Reputation-based Web service orchestration in cloud computing:
A survey

Félix Gómez Mármol* and Marcus Quintino Kuhnen
NEC Laboratories Europe, Kurfürsten-Anlage 36, 69115, Heidelberg, Germany

SUMMARY
Cloud computing is no longer the future but the present. Every day, more and more companies and service providers transfer their businesses and operations to the cloud, benefiting from its multiple advantages. Moreover, the flexibility offered by many cloud services allows to easily build sophisticated services by just composing simpler ones, rather than creating them from zero. Yet, to mitigate potential security threats and keep the maximum performance at any time, a smart selection of those composite services constitutes a key aspect. In this paper, we introduce the reader to the problem of Web service selection based on their reputation scores and subsequently present a survey on some of the most relevant reputation-based Web service orchestration schemes for cloud computing in the literature. For each one of these approaches, a thorough analysis of their pros and cons has been performed, providing a comprehensive comparison amongst all of them leading to the conclusion that, to the best of our knowledge, there is no one single model elegantly fitting to each and every situation that could occur in such a dynamic environment like cloud computing. Finally, we present some current challenges and future research trends in the field of reputation-based service orchestration in cloud computing. Copyright © 2013 John Wiley & Sons, Ltd.

1. INTRODUCTION
Cloud computing has revolutionized the way we understand the Internet and the World Wide Web. Although at the beginning, the intelligence of the whole system resided mainly on the end terminals and the network was merely a transport mechanism to connect and communicate those terminals, nowadays, such intelligence is moving more and more towards the so called ‘cloud’ [1–4].

Thanks to the cloud computing, unprecedent massive computing, processing and storing capabilities can be easily offered to small, medium and large enterprises, as well as to the end users. Hence, many service providers and companies are aiming at cloud computing, transferring their businesses models [5–7] to the cloud, saving thus costs and providing a higher quality of service to their customers.

One of the main advantages of cloud computing consists of the flexibility that many cloud services provide when it is about creating new complex or sophisticated services, just based on simpler ones. There is no longer the need for a developer to build a huge service or cloud application necessarily from scratch. Instead, by appropriately combining and orchestrating the right services [8, 9], a more powerful, functional and even efficient composite service can be easily built.
Yet, to create a successful composite service, such as when building a puzzle, the correct pieces need to be chosen. For one or another reason (failure, malware, exhausted resources, etc), the individual services composing the bigger one might unexpectedly change their behaviour throughout time [10, 11], degrading their performance (and, in turn, the performance of the composite service) or even constituting a security threat or breach. Therefore, it is a wise strategy not to base a composite service on fixed simpler ones, but rather making the latter replaceable whenever needed in order to keep always the maximum performance and level of assurance.

Trust and reputation management has emerged in the last years as a smart and accurate solution to tackle these issues [12–15]. Actually those systems have been widely accepted and deployed, finding them in a multitude of different environments and scenarios [16, 17]. This appealing area of knowledge has captured the attention of both industry and academia, leading to several successful products [13] and patents [18–23].

In this paper, we performed a survey of some of the main research approaches dealing with the problem of reputation-based Web service (WS) orchestration for cloud computing environments. We described each of those solutions, thoroughly analysing their advantages and shortcomings. Moreover, a comprehensive comparative amongst them led us to the conclusion that none of the studied models gracefully fits to each and every situation that could emerge in a highly dynamic environment such as cloud computing. Finally, we also studied some current research challenges, as well as future trends in the field of reputation-based service orchestration in cloud computing.

The remainder of the paper is organized as follows. Section 2 provides a wider background regarding WSs, trust and reputation management and reputation-based WS selection. The description of the main problem addressed by this survey is shown in Section 3, whereas Section 4 actually presents such survey together with a comprehensive comparison of those works. Next, a set of current research challenges, as well as some future trends in this field, are analysed in Section 5. Finally, Section 6 concludes with some final remarks.

2. BACKGROUND

2.1. Web services

We can define a WS as the piece of software that resides in machines and allows them to communicate in a standardized manner without worrying about the networking details. Thus, following this WS paradigm, machines can easily provide and consume services.

This machine-to-machine interaction comes accompanied with a service-oriented architecture [24] concept. A concept that defines how service providers can describe and publish their services in a universal repository, the so-called universal description, discovery and integration (UDDI) [25] component, and how service consumers can look at this repository to select the service that best fits their needs. Figure 1 shows such architecture.

![Service-oriented architecture](image_url)
The service providers can describe their published services using the WS description language (WSDL) [26]. The interaction happens using the standardized simple object access protocol (SOAP) [27] that runs over the Hypertext Transfer Protocol [28]. This application layer protocol, which relies on the extensible markup language (XML) [29], makes possible the exchange of remote procedure calls (RPC) [30] between the machines.

However, we see as a strong characteristic of the previously described architecture that the WS providers might not be necessarily reliable. Although service providers can describe and publish their services using this architecture, a published service might not be necessarily available whenever a service consumer selects it. Moreover, this architecture does not provide any guarantee of the quality of the service being selected. Consequently, there still exist a number of open research questions in this field such as fault handling, fault tolerant, trust and reputation issues about the selected services.

2.2. Trust and reputation management

In turn, trust and reputation management [13, 31–36] has captured the attention of several research groups from both industry and academia in the recent years, leading to a current solid as well as promising research field.

In fact, these novel mechanisms have been applied in a wide variety of scenarios ranging from peer-to-peer (P2P) networks [37–39] to wireless sensor networks [17, 40, 41], mobile ad hoc networks [42–44] or even vehicular networks [45–47], to cite some of them.

A number of different techniques have been applied as well in order to represent, model and, in short, deal with concepts such as trust, reputation, recommendations and feedback. Thus, for instance, several authors relied on fuzzy sets [48, 49] and fuzzy logic [50, 51] in order to develop their models [52–54], whereas others applied Bayesian networks [44, 55, 56], or even bio-inspired algorithms [57–59] (mainly ant colony optimization [60]).

Even network simulators specifically designed to test trust and reputation approaches have been developed [61–63], allowing this way a comprehensive analysis and comparison amongst different proposals [64].

Additionally, some theoretical works haven been performed too, for instance, [65] and [66] where a set of steps to be followed by every trust and reputation model, as well as some recommendation designs, are presented.

Other works, for instance, [14, 67, 68], however, focus on describing and analysing which are those potential security threats that could subvert these specific systems, while proving some recommendations to tackle them too.

In [15], authors analyse the trust establishment process itself, formally describing each element that plays a role in these systems, and therefore providing a way to formally compare different trust and reputation management approaches.

Finally, in order to have a common understanding of the actual meaning of these two concepts (namely ‘trust’ and ‘reputation’), we reproduce next the definition given in [66]:

- **Trust**
  
is a particular level of the subjective probability with which an agent will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action.

