Personalized Reliability Prediction of Web Services

ZIBIN ZHENG and MICHAEL R. LYU, Shenzhen Research Institute, The Chinese University of Hong Kong

Service Oriented Architecture (SOA) is a business-centric IT architectural approach for building distributed systems. Reliability of service-oriented systems heavily depends on the remote Web services as well as the unpredictable Internet connections. Designing efficient and effective reliability prediction approaches of Web services has become an important research issue. In this article, we propose two personalized reliability prediction approaches of Web services, that is, neighborhood-based approach and model-based approach. The neighborhood-based approach employs past failure data of similar neighbors (either service users or Web services) to predict the Web service reliability. On the other hand, the model-based approach fits a factor model based on the available Web service failure data and use this factor model to make further reliability prediction. Extensive experiments are conducted with our real-world Web service datasets, which include about 23 millions invocation results on more than 3,000 real-world Web services. The experimental results show that our proposed reliability prediction approaches obtain better reliability prediction accuracy than other competing approaches.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Software/Program Verification—*Reliability*

General Terms: Design, Reliability

Additional Key Words and Phrases: Reliability prediction, user-collaboration, web service

ACM Reference Format:
Zheng, Z. and Lyu, M. R. 2013. Personalized reliability prediction of web services. *ACM Trans. Softw. Eng. Methodol.* 22, 2, Article 12 (March 2013), 25 pages.
DOI:http://dx.doi.org/10.1145/2430545.2430548

1. INTRODUCTION

Web services are self-contained and self-describing computational components designed to support machine-to-machine interaction by programmatic Web method calls [Zhang et al. 2007]. By discovering and integrating available Web services of different organizations, service-oriented systems can be efficiently built. Service-oriented systems have been widely engaged in a lot of domains, such as business-to-business

A preliminary version of this article was published in *Proceedings of the 32nd International Conference on Software Engineering (ICSE2010)*, Cape Town, South Africa, May 2–8, 2010. See Zheng and Lyu [2010].
The work described in this article was sponsored by the National Basic Research Program of China (973 Project No. 2011CB302603), the National Natural Science Foundation of China (Project No. 61100078), the Shenzhen Basic Research Program (Project No. JCYJ20120619153834216), and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 415410).
Z. Zheng and M. R. Lyu are with the Shenzhen Research Institute and the Department of Computer Science and Engineering, the Chinese University of Hong Kong. M. R. Lyu is also with the School of Computer Science, National University of Defence Technology, Hunan, China.
Authors' address: Z. Zheng and M. R. Lyu, Room 927, Ho Sin-Hang Engineering Building, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong; email: zbzheng, lyu@cse.cuhk.edu.hk.

DOI:http://dx.doi.org/10.1145/2430545.2430548

collaboration [Ye et al. 2006], automotive systems [ter Beek et al. 2008], multimedia services [Scholz et al. 2008], etc. Reliability of service-oriented systems heavily relies on the employed Web services. To optimally select and integrate reliable Web services, we must be able to predict the reliability of Web services. The ability to predict reliability of invoked Web services early at the system architecture design phase can help to reduce re-engineering cost and to produce more reliable service-oriented systems.

Software reliability is the probability of failure-free software operation for a specified period of time in a specified environment [ANSI/IEEE 1991]. In order to predict the reliability of software systems, failure data need to be properly collected by various means during software development and operational phases [Lyu 2007]. A number of software reliability models have been proposed for predicting software reliability by observing, accumulating, and analyzing previous failure data (e.g., Musa's execution time model [Musa et al. 1990], Putnam's model [Putnam and Myers 1992], Rome Laboratory model [Friedman et al. 1992], etc.). Traditionally, comprehensive testing schemes are conducted on the software systems to collect failure data and to make sure that the reliability threshold has been achieved before releasing the software to end users. In the environment of service computing, user-observed reliability of a service-oriented system not only relies on the system itself, but also heavily depends on the remote Web services and the user characteristics (e.g., geographic locations, network conditions, operational profiles [Musa 1993], etc.). Influenced by the unpredictable Internet connections and many other environmental and operational factors, different service users may experience quite different reliability performance on the same Web service. Therefore, personalized Web service reliability prediction is required for different service users.

To obtain the past failure data of Web service for making reliability prediction, Web service evaluation [Tsai et al. 2008] from the client-side is usually required. Unfortunately, traditional exhaustive testing becomes difficult and sometimes even impossible for the service-oriented systems due to the following reasons.

— Invocations of Web services may be charged since Web services are owned and hosted by different organizations. Even if the invocations are free, executing a large number of Web service invocations consume resources of both the service users and the service providers. Moreover, Web service invocations may produce irreversible effects in the real world.
— Since there are typically a lot of alternative Web service candidates in the Internet, it is time-consuming and expensive to conduct real-world evaluations on all the candidates. However, without a comprehensive evaluation, sufficient past failure data of Web services cannot be collected. It is thus difficult for the system designers to make accurate reliability prediction of the Web services.

A number of reliability prediction methods have been proposed for component-based systems [Cheung 1980; Gokhale and Trivedi 2002; Goseva-Popstojanova and Trivedi 2001; Yacoub et al. 1999] and service-oriented systems [Cardoso et al. 2002; Grassi and Patella 2006]. Most of these prediction approaches focus on system-level compositional analysis and assume that reliabilities of the components (or Web services) are known. A few approaches [Cheung et al. 2008; Goseva-Popstojanova et al. 2003; Reussner et al. 2003], which consider component-level reliability prediction, are mainly designed for traditional component-based systems. As discussed in Tsai [2005], reliability prediction of service-oriented systems is quite different from that of traditional systems. Compared with reliability prediction of software components, reliability prediction of Web services is much more challenging, because: (1) it is difficult to collect sufficient past failure data as discussed above; (2) the internal information of Web services are unavailable, since Web services are designed and developed by other organizations;

and (3) user-observed Web service reliability is greatly influenced by the unpredictable Internet connections.

To attack this challenge, we propose two personalized reliability prediction approaches for Web services. By these approaches, the reliability of Web services can be personally predicted for a service user even if this current user has not invoke these services previously and has no idea on their internal information. In the field of service computing, service users refer to the systems/applications that use the Web services. Complementary to the existing reliability prediction approaches [Cardoso et al. 2002; Cheung 1980; Gokhale and Trivedi 2002; Goseva-Popstojanova and Trivedi 2001; Grassi and Patella 2006; Wang et al. 2006b; Yacoub et al. 1999], which mainly focus on system-level compositional analysis, this paper focuses on component-level reliability prediction. The Web service reliability obtained by our approaches can be employed as input to other system-level reliability prediction approaches. This article extends from its preliminary version [Zheng and Lyu 2010], which proposed a neighborhood-based approach for Web service reliability prediction, and includes the following additional contributions.

— First, our preliminary neighborhood-based approach [Zheng and Lyu 2010] has been improved. We propose an enhanced similarity computation approach to improve the similarity computation accuracy, and present an enhanced weight computation to increase the influence of users with larger similarity values.
— Second, beside the neighborhood-based approach, a model-based approach is proposed for making personalized Web service reliability prediction. The premise behind our model-based approach is that there is a small number of factors influencing the user-observed Web service failure probability. Based on the available Web service failure data, we can train a factor model and employ this model to make further reliability prediction.
— Third, our most up-to-date Web service dataset is employed for the experimental study, which contains about 23 millions Web service invocation results from 102 service users on 3568 real-world Web services. Extensive experiments are conducted to compare the neighborhood-based approach, the model-based prediction approach, and other competing approaches.

The rest of this article is organized as follows. Section 2 describes the research problem. Section 3 and Section 4 present our neighborhood-based and model-based reliability prediction approaches, respectively. Section 5 shows the experiments. Section 6 discusses the comparisons between our neighborhood-based approach and model-based approach. Section 7 surveys related work and Section 8 concludes the article.

2. PROBLEM DESCRIPTION

Web service selection aims at selecting an optimal Web service from a set of functionally equivalent service candidates by considering non-functional properties, such as reliability, security, performance, etc. Reliability is one of the most important criteria when making service selection. The most straightforward way for obtaining user-observed Web service reliability is to conduct client-side evaluation on all the service candidates. However, it is time-consuming, expensive, and sometimes even impossible to make a comprehensive evaluation on all the Web service candidates.

Another approach is to employ the past failure data of Web services observed by other service users for predicting the Web service reliability for the current user. For example, for the new service users who did not invoke any Web services previously, the

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0.5	0.2		0.3		0.4
u_2		0.1		0.2	0.5	
u_3	0.4		0.3		0.1	
u_4		0.6		0.1		
u_5	0.5		0.2			0.3

Fig. 1. A 5×6 user-item matrix.

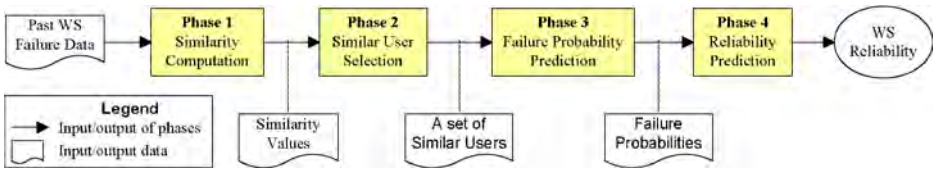


Fig. 2. Procedures of neighborhood-based reliability prediction.

performance of a Web service can be predicted by its *average failure probability* ($\overline{p_i}$), which is defined as:

$$\overline{p_i} = \frac{1}{m} \sum_{a=1}^{m} p_{u,i}, \tag{1}$$

where $p_{u,i}$ is the failure probability of Web service candidate i observed by service user u, m is the number of service users, and $\overline{p_i}$ is the average failure probability of the Web service candidate i. Based on the *average failure probabilities* of the service candidates, the best performing candidate can be determined. However, since different service users have different characteristics (e.g., in different geographic locations, under different network conditions, with different operational profiles [Musa 1993], etc.), the current user may not experience similar failure probability performance as the *average failure probability*. Personalized approaches are therefore needed to accurately predict the Web service failure probability for a certain user.

