Automating the Procurement of Web Services*

Octavio Martín-Díaz, Antonio Ruiz-Cortés,
Amador Durán, David Benavides, and Miguel Toro

Dpto. de Lenguajes y Sistemas Informáticos
E.T.S. de Ingeniería Informática, Universidad de Sevilla
41012 Sevilla, España - Spain
Phone: +34 95 455 3871 Fax: +34 95 455 7139
{octavio,aruiz,amador,mtoro}@lsi.us.es
benavides@us.es

Abstract. As government agencies and business become more dependent on web services, software solutions to automate their procurement gain importance. Current approaches for automating the procurement of web services suffer from an important drawback: neither uncertainty measures nor non-linear, and complex relations among parameters can be used by providers to specify quality-of-service in offers. In this paper, we look deeply into the roots of this drawback and present a proposal which overcomes it. The key point to achieve this improvement has been using the constraint programming as a formal basis, since it endows the model with a very powerful expressiveness. A XML-based implementation is presented along with some experimental results and comparisons with other approaches.

Keywords: software procurement, web services, quality-of-service, traders.

1 Introduction

As government agencies and business become more dependent on web services, software solutions to automate their procurement gain importance. It is generally assumed that decision criteria for choosing software packages stems from the user requirements they should fulfill. There are different types of requirements such as managerial, political, and, of course, quality requirements. There are a number of approaches which automate some activities of the procurement, most of them focus in quality requirements. However, these approaches suffer from several drawbacks that hamper their use when requirements that providers guarantee include uncertainty measures, non-linear and complex relations among parameters. In fact, if we want to achieve a competitive technology based on web services, their quality-of-service is an important issue to be taken into account, becoming one of challenges to be solved in the near future [31].

In this context, software procurement [4,5] becomes web services procurement (WSP), an activity focussed on the acquisition of web services required by a web-

* Supported by the Spanish Interministerial Commission on Science and the Spanish Ministry of Science and Technology under grants TIC2000-1106-C02-01, TIC2003-02737-C02-01 and FIT-150100-2001-78.

M.E. Orlowska et al. (Eds.): ICSOC 2003, LNCS 2910, pp. 91–103, 2003.

service-based system, thus it is a critical activity for current web system developers. Some typical tasks involved in WSP are:

- Specification of demands and offers, which should be checked for consistency in order to verify they do not contain any inner contradiction.
- Search of offers, which should be checked for conformance in order to verify they fulfill the demand, so that the selection is limited to such offers.
- Selection of the best choice according to the assessment criteria which is included in the demand.

In this paper, we present a proposal to automate the procurement of web services. Our proposal improves on others in that it supports a symmetric specification model. Thus, providers can include in their offers requirements as complex as customers include in their demands. The key point to achieve this improvement has been using the constraint programming as a formal basis, since it endows the model with a very powerful expressiveness. A XML–based implementation is presented along with some experimental results and comparisons with other approaches.

The rest of the paper is structured as follows. In Section 2, we introduce the notions of asymmetric and symmetric specification models, as well as an overview of related works. In Section 3, we propose the use of constraint programming as a means of achieving a symmetric specification model. In Section 4, we present briefly the main implementation aspects of our run-time framework, together with some experimental results. Finally, in Section 5 we summarise the presented work and the immediate future work.

2 Symmetric versus Asymmetric Models

2.1 Asymmetric Models

Let S be a multidimensional space whose dimensions are given by domains of quality-of-service parameters. Traditionally, a demand (δ) has been viewed as a subspace in S, whereas an offer (ω) 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 1.a and 1.b, respectively. This checking can be computed easily by evaluating ω in δ. 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 $120 \not> 100$.

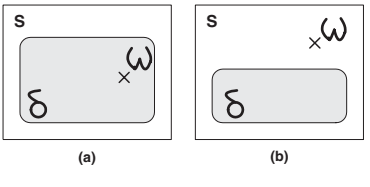

Fig. 1. Conformance in asymmetric models.

This interpretation of conformance results in a model which is asymmetric with regard to the expressiveness of quality-of-service 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.

2.2 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 (ω) is conformant to a demand (δ) whenever the offer's sub-space is inside the demand's sub-space (see Figure 2.a), otherwise the offer is not conformant (see Figure 2.b). As an example, if a web service owns the offer $\omega = \{MTTF >= 120\}$, then it is conformant to the following demand $\delta_1 = \{MTTF >= 100\}$, 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 quality-of-service 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 sub-space, 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.

