Reputation-based Service Composition

Seminar Thesis
Submitted to the Software Engineering Research Group
in Partial Fulfillment of the Requirements for the
Seminar
Cloud Computing and Software Services

by
SIDDHARTHA MOITRA

Thesis Supervisor:
M.Sc. Marie Christin Platenius

Paderborn, September 2014
Contents

1 Introduction 2

2 Background 4
  2.1 Web Services .............................................. 4
  2.2 Trust and Reputation ........................................ 4
  2.3 Reputation Systems and Service Selection .................. 5

3 Orchestration of Services based on Reputation 7
  3.1 Malik and Bouguettaya ...................................... 7
  3.2 Bianculli et al. .............................................. 8
  3.3 Malik and Medjahed .......................................... 10
  3.4 Paradesi et al. .............................................. 10

4 Evaluation of Reputation based systems 12

5 On-The-Fly Computing: Reputation Management 15

6 Discussion 17

7 Conclusion 19

Appendix

A References 20

Bibliography 21
Abstract

Cloud computing is the latest evolution in the field of distributed computing. This advocates the composition of basic distributed web services to form a more complex Second Generation Service encapsulating its complete business workflow. Trust and reputation management have emerged in industry and academia for selecting the best web services available and thus forming the best-in-class compositions. This paper focuses on the state-of-art for managing reputation and how factors involving reputation can be used to construct well-defined compositions. I surveyed some techniques, presented in literature, involving different implementations of reputation managers and their integration with the web service architectures. Every technique is analysed with their advantages and disadvantages and a comparison with other methods, is stated. Finally, I also present On-the-Fly Computing, a research project at the University of Paderborn and the reputation matching approach used there to automatically compose services published in a dynamic market. I also perform a comparison of On-the-Fly Computing reputation technique with other techniques discussed. It can be concluded that the reputation approaches have a major dependency on the domain and technologies used. Hence, there is no single technique which can be termed as the best-in-class.
1 Introduction

Web services are an emerging technology to solve varying tasks ranging from simple requests to complex business processes. With the ability of communication between distributed and heterogeneous systems, web services are considered to be self-contained, modular and platform-independent components that perform a specific business function [W3C] such as searching a flight ticket or transferring an amount from one account to another.

One of the greatest advantages of web service architecture is the combination of basic services to create a more complex service which combines into a well-defined business process. Hence, if basic services are present in the marketplace, the provider just needs to compose them in order to produce a complete and complex business process. An example to highlight the foundation of composition is shown in Figure 1.1.

![Figure 1.1: Service composition using an orchestrator](image)

Imagine a user wants to plan a travel and needs to book a flight ticket and a hotel for his/her stay. To achieve the goal, the user selects an orchestrator and sends the request to it. The role of the orchestrator is to facilitate the service composition such that it could respond to the user’s request and abstracts the information exchange between the basic services from the user. Also, the orchestrator provides the user a single view of the transaction. This architecture enables the basic services to exchange user’s data efficiently and does not require the user to re-enter his/her data again for multiple basic services. However, in a real world
service market multiple services are published which perform similar functionality and the orchestrator’s job is to select one of the services from the bunch of similar services.

The service selection techniques used by the orchestrator primarily depends on the functional aspects of the participating services [WV07]. However if the number of providers of functionally similar service is more, the orchestrator should be able to choose the best and the most reliable service. The reliability of any service depends on its historic performance data and the pre-consumer’s feedback. It is also noted that the service markets are often dynamic in nature [BYV08]. Services tend to change their behavior in terms of performance degradation, new services using better technologies are added. All these constraints leave the orchestrator clueless about which service should be chosen and which must be discarded. The selection of the services should (apart from functional properties) also take into consideration the rank and status of the participating services. Formal mechanisms that could distinguish between “good” and “bad” services are needed [WV07].

