Dealing with Fixable and Non-fixable Properties in Service Matchmaking*

Octavio Martín-Díaz, Antonio Ruiz-Cortés, José Mª García, and Miguel Toro

Dpto. Lenguajes y Sistemas Informáticos
ETS. Ingeniería Informática – Universidad de Sevilla
41012 Sevilla, España – Spain
octavio@lsi.us.es, {aruiz,josemgarcia,migueltoro}@us.es

Abstract. In the context of service discovery, matchmakers check the compliance of service-level objectives from providers and consumers. The problem of bounded uncertainty arises if some property is non-fixable. In this case, the provider is not able to control the value it takes at runtime, so the eventual consumer must not have the choice to select a value and fix it, but only knowing the guaranteed range of values it may take. To the best of our knowledge, there does not exist any approach which deals with this scenario. Most matchmakers work as if all properties were fixable, and a few have assumed the contrary. In either case, the accuracy of their results is likely to be in question since there may be involved both fixable and non-fixable properties at the same time, and there may also exist dependencies between them. In order to improve the accuracy, we present a holistic approach to matchmaking under bounded uncertainty and propose constraint programming as our choice to deal with it, so that matchmaking is transformed into a quantified constraint satisfaction problem.

1 Introduction

Quality-aware enhancement of automated service discovery is one of the main challenges of service-oriented computing [13]. In a common scenario, administrators allow providers and consumers to publish their agreement offers (AO) in a repository. AOs specify both the service-level objectives (SLO) guaranteed by the provider to consumers, and the SLOs required by the consumer to providers. SLOs state assertions over service properties, or properties for short. In this context, matchmakers check the compliance, i.e. the existence of commonalities between the AOs from a provider and a consumer to determine if coming to a service-level agreement (SLA) between them is feasible. In general, SLAs regulate the execution of the services and provide guarantees to that end [2].

* This work has been co-supported by the European Commission FEDER and Spanish Govt. under CICYT projects Web-Factories (TIN2006-00472) and SETI (TIN2009-07366), and the Andalusian Administration under project Isabel (TIC-2533).

1.1 Fixability and Dependency

The fixability of a property is related to the fact that the provider is able to control the value it takes at runtime. If so, it may be declared as fixable, otherwise it must be declared as non-fixable. In turn, the controllability is related to the ability of the service to adapt itself in order to maintain the objectives which are being guaranteed, adjusting dynamically the use of common resources which are shared with other services [1, 11, 18].

As an example, consider a video streaming service whose provider guarantees a SLO such as `IMAGE` $\in$ `{320x200,640x480}` where `IMAGE` stands for *the size of the screen*. This property takes a value from a limited set of options, and the provider is usually able to control the requested value regardless of the environment of the service execution. Thus, the property may be declared as fixable, and the consumer has the choice to fix the value, say `IMAGE = 320x200`.

Consider the provider guarantees another SLO as `FRAMERATE` $\in$ `[22..25]` where `FRAMERATE` stands for *the frame ratio in images/sec to show a video*. In this case, the provider is not usually able to control the value it takes at runtime, being under the influence of the heterogeneous environment of the underlying network. Quality-of-service properties, such as bandwidth, performance, reliability, and availability, are usually dynamic and have an unpredictable nature [13]. This is said to be a scenario of *bounded uncertainty*. At best, the provider is only able to guarantee the range of values the property may take. Thus, the property must be declared as non-fixable, so that the consumer does not have the choice to fix it, but only knowing the guaranteed range of values it may take, e.g. `FRAMERATE` takes any value between 22 and 25 at runtime.