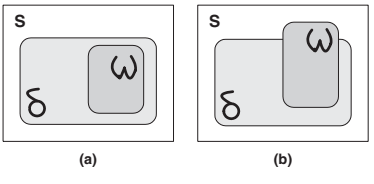

Fig. 2. Conformance in symmetric models.

2.3 Related Work

Figure 3 shows a comparative study among the most prominent (as far as we know) quality-aware approaches to WSP. Briefly:

The Reference Model	Static View: The Lexicon					Dynamic View: The Process Model
	Stakeholders	Quality-of-Service Documents	Catalogues, Parameters & Measures			
			Data Structuring	Customer's	Provider's	
IBM's WSME MME	Providers Costumers	Advertisements Queries Agreements	Data Dictionary: pre-def. basic types sequences records	Name-Value Pair Properties Static/Dynamic Binding Scripts for Rule-based Reqs.		Advertisement/Submission Query/Submission Matchmaking Selecting Providers' Offers
HP's MME Service	Advertisers Requestors	Service Offers & Requests	DAML+OIL Ontology: datatypes and types subsumption	Composition Single-Parameter Constraints on Parameters of Service (expandable)		Advertising Querying Browsing
UDDIe	Providers Consumers	Publishing Inquiry	*Blue Pages*	Single-Par. Conditions on Properties (Qualifiers)	Name-Value Pair Properties	Publishing Search and Discovery
Our Proposal	Providers Costumers	Demands Offers Agreements	Catalogues: pre-def. basic types catalogue extension basic and derived p.	Composition Multiple-Parameter Constraints on Parameters of Service		Creating Catalogues Offers Submission Demands Submission Matchmaking

Fig. 3. A comparison of quality-aware approaches to WSP.

- The *UDDI Extension* (UDDIe) [28] is based on the UDDI (Universal Description Discovery and Integration) services. UDDIe owns an asymmetric model when specifying demands and offers.
- The HP's *Matchmaking Engine* (MME) [10] is based on the DAML (DARPA Agent Markup Language) semantic web language [2]. It is the closest proposal to ours, because it owns a symmetric model to specify quality-of-service, and it uses constraints to do it, so it owns a great expressiveness. As well, it uses a Description Logic DL's solver as a mean of carrying out the WSP-related tasks. Nevertheless, there is not currently any DL's solver version able to process some of the most complex expressions which can be specified in MME.
- The IBM's *Web Services Matchmaking Engine* (WSME) [12], which is related to *Web Service Level Agreement* (WSLA) [15,17], is based on the CORBA/ODP trader service. It owns an asymmetric model and there is no optimation of the selection because search results are only the lists of conformant offers. Nonetheless, there is a difference: relationships between demands and offers are bilateral. In the same way quality-of-service in offers is based on parameter/value pairs whereas demands impose conditions on them, it is also allowed that demands define their own quality-of-service parameters whereas the offers impose conditions on them. As an example, let an offer be given by the following quality-of-service specification $\omega = \{me.MTTF = 120 \ \& \ your.nationality \in \{BE, \ldots, UK\}\}$ and a demand $\delta = \{me.nationality = \{IS\} \ \& \ your.MTTF > 100\}$, then the offer ω is not conformant to the demand δ, because the condition it imposes on the demand (the Europe Union membership) is not fulfilled, despite of the offer fulfills conditions imposed by the demand.
- Other languages for specifying quality-of-service and trader services the *Quality-of-service Modeling Language* (QML) [8], the NoFun language [6], and the CORBA trader service [22]. These proposals are not directly related to WSP.

3 Supporting WSP with Constraint Programming

We have chosen mathematical constraints as the way of specifying quality-of-service in demands and offers. In this way, checking conformance can be carried out just as a constraint satisfaction problem (CSP) or a constraint satisfaction optimisation problem (CSOP) [7,11,18,29]. In general, CSP-based modelling is quite simple and intuitive (in most cases) in the context of problems which we are dealing with. Constraint programming is an excellent support for symmetric specifications models, because it makes possible to check whether a space is included in another one, being these spaces treated as constraints. Our proposal owns a symmetric specification model with a great deal of expressiveness because of using constraints.

