# QoS-based Web Service Description and Discovery

by

Kyriakos E. Kritikos

A dissertation submitted to the faculty of

University of Crete

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

University of Crete

December 2008

UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE

# QoS-based Web Service Description and Discovery

Dissertation submitted by
Kyriakos Kritikos
in partial fulfillment of the requirements for
the PhD degree in Computer Science

Author:

Kyriakos Kritikos, University of Crete

Examination Committee:

Dimitris Plexousakis, Professor, University of Crete, Supervisor

Christos Nikolaou, Professor, University of Crete

Manolis Koubarakis, Associate Professor, National and Kapodistrian University of Athens

Kostas Stergiou, Assistant Professor, University of the Aegean

Evangelos Markatos, Professor, University of Crete

Vassilis Christophides, Associate Professor, University of Crete

Pierluigi Plebani, Assistant Professor, Technical University of Milan

Approved by:

Trahanias Panos
Chairman of Graduate Studies & Professor of University of Crete
Heraklion, November 2008

# Abstract

The success of the *Web Service* (WS) paradigm has led to a proliferation of available WSs. As these WSs are advertised in inter and intra-organizational registries/repositories, they have to be discovered based on the functional needs of the user. However, these registries rely on the current WS standard stack that uses *WSDL* and *UDDI* for WS description and discovery. Unfortunately, these two standard languages are based on the use of static descriptions of service interfaces and other general non-functional service attributes for publishing and finding WSs. This situation creates two problems. Firstly, syntactic discovery efforts return imprecise and inaccurate results. Secondly, no means are provided in order to select among multiple services that appear to perform the same function.

The first problem is solved by combining *Semantic Web* (SW) and WS technologies. Ontologies are used, which provide meaning to concepts and relationships between them and thus lead to semantic WS descriptions and discovery algorithms. These enhanced discovery algorithms equipped with SW technologies provide more precise and accurate results. One recent result of the joint SW and WS efforts is *OWL-S*, which is currently a W3C member submission.

The second problem can be solved by taking into account a big subset of all possible non-functional properties of WSs, which is called *QoS* (Quality of Service). QoS is closely related with the performance of a WS as well as with other features and characteristics of a WS that bear on its ability to satisfy stated or implied needs. Therefore it has a substantial impact on user's expectations from a service and can be used as a discriminating factor between functionally equivalent WS advertisements. Thus WS descriptions must be enhanced with QoS descriptions. Additionally, WS discovery algorithms should perform QoS-based *filtering* (matchmaking) and *ranking* (selection) on WS advertisements in order to produce fewer ranked results.

A *QoS offer* (or *demand*) of a WS is a set of constraints/restrictions on some QoS attributes that restrict them to have certain values (for unary constraints) or certain combinations of values (for n-ary constraints). Actually, the current modeling efforts of QoS offers or demands only differ in the expressiveness of these constraints. However, when it comes to QoS attributes/metrics modeling, these efforts fail. The first reason is because the QoS attribute definition is syntactic. Another reason is that the QoS attribute model is not rich enough, not incorporating the definition of measurement units, currency units, measured properties and measurement methods. The last reason of failure is that the QoS attribute

model is not extensible or when it is extended, the underlying computation (matchmaking and selection) model must be changed as a consequence.

Based on the above inabilities of the current state-of-the-art QoS description models of WSs, the first part of this thesis is dedicated to conducting a research on all QoS-based WS description efforts in order to unveil those features and parts that are necessary for a QoS description model of WSs. The result is a set of requirements for a rich, semantic and extensible QoS-based WS description. Based on these requirements, an ontology language called *OWL-Q* has been carefully designed and created by using the *OWL* ontology specification and by extending *OWL-S*. *OWL-Q* is separated into many facets, each capturing a particular facet of QoS-based WS description, that could be extended independently from each other. As *OWL-Q* is an upper-level ontology, a mid-level ontology has also been created for describing domain-independent QoS attributes like *response time* or *availability*. Moreover, a low-level ontology has been created to describe those QoS attributes that are specific to the *Traffic Monitoring* application domain like *refresh time* and *coverage*.

Although the use of ontologies has great advantages, it also presents some disadvantages. One such disadvantage is the creation of different ontological descriptions of the same concept by two or more users. Concerning the QoS domain, this can be true for QoS metrics (measurement objects of QoS attributes). We argue that the state-of-the-art ontology alignment algorithms cannot be used in this case because of the use of mathematics for deriving complex QoS metrics from simpler ones. For this reason, we have devised a twofold solution to this problem: a) a semantic QoS metric matching algorithm that derives if two QoS metrics specified in different OWL-Q descriptions are equivalent and b) based on this matching algorithm, an alignment algorithm for aligning OWL-Q descriptions of WS providers and requesters. These two implemented algorithms have been evaluated based on a specific scenario taken from the *Traffic Monitoring* application domain. The experimental results showed that our alignment algorithm is quite promising.

There are two main approaches for QoS-based WS matchmaking: onto-logy-based and Constraint Programming (CP) -based. The first approach suffers from performance problems due to the use of ontology reasoners. The second approach is fast but the two research works adopting it return inaccurate results and do not provide advanced categorization of results. Inaccuracy of results is caused both by the syntactic matching of QoS terms like QoS metrics and by wrong matchmaking metrics. In addition, both approaches do not provide useful results for over-constrained QoS demands. In this thesis, we argue that the CP-based approach is better but has to be enhanced in many ways. The first enhancement is the solving technique. Based on previous research results in Mixed-Integer Programming (MIP) and CP fields, MIP must be used when only linear constraints are present on QoS demands and offers while CP must be used when non-linear constraints are also present. The second enhancement is the use of appropriate matchmaking metrics that take into account the fact that some user constraints should not be used in matchmaking. Another enhancement is the use of optimization techniques especially in cases where there are over-constrained demands. All matchmaking algorithms that have been devised and implemented were evaluated based on a framework producing random QoS-based WS descriptions. The results experimentally prove the theoretical properties of our QoS-based WS matchmaking algorithms.

As can be seen from the above description, a semantic framework for QoS-based WS description and discovery was produced in this thesis. The impacts of these results comprise the following. First of all, complete and semantic QoS-based description and alignment of WSs is provided and supported. Secondly, the results of the WS functional matchmaking process are further filtered based on user-provided QoS constraints and by selecting an appropriate matchmaking algorithm according to the nature of these constraints. Thirdly, the filtered results are ranked based on user QoS criteria, thus helping WS requesters in the selection of the best WS advertisement. Another benefit is in WS composition, where our QoS-based discovery algorithms can be exploited for the runtime selection of component services based on QoS criteria/constraints. Last but not least, by supporting the QoS-based description and runtime selection of WSs, QoS management of Web Processes (WPs) becomes more realistic to implement, thus providing many benefits for businesses, such as monitoring of WPs, triggering and evaluating alternative adaptation strategies, better resource utilization and increase of economical profits.

# Περίληψη

Η επιτυχία του παραδείγματος των Διαδικτυακών Υπηρεσιών (ΔΥ) έχει οδηγήσει στην εξάπλωση των ΔΥ. Επειδή οι ΔΥ δημοσιοποιούνται σε ενδό και δια-επιχειρησιακές αποθήκες, πρέπει να ανακαλυφθούν με βάση τις λειτουργικές ανάγκες του χρήστη. Όμως, αυτές οι αποθήκες βασίζονται στη τρέχουσα στοίβα πρότυπων τεχνολογιών ΔΥ, η οποία χρησιμοποιεί τις γλώσσες *WSDL* και *UDDI* για την περιγραφή και ανακάλυψη ΔΥ. Δυστυχώς, αυτές οι δύο πρότυπες γλώσσες βασίζονται στη χρήση συντακτικών περιγραφών των διεπαφών και άλλων γενικών και μη λειτουργικών ιδιοτήτων των ΔΥ για τη δημοσίευση και εύρεσή τους. Αυτή η κατάσταση δημιουργεί δύο προβλήματα. Πρώτον, οι προσπάθειες συντακτικής ανακάλυψης ΔΥ επιστρέφουν μη ακριβή αποτελέσματα. Δεύτερον, δεν υπάρχει τρόπος επιλογής ανάμεσα σε πολλές ΔΥ που εκτελούν την ίδια λειτουργία.

Το πρώτο πρόβλημα επιλύεται με το συνδυασμό των τεχνολογιών του Σημασιολογικού Δικτύου (ΣΔ) και των ΔΥ. Χρησιμοποιούνται οντολογίες, οι οποίες προσφέρουν νόημα στις οντότητες και στις σχέσεις ανάμεσά τους και συνεπώς οδηγούν σε σημασιολογικούς αλγορίθμους περιγραφής και ανακάλυψης ΔΥ. Αυτοί οι εμπλουτισμένοι αλγόριθμοι ανακάλυψης με τεχνολογίες ΣΔ παρέχουν πιο ακριβή αποτελέσματα. Ένα πρόσφατο αποτέλεσμα των κοινών προσπαθειών ΣΔ και ΔΥ είναι η γλώσσα *OWL-S*.

Το δεύτερο πρόβλημα μπορεί να επιλυθεί λογαριάζοντας ένα μεγάλο υποσύνολο όλων των δυνατών μη λειτουργικών ιδιοτήτων των ΔΥ που ονομάζεται *Ποιότητα Υπηρεσίας* (ΠΥ) (Quality of Service). Η ΠΥ σχετίζεται άμεσα με την απόδοση των ΔΥ καθώς επίσης και με άλλα χαρακτηριστικά των ΔΥ που σχετίζονται με την ικανότητα των ΔΥ να ικανοποιούν δηλωμένες ή συνεπαγόμενες ανάγκες. Επομένως, η ΠΥ έχει ουσιώδης αντίκτυπο στις προσδοκίες του χρήστη από μια ΔΥ και μπορεί να χρησιμοποιηθεί ως ένας παράγοντας διαφοροποίησης των λειτουργικά ισοδύναμων ΔΥ. Συνεπώς, οι περιγραφές ΔΥ πρέπει να εμπλουτιστούν με περιγραφές ΠΥ. Επιπλέον, οι αλγόριθμοι ανακάλυψης ΔΥ πρέπει να εκτελούν βασισμένο σε ΠΥ ταίριαγμα και επιλογή των διαφημίσεων ΔΥ ώστε να παράγουν πιο λίγα και ταξινομημένα αποτελέσματα.

Μια *Προσφορά ΠΥ* (ή *Απαίτηση*) ΔΥ είναι ένα σύνολο από περιορισμούς σε ορισμένες ιδιότητες ΠΥ, που τις περιορίζουν να έχουν συγκρικριμένες τιμές (οι μοναδιαίοι περιορισμοί) ή συγκεκριμένους συνδυασμούς τιμών (οι μη μοναδιαίοι περιορισμοί). Μάλιστα, οι τρέχουσες προσπάθειες μοντελοποίησης προσφορών ή απαιτήσεων ΔΥ διαφέρουν μόνο στην εκφραστικότητα αυτών των περιορισμών. Όμως, όσον αφορά την μοντελοποίηση ιδιοτήτων ή μετρικών ΠΥ, αυτές οι προσπάθειες αποτυγχάνουν. Ο πρώτος λόγος είναι ότι ο ορισμός

ιδιοτήτων ΠΥ γίνεται συντακτικά. Ένα άλλος λόγος είναι ότι το μοντέλο ιδιοτήτων ΠΥ δεν είναι αρκετά πλούσιο, μη ενσωματώνοντας τον ορισμό μονάδων μέτρησης, μονάδων συναλλάγματος, ιδιοτήτων και μεθόδων μέτρησης. Ο τελευταίος λόγος αποτυχίας είναι ότι το μοντέλο ιδιοτήτων ΠΥ δεν είναι επεκτάσιμο ή όταν επεκτείνεται, το υποκείμενο μοντέλο υπολογισμού (που αφορά το ταίριαγμα και την επιλογή περιγραφών ΔΥ) πρέπει να αλλαχθεί ως συνέπεια της επέκτασης.

Με βάση τις παραπάνω αδυναμίες των τρέχουσων μοντέλων περιγραφής ΠΥ για ΔΥ, το πρώτο μέρος αυτής της διατριβής αφιερώνεται στην διεξαγωγή έρευνας σε όλες τις προσπάθειες περιγραφής ΠΥ για ΔΥ ώστε να ανακαλύψει εκείνα τα χαρακτηριστικά και μέρη που είναι αναγκαία για ένα μοντέλο περιγραφής ΠΥ για ΔΥ. Το αποτέλεσμα της έρευνας ήταν ένα σύνολο από απαιτήσεις για μια πλούσια, σημασιολογική και επεκτάσιμή περιγραφή ΔΥ με βάση την ΠΥ τους. Με βάση τις παραπάνω απαιτήσεις, μια οντολογική γλώσσα με όνομα *OWL-Q* σχεδιάστηκε και δημιουργήθηκε χρησιμοποιώντας τον προσδιορισμό οντολογιών *OWL* και επεκτείνοντας την *OWL-S*. Η OWL-Q διαχώριζεται σε πολλά μέρη, όπου το καθένα συλλαμβάνει μια ιδιαίτερη όψη της περιγραφής ΔΥ με βάση την ΠΥ ενώ μπορεί να επεκταθεί ανεξάρτητα από όλα τα άλλα. Επειδή η OWL-Q είναι μια οντολογία ανώτερου επιπέδου, μια οντολογία μέσου επιπέδου δημιουργήθηκε για την περιγραφή ιδιοτήτων ΠΥ ανεξάρτητων από το πεδίο εφαρμογής, όπως το *χρόνο απόκρισης* ή τη *διαθεσιμότητα*. Επιπλέον, μια οντολογία κατώτερου επιπέδου δημιουργήθηκε για την περιγραφή εκείνων των ιδιοτήτων ΠΥ που είναι συγκεκριμένες ως προς το πεδίο εφαρμογής της *Παρακολούθησης της Κυκλοφορίας*, όπως είναι ο *χρόνος ανανέωσης* και η *κάλυψη*.

Αν και η χρήση οντολογιών έχει σημαντικά πλεονεκτήματα, παρουσιάζει επίσης και ορισμένα μειονεκτήματα. Ένα τέτοιο μειονέκτημα είναι η δημιουργία διαφορετικών περιγραφών για την ίδια οντότητα από δύο ή παραπάνω χρήστες. Όσον αφορά το πεδίο της ΠΥ, το παραπάνω είναι αληθές για τις μετρικές ΠΥ. Υποστηρίζουμε ότι οι τρέχουσας τεχνολογίας αλγόριθμοι ευθυγράμμισης δεν μπορούν να χρησιμοποιηθούν σε αυτή τη περίπτωση εξαιτίας της χρήσης των μαθηματικών για την παραγωγή πολύπλοκων μετρικών ΠΥ από πιο απλές. Για αυτό το λόγο, έχουμε επινοήσει μια διπλή λύση σε αυτό το πρόβλημα: α) ένα σημασιολογικό αλγόριθμο ταιριάγματος μετρικών ΠΥ που εξάγει αν δύο μετρικές ΠΥ περιγραφόμενες από διαφορετικούς OWL-Q προσδιορισμούς είναι ισοδύναμες, β) με βάση τον αλγόριθμο ταιριάγματος, ένα αλγόριθμο ευθυγράμμισης για την ευθυγράμμιση των OWL-Q περιγραφών προερχόμενων από παροχείς ή αιτούντες ΔΥ. Οι δύο αυτοί υλοποιημένοι αλγόριθμοι έχουν αποτιμηθεί με βάση ένα συγκεκριμένο σενάριο από το πεδίο εφαρμογής της παρακολούθησης της κυκλοφορίας. Τα πειραματικά αποτελέσματα έδειξαν ότι ο αλγόριθμος ευθυγράμμισης είναι αρκετά υποσχόμενος.

Υπάρχουν δύο προσεγγίσεις για το ταίριαγμα ΔΥ με βάση τη ΠΥ: οντολογικές και βασιζόμενες στον Προγραμματισμό με Περιορισμούς (ΠΠ). Η πρώτη προσέγγιση υποφέρει από προβλήματα απόδοσης εξαιτίας της χρήσης των ontology reasoners. Η δεύτερη προσέγγιση είναι αρκετά γρήγορη αλλά οι δύο ερευνητικές εργασίες που την υιοθετούν επιστρέφουν μη ακριβή αποτελέσματα και δεν παρέχουν προηγμένη κατηγοριοποίηση των αποτελεσμάτων. Η ανακρίβεια των αποτελεσμάτων οφείλεται τόσο στο συντακτικό ταίριαγμα των όρων ΠΥ όπως των μετρικών ΠΥ όσο και σε λανθασμένες μετρικές ταιριάγματος. Επιπροσθέτως, και οι δύο προσεγγίσεις δεν παρέχουν χρήσιμα αποτελέσματα για υπερ-περιορισμένες απαιτήσεις. Σε

αυτή τη διατριβή, υποστηρίζουμε ότι η προσέγγιση βασισμένη στον ΠΠ είναι καλύτερη αλλά πρέπει να επεκταθεί με πολλούς τρόπους. Η πρώτη επέκταση αφορά την τεχνική επίλυσης περιορισμών. Με βάση προηγούμενα αποτελέσματα στα πεδία του Ακέραιου Προγραμματισμού (ΑΠ) και του ΠΠ, ο ΑΠ πρέπει να χρησιμοποιείται μόνο όταν γραμμικοί περιορισμοί περιέχονται στις απαιτήσεις και προσφορές ΠΥ, ενώ ο ΠΠ πρέπει να χρησιμοποιείται όταν μη γραμμικοί περιορισμοί επίσης περιέχονται στις περιγραφές ΠΥ των ΔΥ. Η δεύτερη επέκταση αφορά τη χρήση κατάλληλων μετρικών ταιριάγματος που λαμβάνουν υπόψη το γεγονός ότι ορισμένοι περιορισμοί του χρήση δεν πρέπει να χρησιμοποιηθούν κατά το ταίριαγμα. Άλλη μια επέκταση αφορά τη χρήση τεχνικών βελτιστοποίησης ειδικά στις περιπτώσεις όπου υπάρχουν υπερ-περιορισμένες απαιτήσεις. Όλοι οι αλγόριθμοι ταιριάγματος που έχουν επινοηθεί και υλοποιηθεί αποτιμήθηκαν με βάση ένα πλαίσιο παραγωγής τυχαίων περιγραφών ΔΥ με βάση την ΠΥ. Τα αποτελέσματα αποδεικνύουν πειραματικώς τις θεωρητικές ιδιότητες των προτεινόμενων αλγορίθμων ταιριάγματος.

Όπως μπορεί να φανεί από την παραπάνω περιγραφή, ένα σημασιολογικό πλαίσιο για την περιγραφή και ανακάλυψη ΔΥ με βάση την ΠΥ αναπτύχθηκε στα πλαίσια αυτής της διατριβής. Ο αντίκτυπος του εν λόγω πλαισίου και των αποτελεσμάτων του είναι ο ακόλουθος. Πρώτα από όλα, ολοκληρωμένη και σημασιολογική περιγραφή και ευθυγράμμιση των ΔΥ με βάση την ΠΥ παρέχεται και υποστηρίζεται. Δεύτερον, τα αποτελέσματα του λειτουργικού ταιριάγματος των ΔΥ φιλτράρονται επιπλέον με βάση παρεχόμενους από τον χρήστη περιορισμούς αλλά και με την επιλογή του κατάλληλου αλγορίθμου ταιριάγματος ανάλογα με την φύση αυτών των περιορισμών. Τρίτον, τα φιλτραρισμένα αποτελέσματα βαθμολογούνται με βάση παρεχόμενα από το χρήστη κριτήρια, επομένως οι αιτούντες ΔΥ βοηθούνται στην επιλογή της καλύτερης ΔΥ. Ένα άλλο όφελος βρίσκεται στην σύνθεση ΔΥ, όπου οι αλγόριθμοί μας για την ανακάλυψη ΔΥ με βάση την ΠΥ μπορούν να χρησιμοποιηθούν για την επιλογή, κατά την εκτέλεση, των συστατικών μερών ΔΥ της σύνθετης ΔΥ με βάση κριτήρια/περιορισμούς ΠΥ. Τέλος, με την υποστήριξη της περιγραφής και επιλογής κατά την εκτέλεση των ΔΥ με βάση την ΠΥ, η διαχείριση των Διαδικτυακών Διεργασιών (ΔΔ) με βάση την ΠΥ γίνεται πιο ρεαλιστική ως προς την υλοποίησή της, παρέχοντας με αυτό τον τρόπο πολλά ωφέλη στις επιχειρήσεις όπως είναι η παρακολούθηση των ΔΔ, το έναυσμα και η αποτίμηση εναλλακτικών στρατηγικών προσαρμογής, η καλύτερη εκμετάλλευση των πόρων και η αύξηση των οικονομικών κερδών.

# Acknowledgements

Pursuing a PhD is a truly life-changing journey that is not possible to accomplish without the help, support and guidance of many people.

First of all, I would like to thank and express my gratitude to my advisor, Professor Dimitris Plexousakis, who gave me the opportunity to perform and accomplish the biggest step towards an academic or research career, which is the PhD acquisition. His continuous support encouraged and enabled me to surpass all research obstacles that came into my way. In addition, his expert guidance, stimulating encouragement, and valuable suggestions have greatly contributed to this thesis and helped me round off the PhD research successfully.

Secondly, special thanks go to the other two members of my supervising committee, Professor Nikolaou and Associate Professor Koubarakis, for their cooperation, support and guidance in many parts of this work. Moreover, I would like to warmly thank Assistant Professor Stergiou, member of my PhD examination committee, for his valuable suggestions and constructive advice in an important part of my PhD. In addition, many thanks must be given to the other members of my PhD examination committee, Professor Markatos, Associate Professor Christophides, and Assistant Professor Plebani for their helpful comments and criticism that increased the quality of my dissertation.

Furthermore, I would like to acknowledge the support of the Institute of Computer Science (ICS-FORTH) and of University of Crete (UOC) both financially (in terms of scholarships, research trips and software purchase) and in facilities. Many people from ICS-FORTH and UOC also helped me in various ways and they deserve special acknowledgements, namely Maria Moutsaki, Dimitris Aggelakis, Christos Georgis, and George Flouris from ICS-FORTH and Rena Kalaitzaki, and Yannis Agiomyrgiannakis from UOC.

Finally, I would like to thank my wife, Rena Kagiali, for the invaluable love and support that she has unconditionally given to me during the last ten years. She has always been there in both the good and the bad moments and gave me the power and inspiration to move on and fight for my goals. This PhD is without doubt devoted to her.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 General Setting

### 1.1.1 Functional and QoS-based Discovery of Web Services

Web Services (WSs) are modular, self-describing, loosely-coupled, platform and programming language-agnostic software applications that can be advertised, located and used across the Internet using a set of standards such as SOAP [GHM+07], WSDL [CCMW01] and UDDI [BCC+04]. They encapsulate application functionality and information resources, and make them available through standard programmatic interfaces. Service Oriented Architectures (SOAs) promise to enable the creation of business applications from independently developed and deployed (Web) services. A key advantage of SOAs is that they enable the dynamic selection and integration of services at runtime, thus realizing the autonomic attributes of system flexibility and adaptiveness.

However, current standard techniques only partially address the SOA vision. These techniques rely on static descriptions of service interfaces and other general non-functional service attributes for publishing and finding WSs. This situation creates two problems. Firstly, syntactic discovery efforts return results with low accuracy (precision and recall) [DMR02]. Secondly, no means are provided in order to select among multiple services that appear to perform the same function.

The first problem is solved by combining Semantic Web (SW) [BLHL01] and WS technologies. Ontologies are used, which provide meaning to concepts and relationships between them and thus lead to semantic WS descriptions and discovery algorithms. These enhanced discovery algorithms equipped with SW technologies provide more precise and recallable results. Two recent results of the joint SW and WS efforts are OWL-S [ea03] and WSMO [RKL+05], which are currently promoted as W3C submissions.

The second problem can be solved by the taking into account a big subset of all possible

non-functional properties of WSs, which is called QoS (Quality of Service) [Lie94]. QoS is highly related with the performance of a WS as well as with other features and characteristics of a WS that bear on its ability to satisfy stated or implied needs. Therefore it has a substantial impact on users' expectations from a service and can be used as a discriminating factor between functionally equivalent WS advertisements. Thus WS descriptions must be enhanced with QoS descriptions. Additionally, WS discovery algorithms should perform QoS-based filtering (matchmaking) and ranking (selection) on WS advertisements in order to produce fewer ranked results so as to assist WS requesters in finding the most suitable WS.

### 1.1.2 Main Problems in QoS-based Web Service Description and Discovery

Although there are many views of quality [DSGF03], we choose the one named "quality as conformance", which sees quality as being synonymous with meeting specifications. For example, if one service provider had promised to provide 1 Mb/s bandwidth for his stock service and he did provide this value of bandwidth at all times in his service's operation, then this service provider conforms to the promised quality of service. Quality as conformance can be monitored for each service individually and usually requires the user's experience of the service in order to measure the 'promise' against the 'delivery'. So we consider QoS of a WS as a set of non-functional attributes/characteristics that may impact the quality of the service offered by the WS. If a WS is advertised to have certain values in these QoS attributes, then it is said that this WS conforms to provide a certain QoS level.

In general, each particular domain that a service belongs to will have its own set of QoS attributes that apply to all services of that application domain [MS02, Ran03, ZBD+03]. However, certain attributes will be cross-domain attributes [FK98, AN02, Ran03, SA03, LJL+03, TGN+03]. For example, a *Traffic Monitoring* WS will involve QoS attributes belonging to the traffic monitoring application domain [CCP07]. Attributes such as "price" and "availability" are domain independent, but an attribute such as "coverage" has a specific meaning in the traffic monitoring domain. Once a set of attributes has been established for all domains a WS belongs to, it must be ensured that QoS information is collected in a fair manner by the values sent from providers, third parties or requester's feedback based on the characteristic of the quality attribute. For example,"price" can be provided by service providers, "response time" can be computed by third party active monitoring of a WS, and "reputation" [MS02] is derived from service requesters' feedback. Based on this value collection of QoS attributes, a WS violates a certain QoS level if it has promised a "better" (under a domain specific definition of "better") value from that it delivers to the requester.

As it can be seen from the above analysis of QoS, a QoS offering (or demand) of a WS is just a set of constraints on some QoS metrics that restrict them to have either certain values (for unary constraints) or certain combinations of values (for n-ary constraints). A QoS metric [TEPP02, KL03] is a concept that encompasses all measurement details for a QoS attribute. Actually, the current modeling efforts of QoS offers or demands only differ in the

expressiveness of these constraints. However, when it comes to QoS attributes modeling, these efforts fail. The first reason is because the QoS attribute definition is a syntactic one [FK98, MS02, JMS02, Ran03, TGN+03, KL03, TPP03, CMDTT05]. In result, QoS attributes like "application availability" may have different meanings to the parties that describe them (network level of the hosting system, application that implements the service). So there is a need for the incorporation of semantics in QoS attribute definition. Another reason is that the QoS attribute model is not rich enough [FK98, MS02, JMS02, Ran03, TGN+03, TPP03, CMDTT05, ZCL04, MS04, DLS05], not incorporating the definition of metrics, measurement units, currency units, measured properties (like 'time' and "quantity of information") and measurement methods [TEPP02]. This deficiency results in similar QoS attributes that are produced differently or that they use different measurement units leading to problems in the QoS attribute value collection or at the matchmaking of QoS offers and demands. So there is a need for a rich QoS attribute model. The last reason of failure is that the QoS attribute model is not extensible. Although, QoS modelers have discovered a standard set of cross-domain QoS attributes for WSs, they cannot provide standard sets of domain-dependent QoS attributes for every possible WS domain as this requires extreme modeling effort in cooperation with domain modelers. Thus, the QoS attribute model should be extensible [ZBD+03] to incorporate new domain-dependent (or cross-domain) QoS attributes for a particular domain as soon as they are discovered. In addition, this incorporation of QoS attributes must not change the underlying computation (matchmaking and selection) model [ZBD+03]. Thus, there is a need for a semantic, rich and extensible QoS model of WSs.

Based on the above inabilities in QoS-based WS description, the most prominent QoS-based WS discovery algorithms fail to perform accurate semantic QoS metric matchmaking and thus produce results with low precision and recall. Apart from this serious problem, there are also other problems that each category of approaches individually exhibits. There are two main categories of approaches for QoS-based WS matchmaking: ontology-based [ZCL04, OVSH06, GZ07] and Constraint Programming (CP) [VHS96] based [CMDTT05, DSL04]. Approaches of the first category use ontologies and rules (only [OVSH06]) in order to describe the QoS of WSs. As matchmaking is based on inferencing, these approaches suffer from performance problems due to the not-yet-matured technology used in ontology reasoners. The second category of approaches translates QoS-based WS descriptions to Constraint Satisfaction Problems (CSPs) [VHS96, Dec03] and matchmaking is performed by solving these CSPs with Constraint Solving Engines (CSE). The approaches of the second category are quite fast due to the maturity of the technology used but they return inaccurate results and do not provide advanced categorization of results. Inaccuracy of results is caused both by the syntactic matching of QoS terms like QoS metrics and by wrong matchmaking metrics. Finally, both categories of approaches do not provide useful results for over-constrained QoS demands. Thus, there is a need for a (semantic) QoS-based WS matchmaking algorithm that is accurate, has great performance and produces advanced categorization of results even in cases where the QoS demands are over-constrained.

## 1.2 Main Thesis Contributions

The above analysis has shown that there are two main problems exhibited by the current research approaches in QoS-based WS description and discovery: a) lack of a semantic, rich and extensible QoS model for WS description, and b) inability of current state-of-the-art QoS-based WS matchmaking algorithms to successfully address several important objectives including accuracy and performance. In addition, it has been claimed that the solution to the second problem depends in a significant percentage on the solution adopted to the first one. The scope of this thesis is to address [Kri05] both of these problems in the best possible way also taking into account their interconnection. To this end, the following three main contributions have been achieved: *semantic QoS-based WS description*, *alignment of QoS-based WS specifications* and *semantic QoS-based WS discovery*. Our first contribution starts by conducting an extensive research in QoS-based WS description so as to produce a set of requirements that must be satisfied by a semantic QoS model for WSs. Based on these requirements [Kri05, KP07b, KP08c], we carefully design and implement an upper ontology for QoS-based WS description, which is called OWL-Q [KP06, KP08c, KP07a, KP08e]. This ontology combines the best disciplines, requirements and parts of all related research efforts in order to describe in a syntactic and semantic way all possible parts of QoS for WSs. It is an ontological description carefully designed into several facets that can easily be extended and enriched. This ontological description also complements OWL-S, a World Wide Web Consortium (W3C) submission for semantic functional WS description, by subsuming OWL-S concepts (the sub-concept of the OWL-S concept is a QoS concept) or relating them to QoS concepts. As OWL-Q is just an upper-level ontology, a mid-level ontology has also been produced for those QoS attributes that are common across application domains. Finally, a low-level ontology has been produced for those QoS attributes that are specific to the *Traffic Monitoring* application domain. This application domain has been chosen in order to experimentally evaluate our developed algorithms in realistic scenarios where QoS-based WS descriptions are randomly created based on constraints on metrics of specific QoS attributes.

Although the use of semantics provides users with the ability to specify in a clear, consistent and machine-processable and interpretable manner the concepts of their domain, it cannot prevent some problems [MFRW00] from happening. One such problem is based on the fact that people have different conceptualization of the same entities [BBB+04]. In result, users may produce different ontological specifications of the same concept. Concerning the QoS domain, this can be true for QoS metrics and not for QoS attributes or measurement units that are more or less standardized. This argument is also strengthened by the fact that the same QoS metric can be produced either from high or low level readings of the instrumentation of the system hosting the WS (depending on this system capabilities). In the former case, this QoS metric is produced from other QoS metrics while in the latter case, this QoS metric is a resource metric. Based on the above reasons, we argue that the state-of-the-art ontology alignment algorithms [BBB+04] cannot be used in our case because of the use of mathematics for deriving complex QoS metrics from simpler ones. For this reason, our second contribution consists of the following two algorithms: a) a semantic QoS metric matching algorithm [KP06, KP07d, KP07c, KP07e, KP07a, KP08e, KP08c] that derives if

two QoS metrics specified in different OWL-Q descriptions are equivalent and b) by using this matching algorithm, an alignment algorithm [KP07d, KP07c, KP07e, KP08e, KP08c, KP08a] for aligning OWL-Q descriptions of WS providers and requesters based on their specified QoS metrics and on new metrics created by past measurements produced by WS monitoring systems. These two implemented algorithms have been evaluated based on a specific scenario taken from the *Traffic Monitoring* application domain. The experimental results have showed that although the execution time of the alignment algorithm (that encompasses the metric matching algorithm) is not very satisfactory, this is compensated by the increase in the accuracy of the matchmaking algorithms. The last observation leads to the conclusion that not only the first problem (lack of semantic, rich and extensible QoS model) described in the previous section is solved, but its solution also increases the quality of our solution to the second problem (better QoS-based WS matchmaking algorithm). This means that our first two contributions really achieve their goal.

As can be seen from the two previous paragraphs, our first two contributions (OWL-Q and alignment algorithm) try to produce QoS-based WS specifications that have the maximum amount of common QoS metrics. This result is in favor of the CP-based approach in QoS-based WS matchmaking. The main reason is that this approach transforms the common QoS metrics of a QoS offer and demand to the same CSP variables used to solve the CSP problem created by the constraints of the QoS offer and demand. By considering also the fact that the CP-based approach is faster than the ontological one, we argue that the CP-based approach is better and must be chosen as a starting point in solving our second research problem (better matchmaking algorithm). So our third contribution is that we enhance the CP-based approach, especially the work of [CMDTT05] that uses a more correct matchmaking metric than [DSL04]. However, due to the nature of the value types of the QoS metrics, our first step in this third contribution has been to perform a survey on the available solving techniques as CP has some difficulties in dealing with real-valued variables. So based on the research results of the MIP and CP fields, we conclude that MIP must be used when only linear constraints are present on QoS-based WS specifications while CP must be used when also non-linear constraints are present. Our conclusion has been experimentally proven [KP08d] by a comparison of these two techniques on linear and non-linear CSPs (MIPs) created according to the criteria specified in [CMDTT05] and by using the matchmaking metric of conformance [CMDTT05].

The second enhancement is the use of an appropriate matchmaking metric [KP07d, KP08e, KP07c, KP07e, KP08b, KP08d] that takes into account the fact that some user constraints should not be used in matchmaking. For example, there is no reason why there must be an upper bound on *availability* for WS requester constraints while both bounds on *response time* must be respected as the requester's system may not be able to process a very fast provided output (e.g. in case of video streaming WSs and WS requesters having mobile phones). Our random [KP08b, KP08d] and semi-random evaluations (based on the domain of Traffic Monitoring) have showed that there is a significant decrease in matchmaking time if our new matchmaking metric is taken into account with respect to the one of conformance.

Another enhancement is the use of optimization techniques especially in cases where there are over-constrained QoS demands. In many cases, users over-estimate the capabilities of

a system and pose unrealistic constraints to it or to discovery brokers in order to find it. In our situation, based on the previous observation, the QoS capabilities of WSs are overestimated by WS requesters. So the state-of-the-art QoS-based WS matchmaking systems fail to produce any useful result to those WS requesters as the CSPs solved are all infeasible. In this dissertation, we will show two different optimization algorithms [KP07d, KP08e] used to deal with this case. Each one has its strengths and weaknesses and their selection depends on user preferences on matchmaking time and accuracy. This is also proven by our random [KP08d] and semi-random evaluations.

Finally, the last enhancement concerns the situation where non-linear constraints are present in the QoS-based WS advertisements so the CP-based approach must be used. The presence of non-unary and non-linear constraints is not utopian. As a concrete example, the price of a WS can be expressed as non-linear function of many QoS metrics, e.g. $price = \sqrt{((1 - responseTime)/0.01)^2 + (availability/0.01)^2}$ where $responseTime$ and $availability$ take values in $(0.0, 1.0]$. Thus, in the case of non-linear constraints, we have devised and implemented two new algorithms that take into account our new matchmaking metric and techniques from a specific subarea of CP called explanation-based programming (eCP) [VJ05, RvBW06]. The random evaluation of these two algorithms against the simple ones (without eCP) shows the significant gain in matchmaking time. The selection of these two new algorithms depends again on user preferences concerning matchmaking time and advanced categorization of results.

Thus, this thesis proposes the use of two different techniques for QoS-based WS matchmaking depending on the linearity of the QoS constraints of WS specifications. Concerning, QoS-based WS selection, some of our proposed algorithms already solve optimization problems and thus also perform the selection subprocess of WS discovery. On the other hand, the algorithms, that do not offer this functionality and is not clear from their results which QoS offer is better, should be accompanied with a QoS-based WS selection algorithm. To this end, we propose a QoS-based WS selection algorithm [KP06] that computes the score of each advertised QoS offer by solving two optimization problems - a) one finding the score of the worst performance of the offer, and b) one finding the score of the best performance of the offer - and then computing the weighted average of the two optimization results.

## 1.3 Impact of Thesis

Based on the content of the previous sections, we argue that the result of our thesis is the production of a complete semantic framework for QoS-based WS description and discovery. This framework can be used as a black-box by WS registries [KFS06] that already perform functional WS description and discovery in order to filter out and rank the results of their discovery process, thus helping WS requesters in the selection of the best WS advertisement.

Apart from WS registries, our framework can be used by WS negotiation frameworks [CFK+02, DNDPG+07, WA03, CLG+06, CP05]. These frameworks are used in order to establish a Service Level Agreement (SLA) [KL03] between the WS requester and the provider of the best WS found. SLAs are documents that specify the trusted third-party entities mon-

itoring QoS levels delivered, the penalties that will be imposed when one of the two main parties does not keep up with its promises, and the validity period of the promises. So SLAs give confidence and trust to the entities providing and consuming the service and lead and guide the process of WS Execution. If a negotiation framework cannot establish an SLA between the entities providing and consuming the service because of their conflicting requirements, then the negotiation fails and the next WS from the returned list of the QoS-based WS discovery phase is contacted. Thus, our framework provides the appropriate input to the WS negotiation frameworks and this input along with the negotiation rules and preferences of each party drive the WS negotiation process.

Another benefit is in WS composition, where the most suitable from our QoS-based discovery algorithms can be exploited for the design-time and runtime filtering and selection of component services based on local QoS constraints [BSND02]. If local constraints are only required, then the invocation of our algorithm for each WS component is enough for producing the (part of the) concrete composed WS. However, when global constraints [ZBD+03] are also present, our algorithms can be used as a first step for filtering the QoS offers of those WSs that functionally match each task (WS component). Then the execution of a QoS-based WS composition algorithm [ZBN+04, JRGM04, GPEV05, YZL07, MDSR07, BPB08] is also required in order to produce the (part of the) concrete composed WS by also respecting the global constraints.

Our framework can be easily extended to also provide context-aware WS description and discovery [BPvS+04]. This type of discovery is essential in application domains where mobile WSs are offered or where the application of the WS requester is hosted in a small device (PDA, mobile phone, etc). The extensions that have to be made concern mostly the WS description part where some attributes must be associated to context metrics (like precision) and mid and low-level ontologies of context metrics and attributes should be defined. Matchmaking is not actually affected. On the contrary, the WS selection part must be enhanced in order to produce scores that take into account user-provided weights on context metrics.

Last but not least, by supporting the QoS-based description and runtime selection of WSs, QoS management of Web Processes (WPs) [CSM+04] becomes more realistic to implement, thus providing many benefits for businesses like monitoring of WPs, triggering and evaluating alternative adaptation strategies, better resource utilization and increase of economical profits.

## 1.4 Outline of Dissertation

The rest of this dissertation is organized in the following way:

- Chapter 2 exemplifies the main concepts and notions that appear in the WS Description and Discovery processes. In addition, it provides a small introduction to the two solving techniques used for QoS-based WS matchmaking: MIP and CP.

- Chapter 3 provides a comprehensive analysis of the conceptualization of QoS for WSs

and of the main domain-independent QoS attributes. In addition, an example of domain-dependent QoS attributes in the Traffic Monitoring application domain is provided. Finally, the role that QoS can play in the management of WSs is elaborated.

- Chapter 4 analyzes the requirements that must be satisfied by the QoS-based WS description and discovery processes.
- Chapter 5 provides a literature review of the main research approaches in QoS-based WS description and discovery and reveals their main contributions and deficiencies.
- Chapter 6 is the cornerstone of this dissertation as it analyzes theoretically the main contributions of this thesis. It is separated into three sections, each focusing on one of our three contributions.
- Chapter 7 unveils the architecture and main components of this thesis' framework and shortly describes the tools used to implement it. In addition, the biggest part of this chapter is dedicated to experimentally evaluating the matchmaking algorithms constituting one of this thesis main contributions. This experimental evaluation is performed randomly and semi-randomly based on a specific testing framework developed. It must be noted that the semi-random evaluation was actually performed by creating random QoS specifications on QoS metrics of QoS attributes taken from the Traffic Monitoring application domain.
- Chapter 8 concludes this dissertation and draws directions for further research.

In each chapter, there will be an introductory text followed by the main content (consisting of one or more sections) and finally a last section concluding this chapter.

# Chapter 2

# Preliminaries

## 2.1 Introduction

The scope of this chapter is to equip the interested reader with the appropriate knowledge in order to be able to: a) follow up with the remaining chapters of this dissertation; b) categorize our thesis contribution according to the current research on WSs; c) better understand our main contributions, the effort taken to achieve them and their outcome and benefits; d) get acquainted with the research carried out in two different research fields; and e) fix the terminology used throughout this dissertation in order to make it self-contained. The rest of this chapter is separated into two main sections. Section 2.2 introduces the main concepts and their roles in the WS Description and Discovery processes. Moreover, it analyzes the main requirements that these two processes impose in order to be achieved. The next and final section 2.3 analyzes the two main techniques that are used by our thesis in order to perform QoS-based WS matchmaking and selection: MIP and CP.

## 2.2 Web Service Description and Discovery

This section explains what a WS is, in which computing paradigm is utilized and how a software system should be designed in order to provide WSs. This is realized in the first three subsections. Then, it defines and analyzes the WS Description and Discovery processes and reveals the requirements that they pose. This is fulfilled by the last two subsections.

### 2.2.1 Web Service Definition

The term *Web Services* is used very often nowadays but not always with the same meaning. Existing definitions [ACKM03] range from the very generic and all-inclusive to the very

specific and restrictive. Often, a Web Service is seen as an application accessible to other applications over the Web [Fis04]. This is a very open and generic definition stating that anything that has a URL is a Web Service. For instance, it can include a CGI script or a program accessible over the Web with a stable API, published with additional descriptive information on some service directory.

A more precise definition is provided by the UDDI consortium, which characterizes Web Services as "*self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces*" [Con01] . This definition is more detailed, emphasizing on the need for being compliant with Internet standards. In addition, it requires the service to be open that is to have a published interface that can be invoked across the Internet. In spite of this clarification, the definition is still not precise enough because it is not clear what is meant by a modular, self-contained business application.

A better definition of Web Services is provided by the World Wide Web Consortium (W3C) and is the following: "*a software application, identified by a URI, whose interface and bindings are capable of being defined, described, and discovered as XML artifacts. A Web Service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols*" [ABFG02].

The W3C definition is quite accurate and also explains how Web Services should work. The definition stresses that Web Services should be capable of being "defined, described, and discovered", thereby clarifying the meaning of "accessible" and making more concrete the notion of "Internet-oriented, standards-based interfaces". It also states that Web Services should be "services" similar to those in conventional middleware. Not only they should be "up and running", but they should be described and advertised so that it is possible to write clients that bind and interact with them. In other words, Web Services are components that can be integrated into more complex distributed applications. The W3C also states that XML is part of the solution. Indeed, XML is so popular and widely used today that is going and should be the data format used for many Web-based interactions.

Note that even more specific definitions exist. For example, the online technical dictionary Webopedia[1] defines a WS as "*a standardised way of integrating Web-based applications using the XML, SOAP, WSDL, and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available*". This definition references the specific standards that could be used for performing binding and for interacting with a Web Service. These are the leading standards of SOAP, WSDL, UDDI currently used in WSs. However, these standards do not constitute the essence of Web Services technology: the problems underlying Web Services are the same regardless of the standards used.

[1] `http://www.webopedia.com/`

### 2.2.2 Service Oriented Computing and Architecture

Service-Oriented Computing (SOC) [Pap03] is the computing paradigm that utilizes (Web) services as fundamental elements for developing applications. Since WSs may be offered by different enterprises or individuals and communicate over the Internet, they provide a distributed computing infrastructure for both intra and cross-enterprise application integration and collaboration. This application integration and collaboration is achieved only if WSs are technology-neutral, loosely-coupled and they support location transparency. There are two types of WSs: *simple* and *composite*. Simple WSs are the building blocks for composite WSs and provide simple functionality. For instance, we can have a simple mathematical WS taking as input a mathematical expression and returning as a result the expression's evaluation. Composite WSs involve assembling existing WSs in order to accomplish a specific high goal or objective. For example, an enterprise could assemble existing intra-enterprise WSs (applications wrapped as WSs) and some external WSs (outsourced applications or utilities wrapped as WSs) in order to provide a complete distributed e-business application like e-procurement. Thus, composite WSs are publicly available, technology-neutral, loosely-coupled and location-transparent applications that are developed following the computing paradigm of SOC.

To build integration-ready applications the service model relies on the service-oriented architecture (SOA). SOA is a logical way of designing a software system to provide services to either end-user applications or other services distributed in a network through published and discoverable interfaces. The basic SOA defines an interaction between software agents as an exchange of messages between service requesters (clients) and service providers. Clients are software agents that request the execution of a service. Providers are software agents that provide the service. Agents can be simultaneously both service clients and providers. Providers are responsible for publishing a description of the service(s) they provide. Clients must able to find the description(s) of the services they require and must be able to bind to them.

In SOA, a relationship of three kinds of participants (the *service provider*, the *service discovery agency*, and the *service requester* (client)) is inherent and realized by the interactions that involve the *publish*, *find* and *bind* operations [BHM+02] (see Fig. 2.1). These roles and operations act upon two WS objects: the WS description and the WS implementation. The way these roles interact and the sequence of these interactions are clarified by the following typical service-based scenario: A service provider hosts a network accessible software module (i.e. an implementation of a given WS). The service provider also defines a WS description of the WS and publishes it to a client or service discovery agency (i.e. a registry or repository like UDDI). Through this agency, any WS description is published and made discoverable. The service requester uses a find operation to retrieve the WS description from the discovery agency and uses the WS description to bind with the service provider in order to invoke the WS implementation. Service provider and service requester roles are logical constructs and a WS may exhibit characteristics of both.

In the sequel, we are going to investigate the content of the WS description and to analyze the steps and algorithms involved in the WS discovery process. In addition, the requirements posed upon the three basic SOA roles by the WS description and discovery

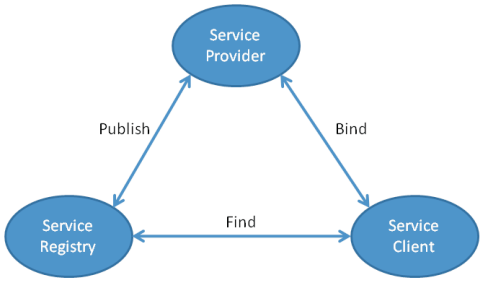



Figure 2.1: Service Oriented Architecture

processes are going to be exposed.

### 2.2.3 Web Service Description

This subsection explains the WS Description process. In particular, it defines the WS Description process and analyzes the requirements that it imposes. These two tasks are realized by the "Definition" and "Requirements" sub-subsections.

#### Definition

*Web Service Description* is the process where a) a service provider describes its service functionality and properties to potential clients (maybe through service brokers or registries) as a service offer b) a service client describes what is expecting from a WS to a service broker/registry or to a sophisticated search (crawling) engine. Service descriptions are used to advertise/query the WS capabilities, interface, behaviour, and quality. Publication of such information provides the necessary means for discovery, selection, binding, and composition of Web Services. In particular, the service capability description states the conceptual purpose and expected results of the WS. The WS interface description publishes the WS's signature (its input/output/error parameters and message types). The (expected) behaviour of a WS during its execution is described by its WS behaviour description. Finally, the Quality of Service (QoS) description publishes important non-functional WS (quality) attributes, such as service metering and cost, performance metrics (e.g. response time), security attributes, (transactional) integrity, reliability, scalability, and availability. A WS Description should also describe the parameters within which both the service provider and client are willing to negotiate. *Web Service negotiation* involves the interaction between a client and

one or more service providers identified through the discovery process or already known to the client. It has the aim to agreeing the terms and the conditions for the supply of the WS.

### Requirements

In the beginning, this subsection unfolds the general requirements that are imposed by the WS Description process. Then, for each of the four parts of a WS Description, additional and part-specific requirements are revealed.

**Common Base Language** As stated in the above definition, a Web Service Description consists of four or five facets/aspects. Each facet can be described by a separate language. This fact imposes the definition/existence of a *Common Base Language* [ACKM03]. This language can be used as a basis for specifying all the languages necessary to describe the different aspects of a service. XML is used for this purpose, both because it is a widely adopted and commonly accepted standard and because it has syntax flexible enough to enable the definition of service description languages and protocols.

**Machine and Human Understandable** A Web Service Description needs to be understandable by humans as well as machines [MSZ01, PTB03, DSGF03]. This means that each service attribute must be described at both the syntactic and semantic level. Syntactic information is concerned with the implementation aspects of a service and thus tailored towards the programmer's requirements. Semantic information is concerned with the conceptual aspects of a service aiming to facilitate end-users by shielding off the lower technical details, as well as to facilitate developers to find services that best match their needs and to enable automatic service selection and composition. Depending whether the service requester is an end-user, a developer or a machine, different kinds of service description are required. For the end-user, only semantic description is needed, whereas developers or machines need both semantic and syntactic information. Thus, the language used for service description needs to provide constructs that enable the description of functional, non-functional and behavioral information in a semantic as well as in a syntactic form of representation.

**Must Support Inferences** The description language should also support inferences on descriptions. This means that automated reasoning and comparison on descriptions should be possible and efficient. The above requirement is crucial for automated service discovery (especially in matching service offers with service requests) and composition. One way to achieve it is by using ontologies [UG96]. Ontologies have been developed in order to facilitate knowledge sharing and reuse. They provide greater expressiveness when modelling domain knowledge and can be used to communicate this knowledge between people and heterogeneous and distributed application systems. Ontologies are based on a class system. Classes are organized in hierarchies and offer extensibility through subclass refinement. This enables automated reasoning on taxonomies of concepts.

**Must Support Four Types of Semantics** In Web services domain, semantics represented by the semantic metadata can be classified into the following types [She03, SSM+03], namely:

- Functional Semantics
- Data Semantics
- Non-Functional Semantics
- Execution (Behavior) Semantics

These different types of semantics can be used to represent the capabilities, requirements, effects and execution patterns of WSs.

The vision of SOC can only be realized when appropriate services are discovered based on the functional requirements. Several semantic Web service discovery algorithms have assumed that the WSs' functionality is characterized by their inputs and outputs. Hence these algorithms look for semantic matching between inputs and outputs of the services and the inputs and outputs of the requirement. Semantic matching of inputs and outputs is not sufficient for discovering relevant services. For example, two services can have the same input/output signature even if they perform entirely different functions. A simple mathematical service that performs addition of two numbers taking the numbers as input and produces the sum as output will have the same semantic signature as that of another service that performs subtraction of two numbers that are provided as input and gives out their difference value as output. Hence matching the semantics of the service signature may result in high recall and low precision. The solution to the above problem is the annotation of Web services with functional semantics. This can be done by having a functional ontology in which each concept/class represents a well-defined functionality. The intended functionality of each WS can be denoted by annotating it with concepts of this ontology. Thus, if WSs are annotated using a functional ontology, then service discovery algorithms can exploit these annotations for a finer and more accurate result set.

All WSs take a set of inputs and produce a set of outputs, which are all represented in the signature of the corresponding WSs' operations in a WSDL file. However the signature of an operation provides only the syntactic and structural details of the input/output data that are used for service invocation. For a more effective discovery of WSs, semantics of the input/output data has to be taken into account. Thus, if the data involved in WS operation/signature is annotated using an ontology, then these data annotations can be used in matching the semantics of the input/output data of the WS with the semantics of the input/output data of the requirement. Data semantics are usually represented by domain ontologies, which describe the concepts and the relations involved in a particular domain of knowledge.

After discovering a list of functionally-equivalent WSs, the most suitable WS must be selected. Each WS can provide different quality (non-functional) levels. Hence WS selection is about locating the WS that provides the best quality criteria match. WS selection demands management of QoS metrics for WSs because it is an important activity in WS

composition [CS03]. Web services in different domains can have different quality metrics, which are called Domain Specific QoS metrics. There can also be some QoS criteria that can be applied to services in all domains irrespective of their functionality or specialty. These criteria are called Domain Independent QoS metrics. Both kinds of QoS metrics need shared semantics for interpreting them as intended by the service provider and requester. This can be achieved by using an ontology that defines the domain specific and domain independent QoS metrics.

WS execution semantics encompasses the constructs of message sequence (e.g., request-response, request-response), conversation pattern of WS execution (peer-to-peer pattern, global controller pattern), flow of actions (sequence, parallel, and loops), preconditions and effects of WS invocation, etc. Some of these constructs may or may not be made publicly available, depending on the nature of the WS application and the policies of the organization that owns it. In any case, the execution semantics are not the same for all WSs and hence they should be verified before WS execution. The execution semantics of WSs can be represented by traditional formal mathematical models (Process Algebra [BPS01]), concurrency formalisms (Petri Nets [ACKM03], state machines [HU79]) and simulation techniques [VSJ01]. Formal modelling for workflow scheduling and execution is also relevant [ASSR93]. By using execution semantics, Web processes (i.e. composite WSs) need not be statically bound to component WSs. Instead, based on the functional, data, QoS and execution (preconditions and effects) semantics a list of appropriate WSs can be provided, then QoS semantics can be used to select the most appropriate WS, and then by using execution semantics the WS can be bound to a process and the corresponding Web process execution can be monitored.

Now, additional requirements are going to be analyzed, separated into four subsections. Each subsection reveals the requirements of the corresponding part of a WS Description (interface, capability, QoS and behavior).

**Interface Description Requirements** Service description in conventional middleware is based on interfaces and interface definition languages (IDLs). In WSs, interface definitions resemble CORBA-like IDLs, although there are some differences between the two [ACKM03], like the availability of different interaction modes in the interface definition language and the XML schema-driven type system. In addition, since WSs lack an implicit context, their description needs to be more accurate.

As WSs are still software, we can use the requirements for ordinary software interface descriptions as the basis for WS interface descriptions [Oak03]. This is done to ensure that the description of WS interfaces is at least as comprehensive and appropriate as the description provided for other types of software. In this section, we give an overview of the interface description elements presented in [CBB+02] and comment how these requirements apply in the WSs context. Each element is necessary to ensure that the developers implementing the interface, and third parties using/invoking the interface, can fully understand what the interface requires, what it provides and its constraints.

1. **Interface identity**: We can uniquely identify an interface by its name and by its version numbers if appropriate. A unique identity for an interface is particularly

important when many implementations of the same interface are supplied or if different interfaces to the same WS are provided for different classes of users. The identity can be used by WS advertisements in registries to indicate that the WS complies with a particular interface.

2. **Resources provided**: These are the operations or methods provided by the interface. The description of resources should be sufficient to aid the discovery, evaluation and selection of services that meet the users' needs. Each resource needs to describe its:

   (a) **Syntax**: The signature including its name and the logical data-types of arguments.

   (b) **Semantics**: A description of what happens when the resource is used i.e. what is visible to the user, and what are the restrictions on use of the resource. For example, the semantics describe the assignment of values to data that the user can access; changes in state, either to this, or another, element; the events signalled and messages sent by the resource; details of how other resources will behave differently when this one is used; and the execution style, whether the operation is atomic, interruptible or suspendible. The semantics of the operations is perhaps the most important part of the description in terms of enabling automated interaction. The semantics will be necessary in most phases from evaluation and selection to interaction.

   (c) **Usage restrictions**: Similar to pre-conditions, these state the assumptions about the environment that must be true, or describe the side effects of the operation. Exceptions should also be described here, detailing how errors are handled. For example, the number of retries or the meaning of returned status indicators.

3. **Locally defined data-types**: This section describes how to declare variables, constants and literal values of the data-types defined and used in the interface, and the operations, comparisons and conversions that can be performed on instances of those types. Data-types in the web context can be viewed as both traditional programming language constructs and XML documents. In the case of XML documents, a reference to the document schema or a document template should be provided to allow creation of required documents or to enable understanding of the documents supplied by resources.

4. **Error handling**: The errors that can be raised by the resources of the interface and error handling behaviour. This information will be necessary during the interaction phase to determine the cause of unexpected results and how to deal with them. Compensating actions could also be defined in order to enable transactional behaviour [LLA+03]. However, the inclusion of compensation information should be optional. In addition, locating the compensating service could be quite involved in case compensational information is absent because reasoning mechanisms must be developed to figure out the required capability and constraints to compensate the service. Even if

the compensating capability is described, valuable time could be spent in discovering the compensating service determined by the compensating capability.

5. **Variability provided by the interface**: Details of what configuration is possible and range of allowable values for each configuration parameter should be provided. In addition, a description of how configuration affects the semantics of the interactions must be included. WS users will be of many different types, operating in different contexts, each with different capabilities. Consequently, web service descriptions must provide the facility to describe what can be configured and how that configuration can be achieved. In addition to operation attributes, there are other aspects of service delivery that could be configurable such as, the interaction mechanism or the security and transaction management protocols.

6. **Quality attributes of the interface**: Quality attributes include such things as the level of performance and reliability that the interface provides. WSs may be selected based on their certified conformance to specific standards, or their ability to provide various quality-of-service attributes.

7. **What each interface element requires**: Either, specific named resources (described as above with syntax, semantics and restrictions), or other preconditions or assumptions about the environment. The usage restrictions (or non-functional requirements) of the operations will be used in the evaluation and interaction phases to determine if the constraints can be satisfied by the user.

8. **Rationale**: The motivation for the design, the constraints, compromises and alternatives considered. This is mainly for the benefit of developers implementing the interface.

9. **Usage guide**: the protocol of interaction or patterns of use for the entire interface. This information will be used during the interaction phase to ensure the correct order of interaction. This information could also be used to describe how to initiate a dialogue with the service, how to negotiate with or configure the service, and how to manage the operation of the service. Alternatively, the information may be used to select services that provide compatible interaction mechanisms.

The interface requirements described here can be considered the minimum requirements for the interfaces of WSs.

**Capability Definition Requirements** A set of criteria for evaluating capability description languages were described by Sycara et.al. [SWKL02] in reference to agent capabilities. These requirements include expressiveness, abstraction, support for inferences, ease of use, application on the web, and avoiding reliance on keyword extraction and comparison. According to [OtHE03], these high level criteria are also relevant in the context of semantic WSs but they do not address the specific requirements of dynamic WS discovery. To address these requirements, a capability language should provide:

1. The ability to declare what action a service performs
2. The ability to allow a capability to have different sets of inputs
3. The ability to declare preconditions and effects in some named rule definition language
4. The ability to describe objects that are not input but are used or affected by the capability
5. The ability to refer to ontological descriptions of the terms used in the description and thus place the use of the terms in context
6. The ability to make explicit the domain or context in which the service operates
7. The ability to classify capabilities based on aspects of the description enabling exact or partial matches between required and provided capability descriptions

Different WSs can provide the same capability, e.g. book a flight, and different capabilities may be provided by the same WS. In this sense, capabilities must be naturally described separately from specific WS descriptions [FB02, Hos02b, MDCG03] for several reasons:

- *Express generic functionality*: Several WSs offering the same functionality but with different specific refinements should be related to the same generic high-level capability. This approach is related to concepts taken from Problem Solving Methods (PSM) research, and it inherits some of their advantages, as the ones highlighted in [FM01, FMvH$^+$03].
- *Use different terminologies*: Refinements done by a given WS can be expressed using a different terminology from the one used to describe their capability, thus increasing flexibility, as requiring the use of the same terminology is sometimes unrealistic.
- *Allow a given service to present many different capabilities*, while exposing only one service description.
- *Support discovery process*: Discovery first needs capability descriptions. Further filtering based on actual input, output and requirements is performed in subsequent steps. Thus, separating capabilities and linking them to WS descriptions follows the steps of the discovery process.

**Non-Functional Definition Requirements** According to [DOH$^+$01, OEtH03], non-functional properties are constraints exhibited over the functionality of the service. This kind of properties includes temporal and spatial availability, channels, charging styles, pricing factors, settlement models, settlement contracts, payment, service quality, trust and ownership. Each of these properties is deserving of a separate paper because of the complexity involved in accurately describing them. In this dissertation, we are going to analyze the set of non-functional properties that characterize the service quality of a WS. More specifically, we are going to closely examine the requirements of QoS-based WS description. Then, we are going to analyze a language we have devised that satisfies these requirements.

**Behavioral Definition Requirements** The behavioral definition semantics were covered previously in the part describing execution semantics. Here, we give a brief overview of what is needed stressing the separation between private and public process models.

Business collaborations require long-running interactions driven by an explicit process model. Thus, a service must explicitly model its business process which will contain decision mechanisms for the execution of the service. However, no internal details of the organization business logics should be made publicly available. Therefore, while a process model and its data and control flow must be designed explicitly to ground the execution and public behavior of a given service, it must not be exposed.

Nevertheless, the external behavior of the service in terms of message interchange must be made public in order to enable automatic inter-operation of the service with any other service. In this sense, the public description of the service must include a conversational interface which allows inter-operation while not revealing any private detail.

#### Related Work and Main Problems

None of the current approaches for WS Description satisfies all the requirements that were imposed. The main problem seems to be the lack of semantics, especially for the standard approaches (UDDI and WSDL). But even if semantics is incorporated, the WS description model is not rich enough as one or more aspects are not very-well covered if not at all. From now on, we will focus on the non-functional aspect of WS Description. For a complete reference to the related work and its limitations, please see [Kri04].

### 2.2.4 Web Service Discovery

This subsection explains the WS Discovery process. In particular, it defines the WS Discovery process and its processing steps. For each processing step, the requirements that it imposes are unfolded.

#### Definition

Clients use WS discovery to locate appropriate WSs, according to their functional and non-functional requirements and selection criteria. Using this process, a client identifies those potential service providers whose offerings meet its functional needs and who are prepared to negotiate within some acceptable bounds. Discovery can involve the recursive use of other services, including brokers, and will result in a list of candidate services and providers. Service negotiation involves the interaction between a client and one or more of the service providers identified through the discovery process or already known to the client. This negotiation aims to achieve agreement on the terms and conditions for supplying the service.

### Discovery Processing Steps and Their Requirements

In this subsection, the actual steps performed by the components/roles of the SOA architecture are defined which lead to a successful discovery process. These steps are classified into: pre-matchmaking steps, the matchmaking step and selection or brokering step [Hos02b]. The matchmaking step is the step where the needs of a client are matched against available offers (gathered either by the client himself or by a registry broker). The pre-matchmaking steps are: description, presentation and publication. For each step defined, the requirements that this step imposes to the Web Service components/resources are analyzed.

**Web Service Description** This step is already analyzed at the section 2.2.3 of this report. What is left to say is that both service providers and service clients must describe their services/needs. Maybe service clients describe their needs in natural language. This is not a serious problem as sophisticated language processing techniques exist nowadays that can map a natural language sentence to concepts and relations of an ontology (e.g. a service or domain ontology). We also assume that WS offers and requests are described in a complete and semantic Web Service description language like OWL-S that will enable the use of (DL) reasoners at the matchmaking process.

**Web Service Presentation** This step is not obligatory as Web Service Description and Publication are the most necessary steps to be taken before the matchmaking process. However, it may be useful in cases where a client wants to retrieve specific parts of descriptions of interested services or in cases where a registry/client wants to retrieve the current (up-to-date) description of a service. So it would not be a burden for a service provider to present his service.

Hence, a service description of a service provider should be presented to potential clients. For this presentation, two things are needed: a) a unique identifier for the service so as to enable global context querying of clients; b) a description retrieval mechanism should be associated with each service identifier. This service presentation can be regarded as another WS interface. In this way, WS's functionality and its presentation are separated into two different interfaces.

WS descriptions should be made publicly available so as to enable every client to be able to access them in the Web. This can be realized by choosing the WS's unique identified to be a URL and the WS's description retrieval mechanism to be HTTP(s). If all WSs are armed with this capability and they have published it as an interface, their current descriptions will be available to all potential clients across the Web.

Taking one step further, instead of retrieving the whole service description, only the values of all the dynamic (QoS) properties could be retrieved in the same manner. To be more specific, a separate interface for each of these dynamic properties could be added and presented. Maybe for the evaluation of some of these dynamic properties, some input should be given to the service. But the type of this input can be described in the interface corresponding to the appropriate dynamic property. In this way, the matchmaking/selection steps will be more successful, as the matchmaker/broker will retrieve the up-to-date values of these properties in order to rank or select a service according to service client needs. In

addition, registries will not be burden with updates to service descriptions for these dynamic properties. Only other updates will be considered useful and thus should be propagated by the service providers to the publishing registries.

**Web Service Publication** Publication is the process of making the presence of services, resources, user communities and other service-related objects known and reachable to potential clients by the registration and storage of these object descriptions in one or more well known registries. The potential clients of a registry are service providers, service requesters and other registries. Service providers use the registry in order to publish information about their business and about what services they are offering. Occasionally, they also update the information they have published because some aspects of it may have changed. In addition, service providers want parts of their published information not to be available to some clients or to some other service providers. Service requesters use service registries for querying about businesses and services. They expect that registries give the most accurate and up-to-date results concerning their queries as quickly as possible. Service registries use other service registries to exchange information or to give away their ownership of some information for replication or distribution purposes.

To meet the above functionalities and requirements, a service registry must be controlled by sophisticated Data Base Management System (DBMS) mechanisms including:

- *Security mechanism*: It is used to authorize and authenticate both service providers, that want to use the registry's publishing capabilities, and service requesters, that want to use the registry's querying capabilities. Other service registries could also be checked

- *Access control and concurrency mechanism*: It is used for two reasons. The first is to let service providers to hide some part of their information from service requesters. The second reason is to control the update/query ratio and concurrent actions. Actions that alter the same information should be performed in a transactional manner. In addition, read actions should not read temporary values of information. Moreover, queries should be let to occur more often that updates even when update requests happen more often than query requests (as it is in dynamic environments). This is because updates are more costly than queries and because querying is the most important function of a registry. Controlling the update/query ratio and concurrent actions can be possible using session techniques as one described in [CBF04]

- *Indexing, update and retrieval mechanisms*: The registry should use a special-purpose data model and techniques for efficient indexing, update and retrieval of published information

- *Soft-state publication (leases) mechanism*: It is of great importance and value if a registry returns up-to-date results. However, in a dynamic and distributed environment, updates occur very often and it is very difficult to maintain a reliable, predictable, and simple distributed registry state. Typical update cases include the following:

- failing or misbehaved services
- change of service URI or service description but service provider does not alter published information at appropriate registries
- change in a registry's authorization policy may result in a case where a published service may no longer be in a position to de-publish itself

All of these cases leave inconsistent or stale registry state behind and can not be captured by a single update model where a service provider publishes, updates or de-publishes information in a registry. So it is difficult for a service registry to detect such situations and to determine when and how they can be resolved. For example, one can envision a strategy in which a registry drops one service's description information if a client finds this service to be unavailable. However, the unavailability may be due to an authorization policy denying access to some but not all clients, or due to problems in a small network segment or simply due to service reboot. The service owner may be offended and claim violation of a service level agreement (SLA), because, in his opinion, there is no reason for dropping its service. Even worse, the service owner might not even notice for quite some time that he has been dropped. To summarize, so-called *hard state* distributed information systems populated from many independent autonomous and heterogeneous distributed sources typically evolve into garbage dumps where valid information is hard to distinguish from trash, decreasing overall utility dramatically.

Elaborate mechanisms can be designed to cope with problems of reliable and consistent state maintenance. Such mechanisms typically face complex and subtle problems. However, one can elegantly avoid much complexity by using a simple *soft state* mechanism for reliable distributed garbage collection: State established at remote location may eventually be discarded unless refreshed by subsequent confirmation notifications. In this manner, component failures and changes are tolerated in the normal mode of operation rather than addressed through a separate recovery procedure. Lack of refresh indicates service failure, shutdown or change.

The responsibility of state maintenance is displaced by moving it from the registry to the publishing services. Registries keep service descriptions as soft state, that is, they are kept for a limited amount of time only. Service descriptions are expired and dropped unless explicitly renewed via periodic publication, henceforth termed *refresh*. Services refresh by essentially saying "I am here" or by updating the information about them or by saying "Description has changed". In the third case, it is the responsibility of the registry to decide when to get the up-to-date description of the service but this case resides on the basis that services are presented via URIs to potential clients.

Clearly, there is a trade off between the resource consumption caused by refreshes and state consistency [Hos02a]. The higher the refresh frequency, the more consistent and up-to-date the state, and the more resources are consumed. High frequency refresh can consume significant network bandwidth, due to pathological client misbehavior, denial-of-service attacks, or sheer popularity. Implementations using high frequency refresh rates can encounter serious latency limitations due to the very expensive nature of secure (and even insecure) network connection setup for publication.

Keep-alive connections should be used to minimize setup time but only partially address the problem. To condition for overload, limit resource consumption and satisfy minimum requirements on content freshness, mechanisms to *throttle* refresh frequency are proposed, adaptively inviting more or less traffic over time. For example, the publication operation could return two timestamps TS1 and TS2, which we call *minimum idle time* and *maximum idle time*. The semantics of the minimum idle time would be that publication was successful but you must at least wait until TS1 before the next refresh. The semantics of the maximum idle time would be that publication was successful but you may be dropped or denied service if you do not refresh before time TS2. Analogously, query operation could return a minimum idle time TS1 as part of the result set indicating that the next query must happen after time TS1.

- *Replication/distribution mechanisms*: Current proposals for Web Services infrastructures/registries focus on centralized approaches [SSDN02] such as UDDI. Service descriptions are stored in a central repository that has to be queried in order to discover or, in a later stage, compose services. Centralized systems introduce single points of failure, hotspots in the network and expose vulnerability to malicious attacks. In addition, due to their limited capacity, they do not scale gracefully to large number of services. This difficulty is severed by the evolving trend to ubiquitous computing in which more and more devices and entities become services and service networks become extremely dynamic due to constantly arriving and departing service providers. The solution to this problem is the organization of a distributed registry, a registry that is separated in many nodes. These nodes either replicate the information they contain or they distribute information according to some semantically meaningful parts or they use a combination of replication and distribution strategies. If a distributed registry organization/structure follows a distributed data strategy, then it should also use a distributed query mechanism to answer user requests. This distributed query mechanism/strategy should route the user request only to nodes that have related-to-user-query data. These distributed organizations and strategies are characteristics of a pure Peer-to-Peer (P2P) infrastructure. Later, P2P decentralized registry approaches will be analyzed.

