Maintaining Trustworthiness of Service Compositions

Zaki Malik
Department of Computer Science
Wayne State University
Detroit, MI. 48202, USA
zaki@wayne.edu

Brahim Medjahed
Department of Computer Science
University of Michigan
Dearborn, MI. 48120, USA
brahim@umd.umich.edu

ABSTRACT

We present a model for managing the trustworthiness of Web services involved in service compositions. We introduce the propagation of reputation information throughout the composition to aid all the services involved, in making informed decisions regarding the selection of their respective component services. In decreasing/increasing service reputations, our aim is to ensure that no service is wrongfully blamed. Our model is based on the "statistical cloud model" defined for uncertain situations. The model can uniformly describe the concepts of randomness, fuzziness, and their relationship in quantitative terms. Experiment results show the applicability of the proposed model in selecting and maintaining an optimal composition of services.

1. INTRODUCTION

High volatility of demand and constant change in user needs have prompted providers to undertake new forms of collaborations in which the providers are not tied down to a fixed list of suppliers. This new paradigm allows providers to form "dynamic partnerships with suppliers" (known as *virtual enterprises*) according to the particular needs of a customer, and disband the partnership after customer demands are met. With the introduction of *Web services*, applications can now be automatically invoked by other Web clients. A Web service is a self-describing software application that can be advertised, located, and used across the Web using a set of standards (such as WSDL, UDDI, and SOAP) [9].

It is expected that future Web enterprises would exhibit a loose coupling of smaller applications offered by autonomous providers [7][9]. A primary goal of the Web services technology is therefore enabling the use of Web services as independent components in Web enterprises, that are automatically (i.e., without human intervention) formed as a result of consumer demand and which may dissolve post demand-completion [7]. The component service responsible for "forming" the VE, i.e., invoking and integrating other services to fulfill a certain task, is known as the orchestrator.

FIT'10, December 21-23, 2010, Islamabad, Pakistan.

As Web services proliferate, several services may provide the functionalities requested by an orchestrator. Naturally, the orchestrator should select a Web service that provides the "best" service in comparison with other candidates. However, the selection process is not straightforward and is not based only on how or what the service offers. Component services may make promises about the provided service and its associated quality but may fail partially or fully to deliver on those promises. Thus, the challenge lies in providing a framework for enabling the selection and composition of Web services based on *trust* [6].

Research results show that reliable *reputation* systems increase users' trust in the Web [10]. We also anticipate that reputation will play a key role in the selection of component services, as the higher the reputation of a service provider, the *better* the service (high expectance for performance and delivery) [5]. The accurate management of component reputations is therefore a requirement for the optimal working of a composition (e.g., a VE). In this paper, we study the relationship between the reputation of a VE (the composed service) and its components. We attempt to provide an accurate assessment of the *real* culprit (i.e., the component service that performed badly in the overall composition of services albeit the fact that its own direct reputation is considered satisfactory by the orchestrator). In doing so, we also aim to ensure that no service is wrongfully blamed and persecuted.

In recent years, research regarding reputation-based trust has been primarily rooted in probability theory, evidence/ belief models, or fuzzy logic. Probability based models usually do not consider the element of fuzziness in building trust [1] [12]. Since the reasoning is done in a purely statistical manner, they tend over-formalize trust's subjectiveness. For example, Bayesian systems take binary ratings as input and assess trust through updating of the beta probability density function [13] [11]. This process is fairly complex to comprehend and implement, and loses the component of fuzziness inherent in trust assessment. Models based on evidence and belief theory exhibit similar characteristics with added complexity [12]. On the other hand, fuzzy logic based systems use precise set memberships for defining fuzziness of subjective trust. However, these solutions fail to consider the randomness and uncertainty of membership in those fuzzy sets [2] [8]. We propose a solution that incorporates uncertainty and fuzziness of trust to provide a more unified and holistic assessment. Our model employs the statistical

cloud model which defines a way for modeling the transition between a linguistic term of a qualitative concept and its quantitative representation under uncertain and fuzzy conditions.

The paper is organized as follows. In Section 2, we motivate the problem with the help of a scenario. In Section 3, we present our reputation propagation model to maintain optimal trust in service compositions, along with some background of our chosen approach. Section 4 provides experiment results, and verifies the applicability of our proposed model, while Section 6 concludes the paper.

