Dealing with Fuzzy QoS Properties in Service Composition

Ioana Şora and Doru Todinca
Department of Computer and Software Engineering,
Politehnica University of Timisoara, Romania

***Abstract*—Service oriented applications are composed by orchestrating sets of cooperating services and are further offered as services themselves. Many applications have a dynamic character, needing to chose at runtime which service implementations to compose. When multiple possibilities exist, the choice is made according to the user preferences for certain Quality of Service (QoS) parameters.**

In this work we propose an approach for QoS-aware selection and composition of services, in the presence of both vague user preferences and vague service descriptions. The central elements of our approach are an extensible QoS ontology and a compositional model for vague QoS properties in workflow service composition.

I. Introduction

Service Oriented Computing is an emerging computing paradigm promoting construction of distributed complex applications out of loosely-coupled reusable services [1]. This is the essence of the service oriented paradigm, and is a constant property of it independent from the evolving standards and technologies in the domain.

The early scenario in service oriented computing was that Service Providers design and implement services, hosting them as network-accessible modules, and advertise them by defining service descriptions that are published in Service Registries. Clients or Service Requesters use the published service descriptions in order to find the needed services. A major research effort in this field was directed towards enhancing the automatic web service discovery and selection with semantic and quality of service aspects [2]. Achieving this requires adding semantic annotations to web service descriptions and including user preferences related to quality of service characteristics in the selection process.

Contemporary Service Oriented Architectures have evolved, being much more complex. New Service Oriented Applications are composed by orchestrating sets of cooperating services and have a dynamic character, needing to chose at runtime which service implementations to use. The problem of QoS-aware service composition [3], [4], [5] consists in searching candidates such that the overall composition meets the user preferences for QoS or globally optimizes the QoS. It is NP-hard and is one of the main research topics in the field of service oriented computing [2]. This problem also needs support in form of compositional models for QoS properties and needs QoS specification solutions developed for the selection problem to be be adapted in this context.

In our previous works [6], [7], [8] we have developed a novel fuzzy logic approach for the specification, selection and ranking of services according to individual QoS preferences. Our method can handle uncertainity at all levels: user preferences and service descriptions. In [8] we have proven the feasability of our fuzzy selection and ranking approach, both from the point of view of the integration with current standards that govern the domain of web services and the performance of the implementation of fuzzy selection and ranking in service registries. Many other works proposed solutions for including QoS in service selection, some of them also recognizing the advantages of fuzzy approaches in this domain [9], [10], [11] for handling vague preferences. However, none of these works deal with fuzzy service descriptions, and existing state-of-the-art compositional models for QoS properties are not implemented to cope with vague service descriptions.

In this paper, we extend our previous work with the capability of handling service composition in the context of service selection and ranking with QoS properties. The global FQ (*F*uzzy-*Q*oS) architecture of our approach is presented in Section III. Section IV introduces our FQ ontology, while Section V details an important part of the FQ ontology, the compositional model for vague QoS properties in workflow service composition. Conclusions are outlined in section VI.

II. Background and Motivation

Service providers and service consumers are the main actors in the broad landscape of service oriented computing. Independent from the technologies used for services, the following scenario describes the fundamental use case involving services

Service providers implement and publish services of different service types. Each service type offers a functionality from a certain functionality domain. Several implementations from different providers and with different QoS attributes may be available for each service type. Further, a service implementation may be either a monolithical implementation or it may point to the description of an abstract composition, given in form of a workflow description or as an abstract business process. An abstract process describes a service composition in terms of service types of partener services involved. In order to get a service implementation as an executable process, all partners have to be linked to service implementations.

A service consumer searches for a service implementation that offers a needed functionality and also has a set of preferred QoS properties.

Mechanisms for service selection and composition have to solve such consumer requests: they have to discover service types that provide the required functionality and select the implementations that match best to the user QoS preferences. In case of service types that point to abstract processes as implementations, there is also a composition step, in order to produce the executable process. This will be preceded, recursively, by selection of implementations for all its parts. The QoS attributes of the resulting executable composition have to be matched against the QoS preferences of the consumer request.

Several research questions are included in this scenario:

- How are QoS properties specified ?
- Which degrees of uncertainity are allowed and where ? (in the specification of QoS attributes of service implementations and/or user preferences ?)
- How are QoS attributes of a composition computed starting from QoS attributes of their parts ?