There may be also eventual dependencies between properties. Hereafter, we focus on whether non-fixable properties are dependent on fixable properties, so that the quality-of-service may vary according to the values the fixable properties take. As an example, consider the provider guarantees several SLOs as:

$$\texttt{IMAGE} \in \texttt{\{320x200,640x480\}}$$
$$\texttt{IMAGE = 320x200} \implies \texttt{FRAMERATE} \in \texttt{[22..25]}$$
$$\texttt{IMAGE = 640x480} \implies \texttt{FRAMERATE} \in \texttt{[20..23]}$$

where the `IMAGE` property is declared as fixable, and `FRAMERATE` is non-fixable. On the one hand, the consumer is allowed to fix a value for `IMAGE` by choosing between `320x200` and `640x480`. On the other hand, the value taken by `FRAMERATE` is not controllable at runtime, but it depends on the value taken by `IMAGE`, so that if `IMAGE = 320x200` then `FRAMERATE` ranges between 22 and 25 at runtime, else if `IMAGE = 640x480` then `FRAMERATE` ranges between 20 and 23 instead.

1.2 Inaccuracy of Matchmaking

Matchmakers may not work properly, mainly due to mis-assumptions taken regarding with fixability. Most approaches treat all properties as if they were fixable, and only a few assume the contrary, i.e. that all properties are non-fixable.

However, the accuracy of their results is likely to be called into question since there may be involved both fixable and non-fixable properties at the same time. This issue is related to the problem of bounded uncertainty, which arises if a non-fixable property is involved in guaranteed SLOs. According to [5], if bounded uncertainty is not properly dealt with, matchmaking may be inaccurate.

Thus, two scenarios are distinguished. Most usually, matchmakers have not taken fixability into account at all. As a matter of fact, they have optimistically defined matchmaking as if all properties were fixable, so that it is simply based on searching for common values between guaranteed and required SLOs. However, once a matching service is found, the consumer may be possibly using the service under non-satisfying values: if some of the properties are actually non-fixable, the provider can not control their values at runtime in order to avoid the values which do not satisfy the required SLOs. This is a matter of false positives.

As an example, consider a provider guarantees a SLO as the following `FRAMERATE ∈ [22..25]`, being the property known as non-fixable, and a consumer states a required SLO as `FRAMERATE ≥ 25`. Under the optimistic scenario, the matching is possible because there exists a common value, say `FRAMERATE = 25`. But this is a false positive because there are also some values which do not satisfy the required SLOs, any value in `[22..24]`, that may occur at runtime because the provider is not able to control them.

On the contrary, there are only a few matchmakers which pessimistically assume all properties to be non-fixable [7, 14]. As the provider does not control the values properties take at runtime, matchmaking is based on checking that every value satisfying the guaranteed SLOs also satisfies the required SLOs. However, the matchmaker may be turning down a service which can be used under satisfying values: if some of the properties are actually fixable, the provider will control their values at runtime, hence the values which do not satisfy the required SLOs will be completely avoidable. This a matter of false negatives.

As an example, consider the provider guarantees a SLO as the following `IMAGE ∈ {320x200, 640x480}`, being the property known as fixable, and a consumer states a required SLO as `IMAGE = 320x200`. Under the pessimistic scenario, the matching is surprisingly not possible (!) because there exists a value, say `IMAGE = 640x480`, which does not satisfy the required SLO. This is a false negative because the other value, `IMAGE = 320x200`, does satisfy the required SLO. As the provider does control it at runtime, the non-satisfying value is avoidable, and thus the provider's SLO should be considered as a match.

1.3 Holistic Approach to Matchmaking under Bounded Uncertainty

To avoid inaccurate results, matchmakers must take into account the fact that (1) there may be both fixable and non-fixable properties at the same time, and (2) there may be also eventual dependencies between fixable and non-fixable properties. To the best of our knowledge, there are quite a few approaches which have tackled uncertainty but from different perspectives: non-probabilistic set theory [5], historical records [6], fuzzy ranking [10], predictive and probabilistic models based on workflows [4, 17] or trustworthiness notions [12, 16].