2. REPUTATION-BASED SERVICE INTERACTIONS

In traditional Web service models, service selection is not trust-based. Web service consumers discover a list of providers that can provide the required functionality through service registries, select a provider arbitrarily, and then invoke, i.e. communicate with the provider, expecting to retrieve the desired results. However, in our model we use service providers' reputations as the fundamental selection criteria. Service consumers gather feedbacks of providers from their peers, assimilate this information and derive corresponding provider reputations, and sort the providers according to the assessed reputations. The higher the reputation of a service provider, the better the service is expected to behave. Service consumers then invoke the best available Web service, and at the end of the interaction, service consumers rate the providers according to pre-determined criteria. These service ratings are used to compute provider reputations accordingly.

Each service consumer records its own perceptions of the reputation of only the services it actually invokes. This perception is called personal evaluation. For each service s_j that it has invoked, a service consumer t_x maintains a p-element vector $PerEval_j^x$ representing t_x's perception of s_j's behavior. Different strategies may be adopted in updating $PerEval_j^x$. A simple one may be a *per-invocation* update. Upon an invocation of service s_j, the delivered quality $QRef_d$ is compared to service s_j's promised quality $QRef_p$ and, if necessary, a trust updating algorithm is run to compute the new personal evaluation of service s_j. In essence, personal evaluation reflects the *Quality* performance of the provider in consumer's views. The personal evaluation $PerEval_j^x$, represents only consumer t_x's perception of the provider s_j's reputation. Other service consumers may differ or concur with t_x's observation of s_j. A service consumer that inquires about the reputation of a given service provider from its peers may get various differing personal evaluation "feedbacks." To get a correct assessment of the service provider's behavior, all the personal evaluations for s_j need to be aggregated. Assume L denotes the set of service consumers which have interacted with s_j in the past and are willing to share their personal evaluations of s_j. We assume that L is not empty, i.e., some service willing to share information can be found. Thus, $L \subseteq T$ with $L \neq \emptyset$ and each service x in L has $PerEval_j^x$ values for s_j. Then, consumer x's trust over s_j's ability to deliver is defined as:

$$Trust(s_j) = \Gamma_{x \in L}(PerEval_j^x) \quad (1)$$

where Γ represents the aggregation function. Equation 1 provides a first approximation of how the trust may be assessed. However, it involves various factors that need to be precisely defined and measured. We have previously defined such a model in [6].

In service compositions, service orchestrators need to be aware of their own (i.e., the composition's) reputation so that in cases where the component services negatively effect the reputation of the orchestrator, it can take suitable actions like replacing the component service, penalizing the defaulting component, etc. In other words, the orchestrator should transfer the (part of the) blame, incurred in the form of loss in reputation to the actual component service responsible (defined: culprit). Similarly, the orchestrator should also recognize the component services working honestly, thereby increasing the orchestrator's reputation. This recognition can come in the form of reputation increment for the component services responsible. In this paper we focus on "blame sharing." The findings are also potentially applicable to reputation increment scenarios.

A service composition may involve component services that are themselves compositions. Thus, we may have nested compositions, and a composition orchestrator may not be aware of *all* the services involved. In these situations, blame sharing is not straight forward as there are many "hidden" variables involved. For example, consider Figure 1, where a user plans to attend a conference at City-X, which is totally new to the user. Since the user does not have enough time to invest in planning the whole trip, the *Trip Planner* (TP) virtual enterprise is consulted. The user communicates some constraints to TP. For example, the user may be interested in (a) touring the city using a bus tour guide (no cruise required), (b) be interested in watching a movie, and (c) plan to go to a good theater in the vicinity, and (d) be keen on exploring the area museums also. The services selected to meet the user's demands are represented by straight lines in Figure 1. Note that *Museums*, *Downtown Tours*, *Cinema* and *Theatre* are individual services that do not outsource any functionality, while *Sightseeing Service*, *Area Tours*, *Arts*, *Bus Tour* are composed services that delegate some part of their assigned work to other services (all services not mentioned here). Since the user only interacts with *Trip Planner*, it is unaware of other services due to privacy, trade secrets, etc. considerations. Similarly, *Trip Planner* is unaware that which of its component services is a composed service.

