Introducing Preferences over NFPs into Service Selection in SOA*

Christian Schröpfer[2], Maxim Binshtok[1], Solomon Eyal Shimony[1], Aviram Dayan[1], Ronen Brafman[1], Philipp Offermann[2], and Oliver Holschke[2]

[1] Dept. of Comp. Sci. Ben-Gurion University Beer-Sheva Israel
[2] Dept. of Comp. Sci. and Electrical Eng. Technische Universität Berlin

Abstract. When implementing a business or software activity in SOA, a match is sought between the required functionality and that provided by a web service. In selecting services to perform a certain business functionality, often only hard constraints are considered. However, client requirements over QoS or other NFP types are often soft and allow tradeoffs. We use a graphical language for specifying hard constraints, preferences and tradeoffs over NFPs as well as service level objectives (SLO). In particular, we use the TCP and UCP network formalisms to allow for a simple yet very flexible specification of hard constraints, preferences, and tradeoffs over these properties. Algorithms for selecting web services according to the hard constraints, as well as for optimizing the selected web service configuration, according to the specification, were developed.

1 Introduction

Scientists and practitioners emphasize the potential of SOA (service-oriented architecture) for reconciling business requirements and IT infrastructures. SOA definitions range from a technology-driven approach to a management school approach on how to run an enterprise. A proper description of services is a fundamental precondition for a functioning SOA. Several advantages of SOA, like agility, fast implementation times, cost efficiency, high software quality, and good functional support of needed business functionality, become effective only if reuse of the services is improved and the business can rely on the quality of the IT functionality delivered. Allowing non-functional selection among services delivering the same functionality in a transparent way could even strengthen the quality of the service portfolio via competition between the service providers.

When selecting web services to perform a certain business functionality, often only hard constraints are considered, e.g., response time should be lower than 100ms. However, in reality, client requirements over QoS or other NFP types are often soft and allow tradeoffs. For example, a client may wish for short response time, but may be willing to temporize if another service is much cheaper. Or she may be willing to allow that, but only in certain contexts. A non-trivial set of requirements for the tradeoffs can exist.

* Funded by Deutsche Telekom Laboratories at Ben-Gurion University and at Technische Universität Berlin.

E. Di Nitto and M. Ripeanu (Eds.): ICSOC 2007 Workshops, LNCS 4907, pp. 68–79, 2009.

A simple way to implement optimization over NFP configurations is for the user to specify hard constraints over NFPs, get a list of service configurations that match, and then select among them manually. However, this scheme is undesirable for several reasons. First, the number of possible configurations may be large, making manual selection of unranked configuration a bad idea. Second, the inverse may be true, the client can define constraints which are too restrictive, and get no candidates. Although repeating the cycle several times can result in a reasonable selection (say through a negotiation process), this process is undesirable as it may be too time consuming.

In a service-oriented architecture, business processes and software systems are built out of orchestrated business or software activities. The work described in this paper is part of such a larger system, and each service integration process involves matching and optimization with respect to more than one activity and web service. Since the configuration process is part of a larger process, involving intermediate agents, it is even more important to have as few repeated cycles of constraint redefinition and selection. It is better if the tradeoff can be specified formally, and an automated ranking or selection can be made in a single pass.

In this work, we use a graphical language for specifying hard constraints, preferences and tradeoffs over NFPs and service level objectives (SLO). In particular, we use the TCP and UCP network formalisms to allow for a simple yet very flexible specification of both hard constraints, preferences, and tradeoffs over these properties. Algorithms for selecting web services according to the hard constraints as well as for optimizing the selected web service configuration according to the specification were developed. The scheme has been implemented as part of a system for web service integration.

Our research methodology follows the guidelines of design research [1]. Building blocks of our concept are: methods describing the general sequence of the steps to be performed, how the SLOs and preferences are modeled, and what algorithms can be used for ranking the services from a non-functional point of view. We expect our preference- and NFP-based approach of finding services for business process steps to yield better matching results than the existing algorithms, i.e. that there are cases in which our approach incurs less overall effort than the negotiation-based selection of the services. The first step is also to show that the concept is usable, which we have done by developing a prototype and performing test scenarios. Comparison with other tools and user tests are planned for future research.

