# Collaborative Reliability Prediction of Service-Oriented Systems

Zibin Zheng
Dept. of Computer Science & Engineering
The Chinese University of Hong Kong
Hong Kong, China
zbzheng@cse.cuhk.edu.hk

Michael R. Lyu
Dept. of Computer Science & Engineering
The Chinese University of Hong Kong
Hong Kong, China
lyu@cse.cuhk.edu.hk

## ABSTRACT

Service-oriented architecture (SOA) is becoming a major software framework for building complex distributed systems. Reliability of the service-oriented systems heavily depends on the remote Web services as well as the unpredictable Internet. Designing effective and accurate reliability prediction approaches for the service-oriented systems has become an important research issue. In this paper, we propose a collaborative reliability prediction approach, which employs the past failure data of other similar users to predict the Web service reliability for the current user, without requiring real-world Web service invocations. We also present a user-collaborative failure data sharing mechanism and a reliability composition model for the service-oriented systems. Large-scale real-world experiments are conducted and the experimental results show that our collaborative reliability prediction approach obtains better reliability prediction accuracy than other approaches.

## Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification—*Reliability*

## General Terms

Design, Reliability

## Keywords

Reliability Prediction, User-Collaboration, Web Service, Reliability Composition

## 1. INTRODUCTION

Service-oriented architecture (SOA) is becoming a major framework for building versatile and complex distributed systems by discovering and integrating Web services provided by different organizations. SOA has been widely used in e-business, e-government and also a lot of other domains, such as automotive systems [28], multimedia services [26], etc. The ability to predict reliability of service-oriented systems early at the architecture design phase can help to reduce re-engineering cost and to produce more reliable systems.

Software reliability prediction is a task to determine future reliability of software systems based on the past failure data [17]. Various software reliability prediction models, such as Musa's execution time model [20], Putnam's model [22], Rome Laboratory model [10], and so on, have been proposed for predicting software reliability by observing, accumulating, and analyzing previous failure data. 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 the customers or end users. However, reliability of a service-oriented system not only relies on the system itself, but also heavily depends on the remote Web services and the unpredictable Internet. Influenced by communication links, difference service users may experience quite different reliability performance on the same Web service. Web service evaluation [29] from the client-side is usually required for assessing performance of target Web services. However, traditional exhaustive testing becomes difficult and sometimes even impossible for the service-oriented systems due to: (1) Web service invocations may be charged since the Web services are owned and hosted by other organizations. Even if the Web service invocations are free, executing a large number of Web service invocations imposes costs for the service users and consumes resources of the service providers; and (2) it is time-consuming and expensive to conduct evaluation on all the service candidates, since there may exist a lot of alternative Web service candidates in the Internet with similar or identical functionalities. However, without comprehensive evaluation, we cannot collect sufficient past failure data of the Web service components. It is thus difficult for the system designer to determine whether the service-oriented system is reliable enough for release.

The previous reliability prediction methods for component-based systems [7, 11, 13, 31] and service-oriented systems [4, 14] are mainly focused on system-level compositional analysis. These approaches assume that reliabilities of the individual components/Web services are known. The detailed procedures for obtaining the component reliabilities are usually ignored. A few approaches [6, 12, 23], which consider component-level reliability prediction, are mainly designed for traditional component-based systems. We argue that reliability prediction for service-oriented systems is more chal-


*ICSE* '10, May 2-8 2010, Cape Town, South Africa

lenging, because: (1) there is a lack of internal information of the service components, since the remote Web services are provided by other organizations without any design and implementation details; and (2) different service users may obtain different performance from a same Web service, since the Web service performance is greatly influenced by communication links.

Without sufficient past failure data, without internal information of the service components, and influenced by the unpredictable communication links, it is thus much more difficult to make accurate reliability prediction on the service-oriented systems than traditional component-based systems. To attack this challenge, we propose a collaborative reliability prediction approach. The idea is that the Web service reliability for the current service user (service user is usually designer of the service-oriented system, not end-user of the system) can be predicted by employing past failure data of other similar service users. A service user is selected as a similar user with the current user if he/she has experienced similar reliability performance on the same set of commonly-invoked Web services. By our approach, reliabilities of Web service components can be predicted even if the current user does not invoke these components previously and has no idea on their internal information.

Complementary to the existing reliability prediction approaches [4, 7, 11, 13, 14, 31], which mainly focus on system level compositional analysis, this paper focuses on both component level reliability prediction and system level reliability compositional analysis. The service component reliabilities obtained from our framework can be employed by other previous reliability prediction approaches. Our reliability prediction framework can be employed at the early phase of system development to help produce more reliable system. More importantly, it can also be employed at runtime to enable optimal system reconfiguration.