The distinction between good and bad service can be understood by observing the historic performance of the services and consumer feedback provides the backbone to observe the service’s trend in terms of service quality [JIB07]. As a result, trust and reputation management has gained enormous attention in recent years. The approach help participants to know their providers better, if they don’t have prior knowledge and help facilitate efficient business transaction [LS10]. Some examples of real world e-commerce applications, such as eBay, Amazon, have implemented reputation management. In this paper, I discuss about different applications involving trust and reputation models and how reputation is taken into consideration during service compositions. An analysis of classifying reputation-based service selection, serve as a major section of this paper. I also discuss the application of reputation during service matching in On-The-Fly Computing, a research project at the University of Paderborn that aims to develop techniques for automatic service compositions.

The paper is organized as follows. Chapter 2 provides the detailed, yet summarized, background on web services and defines trust and reputation management. The survey and description of reputation-based service selection forms the content of Chapter 3 as the major chapter of this paper. Chapter 4 talks in detail about On-The-Fly Computing, the reputation system used there and compares the reputation system with others presented in this paper. Concluding with Chapter 5, I discuss some challenges in the field of reputation-based service composition and the future trend of reputation models in this field.
2 Background

This section of the paper serves as a wider background provided to create a base for further chapters. I discuss the foundation of web services, followed by defining trust and reputation. Reputation systems and service selection based on trust and reputation are also explained with reference to web services.

2.1 Web Services

A Web Service is defined as a piece of code designed to fulfill a specific functionality encapsulating the business logic from the consumer and providing the access through an API invocation over network [W3C]. Web Services form an implementing component of the loosely coupled architecture stack called Service-oriented Architecture (SOA) [SOA]. The stack defines an universally accessible registry, UDDI, for service providers to publish their services’ meta-data [UDD]. Web Services Description Language or WSDL specification is entrusted upon for describing a contract between service provider and requestor [WSD].

However, the Web Service Technology Stack does not provide any standard for specifying the Quality of Service or does not guarantee on the availability of the service [MK13]. With no formal definitions available on the quality of services, it is quite possible that the client chooses a “bad” service. Therefore, we require metrics to measure and declare a 360° view on the quality of service. These metrics should be able to answer service consumers whether they should choose a specific provider in the composition or not and also resolves the dilemma to choose among several functionally similar service providers.

2.2 Trust and Reputation

As the software development moves from monolithic structures to distributed applications with business compositions providing solutions, the terms trust and reputation become very important for these compositions. Broadly defining, trust is the ability by which a component’s behavior can be aligned to that of a predicted value. A trust factor defines and predicts the quality of service of the component in future. Grandison and Sloman [GS00, p4] provides the following definition for trust, as

“...the firm belief in the competence of an entity to act dependably, securely and reliably within a specified context.”
Unlike trust, reputation is the combined value of all the consumers using that component to define its history. Hence, trust is the quality information predicting the future whereas reputation can be viewed as the quality information of the past [Tru].

The trust and reputation parameters have been of great importance in service composition. Formally described metrics are used by intelligent systems in determining which services are to be used from service market in composition. Trust is always defined based on some context. For example, Component A trusts Component B for all Hotel Bookings in Dubai. Reputation, on the other hand, can be explained as one of the outcome of trust [AH14].

There are numerous discussions on whether or not there exists a relation between trust and reputation. Jøsang et al. [JIB07] defines trust as an individual’s knowledge on trustee’s behavior and depends on numerous other factors besides reputation. It is quite evident that the “private knowledge” of the trustor exceeds the collaborative feedback of reputation. In this case, the trust factor is not dependent on the reputation earned by the trustee. On the contrary, it is quite possible that the trustor has no previous knowledge of the trustee before any communication. In that case, a system having a high reputation value, will eventually attract more consumers and if the subsequent interactions follow high quality of service, it will create a high reputation in the marketplace. Authors, however, also mention that trust and reputation are directly proportional to each other and follow a linear graph [DL02].

Maintaining a highly robust system, flexible enough to tackle internal changes smoothly and adherence to the SLA with high quality, remains the major focus in service composition where trust and reputation remains one of the major pillars for assisting a consumers’ decision of selecting the best, trusted and highly ranked services from the service market.

