EFPL Technical Report IC/2005/029

QoS-based Service Selection and Ranking with Trust and Reputation Management*

Le-Hung Vu, Manfred Hauswirth and Karl Aberer

School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{lehung.vu, manfred.hauswirth, karl.aberer}@epfl.ch

Abstract. QoS-based service selection mechanisms will play an essential role in service-oriented architectures, as e-Business applications want to use services that most accurately meet their requirements. Standard approaches in this field typically are based on the prediction of services' performance from the quality advertised by providers as well as from feedbacks of users on the actual levels of QoS delivered to them. The key issue in this setting is to detect and deal with false ratings by dishonest providers and users, which has only received limited attention so far. In this paper, we present a new QoS-based semantic web service selection and ranking approach with the application of a trust and reputation management method to address this problem. We will give a formal description of our approach and validate it with experiments which demonstrate that our solution yields high-quality results under various realistic cheating behaviors.

1 Introduction

One key issue in the Semantic Web Service area is to discover the most relevant services meeting the functional requirements of users. Equally important, e-Business applications also would like to discover services which best meet their requirements in terms of QoS, i.e., performance, throughput, reliability, availability, trust, etc. Thus QoS-based selection and ranking mechanisms will play an essential role in service-oriented architectures, especially when the semantic matchmaking process returns lots of services with comparable functionalities. To support the user in selecting the best services in terms of QoS, service discovery basically has to choose only those services fulfilling all user's quality requirements and rank the services by their expected guarantees in meeting the QoS criteria. As a prerequisite it is necessary to provide expressive semantic means for describing web services including functional and non-functional properties which the system can use in the discovery process. Additionally, it should also allow users to provide feedback on the perceived QoS of a service. In conjunction with an approach to assess the trustworthiness of both service providers and users, this makes it possible to estimate the future quality of a service.

* The work presented in this paper was (partly) carried out in the framework of the EPFL Center for Global Computing and was supported by the Swiss National Funding Agency OFES as part of the European project DIP (Data, Information, and Process Integration with Semantic Web Services) No 507483. Le-Hung Vu is supported by a scholarship of the Swiss federal government for foreign students.

In this paper we present a QoS-based web service selection and ranking approach which uses trust and reputation evaluation techniques to predict the future quality of a service. Our work is based on requirements from a real-world case study on virtual Internet service providers (VISP) from an industrial partner in one of our projects [1]. In a nutshell, the idea behind the VISP business model is that Internet Service Providers (ISPs) describe their services as semantic web services, including QoS such as availability, acceptable response time, throughput, etc., and a company interested in providing Internet access, i.e., becoming a VISP, can look for its desired combination of services taking into account its QoS and budgeting requirements, and combine them into a new (virtual) product which can then be sold on the market. This business model already exists, but is supported completely manually. Since many ISPs can provide the basic services at different levels and with various pricing models, dishonest providers could claim arbitrary QoS properties to attract interested parties. The standard way to prevent this is to allow users to evaluate a service and provide feedbacks. However, the feedback mechanism has to ensure that false ratings, for example, badmouthing about a competitor's service or pushing one's own rating level by fake reports or collusion with other malicious parties, can be detected and dealt with. Consequently, a good service discovery engine would have to take into account not only the functional suitability of services but also their prospective quality offered to end-users by assessing the trustworthiness of both providers and consumer reports. According to several empirical studies [2, 3], the issue of evaluating the credibility of user reports is one of the essential problems to be solved in the e-Business application area. In our case, solving this problem will increase the accuracy of the QoS-enabled discovery engine and help users to select the most relevant services among numerous service advertisements in the registries without involving in the negotiation and establishing business contracts with bad or unpromising providers. Also, this will guarantee the fairness among competing service providers themselves and thus indirectly stimulate the real improvement in their quality offered to end-users.

We develop the QoS-based service selection algorithm under two basic assumptions which are very reasonable and realistic in e-Business settings: First, we assume *probabilistic behavior of services and users*. This implies that the differences between the real quality conformance which users obtained and the QoS values they report follow certain probability distributions. These differences vary depending on whether users are honest or cheating as well as on the level of changes in their behaviors. Secondly, we presume that *there exist a few trusted third parties*. These well-known trusted agents always produce credible QoS reports and are used as trustworthy information sources to evaluate the behaviors of the other users. In reality, companies managing the service searching engines can deploy special applications themselves to obtain their own experience on QoS of some specific web services. Alternatively, they can also hire third party companies to do these QoS monitoring tasks for them. In contrast to other models [4–8] we do not deploy these agents to collect performance data of all available services in the registry. Instead, we only use a small number of them to monitor QoS of some selected services because such special agents are usually costly to setup and maintain.

The QoS-based service selection and ranking algorithm we describe in this paper is a part of our overall distributed service discovery approach [9] and relies on the following input data: A list of web services with similar functionalities obtained from the matchmaking of our framework, i.e., the services fulfill all user's functional requirements. Second, we need the predicted QoS values of every quality attribute of each web service to enable QoS processing. During the service discovery phase, after the functional matchmaking at a specific registry, we select and rank services based on their predicted QoS values, taking into consideration the explicit quality requirements of users in the queries. The output of the selection and ranking algorithm is the list of web services fulfilling all quality requirements of a user, ordered by their prospective levels of satisfaction of the given QoS criteria. In order to perform this selection and ranking accurately, we collect user reports on QoS of all services over time and compute their predicted quality. This prediction, which is based on the quality promised by the service provider, also takes into consideration trust and reputation issues. Since this step is computationally expensive but not time-intensive, it is performed off-line on a periodical basis so as not to affect system performance.

The major contribution of our work is a new QoS-based service selection and ranking approach which is expected to be accurate, efficient and reliable. We have taken into account the issue of trust and reputation management adequately when predicting service performance and ranking web services based on their past QoS data. Experimental results have shown that the newly proposed service selection and ranking algorithm yields very good results under various cheating behaviors of users, which is mainly due to the fact that our use of trusted third parties observing a relatively small fraction of services can greatly improve the detection of dishonest behavior even in extremely hostile environments. This is particularly important as without cheating detection, service providers will be likely to generate lots of false reports in order to obtain higher ranks in the searching results, thereby having higher probability to be selected by clients and gain more profits. Additionally, our algorithm is semantic-enabled by offering support for semantic similarity among QoS concepts advertised by providers and the ones required by users. This allows the QoS-based service selection process to work more flexibly and produce more accurate results. We also adapt the idea of Srinivasan et al [10] to pre-compute all matching information between QoS capabilities of published services and possible QoS requirements of users to avoid time-consuming reasoning and to minimize the searching costs.