Regarding the first of the research questions, the specification of QoS properties, most approaches rely on the concept of domain ontologies [12] [13], [14]. This enables automatic matching while keeping a low degree of formalism in the description of services.

However, the state-of-the-art approaches allow only a limited degree of uncertainity, mostly reduced to the user request and the partial matching [9], [10]. We consider that the specification of services may also be subject to imprecise description. This use case, of services with imprecise descriptions, appears both in the situation when service providers can not or want not to give exact values for all the QoS properties, but even more in the situation when service descriptions are annotated with values for QoS properties as a consequence of service monitoring. In this case, different monitoring components could not establish or measure precise values of some QoS properties but they may give estimations in fuzzy terms. Thus, we have proposed an approach where uncertainity is contained in both consumer request and service descriptions, as well as the ranking and selection algorithm that has to provide the best matching in these conditions [8], [6].

Another issues is raised by the abstract composite services: before they can be used, adequate candidates have to be bound to all abstract services. Many works have developed search algorithms as optimization problems, but have put a reduced interest in using cost functions based on detailed and realistic models of the QoS properties. The QoS values of a composite service should be determined not only by the QoS values of the composed services, but these have to be aggregated taking into account both the types of QoS properties and the composition structure used. Works such as [15] do not take into account the particular ways the services in a composition can interact. In [16], an approach to annotate the syntactical BPEL constructs with semantic information is shown, but it relies on manual annotation of the composition. Different approaches defined models for automatically composing a few types of QoS properties in the case of services interacting in certain workflow patterns [17], [18], [19], [20]. However, none of these took into account specifications of QoS properties that are described as fuzzy terms. It is the goal of this work to extend our FQ architecture and ontology with capabilities for the automatic generation of QoS annotations for service composition in order to extend the service selection to selection of composites.

In summary, our approach starts with the following assumptions:

- Consumer requests may have imprecise (vague) preferences with personalized weighting of QoS attributes
- Service registries know about service implementations and abstract composite services
- Service descriptions are attached to service implementations and may be imprecise (vague) with regard to their QoS properties
- QoS properties may be either general ones or specific to certain domains

Starting from these assumptions and requirements, we define a solution for service matching and QoS aware service ranking and selection, including by composition. This solution comprises:

- A global FQ (*F*uzzy-*Q*oS) architecture of implementing our approach over web services technologies
- An ontology for working with vague QoS properties at all levels
- A Compositional model defining aggregation functions for vague QoS properties

III. The global FQ architecture for selection and composition of services with fuzzy QoS properties

The core of our QoS aware services approach is represented by a customized service registry. In the FQ architecture, this customized service reqistry is formed by wrapping a non-QoS aware service registry with a set of additional components. In this way, backwards compatibility can be assured and the wrapper can be adapted to several different technologies for registriers. The wrapper is composed by following main components: the *FunctionalityFindingService*, which is using an *AnnotationComposer* and *FuzzyRanker*, and a *DomainOntologyService*, as depicted in Figure 1.

The *Functionality Finding Service (FFS)* is responsible for retrieving all functionally equivalent services from the registry in response to requests coming from consumers. It interacts with the *Domain Ontology Service* in order to complete its information about the semantics of QoS parameters found in the description of service candidates. In the case of candidates that are abstract composites, it recursively searches candidates for all the needed bindings. The resulting composed service is annotated with QoS properties by the *Annotation Composer*, applying the specific aggregation functions to the values of QoS properties of the composed services.

The *Domain Ontology Service (DOS)* is responsible for managing all the ontology data and it is the enabling element for our entire QoS-aware approach. QoS ontologies define the semantics of QoS parameters for different service types. Ontology data are found in the *QoS Properties Directory* and the *Domain Ontologies*. They will be detailed in section IV.