- **Reputation**
  
is an expectation about an agent’s behaviour based on information about it or observations of its past behaviour.

2.3. Reputation-based Web service selection

Finally, as a combination of the last two subsections, trust and reputation management has been applied as well with the purpose of aiding a user in the process of selecting the most trustworthy or reputable WS amongst a pool of them [69].
As it can be observed in Figure 2(a), without the assistance of an accurate trust and reputation management, users are ‘blind’ when selecting the most reliable WS to access. Yet, such management performed in an appropriate way might help to find out which WS are actually benevolent and which might have a malicious or fraudulent behaviour, isolating this way the latter.

Thus, for instance, a first approach made in [70] suggests to apply trust and reputation management in order to efficiently filter dishonest ratings when evaluating the QoS of WSs. Authors then proposed to use such certified QoS as the criterion for selecting one or another WS.

Authors of [71] addressed a certainly important although commonly neglected issue regarding trust and reputation management: bootstrapping the system (i.e. dealing with newcomers with lack of historical information). They presented different mechanisms in order to bootstrap the reputation of newcomers in a service-oriented environment in a fair and accurate fashion.

In turn, [72] presented a trust model built upon Bayesian networks aimed to help users to capture the dynamism from not only non-functional QoS properties but also service orchestration in service-oriented environments.

Finally, a survey of trust and reputation-based WS selection is presented in [73]. It proposes a typology to classify those selection mechanisms from three different aspects, namely, centralized versus decentralized, persons/agents versus resources and global versus personalized.

In summary, all these approaches actually constitute the basis and the prelude for the works to be analysed as part of the survey conducted in the paper at hand, which are particularly focused on reputation-based WS orchestration.

Nevertheless, besides the concrete aforementioned reputation-based WS selection process, there are other aspects that need to be addressed as well in order to accomplish the overall reputation-based WS orchestration operation. Although this complementary processes fall out of the scope of this paper, we list next the most relevant ones for the sake of completeness:

- **Performance monitoring.** In order to determine whether a WS currently working as part of the composite WS needs to be replaced or not because of its performance deterioration, a mechanism to actually assess and monitor the current performance of the individual WS is required.

- **Actual service integration and interoperability.** In order to make the set of selected WS actually work all together as a single composite WS, a conglomerate of technologies will need to be taken into play, as mentioned before, such as SOAP, WSDL and UDDI.

- **Decision-making engine for WS selection and replacement.** Taking the performance monitoring mechanism described earlier as an input, there should be a module as well in charge of deciding (1) whether an individual WS has degraded its performance so much that needs to be replaced; and (2) which other individual WS, based on its current reputation score, should replace the misbehaving one.

- **Seamless and secure migration and portability of data.** Whenever a WS needs to be replaced, because of misbehaviour and/or underachievement, a check is required on whether such WS was handling some sort of data that has to be, therefore, migrated to the new selected WS.
to operate. Such migration and portability of the data should occur in a seamless and secure manner, avoiding any type of information leakage.

- Resources allocation. Bearing in mind that we are considering a cloud computing scenario where the WS composing the bigger one might be actually either running in the same machine or distributed across different ones, a phase of resources allocation needs to be considered as well when selecting each one of those individual WS.

2.4. Cloud service orchestration and reputation management

Cloud computing [74] has been extensively studied so far [75], addressing a multitude of challenges as well as identifying a number of open research questions [76] from many different perspectives. More specifically, several authors have recently faced the problem of cloud service orchestration, proposing novel alternatives to solve such issue. Yet, in order to achieve an efficient and effective service orchestration solution within the context of cloud computing environments, it is crucial not to overlook the intrinsic characteristics of the latter, namely, virtualization, multitenancy, elasticity, the service models available (infrastructure as a service, platform as a service, software as a service, etc) and the deployment models available (public, private, community and hybrid). Thus, for instance, in [77] authors relied on a three-layered self-organizing multi-agent system in order to establish a cloud service orchestration framework, whereas authors of [78] extended the existing concept of template-based service orchestration and focus on testing the same workflow of service orchestration. In turn, [79] presented an interesting solution consisting of exploiting the potential of semantic models in supporting service and application linkage by studying links between the complementary services in order to enable the orchestration of cloud services.

Moreover, the security threats inherent to the cloud systems have also captured the attention of a wide number of researchers worldwide [80, 81]. Concretely, reputation management has also been considered as a very effective tool to cope with certain risks jeopardizing these cloud environments. Just to name a few of these works, [82] proposed a hierarchy of P2P reputation systems to protect clouds and data centres at the site level and to safeguard the data objects at the file-access level. Moreover, different security countermeasures are suggested to protect cloud service models: infrastructure as a service, platform as a service and software as a service. In turn, authors of [83] proposed to use reputation management to minimize the cost of computing resources, while satisfying the desired QoS metrics, basing their solution on the Dirichlet multinomial model.

Yet, although the application of reputation management in order to orchestrate composite WS has led to a profuse number of research works, as we will see later in Section 4, the natural evolution of such application to cloud services still remains at an early research stage.

3. PROBLEM STATEMENT

3.1. Reputation-based Web service orchestration

One of the main advantages provided by WSs is that, on the one hand, they can perform very simple tasks, but, on the other hand, those simple WSs can be used to build or compose a more complex WS. And such complex WS can be in turn used as well to create an even more complex WS, and so on. They can be seen as the individual gears of a whole engine, as depicted in Figure 3.

This fringe benefit is of high importance because it allows developers to easily and quickly create new and novel composed WSs on the basis of previous existing ones. Moreover, the WS selection feature described before enables a real time and dynamic choosing of such components. Thus, if the performance of one of those elements is currently downgrading, for instance, it can be replaced on-the-fly by other more reliable one (as we would do with the gears of an engine).

There is a number of works describing in detail the problem of WS orchestration [84, 85]. Thus, for instance, authors of [86] showed a survey of some existing WS orchestration solutions and compared them with respect to four key requirements, namely, connectivity, nonfunctional quality-of-service properties, correctness and scalability. Srivastava and Koehler [87], however, focused on comparing WSs described with WSDL and those ones more oriented to the semantic
Web service orchestration overview.

Figure 3. Web service orchestration overview.

Web and therefore described through resource description format (RDF) expressions. Additionally, they analyse the solutions of each approach to the problems of modelling, composing, executing and verifying WSs.

Nevertheless, the vast and increasing number of potential candidates for each element of a composed WS impels the necessity of robust and scalable mechanisms able to help in the final selection of those WSs. To this end, trust and reputation management has arisen as a very effective and innovative tool in order to assist in an accurately selection of the most reliable composing WSs at any time. And this is specifically the problem this survey will focus on.

3.2. Real scenario: identity federation

In today’s identity federated systems, service level agreements (SLAs) have to be exchanged between the service and identity providers of a federation in order to make the system working. Besides the fact that those configurations are extremely complex, it is open to the service and identity providers to find a solution in case the agreement is not respected. That is to say, usually there is no definition of the procedures to apply in case there is a break in the agreement between the companies.