The rest of this paper is organized as follows. In section 2, we have a review on related work. Section 3 presents our trust and reputation management model in a Web Service Discovery scenario. Our QoS-based service selection and ranking approach is described in detail in section 4. We discuss various experimental results in section 5 and conclude the paper in section 6.

2 Related Work

Although the traditional UDDI standard [11] does not refer to QoS for web services, many proposals have been devised to extend the original model and describe web services' quality capabilities, e.g., QML, WSLA and WSOL [12,13]. The issue of trust and reputation management in Internet-based applications has also been a well-studied problem [14,15]. However, current QoS-enabled service discovery solutions have not sufficiently considered the problem of evaluating the credibility of reporting users while predicting performance of services from their past quality information given by user feedback. The existing discovery approaches either ignore this issue totally or employ simple methods which are not robust against various cheating behaviors.

The UX architecture [16] suggests using dedicated servers to collect feedback of consumers and then predict the future performance of published services. [17] proposes an extended implementation of the UDDI standard to store QoS data submitted by either service providers or consumers and suggests a special query language (SWSQL) to manipulate, publish, rate and select these data from repository. According to Kalepu et al [18], the reputation of a service should be computed as a function of three factors: ratings made by users, service quality compliance, and its verity, i.e., the changes of service quality conformance over time. However, these solutions do not take into account the trustworthiness of QoS reports produced by users, which is important to assure the accuracy of the QoS-based selection and ranking results. [19] rates services computationally in terms of their quality performance with QoS information provided by monitoring services and users. The authors also employ a simple approach of reputation management by identifying every requester to avoid report flooding. In [20], services are allowed to vote for quality and trustworthiness of each other and the service discovery engine utilizes the concept of distinct sum count in sketch theory to compute the QoS reputation for every service. However, these reputation management techniques are still simple and not robust against various cheating behaviors, e.g., collusion among providers and reporters with varying actions over time. Consequently, the quality of ranking results of those discovery systems will not be assured if there are lots of colluding, dishonest users trying to boost the quality of their own services and badmouth about the other ones. [21] suggests augmenting service clients with QoS monitoring, analysis, and selection capabilities, which is a bit unrealistic as each service consumer would have to take the heavy processing role of a discovery and reputation system. Other solutions [4–8] use mainly third-party service brokers or specialized monitoring agents to collect performance of all available services in registries, which would be expensive in reality. Though [8] also raises the issue of accountability of Web Service Agent Proxies in their system, the evaluation of trust and reputation for these agents is still an open problem.

Our QoS provisioning model is grounded on previous work of [4, 5, 16, 18, 19]. The trust management approach in our work is most similar to that of [22, 23] but we employ the idea of distrust propagation more accurately by observing that trust information from a user report can also be used to reveal dishonesty of other reporters and by allowing this distrust to be propagated to similar ones. Other ideas of the trust and reputation management method are based on [24, 25], whereas our probabilistic-based user behavior model is derived from [26]. The idea of clustering user QoS reports originates from [26, 27].

3 A Trust and Reputation Management Model for QoS-based Service Discovery

The interaction among agents in our system is represented in Fig. 1 where S_0,..,S_n are web services, U_0,...,U_m are service users, RP_0,...,RP_k are service registries in a P2P-based repository network, based on our P-Grid P2P system [28]. T_0,...,T_r are trusted QoS monitoring agents from which we will collect trustworthy QoS data to use in our algorithm. After U_j's perception of a normalized QoS conformance value x_{ij} (a real-valued number as defined in Definition 1 in the following section) from a service S_i, it will report the value y_{ij} to the registry RP_k, which manages the description information of S_i. This registry peer will collect users' quality feedbacks on all of its managed web services to predict their future performance and support the QoS-enabled service discovery based on these data. Note that a user generally reports a vector of values representing its perception of various quality parameters from a service. Another noticeable point is that x_{ij}s and y_{ij}s are QoS conformance values, which already take into account the quality values advertised by providers (see Definition 1 in the next section). Since the prediction of QoS in our work mostly depends on reports of users to evaluate service performance, the cheating behaviors of providers are not explicitly mentioned henceforth. In this paper, we only consider the selection and ranking of services with reputation management in one registry peer. The study of interaction among different registries is subject to future work and beyond the scope of this paper.

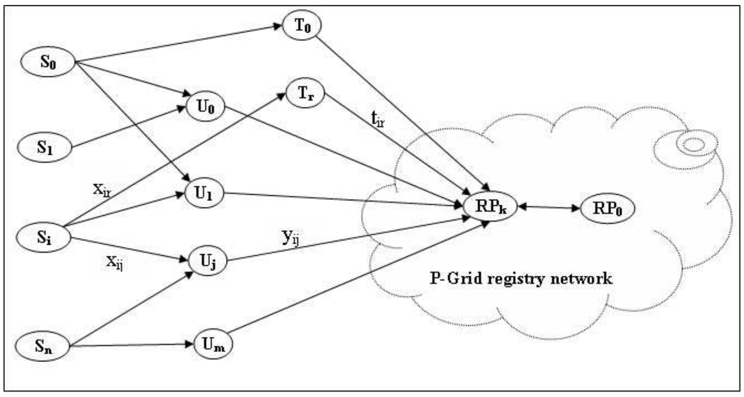

Fig. 1. Interaction among agents in a QoS-based service discovery scenario

Given the above interaction model, we can make a number of observations. An honest user will report $y_{ij} = x_{ij}$ in most cases (in reality, they may not report if not willing to do so). On the other hand, a dishonest reporter who wants to boost the performance of S_i will submit a value $y_{ij} > x_{ij}$. Similarly, cheating reports generated by S_i's competitors will report $y_{ij} < x_{ij}$ to lower its QoS reputation. In addition, (colluding) liars may sometimes provide honest reports while behaving dishonestly in other cases [15]. Accordingly, we presume that the differences between y_{ij}s and x_{ij}s follow certain distribution types which *expected values* depend on the behavior of the user, i.e., are equal to 0.0 for honest users and different from 0.0 in the case of liars. We use this assumption since in general dishonest users are likely to change their actions over time in order to hide their cheating behaviors with *occasionally* credible reports. Moreover, as the quality parameter values of a service really depend on many environmental factors, even trusted agents and honest users may obtain and report those values a little different to each other when talking about the same service. However, the expected value of trustworthy reports on QoS of S_i, e.g., the average of corresponding credible y_{ij} values, will reflect its real quality capability. In our opinion, the expected values of these distributions, e.g., means, normally reveal the behaviors of users, whereas other parameters, e.g., standard deviations, will represent the uncertainty in actions of users, either accidentally or by purpose. Our goal is not only to evaluate whether a user is honest but also to compute the expected conformance values from the reports of the most honest users, e.g., the average of values y_{ij}s by the most credible reporters, from which we will predict the future quality of S_is.