- *transfer of ownership or custody mechanisms*: In some cases, the owner of a service is changed, so this change must also be updated to the information stored in a registry. In addition, the custody of a service information tuple is attributed to only one registry. The service owner should contact this registry in order to update his service information. When a service registry ceases information or its contract with the service publisher expires, the service publisher can change the custody of his service and hand it to another registry. This mechanism is extremely useful both for centralized and decentralized registries.

- *subscription mechanism*: It is often very practical and profitable to use a subscription mechanism for clients. The usage of this mechanism is twofold:

  – it can be used for informing clients when a result set based on a specific query

has changed

– it can be used for informing clients that a particular entity such as a business or service was created or changed.

In registries, appropriate service taxonomies must be built. The burden of categorizing services in respect with these service taxonomies should be attributed to the service provider although in some cases this task is performed by the registry operator through automated reasoning procedures (based on the structure and content of the service description) or by the matchmaker. The more accurate and rich are these service taxonomies, the more is the added value of the registry.

Now, the WS Matchmaking processing step and the requirements that it imposes are going to be analyzed.

**Web Service Matchmaking** The matchmaking process matches existing Web Service descriptions (stored in a registry) with requester's needs (as expressed in a registry query). In order for this to happen, both the description of the service and requester needs must be written in the same language. The requirements for this language were analyzed in the section 2.1.3. In addition, both advertisers and requesters should be encouraged to be honest with their descriptions. Otherwise, they will pay the price of either not being matched or being matched inappropriately. The matching should not be based on keyword search only [PTB03]. Instead, semantic and structural information about each attribute in the service request and advertisement must be taken into consideration. This is essential, as equality of concept names does not necessarily mean equality of their semantics. Other properties that must hold in a matchmaking system are the following:

- *Open-world descriptions*: This property states that the absence of a characteristic in the description of an advertisement or request should be dealt as something that either will be refined later or is irrelevant for the user [NSDM03]. In other words, if a characteristic is missing from a WS advertisement, then the matchmaking engine should try to obtain it from the service provider instead of regarding as absent. Otherwise, if a characteristic is missing from a WS request, then the matchmaking engine will consider it irrelevant and will not use it in the matchmaking algorithm.

- *Non-symmetric evaluation*: This property states that the evaluation may be different depending on whether the matchmaking engine matches a request with an advertisement or the opposite [NSDM03].

- *User preference consideration*: When the user is not actually exact with the description of what he wants or when the results of his query are huge, the user query must be refined based on user preferences and generic usage patterns [BW03]. User preferences depend on the intensions a user has in a specific domain. For example, when he is at New York, he usually eats in an Italian restaurant. Generic usage patterns can also be considered that state what are the most common user preferences in a specific domain. For example, if someone wants to organize a business dinner, then he must book a French restaurant.

- *Symmetry of information exchange and selection*: The process of finding the right service for a given service consumer is not necessarily a one-way process of having the consumer state their requirements and select a winner from the matching services. Service providers may wish to receive information from the consumer before deciding to make a particular service available to that consumer. The input to the matchmaking process therefore needs to take account of the demands of both service consumers and providers, relating these demands to information provided by both parties – resulting in a symmetric exchange by service consumers and providers of both information and demands [FFH+03].

- *Dynamic service configuration*: Matchmaking should allow a provider to describe its offer as a skeleton or a generating function that can be used to offer different service configurations [FFH+03]. This can be done in the form of a reference to an external system or alternatively by supplying a script that the MME (Matchmaking Engine) can evaluate locally. Thus, the MME can generate the specific service offer dynamically at the time of searching. Input to the process that provides the specific value can contain information from the potential consumer, each service configuration can be tailored to the circumstances of the specific consumer. This is needed for several reasons:

  - It facilitates an up-to-date description of the service where service properties such as the cost, availability or quality of service may be subject to variations. Such variations can for example be due to load, maintenance, etc.
  - It provides a way to specify a range of services without having to enumerate all the options associated with them in the MME as this may overload the MME
  - It provides a way to configure the service and the consumer application according to the needs and properties of both parties. This facilitates personalization of the service
  - It provides a way to integrate existing applications that reside on back-end systems (legacy problem)

The process of matchmaking is quite different from simply finding, given a request a perfectly matching advertisement (or vice versa). In practise, it is very unlikely that such a service/advertisement is available [PKPS02, KBH+03, NSDM03]. Instead, the matchmaker will retrieve a service whose capabilities are *similar* to the capabilities expected by the requester. One of the challenges of matchmaking is to locate those services that the requester would choose/select despite their differences from the request. Furthermore, the matchmaker should be able to characterize the distance between the request and the matches found, so that the requester can make an informed decision on which service to invoke. That is the matchmaker should rank the matches found. The ranking (function) should be based on the following properties [NSDM03]:

- *Syntax independence in ranking potential matches*: A ranking of potential matches is syntax independent if for every pair of advertisements $A_1$ and $A_2$, request $R$, and ontology $T$, when $A_1$ is logically equivalent to $A_2$ with respect to $T$, then $A_1$ and

$A_2$ have the same ranking for $R$, and the same holds also for every pair of logically equivalent requests $R_1$, $R_2$ with respect to every advertisement $A$. In other words, if a specific logic gives meaning to descriptions of requests and advertisements, then descriptions with the same meaning should have the same ranking, independently of their syntax.

- *Syntax independence in ranking partial matches*: A ranking of partial matches is syntax independent if for every pair of advertisements $A_1$ and $A_2$, request $R$ and ontology $T$, when $A_1$ is logically equivalent to $A_2$ with respect to $T$, then $A_1$ and $A_2$ have the same ranking for $R$, and the same holds for every pair of logically equivalent requests $R_1$ and $R_2$ with respect to every advertisement $A$. This property states that when we do not have exact matches, then partial match descriptions with the same meaning should have the same ranking, independently of their syntax.

- *Monotonicity of ranking potential matches over subsumption*: A ranking of potential matches is monotonic over subsumption whenever for every request $R$, for every pairs of advertisements $A_1$ and $A_2$ and ontology $T$, if $A_1$ and $A_2$ are potential matches for $R$ and $T \models A_2 \sqsubseteq A_1$, then $A_2$ should be ranked either the same, or better that $A_1$, and the same should hold for every pair of requests $R_1$ and $R_2$ with respect to an advertisement $A$. This property states that for two advertisements that satisfy a request, the more specific (i.e. constrained one) should have a better ranking as it is considered better and more suitable.

- *Anti-monotonicity of ranking partial matches over implication*: A ranking of partial matches is antimonotonic over implication whenever for every request $R$, for every pair of advertisements $A_1$ and $A_2$, and ontology $T$, if $A_1$ and $A_2$ are partial matches for $R$ and $T \models A_2 \sqsubseteq A_1$, then $A_2$ should be ranked either the same, or worse that $A_1$, and the same holds for every pairs of requests $R_1$ and $R_2$ with respect to an advertisement $A$. This property states that for two advertisements that do not satisfy completely a request, the more specific one is worse and should have a worse ranking.

It must be stressed that the above properties are independent of the particular DL employed, or even the particular *logic* chosen. This is extremely important as it frees service provides, requesters and matchmaking engines from using a specific logic to describe and match WS specifications.

Most matchmaking algorithms rank/categorise Web Services in four categories [PKPS02, KBH+03]:

1. **Exact match** is the highest degree of matching; it results when the two descriptions are equivalent.

2. **Plug-in match** results when the service provided is more general than the service requested, but in practice can be used in place of the ideal system that the requester would like to use. To this extent the result can be "plugged in" place of the correct match. A simple example of a plug-in match is the match between a requested service

that sells books and a service that sells printed materials. Since books are printed materials, chances are that the latter service can be used instead.

3. **Subsumes match** results when the service provided is more specific than the service requested. The service requester may use the provider to achieve its goals, but it likely needs to modify its plan or perform other requests to complete its task.

4. **Fail/Relaxed match** occurs when there is no subsumption relation between advertisement and request.

Ideally, the matchmaker would filter the available services at the following order [Elg03]:

1. *By domain*: Services that do not match the domain of the user will be discarded. Maybe, service advertisement and service request use different ontologies to describe their domain. If this is the case, then ontology mapping and mediation must be performed.

2. *By Role*: Services that do not match the role the user plays in his domain will be discarded.

3. *By goal/capability*: Services that do not match user goals and their set of rules (pre-condition and post-conditions) will be discarded. If "subsumes" relationships exist between user and service goals, then the abovementioned categorization of matches applies. This is the most computation intensive and time-consuming filter.

After the last step/filter, the services/advertisements will be separated into the four abovementioned categories. Except from these three filters that must be applied in this order, other filters can also be applied usually before the capability filter to narrow the search space and to reduce the actual computation time. Actually, the requester should be able to decide which filters to apply and in what order. In selecting any combination of the filters at the search time, there is a trade off between accuracy and speed of the matching process. Other filters that could be applied before the three filters (to discard an initial number of advertisements) are the following:

- *Namespace filter*: It determines if whether or not the requested service and the registered ones have at least one shared namespace (a URL of an ontology file). The intersection of namespaces can be considered shared knowledge between the request and the advertisement. Therefore, only the registered services which have at least one shared namespace with the request go into the next filter. Namespaces of default like rdf, rdfs, xsd, etc. are not considered the intersection. Of course, there is a case where the relation between two nodes of different ontology files does exist, although the distance would be relatively long.

- *Text filter*: This is a pre-checking process for human-readable service explanation parts such as comment and text descriptions. It utilizes the well-known IR (Information Retrieval) technique called TF/IDF (Term Frequency Inverse Document Frequency Method). This filter helps to minimize the risk to miss the services which have any relation to the requested one.

- *Business Model Compatibility filter*: This filter is used to discard advertisements that have a business model that is not compatible with the one the user needs. Maybe the requester does not want to negotiate with the non-compatible services to ensure successful/predictable invocation because this comes with the penalty of extra cost (time and maybe money). Besides, negotiation does not always result in a happy outcome. Compatibility of business models is successfully dealt in [BCT04].

- *QoS filter*: It can be used to discard advertisements that do not satisfy the expected QoS performance the requester wants. Here, the requester should give the range of values or distinct values that the desired QoS properties of advertisements ought to have. In addition, because some QoS properties are dynamic (they change continuously or they are functions of some factors), they must be evaluated at search time by invoking a "getProperty" interface of an advertisement or by invoking locally a script that is associated with the property. However, these "dynamic property value extraction" ways are time consuming. It could be better if the service provider was updating more often the QoS property values of his service (not preferable in highly dynamic environments) or if a service description was annotated with third party metadata provided by experts or users that have used the service. To summarize, this filter needs additional input from the requester (except from the description of the actual request) and it is used in cases where QoS performance is critical e.g. for video services.

Besides using pre-checking filters, some of the filters like the Business Model Compatibility and the QoS filters can be used in further ranking the matches found at the last filter (capability filter), which are separated in four categories. This feature can help users in selecting the best service that suits their needs. The (additional) rank function can be given as extra input by the requester in order to specify which QoS properties and compatibility classes interest him or to determine what percentage each filter will contribute to the ranking function ($20\% * compatibilityRank + 80\% * qosRank$). The rank function can also be predetermined by the matchmaking system. However, this after-checking ranking, although it can have a predetermined ranking function, depends on user needs and only the user knows what he exactly wants. Thus, this after-checking ranking can be omitted from the matchmaking process and be dealt with at the Web Services Selection step, which is clearly user-initiated and user-supported.

The matchmaking process must support both early (design time) and late (runtime) binding to web services. In case of early binding, the matchmaker could be queried at design time to locate the appropriate service and the located service is statically bound with the application being developed. In case of late binding, there is no "hard-wired" function call in the program code but instead a "syntactic and semantic description" of what kind of operation to use, which will be dissolved just in time before execution.

The matchmaking process should support both volatile and persistent queries. In case of a volatile query, the matchmaker immediately returns matching advertisements that are currently present in the repository / registry. On the other hand, the persistent query is a query that will remain valid for a predefined period. Within the validity period of the query, whenever the advertisement that matches the query is added or updated, the matchmaker

will notify the requester. This type of query is exactly what a subscription mechanism of a registry must support. In addition to supporting persistent queries, user queries and their results can be cached. In this way, when the same query is posed again, the matchmaker will respond instantaneously.

The matchmaking process must also support service composition. Service composition is an important issue since it is very likely that the offered services will not satisfy user needs and therefore the combination of basic web services (possibly offered by different companies) into value-added services is needed. In order to automate service composition, the service description must provide declarative specifications of service capabilities. The usual techniques used for web service composition come from Artificial Intelligence (AI) planning like backward or forward chaining [PSK03, AGDR03, CBF04, MWG04]. However, in [SK03] several problems are pointed out when applying current AI planning technology to the service composition problem.

The matchmaking process imposes some additional requirements from the ones described in the Web Service Publication subsection. These requirements are the following:

- *Support of persistent and volatile queries.* Especially persistent queries need special treatment as user queries must be stored and re-assessed every time there is an update in service descriptions that affects the user query. In addition, after the predefined validity period of the volatile query, all the information stored in the registry about this query must be eliminated.

- *Support of caching of user queries.* User queries and their results should be cached in order to answer instantaneously when the same query is posed again. This also means that the results of the cached queries should be changed when an update to a service description that affects these queries happens. Moreover, cached queries and results should be erased some time after their posing.

- *Management of user profiles and general usage patterns.* The data model for user profiles must contain information concerning the user role, identity, age, location, preferences, social situation, etc. User preferences describe user intentions in specific domains. The general usage patterns describe common usage intentions in various domains. Information about the user profile must be updated every time a user makes a request and must be erased after a long period of time where the user has not contacted the registry. On the other hand, general usage patterns are never erased but simply updated every time there is a query in a specific domain of knowledge. Of course, in order to get user information, the user must allow for uncovering of his personal information. In addition, the user should be able to manipulate his profile any time or even erase it.

- The web services data model used in the registry must *hold information about all the possible aspects* of a web service description. In addition, the registry's query language must enable users to *describe in every possible aspect their request.* Moreover, the matchmaking component of the registry must support all the matchmaking require-

ments mentioned above for guarantying a successful discovery process. For instance, matchmaking must be done both at the syntactic and semantic level.

- *Management of general or domain-specific QoS properties for a web service description.* This means that the registry's data model must associate a web service description with general and arbitrary QoS metadata. The update of these metadata can be executed by the particular service provider. However, there must be allowance of third-party annotations of QoS metadata to the web service description.

To support early and late binding, both a programming and web interface are needed for the registry. Applications should see a registry as a Web Service with a particular searching functionality that will help them in finding the specific operation that they need to perform. Other operations of the registry must also be called in the same way in order to enable the automation of user applications and the automated control of registry's mechanisms by the registry (node) operator.

Now, the WS Selection/Brokering processing step and the requirements that it imposes are going to be analyzed.

**Web Service Selection/Brokering** For each operation of a request of a given user, the WS matchmaking step returns a list of candidate WSs that implement the operation. The next step is to select a single WS from each list of candidate WSs and is called "Web Service Selection/Brokering" step. In this step, the selection can be as simple as randomly picking a single service from the best candidate matchmaking category of each result list, or as sophisticated as initiating a complex auction where participating services place bids and negotiate a resolution based on economic models [FNSY96, LNPM98], Quality of Service (QoS) and/or Service Level Agreements (SLAs). However, a logical selection procedure operated in any result list must be composed of the following steps:

1. Rank all advertisements that belong to the "best match" category by specific user criteria concerning QoS properties, business protocol compatibility and user context adaptability. If some advertisements miss some criteria information, then try to get them from the actual services either by invoking a "get property" interface, or by exchanging information with the service in stages as described in [KKRO03], or by negotiating with the service in order to establish a SLA that guarantees that the user and service criteria will be satisfied. If none of the above methods works, discard the incomplete advertisement. When the ranking finishes, select the service with the highest ranking and then invoke it.

2. If there is none "best match" advertisement, then rank all advertisements that belong to the "plug in match" category as above and then invoke the selected service.

3. If there is none "plug in match" advertisement, then either immediately negotiate with the "subsumes match" services or reassess / reconsider your query and execute it again or try to find another registry / matchmaker.

4. If there is none "subsumes match" advertisement, then either you should change your query and repose it or try to find and contact another registry / matchmaker.

The above selection procedure imposes some requirements on the web service description language, the service (provider), the user and the registry:

- The web service description language must be able to describe QoS (dynamic) properties and what is needed (other properties values maybe) to get their values from the service. It also must include the description of the business protocols that a service obeys.

- The service (provider) must describe as separate interface of the service the code to get the values of the dynamic properties (along with input that must be provided in order to correctly assess the property). If this is not done, then it must include in the service description properties whose values must be filled in by the user, as it is described in [KKRO03]. Whatever is the case, a service must be able to negotiate with a user a SLA that will guarantee that the needs of both parties will be satisfied. In addition to these requirements, the service must gather a user's context in some automated way (or by the actual user) in order to adapt its behavior according to user preferences and user machine capabilities [KK04]. The extra cost of achieving this is inconsiderable in respect with the increasing turnover that would result from the increased adaptability and quality of your service. Having satisfied customers means good reputation and this results in more profit.

- The user/requester must be able to negotiate with a service.

- The registry must provide mechanisms for QoS management of web service descriptions. QoS must not be provided only by service providers. Third-party annotations to a web service description must be possible that will assess the QoS of the service in a more "neutral" and non-deceptive way. Service providers tend to lie when revealing the performance of their services in order to get more customers. Third-party annotations can really help the selection procedure as time-consuming stages/steps (to get QoS data from services) can be omitted.

The concept **context** was mentioned previously. According to our perception, a context encompasses all information about the client of a Web Service that may be utilized by the Web Service to adjust execution and output to provide the client with customized and personalized behavior [KK04]. In [MH03], context is defined as "*any information that can be used to characterize the situation of an entity, where an entity can be a person, a place, a physical or a computational object*". In addition, a context model is defined to consist of a set of elements along five axes:

1. *User context*: such as the user role, identity, age, location, preferences, social situation ... etc.

2. *Computing context*: such as network connectivity, nearby resources (printers, displays and workstations, processor speed ... etc.

3. *Time context*: such as time of day, week, month ... etc.

4. *Physical context*: such as weather, temperature ... etc.

5. *Context history*: more importantly, when the user, computing and physical contexts are recorded across a time span, we obtain a *context history*, which is also useful when discovering and composing services.

Context is different from the parameters of a Web Service: First of all, the same context information is interesting for a number of Web Services, whereas parameters are only used by exactly the Web Service they belong to. As a consequence, context can often be evaluated automatically, e.g., by the service platform. This simplifies the development of Web Services as the evolution of such context does not need to be integrated into the Web Services themselves. A further difference is that context information is optional whereas parameters are mandatory. Context information does not need to be passed to a Web Service and if it is, the Web Service does not necessarily need to understand it and process it.

### Related Work and Main Problems

The focus of the work of this proposal is the theoretical foundation and implementation of both the QoS filter of the WS matchmaking sub-process and the WS selection sub-process. More specifically, it is expected that a WS matchmaking process is executed and returns a list of functionally equivalent WS advertisements. Then, the QoS filter is applied to refine this list. Finally, the QoS-based WS selection sub-process is executed to sort the refined list in order to help the requester in selecting the best WS for his purposes. Thus, it is more appropriate to present and analyze the related work in QoS filtering and selection and this is done in chapter 5. For more detailed information regarding WS discovery research approaches, please refer to [Kri04].

## 2.3 Constraint Solving

One of the main contributions of this thesis is the use of two different techniques for matchmaking QoS offers and demands of WSs: Mixed-Integer Programming (MIP) [Sch86] and Constraint Programming (CP) [VHS96]. As QoS specifications are represented by a set of constraints on QoS metrics(i.e. variables), both of these techniques are appropriate for solving the matchmaking problem. Each technique is optimized in different circumstances concerning the nature of the constraints and the values the variables take. For this reason, this section is separated into three subsections: the first two are dedicated in shortly analyzing each technique while the last is dedicated in comparing them by indicating their advantages and shortcomings.

### 2.3.1 Mathematical and Mixed Integer Programming

Mathematical Programming (MP) [GPSH02] is a technique of mathematical optimization. Many real-world problems in such different areas as industrial production, transport, telecommunications, finance, or personnel planning may be cast into the form of a MP problem: a set of decision variables, constraints over these variables and an objective function to be maximized or minimized.

MP problems are usually classified according to the types of the decision variables, constraints, and the objective function.

A well-understood case for which efficient algorithms (Simplex, interior point) are known comprises Linear Programming (LP) problems. In this type of problem all constraints and the objective function are linear expressions of the decision variables, and the variables have continuous domains -i.e., they can take on any, usually non-negative, real values. Luckily, many application problems fit into this category. Problems with hundreds of thousands, or even millions of variables and constraints are routinely solved with commercial MP software like ILOG's CPLEX or Dash Optimization's Xpress-Optimizer.

Researchers and practitioners working on LP quickly found that continuous variables are insufficient to represent decisions of a discrete nature ("yes"/"no" or 1,2,3,...). This observation lead to the development of MIP where constraints and objective function are linear just as in LP and variables may have either discrete or continuous domains. To solve this type of problems, LP techniques are coupled with an enumeration (known as Branch-and-Bound) of the feasible values of the discrete variables. Such enumerative methods may lead to a computational explosion, even for relatively small problem instances, so that it is not always realistic to solve MIP problems (MIPP) to optimality. However, in recent years, continuously increasing computer speed and even more importantly, significant algorithmic improvements (e.g. cutting plane techniques and specialized branching schemes) have made it possible to tackle even larger problems, modeling even more exactly the underlying real-world situations.

Another class of problems that is relatively well-handled are Quadratic Programming (QP) problems: these differ from LPs in that they have quadratic terms in the objective function (the constraints remain linear). The decision variables may be continuous or discrete, in the latter case we speak of Mixed Integer Quadratic Programming (MIQP) problems.

More difficult is the case of non-linear constraints or objective functions, Non-linear Programming (NLP) problems. Frequently heuristic or approximation methods are employed to find good (locally optimal) solutions.

### 2.3.2 Constraint Programming

CP is the study of computational models and systems based on constraints. CP has been widely used in areas such as planning, scheduling and optimization. Currently, it is becoming the standard method for modeling optimization problems and is attracting widespread commercial interest as it is based on a strong theoretical foundation and can solve real-hard

problems.

A constraint is a relation among several variables, each of which ranges over a given domain. So a constraint restricts the values that its variables can take. In CP, a problem can be solved by first stating constraints about the problem area and, consequently, finding a solution that satisfies all the constraints. The last task is carried-out by a *solver*. Thus, CP uses constraints to declaratively state the problem without specifying a computational procedure to enforce them. Constraints in CP are generally expressed by a rich language that includes linear and non-linear constraints or logical combinations of constraints. Actually, the expressiveness of this language depends on the capabilities of the underlying solver.

A problem in CP expressed as a set of constraints is called Constraint Satisfaction Problem (CSP). A CSP is formalized as a set of variables $V$, a set of domain of values $D$ and a set of constraints $C$. Domain of values can be reals, integers, boolean, enumerations or powersets and are associated to one or more variables. Constraints are mathematical expressions over a subset of $V$ restricting the values these variables can take. A *solution* to a CSP is an assignment in which each variable in $V$ takes a value from its corresponding domain in $D$ as long as it does not violate the constraint set $C$. The *solution space* of a CSP is the set of all its solutions. A CSP is *satisfiable* if its solution space is not empty, i.e. it has at least one solution. For example, the CSP $(\{x, y\}, \{[0 \dots 2], [0 \dots 2]\}, \{x < y, x > 0\})$ is satisfiable as its solution space contains the sole solution $\{x \mapsto 1, y \mapsto 2\}$.

Instead of getting one or all solutions of a CSP, a user goal could be to find those solutions that satisfy an objective function. This type of CSP is called Constraint Satisfaction Optimization Problem (CSOP) as it actually defines an optimization problem. In relation to CSPs, the solution space of a CSOP is called *minimum space*. The objective function is just a function over a subset of $V$.

In CP, constraints are used actively to deduce infeasible values and delete them from the domains of variables. This mechanism is called *constraint propagation*. It represents the core of CP systems. Each constraint computes impossible values for its variables and informs other constraints. This process continues as long as new deductions are made. Constraint propagation is associated with *tree search techniques* in order to find solutions or prove optimality. Each node and each decision will induce constraint propagation automatically. Many specific and efficient algorithms are used in this propagation, but do not need to be known by the end-user. This allows persons who are familiar with problem modeling to quickly use such techniques for optimization or generation of solutions.

Apart from the basic CP research domain, various sub-domains have been created for deriving and implementing different solving techniques for different types of problems that can be modeled with CP. On such domain is the domain of *Dynamic Constraint Satisfaction* (DSC) [VJ05] that was created to deal with dynamic application domains whose problems change at design or even at run time of the solving engine. In DSC, a *dynamic constraint satisfaction problem* (DSCP) is a sequence of CSPs, each one produced from some changes in the definition of the previous one. A change may affect any part of problem definition: variables, domains of variables, constraints, constraint scopes (addition or removal of a variable in constraint's definition), or constraint definition (e.g change in the mathematical expression). This domain will be analyzed in detail in chapter 6 as its techniques and

algorithms will be used as one of the basic implementing techniques of our developed and implemented QoS-based WS matchmaking algorithms.

Another interesting sub-domain of CP is the one dealing with soft-constraints and is called *preference reasoning* [Ros05]. This domain has been created to deal with application domains where the problems are usually over-constrained or they can be described with preferences rather than hard constraints/statements or the solutions to these problems have different desirability. However, as preferences can be expressed in a quantitative or qualitative way according to the problem we want to model, they can be conditional or unconditional, and can co-exist with constraints, many formalisms have been proposed in order to be specify them efficiently and reason with them effectively. For example, *soft constraints* [BMR97] are most suited for reasoning about constraints and quantitative preferences, while *CP-nets* [BBHP04] are most suited for representing qualitative and possibly conditional preferences. Soft constraints and one of their representation formalisms (semiring-based CSPs [BMR97]) will be further investigated in chapter 6 as they can be used, in our opinion, to represent optimization problems used to solve over-constrained QoS demands.

### 2.3.3 Comparison

CP has the advantages that: **a)** it can solve linear and non-linear constraints; **b)** constraints can be combined conjunctively and disjunctively while the negation of constraints is also allowed; **c)** it is quite efficient in solving constraints involving integer variables with small domains; **d)** the representation of a CSP is very close to the original problem since the CSP variables correspond directly to domain entities and the constraints can be expressed naturally without translating them into linear inequalities. Unfortunately, CP also presents the following disadvantages: **a)** the use of real-valued variables in constraints is under ongoing research and no commercial product really supports this feature yet; **b)** when integer variables with medium-sized and large domains are involved in constraints, constraint propagation can be slow, especially when the constraints involving them are not very restrictive; **c)** CP solvers cannot in general deal with optimization problems in an efficient way, as their focus is on satisfaction problems.

On the other hand, MIP has the advantages that: **a)** MIP variables can be integer or real-valued; **b)** the size of the domain of the variables does not play a very important role in solving; **c)** feasibility (consistency) checking is very efficient as it is inherited from LP which uses very mature technologies. However, MIP also has the following disadvantages: **a)** non-linear problems are hard to solve; **b)** only conjunctions of constraints can be naturally expressed; **c)** constraints must be expressed with (linear) inequalities.

Based on the disadvantages of both CP and MIP, there is now a trend in combining these techniques for solving real-life hard problems. However, the subject of this subsection and of course this thesis is not to propose a technique for combining CP with MP. On the contrary, the objective is to evaluate and compare the behavior of consistency (feasibility) checking and matchmaking algorithms implemented using CP and MIP so as to reveal: a) which is the best from the two techniques and in which case, and b) by choosing a specific

technique according to a specific case, which from our developed matchmaking algorithms is better according to the WS requester preferences.

One last but important remark to be made is that solving non-linear constraints with any technology is a NP-complete problem. Thus, in the worst case any QoS-based WS matchmaking algorithm will need non-polynomial time to solve models (i.e. QoS descriptions) that contain non-linear constraints.

## 2.4 Conclusions

This chapter introduced the main concepts and their roles in the WS Description and Discovery processes, after providing the exact definitions of both of these processes. Moreover, it analyzed the main requirements that these two processes impose in order to be achieved. Then it explicated and theoretically compared the two main techniques that are used by our thesis in order to perform QoS-based WS matchmaking and selection: MIP and CP. The scope of this chapter was to equip the interested reader with the appropriate knowledge in order to be able to follow up with the remaining chapters of this dissertation and to better understand our main contributions, the effort taken to achieve them and their outcome and benefits.

# Chapter 3

# Role of QoS for Web Services

As Quality of Service (QoS) is the dominant concept used throughout this dissertation, this chapter provides a comprehensive analysis of the conceptualization of QoS for WSs. It also reveals the main domain-independent QoS attributes that are appropriate for the description of WSs. In addition, some domain-dependent QoS attributes are also shortly analyzed taken from the application domain of Traffic Monitoring. Finally, the role that QoS can play in the management of WSs is elaborated. For this reason, this chapter is organized into three sections. Section 3.1 defines QoS for Web Services. The next section 3.2 exposes the main domain-independent and some domain-dependent QoS attributes. The last section 3.3 analyzes the benefits of using QoS in WSs. Similarly to the scope of the previous chapter 2, the purpose of this chapter is to equip the interested reader with the appropriate knowledge in order to be able to a) follow up with the remaining chapters of this dissertation, b) understand the benefits of our thesis and c) fix the terminology used throughout this dissertation in order to make it self-contained.

## 3.1 QoS Definition

The international quality standard ISO 8402 (part of the ISO 9000 [Lie94]) describes quality as "*the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs*". According to [DSGF03], what defines quality is vague, and different views exist in different studies and from different perspectives. The following three views are most common ones:

1. **Quality as Functionality**. According to this view, quality is considered as the amount of functionality that a service can offer to its users. For example, if SP1 allows you to rent a car besides booking flights and hotel rooms, and if this functionality is not provided by SP2 or SP3, then SP1 is offering a better quality than SP2 and SP3.

2. **Quality as Conformance**. According to this view, quality means meeting specifications. For example, if SP1 has specified that his WS will be available 0.9999 of the time and this was true, then SP1 is considered as offering good quality of service.

3. **Quality as Reputation**. According to this view, quality depends on users' experience and expectation from a WS and its value is built collectively over the time of the service's existence from users' feedback [MS02]. For example, if WSs have consistently provided specific functionality with specific performance levels at all time of their operation, then they provide good quality of service.

These different views of quality require QoS to be monitored and measured differently. Quality as functionality characterizes the design of a service and can only be measured by comparing the service against other services offering similar functionalities. Quality as conformance, on the other hand, can be monitored for each service individually, and usually requires the user's experience of the service in order to measure the "promise" against the "delivery". Finally, reputation can be regarded as a reference to a service's consistency over time in offering both functionality and conformance qualities, and can therefore be measured through the other two types of quality over time.

While it is possible to establish all three types of quality for a service in an SOC environment, it is perhaps most interesting and relevant to understand how quality as conformance may be monitored and measured. The reasons for this are the following: Firstly, the first type of quality is already established in the SOC environment with the usage of the WSDL and UDDI standards, although nobody reassures that the functionality exposed by a service is the same with the one advertised by the service's provider. Secondly, the reputation of a service is just an indicator of the overall QoS of the service over time that depends on user expectations and other imponderable factors (advertisement, financial and political interests, etc.). So reputation can only be used as a secondary QoS property that can be measured and computed by monitoring and other authorities.

Thus, by the above definition and view, we consider QoS of a Web Service (WS) as **a set of non-functional characteristics/attributes that may impact the quality of the service offered by the WS**. If a WS is advertised to have certain values (or range of values) in these QoS attributes, then we say that the WS **conforms** to provide a certain **QoS level**.

## 3.2 QoS Attributes Exposure

Quality of Service research has been an active research area for several domains. The term "quality of service" has been used for expressing non-functional requirements for different areas such as network research community [Cru95, GGPS96, SF01] and in real time issues [CSZ92]. There is some research effort in defining QoS in distributed systems. The focus is primarily on how to express the QoS for a system, and how these requirements are propagated to the resource manager to fulfil the QoS requirements [TVB00]. [HHR+97] presents

a layered model for representing QoS for telecommunication applications. It presents service quality function, QoS schema mapping and price-QoS trade-off. [FK98] present a QoS specification language. They advocate its use for designing distributed object system, in conjunction with the functional design. [SCD+97] categorizes the QoS from different viewpoints: application, system and resource. It specifies QoS in terms of metrics and policy.

All the above research efforts categorize and define QoS from their perspective with some overlap between them. There is not great consensus about a set of QoS important to distributed systems. There is even less research done on the QoS for SOA. However, some QoS attributes have been found that are common in most of the above research efforts and are directly applicable to WSs. These are general, cross-domain, QoS attributes, i.e. they are independent of the domain the WS belongs to. In the first – section 3.2.1 – of the next two subsections, we are going to analyze this set of common domain-dependent QoS attributes.

Of course, there will be and some domain-dependent QoS attributes that will depend on the domain(s) of knowledge a WS applies to. For example, QoS attributes regarding information quality will be important for a data warehouse WS [LSKW02]. Another more specific example is the following: a service such as a car rental service will involve attributes belonging to multiple domains, such as the travel and retail domains. Attributes such as price belong to both domains, but an attribute such as "flexibility of reservation changes" has a specific meaning in the travel domain. However, determining the attributes that apply to a particular domain is nontrivial and needs to be decided by the community of users and providers as they settle upon ways to distinguish and evaluate different offerings. For this reason, we are not going to enumerate and explain the meaning of any possible QoS domain-dependent attribute that characterizes any possible domain. On the contrary, we are only going to enumerate the most important domain-dependent QoS attributes that are required to be specified for the Traffic Monitoring application domain in subsection 3.2.2. This is done not only for providing a specific example of domain-dependent QoS attributes but also because most of the paradigms and testing frameworks of the following chapters of this dissertation will use this application domain and its QoS attributes.

### 3.2.1 Domain-Independent QoS Attributes

There are many aspects of QoS important to Web services. We are starting to organize them into QoS categories. Each category needs to have a set of quantifiable parameters or measurements. Further research is needed in this area. For illustration purposes, the categories are defined here. They are described briefly, followed by a question as an example to show what type of questions the particular QoS can address. To facilitate the description, the categories are grouped into different types, i.e. QoS related to runtime, transaction support, configuration management and cost and security. This categorization is based on the research works of [FK98, AN02, SA03, Ran03, LJL+03].

### Runtime Related QoS

*Scalability* – The capability of increasing the computing capacity of service provider's computer system and system's ability to process more operations or transactions in a given period. It is related to performance. Web services should be scalable in terms of the number of operations or transactions supported.
*Q*: Will the system scale up to handle X transactions per second? This is closely related to throughput and performance.

*Capacity* – Limit of concurrent requests which should be provided with guaranteed performance.
*Q*: How many concurrent connections does the service support?

*Performance* – a measure of the speed in completing a service request. It is measured by:

- *Response time* – the guaranteed max (or average or min) time required to complete a service request (related to capacity).
- *Latency* – Time taken between the service request arrives and the request is being serviced.
  *Q*: What is the average delay on servicing a request?
- *Throughput* – The number of completed service requests over a time period. Throughput is related to latency/capacity.
- *Execution time* – the time taken by a WS to process its sequence of activities.
- *Transaction time* – the time that passes while the WS is completing one complete transaction. This time may depend on the definition of WS transaction.

In general, high quality web services should provide higher throughput, faster response time, lower latency, lower execution time, and faster transaction time.

*Reliability* – The ability of a service to perform its required functions under stated conditions for a specified period of time. It is the overall measure of a WS to maintain its service quality. Reliability is also related to the assured and ordered delivery of messages being transmitted and received by service requestors and providers. It can be measured by:

- *Mean time between failure* (MTBF)
- *Mean Time to Failure* (MTF)
- *Mean Time To Transition* (MTTT)
- *Availability* – It is the probability the system is up and related to reliability. It can be measured as [Gun00]:

$$A = \frac{\langle upTime \rangle}{\langle totalTime \rangle} = \frac{\langle upTime \rangle}{(\langle upTime \rangle + \langle downTime \rangle)}$$

Where: $\langle upTime \rangle$ is the total time the system has been up during the measurement period, $\langle downTime \rangle$ is the total time the system has been down during the measurement period, and $\langle totalTime \rangle$ is the total measurement time (sum of $\langle upTime \rangle$ and $\langle downTime \rangle$).
$Q$: What is the chance that the service is available when I invoke it?

- *Continuous availability* – It assesses the probability with which a client can access a service an infinite number of times during a particular time period. The service is expected not to fail and to retain all state information during this time period. We could for example require that a particular client can use a service for a 60 minute period without failure with a probability of 0.999. Continuous availability is different from availability in that it requires subsequent use of a service to succeed but only for a limited time period.

- *Failure masking* – It is used to describe what kind of failures a server may expose to its clients. A client must be able to detect and handle any kind of exposed failure. The types of failure than can be exposed are the following: *failure*, *omission*, *response*, *value*, *state*, *timing*, *late*, and *early*. The QoS specification for a particular WS will list the subset of failures exposed by that service. The above types of failures are categorized – as it can be seen by Fig. 3.1. If a service exposes *omission* failures, clients must be prepared to handle a situation where the service simply omits to respond to requests. If a service exposes *response* failures, it might respond with a faulty return value or an incorrect state transition. Finally, if the service exposes *timing* failures, it may respond in an ultimately manner. Timing failures have two subtypes: *late* and *early* timing errors.

- *Operation semantics* – they describe how requests are handled in the case of failure. We can specify that issued requests are executed *exactlyOnce*, *atLeastOnce*, and *atMostOnce*.

- *Server failure* – it describes the way in which a service can fail. That is, whether it will *halt* indefinitely, restart in a well defined *initialState*, or restart *rolledBack* to a previous checkpoint.

- *Data policy* – When a service fails and then restarts, the client needs to know if data returned by the service is still valid. To specify this, we need to associate *data policy* with entities such as return values or arguments.

*Robustness/ Flexibility* – It is the degree to which a service can function correctly in the presence of invalid, incomplete or conflicting inputs.
$Q$: Will the service still work if incomplete parameters are provided to the service request invocation?

*Exception handling* – Since it is not possible for the service designer to specify all the possible outcomes and alternatives (especially with various special cases and unanticipated possibilities), exceptions can be expected. Exception handling is how the service handles these exceptions. It can be in a brutal or a graceful way.

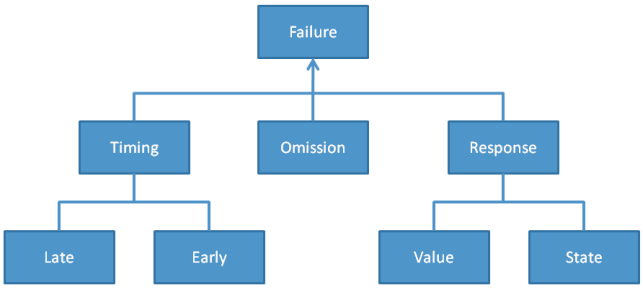



Figure 3.1: Types of failure of a server

*Q*: How will the service still work correctly if I give less number of parameters than it requires?

*Accuracy* – Defines the error rate produced by the service. The number of errors that the service generates over a time interval should be minimized.
*Q*: How many errors does the service produce over a period of time?

### Transaction Support Related QoS

*Integrity* – Transactions can be grouped into a unit in order to guarantee the integrity of the data operated on by these transactions. The unit can either be successful where all transactions in the unit "commit" or all "roll back" to their original state in case of a transaction failure. This is described by the ACID properties: *Atomicity* (executes entirely or not at all), *consistency* (maintains the integrity of the data), *isolation* (individual transactions run as if no other transactions are present) and *durability* (the results are persistent)).

A *two-phase commit* capability is the mechanism to guarantee the ACID properties for distributed transactions running over tightly coupled systems as if they were a single transaction. It is more difficult in the Web services environment, as the transactions may involve more than one business partner with the possibility of transactions spanning over long time (hours or days) - *Long Running Transactions* (LRT). The transaction integrity is still described by ACID properties, although it is a much harder to achieve in this case. It may require different mechanisms.

### Configuration Management and Cost Related QoS

*Regulatory* – It is a measure of how well the service is aligned with regulations.
$Q$: How aligned is the service with appropriate regulations?

*Supported Standard* – A measure of whether the service complies with standards (e.g. industry specific standards). This can affect the portability of the service and interoperability of the service with others. One example is ISO 8583, which is a standard for creating and reading financial transaction messages including Point of Sale (POS) transactions [Lie94].
$Q$: How much does the service adhere to applicable standards? Or what standards does the service comply?

*Stability/change cycle* – A measure of the frequency of change related to the service in terms of its interface and/or implementation.
$Q$: How stable is the service, how often it changes (interface and implementation)?

*Guaranteed messaging requirements* – does it ensure the order and persistence of the messages?

*Cost* – It is a measure of the cost involved in requesting the service.
$Q$: What is the cost based on (per request or per volume of data)?

*Completeness* – A measure of the difference between the specified set of features and the implemented set of features.
$Q$: How many of the specified features are currently available?

*Reputation* – The reputation $q_{rep}(s)$ of a service $S$ is a measure of its trustworthiness. It mainly depends on end user's experiences on using the service $S$. Different end users may have different opinions on the same service. The value of the reputation is defined as the average ranking given to the service by end users, i.e. $q_{rep}(s) = \frac{\sum_{i=1}^{n} R_i}{n}$, where $R_i$ is the end user's ranking on a service's reputation and $n$ is is the number of times the service has been graded. Usually, end users are given a range to rank Web Services. For example, in Amazon.com, the range is [0, 5]. Alternatively, a ranking of an end user to a service can be a vector of values of some QoS attributes. From this vector, a QoS value can be obtained which is then averaged over all user rankings to get the overall reputation of the service.

### Security Related QoS

WSs should be provided with the required security. With the increase in the use of WSs which are delivered over the public Internet, there is a growing concern about security. The WS provider may apply different approaches and levels of providing security policy depending on the service requestor. Security for WSs means providing *authentication*, *authorization*, *confidentiality*, *traceability/auditability*, *data encryption*, and *non-repudiation*. Each of these aspects is described below: *Authentication* – How does the service authenticate principals (users or other services) who can access service and data?
*Authorization* – How does the service authorize principals so that only these can access the protected services?
*Confidentiality* – How does the service treat the data, so that only authorized principals can access or modify the data?

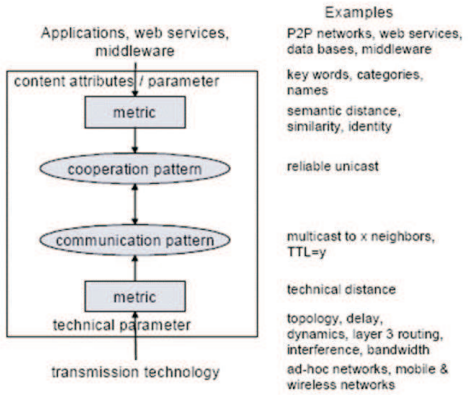



Figure 3.2: Mapping of application level QoS parameters to network level QoS parameters

*Accountability* – Can the supplier be hold accountable for their services?
*Traceability and Auditability* – Is it possible to trace the history of a service when a request was serviced?
*Data encryption* – How does the service encrypt data?
*Non-Repudiation* – A principal cannot deny requesting a service or data after the fact.
*Q*: How does the service provider ensure these security requirements?

### Network Related QoS Attributes

Today, research activities in applications, Web services, and communication networks are running in many aspects widely independent from each other. In most cases, researchers of applications and Web service technologies assume that existing communication infrastructures provide reliable communication. Furthermore, researchers in middleware, Web services, and applications are not very considerate of the resources provided by the underlying networks. On the other hand, research activities in certain communication architectures and protocols are performed with less attention to requirements of actual applications. Therefore, most applications cannot actively consume the Quality of Service (QoS) that may be supported in the communication networks, and on the other hand common network technologies do not support application-dependent requirements.

To achieve desired QoS for web services, the QoS mechanisms operating at the web service application level must operate together with the QoS mechanisms operating in the transport network (e.g., RSVP, DiffServ, MPLS, etc.) which are rather independent of the application. In particular, application level QoS parameters should be mapped appropriately to corresponding network level QoS parameters [TGN+03, LJL+03]. Figure 3.2 (copied from [TGN+03]) gives examples for parameters on different layers when mapping

applications and services onto certain transmission technologies or when pushing performance parameters from transmission technologies up to applications, respectively. The communication and cooperation between different layers allows an efficient utilization of the underlying network resources as well as a better support of application-dependent requirements. It must be highlighted that the abovementioned mapping of parameters is not the concern of this report. More information on how to achieve this mapping can be found in [TGN+03, LJL+03].

Basic network level QoS parameters include network delay, delay variation, and packet loss, and they are described as follows:

*Network delay* – it is the average length of time a packet traverses in a network. The network delay can be handled by a good network design that minimizes the number of hops encountered and by the advent of faster switching devices like Layer 3 switches and tag switching system such as MPLS systems and ATM switches.

*Delay variation* – it is the variation in the inter-packet arrival time (leading to gaps, known as jitter, between packets) as introduced by the variable transmission delay over the network. Removing jitter requires collecting packets in buffers and holding them long enough to allow the slowest packets to arrive in time to be played in correct sequence. Jitter buffers may cause additional delay, which is used to remove the packet delay variation as each packet transits the network.

*Packet loss* – The Internet does not guarantee delivery of packets. Packets will be dropped under peak loads and during periods of congestion. Approaches used to compensate for packet loss include replay of the last packet, and transmission of redundant information. Out of order packets may need to be re-ordered at the receiver.

In addition, network management mechanisms may also be involved in controlling and managing QoS for web services.

### 3.2.2 Domain-Dependent QoS Attributes

The examples used for validating theoretically and evaluating experimentally the main contributions of this thesis rely on the application domain of *Traffic Monitoring* [CCP07]. In this domain, WSs provide up-to-date information about local and inter-state traffic to business and retail customers across the US. The QoS of these WSs is characterized by two different sets of attributes: domain-dependent and domain-independent.

Domain-independent QoS attributes refer mostly to technical aspects of service provisioning irrespective of the application domain used. For the sake of simplicity, we consider seven (7) domain independent QoS attributes: *reliability*, *data encryption*, *refresh time*, *execution time*, *availability*, *reputation*, and *price*. *Data encryption* and *price* are static QoS attributes whose values are known at design time so it's not required to be measured by a QoS metric. *Data encryption* refers to the algorithms adopted for protecting data from malicious accesses while *price* is the subscription or monetary cost for using the WS. On the other hand, *reputation* is a QoS attribute that is not advertised by WS providers but is computed by WS registries by taking the average of WS executors feedback. However, WS requesters may give constraints for this QoS attribute. The latter is the main reason for

taking it into account. *Reliability* is measured by the *MTBF* metric that measures for how long a WS can function in an error-free manner. *Availability* is measured with the help of the $upTime$ and $downTime$ metrics based on the formula previously analyzed that gives a raw metric of *availability* (which has a timeseries value type). Of course, metrics computing the average, min and max values of the raw metric can also be used. *Execution time* can be measured by a raw metric but – similarly to *availability* – it can also be measured by high-level statistical metrics.

Domain-dependent QoS attributes strongly rely on the application domain of the WS advertized/queried. For the Traffic Monitoring domain, we also consider eight (8) QoS attributes that characterize mainly the quality and other domain-dependent characteristics of the data processed and produced. These QoS attributes are: *covered area*, *routes set*, *detail level*, *accuracy*, *completeness*, *validity*, *timeliness*, and *coverage*. The first three QoS attributes are static. The *covered area* QoS attribute characterizes the extensiveness of the area over which the WS is able to provide traffic information. *Routes set* is the set of route types that a WS supports. For instance, a WS may provide information only on national highways while other WSs may also consider inter-state or local routes. Similarly, the *detail level* of traffic information provided by a WS may also vary. A WS may provide information on accidents and traffic jams while other WSs may also provide information about closed routes, detours and predictions about future conditions of local traffic.

The other four QoS attributes are highly dynamic and there are many different QoS metrics for measuring them. *Accuracy* is the measure or degree of agreement between a data value or set of values and a source assumed to be correct. It is also defined as a qualitative assessment of freedom from error, with a high assessment corresponding to a small error. *Accuracy* can be measured using one of the following three error metrics: *Mean Absolute Percent Error* (MAPE), *Signed Percent Error* (SPE) and *Root Mean Squared Error* (RMSE). These three metrics are calculated based on the following three formulas:

$$MAPE(\%) = \left(\frac{1}{n}\right) \times \left(\sum_{i=1}^{n} \left|\frac{X_i - X_{reference}}{X_{reference}}\right|\right)$$

$$SPE(\%) = \left(\frac{1}{n}\right) \times \left(\sum_{i=1}^{n} \frac{X_i - X_{reference}}{X_{reference}}\right)$$

$$RMSE = \sqrt{\left(\frac{1}{n}\right) \times \left(\sum_{i=1}^{n} (X_i - X_{reference})^2\right)}$$

where $X_i$ is the observed data value, $X_{reference}$ is the reference (i.e. ground truth) value, and $n$ is the total number of observed data values. $RMSE$ can also be expressed as a percentage. These different error formulations are all valid measures of accuracy but may reveal slightly different interpretations. The *mean absolute percent error* (MAPE) and *signed error* (SPE) are expressed as percentages; thus, these formulations may be used to compare the relative

accuracy of different attributes (e.g., traffic volume count and speed measurement accuracy). Because the signed error does not use absolute error values (as MAPE does), the signed error formulation may reveal whether there is a consistent bias in measurements. The root mean squared error (RMSE) is an error formulation that is commonly available in many statistical software applications. The correct source of data is typically referred to as ground truth, reference, or baseline measurements. Ground truth data can be collected in several different ways for each traffic data element. In many cases, ground truth data are collected from specialized equipment and reduced in a rigorous manner that minimizes error.

The QoS attribute of *completeness* represents the degree to which data values are present in the attributes (e.g., volume and speed are attributes of traffic) that require them. *Completeness* is typically described in terms of percentages or number of data values. *Completeness* can refer to both the temporal and spatial aspect of data quality, in the sense that completeness measures how much data is available compared to how much data should be available. It can be measured using a percentage metric computed from the following formula:

$$PercentComplete\,(\%) = \frac{n_{availableValues}}{n_{totalExpected}} \times 100$$

where $n_{availableValues}$ is the number of records or rows with available values present and $n_{totalExpected}$ is the total number of records or rows expected. The number of data records expected is a function of the application. The percent complete statistic is defined to include all "values present". In this respect, completeness is defined as including both valid and invalid data values, as long as both types of data values are present in the version of data being evaluated. However, if a particular data process removes invalid data values from a database instead of flagging them as invalid and permanently storing them, then these purged invalid data values would not be included in the completeness statistic because they are not "present".

*Validity* is the degree to which data values satisfy acceptance requirements of the validation criteria or fall within the respective domain of acceptable values. *Data validity* can be expressed in numerous ways. One common way is to indicate the percentage of data values that either pass or fail data validity checks. It can be measured by a metric computing the percentage of data passing validity criteria with the following formula:

$$PercentValid\,(\%) = \frac{n_{valid}}{n_{total}} \times 100$$

where $n_{valid}$ is the number of records or rows with values meeting validity criteria and $n_{total}$ is the total number of records or rows subjected to validity criteria. Validity criteria (also referred to as business rules or validation checks) are defined in many data management applications and can range from a single simple rule to several levels of complex rules. A simple rule might specify that traffic volume counts cannot exceed a maximum value associated with road capacity (such as 2,600 vehicles per hour per lane) or that traffic speeds can not exceed a reasonable threshold (such as 100 mph). Other validity criteria for traffic data could include the following:

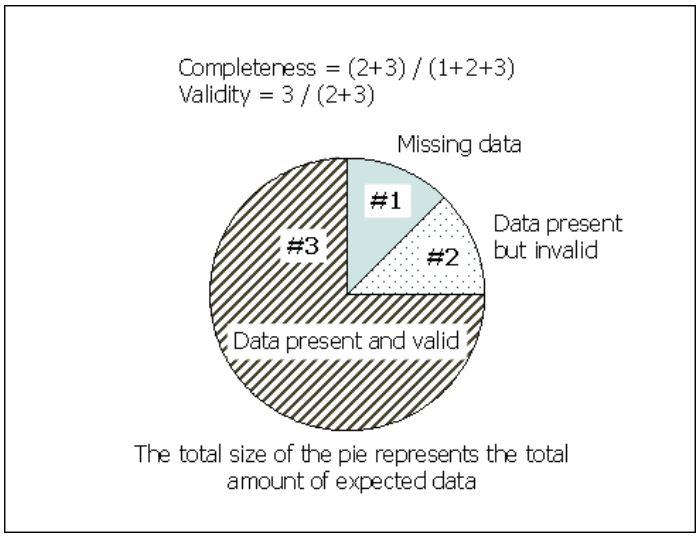



Figure 3.3: Illustration of *Completeness* and *Validity* metrics

- Compare multiple data elements (i.e., volume, occupancy, and speed) to check for inconsistency among traffic data values;
- Compare traffic data to historical averages or trends from previous days, months, or years;
- Compare traffic data to similar nearby locations (upstream or downstream) to check for continuity of traffic flow;
- Compare traffic data in consecutive time periods to identify rapid fluctuations.

Validity criteria are often based on "expert opinion" and are generally viewed as "rules of thumb" although some validity criteria may be based on established theory (e.g., road capacity) or scientific fact (e.g., cannot record a zero volume and non-zero speed). The specific validity criteria will likely vary from place to place, as each traffic data collector or manager brings experience with certain roadway locations, traffic data collection equipment, or collection hardware and software.

The difference between *completeness* and *validity* is best represented with the example of Figure 3.3. In this figure, the pie represents the total amount of data that is expected to be collected (based on data collection plan or data polling rates). The percent complete statistic includes both valid (slice #3) and invalid (slice #2) values, divided by the total expected number of values (entire pie). The percent valid is the valid values (slice #3) divided by the total values checked (slice #2 and #3).

The last QoS attribute, *timeliness*, represents the degree to which data values or a set of values are provided at the time required or specified. *Timeliness* can be expressed in absolute or relative terms. It can be measured by two different metrics: *Percentage of data received within acceptable time limits* (PTD) and *average delay for late data* (ADLD). The

PTD metric is expressed by the following formula:

$$PercentTimelyData(\%) = \frac{n_{on-time}}{n_{total}} \times 100$$

where $n_{on-time}$ is the number of data messages or packets received within acceptable time limits and $n_{total}$ is the total number of data messages or packets received. The ADLD metric is expressed by the following formula:

$$ADLD = \left(\frac{1}{n_{late}}\right) \times \left(\sum (t_{late}) - (t_{exprected})\right)$$

where $n_{late}$ is the number of data messages or packets received outside acceptable time limits, $t_{late}$ is the actual arrival time of a late data message or packet, and $t_{exprected}$ is the expected arrival time of a late data message or packet.

Besides the last four QoS attributes, another QoS attribute that can be modeled is the one of *coverage* which represents the degree to which data values in a sample accurately represent the whole of what is to be measured. As with other measures, coverage can be expressed in absolute or relative units. *Coverage* can be expressed as the percent of roadways (or the transportation system) represented by traffic data. For illustration purposes, in our examples *coverage* is computed from the following formula:

$$cov = \frac{coveredarea_{req} \cap coveredarea_{adv}}{coveredarea_{req}}$$

where $coveredarea_{req}$ is the set of values determining the area the WS requester wants to cover while $coveredarea_{adv}$ is the set of values determining the area the WS covers.

In certain cases, composite metrics can be used by combining the metrics used in measuring the last five (dynamic) QoS attributes. The computed metrics would be useful in relative comparisons or rankings. A composite data quality score that may be useful for performance monitoring applications combines the *completeness*, *coverage* and *validity* attributes to create a "composite system completeness" metric:

$$CompositeSystemCompleteness(\%) = PercentCoverage \times PercentComplete \times PercentValid$$

It must be stated that as WSs may produce data or information products, their QoS should include quality attributes characterizing the quality of the data or the information they produce. However, this type of QoS attributes is domain-dependent as it depends on the application domain of the WS. A small subset of the possible list of these QoS attributes was actually included in our modeling of the Traffic Monitoring application domain and included the attributes of *accuracy*, *completeness*, *validity*, and *timeliness*. As we show in our modeled

application domain, these attributes had concrete QoS metrics so it was straightforward to express WS capabilities or user requirements with constraints on the QoS metrics of these attributes. However, as data and information quality constitutes a vast area of research, we have to further investigate if all the QoS attributes of this research area can be correctly modeled with our proposed QoS model and language. Thus, we can safely state that until now some of the QoS attributes of data or information quality can be represented in our QoS model and language and we will further investigate in the near future if our QoS model is enough for capturing any QoS attribute of this research area.

## 3.3 Benefits of QoS Usage for Web Services

According to [CSM+04], several researchers have identified Web Processes (WPs) as the computing model that enables a standard method of building WSs applications and processes to connect and exchange information over the Web. For organizations, being able to characterize WPs based on QoS has four distinct advantages. **First**, it allows organizations to translate their vision into their business processes more efficiently, since a WP can be designed according to QoS metrics. For e-commerce processes it is important to know the QoS an application will exhibit before making the service available to its customers. **Second**, it allows for the selection and execution of WPs based on their QoS, to better fulfil customer expectations. As WP systems carry out more complex and mission-critical applications, QoS analysis serves to ensure that each application meets user requirements. **Third**, it makes possible the monitoring of WPs based on QoS. WPs must be rigorously and constantly monitored throughout their life cycles to assure compliance both with initial QoS requirements and targeted objectives. QoS monitoring allows adaptation strategies to be triggered when undesired metrics are identified or when threshold values are reached. **Fourth**, it allows for the evaluation of alternative strategies when adaptation becomes necessary. The unpredictable nature of the surrounding environment has an important impact on the strategies, methodologies, and structure of business processes. Thus, in order to complete a WP according to initial QoS requirements, it is necessary to expect to adapt, re-plan, and reschedule a WP in response to unexpected progress, delays, or technical conditions. When adaptation is necessary, a set of potential alternatives is generated, with the objective of changing a WP as its QoS continues to meet initial requirements. For each alternative, prior to actually carrying out the adaptation in a running WP, it is necessary to estimate its impact on the WP's QoS. For example, when a WP becomes unavailable due to the malfunction of its components, it is indispensable to evaluate the adaptive strategies that can be applied to correct the process. **It is essential that the services rendered follow customer specifications to meet their expectations and ensure satisfaction. Customer expectations and satisfaction can be translated into the quality of service rendered.** Organizations have realized that quality of service management is an important factor in their operations.

As WPs are composed or single WSs, all the above advantages of QoS management of WPs also apply to Web Services. That is Web Services can be designed and implemented

according to QoS metrics (properties). They can also be discovered and selected based on their QoS capabilities. In addition, they can be monitored in order to reassure the promised QoS levels to the customers. Moreover, monitoring of QoS for Web Services can trigger adaptation strategies when undesired metrics are identified, threshold values are reached, network or software or hardware errors happen. Adaptation strategies include: repair of malfunctioning components, usage of back-up machines, increase of CPU usage for a customer's WS running operation, using alternative network paths, preference of undesired QoS levels for a WS operation instead of improving them, interrupting clients' operations and starting an operation of a "better" client, etc. The last strategy reveals one very important advantage: A service provider can offer different QoS levels to different customers according to QoS customer needs, the price the customer is willing to pay, his previous usage of the service and even the social status of the customer. This means that the service provider can actually control the profit for the usage of his service and he can eventually maximize it. Research work on this last issue can be found in [Moo01].

Now let us look more closely at the advantages of QoS Management in basic and other SOA activities/functions. Let us start with *Web Service Discovery*. The Discovery of Web Services process tries to find a WS providing specific capabilities, interfaces and other characteristics, requested by a WS client. Its result is a set of Web Services satisfying any or most of the client requirements. The WS Discovery process provides better results if it is supported by a complete and accurate WS Description. That is a description which is formal, complete and accurate and provides any possible aspect that differentiates one WS from the other besides functionality. One aspect that currently is not supported by the Web Services standards and not fully supported by other (complementary or not) research efforts is the QoS description of a WS. The WS Discovery process may return many functionally-equivalent results. In this way, it will harden the user task of selecting the most appropriate WS. If QoS description was incorporated into WS Description, then the returned result list would be filtered out to include only the results providing better QoS performance. Furthermore, the filtered results list can be ordered based on a user function specifying which QoS characteristics are more important to him. In this fashion, the discovery step would not only provide a more constrained result list but would also suggest the best result for the user, thus helping him in the task of *WS Selection*.

After the step of WS Selection, even if the WS client has found the appropriate WS for him, he is not confident that the WS described QoS levels will actually be delivered to him during WS execution. For this reason, the WS client and provider enter a multi-step negotiation phase, where they try to agree on a trusted third-party entity monitoring QoS levels delivered, on the penalties that will be imposed when one of the two main parties does not keep up with its promises, on which situation each penalty applies, and on the validity period of the promises. The result of this negotiation phase is a *contract* or a *Service Level Agreement* (SLA) document that will give confidence and trust to the entities providing and consuming the service and will lead and guide the process of WS Execution. If the WS client and provider do not come into agreement, the negotiation is stopped and the WS client must try to contact the second WS from the returned list of the WS Discovery phase.

A composed WS consists of a combination of WSs and internal operations/processes.

The process model underlying a composite WS identifies the functionalities required by the service to be composed (i.e. the tasks of the composite WS) and their interactions (e.g. control-flow, data-flow, and transactional dependencies). Component services that are able to provide the required functionalities are then associated to the individual tasks of the composite WS and invoked during each execution of the composite service. However, the number of services providing a given functionality may be large and constantly changing and some of these services will not always be available due to network problems (e.g. partitions), software evolution and repair, and hardware problems. Consequently, approaches where the development of composite services requires the identification at design-time of the exact service to be composed (after its discovery) are not appropriate.

The runtime selection of component services during the execution of a composite WS has been put forward as an approach to address this issue [CIjJ+00, BSND02]. The idea is that component services are selected by the composite service execution engine based on a set of criteria. However, previous approaches in this area have not identified a set of criteria (other than price and application-specific criteria) and adopt a local selection strategy, i.e. they assign a component service to an individual task one at a time. A new approach [ZBD+03] overcomes the above deficiencies and can handle global user constraints and preferences (like overall duration of the composite service execution or a budget constraint). The salient features of this approach are a multi-dimensional WS quality model (for WS description) and a global planning method/approach. In this method, quality constraints and preferences are assigned to composite services rather than to individual tasks within a composite service. Service selection is then formulated as an optimization problem and a linear programming method is used to compute optimal service execution plans for composite services. However, it must be indicated that this new approach works only for some of the QoS metrics. That is global constraints can only be set for QoS metrics that can be derived from the corresponding QoS metrics of the individual tasks and Web Services. These QoS metrics include execution time, cost, reliability and reputation [CSM+04]. To conclude, we can see that by having a QoS model for WS Description and QoS-based WS Discovery, the task of WS Composition can be automated and individual service selection takes place as late as possible i.e. at run time.

## 3.4 Conclusions

This chapter provided a comprehensive analysis of the conceptualization of QoS for WSs by analyzing what is QoS, which views exist on QoS and the reason we chose one of these views. It also revealed the main domain-independent QoS attributes that are appropriate for the description of any WS. Moreover, it also shortly analyzed some domain-dependent QoS attributes taken from the application domain of *Traffic Monitoring*, which is the application domain we chose for modeling QoS specifications and providing them as input to the experimental evaluation of our proposed discovery algorithms. Finally, the role that QoS can play in the management of WSs was elaborated. The scope of this chapter was the same as with the previous one: to make the interested reader capable of with the remaining chapters

of this dissertation, to understand the benefits of our thesis and to fix the terminology used throughout this dissertation in order to make it self-contained.

# Chapter 4

# Requirements for QoS-based Web Service Description and Discovery

After reviewing related work in QoS-based WS description and discovery, we identified and came up with a set of requirements [KP07b] that must be satisfied in order for these two processes to be successful, accurate and complete. The goal of this chapter is to analyze these requirements. The outcome of this chapter, i.e. the requirements, will be used in the following two chapters – 5 and 6 – in order to compare the related work against these requirements and to design and implement our contributions according to them. So this chapter's importance deserves and is enough for making it a stand-alone chapter of this dissertation.

This chapter is organized as follows. The first section 4.1, is dedicated to the QoS requirements of the WS description process. On the other hand, the last two – sections 4.2 and 4.3 – are dedicated to the QoS requirements of the two sub-processes (matchmaking and selection) of the WS discovery process.

## 4.1 Requirements for QoS-based Web Service Description

In order to support QoS-based WS discovery, the QoS offer or demand of a WS must be described. Therefore, we need a QoS (description) model that will be incorporated to the overall WS description model. After studying and processing several research efforts, we have come up with some general and specific requirements [KP07b] that this QoS model must satisfy. An analysis of these requirements is given below, from the most general to the most specific. For each requirement, the possible implications it raises regarding the functionalities or obligations of the components of the WS Discovery Architecture are discussed.

### 4.1.1 Extensible and Formal QoS Model

In the presence of multiple Web Services with overlapping or identical functionality, service requesters need objective QoS criteria to distinguish one service from another. It is supported in [LNZ04] that it is not practical to come up with a standard QoS model that can be used for all web services in all domains. This is because QoS is a broad concept that can encompass a number of context-dependent non-functional properties such as privacy, reputation and usability. Moreover, when evaluating QoS of web services, domain specific criteria must be taken into consideration. For example, in the domain of phone service provisioning, the penalty rate for early termination of a contract and compensation for non-service, offered in the service level agreement are important QoS criteria in that domain. Therefore, an extensible QoS model must be proposed that includes both the generic and domain specific criteria. In addition, new domain specific criteria should be added and used to evaluate the QoS of web services without changing the underlying computation (i.e. matchmaking and ranking) model. Last but not least, the semantics of QoS attributes/concepts must be described in order to ensure that both WS provider and consumer talk about the same QoS attribute/concept. Sometimes, generic QoS attributes with the same name like "application availability" may have different meanings to the parties that describe them (network level of the hosting system, application that implements the service) or they may be computed differently. Other times, domain-dependent QoS attributes may have the same name but obviously different meaning. So it is important to describe QoS attributes/concepts not only syntactically but also semantically in order to have a better discovery (matchmaking) process with high precision and recall.

The solution to the above problems is the use of ontologies. Ontologies provide a formal, syntactic and semantic description model of concepts, properties and relationships between concepts. They give meaning to concepts like QoS attributes, QoS offers, domains, services so that they are human-understandable and machine-interpretable while they provide the means for interoperability [TEPP02, MS02, TGN+03, ZCL04]. Moreover, they are extensible as new concepts, properties or relationships can be added to an ontology. In addition, SW techniques can be used for reasoning about concepts or for resolving or mapping between ontologies. These techniques can lead to syntactic and semantic matching of ontological concepts and enforcement of class and property constraints (e.g. type checking, cardinality constraints, etc.). Therefore, by providing semantic description of concepts and by supporting reasoning mechanisms, ontologies cater for better discovery process with high precision and recall. Last but not least, ontologies can help specialized agents/brokers in performing very complex reasoning tasks like WS discovery or mediation or negotiation.

#### Possible Implications – Obligations

Current WS registries are based on the UDDI model [BCC+04], which incorporates a description model that is poor and mainly syntactic allowing only taxonomies of concepts to be described. Due to this description model, querying facilities of UDDI registries allow only keyword search i.e. purely syntactic search. Therefore, WS registries must resort to

ontological representation formalisms like OWL [BDH+03] to describe Web Services. In fact, there has been developed a Web Service description language based on an OWL ontology, which is called OWL-S [ea03] and can be used for the description of Web Services in registries. Fortunately, in order for a registry to remain compatible with its previous UDDI technology, a mapping mechanism has been discovered [PKPS02] that maps OWL-S descriptions to UDDI ones. Unfortunately, OWL-S comes with many deficiencies [SRvS03]. Nevertheless, it is a first step towards semantic description and reasoning of Web Services. However, OWL-S must be extended to include (or reference other ontologies that include) a QoS-based model as it only now contains one attribute used for establishing a rating for a service.

### 4.1.2 Syntactical Separation

QoS specifications should be syntactically separate from other parts of service specifications, such as interface definitions (WSDL). This separation allows us to specify different QoS properties for different implementations of the same interface [FK98]. Moreover, while functional constraints rarely change during runtime, QoS constraints can be changed during runtime. So the separation of service offerings from WSDL descriptions enables that, if needed, service offerings can be deactivated, reactivated, created, or deleted dynamically without any modification of the underlying WSDL file [TMPE03]. Last but not least, an offer could be referenced from multiple WSDL files and thus be reused for different services [TGN+03].

#### Possible Implications – Obligations

Syntactical separation of QoS specifications from interface definitions can cause problems. Entities that use/describe/store QoS service offerings should be responsible for QoS service offering dynamic management and their mapping to the corresponding interface definitions. To be more specific about syntactical separation and following the paradigm of OWL-S registries, separate OWL ontologies must be developed for QoS specification that would be referenced by the ServiceProfile element (it specifies what the service does) of an OWL-S WS description [ZCL04].

### 4.1.3 Both Client and Service QoS Specification

It should be possible to specify both the QoS properties that clients require and the QoS properties that services provide [FK98]. Moreover, these two aspects should be specified separately so that a client-server relationship has two QoS specifications: a specification that captures the client's requirements and a specification that captures the service's provisioning. This separation allows us to specify the QoS characteristics of a component, the QoS properties that it provides and requires, without specifying the interconnection of components. The separation is essential if we want to specify the QoS characteristics of components

that are reused in many different contexts. Talking in terms of ontologies, this separation can be established by having two separate concepts (*ProviderQoS* and *RequesterQoS*) to be subclasses of the more general concept of *QoSDescription*; the one describing the QoS offer of a service provider and the other describing the QoS demand of a requester.

### 4.1.4 Refinement of QoS Specifications

As previously described, syntactical separation provides reusability. But except from reusability, another form of extensibility is equally important. QoS specifications should not only be reused but also be refined. This means that we can create a new WS QoS offering by referencing an older one and by adding constraints like refinement of an older QoS restriction or creation of a new one [FK98, TPP03]. In addition, templates of QoS offerings can be created, usually for every domain of knowledge [FK98, ZCL04]. These templates can be extended by Web Services applied to a domain that want to define their QoS offerings. This is another sign for the ontology usage in QoS offering description as this kind of extensibility is tightly related to concept inheritance.

