), `https://forge.gridforum.org/projects/graap-wg/.posted-at2006-09-05`

[2] Chavez A., Dreilinger D., Guttman R., Maes P.: A Real-Life Experiment in Creating an Agent Marketplace. In: Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97) (1997)

[3] Lomuscio A.R., Wooldridge M., Jennings N.R.: A Classification Scheme for Negotiation in Electronic Commerce. Journal of Group Decision and Negotiation (pp.31-56) **12**(1) (2003)

[4] Ardagna D., Pernici B.: Global and Local QoS Guarantee in Web Service Selection. In: Proceedings of the Third International Conference on Business Process Management (2005)

[5] Bonatti P.A., Festa P.: On Optimal Service Selection. In: Proceedings of the 14th international conference on World Wide Web (2005)

[6] Chen E., Kersten G. E., Vahidov R.: An E-marketplace for Agent-supported Commerce Negotiations. In: Proceedings of 5th World Congress on the Management of eBusiness (2004)

[7] Dubey A., Wagle D.: Delivering Software as a Service. White paper, The McKinsey Quarterly (2007)
[8] Di Nitto E., Di Penta M., Gambi A., Ripa G., Villani M.L.: Negotiation of Service Level Agreements: An Architecture and a Search-Based Approach. In: Proceedings of the 7th International Conference on Service Oriented Computing, ICSOC 2007 (2007)
[9] Elfatatry A., Layzell P.: Negotiating in Service-Oriented Environments. Communications of the ACM **47**(8), 103–108 (2004)
[10] Ludwig H., Keller A., Dan A., King R.P., Franck R.: Web Service Level Agreement (WSLA) Language Specification 1.0 (wsla-2003/01/28) (2003)
[11] Nash Jr.J.F.: The Bargaining Problem. Journal of the Econometric Society **18**(2) (1950)
[12] Bennett K., Layzell P., Budgen D., Brereton P., Macaulay L., Munro M.: Service-based Software: The Future for Flexible Software. In: Proceedings of the Seventh Asia-Pacific Software Engineering Conference (2000)
[13] Karaenke P., Kirn S.: Towards Model Checking and Simulation of a Multi-tier Negotiation Protocol for Service Chains (extended abstract). In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010) (2010)
[14] Kersten G.E., Lo G.: An Integrated Negotiation Support System and Software Agents for E-Business Negotiation. International Journal of Internet and Enterprise Management **1**(3) (2003)
[15] Kersten G.E., Noronha S.J.: WWW-based Negotiation Support: Design, Implementation and Use. Journal of Decision Support Systems **25**(2) (1999)
[16] Hauswirth M., Jazayeri M., Miklos Z., Podnar I., Di Nitto E., Wombacher A.: An Architecture for Information Commerce Systems. In: Proceedings of the Sixth International Conference on Telecommunications (ConTEL) (2001)
[17] Turner M., Budgen D., Brereton P.: Turning Software into a Service. Proceedings of the IEEE Computer Society **36**(10), 38–44 (2003)
[18] Chhetri M.B., Mueller I., Goh S.K., Kowalczyk R.: ASAPM An Agent-based Framework for Adaptive Management of Composite Service Lifecycle. In: Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, 2007 (2007)
[19] Ncho A., Aimeur E.: Building a Multi-Agent System for Automatic Negotiation in Web Service Applications. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (2004)
[20] Waeldrich O., Battre D., Brazier F., Clark K., Oey M., Papaspyrou A., Wieder P., Ziegler W.: WS-Agreement Version Negotiation 1.0 (2007). URL `https://forge.gridforum.org/sf/go/doc15831?nav=1`
[21] Wurman P.R., Wellman M.P., Walsh W.E.: The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents. In: Proceedings of the Second International Conference on Autonomous agents (1998)
[22] P.R. Wurman, M.P. Wellman, and W.E. Walsh: Specifying Rules for Electronic Auctions (2002)

[23] Raiffa H.: The Art and Science of Negotiation. Harvard University Press (1982)
[24] Raiffa H.: Lectures on Negotiation Analysis. PON Books, Harvard Law School (1996)
[25] Hudert S., Eymann T., Ludwig H., Wirtz G.: A Negotiation Protocol Description Language for Automated Service Level Agreement Negotiations. In: Proceedings of the IEEE Conference on Commerce and Enterprise Computing (2009)
[26] Weiss G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press (2000)
[27] Theilmann W., Happe J., Kotsokalis C., Edmonds A., Kearney K., Lambea J.: A Reference Architecture for Multi-Level SLA Management. Journal of Internet Engineering **4**(1) (2010)
[28] Yaqub E., Barosso A.: Distributed Guidelines (DiG): A Software Framework for Extending Automated Health Decision Support to the General Population. Journal of American Health Information Management Association (AHIMA) (2010)