This can be exemplified by the following scenario, as shown in Figure 4: imagine that the user federates his/her account with a flight company and a hotel service. In such use case, when the user wants to book a flight, he/she goes to the website of the flight company, enters all his/her data (such as name, passport id and credit card) and buys his/her flight ticket. After the ticket has been bought, he/she goes to the federated hotel to book his/her room. Because both companies are federated, the user does not need any more to enter his/her data again in the hotel Web portal. The hotel and flight company are responsible for exchanging the user’s attributes, for example, name and passport id. Therefore, the book reservation should run smoothly and automatic.
However, in order that the transaction of the user’s attributes between hotel and flight company occurs without any problems, the hotel has to rely on the flight company that the service will be running all the time and that the transaction will happen in a reasonable time. Otherwise, the booking of the hotel will fail because the flight company did not deliver the needed information. Thus, the hotel takes the risk of losing a client in case the flight company does not deliver the user’s attribute information. Regarding the flight company, in case the attributes’ exchange does not work, and therefore, a SLA was broken, the project Liberty Alliance does not specify how the relation between both patterns should proceed.

Looking at the Internet, we see that reputation systems are becoming very popular and have been integrated into many websites, being some of the most popular ones Amazon and eBay. The result is well known: every buyer looks at the reputation of the sellers before doing any transaction, and the most reputable sellers are the top sellers. Such system is well known and widely deployed in the Internet.

If we further analyse such reputation system, we can see that its core concept can be applied in a circle of trust of members of a federation in case service levels are not respected between its members. Taking the example of the hotel and flight company, in case that the flight company for any reason does not deliver the user’s attribute information needed by the hotel service, the hotel service could use a reputation system available in the federation and register the fact or even rate the service of the flight company such that other companies of the federation could be notified and informed about the behaviour of the flight company.

Therefore, at first glance it looks promising to have a reputation system integrated in federated circle of trust, to enhance the SLA exchanged between its members. For instance, we could have members of the federation punished in case those agreements are not respected.

4. SURVEY ON REPUTATION-BASED WEB SERVICE ORCHESTRATION

4.1. Malik and Bouguettaya [88]

Contrary to most of the approaches when applying reputation systems into service orchestration, Malik and Bouguettaya proposed a decentralized reputation system for WS orchestrations called RateWeb. This decentralized approach is based on a P2P WS model. WSs can act as consumers and providers of services. The proposal consists not using a centralized trusted third party as the reputation manager. Here, we can see that, with this approach, authors avoid the single point of failure and also the fact that all members have to trust a centralized entity.

Therefore, there is no central entity, which is responsible for collecting, updating and analysing the reputation values of the services. Each member of the system is responsible for this task by storing its own experience values. This allows consumers to apply independently their own aggregation methods or strategy. Hence, there is no global common defined agreed reputation value for a service. Each consumer has its own view of the reputation of the service provider.
Following the decentralized approach, the service registries are only used to locate the service provider, and therefore do not store any reputation value. This is achieved by a community model based on ontologies. This community model acts as a directory of raters. Consumers composing services get information about the raters, and not the ratings themselves. When the consumer decides to get reputation information about a certain service, it asks those raters.

Authors proposed as well in this system that, instead of rating only the provider, the service consumer can also rate how useful was the rating of the other consumers that have had past experiences with the service provider. This allows mitigating the effects of malicious raters in the system like collusion attacks [67].

Malik and Bouguettaya also considered important for WS orchestrations to have a temporal sensitivity system. The idea is to fade the value of old observations, so that the system can consider and therefore give more weight to the newest recommendation values.

This research also shows that the overhead suffered by the system is strongly influenced by the collection method used by the reputation system. Authors analyse three different methods:

- **Publish-subscribe model**: consumers publish in the registry the provider they have interacted with. That means the registry is composed by consumer raters and the services ratings. Therefore, consumers can subscribe to the registry to get a notification if a new consumer provided a new rating for a certain service.

- **Community broadcast model**: each consumer service receives the rating values once subscribed into the community. In this model, we can observe that certainly a lot of traffic is generated and consumers get additional information that they do not really need.

- **Credibility-based model**: it is a proposed collection model where service consumers are divided into groups of credible raters. When a consumer requests a rating, it requests to a certain credible rater group. Therefore, a list of credible rater group is kept by each consumer. If the particular credible rater group cannot provide the answer, the request is forwarded to the next community hop by hop. Here, the consumer can set a limit on the number of hops to be queried.

The result of the experiments and analysis showed that the best performance was achieved by the publish-subscribe model. However, the best collection model will still depend on the domain and type of interactions of the particular deployed system.

4.2. Bianculli et al. [89]

Bianculli et al. proposed to improve the orchestration of services by leveraging monitoring techniques of functional and non-functional properties at the client. They introduce a reputation manager, which is a centralized collector and aggregator of all client-collected information. This reputation manager is also responsible for sharing the monitoring information amongst the clients that are willing to composite services. According to the authors, this enables a pro-active replacement before the service totally misbehaves during the workflow. Hence, they believed that the failure of a system will happen gradually and not abruptly.

The architecture of the system proposes a centralized reputation manager integrated with a UDDI service directory and uses publish and subscribe mechanisms to disseminate the reputation information amongst the subscribed clients. The subscription considers new services that appear in the UDDI as well as services that have their reputation degraded by a factor when a Business Process Execution Language (BPEL) [90] process starts. Clients are responsible for giving a time stamp report containing a feedback value for each transaction. Positive if the service met the SLA agreement, negative otherwise. The architecture is based on the assumption that the services are composed using the BPEL language, and therefore, they proposed to adapt the BPEL engine in the following way:

- **Aspect-oriented programming** is used for subscribing the client each time a BPEL process is started.

- Clients are required to give a time stamp report containing a feedback value for each transaction. Positive if the service met the SLA agreement, negative otherwise.
• Notifications are inserted by using aspect-oriented programming techniques that modifies the BPEL engine by inserting a global notification handler.

The reputation manager is modelled by representing the chain of the composed WSs with a stochastic state machine following the hidden Markov model with two states: a good state and a bad state. The good state is assigned with an unknown probability to the service if client requests are satisfied, and a bad state is assigned otherwise. The system assumes that the probability of the service to go from a bad state to a good state is fixed and known. Therefore, according to this model, the reputation manager can estimate the probability that the requested service will be in a good state, satisfying the client. This estimation is then used to optimize the workflow of the BPEL engine in the future.

This model relies on the fact that clients have to provide trustworthy feedbacks. However, bad reports are good for the clients because they would force bad services to be cheaper in the market place. The system solves this problem by introducing a payment scheme. The reputation manager pays clients for reports only if the client report has the same value as a randomly chosen report provided by another client in the past.

