Enabling Self-organising Service Level Management with Automated Negotiation

Lei Liu, Hartmut Schmeck
Institute AIFB
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
[lei.liu, hartmut.schmeck]@kit.edu

Abstract—Automated end-to-end Service Level Management (SLM) is crucial to enable self-organising Service-Oriented Architectures (SOA). In this paper, we present an approach based on Organic Computing to enable automated SLM by using automated service level negotiation. The evaluation results in a simulated SOA environment are presented to show the applicability of our approach.

Keywords-component; Service-Oriented Systems; Service Level Management; Automated Negotiation; Self-organisation

I. INTRODUCTION

Non-functional aspects of self-organising service-oriented systems at runtime, i.e. service levels between providers and consumers, are important to enable SLA-driven IT management with constant alignment of IT capabilities with business goals. This work proposes a negotiation-based approach to enable end-to-end SLM within service-oriented environments. The key characteristic of our approach is to utilise automated negotiation to propagate end-to-end service level requirements for business processes across all related technical components in support of a service-oriented application.

Hence, the remainder of this paper is organised as follows: Section II gives an overview on automated negotiation. Section III focuses on the negotiation model for automated SLA negotiation, while Section IV presents the experimental results from the implementation. At last, Section V summarises the work with an outlook.

II. RELATED WORK

In the context of SOA, research on SLM focuses mainly on providing frameworks that establish and enforce SLAs between service consumer and service provider. WS-Agreement [1] provides an extensible framework for specifying agreements between negotiating parties within SOA. Similar approaches for specifying and monitoring SLAs for Web services are e.g. Web Service Level Agreement (WSLA) as well as Web Service Management Network Agent (WSMN). Yan et al. [2] proposed an agent-based approach to facilitate negotiation of SLAs for service composition to achieve end-to-end QoS requirements. For coordinated negotiation between agents, they constructed a compatible negotiation protocol that is adjusted to the agent-based nature of their negotiation scenarios. These frameworks cited above focus on a comprehensive approach for creating and monitoring SLAs rather than on automated negotiation of SLAs at runtime. Hence, this work focuses on negotiation aspects of automated SLM in self-organising service-oriented environments.

III. SLA NEGOTIATION

This section introduces the model for automated negotiation in SOA, in particular the multi-round and iterated negotiation protocol as well as negotiation strategies.

A. Mathematical Model

The mathematical model introduced in this section is adapted from the work of Sierra et al [3] to model a bilateral negotiation between a service consumer c and its provider p (i.e. agent i ∈ {c, p}) on multiple issues j ∈ {1, 2, ..., n} of an SLA, such as availability, response time, etc. For each issue j, there is a continuous value range [min_j, max_j]. It is assumed that the ranges of both negotiating parties have overlaps. Furthermore, an offer sent from component a to component b in a negotiation thread at time t is denoted as x_{a→b}^t, where a, b ∈ {c, p}, t ∈ [1, t_{max}], and a ≠ b.

For each negotiation issued j, each agent i has a pre-defined utility function V^i_j : [min_j, max_j] → [0, 1] that maps assigned values of the issue to the continuous interval [0, 1] as follows:

\[
V^j_i(x_j) = \begin{cases} 
\left( \frac{\max_j - x_j}{\max_j - \min_j} \right)^\alpha & \text{if } j \text{ is decreasing} \\
\left( \frac{x_j - \min_j}{\max_j - \min_j} \right)^\alpha & \text{if } j \text{ is increasing}
\end{cases}
\] (1)

Increasing means that the utility increases if the issue’s value increases, such as availability for consumers; and decreasing means utility decreases if the issue’s value increases, such as service cost for consumers. \( \alpha \in R \) is a constant with \( \alpha \neq 0 \) and controls the increasing/decreasing behaviour of the utility function. Hence, an agent’s utility function \( V^j \) to estimate the quality of a given agreement \( x = (x_1, x_2, ..., x_n) \) is defined as weighted sum of utilities of all QoS issues, namely:

\[
V_j = \sum_{j=1}^{n} \omega_j \cdot V^j_i(x_j)
\]

with \( \sum_{j=1}^{n} \omega_j = 1 \).