It may happen that the services chosen by *Trip Planner* do not provide an optimal combination when invoked together, and the quality (hence reputation) of the *Trip Planner*-composition drops. Note that although a component service's selection is based on its reputation (i.e. highly reputable services are selected), the combination thereof may not prove to be ideal due to a variety of reasons (as trade constraints, legislative or geographical incompatibilities, etc).

We identify the reputation of composed services in a top-down manner. This means that post-transaction completion, apart from rating the component services, the invoking service (*Trip Planner* in the running example) identifies the operations that did not meet its expected QoWS values.

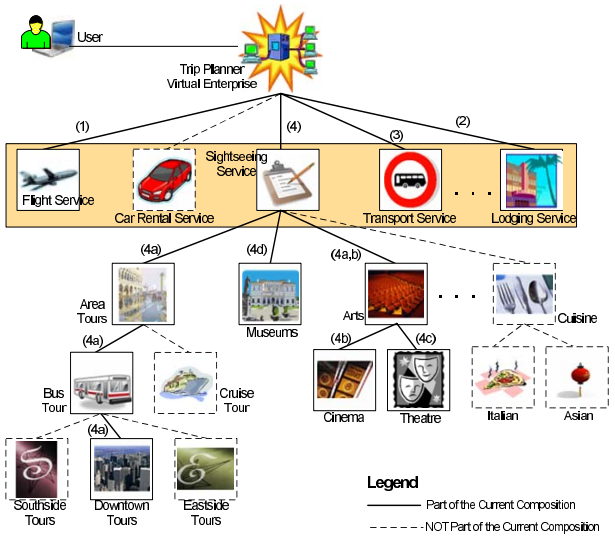

Figure 1: Sample Composition

Based on this experience, the invoker adjusts the reputation of the component service(s) in its knowledge scope responsible for delivering that operation. For example, suppose *Trip Planner* identifies that something went wrong with the user's sightseeing experience which causes a degradation in *Trip Planner*'s reputation. We assume that *Bus Tour* is the real "culprit", but since *Trip Planner* only communicates with *Sightseeing Service*, it transfers the blame to the *Sightseeing Service* by reducing its reputation. Since we assume the services can observe their reputations (through indirect means), *Sightseeing Service* would in turn calculate the difference in its reputation. Based on the percentage decrease in its reputation, *Sightseeing Service* would then transfer the blame to its component services. Each service that transfers or propagates the blame down the composition chain may deploy a different strategy in calculating the amount of blame to be transferred. For example, *Sightseeing Service* may decide to decrease the reputation of component services linearly, exponentially, etc. The component services would in turn employ similar procedures of transferring the blame down to their component service(s). The rationale for employing such a reputation degradation chain is that each component service is held responsible for the composition's reputation degradation.

In situations where a single operation branches into several sub-operations, the blame has to be transferred to the actual culprit and service(s) responsible for other sub- operations should not take undue blame. However, since the invoker only rates the operations atomically, the component service cannot determine which sub-operation defaulted. In our example, *Sightseeing Service* outsources the sightseeing activities to three component services namely *Area Tours*, *Arts*, and *Museums*. If *Trip Planner* labels sightseeing as the faulty operation, it will decrement *Sightseeing Service*'s reputation as it outsources this functionality only from *Sightseeing Service*. Since sightseeing is outsourced to three services (*Area Tours*, *Arts*, and *Museums*), *Sightseeing Service* has no way of identifying the culprit sub-operation directly. Hence *Area Tours*, *Arts*, and *Museums* would all be blamed for the fault. Since, in the discussion above we assumed *Bus Tour* was the real culprit, *Arts*, and *Museums* are blamed unjustly (*Area Tours* should be responsible because of *Bus Tour*). In other words, blaming each component service equally may not prove to be the ideal solution in terms of fairness.

3. TRUST MAINTENANCE IN COMPOSITIONS

We propose to optimize fairness in blame-forwarding through historical knowledge. Due to the subjective nature of trust, uncertain and incomplete trust information regarding all component services, this is not a straight forward task. Thus, we employ a statistical model to incorporate fuzziness and randomness in trust as complementary and essentially inseparable concepts. In the following, we first give a brief overview of this approach, and then describe our model.