4 QoS-based Service Selection and Ranking

Our QoS-enabled distributed service discovery framework is presented in detail in [9]. Quality properties of web services are described by concepts from a QoS ontology and then embedded into service description files using techniques suggested by WS-QoS [4] and Ran [5]. In our work, the value of a quality parameter of a web service is supposed to be *normalized* to a non-negative real-valued number regarding service-specific and call-specific context information where higher normalized values represent higher levels of service performance. We are aware that the issue of normalizing the values of various quality attributes is complicated, but this is out of the scope of our current study. However, with most frequently used QoS concepts, e.g., reliability, execution-time, response-time, availability, etc., the answer is well-defined and straight-forward. For instance, a web service with a normalized QoS parameter value for reliability of 0.99 will be considered as more reliable to another one with a normalized reliability value of 0.90. In this case the normalized reliability is measured as its degree of being capable of maintaining the service and service quality over a time period T. Another example would be a web service with a normalized execution-time value 0.50 which will be considered as faster than another service with a normalized execution-time value 0.20. In this case the normalized value is computed as the reciprocity of the execution-time w.r.t. some client-dependent conditions. The ontology language we use to describe service semantics and to define the QoS ontology is WSMO [29], but other models, e.g., OWL-S [30], would also be applicable. For experimental evaluations, we have developed a simple QoS ontology for the VISP use case including the most relevant quality parameters for many applications, i.e., availability, reliability, execution time, etc. We currently assume that users and providers share a common ontology to describe various QoS concepts. However, this could be relaxed with the help of many existing ontology mapping frameworks [31,32].

4.1 Predicting Service Performance

In order to predict the quality of a web service S_i, we collect all QoS feedbacks on its performance over a time period W and use a *real-valued time-based series forecasting technique* to predict its future quality conformance from past performance data. To understand the concepts in our algorithms we start with two fundamental definitions.

Definition 1. The quality conformance value c_{ij}^k of a service S_i in providing a quality attribute q_{ij} at time k is defined as $c_{ij}^k = \frac{d_{ij}^k - p_{ij}^k}{p_{ij}^k}$ where the d_{ij}^k is the normalized value of q_{ij} that S_i really delivered to a specific user at time k and p_{ij}^k is the corresponding normalized QoS value promised by S_i's provider at that time.

Accordingly, c_{ij}^k is a real-valued number, which may be positive if the service has better performance than as promised and negative if its performance is lower than as advertised by its provider.

Definition 2. A user QoS report R, either by a normal service consumer or by a trusted monitoring agent, is a vector $\{u, S_i, t, L\}$, where u is the user that produced this report, S_i is the corresponding web service, t is the timestamp of the report and L is a quality conformance vector of $\{q_{ij}, c_{ij}^t\}$ pair values, with q_{ij} being a QoS attribute offered by S_i and c_{ij}^t being q_{ij}'s quality conformance at time t to this user.

In order to filter out as much dishonest reports as possible and to take only the most credible ones in the QoS predicting process, we apply our trust and reputation management techniques comprising of two steps: a report preprocessing and a report clustering phase. The first phase evaluates the credibility of collected user reports by applying a trust-and-distrust propagation approach, which relies on some initial trusted reports produced by special monitoring agents. We consider two QoS reports as *comparable* if they are related to the same service during a specific time interval δ_t and as *incomparable* otherwise. Generally, we can set this δ_t as big as the length of the period during which the corresponding service provider does not change the promised quality values of this service. Two *comparable* QoS reports are considered to be *similar* if the squared Euclidean distance between their conformance vectors is less than a specific threshold. On the contrary, they are regarded as *dissimilar* if this distance is greater than another threshold.

The report preprocessing step is done according to Algorithm 1. n_{ch} and n_h are threshold values to estimate a user as cheating or honest regarding to the similarity of its reports to other cheating/honest ones during the distrust and trust propagation (line 9 and line 17). N and T represent for how long and how frequent a user stay in and submit its QoS reports to the system. The values of n_h, n_{ch}, N and T are design parameters to be chosen depending on properties of the collected reports after running the algorithm several times. Generally, higher values of these parameters stand for higher level of caution when estimating behaviors of users regarding current evidences against them.

After finishing the preprocessing phase, we can identify a certain number of cheaters and honest users. However, this trust-distrust propagation phase may not be able to evaluate the credibility of all reports in case the user communities of certain services are isolated from other communities as well as if we set the values of n_h and n_{ch} too high in Algorithm 1.

Therefore, in the next step we have to estimate the trustworthiness of the remaining reports of which credibility has not been evaluated. To achieve this, we reason that colluding cheating users will cooperate with each other in order to influence the system with their dishonest feedbacks. As a result, users within

Algorithm 1 QosReportsPreprocessing()

```
1: All trusted agents are marked as *honest* users;
2: All reports of trusted agents are marked *credible*;
3: **while** there are still new cheaters discovered **do**
4:   All unmarked reports of each cheating user are marked *incredible*;
5:   **for** each unmarked report **do**
6:     If this report is *dissimilar* from a credible report then mark it *incredible*;
7:     If this report is *similar* with an incredible report then mark it *incredible*;
8:   **end for**
9:   Users with at least $n_{ch}$ reports similar with incredible ones are marked *cheating*;
10:  Users with at least $N$ reports in at least $T$ different time are marked *stable*;
11: **end while**
12: **while** there are still new honest users discovered **do**
13:  All unmarked reports of each honest user are marked as *credible*;
14:  **for** each unmarked report and report marked cheating **do**
15:    If this report is *similar* with a credible report then mark it *credible*;
16:  **end for**
17:  Users with at least $n_h$ reports similar with credible ones are marked *honest*;
18: **end while**
```

each group will produce similar values and naturally form different clusters of reports. Thus it is possible to apply data-mining techniques in this situation to discover various existing clusters of reports related to those user groups. In our work, we apply the *convex k-mean clustering algorithm* on each set of QoS reports related to a service during the time interval δ_t with the following metrics: The distance between two *comparable* reports is defined as the Euclidean squared distance between two corresponding quality conformance vectors and the distance between two *incomparable* ones is assigned a large enough value so that these reports will belong to different clusters.