In this paper, we present a holistic approach to matchmaking under bounded uncertainty, incorporating both fixability and dependency of properties, which improves the accuracy of the results. Our contribution is twofold:

- *Problem space.* We provide an abstract, rigorous definition of matchmaking by means of a geometrical interpretation. This makes easier the understanding of matchmaking and the comparison between the basic scenarios, and it justifies the necessity of a more complex, holistic vision of matchmaking.
- *Solution space.* We propose constraint programming as a feasible technique to solve the matchmaking under different scenarios. In particular, we emphasize the use of quantified constraint satisfaction problems [8] as a means to solve the matchmaking under bounded uncertainty.

The rest of the paper is structured as follows. First, Sec. 2 makes an overview of basic scenarios of matchmaking by means of a geometrical interpretation. Then, Sec. 3 presents our holistic approach to matchmaking under bounded uncertainty. Finally, Sec. 4 exposes our conclusions and future work.

2 Preliminaries

Geometrical interpretation. Because of the multiple ways by which AOs/SLOs can be specified, a geometrical interpretation based on set theory is given in order to abstract from the language to describe them [14]. As illustrated in Fig. 1(a), the AO property domains configure a space of points–of–agreement, or points for short, so that the AO is represented by means of a region in such space. In this region, each point is determined by assigning a value to every property, so that all SLOs in the AO are satisfied as a whole. Thus, each point refers to a possibility, either from the provider or the consumer, to achieve a SLA with the other party. These regions may adopt different shapes, which act as indicators of the degree of expressiveness to specify SLOs.

Concerning expressiveness, symmetry refers to the fact that both guaranteed and required SLOs can be specified with the same expressiveness. Fig. 1(b) illustrates both cases, namely, asymmetric and symmetric SLOs. In the former case, it is usual for guaranteed SLOs to be defined by means of property/value pairs, so that their regions are given by single points in the agreement space.

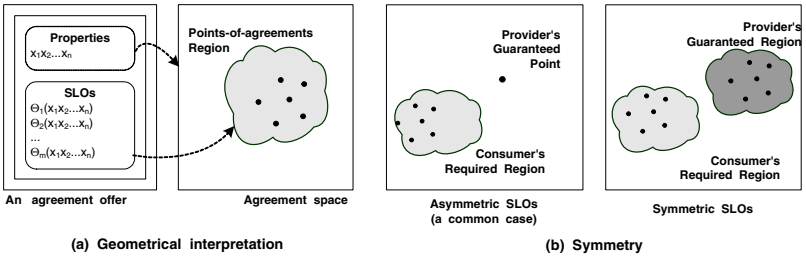

Fig. 1. Geometrical interpretation of agreement offers

Both expressiveness and symmetry have severe influences on matchmaking perfomance, as shown in the following.

Optimistic scenario. Matchmaking is defined as if all properties were fixable. Let G be the provider's guaranteed region, and R the consumer's required region. In case of symmetric SLOs, matching is possible provided that the intersection between these regions is non-empty:

$$\text{matches}(G, R) \Leftrightarrow G \cap R \neq \emptyset$$

That is to say, matching refers to searching for a common point between both regions. This scenario is computationally less costlier from an operational perspective, and also much easier to address.

Pessimistic scenario. Matchmaking is defined as if all properties were non-fixable instead. Let G be the provider's guaranteed region, and R the consumer's required region. In case of symmetric SLOs, matching is possible provided that the former region is a subset of the latter:

$$\text{matches}(G, R) \Leftrightarrow G \subseteq R$$

In other words, matching refers to checking that all points in the guaranteed region belong to the required region. This scenario is much more difficult to address, depending on the shapes which may have the involved regions. The more expressive is the specification of SLOs, the more complex are the shapes which their regions take. To sum up, the comparison of regions with complex shapes is a problem which is computationally costly, so matchmaking is usually reduced to less expressive specifications, leading towards optimistic scenarios. Thus, most approaches restrict regions to be either punctual or having limits parallel to axis.