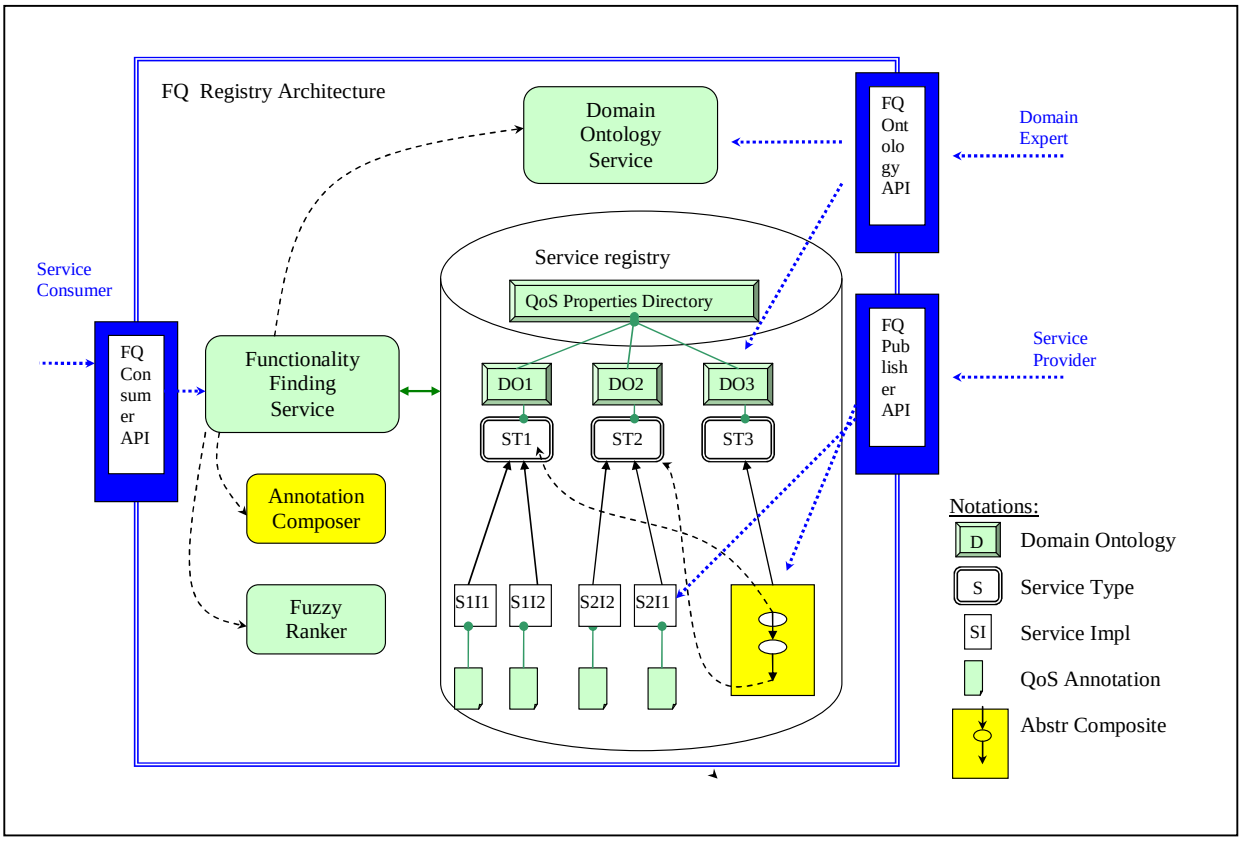

Fig. 1. Arhitecture of our approach

The *Annotation Composer* analyzes the workflow structure of composite services and computes the values of QoS properties of the composite, starting from the values of these properties in the composed services. The Annotation Composer applies the specific aggregation functions defined in the *QoS Properties Directory*. The ontology concepts will be detailed in section V.

Finally, the functionally equivalent web service candidates are ranked according to their QoS properties matching the individual client request. The ranking is done by the *FuzzyRanking* subsystem (FR), which implement the approach based on automatically generated fuzzy rules starting from individual user preferences and using them in a fuzzy inference process ranking the web service candidate, as described in our previous work [6].

The current prototype of the FQ-architecture is built around web services technologies, uses UDDI for traditional registry technology, SAWSDL for semantically annotating WSDL descriptions, and BPEL for defining the abstract composites, but the FQ wrapper can be easily adapted to work over other technologies as well.

IV. FQ Domain Ontologies

Many approaches use ontologies to define QoS related aspects in service computing [12] [13], [14]. The concepts defined by an QoS ontology are used in the description of QoS related aspects by all the actors of service computing: providers, consumers, and third-parties such as domain experts or standardization agencies.

In this work we define a FQ - a fuzzy QoS ontology which has the power not only to help describing consumer requests and service descriptions but which is also able to support composition of services by calculating the QoS annotations of the composite. FQ has been defined in order to respond to following needs: consumer requests and service descriptions may be done in vague terms, QoS properties may be universal or may apply only to certain application domains, or may have different meanings inside different application domains, and new QoS properties or application domains may be added at any time.