After the trust and reputation evaluation in the two above phases, for each service S_i, we will have a list G_i of groups of reports on QoS conformance of S_i over time. Generally, G_i includes the groups containing those reports that were previously marked as honest/cheating during the trust-distrust propagation phase, as well as other clusters of reports obtained after the clustering step. We will assign the credibility w_i^g of a report group $g_i \in G_i$ as follows. Firstly, we filter out all dishonest ratings by assign $w_i^g = 0.0$ for all group of reports marked as cheating during the trust-distrust propagation. If there exists the group g_i^0 of reports previously marked honest during that step, we assign $w_i^{go} = 1.0$ whereas letting $w_i^g = 0.0$ for the remaining groups. Otherwise, we try to compute w_i^g so that this value would be proportional to the probability that the group g_i is trustworthy among all of the others. Our heuristic is to assign higher weight to clusters which are populated more densely, having bigger size and with more stable users. This assumption is reasonable, as the reports of independent cheaters are likely to be scattered, and in the case liars cooperate with each other to cheat the system, the size of their corresponding clusters will not exceed those consisting only of honest reports as it would be too costly to dominate the system with numerous and clustered dishonest ratings. Even if dishonest providers try to produce lots of more condense reports so that they could get high influences to the final ranking results at any cost, these values will be separated from honestly reported values and therefore are likely to be discovered during the distrust-propagation phase (line 3 to line 11 in Algorithm 1), provided we have enough trustworthy reports to use.

Specifically, w_i^g could be estimated based on the following information: the number of users in the cluster g_i ($size_i^g$), the number of all users producing reports in all clusters of G_i ($allusers_i$), the number of stable users in this cluster ($stable_i^g$), the total number of stable users in all clusters of G_i ($allstableusers_i$), as well as the average distance d_i^g from the member reports of cluster g_i to its centroid values. Based on our model in section 3, the credibility of one report would depend on the distance between its conformance vector and that of an honest report. Therefore, the similarity among credibility of different reports in one cluster g_i would be proportional to its d_i^g value. Furthermore, a report in g_i would be honest in two cases: (1) it is submitted by a *stable and honest* user; (2) it is produced by an *unstable and honest* user. Let P_A be the probability that this report is of a stable user and P_B for an unstable user, and let P_{stable} and $P_{unstable}$ be the probability that stable and unstable users report credibly, then we have:

$$w_i^g = \frac{C}{d_i^g}.(P_A.P_{stable} + P_B.P_{unstable}) \tag{1}$$

where $P_A = \frac{stable_i^g}{size_i^g}$ and $P_B = 1 - P_A$. P_{stable} and $P_{unstable}$ could be estimated by comparing reports of trusted agents with those of sample stable/unstable users to derive an appropriate value at the whole network level. The values of C represent our belief in the relation between the density of a report cluster and the credibility of its members, which is considered as parameters of the reputation system and to be set by experience.

The future quality conformance $\hat{C}_{ij}$ of a service S_i in providing a QoS attribute q_{ij} is predicted using a linear regression method, thus we have: $\hat{C}_{ij} = LeastSquaredEstimation(\overline{C}_{ij}^t)$ where $\overline{C}_{ij}^t$ is the evaluated QoS conformance value of the quality parameter q_{ij} at time $t \in \{0, \delta_t, \ldots, (W-1).\delta_t\}$. We define $\overline{C}_{ij}^t$ as the average of conformance values reported by various user groups in the system at that specific time point, using the evaluated credibility w_i^gs as weights in the computation.

$$\overline{C}_{ij}^t = \frac{\sum_{g_i \in G_i} w_i^g C_{ij}^t}{\sum_{g_i \in G_i} w_i^g} \tag{2}$$

In the above formula, C_{ij}^t is the mean of conformances of a report group g_i on S_i's quality attribute q_{ij} at time t, i.e., a centroid value of a cluster/group of reports produced after the trust and reputation evaluation phase. Regarding the probabilistic-based behavior of services as in section 3, we consider $\hat{C}_{ij}$ as an approximate estimate of the expected value of S_i's QoS conformance in providing quality attribute q_{ij} to users.

4.2 QoS-based Service Selection and Ranking

We define QoS requirements in a user query as a vector Q of triples $\{q_j, n_j, v_j\}$ where q_j represents for the required QoS attribute, n_j is the level of importance of this quality attribute to the user and v_j is the minimal delivered QoS value that this user requires. To rank services according to its prospective level of satisfying user's QoS requirements, we utilize the Simple Additive Weighting method, which produces ranking results very close to those of more sophisticated Decision Making techniques [6, 34]. Note that although it is possible to use QoS properties as ranking criteria for service queries without explicit QoS requirements, we have not yet employed this in our current study. Thus, the QoS rank of a service S_i in fulfilling all quality criteria is defined by the following weighted sum:

$$T_i = \frac{\sum_{q_j \in Q} n_j.P_{ij}}{\sum_{q_j \in Q} n_j} \tag{3}$$

where $P_{ij} = w_{ij}.\widehat{nd}_{ij}$ represents for the capability of S_i in providing the QoS concept q_{ij} for users at the query time. The value $\widehat{nd}_{ij} = \frac{\hat{d}_{ij} - v_j}{v_j}$ evaluates the difference between the QoS value $\hat{d}_{ij}$ of the quality attribute q_{ij} that S_i predictably delivers to its user and the corresponding value v_j of q_j required by the service query. w_{ij} is a weight proportional to the semantic similarity m_{ij} between q_{ij} and the QoS ontology concept q_j required by the user, i.e., the degree of match as in [33]. The idea behind is that we will give higher ranks for services which offer the most accurate QoS concepts at the higher levels compared to the ones required by users. In our program we simple use the following definition to evaluate w_{ij}:

$$w_{ij} = \begin{cases} 1.0 & \text{if } m_{ij}{=}\textit{exact}; \text{(i.e., } q_{ij} \text{ is equivalent to } q_j) \\ 0.5 & \text{if } m_{ij}{=}\textit{pluggin}; \text{(i.e., } q_{ij} \text{ is more general than } q_j) \\ 0.0 & \text{if } m_{ij} \in \{\textit{subsume, failed}\}; \text{(i.e., otherwise)} \end{cases} \tag{4}$$