3 Holistic Matchmaking under Bounded Uncertainty

In this section, we present our approach to matchmaking under bounded uncertainty, incorporating both fixability and dependency of properties, as well as taking high expressiveness and symmetry of SLOs into account. As these features can not be separately dealt with, because of the existing relationships among them, a holistic approach is needed.

3.1 The Problem Space

Fixable points & regions of bounded uncertainty. Every property is declared either as fixable or non-fixable. Thus, any point-of-agreement may be split into its fixable and non-fixable parts. The former part is specifically called fixable point (FP) since the provider controls the actual values the fixable properties take during the service execution, so that the consumer is allowed to fix them.

There may be also eventual dependencies among properties, so that non-fixable properties take values which are dependent on values which fixable properties take. As a result, a FP is associated with a region of bounded uncertainty (RBU) whose dimensions are given by the non-fixable properties. Thus, a RBU is composed of the non-fixable part of all points which share the same FP.

As an example, consider the AO in Section 1 whose provider declares that `IMAGE` as fixable, and `FRAMERATE` as non-fixable. The points are given by the tuples `(IMAGE,FRAMERATE)` which satisfy the guaranteed SLOs, so that the fixable part is given by `(IMAGE)` and the non-fixable part by `(FRAMERATE)`. Thus, the FPs are `IMAGE = 320x200` and `IMAGE = 640x480`, so that all values of `FRAMERATE` between 22 and 25 constitute the RBU associated with `IMAGE = 320x200`, and all values of `FRAMERATE` between 20 and 23 constitute the RBU associated with `IMAGE = 640x480`, as shown in Fig. 3.

Filtering and projection. These auxiliary operations are needed to define both FPs and their RBUs, which are illustrated in Fig. 2.

Let Z and X stand for properties which are fixable and non-fixable, respectively. Let Q be a region of points, the region of FPs C_Q is obtained by projecting Q on the Z-axis. In turn, if z^* stands for a FP in C_Q, its associated RBU $N_Q(z^*)$ is obtained by filtering Q through z^* and then projecting on the X-axis.

Holistic matchmaking under bounded uncertainty. Matching is interpreted as the search for a FP common to the provider's guaranteed and consumer's required regions, so that their associated RBUs also match. Note it adopts the optimistic scenario regarding with FPs, and the pessimistic one regarding with RBUs.

Let G be the provider's guaranteed region, and R the consumer's required region. Matching is possible provided that the following holds:

$$\text{matches}(G, R) \Leftrightarrow \{z^* \in C_G \cap C_R \mid N_G(z^*) \subseteq N_R(z^*)\} \neq \emptyset$$

where: (i) z^* is a FP which belongs to the intersection between the provider's and consumer's regions of FPs, C_G and C_R, respectively, and (ii) the RBUs associated with z^* match if $N_G(z^*)$ (the RBU associated with z^* in the guaranteed region) is a subset of $N_R(z^*)$ (the RBU associated with z^* in the required region).