The process of personalized Web service failure probability prediction can be perceived as involving a user-item matrix [Zheng and Lyu 2010] such as shown in Figure 1, where each entry in the matrix represents the value of failure probability of a Web service (e.g., i_1 to i_6) observed by a service user (e.g., u_1 to u_5). In Figure 1, each service user observes several failure probability values of the invoked Web services, while a missing entry represents an uninvoked service by the user. The problem we investigate in this article is how to accurately predict the missing failure probability values in the user-item matrix by employing the available values. By predicting the Web service failure probability values in the user-item matrix, we can provide personalized reliability prediction on the uninvoked Web services for both existing and new service users.

To predict the missing values in the user-item matrix, we propose a neighborhood-based prediction approach in Section 3 and a model-based prediction approach in Section 4.

3. NEIGHBORHOOD-BASED RELIABILITY PREDICTION

In this section, we propose a neighborhood-based reliability prediction approach for Web services, which is designed as a four-phase process. As shown in Figure 2, in

Table I. Notations of Neighborhood-Based Reliability Prediction

Notations	Descriptions
u and a	service user u and service user a
i and j	Web service i and Web service j
$p_{u,i}$	failure probability of Web service i observed by user u
$\overline{p_i}$	average failure probability of Web service i observed by different users
$\overline{p_u}$	average failure probability of service user u on different Web services
n	number of Web services
m	number of users
I_u and I_a	Web services invoked by user a and u, respectively
U_i and U_j	service users who invoked Web service i and j, respectively
$S(u)$	a set of similar service users of user u
$S(i)$	a set of similar Web services of Web service i
$Sim(u,a)$	the similarity between user u and a via Pearson Correlation Coefficient (PCC)
$Sim'(u,a)$	the similarity between user u and a via enhanced PCC
λ	a parameter for combining prediction results from UPCC and IPCC.
$R(t)$	reliability of a Web service for a time period t
β	the number of Web service failures during a certain time duration (failure rate)
f	invocation frequency of Web services

Phase 1, similarities between the current user with other service users are calculated. In Phase 2, a set of similar users are identified based on the calculated similarities. In Phase 3, the invocation failure probabilities of Web services are predicted for the current user by employing the past usage experiences of the similar users. Finally, in Phase 4, reliability of the target Web service is predicted. Details of these phases are presented at Section 3.1 to Section 3.4, respectively. The notations of our neighborhood-based approach are summarized in Table I.

3.1. Phase 1: Similarity Computation

We assume that there are m service users, n Web services, and the relationship between the users and the Web services is denoted by an $m \times n$ user-item matrix. Figure 1 shows an example of user-item matrix. Each entry $p_{u,i}$ in the matrix represents the failure probability of Web service i observed by the service user u. $p_{u,i} = null$ if user u did not invoke Web service i previously.

Pearson Correlation Coefficient (PCC) is a commonly used approach for similarity computation. The purpose of PCC is to indicate a linear relationship between two variables. PCC has been widely applied in a number of recommendation systems [Shardanand and Maes 1995], since it considers the differences in the user value style and can achieve high accuracy. PCC employs the following equation to compute the similarity between service user u and service user a based on their commonly invoked Web services:

$$Sim(u,a) = \frac{\sum_{i \in I_u \cap I_a} (p_{u,i} - \overline{p_u})(p_{a,i} - \overline{p_a})}{\sqrt{\sum_{i \in I_u \cap I_a} (p_{u,i} - \overline{p_u})^2} \sqrt{\sum_{i \in I_u \cap I_a} (p_{a,i} - \overline{p_a})^2}}, \quad (2)$$

where $I_u \cap I_a$ is a set of commonly invoked Web services by both user u and user a, $p_{u,i}$ is the failure probability of Web service i observed by the service user u, and $\overline{p}_u$ represents the average failure probability of all the Web services invoked by user u. The PCC similarity $Sim(u,a)$ is in the interval of [-1,1], where a larger value indicates

higher similarity. PCC value of 1 indicates a perfect positive correlation between two users. PCC value of -1 means a complete negative correlation between two users. PCC value of 0 indicates that two users are independent and have no correlation with each other. For the example shown in Figure 1, $I_{u_1} = \{i_1, i_2, i_4, i_6\}$ and $I_{u_2} = \{i_2, i_4, i_5\}$. Therefore, $I_{u_1} \cap I_{u_2} = \{i_2, i_4\}$, indicating that the co-invoked Web services of u_1 and u_2 are i_2 and i_4. Employing the failure probability values of these co-invoked Web services, PCC can calculate the similarity of u_1 and u_2.

Similar to this procedure, PCC can also be employed to calculate the similarity between Web service i and Web service j by using:

$$Sim(i,j) = \frac{\sum_{u \in U_i \cap U_j} (p_{u,i} - \overline{p_i})(p_{u,j} - \overline{p_j})}{\sqrt{\sum_{u \in U_i \cap U_j} (p_{u,i} - \overline{p_i})^2} \sqrt{\sum_{u \in U_i \cap U_j} (p_{u,j} - \overline{p_j})^2}}, \tag{3}$$

where $U_i \cap U_j$ is a set of service users who invoke both the Web services i and j, and $\overline{p_i}$ is the average failure probability of Web service i, which can be calculated by Eq. (1).

When the number of co-invoked Web services is large, PCC can provide accurate similarity computation. However, when the number of co-invoked Web services is small and the observed failure probability values happen to be similar, it is possible that PCC provides high similarity value for the users who are actually not similar, since PCC does not consider the number of co-invoked Web services when making the similarity computation. Consequently, service reliability can be overestimated. To address this problem, we employ a logistic function to reduce the influence of a small number of similar yet co-invoked Web services. An enhanced PCC for the similarity computation between two service users is defined as:

$$Sim'(u,a) = \frac{1}{1 + e^{-|I_u \cap I_a|}} Sim(u,a), \tag{4}$$

where $Sim'(u,a)$ is the new similarity value, $|I_u \cap I_a|$ is the number of Web services that are invoked by both user u and user a. When two users have a small number of co-invoked Web services (i.e., the value of $|I_u \cap I_a|$ is small), the logistic function $\frac{1}{1+e^{-|I_u \cap I_a|}}$ will decrease the similarity estimation between the service users u and a. When the value of $|I_u \cap I_a|$ is large, the logistic function $\frac{1}{1+e^{-|I_u \cap I_a|}}$ is approximately 1, which has no impact on the PCC similarity estimation. For the example shown in Figure 1, since $I_{u_1} \cap I_{u_2} = \{i_2, i_4\}$, therefore $|I_{u_1} \cap I_{u_2}| = 2$.

Just like the user-based methods, an enhanced PCC for the similarity computation between different Web services is defined as:

$$Sim'(i,j) = \frac{1}{1 + e^{-|U_i \cap U_j|}} Sim(i,j), \tag{5}$$

where $|U_i \cap U_j|$ is the number of service users who invoked both Web service i and item j previously.

3.2. Phase 2: Similar User Selection

After calculating and ranking the PCC values between the current user and other users, a set of similar users can be identified. In our neighborhood-based approach, a parameter Top-K is employed to select similar users. Top-K indicates that k users, who have larger PCC values than the others, will be selected as similar users. In

reality, a service user may have limited similar users and the dissimilar users (e.g., with negative PCC values) may be selected when the number of similar users is less than k. However, including dissimilar users may greatly reduce the prediction accuracy. In our approach, the service users who have negative correlation (negative PCC values) with the current user will be excluded.

To predict a missing entry $p_{u,i}$ in the failure probability matrix, a set of similar service users $S(u)$ can be identified as:

$$S(u) = \{a | Sim'(u,a) \geq Sim'_k, Sim'(u,a) > 0, a \neq u\}, \tag{6}$$

where Sim'_k is the kth largest PCC value with the current user u, $Sim'(u,a) > 0$ is to exclude the dissimilar users, and $Sim'(u,a)$ can be calculated by Eq. (4).

Similar to the above procedure, a set of similar Web services $S(i)$ with the current Web service i can also be identified as:

$$S(i) = \{j | Sim'(i,j) \geq Sim'_k, Sim'(i,j) > 0, j \neq i\}, \tag{7}$$

where Sim'_k is the kth largest PCC value with the current Web service i and $Sim'(i,j)$ can be computed by Eq. (5).

3.3. Phase 3: Failure Probability Prediction

Employing the information from similar users, the user-based approaches [Breese et al. 1998] (named as *UPCC*) predict the missing value $p_{u,i}$ in the user-item matrix by the following equation:

$$p_{u,i} = \overline{p_u} + \sum_{a \in S(u)} w_a (p_{a,i} - \overline{p_a}), \tag{8}$$

where $\overline{p_u}$ and $\overline{p_a}$ are average failure probabilities of different Web services observed by users u and a, respectively, and w_a is the weight of the similar user a. In our preliminary version [Zheng and Lyu 2010], the weight w_a is calculate linearly by:

$$w_a = \frac{Sim'(a,u)}{\sum_{b \in S(u)} Sim'(b,u)}. \tag{9}$$

In this article, to increase the influence of users with higher similarity values, we calculate w_a by

$$w_a = \frac{Sim'(a,u)^2}{\sum_{b \in S(u)} Sim'(b,u)^2}. \tag{10}$$

By squaring the similarity values, Eq. (10) gives a relatively high weight to users with large similarity values, which will impose stronger influence on the missing value prediction. Equation (10) is effective when similar users are particularly desirable. Section 5.4.2 will provide performance comparison of Eq. (9) and Eq. (10) by employing real-world Web service failure probability values.