2.3 Reputation Systems and Service Selection

As the distributed computing gained more popularity, there has been numerous research done on the parameters that should be taken into consideration while composing services. Trust and reputation form one of the pillars in defining systems with a sole responsibility of accepting, processing and storing feedbacks from consumers and providing those ratings to the orchestrator in order to select basic services [WV07]. The benefits achieved in using these systems enable dynamic replacement of error-prone or degraded services with newer ones. There exist other advantages like selecting services with ratings greater than threshold in order to compose best-in-breed solutions at any given point of time [BJB07].

The integration of reputation system with the cloud architecture can be exemplified by the scenario, as shown in Figure 2.1 which is an extention of the example described by Figure 1.1: consider a real world service market scenario, where a user delegates his/her request to book a flight and a hotel to an orchestra-
2. Background

In turn, the orchestrator needs to use functional property matching to select and compose services from the service marketplace that could respond to the user request. Here, we assume the orchestrator selects service from Country Inn and Ibis Hotels for staying purpose and AirBerlin and AirFrance for travel. Since the orchestrator selects more than one service for each domain - flight and hotel, it must create a best-in-breed service composition. However, the orchestrator has no means of identifying the best service in terms of quality of service and the reputation of the provider. Therefore, it is suggested to introduce a new system in the architecture, a *Reputation Management System*. This system is responsible for storing, providing and updating the reputation values of the services published in the market. The orchestrator, thus, requests reputation manager and gets the *reputation scores* of the services it has selected from the market. Based on these scores the orchestrator chooses the highest reputation values of the services from each domain and sends *response* to the user’s request. Once the transaction is complete, it is important for the user to provide a *feedback* as a result of his/her experience with the service quality. The feedback is forwarded to the reputation system which then *updates the reputation scores* of the basic services which took part in the transaction.

Because of the advantages and enormous literature available on reputation management, it has been able to place itself firmly in the industry with e-commerce applications like Amazon and eBay using them.
3 Orchestration of Services based on Reputation

In this section of the paper, I present a survey of some of the most important reputation-based web service selection and assess different methods that I find in literature review. In Section 3.1, I present a de-centralized reputation system which uses the concept of communities and stores reputation information in a distributed manner. Section 3.2 uses a more centralized approach and stores the reputation information in a centralized entity. In Section 3.3, I discuss the reputation system which is more suited for hierarchical compositions of services where the default service only needs to take the blame for fall in performance of the entire composition. Section 3.4 uses a trust model and considers trust as the major reason in selecting a service, rather using its reputation value.

3.1 Malik and Bouguettaya

Malik and Bouguettaya [MB09] propose a distributed approach of managing reputation and selections based on it. The architecture, RateWeb, introduces a concept of Communities - a group of services bound by similar semantics and functionality. Technically realized using ontologies, the communities also act as services that are registered with the service registry and manage the list of consumers who have already used and rated the services. The ratings are not maintained by any centralized system or the community system but for any given provider, stored local with consumers called personal evaluation. The consumer interacts with all the peer ex-consumers of the service and gets the ratings from all of them and aggregates them using its own strategy. This aggregated value is also called assessed reputation.

Figure 3.1 describes a high-level architecture of the reputation management system as suggested. The service orchestrator requests data from the service registry for the list of communities which contain the desired service. It is noted that all the communities must register to the service registry. Once the consumer receives the list of communities, it requests the communities directly for the list of all the services and also the list of pre-consumers of those services. The community stores the list of pre-consumers and sends it to the client. On receiving, the client can query the pre-consumers directly and get the reputation scores. Aggregation of the reputation scores depend upon the algorithm the client wants to implement.

The authors also implement strategies to counter possibilities of malicious ratings and assess the credibility of consumers who rate the service. This enables the
3. Orchestration of Services based on Reputation

Figure 3.1: Community-based architecture by Malik and Bouguettaya

consumers to not only rate the service but also judge the feedback of other peer services. This is quite an important differentiating factor because of its nature of combating “reputation attacks”. However the overhead costs involved depends upon the collection model used in reputation systems.