3.1 Constraint Programming in a Nutshell

Constraint Programming (CP) has recently attracted high attention among experts from many areas because of its potential for solving hard real-life problems. Not only it is based on a strong theoretical foundation, but it is an attracting widespread commercial interest, as well. Constraints formalise those dependencies in physical worlds and their mathematical abstractions naturally and transparently. A constraint is simply a logical relation among several variables, each taking a value in a given domain. The constraint thus restricts the possible values that variables can take, and it represents a partial information about the variables of interest. An important feature of constraints is their declarative manner, i.e., they specify what relationships must hold without specifying a computational procedure to enforce them. CP is the study of computational systems based on constraints. The idea of CP is to solve problems by stating constraints (requirements) about the problem area and, consequently, finding solution satisfying all the constraints.

The earliest ideas leading to CP may be found in the Artificial Intelligence dating back to sixties and seventies. The scene labelling problem [30] is probably the first constraint satisfaction problem that was formalised. The main step towards CP was achieved when Gallaire [9] and Jaffar & Lassez [14] noted that logic programming was just a particular kind of constraint programming. The basic idea behind Logic Programming (LP), and declarative programming in general, is that the user states what has to be solved instead of how to solve it, which is very close to the idea of constraints. Therefore the combination of constraints and logic programming is very natural, and Constraint Logic Programming (CLP) makes a nice declarative environment for solving problems by means of constraints. However, it does not mean that CP is restricted to CLP. Constraints were integrated to typical imperative languages like C++ and Java, as well.

The nowadays real-life applications of CP in the area of planning, scheduling and optimisation rise the question if the traditional field of Operations Research (OR) is a competitor or an associate of CP. There is a significant overlap of CP and OR in the field of NP-Hard combinatorial problems. While the OR has a long research tradition and (very successful) method of solving problems using linear programming, the CP emphasis is on higher level modelling and solutions methods that are easier to understand by the final customer. Most recent advances promise that both methodologies can exploit each

other, in particular, the CP can serve as a roof platform for integrating various constraint solving algorithms including those developed and checked to be successful in OR. As the above paragraphs show, the CP has an inner interdisciplinary nature. It combines and exploits ideas from a number of fields including Artificial Intelligence, Combinatorial Algorithms, Computational Logic, Discrete Mathematics, Neural Networks, Operations Research, Programming Languages, and Symbolic Computation.

Currently, we see two branches of CP, namely constraint satisfaction and constraint solving. Both share the same terminology but the origins and solving technologies are different. The former deals with problems defined over finite domains and, currently, probably more than 95% of all industrial constraint applications use finite domains. Therefore, we deal with constraint satisfaction problems mostly in this paper. The latter shares the basis of CP, i.e., describing the problem as a set of constraints and solving these constraints. But now, the constraints are defined (mostly) over infinite or more complex domains. Instead of combinatorial methods for constraint satisfaction, the constraint solving algorithms are based on mathematical techniques such as automatic differentiation, Taylor series or Newton method.

Constraint Satisfaction Problems [29] have been a subject of research in Artificial Intelligence for many years. A Constraint Satisfaction Problem (CSP) is defined as a set of variables each ranging on a finite domain, and a set of constraints restricting all the values that variables can simultaneously take. A solution to a CSP is an assignment of a value from its domain to every variable, in such a way that all constraints are satisfied at once. We may want to find: i) just one solution, with no preference as to which one, ii) all solutions, iii) an optimal, or at least a good solution, given some objective function defined in terms of some or all of variables. Solutions to a CSP can be found by searching (systematically) through all possible value assignments to variables.

In many real-life applications, we do not want to find any solution but a good solution. The quality of solution is usually measured by an application dependent function called objective function. The goal is to find such solution that satisfies all the constraints and minimise or maximise the objective function, respectively. Such problems are referred to as Constraint Satisfaction Optimisation Problems (CSOP), which consists of a standard CSP and an optimisation function that maps every solution (complete labelling of variables) to a numerical value [29].

3.2 Consistency and Conformance

Whenever a new demand or offer is submitted, its consistency needs to be checked, i.e., whether or not it contains any inner contradiction. This is interpreted as a CSP, so that if the corresponding CSP is satisfiable, then the demand or offer can be considered as consistent. The corresponding CSP for a demand or offer is composed of all the constraints it contains. On the other hand, the best choice selection regarding with a demand implies the previous checking for conformance, because the search is reduced to conformant offers. As we are using constraint programming, checking of conformance lies in determining whether each and every solution to the offer's CSP is also a solution to the demand's CSP.