Similar to the user-based approach, we can employ the information of similar Web services $S(i)$, also known as item-based approaches [Sarwar et al. 2001] (named as *IPCC*) to predict the missing value $p_{u,i}$ by:

$$p_{u,i} = \overline{p_i} + \sum_{k \in S(i)} w_k (p_{u,k} - \overline{p_k}), \tag{11}$$

where $\overline{p_i}$ and $\overline{p_k}$ are average failure probabilities of Web services i and k observed by different service users, respectively, and w_k is the weight of the similar Web service k with respect to Web service i, which is defined as:

$$w_k = \frac{Sim'(i,k)^2}{\sum_{j \in S(i)} Sim'(i,j)^2}. \tag{12}$$

The values of Web service failure probability is in the range of [0,1]. However, the predicted values by UPCC (Eq. (8)) and IPCC (Eq. (11)) may not fall in this value range. Therefore, the predicted value is set to be 0 when it is smaller than 0, and set to be 1 when it is larger than 1.

Due to the sparsity of the $m \times n$ user-item matrix, predicting failure probability only by similar service users (UPCC) or similar Web services (IPCC) will potentially ignore valuable information that can make the prediction more accurate. To address this problem, we combine the prediction results by the user-based approach in Eq. (8) and the item-based approach in Eq. (11) to fully utilize the information of both similar users and similar Web services. When $S(u) \neq \emptyset \wedge S(i) \neq \emptyset$, we employ both similar users and similar Web services to predict the missing value by employing the following equation:

$$p_{u,i} = \lambda(\overline{p_u} + \sum_{a \in S(u)} w_a(p_{a,i} - \overline{p_a})) + (1-\lambda)(\overline{p_i} + \sum_{k \in S(i)} w_k(p_{u,k} - \overline{p_k})), \tag{13}$$

where λ ($0 \leq \lambda \leq 1$) is a user-defined parameter for determining how much the missing value prediction relies on the similar users or the similar Web services. The value of λ should be set based on experiences and experiments. Different applications and datasets may have their own characteristics, and consequently inherit different optimal λ values. The detailed experimental studies on the impact of the parameter λ are presented in Section 5.4.3.

In practice, a missing $p_{u,i}$ value may not have similar users or similar items. To predict the missing value as accurate as possible, when $S(u) \neq \emptyset \wedge S(i) = \emptyset$, we only include the information of similar users for making prediction. In this case, our approach degrades to the UPCC approach as shown in Eq. (8). When $S(u) = \emptyset \wedge S(i) \neq \emptyset$, we only engage the information of similar items for making prediction. In this case, our approach reduces to the IPCC approach as shown in Eq. (11). When $S(u) = \emptyset \wedge S(i) = \emptyset$, there are neither similar users nor similar Web services. Consequently, we predict $p_{u,i}$ by employing the following equation:

$$p_{u,i} = \begin{cases} \lambda\overline{p_u} + (1-\lambda)\overline{p_i}, & \overline{p_u} \neq null \ \& \ \overline{p_i} \neq null \\ \overline{p_u}, & \overline{p_u} \neq null \ \& \ \overline{p_i} = null \\ \overline{p_i}, & \overline{p_u} = null \ \& \ \overline{p_i} \neq null \\ NoPrediction, & \overline{p_u} = null \ \& \ \overline{p_i} = null \end{cases}, \tag{14}$$

where $\overline{p_u}$ is the average failure probability of different Web services observed by the service user u (named as *UMEAN*), and $\overline{p_i}$ is the average failure probability of Web service i observed by different service users (named as *IMEAN*). Equation (14) includes four situations: (1) When $\overline{p_u} \neq null \ \& \ \overline{p_i} \neq null$, we combine $\overline{p_u}$ and $\overline{p_i}$ for the missing value prediction; (2) when the target Web service is never invoked by any service users, and the service user has invoked other Web services previously ($\overline{p_u} \neq null \ \& \ \overline{p_i} = null$), we use UMEAN $\overline{p_u}$ for making prediction; (3) in case a new service user who did not invoke any Web services previously, but the target Web service has been invoked by other users ($\overline{p_u} = null \ \& \ \overline{p_i} \neq null$), we apply IMEAN $\overline{p_i}$ for making the prediction;

and (4) when $\overline{p_u} = null$ & $\overline{p_i} = null$, we will not provide any prediction since there is no information available.

3.4. Phase 4: Reliability Prediction

By these phases, we obtain the failure probability of the target Web service. To predict the Web service reliability, we adopt the commonly used exponential reliability function [Lyu 1996]:

$$R(t) = e^{-\beta t}, \tag{15}$$

where β (*failure-rate*) is the number of failures of the Web service during a certain time duration, and t is the time period for which the reliability is to be calculated. The value of β can be calculated by pf, where p is the failure probability of the Web service and f is the invocation frequency of the Web service (e.g., the number of invocations per hour). Therefore, we obtain the following equation for the reliability prediction of a Web service:

$$R(t) = e^{-pft}. \tag{16}$$

By these four-phase procedure, the designers of service-oriented systems are able to predict reliability of Web services early at the architectural design phase and select the optimal configuration. Moreover, the prediction of Web service reliability can be dynamically updated at runtime after the release of the system, which provides valuable information for dynamical system reconfiguration, load balancing, and fault tolerance needs [Lyu 1995].

3.5. Computational Complexity Analysis

This section discusses the upper bound on the worst-case computational complexity of the neighborhood-based reliability prediction algorithm. We assume there are m service users and n Web services.

3.5.1. Complexity of Similarity Computation. The computational complexity of the similarity computation between two users is $O(n)$, since there are at most n intersecting Web services between service user a and service user u. The computational complexity of the similarity computation between two Web services is $O(m)$, since there are at most m intersecting service users between Web service i and Web service j.

3.5.2. Complexity of UPCC. When predicting the missing values in the user-item matrix employing user-based PCC algorithm (Eq. (8)), for each user, we need to compute similarities of the current user with all the $m-1$ existing users (totally $m-1$ times of similarity computations). As discussed in Section 3.5.1, the computational complexity of each similarity computation is $O(n)$. Therefore, the computational complexity of the similarity computation of each user is $O(mn)$.

After the similarity computation, missing value prediction is conducted. For a user, there are at most n missing values. The computational complexity of each missing value prediction is $O(m)$ (at most $m-1$ similar users will be employed for the prediction). Therefore, the computational complexity of missing value prediction for a user is $O(mn)$.

Based on this analysis, the total computational complexity of UPCC (including similarity computation and missing value prediction) for a user is $O(mn)+O(mn) = O(mn)$. When predicting all the missing values in the user-item matrix, since there are totally m users, the computational complexity is $O(m^2n)$.

3.5.3. Complexity of IPCC. When predicting the missing values for a Web service employing item-based PCC algorithm (Eq. (11)), we need to compute similarities of the

Table II. Notations of Model-Based Reliability Prediction

Notations	Descriptions
P	the user-item matrix
p_{ij}	an entry in P, i.e., failure probability of Web service j observed by user i
W and H	two matrices which are used to approximate the user-item matrix P
W_i	the ith row of matrix W, representing the user-specific coefficients for user i
H_j	the jth column of matrix H, representing the factor vector for Web service j
n	the number of Web services
m	the number of users
l	the number of factors
I_{ij}^P	indicator function, equal to 1 if p_{ij} is available and equal to 0 otherwise
γ	a parameter for penalizing large values in the matrices W and H
$\|X\|_F^2$	The Frobenius norm of matrix X: $\|X\|_F^2 = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left\|x_{ij}\right\|^2}$
α	learning rate, which controls the speed of gradient descent iteration
ρ	the number of available entries in the user-item matrix P

current Web service with all the other $n-1$ Web services (totally $n-1$ similarity computations). As discussed in Section 3.5.1, the computational complexity of each similarity computation is $O(m)$. Therefore, the computational complexity of similarity computation is $O(mn)$ for a Web service.

After the similarity computation, for each missing value of the Web service, the value prediction computational complexity is $O(n)$, since at most $n-1$ similar Web services will be employed for the value prediction. There are at most m missing values for a Web service, so the computational complexity of value prediction for a Web service is $O(mn)$. Therefore, the computational complexity of IPCC for a Web service is $O(mn)+O(mn) = O(mn)$. When predicting all the missing values in the user-item matrix, since there are totally n Web services, the computational complexity is $O(mn^2)$.

3.5.4. Complexity of UIPCC. Since our UIPCC prediction approach is a linear combination of UPCC and IPCC, the computational complexity of our approach for predicting the missing values in the user-item matrix is $O(m^2n + mn^2)$.

4. MODEL-BASED RELIABILITY PREDICTION

The neighborhood-based approach proposed in Section 3 employs information of similar users (or items) to make reliability prediction. When the user-item matrix is very sparse, similar users or items can hardly be identified. Moreover, as discussed in Section 3.5, the computational complexity of the neighborhood-based approach is high. To address these drawbacks, in this section, we present a model-based approach for predicting the missing failure probability values in the user-item matrix. Our model-based approach is designed as a two-phase process. In Phase 1, the missing value problem is modeled as an optimization problem; and in Phase 2, an algorithm is proposed for solving the problem. These two phases will be presented in Section 4.1 and Section 4.2, respectively. Section 4.3 will analyze computational complexity of the model-based approach. Table II summarizes the notations of the model-based reliability prediction approach.