The solution comprises a two level ontology: a Global QoS Properties Directory and specialized Domain Ontologies.

Every possible property is defined inside the Properties Directory. A Property Directory Entry corresponds to a QoS property and has the following attributes: a name, a measurement unit, an optimization direction, a globality flag, and aggregation functions that describe how their values are composed in different workflow sequences.

The Domain Ontologies allow the domain experts to establish and describe the valid non-functional properties for each functionality domain. A Domain Ontology contains the allowed properties for a specific functionality. Each property is specified by its range of values and set of linguistic variables. The fuzzy terms which can be used to describe this property have to be enumerated, for each term providing its name and the shape of its membership function as defined by a domain

```
<ontology xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="ontology.xsd">
 <functionality name="Translator">
  <property name="ExecutionTime" start="Infinity" end="0">
      <term name="Fast" a="0" b="0" c="15" d="20" />
      <term name="Medium" a="10" b="15" c="30" d="35" />
      <term name="Slow" a="30" b="40" c="Infinity" d="Infinity" />
  </property>
  <property name="Availability" ...
  ...
```

Fig. 2. Example: fragment of the Translator domain ontology

```
<service name="MyFrench2RomTranslation" functionality="Translator">
         <property name="EecutionTime" fuzzy="Medium" />
         <property name="Availability" fuzzy="0.75 0.8 0.9 0.95" />
         <property name="GrammarQuality" fuzzy="0.8 0.85 0.9 0.95" />
</service>
```

Fig. 3. Example: annotation of a Translator service implementation

expert. In the current implementation, we assume that the terms have to be defined of trapezoidal shape. The trapezoidal fuzzy numbers are represented as quadruplets $n = (a, b, c, d)$, where $0 \leq a \leq b \leq c \leq d$ are the x-coordinates of the trapeze points.

Characteristic to our approach is that we need domain ontologies for defining value categories for different non-functional or QoS properties through linguistic variables. For example, the response time of a service could be described with the terms *very fast*, *fast*, *slow*, *very slow*. But values of QoS attributes can be categorized only in a well defined context of a certain functionality. The same measured value of 50 sec as a response time, for example, has different values for different service types: while it corresponds to *very fast* for a text translation service, it means *slow* for a currency converter service. The membership functions of these terms can be established by domain experts, separately for each type of service - as the expectations are different for different types of services.

Also, different functionalities may have different sets of QoS attributes. There can be attributes which can be defined only in the context of a certain functionality. For example, a text translation service can be described by an attribute such as *grammar quality*, but this attribute does not apply to the currency converter service type.

For example, a Translator service can be described by following non-functional properties: *Execution Time, Availability, Grammar Quality*. The *Translator Ontology* defines these properties as valid for implementations of the Translator functionality. It also defines value categories of these properties in the context of translation services. The properties are described according to the corresponding Translator domain ontology. A fragment of it is presented in Figure 2. The Global Properties Directory establishes an increasing optimization direction for *Availability, Grammar Quality* and a decreasing one for *Execution Time*. It also defines the aggregation functions for every property, for sequence, parallel and choice structures, like they will be defined in Table I in the next section.

A service implementation corresponding to the Translation type will be annotated with QoS specifications according to the constraints imposed by the domain ontology. A Web Service already has a syntactical description which is provided by the Web Services WSDL. It describes, syntactically, what the Web Service supports in terms of: what operations it offers, the format of the messages that are exchanged, the communication protocol, etc. The annotation that we are discussing completes the description of the Web Service by providing QoS information. An example of a Translator service annotation is shown in Figure 3.

V. Compositional model for vague QoS properties

The *Annotation Composer* approaches the QoS aware composition in a bottom-up manner. The structure of the composite is determined first and the problem is now how to derive the QoS properties of the composite starting from the properties of the composed services.

This problem is a complex one because composition rules depend heavily on the QoS property being considered and on the workflow pattern of the composition. To support this kind of QoS aware service composition, there are needed models for workflow QoS computation, defining aggregation functions for every type of QoS property in every type of workflow pattern.

Workflow patterns that appear in service compositions, including BPEL description of composed business processes, can be reduced to simplified models based on only three fundamental structures: sequential, parallel and alternative.