2 Background: TCP and UCP Networks

A major issue in this paper is compact and intuitive representation of preferences and utilities. Several schemes exist, briefly discussed in the related work section. Of these models, we use both TCP networks [2] and a simplified version of UCP networks [3]. These networks represent preferences (resp. utilities) of a set of *outcomes*, each being an assignment of values to a set of variables. Note that

the set of outcomes is exponential in the number of variables, but these use an independence semantics for a very compact representation.

2.1 TCP Networks

We illustrate TCP networks through an example of selecting a flight, the variables may be *Airline, Departure time, Stop-over, Ticket class* etc., where *Airline* has the domain {*British Airways, KLM,...* }, *Departure time* has the domain {*morning, noon, evening, night*}, etc. Each complete assignment to the set of variables specifies an outcome – a particular flight configuration.[1]

The representation is compact due to a preferential independence assumption: in our flights example, the client may assess *Airline* to be preferentially independent of *Departure time*. This could be the case if the user always prefers to fly British independent of the departure time. As preferential independence is usually too strong a property, a notion of conditional preferential independence is also used. For example, the user may always prefer to fly business class given that this is a KLM flight (independent of other variables), but prefer economy class if this is a British Airways flight.

Relative Importance: In many typical scenarios, we have auxiliary or user constraints that prevent us from providing the user with the most preferred (unconstrained) outcome. A simple and common example is that certain flight combinations may be unavailable, e.g. there may be no business class KLM flights to the USA in the morning. In such cases, it is important to know which attributes the user cares about more strongly, and to try to maintain good values for these attributes, compromising on the others.

In our flights example, suppose that the attributes *Stop-overs* and *Ticket class* are mutually preferentially independent. We might say that *Ticket class* is more important than *Stop-overs*, e.g, because we would rather temporize and have to wait some time in the airport, as long as we can get a good sleep on the flight due to our business class seat.

Relative importance can be refined naturally to a notion of *conditional relative importance*. For instance, if the relative importance of *Ticket class* and *Stop-overs* depends on departure time (see detailed example below).

TCP-nets: The TCP-net (for *Tradeoff-enhanced Conditional Preference-nets*) model is a graphical model that encodes conditional relative importance statements, as well as the conditional preference statements. Each node X in a TCP-net stands for a variable, and is annotated with a *conditional preference table* (CPT). The graph has three type of arcs: directed conditional preference edges, "strongly directed" importance arcs (i-arcs), and undirected conditional importance arcs (ci-arcs). The CPT for a variable X associate preferences over $\mathcal{D}(X)$ for every possible value assignment to the parents of X (denoted $Pa(X)$), where only preference arcs are considered in determining $Pa(X)$.

[1] Alternatively, one could consider variables from the web-service NFP domain, such as performance or price (see Section 3).

In addition, in TCP-nets, each undirected arc is annotated with a *conditional importance table* (CIT). The CIT associated with such an arc (X, Y) describes the relative importance of X and Y given the value of the corresponding importance-conditioning variables $\mathbf{Z}$.

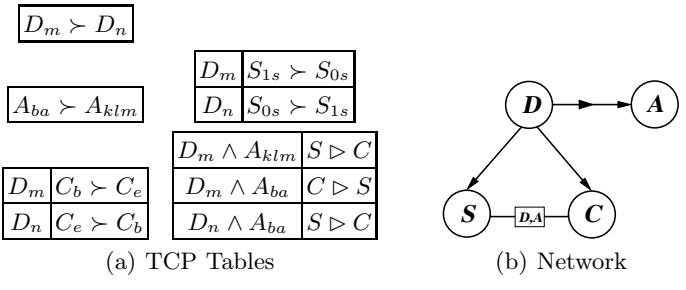
*Example 1 (*TCP-net Example: Flight to the USA). Figure 1 illustrates a TCP-net, describing my preference over the flight options to a conference in the USA, from Berlin. We consider 4 flight parameters (variables):

Airline. The variable A represents the airline. I prefer to fly with British Airways (A_{ba}) than with KLM (A_{klm}).