From Definition 1, we have $\hat{d}_{ij} = (1 + p_{ij}).\hat{C}_{ij}$, with $\hat{C}_{ij}$ is the predicted QoS conformance value for quality attribute q_{ij} and p_{ij} is the corresponding QoS value promised by provider of S_i at current time. Thus:

$$P_{ij} = w_{ij}.\frac{(1 + p_{ij}).\hat{C}_{ij} - v_j}{v_j} \tag{5}$$

In order to accelerate the selection of only services fulfilling all required QoS parameters, we use the idea of Srinivasan et al [10] to avoid the time-consuming semantic reasoning step. Specifically, we use a QoS matching table to store the matching information for all frequently accessed QoS attributes. With each QoS attribute q_j in this QoS matching table, we have a list $Lqos_j$ of records $\{S_{ij}, w_{ij}, \hat{d}_{ij}\}$ where w_{ij}, $\hat{d}_{ij}$ are computed as above and S_{ij} identifies a service having certain support for q_j. Given the list L of services with similar functionalities, the discovery engine performs the QoS-based service selection and ranking process as in Algorithm 2.

5 Experimental Results and Discussions

To evaluate the selection and ranking algorithm, we have implemented it as a QoS support module in a registry peer of our distributed discovery framework [9]

Algorithm 2 QosSelectionRanking(ServiceList L, ServiceQuery Q)

```
1: Derive the list of QoS requirements in Q: $L_q = \{[q_1, n_1, v_1], ..., [q_s, n_s, v_s]\}$
2: for each quality concept $q_j \in L_q$ do
3:   for each service $S_{ij} \in L$ do
4:     $Score[S_{ij}] = 0.0$;
5:     Search the list $L_{qos}$ of $q_j$ for $S_{ij}$;
6:     if $S_{ij}$ is found then
7:       $Score[S_{ij}] = Score[S_{ij}] + \frac{n_j \cdot w_{ij}}{\sum n_j}(\frac{\hat{d}_{ij} - v_j}{v_j})$;
8:     else
9:       Remove $S_{ij}$ from $L$;
10:    end if
11:  end for
12: end for
13: Return the list $L$ sorted by $Score[S_{ij}]$ s;
```

and studied its effectiveness under various settings. The service selection and ranking is performed on three representative quality parameters, namely *availability*, *reliability* and *execution-time* taken form the VISP case study.

We observed the dependency between the quality of ranking results and other factors, such as the percentage of trusted users and reports, the rate of cheating users in the user society and the various behaviors of users. Specifically, we evaluated the effectiveness of the service discovery by studying the differences in the quality of results when running evaluations in four different settings: In the *ideal* case the discovery engine has complete knowledge, such that it knows all correct QoS conformance values of all published services in the system over a time window W and performs the selection and ranking of services from this ideal data set. In the *realistic* case, we ran our algorithm with the application of trust and management techniques to filter out incredible reports and to evaluate the credibility for the others. The *optimistic* case corresponds to the QoS-based discovery of services without regarding to trust and reputation issues, i.e., the system simply uses the average of all reported conformance values to predict services' performance and to perform the QoS ranking. The *naive* case corresponds to the selection of services based only on their QoS values promised by the providers, i.e., the system trusts all providers completely. Our goal was to show that the obtained results of the QoS-based service discovery process are more accurate and reliable in the realistic case with various cheating behaviors of users, they would be much worse in the optimistic case, and the worst with the naive method thus clearly showing the contribution of our approach.

The quality of selection results produced by service discovery can be measured by four parameters, namely *recall*, *precision*, *R-precision* and *Top-K precision*, of which the *R-precision* is the most important quality parameter as recognized by the Information Retrieval community. In our settings, these parameters generally represent for the fraction of services that are most relevant to a specific user among all returned services in terms of *real* QoS capabilities. Apparently, the results would be the best in the ideal case, i.e., its recall, precision, R-precision and Top-K precision parameters are all equal to 1.0. Therefore, we use the results of the ideal case as a reference to compare with the quality parameters in the other three situations. Due to space limitations we only show the R-precision values as the most representative experimental results in this section. We also measured the other parameters as well as computed the absolute QoS ranks of returned services using weighting Spearman's footnote method and had similar results.

We prepared our experiments with the probabilistic-based assumption on the behavior of service providers and consumers. In this paper we only present the experiments with Gaussian (normal) distributions. The extension to other probabilistic distributions is subject to future work. The society of service consumers is modeled as a collection of different types of users. As mentioned in section 3, due to environmental factors, honest users and trusted agents would reported values with the difference $D_h \sim Normal(0.0, {\sigma_h}^2)$ to the real QoS conformance capabilities of services. On the contrary, cheaters would report values with the difference $D_c \sim Normal(M_c, {\sigma_c}^2)$ to the real quality conformances that they had obtained. The values of the mean M_c and the standard deviation σ_c varied according to their cheating behaviors. In our experiments, the values of σ_c represent the variation in reported values of users among different quality attributes of different services, i.e., users with higher values of σ_c have higher levels of inconsistency in their behaviors and therefore are harder to be detected. We divided these liars into three sub-types: *badmouthing users* who *mostly* reported badly about services of their competitors, *advertising users* who *usually* exaggerated performance of their own services and *uncertain users* with indeterministic actions and who might act as advertising, badmouthing or even as honest users. We set the values of M_c for each type of cheaters in the most pessimistic situation, making our testing environment be very hostile. Specifically, cheaters have their corresponding M_cs set to high/low enough values such that badmouthing users would succeed in pushing services of their competitors out of the ranking results and advertising users would be able to raise the QoS ranks of their own services, provided that their reports had been taken into the predicting process of the QoS-based service discovery engine. We should be aware that this setting is realistic because in business, companies generally have knowledge of the base requirements of their users as wells as owning certain statistics of their competitors' capabilities. More complicatedly, uncertain users had their M_cs values belonging to N_c specific values each of which was a randomized real value, with N_c is the number of groups of cheaters with varied behaviors. These types of liars would be harder to be detected since their M_c values were uniformly distributed around 0.0. Though they did not contribute directly to the boosting and lowering the reputation of certain services, their reports were not so dissimilar from honest reports in most cases and therefore they would act as good recommenders for other badmouthing/advertising guys. To assure the scalability of the approach, we also tested it with an increasing number of services, users and QoS reports while keeping the percentage of user types and other parameters the same. The remaining experiments were run in a society of 1000 users which produced a total of 50000 QoS reports on 200 services during a time window of length $W = 5$ and $\delta_t = 1$. The results of each experiment were averaged over 10 runs.