3.1 Background

The statistical cloud model (or simply, the cloud model) states that the concept of fuzzy membership functions is not sufficient for representing the uncertainty and imprecision in real world settings, and probability theory needs to be incorporated to overcome this inadequacy. In essence, a cloud model can uniformly describe the concepts of randomness, fuzziness, and their relationship in quantitative terms. Experiment results have shown that the cloud model exhibits higher levels of simplicity and robustness in comparison with traditional fuzzy logic and probability based methods [3] [4]. In the following, we provide a brief overview of the cloud model.

Let U be the quantitative universe of discourse, and C denote a qualitative concept associated with U. If $x \in U$ is a random realization of C, and $\mu(x) \in [0,1]$ is a random variable with stable tendency denoting the degree of certainty for x belonging to C, that is:

$$\mu : U[0,1] \qquad \forall x \in U \qquad x \rightarrow \mu(x)$$

The distribution of x in U is called the cloud (denoted $C(X)$) and each x is called a cloud drop. Note that in probabilistic terms, $x \in U$ is not a simple random number but it has a certainty degree, which itself is also random and not a fixed number. The cloud is composed of a number of drops, which are not necessarily ordered. The underlying character of the qualitative concept is expressed through all cloud drops. Hence the overall feature of the concept is more precisely represented by a large number of drops. The certainty degree of each cloud drop defines the extent to which the drop can represent the concept accurately. Formally, a cloud's quantitative representation is defined over a set of N ordered pairs (x_i, y_i), where x_i is a cloud drop, and y_i is its certainty degree, with $1 \leq i \leq N$.

A one-dimension normal cloud model's qualitative representation can be represented by a triple of quantitative characteristics: Expected value (Ex), Entropy (En) and Hyper-Entropy (He). Ex is the expectation of the cloud drops' distribution, i.e., it corresponds to the center of gravity of the cloud (containing elements fully compatible with the qualitative concept). En represents the uncertainty measurement of a qualitative concept. It is determined by both the randomness and fuzziness of the concept. En indicates how

many elements could be accepted to the qualitative linguistic concept. He is a measure of the dispersion on the cloud drops. It can also be considered as En's uncertainty. Vector $\vec{v} = (Ex, En, He)$ is called the eigenvector of a cloud [3].

The transformation of a qualitative concept expressed by Ex, En, and He to a quantitative representation expressed by the set of numerical cloud drops is performed by the forward cloud generator [4]. Given these three digital characteristics (Ex, En, He), and the number of cloud drops to be generated (N), the forward cloud generator can create these N cloud drops in the data space with a certainty degree for each drop that each drop can represent the qualitative concept. The procedure is:

1. Generate a normally distributed random number F with mean En and standard deviation He.
2. Generate a normally distributed random number x with mean Ex and standard deviation F.
3. Calculate $y = e^{-\frac{(x-Ex)^2}{2(F)^2}}$.
4. (x, y) represents a cloud drop in the universe of discourse.
5. Repeat Steps 1-4 until N cloud drops are generated.

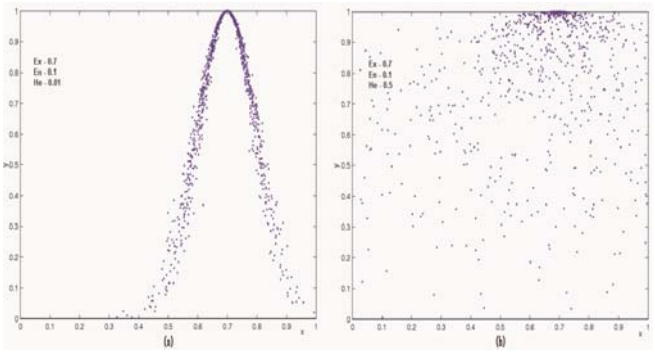

Figure 2: Normal Cloud with Same Ex and En, but Different He Values

Figure 2(a) shows the graph of a one-dimensional cloud whose digital characteristics are (0.7, 0.1, 0.01). A similar cloud with same Ex and En, but a different He (0.7, 0.1, 0.5) is shown in Figure 2(b). As defined in the above algorithm, the quantitative value of cloud drops is determined by the standard normal form distribution function. Hence, the certainty degree function adopts a bell-shaped curve. This is similar to the one adopted in fuzzy set theory. As mentioned earlier, the normal cloud model is therefore an inclusive model based on probability theory and fuzzy set theory, and is able to depict randomness in the former and fuzziness in the latter.