Departure Time. The variable D distinguishes between morning/noon (D_m) and evening/night (D_n) flights. I prefer a morning/noon flight.

Stop-over. The variable S distinguishes between direct (S_{0s}) and indirect (S_{1s}) flights, respectively. On day flights I am awake most of the time and, being a smoker, prefer a stop-over in Europe. However, on night flights I sleep, leading to a preference for direct flights, since they are shorter.

Ticket Class. The variable C stands for ticket class. On a night flight, I prefer economy (cheaper...) (C_e), while on a day flight I prefer business class (C_b).



(a) TCP Tables (b) Network

Fig. 1. "Flight to the USA" TCP from Example 1

The TCP-net in Figure 1 captures all these preference statements, and the underlying preferential dependencies, as well as the relative importance relations between some parameters of the flight. First, there is an i-arc from D to A, because a suitable flying time is more important to me than getting the preferred airline. Second, there is a ci-arc between S and C, where the relative importance of S and C depends on the values of D and A:[2]

1. On a KLM day flight, an intermediate stop in Amsterdam is more important to me than flying business class.
2. For a British Airways night flight, the fact that the flight is direct is more important to me than getting a cheaper economy seat.

[2] For clarity, the ci-arc in Figure 1(b) is *schematically* labeled with its importance-conditioning variables D and A.

3. On a British Airways day flight, business class is more important to me than having a short intermediate break.

The semantics of a TCP-net is straightforward, defined in terms of the set of strict partial orders consistent with the set of constraints imposed by the preference and importance information captured by this TCP-net. For an exact definition of the semantics see [3].

In the above example, the TCP net semantics imply that the most preferred outcome is: morning/noon flight, 1 stop-over, business, and flying BA. The next-best outcome is the same as above, except with no stop-overs, etc. A simple top-down backtracking algorithm (described later on) can be used to enumerate outcomes in order of non-increasing desirability.

2.2 UCP Networks

Although in many cases qualitative preferences are sufficient, there are still cases where a user might wish to specify quantitative values, i.e. utilities. However, rather than having to specify an exponential number of utilities (one for each possible outcome), one can use the structure of the preferential dependencies used in TCP-nets to compactly specify the utilities. Consider a TCP network with just the conditional preference arcs (i.e. without importance or conditional importance arcs). Now, if we replace the conditional preference tables (CPT) by similar tables that contain utility values (i.e. conditional utility tables, CUT) instead of orderings, we obtain the compact representation of utility functions known as (simplified) UCP networks.

Figure 2 provides an example UCP network, specifying utilities similar (though not equivalent) to the TCP network version of the example. It is evident that such a network is roughly as easy to specify as the respective TCP network. Its semantics is rather simple: given an outcome, its utility is the sum of the utilities for each attribute values as specified by the utility tables.

Observe that importance tradeoffs can be enforced in UCP networks by having larger value differences in tables for the more important variables. In this example, the highest-utility flight configuration would be a morning/noon flight (100) by British Airways (10), 1-stop (20), business class (20) for a total utility of 150. The 2nd best would be the same configuration, except flying with KLM.

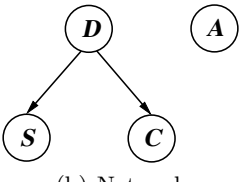
D_m	D_n
100	0

	S_{0s}	S_{1s}
D_m	0	20
D_n	20	0

A_{ba}	A_{klm}
10	5

	C_b	C_e
D_m	20	0
D_n	0	10

(a) UCP Tables



(b) Network

Fig. 2. "Flight to the USA" UCP-net

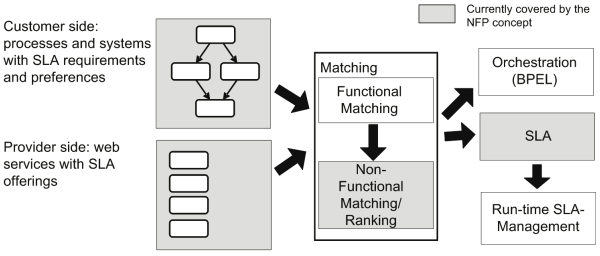


Fig. 3. System overview (part related to handling of NFPs)