The contributions of this paper are twofold:

- Firstly, a collaborative framework is proposed for predicting reliability of service-oriented systems. Different from previous reliability prediction approaches, our approach employs past failure data of similar service users for making reliability prediction for the current service user.

- Secondly, extensive experiments are conducted using real-world Web service dataset, which contains 1.5 millions real-world Web service invocation results from 150 distributed service users on 100 real-world Web services. Our real-world Web service dataset has been published online[1], which provides detailed experimental information for future research and makes our experiments reproducible.

The rest of this paper is organized as follows: Section 2 introduces a collaborative framework. Section 3 presents our collaborative reliability prediction approach. Section 4 describes our implementation and experiments. Section 5 introduces related work and Section 6 concludes the paper.

## 2. COLLABORATIVE FRAMEWORK

In this paper, *failure probability* of a Web service is defined as the probability that an invocation to a Web service will fail. The value of failure probability is in the interval of 0 and 1. Figure 1 shows an example of a simple service-oriented system which is presented as a *service flow*. The service flow in Figure 1 includes a set of abstract tasks ($t_1$,...,$t_6$) with certain control flow structures (*sequence, loop*, and *parallel*). For each abstract task, an optimal candidate will be selected from a set of functionally equivalent Web service candidates. An *execution plan* is obtained by composing the selected Web services ($s_1$,...$s_6$), which will be invoked to implement the abstract tasks. Similar to work [32], we assume that the alternative service candidates are available. The problem of obtaining the functionally equivalent Web service candidates, which has been discussed by a lot of previous work [24, 33], is out of the scope of this paper.

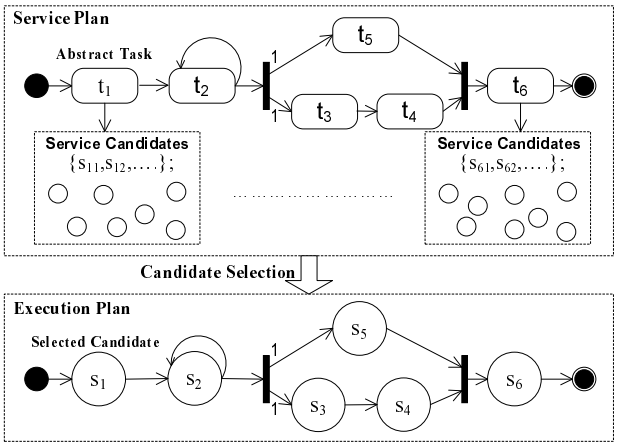



**Figure 1: Service-Oriented System Example**

Reliability of the service-oriented system highly depends on the selected Web services. Our approach employs past failure data from other similar service users to predict the failure probabilities of Web services for the current service user. Therefore, mechanisms are required for enabling past failure-data collection from different service users.

A user-collaborative mechanism is proposed in our previous work [39] for Web service recommendation. This mechanism can also be applied to the Web service past failure data collection. The design of this user-collaborative mechanism is inspired by the recent success of *YouTube*[2] and *Wikipedia*[3]. The key idea is that, instead of contributing videos (*YouTube*) or knowledge (*Wikipedia*), the service users are encouraged to contribute their individually observed Web service past failure data. By contributing more past failure data on the used Web services, the service users can obtain more accurate failure probability prediction of the Web services which they do not use before, since more characteristics of the current service user can be discovered from the provided data. By this way, the service users are encouraged to contribute their observed past failure-data.

To speedup the process of Web service failure data contributions, client-side middleware can be designed for the service users to automatically record the failure information of Web service invocations and contribute this information to a centralized server to exchange for failure probability prediction services. Detailed middleware design can be re-

[1]http://www.wsdream.net

[2]http://www.youtube.com

[3]http://www.wikipedia.org

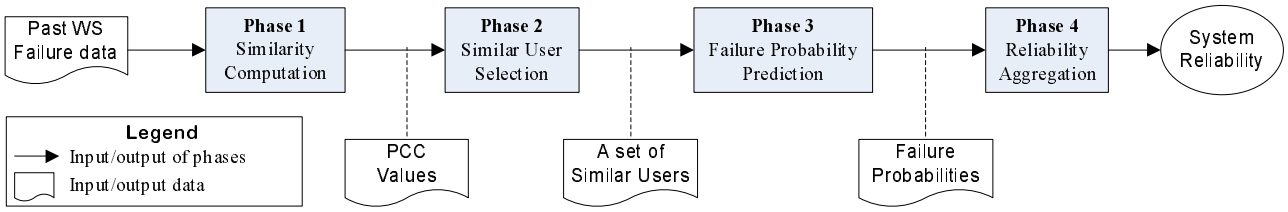



**Figure 2: Collaborative Reliability Prediction Procedures**

ferred to our previous work [36].