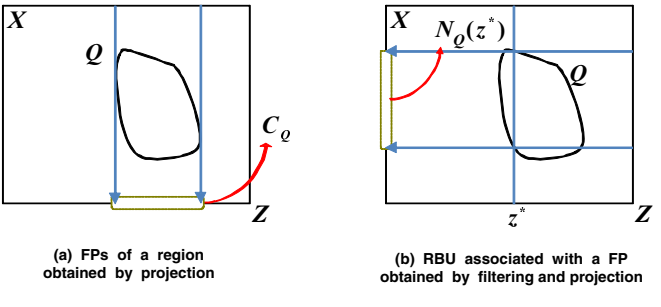

Fig. 2. Filtering and projection

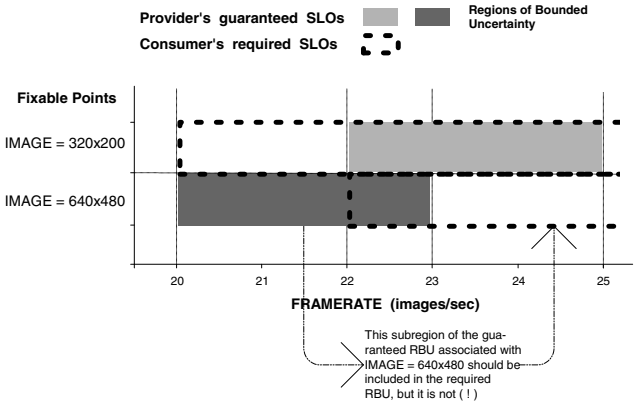

Fig. 3. Matching of FPs associated with RBUs

As an example, consider the AO from a consumer whose required SLOs are:

$$\texttt{IMAGE = 320x200} \implies \texttt{FRAMERATE} \geq 20$$
$$\texttt{IMAGE = 640x480} \implies \texttt{FRAMERATE} \geq 22$$

It matches the provider's AO (see Fig. 3) because there exists a FP, `IMAGE = 320x200`, so that each value in the associated guaranteed RBU (`FRAMERATE` $\in$ `[22..25]`) belongs to the associated required RBU (`FRAMERATE` $\geq$ 20). However, the other FP, `IMAGE = 640x480`, is not a match because of the values in the associated guaranteed RBU (`FRAMERATE` $\in$ `[20..23]`) which do not belong to the associated required RBU (`FRAMERATE` $\geq$ 22).

3.2 The Solution Space

Contraint programming (CP) [9, 15] is our choice to deal with matchmaking. Most CP solvers are able to process highly-expressive constraints, so that the matchmaking can be usually carried out even if SLOs are not restricted to property/value or property/range pairs. This approach was demonstrated to be feasible in [14] where we presented a pessimistic scenario of matchmaking, which we extend in this paper by considering the new scenario.

CP in a nutshell. A constraint satisfaction problem (CSP) is defined by means of a set of variables and their domains, together with a set of constraints specifying which combinations of variables and values, or *solutions*, are acceptable.

Let ψ be a CSP, it is satisfiable iff its solution space is not empty:

$$\text{sat}(\psi) \Leftrightarrow \text{sol}(\psi) \neq \emptyset$$

Due to the declarative nature of CP, computational procedures to enforce CSPs need not to be programmed. Given a problem, the idea is to solve it by stating a CSP which models the problem itself, and then trying to check its satisfiability by means of a CP solver.

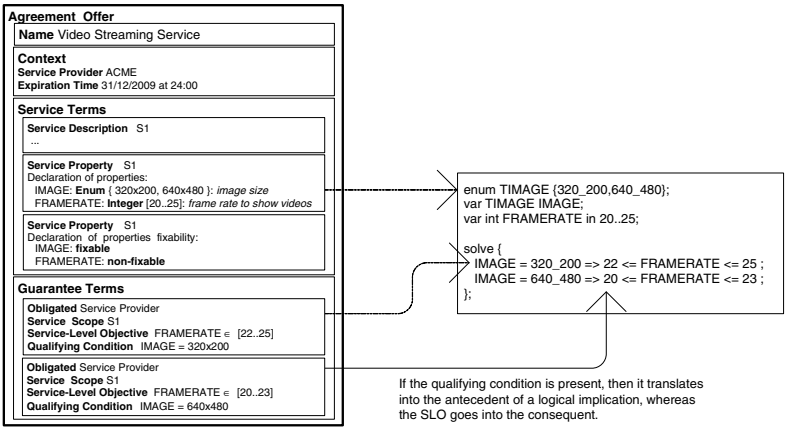

Fig. 4. Translation from an AO to the corresponding CSP

Translating from AOs to CSPs. The key point of the approach is the translation from an AO to a CSP. In short, the CSP is obtained by translating AO properties to CSP variables and SLOs to CSP contraints, so that the CSP solution space is equivalent to the AO region. As an example, Fig. 4 illustrates it. On the one hand, the AO is written in a schematic representation of WS-Agreement, the state-of-the-art recommendation by the Open Grid Forum [2][1]. On the other hand, the CSP is written in OPL, a language designed to specify CSPs [9].