3.2 Reputation Propagation

The premise behind our approach is that past defaulters are *likely* to defect again in a composition. We assume that composition orchestrators can monitor their own reputations throughout the community, in a manner similar to obtaining the reputations of other service providers. A negative change in the orchestrator reputation indicates that one (or more) of the component services did not function as expected in the final composition. When the amount of change is above a defined threshold, the orchestrator transfers the blame to the component services involved. The statistical cloud defined above is used to assess the amount of blame to be transferred (in form of reputation loss), and the "inferred identity" of the real culprit component service. After blame is forwarded, the orchestrator monitors the reputations of the component services for a defined time period to ascertain the validity of its decision. The objective of this activity is two-fold: First, to assess whether the blame was assigned to the correct component. Second, to establish that blame forwarding helped in correcting the component's behavior. If the component service does not ameliorate its behavior, the amount of blame is increased.

The model's recommendations are guided by the component services' profiles maintained by the orchestrator. The profile consists of the component services invoked by the orchestrator, the interaction time-stamp, component reputations, and the sequence in which those component services were invoked. Since outsourced operations are not equal in terms of impact in a composition (some operations are more important than others), amount of blame to be transferred should also reflect that. The orchestrator also assigns an "operational significance" $(OpSign)$ to each outsourced sub-operation, denoting the importance of the sub-operation in completing the parent operation, with $\sum OpSign_k = 1$. $OpSign$ for each component is also part of its historical profile.

We incorporate $OpSign$ with the stored reputations for previous interactions to produce the three digital characteristics of the cloud (Ex, En, He). The backward cloud generator allows transformation of the cloud model from its quantitative representation to a qualitative one. Given a set of N ratings for a component service $PerEval_j^x (x = 1, 2, ..., N)$, we can extract the three characteristics as:

1. $$Ex = \frac{\sum_{x=1}^{N}(OpSign_x PerEval_j^x)}{\sum_{x=1}^{N} OpSign_x}$$
2. $$En = \sqrt{\frac{\pi}{2}} \times \frac{\sum_{x=1}^{N} OpSign_x |PerEval_j^x - Ex|}{\sum_{x=1}^{N} OpSign_x}$$
3. $$He = \sqrt{\frac{\sum_{x=1}^{N} OpSign_x (PerEval_j^x - Ex)^2}{OpSign_x} - (En)^2}$$

The next step is using the three discovered characteristics to make a subjective assessment of the component's role in orchestrator's reputation degradation. Since He is a measure of En's uncertainty, we only use Ex and He to quantify the provider's trust and the associated uncertainty. This allows us to consider the latest majority view of the provider's reputation and the decentralization of ratings from it. A

higher value of Ex therefore indicates high reputation, i.e., the component service is not blame-worthy, while a small He indicates the stability of the ratings around this decision. Intuitively this makes sense, but for a large N, making these comparisons is non-trivial. For instance, Ex and He can occur together in one of four forms: one is high/low the other is low/high, both are high, or both are low. Therefore, to quantify the relationship between the two characteristics, i.e., the component (s_j)'s assessment, we use:

$$ProfileTrust(s_j) = \begin{cases} 1 - \frac{He}{Ex+He} & if Ex \neq 0 \ \& \ He \neq 0; \\ Ex & if He = 0; \\ 0 & if Ex = 0; \end{cases} \quad (2)$$

where both $Ex \neq 0$ and $He \neq 0$. The component service (s_j) with the lowest *ProfileTrust* is thus considered "at fault". In cases where multiple services have equally low values, blame is shared. Moreover, in cases where no profile information is available (e.g. first time interactions with components), blame is shared between the "new" component(s), and the existing lowest performing one (if any exists).

The next step is assessing the amount of blame to be transferred. The orchestrator (*Sightseeing Service* in this case) may use a number of strategies: Assume that *Trip Planner*'s feedback about *Sightseeing Service* reduces its reputation by a factor Δ. In order to "forward" this blame, *Sightseeing Service* may use: a *constant* strategy (i.e., component service's reputation is reduced by at least Δ), a *linear* strategy (i.e., component service's reputation is reduced by at least 2Δ), an *exponential* strategy (i.e., component service's reputation is reduced by at least Δ^2), etc. The component services will also choose a strategy to share the blame with their component services (if any).