3 Handling NFPs in Our System

Figure 3 describes the overall scheme for handling NFPs in our system. The provider uses SLO structures to service capabilities, and combines them into SLA offerings of a certain service class. The consumer describes his SLO requirements and preferences. After finding the functional matches, the NFP matching algorithm finds whether the service with its service class given its SLO capabilities complies with the SLO requirements, and then ranks the services according to the preferences specified by the customer. Once the final service selection has been authorized by a human, the SLA is constructed out of the information from the SLA offering of the selected service class of the service. This SLA can then be used to monitor run-time compliance of the service.

3.1 Provider-Side Description

On the service provider side the service capabilities are described using SLA offerings. An SLA offering contains a set of SLO offerings that state conditions over NFPs. The structure of the SLO offerings is depicted in Figure 4. Figure 4 also contains an example for a ResponseTime SLO offering for the service class "Gold": "ResponseTime is guaranteed to be less than 100ms in 95% of the cases if the TransactionRate caused by the customer is lower than 10000 per second." Note that although in our implementation we have limited the structure in order to preserve simplicity of use (by a business analyst) and low implementation complexity, there is no *inherent* reason for this limitation. The NFPs that can be used are defined in an extensible NFP ontology, formalized in XML, since customer and provider need to talk the same language. An extract from the initial list of NFPs is depicted in Figure 5. Further categories and NFPs are: Security (Authentication, Authorization, Encryption, Integrity, Non-Repudiation), Cultural (Language), Legals (SOXCompliance, BafinCompliance, BaselIICompliance), Organizational (PartnerList), Service Usage (GUISimplicity, InputFlexibility, OutputFlexibility), Trust(Customer Rating, Experience, Membership), and Price. This list can be extended in our system by simply changing the (reconfigurable) ontology. To lower the effort of defining SLAs on

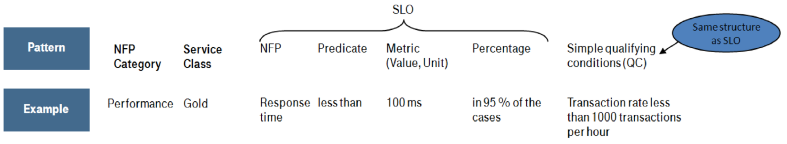


Fig. 4. Structure of an SLO offering/requirement and example SLO offering for the service class "Gold" regarding the NFP "ResponseTime"

NFP category	NFP			Default values								
	Name	Unit	Data type	Platinum				Gold				Silver
				Value	P	%	QC	Value	P	%	QC	Value
Availability	DateTime	Default set	Enum	24/7	>=	-	-	10/7	>=	-	-	
Reliability	MTTR	hrs	Floating point	0.2 hrs	<=	-	-	0.5	<=	-	-	
	MTBF	hrs	Floating point	6360 hrs (=1 yr)	>=	-	-	4000 hrs	>=	-	-	100
	Robustness	Default set	Enum	very high	>=	-	-	high	>=	-	-	medi
	UpTime	Percent	Floating point	99,99%	>=	-	-	99.9%	>=	-	-	99
	Correctness	Percent	Floating point	99,99%	>=	-	-	99.9%	>=	-	-	99
Performance	Delay	s	Floating point	0.01 s	<=	99	-	0.1	<=	97	-	
	ResponseTime	s	Floating point	0.1	<=	99	-	0.5	<=	97	-	
	Throughput	kB/s	Floating point	-	-	-	-	-	-	-	-	-
	TransactionRate	#Transactions/s	Floating point	-	-	-	-	-	-	-	-	-

Fig. 5. NFPs, service classes, and default values (extract)

the consumer and provider side, there are default values for the service classes that can be overwritten in the GUI. The SLA offering regarding one service at a certain service class is the basis for the SLA between provider and customer that gets monitored during runtime.