As a first question, we wanted to study the effects of the trusted reports on the quality of results in the realistic case. Specifically, we wanted to observe the effects of the percentage of the services monitored by trusted agents $F_{special}$ to the results of our QoS-based service selection and ranking algorithm expressed by *R-Precision* values. We increased $F_{special}$ from 1.0% to 10.0%. The results of this experiment are shown in Fig. 2.

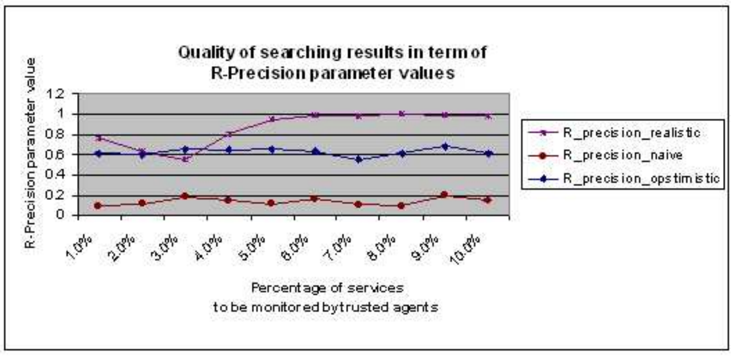

Fig. 2. $F_{special}$ vs. *R-Precision*

Correspondingly, Fig. 3 shows the percentage of cheating and honest reports correctly identified during the report preprocessing phase with our trust-distrust propagation method.

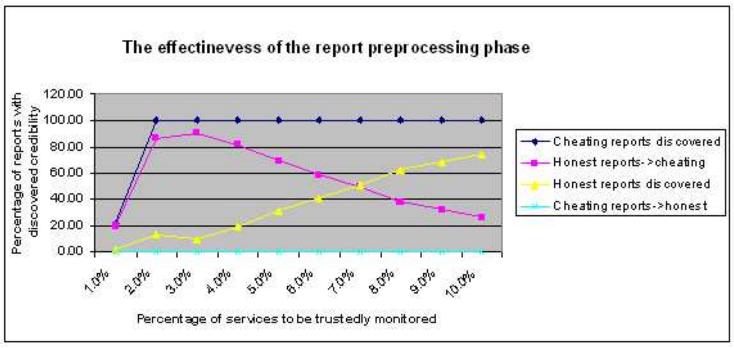

Fig. 3. $F_{special}$ vs. *Correctness of the report preprocessing phase.*

In this experiment, the percentage of trusted users/reports was increased from 0.1% to 1.0% whereas we assumed a very high number of cheaters (74.0% of the total users) consisting of badmouthing users, advertisers and five different groups of uncertain users. We also did various experiments to study the effects of σ_c's and σ_h's values and found that our algorithm performs very well with different settings of σ_c provided that the quality of the honest user population is good, i.e., σ_h is low. Given the fact that the QoS conformance values in our simulation are in the interval $[-1.0, 1.0]$, we kept the standard deviations of cheating reports very high ($\sigma_c = 0.5$) and those of trusted and honest users at an acceptably low level ($\sigma_h = 0.01$). With the increase of $F_{special}$, we could correctly discover almost all cheating reports and an increasing percentage of honest reports. Accordingly, the quality of the results was significantly increased as well. The clustering phase was actually not performed with $F_{special} > 1.0\%$ because above this threshold, using only the trust-distrust propagation was enough to evaluate the credibility of all reports. Although a lot of honest reports were wrongly identified as cheating, which was due to our cautious approach in estimating report credibility, the quality of the results was always very good if $F_{special}$ was kept high enough (about 5.0%). The results were the worst with $F_{special}$ around 2.0% and 3.0%. In these cases, the trust-propagation was not effective and we did not identify enough honest reports to predict service performance and we had to (partly) use the quality value advertised by providers instead. When the number of specially monitored services was very small (1.0%), the result was still acceptable (0.8), since in this situation we could actually combine the trust-propagation and the report clustering phase together. i.e., there were enough reports with unknown credibility after the preprocessing of reports s.t. the clustering step had enough input data to process. As a whole, the result of our algorithm with the trust and reputation management scheme in place is much better than that of the optimistic and the naive cases, as we expected.

Next, we studied the effects of the fraction of cheaters to the quality of results. We increased the total percentage of all types of cheating users ($F_{cheating}$), which consists of badmouthing, advertising and uncertain users, from 4.0% to 94.0% in increments of 10% each step. More specifically, we raised the percentage of badmouthing and advertising users/reports, from 3.33% to 78.33% while generating five different groups of uncertain users with corresponding percentage of dishonest reports increased from 0.67% to 15.67%. These setting represents the realistic case when there are various types of dishonest providers colluding with the generated cheating users to boost their own services and different providers intentionally produced lots of fake ratings to badmouth other services. This experiment might be considered as the most important one with various types of users with changing behaviors, giving us the results shown in Fig. 4. We kept the percentage of trusted reports at 0.5% and let $F_{special} = 5.0\%$, as an acceptable fraction collected from previous observations in the first experiment. The standard deviations of of cheating and honest reports were kept at $\sigma_c = 0.5$ and $\sigma_h = 0.01$ respectively.