B. Negotiation protocol

To propagate end-to-end service level requirements across the complete SOA-based system, we construct a coordinated and iterative negotiation protocol to facilitate negotiation activities across components in an SOA. The complete negotiation process is triggered by the business process, which gets external operational objectives in terms
of QoS as input. These external operational objectives specify the set of QoS issues as well as their value ranges for negotiation (e.g. minimal availability, or maximal cost).

The negotiation is multi-round. That is, a service consumer exchanges offers and counter offers with its service provider, until a mutually acceptable agreement is reached. Furthermore, the negotiation protocol is iterated. That is, if a mutually acceptable agreement is found between a consumer and its provider, the corresponding provider continues to negotiate in turn with its provider(s), before it commits to the SLA with its consumer. In this new negotiation process, the agreement negotiated with its consumer is used as operational objective in the negotiation process with its providers.

By following this iterated negotiation protocol at runtime, the negotiation initialised by a business process is propagated recursively across the complete system, until infrastructural components are reached that have no further providers. Due to the limited number of components that are involved in a service-oriented system and a predefined time limit for negotiation, such an iterated negotiation process terminates always after a certain number of iterations. At the end of such a chained negotiation process, either each consumer/provider pair in the system has a mutually accepted and confirmed SLA; or there are no established SLAs between related components along the consumer/provider chains.

C. Negotiation Space

Given a set of overall service level requirements, a consumer has to determine the appropriate portions of non-functional requirements for each service provider, so that they can fulfil the non-functional requirements collectively. However, this task is not trivial. Theoretically, there are an infinite number of possibilities to decompose the non-functional requirements for each provider. Therefore, it has to incorporate the nature of each service provider into the decomposition process, so that the resulting requirements for each service provider comply with its behaviour pattern, in particular, characteristics of the corresponding QoS parameter, composition patterns specifying relationships between a service consumer and its providers, and runtime behaviour of a particular service provider or service type.

This work uses BPMN’s definitions for diverging gateways to determine composition patterns of a service consumer with its providers. Table 1 shows the correlation between BPMN definitions and the compositions patterns defined in this work.

<table>
<thead>
<tr>
<th>BPMN Pattern</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>exclusive XOR</td>
<td>Among all service providers, only a single provider will be consumed, such as one out of two load-balanced Web servers.</td>
</tr>
<tr>
<td>inclusive OR</td>
<td>One or more service providers are involved to serve the service consumer, such as a Web service with two redundant DB servers.</td>
</tr>
<tr>
<td>parallel AND</td>
<td>All parallel service providers are crucial for the proper functionality of the consumer.</td>
</tr>
<tr>
<td>- SEQ</td>
<td>All service providers are consumed one after another.</td>
</tr>
</tbody>
</table>

That is, if provider-specific historical information is available, such information is used; otherwise, the historical information of the corresponding service type is used as reference value. Furthermore, let \( x_c \) be the QoS requirement for the consumer, and \( x_1 \) be the decomposed QoS value for the provider \( i \). In the following, we investigate one representative QoS parameter - cost. Decomposing other QoS parameters works in a similar way.

The composite cost of a service consumer is the sum of service cost of all involved service providers. Since with each of the patterns OR, AND, and SEQ, a service consumer invokes all service providers, the composite cost satisfies the following conditions:

\[
\begin{align*}
\{ x_1[j] + x_2[j] + \ldots + x_n[j] = x_c[j] \\
x_1[j]/h_1[j] = \ldots = x_n[j]/h_n[j] \\
\Rightarrow x_i[j] = h_i[j]/\sum_{k=1}^{n} h_k[j] \cdot x_c[j].
\end{align*}
\]

For the pattern XOR, where only a single service provider is invoked by the consumer, it applies in general that \( x_i[j] \leq x_c[j] \) with \( i \in \{1,2,\ldots,n\} \). Hence, additional information is required in order to estimate \( x_i[j] \) more precisely. Based on the historical information collected at runtime, it is possible to estimate the absolute probability a service provider \( i \) is invoked at runtime. Let it be \( f_i \) with \( \sum_{k=1}^{n} h_k[j] = k \), then we get:

\[
\begin{align*}
\{ f_1 \cdot x_1[j] + f_2 \cdot x_2[j] + \ldots + f_n \cdot x_n[j] = x_c[j] \\
x_1[j]/h_1[j] = \ldots = x_n[j]/h_n[j] \\
\Rightarrow x_i[j] = \frac{h_i[j]}{\sum_{k=1}^{n} h_k[j] \cdot f_k \cdot x_c[j]}.
\end{align*}
\]