3.2 Consumer-Side Description

The consumer describes his service level requirements and preferences. The service level requirements for one software component consist of several SLO requirements. Preferences can be expressed over these SLOs or on specific values of NFPs directly. The SLO requirements used on the service consumer side have the same structure and formalization as the ones on the provider-side. Yet, there is one difference. While the service provider can define several service classes with different capabilities for one service, the service consumer expresses one set of requirements plus the preferences. An example SLO requirement is: "ResponseTime (RT) should be less than 100ms in 95% of the cases if the TransactionRate (TR) caused by the customer is lower than 1000 per second." Using the SLO inside an SLO node of the preference model allows for richer expression of the needs of the customer.

Either TCP or UCP networks (depending on whether the user wishes to specify qualitative or quantitative preferences) can be used in our system to represent both hard constraints and preferences (or utilities) over non-functional properties and over SLO statements. A regular TCP (resp. UCP) network node encodes some property over which we express preferences (resp. utilities). In addition,

we add special nodes that can allow us to express context. Thus, we have three types of nodes (see Figure 6):

1. SLO nodes (dashed ovals in figure). These have binary domain, and each stands for a previously defined SLO. A value of "true" would signify that a web service configuration can support the SLO statement.
2. Normal NFP nodes (solid ovals). These have arbitrary domain, depending on the NFP type. For example, an "Interface Language" node would have all possible interface languages in its domain.
3. Auxiliary nodes (dotted ovals). These represent auxiliary variables, usually for allowing dependence on arbitrary context, rather than NFPs.

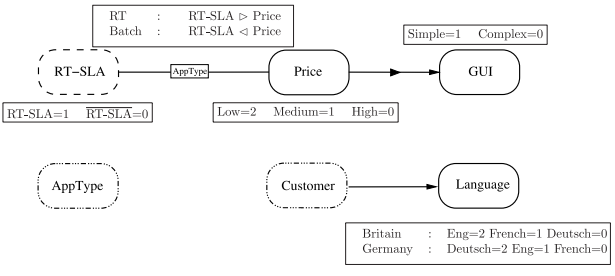


Fig. 6. Preferences over NFPs example using TCP-net

Rather than provide preferences over domain values, the CPTs contain numbers, where a higher value stands for a more preferred value. For example, preference over the interface language NFP (Language node in figure) depends on the auxiliary variable "customer nationality" (Customer node in figure). The values in the first row state that for British customers, English is the most preferred interface language for the web service. The strongly directed edge between Price and GUI means that the client prefers a better price than a better GUI. Finally, the undirected (interrupted) edge between RT-SLA and Price (with the auxiliary AppType variable as a conditioning variable) means that meeting the RT-SLA is more important than price if the application type is real-time, otherwise price is a more important attribute. Using the TCP-net semantics, the best web service configuration (if available), for a real-time application and a German client, would be that the RT-SLA be met, that the price be low, a simple GUI, and German as an interface language. If the only low-priced service available does not meet the RT-SLA, the next-best configuration would be a higher priced configuration that does. However, for a non-RT application, we would prefer a configuration that does not meet the RT-SLA as long as the price is low.

Using numbers in the table, allows us to represent both utilities and hard constraints with no change to the system. A value of $-\infty$ denotes a value

completely disallowed. To represent quantitative preferences, the numbers are simply treated as utility values by the ranking algorithms (see below).

4 Matching and Ranking Algorithms

After specification of both the functional and non-functional requirements (including preferences) have been specified a matching algorithm attempts to find the best match (both web service operation and its configuration) to perform the activity. In our system the functional match is first performed (this part is beyond the scope of the paper), resulting in a list of web service operation configuration candidates. Our goal is now to filter and rank the configurations according to constraints and preferences over the non-functional properties. Although both TCP and UCP networks use the same interface for defining the network, due to the different semantics of the networks, each type of network requires a different optimization algorithm.

UCP Ranking: The UCP ranking algorithm simply finds the utility value of each web service configuration offered, and sorts the services according to utility values, discarding any with infinite negative utility. Since only conditional preference edges (CP) are important, all other edges are dropped.