### 4.1.5 Fine-Grained QoS Specification

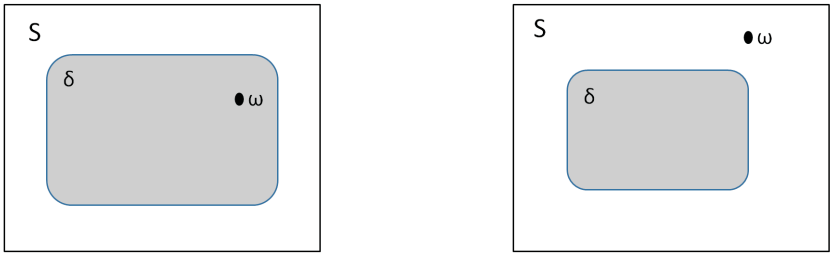
It should be possible to specify QoS properties at a fine-grained level [FK98]. As an example, performance characteristics are commonly specified for individual operations. As another example, the data policy dimension (described in QoS Attributes Exposure section) is applicable to arguments and return values of operations. A QoS model must allow QoS specifications for interfaces, operations, attributes, operation parameters, and operation results. Generally speaking, any service object can have QoS attributes (e.g., elements defined in WSFL [Ley01]) [KL03].

### 4.1.6 Symmetric Model of QoS Specification

In [CMDTT05], it is endorsed that a symmetric model of QoS specification must be adopted. The reason for this lies beneath.

#### Asymmetric Models

Let $S$ be a multidimensional space whose dimensions are given by domains of QoS parameters. Traditionally, a demand ($\delta$) has been viewed as a subspace in $S$, whereas an offer ($\omega$) has been viewed as a point in $S$. Thus, checking the conformance amounts to checking whether the point (the offer) belongs to the subspace (the demand) or not. See Figures 4.1(a) and 4.1(b), respectively. This checking can be computed easily by evaluating $\omega$ in $\delta$. As an example, if a web service owns the offer $\omega = \{MTTF = 120\}$, then it is conformant to the demand $\delta_1 = \{MTTF \geq 100\}$ because $120 \geq 100$, but not to the demand $\delta_2 = \{MTTF > 120\}$ because $MTTF_{\delta_2} > MTTF_{\omega}$.




(a) Conformant (b) Non-conformant

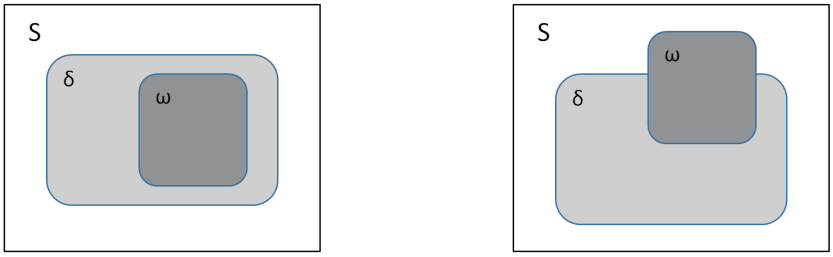
Figure 4.1: Conformance in asymmetric models

This interpretation of conformance results in a model which is asymmetric with regard to the expressiveness of QoS specifications. This semantics makes very difficult to specify offers when it is needed something else than a point, as an example to specify some uncertainty or a space. As most of programming languages are able to check if a point is inside a space, whereas checking if a space includes another space is a hard question, most of platforms have adopted an asymmetric specification model. As well, these approaches with an asymmetric model usually own a limited expressiveness because conditions are restricted to simple expressions involving single parameters, so complex expressions are not allowed.

### Symmetric Models

Alternatively, an offer can be also considered as a sub-space, just as demands, so that it represents the ranges of quality-of-service values that the corresponding web service guarantees to supply. In this way, an offer ($\omega$) is conformant to a demand ($\delta$) whenever the offer's sub-space is inside the demand's sub-space (see Figure 4.2(a)), otherwise the offer is not conformant (see Figure 4.2(b)). As an example, if a web service owns the offer $\omega = \{MTTF \geq 120\}$, then it is conformant to the demand $\delta_1 = \{MTTF \geq 100\}$, but not to the demand $\delta_2 = \{MTTF > 120\}$ because the offer's instance value $\{MTTF = 120\}$ is out of the demand's space.

This interpretation of conformance results in a symmetric model because QoS in demands and offers can be specified in the same way. This semantics makes the offer guarantee the complete range, not only a concrete value, i.e., we can not make any assumption on a concrete value, because it is equally possible any value in the subspace, and there is no control to get a concrete value. As well, symmetric approaches usually achieve a greater deal of expressiveness to specify quality-of-service, since there is usually no restriction on the number of involved parameters or type of operators, so that non-linear or more complex expressions are allowed.




(a) Conformant (b) Non-Conformant

Figure 4.2: Conformance in symmetric models

### 4.1.7 QoS Attributes Model

For each domain, the attributes in that domain are important inputs to the overall QoS of a service. Some attributes are common across domains and some are specific to domains. Each QoS attribute is measured with the help of a QoS metric. Each attribute/metric has the following aspects (the first five are described in [MS02] while the rest are ours):

- The value set for the metric (and its allowed value range). For instance, a metric such as failure rate measuring availability can be expressed as a percentage. The speed of service function execution could be instead a simple bounded integer.
- The domains that this attribute belongs to. For instance, is it a cross-domain attribute or an attribute specific for a domain? And within each domain, some attributes will be of greater importance than others-this can depend on some standard definition of attributes for a domain.
- The weight of the metric relative to its domain and user preferences. This weight would determine the impact of this attribute on the final decision regarding a provider. This weight can also help in calculating the rank of a QoS offering.
- The characteristic of the function from metric values to overall QoS values. For instance, some metrics such as the one measuring availability are monotonic, at least in typical business scenarios. That is, the more the metric of availability increases the better. Generally all agents will consider availability's metric in the QoS value calculation and have a preference for higher values. Of course, if availability were to increase in conjunction with reductions in the values for other positively-monotonic metrics of attributes, the overall QoS might not improve. For instance, for a trip service, if the price of trips were to decrease while the promptness (on arrival and departure times) metric were to become worse, then this decrease in price might not help the overall QoS of the trip service. The characteristics of metrics can be quite rich and need to be further categorized.

- The temporal characteristic of the metric value. Metrics may have decaying values where the decay function can vary from exponential to a step function. For instance, a metric measuring an attribute such as accuracy in the travel domain might be acceptable to allow a range of minutes up to a certain point. That is, the trip that is scheduled for 3:00 PM does not cause major harm if the actual departure time is 3:20 PM. However, a departure of 4:00 PM might become unacceptable for a user that depends on an on time arrival with a gap of 30 minutes in order to catch a connecting flight. Other metrics might have values that decay more progressively rather than in a step-wise fashion.

- QoS attributes must have unique names in their domain. This must also be true for cross-domain attributes.

- QoS attributes could be grouped into QoS Groups. QoS Groups can contain other QoS attributes and QoS Groups. For example, QoS attributes *response time*, *latency*, *throughput*, *execution* and *transaction time* can belong to the *performance* QoS Group. As another example, the *performance* QoS Group could be contained in the more general QoS Group of *Runtime Related QoS*.

- There must be a description (mathematical or otherwise formal) of how a QoS metric's value of a complex WS can be derived from the corresponding QoS metrics' values of the individual WSs that constitute the complex one. For example, the execution time metric $T_C$ of a complex WS $C$, which is defined as a sequence of two WSs $A$ and $B$, can be computed as the sum $T_A + T_B$ of the execution time metrics of the two individual WSs. This description is essential for the automated estimation of the values of QoS metrics for a complex WS that is composed of other WSs and individual operations. So this description is needed for automating the QoS analysis process, a prerequisite for a successful QoS-based WS discovery. In addition, it helps automating the WS composition process and delaying individual WS selection as late as possible (i.e., at runtime).

In the QoS bibliography, QoS attributes are also referred to as QoS metrics. The above aspects of QoS metric description are not enough. Additional semantics must be established. In [TEPP02], it is described that there is a need for several ontologies that would be used in the formal representation of QoS and other constraints. These ontologies include: ontology of QoS metrics, ontology of measurement units, ontology of currency units, ontology of measured properties and ontology of measurement methods. The last ontology is for WS management purposes and will not be further discussed. In the QoS metric ontology, a metric has a name and a short human-readable textual description. In addition, the measured property (e.g. time, quantity of information, information transmission rate, etc.) that the metric quantifies is referenced. Moreover, metrics can be composite or simple (resource). Examples of composite metrics are: *maximum response time* of a service, *average availability* of a service, or *minimum throughput* of a service. Examples of resource metrics are: *system uptime*, *service outage period*, or *number of service invocations*. Composite metrics can be computed by other QoS metrics with the help of a formula. In one ontological definition

of a QoS metric, several such formulae can be specified. Each formula can contain mean, median, sum, minimum, maximum, and various other arithmetic operators, or time series constructors. It should also be accompanied by appropriate unit conversion rules. [KL03] backs that each formula must reference a Schedule or a Trigger. A schedule defines the time intervals during which the formulas are executed to compute the metrics, while a trigger defines a point in time to which the execution of monitoring activity can be tied. In [KL03], Measurement Directives specify how individual (resource) metrics are calculated from the source but this feature is more like a management statement so it can be omitted. Measurement Directives are surely related to the omitted "measurement methods" ontology.

According to [TEPP02], the ontological definition of a QoS metric can also specify invariant relationships with other QoS metrics, particularly those that measure the same property. For example, "average response time" should always be less than or equal to "maximum response time". While such information is probably redundant, it is very important for quick and easy discovery of conflicts. In general, when several QoS metrics are specified in the same service offering, checking various dependencies and relationships given in the ontology helps to avoid various conflicts.

The ontology of QoS metrics should be accompanied by an appropriate ontology of measurement units. Such an ontology should define three types of units: base units like "second" and "byte", multiples of base units like "millisecond" and "megabyte", and derived units like "transactionsPerSecond" and "bytesPerSecond". Synonyms, including abbreviations, should be specified for all three types. Ontological definitions of all measurement units should contain information about what kind of property is measured. When a measurement unit is used for a particular QoS metric, one can check whether the definition of the measurement unit and the definition of the QoS metric refer to the same measured property. The three types of measurement units differ in what additional information should be specified in their ontological definition.

Special types of measurement units are monetary units representing currencies, and units derived from them. Examples are "CanadianDollars" and "CanadianDollarsPerHour". Usually countries give a special name to 1/100 (or sometimes 1/1000) part of their currency (e.g., "CanadianCents") and this information should also be represented in the ontology. Also, some multilingual countries have several synonyms (from different languages) for their currency. A very important special characteristic of monetary units is dynamism of relationships between various currencies. Instead of specifying fixed formula for conversion between different currencies, an ontology could reference one (or maybe more) currency conversion Web Services that can be consulted for up-to-date conversions. Due to this feature, the ontology of monetary units might be separated from the ontology of other measurement units.

[TEPP02] does not describe what are the properties and concepts of an ontology of measured properties. This kind of ontology must be established because not only it is a connecting point of the other metric ontologies but also because the meaning of the measured properties must be provided to the entities participating in the WS discovery process. So further research must be done in order to reveal what are the basic concepts and properties of an ontology of measured properties.

According to [TEPP02], an important issue that must be raised is the development of independent, third-party Web Services for ontological translations between different QoS metrics, measurement units, and currencies. For general usability, these conversion Web Services should be able to communicate using WSDL, not only DAML-S. For example, a Web Service can provide QoS guarantees using "average throughput [invocations/s]" for a particular "number of invocations [invocations]", while one of its consumers can reason about QoS using "average response time [ms]". To understand each other, they have to perform an ontological translation. However, in some cases Web Services will not be able perform ontological translations themselves. For example, they might not understand the language (e.g., OWL) used to represent different ontologies. In such cases, the Web Service or its consumer should consult a specialized external ontology translation Web Service for help during the process of service offering negotiation.

**Possible Implications – Obligations**

From the above discussion of the metrics model and other additional models attached to it, two new requirements come up in relation to the basic interacting entities of the discovery architecture. First of all, while the evolution of the metric ontologies will reach a final point, their metadata will not be exactly the same for providers and clients involved in the same domains. This is quite rational as humans may have different perceptions or terminologies of the same concepts. Especially, metrics like "availability" can have different names or different definitions. So it is important for a WS Client not content with the metrics exposed by the service provider to define new metrics or change some of them and to negotiate with the WS Provider in order to get his new (compatible) offers. The WS Client must have in mind that the above act of guarantee and more satisfaction can cost him more money as more constraints lead to more costs for the service provider. On the other hand, the WS Provider must be able to understand the changes made by the client and as well as to modify his QoS offers. However, he must be careful. Exposure of new metrics can reveal important information of his system. In addition, adoption of new metrics can increase the management efforts of his system and this can even have a serious impact on the QoS offered to his customers.

The other new and important requirement is the standardization of these metrics metadata. WS Providers and Consumers must sit on the same table along with domain ontology modellers and define what are the most important metrics for a domain. As we saw previously, some already generic metrics exist for Web Services, although the way they can be measured differs depending on their definition or the way someone conceptualizes them. Even if someone is not pleased with the metrics defined in his domain, he can define new metrics not by creating them from scratch but by altering or extending the definition of existing ones already provided.

### 4.1.8 Great expressiveness and correct constraint definition

A QoS offer or demand (i.e. specification) must be comprised of QoS constraints. Each QoS constraint consists of a name, an operator, and a value [FK98]. The name is typically the name of a QoS metric, although it can also be the name of a metric *aspect*. We will see in a while what an aspect is. The permissible operators and values depend on the QoS metric type. A metric type specifies a domain of values. These values can be used in constraints for that dimension. The domain may be ordered. For example, a numeric domain comes with a built-in ordering ("<") that corresponds to the usual ordering on numbers. Set and enumeration domains do not come with a built-in ordering; for those types of domains we have to describe a user-ordering of the domain elements. The domain ordering determines which operators can be used in constraints for that domain. For example, we can not use inequality operators ("<", ">", "≥", "≤") in conjunction with an unordered domain.

Aspects are complex statistical characterizations of QoS constraints like: *percentile*, *mean*, *variance*, and *frequency*. They are used for characterization of measured values over some time period. For example, the percentile aspect could be used to define an upper or lower value for a percentage of the measurements or occurrences that have been observed. Aspects can be proved to be very useful in cases where we want to guarantee that the measurements or occurrences of a QoS metric present some special characteristics and we do not want to produce a new complex metric from the basic QoS metric for each of these characteristics. However, they must be used carefully especially in cases where many aspects are created for one metric.

QoS constraints are usually connected by the "and" logical operator, although they can also be connected by other logical operators, into expressions. A QoS offer or demand should contain one complete expression or just one constraint.

QoS constraints should be joined into *Constraint Groups* (CG) or *Constraint Group Templates* (parameterized CGs) in order to be reused by many QoS specifications [TMPE03]. Other reusability constructs can also be created even for expressions.

#### Possible Implications – Obligations

The consistency of every QoS demand or offer needs to be checked either before or after its submission to the WS registry. This means that the there must be at least one value assignment for every QoS metric such that the whole expression, the offer or demand contains, is satisfiable [CMDTT05]. In addition, the references to external ontological concepts need to be checked for validity and consistency, i.e. if they exist in the reference ontology, if they have the right attributes and the right values for these attributes. Finally, syntactical errors need to be checked. Thus all the components/entities of the WS Discovery Architecture need to check the QoS specification they submit or receive for any kind of inconsistencies that may occur in this specification. After all, common human mistakes (logical or syntactical) always happen and the (interoperability) problems due to ontological evolution have not been successfully solved yet.

### 4.1.9 Allow Classes of Service Specifications

*Class of Service* means the discrete variation of the complete service and QoS provided by one WS [TPP03]. Classes of service make sense to be discussed at the level of Web Services and not at the level of constraints or guarantees (e.g. response time) that are part of the overall service and QoS. Classes of service can differ in usage privileges, service priorities, response times guaranteed to consumers, verbosity of response information, etc. The concept of classes of service also supports different capabilities, rights, and needs of potential consumers of the Web Service, including power and type of devices they execute on. Further, different classes of service may imply different utilization of the underlying hardware and software resources and, consequently, have different prices. Additionally, different classes of service can be used for different payment models, like pay-per-use or subscription-based.

The issues of QoS and balancing of limited underlying resources are particularly motivating for having multiple classes of service for Web Services. If the underlying resources were unlimited, all consumers would always get the highest possible QoS. Unfortunately, this is not the case, so it is suitable to provide different QoS to different classes of consumer. Providers of Web Services want to achieve maximal monetary gain with optimal utilization of resources. Providing different classes of service and their balancing helps in achieving this goal because of the flexibility to accommodate several classes of consumer. On the other hand, consumers of such Web Services can better select service and QoS they need and are willing to pay for, while minimizing their price/performance ratio.

Providing classes of service is not the only possible way to customize constraints and management statements that a WS offers to its consumers. There are various alternatives, including custom-made Service Level Agreements (SLAs), user profiles, parameterization, and separate ports. However, the practice of telecommunication service provisioning shows that classes of service have relatively low overhead and complexity of management.

From the above analysis, it is claimed that a WS must provide multiple classes of service (i.e. constraints and management statements). So it is important to have a description of class of service of a WS included in the overall WS description model. However, while the above discussion is interesting, we limit ourselves to QoS constraints only and we do not take into consideration other non-functional constraints and management statements. Moreover, we consider that these other non-functional constraints and management statements are not very important to users as QoS is, so they do not play an important role in the discovery process. Therefore, we sustain that it is of great interest to service providers that they provide multiple QoS offerings. In this way, they service a wider range of consumers and they broaden their market segment. In addition, WS consumers can have a wider solution space and (with the help of efficient QoS-based discovery mechanisms) they can find the best solution (combination of the functional capability of WS and its QoS offering) that minimizes their price/performance ratio. Thus, the QoS specification must support multiple QoS offerings of the same WS of a service provider, an idea also adopted by other research efforts [TGN+03, ZCL04].

### 4.1.10 Other Useful Information

A WS QoS offering must provide references to protocols needed for service management and QoS monitoring as well as entries of third parties that one side would be willing to trust. This information is important as it will be used in the contract negotiation phase, after the WS client has selected the particular service and its offering. Moreover, a QoS offering must be related to the *price* element in order to relate the specified QoS level to the cost of service usage per invocation. Finally, a QoS offering must have an *expires* attribute denoting the point in time until which the offer will be valid [TGN+03]. Of course, other non-functional properties and management statements can be added to a QoS offering/specification, upgrading the QoS offer to a service class but this is out of the scope of this report.

## 4.2 Requirements for QoS-based Web Service Matchmaking

The QoS-based WS matchmaking process starts when the functional WS matchmaking process ends. To put it in another way, the WS matchmaking process is a composed process consisting of two sequential processes: functional and QoS-based. The functional process filters WS advertisements based on the functional restrictions of the requester. The QoS-based process further filters the results of the functional process based on the QoS restrictions of the requester. So the first requirement is that the output of the functional WS matchmaking process should be the input of the QoS-based WS matchmaking process.

One important thing that must be discussed is what happens if both the goal/capability and the QoS filters are applied and if their order matters. First of all, we believe that the order does not play a significant role here as these filters are independent of each other. The order we described above is just preferable as it is better if we know first which advertisements satisfy the request functionally and then to see if one of the QoS offers of each qualified advertisement suits the needs of the QoS demand of the request. In addition, this order is mathematically appropriate as Web Services will provide many QoS offers so the probability that one offer suits the needs of the requester is increased. Moreover, there is no guarantee that the service provider is frank about his QoS offers. Therefore, an application of the QoS filter before the goal filter will not prune as many results as those that will be pruned by the application of the goal filter.

After the application of both filters, we may come into a situation where an advertisement that had a failed match regarding its capabilities to have one of the best QoS offers. We believe that there must be some groundwork between and after the application of these filters in order to remove advertisements or offers of advertisements that do not suit the needs of the requester. That's why the participation of the requester is needed either by providing the appropriate input to the matchmaking engine or by interacting with the matchmaking engine. Therefore, the requester must determine which categories of the results of each filter are preferable or satisfying. In addition to the before-mentioned groundwork, the presentation and total characterization (in respect with the degree of match) of the returned results is an inevitable choice for the matchmaking engine because the requester must know

what are the best results and for each result in what category of match of each filter it was put. Notice that after matchmaking (that used both filters) we may come into a situation where there is one result in the best combined category. This will mean that further ranking is not needed and the selection step of the discovery process will be omitted. In practice, the selection step should not be omitted as the best discovered service could have been recovering from a failure, could not be available, or could not exist (the provider or registry did not erase its description).

QoS offers and demands that are matched during the QoS-based WS matchmaking process should be specified by the same language. The requirements for this language were expressed in section 4.1. In addition, both advertisers and requesters should be encouraged to be honest with their descriptions. Otherwise, they will pay the price of either not being matched or being matched inappropriately. The matching should not be based on keyword search only. Instead, semantic and structural information about each attribute in the service request and advertisement must be taken into consideration.

More specifically, semantic matching of different QoS metric descriptions must be provided. The basic part of QoS-based WS description is the one where WS providers advertise their capabilities and WS requesters provide their requirements as a set of constraints on specific QoS metrics. So in order for the discovery process to be accurate, these sets of QoS metrics must be as similar as they can get. For this reason, semantic QoS metric matching algorithms must be developed and used before the QoS-based WS matchmaking is performed in order to produce QoS-based WS specifications containing (almost) the same set of QoS metrics. Moreover, knowledge about how QoS metrics are derived must also be taken into account as it can help the metric matching process by enriching the processed WS specifications. The main result will be QoS-based WS specifications aligned with each other. Then, these aligned specifications will be used in the matchmaking and selection algorithms increasing their precision and recall.

Most of the requirements that were analyzed in chapter 2 and more specifically in section 2.2.4 for QoS registries and WS matchmaking also stand for QoS-based WS matchmaking. For this reason, in the sequel only the requirements that are specific to the QoS-based WS matchmaking process will be analyzed.

In a perfect world, the QoS-based matchmaking system would return to the requester those QoS offers that perfectly match his request. In practice, this fact is highly unlikely to happen. One of the challenges of matchmaking is to locate those services that the requester would choose/select despite their differences from the request. Furthermore, the matchmaker should be able to characterize the distance between the request and the matches found, so that the requester can make an informed decision about which service to invoke. So QoS offers should be grouped according to their rank (distance from request). Suppose that there is a measure or function that tells us if one solution (comprised by a specific value for every user metric) is better than another one (see next subsection). To the best of our knowledge, at least the following categories of results should be provided by the QoS-based matchmaking system with decreasing order of significance:

- *Super matches*: This category contains those QoS offers that have at least one better solution and all other equivalent to those solutions of the QoS request.

- *Exact matches*: This category contains those QoS offers that have a subset of the solutions expressed by the QoS request.
- *Partial matches*: This category contains those QoS offers that have at least one worse solution than those of the QoS request.
- *Fail matches*: This category contains those QoS offers that have only worse solutions with respect to the solution of the QoS request.

Of course, a MME could provide more fine-grained categories than the proposed ones. However, their ranking must be based on the following properties:

- QoS offers expressing the same set of solutions for the same QoS metrics should be given the same rank.
- If a QoS offer $O_1$ has at least one better and no worse solution than the solutions expressed by the QoS offer $O_2$, then $O_1$ should have a better rank than $O_2$.

Suppose that after the matchmaking process has been executed, there are only partial and fail matches. This is an indication of an over-constrained QoS request. Fail matches are of no use and should be discarded. However, partial matches are promising. The reason is that if one (not very important) QoS constraint of the QoS request is relaxed or deleted, then it is possible that some partial QoS offers are promoted to higher categories. This process is called constraint relaxation and must be incorporated in the QoS-based MME when the aforementioned case occurs in order to provide more value-added results to the WS requester. More details about constraint relaxation can be found in [BMR97].

### 4.2.1 Possible Implications – Obligations

It is clear from the discussion above that the WS requester must be actively involved in the matchmaking process as he only knows what he exactly needs (what filters to call and in what order, how many results to be returned, how query relaxation will be carried out, etc.). However, if a WS registry provides alternative complete (i.e. customizable) ways of invoking the matchmaking operation, then the WS requester would select the most appropriate one for him and invoke it while he would not further participate in the matchmaking process. So a clear goal is to automate as possible as it can get the matchmaking process. To this end, there must be a way of collecting user preferences or user context in general while respecting user's privacy and his personal info. Users are not always accurate with their descriptions or it is not completely understood what they really want. As a result, the outcome of the matchmaking process is not satisfactory (too many usually irrelevant results or too few unsatisfactory results) and the user is called to refine his query. However, if user's context was available during matchmaking then user query could be refined automatically by the system and more importantly the results would get more satisfactory. For example, if user's location was available then the matchmaker would return services that were more "near" to the user. As another example, if the user had a preference of using only secure

transactional services, then the matchmaker would return services that implement secure and transactional protocols. As a last example, if the user's dealings were involving only highly reputable services, then the matchmaker would return services that had a value of the 'reputation' QoS attribute above a specific threshold. Thus, by collecting and describing (user and service) context, user's query can be expanded and be more refined in both the functional and QoS-based parts. As a result, a more automated context-aware and QoS-based matchmaking process will take place that will provide better matches to the WS requester.

Another issue that must be raised, which is related to the above, is "who is responsible for the application of the QoS filter". Is it the user, the registry or probably an agent? From the discussion of this section, it is clear that we assign the whole responsibility to the matchmaking engine, which usually resides at the registry. This is because the WS requester (user) does not want to get all the available Web Services from a WS registry and do all the filtering by himself. But when we add the QoS filter to the matchmaker, things change. The matchmaking process becomes more personalized and there are situations where negotiation must be established between the WS provider and requester, especially in cases where the requester adds another metric and wants from the provider to add a constraint about this new metric to his offer(s) or when the ontological representation of the metrics of the provider and requester are different. We do not want registries to be further burdened with mediations and ontological translations and mappings. The solution to this problem is the advent of user-customized agents who will either carry the whole matchmaking engine or perform only the QoS filtering sub-process. These agents will mediate, negotiate and in general perform complex reasoning tasks. Even when they cannot understand one ontological representation, they will know which Web Service(s) to invoke in order to translate the "unknown" ontological representation to a "known" one. In this way, WS registries will not be further burdened with QoS matchmaking or even with matchmaking and users can perform other tasks while waiting from their agent to come with the results of their WS enquiry. In addition, these agents will be equipped with user preferences (context). Therefore, they will apply more sophisticated queries to WS registries and they will perform a more personalized QoS filtering of the results from the previous sophisticated queries. Last but not least, do not forget that if user context is collected only by user agents, then it does not end up in the hands of the WS registry so the privacy problems are solved (maybe partially if goal filtering is performed at the registries and the registries 'pump' user context information from WS enquiries of the agents). Thus, our opinion is that user agents should be involved in the matchmaking process taking control of at least the sub-process of QoS filtering. Of course, if a WS provider and requester use the same metric ontology (maybe pertaining to the same domain) and if the requester does not want to launch an agent, then the QoS filtering sub-process can be performed by the contacted WS registry.

## 4.3 Requirements for QoS-based Web Service Selection

In the selection process, the results (especially the best ones) obtained from the discovery process are further ranked in order to get the best result. The criteria for ranking are usually non-functional but in the context of this paper we will stick only to the QoS-based ones. The selection process is completely personalized (preference-oriented) as the ranking is based on user preferences regarding the QoS criteria that are more important to him (along with the degree of importance) and different users may have different preferences. User preferences can be given separately to the selection algorithm or they can be obtained from the metric ontology. Alternatively, they can be collected and derived from user context (remember our previous discussion). User preferences should include: weights given to metrics (either generally or with respect to their domain or group), preferred domains or QoS groups (along with their weight if the importance of domains or groups differs), information about if the increase of a metric benefits the requester, maximum normalized values of metrics, etc. Apart from user preferences, other information about the QoS criteria/metrics must be available like the type and range set of a metric, the groups or domains the metric belongs to, what is the ordering of the range set of the metric, etc. This other type of information is collected/derived by the description of QoS metrics in (user or provider-defined or commonly agreed) metric ontologies.

When the selection algorithm collects all the appropriate input, it starts processing available QoS offers in order to give to each of them a suitable rank/degree according to user preferences. The degree is usually computed from the following formula: $\sum_{\forall metric} weight * degreeOfPromisedVal$, where the degree of promised value is determined by the metric's utility function [CMDTT05]. For example, if the metric is positively monotonic, then it can be computed by the next formula: $degreeOfPromisedVal = \frac{promisedVal - min}{max - min}$ [ZBD+03], where $promisedVal$ is the promised value of the metric, $min$ and $max$ are the minimum and maximum value of the metric with respect to all the available QoS offers. If the offer promises a range of values for a metric, then the $promisedVal$ can be the worse or the best or the average or a combination of worse and best values [KP06]. Other sophisticated selection algorithms are based on normalizations and grouping of QoS metrics (in domains or functional groups), as the ones described in [LNZ04]. The purposes of normalization are: **1)** to allow for a uniform measurement of service qualities independent of units, **2)** to provide a uniform index to represent service qualities for each provider, and **3)** to allow setting a threshold regarding the qualities. The number of normalizations performed depends on how the quality criteria are grouped (i.e. the nesting degree of the groups).

In case where there is a nesting of QoS groups, the WS selection algorithm calculates the above formula for every group (at the same level of the metric tree) taking into account the weight of each metric/group in its group, and do this recursively starting at the bottom of the metric ontology and ending at its root (consider that there is an imaginative root group, where groups and metrics make up a metric tree, and that the weights of all groups and metrics belonging in the same group have a sum of 1). It is important to note that according to the metric ontology, a metric tree (consisting of metrics and groups - all having weights) can be constructed. However, the user can give a different tree consisting of the

metrics and groups he prefers and he can give different weights from the defaults given by the metric ontology modelers. We could say that this metric tree can be described in a different ontological model than the metric model or can be given with the help of XML graphs (i.e. it is described in XML format).

### 4.3.1 Possible Implications – Obligations

Two issues must be discussed. The first one is who will be responsible for the ranking/selection process? To our opinion, the WS Selection process is a completely personalized process so it must be performed by the user or the user's "dedicated" agent, as the user or agent has all the available input to this process. Of course, in special cases (user provides all the appropriate input, metrics ontology and its representation are common between the registry, the provider and the requester), ranking can be performed by WS registries. So WS registries must be equipped with a WS ranking/selection process/operation.

The other issue is that the ranking process must be equipped with browsing/graph tools that will help the user in order to build the metric tree. In fact, with the help of these tools the user could provide in a complete and accurate mathematical way how the ranking of the matchmaking results will take place. In addition, QoS analysis tools must be available to the user in order for him to comprehend which are the most important QoS metrics for his system, what is their significance, and what values or thresholds these metrics should take. Moreover, the outcome of this analysis could be used by QoS demand specification tools. In other words, from QoS analysis until QoS demand specification (and possibly WS selection), WS requesters must be assisted by software packages, which will automate the above processes and will require minimum human intervention. In the same manner, WS providers must be assisted in QoS analysis and simulation and in QoS offer specification by appropriate software packages.

## 4.4 Conclusions

The goal of this chapter was to analyze the set of requirements we have identified and came up with after reviewing related work in QoS-based WS description and discovery. So this chapter was actually separated into two main parts: a) the first one analyzing both a set of requirements for a semantic, rich and extensible QoS model that encompasses a QoS metric model with the same features as the parent model and also other important features and capabilities of the QoS description language implementing the QoS model; b) the second one analyzing a set of requirements for a QoS-based WS matchmaking (including requirements of good performance, fine accuracy and great granularity/usufulness of the results) and selection (including requirements of user inequality constraints, normalization, best and worse performance consideration) algorithm. This set of requirements will be used in the following two chapters – 5 and 6 – in order to compare the related work against these requirements and to design and implement our contributions according to them.

# Chapter 5

# Related Work

Based on the requirements set on the previous section 4, in this chapter we provide a literature review of the current state-of-the-art research approaches in QoS-based WS description and discovery. Each approach will be analyzed in detail, bringing into light its main strengths and deficiencies (regarding our requirements). This chapter's purpose is to prove that there is a significant research gap in QoS-based Web description and discovery, paving the way to the next chapter that discusses how this thesis contributions close this gap.

This chapter is organized into six main sections. Each section except from the last one groups all the analyzed research approaches according to the amount of their functional contributions, from those offering the least functionality until those that are complete. However, approaches offering the least functionality does not mean that are not better in this functionality with respect to those that are functionally complete. More specifically, section 5.1 analyzes the work performed in QoS-based Web Service description while section 5.2 analyzes approaches offering a QoS-based WS selection algorithm. Section 5.3 discusses research approaches offering both a QoS description model and a matchmaking algorithm for WSs. Approaches offering both a QoS description model and a WS selection algorithm are analyzed in section 5.4. Then, section 5.5 analyzes research approaches that are complete, i.e. offering a complete framework for QoS-based Web Service description and discovery. Finally, section 5.6 summarizes the whole spectrum of research approaches and compares them against the requirements set in chapter 4.

## 5.1 Research Approaches for QoS-based Web Service Description

This subsection is going to analyze the related work in QoS-based WS Description, from simple ontologies to standard and non-standard approaches.

### 5.1.1 QoS Metrics/Measurement Ontologies

In the *DAML* Ontology Library[1] there is a DAML representation of the *Cyc* Upper Ontology [2]. In this ontology, the *UnitOfMeasure* as well as a number of its abstract subclasses are described. Among the subclasses are *UnitOfMoney* and *UnitOfMonetaryFlowRate* representing currency. Descriptions of concrete measurement units and currencies were not found. Consequently, it is not clear whether this ontology contains clear and easy specification of relationships of measurement units and how it solves dynamism of the relationships between currencies. In addition, the Cyc Upper Ontology is not modularized. This ontology can be the basis for future work on ontological representation of measurement units and currencies.

The *SHOE* (Simple HTML Ontology Extensions) measurement ontology [3] is much simpler than the CyC ontology. It contains some measurement units (metric system) in four categories: length, time, volume, and weight. No relationships (except the one stating a metric belongs to a category) and no derived measurement units are defined.

The work described in [KF02] is an ontology of measurement in enterprises, not an ontology of measurement of quality of Web Services. While this ontology can be used by third-party quality measurement Web Services, it does not seem directly applicable for specification of QoS constraints for Web Services.

[TEPP02] describe that for the specification of constraints for QoS metrics, five ontologies must be developed from which the most important (the top one) is the metrics ontology. They describe the structure and involved elements in four out of the five ontologies. However, they just stayed on the requirements for the specification of the metrics ontologies. They did not develop any ontology. In addition, the requirements specified are incomplete according to the requirements we posed. For example, the "metric" class consists only of five attributes while other important attributes/properties are missing. As another example, they imagine that metrics should be related to each other. However, they do not describe all the types of relationships that can appear between QoS metrics.

### 5.1.2 Standard Approaches

The Web Service Description Language [CCMW01] is a WS standard dedicated to the syntactic description of the signature of a WS operation. It does not describe the semantics of the components of the signature. In addition, it does not describe QoS values of QoS metrics for a WS operation.

The UDDI [BCC+04] WS standard is dedicated to the description and discovery of Web Services. However, it is based on the tModel concept which leads to purely syntactic queries. In addition, there is no QoS description of offers or demands in the UDDI description model.

The OWL-S [ea03] ontology is a semantic approach for the description of Web Services. It has many advantages in respect with the other WS description standards but it does not describe QoS offers or demands. It only contains an attribute used for rating a WS.

[1] On line at: `http://www.daml.org/ontologies`

[2] More information can be found at: `http://www.opencyc.org/`

[3] Available at: `http://www.cs.umd.edu/projects/plus/SHOE/onts/measure1.0.html`

However, as it is an ontological approach, it can be extended in order to describe QoS offers or demands.

### 5.1.3 Other Approaches

In [Ran03], an extension to UDDI is proposed. A new data structure type - called *qualityInformation* - is added to the UDDI model that represents description of QoS information about a particular WS. This proposed data structure is under the *businessService* data structure type, in addition to *bindingTemplate* data structure type, which provides binding information for a particular service. The *qualityInformation* data structure type also refers to *tModel*s as references to QoS taxonomies which also need to be defined in the extended UDDI registry. These taxonomies define the new terminologies or concepts about the proposed QoS information, which do not exist in the existing UDDI registries.

The proposed approach has many disadvantages. First of all, there is no actual description about the contents of the *qualityInformation* data structure type and its referenced *tModel*s. Secondly, it relies on the UDDI technology and UDDI's *tModel*, so it can be used only for syntactic matchmaking of Web Services. Thirdly, metrics and other aspects of QoS-based WS description are not defined. Fourthly, constraints on QoS parameters are not defined. Last but not least, there is no clarification of how the actual QoS matchmaking (of offers and demands) will take place.

The work described in [DSGF03] refers to calculation of values of those QoS attributes of a WS that it is not realistic to be provided/measured by service providers or third-party software modules. These attributes are usually measured by the values users submit from their experience of using the particular service. So the calculation of the value of the described QoS attribute will be a function of what value it should had taken and what value was eventually offered. The overall QoS can be given by the (weighted) sum of every QoS value of each attribute.

This work suggests a different mechanism based on user ratings and expectations for the calculation of the value of a QoS attribute. Usually, the value of a QoS attribute is calculated as a weighted sum of all user ratings for this attribute, where the weights denote the trustfulness we have for these users. This paper proposes that we should take into account (when calculating the value of a QoS attribute) only the ratings of users that have the same expectation for the value of a QoS attribute as with the one of the requesting user. This expectation does not only depend on the advertised value of a QoS attribute but also on other (contextual) factors like previous user experience from the service, the service usage cost, and recommendations from friends. So this expectation depends on the context of the user. Thus, when matchmaking, we should take into account the context of users which has an important role/impact in their ratings.

This work presents many disadvantages. First of all, QoS calculation is based on triples stored in a registry. These triples have the form ($expectedValue$, $perceivedValue$, $rating$) and are related to a QoS attribute. However, all parts of the triple take values from the [0, 1] space of real numbers. In addition, we trust all users who submitted ratings. Moreover, user expectation is only modeled by the $expectedValue$ of a user. So what is needed is better and

richer representation of user expectations and better techniques for matching expectations. Last but not least, this work does not explain what happens when the expectation for a QoS attribute of the requesting user is totally different from the ones collected.

In [MS02], an architecture and model of Web Service reputation (QoS) is presented. It is proposed that for successful description of QoS, three challenges must be dealt:

- definition of a QoS conceptual model for WS attributes which is reusable across domains such that weights, threshold to QoS attribute values, and user risk tolerance can be defined;
- semantics to QoS service attributes must be added in order for new attributes to be dynamically discovered and incorporated to the conceptual model of QoS attributes and for successful and more accurate discovery of Web Services;
- reputations should consider time and history of endorsements and ratings. Time must be considered as ratings should be taken into account when they are up-to-date, considering that QoS of a service changes over time. History of previous service usage is important as a service that is used many times must have high reputation. Endorsements of trusted agents/parties should also be weighted to compute reputation especially for new services that are not rated/used yet.

Based on the above requirements/challenges, a conceptual model of WS reputation is proposed which is used for the calculation of a WS reputation and is influenced on the following factors:

- relative weights given to QoS service attributes by the requesting user;
- QoS attribute aggregation algorithm for each QoS attribute;
- the set of endorsers of the service and the list of trusted endorsers of the user;
- the history of the service;
- damping for the rating such as older ratings matter less than newer ratings;

The suggested conceptual model enclosed a QoS attributes model. In this model, for each attribute the following aspects are defined:

- its type and allowed values;
- the domains it belongs along with a weight of this attribute in relation to the enclosing domain and user preferences;
- the characteristic function from attribute values to ratings;
- the time characteristics of the values of this attribute;

The main disadvantages of this work are the following. First of all, the reputation of a WS is calculated and not it's QoS. As it was described in previous sections, reputation can be considered one QoS attribute. Another disadvantage is that there is no explicit clarification of how the reputation of a WS is calculated. In addition, concepts like QoS constraints and QoS offers and demands are not modeled. Last but not least, QoS metrics conceptual model does not contain the classes and properties described in [TEPP02].

Work described in [TPP03, TMPE03], which is called *Web Service Offerings Language* (WSOL), proposes that a WS must offer different classes of service in order to satisfy a greater amount and type of customers and in order to deal successfully with situations where there is a variation in QoS due to network problems or mobility reasons. The authors of this work introduce the concept of "service offering", which is a formal representation of one class of service for a WS and contains formal definitions of various constraints (functional constraints, QoS constraints, and access rights), management statements (management responsibility, subscription prices, pay-per-use prices, monetary penalties to be paid if constraints are not met), as well as different reusability constructs (extension of service offerings, inclusion of constraint groups and parameterization of constraint group templates). A QoS constraint contains specification of what QoS metrics are monitored, as well as when and by what entity. However, definition of QoS metrics is done in external, reusable and extensible ontologies. QoS constraints usually describe QoS guarantees. Access rights specify conditions under which any consumer using the current service offering has the right to invoke a particular operation. In addition to all of these, specification of dynamic relationships between service offerings is enabled outside the specification of the service offerings. One such relationship could be that: if some constraints of the current service offering are not satisfied, then substitute the offering with another one.

This last feature is very important because in the usual case, when a service offering of a WS does not satisfy us, we try to find another suitable WS. This could cause substantial delay of desired execution/task. One the other hand, switching between different service offerings of the same WS seems more appropriate and faster and empowers the trust relationship between provider and consumer in contrast to re-negotiation of SLAs, switching between Web Services or re-composition of Web Services. However, it should be noted that appropriate alternative service offerings cannot always be found. Therefore, manipulation of service offerings is a complement to re-composition of Web Services or re-negotiation of SLAs or a lightweight replacement of re-negotiation of SLAs.

This work comes with two stated shortcomings/open issues. The first one is separation and integration of constraint dimensions, without conflicts and with straightforward implementation of constraint-checking code, which currently is not supported. The second one is the improvement of the specification of relationships between service offerings to support both easier and more flexible specification and dynamic adaptation. In addition to these shortcomings, there are some other problems. First of all, there is no specification of the QoS demand of the consumer. Secondly, the matchmaking process is not defined. Additionally, the metrics ontologies are not yet developed. Finally, we were not able to find the WSOL's complete specification so we cannot come into safe conclusions about its stated supported features.

The research approaches of *Web Service Level Agreement*(WSLA) [KL03] and *Web Service Management Language* (WSML) [JMS02] are similar. They try not only to provide a specification of SLAs but also to develop a complete framework for the management of SLAs. SLAs are agreements between service providers and consumers that provide guarantees for a particular service and define the common perceptions and expectations for that service. Comparing these efforts on the specification of SLAs, WSLA provides a better and more accurate language, providing more features and constructs, and satisfies a lot of the requirements we have set for QoS-based WS description. However, we should note that a SLA is different from a QoS offer or demand. It is an electronic document that is the outcome of the negotiation between a service provider and a service consumer (which negotiation is after the step of WS selection) and belongs to both of them. So it contains more management and responsibility statements, it is more technical and refers to implementation details that are not relevant to the context of WS discovery. For example, it states when and who is going to compute a resource metric accessing which resource, when QoS constraints should be evaluated, and what will be done in case one of the contracting parties fails to meet its obligations. All this information is not needed for matchmaking, but it could be incorporated in a QoS specification in a limited form and used as a starting point for the more involved task of WS negotiation. Last but not least, due to the existence of more management and monitoring constraints, the framework supporting SLAs is more heavy-weight than the framework for QoS-based discovery.

The research effort described in [TGN+03] analyzes what must be enclosed into the QoS information for a WS request or advertisement with the help of a QoS ontology. Important elements of this ontology are *QoSInfo* and *QoSDefinition*. *QoSInfo* describes standard or user-defined *serverQoSMetrics* and *transportQoSPriorities* and the values they will take. It also references protocols used by a WS for security and transaction support. The *QoSDefinition* element describes QoS information (*QoSInfo*) either for the whole service or for every operation of the service. Additionally, it includes information about protocols supporting service management and QoS monitoring and about the trusted third-parties that will participate in these protocols. It ends with the price for the usage of this service supporting the QoS offer. One WS advertisement is related to many service offers (*QoSDefinition*) while one service request enquires one particular service offer.

One important feature of this research effort is that it supports the mapping of QoS requirements from higher layers onto the underlying network in terms of the Internet model. This mapping is achieved by the help of proxies (residing at the provider and consumer) and by the existence of a QoS-aware network. QoS network parameters are given as guidelines to QoS-aware routers while the client proxy calculates the network performance by taking into account the server performance information provided by the server proxy.

This research work comes with three main deficiencies. First of all, there is not a complete and accurate description of QoS constraints as QoS constraints are just equality constraints. Secondly, metrics ontologies are not developed but are just referenced. Finally, there is no specification of how WS matchmaking or selection will take place.

QoSOnt [DLS05] is another carefully designed ontology for semantic QoS-based WS description. Its main features are: a) Measurable QoS attributes can be measured by many

QoS metrics; b) A QoS metric is applied either to the whole WS or just one operation; c) Proposal of converting units based on SWRL rules; d) Direct connection to OWL-S. This is a very good approach but not a complete one as it does not capture most of the requirements set in section 4. In addition, it is not accompanied by a formal WS discovery framework.

The work in [TRPA06] proposes an upper-level ontology that uses the main features of the ontologies produced by the works of [MS04, DLS05]. In addition, a QoS ontology vocabulary (actually a mid-level ontology) has been designed for domain-independent QoS properties. This is a rich ontology that is also connected to OWL-S. However, it lacks information on how QoS constraints are specified and it is not publicly available. In addition, it is not supported by a WS discovery framework.

## 5.2 Research Approaches for QoS-based Web Service Selection

Zeng et. al. [ZBD+03] propose a QoS-based WS selection algorithm that is the backbone of most QoS-based WS selection approaches. This algorithm considers the case that each WS $S_i$ advertises one value $Q_{ij}$ for each QoS attribute $Q_j$. It consists of two sequential phases: *scaling* and *weighting*. The *scaling* phase normalizes the value $Q_{ij}$ of each WS and produces a new value $V_{ij}$ in [0, 1] according to two cases based on the monotonicity of the QoS attribute $Q_j$:

$$V_{ij} = \begin{cases} \frac{Q_j^{max} - Q_{ij}}{Q_j^{max} - Q_j^{min}}, \text{ if } Q_j^{max} - Q_j^{min} \neq 0 \\ 1, \text{ if } Q_j^{max} - Q_j^{min} = 0 \end{cases} \tag{5.1}$$

$$V_{ij} = \begin{cases} \frac{Q_{ij} - Q_j^{min}}{Q_j^{max} - Q_j^{min}}, \text{ if } Q_j^{max} - Q_j^{min} \neq 0 \\ 1, \text{ if } Q_j^{max} - Q_j^{min} = 0 \end{cases} \tag{5.2}$$

where $Q_j^{max} = \max\{Q_{ij}\}$ and $Q_j^{min} = \min\{Q_{ij}\}$ for each $i$. The first case (Eq. 5.1) is for negatively monotonic QoS attributes (e.g *price*) while the second case (Eq. 5.2) is for positively monotonic QoS attributes (e.g *availabity*). Finally, at the *weighting* phase the score $sc_i$ for each WS $S_i$ is produced according to WS requester's weight $w_j$ given to each QoS attribute $Q_j$ by the following formula:

$$sc_i = \sum_{j=1} w_j \times V_{ij}$$

The algorithm of Zeng et. al. is quite simple not taking into account user constraints (the value that the WS requester expects for each QoS attribute – values near this value are better than values more distant) and other user preferences or general criteria (user's QoS tree, QoS groups, etc.). In addition, this algorithm only considers the case where each WS provider advertises one value for each QoS attribute for his WS and not a range of values.

The research work described in [LNZ04] refers to the need for an extensible QoS model that not only contains general but also domain-specific QoS criteria. It sustains that QoS must be represented to users according to user preferences and users should express accurately their preferences with this QoS model without resorting to complex coding of user-profiles. It also suggests that QoS computation must be fair and open for providers and requesters. Then it proposes an extensible QoS model. However, this model is just a description of some general QoS criteria and other criteria that can be grouped and not a formal ontological description of QoS constraints regarding some metrics that also defined by suitable ontologies. Finally, based on this model and the above requirements, it develops a QoS computation algorithm. This algorithm takes as input QoS descriptions of available Web Services (metric=value descriptions) and user preferences (metrics that benefit the user, maximum normalized values for metrics, in what groups metrics should be contained, etc.). This algorithm performs two sequential normalizations in order to derive a unique value for the QoS description of each service: a) the first one normalizes the value of each QoS attribute of a WS; b) while the second one produces normalized values for each QoS group for each WS according to the normalized values of the QoS attributes of these WSs. The unique value of each WS description is produced by the weighted sum of the multiplication of the weight of every QoS group with its normalized value calculated according to this WS. This selection algorithm presents the same disadvantages as the algorithm of Zeng et. al. [ZBD+03] although it goes one step further by calculating the score of each WS based on one level nesting of QoS groups (that is QoS attributes belong to their parent QoS groups and there is no further nesting i.e QoS groups belonging into other QoS groups).

## 5.3 Research Approaches for QoS-based Web Service Description and Matchmaking

*QoS Modelling Language* (QML) [FK98] is another research effort for the specification of QoS. It was designed according to some basic principles for the support of QoS specification. It contains the following constructs:

- **contract type**: Definition of a QoS dimension that includes definitions for the metrics of this QoS dimension.
- **contract**: Gives particular values/constraints to the fields of a contract type. This is where the idea of contract inheritance is implemented.
- **profile**: One service is associated with many (QoS) profiles. Each profile consists of one list of contracts/requirements. Each contract may describe constraints for a QoS dimension either for the whole service or just for one service operation. But for every QoS dimension, at most one constraint will be valid for one operation of the functional interface of the service.

One (QoS) Service Profile $P$ is matched with one client profile $Q$ if all contracts of $P$ conform to all the contract of $Q$. Contract conformance is translated into constraint conformance. Constraint conformance is separated into two cases depending on the QoS metric:

- for set domains, we first have to define if superset stronger than subset or the opposite;
- for numeric domains, we have to know if the metric is increasing or decreasing. If increasing, then $d > 10$ is stronger than $d > 5$. If decreasing, then $d < 20$ is stronger than $d < 10$.

In general, if $Q$ is a refinement of $P$, $Q$ will also conform to $P$. Considering the language for specifying QoS offers or demands, it conforms to many of our requirements. However, it does not use ontologies (especially metrics ontologies) so it lacks the semantics needed for better matchmaking.

Matchmaking is based on the concepts of contract and constraint conformance. However, the matchmaking process if it is based only on these concepts will provide two results sets: matched service profiles and not matched service profiles. That is no additional ranking is performed to see if the matched service profiles are better or equivalent to the client profile. Thus the user is not assisted very much and could be provided with a huge result set of conformant (matched) service profiles.

In [ZCL04], DAML-S Web Service description language is extended to include a QoS specification ontology. This is achieved by the following:

- A *ServiceProfile* element is associated to many QoS profiles (service offerings).
- External ontologies in DAML for metrics and units are referenced or developed.
- Existence of a *BasicQoSProfile* containing all the basic metrics and ability to inherit/extend this type of profile to provide constraints and/or include custom-made metrics.

In addition to the DAML-S extension, a novel QoS matchmaking algorithm is proposed, which is performed after functional matchmaking, and separates the result-set in five categories. This QoS matchmaking algorithm is based on the concept of QoS profile compatibility that states the following: "Two QoS ontology descriptions, say $C_1$ and $C_2$, are compatible if and only if their intersection is satisfiable: compatible $(C_1, C_2) \Leftrightarrow \neg (C_1 \cap C_2 \sqsubseteq \bot)$". Matchmaking is performed by a DL reasoner that computes the subsumption relationship of a request $R$ (and $\neg R$) with all available QoS advertisements.

The deficiencies of this research effort are the following:

- The metrics and units classes described do not contain all the appropriate properties and relationships so further work must be done.
- The QoS metrics values are restricted to have the set $\mathbb{N}^+$ as their range in order to help the DL reasoner in calculating the T-Box subsumption relationships. However, this leads to imprecision and errors that can reach one half of measurement unit. In addition, this feature is based on a misuse of the OWL cardinality constraints to

express bounds on QoS metrics. A cardinality constraint restricts the number of values a property can take, not the values themselves.

- DL reasoners are not very quick and do not support the most complex mathematical expressions.
- No further WS selection according to user preferences is performed.

In [DSL04], there is an extension to WSDL in order to include constructs for constraint specification of WSs. By using these constructs, a service provider can specify meaningful attributes for characterizing a WS and its operations. Attribute constraints and inter-attribute constraints can be explicitly defined. Attribute constraints are of the form: $(x >$ or $\geq a)$ or $(x <$ or $\leq a)$, where $x$ is a numeric or string-based variable mapped to a QoS variable and $a$ is a value taken from the domain of variable $x$. Inter-attribute constraints are of the form: IF $A$ THEN $B$, where $A$ and $B$ are boolean expressions of constraints constructed from the logical operators AND and OR. In addition, a Constraint Satisfaction Processor was developed for matching the requirements given in a service request against the constraints of registered services in the service discovery process. This processor and some additional components are integrated with the IBM's UDDI registry to form a Constraint-based Broker. The drawbacks of this approach are the following:

- The QoS model is not rich enough as it does not cover every possible aspect, especially that of a QoS metric.
- The QoS model is structural and not semantic.
- The constraint specification language is not rich enough as it only allows attribute constraints and simple inter-attribute constraints. This language does not also include linear and non-linear attribute functions.
- The matchmaking metric used is incorrect. A QoS offer is matched with a QoS request if all its QoS metrics have common values with the same QoS metrics of the request. This is wrong as a QoS offer may promise lower quality values for a metric that are not acceptable by the QoS request.

## 5.4 Research Approaches for QoS-based Web Service Description and Selection

WSMO-QoS [WVKT06] is an upper level ontology complementary to the WSMO (`www.wsmo.org`) semantic language for functional WS description. Besides this upper-level ontology, a vocabulary of general domain-independent QoS attributes has also been developed. WSMO-QoS is a very rich ontology capturing many aspects of QoS metric description. It includes and allows many metric value types (linguistic, numeric, boolean), dynamic calculation of metrics values, the attachment of units to metrics, unit transformation functions,

the expression of the tendency of the metric's value from the user's perspective and grouping of QoS attributes. The main deficiencies of this ontology are the following: a) there is a one-to-one mapping of QoS attributes and metrics, which is incorrect; b) no measurement modeling (functions and measurement directives of metrics are not expressed); c) only equality constraints on metrics are allowed, which is quite restrictive; d) not publicly available yet.

This ontology is supported by a QoS-aware selection framework of WSs that uses a WS selection algorithm similar to the one of [ZBD+03]. The only difference with respect to the algorithm of [ZBD+03] is that the proposed algorithm is different when the WS requester defines a QoS attribute/metric with tendency="given" instead of "low" (negatively monotonic QoS attribute) or "high" (positively monotonic QoS attribute). In this case, the score $sc_{ij}$ of the value $v_{ij}$ of each QoS attribute $Q_j$ of each WS $W_i$ is calculated by the following formula:

$$sc_{ij} = \begin{cases} 1 - \frac{q_{max} - q_{ij}}{q_{max} - q_{min}}, & \text{if } r_j \geq q_{max} \\ \frac{q_{ij} - q_{min}}{q_{max} - q_{min}}, & \text{if } r_j \leq q_{min} \\ 1 - \left( \left| \frac{|q_{ij} - r_j| - m}{n - m} \right| \right), & \text{if } r_j \in (q_{min}, q_{max}) \end{cases} \tag{5.3}$$

where $1 \leq i \leq N$, $1 \leq j \leq k$, $r_j$ is WS requester's value for QoS attribute $Q_j$, $q_{max} = \max\{q_{ij}\}$, $q_{min} = \min\{q_{ij}\}$, $n = \max\{|q_{ij} - r_j|\}$ and $m = \min\{|q_{ij} - r_j|\}$. This is actually the case where a WS requester requires the value of a quality property to be as close as possible to his value. By closely examining Eq. 5.3, it is easy to see that the first case can be simplified while the second case is wrong. Thus, Eq. 5.3 can be rewritten as:

$$sc_{ij} = \begin{cases} \frac{q_{ij} - q_{min}}{q_{max} - q_{min}}, & \text{if } r_j \geq q_{max} \\ \frac{q_{max} - q_{ij}}{q_{max} - q_{min}}, & \text{if } r_j \leq q_{min} \\ 1 - \left( \left| \frac{|q_{ij} - r_j| - m}{n - m} \right| \right), & \text{if } r_j \in (q_{min}, q_{max}) \end{cases} \tag{5.4}$$

This last feature represents an excellent extension to the WS selection algorithm of Zeng et. al. [ZBD+03]. The main drawback of the selection algorithm under examination is that it does take into account the QoS groups that were actually accounted and specified in WSMO-QoS in order to perform a second tree-based normalization producing the final score for each WS.

Work analyzed in [MS04] is actually a continuation of the work in [MS02]. The requirements of the work in [MS02] have been translated to a quite expressive ontology language. Its main highlight is the formalization of relationships between QoS attributes. When a QoS attribute depends on another one, then either its values influence the values of the other with a specific impact or the values of these attributes change in a parallel or in inverse parallel way. A framework using the ontology to support dynamic web services selection is also outlined. The main drawback of the proposed ontology is the lack of a metric model.

In addition, this ontology lacks both an openly available implementation and links to a semantic description WS language like OWL-S.

## 5.5 Complete Research Approaches

The research effort described in [CMDTT05] refers to QoS-based Web Service Description and Discovery so it is a complete approach. It uses a symmetric QoS model expressing mathematical constraints for QoS metrics and user preferences (weights given to metrics and a function expressing the utility assessment of the metric's value with a range of [0, 1]). However, QoS metrics are not ontologically defined so semantics are missing. Before matchmaking, a QoS specification is transformed to a Constraint Satisfaction Problem (CSP) [VHS96] which is checked for *consistency* i.e. if there is an assignment of values to metrics (a solution) such that all the constraints are satisfied. Matchmaking is performed according to the concept of *conformance*, which is used for checking out if every solution to the CSP of the offer is also a solution to the CSP of the demand. After matchmaking, two main result-sets are produced: matched offers w.r.t the demand and not (completely) matched offers. Unfortunately, the consistency and conformance of CSPs does not always lead to polynomial computation of solutions especially if there are non-linear expressions at the QoS constraints. This is a characteristic of the class of CSP problems. In addition, the matchmaking metric is not totally correct, in these cases where the user provides two unary range constraints to specific QoS metrics and these constraints should not be both respected. For example, if the WS requester provides the constraints: $x \geq 0.8$ and $x \leq 0.9$ to a QoS metric measuring *availability*, then only the first one should be respected as the second one may discard those QoS offers promising a better value for the same metric of *availability*.

Concerning Web Service Selection, the (QoS) score of a Web Service advertisement is calculated as a weighted sum of the weight of each metric multiplied with its utility assessment value, where the assignment of values to metrics is chosen so that the sum is the minimum. That is Web Service Selection is expressed as a Constraint Satisfaction Optimization Problem (CSOP) [VHS96] where from all solutions to the CSP of an offer we try to find the one that minimizes the previously described weighted sum. Utility assessment values are calculated by utility functions that are specific to the QoS metric examined. For example, the two cases of the selection algorithm of Zeng et. al. [ZBD+03] are actually utility functions for positively and negatively monotonic QoS metrics respectively. Of course, different and even non-linear functions can also be used, especially when the monotonicity of the QoS metric is not trivial. This selection algorithm has worse performance with the corresponding matchmaking algorithm as it tries to solve optimization problems using CP techniques. Additionally, both algorithms are purely syntactic i.e. they can not figure out if two metrics that have different names are exactly the same. Finally, this selection algorithm does not perform a second normalization based on QoS groups enclosing QoS attributes.

The work in [OVSH06] semantically enriches the WS-Agreement language in order to develop a semantic framework for matching agreement specifications of providers and re-

quester automatically. The WS-Agreement language is extended in important areas such as the *SLO* and *QualifyingCondition* with the addition of the expression, predicate, parameter, and value tags as defined in the WSLA specification [KL03]. In addition, 4 ontologies are used to provide a commonality of terms between agreement parties and to provide rich domain knowledge to the search engine so that it may achieve the best possible match results: 1) an OWL ontology for representing the WS-Agreement schema; 2) an OWL-based QoS ontology encompassing QoS concepts used in guarantees; 3) a third OWL ontology representing domain knowledge; 4) the OWL Time ontology [HP04] for representing temporal constructs such as *endTime*, *interval*, *dayOfWeek*, and *seconds*. Moreover, this approach uses SWRL rules in order to: a) transform one *SLO* to another one that is semantically similar but syntactically heterogeneous with respect to the first one; b) to compare *SLO*s according to the semantics of a domain specific predicate; c) to derive new domain knowledge by e.g. producing a new *SLO* from one or more other *SLOs*; d) to enable user assertions over subjective personal preferences. Last but not least, it must be stated that this extended language is connected to WSDL-S (`www.w3.org/Submission/WSDL-S/` and is supported by a complete semantic QoS-based WS discovery framework. QoS-based WS matchmaking is actually performed according to the semantics of *conformance* [CMDTT05]. The WS selection process is performed by marking some *SLO*s as *preferred* and producing higher scores to those WSs having the higher number of *preferred* SLOs. This work has the following deficiencies: a) QoS metrics are not modeled at all; b) *SLO*s of guarantees are expressed only in terms of unary constraints (i.e. containing just one QoS concept); c) Although timing conditions are expressed in *guarantee*s, this does not happen with the whole *alternative*; d) the WS Selection process is quite simplistic and may produce the same scores to two different WS offers – this process should assist the user by providing different scores to different WS offers; e) as explained above, the *conformance* matchmaking metric is not correct for some QoS metrics.

The onQoS-QL [GZ07] ontology is very rich encompassing all appropriate aspects of QoS-based WS description by respecting the requirements set in section 4. It is also supported by a semantic QoS-based WS discovery framework. Its main highlights are: a) the use of scales for restricting the metric value types and the comparison predicates on metrics; b) the use of unary and binary predicates for constructing metric constraints; c) metric constraints have both retrieval and ranking semantics; d) many important types of measurement processes are supported. According to the c) point, each QoS metric predicate not only filters out those QoS offers not satisfying the constraint but also returns a value in the real set [0 1] indicating the satisfaction degree of the constraint. The main drawbacks of this work are: a) not all appropriate details are expressed for measurement functions and directives b) only unary and binary metric comparison predicates are used for expressing QoS constraints on QoS demands while function predicates (e.g statistical functions) are used only in metric definition; c) the QoS profile of a WS only contains a set of metrics and not a set of constraints on QoS metrics – this means that the onQoS-QL framework should know and store all measurements performed on a QoS parameter or to explicitly call external functions each time it must evaluate a constraint on a QoS metric of a WS (quite time consuming). In addition, another problem caused by the last drawback is that the WS provider cannot

enforce constrains on the WS requesters that perform the querying. Finally, this research approach also proposes a QoS metric derivation algorithm that produces new QoS metrics and values on them based on semantic rules. This algorithm is used in order to produce the same set of metrics on both the providers and requesters QoS specifications so as to increase the precision and recall of the QoS-based WS discovery process. So this algorithm can be regarded as a first good step in aligning QoS-based WS specifications as it also performs limited QoS metric matching but only in obvious cases. Unfortunately, more details about this algorithm are not revealed in order to come to more safe conclusions. In addition, the onQoS ontology is not publicly available.

## 5.6 Comparison

The goal of this section is to compare all of the reviewed research approaches against the requirements set in chapter 4. However, as it is not right to bore the reader with repetition of information already provided in the previous subsections, this comparison is established with the use of three different tables. Each of these tables focuses on the requirements set for one functionality – that is QoS-based WS description, matchmaking, and selection – and informs the user about the offerings of each research approach. By providing these tables, an overview of the research approaches and their achievements is supplied and the conclusion of the literature review is easily proved and comprehended.

Before proceeding to presenting the three tables, it must be stated that some approaches will be present in two or even the three tables as they offer more than one functionality. In addition, the general notation used in each table must be explained. For each requirement of a functionality, the symbol "–" denotes that this requirement is not satisfied at all if it is present in the row of a particular research approach while the symbol "yes" denotes that this requirement is specified. For requirements where simple no or yes answers are not adequate, we have the following symbols and their explanations: the word "poor" denotes that the requirement is poorly supported, the word "fair" denotes that the requirement is partially supported, in cases where a requirement is basically supported, the word "good" will appear, while in cases where a requirement is totally supported, the word "excellent" will be present.

The first table is split into tables 5.1 and 5.2 summarizes the satisfaction of each requirement by each research approach reviewed for the QoS-based WS description process. The main requirements are: *extensible and formal QoS model*, *syntactical separation*, *both client and service QoS specification*, *refinement of QoS specifications*, *symmetric model of QoS specification*, *QoS attributes model*, *constraint definition and expressiveness*, and *classes of service specification*. As these requirements are already explained in section 4.1, they do not need any special introduction and analysis. Each approach and its achievements appears at the rows of this and the other tables.

The second table 5.3 summarizes the satisfaction of each requirement by each research approach reviewed for the QoS-based WS matchmaking process. The main requirements are: *QoS metric matching*, *precision*, *recall*, *performance*, *result categorization* and *optimization*.

| Research Approach | Extensible & Formal QoS Model | Synt/cal Sep/tion | Both Client & Service Spec | Ref/ment of QoS Specs | Fine-Grained QoS Spec | Symmetric Model of QoS Spec | QoS Attr/s Model | Constraint Def/tion & Expr/ness | Classes of Service Spec |
|---|---|---|---|---|---|---|---|---|---|
| Cyc | – | – | – | – | – | – | fair | – | – |
| SHOE | – | – | – | – | – | – | poor | – | – |
| [KF02] | – | – | – | – | – | – | fair | – | – |
| [TEPP02] | – | – | – | – | – | – | good | – | – |
| WSDL | – | – | – | – | – | – | – | – | – |
| UDDI | – | – | – | – | – | – | – | – | – |
| OWL-S | – | – | – | – | – | – | – | – | – |
| [Ran03] | yes but syntactic | – | yes | – | – | yes | poor | poor | – |
| [DSGF03] | – | – | – | – | – | – | poor | – | – |
| [MS02] | yes | – | yes | – | – | – | fair | – | – |
| [TPP03] | yes but syntactic | yes | – | yes | – | – | good | good | yes |
| [JMS02, KL03] | yes but syntactic | yes | – | yes | – | – | good | excellent | yes |

Table 5.1: Comparison of Research Approaches in QoS-based WS description

| Research Approach | Extensible & Formal QoS Model | Synt/cal Sep/tion | Both Client & Service Spec | Ref/ment of QoS Specs | Fine-Grained QoS Spec | Symmetric Model of QoS Spec | QoS Attr/s Model | Constraint Def/tion & Expr/ness | Classes of Service Spec |
|---|---|---|---|---|---|---|---|---|---|
| [TGN+03] | yes but syntactic | yes | yes | – | – | yes | fair | poor | yes |
| [DLS05] | yes | yes | yes | yes | yes | – | fair | poor | yes |
| [TRPA06] | yes | yes | yes | yes | – | yes | good | poor | yes |
| [FK98] | yes but syntactic | yes | yes | yes | yes | yes | good | good | yes |
| [ZCL04] | yes | yes | yes | yes | yes | yes | good | poor | yes |
| [DSL04] | yes but syntactic | – | yes | yes | – | yes | fair | fair | – |
| [WVKT06] | yes | yes | yes | yes | – | yes | good | poor | yes |
| [MS04] | yes | yes | yes | yes | – | yes | good | poor | – |
| [CMDTT05] | yes but syntactic | – | yes | – | – | yes | fair | good | – |
| [OVSH06] | yes but | – | yes | yes | – | yes | good | poor | yes |
| [GZ07] | yes | yes | yes | yes | – | yes | excellent | good | yes |

Table 5.2: Comparison of research approaches in QoS-based WS description (cont.)

| Research Approach | Semantic QoS Metric Matching | Precision | Recall | Performance | Results Cat/tion | Optim/tion |
|---|---|---|---|---|---|---|
| [FK98] | – | excellent | good | – | fair | – |
| [ZCL04] | – | good | good | good | good | – |
| [DSL04] | – | fair | excellent | excellent | fair | – |
| [CMDTT05] | – | excellent | good | excellent | fair | – |
| [OVSH06] | – | excellent | good | good | fair | – |
| [GZ07] | yes (limited) | excellent | good | good | fair | – |

Table 5.3: Comparison of research approaches in QoS-based WS matchmaking

The first requirement express the fact if semantic QoS metric matching is performed in order to align the QoS-based WS specifications before the actual matchmaking is executed. The second and third requirements define the accuracy of the matchmaking metric and subsequently of the matchmaking algorithm with the use of two metrics [DMR02]: *precision* (indicating if all results returned are correct) and *recall* (indicating if all correct results are returned). By expressing *precision* or *recall* as "excellent", their values are equal or very close to 1.0. If they are "good", their value is above 0.5. "Fair" notations for these two properties indicate that the values of these properties can go below 0.5. The *performance* requirement affects how fast the matchmaking algorithm executes while the next requirement expresses the need for an *advanced categorization* of results assisting the user in understanding and using these results. Finally, the last requirement affects the ability of the matchmaking system to deal with over-constrained demands and return meaningful results.

The third table 5.4 summarizes the satisfaction of each requirement by each research approach reviewed for the QoS-based WS selection process. The main requirements are: *user preference specification*, *user constraints enforcement*, *normalization*, and *QoS grouping*. The first requirement specifies how well the user requirements are captured and defined by the research approaches. The second requirement specifies if user constraints are taken into account by the approach's selection model. The third requirement specifies if the values for all QoS properties/metrics are normalized in order to have a fair selection model. The last requirement specifies if the selection model follows a tree-based computation approach in order to produce a score for a QoS offer.

As can be understood from all the tables shown, there is no research approach that supports all the requirements we have set for each process of QoS-based WS description and discovery. This identified gap can be covered by an approach that is carefully designed and implemented based on all the requirements of chapter 4. Such an approach is the one that was carried out in terms of this PhD thesis and is revealed in the next chapter. As it will be shown, our approach satisfies in the best way all of these requirements. Moreover, the evaluation of our approach empirically proves our claims.

| Research Approach | User Preference Spec | User Constraints | Normalization | QoS Grouping |
|---|---|---|---|---|
| [ZBD+03] | fair | – | yes | – |
| [LNZ04] | good | – | yes | yes (one-level) |
| [WVKT06] | good | yes | yes | – |
| [MS04] | fair | – | – | – |
| [CMDTT05] | good | – | yes | – |
| [OVSH06] | good | yes | – | – |
| [GZ07] | fair | yes | yes | – |

Table 5.4: Comparison of research approaches in QoS-based WS selection

## 5.7 Conclusions

In this chapter, we have provided a literature review of the current state-of-the-art research approaches in QoS-based WS description and discovery, based on the requirements set in the previous chapter. Each research effort was analyzed in detail, bringing into light its main strengths and deficiencies in accordance to our requirements. The chapter was organized into six main sections, where in each section from the first five its corresponding research approaches were analyzed. So the research efforts were actually grouped thematically according to the amount of their functional contributions, from those offering the least functionality until those that are complete. However, approaches offering the least functionality does not mean that are not better in this functionality with respect to those that are functionally complete. This latter fact along with other details could be derived from the three unitary tables of the last section with which we tried to compare in a summarized way the corresponding research approaches according to the requirements of section '4. To be more specific, each table was devoted to comparing the presented research efforts (whenever this was applicable) against the requirements of either QoS-based WS description or QoS-based WS matchmaking or QoS-based WS selection. In this way, we could prove and actually it was easily perceivable from the three comparison tables that there was a significant research gap in QoS-based Web description and discovery. This research gap is aimed to be closed by the research approach we propose and analyze in the next chapter.