D. Negotiation strategies

A much more interesting aspect of automated negotiation is negotiation behaviour of a technical component at runtime, i.e. how a component responds to incoming SLA offers. In general, negotiation behaviour is determined by Conceding strategies that specify in case of a counter offer the utility of the offer, and Trade-off strategies that allow an agent finding optimal counter offers by taking the behaviour of its negotiation partner into consideration.

1) Conceding Strategies

Due to the fact that both negotiating parties have conflicting interests on the negotiation issues and both of them start with their respective optimum (i.e.
\[ V^m(x_{a\rightarrow b}) = 1 \] in the negotiation, both parties have to concede in the course of negotiation, so that a compromise can be reached at all. The major aspect of conceding strategies is to find the extent of concession in each step. An adequate conceding strategy should produce appropriate concession pressure on the agent’s negotiation behaviour, depending on e.g. the time left until a given deadline. In general, we distinguish between time-dependent tactics that assess the extent of concession for issue \( j \) in relationship with negotiation time, and behaviour-dependent tactics that assess the extent of concession according to the negotiation behaviour of an agent’s negotiation partner. At runtime, an agent can combine both tactic types to use several criteria simultaneously to support its decision-making process.

2) Trade-off Strategies

Conceding strategies do not optimise outgoing offers. In addition, for a given utility value, there are an infinite number of possible value assignments for outgoing offers. For instance, without changing an offer’s utility, an agent can increase the cost to get shorter response time in the offer. Hence, an agent can find trade-offs that are more attractive for the counter party to increase possibility of an agreement.

The obstacle to find optimal trade-offs is missing information of an agent about its negotiation partners, such as their preferences on negotiation issues. Agents can only use offers proposed by its negotiation partner so far for decision making. Hence, a reasonable way is to find counter offers that are similar to incoming offers from negotiation partners. To this end, we propose to use normalised Euclidean distance between two offers in the negotiation space. The function in SA is the normalised distance(s) between the offers with minimal distances to them.

\[
D_{\text{norm}}(x_{b\rightarrow a}^{t-1}, x_{a\rightarrow b}^{t}) = \sqrt{\sum_{j=1}^{n} \left( \frac{x_{b\rightarrow a}^{t-1} - x_{a\rightarrow b}^{t} - x_{b\rightarrow a}^{t} - x_{a\rightarrow b}^{t}}{x_{b\rightarrow a}^{t} - x_{a\rightarrow b}^{t}} \right)^2} \tag{2}
\]

This formula takes both initial offers \( x_{b\rightarrow a}^{t} \) proposed by the consumer \( c \) and \( x_{a\rightarrow b}^{t} \) by the provider \( p \) as reference points to eliminate the influence of different scales of QoS parameters on the estimation.

On the base of (2), the following different strategies can be applied to find trade-offs:

- **Trade-off strategy 1**: an agent aligns itself to the two initial offers \( x_{c\rightarrow p}^{0} \) and \( x_{p\rightarrow c}^{0} \), and tries to find trade-offs with minimal distances to them.
- **Trade-off strategy 2**: an agent aligns its search to the last incoming offer and tries to find trade-offs with minimal distance to the offer.
- **Trade-off strategy 3**: an agent combines the second strategy with given business objectives. For example, for the given business objective “quick response”, an agent finds at first trade-offs with response time as low as possible, then finds trade-offs with minimal distance to the last incoming offer by keeping the value of response time determined in the first phase unchanged.

Furthermore, Simulated Annealing (SA) is used to find desirable trade-offs in the negotiation space. The evaluation function in SA is the normalised distance(s) between the trade-off and reference offers in the negotiation space. The neighbourhood of a given offer is generated by randomly selecting an issue, changing its value, and adjusting another randomly selected issue to compensate the utility change.