4.1. Matrix Factorization

As introduced in Section 2, the process of failure probability prediction usually includes a user-item matrix, where users are corresponding to rows and items (Web services) to columns. The matrix entries specify the failure probability values of each user on

each Web service. The problem we investigate is how to accurately predict the missing failure probability values in the user-item matrix by employing the available values.

Matrix factorization refers to a group of algorithms, where a matrix is factorized into a product of two matrices [Salakhutdinov and Mnih 2007]. When applying matrix factorization to address our research problem, the premise behind is that there is a small number of factors influencing the user-observed Web service failure probability, and that a user's observed failure probability on a Web service is determined by how each factor applies to the user. More specifically, a user's failure probability value on a Web service corresponds to a linear combination of these factors with user-specific coefficients.

Considering an $m \times n$ user-item matrix P, an *l-factor* model is attempted to find two matrices W (m rows and l columns) and H (l rows and n columns), such that:

$$P \approx WH, \tag{17}$$

where l is the number of factors. The number of columns of W and the number of rows of H are selected so the product of W and H will become an approximation to P. This matrix factorization procedure (i.e., decompose the user-item matrix P into two matrices W and H) has clear physical meanings: Each column of H is a factor vector including the values of the l factors for a Web service, while each row of W is the user-specific coefficients for a user. A user's Web service usage experience corresponds to the linear combination of the user-specific coefficients and the Web service factor vector.

For example, employing matrix factorization technique with four factors ($l = 4$), the user-item matrix in Figure 1 can be decomposed into two matrices W and H:

$$W = \begin{bmatrix} 0.135 & -0.747 & -0.430 & 0.245 \\ 0.281 & -1.049 & -0.527 & 0.489 \\ -0.029 & -0.622 & -0.224 & -1.135 \\ -0.434 & -0.404 & 0.637 & 0.921 \\ 0.209 & -0.423 & -0.995 & -0.303 \end{bmatrix},$$

$$H = \begin{bmatrix} -0.028 & -0.457 & -0.234 & 0.265 & 0.087 & -0.041 \\ 0.142 & 1.020 & 0.407 & 0.935 & 0.481 & 0.248 \\ -0.117 & 0.894 & 0.813 & -0.280 & 0.156 & 0.574 \\ 0.260 & -0.028 & 0.346 & -1.030 & 1.116 & 0.155 \end{bmatrix},$$

where each row of W includes four user-specific coefficients for a user and each column of H has values of the four factors for a Web service. The product of W and H is an approximation to the original user-item matrix P:

$$P = \begin{bmatrix} 0.5 & 0.2 & & 0.3 & & 0.4 \\ & 0.1 & & 0.2 & 0.5 & \\ 0.4 & & 0.3 & & 0.1 & \\ & 0.6 & & 0.1 & & \\ 0.5 & & 0.2 & & & 0.3 \end{bmatrix} \approx \begin{bmatrix} 0.501 & 0.229 & 0.354 & 0.311 & 0.465 & 0.401 \\ 0.508 & 0.157 & 0.320 & 0.221 & 0.496 & 0.378 \\ 0.412 & 0.312 & 0.306 & 0.655 & 0.168 & 0.388 \\ 0.530 & 0.582 & 0.684 & 0.165 & 0.710 & 0.605 \\ 0.493 & 0.197 & 0.243 & 0.562 & 0.337 & 0.325 \end{bmatrix} = WH.$$

From this introduction, we can see that the matrices W and H are unknown, and need to be estimated using the available values in the user-item matrix P. The matrices W and H are generally nonunique and we need to identify the optimal ones by minimizing the distance between WH and P. Since the user-item matrix P is only partially observed (e.g., not all users have the failure probability values of all Web

services), if we can find out the matrices W and H, the product of these two matrices can be employed to predict the unobserved entries in the user-item matrix P.

The most common measure of the discrepancy between P and WH is the sum-squared error, which can be calculated by:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{P}(p_{ij} - W_i H_j)^2, \tag{18}$$

where I_{ij}^{P} is the indicator function that is equal to 1 if the value p_{ij} is available in the user-item matrix (indicating that Web service j has been invoked by user i previously) and equal to 0 otherwise, W_i is the ith row of matrix W (representing the user-specific coefficients of user i), and H_j is the jth column of matrix H (representing the factor vector of Web service j).

The factorization problem can be stated as: Given a partially observed user-item matrix P, find two matrices W and H that minimize the sum-squared errors, which can be calculated by Eq. (18). Since the goal of finding W and H is to predict the unknown missing values in the matrix, these two matrices should avoid overfitting the observed vales, so that the resulting model has a good generalization performance for the unknown values. By adding the constraints of the norms of W and H to penalize large values of W and H, we have the following optimization problem:

$$\min_{W,H} \mathcal{L}(W,H) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{P}(p_{ij} - W_i H_j)^2 + \frac{\gamma}{2}\|W\|_F^2 + \frac{\gamma}{2}\|H\|_F^2, \tag{19}$$

where γ controls the extent of regularization for penalizing large values in the matrices W and H to avoid the overfitting problem, and $\|\cdot\|_F^2$ denotes the Frobenius norm [Golub and Loan 1996], which is defined as the square root of the sum of the absolute squares of values in a matrix. For example, $\|W\|_F^2$ (m rows and l columns) can be calculated by: $\|W\|_F^2 = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{l} |w_{ij}|^2}$. The optimization problem in Eq. (19) minimizes the sum-squared-errors objective function with quadratic regularization terms.

4.2. Gradient Descent

A local minimum of the objective function given by Eq. (19) can be found by performing gradient descent. Gradient descent is an optimization method for minimizing an objective function that is written as a sum of differentiable functions. The gradient descent algorithm loops through all available values in the user-item matrix and train the l factors one by one. One gradient step intends to decrease the square of prediction error of only one value.

The gradients can be computed by:

$$\frac{\partial \mathcal{L}}{\partial w_{il}} = \gamma w_{il} - \sum_{j=1}^{n} I_{ij}^{P}(p_{ij} - W_i H_j)h_{lj}, \tag{20}$$

$$\frac{\partial \mathcal{L}}{\partial h_{lj}} = \gamma h_{lj} - \sum_{i=1}^{m} I_{ij}^{P}(p_{ij} - W_i H_j)w_{il}, \tag{21}$$

where w_{il} is the lth coefficient of user i in the matrix W, h_{lj} is the lth factor of Web service j in the matrix H, W_i is the ith row of the matrix W, H_j is the jth column of the matrix H, and the prediction error is computed by $p_{ij} - W_i H_j$.

Algorithm 1: Gradient Descent Algorithm

Input: User-item matrix: P; number of factors: l
Output: User-specified coefficient matrix: W; Factor matrix: H

Initialize W and H with small random numbers;
repeat
  for *each factor* l **do**
    for *each available* p_{ij} *in* P **do**
      $w_{il}^{t+1} = w_{il}^{t} - \alpha \frac{\partial \mathcal{L}}{\partial w_{il}^{t}}$;
      $h_{lj}^{t+1} = h_{lj}^{t} - \alpha \frac{\partial \mathcal{L}}{\partial h_{lj}^{t}}$;
    end
  end
until *Converge* ;

Algorithm 1 shows the iterative process of the gradient descent algorithm for solving the optimization problem in Eq. (19). We first initialize the matrices W and H with small random values. Then, the algorithm iteratively updates the matrices W and H by a magnitude proportional to α in the opposite direction of the gradient. w_{il}^{t} means the value of w_{il} in the tth iteration. The parameter α is named *learning rate*, which controls the speed of iteration. The process stops when the local minimum of the objective function given in Eq. (19) is reached.

4.3. Complexity Analysis

The main computation of the gradient methods is to evaluate the objective function $\mathcal{L}$ and its gradients against the variables. Because of the sparsity of matrix P, the computational complexity of evaluating the objective function $\mathcal{L}$ is $O(\rho l)$, where ρ is the number of available entries in the matrix P and l is the number of factors. When performing gradient descent, the computational complexity is $O(\rho l)$ for each iteration, since there are totally ρ available entries and each entry requires $O(l)$ computation for updating the latent features. Therefore, the computational time of the model-based method is linear with respect to the number of observations in the user-item matrix P. This complexity analysis shows that the proposed model-based approach can scale to very large datasets when making Web service reliability prediction.

5. EXPERIMENTS

5.1. Dataset Description

To evaluate the prediction accuracy of various prediction approaches, we employ our real-world Web service QoS dataset[1] for conducting experiments. This dataset is collected in Feb. 2011, which includes 23 million Web service invocation results from 102 distributed service users on 3568 publicly available real-world Web services. We employ the distributed computers from Planet-Lab[2] to observe, collect, and contribute evaluation results of the target Web services to our centralized server, which is implemented by JDK, Eclipse, Axis2,[3] Apache Tomcat, Apache HTTP Server, and MySQL. Planet-Lab is a global research network that consists 1095 nodes at 531 sites. In our Web service evaluation, each user (i.e., a computer from Planet-Lab) conducts a number of invocations on each selected Web service and the invocation results are recorded.

[1]http://www.wsdream.net
[2]http://www.planet-lab.org
[3]http://ws.apache.org/axis2

The failure probability of a Web service obtained by a service user can thus be obtained. By processing the evaluation results, failure probabilities of the 3568 Web services observed by the 102 service users can be represented as a 102×3568 user-item matrix. The range of failure probability is from 0 to 100%.