In this way, the corresponding CSP for checking the conformance is constructed according to the definition given in [18]:

$$conformance(\omega, \delta) \Leftrightarrow sat(c_\omega \wedge \neg\, c_\delta) = false$$

where ω is the offer and c_ω its corresponding CSP, δ is the demand and c_δ its corresponding CSP, and sat is a function that we identify with the CSP solver which is being used. It can be applied on a CSP c so that it returns one of the following results: $true$ if c is satisfiable, $false$ if not, and $\perp$ if the solver cannot determine whether c is satisfiable or not.

3.3 Optimality

More often than not, it is possible to have several offers which are conformant to the same demand for a web service, then we should select that offer which is the best choice. This selection is carried out according to the assessment criteria the customer includes in his or her demand. These criteria may be given by utility functions [3,16,21] which, in general, have the signature $\mathcal{U} : \pi \rightarrow [0, 1]$ where π is the measuring domain of a quality-of-service parameter. Utility functions assign an utility assessment (ranging from 0 to 1) to every quality-of-service value it can take, so the greater the assessment, the better the consideration of the customer. Therefore, utility functions allow the establishment of an objective criteria, given by customers, in order to select those offers which better fulfill the demands. Figure 4 shows several utility functions corresponding to examples in this section.

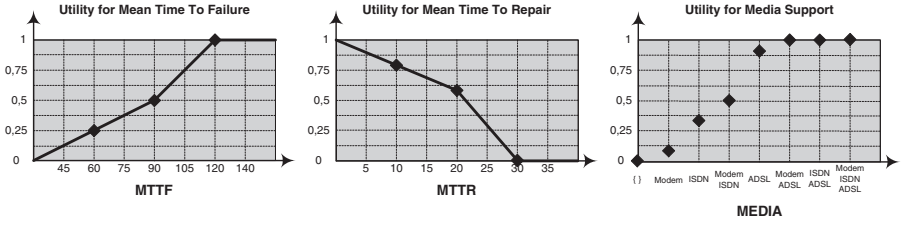

Fig. 4. Utility functions for $MTTF$, $MTTR$, and $MEDIA$.

Although we can make use of any kind of function to specify utility functions, linear piecewise functions are often the preferred. As an example, the utility function for a numeric quality-of-service parameter can be defined by means of polylines determined by a sequence of coordinate points such as $(x_1, u_1), (x_2, u_2), \ldots, (x_n, u_n)$, where every x represents a value in the measuring domain of the quality-of-service, and u its assessment in the range [0,1]. The corresponding utility function is then given by:

$$\mathcal{U}(x) = \begin{cases} u_1 + \frac{u_2 - u_1}{x_2 - x_1}(x - x_1) & \text{if} \quad x_1 \leq x < x_2 \\ \cdots & \\ u_{n-1} + \frac{u_n - u_{n-1}}{x_n - x_{n-1}}(x - x_{n-1}) & \text{if} \quad x_{n-1} \leq x \leq x_n \\ \perp & \text{if } x < x_1 \text{ o } x > x_n \end{cases}$$

We are not usually interested in computing the utility assessment of an unique quality-of-service parameter, but on maximising the global assessment of offers in order to select the best one, being these offers conformant to the demand. Nevertheless, we can not compute the maximum offers' utility assessments when comparing them. As an example, let the following offers $\omega_1 = \{60 \leq MTTF \leq 120\}$ and $\omega_2 = \{90 \leq MTTF \leq 110\}$. Intuitively, the first is better, because if $MTTF = 120$ then $\mathcal{U}(\omega_1) = 1$. However, the offer is guaranteeing the complete range, not only a concrete value, so we can not make such assumption because it is equally possible that $MTTF = 60$, and there is no control to get a concrete value. Therefore, we compare the minimum utility assessments of offers. In this way, the latter offer is the better, because if $MTTF = 90$ then $\mathcal{U}(\omega_2) = 0.5$, whereas the worst assessment of the first offer is 0.25, at most. Formally, the best offer (ω_S) can be defined as:

$$\omega_S = \omega \in \Omega_\delta \cdot \ \forall \omega_i \in \Omega_\delta - \{\omega\} \ \ \mathcal{U}^\delta(\omega) \geq \mathcal{U}^\delta(\omega_i)$$

where ω and ω_i stand for offers in the set Ω_δ of conformant offers to the demand δ, and the $\mathcal{U}^\delta(\omega)$ utility function of an offer ω according to assessment criteria in demand δ is defined as:

$$\mathcal{U}^\delta(\omega) = \underset{st\ c_\omega}{min} \sum_{\pi \in c_\omega} w_\pi^\delta U^\delta(\pi)$$

where π represents a quality-of-service parameter which is involved in the offer's CSP c_ω, and $U^\delta(\pi)$ its utility function, and w_π^δ its assigned weight, according to assessment criteria in demand δ. On the other hand, weights are needed to express that a quality-of-service parameter is preferred to another.