There are no hard and fast rules for deciding which strategy is the best. It is thus at the discretion of the invoking service to decide which strategy suits its business model. We propose to use the *rate of maliciousness* in the service community as a measure of which strategy to use. The rate of maliciousness (denoted $\Re$) is defined as the ratio of the number of transactions where the providers defect, to the total number of transactions ($\Re$ thus lies in the range [0, 1]). A provider's "defection" is measured after each individual transaction, by the service consumer (denoted rater). If the provider performs satisfactorily in the transaction, the rater can label the transaction as "acceptable." Alternatively, the transaction is labelled as "defective." Thus, defection (denoted D) can be represented as a binary. Since service raters can differ in their total number of transactions, and the number of defective transactions experienced, we can expect a variation in the value of $\Re$ across different service raters. In essence, the value of $\Re$ would depend on each rater's personal experience, and the manner in which it estimates D after each transaction. The basic idea of the proposed scheme is for the component service invoker to forward less blame when $\Re$ is low, and higher degrees of blame when $\Re$ is high. This allows the consumer to adapt to the current state of the system (i.e., defective vs. acceptable transactions). Formally, for each service consumer i, $\Re$ is defined as:

$$\Re_i = \frac{D_i}{T_i} \quad (3)$$

where D_i is the number of transactions where providers have defected for consumer i, and T_i is the total number of transactions that consumer i has undertaken. Since the severity of defective transactions varies, the service rater can also assign relative weights to the transactions. For example, in a "high impact" transaction where the consumer suffers a huge loss as a consequence of the provider's defection, the consumer may count two (or more) defective transactions instead of one (while increasing the T_i count by only one), to increase $\Re_i$. The assignment of such weights is left at the discretion of the service rater.

4. EXPERIMENTS

We have performed preliminary experiments to show the applicability of the proposed model. We assume that reputations for services are evaluated and available using assessment techniques defined for individual services (e.g. [6] [13] [11], etc). Through these experiments we intend to answer two main questions: (1) Does blame propagation help? (2) What is the impact of wrongfully blaming a component service, i.e., when the service is not at fault but gets punished anyhow? Due to space restrictions, complete details of the experiment environment are not listed here. The interested reader is referred to [6].

We assume that each service can obtain a measure of its "true" reputation. After any dissatisfying transaction, the composition orchestrator Web service (acting as the invoker) may leave a negative feedback for the component service, thereby reducing its reputation. Since reputation is defined as an aggregation of a number of feedbacks, the effect of a single composition orchestrator may not be profound. However, for experimental purposes we assume that a negative feedback by even one rater reduces the reputation of the component service. Moreover, if more than one operation are outsourced from one service, the invoker leaves feedback for each operation. This allows the component service to know which of its operations defaulted in the invoker's view. To the best of our knowledge, no comparable work for "reputation propagation" in service compositions exists yet. Thus, experiment results cannot be validated against an existing model.

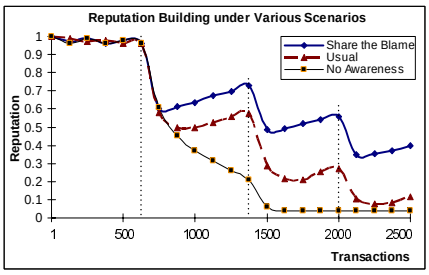

Figure 3: Reputation Propagation Effects on a Composed Service's Reputation

Figure 3 shows the effects of blame propagation on the services involved in a composition. The reputation building process is shown for three scenarios namely share the blame, usual, and no awareness. The process is shown for a single service in a composition of ten or more services for each scenario. The services experience a (forced) drop in their

reputation in the 650th., 1350th. and 2000th. transaction. In Share the Blame scenario, the composed service identifies the faulty service based on the information received, and propagates the "blame" to its components. In these experiments we force the assumption that once a service becomes aware of its reputation loss due to blame propagation (reputation assessment through direct monitoring is different from this), it does not repeat its mistake and rectifies its behavior. This causes a gradual elevation in the composed service's reputation right after it suffered a degradation. In the Usual case, the service does not identify the faulty service immediately, and thus experiences a loss in its reputation due to the faulty operation for extended periods. Upon identification, and new service selection, the composed service starts its gradual reputation ascend. In the third case, since the service is not aware of the fault, it keeps repeating its mistake and hence keeps losing its reputation. When the reputation reaches a very low value, the service is not selected again, and its (low) reputation remains constant. We can clearly see from Figure 3 that our proposed technique produces favorable results in terms of the composition's reputation.