The benefit of this approach is: there exists no central repository for ratings and hence no “single point of failure”. Each consumer has its own set of protocols to assess the service quality of any provider. The consumer also has the ability to aggregate and collect the ratings from different peer consumers. This property could be as well argued as a down-side for consumers: First, every consumer implements its own vision of reputation specification and there exists no common “model” for reputation assessment. Second, every consumer needs to implement a strategy for reputation aggregation. Hence, a common platform for reputation assessment might stay absent. Third, the overhead involved is quite large because of an excessive request-response cycle between client and other components of the market.

3.2 Bianculli et al.

This paper relies on a centralized reputation manager for collecting and aggregating the reputation scores from the consumers. Bianculli et al. [BJB+07] proposes the use of monitoring by the clients for both the functional and non-functional qualities of the providers. The information is then published with the reputation manager which then is processed and analyzed for further use. It is also the job of reputation manager to provide aggregated values to any consumer requesting for the ratings. The service consumers register themselves with the subscription manager. The role of subscription manager is to notify all the registered clients whenever there is a change in the reputation value (computed by reputation manager) and the value falls below a certain threshold. It also notifies the consumers whenever a new provider with better reputation value, subscribes into the registry.

Figure 3.2 shows the architecture of the reputation management system as sug-
gested. The service orchestrator sends the request with reputation expectations and it is responsible for composing services in accordance with reputation information from the reputation manager, which is a centralized component processing and providing reputation scores. The service directory is a component of the registry which holds the information of all the services in marketplace. Whenever a new service enters the market, it needs to register itself with service directory. The subscription manager is the component that deals with providing notifications to the orchestrator. The subscription manager receives information from reputation manager, if the reputation value falls below a threshold and service directory, if a new service enters the market. The feedback from the consumer is sent by the orchestrator to the reputation manager which then processes it to compute a new value of the services taken part in the transaction.

One of the important dependencies of this architecture is that the client provide honest feedback to the system. Realization for client feedback is done using Hidden Markov Model with two states: good and bad. A good state is received when the provider adheres to the service contract (including functional and non-functional properties) and is assigned with an unknown probability. “The probability of transition from the good state into the bad state is assumed fixed and known…” [BJB+07]. Payment schemes are introduced in the architecture to combat malicious feedbacks by clients and governed by reputation manager. According to the authors, the current feedback is supposed good if it has same value with an earlier randomly chosen feedback.

The authors claim the advantage of this architecture as pro-active because the notification service provided enables the orchestrator to swap between services before any provider’s Quality-of-Service degrades. It can also be noted that because of the use of centralized reputation registry, the architecture has a low scalability quotient.

Figure 3.2: Components of reputation system and their interactions
3. Orchestration of Services based on Reputation

3.3 Malik and Medjahed

An idea of propagating reputation information to down-stream combining services is introduced by Malik and Medjahed [MM10] in this paper. The paper focuses majorly on issues where the compositions are hierarchical in nature, i.e., compositions of compositions. As the reputation information of the compositions are also stored, it helps the compositions to make known selections of their services. Authors claim that the reputation of compositions must be known by the orchestrator. Therefore, if the reputation falls below the orchestrator’s perceived or assessed reputation, then it should be able to diagnose the reason and must be able to propagate the reason of decrease in reputation to its sub-services. This can be done in terms of replacing the bad service or penalizing it.

There is a strong motivation of implementing such a combating proposal using a Statistical Cloud Model where the fuzzy membership functions are integrated with probability theory. Since the Cloud Model is a hybrid approach which takes the advantages of both fuzzy membership functions as well as probability theory, the results prove it to be more robust, simple and efficient in nature. The basic idea of this approach is “... past defaulters are likely to defect again in a composition.” [MM10]. Once the reputation of the orchestrator falls below its perceived reputation, the orchestrator passes the blame to its sub-services in a manner defined by the Cloud Model. It is assumed that the highest blame is passed to the most likely default service, with continuous post monitoring of this service. The post-monitoring also helps the orchestrator to pick out the services which performed as per expected SLA.