In the experiments, we randomly removed entries from the user-item matrix and employed different prediction approaches to determine the missing values in the matrix. The removed values are used as ground truth to evaluate the prediction accuracy. *Matrix density* refers to the percentage of unremoved entries in the user-item matrix. By predicting missing values for the users in the matrix, their personalized Web service failure probability values are consequently obtained. In practice, a centralized server can be built to provide personalized reliability prediction service to users. First, a user requests personalized reliability prediction by sending observed Web service failure probability values to the server. Second, the proposed prediction approaches are employed by the server to make personalized reliability prediction on the uninvoked Web services for the user. Finally, the prediction results are sent to the user to help decision making.

5.2. Metrics

We use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics to measure the prediction accuracy of different prediction approaches. MAE is defined as:

$$MAE = \frac{\sum_{u,i} |p_{u,i} - \widehat{p}_{u,i}|}{N}, \tag{22}$$

where $p_{u,i}$ denotes the observed failure probability of Web service i observed by user u, $\widehat{p}_{u,i}$ is the predicted value, and N is the number of predicted values. MAE is the average of the differences between a prediction result and the corresponding observed value. MAE is a linear score, which means that all the individual differences are weighted equally in the average.

RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{u,i} (p_{u,i} - \widehat{p}_{u,i})^2}{N}}. \tag{23}$$

In RMSE, the differences between prediction results and the corresponding observed values are each squared and then averaged over the sample. Finally, the square root of the average is taken. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means RMSE is more applicable when large errors are particularly undesirable.

5.3. Comparison of Prediction Accuracy

To study the prediction accuracy of failure probability, we implement and compare seven prediction approaches.

— *UMEAN* (*User Mean*). This method employs a user's average failure probability value of the used Web services to predict the missing values of the unused Web services.
— *IMEAN* (*Item Mean*). This method employs the average failure probability value of the Web service observed by other service users to predict the failure probability value for a service user who never invoked this Web service previously.
— *UPCC* (*User-Based Collaborative Filtering Method using Pearson Correlation Coefficient*). This method is a classical method. It employs information of similar

Table III. MAE Performance Comparison

Methods	den=5%	den=10%	den=15%	den=20%	den=25%	den=30%
UMEAN	0.1861	0.1864	0.1867	0.1867	0.1866	0.1865
IMEAN	0.0504	0.0472	0.0462	0.0459	0.0455	0.0453
UPCC	0.0450	0.0386	0.0371	0.0363	0.0358	0.0353
IPCC	0.0491	0.0406	0.0373	0.0362	0.0345	0.0328
ICSE2010	0.0448	0.0384	0.0366	0.0357	0.0344	0.0327
UIPCC	0.0440	0.0371	0.0352	0.0335	0.0315	0.0301
MF	0.0420	0.0326	0.0288	0.0257	0.0245	0.0235

Table IV. RMSE Performance Comparison

Methods	den=5%	den=10%	den=15%	den=20%	den=25%	den=30%
UMEAN	0.2670	0.2666	0.2663	0.2663	0.2662	0.2663
IMEAN	0.1372	0.1262	0.1234	0.1221	0.1215	0.1210
UPCC	0.1081	0.0962	0.0938	0.0927	0.0921	0.0915
IPCC	0.1342	0.1071	0.0990	0.0939	0.0889	0.0841
ICSE2010	0.1080	0.0962	0.0935	0.0916	0.0885	0.0840
UIPCC	0.1057	0.0926	0.0897	0.0881	0.0854	0.0812
MF	0.1049	0.0816	0.0741	0.0677	0.0654	0.0633

users for the failure probability value prediction [Breese et al. 1998; Shao et al. 2007].

— *IPCC* (*Item-Based Collaborative Filtering Method using Pearson Correlation Coefficient*). This method is widely used in companies like Amazon.[4] It employs information of similar Web services (items) for the failure probability value prediction [Resnick et al. 1994; Sarwar et al. 2001].
— *ICSE2010*. This is our preliminary version presented in Zheng and Lyu [2010]. It systematically combines the UPCC and IPCC approaches for making missing failure probability value prediction.
— *UIPCC* (*User-Based and Item-Based Collaborative Filtering Method using Pearson Correlation Coefficient*). This is the neighborhood-based approach presented in Section 3. This approach is an enhanced version of its preliminary version (i.e., the ICSE2010 approach [Zheng and Lyu 2010]). The enhancements include: (1) employing a logistic function to overcome the PCC similarity overestimate problem (Eq. (4) and Eq. (5)), and (2) new weight computations of similar users and items (Eq. (10) and Eq. (12)).
— *MF* (*Matrix Factorization*). This is the model-based approach presented in Section 4. It factorizes the user-item matrix into two matrices and employs these two matrices for making missing value prediction.

Applying our new Web service QoS Dataset,[5] Table III and Table IV show the MAE and RMSE experimental results of different prediction approaches under different matrix density settings, respectively. As shown in Table III and Table IV, we can see the following.

— The MF prediction approach obtains the best prediction accuracy (smaller MAE or RMSE values) in all experimental settings. This is because the MF approach employs all the available information in the user-item matrix for making prediction,

[4]http://www.amazon.com
[5]http://www.wsdream.net

while the neighbor-based approaches only employ the information of similar neighbors (users or items) for making prediction.
— Compared with UPCC and IPCC, ICSE2010, and UIPCC achieve better prediction accuracy, indicating that prediction accuracy can be improved by systematically combining the information from similar users and similar items. Since different datasets have their own characteristics, the parameter λ can help our ICSE2010 and UIPCC approaches achieve optimal performance in different datasets.
— Compared with ICSE2010, UIPCC provides better prediction accuracy under different matrix density settings, indicating that our enhancements of the UIPCC approach is reasonable and effective.
— When the matrix density is increased from 5% to 30%, the prediction accuracies of our UIPCC and MF approaches are enhanced, since a denser matrix provides more information for the missing value prediction.
— When the matrix density is small (e.g., 5%, 10%, etc.), the prediction accuracy of UPCC is better than that of IPCC. When the matrix density is large (e.g., 25%, 30%, etc.), the prediction accuracy of IPCC is better than that of UPCC. This observation result indicates that information of similar users and similar items have different effect in different situations. By fusing on the information of similar users and similar items, our ICSE2010 and UIPCC approaches can achieve better prediction accuracy.

5.4. Neighborhood-Based Approach: Impact of Parameters

5.4.1. Impact of Enhanced PCC. PCC may overestimate the similarity between two users who happen to observe similar failure probability values on a small number of co-invoked Web services. Our enhanced PCC similarity computations (i.e., Eq. (4) and Eq. (5)) make the similarity computation more reasonable in practice by devaluing the similarities which look significant but are actually not. To study the performance of our enhanced PCC similarity computation, we implement two versions of UPCC, IPCC, ICSE2010, and UIPCC. One version employs traditional PCC for the similarity computation, while the other version employs our enhanced PCC for the similarity computation. In the experiment, we vary the density of the training matrix from 5% to 50% with a step value of 5%.

Figure 3(a) to Figure 3(d) show the experimental results of UPCC, IPCC, ICSE2010, and UIPCC, respectively. As shown in Figure 3, we can see the following.

— Enhanced PCC obtains better prediction accuracy consistently than traditional PCC in all the UPCC, IPCC, ICSE2010, and UIPCC prediction approaches.
— The cases of excluding dissimilar neighbors do not happen very often compared with the normal cases. Therefore in Figure 3, the improvement of excluding dissimilar neighbors is alleviated by the existence of many normal cases.

5.4.2. Impact of Enhanced Weight. When predicting failure probability values, weights are calculated for different similar users (or items) based on their similarity values, so that users (or items) with large similarity values will impose stronger influence on the missing value prediction. In this article, we propose an enhanced weight computation to enlarge the influence of users (or items) with higher similarity values by squaring the similarity values. To study the performance of our enhanced weight computation in Eq. (10), we implement two versions of UPCC, IPCC, ICSE2010, and UIPCC. One version employs linear weight computation (Eq. (9), proposed in our preliminary version [Zheng and Lyu 2010]), while the other version employs our enhanced weight computation (Eq. (10) and Eq. (12)). In the experiment, we vary the density of the training matrix from 5% to 50% with a step value of 5%.

Figure 4(a) to Figure 4(d) show the experimental results of UPCC, IPCC, ICSE2010, and UIPCC, respectively. As shown in Figure 4, we can see the following.