# Chapter 6

# Proposed Approach

Based on the literature review of the previous chapter, there is a significant research gap in QoS-based Web Service description and discovery. This gap is covered by the contributions of this thesis which are the following: **a)** a semantic, rich and extensible QoS description model for WSs; **b)** a semantic alignment algorithm for QoS-based WS specifications; **c)** development of different QoS-based WS discovery algorithms optimized according to the solving technique used and the user requirements.

Before analyzing our three thesis's contributions, the first section 6.1 is dedicated to the analysis of which instance of the main problem was solved by the provision of assumptions and restrictions. The remaining three sections are dedicated to the theoretical analysis for the three main contributions of this PhD thesis providing answers to the questions of in which way, by which mean and how well the instance of the main problem is solved. The practical or experimental evaluation of our contributions is provided in chapter 7.

## 6.1 Hypothesis

First of all, it must be noted that the problem of QoS-based WS description and discovery is very complex and many factors should be taken into account in order to solve it successfully. Thus, a complete solution cannot be accomplished easily and requires the contribution of many researchers or scientists. The main purpose of this thesis was to provide a theoretical solution of a specific instance of this problem. This instance is set by posing constraints and assumptions both on the WS description and discovery processes and on the entities that participate in these processes. The theoretical solution of this instance of the main problem was accomplished by:

- a semantic, rich and extensible QoS-based WS description language and a methodology of how to use it;

- an algorithm aligning QoS-based WS specifications using a "QoS metric matching" algorithm as a component that offers the functionality of inferring if two QoS metrics are equivalent;
- QoS-based WS matchmaking and selection algorithms for the QoS-based WS discovery process exploiting the CP and MIP solving techniques

In addition to the theoretical solution, a software system was developed implementing the above algorithms and solutions that takes as input a list of QoS-based WS advertisements and requests.

Based on the fact that the QoS-based WS description sub-process is part of the WS description process, its relative order in relation to the other sub-processes must be determined along with the actions and restrictions of the SOA entities regarding the manipulation of their objects. Below we provide a complete list of our assumptions and constraints:

1. Obviously, a Service Provider (SP) has specific software. So he knows the functional capabilities and behavior of his software. When he makes it available on the Internet as a WS, then he also knows its signature. After using a QoS analysis tool, the SP is aware of what QoS can deliver to his customers and of what is its impact to the resources used. So he can estimate the QoS offers that are associated to his WS. Therefore, the SP knows in advance the functional capabilities and the behavior of his service while then he can determine the signature and the QoS of his service.

2. The SP registers (in a registry or other type of repository) a complete WS description or only its functional part (interface, capability, behavior). In the first case, the WS description/advertisement contains one functional part that is associated with one or more QoS-based offers. In the second case, the SP can later complete his advertisement by registering QoS offers related to the functional part of his WS advertisement. Therefore, the functional description registration must precede or be concurrent with the QoS one.

3. A QoS offer has time validity. It is the responsibility of the registry and its underlying mechanisms to discard QoS offers that have expired. However, in the developed QoS-based WS discovery algorithms we did not check the time validity of the QoS offers but as this is a very easy task, it can be performed in the future.

4. A QoS offer must conform to the QoS-based WS description requirements that were specified in chapter 4. It should also be described by the proposed and developed QoS-based WS description language called OWL-Q (which also conforms to the description requirements). It may use the mid-level (generic) concepts (metrics, units, measurement functions, e.t.c.) specified with OWL-Q. It may also use low-level (domain dependent) concepts developed by domain modelers with the format of OWL-Q.

5. The specific values or range of values of QoS metrics that appear in QoS constraints in a QoS offer are usually given by the SP. However, there will be cases where a third-party entity monitors the delivery of the WS under the promise of a specific QoS

offer. In this case, the third-party entity renders some (if not all) of the values of the QoS metrics. In addition, there will be cases where a QoS offer does not contain constraints of specific QoS metrics as these metrics can only be measured by the WS requester. To meet this, the registry can collect these requesters' observations (either by asking the requester or the requester submits them), transform them to the appropriate default unit, estimate the range of values of these QoS metrics and add the corresponding constraints to the appropriate QoS offer. Of course, this last addition of QoS constraints to a QoS offer is not safe because the requester may not be telling the truth about his observations or he may not have used the WS or the specific QoS offer was not promised to him. According to the context of this thesis, it was expected that the registry delivers to the proposed discovery algorithms the best possible, safest and fair description of QoS offers.

6. When the SP erases the functional part of a WS advertisement, then all the corresponding QoS offers should be erased, too, by the registry. In reverse, if the SP erases all the QoS offers of a WS advertisement, then the registry should erase the functional part, too. It may be more appropriate for the registry to provide a single function for erasing WS advertisements.

7. All the before-mentioned new underlying mechanisms (time validity, registration of QoS offers, association of parts of an advertisement, erasure of advertisements) of a registry were not implemented during the time of this PhD. Thus we consider them as future work and hypothesize that are implemented by the registry hosting our developed prototype software.

As QoS-based WS discovery is a sub-process of the WS discovery process, we follow the same procedure as above and supply a complete list of our assumptions and constraints:

1. It is assumed that the WS (functional) matchmaking sub-process precedes the QoS-based one. This is because the WS requester is mainly interested in finding out which WSs satisfy particular functional criteria in order to achieve his goal. Then he is interested in finding out which of the returned WS advertisements has the best performance to price ratio.

2. In consequence, it is expected from the WS functional matchmaking sub-process to hand over a list of candidate WS offers. These offers are derived from the calculated list of WS advertisements that satisfy the functional criteria of the WS requester. In addition, the aforementioned sub-process must hand over the QoS demand/request and QoS selection criteria of the WS requester. These QoS request and selection criteria are part of the whole WS request, which is submitted to this sub-process initially by the WS requester.

3. Thus, the input to the QoS-based discovery process is a list of QoS offers, a QoS demand and a set of QoS selection criteria.

4. It is assumed that the WS functional matchmaking sub-process was perfectly executed, returning ideal results. Currently, this sub-process is not completely recallable and

precise due to reasons that were previously analyzed in chapter 2. The results returned are expected to be perfect (functional) matches as this is the most interesting result category for the WS requester.

5. The QoS-based WS matchmaking and selection sub-processes conform to the requirements posed in the 4.2 and 4.3 sections of chapter 4.

6. The results returned from the QoS-based WS matchmaking sub-process must be at least of three types: *perfect* matches, *partial* matches and *failed* matches. Apart from the hand over of the results along with their characterization, this sub-process should also deliver to the QoS-based WS selection sub-process the QoS selection criteria of the WS requester. So we also assume that the QoS-based WS matchmaking process precedes the QoS-based WS selection process.

7. As we are not aware of a WS selection sub-process that does not rank WS offers according to QoS criteria, from now on the notions of WS selection and QoS-based WS selection are considered equivalent.

8. The result of the WS selection sub-process is a ranked list of QoS offers. Each rank must be unique based on the requester's QoS criteria unless there are identical QoS offers. In this case, the identical QoS offers must get the same rank.

9. It is the responsibility of the whole WS discovery process not only to control the data and workflow of its sub-processes but also to provide a complete, custom-tailored characterization of the results returned. In the context of this thesis, the whole WS discovery process was not going to be implemented but only the QoS-based WS matchmaking and selection sub-processes. For evaluation purposes, a program was implemented to control the data and program flow of the two implemented sub-processes. This program returned to the WS requester a list of the following format of results: [WS Advertisement URI, QoS offer URI, (perfect or partial QoS match?), QoS rank].

## 6.2 OWL-Q for QoS-based Web Service Description

Based on the design principles and requirements of QoS-based WS Description set in section 4.1 and on the assumptions and constraints of the previous section, we have developed an *OWL-S* extension (the requirement *syntactical separation* is satisfied as our ontology can be developed independently from OWL-S), named OWL-Q [KP06], for QoS-based WS description of both requests and offers. We have extended *OWL-S* ontological description for two reasons: to comply with Semantic WS description standards (*standards compliance*) and to use the *OWL* [BDH+03] ontology formalism (*extensible and formal semantic QoS model*). OWL is one of the most expressive ontology languages and it is a W3C recommendation. There have been developed various reasoning tools for OWL enabling the enforcement of various class and property constraints, subsumption reasoning, type-checking. Using these

tools the syntactic and semantic validity of QoS descriptions can be checked and the processes of QoS matchmaking and selection can be properly supported.

Apart from this upper-level ontology, which is analyzed in subsection 6.2.1, mid-level and low-level ontologies were also created in order to support the evaluation of our QoS-based WS alignment and discovery algorithms and to ease the adaption of OWL-Q. In addition, SWRL[1] rules were also developed for supporting the semantic type-checking and validity of QoS specifications and for inferencing new knowledge like the equivalence of QoS metrics. The reasons of why rules were also needed apart from OWL-Q are given in subsection 6.2.2 along with the main rules used (except from those inferencing the equivalence of QoS metrics that are analyzed in section 6.3).

### 6.2.1 OWL-Q

During the implementation of this thesis proposal, OWL-Q design was finalized [KP07d, KP07a]. In its new form, OWL-Q is carefully separated into several facets. Each facet can be developed and extended independently of the other (*syntactical separation* and *refinement of QoS specifications*). Each facet concentrates on a particular part of QoS-based WS description. A document describing a QoS WS advertisement or request should reference all the facets of our ontology.

In the sequel, we analyze all facets of the OWL-Q ontology commenting on eight figures-snapshots of this ontology. Each figure contains classes (circles) and properties (arrows). It does not include cardinality constraints and other type of OWL constraints. The reason for using this formalism is not to present very complicated figures and in order to show that our ontology can easily be expressed in another ontology language apart from OWL.

#### Connecting Facet

As can be seen in Fig. 6.1, the *Connecting* facet provides two points of connection of OWL-S with OWL-Q and provides some high-level QoS concepts. The first point of connection of the two ontological descriptions is that the *ServiceAttribute* class is a subclass of OWL-S *ServiceParameter* and references a *ServiceElement*. Subclasses of the latter class are *ConditionalOutput*, *Parameter*, *Input*, *Precondition*, *Effect*, and *Service*. That is a *ServiceAttribute* can reference any *ServiceElement* of a service's functional description (*fine-grained QoS specification*). Another point of connection is that the *Actor* class is separated into three subclasses: *Provider*, *Requester*, *ThirdParty* so as to define the main actors involved in QoS-based WS description and measurement. A Service Attribute contains two subclasses: *QoSAttribute* and *ContextAttribute* and is a subclass of the general class *Attribute*. QoS attributes can be static or dynamic and are measured by static or dynamic QoS metrics respectively. Moreover, QoS attributes belong to QoS groups which can be inside other QoS groups. For example, *Response Time* QoS attribute belongs to the QoS group of *Performance*. An attribute can be separated into **a)** physical or service attributes, **b)** measurable

[1] `http://www.w3.org/Submission/SWRL/`

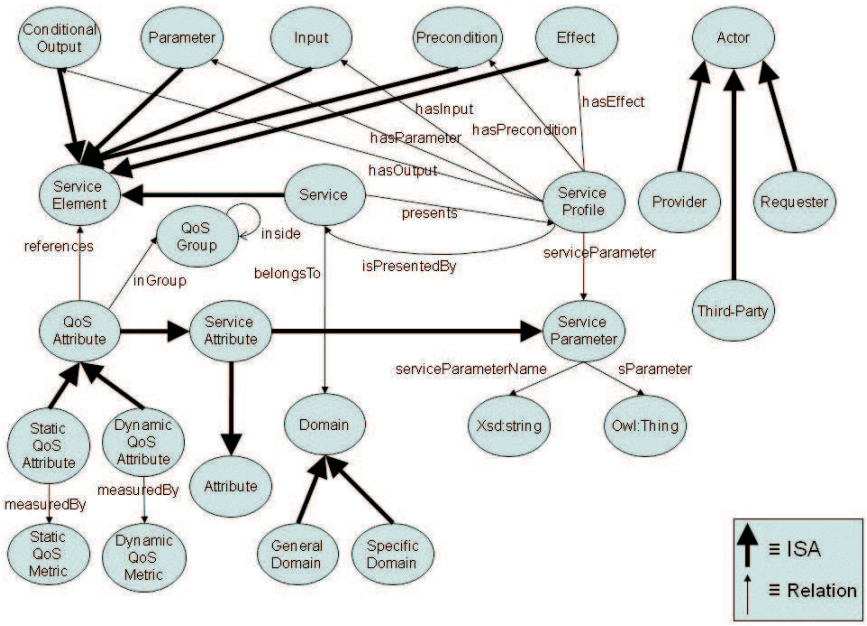



Figure 6.1: Connecting OWL-Q facet

or unmeasurable attributes and **c)** unique or derived attributes. Physical attributes like *Time*, *Temperature* and *Location* characterize environmental (contextual) factors of a WS or its requester while service attributes like *Availability* or *NumOfInterfaces* are functional or non-functional characteristics of a WS. Measurable attributes like *Time* are measured by specific metrics while unmeasurable attributes like *Manageability* cannot be measured. Unique attributes like *Time* are not derived by other attributes and are measured by resource metrics while derived attributes like *Throughput* are produced by complex metrics computed by functions using metrics of other attributes. We have not included all classes referenced here in the displayed figure in order not to create many images and distract user's attention from the main description and analysis. The *Domain* class represents the domain of knowledge that a service applies to and is separated into two subclasses: **a)** *GeneralDomain* and **b)** *SpecificDomain*. The *GeneralDomain* stands for every possible WS. Specific Domain can be further specialized/subsumed, for example a possible subclass could be the *Traffic Monitoring* domain.

### QoSSpec Facet

In this facet, that is shown in Figure 6.2, the classes representing QoS offers and requests are defined. Moreover, the final point of connection of OWL-S with OWL-Q is established by the fact that a *ServiceProfile* contains one or more *QoSOffers*'s or one QoS Request (*classes of service requirement*). The main class *QoSSpec* is separated into two subclasses: *QoSOffer* and *QoSDemand* in order to enable WS providers and requesters to define in the same way

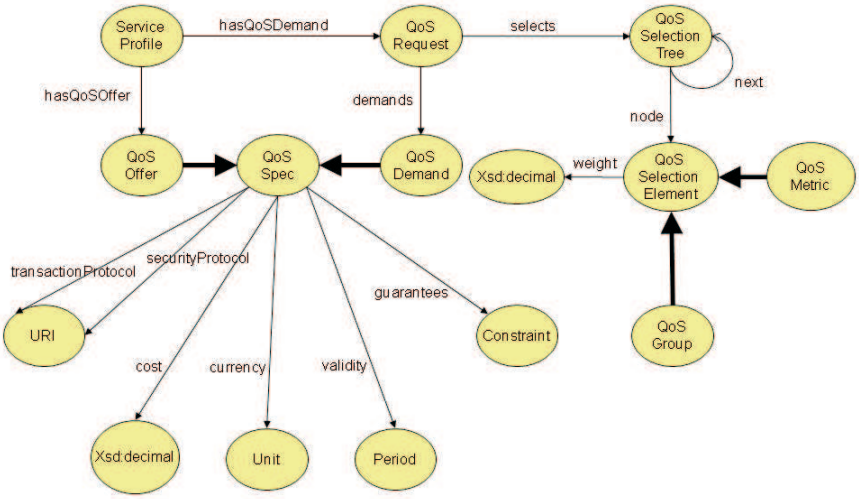



Figure 6.2: Basic OWL-Q facet

their QoS constraints *(both provider and requester QoS specification)*. Of course, the WS requester is enabled not only to specify constraints (by the *QoSDemand* class) but also to provide weights to QoS metrics and groups of his interest (by the fact that the *QoSRequest* class points to the class *QoSSelectionTree*). The *QoSSelectionTree* class is actually a tree of nodes called *QoSSelectionElement*s. Each node is either a QoS metric or a QoS group and has a *weight* label. The *QoSSpec* class represents the actual QoS description of a WS. It describes the security and transaction protocols used (URIs), the cost of using the service (double) and the associated currency for the cost (unit), the validity period of the offer or demand (*Period* class defined in a following facet) and a QoS constraint (which can be complex or simple) that must be guaranteed.

### QoS Metric Facet

The *QoS Metric* Facet (shown in Figure 6.3) describes all the appropriate classes and properties used for a proper formal definition of a QoS metric (*QoS metric model*). This metric facet is actually an upper ontology representing any abstract QoS metric. A specific QoS metric can be created by instantiating the *QoSMetric* class. Many specific QoS metrics (especially the ones that are domain-independent) can be part of a mid-level ontology created for QoS metric reuse. We have developed a mid-level ontology defining cross-domain QoS metrics and a low-level ontology for defining QoS metrics for the particular domain of *Traffic Monitoring*.

The *QoSMetric* is one of the most important classes of OWL-Q representing a QoS met-

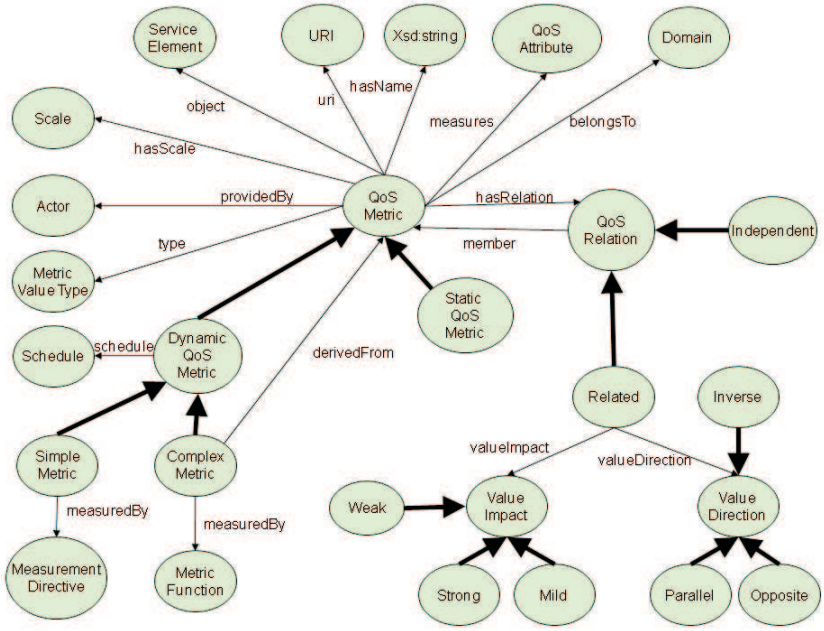



Figure 6.3: QoS Metric facet

ric. The values of a QoS metric are provided by an *Actor*. A QoS metric belongs to a *Domain* of knowledge. It has only one name. It measures a $QoSAttribute \cup MeasurableAttribute$ (in the figure we have omitted that the attribute is measurable due to space limitation) on a specific *ServiceElement*. The value type of a *QoSMetric* is an instance of the *MetricValueType* class (analyzed in a following facet) while the scale of the value is an instance of the *Scale* class. A *QoSMetric* is separated into static and dynamic metrics. A *StaticQoSMetric* is computed only once according to a *Trigger* in order to produce a value for a *StaticQoSAttribute*. A *DynamicQoSMetric* is computed repeatedly according to a *Schedule* to produce values of a *DynamicQoSAttribute* that change over time. It can be a simple QoS metric *measuredBy* a *MeasurementDirective* or a complex one. *ComplexMetrics* are derived from other metrics with the help of a *MetricFunction*. Last but not least a *QoSMetric* is related to other metrics according to two types of *Relationships*: *Independent* and *Related*. When two metrics are related, we can specify the direction of their values or the impact of one's value to the other's value. According to the scale it uses, a metric can be categorized into absolute, interval, nominal, ordinal and ratio metrics (this information was also omitted in the figure due to space limitation).

### Scale Facet

A measurement scale – main class of the *Scale* facet shown in Figure 6.4 – controls the value type and the type of operations allowed for a metric and belongs to a specific *Attribute*. It also specifies indirectly the way one value expression bound to one scale can be transformed

to another value expression of another compatible scale (both scales belonging to the same metric). So specific scales can be compatible if they belong to the same scale type and there is a *ScaleTransformationFunction* that transforms their expressions into each other. *Scale* is a more general notion with respect to *Unit*. A scale can be categorized into five disjoint subclasses: *NominalScale*, *OrdinalScale*, *IntervalScale*, *RatioScale* and *AbsoluteScale* [Fen96]. *Nominal* scales concern metrics that have as value type a set of numbers or strings. The members of this set cannot be compared (no ordering). Specific nominal scales can be compatible if there is a *one-to-one mapping function* between their corresponding value types. *Ordinal* scales apply to metrics that have an ordered set as value type. Metrics belonging to different ordinal scales cannot be added, multiplied, divided or abstracted in QoS constraints. We can transform one ordinal scale expression into another one with the help of *monotonic functions*. *Interval* scales preserve not only ordering but also differences. However, they do not preserve ratios. The operations of addition and substraction are allowed between different ordinal metrics. We can transform one interval scale expression into another one with the help of *affine transformation functions* of the form: $M = a*M^{'}+b$. *Ratio* scales preserve ordering, size of intervals and ratios. In a ratio scale there is always a zero element representing the total lack of the measured attribute. All arithmetics are allowed between different ratio metrics. We can transform one ratio scale expression into another one with the help of *mapping functions* of the form: $M = a * M^{'}$. Finally, the following facts are true for an absolute scale: **a)** measurement is made simply by counting the number of elements in the measurement set; **b)** measured attribute takes the form: "num of occurrences of $x$ in the entity"; **c)** all arithmetic analysis is meaningful; d) the set of acceptable transformations between different absolute scale expressions is the *identity transformation function*.

### Unit Facet

The *Unit* Facet (shown in Figure 6.5 formally describes the unit of a ratio scale of a ratio QoS metric. A *Unit* has one name, several abbreviations and synonyms (even in different languages). A *Unit* belongs to a *System of Units*, which system can be *SelfConsistent* or *NonSelfConsistent*, and is associated with a *MeasurableAttribute*. In case this latter attribute is actually a *QoSAttribute*, then it is measured by the ratio QoS metric that indirectly references this unit (this information is missing from the figure). A *Unit* is separated into *BasicUnit*s and *MultipleUnit*s. The *BasicUnit* class is separated into *UniqueAttributeUnit*s and *DerivedAttributeUnit*s, depending on the type of *Attribute* measured. A *MultipleUnit* is associated with a *BaseUnit* and converted to it by a constant (*magnitude*). It has a name composed of the name of its *BaseUnit* and a prefix. A *DerivedUnit* is proportional to some *Unit*s and inverse proportional to other *Unit*s. It also has a *magnitude* that is used to express its mathematical definition in relation to the other (inverse) proportional units. An unit is equivalent to another unit and can be converted to it with the help of their ratio scale and its ratio transformation functions. In addition, we have defined a *Unit Equivalence* class capturing those unit pairs that contain equivalent units and we have provided a property formula pointing to a XML-based string that explains in a mathematical

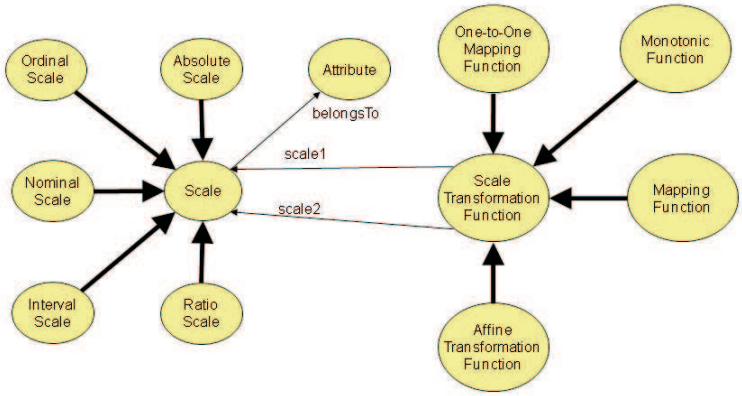



Figure 6.4: Scale facet

language like OpenMath[2] how the conversion should be performed mathematically.

### Metric Value Type Facet

The *MetricValueType* facet – shown in Figure 6.6 describes the types of values a QoS metric can take. The *MetricValueTypes* can be *Scalar* or *List-Based* types. *Scalar* value types are simple value types that can be *NumericScalar* or *String*. *Unconstrained* value types actually represent the infinite integer set or the decimal set $(-\infty, +\infty)$. *Constrained* value types represent *NumericScalar* value types that have (upper or low or both upper and low) limits. Constrained value types are separated into *SemiConstrained* and *TotallyConstrained* value types. *SemiConstrained* value types represent integer or decimal data types that are constrained in one of their limits. For example, the integer set $[0, \infty)$ is an instance of this type. *TotallyConstrained* value types represent integer or decimal data types that are constrained in both of their limits. For example, the decimal set [0.0, 1.0] is an instance of this type. The *List-Based* class represents list value types that have a specific size and whose elements are of a specific value type. Subclasses of the *List-Based* class are: numeric or string lists, string double-lists and timeseries. Numeric or string lists represent value types for nominal or ordinal scales. String double-lists represent string lists that can be mapped to decimal or integer lists again both for nominal or ordinal scales. For example, {[“one”, “two”, “three”], [1, 2, 3]} is an instance of a String double list. A *TimeSeries*

[2] `www.openmath.org`

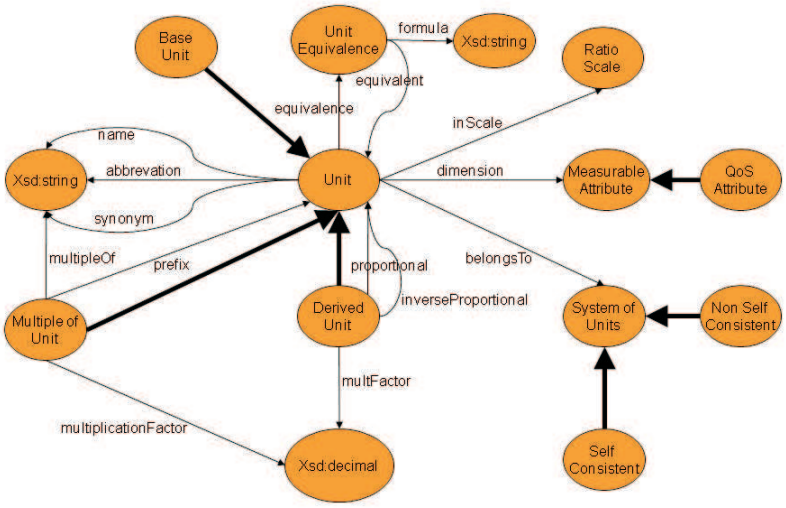



Figure 6.5: Unit facet

object is a set of *TSElement*s representing measurements performed on a specific time point and producing a specific value. Finally, the *MetricValueType* class has two subclasses (not shown in the figure) indicating the direction of values for the QoS metric that owns one of these two subtypes: *PositivelyMonotonic* value types have a direction of values from the lowest to the highest value where the highest value is mapped to the highest quality level that can be achieved while a *NegativelyMonotonic* value type has an opposite direction of values where the highest quality level is mapped to the lowest value. As an example, a QoS metric measuring the *Availability* QoS attribute like the one presented in chapter 3 has as value type a positively monotonic value type of *CDouble* (totally constrained double) [0.0, 1.0]. Due to the complexity of the figure we have not included the latest subclasses of value types.

### Function, Measurement Directive and Schedule facets

These three facets are shown in Figure 6.7. Functions in OWL-Q are separated into functions applied to metrics and functions applied to scales. Metric functions have specific name, contain a list of input arguments that are either *QoS Metric*s, *Decimal* values or other Metric Functions and produce a single argument (QoS metric or decimal) as an output. Metric functions are further categorized into *Metric Construction Function*s and *Metric To Numeric Function*s. The former subclass contains those functions that construct *TimeSeries* (TS) metrics like *TSContructor* (produces a series of time-based measurements for a resource metric) while the latter contains metric to numeric functions like *Size* (produces the size of

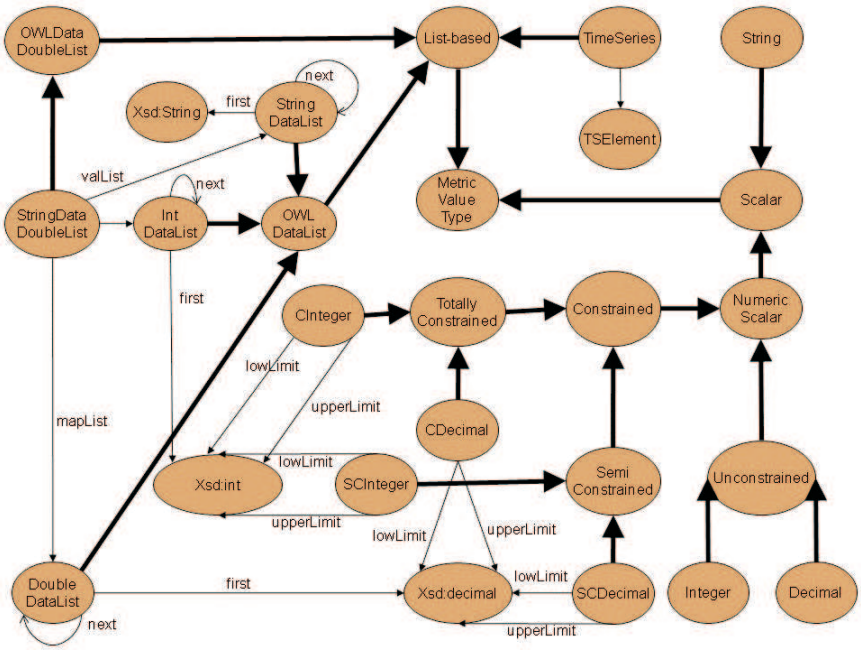



Figure 6.6: Metric Value Type facet

the TS) that are further separated into statistical ones like *Mean* (produces the mean of the values of the TS). Scale transformation functions are further categorized into five disjoint subclasses and are used for converting one scale expression to an expression of another scale of the same type.

The *MeasurementDirective* class specifies the way simple metrics are measured. It specifies by a URI how the value of a managed resource is going to be retrieved and by a *ValueType* the type of the return value. In addition, it specifies if the party responsible for the measurement will ask for the value or get it when it is ready (i.e it specifies the access model, where $AccessModel = Pull \cup Push$). This class can have many subclasses, some of which may require a possible extra attribute (*timeOut*) specification concerning the time duration (of type *interval* specified in next paragraph) that the measurement party will wait for in order to get the measurement value (consider for example the *Status* measurement directive [KL03]).

A *Schedule* is used to compute the frequency of the computation of a *Complex Metric*'s value. It has a specific name and is defined either by a starting and ending *Period* of xsd:dateTime type or by a time *Interval* that is expressed in specific time units.

### Constraint Facet

This facet that is shown in Figure 6.8 represents the actual constraint specification of QoS-based WS specifications. A *Constraint* can be simple or complex. *Simple Constraints* compare two arguments (that – as analyzed in the previous subsubsection – can be metric

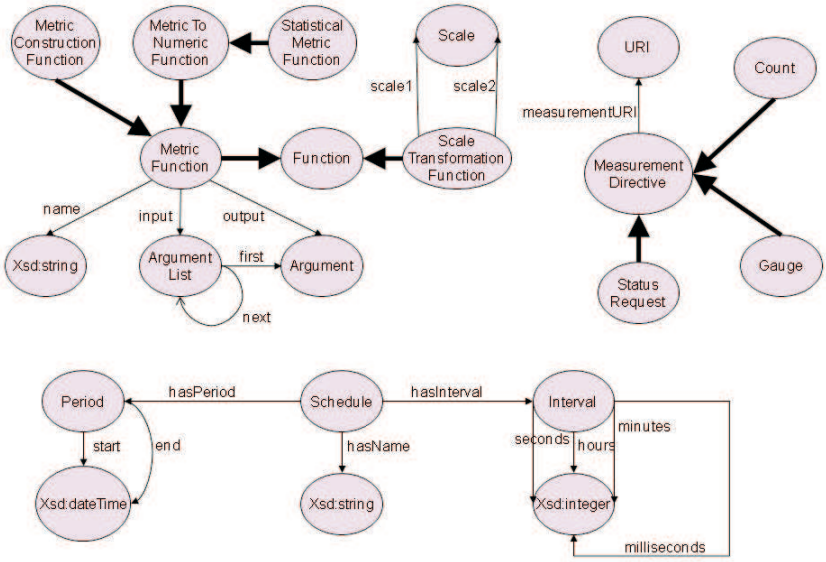



Figure 6.7: Function, Measurement Directive and Schedule facets

functions, QoS metrics or Decimal values) with the help of a binary comparison predicate like $\leq$. On the other hand, complex constraints are specified with the enforcement of a constraint predicate (like "and", "or", "not" or "if A then B") to a list of constraints. In this way, a conjunction of simple constraints can be represented by a complex constraint using the "and" constraint predicate and referencing the constraint list containing these simple constraints. Finally, the *Predicate* class is used to represent all possible predicates used in constraint specification [KL03]. Its subclasses are unary, binary or n-ary predicates.

### 6.2.2 Rules

The most significant change adapted in OWL-Q is the incorporation of rules. It is well-known at the Semantic Web community that OWL supports very well reasoning about concepts but not about properties. For example, there is no way we can specify that a fact $p(x, y)$ can be true, where $x, y$ are instances, if other property or instance facts are true. As another example, there is no way to specify that two or more property or class instance facts (or a mixture of them) cannot be both part of the semantic database. However, it was imperative in OWL-Q to reason about properties with rules because: **a)** relations between temporal properties like duration [HP04] had to be expressed and reasoned about; **b)** operations or comparisons on metrics had to be restricted according to the scale that they use; **c)** integrity constraints between property facts and/or instance facts had to be enforced (e.g. in lists we have prevented cycles with the help of rules); **d)** compatibility or equivalency of scales and compatibility of metrics' value types were expressed by OWL property facts fired by rules;

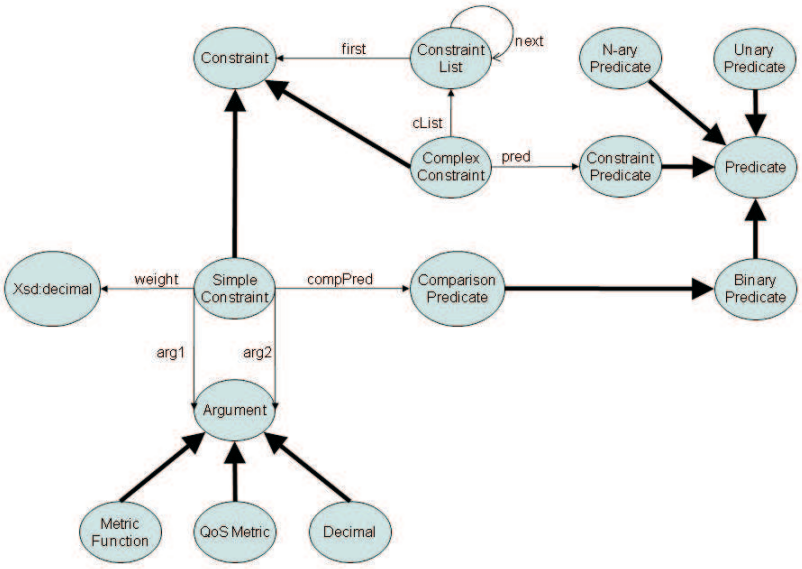



Figure 6.8: Constraint facet

**e)** rule-based algorithms like the metric matching one (see next section) were specified with rules. So we have extended OWL-Q with rules, which are expressed in SWRL – the most widely used SW rules proposal at present.

### 6.2.3 Achievements

OWL-Q is among the most rich semantic QoS-based WS languages. In the sequel, we are going to analyze why we have selected OWL-S instead of WSMO (or any other SWS language) and how OWL-Q satisfies all the requirements set in chapter 4 in the best possible way.

OWL-Q extends OWL-S and adopts OWL for semantically describing QoS-based WS specifications. The extension of OWL-S is performed in a way that OWL-Q can be syntactically separated from OWL-S. In this way, OWL-Q achieves *syntactical separation* and *adoption of world-wide standards*. Of course, besides OWL-S there are other SWS languages from which the most competitive one is WSMO. Let us now compare OWL-S and WSMO. In summary, OWL-S mainly uses the agent planning approach and models WSs as processes, whereas WSMO is mainly based on problem solving techniques and models WSs as state machines. OWL-S is better integrated with the existing Web standards and has defined a specific process model and grounding mechanism. WSMO provides some relatively mature execution environments; the explicit definitions of choreography and orchestration enable the better service reuse and composition; also PSM appears to be a more nature choice of formalisms to describe WSs. Lastly, OWL-S is just a service specification ontology, whereas

WSMO is a complete SWS framework [GPSH08].

Based on the above comparison, we cannot conclude a clear winner of the two important SWS alternatives. OWL-S and WSMO currently each deliver parts of what users need for effective Web services representation and fail to deliver other parts [GPSH08]. There are two reasons why we have selected OWL-S from WSMO. The first one is that at the time we started our thesis, OWL-S had a better tool support than WSMO. The second reason concerns ontology reasoning. OWL-S is tightly coupled with OWL, while WSMO defined as a conceptual framework aims to support a family of Ontology specification languages, which include Description Logics, First-Order Logic and Logic Programming based ontology language variants. Despite of this, the syntax of WSMO ontology language (WSML) is based on F-logic. Furthermore, as far as we know, all most all the WSMO use cases and execution environments have adopted F-Logic based WSML variant as the ontology languages. So let us compare F-Logic with OWL. While most users find F-logic systems more intuitive and easier to use, at least initially and for simpler applications, for large ontologies with multiple inheritances, composition of entities allows ontologies implemented in OWL to be regular and parsimonious. Moreover, for large ontologies written in frames and related formalisms, the number of enumerated entities tends to explode exponentially. In OWL ontologies many operations can be performed automatically by a reasoner; whereas ontologies implemented in frames must be maintained largely manually. If reasoning is required, OWL provides built in inference (e.g. consistency checking, instantiation reasoning, etc.) through the reasoner, whereas frames use a variety of external query and constraint languages. Where required for use in software, OWLŏs more rigid semantics and global consistency checking provide more support than is possible using the local reasoning typical of frames and their associated query and constraint languages. Finally, OWL is a W3C standard which may well promote its acceptation in the future. So based on the above reasons, we prefer OWL to F-Logic and choose OWL-S as it is tightly coupled with OWL. Of course, WSMO also defines some mappings from the F-logic to OWL (DL, Lite and Full), but most of these mappings are only at the syntax level. To end this discussion, we have to note that OWL-Q is specified in such a way that it can be easily connected to WSMO and we plan to make this connection in the near future.

Let us now concentrate on how well the requirements set in chapter 4 are satisfied by OWL-Q. OWL-Q offers a *semantic, rich and extensible QoS model*. The QoS model is semantic as OWL is adopted. It is rich as there are many facets developed each capturing an aspect of QoS-based WS description in great detail. Last but not least, it is extensible as all these facets can be refined and specialized according to the requirements of the users. The latter advantage is mainly based on the fact that ontologies can be structured and developed independently of the others and can be connected via the *namespace* construct. In this way, refinement of QoS specifications is easily allowed and realized with ontologies (refinement of QoS specifications).

Another advantage of OWL-Q is that its semantic QoS model contains a *rich and semantic QoS attributes and metrics model*. The richness of this model is undoubted as there are many aspects supported for both of these entities and in great detail. Especially, the *QoS Metric*, *Scale*, *Unit*, *MetricValueType* and *Function* facets are the most rich and detailed

facets with respect to all other research approaches.

Concerning specification of constraints, OWL-Q supports a quite rich, extensible and expressive constraints model. Simple constraints are defined by the expression: $[arg_1$ $binaryPredicate$ $arg_2]$ where $arg_1$ and $arg_2$ are QoS metrics, metric functions or decimal values and $binaryPredicate$ is a binary comparison predicate between these two subexpressions like $\leq, <, \geq, >, ! =, ==$. If someone has studied the research approach of [FK98], it is easy to derive that our approach is equivalent to it with respect to the simple constraint specification. Moreover, metric functions can play the role of *aspects* in the same way they are defined in [FK98] because we have defined most of the TS construction and statistical functions specified in [KL03]. Complex constraints can be constructed by simpler ones with the help of constraint predicates. We have defined many *unary*, *binary* and *n-ary* constraint predicates. An example of a unary constraint predicate is the "not" predicate, examples of binary constraint predicates are the "$\Leftrightarrow$" and "$\Rightarrow$" predicates and examples of n-ary constraint predicates are the "and" and "or" predicates.

Finally, OWL-Q can connect many *QoSProfiles* to a single *ServiceProfile* satisfying the requirement of *class of service specification*. Moreover, as QoS Metrics can be defined to measure QoS properties of different *Service Elements*, OWL-Q also satisfies the requirement of *fine-grained QoS specification*. Table 6.1 summarizes the achievements of OWL-Q. If this table is compared to the tables 5.1 and 5.2 of chapter 5, it is easy to see that OWL-Q not only outstands all research approaches in QoS-based WS description but also satisfies all the requirements set in chapter 4.

## 6.3 Semantic Alignment of QoS-based Web Service Specifications

UDDI-based registries have failed as they use a structural WS description model leading to accuracy problems in WS discovery. Semantic WS registries have been introduced for this reason, providing semantic WS publication and discovery functionalities. For functional WS discovery, the latter type of registries is enough. However, this is not the case for QoS-based WS discovery.

QoS is dynamic in nature so QoS offers will rapidly change depending mostly on their application domain. Lease mechanisms must be used for removing or updating these QoS offers. In addition, as QoS offers are associated to one WS functional specification, removal of the latter must cause the removal of all its QoS offers. Moreover, QoS offers will not always specify constraints on any general or domain-specific QoS metric. For this reason, measurements should be performed and stored. Their statistical processing will produce constraints on missing QoS metrics. Last but not least, all QoS-based WS discovery algorithms require that each QoS offer should use at least all QoS metrics of the QoS demand. Those QoS offers that do not satisfy this requirement are rejected from the discovery result list. This problem can be solved by aligning all candidate QoS offers with the QoS demand.

The subject of this PhD thesis was to solve the last two problems by introducing a com-

| Research Approach | Extensible & Formal QoS Model | Synt/cal Sep/tion | Both Client & Service Spec | Ref/ment of QoS Specs | Fine-Grained QoS Spec | Symmetric Model of QoS Spec | QoS Attr/s Model | Constraint Def/tion & Expr/ness | Classes of Service Spec |
|---|---|---|---|---|---|---|---|---|---|
| OWL-Q | yes | yes | yes | yes | yes | yes | excellent | excellent | yes |

Table 6.1: OWL-Q achievements in QoS-based WS description

plete alignment process for QoS-based WS specifications that uses OWL-Q, our QoS metric matching algorithm [KP06, KP07d] and stored measurements on WSs. This alignment process is separated into two sub-processes: *global alignment* and *local alignment*. Both of these sub-processes will be analyzed in detail in the two following subsections. But before this analysis, let us explain how this alignment process is exploited in our under-implementation QoS-based WS discovery engine.

Our QoS-based WS discovery engine [KP07d] is actually a WS registry offering its functionalities as a WS. When any QoS-based WS offer or demand is issued to our engine, the *global alignment* process is executed on it in order to align it with all specifications already processed and stored. Then this aligned specification is transformed into a CSP/MIP so as to check its consistency i.e. if it has any solutions at all. If not, then it is discarded and the issuer is informed to change it. Otherwise it is stored in our engine. If the issued specification was actually a QoS demand, then there will be some QoS offers related to it based on the results of a functional WS matchmaker. In this case, the *local alignment* process is executed on these QoS offers and demand so as to ensure that all QoS offers use at least the same set of QoS metrics with respect to the QoS demand. Then all specifications processed by this latter process are transformed again to CSPs/MIPS and our QoS-based WS discovery algorithm/process is finally executed on these CSPs so as to produce the appropriate results back to the issuer.

### 6.3.1 Global Alignment

The *global alignment* process is executed when a QoS-based WS specification is issued to our QoS-based WS discovery engine. Its goal is to align it with all previous specifications processed by our engine by exploiting our QoS metric matching algorithm. This process relies on the concept of the Metric Store (MS). MS is actually a storage space (inside our engine) of all QoS metrics encountered so far. So when this new specification arrives, we do not need to examine if any of its metrics matches with any metric of all previous specifications stored but with any metric in the MS. In this way, there is a minimization of all possible metric-to-metric comparisons and a common terminology across all processed specifications is enforced. In the end, the matched QoS metrics of this new specification will be replaced with the corresponding MS metrics.

Imagine MS is a forest of *derivation trees*. Each *derivation tree* contains QoS metrics as nodes that measure the same QoS property on the same WS element/construct. The relationship between a parent node and its child is that the parent QoS metric is produced by the child QoS metric with the help of a function. So at the leaves of each tree will be resource metrics that are only produced from measurement directives while all other nodes represent composite metrics. The level of each node equals the maximum level of its children plus one. For example, Figure 6.9 shows an *AverageExecutionTime* metric measuring the QoS property of *ExecutionTime* on a operation of a WS and having its level equal to three. This metric is produced by a *TimeSeriesExecutionTime* metric of level two. The latter metric is produced by a *RawExecutionTime* resource metric of level one.

Suppose a new QoS-based WS specification $Q$ is given as input to the global alignment

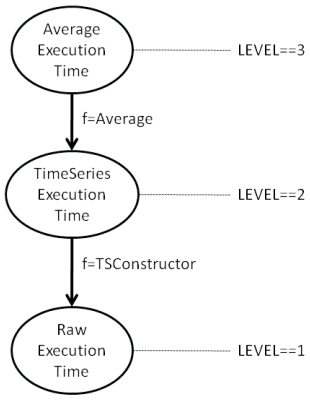



Figure 6.9: Metrics measuring *ExecutionTime* QoS property of a specific WS operation with different levels

process. Before analyzing the algorithmic steps of execution, let us explain how OWL-Q and the QoS metric matching algorithm equip our process. OWL-Q contains a property named *match* from the $QoSMetric$ to the same class. If a metric $M$ is contained in MS, then there will be a property fact $match(M, M)$ indicating this. If metric $M_2$ matches with a MS metric $M_1$, then there will also be a property fact $match(M_1, M_2)$ indicating this. Thus, the domain and not the range of the *match* property indicates the metrics that are contained in MS. The QoS metric matching algorithm compares two QoS metrics $M_1$ and $M_2$. If these metrics match, then it adds the property fact $match(M_1, M_2)$. More details about this algorithm will be provided later on this subsection.

A sketch of the algorithm of the global alignment process with input $Q$ is now analyzed while it is visually explained in Figure 6.10. We start the analysis of the algorithm by providing its pseudo-code, then we move on to explaining what it does and finally we supply a small example of its usage.

### Pseudo-code

```
Metric[] M = findUniqueMetrics(Q);

sort(M,'level');

for (i=1; i<=M.length; i++){

  if (match(M[i],M[i])) break;
```

```
  Metric[] Mi = getAncestorMetrics(M[i],M);
  sort(Mi,'level');
  Tree Tk = getMSMetrics(M[i].measures,M[i].object);
  Metric[] MS = Tk.getMetrics();
  sort(MS,'level');

  for (k=1; k<=Mi.length; k++){
    if (!match(Mi[k],Mi[k])){
      matched=false;
      for(j=Mi[k].level-1; j<=Mi[k].level+1; j++){
do{
  MS[l]=findInLevelNextMetric(MS,j);
  if (MS[l] != null && match(MS[l],Mi[k])){
    matched=true;
    updateFormula(M[i].derivedFrom,Mi[k],MS[l]);
    break;
  }
}
while(MS[l] != null);
if matched break;
      }
      if !matched infer(match(Mi[k],Mi[k]));
    }
  }

  matched=false;
```

```
  for(j=M[i].level-1; j<=M[i].level+1; j++){
    do{
      MS[l] = findInLevelNextMetric(MS,j);
      if (MS[l] != null && match(MS[l],M[i])){
matched=true;
updateSpec(Q,M[i],MS[l]);
break;
      }
    }
    while (MS[l] != null);
    if matched break;
  }
  if !matched infer(match(M[i],M[i]));
}
```

**Explanation** Let us explain the symbols that appear in the pseudo-code: $M$ is a metric array, $findUniqueMetrics$ is a function returning all metrics that are contained in a QoS specification $Q$, *sort* is a function that sorts a metric array by the level of its metrics (from smallest to biggest), *match* is a function that calls the metric matching algorithm on two metrics, $getAncestorMetrics$ is a function that returns the ancestor metrics of a metric excluding the ones that are contained in a metric array given as a second input, $getMSMetrics$ is a function returning a derivation MS metrics tree where each metric measures a specific QoS property on a specific WS element, $getMetrics$ is a function that returns a metric array from a derivation tree, $findInLevelNextMetric$ is a function that finds the next metric of a specific level of a metric array, $updateFormula$ is a function that updates a derivation formula of a metric by substituting one metric with another one, $infer$ adds a fact in the semantic database, and *updateSpec* is a function that updates a QoS specification by replacing one metric with another one.

Now we are going to explain what the algorithm does. For each QoS metric $M_i$ of $Q$ of level $y$ that appears in the constraints of $Q$, from the lowest level until the upmost, do the following: Find if $M_i$ is already in MS by asking if the property fact $match(M_i, M_i)$ is present. If yes, then go to the next metric of $Q$. Otherwise, find the corresponding derivation tree $Tk$ in MS containing metrics $MS_l$ measuring the same QoS property on the same WS element as $M_i$. For each ancestor metric $M_{ik}$ of $M_i$ (which is not already present in $Q$'s constraints), from the lowest level to the upmost, that has level $x$ find if it is in MS by

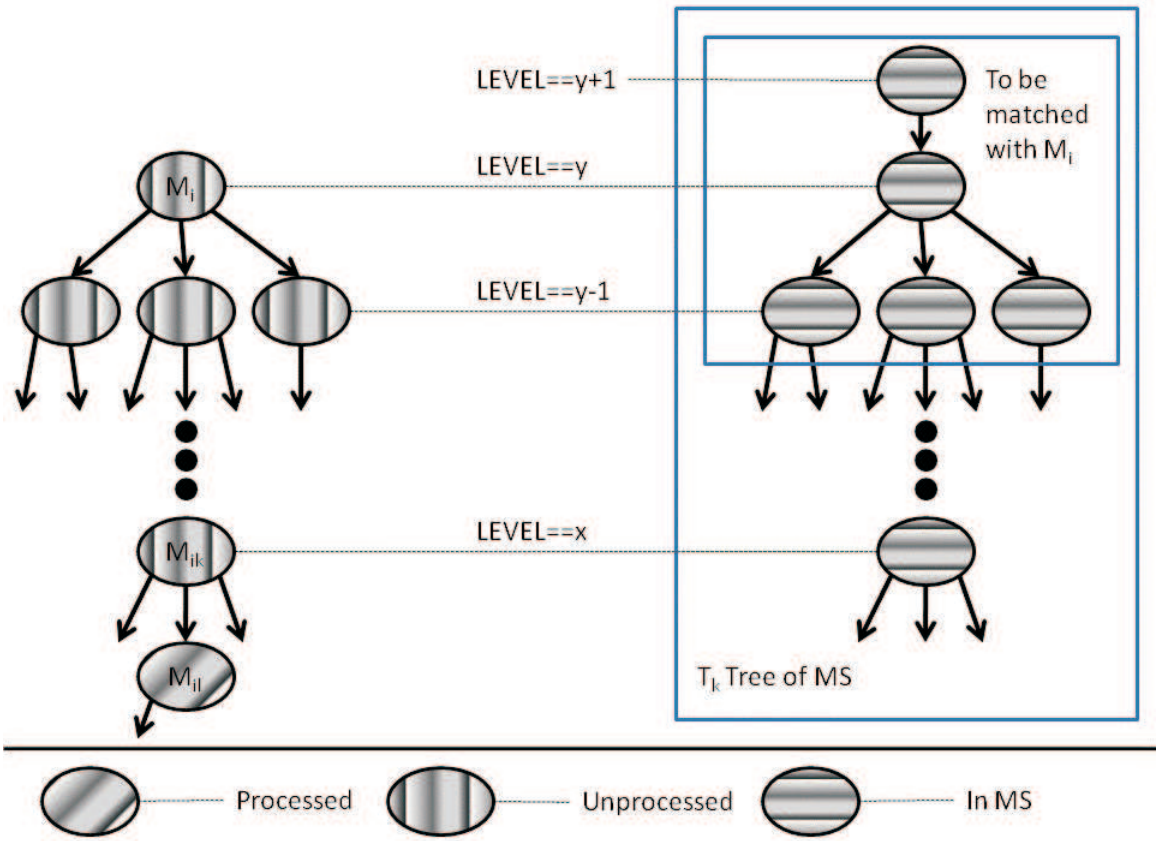



Figure 6.10: Sketch of the Global Alignment process

asking if the property fact $match(M_{ik}, M_{ik})$ is present. If yes go to the next ancestor metric of $M_i$. Otherwise, compare $M_{ik}$ with those metrics $MS_l$ in $T_k$ that have levels $x-1$, $x$ and $x+1$ with the QoS metric matching algorithm. If a match is found between $M_{ik}$ and one $MS_l$, stop the search for this metric $M_{ik}$. If no match is found at all, put $M_{ik}$ in MS by adding the property fact $match(M_{ik}, M_{ik})$. After all ancestor metrics have been processed, compare $M_i$ with those metrics $MS_l$ in $T_k$ that have levels $y-1$, $y$ and $y+1$ with the QoS metric matching algorithm. If no match is found at all, put $M_i$ in MS by adding the property fact $match(M_i, M_i)$ and go to the next metric of $Q$. Otherwise, if a match is found between $M_i$ and one $MS_l$ stop the search for $M_i$ and make two alignment actions in $Q$: a) change all references of $M_i$ to $MS_l$; b) perform a scale-to-scale transformation function from the scale of $MS_l$ to the scale of $M_i$ on all variables previously bound to $M_i$ that appear in $Q$'s constraints.

**Example** For example, suppose that MS is initially empty and that a specification $S_1$ containing constraints on the previously referenced *AverageExecutionTime* metric is issued. The aforementioned alignment algorithm would first add the resource metric *RawExecu-*

*tionTime* to the MS, then the composite metric *TimeSeriesExecutionTime* and finally the *AverageExecutionTime* metric. So $S_1$ is not actually affected. Further, suppose that the *AverageExecutionTime* QoS metric has as scale *MinutesDuration* and value type the real set $[0.0, 2.0]$. Finally, suppose that a new specification $S_2$ is issued that contains the constraint $X >= 100$, where $X$ is bound to a new metric *AvgExecTime* that has as scale *Seconds-Duration* and value type the integer set $[1, 120]$. This new metric is also derived from the *TimeSeriesExecutionTime* metric. When the alignment algorithm executes on $S_2$, it will find out that the ancestor metrics of *AvgExecTime* are already contained in MS. Then, it will discover that there is a match between *AverageExecutionTime* and *AvgExecTime*. So it will rebind $X$ to the *AverageExecutionTime* metric and will transform the $S_2$'s constraint to the constraint $60 \cdot X >= 100$.

### QoS Metric Matching Algorithm

It must be stressed out that inferring the semantic equivalence of QoS metrics is essential for three main reasons: a) People tend to have different perception of the same concept; b) Different types of system readings can be available for the same QoS metric; c) two mathematical expressions for producing the values of QoS metrics can be equivalent and thus these QoS metrics should be equivalent. For example, equivalent response time metrics could be associated to different scales and to different value types. This could happen not only in different but also in the same QoS specification if this specification is edited by different people. As another example, a *DownTime* metric can be either obtained in the form of high-level reading from a system with advanced instrumentation or can be derived from a resource metric of a system's *Status* obtained from low-level reading of systems with basic instrumentation. As a final example, suppose that one QoS metric $A_1$ is produced from metric $B_1$ and $B_2$ by the mathematical expression: $A_1 = \frac{B_1}{B_1+B_2}$ and another QoS metric $A_2$ is produced by the expression: $A_2 = 1 - \frac{B_2}{B_1+B_2}$. It is very easy to see that these two QoS metrics $A_1$ and $A_2$ should be equivalent as they are produced by the same QoS metrics used in two different but equivalent mathematical expressions.

Based on the above reasons, we have developed [KP06] and updated [KP07d] a semantic QoS metric matching algorithm that infers the equivalence of two QoS metric descriptions defined in OWL-Q. Of course, our algorithm is quite general and another ontology formalism can be used if there is an appropriate mapping of concepts. This algorithm is composed of three main rules, each corresponding to a different case in a two metrics comparison. The last rule – used in composite-to-composite metric matching – reaches the final point of either a) simplifying the difference of two derivation mathematical expressions by using mathematical engines like Matlab[3] in order to see if it equals zero or b) solving a CSP which contains the following constraint: $|expr_1 - expr_2| > threshold$, where $|x|$ is a function calculating the absolute value of $x$, $expr_1$ and $expr_2$ are the two derivation mathematical expression of the two composite QoS metrics respectively and $threshold$ is a user supplied limit ranging between $[10^{-1}, 10^{-12}]$. In the first case, symbolic computation techniques are used in order

[3] `www.mathworks.com`

to see if the two mathematical metric derivation expressions are equivalent. In the second case, the user does not care if the mathematical derivation expressions are equivalent but if they can produce very close values. So if the CSP is solved, this means that a pair of values from these two expressions was found that is very distant i.e. it does not satisfy the requirements of the user so the two composite metrics are different. Of course, both of these techniques can be used in parallel so that we can have the quickest possible solution, especially if the mathematical and CSP engines are hosted on a different computer.

It must be noted that based on the global alignment process and the way this process is performed, recursiveness is eliminated from the last two of three main rules. In addition, the second rule was reduced to the first one, so we now actually have two main rules. The first rule now compares resource-to-resource and resource-to-composite metrics. This rule just checks if the scale and value types of the two metrics are compatible. Compatibility of scales is easily inferred by a reasoner from the OWL-Q *compatible* property. Please note that it is assumed that *compatible* property facts are already feeded into OWL-Q specifications from domain modelers before metric matching. If the scales are compatible, then we transform one value type into the other one. Next, we check if the percentage of common values of the two types with respect to the size of the biggest value type is greater than a threshold provided by the user. If yes, then the match of the two metrics is inferred.

At the second rule, the first thing to be checked is scale and value type compatibility. Then matching of derivative metrics of the two compared QoS metrics is performed. This matching offered as a logical procedure, as the matches have already been computed by the code of the "global alignment" algorithm, will just check if the corresponding *match* facts have been produced. For each pair of matched derivative metrics, the corresponding derivation formulas of the two compared formulas are updated to include the same variable. More specifically, suppose that all possible matches between the metrics – from which our two compared metrics are derived – have been inferred. Then, from the derivation lists $M_1.derivedFrom$ and $M_2.derivedFrom$ of the compared metrics $M_1$ and $M_2$ and their corresponding measurement formulas $M_1.measuredBy$ and $M_2.measuredBy$, a CSP or simplification is produced by the following transformation:

1. $\forall M_i \in M_1.derivedFrom \notin M_2.derivedFrom$ map $M_i$ to CSP/symbolic variable $X_{i,1}$ and assign the value type of $M_i$ to the value domain of this variable (the latter only for the CSP)

2. $\forall M_j \in M_2.derivedFrom \notin M_1.derivedFrom$ map $M_j$ to CSP/symbolic variable $X_{j,2}$ and assign the value type of $M_j$ to the value domain of this variable (the latter only for the CSP)

3. $\forall M_k \in M_1.derivedFrom \cap M_2.derivedFrom$ map $M_k$ to CSP/symbolic variable $X_{k,12}$ and assign the value type of $M_k$ to the value domain of this variable (the latter only for the CSP)

4. Add the CSP constraint $|f(X_{i,1}, X_{k,12}) - g(X_{j,2}, X_{k,12})| > 10^{-8}$ to the CSP, where $f = M_1.measuredBy$, $g = M_2.measuredBy$, or ask the mathematical engine if $simplify(f(X_{i,1}, X_{k,12}) - g(X_{j,2}, X_{k,12})) == 0$.

Finally, when all variables of the two compared metrics are checked, the symbolic calculation or CSP solving technique is used on the derivation formulas in order to finally infer if the compared metrics are equivalent. In the sequel, the two main rules are going to be presented in a formal form and a small example of the application of the second rule will be provided (please notice that we use the CSP technique for the second rule).

$$match\,(M_1, M_2) \Leftarrow rrcm\,(M_1, M_2) \vee ccm\,(M_1, M_2) \tag{6.1}$$

$$sm\,(M_1, M_2) \Leftarrow svm\,(M_1.scale, M_2.scale, M_1.type, M_2.type) \\ \wedge M_1.object = M_2.object \wedge M_1.measures = M_2.measures \tag{6.2}$$

$$rrcm\,(M_1, M_2) \Leftarrow RM\,(M_1) \wedge RCM\,(M_2) \wedge sm\,(M_1, M_2) \tag{6.3}$$

$$ccm\,(M_1, M_2) \Leftarrow CM\,(M_1) \wedge CM\,(M_2) \wedge sm\,(M_1, M_2) \\ \wedge msm\,(M_1.derivedFrom, M_2.derivedFrom) \\ \wedge \neg solveCSP(M_1.derivedFrom, M_2.derivedFrom, \\ |M_1.measuredBy - M_2.measuredBy| > 10^{-8}) \tag{6.4}$$

Let us explain in detail the notation used: $M_1$ and $M_2$ are Metrics, $RM\,(M)$ and $CM\,(M)$ are rules inferring if metric $M$ is resource and composite respectively, $RCM\,(M)$ is a rule inferring if metric $M$ is resource or composite, $svm\,(M_1.scale, M_2.scale, M_1.type, M_2.type)$ is a rule that infers if the scales and value types of metrics $M_1$ and $M_2$ are compatible, $msm\,(M_1.derivedFrom, M_2.derivedFrom)$ is a logical procedure that matches one by one the $M_1$'s list of derivative metrics with the corresponding metrics list of $M_2$ and transforms the matching pairs in the same variables at the compared metrics derivation formulas, and $solveCSP\,(List_1, List_2, constraint)$ is a logic procedure that solves the CSP defined by the two first metric lists and the constraint given by third argument. When the latter procedure finds a solution, it returns true, otherwise it returns false.

**Example** Assume that a WS provider defines composite metric $Avail_1$ that measures the QoS Property of *Availability* of his WS and is derived from two Resource metrics $Downtime_1$ and $Uptime_1$ based on the formula: $1 - Donwtime_1/(Downtime_1 + Uptime_1)$. In addition, assume that a WS requester defines composite metric $Avail_2$ that also measures the QoS Property of *Availability* and is derived from two Composite metrics $Downtime_2$ and $Uptime_2$ based on the formula: $Uptime_2/(Uptime_2 + Downtime_2)$. Further assume that all metrics have as value type the interval $[0.0, 1.0]$ and that the two compared metrics use the same unit. Finally, assume that the following matches are true: $match(Downtime_1, Downtime_2)$, $match(Uptime_1, Uptime_2)$ and that all previous clauses of the third main rule are satisfied. We want to see if composite metrics $Avail_1$ and $Avail_2$ are matched based on the satisfiability of the last clause of the rule. Based on the aforementioned transformation procedure, a CSP is created and solved that has the following definitions: $D, U :: [0.0, 1.0]$ and constraints: $|1 - D/(D + U) - U/(U + D)| > 10^{-8}$. This CSP is unsatisfiable so the last clause is satisfiable and finally the fact $match\,(Avail_1, Avail_2)$ is inferred.

### 6.3.2 Local Alignment

It can be argued that the global alignment process is not actually needed if all QoS specifications use QoS metrics from the same ontology. This is in part true but only if there is one to one correspondence between the metric ontology developed by the domain expert and the application domain itself. To be more specific, this is the case where there is only one metric ontology for each application domain and another one for the domain independent QoS metrics. However, the necessity of the *local alignment* process is undebatable as its execution dramatically improves the accuracy of the QoS-based WS discovery algorithms.

The *local alignment* process takes as input a set $O$ of several QoS offers $O_i$ and one QoS demand $D$. Its goal is to align all QoS offers in such a way that the number of the demand's QoS metrics that are not referenced by any QoS offer is minimized. The alignment is made locally to the QoS offers (that's why we have given this name to this process). This means that the altered QoS offers are not stored back to the discovery engine but they are just returned as output from this process (and then as input to the matchmaking process).

The main idea of this process is the following. Suppose we want to align QoS offer $O_i$ with the QoS demand $D$. Further, suppose that $M$ is a list of all QoS metrics $M_j$ in $O_i$. Some of these metrics will be the same with some of the metrics of $D$. In addition, suppose that $N$ is a list of QoS metrics $N_k$ in $D$ that is disjoint with the list $M$. Our goal is to alter $O_i$ so as to add and express constraints on every metric $N_k$ in this spec with the help of some of the metrics in $M$. To the best of our knowledge, there are three strategies that we can adopt.

#### Mathematical Derivation Strategy

This strategy is actually decomposed into two alternative sub-strategies. The first sub-strategy is quite simple. Every QoS-based WS specification is expressed with constraints only on resource metrics. For example, suppose there is a composite metric $A$ that is derived from resource metric $B$ by the expression $f(x)$, where $x$ is bound to $B$. Then every specification containing $A$ will replace it with $B$ and every constraint of this specification referencing variables bound to $A$ will replace them with $f(x)$. It is easy to comprehend what happens to composite metrics of level greater than two. As another (more concrete example), suppose that a QoS-based specification has the constraint $AvgRespTime <= 10$, where $AvgRespTime$ is a QoS metric measuring the average response time of a WS's operation. Further, suppose that $AvgRespTime$ is computed by applying the $avg$ function on a metric $TSRespTime$ (Time series of response time values) which in turn is produced by applying the $tsConstruct$ function (builds a time series) on a resource metric $RespTime$ measuring the operation's response time. In this case and according to this sub-strategy, the QoS-based specification will replace all references to $AvgRespTime$ with the expression $avg(tsConstruct(RespTime))$ (that is with a different and resource metric) so the $AvgRespTime <= 10$ constraint will be replaced with the new one $avg(tsConstruct(RespTime)) <= 10$ in the processed specification.

The pseudo-code for this sub-strategy follows:

```
Metric[] M = findCompositeMetrics(Q);

for (i=1; i<=M.length; i++){

  Metric[] R = getResource(M[i]);

  String expr = exprWithResource(M[i].derived);

  updateSpec(Q,M[i],R,expr);

}
```

where $findCompositeMetrics$ is a function that returns the composite metrics that are contained in the constraints of a specification, $getResource$ is a function that returns the direct and indirect resource metrics from which a composite metric is derived from, $exprWithResource$ is a function that returns the expression with which a composite metric is derived from its direct and indirect resource metrics, and $updateSpec$ is a function that updates a QoS specification a) by removing a composite metric definition and substituting it with the definition of its direct and indirect resource metrics (if they are not already defined in the specification) and b) by replacing the reference of this metric in the constraints of the specification with the expression containing the derivation formula of the composite metric containing only references to its direct and indirect resource metrics.

Although this strategy is simple, quickly enforced and may produce specifications with lower number of QoS metrics, it may lead to non-linear constraints on metrics if non-linear derivation formulas for composite metrics are present. This is bad for the efficiency of the discovery engine as solving non-linear CSPs takes significatively more time with respect to linear CSPs (or even better to linear MIPs).

The second sub-strategy tries to find patterns in the derivation trees. It scans for the following three cases with decreasing order of preference:

1. A metric $M_i$ in $M$ has one or more $N_k$ ancestor metrics in its derivation tree. For example, suppose $M_i$ is produced from metrics $L_1$ and $N_k$ by the expression $f(x, y)$, where $x$ is bound to $L_1$ and $y$ to $N_k$. Spec $O_i$ will be altered as follows: If $M_i$ is a common metric of $O_i$ and $D$, then we just add the constraint $z = f(x, y)$, where $z$ is bound to $M_i$, $x$ to $L_1$ and $y$ to $N_k$. Otherwise, we update $O_i$'s constraints having variables bound to $M_i$ by replacing these variables with the expression $f(x, y)$. In all instances, $N_k$ is erased from $N$. The big question here is the status of the $L_1$ metric because we want to restrict the value domain of $N_k$ as it is already constrained in $D$. It would be nice if $L_1$ is already contained in the $M$ list. In this way, $O_i$ would contain constraints on $L_1$ and in combinations of $L_1$ and $N_k$, so $N_k$ would get further restricted. But if $L_1$ is not contained in $M$, then we either hope that it has a restrictive value type or we add it to $N_k$. The last option would surely increase the execution time of the local alignment process. As another more concrete example, suppose $M_i$ is $AvgServiceLatency$ and $N_k$ is $AvgExecutionTime$. In addition, suppose we have that: $M_i = N_k + L_1$, where $L_1$ is $AvgServiceQueueDelay$, from the definition of $M_i$. Finally, suppose that $M_i$ is not common between $O_i$ and $D$. In this case, we will

| Case | Specs | Before | After | Condition |
|---|---|---|---|---|
| Case 1 | | | | $N_k$ in ancestors of $M_i$ |
| 1.1 | $O_i$ | $M_i > 10$ | $M_i > 10, M_i = \mathrm{f}(N_k, L_1)$ | $M_i$ contained in $D$ (before) |
| | $D$ | $N_k > 5, M_i > 10$ | $N_k > 5, M_i > 10$ | $L_1$ may be added in $N$ (after) |
| 1.2 | $O_i$ | $M_i > 10$ | $\mathrm{f}(N_k, L_1) > 10$ | $M_i$ not contained in $D$ (before) |
| | $D$ | $N_k > 5$ | $N_k > 5$ | $L_1$ may be added in $N$ (after) |
| Case 2 | | | | $M_i$ and $N_k$ have common ancestor |
| | $O_i$ | $M_i > 10$ | $M_i > 10, M_i = \mathrm{f}(L_1)$ | When matchmaking |
| | $D$ | $N_k > 5$ | $N_k > 5, N_k = \mathrm{g}(L_1)$ | $O_i$ with $D$ must ignore $N_k$ |
| Case 3 | | | | $M_i$ is ancestor of $N_k$ |
| | $O_i$ | $M_i > 10$ | $M_i > 10, N_k = \mathrm{f}(M_i, L_1)$ | $L_1$ may be added in $N$ |
| | $D$ | $N_k > 5$ | $N_k > 5$ | |

Table 6.2: The three cases of the second sub-strategy of the *Mathematical Derivation* strategy

replace every occurrence of $M_i$ in the constraints of $O_i$ with $N_k + L_1$, we will remove $N_k$ from $N$ and we will hope that $L_1$ has a restrictive value type.