3.4 An Example of Contraint-Based Quality-of-Service Specification

Figure 5 shows several catalogues, demands, and offers written in QRL [23,26], the language which we have defined for specifying quality requirements. Figure 4 shows the graphical representation of utility functions appearing in Figure 5. These demands and offers will be used in the examples along these paragraphs.

In this case, the involved quality-of-service parameters are the `Mean Time To Failure` (MTTF), the `Mean Time To Repair` (MTTR), and the `Media Support` (MEDIA). Note the included demand and offers are all consistent, because their corresponding CSP are satisfiable, as well as offers are conformant to the demand, because the corresponding CSP for checking the conformance are not satisfiable, according to definitions in Section 3.2.

Since both offers are conformant to the demand, we will have to compute the utility functions to compare them. According to definitions in Section 3.3, both offers own $U(MTTF = 110) = 0.83$ and $U(MTTR = 10) = 0.8$, `velazquez` owns $U(MEDIA) = 1$, and `zipi` owns $U(MEDIA) = 0.5$. Therefore, utility assessment of `velazquez` is $0.9 * 0.83 + 0.05 * 0.04 + 0.05 * 1 = 0.84$, and utility assessment of `zipi` is $0.9 * 0.83 + 0.05 * 0.04 + 0.05 * 0.5 = 0.815$, so the best offer is `velazquez`.

```
// A catalogue of Reliability-related QoS parameters
catalogue Reliability {
  MTTF {
    description: "Mean Time to Failure";
    domain: real [0,+inf) minute;
  };
  MTTR {
    description: "Mean Time To Repair";
    domain: real [0,+inf) minute;
  };
}

// A catalogue of Multimedia-related QoS parameters
catalogue Multimedia {
  MEDIA {
    description: "Media Support";
    domain: set { modem, ISDN, ADSL };
  }
}
```

a) Catalogues of quality-of-service parameters.

```
// Web service demand for IVideoServer
using Reliability, Multimedia;
demands for IVideoServer {
  D1: MTTF / (MTTF + MTTR) >= 0.9;
  D2: MEDIA includes {modem,ISDN};
}
assessment {
  MTTF {90, { (0,0), (90,0.5), (120,1) } };
  MTTR {05, { (0,1), (20,0.6), (30,0) } };
  MODEM {05,
    case MEDIA = { } : 0.01;
    case MEDIA = {modem} : 0.1;
    case MEDIA = {ISDN} : 0.3;
    case MEDIA = {ISDN,modem} : 0.5;
    case MEDIA = {ADSL} : 0.9;
    case MEDIA = {modem, ADSL} : 1;
    case MEDIA = {ISDN, ADSL} : 1;
    case MEDIA = {modem, ISDN, ADSL} : 1;
  }
}
```

b) A demand.

```
// Web service offer supplied by Velazquez
using Reliability, Multimedia;
offer for IVideoServer {
  O1: MTTF >= 110 and MTTF <= 120;
  O2: MTTR > 5 and MTTR <= 10;
  O3: MEDIA = {ADSL,ISDN,modem};
}
```

```
// Web service offer supplied by Zipi
using Reliability, Multimedia;
offer for IVideoServer {
  O1: MTTF >= 110 and MTTF <= 120;
  O2: MTTR > 5 and MTTR <= 10;
  O3: MEDIA = {ISDN,modem};
}
```

c) Several offers.

Fig. 5. Demands and offers written in QRL.

4 Implementation and Experimental Results

4.1 Overview of the Prototype's Architecture

We are developing a prototype of a run-time framework for WSP [19,20,24,27], whose preliminary version is available at `http://www.lsi.us.es/~octavio`. In this paper, we give a brief review, together with some experimental results we have recently obtained. A components view of the run-time framework is shown in Figure 6.