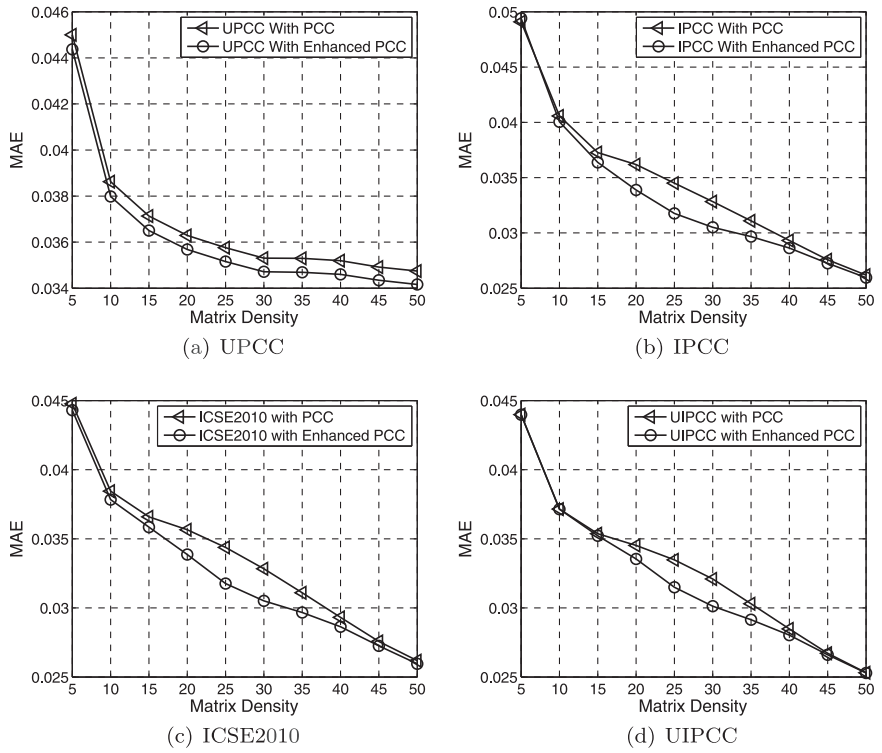


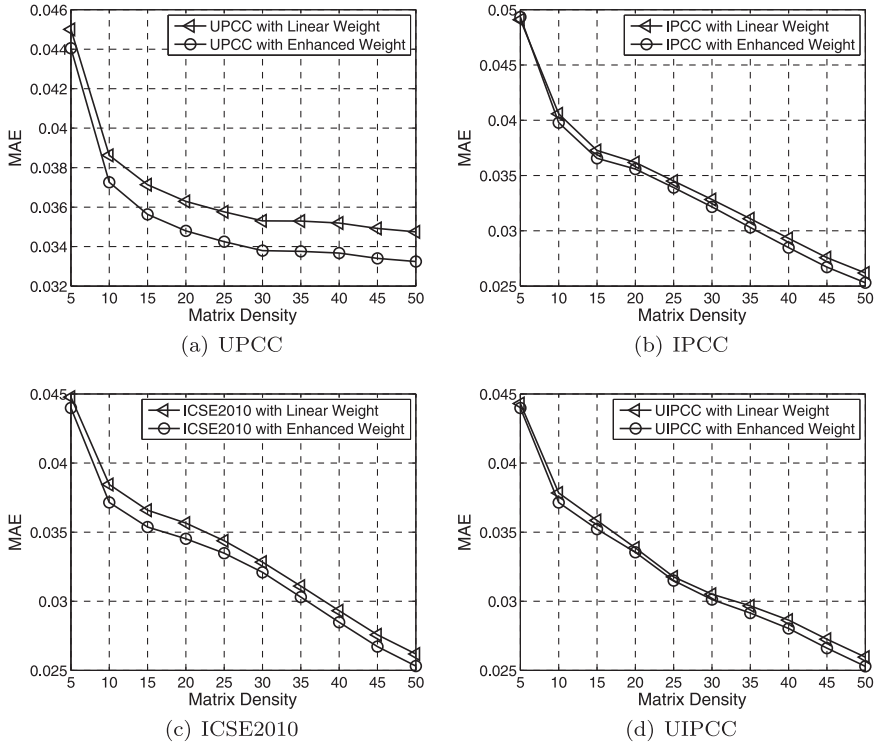
Fig. 3. Impact of enhanced PCC.

— Enhanced weight obtains better prediction accuracy consistently than linear weight in all the UPCC, IPCC, ICSE2010, and UIPCC prediction approaches under different matrix density settings.
— When the training matrix density increases from 5% to 50%, the prediction accuracy of all the UPCC, IPCC, ICSE2010, and UIPCC becomes better, since more similar users (or items) can be found in a denser user-item matrix for making missing value prediction.

5.4.3. Parameter λ. The experimental results of Table III and Table IV show that UPCC and IPCC have different effect in different situations, indicating that different datasets may inherit different data distribution and correlation characteristics. Our UIPCC approach is adaptable to different datasets by using a parameter λ. When $\lambda = 1$, we only extract information from the similar users. When $\lambda = 0$, we only consider the similar Web services. When $0 < \lambda < 1$, we combine the information from both similar users and similar Web services. To study the impact of λ on the prediction result, we implement our UIPCC approach and its preliminary version (named ICSE2010, which does not employ enhanced PCC and enhanced weight). We vary the value of λ from 0 to 1 with a step value of 0.1.

Figure 5(a) and 5(b) show the MAE results of 10% and 20% matrix densities. Figure 5 shows the following.

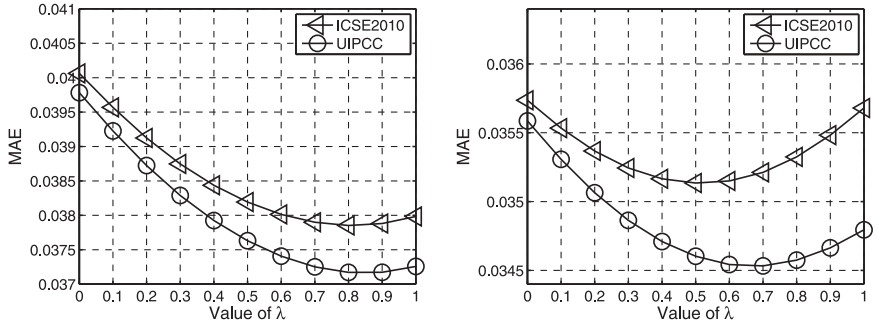
— The value of λ impacts the recommendation results significantly. Suitable λ values will provide better prediction accuracy, since it enables a proper combination of the user-based method and the item-based method.



(a) UPCC (b) IPCC

(c) ICSE2010 (d) UIPCC

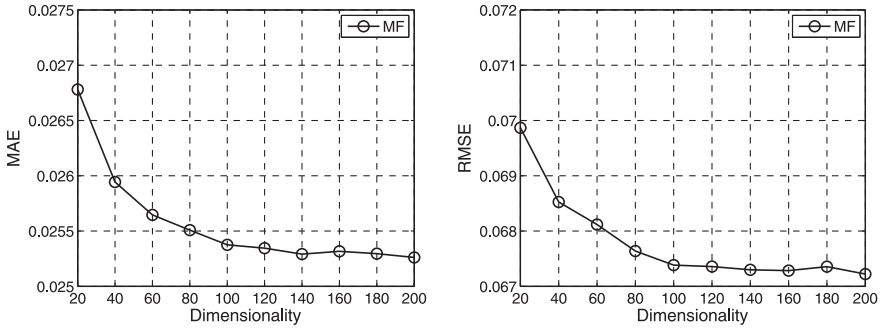
Fig. 4. Impact of enhanced weight.



(a) 10% Density (b) 20% Density

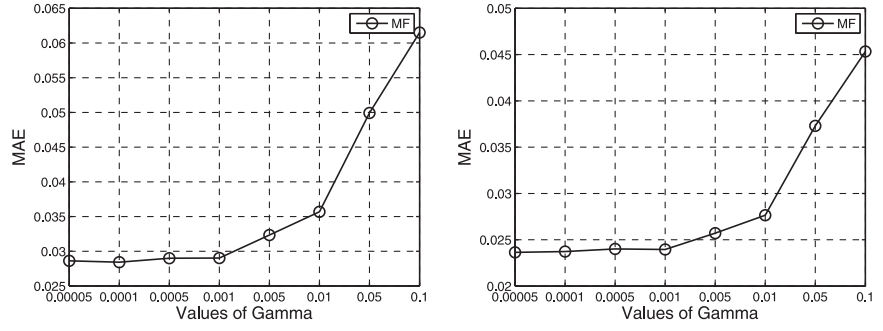
Fig. 5. Impact of parameter λ.

— The optimal λ values (the minimal MAE values of the curves in Figure 5) are different under different matrix density settings. For example, in Figure 5(a), the optimal λ value for UIPCC is 0.8, while the optimal λ value for UIPCC in Figure 5(b) is 0.7. This observation indicates that the optimal λ value is influenced by the matrix density. Information of similar users and similar Web services impose different effect in different matrix density settings.
— Our UIPCC approach outperforms ICSE2010 approach consistently under different λ value settings, indicating that our enhanced PCC and enhanced weight computations can provide better prediction accuracy.



(a) MAE of 20% Matrix Density (b) RMSE of 20% Matrix Density

Fig. 6. Impact of factor dimensionality.



(a) MAE of 10% Matrix Density (b) MAE of 20% Matrix Density

Fig. 7. Impact of parameter γ.

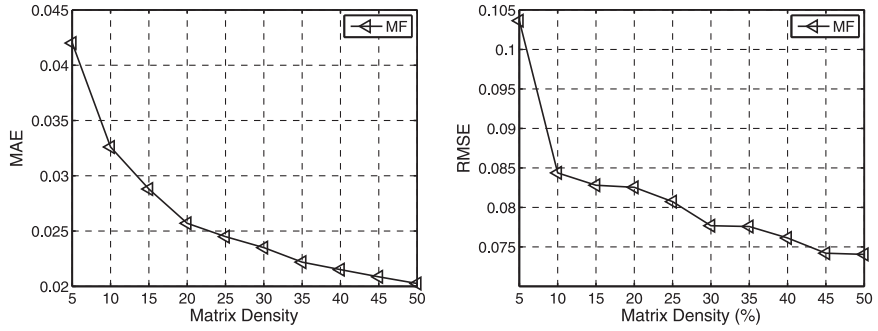
5.5. Model-Based Approach: Impact of Parameters

5.5.1. Impact of Dimensionality. Dimensionality refers to the number of factors when factorizing the user-item matrix. To study the impact of dimensionality, we vary the values of dimensionality from 10 to 100 with a step value of 10. We set γ=0.005 and matrix density=20% in this experiment.

Figure 6(a) and Figure 6(b) show the MAE and RMSE results of our model-based approach (named MF) under different dimensionality settings. As shown in Figure 6, the values of MAE and RMSE decrease when the dimensionality is increased from 20 to 200, indicating that a larger dimension value generates more accurate prediction results. However, as discussed in Section 4.3, larger dimensionality values will lead to longer computation time.