2. Metrics $M_i$ and $N_k$ have a common ancestor. For instance, recall an example from a previous subsection and suppose that $M_i$ is *AverageExecutionTime* produced by *TimeSeriesExecutionTime* with the expression $avg(x)$ and that $N_k$ is *MaxExecutionTime* produced from *TimeSeriesExecutionTime* with the expression $max(x)$, where $x$ is bound to *TimeSeriesExecutionTime*. Then we update both $O_i$ and $D$ specs by adding the constraints $M_i = avg(x)$ and $N_k = max(x)$, respectively. Notice that we alter $D$ by not removing a metric reference but by adding a new one. So we just increase the solution space of $D$. In addition, we comment on $O_i$ that when it is matched with $D$, then the $N_k$ metric must be ignored. Finally, we remove $N_k$ (*MaxExecutionTime*) from $N$.

3. $M_i$ is ancestor of $N_k$. For example, suppose $N_k$ is produced from $L_1$ and $M_i$ with the expression $f(x, y)$, where $x$ is bound to $L_1$ and $y$ to $M_i$. Then we update $O_i$ by adding the constraint $z = f(x, y)$, where $z$ is bound to $N_k$, $x$ to $L_1$ and $y$ to $M_i$. We also remove $N_k$ from $N$. The problem here is again the status of $L_1$. If $L_1$ is in $M$, then $O_i$ contains constraints on $L_1$ and $M_i$ so these constraints are propagated to the values of the added $N_k$ metric. But if $L_1$ is not contained in $M$, then we either hope that it has a restrictive value type or we add it in the $N$ list. As a more concrete example, suppose that $M_i$ is $AvgServiceLatency$, $N_k$ is $AvgRespTime$, and $N_k = M_i + L_1$, where $L_1$ is $AvgNetworkLatency$. In this case, we will add to $O_i$ the constraint $N_k = M_i + L_1$, we will remove $N_k$ from $N$ and we will add $L_1$ to $N$.

Table 6.2 summarizes the three cases of the second sub-strategy of the *Mathematical Derivation* strategy. This table firstly explains the main conditions for entering each case, then it shows how the QoS specs $O_i$ and $D$ are affected by displaying their content before

and after the application of each case and finally what are the other conditions that hold after the application of each case. If for each case there are also subcases, then we explicate for each subcase (and not for the whole case) how the specs are altered and what conditions hold before and after the subcase.

The pseudo-code for this sub-strategy follows:

```
Metric[] M = findMetrics(Oi);

Metric[] N = minus(findMetrics(D),M);


Time t1=now();

while (notEmpty(N) && now()-t1<=limit){

  Nk=N[1];

  for(i=1; i<=M.length; i++){

    if inAncestorsOf(Nk,M[i]){

      if containedIn(M[i],D) addIn(M[i]=f(Nk,L1),Oi);

      else replaceIn(M[i],f(Nk,L1),Oi);

      removeFrom(Nk,N);

      addTo(L1,N);

      break;

    }

    else if commonAncestors(Nk,M[i]){

      addIn(M[i]=f(L1),Oi);

      addIn(Nk=g(L1),D);

      removeFrom(Nk,N);

      addTo(L1,N);

      break;

    }

    else if inAncestorsOf(M[i],Nk){

      addIn(Nk=f(M[i],L1),Oi);
```

```
        removeFrom(Nk,N);
        addTo(L1,N);
        break;
      }
    }
}
```

where $findMetrics$ is a function that returns the metrics referenced by the constraints of a QoS specification, *minus* is a function that subtracts one metric array from another one, *now* is a function that returns the current timestamp, $notEmpty$ is a function that returns true if a metric array is not empty and false otherwise, $inAncestorsOf$ is a function that returns true if a metric is in the ancestors of another metric, $containedIn$ is a function that returns true if a metric is contained in a QoS specification, $addIn$ is a function that adds a constraint in a QoS specification, $replaceIn$ is a function that replaces all metric references with a specific expression in a QoS specification, $removeFrom$ is a function that removes a metric from an array, $addTo$ is a function that adds a metric to an array, and *commonAncestors* is a function that returns true if two metrics have a common ancestor or false otherwise.

The second sub-strategy is better than the first one as it adds none or fewer non-linear constraints. However, this advantage is annihilated by the fact that two out of three cases of this sub-strategy do not exactly achieve their goal as they remove one metric from $N$ and may add another one. So the execution time of this sub-strategy can be better but also can be even worse than that of the first sub-strategy. In addition, it must be stated that both sub-strategies fail if there is any metric $N_k$ in $N$ that has a derivation tree disjointed with all the derivation trees of the metrics in $M$. This problem can be solved by the next strategy.

### Statistical Strategy

If measurements on metrics for a WS are stored in a discovery engine, then the WS's QoS offer can be produced dynamically from the required QoS metrics of the QoS demand. More specifically, if there are measurements on specific resource metrics for a WS, where these metrics are actually the leaves of the derivation tree of a required composite metric, then we can actually build up the whole derivation tree and also create constraints on the values of all of its nodes, including the composite metric. Statistical processing is used for this reason. In addition, temporal database techniques are required for producing values/measurements on metrics for specific time periods or time instances where these measurements are not available.

To go on one step further, WS providers usually provide different QoS for their WSs to different classes of consumers. So we can associate each measurement to a class of consumer. In this way, we can relate a QoS demand to a specific class of consumer (e.g. by checking

the constraints on the 'price' metric or if the requester was a previous good customer) and dynamically produce a specific QoS offer for each WS that functionally satisfies the WS request.

This strategy is very good if QoS offers can be constructed that contain constraints on all the QoS metrics of the QoS demand. Its advantage is that dynamic QoS offers constructed contain only unary constraints on metrics. In this way, linear offer CSPs are constructed so QoS-based WS discovery takes significantly less time even if the QoS demand contains some non-linear constraints on metrics. However, this strategy has two main disadvantages. The first one is that measurements can not always be produced on all metrics and require effort not only from WS providers but also from third-party monitoring applications and WS requesters that execute these WSs. The second one is that malicious users can take advantage of this strategy. They can issue many QoS demands with different QoS metrics and constraints each time in order to predict the complete performance behavior of offered WSs. As a result, they can make more focused Denial of Service (DoS) attacks to WSs.

### Mixed Strategy

In our opinion, a combination of the previous techniques is the best strategy for local alignment and this is what we are trying to implement in our discovery engine. To be more specific, the best choice is to first use the second sub-strategy of the *mathematical derivation* strategy in order to minimize the $N$ metrics list for each offer $O_i$ (with a time limit for each offer if we have metrics added to $N$ at execution) and then to use the *statistical* strategy in order to produce constraints on the remaining $N_k$ metrics in each $O_i$ offer. The next paragraph outlines a pseudo-code of our proposed best choice for mixing strategies. Of course, the reverse combination of the above strategies could be a serious alternative. Another but not very preferable alternative would be to use the first sub-strategy of the *mathematical derivation* strategy in order to produce constraints only on resource metrics in all $O_i$ and $D$ specs and then to use the *statistical* strategy if there are resource metrics of the demand that are not constrained in some $O_i$ offers.

The pseudo-code of our proposed mixed strategy is:

```
N=callMathPattern(Oi,D,timeLimit);

if notEmpty(N){

  N=statDeriv(N);

}
```

where $callMathPattern$ is a function calling the Mathematical pattern strategy on a QoS offer and demand with a time limit and $statDeriv$ is a function calling the statistical derivation strategy on a metric array.

Of course this mixed strategy may have better results than the other strategies but it increases the execution time of the local alignment process. In addition, nothing ensures us that the final goal of perfectly aligning the QoS offers will be achieved.

### 6.3.3 Discussion

In this section we have presented a complete alignment process of QoS-based WS specifications. As it can be understood, this alignment process may not be needed if all specifications use the same set of metrics. However, reality has shown that this is not the case as people tend to understand in a different way the same concepts and usually write different mathematical expressions to express the same computations.

Another point that must be made is that the success of the alignment process depends on the (semantic) knowledge that is described and on the available measurements of QoS metrics. If some information is missing, then it is most probable that some matches between QoS metrics may never be inferred by a rule engine. However, it is obvious that this alignment process does its best to increase the accuracy of the QoS-based WS matchmaking process by inferring the equivalent QoS metrics of two QoS-based WS specifications based on the available knowledge and measurement values.

## 6.4 QoS-based Web Service Discovery

As the QoS-based WS discovery process is separated into two sequential sub-processes: WS matchmaking and selection, our goal was to contribute to these two sub-processes in our thesis. This goal was achieved and in the next two subsections we are going to analyze theoretically the QoS-based WS matchmaking and selection algorithms that we propose.

### 6.4.1 QoS-based Web Service Matchmaking

Based on the analysis of chapter 5 there are two main categories of approaches in QoS-based WS matchmaking: ontology-based [ZCL04, OVSH06, GZ07] and CP-based [DSL04, CMDTT05] ones. As it was explained, the ontology-based category of approaches has all the potentials to be more accurate as it can exploit a QoS metric matching algorithm but has performance problems. In addition, the research approaches adopting it cannot express more complex constraints like "if $A$ then $B$", where $A$ and $B$ can be complex or simple constraints. On the contrary, the other category presents better performance and exploits better expressivity concerning constraint specification but the two research approaches adopting it are purely syntactic (so they have low accuracy) and do not use appropriate/correct matchmaking metrics. For the above reasons, we chose to extend the second category of approaches as it is more promising and we did not want to use a pure ontological approach that suffers from performance problems. Thus, our approach is a mixture of the main categories of approaches: alignment is performed on semantic QoS-based WS specifications and then the altered specifications are transformed to constraint (optimization) problems to be solved. To the best of our knowledge, none research approach has adopted a hybrid approach in QoS-based WS matchmaking and this fact makes our approach unique and original.

Before extending and proposing our QoS-based WS matchmaking algorithms, we made

a small survey of what exactly solving technique to use for solving constraints. Our survey was motivated by the fact that the CP technique does not have great performance when the constraint model (i.e specification) to be solved contains local linear constraints and real-valued variables. This inefficiency of the CP technique lead us to MP and more specifically to MIP. The MIP technique presents great performance for local linear constraints and for both integer-based and real-valued-based variables of constraint models. Thus, our first proposal was that the MIP technique must be used when only (local) linear constraints are present in constraint models. However, when also global linear or (global and/or local) non-linear constraints are present, then the CP technique must be used for solving. Our proposal was experimentally evaluated and the results are analyzed in chapter 7. The results confirmed our assumptions.

Based on the latter proposal and on the fact that we had already defined an alignment algorithm for QoS-based WS specifications, our next step was to choose an appropriate matchmaking metric in order not to destroy the increase of the precision offered in advance by the alignment algorithm. We started our new survey by inspecting the matchmaking metrics offered by the two research approaches of the CP-based category. The first approach [DSL04] uses a simple matchmaking metric: A QoS offer $O$ matches a QoS demand $D$ if and only if $O$ has common values in all QoS metrics used by $D$ with $D$. This matchmaking metric is wrong as it includes in its results QoS offers that may perform worse i.e. promise lower and equal quality values (for one or more QoS metrics) with respect to the ones requested by the QoS demand (for the same set of QoS metrics). Thus, the approach of [DSL04] surely presents low precision as it includes "positive negatives" in the result set [DMR02]. In addition, this approach presents low recall as it excludes from the result set those QoS offers that perform also better (for all QoS metrics) than the performance expected from the QoS offer ("negative positives" effect). Thus, this approach uses a totally faulty matchmaking metric.

The approach of Cortés et. al. [CMDTT05] proposes the matchmaking metric of conformance which is expressed by the following equivalence:

$$conformance\,(O_i, D) \Leftrightarrow s\left(P_i \wedge \neg P^D\right) = \mathrm{f} \tag{6.5}$$

where $s$ is a procedure returning true if the input Constraint Satisfaction Problem (CSP) [VHS96] is satisfiable or false otherwise, and f=false. To explain, an offer $O_i$ matches a demand $D$ when there is no solution to the offer's CSP $P_i$ that is not part of the solution set of the demand's CSP $P^D$. In other words, the offer $O_i$ matches demand $D$ if it promises some of the values of all QoS metrics that are expected by this demand. This matchmaking metric is a little faulty as it excludes from the result set QoS offers that provide better and/or equal solutions with respect to the solutions of the demand. For example, suppose that provider and requester use the same metric $X$, measuring the QoS Property of Availability, that has as value type the set $(0.0, 1.0)$. Further assume that the WS provider's CSP has the constraint: $X \geq 0.96$ while the WS requester's CSP has the constraint: $0.95 \leq X \leq 0.999$. Based on the metric of *conformance*, the provider's offer does not match the request as it contains solutions greater than that of the request's, although these solutions are better. As

accuracy is measured based on the metrics of *precision* and *recall* [DMR02], this approach has perfect precision (equal to 1.0) but recall less than 1.0 as it suffers from the *false negatives* effect.

Of course, better solutions are not always welcome by the WS requester for some QoS metrics and in specific application domains. For example, for a metric measuring *refresh time* both constraints of the WS requester must be respected especially when this user has a device not able to process the WS's output more quickly. For the above reason, we have introduced the notions of "sensitive" and "insensitive" QoS metrics. A QoS metric is "sensitive" when both its worse and better values (w.r.t the ones requested) have an impact on the WS requester and his application/system. Thus, both WS requester's bounds on a "sensitive" QoS metric must be respected. Examples of "sensitive" metrics are the ones measuring *refresh time* and *throughput*. On the other hand, "insensitive" QoS metrics are metrics whose "better" values (w.r.t the ones requested) are welcome by the WS requester. Examples of "insensitive" metrics are the ones measuring *response time* and *availability*.

Based on the previous analysis, we have proposed the following matchmaking metric: an offer $O_i$ matches a demand $D$ when its CSP $P_i$ has solutions that are either contained in the solution set of the demand's CSP $P^D$ or are 'better' than the demand's solutions but the latter condition should be taken into account only for specific QoS metrics. We call this matchmaking metric *conditional conformance*. In other words, when matchmaking, we should respect all constraints of the WS requester for "sensitive" QoS metrics and only the ones restricting lower quality values for "insensitive" QoS metrics. A 'better solution' contains a value for a metric that is 'better' than all the values of this metric in the demand's solutions. For unary constraints of arithmetic "insensitive" metrics, 'better' is translated to 'greater than' for positively monotonic metrics (e.g. Availability) or to 'less than' for negatively monotonic metrics (e.g. Response Time) while 'worse' is translated to 'less than' or 'greater than' respectively. Of course, if we use the MIP technique we do not solve CSP problems but MIPP problems.

Continuing the example of the previous paragraph, we search for those constraints of the QoS demand whose negation 'scans' the worse solutions of the QoS offer. The object of research is the constraint $0.95 \leq X$ while its negation is the constraint $0.95 > X$. Observe that if we try to solve the CSP containing the constraint $0.95 > X$ and the constraint of the QoS offer $X \geq 0.96$, we get no solution. So 'worse' solutions are not found. Conversely, the negation of the constraint $X \leq 0.999$ searches for better solutions than that of the demand's. Thus, for positively monotonic metrics $X$ like the one of the example, we are interested in QoS demand's unary constraints of the form: $a \leq X$, as the negation of these constraints scans for worse solutions of the QoS offer under consideration. Symmetrically, for negatively monotonic metrics $X$, the demand's 'interesting' constraints have the following form: $X \leq b$.

Even if our *conditional conformance* matchmaking metric is used to implement a MIP or CP algorithm that will exploit perfect accuracy, there are still two problems that must be corrected: a) our algorithm should not only return *matched* and *failed* QoS offers but should provide an *advanced categorization of results*, and b) when over-constrained QoS demands are issued, meaningful results should be presented to the user. Taking into account these two problems to be solved and our new metric, our effort was now concentrated on devising

and implementing QoS-based WS matchmaking algorithms exploiting the CP and MIP techniques individually. The next two sub-subsections unveil the algorithms that we have devised exploiting the MIP and CP techniques respectively.

Before continuing, it must be noted that additionally to the CP/MIP algorithms we propose for implementing the conformance matchmaking metric, one could easily implement two different approaches. The first approach is the naive one where firstly all the solutions of the offer are found and then checked if they are also solutions of the demand. This approach may scale well only for small solutions spaces. So, it is not general and not suitable for large solution spaces. The second approach relies on the next sub-subsection analysis of the conformance equivalence. Based on this analysis, the conformance of an offer $O_i$ to a demand $D$ is transformed to checking of the following expression:

$$s\left(\left(P_i^O \wedge \neg c_1^D\right) \vee \left(P_i^O \wedge \neg c_2^D\right) \vee \cdots \vee \left(P_i^O \wedge \neg c_Z^D\right)\right) = \text{false}$$

where $P_i^O$ is the CSP/MIP of the offer and $c_j$ are the constraints of the demand while $s$ is a satisfiability checking procedure (i.e. solving procedure). So this second approach tries to check if this expression, that contains disjunctive constraints, is true. Unfortunately, this approach has the disadvantage that disjunctive constraints do not propagate at all well in CP/MIP solvers. Thus, for the above reasons we decided not to further investigate, implement and experimentally evaluate these two alternative approaches.

### MIP Algorithms

The main problem we had to confront before devising our new algorithms was how to translate the *conditional conformance* matchmaking metric into a formalism that could be exploited by MIP algorithms as in its current form is more suitable for CP algorithms because it uses negation. Fortunately, by mathematically analyzing the *conformance* expression-equivalence (6.5) and by assuming that the QoS demand is expressed by conjunctions of $Z$ constraints, we obtain:

$$\begin{aligned}
&conformance\,(O_i, D) \Leftrightarrow s\left(P_i^O \wedge \neg P^D\right) = \text{f} \\
&\Leftrightarrow s\left(P_i^O \wedge \neg\left(c_1^D \wedge c_2^D \wedge \cdots \wedge c_Z^D\right)\right) = \text{f} \\
&\Leftrightarrow s\left(P_i^O \wedge \left(\neg c_1^D \vee \neg c_2^D \vee \cdots \vee \neg c_Z^D\right)\right) = \text{f} \\
&\Leftrightarrow s\left(\left(P_i^O \wedge \neg c_1^D\right) \vee \left(P_i^O \wedge \neg c_2^D\right) \vee \cdots \vee \left(P_i^O \wedge \neg c_Z^D\right)\right) = \text{f} \\
&\Leftrightarrow \left(s\left(P_i^O \wedge \neg c_1^D\right) = \text{f}\right) \wedge \left(s\left(P_i^O \wedge \neg c_2^D\right) = \text{f}\right) \wedge \ldots \\
&\wedge \left(s\left(P_i^O \wedge \neg c_Z^D\right) = \text{f}\right)
\end{aligned}$$

So the conformance of offer $O_i$ to demand $D$ is translated into checking if all of the CSPs– constructed by offer's CSP $P_i^O$ and the negation $\neg c_j^D$ of a demand's constraint $c_j^D$ – are unsatisfiable. If any of the CSPs is satisfiable then offer $O_i$ does not match with the demand

$D$. As satisfiability has the same meaning as feasibility and unary CSPs can be easily transformed to MIPPs, then we can translate conformance of offer $O_i$ to demand $D$ to checking if all of the MIPPs– constructed by offer's MIPP $P_i^O$ and the negation $\neg c_j^D$ of a demand's constraint $c_j^D$ – are infeasible. Thus, we have that:

$$conformance\,(O_i, D) \Leftrightarrow \left(fe\left(P_i^O \wedge \neg c_1^D\right) = \text{f}\right) \wedge$$
$$\wedge \left(fe\left(P_i^O \wedge \neg c_2^D\right) = \text{f}\right) \wedge \cdots \wedge \left(fe\left(P_i^O \wedge \neg c_Z^D\right) = \text{f}\right)$$

where $fe$ is a feasibility checking procedure of a MIPP executed by a MIP engine (MIPE). Thus, we have successfully translated the CP-based approach of conformance to a MIP-based approach.

Based on the above analysis, our *conditional conformance* matchmaking metric can be expressed as: Conformance of a QoS offer $O_i$ and a QoS demand $D$ is true if and only if all MIPP problems $P_i^O$ – constructed from all the constraints of the offer and the negation $\neg c_j^D$ of an 'interesting' demand's constraint $c_j^D$, where 'interesting' means the $a \leq X$ demand constraints for "insensitive" positively monotonic metrics, the $X \leq b$ demand constraints for "insensitive" negatively monotonic metrics, and both $a \leq X$ and $X \leq b$ demand's constraints for "sensitive" metrics– are all infeasible. This definition is based on the assumption that the QoS demand not only contains a conjunction of constraints but also these constraints are unary.

After expressing mathematically the *conditional conformance* matchmaking metric, we concentrated on the algorithmic part and proposed [KP07d] and implemented [KP08d] two different QoS-based WS matchmaking algorithms. The first one applies only on QoS demands containing unary constraints (i.e. each demand's constraint has only one QoS metric – this is the most often case for QoS demands in reality) while the other one is more generic and allows for n-ary constraints on QoS demands. Of course, both of the proposed algorithms handle QoS offers with n-ary constraints. In the sequel, we are going to theoretically analyze these algorithms based on the following factors: execution time, accuracy, advanced categorization of results, and size of matchmaking results. In order to make the comparison more interesting, we also compare these two algorithms with a MIP version of both the *conformance* and the *conditional conformance* matchmaking metrics. The notation used is explained in the next paragraph while the algorithms and their analysis is given in separate paragraphs. The experimental evaluation of these four algorithms will be analyzed in chapter 7.

**Assumptions and notation** Our basic assumption is that the matchmaking scenario contains $N$ QoS offers and one demand that involves a conjunctive set of unary constraints. We also assume that these QoS specifications have already been aligned with the QoS metric matching algorithm and that they eventually involve the same number of QoS metrics $M$, from which $K$ are "sensitive" and $L$ are "insensitive". In addition, we assume that there are only numeric (real or integer) QoS metrics. So in the worst case, there will be $2 \cdot M$ unary constraints in the QoS demand as each QoS metric will have at most two constraints (one

restricting its upper limit and another one restricting its lower limit). Finally, we assume that the average execution time of the MIP engine (MIPE) for solving: **a)** a MIP problem (MIPP) of $2 \cdot M + 1$ constraints that involve $M$ metrics is $T_M^{2M+1}$; **b)** a MIP optimization problem (MIPOP) of $2 \cdot M$ constraints and one objective that involve $M$ metrics is $To_M^{2M}$.

**Algorithm of Cortés et. al.** Based on the analysis of the previous subsection, conformance [CMDTT05] of offer $O_i$ to demand $D$ is translated to checking if all of the MIPPs–constructed by offer's MIPP $P_i^O$ and the negation $\neg c_j^D$ of a demand's constraint $c_j^D$ – are infeasible. The pseudo-code for the conformance algorithm is the following:

```
boolean conform(O_i,D){
  for each c_j in D{
    stat=solveMIP(O_i,negate(c_j));
    if stat return false;
  }
  return true;
}

<match,fail> conform_algo(O,D){
  match=[];
  fail=[];
  for each O_i in O{
    if conform(O_i,D) add(O_i,match);
    else add(O_i,fail);
  }
}
```

where $negate\,(c_j)$ is a function that negates a constraint and returns it, $solveMIP(prob, c)$ is a function that solves a MIPP constructed by another MIPP $prob$ plus an additional constraint $c$ and returns true if the new MIPP is feasible, $conform$ is a function that checks the conformance of an offer $O_i$ with the demand $D$, $conform\_algo$ is the actual conformance algorithm that takes as input an array of offers $O$ and the demand $D$ and returns two arrays containing the conforming and the non-conforming offers, respectively, and $add$ is a function that adds an offer $O_i$ to an array.

In the worst case, as the QoS demand will contain $2 \cdot M$ unary constraints, conformance checking will be transformed to solving at most $2 \cdot M$ MIPPs, each containing $2 \cdot M + 1$ constraints. In addition, this conformance checking will be performed $N$ times, one time for each of the $N$ QoS offers. For brevity, let us call this matchmaking algorithm '$MIP_{conf}$'. Hence, Algorithm $MIP_{conf}$ will take $O\left(2 \cdot N \cdot M \cdot T_M^{2M+1}\right)$ time in the worst case. It is easy to see that in the best case this algorithm will solve just one MIPP for each QoS offer so it will take $\Omega\left(N \cdot T_M^{2M+1}\right)$ time. This algorithm produces two types of results: **a)** *exact* results containing QoS offers conforming to the QoS demand; **b)** *fail* results containing non-conforming QoS offers. So Algorithm $MIP_{conf}$ does not provide advanced categorization of matchmaking results. In addition, this algorithm will produce only *fail* results in case of over-constrained QoS demands. Finally, as explained in the previous subsection, this algorithm presents perfect recall but imperfect accuracy.

**Conditional Conformance Algorithm** A MIP algorithm implementing the matchmaking metric of *conditional conformance*, which we will call $MIP_{cond-conf}$ for brevity, will have perfect accuracy. The pseudo-code for this algorithm is the following:

```
boolean cond-conform(O_i,D){
  for each c_j in D{
    Metric m = c_j.metric;
    String pred = c_j.binPred;
    if (sensitive(m) || ((pmon(m) && pred=='>=') || (!pmon(m) && pred=='<='))) {
      stat=solveMIP(O_i,negate(c_j));
      if stat return false;
    }
  }
  return true;
}

<match,fail> cond-conform_algo(O,D){
  match=[];
  fail=[];
  for each O_i in O{
```

```
    if cond-conform(O_i,D) add(O_i,match);
    else add(O_i,fail);
  }
}
```

where besides the previously analyzed functions (*negate*, $solveMIP$ and *add*) we introduce four new ones: a) $pmon(m)$ which returns true if a metric $m$ is positively monotonic and false otherwise; b) $sensitive(m)$ that returns true if a metric $m$ is "sensitive" and false otherwise, c) *cond-conf* checks the conditional conformance of an offer $O_i$ with the demand $D$, and d) $cond\text{-}conform\_algo$ is actually the same function as $conform\_algo$ except that it calls the $cond\text{-}conform$ function instead of the $conform$ one. Moreover, for a unary constraint $c$ the expression $c.m$ returns its involved metric and the expression $c.binPred$ returns its involved binary predicate.

Concerning matchmaking time, we have the following analysis: In the worst case, as the QoS demand will contain $2 \cdot M$ unary constraints, conformance checking will be transformed to solving at most $2 \cdot K + L$ MIPPs. So the time complexity of the matchmaking Algorithm $MIP_{cond-conf}$ will be: $O\left(N \cdot (2 \cdot K + L) \cdot T_M^{2M+1}\right) < O\left(2 \cdot N \cdot M \cdot T_M^{2M+1}\right)$. In the best case, this algorithm will solve one MIPP for every QoS offer so it will take $\Omega\left(N \cdot T_M^{2M+1}\right)$ time. Thus, in the average case (half of the worst number of problems is solved for each QoS offer) this algorithm will perform a little better as it will solve some less MIPPs than the $MIP_{conf}$ algorithm because $\frac{2 \cdot K + L}{2} < \frac{2 \cdot K + 2 \cdot L}{2}$. However, this algorithm still does not offer advanced categorization of results and does not produce results for over-constrained QoS demands.

**Unary Algorithm** To enable advanced categorization of results, we had to find offers with better solutions than those of the demand. So we had to solve all possible MIPPs ($2 \cdot M$ in the worst case and $2 * K + 1$ in the best) instead of $M$ so as to produce the following categories of conforming offers: **a)** *super* offers promising at least one better solution than the demand's; **b)** *exact* offers containing a solution set that is a subset of the demand's solution set. For producing **exact** matching results for over-constrained demands, we had to first split non-conforming results into two categories: **a)** *partial* results that do not violate all 'interesting' QoS demand's constraints of "insensitive" QoS metrics and all QoS demand's constraints of "sensitive" QoS metrics; **b)** *fail* results that violate all $L$ 'interesting' constraints of "insensitive" QoS metrics and $2 * K$ constraints of the "sensitive" QoS metrics of the demand.

We also had to introduce weights to the constraints where a weight of 10.0 meant that the constraint was "hard" while a weight in $(0, 1.0)$ meant that the constraint was "soft". We have put a weight of 10.0 to hard constraints in order to show their significance with respect to the soft constraints. In this way, we declare that a hard constraint violation costs more than 10 soft constraints violations. A counter *cn* was dedicated for each offer adding to its total the weight of the violated constraint. Then, if an over-constrained demand was

issued, we executed a constraint relaxation task [KP07d] that promoted one or more *partial* results as *exact* according to the *cn* counter. Obviously, the *partial* offer with the smallest number in the *cn* counter was promoted.

The pseudo-code for this algorithm is the following:

```
<stat,weight,hards> solve_not_probs(O_i,D){
  weight=0.0;
  hards=0.0;
  cons=0;
  bcons=0;
  for each c_j in D{
    Metric m = c_j.metric;
    String pred = c_j.binPred;
    if (sensitive(m) || ((pmon(m) && pred=='>=') || (!pmon(m) && pred=='<='))) {
      cond=solveMIP(O_i,negate(c_j));
      if (cond) {
cons++;
if (c.weight==10.0) hards++;
weight=weight+c.weight;
      }
    }
    else if (bcons==0 && !sensitive(m) && ((pmon(m) && pred=='<=')
    || (!pmon(m) && pred=='>='))){
      cond=solveMIP(O_i,negate(c_j));
      if cond bcons++;
    }
  }
  if (cons==2*sens(D)+non_sens(D)) stat=0;
```

```
  elseif cons!=0 stat=1;
  elseif bcons==0 stat=2;
  else stat=3;
  return <stat,weight,hards>;
}

<super,exact,partial,fail> unary(O,D){
  super=[];
  exact=[];
  partial=[];
  fail=[];
  for each O_i in O{
    <stat,weight,hards>=solve_not_probs(O_i,D);
    switch stat{
      case 0: add(O_i,fail);break;
      case 1: add(<O_i,weight,hards>,partial);break;
      case 2: add(O_i,exact);break;
      case 3: add(O_i,super);break;
    }
  }
  if (super==[] && exact==[]) add(promote(partial),exact);
}
```

where except from the same functions that were previously defined on the previous algorithms, five new functions are defined: a) $sens(D)$ returns the number of sensitive offers referenced at the constraints of the demand $D$, b) $non\_sens(D)$ returns the corresponding number of non-sensitive metrics of the demand $D$, c) $solve\_not\_probs(O_i, D)$ is a function that calculates for an offer $O_i$ the category that this offer should belong, the total weight of the demand's violated constraints and the number of hard constraints of the demand violated, d) $promote(partial)$ is a function that ranks the contents of the array $partial$,

finds the offer that has the least weight and violates the least amount of hard constraints, removes the corresponding offer from this array and returns it, and e) $unary(O,D)$ is the actual algorithm that returns the four categories of results according to an offer array $O$ and the demand $D$. Finally, for a unary constraint $c$ the expression $c.weight$ returns the weight of this constraint.

Let us call this algorithm $MIP_{unary}$ because while it can handle n-ary constraints in QoS offers, it works only with unary-constrained QoS demands. In the best case, Algorithm $MIP_{unary}$ will solve $2 * K + 1$ MIPPs ($K$ MIPPs for distinguishing between *partial* and *fail* QoS offers and 1 MIPP for distinguishing between *exact* and *super* QoS offers) of $M$ metrics and $2 \cdot M + 1$ constraints for each QoS offer. In the worst case, it will solve $2 \cdot M$ MIPP problems of the same form. So its complexity is $\Omega\left(N \cdot (2K+1) \cdot T_M^{2M+1}\right)$ in the best case and $O\left(2 \cdot N \cdot M \cdot T_M^{2M+1}\right)$ in the worst. As it can be seen, Algorithm $MIP_{unary}$ is the slowest of the three algorithms already introduced. However, its main advantages is that it presents advanced categorization of results, it has perfect accuracy and it produces one or more *exact* matches for over-constrained QoS demands.

**Optimization Algorithm** Algorithm $MIP_{unary}$ solves all disadvantages of the previous two algorithms but as we will see in chapter 7, it is very slow. So we pay a performance penalty for being accurate and for offering a variety of solutions to WS requesters even if their QoS demands are over-constrained. In addition, $MIP_{unary}$ (and $MIP_{cond-conf}$) enforces that QoS demands should be composed of only unary constraints. For the next algorithm, we relax precision in order to increase performance while trying to preserve all other matchmaking advantages.

The main idea of the new algorithm is modification of the notion of 'better' or 'worse' that is applied to MIPP solutions. To remind, all previous algorithms check if a tuple of a n-tuple solution of the offer is worse than all corresponding tuples of the demand. This algorithm checks if the whole solution of the offer is worse than all the solutions of the demand by assigning a preference or value to each MIPP solution. The technique for assigning preferences to solutions is based on utility functions and weights on MIP variables [CMDTT05]. Each MIP variable (a map of a metric) is given a (user) weight or preference (taking values from the set [0.0,1.0]) to reflect the significance of this variable to the preference/value of the solution. In addition, each possible value of this variable is also given a preference ($\in [0.0, 1.0]$) by the variable's utility function. The preference of a MIPP solution is given by the following formula-sum on all variables $X_j$: $p_s = \sum_{X_j}(w_{X_j} \cdot uf_{X_j}(X_j))$, where $w_{X_j}$ is the weight of the variable $X_j$ and $uf_{X_j}()$ is its utility function. This formula is instantiated by any admissible value from the domain of values of the variables $X_j$ in order to produce any preference value. In the next subsection analyzing our QoS-based WS selection approach, you will see that we use two types of utility functions depending on the value monotonicity of the metric/MIP variable.

Based on the above technique, a partial ordering of all solutions of a MIPP can be inferred. This is the appropriate mean in order to define matchmaking: an offer's MIPP $P_i$ matches the MIPP $P^D$ of the demand if its worst solution has a preference of greater or equal

value with respect to the preference of the worst solution of the demand. So our new algorithm's logic is as follows: we compute the preference $p_s^D$ of the demand's MIPP $P^D$ worst $s^D$ solution by solving one MIP Optimization Problem (MIPOP) (minimization) [KP06]. For each offer's MIPP $P_i$, we compute the preference $p_s^i$ of its worst $s^i$ solution in the same manner as above. Then, we consider two cases:

1. If $(p_s^i < p_s^D)$, then the offer is put in the *fail* match list.
2. If $(p_s^i \geq p_s^D)$, then the offer is put in the *exact* match list.

The pseudo-code for this algorithm follows:

```
obj getObjective(D){
  obj=0;
  for each Metric m in D{
    obj=obj+m.weight*uf(m);
  }
}


<match,fail> optimization(O,D){
  match=[];
  fail=[];
  c=getObjective(D);
  min=solveMIOP(D,c,'minimize');
  for each O_i in O{
    min_i=solveMIOP(O_i,c,'minimize');
    if min_i>=min add(O_i,match);
    else add(O_i,fail);
  }
}
```

where function $getObjective(D)$ creates an objective expression composed from the metrics contained in $D$, $m.weight$ is the weight of metric $m$ and $uf(m)$ is its utility function, function $solveMIOP(spec, obj, str)$ solves a MIPOP from a specification $spec$ according to

an objective *obj* and the type of optimization *str*, and *optimization*$(O, D)$ is the function implementing our algorithm that returns two categories/arrays of results.

This new algorithm, called '$MIP_{opt}$', is actually a simpler form of the algorithm proposed in [KP07d]. In all cases, Algorithm $MIP_{opt}$ will solve $N+1$ MIPOPs of $M$ metrics and $2 \cdot M + 1$ constraints for each QoS offer and the QoS demand. Thus, its time complexity is $\Theta\left((N+1) \cdot To_M^{2M}\right)$. We expect that for unary constraints, it is the case that: $To_M^{2M} \approx T_M^{2M+1}$, so algorithm $MIP_{opt}$ will be faster than all other algorithms. However, this must be proved in the evaluation of the matchmaking algorithms provided in chapter 7.

Algorithm $MIP_{opt}$ does not require the QoS demand to contain only a conjunction of unary constraints so it is more generic but provides only two types of results. However, if we order each result based on its preference of its worst solution, then we actually have one category for each result. So two types of categorizations are offered. Moreover, as all results returned are ordered, the WS selection subprocess does not need to be executed. In addition, constraint relaxation is inherent to the optimization of MIPOPs based on linear utility functions. In this way, some QoS offers that violate some constraints affecting metrics of less significance of the QoS demand but also provide better values for other more important metrics are promoted from *failed* matches to *exact*. On one hand, this feature enables the return of matching results for over-constrained QoS demands. On the other hand, it reduces the precision (but not the recall) of the algorithm as now more matching results are returned with respect to the precise number of matching results.

**Example** To demonstrate the functionality of our two proposed QoS-based WS discovery algorithms, we supply a simple example of their application to a small set of four QoS offer MIPPs $P_i$ and one demand MIPP $P^D$. Assume that all MIPPs have the following three definitions: $X_1 :: (0.0, 86400.0] \downarrow$, $X_2 :: (0, 100000] \uparrow$ and $X_3 :: (0.0, 1.0) \uparrow$, where all variables are actually mapped to "insensitive" QoS metrics. Based on these variable definitions, assume that each MIPP has the following constraints: $P_1 : [X_1 \leq 10.0, X_2 \leq 100, X_2 \geq 50, X_3 \geq 0.9]$, $P_2 : [X_1 \leq 4.8, X_2 \leq 50, X_2 \geq 40, X_3 \geq 0.95]$, $P_3 : [X_1 \leq 16, X_2 \leq 40, X_2 \geq 30, X_3 \geq 0.98]$, $P_4 : [X_1 \leq 16, X_2 \leq 50, X_2 \geq 40, X_3 \geq 0.98]$, and $P^D : [X_1 \leq 15.0, X_2 \geq 40, X_2 \leq 60, X_3 \geq 0.99]$. Moreover, assume that the WS requester does not provide weights to the constraints of his demand (so they are all considered "hard") and associates the following weights to the three metrics/variables: $X_1 \leftarrow 0.3$, $X_2 \leftarrow 0.3$, $X_3 \leftarrow 0.4$, while $a = 0.7$ and $b = 0.3$ for the QoS-based WS selection process [KP06]. In addition, assume that the following utility functions are applied to the MIPOPs: $uf_{X_1} = (16 - X_1)/16$, $uf_{X_2} = (X_2 - 30)/70$, $uf_{X_3} = (X_3 - 0.9)/0.1$ [KP06].

We firstly apply the $MIP_{unary}$ QoS-based WS matchmaking algorithm. This algorithm initially produces the following four results lists: $Super = [\,]$, $Exact = [\,]$, $Partial = [(O_1, P_1, 10.0), (O_2, P_2, 10.0), (O_4, P_4, 20.0)]$, $Fail = [(O_3, P_3)]$, as offers $O_1$ and $O_2$ violate only one constraint, offer $O_4$ violates two and offer $O_3$ violates all "interesting" constraints. Then it sorts the partial list based on the third argument of each entry and produces the following two lists: $Exact^* = [(O_1, P_1), (O_2, P_2^O)]$ and $Partial = [O_4, P_4^O]$. Finally, the selection process (which is called in order to sort the two entries of the best category of results) solves two MIPOPs for each of the first two offers, produces one score for each:

| Spec | Constraints | $MIP_{unary}$ | $MIP_{opt}$ |
|---|---|---|---|
| $P_1$ | $X_1 \leq 10.0$, $X_2 \leq 100$, $X_2 \geq 50, X_3 \geq 0.9$ | exact | partial |
| $P_2$ | $X_1 \leq 4.8$, $X_2 \leq 50$, $X_2 \geq 40, X_3 \geq 0.95$ | exact | exact |
| $P_3$ | $X_1 \leq 16$, $X_2 \leq 40$, $X_2 \geq 30, X_3 \geq 0.98$ | fail | partial |
| $P_4$ | $X_1 \leq 16$, $X_2 \leq 50$, $X_2 \geq 40, X_3 \geq 0.98$ | partial | partial |
| $P^D$ | $X_1 \leq 15.0$, $X_2 \geq 40$, $X_2 \leq 60, X_3 \geq 0.99$ | – | – |

Table 6.3: Example of the application of the $MIP_{unary}$ and $MIP_{opt}$ matchmaking algorithms

$Score_1^Q \simeq 0.44$ and $Score_2^Q \simeq 0.55$, and returns the following ordered list to the requester: $Exact^* = [(O_2, 0.55), (O_1, 0.44)]$ along with the other two (failing) lists. However, the WS requester is warned that this list contains offers that do not satisfy his constraint: $X_3 \geq 0.99$.

As it can be seen, offer $O_2$ is at the top of the result list as it violates in a significantly lower amount the last constraint of the demand with respect to the amount of violation of $O_1$. So if $O_2$ is selected by the WS requester, then the minimum possible constraint relaxation will have been achieved.

Let us now focus on the application of the $MIP_{opt}$ algorithm. This algorithm will solve one MIPOP for each QoS specification (including that of the QoS demand). So for each MIPP the following preferences will be calculated: $[P_1 : p_s^1 = 0.1982]$,$[P_2 : p_s^2 = 0.4528]$,$[P_3 : p_s^3 = 0.32]$, $[P_4 : p_s^4 = 0.3628]$, and $P^D : [p_s^D = 0.4216]$. Thus this algorithm will produce the following results lists: $Super = [\,]$, $Exact = [O_2]$, $Partial = [(O_1), (O_3), (O_4)]$, $Fail = [\,]$.

As it can be seen, offer $O_2$ is in the *exact* match list although it violates the last constraint of the demand. The reason for this is that the preference of its worse solution is greater than the preference of the worse solution of the demand. To put it in another way, $O_2$ provides a far better lowest value for the $X_1$ attribute with respect to the worse lowest value for the $X_3$ attribute (compared to the corresponding lowest values of the demand). Another observation is that $O_1$ pays the penalty of providing the minimum possible value for the $X_3$ attribute and is considered a *partial* result. The last observation is that $O_3$ is promoted as a *partial* result. $O_3$'s promotion is due to the fact that its worse solution violates only in a small amount the constraints of the demand.

Table 6.3 summarizes the contents of this example. This table displays for each QoS specification what were its constraints and in which category of results was put by the two compared algorithms. Of course, for the QoS demand only its constraints are shown.

**Achievements** For QoS-based WS matchmaking, we have proposed two algorithms that use the MIP solving technique. The actual system that will exploit these two algorithms will have to choose between them based on user requirements of the user that issues the QoS request. If the user wants a very fast execution and is not interested much if the accuracy

| Research Approach | Semantic QoS Metric Matching | Precision | Recall | Performance | Results Cat/tion | Optim/tion |
|---|---|---|---|---|---|---|
| $MIP_{conf}$ [CMDTT05] | – | excellent | good | excellent | fair | – |
| $MIP_{cond-conf}$ | yes | excellent | excellent | excellent | fair | – |
| $MIP_{unary}$ | yes | excellent | excellent | good | excellent | yes |
| $MIP_{opt}$ | yes | good | excellent | excellent | excellent | yes |

Table 6.4: Comparison of our proposed QoS-based MIP-based WS matchmaking algorithms

is not perfect, then the $MIP_{opt}$ algorithm should be used. Otherwise, if the user wants high accuracy and does not care too much if the results are returned two to ten seconds later, then the $MIP_{unary}$ algorithm should be used. Table 6.4 summarizes the features (based on the requirements we have set in chapter 4) of the two proposed algorithms in order to compare them. In order to make the comparison even more interesting and complete, the $MIP_{conf}$ and $MIP_{cond-conf}$ matchmaking algorithms are also included and analyzed in the table.

If this table is compared against table 5.3, it is easy to see that our two proposed algorithms outperform all other research approaches. This is mainly due to four reasons: a) use of the alignment algorithm; b) high accuracy of results; c) advanced categorization of results; d) return of useful results in case of over-constrained QoS demands. Moreover, the performance of the proposed algorithm is from good to the best. So our QoS-based MIP-based WS matchmaking algorithms have achieved their goal as they satisfy all the requirements set in chapter 4. Their evaluation in chapter 7 of this dissertation will verify our claims.

### CP Algorithms

As already mentioned in several places in this dissertation, the CP technique must be used for solving non-linear QoS constraints involving QoS metrics (i.e constraint variables). Non-linear constraints in QoS specifications can be used to express relations between QoS metrics and objective functions (for optimization). For example, the price of a WS (i.e WS's pricing model) can be expressed as a non-linear function of many QoS metrics, e.g. $price = \sqrt{((1 - responseTime)/0.01)^2 + (availability/0.01)^2}$ where $responseTime$ and $availability$ take values in $(0.0, 1.0]$.

Based on the analysis of the previous sub-subsection, the matchmaking metrics of *conformance* and *conditional conformance* have been expressed in terms of a series of MIPPs which have to be all *infeasible*. As *infeasibility* in MIP (i.e MIPP has no solutions) has almost the same meaning as *unsatisfiability* in CP and MIPP problems can be easily transformed to CSPs (also the case for MIPOPs and CSOPs), these two matchmaking metrics can be easily expressed in terms of a series of CSPs that all have to be unsatisfiable. Moreover, by

relying on the above observation, it was easy to transform all of our MIP-based QoS-based WS matchmaking algorithms to their CP counterparts. Thus, we have implemented the CP counterparts of our MIP-based algorithms $MIP_{conf}$, $MIP_{cond-conf}$, $MIP_{unary}$ and $MIP_{opt}$ that we name $CP_{conf}$, $CP_{cond-conf}$, $CP_{unary}$ and $CP_{opt}$ respectively.

These four CP algorithms will exhibit the same behavior concerning the characteristics of accuracy, categorization of results and optimization for over-constrained demands with respect to their MIP counterparts. In addition, we expect that the relevant order of these CP algorithms is not going to change concerning matchmaking time and we base our claims on the formal performance analysis performed on the corresponding MIP algorithms. This performance analysis can be easily changed (by altering the notations) to reflect the same results on the proposed CP matchmaking algorithms. However, one suspicion that can be oscillated concerns the matchmaking time of the $CP_{opt}$ optimization algorithm. Solving CSOPs quickly requires [RvBW06] that constraint propagation techniques must be applied to the objective function (which is usually represented as a constraint). Special techniques like *local search methods* [RvBW06] must be used in order to have an effective solving procedure. So the matchmaking time of $CP_{opt}$ will mostly depend on the CSP engine used and on which special techniques this engine uses to solve CSOPs.

**Different CSP Frameworks Investigation** Apart from these four proposed CP algorithms, as CP is a broad field, we conducted an extensive research in CP in order to find solutions to two research problems. The first research problem was how to solve over-constrained QoS demands apart from solving CSOPs (with the $MIP_{opt}$ algorithm) or exhaustively solving many CSPs (with the $MIP_{unary}$ algorithm). Our research concentrated both on representation formalisms and their implementing frameworks. For representation formalisms, we found out that there are many possible representation frameworks of optimization problems for CP but the most appropriate ones for QoS-based WS optimization are two: a) *Weighted MAX-CSP* [RvBW06], and b) *Semiring-Based CSP*s [BMR97]. In *MAX-CSP*, the optimization problem is represented by a CSP with the objective to find a variable assignment that satisfies the maximal number or constraints. This is equivalent to finding a variable assignment that minimizes the total number of violated constraints. In case where not all constraints are important, weights are associated with individual constraints and the objective is to maximize the total weight of the satisfied constraints. In this case, we have a *Weighted MAX-CSP* problem. Concerning QoS-based WS matchmaking, if all QoS-based WS specifications have unary constraints, then all CSP variables are independent with each other so instead of solving many CSPs only one CSP is required to be solved. This CSP will contain all constraints of the offer and the negation of all the constraints of the demand along with their weights so it will certainly be represented as a *Weighted MAX-CSP*. The goal will be to maximize the total weight of all the unsatisfied constraints of the demand. If the total weight equals the sum of the weights of all constraints of the demand, then the offer is conforming with the demand. Otherwise, it will be a *partial* or *fail* (weight equals zero) result. As each offer is not only categorized but also gets a score, in case of over-constrained demands, the offer with the maximum score is the one which matches better with the demand. *Local Search Methods* [RvBW06] can be used to solve both *MAX-CSP*s

and *Weighted-MAX-CSP*s.

A *semiring constraint network* is actually a *constraint network*. What differentiates it from classical constraint networks is the fact that each constraint maps the assignment of its variables to values in the c-semiring. A c-semiring resembles the classical semiring notion and is used to formalize the notion of *satisfaction degrees* or *preference levels*. This simple algebraic structure is specified by a set $E$ of satisfaction degrees, where two binary operators are defined: a) $\times_s$ specifies how to combine preferences, and b) $+_s$ is used to induce a partial ordering on E. In addition, two special values are defined: **0** and **1**. The former is used to indicate that a constraint is not satisfied at all while the latter is used to indicate that the constraint is satisfied completely. Thus, a c-semiring is a 5-tuple of the form: $\langle E, +_s, \times_s, 0, 1\rangle$. Actually, the c-semiring framework is quite general and can express *MAX-CSP*s and other types of *soft constraint* CSPs. In our case, we could use this framework to represent and solve CSOPs by providing appropriate functions for the $times_s$ and $+_s$ operators. Then we could develop a matchmaking algorithm that would compare offers based on their lowest preference against the lowest preference of the demand, much like our $MIP_{opt}$ algorithm. The *consistency level* of a complete assignment $t$ (i.e a solution), $val_s(t)$, is obtained by combining the individual level of each constraint: $val_s(t) = \times_s(t[V])$. Then the $+_s$ is used to compare all solutions in order to find one or more *optimal* solutions. Unfortunately, solving a c-semiring CSP is a **NP-hard** task which can be reduced to **NP-complete** if the computations of $a \times_s b$ and $a +_s b$ are polynomial in the size of their arguments.

Based on the findings of our research on the first problem, we have found two different CSP frameworks with respect to the classical one. So we could use the representation formalisms of these frameworks and their solving engines in order to represent and solve our matchmaking problems. Unfortunately, concerning the semiring-based CP approach, we did not find any stable free or commercial solving engine that adopts it and uses it. So we had to develop prototype algorithms from scratch which could be very time-consuming and not very efficient as we wanted to use stable and widely tested algorithms in our experimental evaluation without taking us a lot of time in order to evaluate them. A same more-or-less situation also applied to the *Weighted-MAX-CSP* approach. Thus, our decision was to use these two representation frameworks in the near future when stable and widely-tested tools or algorithms will be available in free or commercial solving engines.

**Dynamic Constraint Satisfaction** The second research problem concerned the fact that the *conditional conformance* matchmaking metric requires solving a series of very similar CSPs which differ in only one constraint. So the focus of our research was to find a specific solving technique-framework in CP that could take advantage of both the knowledge of solving the previous CSP and the small change that takes us from the previous CSP to the next one. In this way, the CSP engine would not have to solve each new CSP from scratch but could use the knowledge from the solving procedure of the previous CSP in order to solve the new one. In addition, it would be wonderful not to create the new CSP from scratch but have the ability to delete one constraint and create a new one so as to build the new CSP. This CP framework would help us achieve significant time reductions in the

matchmaking time of our algorithms. Thus, our research concentrated on a specific area of CP called *Dynamic Constraint Satisfaction* (DSC) [VJ05].

In DSC, a *dynamic constraint satisfaction problem* (DSCP) is a sequence of CSPs, each one produced from some changes in the definition of the previous one. A change may affect any part of problem definition: variables, domains of variables, constraints, constraint scopes (addition or removal of a variable in constraint's definition), or constraint definition (e.g change in the mathematical expression). As all of these types of changes can be basically expressed as constraint additions or removals, a DSCP is a sequence $\{P_0, P_1, \ldots, P_i, \ldots, P_n\}$ where, for each $i$, $1 \leq i \leq n$, $P_i$ is a CSP, $C_{a_i}$ is a set of added constraints, $C_{r_i}$ is a set of removed constraints, such as $C_{r_i} \subseteq P_i$, and $P_i = P_{i-1} + C_{a_i} - C_{r_i}$. This definition is quite general because it covers the case where a new CSP $P_i$ is completely different from the previous one $P_{i-1}$. There are two categories of approaches devised for solving DCSPs: *proactive* and *reactive*. *Proactive* approaches assume that the modeler provides to the system not only the definition of constraints but also information about the possible changes and sometimes about their absolute or relative likelihood of occurrence. On the other hand, *reactive* approaches use no information about the possible directions of the future changes but try to record, when solving, some potentially useful information that will help them anticipate any possible change. Based on the fact that *proactive* approaches try either to produce robust solutions, that will remain valid resisting to many changes, or to produce flexible solutions, that can be easily modified after some changes to produce solutions, we decided not to further investigate them. The reason for our act was that our matchmaking metric requires to solve a series of CSPs that all have to be unsatisfiable. If one is found satisfiable, then the offer is not conforming to the demand. So we did not want to (re)use a solution of a previous CSP in order to find the solution to the next one but to stop after a solution was found. Thus, we only further investigated the *reactive* approaches.

*Reactive* approaches try to reuse as much as possible what has been produced by previous problem solvings in order to face a new problem after a change in the problem definition. There are four kinds of information that are produced when solving a problem: a) *local consistency or inconsistency of partial assignments*; b) *global consistency or inconsistency of partial assignments*; c) *consistency or inconsistency of complete assignments*; and d) *problem consistency or inconsistency*. Moreover, it must be noted that *consistency* and *inconsistency* information exhibits a symmetric behavior with respect to problem relaxation and restriction. For example, consistency information is preserved by problem relaxation while inconsistency is not. Based on the above observations, *reactive* approaches are separated into *solution reuse* and *reasoning reuse* approaches. *Solution reuse* approaches try to reuse the previous solution in order to build the new one based on the assumption that as the previous CSP $P_{i-1}$, that is solved and has the solution $S_{i-1}$, is very close to the new one $P_i$, then a solution $S_i$ of $P_i$ can be searched for in the vicinity of $S_{i-1}$. On the other hand, *reasoning reuse* approaches try to reuse the set $I$ of implied constraints (consequences of the constraints) of CSP $P_{i-1}$, which CSP was found to be consistent or inconsistent, based on the assumption that because $P_{i-1}$ is very close to the new CSP $P_i$ then most of the constraints in $I$ will remain valid in $P_i$. In this way, by recording the constraints in $I$ that will certainly be valid and not producing them again, search efficiency is greatly favored. Based

on the same reasons we eliminated *proactive* approaches, we did not further investigate *solution reuse* approaches. Thus, we finally concentrated on the *reasoning reuse* approaches, which actually perform what we actually want: reuse reasoning information (inconsistency information in our case) in order to decrease solving time of the search-solving procedure and consequently matchmaking time.

Reasoning reuse techniques differ from each other only in the information they use to determine whether an implied constraint in $I$ becomes questionable: *constraint graph*, *constraint justification*, or *constraint explanation*. This results in three families of methods: 1) *graph-based*, 2) *justification-based*, and 3) *explanation-based* methods. The latter two methods are more time and space consuming when producing implied constraints and their justifications or explanations. However, they exploit recorded information in such way that allows fewer constraints in case of *justifications*, and even no constraints in case of *explanations*, to be checked in order to maintain valid implied constraints. Based on the latter reason and on the fact that we discovered a free solving framework, called Choco [Lab00], for normal and dynamic CSPs that uses explanation-based methods (actually a Java version of the Palm system [JB00]), we chose to finally select this CP technique in order to re-implement our algorithms. In result, our proposed $CP_{cond-conf}$ and $CP_{unary}$ matchmaking algorithms were re-implemented in order to use the explanation-based methods of defining and solving CSPs of the Choco framework, thus producing in this way the algorithms $CP_{expl-cconf}$ and $CP_{expl-unary}$ respectively. It must be noted that we used the Choco framework in order to implement all the CP algorithms that we propose, CP and e-CP (explanation constraint programming) ones. In this way, as the explanation-based mechanisms are built on top of the normal CP ones, we could run and experimentally evaluate our matchmaking algorithms in a fair manner, not using different CSP engines for different types of CSPs (i.e normal and dynamic ones). In addition, it must be noted that we did not transform our $CP_{opt}$ algorithm into an e-CP one because of the lack of the appropriate facilities of the Choco framework.

**Achievements** By using CP solving techniques, we have proposed four matchmaking algorithms named $CSP_{unary}$, $CSP_{opt}$, $CSP_{expl-cconf}$, and $CSP_{expl-unary}$. The actual system that will exploit these algorithms will have to choose between them based on user requirements of the user that issues the QoS request. If the user wants a very fast execution time and is not interested much if there is no advanced categorization of results, then the $CSP_{expl-cconf}$ algorithm should be used. On the other hand, if the user wants fast execution time, advanced categorization of results and optimization, then the $CSP_{expl-unary}$ (best if the highest accuracy is required) or the $CSP_{opt}$ (best if n-ary constraints are contained in the QoS demand) algorithms should be used. Table 6.5 summarizes the features (based on the requirements we have set in chapter 4) of the proposed algorithms in order to compare them. In order to make the comparison even more interesting and complete, the $CSP_{conf}$ and $CSP_{cond-conf}$ matchmaking algorithms are also included and analyzed in the table. The "excellent*" phrase denotes that the algorithm that exhibits it has the fastest execution from all compared algorithms.

If this table is compared against table 5.3, it is easy to see that our proposed algorithms outperform all other research approaches. This is mainly due to four reasons: a) use of the

| Research Approach | Semantic QoS Metric Matching | Precision | Recall | Performance | Results Cat/tion | Optim/tion |
|---|---|---|---|---|---|---|
| $CSP_{conf}$ [CMDTT05] | – | excellent | good | excellent | fair | – |
| $CSP_{cond-conf}$ | yes | excellent | excellent | excellent | fair | – |
| $CSP_{unary}$ | yes | excellent | excellent | good | excellent | yes |
| $CSP_{opt}$ | yes | good | excellent | excellent | excellent | yes |
| $CSP_{expl-cconf}$ | yes | excellent | excellent | excellent* | fair | – |
| $CSP_{expl-unary}$ | yes | excellent | excellent | excellent | excellent | yes |

Table 6.5: Comparison of our proposed QoS-based CP-based WS matchmaking algorithms

alignment algorithm; b) high accuracy of results; c) advanced categorization of results; d) return of useful results in case of over-constrained QoS demands. Moreover, the performance of the proposed algorithms is from good to the best. So our QoS-based CP-based WS matchmaking algorithms have achieved their goal as they satisfy all the requirements set in chapter 4. In addition, their evaluation in chapter 7 will verify the claims that are posed now.

### 6.4.2 QoS-based Web Service Selection

In some cases, there will be several QoS offers which will be returned from a QoS-based WS matchmaking algorithm, from the best matchmaking categories to the worst. Therefore, a QoS-based WS selection mechanism is required in order to rank QoS offers according to the requester requirements and present to the WS requester a sorted WS advertisement list, thus helping him to select the best WS according to his QoS needs (usually expressed as a performance to price ratio). For this reason, we have devised and implemented a QoS-based WS selection algorithm, which will be analyzed in the sequel. It must be noted that this algorithm does not need to be executed when the best matchmaking categories contain only one or two results. In addition, our two proposed matchmaking algorithms $CSP_{opt}$ and $MIP_{opt}$ not only produce two categories of results, but they also rank all results of each category. Thus, when one of these two algorithms is executed, the execution of our selection algorithm is not needed.

Before presenting and analyzing our selection algorithm, two things need to be cleared out. The first one is that the selection algorithm checks if a QoS specification contains linear or non-linear constraints and then solves two optimization problems by using a MIP or a CSP engine respectively. So this algorithm is quite generic as it does not only handle unary QoS specifications but also n-ary ones. This feature is not present in all other research approaches except from the one in [CMDTT05] which, however, uses only a CSP engine to solve linear and non-linear QoS specifications. Thus, the latter research approach will exhibit lower performance than our algorithm in case n-ary QoS specifications with only

linear constraints are present. The form of two optimization problems and the usefulness of their results will be analyzed later on. Based on the above reason, from now on we specify that an optimization problem is solved and we do not clarify what type of engine is used to solve it. The second thing to clear out is that it is expected that the semantic alignment of QoS specifications is called before the execution of the selection algorithm. Thus, the QoS specifications will have as many common QoS metrics as possible. If a QoS metric of the QoS demand is not present in a QoS offer, then the contribution of this metric to the score of this offer will be inconsiderable. This will be also understood from the analysis of the proposed selection algorithm.

Our QoS-based WS selection algorithm consists of four sequential phases called: a) *Selection Tree Reduction*; b) *Determination of Requester's Preferred Values*; c) *Production of Min and Max Scores of the Offers*; and d) *Production of Final Score of the Offers*. Before analyzing all these phases, the next subsubsection will be dedicated to explicating the input to this algorithm and the notation used while the last four subsubsections will be dedicated to the four phases of the algorithm.

#### Input and Notations

The input to our selection algorithm is: all QoS offers that need to be ranked, the QoS demand of the WS requester and his WS selection tree. This input is provided by our framework either as a result of a previous matchmaking step or immediately after the QoS-based WS alignment process is executed. In the former case, the matchmaking algorithm may hand over these QoS offers by also placing them in different but "interesting" matchmaking categories (that is *super*, *exact* and/or *partial* categories), so the selection algorithm will have to preserve this categorization by placing each ranked QoS offer in its matchmaking category. In the latter case, the WS requester has asked our framework to only execute the WS selection process and not the matchmaking one.

Now suppose that there are $N$ QoS offers $P_i$ (MIPPs or CSPs), one QoS demand $P_D$ (MIPP or CSP) and one QoS selection tree $S$. Moreover, suppose that the QoS selection tree has $M$ QoS leaf nodes-metrics $Q_j$. As we will see in the sequel, these $M$ QoS metrics play the role of the variables in the production of the scores of the QoS offers. Each $Q_j$ from the $M$ QoS metrics will have a domain of values which has one minimum $Q_j^{min}$ and one maximum value $Q_j^{max}$. Suppose, now, that $P_D$ contains surely constraints on these $M$ QoS metrics. It may also reference some other QoS metrics, but these metrics will not play a very important role in the selection process, especially if they do not appear in any constraint involving any metric from the $M$ particular ones. Concerning the QoS offers $P_i$, they may reference all of the $M$ QoS metrics or not. Let us denote with $X_{ij}$ the constraint problem variable referencing QoS metric $Q_j$ for QoS offer $P_i$. This variable will have the same value type as $Q_j$ and consequently the same min and max values in its domain. However, based on the constraints of $P_i$, its actual min and max values may be more strict so we denote them as $X_{ij}^{min}$ and $X_{ij}^{max}$ respectively. In case where $P_i$ does not reference QoS metric $Q_j$, then the variable $X_{ij}$ will be set to the lowest quality value from the domain of values of

$Q_j$, that is: $X_{ij} = \begin{cases} Q_j^{max}, & \text{for negatively monotonic metrics} \\ Q_j^{min}, & \text{for positively monotonic metrics} \end{cases}$. In this way, this offer's partial score for the QoS metric $Q_j$ will always be zero. In the same way, the QoS demand $P_D$ will contain variable $X_j$ indicating the reference to QoS metric $Q_j$. Each variable $X_j$ will have the same domain of values as metric $Q_j$ but based on the constraints of $P_D$ its min $X_j^{min}$ and max $X_j^{max}$ values may be stricter. Logically, $P_D$ will reference all $M$ QoS metrics.

The QoS selection tree $S$ of the WS requester is a tree with nodes being QoS groups or QoS metrics. This tree should be constructed with the help of a visual tool that will guide the requester in providing the appropriate input. The reason for the existence of this tool is the following main restriction on the tree: in each level $l$ of the tree (except the topmost one), the sum of the weights of the nodes that belong to the same parent must equal to one. The root node also has weight equal to 1. Another restriction is that a QoS metric or group cannot appear twice in the tree. A final restriction is that the leaf nodes of $S$ must only be QoS metrics. This tree can be used by the WS requester in order to create groups of QoS metrics that share a common feature. For example, imagine a tree where the root node is the QoS group named "QoS" having two QoS groups as children named "Domain Dependent" and "Domain Independent". The former children node will have a weight of 0.6 and all of the domain dependent QoS metrics of the QoS demand as children, while the latter one will have a weight of 0.4 and all of the domain independent QoS metrics of the demand as children. In this way, it is denoted that the domain dependent QoS metrics should contribute with a 0.6 factor to the score of each QoS offer while the domain independent ones will contribute with a 0.4 factor to this score. This example is just indicative of what type of nesting this tree will have.

### Selection Tree Reduction Phase

Based on the last paragraph of the previous subsection, we argue that the QoS selection tree $S$ can be reduced to a weighted sum of all the leaf-nodes (i.e QoS metrics) of the tree. We are going to show this reduction by enhancing the example of the previous paragraph and analyzing how the weighted sum is produced. The enhanced example is shown in Figure 6.11, where non-leaf nodes have blue color and leaf nodes have green color. This example is taken from the Traffic Monitoring application domain, analyzed in chapter 3. As it can be seen from this figure, domain-dependent QoS metrics will have a 0.5 impact of the final score while domain-independent ones have a 0.5 impact. The domain-dependent QoS metrics are further categorized into two categories: a) Application Functionality, and b) Application Data related, with the first group having more significance than the second one (0.6 versus 0.4 weight). By taking into account the impact of the domain-dependent QoS metrics group, the overall weight (i.e impact to the final score) for these two subgroups will be $0.5 * 0.6 = 0.3$ and $0.5 * 0.4 = 0.2$ respectively. So as we go down the tree, it is easily understood that we multiply the weight of the parent with the weight of the child node in order to see the overall weight-impact of the child node, until we reach a leaf-node (i.e QoS metric). Thus, leaf nodes will have as overall weight the multiplication of the weight of their

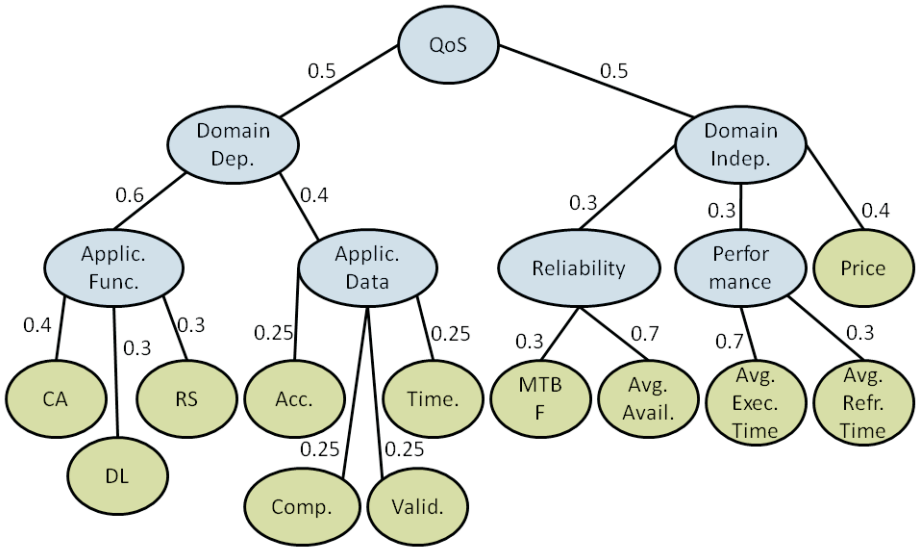



Figure 6.11: Example selection tree for the Traffic Monitoring application domain

ancestors with their local weight.

In this way, let us calculate the overall weights of all the QoS metrics: *CA* (measures the *covered area* QoS property) has overall weight $0.5 * 0.6 * 0.4 = 0.12$, *DL* (measures the *detail level* QoS property) has overall weight $0.5 * 0.6 * 0.3 = 0.09$, *RS* (measures the *routes set* QoS property) has overall weight $0.5 * 0.6 * 0.3 = 0.09$, *Acc.* (measures the *accuracy* QoS property) has overall weight $0.5 * 0.4 * 0.25 = 0.05$, *Comp.* (measures the *completeness* QoS property) has overall weight $0.5 * 0.4 * 0.25 = 0.05$, *Valid.* (measures the *validity* QoS property) has overall weight $0.5 * 0.4 * 0.25 = 0.05$, *Time.* (measures the *timeliness* QoS property) has overall weight $0.5 * 0.4 * 0.25 = 0.05$, *MTBF* (measures the *reliability* QoS property) has overall weight $0.5 * 0.3 * 0.3 = 0.045$, *Avg. Avail.* (measures the *availability* QoS property) has overall weight $0.5 * 0.3 * 0.7 = 0.105$, *Avg. Exec. Time.* (measures the *execution time* QoS property) has overall weight $0.5 * 0.3 * 0.7 = 0.105$, *Avg. Refr. Time.* (measures the *refresh time* QoS property) has overall weight $0.5 * 0.3 * 0.3 = 0.045$, and *Price* (measures the *price* QoS property) has overall weight $0.5 * 0.4 = 0.2$.

Now, another observation that must be made is that, based on the main restriction of the tree and on the way the overall weights on the metrics are calculated, the sum of all the overall weights of the QoS metric equals to 1.0. Indeed, we have that: $0.12 + 0.09 + 0.09 + 0.05 + 0.05 + 0.05 + 0.05 + 0.045 + 0.105 + 0.045 + 0.105 + 0.2 = 1.0$. So we have visually proven that the selection tree can be exploited and reduced to a one level tree consisting only of the QoS metrics referenced and their overall weights. So at the end of this phase, each $Q_j$ from the $M$ QoS metrics (i.e the leaf nodes of the selection tree) will have an overall

(significance) weight $w_j$ apart from the one it had for expressing its significance in the QoS group that it belonged to. In addition, we have that: $\sum_{j=1}^{M} w_j = 1$.

One remark that must be made concerns the QoS attribute of *price*. In some research efforts, we have seen that *price* is not considered as a QoS attribute because it is argued that it is related both to non-functional and to functional characteristics of a service. So in these efforts, service selection is firstly performed by ranking services based on the QoS attributes (that is their values and weights) and then by selecting from the highest ranked services the one with the cheapest price. Even in this case, we believe that our approach can be easily adopted to take into account this restriction. The change that must be made concerns the top of the tree: we have to add one higher node (we can call it "Selection") and make its children the nodes of *QoS* and another (new) node representing the price. In this way, if we put weights on the edges connecting the new root node with its children, then we actually express the trade-off that must be performed between QoS and price in order to select the best service according to the user preferences and his budget.

**Determination of Requester's Preferred Values Phase**

In this phase, the selection algorithm calculates the maximum $X_j^{max}$ and minimum $X_j^{min}$ values constrained by the QoS demand of the WS requester for all $M$ QoS metrics $Q_j$. If the QoS demand $D$ has only unary constraints, then the calculation of the maximum and minimum values of the QoS metrics can be easily performed as follows: if inside $P_D$ there is a constraint $X_j >= a$, then we have that $X_j^{min} = a$; if there is a constraint $X_j <= b$, then we have that $X_j^{max} = b$; if there is a constraint $X_j = c$, then we have that $X_j^{min} = X_j^{max} = c$. However, if $P_D$ contains also binary constraints including some of the $M$ metrics, then the calculation of the maximum and minimum values for all $M$ QoS metrics is a more troublesome process that requires the solving of one maximization and one minimization problem respectively. These two optimization problems and their outcomes will be analyzed in the sequel.