Selecting a multi-level architecture along with the deployment of the components as web applications or web services have been critical design decisions. Components are split up among the upper user-interface level, the intermediate service and utility levels, and the bottom repository level. These components can be reusable and interchangeable. Service level includes those components which implement the `IImportService` interface (functions related to submission of demands and searching for best conformant offer), and the `IExportService` interface (functions related to submission of offers).

These components have need of invoking checkings for consistency, conformance, and optimum search. These functions are implemented by the Quality Trader Web Service [19] at the utility level. Each function has a similar operation:

1. It takes the involved demands and offers written in XML as parameters.
2. It invokes the appropriate XSLT transformations in order to generate automatically the corresponding CSP.

3. It invokes a CSP solver which processes the CSP in order to get the result, which is finally returned.

The CSP solver which is invoked is ILOG's OPL Studio [13], which is an integrated development environment for mathematical programming and combinatorial optimisation applications. The OPL language (OPtimisation Language) is used to define CSP and CSOP models.

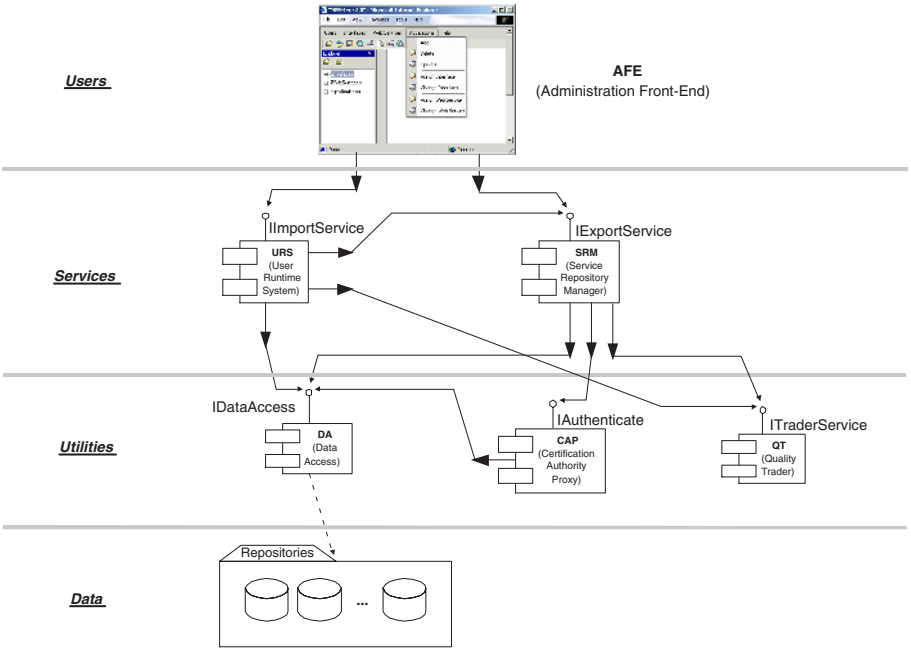

Fig. 6. Architecture of the run-time framework.

4.2 Experimental Results

Recently, we have carried out several tests, in order to get measures about performance of quality trader. We have implemented the first prototype in a Microsoft .NET environment, using the Visual Studio C#-based utilities and compilers. The main characteristics of the server computer are an AMD Athlom XP 1.8Gb processor with 560 Mb RAM. These tests have been focussed on latency of consistency, which is possibly the simplest of operations which are involved in WSP. In this paper, latency means the time from invokation of operation to return of a result. Figure 7 summarises the time our implementation took to check the consistency of a demand (of course, it could have been an offer).

Experimental data have been specified in this way: the N^{th} execution involves a QoS specification containing N constraints. Each QoS specification is constructed according

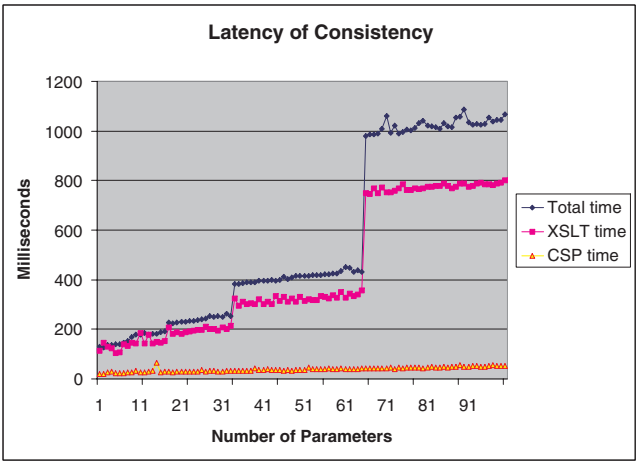

Fig. 7. Average time to check the consistency.

to this criteria: the first constraint involves a single integer-typed parameter, the second constraint involves a real-typed parameter, the third a set-typed parameter, the fourth an enumerated-type parameter, the fifth a boolean parameter, and so on, up to N constraints. Consistency of every QoS specification has been checked up to 50 times, so that 7 shows the average latency of invokations.