The theory suggested in this paper has been also supported by experiments. It has been shown that, if any service receives a bad report, it tries to rectify its behavior. The work presented in this paper improves a lot on the current approaches discussed earlier, as the architecture does not depend on any specific third party reputation system.

3.4 Paradesi et al.

In their paper, Paradesi et al. [PDS09] propose the use of Trust Models, which help in composing service and deriving trust based on those models. In addition to this, they also try to correlate predicting trust value and actual experiences of the consumers. Trust, as described in the paper, is a multifaceted attribute corresponding to the security and authenticity of the service and also acts as judgemental attribute stating whether the service conforms to the promises it has made. The use of Trust Models using a second order probability density function is a way of formalizing their belief of whether a service with high probability is trustworthy. The authors claim not to treat trust as one of the parameters of QoS, but rather an additional function depending on which the compositions take place. Hence, the authors develop a separate trust framework Wisp that computes trust
of a composition and selects the composition with a higher trust quotient. This approach is helpful, when there exists a hierarchy of compositions, for example in On-the-Fly Computing as discussed later in Chapter 5.

The framework suggests that the trust definition must be revised for every service based on its past feedback and thus, the authors specify an update function based on trust parameter. The basis of this framework is formally defining a Trust Vector which is a three dimensional function; one dimension each for Trust, Distrust and Uncertainty. An experience of any consumer with a service is also defined based of three parameters: Competency, Reliability and Honesty as predictors of positive experience.

To derive trust quotients for compositions, the authors analyzed four types of basic flows of services (also called constructs): Sequential flow, Concurrent flow, Conditional flow and Loops. For every construct, a belief density could be formulated and composing the constructs together will provide the belief density for the entire composition. This process may be iterated to find the aggregated trust quotient of the overall orchestration because of its hierarchical nature. One of the fine developments here is that the users can also provide feedback to the service components taking part in orchestration. The framework presented is responsible for selecting trustworthy services for deploying in the orchestration.
4 Evaluation of Reputation based systems

This chapter focuses on the evaluation of the reputation architectures presented in Chapter 3. As all the approaches come with their own set of pros and cons, I use four key requirements to rate and compare every approach. As suggested by Zhou et al. [ZH07], these requirements form the basis of designing an efficient and effective reputation system. A point to note is that these ratings are not based on experimental findings but rather a comparison among all the techniques available based on personal evaluations. I also present a comprehensive report of the evaluations and assessment in a tabular format in Table 4.1.

- **Scalability:** The system should be able to efficiently serve an increasing number of peers in terms of Overhead, Accuracy and Convergence Speed.

The reputation system suggested by Malik and Bouguettaya offers a distributed solution to the orchestrator, with the reputation values being stored at client location. In this approach, scalability depends mostly upon the response time in service registries and the communities. Using adequate data structures, help to increase the scalability from high to very high values.

However, the reputation system presented by Bianculli et al. is designed to have a centralized repository of reputation values, scalability is, to a large extend, decreased with a bottleneck forming at the reputation manager component. See Figure 3.2.

The reputation system suggested by Malik and Medjahed involves use of two reputation values - consumer’s “personal evaluation”, which is stored at consumer’s location and provider’s “assessed reputation”, that is available globally and stored as a centralized entity. Hence the scalability can be ranked between low to medium, depending on the data structures used while storing the reputation values.

In the paper by Paradesi et al., the reputation system - Wisp, uses trust vectors to filter out the services and stores these values with itself. However if the number of Wisp nodes are increased as the services increase, the scalability can be improved between medium and high.

- **Overhead:** The ability of the system to consume limited resources and cost while reputation computation and process.

In the reputation system suggested by Malik and Bouguettaya, the reputation values are stored as raw values and need aggregation which is done at the orchestrator. This involves an extra overhead when accumulating the values. Also, the
number of interactions between the orchestrator and cloud entities, before the actual reputation score is sent, is higher. The architecture, thus, has a high overhead in terms of calculating reputation score.