TCP Ranking: The TCP ranking algorithm is (topologically) top-down backtracking algorithm over web service configurations. Pruning is achieved if at a certain branch no web service configuration consistent with the current partial variable instantiation exists in the web service offering list.

4.1 Testing the Compliance of SLA Nodes

Part of the above mentioned algorithms is to find out, whether the service considered would fulfill the SLO requirements mentioned in the SLO nodes. For this, each statement in the SLO node of the preference model is tested with a simple algorithm that directly compares the predicate and values (enumerated and free ones) as well as the qualifying condition of the SLO offering and SLO requirement referring to the same NFP.

5 Implementation

The proposed concept has been shown in a prototype that consists of the following components: service repository (XML based on NFP XSDs), SLA definition tool (Portal application), preference modeling tool (Java), functional matcher (not part of the NFP concept), and non functional matching and ranking component with SLA definition capability. Figure 7 shows the GUI that is used for the SLO offerings definition on the service provider side and for the SLO requirements definition on the service consumer side. Our system allows for default characteristics (including preference specifications) for different client classes. Mapping NFP ontology between provider and client side is also possible in our

Algorithm 1. WS-TCP-RANKING($G, WS, limit$)

```
    G — topologically sorted preference model
    WS — set of matched WS-es
    limit — limit on maximal number of WS-es to output for ranking
 1: wsHist[0] ← WS
 2: i ← 1
 3: outCnt ← 0
 4: while (i > 0) and (outCnt < limit) do                    ▷ loop on nodes (attributes)
 5:     while G[i].setNextBestValue() do                     ▷ loop on values
 6:         WStmp ← wsHist[i-1]
 7:         for each w ∈ WStmp do                            ▷ WS filtering loop
 8:             if not(w.hasAttrValue(G[i].name,G[i].currValue)) then
 9:                 WStmp ← WStmp\{w}
10:         if not (WS = ∅) then
11:             if i = G.size() then                         ▷ full assignment to pref. model
12:                 output: WStmp + attr. assignments trace
13:                 outCnt ← outCnt + |WStmp|
14:             wsHist[i] ← WStmp
15:             i ← i + 1
16:     G[i].setBestValue()                                  ▷ reset the domain before backtrack
17:     i ← i − 1
```

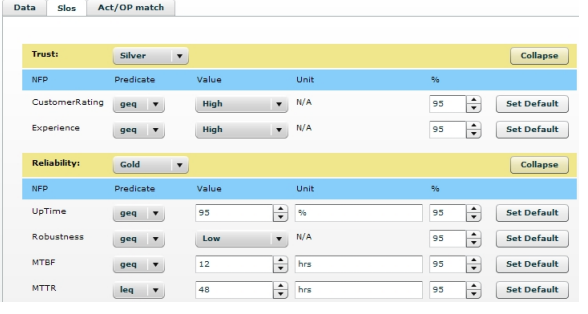


Fig. 7. GUI for defining SLO offerings and SLO requirements

system. Both of these features are handled by the system as a whole, rather than by the NFP tools directly.

6 Discussion

6.1 QoS-Based Service Description, Discovery, and Selection

Specification of QoS characteristics is an important topic in the area of IT systems. The existing standards can be grouped according to their main focus: software design/process description (e.g. UML Profile for QoS and QML – QoS Modeling Language [4]), service/component description (e.g. WS-Policy), and SLA-centric approaches (e.g. WSLA – Web Service Level Agreements [5], WSOL – Web Service Offerings Language [6], SLAng – Service Level Agreement definition language [7], and WS-Agreement [8]). The SLA-centric approaches usually are based on a provider-consumer scenario and thus related to our work. They consider SLA negotiation, specification, and SLA management.

A common way of performing QoS-aware matching is a two-step approach of functional and non-functional matching as proposed by METEOR-S, Grønmo and Jaeger [9], or in "Semantic WS-Agreement Partner Selection" [10]. Maximilien et al. uses an agent-based system [11]. Tian proposes WS-QoS, a comprehensive framework for considering QoS in SOA by extending UDDI and WSDL [12]. We are also using a two step-approach for matching business activities or system functionality with services. However, the novelty in our work are the preferences that are used to consider soft constraints.