4.3. Conner et al. [91]

Conner et al. designed and implemented a trust management framework on the basis of previous service interactions with clients. Instead of client composers rating services, the approach in this work makes services to rate clients. The reputation manager, responsible for storing and calculating the reputation values, is a centralized entity that can apply different trust evaluation metrics, therefore enabling each service to make trust level assessments with its own individual reputation. To reduce the number of interactions of the services with the centralized trust management framework, they proposed a sophisticated trust evaluation caching mechanism.

A bloom filter storing all the history is available for each service instance. Therefore, the centralized trust management framework can keep track of the clients’ requests and evaluation by sending bloom histograms [92] to the client services that recently requested the trust score evaluation. Although this mechanism adds some complexity to the system, it makes much faster the access to the reputation history. This caching mechanism allows that only after enough transactions, the services will make a request to the trust management framework, therefore reducing the overhead in the network because of the message exchanges. The system also proposes to include optional attributes to provide in this way additional contextual information to the transaction. This additional information is relevant because it could serve as proof that the transaction took place.

Because the trust management framework is a centralized solution, load balancing is therefore used in the system. Each client has his/her own instance of the reputation history. Availability is also achieved using replication techniques.

Because clients can dictate the type of scoring function the trust management framework has to apply for calculating the trust score values, this system considers at the same time three different types of scoring functions: eBay [16, 93, 94], PeerTrust [38] and their own developed exponentially weighted moving average (EWMA). The proposal of the EWMA is to quickly degrade the reputation of the client in case of negative feedback and slowly increase the reputation in case of positive feedbacks. This approach makes it possible to identify the clients that have a continuous good transaction behaviour.

4.4. Paradesi et al. [95]

Paradesi et al. showed how to derive trust for WS orchestrations from trust models of individual services. Their proposal is to model trust between WSs using a stochastic model as proposed by Wang and Sigh [96]. The model uses a second-order probability certainty density function (PCDF) [97]. Contrary to the already described approaches, the system does not treat trust as additional QoS parameter because they claim that the already existing WS orchestration techniques should be adapted in order to apply this approach. Thus, they developed a trust framework, which is capable
of computing the aggregate trust of a service orchestration, and not of single services, treating trust as an additional parameter.

The approach is based on a trust vector that expresses the trust, distrust and uncertainty about having a positive experience with a WS. Therefore, the vector represents the probable certainty to have a positive experience with a WS. This trust vector is generated for each one of the possible orchestrations available that forms a composed service. Authors assume here the existence of WS orchestration algorithms that can build multiple orchestrations satisfying the same functional requirements.

The authors considered in the study four types of basic flows of services: sequence flow, concurrent flow, conditional flow and loops. They show that, for each of these basic flows, it is possible to derive the belief density function. Consequently, it is possible to calculate the probability of a positive experience with the user over the total orchestration. The method is analogous to calculating the aggregate QoS parameters of an orchestration in [98].

On the basis of this mathematical model, the authors proposed a framework for considering trust into WS orchestrations, which is able to select the most trustworthy orchestration for deployment. The novelty here is that the framework also considers the feedback of the users of the individual WSs that took part in the final orchestration. Users can indicate their experiences with the service components that were triggered during the WS orchestration chain.

Summarizing, the framework bases its reputation mechanism by integrating the subjective feedback of the users as well the objective measurements of the composed services. The authors argue that solely relying on user feedback does not take into account the subjective fact that what appears to somebody might not appear to another one. Users have different perceptions of a provided service, and this is especially difficult to model.

Moreover, such systems relying only on user feedback might suffer from negative or positive feedback on purpose. Therefore, correlating objective measurements with subjective feedback in the orchestrations allows the system to mitigate those drawbacks coming only from the subjective feedback. The model also allows modelling the certainty about trusting a WS, even though no prior experience may exist, which constitutes a considerable step forward in the integration of reputation systems in WS orchestrations.

4. Wang et al. [99]

According to Wang et al., QoS may serve as a key benchmark in order to discern differences amongst alternatives in the process of selecting WSs in open and distributed environments. To this end, traditional QoS entails a whole range of definitions such as response time, accessibility, availability or reliability. Yet, they consider that, in an open Internet environment, it is also necessary to objectively link service quality with the users’ subjective perceptions.

Moreover, they state that current research regarding modelling and evaluating trust within the context of WSs does not accurately and rationally reflect some essential characteristics of trust such as subjective uncertainty and dynamism. Therefore, they proposed a trust model on the basis of cloud model theory in order to model the subjective uncertainty of trust factors. Regarding the dynamism aspect, it is expressed by means of a time-related backward cloud generation algorithm. Furthermore, according to the trust model and algorithm, a formalized calculation approach is provided to evaluate the trust degree of service requestors in providers.

Thus, the cloud model can show the uncertain mechanism during the transformation between qualitative concepts and quantitative values. This characteristic of the cloud model makes it suitable to express the subjective uncertainty during the perception of trust degree. Formally, let $U$ be a universal set described by precise numbers and $C$ be the qualitative concept related to $U$. If there is a number $x \in U$, which randomly realizes the concept $C$, and the certainty degree of $x$ for $C$, that is, $\mu(x) \in [0, 1]$, is a random value with stabilization tendency $\mu : U \rightarrow [0, 1]$ $\forall x \in U$ $x \rightarrow \mu(x)$, then the distribution of $x$ on $U$ is defined as a cloud, and every $x$ is defined as a cloud drop.

In this direction, reputation cloud $RepC(x)$ (reputation of services, reflecting the satisfaction degree of the end-user), trustworthiness cloud $TWC(x)$ (expressing whether the value of an SLA parameter was greater or less than the expected value) and risk cloud $RiskC(x)$ (the risk that
requestors face in future interaction by means of the change state of reputation or capability of services in adjacent time slots) are defined.

Regarding the time-related backward cloud generation algorithm, used to express the dynamism aspect of trust, authors designed a weighing mechanism that, on the basis of the distance from historical time to the current trust decision time, assigns different weights to drop values of $RepC$, $TWC$ and $RiskC$. The basic weighting rule of this algorithm is that the newer the drop value is, the bigger its weight and vice versa.

Finally, through reputation, trustworthiness and risk, the trust degree of requestors in providers can be estimated. In general, the trust degree increases with reputation and trustworthiness, whereas risk has a negative effect on the trust decision of requestors. Authors proposed the following equation for computing this trust degree:

$$TS = \frac{RepC + TWC - e^{k \times RiskC(RepC)} - e^{k \times RiskC(TWC)}}{2 \times e^k} + 1$$

Such formula satisfies the following constraints:

1. $TS$ increases or decreases with $RepC$ and $TWC$ when the other variable is fixed.
2. It is a decreasing function of $RiskC(RepC)$ and $RiskC(TWC)$.
3. When $RepC$ and $TWC$ are equal to 1, and $RiskC(RepC)$ and $RiskC(TWC)$ are equal to 0, the trust degree reaches its maximal value, and is equal to 1.
4. In contrast to the previous point, when $RepC$ and $TWC$ are equal to 0, and $RiskC(RepC)$ and $RiskC(TWC)$ are equal to 1, the trust degree meets its minimal value, and it is equal to 0.
5. $k$ parameter could be used so that the trust score function can be adjusted to control the effect of risk on trust-decision making.