# 3. RELIABILITY PREDICTION

For an abstract task in a service flow, an optimal Web service needs to be identified from a set of functionally equivalent candidates for executing the task. The most straightforward way is to comprehensively evaluate all the service candidates and select out the best performing one. However, as discussed in Section 1, it is time-consuming, expensive, and sometimes even impossible to make comprehensive evaluation on all the candidates.

Another approach is to employ the Web service past failure data observed by other service users for predicting the Web service performance for the current user. For example, for the new service users, who do not invoke any Web services previously, the performance of a Web service can be indicated by its *average failure probability* ($\overline{p_i}$), which is defined as:

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

where $p_{a,i}$ is failure probability of Web service candidate $i$ observed by the service user $a$, $m$ 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 probability* values of the service candidates, the best performing candidate can be determined.

However, since service users are in different geographic locations and are under different network conditions, the current user may not be able to experience similar failure probability performance as the *average failure probability*. In case that the service users have invoked some Web services previously, we can take advantage of the past invocation information to enable more accurate failure probability prediction on the uninvoked Web services. Our collaborative reliability prediction approach is designed as a four-phase process as shown in Figure 2. In phase 1, we calculate the similarity of the service users with the current user based on the past failure data of their commonly-invoked Web services. Then, in Phase 2, a set of similar users are identified. After that, in phase 3, the invocation failure probabilities of Web services are predicted for the current user by using the invocation failure probabilities observed by similar users. Finally, in phase 4, the reliability of the service-oriented system is predicted by aggregating the failure probabilities of the service components. Details of these phases are presented at Section 3.1 to Section 3.4, respectively.

## 3.1 Phase 1: Similarity Computation

We assume that there are $m$ service users, $n$ Web services, and the relationship between users and Web services is denoted by an $m \times n$ matrix. Each entry $p_{a,i}$ in the matrix represents the failure probability of Web service $i$ observed by the service user $a$. $p_{a,i} = null$ if user $a$ did not invoke Web service $i$ before. Pearson Correlation Coefficient (PCC), which has been widely used in a number of recommendation systems [27], is employed for the similarity computation. PCC employs the following equation to computes the similarity between service user $a$ and service user $u$ based on their commonly invoked Web services:

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

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

Similar to the above approach, we also employ PCC 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 Equation (1).

## 3.2 Phase 2: Similar User Selection

After calculating and ranking the PCC similarity values between the current user and other users, a set of similar users can be identified by setting a parameter Top-K, which indicates that $k$ users, which have larger PCC values than others, will be selected as similar users. In reality, a service user may have limited number of similar users and the dissimilar users (e.g., with negative PCC value) may be selected when the number of similar users is less than $k$. However, including dissimilar users will greatly influence 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

by:

$$S(u) = \{a | Sim(u,a) \geq Sim_k, Sim(u,a) > 0, a \neq u\}, \quad (4)$$

where $Sim_k$ is the $k^{th}$ largest PCC value with the current user $u$, $Sim(u,a) > 0$ is to exclude the dissimilar users, and $Sim(a,u)$ can be calculated by Equation (2). A set of similar Web services $S(i)$ with the current Web service $i$ can also be identified by:

$$S(i) = \{k | Sim(i,k) \geq Sim_k, Sim(i,k) > 0, k \neq i\}, \quad (5)$$

where $Sim_k$ is the $k^{th}$ largest PCC value with the current Web service $i$ and $Sim(k,i)$ can be computed by Equation (3).

## 3.3 Phase 3: Failure Probability Prediction

Employing the similar users, the user-based approaches [2] (named as *UPCC*) predict the missing value $p_{u,i}$ by the following equation:

$$p_{u,i} = \overline{p_u} + \sum_{a \in S(u)} w_a \times (p_{a,i} - \overline{p_a}), \quad (6)$$

where $\overline{p_u}$ and $\overline{p_a}$ are average failure probabilities of different Web services observed by user $u$ and $a$, respectively, and $w_a$ is the significant weight of the similar user $a$, which is defined as:

$$w_a = \frac{Sim(a,u)}{\sum_{b \in S(u)} Sim(b,u)}. \quad (7)$$

Equation 7 ensures that a user with higher similarity has stronger influence on the value prediction (larger weight $w_a$). Similar to the user-based approach, employing the similar Web services $S(i)$, item-based approaches [25] (named as *IPCC*) predict the missing value $p_{u,i}$ by:

$$p_{u,i} = \overline{p_i} + \sum_{k \in S(i)} w_k \times (p_{u,k} - \overline{p_k}), \quad (8)$$