In the same way the preference of a QoS specification is produced in the $MIP_{opt}$ algorithm, we calculate the worst and the best preference of $P_D$ so as to find its worst and best solutions respectively. In this way, we actually make the assumption that the worst and best solutions will contain the worst and best values of all $M$ QoS metrics respectively. The worst solution is calculated by solving a minimization problem (CSOP or MIPOP) which contains all the constraints of $P_D$ and uses the following formula-objective: $p_D^{min} = \min \sum_{X_j}(w_j \cdot uf_{X_j}(X_j))$, where $w_j$ is the weight of the QoS metric $Q_j$ to which variable $X_j$ is mapped and $uf_{X_j}()$ is the utility function of variable $X_j$. Similarly, the best solution is calculated by solving a maximization problem which contains all the constraints of $P_D$ and uses the following formula-objective: $p_D^{max} = \max \sum_{X_j}(w_j \cdot uf_{X_j}(X_j))$. One thing that must be cleared out is which utility function to use for the $M$ QoS metrics. Based on the assumption that QoS metrics are positively or negatively monotonic, we use two types

of utility functions taken from the research work of Zeng et. al [ZBD+03]:

$$uf_{X_j} = \begin{cases} \frac{max_j - X_j}{max_j - min_j}, \text{ if } max_j - min_j \neq 0 \\ 1, \text{ if } max_j - min_j = 0 \end{cases} \quad (6.6)$$

$$uf_{X_j} = \begin{cases} \frac{X_j - min_j}{max_j - min_j}, \text{ if } max_j - min_j \neq 0 \\ 1, \text{ if } max_j - min_j = 0 \end{cases} \quad (6.7)$$

The values of $max_j$ and $min_j$ can be computed from the QoS offers (if all offers contain unary constraints) as: $max_j = \max\{X_{ij}^{max}\}$ and $min_j = \min\{X_{ij}^{min}\}$ where the values of $X_{ij}^{max}$ and $X_{ij}^{min}$ are computed in the same way we have described for unary QoS demands. However, if the QoS offers contain also binary constraints, instead of solving two optimization problems for them in order to find the $X_{ij}^{max}$ and $X_{ij}^{min}$ values, we compute $max_j$ and $min_j$ as: $max_j = Q_j^{max}$ and $min_j = Q_j^{min}$ respectively. The first type of utility functions (Eq. 6.6) is for negatively monotonic QoS metrics (e.g measuring *price*) while the second type (Eq. 6.7) is for positively monotonic QoS metrics (e.g measuring *availabity*). So, in the end, after solving the two optimization problems, we will have computed the minimum preference $p_D^{min}$ of the worst solution $(X_{11}, X_{12}, \ldots, X_{1M})$ and the maximum preference $p_D^{max}$ of the best solution $(X_{21}, X_{22}, \ldots, X_{2M})$ of $P_D$. Thus, in this way we will eventually have that $X_j^{min} = \min(X_{1j}, X_{2j})$ and $X_j^{max} = \max(X_{1j}, X_{2j})$ for every $j$ in $[1, M]$.

### Production of Min and Max Scores of the Offers Phase

The goal of the WS selection algorithm is to produce a score for each QoS offer $P_i$ in order to present an ordered list of WSs back to the WS requester. We believe that the score of each offer should consist of two parts: one part dedicated to computing the worst score of the offer and another part calculating the best score of the offer. The reason for this will be given in the next subsubsection. Similarly to the previous paragraph where the worst and best preference of a n-ary-based QoS demand were calculated, we calculate the the worst $p_i^{min}$ and best $p_i^{max}$ preferences of offer $P_i$ by the following formulas: $p_i^{min} = \min \sum_{X_{ij}} (w_j \cdot uf_{X_{ij}}(X_{ij}))$ and $p_i^{max} = \max \sum_{X_{ij}} (w_j \cdot uf_{X_{ij}}(X_{ij}))$, where where $w_j$ is the weight of the QoS metric $Q_j$ to which variable $X_{ij}$ is mapped and $uf_{X_{ij}}()$ is the utility function of variable $X_{ij}$. If $P_i$ contains only unary constraints, then the min $X_{ij}^{min}$ and max $X_{ij}^{max}$ values of all metrics $Q_j$ can be computed very easily by the process we analyzed in the previous subsubsection for QoS demands. In this way, the computation of the above formulas takes $O(1)$ time as we just instantiate each $X_{ij}$ variable with its worst $X_{ij}^{min}$ and best $X_{ij}^{max}$ value and the formulas are expressed as: $p_i^{min} = \sum_{X_{ij}} (w_j \cdot uf_{X_{ij}}(X_{ij}^{min}))$ and $p_i^{max} = \sum_{X_{ij}} (w_j \cdot uf_{X_{ij}}(X_{ij}^{max}))$. Otherwise, if $P_i$ contains also n-ary constraints, then two optimization problems, constructed from $P_i$, must be solved with the first one using the objective $p_i^{min}$ and the other one using the objective $p_i^{max}$.

Until now, the only unknown component that was not analyzed is the utility function $uf_{X_{ij}}$ of the $X_{ij}$ variables. We intentionally have left this component uncommented because

it is the most important one. Our approach in determining this utility function resembles the one of Toma et. al. [WVKT06] as we take into account the values the WS requester provides for the $M$ QoS metrics (of the QoS selection tree). However, what differentiates our approach from theirs is that we respect the min $X_j^{min}$ and max $X_j^{max}$ values the requester provides for these metrics $Q_j$, when the QoS metric is "sensitive", and only the min or max values for "insensitive" positively or negatively monotonic QoS metrics respectively. Let's stick for a moment to the "sensitive" QoS metrics. Suppose we have a "sensitive" QoS metric $Q_j$ and that the WS requester has provided a min $X_j^{min}$ and max $X_j^{max}$ value for it. Our idea is to choose a utility function that "respects" the user given limits by giving a preference between $a$ and 1.0 to values of variables $X_{ij}$ that are between them. Moreover, the distance $x = X_j^{max} - X_j^{min}$ between the min and max values plays an important role as it determines that for values of $X_{ij}$ in sets $\left[\max\left(X_j^{min} - x, Q_j^{min}\right), X_j^{min}\right]$ and $\left[X_j^{max}, \min\left(X_j^{max} + x, Q_j^{max}\right)\right]$ the preference will fall between $a$ and $\max\left(a - (1-a), 0\right)$. By combining $a$ and $x$ we get the big picture: by moving away from the user-provided limits of a metric, the preference of a value will decrease from $a$ with the amount $1 - a$ so many times as approximately is the distance of the value from the nearest limit divided by $x$.

For example, suppose that we have $a$=0.6, $x = 10$, $X_j^{min} = 30$, $X_j^{max} = 40$, $Q_j^{min} = 10$, and $Q_j^{max} = 100$. Then: if $X_{ij}$ is in $[30, 40]$, its preference will vary between 0.6 and 1.0; if $X_{ij}$ is in $[20, 30]$ or $[40, 50]$, its preference will vary between 0.2 and 0.6; if $X_{ij}$ is in $[10, 20]$ or $[50, 60]$, its preference will vary between 0.0 and 0.2; all other values of $X_{ij}$ will have their preference equal to 0.0. In the previous example, if we choose $a$=0.5, then the values of $X_{ij}$ in the sets $[20, 30]$ or $[40, 50]$ will get a preference between 0.5 and 0.0 while all values below 20 or above 50 will have preference equal to 0.0. So high values of $a$ that are greater than 0.5 enable more distant values (w.r.t to distance $x$) of $X_{ij}$ to get a non-zero preference. Of course, if $a$ equals to 1.0 then all values of $X_{ij}$ will have a preference of 1.0. On the other hand, small values of $a$ less than 0.5 enable only values of $X_{ij}$ that have less distance than $x$ from the nearest user-provided limits to get a non-zero preference value. If $a$ equals 0.0, then only the values of $X_{ij}$ that are between the user-provided limits will get a non-zero preference. In the same way as $a$, small values of $x$ will decrease the possibility that distant values of $X_{ij}$ will get a non-zero preference, while big values of $x$ will increase it. In the special case that $x$ equals to 0, then only the value of $X_{ij}$ that is equal to $X_j^{min} = X_j^{max}$ will have a preference of 1.0 while all other values will have a zero preference.

Based on the above analysis, we now define the utility function $uf_{X_{ij}}$ of very variable

$X_{ij}$ as:

$$uf_{X_{ij}} = \begin{cases} a_j + \left(\frac{X_{ij}-min_j}{max_j-min_j}\right) \cdot (1-a_j), \text{ if } min_j \neq max_j \ \& \ Q_j \uparrow \ \& \ min_j \leq X_{ij} \leq max_j \\ a_j + \left(\frac{max_j-X_{ij}}{max_j-min_j}\right) \cdot (1-a_j), \text{ if } min_j \neq max_j \ \& \ Q_j \downarrow \ \& \ min_j \leq X_{ij} \leq max_j \\ \max\left(a_j - \left(\frac{X_{ij}-min_j}{max_j-min_j}\right) \cdot (1-a_j), 0.0\right), \text{ if } min_j \neq max_j \ \& \ X_{ij} > max_j \\ \max\left(a_j - \left(\frac{max_j-X_{ij}}{max_j-min_j}\right) \cdot (1-a_j), 0.0\right), \text{ if } min_j \neq max_j \ \& \ X_{ij} < min_j \\ 1.0, \text{ if } min_j = max_j = X_{ij} \\ 0.0, \text{ if } min_j = max_j \neq X_{ij} \end{cases}$$

where the symbol $Q_j \uparrow$ denotes that the metric $Q_j$ is positively monotonic while the symbol $Q_j \downarrow$ denotes that metric is positively monotonic. As it can be seen from this type, we define $a_j$ for very QoS metric $Q_j$. Actually, these values will be provided either by the WS requester or by a domain modeler that has the experience and the exact knowledge of all entities of his application domain. Until now, we have not modeled these values in OWL-Q so we do not actually capture them but we hardwire them into the code of the selection algorithm. The modeling of these values will be performed in the near future. We have deliberately left the $min_j$ and $max_j$ values undefined as they vary according to the sensitivity of the QoS metric $Q_j$. If $Q_j$ is "sensitive", then we have that: $min_j = X_j^{min}$ and $max_j = X_j^{max}$ so we respect both of the limits of the WS requester. If $Q_j$ is "insensitive", then we respect only one of the limits of the requester, that is we have: $min_j = X_j^{min}$ and $max_j = Q_j^{max}$ for positively monotonic metrics, and $max_j = X_j^{max}$ and $min_j = Q_j^{min}$ for negatively monotonic metrics. So the second (non interesting) limit gets so high or low until the max or min value of the domain of the QoS metric.

### Production of Final Score of the Offers Phase

The overall preference or score $p_i$ of an offer $P_i$ based on a demand $P_D$ is given by the following formula: $p_i = a \cdot p_i^{min} + b \cdot p_i^{max}$, where $0 \leq a, b < 1$ and $a + b = 1$. This formula states that the overall score of an offer depends on the percentage $a$ of the minimum score of the offer and on the percentage $b$ of the maximum score of the offer.

### Time Complexity

Let us now examine the time complexity of our QoS-based WS selection algorithm. We assume that the reduction of the selection tree phase is trivial so we do not take it into account for computing the time complexity. If we have only unary QoS offers and demands, then we will perform $2 \cdot (N+1) \cdot M$ operations for computing the min and max values of the $X_j$ and $X_{ij}$ variables that take $O(1)$ time and $3 \cdot N$ operations for computing the score of each offer that take also $O(1)$ time. Thus, in this case which is the best, the time complexity will be $\Theta(N \cdot M)$. In the worst case, all QoS specification will be n-ary so we

| Research Approach | User Preference Spec | User Constraints | Normalization | QoS Grouping |
|---|---|---|---|---|
| *Sel* | good | yes | yes | yes |

Table 6.6: Satisfaction degree of the QoS-based WS selection criteria by our algorithm

will have to solve $2 \cdot N + 2$ optimization problems that take time at most $T$ and we will have to compute $N$ final scores in time $O(1)$. So the time complexity of the worst case will be $\Theta(N \cdot T)$. It is quite obvious that solving optimization problems is more difficult than computing simple weighted formulas so non-unary QoS specifications significantly increase the time complexity of the selection algorithm.

#### Achievements

Although most QoS-based WS selection algorithms are interested in the worst score that can be given by an offer, we believe that this is not enough; we have to pick among the same "worst-score" offers the one that gives the best of the maximum-best scores. This feature is considered only in our algorithm. Another advantage of our selection algorithm is that it takes into account both of the limits of the WS requester and not only one value. This is very important, especially in highly dynamic application domains where just one value of a QoS metric cannot be guaranteed but only a range of values. Finally, one of the biggest advantages of our algorithm is that it not only deals with unary QoS specifications but also with n-ary ones. This is something that is not offered by any other selection algorithm.

Let us denote with *Sel* our proposed QoS-based WS selection algorithm. Table 6.6 summarizes the features of our algorithm according to the criteria set in chapter 5 and section 5.6. Apart from the above unique advantages of our algorithm, if we compare the contents of table 6.6 against the contents of table 5.4 we can surely see that our algorithm satisfies all the requirements-criteria set in chapter 5 and is the only one achieving this.

## 6.5 Conclusions

The three main parts of this chapters were dedicated to the analysis of the three main contributions of this thesis which are the following: **a)** a semantic, rich and extensible QoS description model for WSs; **b)** a semantic alignment algorithm for QoS-based WS specifications; **c)** development of different QoS-based WS discovery algorithms optimized according to the solving technique used and the user requirements.

In the first part of the chapter, OWL-Q, an upper-level ontology consisting of many facets – each capturing a specific aspect of QoS-based WS description – was described and analyzed. Then, the reasons why this language was extended with SWRL rules was provided. Finally, the first part concluded by explicating why our ontology language realizes a semantic, rich

and extensible QoS description model and which of the requirements set in chapter 4 it satisfies.

The second part of this chapter explains why the alignment of QoS-based WS specifications is necessary for ensuring the accuracy of the QoS-based WS matchmaking and selection algorithms. After this justification, a complete alignment algorithm of QoS-based WS specifications is provided and analyzed in text, pseudo-code samples and figures. The main core, on top of which the alignment algorithm is built, is also analyzed, which constitutes the semantic QoS metric matching algorithm that is able to infer if two QoS metrics (that are part of different QoS specifications) are equivalent.

Finally, the third part of this chapter was devoted to the description and theoretical analysis of the QoS-based WS matchmaking we have implemented and experimentally evaluated. We have actually provided and implemented various versions of different algorithms that use different techniques (CP or MIP), have different performance and accuracy and provide results of different granularity or usability. Last but not least, it must be remarked that we have also proposed a novel WS selection algorithm that considers not only equality but also inequality constraints of the WS requester, it deals with n-ary QoS specifications and it takes into account both the best and worst performance of each QoS offer.

# Chapter 7

# Evaluation

This chapter starts by explaining the prototype system we are currently implementing for supporting the semantic QoS-based description, alignment and discovery of WSs. Then, the remaining of the chapter is dedicated to experimentally evaluating all the algorithms we have devised and implemented. We focused our study mainly in the QoS-based WS matchmaking algorithms and conducted many series of experiments in order to evaluate their performance and accuracy. The majority of the results were in concert with the theoretical analysis we performed in chapter 6 for these algorithms.

## 7.1 Implementation

We are currently in the development phase of our QoS-based WS discovery engine that is going to support the semantic QoS-based WS description, alignment and discovery. This engine uses three main assisting components: a) the Pellet reasoner (`http://pellet.owldl.com`) for ontology reasoning, b) the XpressMP system for solving MIPPs having linear constraints and c) the Choco CSP engine for solving (e)CSPs with non-linear constraints, while the Java programming language is used as a bridge between them. Pellet is chosen because it is one of the best three ontology reasoners supporting the tasks of ontology validation and reasoning, OWL 1.1 data-type reasoning (needed for reasoning on QoS metric data-types) and partial SWRL inferencing (for rules support and inferencing). XpressMP is one of the best commercial engines for solving LP and MIP programs while it also supports constraint solving through its interface with the Kalis constraint solving engine developed by Artelys SA. Finally, the Choco system consists of both a CSP and a eCSP engine so it is ideal for using any of our developed and implemented CP-based matchmaking algorithms.

The architecture of the system under development is shown in Fig. 7.1. In the sequel, an overview of the functionality of each component of our system is provided by analyzing the scenario where a WS provider wants to publish the QoS offer of his WS while a WS

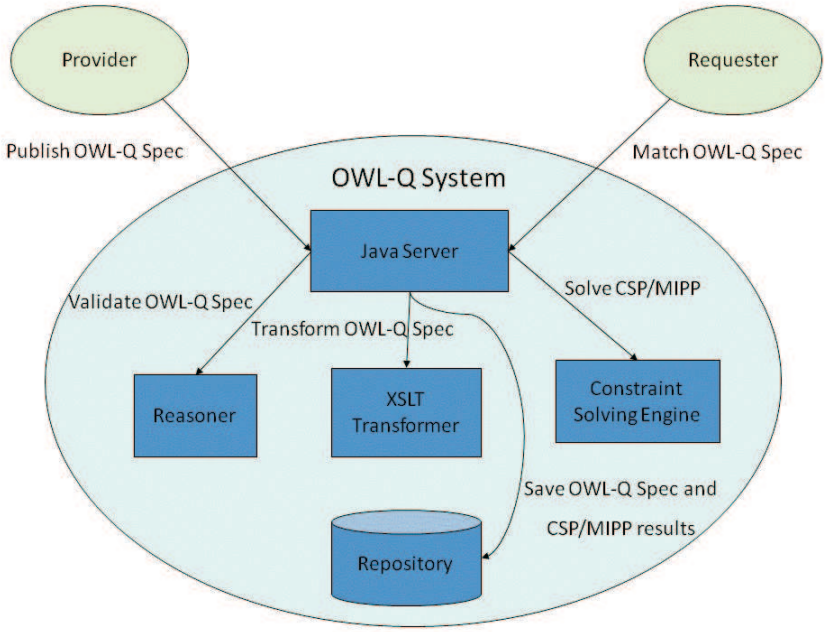



Figure 7.1: QoS-based WS Discovery Engine.

requester wants to find available QoS offers based on his request.

**Publication.** The *WS Provider* (WSP) describes his QoS offer in OWL-Q and sends it to the *Java Server*, the main component of our system. The *Java Server* (JS) sends this offer to the *Reasoner* (RS) in order to validate its (ontological) consistency against the common OWL rules and the accompanying SWRL rules. If the offer is not consistent, then an error message is returned to the WSP. Otherwise, JS aligns the OWL-Q offer according to the contents of the *Metric Store* (MS), which is part of the *Repository* (R), and with the help of the *XSLT Transformer* (XSLT) it transforms it into an MIPP or (e)CSP. This MIPP or CSP is send to the appropriate Constraint Solving Engine (CSE) (MIP or CP-based) in order to find if it is satisfiable/feasible. If not, then an appropriate error message is returned to the WSP. If yes, then the aligned OWL-Q offer and its accompanying CSP/MIPP are stored at R and an appropriate positive message is returned to the WSP.

**Matching.** The *WS Requester* (WSR) sends his OWL-Q demand to the JS, which follows exactly the same procedure as above (ontology reasoning, global alignment and consistency/feasibility checking). If everything is alright, then the demand and its CSP/MIPP are stored at R for caching purposes. Then the remaining four processes of local alignment, matchmaking, (possibly) constraint relaxation and selection are executed at JS, which cooperates with the CSE in order to solve appropriate CS(O)Ps/MIP(O)Ps and evaluate the returned results. In the end, the user gets an ordered list of matching OWL-Q offers or a list of partial matches along with a suggestion or a matching failure message.

## 7.2 Evaluation of QoS-based Web Service Matchmaking Algorithms

Before analyzing the series of experiments we conducted on our algorithms and their results, our first experimental evaluation had the goal to prove that the MIP solving technique is better than CP for solving only linear constraints. This was one of our main arguments so it had to be backed up by an experimental evaluation. Our experimental evaluation was actually based on the experimental evaluation of Cortés et. al. [CMDTT05] in which the performance of consistency checking and conformance checking of linear CSPs was evaluated. The next two subsections reveal these two evaluations starting from the one of Cortés et. al. [CMDTT05] and ending with ours.

For evaluating our QoS-based WS matchmaking algorithms we used a generic testing framework that produces randomized QoS-based WS specifications in a control way so as to be able to test the performance and accuracy of our algorithms. This framework accepts as parameters the matchmaking algorithms to be evaluated and can be altered easily so as to change what kinds of specifications it produces as it provides them as input to the matchmaking algorithms. We conducted four series of experiments based on this testing framework. The first three series of experiments evaluated the MIP algorithms we have implemented ($MIP_{conf}$, $MIP_{cond-conf}$, $MIP_{unary}$, and $MIP_{opt}$) while the last one evaluated the (e)CP-based ones ($CP_{conf}$, $CP_{cond-conf}$, $CP_{expl-cconf}$ and $CP_{expl-conf}$). These four series of experiments are presented in the last four subsections of this chapter.

### 7.2.1 Cortés et. al. experimental evaluation

Cortés et. al. [CMDTT05] carried out a series of experiments in order to experimentally analyze the behavior of their CP-based matchmaking algorithm focusing on CSPs (transformed from QoS-based WS specifications) containing linear constraints. The experiments were implemented in the J# programming language while the powerful CP engine 'ILOG CP' was used for solving CS(O)Ps. Each experiment was carried out 30 times in order to reduce interferences in the form of outliers from the host's operating system or network and the average solving time was registered.

In these experiments, the worst case that was considered was a CSP whose constraints could reduce the *initial search space* as less as possible. For this reason, the following constraint factors were taken into account: **a)** The number $N$ of variables of the CSP along with their domains (*small*-256 values, *medium*-65536 values, *large*-$4.2 \cdot 10^9$) that determine the initial search space; **b)** the number $C$ of constraints of the CSP; **c)** The arity $A$ of each constraint i.e. the number of variables involved.

In the first series of experiments, the performance of consistency checking was measured for both consistent and inconsistent CSPs. In order to keep the initial search space as big as possible, the tested CSPs contained $C_{min} = N - A + 1$ constraints in order to bind all variables. Figure 7.2 shows the different CSPs created for each value of $A$ and $N$ for the first (consistency checking) and second (conformance checking) series of experiments.

Consistency and Non-Consistency Checking:

| A=1 | A=2 | A=3 | | A=N |
|---|---|---|---|---|
| $X_1>10$ | $X_1+X_2>10$ | $X_1+X_2+X_3>10$ | | $X_1+X_2+ \dots +X_N>10$ |
| $X_2>10$ | $X_2+X_3>10$ | $X_2+X_3+X_4>10$ | | |
| : | : | : | ... | |
| $X_N>10$ | $X_N+X_{N-1}>10$ | $X_{N-2}+X_{N-1}+X_N>10$ | | |
| $X_1<10$ | $X_1+X_2<10$ | $X_1+X_2+X_3<10$ | | $X_1+X_2+ \dots +X_N<10$ |

Conformance Checking/Solving:

| |
|---|
| A=1: $X_1>10 \wedge X_2>10 \wedge \dots \wedge X_N>10 \wedge \neg(X_1>10 \wedge X_2>10 \wedge \dots \wedge X_N>10)$ |
| A=3: $X_1+X_2+X_3>10 \wedge X_2+X_3+X_4>10 \wedge \dots \wedge X_{N-2}+X_{N-1}+X_N>10 \wedge \neg (X_1+X_2+X_3>10)$ |
| ... |
| A=N: $X_1+X_2+ \dots +X_N>10 \wedge \neg (X_1+X_2+ \dots +X_N>10)$ |

Figure 7.2: Actual CSP Model for every value of $A$ and $N$.

For example, a consistent CSP with $N = 3, A = 1$ contained the following constraints: $x_1 > 10; x_2 > 10; x_3 > 10$ while the corresponding non-consistent CSP also contained the constraint: $x_1 < 10$. As another example, if $N = 4, A = 3$, then a consistent CSP contained the constraints: $x_1 + x_2 + x_3 > 10; x_2 + x_3 + x_4 > 10$ while the inconsistent one also contained the constraint: $x_1 + x_2 + x_3 < 10$. For consistent CSPs having variables of small domains with $N$ ranged from 1 to 1500 and $A$ ranged from 1 to 10, the results revealed that the performance of consistency checking presents a slightly linear increasing behavior. For consistent CSPs having large-domain variables with $N$ ranging from 1 to 30 and $A$ ranging from 1 to 10, the results revealed that for arities above 4 execution time gets considerably worse. For non-consistent CSPs having small-domain variables with $N$ ranging from 1 to 20 and $A$ ranging from 1 to 10, the results showed that consistency checking performance presented an exponential behavior. The overall observation from this series of experiments was that consistency checking is acceptable for QoS specifications having variables with small-domain variables and constraints with arity below 7.

In the second series of experiments, the performance of conformance checking of both conforming and non-conforming CSPs was measured. CSPs were created according to the way it is presented in Figure 7.2. For example, if $N = 3, A = 1$, a conforming CSP contained the constraints: $x_1 > 10; x_2 > 10; x_3 > 10; not(x_1 > 10 \ \& \ x_2 > 10 \ \& \ x_3 > 10)$ while a non-conforming CSP contained the constraints: $x_1 > 10; x_2 > 10; x_3 > 10; not(x_1 < 10 \ \& \ x_2 > 10 \ \& \ x_3 > 10)$. For $A = 1$ and $N$ ranging from 1 to 20 and small and medium domains, the performance of conformance checking presented a slightly increasing behavior. However, with $A = N$ and ranging from 1 to 10, the performance of conformance checking presented

an exponential increasing behavior for medium domains and arity above 3 and for small domains and arity above 6. Another important observation is that conforming CSPs take more time for solving compared to non-conforming CSPs.

The authors of this experimental evaluation did not explain why non-consistent linear CSPs and consequently conforming linear CSPs (actually consisting of a set of non-consistent CSPs) exhibit an exponential behavior. However, this behavior can be easily explained. The default solving technique in the ILOG's CP engine, that is apparently used in these experiments, is performing *bounds consistency* for *constraint propagation* and then *tree search*. Bounds consistency makes arc-consistent the bounds on the domain of each variable and is weaker than *full arc consistency*. However, for the type of constraints of the conducted experiments these two constraint propagation techniques are equivalent and have exponential complexity with respect to the arity of the constraints. Let us give an example in order to assist the comprehension of our explanation. Assume that there are two constraints in a non-consistent CSP: $x_1+x_2+\ldots+x_n > 10$ and $x_1+x_2+\ldots+x_n < 10$ and all variables take values from the domain $[0, 65365]$. Bounds consistency will start from the first variable $x_1$ and its upper bound (65365) and will seek support in both of the constraints. As this value is not supported by the second constraint, it will be deleted. Then bounds consistency will continue with the new upper bound (65534) of the $x_1$ variable and it will continue to delete values until it goes to the value of 9 for this variable. This procedure is then repeated for the remaining variables. So it is very easy to understand that bounds consistency exhibits an exponential behavior with respect to the arity of the constraints.

### 7.2.2 Experimental evaluation of conformance versus feasibility

We carried out a series of experiments in order to experimentally analyze the behavior of the conformance matchmaking algorithm $CP_{conf}$ and its MIP counterpart $MIP_{conf}$ focusing on constraint problems (CSPs and MIPPs respectively) containing linear constraints. The experiments were implemented in the Matlab programming language while the powerful CP engine 'Xpress-Kalis' was used for solving CS(O)Ps and the MI(Q)P engine 'Xpress-Optimizer' was used for solving MIPPs. Each experiment was carried out 20 times in order to reduce interferences in the form of outliers from the host's operating system or network and the average solving time was registered.

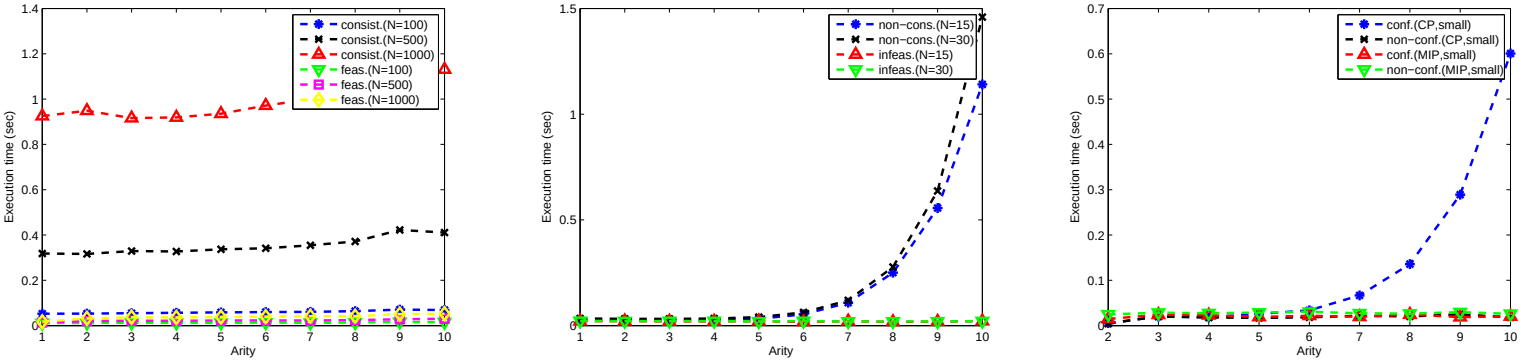
As far as constraint factors were concerned, we had the same as the ones used in Cortés et. al. evaluation process. In addition, the $C_{min}$ criterion for CSPs was also preserved. However, in order to compare the CP and MIP approaches in a fair manner, we transformed the 'less than' and 'greater than' operators to the inequalities $\leq$ and $\geq$ respectively. For example, if $N = 3, A = 1$, then a consistent CSP/MIPP contained the constraints: $x_1 \geq 10; x_2 \geq 10; x_3 \geq 10$ while an inconsistent one contained also the constraint: $x_1 \leq 9$. As another example, if again $N = 3, A = 1$, then a conforming CSP contained the constraints: $x_1 \geq 10; x_2 \geq 10; x_3 \geq 10; not(x_1 \geq 10 \;\&\; x_2 \geq 10 \;\&\; x_3 \geq 10)$ while a non-conforming one contained the constraints: $x_1 \geq 10; x_2 \geq 10; x_3 \geq 10; not(x_1 \geq 10 \;\&\; x_2 \geq 10 \;\&\; x_3 \leq 9)$.

For (in)consistency-(in)feasibility checking, we performed four experiments. In the first experiment, we had consistent CSPs (and feasible MIPPs) with variables of small domain,

$N$ was 100, 500 and 1000 and $A$ ranged from 1 to 10. As can be seen from Figure 7.3(a), feasibility checking for all values of $N$ exhibited a stable behavior and this was also the case for consistency checking with $N = 100$. But consistency checking for the other values of $N$ exhibited a slightly linear increasing behavior. Moreover, the MIP-based approach outperformed the CP-based approach especially for the two biggest values of $N$. In the second experiment, we had inconsistent CSPs (and infeasible MIPPs) with variables of small domain, $N$ was 15 and 30 and $A$ ranged from 1 to 10. As can be seen from Figure 7.3(b), feasibility checking exhibited a stable behavior while consistency checking exhibited an increasing exponential behavior. In addition, for arities below 6 both approaches had approximately the same performance. In the third experiment, we had consistent CSPs (and feasible MIPPs) with variables of medium domain, $N$ was 10, 20 and 30 and $A$ ranged from 1 to 10. As can be seen from Figure 7.4(a), feasibility checking for all values of $N$ exhibited a stable behavior and this was also the case for consistency checking. However, the MIP-based approach greatly outperformed the CP-based approach. In the fourth experiment, we had inconsistent CSPs (and infeasible MIPPs) with variables of medium domain, $N$ was 15 and $A$ ranged from 1 to 12. As can be seen from Figure 7.4(b), feasibility checking exhibited a stable behavior while consistency checking exhibited an increasing exponential behavior. In addition, for all arities the MIP-based approach greatly outperformed the CP-based one. So the conclusion from this series of experiments was that feasibility checking is better even for small domains and greatly outperforms consistency checking. In addition, feasibility checking exhibits the same performance for both feasible and infeasible MIPPs while consistency checking exhibits stable performance for consistent CSPs and exponential behavior for inconsistent CSPs.

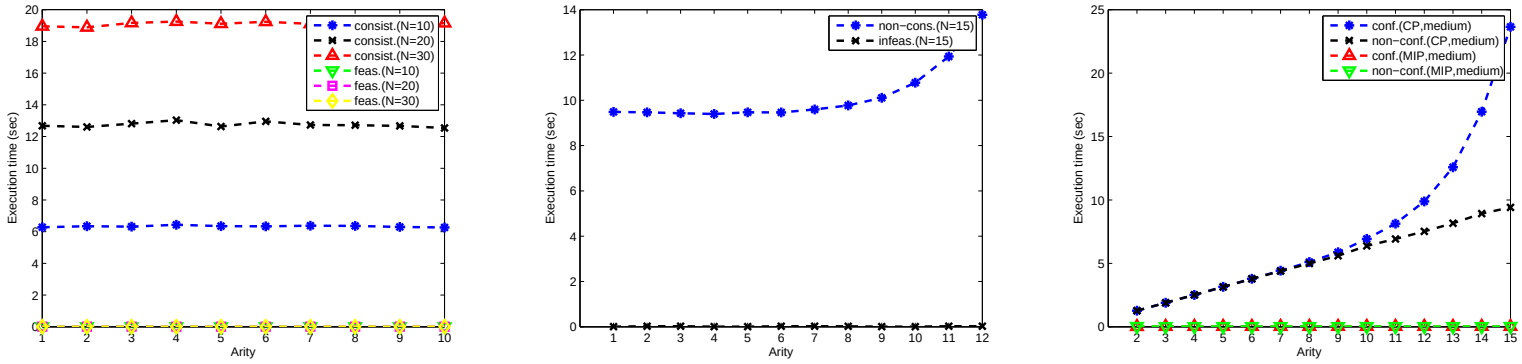
For CP and MIP-based (non)conformance checking, we performed two experiments. In the first experiment, we had both conforming and non-conforming CSPs (and MIPPs) of only small domains while $A = N$ and ranging from 1 to 10. As can be seen from Figure 7.3(c), CP-based conformance checking presented an exponential behavior for arities above 6 and conforming CSPs while MIP-based conformance checking presented a stable behavior for all arities and CSPs. Moreover, both $MIP_{conf}$ and $CP_{conf}$ algorithms had the same performance for non-conforming CSPs. In the second experiment, we had both conforming and non-conforming CSPs (and MIPPs) of medium domain while $A = N$ and ranging from 1 to 15. As can be seen from Figure 7.4(c), CP-based conformance checking presented an exponential behavior for arities above 8 and conforming CSPs while MIP-based conformance checking presented a stable behavior for all arities and CSPs. Moreover, while the $CP_{conf}$ algorithm presented a stable performance for non-conforming CSPs, its performance was greatly outperformed by the $MIP_conf$ algorithm. So the conclusion from this experiment was that the MIP-based conformance checking approach is better than the CP-based one.

Thus, based on the above experiments and their results, it can be definitely stated that MIP is better than CP for QoS-based WS specifications with linear constraints. The main reason is that CP has a hard time solving inconsistent or conforming CSPs (it presents exponential behavior) while MIP solves any kind of MIPP (infeasible, conforming, etc.) with the same stable and excellent performance. Two main remarks must be made. Firstly,




(a) Consistency checking (b) Inconsistency checking (c) Conformance checking

Figure 7.3: Experiments results for small domains




(a) Consistency checking (b) Inconsistency checking (c) Conformance checking

Figure 7.4: Experiments results for medium domains

the 'Xpress-Kalis' CP engine performs bound consistency so this is the main reason (as explained in the previous subsection) of the exponential behavior of CP. Of course, even if a better technique was used, solving time would be again greater for CP with respect to MIP. Secondly, MIP uses firstly the linear relaxation of a MIPP in order to solve it so it quickly detects the inconsistency or conformance of linear MIPPs. This is the main reason why MIP is much more quicker than CP in these type of CSPs/MIPPs.

### 7.2.3 Randomized Evaluation of MIP-based Matchmaking Algorithms

In this subsection, we present and analyze the results of a set of experiments carried out in a randomized way for testing the performance and accuracy of our MIP-based QoS-based WS matchmaking algorithms.

### Experimental Setup

The algorithms and their experiments were implemented in Matlab. Matlab was selected as it offers a user-intuitive modeling, building, runtime and debugging environment and convenient facilities for creating graph files and saving them as images. However, its optimization package (providing optimization solving routines) could not solve MIP problems so we decided to use a free evaluation version of DashOptimization's XPressMP product and Java as the 'glue' language. The tests were performed on a computer with Microsoft Windows XP Home as the operating system, a 1.8Ghz AMD Athlon microprocessor and 512 megabytes of RAM.

In each experiment a series of tests were executed. The number of these tests depended on the step of increase on one tuning parameter (e.g. a value from 10 to 50 with step 10 translates to five tests) while the other tuning parameters had specific values. This type of parameters was used for constructing a specific random input to the matchmaking algorithms. Each test was executed 30 times producing 30 values for each measured metric for every algorithm. From these 30 values, only the average or minimum was registered depending on the nature of the measured metric. So in the end of each test, every algorithm had exactly one value registered for each metric.

**Tuning parameters** Tuning parameters were used for constructing a specific random input, common to every matchmaking algorithm at each run of a test. The tuning parameters taken into account were the following:

- *adscn*: this parameter indicates the number of QoS offers that have to be constructed from a specific random QoS demand.
- *matchper*: this parameter indicates the percentage of QoS offers that are conforming to the random QoS demand. In addition, it indirectly indicates the percentage of non-conforming QoS offers. There will be a total of $m = \text{round}\,(adscn \cdot matchper)$ matching QoS offers and a total of $nm = adscn - m$ non-matching QoS offers in the randomly constructed input.
- *bmper*: this parameter indicates the percentage of *super* offers among all matching offers. In addition, it indirectly indicates the percentage of *exact* offers among all matching offers. There will be a total of: $b = \text{round}\,(bmper \cdot m)$ *super* offers and a total of $e = m - b$ *exact* offers in the randomly constructed input. This parameter also controls the *recall* accuracy metric of the $MIP_{conf}$ algorithm as it expresses the percentage of 'false negatives' that will be produced from this algorithm.
- *partper*: this parameter indicates the percentage of *partial* offers among all non-conforming QoS offers. It also indirectly indicates the percentage of *fail* offers among all non-conforming QoS offers. There will be a total of $p = \text{round}\,(partper \cdot nm)$ *partial* offers and a total of $f = nm - p$ *fail* offers in the randomly constructed input.
- *arity*: this parameter indicates the arity of the constraints i.e. the max number of variables participating in the constraints set.

- *metrcn*: each randomly produced QoS-based WS specification contains *metrcn* number of metrics and $4 \cdot metrcn$ constraints. For every QoS specification, each metric has 2 constraints indicating the limits of its type (e.g. for a real valued metric in [1,3] we have the constraints: $x \geq 1$ and $x \leq 3$) and 2 constraints indicating the limits of the specification (e.g. for the same metric we can have $x \leq 2$ and $x \geq 2$).

- *realper*: this parameter indicates the percentage of real valued metrics among all metrics. In addition, it indirectly indicates the percentage of integer valued metrics among all metrics. There will be $r = \text{round}\,(realper \cdot metrcn)$ real valued metrics and $i = metrcn - r$ integer valued metrics in each randomly constructed QoS-based WS specification. Each real valued metric takes just one of the three available real types randomly. These types are: $[0.0, 1.0]$, $[0.0, 100.0]$ and $[0.0, 1000.0]$.

- *sintper*: this parameter indicates the percentage of short int valued metrics (values in [0,255]) among all metrics. In addition, it indirectly indicates the percentage of long int valued metrics (values in [0,65535]) among all metrics. There will be $si = \text{round}\,(sintper \cdot i)$ short int valued metrics and $li = i - si$ long int valued metrics in each randomly constructed QoS-based WS specification. According to Cortés et. al. [CMDTT05], increasing the percentage of long integers causes matchmaking execution time to increase.

- *hardstat*: this parameter indicates if all constraints in the QoS demand are hard or not. If yes, then all constraints have weight equal to 10.0. If not, then some constraints are randomly selected to be hard and some to be soft (weight is in (0.0,1.0)). In all the conducted experiments, all constraints were considered to be hard.

**Comparison metrics** Comparison metrics are actually parameters used to compare the performance of the algorithms under consideration. For our experiments, we used the following:

- *Matchmaking time*: this parameter indicates the average execution time of a matchmaking algorithm for the 30 runs of each test.

- *Precision*: QoS-based WS matchmaking is an instance of an information retrieval problem. There are two popular metrics for measuring the accuracy of an information system: *precision* and *recall*. If *correct* is the set of correct results and *ret* is the set of returned results, then precision is expressed as: $\frac{|correct \cap ret|}{|ret|}$. Each run not only randomly creates a set of QoS offers and one demand but also creates the accurate categories of results for the purpose of our evaluation. So it creates the following sets: *super*, *exact*, *partial* and $fail$. In each run, algorithms $MIP_{conf}$, $MIP_{cond-conf}$ and $MIP_{opt}$ produce the following results: $exact_i$ and $fail_i$, where $i$ is one of $MIP_{conf}$, $MIP_{cond-conf}$ and $MIP_{opt}$. Thus, their precision is: $prec_i = \frac{|(super \cup exact) \cap exact_i|}{|exact_i|}$. In each run, algorithm $MIP_{unary}$ produces the following results: $super_i$, $exact_i$, $partial_i$ and $fail_i$, where $i$ is 3. Its precision is: $prec_i = \frac{|(super \cup exact) \cap (super_i \cup exact_i)|}{|super_i \cup exact_i|}$. After the

execution of 30 runs, each algorithm will have as its precision the minimum of the precision over all runs. In other words, $precision_i = \min(prec_i)$.

- *Recall*: Based on the previous analysis, recall is expressed as: $\frac{|correct \cap ret|}{|correct|}$. In each run, the recall of algorithms $MIP_{conf}$, $MIP_{cond-conf}$ and $MIP_{opt}$ is: $rec_i = \frac{|(super \cup exact) \cap exact_i|}{|super \cup exact|}$ and the recall of algorithm $MIP_{unary}$ is: $rec_i = \frac{|(super \cup exact) \cap (super_i \cup exact_i)|}{|super \cup exact|}$. In the same manner, after 30 runs, all algorithms will have as their recall: $recall_i = \min(rec_i)$.

- *Len*: this parameter indicates the average size of the matching results for the 30 runs of each test. For algorithms $MIP_{conf}$, $MIP_{cond-conf}$ and $MIP_{opt}$ each run has: $Len = |match_i|$. For algorithm $MIP_{unary}$ each run has: $Len = |super_i \cup match_i|$.

### Input creation

As explained previously, each test of an experiment created a series of 30 runs. At each execution of a run, a new random input was created based on the values of the tuning parameters. Then this input was fed to the matchmaking algorithms. After the execution of these algorithms, the values of the comparison metrics for every algorithm were derived. A new random input was created for each run in order to test the matchmaking algorithms with different input elements each time. However, it must be indicated that the number of input elements remained constant at each test.

Each random input construction created one QoS demand and $adscn$ QoS offers. The QoS demand was created first having $metrcn$ QoS metrics and $4 \cdot metrcn$ unary constraints, as was explained earlier. Each metric was granted with a random value type (real, short or long int) according to the *realper* and *sintper* tuning parameters. In addition, the monotonicity of each metric was assigned randomly. Suppose $x$ is the $1 \times metrcn$ array of metrics, then the QoS demand could be expressed as: $[blc_D \leq x \leq buc_D, blx \leq x \leq bux]$, where $blx$, $bux$ are $1 \times metrcn$ arrays indicating the low and upper limit respectively of the value types of each metric and $blc_D$, $buc_D$ are $1 \times metrcn$ arrays indicating the low and upper limit respectively of each metric enforced by the QoS demand.

After demand creation, $adscn$ QoS offers were created. Initially, each QoS offer was just a copy of the QoS demand. Then each QoS offer was randomly classified as *super*, *exact*, *partial* or *fail* according to the *matchper*, *bmper* and *partper* tuning parameters:

- If offer $j$ was classified as *super*, then a random number of metrics ($< metrcn$) was selected. If selected metric $i$ was positively monotonic, then its limits were changed to: $blc_D(i) < blc_j(i) < buc_D(i)$ and $buc_D(i) < buc_j(i) \leq bux(i)$. Otherwise, its limits were changed to: $blx(i) \leq blc_j(i) < blc_D(i)$ and $blc_D(i) < buc_j(i) \leq buc_D(i)$. The other not selected metrics had their limits changed to: $blc_D(i) < blc_j(i) < buc_D(i)$ and $blc_j(i) \leq buc_j(i) < buc_D(i)$.

- If offer $j$ was classified as *exact*, then all metrics had their limits changed to: $blc_D(i) < blc_j(i) < buc_D(i)$ and $blc_j(i) \leq buc_j(i) < buc_D(i)$.

- If offer $j$ was classified as *partial*, then a random number of metrics ($< metrcn$) was selected. If selected metric $i$ was negatively monotonic, then its limits were changed to: $blc_D(i) < blc_j(i) < buc_D(i)$ and $buc_D(i) < buc_j(i) \leq bux(i)$. Otherwise, its limits were changed to: $blx(i) \leq blc_j(i) < blc_D(i)$ and $blc_D(i) < buc_j(i) \leq buc_D(i)$. The other not selected metrics had their limits changed to: $blc_D(i) < blc_j(i) < buc_D(i)$ and $blc_j(i) \leq buc_j(i) < buc_D(i)$.

- If offer $j$ was classified as *fail*, then for each metric $i$ we had the following cases: If it was negatively monotonic, then its limits were changed to: $blc_D(i) < blc_j(i) < buc_D(i)$ and $buc_D(i) < buc_j(i) \leq bux(i)$. Otherwise, its limits were changed to: $blx(i) \leq blc_j(i) < blc_D(i)$ and $blc_D(i) < buc_j(i) \leq buc_D(i)$.

Thus, in the end, each offer $j$ could be expressed as: $[blc_j \leq x \leq buc_j, blx \leq x \leq bux]$.

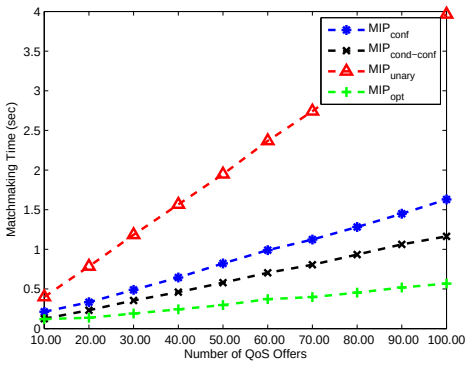
For algorithm $MIP_{opt}$, each QoS specification is enriched with the objective: $\sum_{i \in \uparrow} \frac{x_i - p_{min_i}}{p_{max_i} - p_{min_i}} * w_i + \sum_{i \in \downarrow} \frac{p_{max_i} - x_i}{p_{max_i} - p_{min_i}} * w_i$ ,where $i \in \uparrow$ means metric $i$ is positively monotonic, $i \in \downarrow$ means metric $i$ is negatively monotonic, $x_i$ is metric's $i$ MIP variable, $w_i$ is the weight of the metric, $p_{max_i} = max\left(max_j\left(buc_j\left(i\right)\right), buc_D\left(i\right)\right)$ and $p_{min_i} = min\left(min_j\left(blc_j\left(i\right)\right), blc_D\left(i\right)\right)$ for every metric $i$ and offer $j$.

It must be noted that all QoS metrics were considered "insensitive". This means that only one of the constraints of the QoS demand was respected for each of the QoS metrics. For this reason, we expect that the performance of the $MIP_{cond-conf}$ will be a little better than expected. In addition, it must be remarked that in our experiments we considered that all QoS offers and demands had constraints on the same set of QoS metrics. It's like assuming that the QoS-based WS alignment algorithm was totally successful in achieving his goal. In reality, this is not the right case.

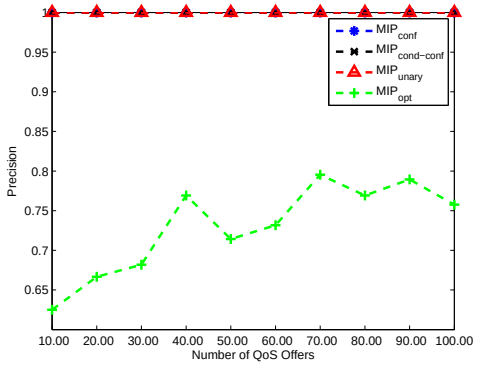
### Results

In order to experimentally evaluate the performance and accuracy of the four MIP-based QoS-based WS matchmaking algorithms in different settings, a series of eight (8) experiments were conducted. Each experiment had the following values for the tuning parameters: $adscn = 10$, $matchper = 0.5$, $bmper = 0.4$, $partper = 0.5$, $arity = 1$, $metrcn = 10$, $realper = 0.5$, $sintper = 0.5$, $hardstat = 1$, unless otherwise stated. Note that at each experiment a tuning parameter was increasing its value according to a specific step until a specific upper limit.

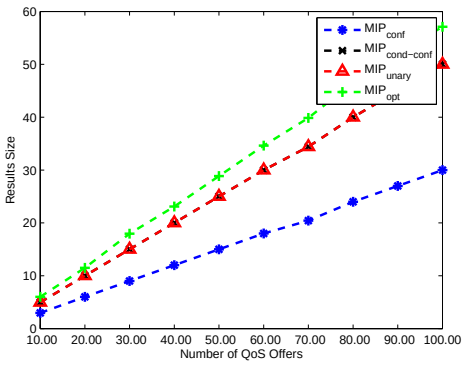
In the first conducted experiment, we increased the number of QoS offers in order to observe the performance of the matchmaking algorithms. We expected a linear increasing performance behavior for all algorithms according to the time complexity analysis of the previous section (time was proportional to the number of QoS Offers). Indeed, as can be seen in Figure 7.5(a), this is the case. In addition, algorithm $MIP_{opt}$ was the fastest followed by algorithms $MIP_{cond-conf}$, $MIP_{conf}$ and $MIP_{unary}$ in decreasing order of performance. This was also indicated by the time complexity analysis of the previous section, although we were not quite sure if $To_M^{2M} \approx T_M^{2M+1}$. As far as accuracy is concerned, all algorithms




(a) Matchmaking time




(b) Precision

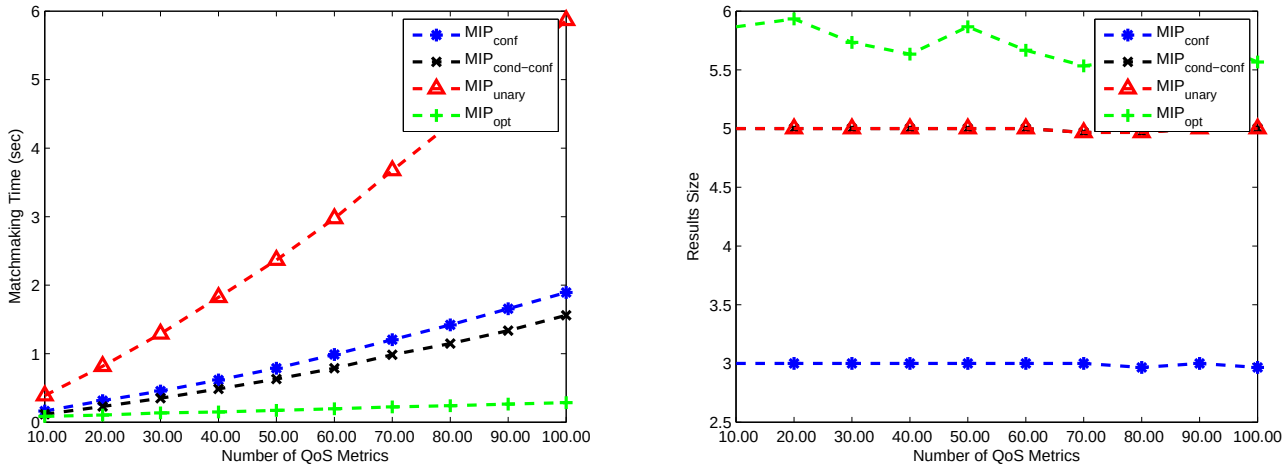



(c) Size of matchmaking results

Figure 7.5: 1st Experiment results

had $precision = 1.0$, except from algorithm $MIP_{opt}$ whose precision ranged from 0.60 to 0.76. This is indicated in Figure 7.5(b). Recall was also 1.0 for all algorithms except from algorithm $MIP_{conf}$ that had recall always equal to 0.6. This was also predicted as the percentage of *super* matches among all matches is the same as the percentage of 'false negatives' produced by algorithm $MIP_{conf}$. For the majority of the experiments, we had $bmper = 0.4$ and $matchper = 0.5$, so the accuracy of the algorithms did not change unless these two parameters were altered. As far as size of matchmaking results is concerned, algorithms $MIP_{cond-conf}$ and $MIP_{unary}$ always returned the right number of matchmaking results while algorithm $MIP_{conf}$ returned 40% less results and algorithm $MIP_{opt}$ returned more results (from 1 to 6 more increasingly). This is indicated in Figure 7.5(c).

In the second experiment, we increased the number of QoS metrics in order to observe the performance of the matchmaking algorithms. We expected a linear increasing performance behavior for all algorithms according to the time complexity analysis of the previous section. However, as can be seen in Figure 7.6(a), this is not exactly the case. Indeed, algorithms $MIP_{conf}$, $MIP_{cond-conf}$ and $MIP_{unary}$ had a linear increasing performance behavior (their matchmaking time is proportional to the number of metrics) and their performance order did not change. But algorithm's $MIP_{opt}$ matchmaking time increased in a very little manner. This can be explained as follows: matchmaking time of algorithm $MIP_{opt}$ is not proportional to the number of QoS metrics but is proportional to the solving time of a MIPOP with $2 * M$ unary constraints, $M$ metrics and one linear objective referencing all metrics. While the number of metrics increases, this solving time increases very slightly due to the advanced MIP solving techniques and to the fact that only unary constraints are dealt with. Concerning accuracy, recall did not change for all algorithms. Precision was again 1.0 for algorithms $MIP_{conf}$, $MIP_{cond-conf}$ and $MIP_{unary}$ while algorithm $MIP_{opt}$ has its precision ranging from 0.62 to 0.72. Concerning size of matchmaking results, algorithms $MIP_{conf}$, $MIP_{cond-conf}$ and $MIP_{unary}$ were returning a specific amount of results while algorithm $MIP_{opt}$ was returning almost 1 result more than the amount of results of



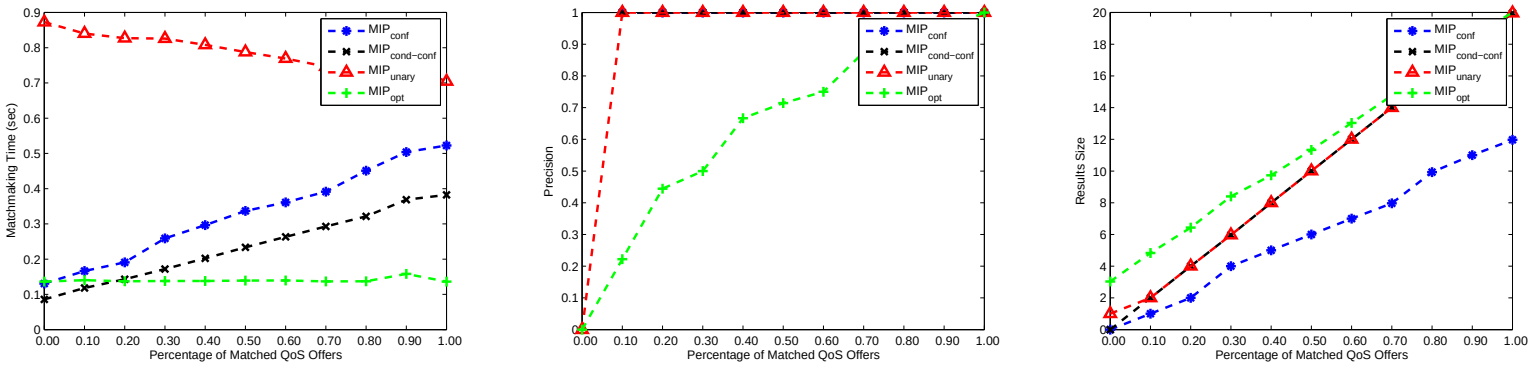

(a) Matchmaking time

(b) Size of matchmaking results

Figure 7.6: 2nd Experiment results

Algorithms $MIP_{cond-conf}$ and $MIP_{unary}$. However, it can be seen from Figure 7.6(b), as the amount of metrics increases, the probability of returning 1 more result decreases for algorithm $MIP_{opt}$. For 100 metrics, algorithm $MIP_{opt}$ did not produce more results for almost half of the (30) runs.

In the third experiment, we increased the percentage of matched QoS offers among the fixed-sized population of QoS offers ($adscn = 30$). As can be seen in Figure 7.7(a), algorithm $MIP_{opt}$ exhibited a stable behavior while algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ exhibited a linear increasing performance behavior and algorithm $MIP_{unary}$ exhibited a linear decreasing behavior. There is an explanation for this kind of behavior: algorithm $MIP_{opt}$ spends a constant amount of time in order to figure out if one QoS offer is conforming or not to the QoS demand and as long as the amount of QoS offers and metrics remain constant so does its whole matchmaking time. On the other hand, for each QoS offer, algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ spend time depending on the position of the first violating constraint in the order of the constraints. When the percentage of matching QoS offers increases, the total percentage of violating constraints decreases so these algorithms will have to do more and more work. Algorithm's $MIP_{unary}$ performance is explained based on the fact that solving consistent problems takes more than solving inconsistent problems. So when only non-conforming QoS offers are present, the ratio of consistent to inconsistent problems is the highest and the matchmaking time is the highest. But as the amount of conforming offers increases, the previous ratio decreases and so does the matchmaking time. So, when there are only conforming offers, the matchmaking time gets its smallest value. Another thing that must be pointed out is that algorithms $MIP_{conf}$ and $MIP_{unary}$ have the same performance when $matchper = 0.0$ while algorithm $MIP_{cond-conf}$ has the best performance. But as $matchper$ increases, algorithm $MIP_{opt}$ shows stable performance and



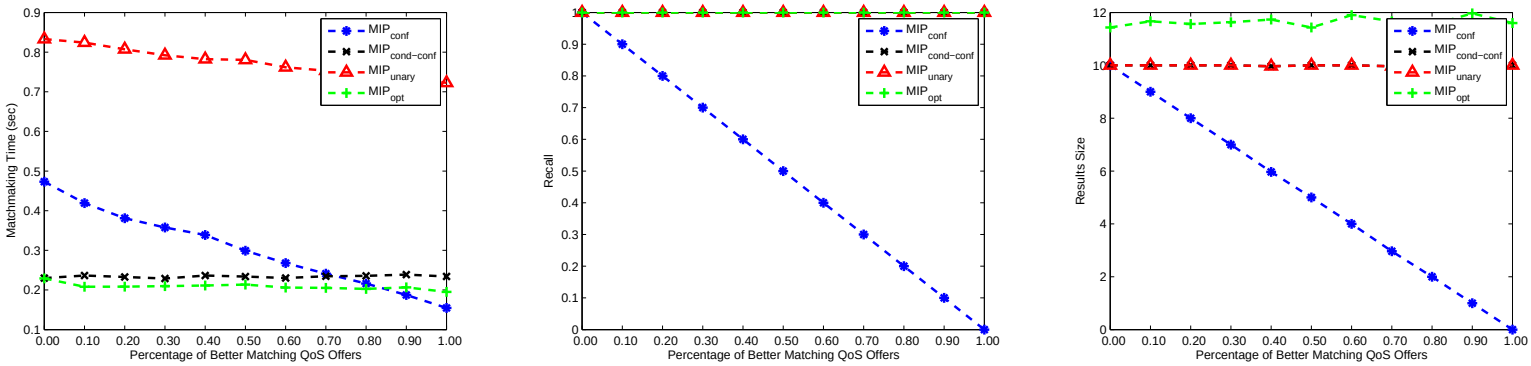

(a) Matchmaking time (b) Precision (c) Size of matchmaking results

Figure 7.7: 3rd Experiment results

outperforms algorithm $MIP_{cond-conf}$ when *matchper* takes the value of 0.3.

As far as accuracy is concerned, recall for all the algorithms was the same as in the previous experiments. But precision was 1 for all the algorithms except from algorithm $MIP_{opt}$. As can be seen from Figure 7.7(b), precision of algorithm $MIP_{opt}$ increased from 0.0 to 1.0 step-wise linearly. This increase is explained as follows: when the number of conforming results increases, the corresponding number of non-conforming results decreases and so does the possibility of promoting a partial result to the *exact* matches list. Concerning size of matching results, algorithms $MIP_{cond-conf}$ and $MIP_{unary}$ returned the same amount of results. Of course, this amount was increasing as the number of conforming offers was increasing. Algorithm $MIP_{conf}$ presented the same behavior but it was returning less results than the other two algorithms based on the 'negative positives' effect. Finally, algorithm $MIP_{opt}$ had the same behavior but it was returning more results than algorithms $MIP_{cond-conf}$ and $MIP_{unary}$ until *matchper* got the value of 1.0. This is indicated in Figure 7.7(c).

In the fourth experiment, we increased the percentage of *super* matches among the fixed-sized population of QoS offers ($adscn = 20$). Concerning matchmaking time, algorithms $MIP_{cond-conf}$ and $MIP_{opt}$ exhibited an almost stable behavior while algorithm $MIP_{conf}$ and $MIP_{unary}$ exhibited a decreasing linear behavior. The behavior of algorithm $MIP_{conf}$ can be explained as follows: better matching QoS offers are 'false negatives' for this algorithm so their increase causes a decrease in the number of matched QoS offers and finally in the matchmaking time. The behavior of algorithm $MIP_{unary}$ can be explained as follows: each QoS offer is ranked as *super* match when all $M$ 'interesting' problems are infeasible and 1 out of $M$ 'non-interesting' problems is feasible. On the other hand, a QoS offer is ranked as *exact* match when all $M$ 'interesting' problems are infeasible and all $M$ 'non-interesting' problems are infeasible. So matchmaking takes longer for exact matches than for super matches. Thus, as super matches increase and exact matches decrease, the matchmaking time will decrease. Algorithm's $MIP_{opt}$ behavior was totally expected as this algorithm's




(a) Matchmaking time (b) Recall (c) Size of matchmaking results

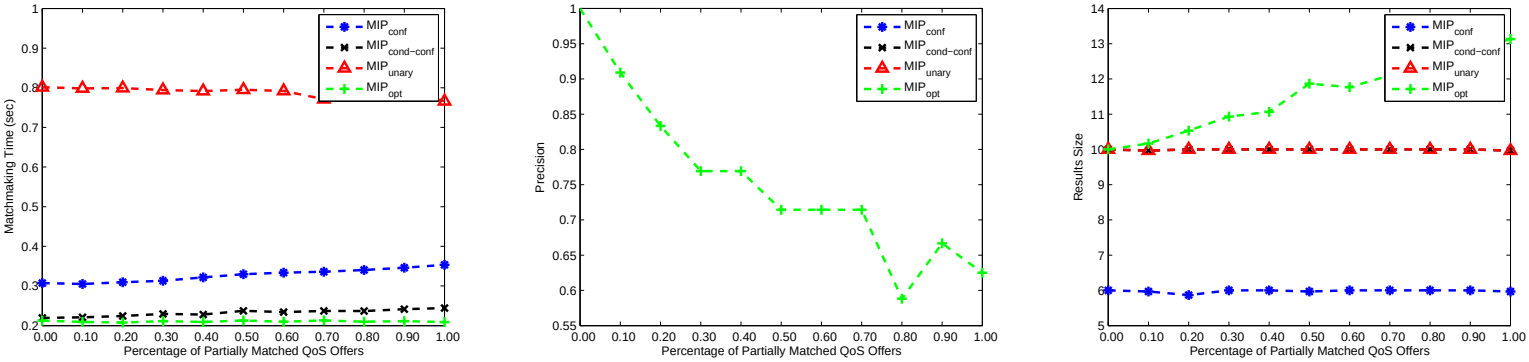
Figure 7.8: 4th Experiment results

matchmaking time depends mainly on the number of QoS offers which remains constant throughout this experiment. In addition, algorithm $MIP_{cond-conf}$ does not distinguish between super and exact matches and takes the same time for each conforming offer. So as long as the number of conforming offers does not change, so does the matchmaking time of this algorithm. The above results can be seen in Figure 7.8(a).

Concerning accuracy, all algorithms had precision equal to 1.0 except from algorithm $MIP_opt$ that its precision ranged from 0.66 to 0.77. Recall was also again 1.0 for all algorithms except algorithm $MIP_{conf}$. As can be seen from Figure 7.8(b), recall of algorithm $MIP_{conf}$ decreases in the same amount as *bmper* increases. So, indeed, *better* match offers represent 'false negatives' for Algorithm 1. Thus, we can definitely express that: $rec_1 = 1 - bmper$. As far as size of matching results is concerned, the 'false negatives' effect influences the results size of algorithm $MIP_{conf}$ which decreases linearly until it gets zero when $bmper = 100\%$. The behavior of the four algorithms on this comparison metric can be seen in Figure 7.8(c).

In the fifth experiment, we increased the percentage of *partial* matches among the fixed-sized population of QoS offers ($adscn = 20$). As can be seen in Figure 7.9(a), algorithms $MIP_{unary}$ and $MIP_{opt}$ had a stable behavior while the rest of the algorithms exhibited a small increasing linear behavior. This kind of behavior can be explained. Algorithm $MIP_{opt}$ does the same work when the number of QoS offers does not change. As long as $matchper! = 0$, algorithm $MIP_{unary}$ performs the same amount of work for every non-conforming QoS offer. On the other hand, the rest of the algorithms do considerably more work for *partial* results than for *fail* results so when the number of *partial* results increases so does the whole matchmaking execution time.

Concerning recall, algorithms $MIP_{cond-conf}$, $MIP_{unary}$ and $MIP_{opt}$ had recall equal to 1.0 while algorithm $MIP_{conf}$ had recall equal to $1 - bmper$. But for precision and size of matchmaking results, although algorithms $MIP_{conf}$, $MIP_{cond-conf}$ and $MIP_{unary}$ had precision equal to 1.0 and a constant amount of results, this was not the case for algorithm



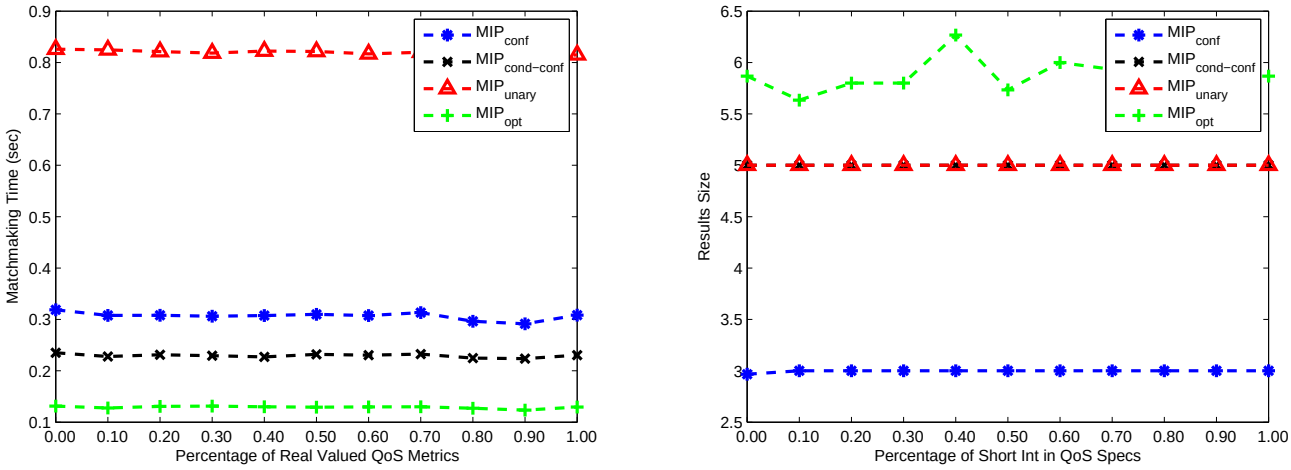

(a) Matchmaking time (b) Precision (c) Size of matchmaking results

Figure 7.9: 5th Experiment results

$MIP_{opt}$. The latter algorithm had its precision decreasing stepwise linearly and the size of matching results increasing stepwise linearly. These results can be seen in Figures 7.9(b) and 7.9(c). The behavior of algorithm $MIP_{opt}$ can be explained as follows: When the number of partial results increases, the possibility of promoting a partial result to the exact match list increases. So the number of matching results increases as more partial results are promoted. Thus, precision decreases as more not accurate results are returned.

In the sixth and seventh experiments, we had $adscn = 10$, $metrcn = 20$ and we increased the percentage of real-valued QoS metrics and short-integer-valued QoS metrics, respectively, in the QoS-based WS specifications. The results were exactly the same and are shown in Figure 7.10(a). More specifically, all algorithms had approximately a stable performance behavior. This can be explained as follows: MIP uses better techniques for feasibility checking than CP. These techniques do not depend on the size of the domain of the variables. In addition, as we are dealing with unary constraints, integer variables do not play an important role in the MIP solving time. The results on accuracy are the same as the results in experiments one and two. Concerning the size of conforming results, all algorithms exhibited a stable behavior and this is indicated in Figure 7.10(b).