4.6. Hwang et al. [100]

Going beyond the simple usual service oriented paradigm in WS orchestration, [100] proposed a hierarchy of P2P reputation to protect data centres at the site level, while safeguarding data objects at the file-access level in the cloud ecosystem.

In the field of cloud computing, they proposed a new security-aware cloud architecture, where one of the key components is a proposed hierarchical reputation system of resources sites from data centres to distributed file systems. The hierarchical reputation system is responsible for controlling the data centre and the distributed file systems access level.

The solution is based on a reputation-based trust overlay network layer infrastructure for data aggregation using an hierarchy of distributed hash table (DHT)-based P2P reputation systems. The reputation system is responsible for protecting the cloud resources and the site level, for data centres, and data objects at the file level.

Thus, this research goes one step further providing a centralized reputation system for WS orchestrations. It shows that the next step in this topic is to apply a distributed reputation framework over cloud services through a trust overlay network.

4.7. Kim and Doh [101]

This work addresses the challenges associated to providing quality guaranteed services, while, at the same time, looking at the trust of this quality, in service orchestrations. The authors proposed a WS trust-based quality model in which mediators, through non-functional properties, evaluate services and assign trust ratings to them. Regarding trust, the research assumes that clients or even a credible trust authority might perform the trust assignment to the services after their execution.

Because WSs can be composed in different ways depending on the business model being executed, and therefore have multiple different paths, the model uses the trust ratings to select optimal services based on different paths of the composite WS execution. The trust model is based on inference rules for the sequential, parallel service orchestration.
The WS quality model is based on the following three non-functional properties inherent to all WSs: response time, reliability and availability. For an execution plan $P_1$, aggregating functions are defined for computing an aggregated value of the three non-functional properties in the composite execution. The optimal execution plan is computed using the Euclidean norm over the three non-functional properties, as shown next:

$$||M_Q(P_1)|| = \sqrt{Q_{time}(P_1)^2 + Q_{availability}(P_1)^2 + Q_{reliability}(P_1)^2}$$

Thereafter, a profit function is defined as follows:

$$profit(P_1) = weight_{trust} * ||M_Q(P_1)||$$

where the value of $weight_{trust}$ is the trust rate of the execution path plan.

The mediator of the model calculates the maximum value of this profit function and thus defines the optimal execution path plan of the service, taking as well into account the weighted trust aggregated in the services.

4.8. Cheikh et al. [102]

The main contribution of the work of Cheikh et al. work consists of developing a formal technique aimed to perform automatic orchestration of WSs when available services are nondeterministic and impose security constraints such as access control and authorization ones, as well as reputation constraints on other component services. In particular, access and authorization control is based on credentials, component services may (or may not) trust the credentials issued by other component services and the service behaviour is modelled by the possible conversations the service can have with its clients. The technique that authors proposed is based on reduction to satisfiability in propositional dynamic logic (PDL), which shares the same basic algorithms behind the success of the description logic-based reasoning systems used for Web ontology language (OWL).

In their work, authors defined a community $S$ as a finite set of available services $\{S_1, \ldots, S_n\}$ that share the same set of shared actions $A$. Such actions are the actions available to the agent that is a client of the community. The client can use such actions to specify a behaviour of its interest, the so-called target service $S_0$. The community will try to realize the target service $S_0$ by suitably orchestrating the available services $\{S_1, \ldots, S_n\}$.

They also consider a reputation matrix $Rep$, which has as rows available services and as columns available services and possibly third parties. The cell $Rep(i, j)$ therefore represents the reputation level (set of all possible levels is finite) that the available service $S_i$ has on the available service $S_j$ or on the third party $P_{j-n}$.

In addition, a client has a set of credentials that allows him to execute various parts of an available service. Such credentials, $C = \{c_1, \ldots, c_m\}$, are the mean to establish trust between a client and the service provider and they represent assertions about the client, issued by a given party. Each $c_h$ is a pair of variables ($Attr, Issuer$) where $Attr \in \Delta$ contains the credential itself (an X.509 certificate, for instance) and $Issuer \in I$ represents the issuer of such credential.

Conditions on a credential specify the security requirements of the available services. Let $S_i$ be an available service and $c_h : (Attr, Issuer)$ be a credential. Then an atomic credential condition in $S_i$ on $c_h$ is an expression of the form $T \odot v$, where $T$ is either $c_h.Attr$ or $Rep(i, c_h.Issuer)$, $\odot$ is a comparison operator and $v \in \Delta$.

Each available service $S_i$ is defined in terms of a finite transition system of the form: $TS_i = (S_i, s_{i0}, G_i, \delta_i, F_i)$, where

- $S_i$ is a finite set of states;
- $s_{i0} \in S_i$ is the single initial state of the service;
- $G_i$ is a set of guards, which are closed first-order logic (FOL) formulas built using as atoms the credential conditions;
- $\delta_i$ is the service transition relation;
- $F_i \subseteq S_i$ is the set of final states of the service.
Thus, the client specifies the service to be provided by the community, called target service, in terms of a finite transition system $T S_0 = (S_0, s_{00}, G_0, \delta_0, F_0)$. To the client, it is also associated an initial assignment $CA_{init}$ of the attribute and issuer variables of the credentials in $C$, describing the value of credentials initially assigned to the agent executing the target service and who issued them.

Finally, an orchestrator programme $OP$ is defined, which is in charge of realizing the target service $S_0$ for available services $\{S_1, \ldots, S_n\}$ in the community, given the previous reputation matrix $Rep$ and the initial credential assignment $CA_{init}$.

4.9. Malik and Medjahed [103]

In this article, Malik and Medjahed introduced the idea of propagating reputation among the services involved in a service orchestration. The orchestrator is the component responsible for forwarding reputation information to the composed services. We believe their assumption is relevant because some services will be only accessible in the context of service orchestrations.

The main assumption and motivation for propagating the reputation information is that the invoker of the composed operation, for example, an operation that is subdivided in a workflow in subservices, cannot know which services were responsible for the bad performance of the invoked service, and blaming all the service components equally is not a good approach in terms of fairness. Moreover, if a service has a defect, it will continue to behave in the same way until the client of this service degrades his reputation.