From our survey on different types of QoS properties, it resulted that the following set of aggregation functions can handle all properties in all types of workflow structures: Sum, Multiplicative, Average, Maximum, and Minimum.

Since our approach allows service descriptions to be vague, fuzzy numbers are used to represent the values of QoS properties. We use trapezoidal fuzzy numbers, represented as $n = (a, b, c, d)$ where $a, b, c, d \geq 0$. Taking into account that these numbers represent values of QoS properties, the restriction on a, b, c, d being positive real numbers is a natural one. Thus, for two trapezoidal fuzzy numbers $n_1 = (a_1, b_1, c_1, d_1)$ and $n_2 = (a_2, b_2, c_2, d_2)$ representing two values of a QoS

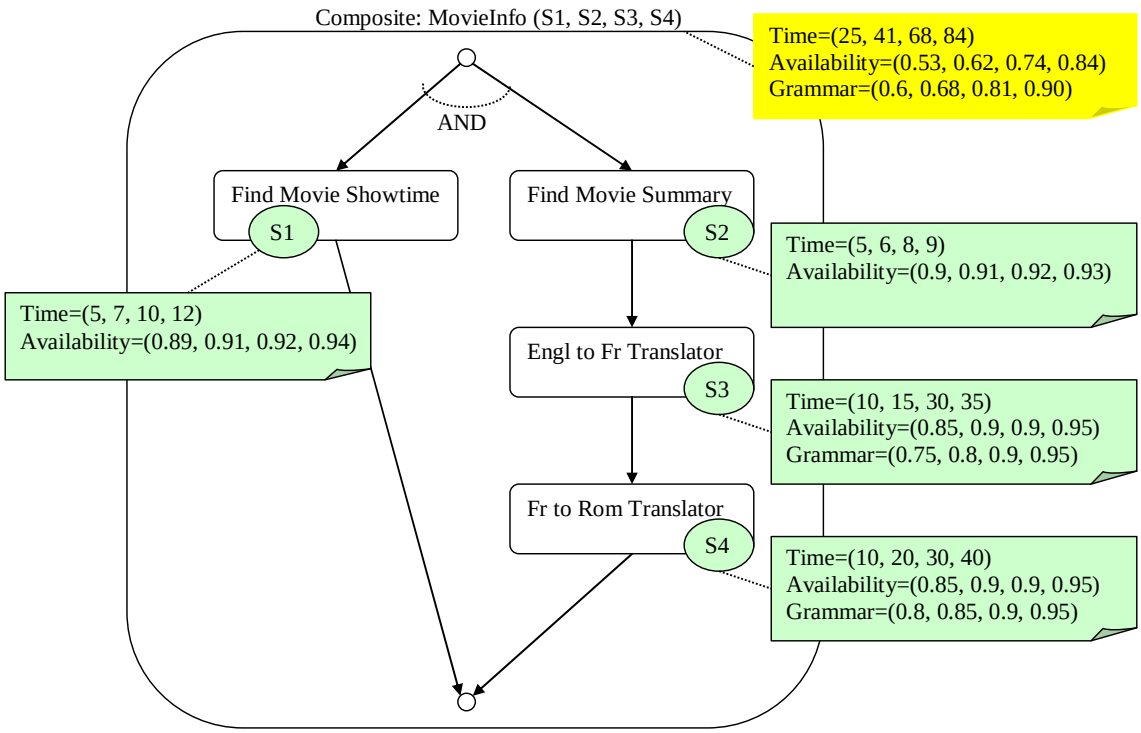

Fig. 4. Example: Composite Service MovieInfo

property, we define the aggregation functions Sum, Multiplicative, Average, Maximum, and Minimum as in the following paragraphs. The result of the aggregation function is also a fuzzy value of the composite QoS property, represented as a trapezoidal fuzzy number as well.

A. The SUM aggregation function

This aggregation function computes the value of a property in the composite as the sum of the values of the same property. Fundamental properties such as *Execution Time* and *Cost* are additive when the services are composed in sequence.

$SUM(n_1, n_2) = (a_1 + a_2, b_1 + b_2, c_1 + c_2, d_1 + d_2)$

B. The AVG aggregation function

This function computes the arithmetic mean of the properties. It is applied to properties such as *Reputation*, *Frequency*, *UserRating*.