Transformations for basic scenarios. Let G be the guaranteed region by a provider, and R the required region by a consumer[2], whose respective CSPs are denoted with ψ_G and ψ_R. The matching formulae is transformed to a CSP whose satisfiability is checked for a result, as shown in Fig. 5(a).

Transformations for holistic matchmaking under bounded uncertainty. Let z and x stand for variables corresponding to fixable and non-fixable properties, respectively, so that the expression $\psi(z, x)$ denotes a CSP based on such variables. The matching formulae is transformed to a CSP which consists of two parts, as shown in Fig. 5(b): (i) the former corresponds to the transformation for the optimistic scenario in order to get the common FPs; (ii) the latter corresponds to the transformation for the pessimistic scenario in order to check whether the guaranteed and required RBUs (associated with such FPs) match.

Unfortunately, checking the satisfiability of this CSP does not yield the expected results. Its parts can not be solved separately since both parts are bound by the z variables, which stand for the common FPs. The solution lies in transforming the latter part to an equivalent CSP based on universal quantification, in order to properly process the projection on z variables.

[1] Note a new service term to declare the fixability of properties has been added.
[2] It is assumed both guaranteed and required SLOs are based on the same properties.

	Problem space	Solution space
Optimistic scenario	$matches\,(G,R) \Leftrightarrow G \cap R \neq \phi$	$matches\,(G,R) \Leftrightarrow sat(\psi_G \wedge \psi_R)$
Pessimistic scenario	$matches\,(G,R) \Leftrightarrow G \subseteq R$	$matches\,(G,R) \Leftrightarrow \neg sat(\psi_G \wedge \neg \psi_R)$

(a) Transformations for basic scenarios of matchmaking

	Problem space	Solution space
Scenario of holistic matchmaking under bounded uncertainty	$matches(G,R)$ $\Leftrightarrow \{z^* \in C_G \cap C_R \mid N_G(z^*) \subseteq N_R(z^*)\} \neq \phi$	$matches(G,R)$ $\Leftrightarrow sat(\psi_G(z,x) \wedge \psi_R(z,x)$ $\wedge \neg(\psi_G(z,x') \wedge \neg\psi_R(z,x')))$

(b) Transformations for holistic matchmaking under bounded uncertainty

Fig. 5. Transformations for scenarios of matchmaking under study

Consequently, the CSP turns out to be as follows:

$$matches(G,R) \Leftrightarrow sat(\psi_G(z,x) \wedge \psi_R(z,x) \wedge \forall x' \cdot \psi_G(z,x') \Rightarrow \psi_R(z,x'))$$

Solving a quantified-CSP is a classic problem in CP. Most solutions are based on particular instances of it. As an example, quantified variables can be restricted to be denumerable. In doing so, an iterative solving of CSPs for holistic matchmaking is made easier, and feasible provided that domains of quantified variables (which correspond to FPs) remain as short as possible [3, 8].

4 Conclusions and Future Work

In this paper, we have introduced a holistic scenario of matchmaking which deals with bounded uncertainty, incorporating both the fixability and dependency of properties. Our approach improves the accuracy of matchmaking, avoiding both false positives and false negatives which are due to mis-assumptions regarding with the fixability of properties. A CP-based solution has been also proposed to perform the service matchmaking in that scenario, whose feasibility was demonstrated regarding the pessimistic matchmaking in [14], though the empirical evaluation have to be repeated to properly find out about the influences of quantified variables in CSP solving.