Therefore, a service orchestrator, the entity responsible for orchestrating the services, and at the same time offering this composed service, should transfer part of the blame, on the basis of the cloud model, to one of the services contained in the orchestration in case the reputation degrades below a certain threshold. If not all the blame has to be transferred, at least part of it. We can also think of reputation punishment system to the services, because if they do not fix their behaviour, the system can also increase the amount of blame to be transferred.

The motivation for using the cloud model for propagating the reputation information is twofold. First, the concept of fuzzy memberships is not sufficient for representing uncertainty and imprecision inherent to trust systems, especially when dealing with reputation propagation. Second, the authors argued that other models like the Bayesian systems, which take binary rating as input as proposed in [36, 95, 104], are far too complex to implement, and additionally do not present a fuzziness component. The cloud model [2] incorporates the missing uncertainty and fuzziness of trust.

The work showed the results of some experiments. The main result was that it demonstrated that once a service receives a bad reputation from the orchestrator, it fixes its behaviour rapidly. This has the effect of fast behaviour correction of the orchestrated services. This fact improves a lot the orchestrated service when compared with the state of the art solutions.

4.10. Bianculli et al. [105]

In this work, authors presented a customizable and generic reputation framework aimed to automatically and transparently monitor the execution of composite services, considering both functional and non-functional properties. The experienced WS QoS is delivered to a configurable reputation mechanism, which is in charge of publishing service rankings. Their reputation mechanism supports as well notifications upon changes in service reputation, enabling this way self-tuning and self-healing properties in the execution of composite services.

At the server side, the architecture of the reputation infrastructure comprises three main components:

- **Enhanced registry.** It is a UDDI-compliant registry extended with the functionality to query for QoS estimations of registered services.
- **Reputation manager.** It provides functionalities to manage the services registered for reputation and to estimate their QoS. It exposes a public message queue where service clients may post their feedback reports. It is also in charge of managing reputation policies, which is an abstraction for an algorithm that estimates service reputation.
• **Subscription manager.** It provides functionalities to notify service consumers when reputation-related events occur and to manage the subscriptions to these events. The reputation infrastructure supports two event types: reputation decrease and availability of a service with better reputation.

On the other hand, at the client side, the architecture comprises three components:

• **Monitor.** It monitors the behaviour of external services used by a BPEL service client, by checking some functional and non-functional assertions.

• **Reputation feeder.** It provides methods to collect feedback reports and to send them to the reputation manager in the server side.

• **Event manager.** It provides functionalities to subscribe to reputation-related events and to react to such notifications.

The reputation infrastructure has been designed in an open and extensible way, so as to support different methods for computing service reputation through the installation of new reputation policies provided as plugins.

The default reputation policy plugin works as follows. Let \( S \) be the set of services published in the reputation infrastructure, \( F \) be the set of feedbacks \( f \), where each \( f \) consists of a tuple \(< s, v, t >\), with \( s \in S \) being the service that is the object of the feedback, \( v \in \{0, 1\} \) the value of the feedback and \( t \in \mathbb{R}^+ \) the time-stamp at which the feedback is received at the server. Let \( F_{s,t} = \{ f \in F | f.s = s \land f.t \leq t \}, s \in S, t \in \mathbb{R}^+ \), be the set of the feedbacks received for a service \( s \) until time \( t \); let \( P(s,t) = \sum_{f \in F_{s,t}} f.v \) be the amount of endorsement received for service \( s \) until time \( t \), and \( N(s,t) = |F_{s,t}| \) be the number of total feedbacks received for a service \( s \) until time \( t \). The reputation \( r(s,t) \) for a service \( s \) at instant \( t \) is then computed using the endorsement-refusals ratio as follows:

\[
    r(s,t) = \frac{P(s,t)}{N(s,t)}
\]

### 4.11. Mokarizadeh et al. [106]

Mokarizadeh et al. presented in [106] a framework for selecting WSs for service orchestration using the trustworthiness of the former as a measure of their quality. Such trustworthiness, in turn, is computed using the reputation scores provided by different users’ profiles, and such profiles are extracted from social networks storing the previous experiences of those users with the services. Moreover, they apply a privacy inference model to protect sensitive information in the social network, when analysing its content in order to extract the corresponding reputation values.

Authors applied a T-index\(^2\) approach [107] in order to compute the inferred trust value of user \( u_1 \) towards user \( u_2 \), denoted by \( trust(u_1, u_2) \). Thus, being \( u_1 \) and \( u_2 \) two individuals in the target social network, and \( p_{u_1} \in \{0.0, 0.1, \ldots, 0.9, 1.0\} \) the privacy value given to profile of individual \( u_1 \), then the inferred privacy rating of \( u_1 \) from the perspective of \( u_2 \) can be computed as follows:

\[
    privacy(u_1, u_2) = \begin{cases} 
    \alpha(1 - trust(u_1, u_2)) + \beta p_{u_1} & \text{if } trust(u_1, u_2) \geq Min_tr \\
    \gamma(1 - trust(u_1, u_2)) + p_{u_1} & \text{otherwise}
    \end{cases}
\]

where \( \gamma \in (0, 1), \alpha, \beta \in [0, 1] \) and \( \alpha + \beta = 1 \). \( Min_tr \) denotes the trust threshold for considering user \( u_2 \) as trusted individual. As it can be observed, authors consider privacy as an inverse function of trust towards the individuals for whom privacy assertion is issued, so less trusted nodes are ignored by shrinking their visibility with a strength based on \( \gamma \) parameter. In contrast, the model can be generous towards highly trusted nodes by enforcing \( \alpha \gg \beta \) as a constraint on the weights. In this case, highly trusted nodes are rewarded by decreasing the privacy level they face to access the content.

---

\(^2\)Similar to \( H \)-index to show the number of trust relationships between a user and its trusters with trust value higher or equal to \( T \).
Regarding the WS trustworthiness computation model, authors adopted the approach of Kuter and Golbeck \[108\] as follows:

\[
t_c(s) = \frac{\sum_{u \in U} r_u(s) \cdot \text{trust}(c, u)}{|U|}
\]

where \(t_c(s)\) denotes the trustworthiness of service \(s\) from the point of view of user \(c\), \(U\) is the set of users and \(r_u(s)\) represents the rating of user \(u\) about service \(s\).

Finally, as for composed services, their trustworthiness is calculated via propagating the trust values of atomic services, upward in the orchestration, computed using the last formula. Such propagation can be performed by means of three different strategies, namely, overly cautious, overly optimistic and average. The first one avoids incorporating low trusted services, whereas the second one, overly optimistic, promotes the influence of highly trusted atomic services into trust of the composite service. The last strategy, however, looks for orchestrations with maximum average trust.

4.12. Bansal et al. \[109\]

In this work, Bansal et al. presented a framework for a trust-based dynamic WS orchestration taking into consideration the following elements:

- Functional attributes described in the WS description document;
- The nonfunctional attributes described as SLA attributes in a WS agreement document; and
- Trust rating of the service provider computed from a popular and widely accepted Web-based social network.