IV. Evaluation

To assess the proposed approach, we implement the framework in an SOA simulation environment [4] to show the feasibility and performance of the approach. For simplicity, three QoS parameters are considered in the simulation: cost, availability, and response time. Furthermore, each management agent utilises the utility function (1) with \( \alpha = 1 \). The negotiation deadline is set to 50 simulation ticks. Fig. 3 illustrates the experimental results between two agents with the following simulation settings:

- **Group A**: evaluate conceding strategies only
- **Group B**: evaluate trade-off strategy 1
- **Group C**: evaluate trade-off strategy 2
- **Group D**: evaluate trade-off strategy 3

The charts in Fig. 3 show the change of agents’ utilities in the course of negotiation, and the ones on the right show the offers proposed by the agents in the negotiation space. For simplicity, the offers are projected on the plane spanned by response time and cost. The stars in the figures indicate the behaviour of the service consumer, and the circles show the behaviour of the service provider. Furthermore, diamonds show the behaviour of agents using the same experiment setting, but without trade-off search. The rectangles in the planes are the projections of the negotiation spaces of corresponding agents on the plane.

In the utility figures, it is obvious that in order to reach a compromise, both agents have to concede by giving up a certain extent of utility in each step and moving away from their optimum towards the optimum of their negotiation partners. Both behaviour patterns provide the prerequisite for a mutually accepted agreement by negotiation.

In the experiment group A, an agent cares only about the utility of incoming offers and the extent of utility it is going to give up. QoS values in incoming offers are not considered by this process. Therefore, as shown in the chart A2, each offer is located on the diagonal of the negotiation space. It shows that conceding strategies take the intentions of the negotiation partner barely into consideration.

The trade-off strategies incorporate external information, such as negotiation history, or business objectives, to find more promising offers. In the experiment group B, an agent aligns its trade-off search to the initial offers of both agents. This ensures that an agent can provide offers with respect to the optimum of its negotiation partner. As shown in the chart B2, both agents tend to place their offers on the line connecting the initial agreements. If they are out of the corresponding negotiation space, the agents put the offer along the boundaries towards the joint negotiation space. Obviously, this trade-off strategy increases the possibility of reaching an agreement. But in favour of this increased possibility, both agents need a longer negotiation thread with slow convergence.

Hence, in the experiment group C, the trade-off strategy is changed to align trade-off search to the last incoming offer.
from the negotiation partner. This change brings in more dynamics in the trade-off search, so that an agent can propose a more attractive offer to its partner in dependence on the most current offer of the partner. As seen in chart C2, it is obvious that by using this strategy, both agents come earlier to a mutually accepted compromise. Without this strategy, both agents have not even reached the joint negotiation space using the same time. The same effect can be observed by comparing the utility charts A1 and C1. Altogether, in comparison to pure conceding strategies this strategy combining conceding strategies helps agents to reach a compromise more quickly.

In the last experiment group D, the trade-off strategy with alignment to incoming offers is combined with external business objectives. The charts of group D show the behaviour of the agents with an emphasis on response time. Each agent tries at first to exhaust its reserve on less prioritised QoS parameters – e.g. availability and cost in the chart, before they begin to concede in the prioritised QoS parameters. This strategy allows an agent to reach a compromise in compliance with external business objectives. Furthermore, as chart D1 shows, this strategy also allows a quicker convergence of negotiation. In comparison to other strategies, both agents have higher utilities at the end.

However, it is noteworthy that the convergence of a negotiation process depends strongly on agents’ negotiation behaviour, in particular their negotiation strategies and negotiation spaces. For the same simulation scenario, dynamic trade-off strategies with alignment to previous incoming offers have a substantially higher possibility for convergence than pure conceding strategies.

V. SUMMARY AND OUTLOOK

This paper presents a negotiation-based approach to support end-to-end SLM in a SOA environment. The negotiation model is implemented and empirical studies are done. The experimental results show the applicability of our approach in a service-oriented environment. Our future work focuses on refining the negotiation model with respect to efficiency. Secondly, we plan to extend negotiation strategies using learning, so that an agent can change its negotiation strategies adaptively depending on the behaviour of its negotiation partner.

REFERENCES