Finally, there are some untouched questions. On the one hand, the optimal selection needs also to be studied under this scenario. On the other hand, we are considering other technologies such as soft constraints and fuzzy constraints to be used, whose feasibility to solve problems under uncertainty is well-known.

Acknowledgements

The authors would like to thank Mr. Carlos Müller, Mr. José Antonio Parejo, and Dr. Manuel Resinas for their helpful discussions, whose comments and suggestions improved the paper substantially.

References

[1] Afandi, R., Zhang, J., Gunter, C.A.: AMPol-Q: Adaptive Middleware Policy to Support QoS. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 165–178. Springer, Heidelberg (2006)

[2] Andrieux, A., Czakowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement) Version 1.1 draft 20 (September 2006)

[3] Bordeaux, L., Monfroy, E.: Beyond NP: Arc-Consistency for Quantified Constraints. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 371–386. Springer, Heidelberg (2002)

[4] Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K.: Quality of Service for Workflows and Web Service Processes. Journal of Web Semantics 1(3), 281–308 (2004)

[5] Cheng, W., Wang, H.: Uncertainty-Aware QoS Description and Selection Model for Web Services. In: 4th IEEE Services Computing Conf., pp. 154–161 (2007)

[6] Cheng, W., Wang, H.: Web Service Decision-Making Model Based on Uncertain-but-Bounded Attributes. In: 4th IEEE SCC Workshop on Semantic Web for Web Services and Processes, pp. 81–86. IEEE CS Press, Los Alamitos (2007)

[7] Frølund, S., Koistinen, J.: Quality-of-Service Specification in Distributed Object Systems. Distributed Systems Engineering Journal 5(4) (1998)

[8] Gent, I.P., Nightingale, P., Rowley, A., Stergiou, K.: Solving Quantified Constraint Satisfaction Problems. Artificial Intelligence 172(6-7), 738–771 (2008)

[9] Hentenryck, P.: Constraint and Integer Programming in OPL. Informs Journal on Computing 14(4), 345–372 (2002)

[10] Hwang, S.Y., Wang, H., Tang, J., Srivastava, J.: A Probabilistic Approach to Modeling and Estimating the QoS of Web-Services-based Workflows. Information Sciences 177(23), 5484–5503 (2007)

[11] Li, B., Nahrstedt, K.: A Control-Based Middleware Framework for Quality of Service Adaptations. Journal on Selected Areas in Communications 17(9), 1632–1650 (1999)

[12] Mandaric, A., Oberweis, A., Perc, P.: Web Services-based Architecture for Reducing Behaviour and Quality Uncertainties. In: 1st IEEE Conf. on e-Science and Grid Computing, Melbourne, Australia, pp. 320–327 (December 2005)

[13] Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F., Krämer, B.: Service-Oriented Computing: A Research Roadmap. In: Dagstuhl Seminar on Service Oriented Computing (2006)

[14] Ruiz-Cortés, A., Martín-Díaz, O., Durán, A., Toro, M.: Improving the Automatic Procurement of Web Services using Constraint Programming. International Journal of Cooperative Information Systems 14(4), 439–467 (2005)

[15] Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, London (1995)

[16] Vu, L., Aberer, K.: A Probabilistic Framework for Decentralized Management of Trust and Quality. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676, pp. 328–342. Springer, Heidelberg (2007)

[17] Wang, P., Chao, K.M., Lo, C.C., Huang, C.L., Li, T.: A Fuzzy Model for Selection of QoS-Aware Web Services. In: 2nd Intl. Conf. on e-Business Engineering, Shanghai, China, pp. 585–593. IEEE Computer Society, Los Alamitos (October 2006)

[18] Wohlstadter, E., Tai, S., Mikalsen, T., Rouvellou, I., Davanbu, P.: GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interations. In: 26th Intl. Conf. on Software Engineering, Edinburg, Scotland, pp. 189–199. IEEE CS Press, Los Alamitos (2004)