5.5.2. Impact of Parameter γ. The parameter γ is employed to avoid the overfitting problem by penalizing large values of the matrices W and H. To study the impact of parameter γ on the prediction results, we vary the values of γ from 0.00005 to 0.1. We set $dimensionality = 50$ in this experiment.

Figure 7(a) and Figure 7(b) show the MAE results of our MF method with 10% and 20% matrix densities, respectively. As shown in Figure 7, the values of MAE increase when the value of γ is increased, indicating that a smaller value of γ generates more accurate prediction results. When the value of γ is smaller than 0.001, the MAE performance results of different settings are similar. In our model-based approach, we can



(a) MAE (b) RMSE

Fig. 8. Impact of matrix density.

simply set the parameter γ to be a small value (e.g., 0.0001) to obtain good prediction accuracy.

5.5.3. Impact of Matrix Density. Matrix density refers to the percentage of available entries in the user-item matrix. To study the impact of the matrix density on the prediction results of our model-based approach, we vary the density from 5% to 50% with a step value of 5%.

Figure 8 shows the experimental results of our model-based approach (named MF) with different matrix density settings. As shown in Figure 8, the prediction accuracy of MF is enhanced when the matrix density is increased from 5% to 50%, because a denser matrix provides more information for making prediction. This observation is the same as the experimental results of the neighborhood-based approach. By obtaining more past failure data from different service users, both our neighborhood-based and model-based approaches can achieve more accurate prediction results.

6. NEIGHBOR-BASED APPROACH VS. MODEL-BASED APPROACH

This article proposes a neighborhood-based approach and a model-based approach for predicting the reliability of Web services. Advantages of the neighborhood-based approach include the following.

— *Simplicity*. Neighborhood-based approaches are widely employed in practice, because they are simple and intuitive on a conceptual level while avoiding the complications of a model-building stage. The neighborhood-based approaches can be learned and implemented quickly by developers without expert knowledge on personalized reliability prediction.
— *Explanations*. The research results in Herlocker et al. [2000] and Yu et al. [2004] indicate that allowing users to know more about the result-generating process can help them understand the prediction approaches. Such explanations can remove the black box process from the prediction approaches and provide transparency. The neighborhood-based approach presents a clear interpretation that can be easily conveyed to users, such as "You seem to be sharing similar past usage experience with user A, who have invoked the following *Web services*. . . ." The neighborhood-based approach demonstrates better explanations than the model-based approach, since it is simple and intuitive on a conceptual level. The benefits of providing better explanations include: (1) better understanding of the reasoning behind a prediction, so that users may decide the level of confidence to place in the prediction; (2) better understanding the strengths and limitations of the prediction approaches;

and (3) greater acceptance of the prediction results, since its limits and strengths are fully visible and the prediction results are justified.

— *Incremental Accommodation to New Data*. The prediction approaches must be able to handle new data, for example, new users or new items. Neighborhood-based approaches can easily accommodate to new users (or new items) by simply storing the new data. On the other hand, the model-based approaches need to train a factor model before making missing value prediction. Retraining the factor model with new data is quite expensive, especially if retraining needs to be performed regularly.

— *Prediction of Specific Missing Values*. When prediction of a specific missing value is needed, neighborhood-based approach can be easily employed since it calculates one missing value at a time. On the other hand, model-based approach needs to compute the whole matrix to train a factor model before making the missing value prediction.

On the other hand, advantages of the model-based approach include:

— *Accuracy*. As shown in Table III and Table IV, the prediction accuracy of the model-based approach is better than that of the neighborhood-based approaches when employing our dataset. The model-based approach takes advantages of using all the available values in the user-items matrix for making prediction, while the neighborhood-based approaches only employ information of similar users and similar items.

— *Computational Complexity*. As introduced in Section 4.3, the computational complexity of the model-based method is linear with respect to the number of available values in the user-item matrix, which is better than the neighborhood-based approach (e.g., as introduced in Section 3.5, the computational complexity of UIPCC is $O(m^2n + mn^2)$).

In summary, the neighborhood-based approaches can be applied to situations such as agile development, small-scale data, dynamic environment where incrementally accommodating to new data and prediction of specific missing values are required. On the other hand, the model-based approaches are suitable for situations where the requirements for prediction accuracy and computational time are high.

7. RELATED WORK

7.1. Software Reliability Prediction

Software reliability prediction is an important research problem in software reliability engineering [Lyu 1996; Lyu and Nikora 1992]. A great deal of software reliability prediction models have been proposed for stand-alone software systems, such as Musa's execution time model [Musa et al. 1990], Putnam's model [Putnam and Myers 1992], Rome laboratory models [Friedman et al. 1992], Jelinski's model [Jelinski and Moranda 1972], Littlewood's models [Littlewood et al. 1986], structure-based analysis [Gokhale and Lyu 2005], Poisson Process Models [Huang et al. 2003], Support Vector Machine based model [Xing et al. 2005], and so on. With the popularization of service computing, a number of reliability prediction approaches are proposed for component-based systems and service-oriented systems [Cardoso et al. 2002; Cheung 1980; Cortellessa and Grassi 2007b; Gokhale and Trivedi 2002; Goseva-Popstojanova and Trivedi 2001; Grassi and Patella 2006; Yacoub et al. 1999]. Cortellessa and Grassi [2007a] analyzed impact of error propagation on the reliability of component-based systems. Filieri et al. [2010] presented a novel approach to the reliability modeling and analysis of component-based systems, which permits component-level reliability specification. Koziolek et al. [2010] applied different reliability analysis methods to a large-scale industrial control system. Most of these previous approaches focus on system-level reliability compositional analysis.

A few approaches [Cheung et al. 2008; Goseva-Popstojanova et al. 2003; Reussner et al. 2003], which consider component-level reliability prediction, are mainly designed for traditional component-based systems. Goseva-Popstojanova et al. [2003] calculated reliability risk of a component by the component complexity and the levels of its failures. Reussner et al. [2003] computed component reliability by the reliabilities of its services, which are assumed to be known. Both these approaches require internal information of the components for making component reliability prediction. Hamlet [2009] implemented tools to approximate the behavior of software components, then engaged these approximations to estimate the properties of component-based systems. The system is neither built, nor is any test execution performed, as the system designers do not require the component sources or the executables as long as historical data of the components are available. Similar to Hamlet's work, in this article, service reliability is not assessed directly but computed from the collected data. Cheung et al. [2008] predicted the component reliability at architecture design phase by exploiting behavioral models from sources of information available at design time. Different from the work in Cheung et al. [2008], which tried to predict reliability of a component that was not even implemented yet, our work targets at predicting reliabilities of remote Web services that are implemented and provided by other organizations. Moreover, the work in Cheung et al. [2008] did not investigate personalized reliability prediction for users since the components were assumed to be invoked locally in the traditional component-based systems. Our approach focuses on personalized reliability prediction for different service users, which significantly enhances the reliability prediction accuracy of Web services and service-oriented systems.

Tsai [2005] proposed a data-driven reliability evaluation process (named DREP). DREP computes reliability in a collaborative manner by employing current data just collected and profiled from participants (e.g., service users, service developers, and service brokers). Bertolino and Polini [2009] provided a first abstract definition of an SOA testing governance notion, where remote stakeholders collaborated with each other to achieve the common utility in validation. In this article, we focus on collaboration among service users for sharing past failure data of Web services and propose two approaches for making personalized service reliability prediction.

7.2. Collaborative Filtering

Collaborative filtering is the process of filtering for information with techniques involving collaboration among multiple agents. Various collaborative filtering techniques are widely adopted in recommender systems [Burke 2002; Ma et al. 2007]. Two types of collaborative filtering approaches are widely studied: neighborhood-based (memory-based) and model-based.

The most investigated neighborhood-based approaches include user-based approaches [Breese et al. 1998], item-based approaches [Sarwar et al. 2001], and their fusion [Wang et al. 2006a; Zheng and Lyu 2010]. User-based approaches employ similar users in making prediction, while item-base approaches employ similar items for making prediction. In the environment of service computing [Zhang et al. 2007], it is possible to collect and make use of the past Web service failure data from different service users for making missing value prediction. To take advantages of these collected data, Section 3 of this article applies the neighborhood-based collaborative filtering techniques [Breese et al. 1998; Ma et al. 2007; Sarwar et al. 2001; Zheng et al. 2009, 2011] to attack the challenging research problem of reliability prediction of service-oriented systems.

In the model-based approaches, on the other hand, training datasets are used to train a predefined model. Examples of model-based approaches include the clustering model [Xue et al. 2005], aspect models [Hofmann 2004], etc. Recently, several matrix

factorization methods [Ma et al. 2009; Rennie and Srebro 2005; Salakhutdinov and Mnih 2007, 2008] have been proposed for collaborative filtering. The matrix factorization methods decompose the user-item matrix into two low-dimensional matrices. The premise behind a low-dimensional factor model is that there is only a small number of factors that will influence the values in the user-item matrix. We have presented a matrix factorization approach for predicting the failure probability of Web services in Section 4.

8. CONCLUSION

In this article, we propose two personalized reliability prediction approaches for Web services. The main idea is to exploit the past failure data of service users in predicting Web service failure probabilities. The comprehensive experimental analysis shows the effectiveness of our reliability prediction approaches.