$AVG(n_1, n_2) = ((a_1+a_2)/2, (b_1+b_2)/2, (c_1+c_2)/2, (d_1+d_2)/2)$

C. The Multiplicative aggregation function

The MUL aggregation function is mainly used for properties with values representing percentages, such as *Availability* or *Reliability*.

$MUL(n_1, n_2) = (a_1 \cdot a_2, b_1 \cdot b_2, c_1 \cdot c_2, d_1 \cdot d_2)$

D. The Minimum Composition Function

The MIN aggregation function computes the minimum value of the properties and the result represents the composition result that will be part of the new composite annotation. It is a function which is much used to take the worst case as the result of the composition. In sequential compositions, it applies to properties such as: *Throughput*, *Request Limit Per Day*. It is also used for many other properties in choice XOR scenarios: *Availability*, *Reputation*, *Frequency*.

$MIN(n_1, n_2) = (min(a_1, a_2), min(b_1, b_2), min(c_1, c_2), min(d_1, d_2))$

E. The Maximum Composition Function

The MAX aggregation function computes the maximum value of the properties and the result represents the composition result that will be part of the new composite annotation. It is a function which is also used to take the worst case as the result of the composition, for these properties where the small values mean optimal values. It is used for some properties in choice XOR scenarios: *Execution time*, *Cost*.

$MAX(n_1, n_2) = (max(a_1, a_2), max(b_1, b_2), max(c_1, c_2), max(d_1, d_2)$

F. Examples of using the aggregation functions

Table I presents examples of using the aggregation functions for different QoS properties and workflow patterns.

The table comprises universal properties as well as domain specific properties. Universal properties are, for example: Execution Time, Cost, Throughput, Availability, Reliability, Reputation, Frequency, Time between requests, Request allowed per day. The table contains two domain specific properties, *Grammar quality* in the domain of natural language processing services such as translators or summarizers, and *Storage Limit*

TABLE I
EXAMPLES OF AGGREGATION FUNCTIONS FOR SOME QOS PROPERTIES IN WORKFLOW PATTERNS

	sequence	parallel	choice
Execution Time	SUM	MAX	MAX
Cost	SUM	SUM	MAX
Throughput	MIN	MIN	MIN
Availability	PROD	PROD	MIN
Reliability	PROD	PROD	MIN
Successful Exec Rate	PROD	PROD	MIN
Reputation	AVG	AVG	MIN
Frequency	AVG	AVG	MIN
Time between requests	MAX	MAX	MAX
Requests allowed	MIN	MIN	MIN
Grammar Quality	PROD	PROD	MIN
Storage Limit	SUM	SUM	MIN

in the domain of file sharing services. Additional domain specific properties with their aggregation functions can be defined with the help of the FQ ontology, as described in Section IV.

G. Example Scenario

Figure 4 presents an example scenario of a composed service, MovieInfo, defined as a workflow involving four other services. In this example workflow, S1 is executed in parallel with the sequence formed by S2, S3 and S4. For *Execution Time*, the SUM function is applied in sequence and MAX in parallel; for *Availability*, the PROD function is applied both in sequence and in parallel; for *Grammar Quality*, the PROD function is applied for the two services in sequence, while for the other it is not applicable, being a domain specific property only. Applying the aggregation functions, the QoS annotation of the composite results as depicted in the figure.

VI. CONCLUSION

In this work we propose an approach for QoS-aware selection and composition of services, in the presence of both vague user preferences and vague service descriptions. The central elements of our approach are an extensible QoS ontology, containing also a compositional model for vague QoS properties in workflow service composition.

Specification of QoS properties, service selection, composition and ranking are fundamental problems of the service oriented paradigm, invariant to technological aspects of the different service technologies, and remain valid also in the context of cloud services [21], [22], [23]. Our solution, although prototyped over Web Service technologies, can be easily transposed in these new technologies as well.

REFERENCES

[1] T. Erl, *Service-oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.

[2] M. Papazoglu, P. Traverso, S. Dustdar, and F. Leyman, "Service-oriented computing: State of the art and research challenges," *IEEE Computer*, Nov. 2007.

[3] A. Strunk, "QoS-aware service composition: A survey," in *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, 2010, pp. 67–74.

[4] M. Alrifai, T. Risse, and W. Nejdl, "A hybrid approach for efficient web service composition with end-to-end QoS constraints," *ACM Trans. Web*, vol. 6, no. 2, pp. 7:1–7:31, Jun. 2012.