The reputation system presented by Bianculli et al. involves storing an aggregated reputation score. The calculation can be performed by the reputation manager offline and, hence, has no effect on the overhead, that ranges from low to medium.

The reputation system developed by Malik and Medjahed uses a comparison of two reputation values - personal and assessed reputation score and hence has a very high overhead in terms of calculations and latency.

Wisp, as suggested by Paradesi et al., performs better in terms of overhead incurred and tends to place itself in a medium range. Since filtering is done at service and composition level based on trust vectors, an increased latency in orchestrating the services is expected.

- **Accuracy**: The system must be able to calculate the reputation score as close as the real trustworthiness.

RATEWeb, as presented by Malik and Bouguettaya, is considered to be very accurate as it calculates reputation score taking into consideration all the peers. However, the framework doesn’t give any information on the calculation of flexible reputation scores. For example, the reputation score of the latest n feedbacks. Hence, the rating should range between low to medium.

The reputation system presented by Bianculli et al. uses an aggregated value for storage and no flexibility in terms of the algorithm used for calculation or the number of ratings to be used. Thus, comparing to other models, this architecture scores medium on accuracy.

Malik and Medjahed propose a reputation system that takes into account a consumer’s perception of a certain provider. As the perception might differ with every consumer based on the context, the accuracy of this approach is not only a function of reputation scores but also of context.

Using probability functions in Wisp by Paradesi et al. and storing these trust functions as a raw value in a distributed format, the accuracy is relatively high than other models discussed. The filter then uses only those service whose trust functions are greater than the threshold.

- **Robustness to malicious attacks/peers**: The system must always accept feedbacks from legitimate service consumers. Illegal feedback must be avoided at any cost, as it will affect the accuracy of the reputation scores.

Malik and Bouguettaya propose in their solution to distinguish between “credible” and “malicious” raters. However, their solution is based on using a probabilistic function and does not give a concrete solution to this problem.

Bianculli et al., however, propose the use of economic schemes and payments to client whose feedback matches to that of a randomly chosen feedback by another
client for the same service. Still, the architecture is not free from attacks and needs further modifications to counter malicious feedbacks.

In the architecture proposed by Malik and Medjahed, consumers do not provide feedback. But if any provider fails to provide Quality of Service as promised, the reputation system penalises the provider. This way, the provider refrains from giving over-expected promise of performance.

Paradesi et al. in their solution, do not provide any discussion on how to prevent malicious feedbacks. If similar architecture is to be followed without any enhancements to include combat for Sybil attacks, the architecture would perform the worst in handling malicious attacks.

We can observe that every model has its own set of pros and cons and there exists no suitable model that performs best against all the criterias. The requirements that are identified has dependencies among each other. For example, the scalability of an architecture can be increased using a de-centralized approach. But this has a consequent negative impact on the overhead because the number of interaction increases. Therefore, a perfect balance needs to be identified here. Similarly, accuracy also has an impact on overhead. We observed that accuracy can be substantially improved if raw values are stored and the final reputation value is computed everytime using an algorithm. Therefore, it is quite important to have a complete understanding of the architecture and make a trade-off decision while choosing any reputation system. Combating malicious attacks remains one common open issue with all the architectures.

Table 4.1: Evaluation summary of reputation-based systems

<table>
<thead>
<tr>
<th></th>
<th>Scalability</th>
<th>Overhead*</th>
<th>Accuracy</th>
<th>Robustness to attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malik and Bouguettaya</td>
<td>↑</td>
<td>↑</td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>Bianculli et al.</td>
<td>↓</td>
<td>∼</td>
<td>∼</td>
<td>∼</td>
</tr>
<tr>
<td>Malik and Medjahed</td>
<td>∼</td>
<td>↑↑</td>
<td>∼</td>
<td>∼</td>
</tr>
<tr>
<td>Paradesi et al.</td>
<td>∼</td>
<td>∼</td>
<td>↑</td>
<td>↓↓</td>
</tr>
</tbody>
</table>

Legend: ↑↑, very high; ↑, high; ∼, medium; ↓, low; ↓↓, very low

*: The lower the values, the better they are.
5 On-The-Fly Computing: Reputation Management

On-The-Fly Computing is a research project at the University of Paderborn which aims at the dynamic composition and provision of services in accordance with client request [JBPP14]. In addition, the project also includes provisions for quality assurance and recognizes the OTF Market as dynamic with support to include interaction among participants [SFB]. Therefore the need of implementing a reputation based service selection is essential.