In our current design, our reliability prediction approaches mainly focus on Web services, since Web services are usually publicly accessible and invoked by a lot of service users. Therefore, it is possible to employ the Web service failure data from different service users in making personalized reliability prediction. Our approach, nevertheless, can be extended to other services or systems such as component-based systems, distributed computing platforms, and cloud computing platforms, in case reusable components are employed by other users previously and the past failure data can be obtained from these users. In our future work, we plan to extend our proposed reliability prediction approaches to cloud platforms, where software components and component users (usually other software components that invoke the current component) are all distributed in the same cloud infrastructure. The component failure data can be easily collected by the cloud infrastructure providers for making reliability prediction.

This article focuses on reliability prediction of Web services. Despite the focus of reliability, the proposed neighborhood-based and model-based approaches can be extended to the prediction of other user-received QoS properties (e.g., response time, throughput, etc.). Our further work also includes exploring failure correlation between different Web services, collecting past Web service failure data from more service users, investigating the combination of the neighborhood-based and model-based prediction approaches to achieve more accurate prediction results, and employing location information of users and Web services to build more accurate reliability prediction models.

REFERENCES

ANSI/IEEE. 1991. Standard glossary of software engineering terminology. ANSI/IEEE STD-729-1991.

Bertolino, A. and Polini, A. 2009. SOA test governance: Enabling service integration testing across organization and technology borders. In *Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation Workshops*. 277–286.

Breese, J. S., Heckerman, D., and Kadie, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI'98)*. 43–52.

Burke, R. 2002. Hybrid recommender systems: Survey and experiments. *User Model. User-Adap. Interact. 12,* 4, 331–370.

Cardoso, J., Miller, J., Sheth, A., and Arnold, J. 2002. Modeling quality of service for workflows and web service processes. *J. Web Semant. 1*, 281–308.

Cheung, L., Roshandel, R., Medvidovic, N., and Golubchik, L. 2008. Early prediction of software component reliability. In *Proceedings of the 30th International Conference on Software Engineering (ICSE'08)*. 111–120.

Cheung, R. C. 1980. A user-oriented software reliability model. *IEEE Trans. Softw. Eng. 6,* 2, 118–125.

Cortellessa, V. and Grassi, V. 2007a. A modeling approach to analyze the impact of error propagation on reliability of component-based systems. In *Proceedings of the 10th International Symposium on Component Based Software Engineering (CBSE'07)*. 140–156.

Cortellessa, V. and Grassi, V. 2007b. Reliability modeling and analysis of service-oriented architectures. In *Test and Analysis of Web Services*, 339–362.

Filieri, A., Ghezzi, C., Grassi, V., and Mirandola, R. 2010. Reliability analysis of component-based systems with multiple failure modes. In *Proceedings of the 13th International Symposium on Component Based Software Engineering (CBSE'10)*. 1–20.

Friedman, M., Tran, P., and Goddard, P. 1992. Reliability techniques for combined hardware and software systems. Tech. rep. RL-TR-92-15, Rome Laboratory.

Gokhale, S. S. and Lyu, M. R.-T. 2005. A simulation approach to structure-based software reliability analysis. *IEEE Trans. Softw. Eng. 31*, 643–656.

Gokhale, S. S. and Trivedi, K. S. 2002. Reliability prediction and sensitivity analysis based on software architecture. In *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE'02)*. 64–78.

Golub, G. H. and Loan, C. F. V. 1996. *Matrix Computations* 3rd Ed. The Johns Hopkins University Press.

Goseva-Popstojanova, K. and Trivedi, K. S. 2001. Architecture-based approach to reliability assessment of software systems. *Perf. Eval. 45,* 2–3, 179–204.

Goseva-Popstojanova, K., Hassan, A., Abdelmoez, W., Nassar, D. E. M., Ammar, H., and Mili, A. 2003. Architectural-level risk analysis using uml. *IEEE Trans. Softw. Eng. 29,* 10, 946–960.

Grassi, V. and Patella, S. 2006. Reliability prediction for service-oriented computing environments. *IEEE Internet Comput. 10,* 3, 43–49.

Hamlet, D. 2009. Tools and experiments supporting a testing-based theory of component composition. *ACM Trans. Softw. Eng. Methodol. 18*, 12:1–12:41.

Herlocker, J. L., Konstan, J. A., and Riedl, J. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW'00)*. 241–250.

Hofmann, T. 2004. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst. 22,* 1, 89–115.

Huang, C.-Y., Lyu, M., and Kuo, S.-Y. 2003. A unified scheme of some nonhomogenous poisson process models for software reliability estimation. *IEEE Trans. Soft. Eng. 29,* 3, 261–269.

Jelinski, Z. and Moranda, P. 1972. Software reliability research. In *Proceedings of the Statistical Methods for the Evaluation of Computer System Performance*. 465–484.

Koziolek, H., Schlich, B., and Bilich, C. 2010. A large-scale industrial case study on architecture-based software reliability analysis. In *Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering (ISSRE'10)*. 279–288.

Littlewood, B., Abdel-Ghaly, A., and Chan, P. 1986. *Tools for the Analysis of the Accuracy of Software Reliability Predictions*. Springer-Verlag, Heidelberg.

Lyu, M. and Nikora, A. 1992. Applying reliability models more effectively [software]. *Software, IEEE 9,* 4, 43–52.

Lyu, M. R. 1995. *Software Fault Tolerance*. Trends in Software, Wiley.

Lyu, M. R. 1996. *Handbook of Software Reliability Engineering*. McGraw-Hill, New York.

Lyu, M. R. 2007. Software reliability engineering: A roadmap. *Prof. Future of Software Engineering, Int'l Conf. Software Engineering (ICSE'07) 0*, 153–170.

Ma, H., King, I., and Lyu, M. R. 2007. Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*. 39–46.

Ma, H., King, I., and Lyu, M. R. 2009. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*. 203–210.

Musa, J. D. 1993. Operational profiles in software-reliability engineering. *IEEE Softw. 10*, 14–32.

Musa, J. D., Iannino, A., and Okumoto, K. 1990. *Software Reliability: Measurement, Prediction, Application*. McGraw-Hill, New York.

Putnam, L. H. and Myers, W. 1992. *Measures for Excellence: Reliable Software on Time, Within Budget*. Prentice-Hall.

Rennie, J. D. M. and Srebro, N. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*. 713–719.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. 175–186.

Reussner, R. H., Schmidt, H. W., and Poernomo, I. H. 2003. Reliability prediction for component-based software architectures. *J. Syst. Softw. 66,* 3, 241–252.

Salakhutdinov, R. and Mnih, A. 2007. Probabilistic matrix factorization. In *Proceedings of the Advances in Neural Information Processing Systems*. 1257–1264.

Salakhutdinov, R. and Mnih, A. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. 880–887.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*. 285–295.

Scholz, A., Buckl, C., Kemper, A., Knoll, A., Heuer, J., and Winter, M. 2008. Ws-amuse - web service architecture for multimedia services. In *Proceedings of the 30th International Conference on Software Engineering (ICSE'08)*. 703–712.

Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., and Mei, H. 2007. Personalized qos prediction for web services via collaborative filtering. In *Proceedings of the 5th International Conference on Web Services (ICWS'07)*. 439–446.

Shardanand, U. and Maes, P. 1995. Social information filtering: Algorithms for automating word of mouth. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

ter Beek, M. H., Gnesi, S., Koch, N., and Mazzanti, F. 2008. Formal verification of an automotive scenario in service-oriented computing. In *Proceedings of the 30th International Conference on Software Engineering (ICSE'08)*. 613–622.

Tsai, W. T. 2005. Service-oriented system engineering: A new paradigm. In *Proceedings of the IEEE International Workshop on Service-Oriented System Engineering (SOSE'05)*. 3–8.

Tsai, W.-T., Zhou, X., Chen, Y., and Bai, X. 2008. On testing and evaluating service-oriented software. *IEEE Comput. 41,* 8, 40–46.

Wang, J., de Vries, A. P., and Reinders, M. J. 2006a. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*. 501–508.

Wang, W.-L., Pan, D., and Chen, M.-H. 2006b. Architecture-based software reliability modeling. *J. Syst. Softw. 79,* 1, 132–146.

Xing, F., Guo, P., and Lyu, M. 2005. A novel method for early software quality prediction based on support vector machine. In *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)*. 213–222.

Xue, G., Lin, C., Yang, Q., Xi, W., Zeng, H., Yu, Y., and Chen, Z. 2005. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*. 114–121.

Yacoub, S. M., Cukic, B., and Ammar, H. H. 1999. Scenario-based reliability analysis of component-based software. In *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE'99)*. 22–31.

Ye, C., Cheung, S. C., and Chan, W. K. 2006. Publishing and composition of atomicity-equivalent services for B2B collaboration. In *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*. 351–360.

Yu, K., Schwaighofer, A., Tresp, V., Xu, X., and Kriegel, H.-P. 2004. Probabilistic memory-based collaborative filtering. *IEEE Trans. Knowl. Data Eng. 16,* 1, 56–69.

Zhang, L.-J., Zhang, J., and Cai, H. 2007. *Services Computing*. Springer and Tsinghua University Press.

Zheng, Z. and Lyu, M. R. 2010. Collaborative reliability prediction for service-oriented systems. In *Proceedings of the IEEE/ACM 32nd International Conference on Software Engineering (ICSE'10)*. 35–44.

Zheng, Z., Ma, H., Lyu, M. R., and King, I. 2009. WSREC: A collaborative filtering based web service recommender system. In *Proceedings of the 7th International Conference on Web Services (ICWS'09)*. 437–444.

Zheng, Z., Ma, H., Lyu, M. R., and King, I. 2011. Qos-aware web service recommendation by collaborative filtering. *IEEE Trans. Serv. Comput. 4,* 2, 140–152.

Received April 2011; revised October 2011, January 2012; accepted January 2012