[5] H. Guo, F. Tao, L. Zhang, S. Su, and N. Si, "Correlation-aware web services composition and QoS computation model in virtual enterprise," *The International Journal of Advanced Manufacturing Technology*, vol. 51, no. 5-8, pp. 817–827, 2010.

[6] I. Sora, D. Todinca, and C. Avram, "Translating user preferences into fuzzy rules for the automatic selection of services," in *SACI*, 2009, pp. 497–502.

[7] I. Sora and D. Todinca, "Specification-based retrieval of software components through fuzzy inference," *Acta Politehnica Hungarica*, vol. 3, no. 3, pp. 121–132, 2006.

[8] I. Sora, G. Lazar, and S. Lung, "Mapping a fuzzy logic approach for QoS-aware service selection on current web service standards," in *Computational Cybernetics and Technical Informatics (ICCC-CONTI), 2010 International Joint Conference on*, 2010, pp. 553–558.

[9] P. Wang, "QoS-aware web services selection with intuitionistic fuzzy set under consumers vague perception," *Expert Systems with Applications*, vol. 36, no. 3, pp. 4460–4466, 2009.

[10] M. C. Platenius, M. von Detten, S. Becker, W. Schäfer, and G. Engels, "A survey of fuzzy service matching approaches in the context of on-the-fly computing," in *Proceedings of the 16th International ACM Sigsoft symposium on Component-based software engineering*, ser. CBSE '13. New York, NY, USA: ACM, 2013, pp. 143–152.

[11] X. T. Nguyen, R. Kowalczyk, and M. T. Phan, "Modelling and solving QoS composition problem using fuzzy DisCSP," in *Web Services, 2006. ICWS '06. International Conference on*, Sept 2006, pp. 55–62.

[12] E. Giallonardo and E. Zimeo, "More semantics in QoS matching," in *Service-Oriented Computing and Applications, 2007. SOCA '07. IEEE International Conference on*, June 2007, pp. 163–171.

[13] G. Dobson, R. Lock, and I. Sommerville, "QoSOnt: a QoS ontology for service-centric systems," in *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on*, Aug 2005, pp. 80–87.

[14] V. X. Tran, H. Tsuji, and R. Masuda, "A new QoS ontology and its QoS-based ranking algorithm for web services," *Simulation Modelling Practice and Theory*, vol. 17, no. 8, pp. 1378 – 1398, 2009, dependable Service-Orientated Computing Systems.

[15] P. Wang, K.-M. Chao, and C.-C. Lo, "On optimal decision for QoS-aware composite service selection," *Expert Systems with Applications*, vol. 37, no. 1, pp. 440 – 449, 2010.

[16] M. Pistore, L. Spalazzi, and P. Traverso, "A minimalist approach to semantic annotations for web processes compositions," in *The Semantic Web: Research and Applications*, ser. Lecture Notes in Computer Science, Y. Sure and J. Domingue, Eds. Springer Berlin Heidelberg, 2006, vol. 4011, pp. 620–634.

[17] M. Jaeger, G. Rojec-Goldmann, and G. Muhl, "QoS aggregation for web service composition using workflow patterns," in *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International*, Sept 2004, pp. 149–159.

[18] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "A framework for QoS-aware binding and re-binding of composite web services," *Journal of Systems and Software*, vol. 81, no. 10, pp. 1754 – 1769, 2008, selected papers from the 30th Annual International Computer Software and Applications Conference (COMPSAC), Chicago, September 721, 2006.

[19] J. M. Ko, C. O. Kim, and I.-H. Kwon, "Quality-of-service oriented web service composition algorithm and planning architecture," *Journal of Systems and Software*, vol. 81, no. 11, pp. 2079 – 2090, 2008.

[20] J. A. Parejo, S. Segura, P. Fernandez, and A. Ruiz-Corts, "QoS-aware web services composition using GRASP with path relinking," *Expert Systems with Applications*, vol. 41, no. 9, pp. 4211 – 4223, 2014.

[21] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012 – 1023, 2013, special Section: Utility and Cloud Computing.

[22] Z. Rehman, F. Hussain, and O. Hussain, "Towards multi-criteria cloud service selection," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, 2011, pp. 44–48.

[23] L. Qi, W. Dou, X. Zhang, and J. Chen, "A QoS-aware composition method supporting cross-platform service invocation in cloud environment," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1316 – 1329, 2012.