With the reduction of honest users and the corresponding increase of dishonest users $F_{cheating}$, the values of the R-Precision parameter were also reduced. However, the quality of the results in the realistic case was always much better than that of the optimistic case and the naive case. Even when the fraction of cheaters $F_{cheating}$ was very high (0.84), the R-Precision parameter value in the realistic case was still acceptable (higher than 0.8). On the other hand, the quality of results without taking into account trust and reputation management issues, i.e., the optimistic and the naive case, dropped dramatically in hostile settings. This phenomenon was due to the fact that in a society with a very high cheating rate, our trust and reputation evaluation mechanism could discover and filter out almost all incredible reports, as shown in Fig. 5.

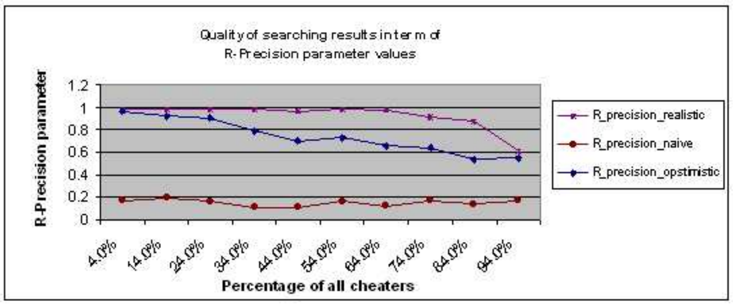

Fig. 4. $F_{cheating}$ vs. *R-Precision*

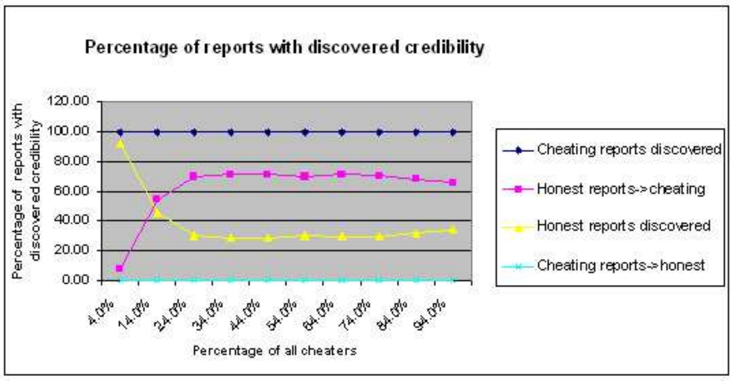

Fig. 5. $F_{cheating}$ vs. *Correctness of the report preprocessing phase.*

From these experiments we can draw a number of conclusions. Regarding efficiency and effectiveness, our selection and ranking approach exhibits the following properties: As the trust and reputation evaluation phase uses social network (trust-distrust propagation) and data-mining (report clustering) methods, it requires high-computational cost. Fortunately, in our case, the computation involves mainly local processing in one registry and thus does not require much communication overheads. Additionally, it could be done *off-line* on a periodical basis and therefore will not much affect the system performance. Another important observation is that almost no cheating report was wrongly evaluated as honest even in very hostile settings due to our cautious reputation evaluation mechanism. Therefore, in reality one can observe the results of the trust-distrust propagation step and incrementally adjust the parameters of the system, e.g., increase $F_{special}$, in order to collect enough honest reports for the performance prediction phase. This task to choose an optimal configuration for the system could be done off-line on a periodical basic as well. The service selection and ranking can be performed fast and efficiently thanks to the pre-computation of the matching information between QoS of existing services with possible quality requirements of users. Similar to [22], we conclude that the use of trusted third parties monitoring a relatively small fraction of services can greatly improve the detection of dishonest behavior even in extremely hostile environments. However, this effectiveness mainly depends on the properties of the social network model among service users and their corresponding reports. In our settings, these properties are represented by:

1. *The overlaps in the set of users for each service*, i.e., whether the services monitored by trusted agents have many users who are also consumers of other services.
2. *The inconsistency in the behavior of users*, i.e., whether a user is honest while reporting on a service but behaves dishonestly on other cases.

These factors suggest that we should deploy trusted agents to monitor the QoS of the most important and most widely-used services in order to get a *"high impact"* effect when estimating behaviors of users. Currently as the user reports are distributed uniformly for services in our experiments, we have not yet taken into account this factor. However, we have employed another technique to fight back the inconsistency behaviors of users by *detecting cheaters first and evaluating honest users later* in the preprocessing phase. This helps us to collect all possible evidences against cheaters by making use of the distrust propagation among reports at the first place. The philosophy behind is that it would be better to filter out a certain number of honest users rather than accept some dishonest reports. However, lots of credible users will be accidentally detected as cheating because of the similarity between their reports with other ones produced by well-disguised liars in the system. Thus, in the honest detecting (trust-propagation) phase, we also give these users a second chance to prove their honesty provided they have produced lots of credible reports which could be certified by trusted reports and other honest users. Additionally, other techniques can also be utilized to make this method more robust. For example, we can *pre-select the important services to monitor and increase the number of them* as well as *keep the identities of those specially chosen services secret* and *change them periodically.* Thus, cheaters will not know on which services they should report honestly in order to become high-reputation recommenders and have to pay a very high cost to have great influences in the system. In reality, this also help us to reduce the cost of setting-up and maintaining trusted agents as we only need to deploy them to monitor changing sets of services at certain time periods. Moreover, we can extend our algorithm so that *neighboring registries are allowed to exchange with each other the evaluated credibility of users* to further find out other possibly well-disguised cheaters who frequently change their behaviors. Note that the adjacent registry peers in our system are assigned to manage services with similar functional characteristics and therefore they are likely to attract comparable sets of users in the system [9].

6 Conclusion

In this paper, we have introduced a new QoS-based service selection and ranking algorithm with trust and reputation management support. We have shown that our selection and ranking solution yields very good results in most cases. As the proposed reputation management mechanism is robust against various cheating behaviors, the results are generally of good quality even in hostile situations in which many different types of cheaters make up a high percentage of the overall users and report values with remarkable variances. By combining a trust-distrust propagation approach with a data-mining method, we could filter out almost all cheaters and find out honest reports to be used in the quality prediction process with very high probability. However, as we limited our work on only some specific quality attributes, it would be necessary to generalize the approach so that it could be used in a more descriptive and flexible QoS model. The selection of services to be monitored by trusted agents is an interesting point not yet to be mentioned. Another open problem is how to accurately predict the performance of newly published services with only few QoS reports. This may relate to the issue of evaluating providers' capabilities in complying with the advertised QoS of their new services from existing reputation data. The selection of an optimal configuration for many design parameters of our proposed solutions is also an important question to be studied in further. We plan to develop a so-called meta-behavior model of users, which is more general to describe user behaviors with various possible probabilistic-based responses to obtain analytical results of the proposed solution. We are also going to deploy our algorithm in a decentralized setting to observe the effectiveness of our trust and reputation techniques where there are many registry peers exchanging among each other the information of users and services' quality data.