Figure 7 shows that XSLT processing is roughly implying up to 80 per cent. In this way, a first conclusion is that another alternative should be studied in case of a final version of the platform, such as the use of compilers or similar. XML and XSLT are good solutions for a prototype version, but it is not an efficient solution at all. On the other hand, XML and XSLT are (nearly) the universal standard of communications on the Internet, and it owns a very high versatility because it makes easier any adaptation, as well as the treatment of corresponding XML schemas as truly ontologies on QoS specification. We have used the Microsoft DOM library, so that another alternative is the use of components with improved XSLT-related functions, such as SAX, or similar.

5 Conclusions and Future Work

In this paper, we have presented our run-time framework for automating WSP. The solution is based on usage of mathematical constraints in order to specify quality-of-service in demands and offers, so we have achieved a lot of interesting properties. First, it owns a great deal of expressiveness, allowing non-linear or more complex expressions involving multiple parameters to be specified. As the same expressiveness is allowed to specify quality-of-service in demands and offers, our approach is symmetric. As well, our approach includes the possibility to express the assessment criteria, which is very important to select the best choice according to a demand.

Currently, we have developed a prototype of the run-time framework. It includes a quality trader web service as the core component, which offers services such as the

checking for consistency and conformance, and the search/selection of the best choice. Preliminar experimental results have been presented, standing out the marked influence of XSLT processing in perfomance of the trader service. However, new experiments have to be carried out to get a more complete vision of the framework perfomance.

Regarding with the future work, we want to point out that our approach can be extended in several ways in order to achieve new characteristics: the inclusion of temporality in constraints, the inclusion of negotiation clauses to improve the flexibility of the model whenever no solution can be found at first, and the inclusion of importance and soft clauses in order to enlarge the solution space of the search. In fact, definitions of temporality and negotiation are currently in study [25], so we are beginning the first phases of the improvement of our prototype to include them.

Finally, the integration of our model on the current technology is also a pending work. We are aware of the uselessness of our approach if we do not have a working prototype integrated with any of them, such as UDDI or similar. In this way, our quality trader is a component leveled at the top of a pyramid, wherein the lowerer levels are devoted to functional-aspects of WSP [1]. This stage of development is currently starting, but we hope to have a completely functional prototype in the very near future.

References

1. A. Beugnard, J-M. Jézéquiel, N. Plouzeau, and D. Watkins. Making components contract aware. *IEEE Computer*, pages 38–45, July 1999.
2. Joint US/EU Agent Markup Language Committee. DARPA Agent Markup Language. Technical report, US's DARPA Defense Advance Research Projects Agency and EU's IST Information Society Technologies, 2000. `http://www.daml.org`.
3. J.J. Dujmovic. A Method for Evaluation and Selection of Complex Hardware and Software Systems. In *Proceedings of the 22nd International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems*, volume 1, pages 368–378, 1996.
4. B. Farbey and A. Finkelstein. Software acquisition: a business strategy analysis. In *Proc. of the Requirements Engineering (RE'01)*. IEEE Computer Society Press, 2001.
5. A. Finkelstein and G. Spanoudakis. Software package requirements and procurement. In *Proc. of the 8^{th} Int'l IEEE Workshop on Software Specification and Design (IWSSD'96)*. IEEE Press, 1996.
6. X. Franch and P. Botella. Putting non-functional requirements into software architecture. In *Proc. of the IX^{th} Intl. Workshop on Software Specification and Design*, Ise-Shima (Isobe), Japan, April 1998.
7. E.C. Freuder and M. Wallace. Science and substance: A challenge to software engineers. *Constraints IEEE Intelligent Systems*, 2000.
8. S. Frolund and J. Koistinen. QML: A language for quality of service specification. Technical Report HPL-98-10, Hewlett–Packard, 1998.
9. H. Gallaire. Logic programming: Further developments. In *Proc. of the IEEE Symposium on Logic Programming*, pages 88–96, Boston, 1985. IEEE-CS Press.
10. J. González-Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. Technical Report HPL-2001-265, Hewlett-Packard, 2001.
11. P. Hentenryck and V. Saraswat. Strategic directions in constraint programming. *ACM Computing Surveys*, 28(4), December 1996.
12. Y. Hoffner, S. Field, P. Grefen, and H. Ludwig. Contract-driven creation and operation of virtual enterprises. *Computer Networks*, (37):111–136, 2001.