For them, a service is described as a sixtuple \((CI, I, A, AO, O, CO)\) of its preconditions, inputs, side effect, affected object, outputs and post-conditions. In turn, a service orchestration is modelled as a directed acyclic graph \(G = (V, E)\) of services from repository \(R\), given query \(Q = (CI', I', A', AO', O', CO')\), and being \(V\) the set of vertices and \(E\) the set of edges of the graph.

Moreover, each vertex \(v_i\) in the graph represents either a service \(s_i\) involved in the orchestration or postcondition of the immediate predecessor service in the graph, whose outcome can be determined only after the execution of the service. On the other hand, each outgoing edge of a node (service) represents the outputs and post-conditions produced by such service, whereas each incoming edge represents the inputs and preconditions of the service.

Regarding the trust rating for each service in the repository \(R\), it is computed as a measure of the centrality degree \((C_D)\) of the social network to which the service provider belongs. It is calculated as the degree or count of the number of adjacencies for a node \(s_k\) as follows:

\[
C_D(s_k) = \sum_{i=0}^{n} a(s_i, s_k)
\]

where \(a(s_i, s_k) = 1\) iff \(s_i\) and \(s_k\) are directly connected by an edge in the graph \(G\), and 0 otherwise.

Thus, it can be considered as a straightforward index of the extent to which \(s_k\) represents a focus of activity. Trust rating of the entire composite service is therefore computed as an average of the individual trust ratings of each of the services involved in the orchestration. Furthermore, authors define a trust threshold so that any service with a trust rating below such threshold will not be used while generating orchestration solutions.

This solution applies SLAs of the individual services involved in the orchestration in order to determine the final SLA of the obtained composite service. Let \(SLA_{s_j}\) be the SLA of a WS \(s_j\). Then \(SLA_{s_j}\) can be represented as a tuple shown in the succeeding text:

\[
SLA_{s_j} = (Up_{s_j}, SRat_{s_j}, SResp_{s_j}, Cost_{s_j}, PreNT_{s_j}, RenegT_{s_j}, Rel_{s_j}, T_{s_j})
\]
where $U_{ps_i}$ is the uptime of the service specified by the agreement (guarantee by the service providers that their services will be available a specified percentage of the time per day or month), $SRate_{ps_i}$ is the allowable service rate, $SResp_{ps_i}$ is the allowable service response time (the time it takes to complete the process by adding the response times of each service in the orchestration), $Cost_{ps_i}$ is the subscription cost or price of the service (sum total price of all services participating in the solution process), $PreNT_{ps_i}$ is the maintenance pre-notification time, $RenegT_{ps_i}$ is the expiration/renegotiation time of the agreement, $Rel_{ps_i}$ is the reliability rating of the service and $T_{ps_i}$ is the trust rating of the service provider. Finally, the SLA for a workflow orchestration or a composite WS is obtained by composing the set of SLAs of all the services participating in the orchestration.

### 4.13. Summary

After studying, describing and analysing a number of solutions advocated to the reputation-based orchestration of WSs, this section presents an overall comparison of all those relevant proposals. To this end, we have selected the following criteria, because of their suitability to the context of cloud computing scenarios:

- **Scalability.** In such an open environment as the cloud, we cannot neglect the fact that the system might suddenly and unexpectedly shrink and grow, having a harmful impact in the performance of the reputation management mechanism applied at each moment. Thus, a good trust and reputation model should be able to dynamically and efficiently adapt to the changes produced in the system in terms of number of WSs, clients, providers, resources allocation and so on.
- **Overhead.** Like any other new system that requires network communication and processing capacity, the overhead introduced by each reputation approach both in terms of computation, data storage and communication should be kept to the minimum in order to achieve an efficient and light solution. This design feature in turn fosters the deployment and acceptance of each proposal in a real scenario.
- **Target services.** We would like to distinguish between those reputation models mainly focused on WSs and those focused on cloud services (which, in our opinion, are the natural evolution of the former). There might be even the case of some proposals, which could be considered as a transition between the former and the latter. Yet, the intrinsic features of one or another type might hinder the adaptation of a specific reputation model into a different environment.
- **Applied technologies.** The application of standard and existing technologies incentivizes the acceptance of new solutions or proposals. To this end, we consider interesting as well to analyse which existing protocols or standards are in use by each one of the studied trust and reputation models.
- **Security flaws.** Finally, there are several security threats that cannot be underestimated, because its success might, in some cases, generate quite harmful effects and even subvert the whole system. Therefore, it is important to study the resilience of each surveyed trust and reputation model with regards to some of the most important threats in this field [67].

Table I shows a summary of the surveyed works on reputation-based WS orchestration, as well as an assessment of the behaviour and/or performance against each one of the aforementioned comparison criteria. As we can observe, there is actually no optimal trust and reputation model that could be ideally applied everywhere and every time. Instead, each model has its pros and cons, and therefore, a careful analysis of the properties, features and conditions of the system or environment where to apply any of these reputation management techniques is required. Upon such comprehensive analysis, the most suitable model for each system could be then accurately determined. Moreover, the high dynamism of cloud computing scenarios suggests that a dynamic and smart selection of the most suitable reputation model from a pool of pre-defined ones, based on the

---

1Such assessment is not based on experiments or simulations, but rather on the understanding of each of these models and observation of their behaviour and properties.
<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Scalability</th>
<th>Overhead</th>
<th>Target services</th>
<th>Applied technologies</th>
<th>Security flaws</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malik and Bouguettaya</td>
<td>↑</td>
<td>~</td>
<td>Web services</td>
<td>Ontologies</td>
<td>Sybil attack</td>
</tr>
<tr>
<td>Bianculli et al. [89]</td>
<td>↓</td>
<td>~</td>
<td>Web services</td>
<td>P2P, BPEL, UDDI, SLA</td>
<td>Partially malicious collectives, Malicious spies, Sybil attack</td>
</tr>
<tr>
<td>Conner et al. [91]</td>
<td>~</td>
<td>↓</td>
<td>Web services</td>
<td>Bloom histograms, PeerTrust</td>
<td>Sybil attack, Partially malicious collectives, Malicious spies</td>
</tr>
<tr>
<td>Paradesi et al. [95]</td>
<td>~</td>
<td>~</td>
<td>Web services</td>
<td>PCDF</td>
<td>Sybil attack, Partially malicious collectives, Malicious spies</td>
</tr>
<tr>
<td>Wang et al. [99]</td>
<td>↓</td>
<td>↑</td>
<td>Web services</td>
<td>SLA</td>
<td>Sybil attack, Partially malicious collectives, Malicious spies, Sybil attack</td>
</tr>
<tr>
<td>Hwang et al. [100]</td>
<td>↑</td>
<td>~</td>
<td>Cloud services</td>
<td>P2P, DHT</td>
<td>Partially malicious collectives, Malicious spies, Sybil attack</td>
</tr>
<tr>
<td>Kim and Doh [101]</td>
<td>↓</td>
<td>↑</td>
<td>Web services</td>
<td>WSDL, UDDI</td>
<td>Partially malicious collectives, Malicious spies, Sybil attack</td>
</tr>
<tr>
<td>Cheikh et al. [102]</td>
<td>↓</td>
<td>↑↑</td>
<td>Web services</td>
<td>PDL, OWL</td>
<td>Partially malicious collectives, Malicious spies, Sybil attack</td>
</tr>
<tr>
<td>Malik and Medjahed [103]</td>
<td>↓</td>
<td>↑</td>
<td>Web services</td>
<td>WSDL, UDDI, SOAP, BPEL</td>
<td>Partially malicious collectives, Malicious spies, Sybil attack</td>
</tr>
<tr>
<td>Bianculli et al. [105]</td>
<td>↓</td>
<td>~</td>
<td>Web services</td>
<td>BPEL</td>
<td>Malicious collectives, Malicious spies, Sybil attack</td>
</tr>
<tr>
<td>Mokarizadeh et al. [106]</td>
<td>~</td>
<td>~</td>
<td>Web services</td>
<td>H-index</td>
<td>Malicious collectives, Malicious spies, Sybil attack</td>
</tr>
<tr>
<td>Bansal et al. [109]</td>
<td>↓</td>
<td>↓</td>
<td>Web services</td>
<td>SLA</td>
<td>Malicious collectives, Malicious spies, Sybil attack</td>
</tr>
</tbody>
</table>