Figure 5.1: High level architecture of OTF reputation system

Figure 4.1 shows a high level architecture of OTF reputation system. The orchestrator requests the Reputation Values component of the system which provides the reputation information of OTF providers, service providers, service compositions and basic participating services. Thus, the orchestrator has the flexibility to choose the best composition hierarchy. The user sends feedback to the Accumulated Ratings component which stores the raw values of the reputation before passing it to the Aggregation System. The aggregation system processes these raw values to combine them and form aggregated reputation values. The reputation management system proposed for OTF requires consumers to rate their experience in 3 different sections, namely, Composed Services, OTF Providers and Base Services. However, it is most often not feasible to have users rate the base services. In that case, the reputation mechanism assigns the reputation values to the entire composition and further decomposes it according to the composition information. In the following section of the chapter, I assess the OTF Computing reputation mechanism against some important expectations that should be addressed in the design of a reputation management system.
I will use similar parameters as in Chapter 4, to rate OTF Computation reputation system. These parameters are derived from the works of Zhou et al. [ZH07].

- **Scalability:** The reputation system in OTF Environment works as a stand-alone component [JBPP14]. To improve scalability of the entire system, it is very important that multiple reputation components exist in the environment. This could follow similar approach as suggested by Conner et al. using TMS nodes [CIM+09]. The actual implementation is beyond the scope of this paper and is still being decided upon. If the above mentioned approach is followed, the scalability of OTF reputation system could range from high to very-high depending upon the factor of number of instances running in the OTF environment.

- **Overhead:** The decision-making process involves 2 stages: OTF Provider selection and Service Provider selection. The overall process must be streamlined using efficient data structures for storage, retrieval and process. Raw ratings can be stored and retrieved using Distributed Hash Tables which involves less overhead. Also, the ratings in OTF Environment are stored as raw values and reputation values are computed on demand only. Caching is one possible solution to decrease overhead and improve throughput as it is suggested by Conner et al. [CIM+09]. Thus, using efficient storage and caching, the overhead can be suppressed from medium to low.

- **Accuracy:** The reputation system in OTF Computing can be termed as highly accurate as it stores all the raw values and computes the aggregated value on-demand. Flexibility is even provided as the orchestrator can choose among a range of algorithms that can be used to calculate the reputation value. Further on, to make the system as scalable as possible, the raw values are stored in a distributable fashion within the system. Therefore, the system can be rated between the range of medium to highly accurate.

- **Robustness to malicious attacks/peers:** The OTF Environment uses a novel idea of transmitting a unique ID for clients taking part in feedback. This unique ID is only provided by the OTF provider if the client has performed successful transaction. This technique successfully curbs malicious clients to send wrong feedbacks without using the service composition. Introducing a payment scheme as suggested by Bianculli et al. [BJB+07], can also be used for clients providing thorough and honest feedback.
6 Discussion

After discussing some reputation-based systems, we conclude that there exists no perfect solution for managing reputation effectively and all the reputation systems discussed so far are in their pre-mature state. There are several unresolved topics that need serious discussion that I would like to present in this section. These challenges and threats pertaining to the current research theories about reputation management systems should be taken into consideration in future work. I also present an idea which could help counter these issues. Though, the practical feasibility will remain out of scope of this paper.

One of the most important threat to reputation-based systems is malicious feedbacks or Sybil attacks. It “...is an attack wherein a reputation system is subverted by forging identities in peer-to-peer networks”[Syb]. Another threat to a service oriented system is the high execution cost of the services. As the execution of a service remains primarily within the system and may affect the execution costs of other services sharing same resources.