In the last experiment, we had $metrcn = 10$, $matchper = 0.0$ and we were increasing the number of QoS offers. We wanted to observe the performance of the algorithms when there are no matches (i.e. the QoS demand is over-constrained). Figure 7.11(a) shows the produced results. Algorithm $MIP_{unary}$ exhibited a big linear increasing performance behavior while the rest of the algorithms exhibited a very small linear increasing performance behavior. This can be explained as follows: Algorithm $MIP_{unary}$ spends a specific amount of time for each non-conforming QoS offer. So as the number of non-conforming QoS offers increases, so does the matchmaking time. On the other hand, algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ need only to solve one to three MIPPs, in average, each of the time for every QoS offer. That's why their matchmaking time increases so slowly. Algorithm's $MIP_{opt}$ behavior is the same as that of the first experiment. This indicates that the latter algorithm's




(a) Matchmaking time

(b) Size of matchmaking results

Figure 7.10: 6th Experiment results

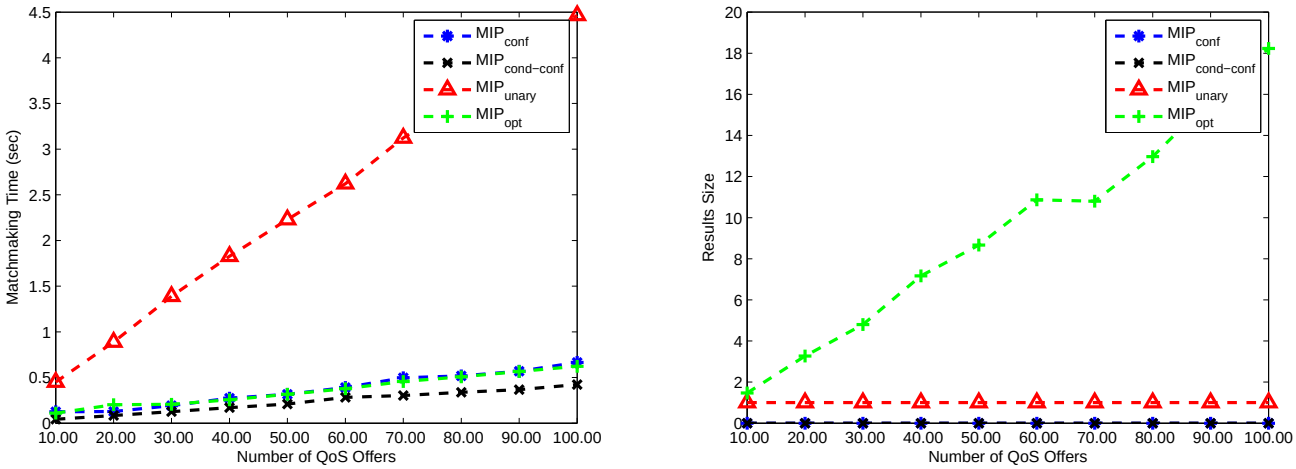
performance is not influenced by the fact that all QoS offers were non-conforming ones. It must also be noted that algorithms $MIP_{conf}$ and $MIP_{opt}$ had exactly the same performance behavior.

In this type of experiment, we cannot define accuracy. However, we can compare these algorithms on the size of conforming results. As expected, algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ do not produce any result. On the other hand, algorithm $MIP_{unary}$ always produces exactly one result that violates the smallest number of QoS demand's constraints. Finally, algorithm's $MIP_{opt}$ number of results was increasing linearly, as can be seen from Figure 7.11(b).

### 7.2.4 Unary Semi-random Evaluation of MIP-based Matchmaking Algorithms

In the second series of experiments we wanted to evaluate our MIP-based QoS-based WS matchmaking algorithms with QoS-based WS specifications that have specific content. With specific content, we mean that specific QoS metrics which have specific sensitivity and monotonicity and specific domain of values were involved in unary constraints in the specifications. From the set of these QoS metrics, some of them were domain-independent and the others were domain-dependent taken from the application domain of Traffic Monitoring. Now we analyze one by one all QoS metric involved:

- The first QoS metric was the *MTBF* metric which is domain-independent and measured the QoS attribute of *Reliability*. This metric was "insensitive" and positively monotonic. Its domain of values was the set of integers $[0, 365]$.




(a) Matchmaking time

(b) Size of matchmaking results

Figure 7.11: 7th Experiment results

- The second QoS metric, $X_{de}$, was again domain-independent and measured the QoS attribute of *Data Encryption*. This metric was "sensitive" and positively monotonic. Its domain of values was mapped to the integers set [1,5].
- The third QoS metric, $X_{rt}$, was domain-dependent and measured the QoS attribute of *Refresh Time*. This metric was "sensitive" and negatively monotonic. Its domain of values was the reals set [0.0,4.0].
- The fourth QoS metric, $X_{et}$, was domain-independent and measured the QoS attribute of *Execution Time*. This metric was "sensitive" and negatively monotonic. Its domain of values was the reals set [0.0,10.0].
- The fifth QoS metric, $X_{avail}$, was domain-independent and measured the QoS attribute of *Availability*. This metric was "insensitive" and positively monotonic. Its domain of values was the reals set [0.0,1.0].
- The sixth QoS metric, $X_{rep}$, was domain-independent and measured the QoS attribute of *Reputation*. This metric was "insensitive" and positively monotonic. Its domain of values was mapped to the integers set [1,5].
- The seventh QoS metric, $X_{price}$, was domain-independent and measured the QoS attribute of *Price*. This metric was "insensitive" and negatively monotonic. Its domain of values was the reals set [100.0,1000.0].
- The eighth QoS metric, $X_{acc}$, was domain-dependent and measured the QoS attribute of *Accuracy*. This metric was "sensitive" and positively monotonic. Its domain of values was the reals set [0.0,1.0].

- The ninth QoS metric, $X_{compl}$, was domain-dependent and measured the QoS attribute of *Completeness*. This metric was "sensitive" and positively monotonic. Its domain of values was the reals set [0.0,1.0].
- The tenth QoS metric, $X_{valid}$, was domain-dependent and measured the QoS attribute of *Validity*. This metric was "sensitive" and positively monotonic. Its domain of values was the reals set [0.0,1.0].
- The eleventh QoS metric, $X_{tmlness}$, was domain-dependent and measured the QoS attribute of *Timeliness*. This metric was "sensitive" and positively monotonic. Its domain of values was the reals set [0.0,1.0].
- The twelfth QoS metric, $X_{ca}$, was domain-dependent and measured the QoS attribute of *Covered Area*. This metric was "insensitive" but did not have monotonicity. Its domain of values was the String set {"NE","NW","SE","SW"}.
- The thirteenth QoS metric, $X_{rs}$, was domain-dependent and measured the QoS attribute of *Routes Set*. This metric was "insensitive" but did not have monotonicity. Its domain of values was the String set {"national","interstate","local","downtown"}.
- The fourteenth QoS metric, $X_{dl}$, was domain-dependent and measured the QoS attribute of *Detail Level*. This metric was "insensitive" but did not have monotonicity. Its domain of values was the String set {"accidents","traffic jams","closed routes", "detours","predictions","localization","tolls"}.

This means that we had 8 out of 14 QoS metrics as "sensitive" (while the last three "insensitive" QoS metrics do not have monotonicity so they do not actually count) so we expected that the execution time difference between the $MIP_{conf}$ and $MIP_{cond-conf}$ algorithms will be the smallest possible. All QoS metrics were taking values in their constraints from their domain of values. Thus, in this series of experiments the code for producing the QoS-based WS specification was altered. However, the algorithms were not changed at all except from one point. This was the fact that the last 3 QoS metrics had as their scale the nominal scale. So matchmaking an offer and a demand with respect to these metrics had to use different semantics: subset semantics (i.e these metrics were represented as string sets and an appropriate facility of XPressMP was used to check if one set was a subset of the other).

Another point that must be made is that the testing framework used was the same as in the previous series of experiments. As we said only the code for producing QoS specs was altered and slightly the matchmaking algorithms. In addition, the parameters for controlling the number of QoS metrics and the percentages of real and int variables remained fixed as we had fixed QoS metrics.

### Results

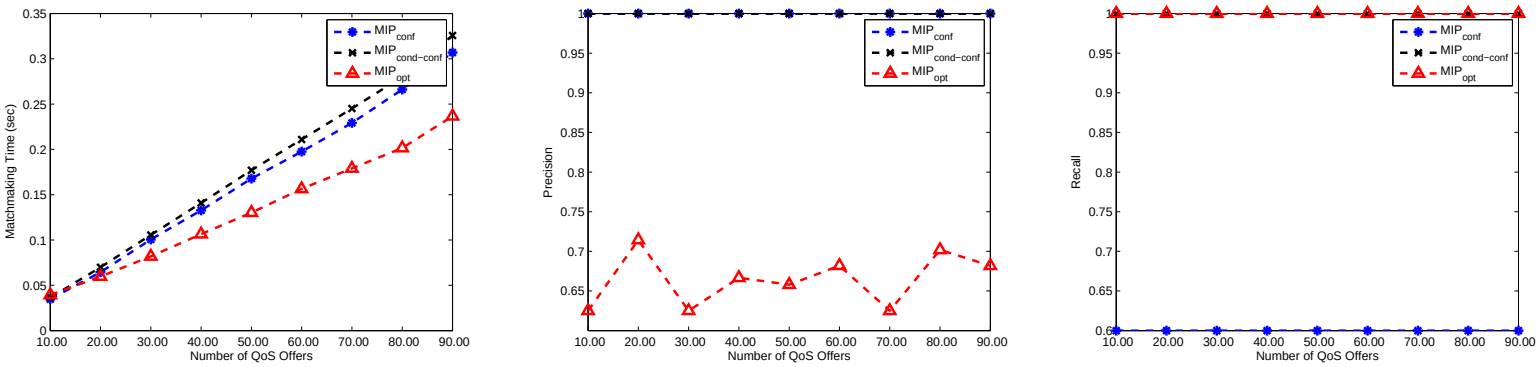
In order to experimentally evaluate the performance and accuracy of the our MIP-based QoS-based WS matchmaking algorithms in different settings, a series of five (5) experiments were

conducted. Each experiment had the following values for the tuning parameters: $adscn = 30$, $matchper = 0.5$, $bmper = 0.4$, $partper = 0.5$, $arity = 1$, $metrcn = 14$, $hardstat = 1$, unless otherwise stated. Each experiment was repeated again for 30 times. Note that at each experiment a tuning parameter was increasing its value according to a specific step until a specific upper limit. It must be remarked that we neglected the $MIP_{unary}$ algorithm from these tests as its performance and accuracy would not be greatly altered so there was no point in evaluating it and commenting again for the same results.

In the first conducted experiment, we increased the number of QoS offers in order to observe the performance of the matchmaking algorithms. We expected a linear increasing performance behavior for all algorithms according to the time complexity analysis of the previous section (time was proportional to the number of QoS Offers). Indeed, as can be seen in Figure 7.12(a), this is the case. In addition, algorithm $MIP_{opt}$ was the fastest followed by algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ in decreasing order of performance. This is slightly different from what were expecting from the theoretical analysis of the latter two algorithms and the corresponding result from the first series of experiments. However, this result can be explained. Although the $MIP_{cond-conf}$ algorithm may solve one or maybe two less MIPPs than the $MIP_{conf}$, it has more complicated code and as the MIP solving engine we use is very fast and reuses the results of the previous MIPP solvings, it reverses the former advantage. As far as accuracy is concerned, all algorithms had $precision = 1.0$, except from algorithm $MIP_{opt}$ whose precision ranged from 0.60 to 0.7 increasingly. This is indicated in Figure 7.12(b). Recall was also 1.0 for all algorithms except from algorithm $MIP_{conf}$ that had recall always equal to 0.6 (see Figure 7.12(c)). This was also predicted as the percentage of *super* matches among all matches is the same as the percentage of 'false negatives' produced by algorithm $MIP_{conf}$. For the majority of the experiments, we had $bmper = 0.4$ and $matchper = 0.5$, so the accuracy of the algorithms did not change unless these two parameters were altered. As far as size of matchmaking results is concerned, algorithm $MIP_{cond-conf}$ always returned the right number of matchmaking results while algorithm $MIP_{conf}$ returned 40% less results and algorithm $MIP_{opt}$ returned more results.

In the second experiment, we increased the percentage of matched QoS offers among the fixed-sized population of QoS offers ($adscn = 30$). As can be seen in Figure 7.13(a), algorithm $MIP_{opt}$ exhibited a stable behavior while algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ exhibited a linear increasing performance behavior. This behavior was explained in the first series of experiments. Another thing that must be pointed out is that again $MIP_{conf}$ algorithm is better than the $MIP_{cond-conf}$. $MIP_{opt}$ comes third but as $matchper$ increases, it shows stable performance and outperforms the other algorithms when $matchper$ goes beyond 0.4. This is slightly different from the first series of experiments where this take over happened a little earlier.

As far as accuracy is concerned, recall for all the algorithms was the same as in the previous experiments. But precision was 1 for all the algorithms except from algorithm $MIP_{opt}$. As can be seen from Figure 7.13(b), precision of algorithm $MIP_{opt}$ increased from 0.0 to 1.0 exponentially. This increase is explained as follows: when the number of conforming results increases, the corresponding number of non-conforming results decreases and so does the




(a) Matchmaking time (b) Precision (c) Recall

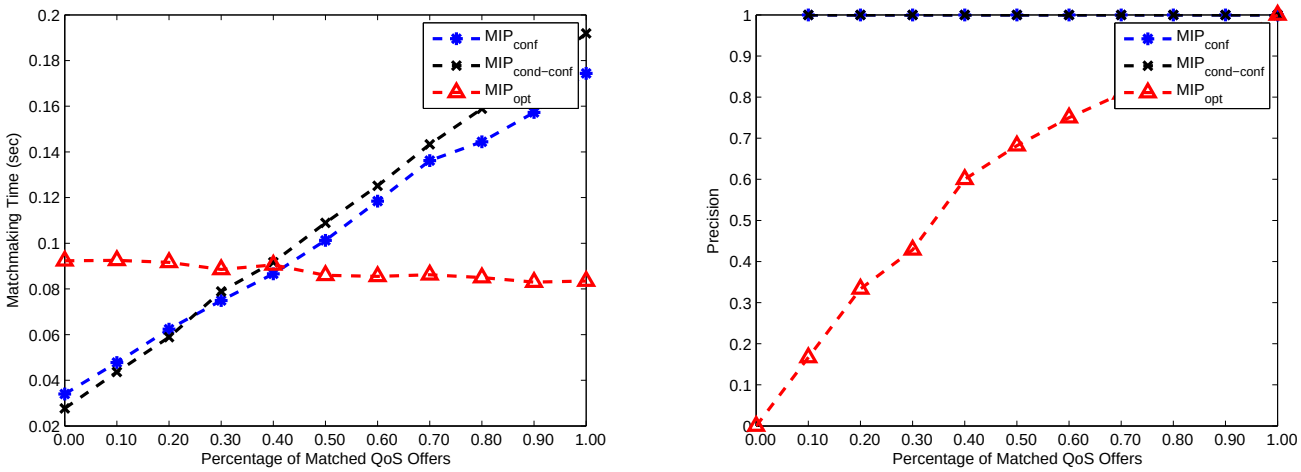
Figure 7.12: 1st Experiment results

possibility of promoting a partial result to the *exact* matches list. Concerning size of matching results, algorithm $MIP_{conf}$ consistently returned less results than the $MIP_{cond-conf}$ algorithm based on the 'negative positives' effect. Finally, algorithm $MIP_{opt}$ returned more results than the other two algorithms until *matchper* got the value of 1.0.

In the third experiment, we increased the percentage of *super* matches among the fixed-sized population of QoS offers ($adscn = 30$). Concerning matchmaking time, algorithms $MIP_{cond-conf}$ and $MIP_{opt}$ exhibited an almost stable behavior while algorithm $MIP_{conf}$ exhibited a decreasing linear behavior. This is much the same results as those of the first series of experiments. The only difference is than now $MIP_{cond-conf}$ outperforms the other algorithms earlier, which is easy to explain as in the first series of experiments we had 10 "insensitive" metrics and now we actually have only 3. The above results can be seen in Figure 7.14(a). Concerning recall (see Figure 7.14(b)), precision and size of results (see Figure 7.14(c)) we have absolutely the same picture with the corresponding results of the previous series of experiments.

In the fourth experiment, we increased the percentage of *partial* matches among the fixed-sized population of QoS offers ($adscn = 20$). As can be seen in Figure 7.15(a), all algorithms exhibited a not very stable behavior. However, similarly to the corresponding result of the first series of experiments, the $MIP_{conf}$ and $MIP_{cond-conf}$ algorithms had an increasing behavior while in average the $MIP_{opt}$ algorithm had a stable behavior. All these facts were justified in the previous set of experiments.

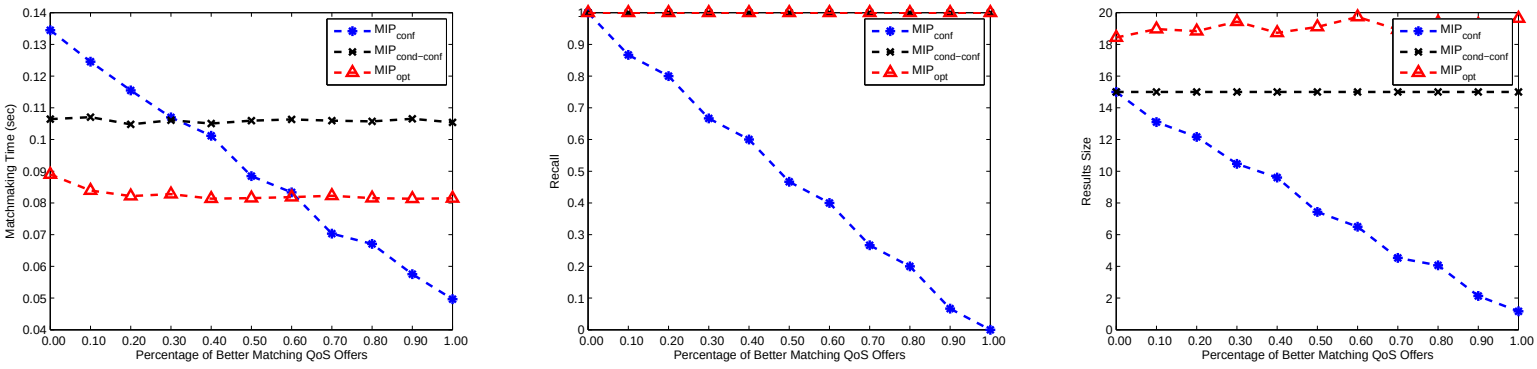
Concerning recall, algorithms $MIP_{cond-conf}$ and $MIP_{opt}$ had recall equal to 1.0 while algorithm $MIP_{conf}$ had recall equal to $1-bmper$. But for precision and size of matchmaking results, although algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ had precision equal to 1.0 and a constant amount of results, this was not the case for algorithm $MIP_{opt}$. The latter algorithm had its precision decreasing stepwise linearly and the size of matching results increasing stepwise linearly. These results can be seen in Figures 7.15(b) and 7.15(c). The behavior of algorithm $MIP_{opt}$ was explained in the first series of experiments.




(a) Matchmaking time

(b) Precision

Figure 7.13: 2nd Experiment results

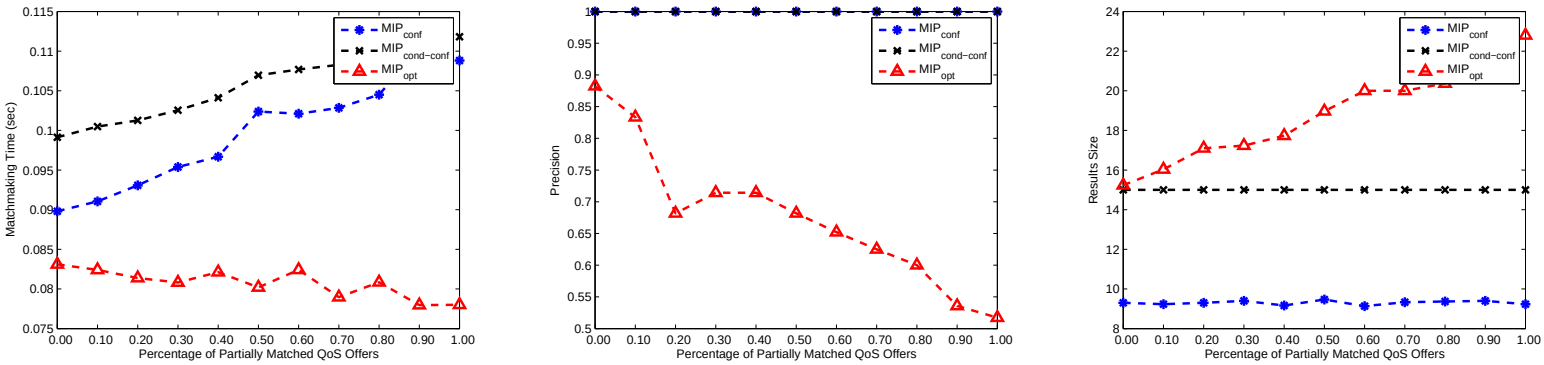



(a) Matchmaking time

(b) Recall

(c) Size of matchmaking results

Figure 7.14: 3rd Experiment results




(a) Matchmaking time (b) Precision (c) Size of matchmaking results

Figure 7.15: 4th Experiment results

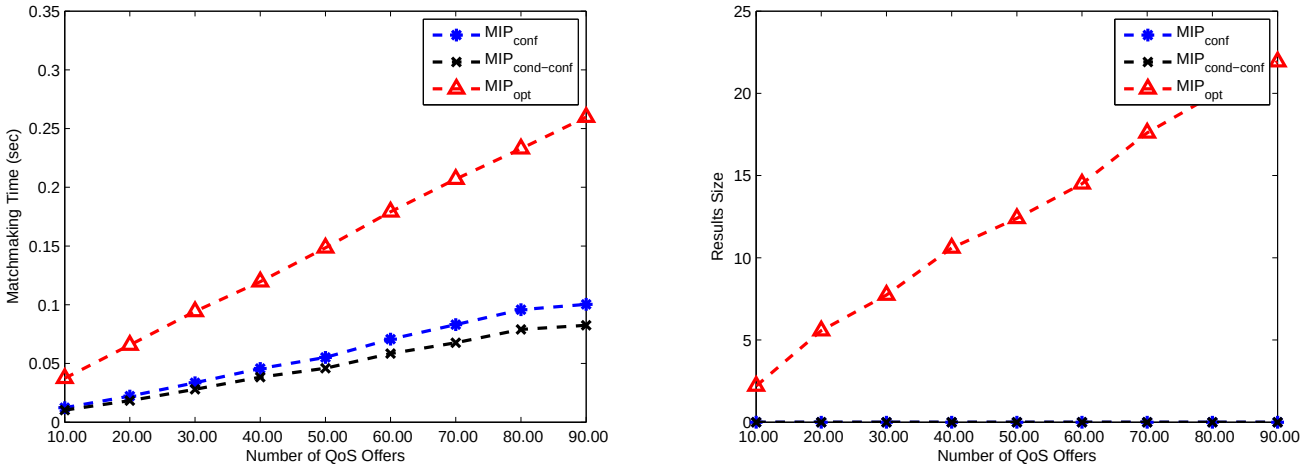
In the last experiment, we had $matchper = 0.0$ and we were increasing the number of QoS offers. We wanted to observe the performance of the algorithms when there are no matches (i.e. the QoS demand is over-constrained). Figure 7.16(a) shows the produced results. This performance is almost the same with the one we discovered in the first series of experiments. The only difference from the picture we got from the previous results of this series of experiments is that now $MIP_{cond-conf}$ is better than $MIP_{conf}$.

In this type of experiment, we cannot define accuracy. However, we can compare these algorithms on the size of conforming results. As expected, algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ do not produce any result. On the other hand, algorithm's $MIP_{opt}$ number of results was increasing linearly, as can be seen from Figure 7.16(b).

As can be seen from the first two series of experiments, the behavior of the $MIP_{opt}$ has remained almost the same and the best from all other algorithms (if we forget about precision). The only observed change is that the $MIP_{conf}$ is slightly faster than the $MIP_{cond-conf}$ algorithm in the second series. We believe that if we had tested the $MIP_{unary}$ algorithm, its results would have been almost identical. So the main conclusion is that our two MIP-based QoS-based WS matchmaking algorithms had a specific performance behavior during these two series of experiments which was totally expected.

### 7.2.5 N-ary Semi-random Evaluation of MIP-based Matchmaking Algorithms

In the third series of experiments our main goal was to introduce n-ary constraints to the semi-random framework of the previous set of experiments in order to observe the changes of the performance and accuracy of the matchmaking algorithms. We tried to produce n-ary constraints for both QoS offers and demands but we failed to do so based on the types of constraints we chose to represent, as it was too difficult to create a matching QoS offer




(a) Matchmaking time

(b) Size of matchmaking results

Figure 7.16: 5th Experiment results

after the creation of the QoS demand. So we chose to create only unary QoS demands. Thus, based on a randomly-created unary QoS demand and the tuning parameters, we were constructing/creating the corresponding n-ary QoS offers. The non-unary constraints we used for populating the QoS offers were the following:

- $X_{avail} = 0.4 + \frac{0.3}{365} * MTBF + 0.03 * (X_{rt} + 6)$, where $X_{avail}$, $MTBF$ and $X_{rt}$ are metrics defined in the previous set of experiments.
- $X_{et} = 1 + (4 - X_{rt}) * 0.9 * 2.5$
- $P_1 = \text{rand}(0, 50) + \text{rand}(10, 50) * (10 - X_{et})$, where by $\text{rand}(x_1, x_2)$ we mean that we produced randomly a number between $x_1$ and $x_2$ exclusive.
- $P_2 = \text{rand}(0, 50) + X_{avail} * \text{rand}(10, 50)$
- $P_3 = \text{rand}(0, 50) + \text{rand}(10, 50) * MTBF * \frac{1}{365}$
- $P_4 = \text{rand}(10, 50) * X_{de}$
- $P_5 = \text{rand}(0, 50) + \text{rand}(10, 50) * (4 - X_{rt})$
- $P_6 = \text{rand}(10, 50) * X_{acc}$
- $P_7 = \text{rand}(10, 50) * X_{compl}$
- $P_8 = \text{rand}(10, 50) * X_{valid}$
- $P_9 = \text{rand}(10, 50) * X_{tmlness}$

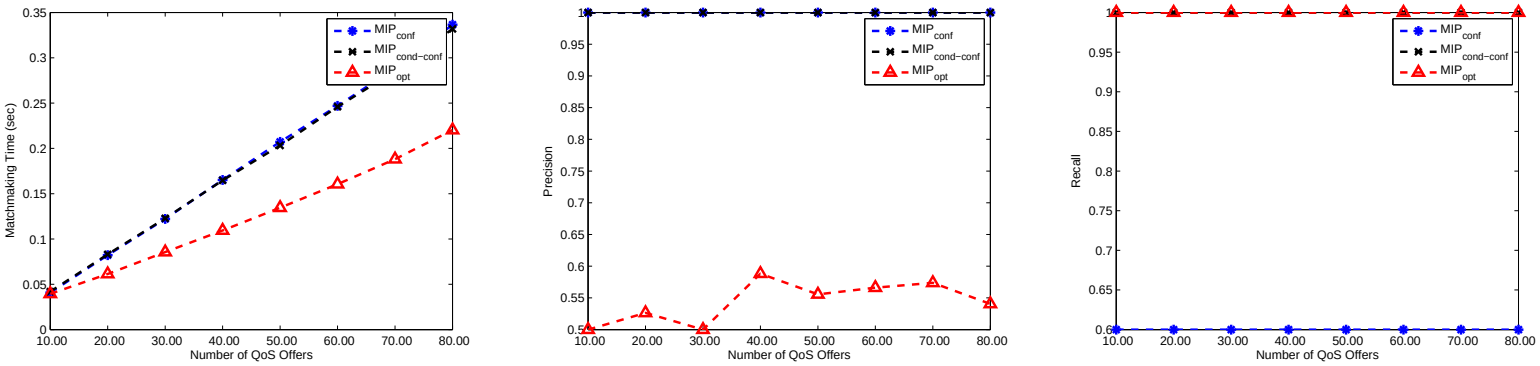
- $P_{10} = \text{rand}(10, 50) * size(X_{ca} \cap X_{ca}^D)$, where $size(X_{ca} \cap X_{ca}^D)$ means the size of the common values of the *Covered Area* QoS metric for both the QoS offer and the demand
- $P_{11} = \text{rand}(10, 50) * size(X_{rs} \cap X_{rs}^D)$, where $size(X_{rs} \cap X_{rs}^D)$ means the size of the common values of the *Routes Set* QoS metric for both the QoS offer and the demand
- $P_{12} = \text{rand}(10, 50) * size(X_{dl} \cap X_{dl}^D)$, where $size(X_{dl} \cap X_{dl}^D)$ means the size of the common values of the *Detail Level* QoS metric for both the QoS offer and the demand
- $X_{price} = P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8 + P_9 + P_{10} + P_{11} + P_{12}$

The first constraint correlates *Availability* with *Reliability* and *Refresh Time* (actually the metrics of these three QoS attributes). The second constraint correlates *Execution Time* with *Refresh Time*. The other constraints actually constitute the *pricing model* of the QoS offer. As it can be understood, the price of the QoS Offer depends and is actually calculated from the other QoS attributes, where each $P_i$ is the part (of the price) corresponding to any of the other 13 QoS attributes. Of course, besides these non-unary constraints for three of the QoS attributes/metrics (*Execution Time*, *Availability* and *Price*), the offer contains 2 unary constraints for these 3 metrics (for indicating their value type) and 4 unary constraints (2 indicating the value type of the metric and 2 for the actual range of values offered) for the rest of the QoS metrics. It must be noted that all other details related to the conduction of this series of experiments are the same with those of the previous set of experiments (the unary semi-random evaluation). So let us now present and explain the results produced.

### Results

In order to experimentally evaluate the performance and accuracy of the our MIP-based QoS-based WS matchmaking algorithms in different settings, a series of five (5) experiments were conducted. Each experiment had the following values for the tuning parameters: $adscn = 30$, $matchper = 0.5$, $bmper = 0.4$, $partper = 0.5$, $arity = 2$ (means not unary), $metrcn = 14$, $hardstat = 1$, unless otherwise stated. Each experiment was repeated for 20 times (less than before because of instability of the software we were using). Note that at each experiment a tuning parameter was increasing its value according to a specific step until a specific upper limit. It must be remarked that we neglected the $MIP_{unary}$ algorithm for the same reason as the one of the previous set of experiments.

In the first conducted experiment, we increased the number of QoS offers in order to observe the performance of the matchmaking algorithms. We expected a linear increasing performance behavior for all algorithms according to the time complexity analysis of the previous section (time was proportional to the number of QoS Offers). Indeed, as can be seen in Figure 7.17(a), this is the case. In addition, algorithm $MIP_{opt}$ was the fastest followed by algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ which had almost the same performance. So the gap between the latter two algorithms experienced in the corresponding experiment of the previous series has closed. Another thing that must be pointed out is that all of the algorithms had almost the same performance as in the previous set of experiments so they were not actually influenced by the introduction of the non-unary constraints in the offers.




(a) Matchmaking time (b) Precision (c) Recall

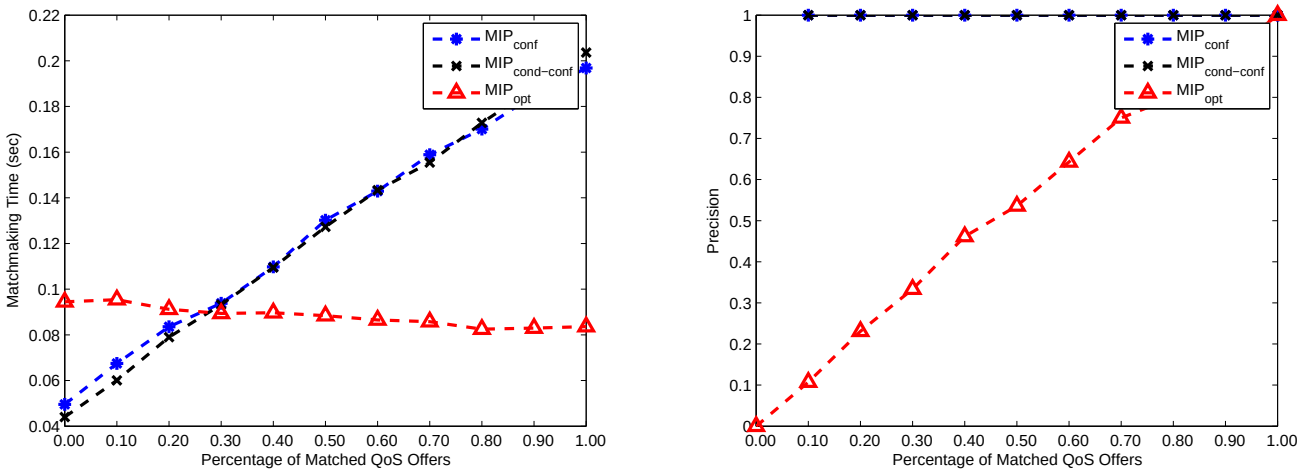
Figure 7.17: 1st Experiment results

This came as a surprise to us (the gap closing was expected) but then we could explain it because a) most of the time the constraints that had to be solved were trivial (the usual case was that the contradictory or not constraints – involving the same variable – were unary as only 3 out of the 11 metrics had complex constraints); b) MIP solving for linear constraints is really fast and as long as there are not a lot of non-unary constraints involving a lot of variables, then the solving time is almost stable.

As far as accuracy is concerned, all algorithms had $precision = 1.0$, except from algorithm $MIP_{opt}$ whose precision ranged from 0.50 to 0.6 increasingly (it is lower with respect to the previous set of experiments). This is indicated in Figure 7.17(b). Recall was also 1.0 for all algorithms except from algorithm $MIP_{conf}$ that had recall always equal to 0.6 (the usual case for specific values of the tuning parameters) (see Figure 7.17(c)). As far as size of matchmaking results is concerned, algorithm $MIP_{cond-conf}$ always returned the right number of matchmaking results while algorithm $MIP_{conf}$ returned 40% less results and algorithm $MIP_{opt}$ returned more results.

In the second experiment, we increased the percentage of matched QoS offers among the fixed-sized population of QoS offers ($adscn = 30$). As can be seen in Figure 7.18(a), algorithm $MIP_{opt}$ exhibited a stable behavior while algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ exhibited a linear increasing performance behavior. This behavior was explained in the first two series of experiments. Another thing that must be pointed out is that again $MIP_{conf}$ and $MIP_{cond-conf}$ algorithms have the same performance. $MIP_{opt}$ comes third but as $matchper$ increases, it shows stable performance and outperforms the other algorithms when $matchper$ goes beyond 0.3. This is slightly different from the second series of experiments where this take over happened a little later and the same with the first series of experiments.

As far as accuracy is concerned, recall for all the algorithms was the same as in the previous experiments. But precision was 1 for all the algorithms except from algorithm $MIP_{opt}$. As can be seen from Figure 7.18(b), precision of algorithm $MIP_{opt}$ increased from 0.0 to 1.0 linearly. This result is different from the other two series of experiments where the in-




(a) Matchmaking time

(b) Precision

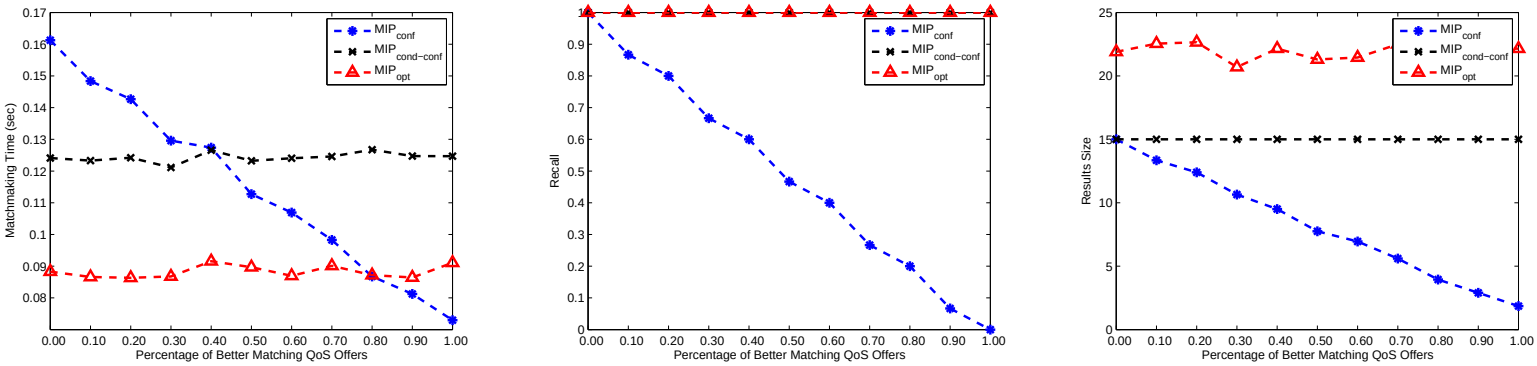
Figure 7.18: 2nd Experiment results

crease was exponential. Concerning size of matching results, algorithm $MIP_{conf}$ consistently returned less results than the $MIP_{cond-conf}$ algorithm based on the 'negative positives' effect. Finally, algorithm $MIP_{opt}$ returned more results than the other two algorithms until $matchper$ got the value of 1.0.

In the third experiment, we increased the percentage of *super* matches among the fixed-sized population of QoS offers ($adscn = 30$). Concerning matchmaking time, algorithms $MIP_{cond-conf}$ and $MIP_{opt}$ exhibited an almost stable behavior while algorithm $MIP_{conf}$ exhibited a decreasing linear behavior. This is much the same results as those of the first two series of experiments. There are two differences observed with the previous set of experiments: a) the performance of all algorithms was decreased; b) now $MIP_{conf}$ outperforms the other algorithms later, and especially $MIP_{opt}$ is outperformed when $bper > 0.8$. The above results can be seen in Figure 7.19(a). Concerning recall (see Figure 7.19(b)), precision and size of results (see Figure 7.19(c)) we have absolutely the same picture with the corresponding results of the previous series of experiments.

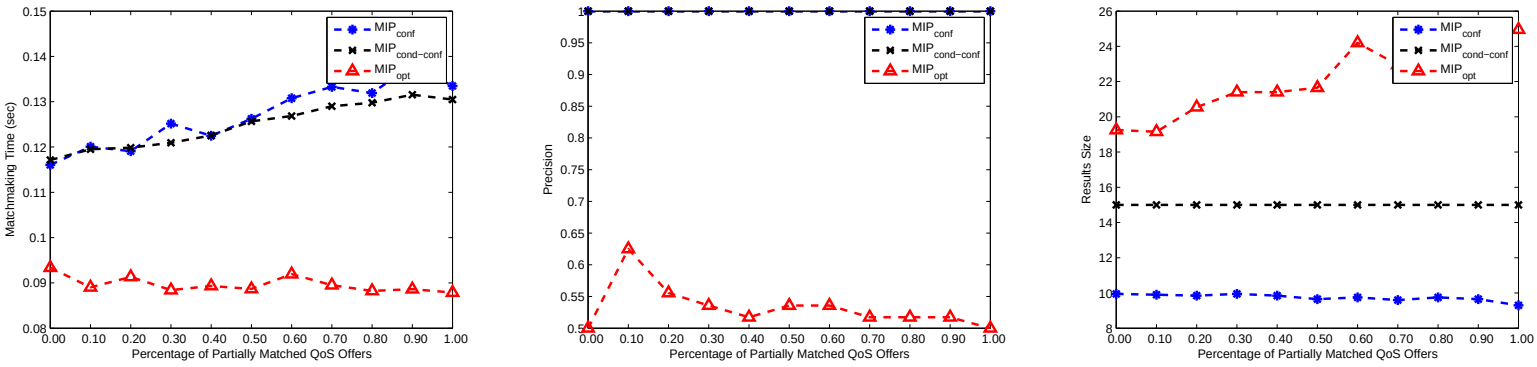
In the fourth experiment, we increased the percentage of *partial* matches among the fixed-sized population of QoS offers ($adscn = 20$). As can be seen in Figure 7.20(a), all algorithms exhibited a not very stable behavior. However, similarly to the corresponding result of the first two series of experiments, the $MIP_{conf}$ and $MIP_{cond-conf}$ algorithms had an increasing behavior while in average the $MIP_{opt}$ algorithm had a stable behavior. Another observation is that it is the first experiment in this series where $MIP_{cond-conf}$ surely outperforms $MIP_{conf}$. Moreover, all algorithms had a decreased performance with respect to the previous set of experiments.

Concerning recall, algorithms $MIP_{cond-conf}$ and $MIP_{opt}$ had recall equal to 1.0 while algorithm $MIP_{conf}$ had recall equal to $1-bmper$. But for precision and size of matchmaking




(a) Matchmaking time (b) Recall (c) Size of matchmaking results

Figure 7.19: 3rd Experiment results



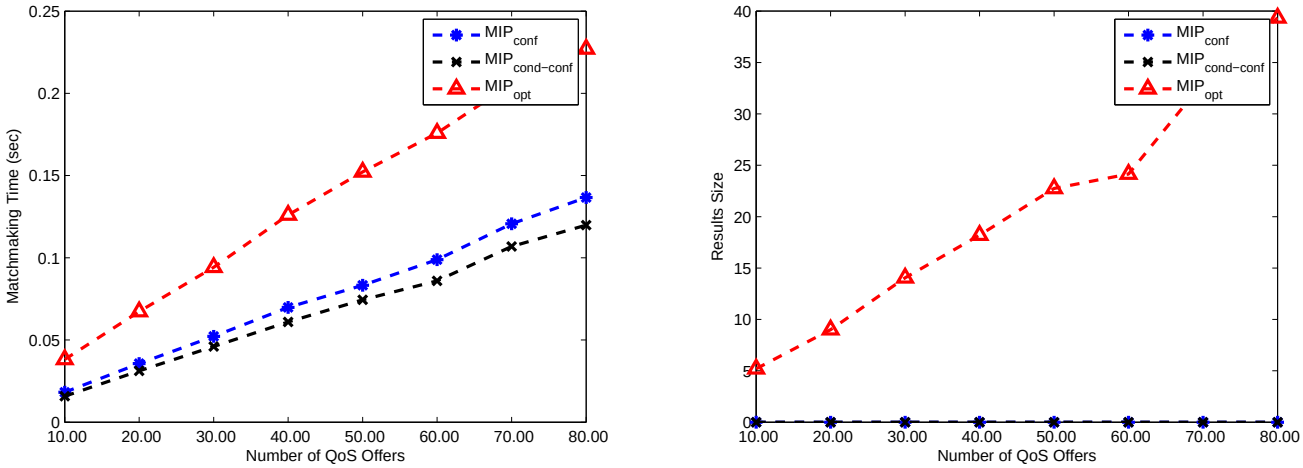

(a) Matchmaking time (b) Precision (c) Size of matchmaking results

Figure 7.20: 4th Experiment results

results, although algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ had precision equal to 1.0 and a constant amount of results, this was not the case for algorithm $MIP_{opt}$. The latter algorithm had its precision decreasing stepwise linearly and the size of matching results increasing stepwise linearly. These results can be seen in Figures 7.20(b) and 7.20(c) and were the same as the ones in the previous set of experiments.

In the last experiment, we had $matchper = 0.0$ and we were increasing the number of QoS offers. We wanted to observe the performance of the algorithms when there are no matches (i.e. the QoS demand is over-constrained). Figure 7.21(a) shows the produced results. This performance is almost the same with the one we discovered in the previous series of experiments. One observation is that $MIP_{cond-conf}$ is much better than $MIP_{conf}$ in accordance with the result of the previous experiment.

In this type of experiment, we cannot define accuracy. However, we can compare




(a) Matchmaking time

(b) Size of matchmaking results

Figure 7.21: 5th Experiment results

these algorithms on the size of conforming results. As expected, algorithms $MIP_{conf}$ and $MIP_{cond-conf}$ do not produce any result. On the other hand, algorithm's $MIP_{opt}$ number of results was increasing linearly, as can be seen from Figure 7.21(b).

Now let us compare the results of this series of experiments with the previous one. First of all, the performance of all algorithms in this series was slightly decreased with respect to the previous one but not on all the experiments. Secondly, now algorithm $MIP_{cond-conf}$ has better or equal performance with $MIP_{conf}$, so the order of these algorithms was reversed with respect to the previous series of experiments. Besides these differences, the other results remain the same. So, from all of these series of experiments, we can can safely conclude that the $MIP_{opt}$ algorithm is the most stable algorithm and of the quickest ones but it suffers from precision problems (it returns more results than needed). Another good choice should be the $MIP_{cond-conf}$ algorithm as it is very quick especially when there are a lot of "insensitive" metrics, it has perfect accuracy but it does not offer advanced categorization of results. Finally, comparable to these two algorithms is the $MIP_{unary}$ algorithm that has perfect accuracy and offers advanced categorization of results but it has a very bad performance with respect to the other algorithms. Thus, based on this analysis we cannot definitely say that we have a "winning" algorithm as this depends much on the user preferences especially regarding the trade-offs we have identified.

### 7.2.6 Randomized Evaluation of CP-based Matchmaking Algorithms

In our fourth set of experiments we used exactly the same testing framework that was used in the first series of experiments producing unary QoS specifications. However, our focus was now to compare and experimentally evaluate the CP and eCP-based (QoS-based)

matchmaking algorithms. So we implemented the four CP-based matchmaking algorithms – $CP_{conf}$, $CP_{cond-conf}$, $CP_{expl-cconf}$ and $CP_{expl-unary}$ – introduced in chapter 6 using the Choco constraint engine, an engine capable of solving both regular and explanation-based constraint models. However, in order to use the eCP solving facilities of Choco, we had to constraint our experiments only with integer variables.

Another thing that must be mentioned concerns the way all the algorithms were implemented. For the first two algorithms, $CP_{conf}$ and $CP_{cond-conf}$, based on our analysis of the metrics of *conformance* and *conditional conformance*, we had to solve a chain of CSPs with each CSP differing from the previous and the next one only with the addition of one constraint and the removal of another one. However, differently from the previous series of experiments that used the MIP technique and with the eCP approach of the next two algorithms, we had to create each CSP from the chain from scratch, as the underlying constraint network after reaching a solution could not be updated with the new changes unless it was totally destroyed and restructured. So we were taking into account this CSP rebuilding time for these two algorithms (and the next two) when calculating their execution time. Moreover, based on the previously analyzed fact and on the fact that MIP solving is quicker than CP solving for linear constraints, we were expecting much worse execution times for these two algorithms with respect to their MIP counterparts.

Fortunately, the next two algorithms, $CP_{expl-cconf}$ and $CP_{expl-unary}$, had a better "fate" as thanks to the eCP technique they used they did not have to create a new CSP from scratch but just add and remove one constraint. However, in contract to the $CP_{expl-cconf}$ algorithm that was finishing when a solution was found, the $CP_{expl-unary}$ algorithm had to create a new CSP after a solution was found. But, again, the latter algorithm did not have to create all the CSPs it needed to solve but just few of them (and only when a solution was found for the previous CSP), with respect to the first two algorithms. So the $CP_{expl-cconf}$ algorithm was creating only one CSP each time it was called and the $CP_{expl-unary}$ was creating some but not all of them. In order to be fair, we were also taking into account the CSP building time and the additions and removals of constraints for these two algorithms. Thus, based on this analysis, we were expecting that $CP_{expl-unary}$ algorithm will be slightly better (according to execution time) than the first two algorithms ($CP_{conf}$ and $CP_{cond-conf}$) and that $CP_{expl-cconf}$ will be much better.

Before presenting and analyzing our experimental results, it must be highlighted that we expect that the only speedup that the $CP_{expl-cconf}$ and $CP_{expl-unary}$ algorithms will have with respect to the other two algorithms is due to the faster modeling of successive CSPs and not to the actual explanation facility. This is because we have only used unary constraints in our experiments, so the explanation facility cannot offer anything but itself. However, in the near future we want to investigate the impact of n-ary constraints to the performance of our (e)CP-based matchmaking algorithms.

### Results

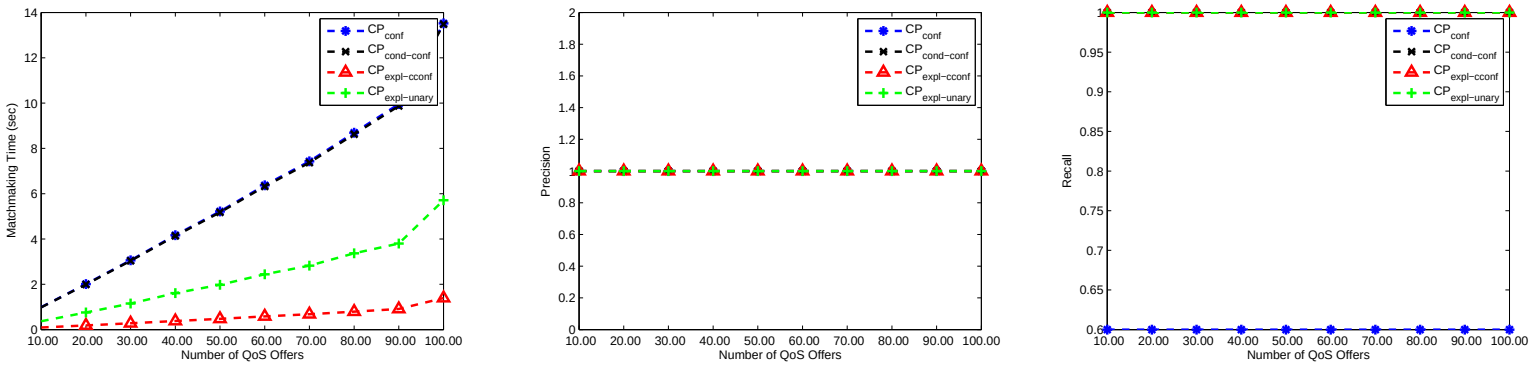
In order to experimentally evaluate the performance and accuracy of the our (e)CP-based QoS-based WS matchmaking algorithms in different settings, a series of six (6) experi-

ments were conducted. Each experiment had the following values for the tuning parameters: $adscn = 10$, $metrcn = 10$, $matchper = 0.5$, $bmper = 0.4$, $partper = 0.5$, $arity = 1$ and $hardstat = 1$, unless otherwise stated. Each experiment was repeated for 30 times. Note that at each experiment a tuning parameter was increasing its value according to a specific step until a specific upper limit.

In the first conducted experiment, we increased the number of QoS offers in order to observe the performance of the matchmaking algorithms. We expected a linear increasing performance behavior for all algorithms. Indeed, as can be seen in Figure 7.22(a), this is the case. In addition, algorithm $CP_{expl-cconf}$ was the fastest followed by algorithms $CP_{expl-unary}$, $CP_{cond-conf}$ and $CP_{conf}$ in decreasing order of performance. This was predicted from the analysis we did previously in this subsection. However, as can be easily seen from Figure 7.22(a), algorithms $CP_{cond-conf}$ and $CP_{conf}$ had exactly the same performance. This result was not actually expected but it can be explained as follows: a) we did a modeling mistake by creating the same amount of CSPs for both algorithms; b) in average $CP_{conf}$ will solve at most one CSP more than the $CP_{cond-conf}$ algorithm. This result was repeated in all experiments as it will be seen in the sequel. Another important remark is that the $CP_{expl-unary}$ algorithm is at least twice as fast than the plain CP algorithms while the $CP_{expl-unary}$ algorithm is 7 times faster (than the plain CP algorithms). So the speedup in execution time because of the faster modeling in the eCP algorithms is already substantial. You can imagine the speedup/time difference that will be caused if also n-ary constraints are present.

As far as accuracy is concerned, all algorithms had $precision = 1.0$, something totally expected. Recall was also 1.0 for all algorithms except from algorithm $CP_{conf}$ that had recall always equal to 0.6 (the usual case for specific values of the tuning parameters) (see Figure 7.22(c)). As far as size of matchmaking results is concerned, algorithms $CP_{cond-conf}$, $CP_{expl-cconf}$ and $CP_{expl-unary}$ always returned the right number of matchmaking results while algorithm $CP_{conf}$ returned 40% less results. Precision was the same for all algorithms in all experiments (except the one where it cannot be measured) while recall was again the same for all algorithms in all experiments (except the one where the percentage of super matches is varying). Thus, in the sequel, we are going to show figures for these two comparison metrics only for the two experiments were the previously analyzed situation changes.

In the second experiment, we increased the number of QoS metrics in order to observe the performance of the matchmaking algorithms. As can be seen in Figure 7.23(a), all algorithms had an exponential increasing performance behavior and their performance order did not change. So this is something different from what we experienced in the results of the MIP-based algorithms where the increase was linear. However, we cannot come to a safe conclusion that CP-based matchmaking algorithms do not scale well with respect to the number of QoS metrics as this may be problem of the Choco constraint engine and the way it is implemented. Another remark is that now the speedup of the eCP algorithms has increased with respect to the plain CP algorithms compared to the results of the first experiment. Concerning precision, the situation is not changed, something totally expected. Concerning size of matchmaking results, Figure 7.23(b) shows that all algorithms had a con-




(a) Matchmaking time (b) Precision (c) Recall

Figure 7.22: 1st Experiment results

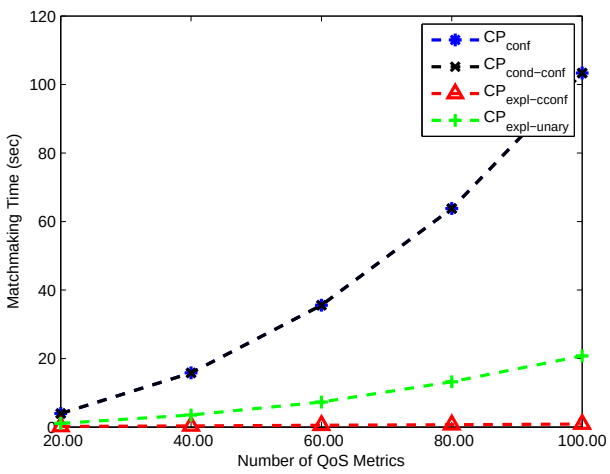
stant behavior. Moreover, algorithms $CP_{cond-conf}$, $CP_{expl-cconf}$ and $CP_{expl-unary}$ returned the right number of results (5) while algorithm $CP_{conf}$ returned less results (0.6*5=3).

In the third experiment, we increased the percentage of matched QoS offers among the fixed-sized population of QoS offers ($adscn$ = 20, $metrcn$ = 20). As can be seen in Figure 7.24(a), algorithm $CP_{expl-unary}$ exhibited a linear decreasing behavior while algorithms $CP_{conf}$, $CP_{cond-conf}$ and $CP_{expl-cconf}$ exhibited a linear increasing performance behavior. The behavior of the latter three algorithms is easy to explain as these algorithms do more work when there are more matching QoS offers. The behavior of the $CP_{expl-unary}$ can also be explained: when more matching QoS offers are present, then this algorithm builds less new CSPs because the matching QoS offers construct inconsistent CSP sequences (so they require the construction of just one CSP). Two remarks must be made: a) the speedup of algorithm $CP_{expl-cconf}$ with respect to the plain CP algorithms is the same as in the first experiment; b) when all QoS offers are matching, the performance of the two eCP algorithms gets the same.

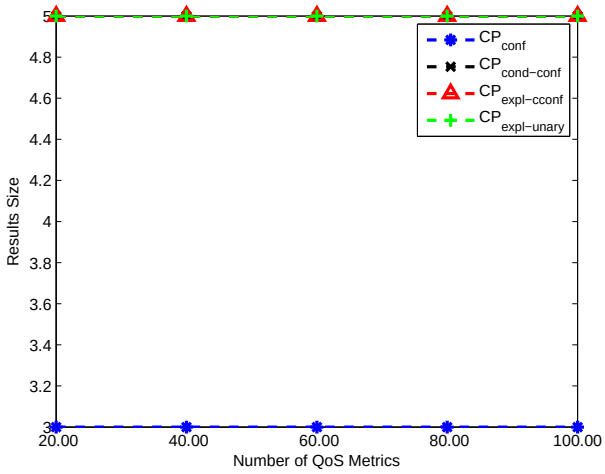
As far as precision is concerned, the situation has not changed. Finally, all algorithms had a linear increase in the size of the matching results, as can be seen from Figure 7.24(b).

In the fourth experiment, we increased the percentage of *super* matches among the fixed-sized population of (matching) QoS offers ($adscn$ = 20, $metrcn$ = 20). Concerning matchmaking time, only algorithm $CP_{expl-cconf}$ had a stable behavior (something rational as it does not care if a matching offer is *exact* or *super* with respect to the number of CSPs it has to build) while all others had a small linear increase in their behavior. Algorithm $CP_{expl-unary}$ had the smallest amount of increase as it only had to build one additional CSP for each additional *super* QoS offer. The behavior of algorithms $CP_{cond-conf}$ and $CP_{conf}$ was totally unexpected as the former should have a stable behavior while the latter a linear decreasing behavior. Moreover, we cannot easily explain why this happened.

Concerning precision, the situation is not changed as all algorithms have precision equal

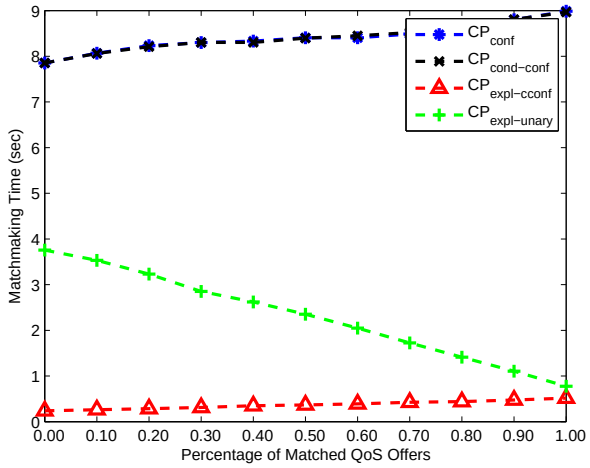



(a) Matchmaking time

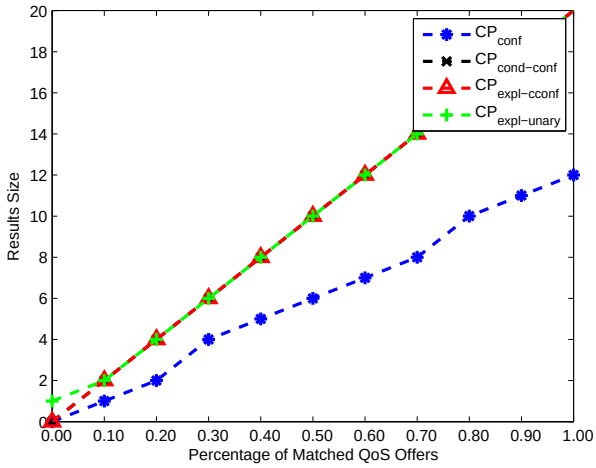



(b) Size of matchmaking results

Figure 7.23: 2nd Experiment results

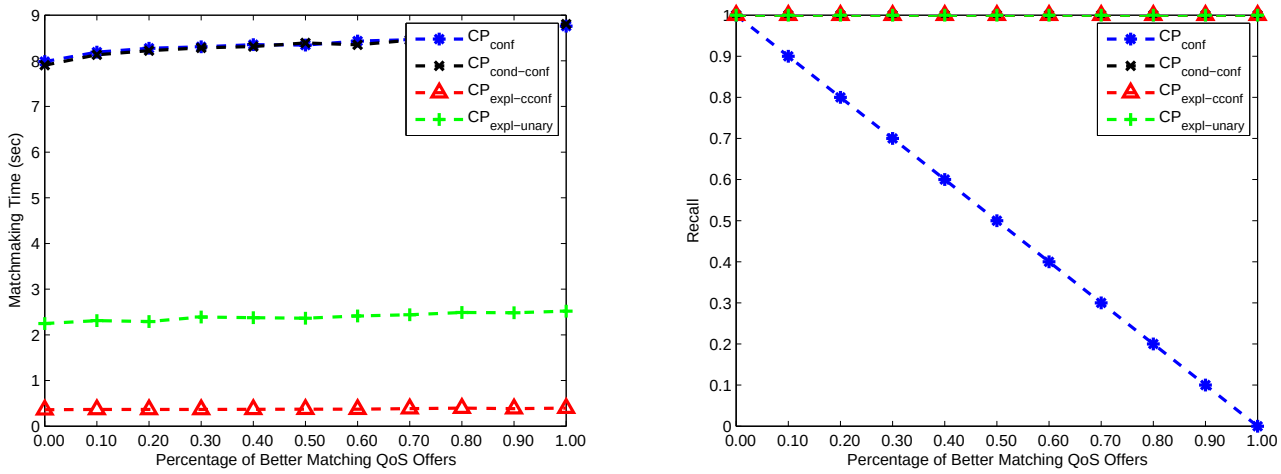



(a) Matchmaking time




(b) Size of matchmaking results

Figure 7.24: 3rd Experiment results




(a) Matchmaking time

(b) Recall

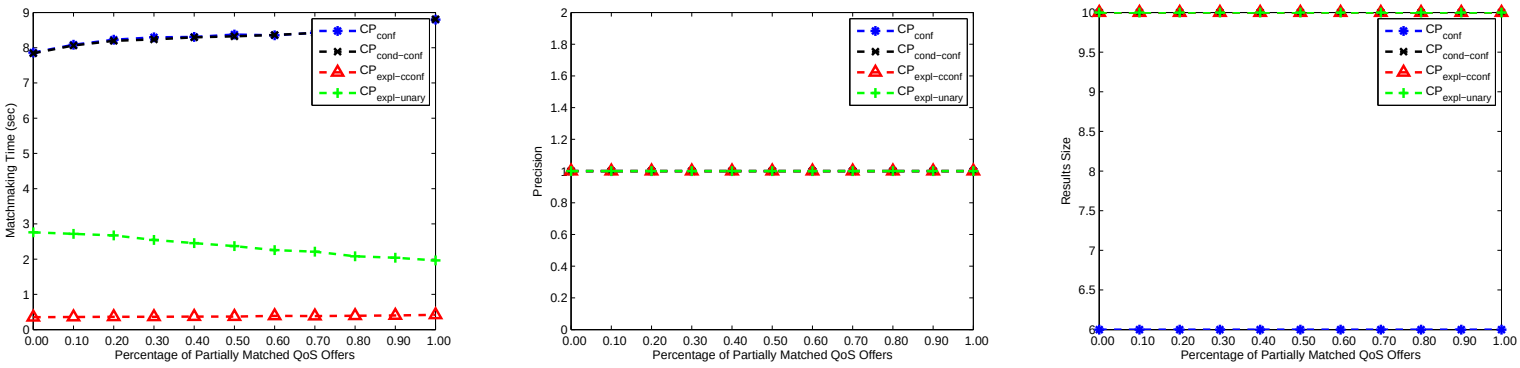
Figure 7.25: 4th Experiment results

to 1.0. However, recall is changed only for algorithm $CP_{conf}$ as this algorithm regards *super* QoS offers as false positives. So this algorithm had a linear decreasing behavior as it can be seen from Figure 7.25(b). Finally, as far as the size of matching results is concerned, obviously all algorithms were returning the same amount of matching results except from algorithm $CP_{conf}$ that was returning less and less results due to the false negatives effect.

In the fifth experiment, we increased the percentage of *partial* matches among the fixed-sized population of (failing) QoS offers ($adscn = 20$, $metrcn = 20$). As can be seen in Figure 7.26(a), algorithm $CP_{expl-cconf}$ has again a stable behavior (as it does not care if an offer is *partial* or *fail*), algorithm $CP_{expl-unary}$ had a small linear decrease in its behavior while the rest of the algorithms exhibited a linear increase in their behavior. Algorithm's $CP_{expl-unary}$ behavior can be explained as it has to create less CSPs for a *partial* offer with respect to a *fail* one. The behavior of algorithms $CP_{conf}$ and $CP_{cond-conf}$ can also be explained as they have to solve more CSPs for a *partial* offer with respect to a *fail* one.

Concerning recall, algorithms $CP_{cond-conf}$, $CP_{expl-cconf}$ and $CP_{expl-unary}$ had recall equal to 1.0 while algorithm $CP_{conf}$ had recall equal to $1 - bmper$. Concerning precision, all algorithms had a perfect precision as in the rest of the experiments (except the last one). Finally, the size of matching results was stable for all algorithms. The latter two results can be seen in Figures 7.26(b) and 7.26(c).

In the last experiment, we had $matchper = 0.0$ and we were increasing the number of QoS offers. We wanted to observe the performance of the algorithms when there are no matches (i.e. the QoS demand is over-constrained). Figure 7.27(a) shows the produced results. As can be seen, all algorithms exhibited a linear increasing behavior and their order was not changed. Moreover, the speedup ratio (of the eCP algorithms with respect to the plain CP ones) remained the same like in all other experiments (except the second one).




(a) Matchmaking time (b) Precision (c) Size of matchmaking results

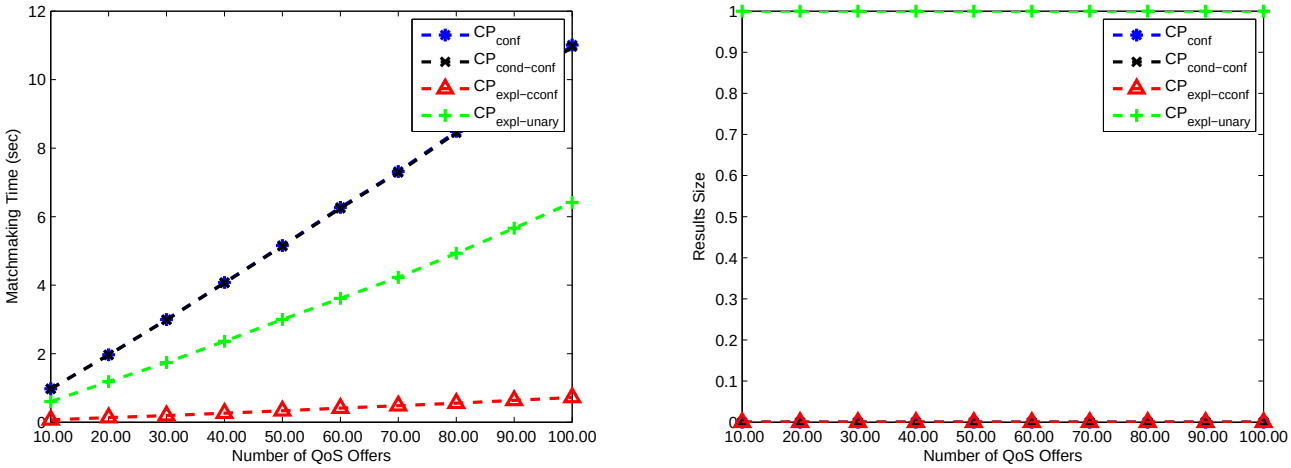
Figure 7.26: 5th Experiment results

In this type of experiment, we cannot define accuracy. However, we can compare these algorithms on the size of conforming results. As expected, algorithms $CP_{conf}$, $CP_{cond-conf}$ and $CP_{expl-cconf}$ do not produce any result. On the other hand, algorithm $CP_{expl-unary}$ always produces one result, as can be seen from Figure 7.27(b).

Let us now shortly discuss the results of this series of experiment. As it was already analyzed, the eCP QoS-based WS matchmaking algorithms are better than their plain CP counterparts because they do not have to create many CSPs in order to perform the matchmaking. Moreover, their advantage/speedup could get greater if we had modeled n-ary constraints in our experiments in order to take advantage of the actual facilities of the eCP technique. So we plan to do this extended modeling and evaluation in the near future. Concerning each algorithm's performance, algorithm $CP_{expl-cconf}$ is the quickest and most stable one, it has better accuracy but it does not offer advanced categorization of the results. Comparable to the latter algorithm is the $CP_{expl-unary}$ one as it has perfect accuracy, it offers advanced categorization of results but it is at least five times slower than the $CP_{expl-cconf}$. The other algorithms (the plain CP ones) do not have great performance and do not offer advanced categorization of results, so they are surely one level below their eCP counterparts. Thus, as it can be understood, we have reached the same situation as with the analysis of the experimental evaluation of the MIP-based QoS-based WS algorithms. There is no "winning" algorithm and the choice of the best one always depends on the user preferences concerning accuracy, execution time and richness/usability of the matchmaking results.

## 7.3 Conclusions

In this chapter, we shortly analyzed our QoS-based WS discovery engine (registry) that is currently under development by providing its architecture and analyzing the role and




(a) Matchmaking time

(b) Size of matchmaking results

Figure 7.27: 6th Experiment results

functionality of each component. Of course, our framework is incomplete as it only supports some of the basic functionalities of a WS registry. However, we plan in the near future to extend it and to make it more complete.

The remaining content of this chapter was devoted to our extensive experimental evaluation of the matchmaking techniques and algorithms we have implemented. Firstly, we started evaluating the performance of the MIP and CP techniques according to the matchmaking metric of conformance. The results showed that the MIP technique is by far better when only linear constraints are present in QoS-based WS specifications. Then, the results of four series of conducted experiments were presented and analyzed. The first three series of experiments were devoted to the random and semi-random experimental evaluation of our implemented MIP-based QoS-based WS matchmaking algorithms. The results showed that all of our implemented algorithms could be used in different settings according to user preferences. The last set of experiments was devoted to the random experimental evaluation of our implemented (e)CP QoS-based WS matchmaking algorithms. The results showed that the eCP technique speeded up the performance of our algorithms. In addition, similarly to the results of the previous series of experiments, the two proposed eCP matchmaking algorithms could be used in different settings/scenarios according to user preferences concerning execution time, accuracy and richness/usability of the matchmaking results.

# Chapter 8

# Conclusion and Future Work

This chapter has the role to conclude this dissertation by providing an overview of its content, by reviewing this thesis contributions and achievements and by drawing directions for further research concerning the research and work of this thesis. There are three sections articulating this chapter, each one with its own goal. Section 8.1 reviews the whole dissertation by analyzing the content of each chapter. Then, next section 8.2 analyzes and discusses the main contributions of this thesis. Finally, section 8.3 explicates all the possible ways in which our thesis research work can be extended.

## 8.1 Dissertation Overview

This dissertation had the sole purpose to describe and analyze in the best possible way the research path towards the achievements of this thesis. It actually reveals the way we approached the general problem and what were our motives, how the research gaps we discovered led us to the specific problem we dealt with, how the literature review of the specific problem tracked down the main research gaps and led us to produce a set of requirements, the approach we followed in order to solve the main problems and our main achievements, the results of the experimental evaluation of our devised and implemented algorithms and how these results prove our theoretical analysis, and the architecture of the implemented framework and the functionality that is exposed by its components. All of the above content was carefully designed and analyzed in seven chapters, each one consisting of many sections and subsections, going from the most general concepts and their analysis to the most specific ones. In the sequel, we are going to analyze in short the main chapters purpose and content without revealing details of the main contributions of this thesis as these are analyzed in the next section.

Chapter 1 explained the way our research started from the general problem and finished at the specific one while it also revealed the main research gaps. Then it analyzed in small

detail what were the main contributions of our thesis and how they closed this research gap. Moreover, it explicated the impact of our contributions and achievements concerning the Web Services research area. Finally, it concluded by providing a short overview of the chapters that followed. As it can be seen, this chapter was independent of the others and its sole purpose was to intricate the user's attention in reading the whole dissertation.

The next chapter, chapter 2, introduced to the reader the main concepts and terms used throughout this dissertation in order to make this dissertation self-contained. In this way, the reader could easily follow up with the content of the remaining chapters. Concerning its content, this chapter explained in detail what is Web Service description and discovery and which entities are involved in these two processes while it also analyzed in detail what are the main requirements imposed on them. By doing so, this chapter not only introduced the main concepts and terms but also enabled the reader to understand and be able to categorize our main thesis contribution according to the current research on WSs. In addition, this chapter provided a small introduction to the two solving techniques that we used for QoS-based WS matchmaking: MIP and CP.