13. ILOG. OPL Studio. `http://www.ilog.fr`.
14. J. Jaffar and J.L. Lassez. Constraint logic programming. In *Proc. of the ACM Symposium on Principles of Programming Languages*, pages 111–119, Boston, 1987.
15. A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. Technical Report RC22456 (W0205-171), IBM International Business Machines Corporation, 2002.
16. J. Koistinen and A. Seetharaman. Worth–based multi-category quality–of–service negotiation in distributed object infrastructures. In *Proceedings of the Second International Enterprise Distributed Object Computing Workshop (EDOC'98)*, La Jolla, USA, 1998.
17. H. Ludwig, A. Keller, A. Dan, and R.P. King. A service level agreement language for dynamic electronic services. Technical Report RC22316 (W0201-112), IBM International Business Machines Corporation, 2002.
18. K. Marriot and P.J. Stuckey. *Programming with Constraints: An Introduction*. The MIT Press, 1998.
19. O. Martín-Díaz, A. Ruiz-Cortés, D. Benavides, A. Durán, and M. Toro. A quality-aware approach to web services procurement. In *Fourth International VLDB Workshop Technologies for E-Services, Springer LNCS 2819*, pages 42–53, Berlin, Germany, 2003.
20. O. Martín-Díaz, A. Ruiz-Cortés, R. Corchuelo, and A. Durán. A Management and Execution Environment for Multi-Organisational Web-based Systems. In *ZOCO: Métodos y Herramientas para el Comercio Electrónico*, pages 79–88, San Lorenzo del Escorial, Spain, 2002.
21. L. Olsina, D. Godoy, G. Lafuente, and G. Rossi. Specifying Quality Characteristics and Attributes for Websites. In *Proceedings of the Web Engineering Workshop, in conjunction with 21st International Conference on Software Engineering (ICSE)*, pages 84–93, May 1999.
22. OMG. Trading Object Service Specification. Technical report, Object Management Group, 2000. Version 1.0.
23. A. Ruiz-Cortés. *A Semi-qualitative Approach to Automated Treatment of Quality Requirements (in Spanish)*. PhD thesis, E.T.S. de Ingeniería Informática. Dpto. de Lenguajes y Sistemas Informáticos. Universidad de Sevilla, 2002.
24. A. Ruiz-Cortés, R. Corchuelo, and A. Durán. An automated approach to quality-aware web applications. In *Enterprise Information Systems IV*, pages 237–242. Kluwer Academic Publishers, 2003.
25. A. Ruiz-Cortés, R. Corchuelo, A. Durán, and M. Toro. Enhancing Win–Win requirements negotiation model. In *Applied Requirements Engineering*. Catedral, 2002.
26. A. Ruiz-Cortés, A. Durán, R. Corchuelo, B. Bernárdez, and M. Toro. Automated Checking of Quality Requirements in Multi-Organisational Systems (in Spanish). In *4th Workshop on Requirements Engineering (WER'01)*, pages 195–201, Buenos Aires, Argentina, 2001.
27. A. Ruiz-Cortés, R. Corchuelo, A. Duran, and M. Toro. Automated support for quality requirements in web-services-based systems. In *Proc. of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'2001)*, Bologna, Italy, 2001. IEEE Press.
28. A. ShaikhAli, R. Al-Ali O. Rana, and D. Walker. UDDIe: An extended registry for web services. In *Proc. of the IEEE Int'l Workshop on Service Oriented Computing: Models, Architectures and Applications at SAINT Conference*. IEEE Press, January 2003.
29. E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London, 1995.
30. D.L. Waltz. *Understanding line drawings of scenes withshadows*. Psychology of Computer Vision, New York, 1975.
31. Gerhard Weikum. The Web in 2010: Challenges and opportunities for database research. *Lecture Notes in Computer Science n° 2000*, 2001.