In addition to looking at existing work, in [13], Dobson derives the requirements for a QoS concept in SOA: extensibility, XML, complex specifications, failure and non-compliance, and integration into environment. These requiremements are covered by our approach although the failure and non-compliance is not explicitly covered in this paper.

6.2 Related Approaches to Preference Specification

We briefly mention some publications most closely related to our representations. Languages for representing constraints, such as constraint networks [14] are a well-known scheme for graphically representing constraints. In recent years, these have been generalized to weighted constraints and to soft constraints. The latter bear some similarity to TCP networks (see [15]).

Representations for qualitative preferences also exist, but the first graphical model for such has been the CP-networks model [3] a precursor of TCP-networks. The TCP networks are also closely related to UCP networks, while UCP networks can be seen as a graphical notational variant of a generalized form of a generalized version of additive utility functions known as "generalized additive utility" (GAI) functions [16].

While a method similar to that done in this paper could have used some of the representations in the other related work, we feel that the graphical representation provided by TCP and UCP networks allowed us to combine preferences and constraints rather painlessly.

7 Conclusion

The paper presents an integrated approach of how formalized SLAs and preferences over NFPs can be used to improve the selection of services for business processes and software systems considering both hard and soft constraints. It describes a new approach compared to the existing work which only considers hard constraints.

Although we show that the approach works and is very useful in our example, the concrete comparison with other tools regarding the matching results and the user tests remain for future work. The user tests would include empirical tests on whether the additional effort necessary for doing the non-functional service description including specifying the preference model for a customer is justified by the added value.

References

1. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information system research. MIS Quarterly (2004)
2. Brafman, R.I., Domshlak, C., Shimony, S.E.: On graphical modeling of preference and importance. Journal of AI Research 25, 389–424 (2006)
3. Boutilier, C., Bacchus, F., Brafman, R.I.: UCP-networks: A directed graphical representation of conditional utilities. In: Proceedings of Seventeenth Conference on Uncertainty in Artificial Intelligence, pp. 56–64 (2001)
4. Frolund, S., Koistinen, J.: Quality of service specification in distributed object systems design. QML of HP Laboratories (1998)
5. Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web services level agreement (wsla) language specification (2003)
6. Tosic, V., Patel, K., Pagurek, B.: WSOL - web service offerings language. In: Bussler, C.J., McIlraith, S.A., Orlowska, M.E., Pernici, B., Yang, J. (eds.) CAiSE 2002 and WES 2002. LNCS, vol. 2512, pp. 57–67. Springer, Heidelberg (2002)
7. Lamanna, D.D., Skene, J., Emmerich, W.: Slang: A language for defining service level agreements (2003)
8. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web services agreement specification, ws-agreement (2005)
9. Grønmo, R., Jaeger, M.C.: Model-driven methodology for building qoS-optimised web service compositions. In: Kutvonen, L., Alonistioti, N. (eds.) DAIS 2005. LNCS, vol. 3543, pp. 68–82. Springer, Heidelberg (2005)
10. Oldham, N., Verma, K., Sheth, A., Hakimpour, F.: Semantic ws-agreement partner selection. In: International World Wide Web Conerence Committee (IW3C2), Edinburgh, Scotland. ACM Press, New York (2006)
11. Maximilien, E.M., Singh, M.P.: A framework and ontology for dynamic web services selection. IEEE Internet Computing 08(05), 84–93 (2004)
12. Tian, M.: QoS integration in Web services with the WS-QoS framework. PhD thesis, Freie Universität Berlin (2005)
13. Dobson, G.: Quality of service in service-oriented architectures. Dependability Infrastructure for Grid Services Project (2004)
14. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, London (1993)
15. Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G.: Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. Constraints 4(3), 275–316 (1999)
16. Bacchus, F., Grove, A.: Graphical models for preference and utility. In: UAI 1995, pp. 3–10. Morgan Kaufmann, San Francisco (1995)