Acknowledgements

We would like to thank Zoran Despotovic, Amit Lakhani and Fabio Machado Porto for many helpful discussions with them as well as for their carefully reading and commenting the paper.

References

1. DIP Integrated project- Data, Information, and Process Integration with Semantic Web Services, http://dip.semanticweb.org/.
2. P. Resnick, R. Zeckhauser: Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System, The Economics of the Internet and e-Commerce, Advances in Applied Microeconomics, Amsterdam, Vol. 11, Elsevier Science, 2002.
3. M. I. Melnik, J. Alm,: Does a Seller's e-Commerce Reputation Matter? Evidence from eBay Auctions, Journal of Industrial Economics 50(3), 2002, p.p. 337-349.
4. M. Tian, A. Gramm, T. Naumowicz, H. Ritter, J. Schi: A Concept for QoS Integration in Web Services, Procs of the 4th Intl. Conf. on Web Inf. Sys. Eng. Workshops, Italy, 2003.
5. S. Ran: A Model for Web Services Discovery with QoS, ACM SIGecom Exchanges, Vol. 4, Issue 1 Spring, pp. 1-10, 2003.
6. M. Ouzzani, A. Bouguettaya: Efficient Access to Web Services, IEEE Internet Computing, March/April, pp. 34-44, 2004.
7. C. Patel, K. Supekar, Y. Lee: A QoS Oriented Framework for Adaptive Management of Web Service based Workflows, Database and Expert Systems 2003 Conf.
8. E. M. Maximilien and M. P. Singh: Reputation and Endorsement for Web Services, SIGEcom Exch., 3(1):24-31, ACM Special Interest Group on e-Commerce, 2002.

9. H. L. Vu, M. Hauswirth and K. Aberer: Towards P2P-based Semantic Web Service Discovery with QoS Support, Proceeding of Workshop on Business Processes and Services (BPS), Nancy, France, 2005 (to appear).
10. N. Srinivasan, M. Paolucci, K. Sycara: Adding OWL-S to UDDI, Implementation and Throughput, Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, USA, 2004.
11. Latest UDDI Version (3.0.2), UDDI Spec Technical Committee Draft, Dated 20041019, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm.
12. H. Ludwig: Web Services QoS: External SLAs and Internal Policies Or: How Do We Deliver What We Promise?, Proceedings of WISEW'03, Roma, Italy, 2003.
13. Glen Dobson: Quality of Service in Service-Oriented Architectures, 2004, http://digs.sourceforge.net/papers/qos.html.
14. A. Jøsang, R. Ismail and C. Boyd: A Survey of Trust and Reputation Systems for Online Service Provision, Decision Support Systems, 2005 (to appear).
15. Z. Despotovic and K. Aberer: Possibilities for Managing Trust in P2P Networks, EPFL Technical Report No. IC200484 , Switzerland, November, 2004.
16. Z. Chen, C. Liang-Tien, B. Silverajan, L. Bu-Sung: UX - An Architecture Providing QoS-Aware and Federated Support for UDDI, Proceedings of ICWS'03.
17. A. S. Bilgin and M. P. Singh: A DAML-Based Repository for QoS-Aware Semantic Web Service Selection, Proceedings of ICWS'04.
18. S. Kalepu, S. Krishnaswamy and S. W. Loke: Reputation = f(User Ranking, Compliance, Verity), Proceedings of ICWS'04.
19. Y. Liu, A. Ngu, and L. Zheng: QoS Computation and Policing in Dynamic Web Service Selection, Proceedings of WWW 2004 Conference.
20. F. Emekci, O. D. Sahin, D. Agrawal, A. E. Abbadi: A Peer-to-Peer Framework for Web Service Discovery with Ranking, Proceedings of ICWS'04.
21. J. Day and R. Deters: Selecting the Best Web Service, the 14th Annual IBM Centers for Advanced Studies Conference, 2004.
22. R. Guha and R. Kumar: Propagation of Trust and Distrust, Proceedings of WWW 2004 Conference.
23. M. Richardson, R. Agrawal, P. Domingos, Trust Management for the Semantic Web, Proceedings of ISWC 2003, LNCS 2870, p.p. 351-368, 2003.
24. K. Aberer and Z. Despotovic: Managing Trust in a Peer-2-Peer Information System, Proceedings of ACM CIKM'01.
25. A. Whitby, A. Jøsang and J. Indulska: Filtering Out Unfair Ratings in Bayesian Reputation Systems, Icfain Journal of Management Research. Vol. IV, No. 2, p.p. 48-64, February 2005.
26. E. Manavoglu, D. Pavlov and C. L. Giles: Probabilistic User Behavior Models, the 3rd IEEE Intl. Conf. on Data Mining, Melbourne, Florida, 2003.
27. F. Cornelli, E. Damiani, S. C. Vimercati, S. Paraboschi and P. Samarati: Choosing Reputable Servents in a P2P Network, Proceeding of WWW 2002 Conf., USA.
28. K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: A self-organizing structured p2p system. *ACM SIGMOD Record*, 32(3), 2003.
29. Web Service Modelling Ontologies, http://www.wmso.org/.
30. Web Ontology Language for Services, http://www.w3.org/Submission/2004/07/.
31. H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner: Ontology-based Integration of Information - a Survey of Existing Approaches, Proceedings of IJCAI'01 workshop on OIS, p.p. 108-117, 2001.
32. Y. Kalfoglou, M. Schorlemmer: Ontology Mapping: the state of the art, The Knowledge Engineering Review 18(1): p.p. 1-31, 2003.
33. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, Semantic Matching of Web Services Capabilities, Proceedings of ISWC'02.
34. F. Naumann: Data Fusion and Data Quality, Proc. New Techniques and Technologies for Statistics Seminar (NTTS), IO-Press, p.p. 147154, 1998.