Chapter 3 provided a comprehensive analysis of the conceptualization of QoS for WSs and of the main domain-independent QoS attributes. In addition, an example of domain-dependent QoS attributes in the Traffic Monitoring application domain was also provided. Finally, the role that QoS can play in the management of WSs was elaborated. This chapter had the same role as the previous one that is to fix the terminology used and enable the reader to follow up with the rest of this dissertation and understand the benefits of our thesis.

The next chapter, chapter 4, revealed the main requirements that must be satisfied by the implementing frameworks and the entities involved in order for the processes of QoS-based WS description and discovery to be successful, accurate and complete. These requirements were set after reviewing related work in QoS-based WS description and discovery and the results of this review were analyzed in the next chapter. In our opinion, the content of this chapter was very important as it not only established a reviewing framework for research curried out in the areas of QoS-based WS description and discovery but also guided the design, features and implementation of our main thesis contributions.

Chapter 5 provided a literature review of the main research approaches in QoS-based WS description and discovery based on the requirements set that was established in the previous chapter and revealed the main contributions and deficiencies of these approaches. In addition, at the end of this chapter, comparison tables for the processes of QoS-based WS description, matchmaking and selection were provided and displayed showing in short-form how each approach performed according to the requirements we have set for the corresponding process. In our opinion this summarizing facility was a very important tool for enabling the reader to comprehend what were the exact research problems-gaps in each of the three under-investigation processes.

The next chapter, chapter 6, was the most important chapter of the whole dissertation. Firstly, it explicated what instance of the main research problem was solved by the provision of assumptions and restrictions. Then it analyzed in great detail what were the main contributions of this thesis and how they solved the identified research gaps. For each contri-

bution, apart from the main analysis, images and examples were provided in order to assist the user in comprehending the main notions, procedures and solutions. Additionally, at the end of the analysis of each contribution, there was a concluding subsection summarizing the contribution's achievements and providing a table exactly the same as the one in chapter 5 showing if all requirements set in chapter 4 were satisfied by the contribution and at what grade or point.

Finally, chapter 7 presented the architecture of the framework we are building and explained the functionality of each implemented or not-yet-implemented component. Apart from the functionality, the main tools used to built or assist each component were also described. Moreover, this chapter analyzed the series of experiments we performed and discussed the results uncovered by pointing out what was expected and what was not and why it happened. In addition, it described in great detail the framework we have implemented that was used for testing the performance and accuracy of the QoS-based WS matchmaking algorithms.

## 8.2 Main Contributions of Our Research

We started our PhD thesis by *conducting a long research on the areas of WS description and discovery*. Based on this research, we proposed a set of requirements that should be satisfied by the WS description and discovery frameworks. By taking into account these requirements, we reviewed related work and came into some important conclusions. Actually, this research identified many research gaps that we could study and solve. The appropriate definitions of the WS description and discovery processes and the analysis of their requirements and of the reviewed related work can be found in [Kri04]. In this dissertation, only the definitions and some of the requirements were analyzed.

From all the identified research gaps of the previous research, we were attracted by those gaps that dealt with QoS. QoS-based WS description and discovery was a very interesting and hot topic as there were not so much research approaches dealing with it with not very satisfying results and a lot of unexplored areas. So we decided to nominate this subarea of WS description and discovery as the main object our research in our PhD thesis.

Thus, we started exploring this area by reviewing related work and studying the main techniques that were used to solve partly the main problem. From all of this research, we *identified a set of requirements that should be satisfied by the QoS-based WS description and discovery frameworks* [KP07b]. Actually, these requirements were used as a reviewing framework for comparing related work and identifying places where further research should be conducted. After reviewing related work in QoS-based WS description and discovery, we discovered that there were two main problems to be solved:

- None of the state-of-the-art research approaches was using a semantic, rich and extensible QoS description model for WSs that was satisfying in the best possible way the description requirements we had set.

- Based on the previous problem, the QoS-based WS matchmaking algorithms were

exposing low precision and recall with respect to the results they were returning to the user. In addition, these algorithms were using wrong matchmaking metrics, not all of them provided advanced categorization of results, some of them experienced performance problems and none of them provided useful results back to the user in case this user was using over-constrained QoS demands.

Moreover, as it can be seen, the solution to the second problem depends in a significant percentage on the solution adopted to the first one. Apart from the two main problems, another problem to address was that most QoS-based WS selection algorithms were using a very simple and naive model of calculating the score/rank of a QoS offer that: a) was not considering the best performance of a WS, and b) was independent of and was neglecting the constraints of the QoS demand. Moreover, these selection algorithms were also experiencing the same problems with the matchmaking ones considering the accuracy of the results. The scope of this thesis was to address [Kri05] both of these two main problems (and of course the third one) in the best possible way while also taking into account their interconnection. To this end, the following three main contributions were achieved: *semantic QoS-based WS description*, *alignment of QoS-based WS specifications* and *semantic QoS-based WS discovery*.

Our first contribution was guided by the description model requirements we had set. Based on these requirements, *we carefully designed and implemented an upper ontology for QoS-based WS description*, which was called OWL-Q [KP06, KP08c, KP07a, KP08e]. This ontology combined the best disciplines, requirements and parts of all related research efforts in order to describe in a syntactic and semantic way all possible parts of QoS for WSs. It was an ontological description carefully designed into several facets that could easily be extended and enriched. This ontological description also complemented OWL-S, a World Wide Web Consortium (W3C) submission for semantic functional WS description, by subsuming OWL-S concepts (the sub-concept of the OWL-S concept is a QoS concept) or relating them to QoS concepts. As OWL-Q is just an upper-level ontology, *a mid-level ontology was also produced* for those QoS attributes that are common across application domains. Moreover, *a low-level ontology was produced* for those QoS attributes that are specific to the *Traffic Monitoring* application domain. This application domain was chosen in order to experimentally evaluate our developed algorithms in realistic scenarios where QoS-based WS descriptions were randomly created based on constraints on metrics of specific QoS attributes. In addition, OWL-Q was extended with SWRL rules because: **a)** relations between temporal properties like duration [HP04] had to be expressed and reasoned about; **b)** operations or comparisons on metrics had to be restricted according to the scale that they use; **c)** integrity constraints between property facts and/or instance facts had to be enforced (e.g. in lists we have prevented cycles with the help of rules); **d)** compatibility or equivalency of scales and compatibility of metrics' value types were expressed by OWL property facts fired by rules; **e)** rule-based algorithms like the metric matching one were specified with rules. Finally, by comparing OWL-Q with all other research approaches, we showed that OWL-Q is by far the most rich of all languages that respects in the best possible way the description requirements we have set.

The use of semantics provides users with the ability to specify in a clear, consistent and

machine-processable and interpretable manner the concepts of their domain. However, it cannot prevent some problems [MFRW00] from happening. One such problem is based on the fact that people have different conceptualization of the same entities [BBB+04]. In result, users may produce different ontological specifications of the same concept. Concerning the QoS domain, this can be true for QoS metrics and not for QoS attributes or measurement units that are more or less standardized. This argument is also strengthen by the fact that the same QoS metric can be produced either from high or low level readings of the instrumentation of the system hosting the WS (depending on this system capabilities). In the former case, this QoS metric is produced from other QoS metrics while in the latter case, this QoS metric is a resource metric. Based on the above reasons, we argued that the state-of-the-art ontology alignment algorithms [BBB+04] cannot be used in our case because of the use of mathematics for deriving complex QoS metrics from simpler ones. For this reason, *our second contribution* consisted of the following two algorithms: a) *a semantic QoS metric matching algorithm* [KP06, KP07d, KP07c, KP07e, KP07a, KP08e, KP08c] that derives if two QoS metrics specified in different OWL-Q descriptions are equivalent and b) by using this matching algorithm, *an alignment algorithm* [KP07d, KP07c, KP07e, KP08e, KP08c, KP08a] for aligning OWL-Q descriptions of WS providers and requesters based on their specified QoS metrics and on new metrics created by past measurements produced by WS monitoring systems.

The QoS metric matching algorithm is composed of three main rules, each corresponding to a different case in a two metrics comparison. The last rule – used in composite-to-composite metric matching – reaches the final point of either a) simplifying the difference of two derivation mathematical expressions by using a mathematical engine in order to see if it equals zero or b) solving a CSP which contains the following constraint: $|expr_1 - expr_2| > threshold$, where $|x|$ is a function calculating the absolute value of $x$, $expr_1$ and $expr_2$ are the two derivation mathematical expression of the two composite QoS metrics respectively and $threshold$ is a user supplied limit ranging between $[10^{-1}, 10^{-12}]$. In the first case, symbolic computation techniques are used in order to see if the two mathematical metric derivation expressions are equivalent. In the second case, the user does not care if the mathematical derivation expressions are equivalent but if they can produce very close values. This QoS metric matching algorithm was evaluated based on a specific scenario taken from the *Traffic Monitoring* application domain and by using the symbolic computation techniques of the Matlab mathematical engine. The experimental results showed that although the execution time of the metric matching algorithm is not very satisfactory, this is compensated by the increase in the accuracy of the matchmaking algorithms.

The proposed alignment algorithm consists of two main subalgorithms/processes: the *global alignment process* and the *local alignment* one. When any QoS-based WS offer or demand is issued to this thesis framework, the *global alignment* process is executed on it in order to align it with all specifications already processed and stored in such a way that all metric-to-metric comparisons are minimized. As it can be seen, this alignment process actually uses the QoS metric matching algorithm. The *local alignment* process is executed on a QoS demand and to those QoS offers that are related to it (these are the offers of WS returned from the functional discovery process) so as to ensure that all QoS offers use at least

the same set of QoS metrics with respect to the QoS demand. To achieve the latter goal, this process uses mathematical derivation or statistical strategies or combinations of these two strategies. Unfortunately, we did not complete the implementation of the alignment algorithm because of some reasoning problems we encountered with the reasoning engine we were using and because we did not find a QoS metric monitoring system that could provide us with the appropriate measurement facts. However, when implemented, it is obvious that this alignment process/algorithm will do its best to increase the accuracy of the QoS-based WS matchmaking (and selection) process by inferring the equivalent QoS metrics of two QoS-based WS specifications based on the available knowledge and measurement values.

As can be seen from above, our first two main contributions try to produce QoS-based WS specifications that have the maximum amount of common QoS metrics. This result was in favor of the CP-based category of approaches [CMDTT05, DSL04] in QoS-based WS matchmaking with the respect to the ontology-based one [ZCL04, OVSH06, GZ07]. The main reason was that in this category of approaches the common QoS metrics of a QoS offer and demand are transformed to the same CSP variables used to solve the CSP (matchmaking) problem created by the constraints of the QoS offer and demand. By considering also the fact that the CP-based approach is generally faster than the ontological one, we argued that the CP-based approach is better and must be chosen as a starting point in solving our second research problem. *So our third contribution was to enhance the CP-based approach*, especially the work of [CMDTT05] that uses a more correct matchmaking metric than [DSL04]. However, due to the nature of the value types of the QoS metrics, our first step in this third contribution was to perform a survey on the available solving techniques as CP has some difficulties in dealing with real-valued variables. So based on the research results of the MIP and CP fields, *we concluded that MIP must be used when only linear constraints are present on QoS-based WS specifications while CP must be used when also non-linear constraints are present.* Our conclusion was experimentally proven [KP08d] by a comparison of these two techniques on linear and non-linear constraint problems (CSPs and MIPs) created according to the criteria specified in [CMDTT05] and by using the matchmaking metric of conformance [CMDTT05].

*The second enhancement* was the use of an appropriate and more correct matchmaking metric [KP07d, KP08b, KP08d] called "conditional conformance" that takes into account the fact that some user constraints should not be used in matchmaking. For example, there is no reason why there must be an upper bound on a metric of availability *availability* (case of an "insensitive" metric) for WS requester constraints while both bounds on a metric of *response time* (case of "sensitive" metric) must be respected as the requester's system may not be able to process a very fast provided output (e.g. in case of video streaming WSs and WS requesters having mobile phones). Based on the above reason, we theoretically proved that our matchmaking metric has perfect accuracy with respect to the other matchmaking metrics that exhibit lower accuracy. Apart from this theoretical analysis, we implemented MIP and CP matchmaking algorithms implementing the conformance and conditional conformance metrics that we named $MIP_{conf}$, $CP_{conf}$ and $MIP_{cond-conf}$, $CP_{cond-conf}$ respectively. We firstly experimentally evaluated the MIP-based algorithms with the help of *a testing framework* [KP08d] we have developed that produces randomized QoS-based WS specifications

in a control way so as to be able to test the performance and accuracy of our implemented algorithms. *This testing framework can be considered as a very good contribution* as it is accepts as parameters the matchmaking algorithms to be evaluated and can be altered easily so as to change what kinds of specifications it produces and provides them as input to the matchmaking algorithms. In fact all experimental evaluations we performed used this testing framework. Back to the evaluation of the MIP-based algorithms, the first series of experiments was conducted with completely randomized unary QoS specifications and with all QoS metrics to be "insensitive". The results [KP08d] showed that the $MIP_{cond-conf}$ algorithm was not only significantly faster than the $MIP_{conf}$ one but also exhibited better (actually perfect) accuracy. The second series of experiments was conducted with semi-random unary QoS specifications that contained domain-independent and domain-dependent QoS metrics (taken from the Traffic Monitoring application domain) which were not only "insensitive" but also "sensitive". The results revealed again that the $MIP_{cond-conf}$ algorithm had better accuracy than the $MIP_{conf}$ one but the performance gap was closed. Finally, the CP algorithms were evaluated by only one series of experiments conducted with completely randomized unary QoS specifications and with all QoS metrics to be "insensitive". The results were similar to those of the second series of experiments of the evaluation of the MIP-based algorithms.

Another enhancement was the use of optimization techniques especially in cases where there were over-constrained QoS demands. In many cases, users over-estimate the capabilities of a system and pose unrealistic constraints to it or to discovery brokers in order to find it. In our situation, based on the previous observation, the QoS capabilities of WSs are over-estimated by WS requesters. So the state-of-the-art QoS-based WS matchmaking systems fail to produce any useful result to those WS requesters as the constraint problems solved are all infeasible. In this thesis, *we devised and implemented two different optimization algorithms* [KP07d, KP08e] used to deal with this case. The first one exhaustively solves many constraint problems in order to categorize a n-ary QoS offer according to a unary QoS demand but exhibits perfect accuracy and produces an advanced categorization of results. We have implemented both MIP and CP versions of this algorithm that we have called $MIP_{unary}$ and $CP_{unary}$. The second algorithm solves one optimization problem for each n-ary QoS offer in order to find the worst score this offer can produce and one optimization problem for the n-ary QoS demand for the same reason. This second algorithm does not have perfect accuracy but produces two different categorizations of results and is the fastest from all other algorithms. This was at least shown by the experimental evaluation of all the MIP algorithms based on randomized [KP08d] and semi-randomized inputs with our testing framework. The latter evaluation also showed that the $MIP_{unary}$ is the slowest of all MIP algorithms. So each of these two proposed algorithms has its strengths and weaknesses and their selection depends on user preferences on matchmaking time and accuracy.

The last enhancement concerned the situation where non-linear constraints are present in the QoS-based WS advertisements so the CP-based approach must be used. In this case, we have devised and implemented two new algorithms that take into account our new matchmaking metric and techniques from a specific subarea of CP called explanation-based programming (eCP) [VJ05, RvBW06]. These algorithms are just eCP versions of

the CP algorithms $CP_{cond-conf}$ and $CP_{unary}$ and we have named them $CP_{expl-cconf}$ and $CP_{expl-unary}$ respectively. The random evaluation of these two algorithms against the simple ones (without eCP) showed the significant gain in matchmaking time. The selection of these two new algorithms depends again on user preferences concerning matchmaking time and advanced categorization of results.

Concerning, QoS-based WS selection, some of our proposed algorithms already solve optimization problems and thus also perform the selection subprocess of WS discovery. On the other hand, the algorithms, that do not offer this functionality and is not clear from their results which QoS offer is better, should be accompanied with a QoS-based WS selection algorithm. To this end, *we have proposed and implemented a QoS-based WS selection algorithm* [KP06] that computes the score of each advertised QoS offer by solving two optimization problems - a) one finding the score of the worst performance of the offer, and b) one finding the score of the best performance of the offer - and then computing the weighted average of the two optimization results. This algorithm is better than all other selection algorithms proposed so far not only because it takes into account the best performance of a QoS offer and not only the worst one in order to produce its overall score but because of the following reasons: a) it satisfies all requirements we have set for QoS-based WS selection algorithms; b) it respects and does not neglect the user constraints on all QoS metrics; c) it can produce the score of a QoS offer expressed with n-ary constraints and not only with unary ones. Unfortunately, we did not experimentally evaluate this selection algorithm against the other proposed ones as not only there was limitation of time but also we could not find practical comparison criteria except from the obvious one of execution time.

Apart from devising and implementing QoS-based WS alignment and discovery algorithms, *we have constructed also a registry* encompassing all of them. In this way, we actually *offer a simple but functionally adequate semantic framework for QoS-based WS description and discovery.* In the next section, we will see ways this framework can be extended in order to become more complete and offer more advanced functionalities.

## 8.3 Future Work

Research performed in terms of this thesis has provided a set of solutions to the research problem under investigation. However, as it was analyzed in chapter 6, these solutions focus on a specific instance of the main research problem. The role of this section is to uncover all the possible ways our work can be extended in order to become more general, practical and complete. By analyzing each extension, not only practical solutions are proposed but also many directions for further research are unveiled. This section is organized into as many subsections as the extensions we envision for our work.

### 8.3.1 Tools

Scientists usually try to produce results of great research importance using prototype programs that are very hard to understand and execute. This is also true for Web Services. Users who want to develop or request and execute a WS come across a situation where they have to understand some XML languages and create XML files based on them; produce many lines of code for just to register a WS advertisement or to send a query, etc. While WS technology has evolved, this situation is now changing. However, very few tools have been developed for the semantic WS description and none for the QoS specification and querying.

The object of research is more automation for the QoS-based WS description and discovery processes and less user involvement in these processes. Therefore, there is an ongoing need for the implementation of (visual) tools. These tools should help the user, whether he is a SP or WS requester, to describe or query QoS for WSs, to understand and analyze the QoS requirements or performance of his application and the impact QoS has on the resources used and to view the results of his query in an appropriate way. All of these activities based on these tools must be done with the less possible human intervention. These tools should also connect with each other in order to cater for a complete QoS-based WS description and discovery process. Finally, these tools should cooperate with the tools that have been developed and are used in functional WS description and discovery.

The purpose of this PhD was to implement as many of these tools as possible, but only after the main research goals were achieved. As the latter task took us a lot of time, apart from implementing our research prototype framework we did not have the time to really implement these tools. But we acknowledge that by only implementing these tools, our work will become a complete and practical semantic framework for QoS-based WS description and discovery. The most important research results can become useless if they are not exploited by appropriate tools that help users in performing their tasks in a quick and user-intuitive way.

### 8.3.2 Registry Implementation

In terms of this thesis, a prototype registry has been partially implemented supporting QoS-based WS description and discovery. Being a prototype, this registry does not support many of the requirements we have set in chapter 2 and section 2.2.4. These requirements are generally applicable to WS registries, either they support functional and/or QoS-based WS discovery. Our registry, in its current form, supports the publication and storage of QoS-based WS advertisements and their querying based on user-specified QoS requests. If it is extended with mechanisms for deleting or updating part of or whole QoS offers, then it can be said that this registry supports the basic functions of a registry. In the sequel, we are going to analyze the most important mechanisms that must be implemented as extensions to our registry in order to make it more advanced and industry-oriented. Moreover, based on this analysis, some important research problems are unveiled.

One category of extensions that should be added to our registry concern user queries.

The first extension mechanism is the *support of persistent and volatile queries. Persistent queries* are queries that must be stored and re-assessed each time there is a change in service descriptions that affects them. Concerning QoS offers, this type of service specifications needs special treatment as not only in its entirety but also its parts can be updated very often. The main reason is that QoS is dynamic in nature. So a QoS metric value can change or the whole QoS offer can change (e.g it is not available due to software malfunction). It is important that these changes are not only performed on the QoS specifications documents (through API calls or by the registry itself if it has the capability) but also reflected on the affected persistent queries. To support a QoS offer that entirely changes is a very easy task as if it is deleted, then the result sets of the affected persistent queries are updated by removing the appropriate entry, while if it is entirely updated then all (more generally related) persistent queries should be reassessed in order to include it or delete it or leave it in their result sets. However, when only parts of a QoS offer are updated, then it is not wise to completely re-execute the persistent queries on this QoS offer. In our opinion, the mechanisms offered by the explanation-based CP should be used in this case. So what we actually propose is the following: Along with the persistent query, all the eCSP matchmaking problems (executed in favor of this query) should be stored in their current state. In this way, when a QoS offer is partially updated, then these updates will be added to the appropriate matchmaking eCSPs and these eCSPs will be reassessed. So, based on the facts that eCSPs have stored the previous reasoning (solving) consequences (implied constraints) and that a small number of the constraints of the persistent query need to be checked for matchmaking, then their solving will not start from scratch and matchmaking time will be severely reduced. Thus, the total time for updating all persistent queries will be reduced and the computing resources of the registry will be saved. It must be noted that as a QoS offer could be a match for one persistent query and a non-match for another one, if this QoS offer was updated then the eCSPs matchmaking problems concerning this QoS offer and these two queries should all be reassessed. So this QoS offer may not be a match any more for the first query and a match for the second one.

As persistent queries introduce a significant overhead in the WS registry, which could be even more if our approach for updating them is adopted, they must be erased after a significant time period either specified by the user (if he has initially determined it) or predetermined by the registry itself. In addition to time periods, another additional strategy would be to erase persistent queries if they are not accessed very frequently by the initiating users. So *persistent queries must be volatile also.* Another mechanism that must be supported by a QoS-based WS registry is *caching of user queries.* The only and most important reason for caching user queries (along with their results) is that the same or similar query may be issued later on by the same or different users. Equality in queries is translated to exact matchmaking if we consider one of the two queries as a QoS offer and the other one as a QoS demand. So in this case the results of the stored query should be returned as the results of the new and equal query. For similarity, we should consider two cases: a) if the new query is a *super* match of the cached one, then surely the matched offers of the cached query are also matched offers for the new one, while the *partial* results (of the cached query) should be reexamined for the new query in a similar way as the one described in

the previous paragraph, b) if the new query is a *partial* match of the cached one, then the not-matched results of the cached query are also not-matched results for the new one, while the matched results (of the cached query) should be reexamined. Cached user queries must be updated in the same way as the one introduced in the previous paragraph. They should also be deleted when they are not matched by similar or equal queries for a specific time period.

*Access control and concurrency mechanisms* are obligatory extensions of our registry. Access control is required as service providers or requesters should be let to hide some part of their QoS information from service requesters or providers. For example, some security information of a WS should not be revealed to its customers because it may enable malicious users to perform more focused DoS attacks on this WS. Concurrency mechanisms are needed for two reasons: to control the update/query ratio and concurrent actions. Actions that alter or alter and read concurrently the same QoS information (e.g constraints or other parts of QoS offers) should be performed in a transactional manner. In addition, user queries should be let to occur more often than updates. The reason for this latter necessity is that updates are more costly than queries and because querying is the most important function of a registry.

Based on the analysis of the user querying mechanisms, it is easily understood that *soft-state publication* facilities should be supported by a QoS-based WS registry. These mechanisms allow the registry to store any kind of (published) information for a specific time period and then discard it if its publication is not renewed for another period. So these mechanisms are really needed to relieve the registry from the enormous amount of stale, invalid or useless information which also causes the spending of useful computing resources, especially in case of QoS-based WS alignment and discovery. Renewal may be caused by both service providers or requesters or by the registry itself. So these mechanisms are really needed for enforcing the lifetime of both persistent but volatile queries but also of the cache-based queries. However, not only queries (QoS demands) but any other information published or queried or produced in the registry should be soft-state published. Especially, parts of QoS offers or whole QoS offers should be kept in the registry in this form. The reasons for this are the following:

- a WS may become unavailable or updated with new software or withdrawn so a QoS offer stored will become *stale* or invalid information. This QoS offer should be renewed in logical time periods (or when the actual event happens) by the WS provider or otherwise deleted. An appropriate input in the publication API of the registry should be anticipated or a default time period should be used instead;

- a WS provider may predict with the use of appropriate tools the QoS offer(s) he is going to publish for his WS. However, this prediction is always time-based (e.g QoS offer may be valid for one day, one week or one month). In this case, the WS provider may use a specific field in his QoS offer in order to declare this lifetime period or the actual expiration date. Alternatively, he may provide a specific value in the appropriate input parameter of the registry's publication API;

- QoS is dynamic in nature. So, in many cases some of the metrics used to measure

QoS properties may get new or updated values. Thus, some constraints of some QoS offers must be altered. For application domains that are highly dynamic in nature (e.g SMS sending services to mobile phones) it may become impossible to measure some highly dynamic QoS properties (like *availability*). The solution to this problem is the following: Parts of a QoS offer are soft-state published according to a specific time period (may be specific of the application domain) and then a local registry function or a WS operation (either of the same WS or of the same provider or of a third-party measurement WS) is called to get the up-to-date value or constraint of a metric and renew the publication of the specific part of the QoS offer in question. This is the case where soft-state publication mechanisms function in concert with the *dynamic service configuration* mechanisms.

From the last reason and its analysis it is obvious that *dynamic service configuration* mechanisms must also be supported by a QoS-based WS registry.

*Replication/distribution mechanisms*: The majority of current research approaches implementing WS registries focus on centralized components. WS descriptions are stored in a central repository that has to be queried in order to discover WSs. Centralized systems exhibit many problems like being single points of failure and hotspots in the network and exposing vulnerability to malicious attacks. Moreover, as their computing and spacial capacity is limited, they are unscalable. The latter problem gets worse in highly dynamic environments where many WSs become available or unavailable and new service providers can register themselves and their services in any time point. The solution to the above problem is to make registries distributed by splitting and/or replicating their functionality and/or stored information into many nodes interconnected through the network. A P2P infrastructure has become the most common decentralized registry approach implemented by many research approaches [SSDN02]. This decentralized approach follows a distributed data strategy, by distributing valuable information to several nodes according to a specific pattern, and a distributed query mechanism strategy, in order to route user requests only to those nodes that contain relevant data. We believe that a P2P infrastructure approach should also be adopted for QoS-based WS registries, either these registries are contained in WS functional registries or they are standalone independent software applications, for the above analyzed reasons. More specifically, we advocate that QoS offers should be distributed and replicated in many registry nodes according to the application domain they cover, the service provider that registers them, the functionality of their WS, the capacity of each node, and many other factors. In this way, QoS-based WS discovery will not take long as a user query will be routed based on the previous factors only to those nodes that contain QoS offers that are related to the user query. Based on the matchmaking technique used, a registry could adopt specialized distribution mechanisms. For example, in case when matchmaking a QoS offer with a QoS demand or solving an optimization problem takes a lot of time, especially when the CS(O)P is sparse and loose, the registry should also consider to use Distributed Constraint Programming (DCP) [RvBW06] techniques in order to solve the CS(O)P problem. In DCP, the variables of the problem are distributed to the nodes of the CSP engine which become responsible of them and then the problem is solved by using the techniques of synchronous or asynchronous backtracking [RvBW06].

### 8.3.3 Context-aware Service Discovery

In chapter 4 and section 4.1, the notion of *Class of Service* was introduced and analyzed. To refresh your memory, classes of service can differ in usage privileges, service priorities, response times guaranteed to consumers, verbosity of response information, etc. The concept of classes of service also supports different capabilities, rights, and needs of potential consumers of the Web Service, including power and type of devices they execute on. Further, different classes of service may imply different utilization of the underlying hardware and software resources and, consequently, have different prices. Additionally, different classes of service can be used for different payment models, like pay-per-use or subscription-based.

While the above definition of a Class of Service is very general, as it includes constraints on all possible non-functional characteristics of a WS, we were constrained into the notion of a Service Offering that only included QoS (and price) constraints. So a further step of this PhD work is to include all possible non-functional characteristics of WSs in WS descriptions. In this way, not only WS descriptions will be richer but also the WS discovery process will be more precise as it will be aware of more characteristics of WSs. One set of non-functional characteristics of WSs (and other entities) that should be added in WS descriptions is called *context*. As defined before, context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application including the user and the application themselves. From this definition of context, it is clear that QoS is a subclass of context. This characterization is also justified in [SLP03], where context interoperability is analyzed. So, if we model context and incorporate it into WS descriptions, we also model QoS and incorporate it into WS descriptions. Thus, context modeling is a step towards a richer and more accurate WS Description process.

While context modeling caters for a better and richer WS description, what are the benefits of the WS discovery process? These benefits are identified with the reasons of having a system being context-aware. *Context-awareness* [Dey00] is defined as a property of a system that uses context to provide relevant information and/or service to the user, whereby relevancy depends on the user's task. So let us now look of what are the benefits of context-aware services. In general, traditional services perform their functionality based on explicit inputs and ignore context. Consider a simplified traditional telephone service. A telephone call would be made based on the dialled telephone number. The result would be a connection to the recipient. Contextual information, such as whether the recipient is busy (e.g. in a meeting) is completely ignored. In a context-aware service discovery, this information could be incorporated in the service functionality. In this way, the call would be forwarded to the user only if he is available, otherwise the voicemail is activated. Therefore, by using the status of the user (e.g. derived from his calendar) we may be able to provide more added-value services.

To be able to provide CA (context-aware) service, these services need to consider also context inputs besides functional inputs. Therefore, the output of a CA service does now also depend on contextual information. Contextual inputs are provided by context providers such as GPS, weather station, etc. Context providers form a new party in the service provisioning process. Besides context inputs, a service exhibits context itself. Consider for instance a

shopping service. This can be a store located in the city center. The physical location can be stored in the service description and can be used when discovering this service. Thus, when a user requests a shopping service nearby, the location of the user and the service can be matched in order to find all nearby services. Services that exhibit context but do not require context inputs are not considered context-aware because they do not use user context to provide tailored information. However, they can be context-aware discovered.

So CA services use context in order to provide more added-value services to its customers. So a *CA WS discovery* service will provide more added-value services to its requesters. But what are these added-value services? Let us, now, explain the *CA WS discovery* process and the benefits of using context in WS discovery. *CA WS discovery* is the process of finding WSs using context and functional descriptions [BPvS+04]. In the beginning, functional matchmaking is performed in order to find out those WS advertisements that satisfy the functional needs of the requester. Then, context constraints (which also include QoS ones) are used to filter out the functionally equivalent WS advertisements (i.e. the user context is matched against the WS context). In the end, the list of context-plus-functionally equivalent WS advertisements is ordered based on QoS criteria of the requester, which criteria can be derived based on the requester's context. The context of the requester or the WS can be gathered from the requester and the SP respectively or from an independent third-party Context Provider (CP). In the last case, the CP defines his context provisioning service and the owner of the CA WS discovery process pays for invoking it in order to gather the appropriate context. Unfortunately, context gathering results in non price-free CA WS registries. Thus, the WS requester has to pay in order to enjoy the benefits of CA WS discovery.

There are many reasons for using context in the WS discovery process. The most important of them are the following:

- *Request completion* (applies to the service requester side): This process should be performed when the WS requester wants to find services based on contextual criteria. For instance, the requester wants to find shopping services nearby. The matching process looks for location information of the WS requesters and functionally-appropriate WS advertisements. This information is used to match the request and the service. By completing the request with contextual information, the precision of the result is improved since the request becomes information-richer and therefore more relevant results can be returned.

- *Input completion* (applies to the SP side): This process should be performed when services require context inputs that WS requesters do not provide. For instance, the requester wants to find open shopping centers. The matching process looks for these services and concludes that the requester does not provide the current time. It then looks for time services that can provide this contextual information. By completing missing inputs, the recall of the result is improved since the services that would be ignored when not using context, are now returned.

- *Request Completion* (applies to the WS discovery process itself): If the WS requester's context is known or is gathered by the WS discovery process, this information can be

used to derive missing request information like QoS criteria. In result, the request is completed and the whole discovery process is executed resulting in an ordered list of appropriate WS advertisements.

- *Output adaptation*: If the WS discovery process is aware of the capabilities of the device the requester is using, then it can adapt its output. For instance, if it knows that the requester has a device with a ten-line display, then it could send ten pages of ten results of the total 100 results of the matching WS advertisements. In another example, if the requester does not hold any device but there is a nearby printer, then it could send the 100 results to this printer and inform the user about it.

- *Added-value service offering*: If the WS discovery process is aware of the context of the requester, then it could offer more services to the requester. For instance, if a user wants to find a service for his current task and his agenda is available to the WS discovery process, then this process could not only search for the best WS matching the current requester's task but also search for the best WS matching the next requester's task. As another example, if the user wants to find a map service and from his status can be inferred that he is driving, then the WS discovery process could suggest that he may use the best map service's output as an input to a best route service in order to find the best route with the less traffic.

While the benefits of using context in WS discovery are huge, in [BPvS+04] there are several observations about the nature of context from which some issues come out. These issues influence the role that context-aware service discovery can play in improving the quality of the discovery result:

- Context information exhibits a range of temporal characteristics: Context can be classified as static or dynamic. Static context is fixed information such as the gender of a person while dynamic context changes and is for instance location. This provides requirement for context acquisition (e.g. dynamic context has to be regularly acquired).

- Context information is imperfect: Pervasive environments are highly dynamic. This means that distributed information can quickly become out of date. Because of the distributed property of CA systems, high delays can result in faulty contextual information. Furthermore, derivation algorithms of context producers can produce faulty information when inferring from crude sensor inputs.

- Context information is highly interrelated: Several relations are evided between contextual information. For instance, speed can be derived by a time internal and distance and the openness of a store can be inferred using the current time.

- Context has many alternative representations: Often context is obtained from sensors. Before this sensor context information can be used, it has to be processed to generate concepts which can be used by high level applications. Different applications can have different requirements leading to different representations of context.

These observations indicate some limiting factors on how precision and recall rates can increase by introducing context in the discovery process. When the context is imperfect, the precision and recall decreases because incorrect information is used. Furthermore, alternative representations among the service requestor, providers and context providers result in mismatches and decrease precision and recall. The solution to the above problems has not been found in total yet but at least the following steps are essential:

- Limit of computational delays such that the probability of faulty context is minimized.
- Common understanding of context has to be established.
- Context descriptions should be able to encapsulate relations between contexts to use the opportunity for context derivation.
- Context modeling must be performed in a rich and formal way. Ontologies are the best format of description to use.
- Context acquisition and gathering should be based on specific quality criteria like precision, probability of correctness, trust-worthiness, resolution and up-to-dateness. These criteria are now called QoC (Quality of Context).

From the above analysis, it is clear that Context-aware WS description and discovery is a very complicated and hard process where its research is still in its infancy. Our future work will proceed to tackle many of the research issues raised in this subsection.

### 8.3.4 CSP Optimization

In chapter 6, it was described that except from classical and explanation-based CP, other representation formalisms and frameworks can be used for implementing our matchmaking algorithms. Especially for those matchmaking algorithms that must return useful results to over-constrained user queries, two frameworks were distinguished called *Weighted-MAXCSP* [RvBW06] and *Semi-ring CP* [BMR97] for their appropriateness. Unfortunately, we did not find any free or commercial CSP engine offering/implementing these frameworks. Moreover, as we were late in discovering these frameworks, we did not have the time to implement and evaluate them based on the facilities of the CSP engine we had selected. So a future research direction would consist of the following sequential tasks:

1. the implementation of the most important solving approaches/techniques of these two frameworks based on the building facilities of an appropriate CSP engine;
2. the transformation of the classical CS(O)Ps we use for matchmaking to the formalisms used by these two frameworks;
3. the reformulation and re-implementation of the $CSP_{unary}$ and $CSP_{opt}$ matchmaking algorithms based on the facilities each framework provides and the ideas we have expressed in chapter 6;

4. experimental evaluation of the new algorithms along with the previous ones in order to compare them and come to safe conclusions about which one is more appropriate and in which case.

### 8.3.5 QoS-based Web Service Composition

According to the analysis of section 1.3 of chapter 1, our QoS-based discovery algorithms can be exploited in QoS-based WS composition for the design-time and runtime matchmaking and selection of component services based on local QoS constraints [BSND02]. However, when global constraints [ZBD+03] are present in the user QoS requirements, our algorithms can be used as an optional first step for filtering the QoS offers of those WSs that functionally match each task (WS component). Then the execution of a QoS-based WS composition algorithm [ZBN+04, JRGM04, GPEV05, YZL07, MDSR07, BPB08] is also required in order to produce the (part of the) concrete composed WS by also respecting the global constraints.

We have already performed a small critical review of the current state-of-the-art QoS-based WS composition algorithms. Our conclusion is that these algorithms have the following disadvantages:

- They do not take into account inter-provider QoS offers. Apart for offering their WSs individually and advertising the QoS offers of these WSs, service providers (SPs) may cooperate. This cooperation may take the form of a bargain. In other words, the SPs specify in a specific formality to the composition engine that if both of their WSs are used as a part of a composed WS that will be executed, then the cost for using these two (or more) WSs will be reduced by say 20 percent. Composition engines should be able to capture this information and exploit it in the QoS-based composition algorithm used. To the best of our knowledge, none of the QoS-based WS composition algorithms proposed so far takes this information into account and no composition engine uses a language to capture this information.

- There may be cases, especially in sequential execution (sub)paths, where the output of a selected WS component does not exactly match with the input of the next-to-be-executed selected WS component. To solve this problem, composition engines use data-transformation services, thus increasing in this way the execution time of the composed WS. All proposed composition algorithms do not take into account data dependencies that may exist between WSs (performing complementary functions) and the time approximately taken to transform the data from one form to the other. So they produce composed WSs that may ultimately violate their global constraints concerning execution (or response) time. This problem is magnified by the fact that the actual execution of some component WSs (which under-perform) can produce time delays that are finally propagated to the total execution time of the composed WS.

- Most of the state-of-the-art research efforts in QoS-based WS composition do not take into account the time taken for sending an output data as input to the next WS to be executed. For application domains like Grids, this time may be very considerable.

- While SPs of WSs may advertise specific QoS capabilities of their requirements, there are many factors that can cause a WS to violate some of its QoS capabilities. So if these under-performing WSs are part of a composed WS, it is of outmost importance of both to be able to inspect if there will be a global QoS violation and to adapt to this change by changing the structure of the composed WS dynamically. Only two of the most important research efforts have tried to solve this problem, where Zeng et. al. [ZBN+04] use a region-based approach and Canfora et. al. [GPEV05] use a slice-based approach. Of course, these two capabilities, violation inspection and dynamic adaptation, can be realized and implemented into a QoS-based WS composition engine only if a cooperating monitoring system exists that has the specific task of monitoring the runtime execution of the component WSs.
- In many cases, it may be possible that not all of the QoS requirements of the requester can be satisfied based on the available QoS offers so a valid execution plan cannot be reduced. This problem is not confronted at all by all QoS-based WS composition approaches. It can be solved by adopting some of the optimization techniques we have already proposed in QoS-based WS discovery to the QoS-based WS composition process.

Thus, an interesting and very prominent future direction is to either extend one of the already proposed composition algorithms by solving these five crucial research problems or to introduce a new one by using a technique or research approach not exploited yet by these composition algorithms. Additionally, the formal and experimental evaluation of all the composition algorithms (and of the one we would like to propose and implement) is another task that would have a great research impact if implemented as it will not only give insight to the time and space complexity of these algorithms but also reveal cases where one algorithm outperforms the other ones and vice versa.

# Bibliography


[ABFG02] Daniel Austin, Abbie Barbir, Christopher Ferris, and Sharad Garg. Web services architecture requirements. W3c working group note, W3C, 2002. Available at `http://www.w3.org/TR/wsa-reqs`.

[ACKM03] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer, November 2003.

[AGDR03] Rama Akkiraju, Richard Goodwin, Prashant Doshi, and Sascha Roeder. A method for semantically enhancing the service discovery capabilities of uddi. In Subbarao Kambhampati and Craig A. Knoblock, editors, *IIWeb*, pages 87–92, 2003.

[AN02] M. Anbazhagan and A. Nagarajan. Understanding quality of service for web services. IBM Developerworks website, January 2002.

[ASSR93] P. Attie, M. Singh, A. Sheth, and M. Rusinkiewicz. Specifying and executing intertask dependencies. In *VLDB'93: 19th International Conference on Very Large Data Bases*, pages 134–145, Dublin, Ireland, August 1993.

[BBB+04] Thanh Le Bach, Jesús Barrasa, Paolo Bouquet, Jan De Bo, Rose Dieng, Marc Ehrig, Manfred Hauswirth, Mustafa Jarrar, Ruben Lara, Diana Maynard, Amedeo Napoli, Giorgos Stamou, Heiner Stuckenschmidt, Pavel Shvaiko, Sergio Tessaris, Sven Van Acker, and Zaihrayeu. *D2.2.3: State of the Art on Ontology Alignment*. knowledgeweb european project, August 2004. final version 1.2 – available at: `www.starlab.vub.ac.be/research/projects/knowledgeweb/kweb-223.pdf`.

[BBHP04] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–192, 2004.

[BCC+04] Tom Bellwood, Steve Capell, Luc Clement, John Colgrave, Matthew J. Dovey, Daniel Feygin, Andrew Hately, Rob Kochman, Paul Macias,

Mirek Novotny, Massimo Paolucci, Claus von Riegen, Tony Rogers, Katia Sycara, Pete Wenzel, and Zhe Wu. *UDDI Version 3.2.* OASIS, http://uddi.org/pubs/uddi_v3.htm, 2004.

[BCT04] Boualem Benatallah, Fabio Casati, and Farouk Toumani. Analysis and management of web service protocols. In *ER 2004: 23rd International Conference on Conceptual Modeling*, volume 3288 of *Lecture Notes in Computer Science*, pages 524–541, Shanghai, China, November 2004. Springer.

[BDH+03] S. Bechhofer, M. Dean, F. Van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, G. Schreiber, and L. Stein. *OWL Web Ontology Language Reference.* W3C, http://www.daml.org/services/owl-s/1.0/, 2003.

[BHM+02] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web services architecture. W3c working draft, W3C, 2002. Available at `http://www.w3.org/TR/ws-arch`.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, May 2001.

[BMR97] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236, 1997.

[BPB08] I. Brandic, S. Pllana, and S. Benkner. Specification, Planning, and Execution of QoS-aware Grid Workflows within the Amadeus Environment. *Concurrency and Computation: Practice and Experience*, 20:331–345, March 2008.

[BPS01] J. A. Bergstra, A. Ponse, and Scott A. Smolka. *Handbook of Process Algebra.* Elsevier Science Inc., New York, NY, USA, 2001.

[BPvS+04] T. H. F. Broens, S. Pokraev, M. J. van Sinderen, J. Koolwaaij, and P. Dockhorn Costa. Context-aware, ontology-based, service discovery. In *European Symposium on Ambient Intelligence (EUSAI), Eindhoven, The Netherlands*, volume 3295 of *Lecture Notes in Computer Science*, pages 72–83, Berlin / Heidelberg, 2004. Springer.

[BSND02] Boualem Benatallah, Quan Z. Sheng, Anne H. H. Ngu, and Marlon Dumas. Declarative composition and peer-to-peer provisioning of dynamic web services. In *ICDE*, pages 297–308, San Jose, CA, USA, 2002. IEEE Computer Society.

[BW03] Wolf-Tilo Balke and Matthias Wagner. Cooperative discovery for user-centered web service provisioning. In Zhang [Zha03], pages 191–197.

[CBB+02] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. *Documenting Software Architectures: Views and Beyond.* Addison Wesley Professional, 2002.

[CBF04] Ion Constantinescu, Walter Binder, and Boi Faltings. Directory services for incremental service integration. In *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004*, volume 3053 of *Lecture Notes in Computer Science*, pages 254–268, Heraklion, Crete, Greece, May 2004. Springer.

[CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. *Web Services Description Language (WSDL) 1.1.* W3C, http://www.w3.org/TR/wsdl, 2001.

[CCP07] Cinzia Cappiello, Marco Comuzzi, and Pierluigi Plebani. On automated generation of web service level agreements. In John Krogstie, Andreas L. Opdahl, and Guttorm Sindre, editors, *CAiSE*, volume 4495 of *Lecture Notes in Computer Science*, pages 264–278, Trondheim, Norway, 2007. Springer.

[CFK+02] K. Czajkowski, I.T. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In *In 8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2002), LNCS Vol. 2537*, pages 153–183, 2002.

[CIjJ+00] Fabio Casati, Ski Ilnicki, Li jie Jin, Vasudev Krishnamoorthy, and Ming chien Shan. eflow: a platform for developing and managing composite e-services. Technical Report HPL-2000-36, HP Labs, Palo Alto, USA, 2000.

[CLG+06] M.B. Chhetri, J. Lin, S. Goh, J. Yan, J. Y. Zhang, and R. Kowalczyk. A coordinated architecture for the Agent-based Service Level agreement Negotiation of Web service composition. In *In Proc. 2006 Australian Software Engineering Conference, ASWEC'06*, 2006.

[CMDTT05] Antonio Ruiz Cortés, Octavio Martín-Díaz, Amador Durán Toro, and Miguel Toro. Improving the automatic procurement of web services using constraint programming. *Int. J. Cooperative Inf. Syst.*, 14(4):439–468, 2005.

[Con01] UDDI Consortium. Uddi executive white paper. White paper, November 2001. Available at `http://uddi.org/pubs/UDDI_Executive_White_Paper.pdf`.

[CP05] M. Comuzzi and B. Pernici. An architecture for flexible Web service QoS negotiation. In *Proceedings of the 9th IEEE Enterprise Computing Conference*, Enschede, The Netherlands, 2005.

[Cru95] Rene L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1048–1056, 1995.

[CS03] Jorge Cardoso and Amit Sheth. Semantic e-workflow composition. *J. Intell. Inf. Syst.*, 21(3):191–225, 2003.

[CSM+04] Jorge Cardoso, Amit P. Sheth, John A. Miller, Jonathan Arnold, and Krys Kochut. Quality of service for workflows and web service processes. *Journal of Web Semantics*, 1(3):281–308, 2004.

[CSZ92] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: architecture and mechanism. *SIGCOMM Comput. Commun. Rev.*, 22(4):14–26, 1992.

[Dec03] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[Dey00] Anind Kumar Dey. *Providing architectural support for building context-aware applications*. PhD thesis, Atlanta, GA, USA, 2000. Director-Gregory D. Abowd.

[DLS05] Glen Dobson, Russell Lock, and Ian Sommerville. Qosont: a qos ontology for service-centric systems. In *EUROMICRO '05: Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 80–87, Porto, Portugal, 2005. IEEE Computer Society.

[DMR02] Hong Hai Do, Sergey Melnik, and Erhard Rahm. Comparison of schema matching evaluations. In Akmal B. Chaudhri, Mario Jeckle, Erhard Rahm, and Rainer Unland, editors, *Web, Web-Services, and Database Systems*, volume 2593 of *Lecture Notes in Computer Science*, pages 221–237. Springer, 2002.

[DNDPG+07] E. Di Nitto, M. Di Penta, A. Gambi, G. Ripa, and M. L. Villani. Negotiation of service level agreements: An architecture and a search-based approach. In *In Proc. ICSOC'07*, pages 295–306, 2007.

[DOH+01] Marlon Dumas, Justin O'Sullivan, Mitra Hervizadeh, David Edmond, and Arthur H. M. ter Hofstede. Towards a semantic framework for service description. In *Proceedings of the IFIP TC2/WG2.6 Ninth Working Conference on Database Semantics*, pages 277–291, Hong Kong, April 2001. Kluwer, B.V.

[DSGF03] Vikas Deora, Jianhua Shao, W. A. Gray, and N. J. Fiddian. A quality of service management framework based on user expectations. In Orlowska et al. [OWPY03], pages 104–114.

[DSL04] Seema Degwekar, Stanley Y. W. Su, and Herman Lam. Constraint specification and processing in web services publication and discovery. In *ICWS*, pages 210–217. IEEE Computer Society, 2004.

[ea03] K. Sycara et. al. *OWL-S 1.0 Release*. OWL-S Coalition, http://www.daml.org/services/owl-s/1.0/, 2003.

[Elg03] Islam Elgedawy. A conceptual framework for web services semantic discovery. In Robert Meersman and Zahir Tari, editors, *OTM Workshops*, volume 2889 of *Lecture Notes in Computer Science*, pages 1004–1016. Springer, 2003.

[FB02] Dieter Fensel and Christoph Bussler. The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[Fen96] Norman E. Fenton. *Software Metrics: A Rigorous and Practical Approach.* International Thomson Computer Press, Boston, MA, USA, 1996.

[FFH+03] Christian Facciorusso, Simon Field, Rainer Hauser, Yigal Hoffner, Robert Humbel, René Pawlitzek, Walid Rjaibi, and Christine Siminitz. A web services matchmaking engine for web services. In Kurt Bauknecht, A. Min Tjoa, and Gerald Quirchmayr, editors, *EC-Web*, volume 2738 of *Lecture Notes in Computer Science*, pages 37–49, Prague, Czech Republic, 2003. Springer.

[Fis04] M. Fischer. Introduction to web services. Technical report, Sun Microsystems, June 2004. Part of the Java Web Services Tuturial, available at `http://java.sun.com/webservices/docs/1.4/tutorial/doc/index.html`.

[FK98] Svend Frølund and Jari Koistinen. Quality of services specification in distributed object systems design. *COOTS'98: Proceedings of the 4th conference on USENIX Conference on Object-Oriented Technologies and Systems (COOTS)*, 5(4):179–202, 1998.

[FM01] D. Fensel and E. Motta. Structured development of problem solving methods. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):913–932, 2001.

[FMvH+03] Dieter Fensel, Enrico Motta, Frank van Harmelen, V. Richard Benjamins, Monica Crubezy, Stefan Decker, Mauro Gaspari, Rix Groenboom, William Grosso, Mark Musen, Enric Plaza, Guus Schreiber, Rudi Studer, and Bob Wielinga. The unified problem-solving method development language upml. *Knowl. Inf. Syst.*, 5(1):83–131, 2003.

[FNSY96] Donald F. Ferguson, Christos Nikolaou, Jakka Sairamesh, and Yechiam Yemini. *Market-based control: a paradigm for distributed resource allocation*, chapter Economic models for allocating resources in computer systems, pages 156–183. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1996.

[GGPS96] Leonidas Georgiadis, Roch Guérin, Vinod Peris, and Kumar N. Sivarajan. Efficient network qos provisioning based on per node traffic shaping. *IEEE/ACM Trans. Netw.*, 4(4):482–501, 1996.

[GHM+07] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. *Simple Object Access Protocol (SOAP) Version 1.2.* W3C, http://www.w3.org/TR/soap, 2007.

[GPEV05] G.Canfora, M. Di Penta, R. Esposito, and M. L. Villani. An Approach for QoS-aware Service Composition based on Genetic Algorithms. In *Proceedings of the Genetic and Computation Conference (GECCO'05), Washington DC, USA*. ACM Press, 2005.

[GPSH02] Christelle Guéret, Christian Prins, Marc Sevaux, and Susanne Heipcke. *Applications of Optimization with XpressMP*. Dash Optimization Ltd., 2002.

[GPSH08] Nicholas Gibbins, Terry R. Payne, Ahmed Saleh, and Hai H.Wang. A comparison between owl-s and wsmo. Deliverable D1.1 – Annex (WP 1), April 2008. TAO – EU-IST Specific targeted research project (STREP) IST-2004-026460.

[Gun00] Neil J. Gunther. *The Practical Performance Analyst*. Authors Choice Press, 2000. Foreword By-Raj Jain.

[GZ07] Ester Giallonardo and Eugenio Zimeo. More semantics in qos matching. In *International Conference on Service-Oriented Computing and Applications*, pages 163–171, Newport Beach, CA, USA, 2007. IEEE Computer Society.

[HHR+97] Takeo Hamada, Stephanie Hogg, Jarno Rajahalme, Carlo Licciardi, Lill Kristiansen, and Per Fly Hansen. Service quality in tina-quality of service trading in open network architecture. In *EDOC '97: Proceedings of the 1st International Conference on Enterprise Distributed Object Computing*, pages 322–333, Marriott Resort, Gold Coast, Australia, 1997. IEEE Computer Society.

[Hos02a] Wolfgang Hoschek. A unified peer-to-peer database framework for scalable service and resource discovery. In *GRID '02: Proceedings of the Third International Workshop on Grid Computing*, pages 126–144, Baltimore, Maryland, USA, 2002. Springer-Verlag.

[Hos02b] Wolfgang Hoschek. Web service discovery processing steps. In *ICWI'02: Proceedings of the IADIS International Conference WWW/Internet 2002*, pages 255–262, Lisbon, Portugal, November 2002. IADIS.

[HP04] Jerry R. Hobbs and Feng Pan. An ontology of time for the semantic web. *ACM Trans. Asian Lang. Inf. Process.*, 3(1):66–85, 2004.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[JB00] Narendra Jussien and Vincent Barichard. The palm system: explanation-based constraint programming. In *TRICS 2000: Proceedings of the Techniques foR Implementing Constraint programming Systems Workshop, a post-conference workshop of CP 2000*, pages 118–133, Singapore, September 2000.

[JMS02] L. Jin, V. Machiraju, and A. Sahai. Analysis on service level agreement of web services. Technical Report HPL-2002-180, Software Technology Laboratories, HP Laboratories, USA, June 2002.

[JRGM04] Michael C. Jaeger, Gregor Rojec-Goldmann, and Gero Mühl. QoS Aggregation for Service Composition using Workflow Patterns. In *Proceedings of*

*the 8th International Enterprise Distributed Object Computing Conference (EDOC'04)*, pages 149–159, Monterey, California, USA, September 2004. IEEE CS Press.

[KBH+03] Takahiro Kawamura, Jacques-Albert De Blasio, Tetsuo Hasegawa, Massimo Paolucci, and Katia P. Sycara. Preliminary report of public experiment of semantic service matchmaker with uddi business registry. In Orlowska et al. [OWPY03], pages 208–224.

[KF02] Henry M. Kim and Mark S. Fox. Towards a data model for quality management web services: An ontology of measurement for enterprise modeling. In Anne Banks Pidduck, John Mylopoulos, Carson C. Woo, and M. Tamer Özsu, editors, *CAiSE*, volume 2348 of *Lecture Notes in Computer Science*, pages 230–244. Springer, 2002.

[KFS06] Matthias Klusch, Benedikt Fries, and Katia Sycara. Automated semantic web service discovery with owls-mx. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 915–922, New York, NY, USA, 2006. ACM Press.

[KK04] Markus Keidl and Alfons Kemper. Towards context-aware adaptable web services. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 55–65, New York, NY, USA, 2004. ACM.

[KKRO03] Michael Klein, Birgitta König-Ries, and Philipp Obreiter. Stepwise refinable service descriptions: Adapting daml-s to staged service trading. In Orlowska et al. [OWPY03], pages 178–193.

[KL03] Alexander Keller and Heiko Ludwig. The wsla framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11(1):57–81, 2003.

[KP06] Kyriakos Kritikos and Dimitris Plexousakis. Semantic qos metric matching. In *ECOWS '06: Proceedings of the European Conference on Web Services*, pages 265–274, Zurich, Switzerland, 2006. IEEE Computer Society.

[KP07a] Kyriakos Kritikos and Dimitris Plexousakis. Owl-q for semantic qos-based web service description and discovery. In Tommaso Di Noia, Rubén Lara, Axel Polleres, Ioan Toma, Takahiro Kawamura, Matthias Klusch, Abraham Bernstein, Massimo Paolucci, Alain Leger, and David L. Martin, editors, *SMRR*, volume 243 of *CEUR Workshop Proceedings*, Busan, South Korea, 2007. CEUR-WS.org.

[KP07b] Kyriakos Kritikos and Dimitris Plexousakis. Requirements for qos-based web service description and discovery. *compsac*, 02:467–472, 2007.

[KP07c] Kyriakos Kritikos and Dimitris Plexousakis. A semantic qos-based web service discovery algorithm for over-constrained demands. In *NWESP '07: Proceedings of the Third International Conference on Next Generation Web Services Practices*, pages 49–54, Seoul, South Korea, 2007. IEEE Computer Society.

[KP07d] Kyriakos Kritikos and Dimitris Plexousakis. Semantic qos-based web service discovery algorithms. In *ECOWS '07: Proceedings of the Fifth European Conference on Web Services*, pages 181–190, Halle, Germany, 2007. IEEE Computer Society.

[KP07e] Kyriakos Kritikos and Dimitris Plexousakis. A semantic qos-based web service discovery engine for for over-constrained qos demands. In Flavio de Paoli, Andrea Maurino, Ioan Toma, Justin O'Sullivan, Marcel Tilly, and Glen Dobson, editors, *Proceedings of the Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC'07) at the 3rd International Conference on Service Oriented Computing (ICSOC 2007)*, Vienna, Austria, September 17 2007.

[KP08a] Kyriakos Kritikos and Dimitris Plexousakis. Alignment of qos-based web service specifications. In *ICSOC '08: Proceedings of the Sixth International Conference on Service-Oriented Computing*, Sydney, Australia, 2008. Springer. submitted.

[KP08b] Kyriakos Kritikos and Dimitris Plexousakis. Evaluation of qos-based web service matchmaking algorithms. *2008 IEEE Congress on Services*, Part I:567–574, 2008.

[KP08c] Kyriakos Kritikos and Dimitris Plexousakis. *Managing Web Services Quality: Measuring Outcomes and Effectiveness*, chapter VI - Enhancing the Web Service Description and Discovery Processes with QoS. Information Science Reference, October 2008.

[KP08d] Kyriakos Kritikos and Dimitris Plexousakis. Mixed-integer programming for qos-based web service matchmaking. *IEEE Transactions on Services Computing*, 2008. under revision.

[KP08e] Kyriakos Kritikos and Dimitris Plexousakis. Qos-based web service description and discovery. *International Journal of Web Services Practices*, 3(2):72–82, 2008.

[Kri04] Kyriakos Kritikos. Web service description and discovery. Internal report, ICS-FORTH, Heraklion, Crete, Greece, 2004.

[Kri05] Kyriakos Kritikos. Extending owl for qos-based web service description and discovery. In Andreas Hanemann, editor, *Proceedings of the IBM PhD Student*

*Symposium at the 3rd International Conference on Service Oriented Computing (ICSOC 2005)*, volume 169, pages 73–78, Amsterdam, The Netherlands, December 12 2005. CEUR.

[Lab00] Francois Laburthe. Choco: implementing a cp kernel. In *TRICS 2000: Proceedings of the Techniques foR Implementing Constraint programming Systems Workshop, a post-conference workshop of CP 2000*, Singapore, September 2000.

[Ley01] Frank Leymann. Web services flow language (wsfl 1.0). Technical report, IBM Corporation, May 2001 2001.

[Lie94] Sandford Liebesman. *ISO 9000 - An Introduction*, chapter 1. AT&T Corporate Quality Office, USA, July 1994.

[LJL+03] KangChan Lee, JongHong Jeon, WonSeok Lee, Seong-Ho Jeong, and Sang-Won Park. Qos for web services: Requirements and possible approaches. World Wide Web Consortium (W3C) note, November 2003.

[LLA+03] Ruben Lara, Holger Lausen, Sinuhe Arroyo, Jos de Bruijn, and Dieter Fensel. Semantic web services: description requirements and current technologies. In *International Workshop on Electronic Commerce, Agents, and Semantic Web Services, In conjunction with the Fifth International Conference on Electronic Commerce (ICEC 2003)*, Pittsburgh, PA, USA, September 2003.

[LNPM98] Spyros Lalis, Christos Nikalaou, Dimitris Papadakis, and Manolis Marazakis. Market-driven service allocation in a qos-capable environment. In *ICE '98: Proceedings of the first international conference on Information and computation economies*, pages 92–100, Charleston, South Carolina, United States, 1998. ACM.

[LNZ04] Yutu Liu, Anne H. H. Ngu, and Liangzhao Zeng. Qos computation and policing in dynamic web service selection. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *WWW (Alternate Track Papers & Posters)*, pages 66–73, New York, NY, USA, 2004. ACM.

[LSKW02] Yang W. Lee, Diane M. Strong, Beverly K. Kahn, and Richard Y. Wang. Aimq: a methodology for information quality assessment. *Information Management*, 40(2):133–146, 2002.

[MDCG03] Enrico Motta, John Domingue, Liliana Cabral, and Mauro Gaspari. Irs-ii: A framework and infrastructure for semantic web services. In *ISWC 2003: Second International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 306–318, Sanibel Island, FL, USA, October 2003. Springer.

[MDSR07] Arun Mukhija, Andrew Dingwall-Smith, and David S. Rosenblum. QoS-Aware Service Composition in Dino? In *Proceedings of the Fifth European Conference on Web Services (ECOWS'07), Halle (Saale), Germany*, pages 3–12, November 2007.

[MFRW00] Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. In *KR*, pages 483–493, 2000.

[MH03] Soraya Kouadri Mostéfaoui and Béat Hirsbrunner. Towards a context-based service composition framework. In Zhang [Zha03], pages 42–45.

[Moo01] Aad Van Moorsel. Metrics for the internet age: Quality of experience and quality of business. Technical Report HPL-2001-179, HP Labs, August 2001.

[MS02] E. Michael Maximilien and Munindar P. Singh. Conceptual model of web service reputation. *SIGMOD Rec.*, 31(4):36–41, 2002.

[MS04] E. Michael Maximilien and Munindar P. Singh. A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5):84–93, 2004.

[MSZ01] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. Mobilizing the semantic web with daml-enabled web services. In *SemWeb*, Hong Kong, 2001.

[MWG04] Shalil Majithia, David W. Walker, and W. A. Gray. A framework for automated service composition in service-oriented architectures. In *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004*, volume 3053 of *Lecture Notes in Computer Science*, pages 269–283, Heraklion, Crete, Greece, May 2004. Springer.

[NSDM03] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. A system for principled matchmaking in an electronic marketplace. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 321–330, Budapest, Hungary, 2003. ACM Press.

[Oak03] Phillipa Oaks. Towards self-describing web services. In *ICWE'03: International Conference on Web Engineering*, volume 2722 of *Lecture Notes in Computer Science*, pages 476–485, Oviedo, Spain, 2003. Springer.

[OEtH03] Justin OŏSullivan, David Edmond, and Arthur H.M. ter Hofstede. Service description: A survey of the general nature of services. Technical Report FIT-TR-2003-01, Queensland University of Technology, Brisbane, Australia, January 2003. Available at: `http://www.bpm.fit.qut.edu.au/about/docs/service\_description.pdf`.

[OtHE03] Phillipa Oaks, Arthur H. M. ter Hofstede, and David Edmond. Capabilities: Describing what services can do. In *ICSOC'03: First International Conference on Service-Oriented Computing*, volume 2910 of *Lecture Notes in Computer Science*, pages 1–16, Trento, Italy, December 2003. Springer.

[OVSH06] Nicole Oldham, Kunal Verma, Amit Sheth, and Farshad Hakimpour. Semantic ws-agreement partner selection. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2006. ACM Press.

[OWPY03] Maria E. Orlowska, Sanjiva Weerawarana, Mike P. Papazoglou, and Jian Yang, editors. *Service-Oriented Computing - ICSOC 2003, First International Conference, Trento, Italy, December 15-18, 2003, Proceedings*, volume 2910 of *Lecture Notes in Computer Science*, Trento, Italy, 2003. Springer.

[Pap03] Mike P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *4th International Conference on Web Information Systems Engineering (WISE 2003)*, pages 3–12, Rome, Italy, 2003. IEEE Computer Society.

[PKPS02] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. Semantic matching of web services capabilities. In *ISWC '02: Proceedings of the First International Semantic Web Conference on the Semantic Web*, pages 333–347, Sardinia, Italy, 2002. Springer-Verlag.

[PSK03] Massimo Paolucci, Katia Sycara, and Takahiro Kawamura. Delivering semantic web services. Technical Report CMU-RI-TR-02-32, Robotics Institute, Carnegie Mellon University, May 2003.

[PTB03] T. Pilioura, A. Tsalgatidou, and A. Batsakis. Using wsdl/uddi and daml-s in web service discovery. In *ESSW 2003: WWW 2003 Workshop on E-Services and the Semantic Web*, Budapest, Hungary, 2003.

[Ran03] Shuping Ran. A model for web services discovery with qos. *SIGecom Exch.*, 4(1):1–10, 2003.

[RKL+05] Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubén Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Cristoph Bussler, and Dieter Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.

[Ros05] Francesca Rossi. Preference reasoning. In Maurizio Gabbrielli and Gopal Gupta, editors, *21st Internation Conference on Logic Programming (ICLP 2005)*, volume 3668 of *Lecture Notes in Computer Science*, pages 5–8. Springer, 2005.

[RvBW06] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[SA03] Rajesh Sumra and D. Arulazi. Quality of service for web services – demystification, limitations, and best practices. Developer.com website, March 2003.

[SCD+97] Bikash Sabata, Saurav Chatterjee, Michael Davis, Jaroslaw J. Sydir, and Thomas F. Lawrence. Taxomomy of qos specifications. In *WORDS '97: Proceedings of the 3rd Workshop on Object-Oriented Real-Time Dependable Systems - (WORDS '97)*, pages 100–107, Washington, DC, USA, 1997. IEEE Computer Society.

[Sch86] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley, New York, N.Y, USA, 1986.

[SF01] Kavé Salamatian and Serge Fdida. Measurement based modeling of quality of service in the internet: A methodological approach. In *IWDC '01: Proceedings of the Thyrrhenian International Workshop on Digital Communications*, pages 158–174, Taormina, Italy, 2001. Springer-Verlag.

[She03] Amit Sheth. Semantic web process lifecycle: Role of semantics in annotation, discovery, composition and orchestration. In *ESSW 2003: WWW 2003 Workshop on E-Services and the Semantic Web*, Budapest, Hungary, 2003. Invited talk available at: `http://lsdis.cs.uga.edu/lib/presentations/WWW2003-ESSW-invitedTalk-Sheth.pdf`.

[SK03] Biplav Srivastava and Jana Koehler. Web service composition - current solutions and open problems. In *ICAPS 2003 Workshop on Planning for Web Services*, pages 28–35, Trento, Italy, June 2003.

[SLP03] Thomas Strang and Claudia Linnhoff-Popien. Service interoperability on context level in ubiquitous computing environments. In *SSGRR 2003w: International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet*, L'Aquila, Italy, January 2003.

[SRvS03] Marta Sabou, Debbie Richards, and Sander van Splunter. An experience report using daml-s. In *ESSW'03: Proceedings of the Twelfth International World Wide Web Conference Workshop on E-Services and the Semantic Web*, Budapest, Hungary, 2003.

[SSDN02] Mario Schlosser, Michael Sintek, Stefan Decker, and Wolfgang Nejdl. A scalable and ontology-based p2p infrastructure for semantic web services. In *P2P '02: Proceedings of the Second International Conference on Peer-to-Peer Computing*, page 104, Linköping, Sweden, September 2002. IEEE Computer Society.

[SSM+03] Kaarthik Sivashanmugam, Amit P. Sheth, John A. Miller, Kunal Verma, Rohit Aggarwal, and Preeda Rajasekaran. Metadata and semantics for web

services and processes. In W. Benn, P. Dadam, S. Kirn, and R. Unland, editors, *Datenbanken und Informationssysteme: Festschrift zum 60. Geburtstag von Gunter Schlageter*, pages 245–271. FernUniversität in Hagen, Fachbereich Informatik, Hagen, 2003.

[SWKL02] Katia Sycara, Seth Widoff, Matthias Klusch, and Jianguo Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5(2):173–203, 2002.

[TEPP02] Vladimir Tosic, Babak Esfandiari, Bernard Pagurek, and Kruti Patel. On requirements for ontologies in management of web services. In *CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*, pages 237–247, Toronto, Ontario, Canada, 2002. Springer-Verlag.

[TGN+03] M. Tian, A. Gramm, M. Nabulsi, H. Ritter, J. Schiller, and T. Voigt. Qos integration in web services. Gesellschaft fur Informatik DWS 2003, Doktorandenworkshop Technologien und Anwendungen von XML, October 2003.

[TMPE03] V. Tosic, W. Ma, B. Pagurek, and B. Esfandiari. On the dynamic manipulation of classes of service for xml web services. Research Report SCE-03-15, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, 2003.

[TPP03] Vladimir Tosic, Bernard Pagurek, and Kruti Patel. Wsol - a language for the formal specification of classes of service for web services. In Zhang [Zha03], pages 375–381.

[TRPA06] Dimitrios T. Tsesmetzis, Ioanna G. Roussaki, Ioannis V. Papaioannou, and Miltiades E. Anagnostou. Qos awareness support in web-service semantics. In *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, pages 128–134, Guadeloupe, French Caribbean, 2006. IEEE Computer Society.

[TVB00] Didier Le Tien, Olivier Villin, and Christian Bac. Corba application tailored manager for quality of service support. In *ISORC '00: Proceedings of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, page 52, Newport Beach, California, USA, 2000. IEEE Computer Society.

[UG96] Mike Uschold and Michael Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[VHS96] Pascal Van Hentenryck and Vijay Saraswat. Strategic directions in constraint programming. *ACM Computing Surveys*, 28(4):701–726, 1996.

[VJ05] Gerard Verfaillie and Narendra Jussien. Constraint solving in uncertain and dynamic environments: A survey. *Constraints*, 10:253–281(29), July 2005.

[VSJ01] Vesna Bosilj Vuksic, Mojca Indihar Stemberger, and Jurij Jaklic. Simulation modelling towards e-business models development. *International Journal of Simulation Systems*, 2(2):191–225, 2001.

[WA03] WS-AGREEMENT. WS-Agreement Framework. `https://forge.gridforum.org/projects/graap-wg`, September 2003.

[WVKT06] Xia Wang, Tomas Vitvar, Mick Kerrigan, and Ioan Toma. A qos-aware selection model for semantic web services. In Asit Dan and Winfried Lamersdorf, editors, *ICSOC*, volume 4294 of *Lecture Notes in Computer Science*, pages 390–401. Springer, 2006.

[YZL07] Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web*, 1(1):6, 2007.

[ZBD+03] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality driven web services composition. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 411–421, New York, NY, USA, 2003. ACM Press.

[ZBN+04] Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, May 2004.

[ZCL04] Chen Zhou, Liang-Tien Chia, and Bu-Sung Lee. Daml-qos ontology for web services. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 472–479, San Diego, CA, USA, 2004. IEEE Computer Society.

[Zha03] Liang-Jie Zhang, editor. *Proceedings of the International Conference on Web Services, ICWS '03, June 23 - 26, 2003, Las Vegas, Nevada, USA*, Las Vegas, Nevada, USA, 2003. CSREA Press.