To address these challenges, let us take the example of Hotel and Flight booking as explained in Figure 2.1 and integrate a new component to resolve some open issues.

Figure 6.1: Integration of Internal Audit System with Reputation Systems and Cloud architecture
Referring to Figure 6.1, I would like to put forth an idea of Internal Runtime Auditing of the services by reputation management systems. As observed, most of the designs are not capable of combating Sybil attacks or malicious feedbacks, the idea of putting an internal check on services might drastically decrease the threats. The proposed auditing system will intercept the communication between consumer-provider and will function offline without interfering in any communication. This will have no impact on the incurred overhead. The reputation manager can only accept feedback from the client whose interaction has been monitored by the auditing system and assigned a unique transaction id for feedback about the interaction. This will ensure authenticity of the transaction and confirm the reputation manager of the legitimacy of the feedback from the client thereby reducing the chances of malicious feedbacks. Another benefit of this idea would be assessing the cost parameters of the services involved. It is quite possible that a service adheres to the SLA and receives positive feedback resulting in high reputation score. However, during the course of its execution it affects the runtime performance of other services by consuming the shared resources such as processor occupancy, network bandwidth, locks on shared files. For implementing a pro-active measure to counter the execution cost threats, it is suggested when any service provider registers itself, it must also provide an estimated cost function of the service. The auditing system will also check the actual cost with the provided cost estimation. The reporting mechanism integrated with the auditing system will report to the reputation manager in case of high difference between the estimated cost and the actual cost of execution. In this way, the reputation manager would be able to take a pro-active measure to combat the threats of reputation deterioration with other services.

In their paper, Márnl et al. [MK13] also discuss some limitations, such as, mechanisms to provide initial trust and reputation values to services and scalability issues. There also exist incompatibilities in the language and vocabulary, semantics used by participants. Therefore the need of systematic trust based on common understanding of semantics will hold an important position going forward. It is also important to point out the effective ways of storing, analyzing and retrieving reputation data is going to be one of the most important areas of interest in near future. With vast amount of raw data being processed, it must be important for researchers to dive deep into defining the data structures for efficient storage and retrieval. The authors claim for immediate attention to these issues so as to have successful deployment and wide acceptance in the industry.
7 Conclusion

Services enable applications to build complex business process by smart selection of simpler base services. Reputation and trust values provide the basis of selecting the best-in-breed service among several set of service with similar functional properties and promised quality of service. In this paper, I have motivated why we need such reputation systems and how we can safeguard the consumer’s interests by providing him with a “good” service. With an understanding of reputation systems and service selections, I have surveyed in detail some important and diverse architectures, compared and rated them on their merits and demerits.

However, I conclude that these methods are still not mature and most of them are in the early stages of definition and many challenges exist that needs defined solution. There also exists a trade-off decision that needs to be made while selecting which approach to choose in designing a reputation system because there exists no approach which scores high in all the parameters. I have observed in many cases that authors have not been able to underpin the implementation and the deployment of their reputation systems with current cloud architectures. Following which, I explained on how reputation management works in OTF Computing and compared the same with other designs presented. Many open issues with other reputation systems have been taken into consideration while designing OTF Computing reputation system. However, the concrete implementation is still under discussion and will have many iterations in coming years. I have also made some suggestions as a part of future work in proposing the use of Internal Audit System which can answer to many open issues related to malicious feedbacks and performance monitoring.

Reputation mechanisms offer an elegant way to select the best-in-class services and form compositions. There has been tremendous research in the area of service composition and cloud architectures, however the models have not attained enough maturity and more investigation in this area will definitely accelerate the research for using trust and reputation systems.
Appendix A

References

API       Application Programming Interface
OTF       On-The-Fly
QoS       Quality of Service
SLA       Service Level Agreement
SOA       Service Oriented Architecture
SOAP      Simple Object Access Protocol
UDDI      Universal Description, Discovery and Integration
WSDL      Web Service Description Language
Bibliography