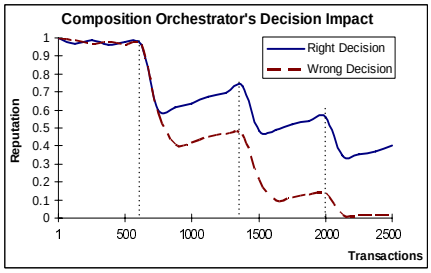

Figure 4: Impact on the Composition's Reputation because of Right/Wrong Decision

Figure 4 shows the impact of a correct or an incorrect decision on part of the composition orchestrator, on the reputation of the composition. Here also, the composition experiences a drop in reputation at the 650th., 1350th. and 2000th. transaction (enforced). When the faulty operation is identified correctly, the composition reevaluates itself (through propagation) quickly and the reputation is maintained/increased. Alternatively, the composition takes some time to recover from a wrong decision. The reputation keeps declining in face of the wrong decision for sometime, and only when the correct fault is detected, does the reputation starts increasing. Although the proposed technique provides a reasonable solution, the decision of the composition orchestrator does play an important role in propagating and hence defining the correct reputation.

5. CONCLUSION

We have presented a framework for managing reputations of Web service compositions. The framework is based on previous recommendations from supply-chain management studies, and uses the propagation of reputation information throughout the composition as the primary tool in deterring malicious behavior by component services. The results indicate the usefulness of the proposed approach. We focused on an environment where Web services can act as both consumers (i.e., requesters) and providers of services, without the need of a *trusted third party*. We have also conducted preliminary experiments to verify the proposed model. Results exhibit strong evidence that our approach provides a fairly accurate assessment of provider trust. In the future, we intend to implement the model in a real Web services environment.

6. REFERENCES

[1] K. Bharadwaj and M. Al-Shamri. Fuzzy computational models for trust and reputation systems. *Electron. Commer. Rec. Appl.*, 8(1):37–47, 2009.

[2] R. He, J. Niu, M. Yuan, and J. Hu. A novel cloud-based trust model for pervasive computing. *Computer and Information Technology, International Conference on*, 0:693–700, 2004.

[3] D. Li, J. Han, X. Shi, and M. Chan. Knowledge representation and discovery based on linguistic atoms. *Knowledge-Based Systems*, 10(7):431 – 440, 1998. KDD: Techniques and Applications.

[4] D. Li, C. Liu, and W. Gan. A new cognitive model: Cloud model. *Int. J. Intell. Syst.*, 24(3):357–375, 2009.

[5] Z. Malik and A. Bouguettaya. Evaluating rater credibility for reputation assessment of web services. In *8th International Conference on Web Information Systems Engineering (WISE 07)*, pages 38–49, 2007.

[6] Z. Malik and A. Bouguettaya. Reputation-based Trust Management for Service-Oriented Environments. *VLDB Journal*, 18(4):885–911, August 2009.

[7] B. Medjahed, A. Bouguettaya, and A. Elmagarmid. Composing Web Services on the Semantic Web. *The VLDB Journal*, 12(4), November 2003.

[8] J. Niu, Z. Chen, and G. Zhang. Towards a subjective trust model with uncertainty for open network. *Grid and Cooperative Computing Workshops, International Conference on*, 0:102–019, 2006.

[9] M.P. Papazoglou and D. Georgakopoulos. Serive-Oriented Computing. *Communcications of the ACM*, 46(10):25–65, 2003.

[10] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation Systems. *Communication of the ACM*, 43(12), December 2000.

[11] J. Sabater and C. Sierra. Bayesian Network-Based Trust Model. In *Proc. of the first Intl. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 475 – 482, Bologna, Italy, 2003.

[12] Z. Shibin, S. Xiang, and Q. Zhi. Subjective trust evaluation model based on fuzzy reasoning. *Electronic Commerce and Security, International Symposium*, 1:328–332, 2009.

[13] A. Whitby, A. Josang, and J. Indulska. Filtering Out Unfair Ratings in Bayesian Reputation Systems. *The Icfain Journal of Management Research*, 4(2):48–64, February 2005.