Legend: ↑↑, very high; ↑, high; ~, medium; ↓, low; ↓↓, very low.
P2P, peer to peer; BPEL, business process execution language; UDDI, universal description, discovery and integration; SLA, service level agreement; PCDF, probability certainty density function; DHT, distributed hash table; WSDL, Web service definition language; PDL, propositional dynamic logic; OWL, Web ontology language; SOAP, simple object access protocol.
current system conditions, would fit very well in here, leading to a remarkable overall performance of the service orchestration.

5. CHALLENGES AND TRENDS

Once we have analysed several models and despite the considerable number of works we found focused on this topic, we must conclude that most of them are still at a preliminary stage. Hence, there are some yet unresolved challenges as well as future trends that we would like to depict in this section.

Regarding the aforementioned challenges, many authors focus on theoretical models, neglecting the implications that a real deployment of their proposals would imply. Thus, next we present some of the limitations or deficiencies that we detected on the analysed models, which in our opinion should be seriously tackled in order to have a real successful acceptance of these solutions:

- **Bootstrapping the system.** Most of the authors focus on describing the algorithm or mathematical formula applied to compute the reputation and/or trust score, obviating the bootstrapping of the system [71]. A negligent initial assignation of reputation scores for newcomers might subvert the whole system, causing harmful damages and making the system vulnerable to certain threats, like the Sybil attack [110], for instance.

- **Recommendations dissemination.** Another commonly neglected factor is the description of how to actually gather or collect reputation information (i.e. recommendations or opinions) from other members within the community. Many authors just assume the data are available, without explaining where to collect it from, how to do it or which might be the implications in terms of communication or storage overhead introduced in the system.

- **Scalability issues.** In reputation management systems, there is always a trade-off between the amount of data considered to calculate a reputation score and the accuracy of such computed value. Usually, the more information taken into account, the more accurate the result will be. However, and specially in centralized solutions, this might end up with some scalability problems, which might constitute an important degradation in the overall performance of the system.

- **Ontology models.** A rigorous definition of an ontology describing the common concepts handled in a reputation management system would drastically increase the interoperability amongst different solutions. As an example, an orchestration language like BPEL [90] could be enhanced with an ontology model in order to better express the preferences and requirements of the composite WSs.

- **Web service orchestration.** As shown in Table I, most of the analysed models are mainly focused on WS selection rather than pure WS orchestration. These are two closely related by still different issues with different requirements. We therefore consider that there is still work to do and room for improvement in the specific field of reputation-based WS orchestration.

With regards to the future research trends in this area of knowledge, we observe there has been a considerable work in the domain of WSs, whereas pure cloud computing scenarios are beginning to receive the deserved attention now. In our opinion, there is a vast set of yet unexplored scenarios within this field where investing research efforts is for sure a wise move.

In fact, a number of international research projects and research groups are currently addressing the security flaws, which are hindering the wide use, acceptance and deployment of cloud computing technologies. In our opinion, such success will not be completed, whereas those deficiencies remain unresolved.

6. CONCLUSIONS AND FUTURE WORK

Cloud computing emerged in the last few years as a revolutionary concept easing the provision of vast amounts of processing, computing and storing capabilities to both enterprises and end
users. This phenomenon has transformed the way many companies run their businesses, switching from traditional scenarios to cloud-based ones, benefiting this way from its multiple advantages and opportunities.

More specifically, the development of sophisticated cloud composite services, on the basis of the orchestration of simpler ones, has become a prominent aspect of the cloud computing, because of the flexibility and immediacy it offers for a rapid creation and deployment of new services in the cloud.

However, such composite services need to constantly meet certain criteria regarding performance and security. And to fulfill such requirements, the reliability and trustworthiness of the composing services must be permanently guaranteed. To this end, trust and reputation management constitutes an excellent solution to tackle this issue.

Thus, in this work, we have analysed some of the main current solutions dealing with reputation-based WS orchestration techniques oriented to cloud computing. We described each one of those solutions in detail, providing an additional comprehensive comparison amongst them. Moreover, some present research challenges within this field, and some future trends have also been studied.

As a final remark, we noticed the wide attention received so far by WSs, whereas the cloud services and scenarios are now starting to gain the deserved consideration and recognition. There is still a number of unaddressed security issues in the field of cloud computing, making this an appealing research topic.

ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPEL</td>
<td>Business process execution language</td>
</tr>
<tr>
<td>DHT</td>
<td>Distributed hash table</td>
</tr>
<tr>
<td>EWMA</td>
<td>Exponentially weighted moving average</td>
</tr>
<tr>
<td>FOL</td>
<td>First-order logic</td>
</tr>
<tr>
<td>OWL</td>
<td>Web ontology language</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer to peer</td>
</tr>
<tr>
<td>PCDF</td>
<td>Probability certainty density function</td>
</tr>
<tr>
<td>PDL</td>
<td>Propositional dynamic logic</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of service</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource description format</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote procedure call</td>
</tr>
<tr>
<td>SLA</td>
<td>Service level agreement</td>
</tr>
<tr>
<td>SOA</td>
<td>Service-oriented architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple object access protocol</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal description, discovery and integration</td>
</tr>
<tr>
<td>WS</td>
<td>Web service</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web service definition language</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible markup language</td>
</tr>
</tbody>
</table>

REFERENCES