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

$$w_k = \frac{Sim(i,k)}{\sum_{j \in S(i)} Sim(i,j)}. \quad (9)$$

The predicted values by Eq. (6) and Eq. (8) may not fall in the probability range of 0 to 1. 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 $m \times n$ failure probability 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, the prediction results by the user-based approach (Equation (6)) and the item-based approach (Equation (8)) can be combined to fully utilize the information of both similar users and similar Web services [39]. When $S(u) \neq \emptyset \wedge S(i) \neq \emptyset$, we employ both the similar users and similar Web services to predict the missing value by employing the following equation:

$$p_{u,i} = \lambda \times (\overline{p_u} + \sum_{a \in S(u)} w_a \times (p_{a,i} - \overline{p_a})) + (1-\lambda) \times (\overline{p_i} + \sum_{k \in S(i)} w_k \times (p_{u,k} - \overline{p_k})), \quad (10)$$

where $\lambda$ ($0 \leq \lambda \leq 1$) is a user-defined parameter for determining how much the missing value prediction relies on the similar users or the similar Web services.

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 $S(u) \neq \emptyset \wedge S(i) = \emptyset$, our approach only employs the information of similar users for making prediction. In this case, our approach degrades to the user-based approach as shown in Equation (6). When $S(u) = \emptyset \wedge S(i) \neq \emptyset$, we only use the information of similar items for making prediction. In this case, our approach degrades to the item-based approach as shown in Equation (8). When $S(u) = \emptyset \wedge S(i) = \emptyset$, there are no similar users or similar Web services. Consequently, we predict the $p_{u,i}$ by employing the following equation:

$$p_{u,i} = \begin{cases} \lambda \times \overline{p_u} + (1-\lambda) \times \overline{p_i}, & \overline{p_u} \neq null \ \& \ \overline{p_i} \neq null \\ \overline{p_u}, & \overline{p_u} \neq null \ \& \ \overline{p_i} = null \\ \overline{p_i}, & \overline{p_u} = null \ \& \ \overline{p_i} \neq null \\ NoPrediction, & \overline{p_u} = null \ \& \ \overline{p_i} = null \end{cases}, \quad (11)$$

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 (11) includes four situations: (1) When $\overline{p_u} \neq null \ \& \ \overline{p_i} \neq null$, we combine the $\overline{p_u}$ 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 the *UMEAN* $\overline{p_u}$ for making prediction; (3) in case a new service user who did not invoked 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 use the *IMEAN* $\overline{p_i}$ for making prediction; and (4) when $\overline{p_u} = null \ \& \ \overline{p_i} = null$, we cannot provide any prediction since there is no information available.

## 3.4 Phase 4: Reliability Aggregation

In a service flow, compositional structures describe the order in which a collection of abstract tasks are executed. There are four types of basic compositional structures, i.e., *sequence, branch, loop*, and *parallel* [30]. These basic structures are included in BPMN [21] and can be mapped to BPEL [19] easily. Similar to the work in [5, 33, 38], we assume independent task failure in the service flow. The effect of error propagation (e.g., the problem studied in [9]) is not considered in this paper. The aggregation of component failure probabilities of these compositional structures is introduced in the following:

- **Sequence.** The tasks in a sequence structure are executed one by one. The sequence structure fails if any of its sub-task fails. The composed failure probability

of a sequence structure is calculated by

$$p = 1 - \prod_{i=1}^{n} (1 - p_i), \tag{12}$$

where $n$ is the number of sequential tasks and $p_i$ is the failure probability of the $i^{th}$ task.

- **Branch.** In the branch structure, only one branch will be executed for each execution. The composed failure probability of the branch structure is calculated by

$$p = 1 - \sum_{i=1}^{n} b_i(1 - p_i), \tag{13}$$

where $n$ is the number of branches, $p_i$ is the failure probability of the $i^{th}$ branch, and $b_i$ is the execution probability of the $i^{th}$ branch ($\sum_{i=1}^{n} b_i = 1$).

- **Loop.** The composed invocation failure probability of the loop structure is calculated by

$$p = 1 - \sum_{i=0}^{n} l_i(1 - p_1)^i, \tag{14}$$

where $p_1$ is the failure probability of the task in the loop structure, $l_i$ is the probability of executing the loop for $i$ times, $n$ is the maximum looping times, and $\sum_{i=0}^{n} l_i = 1$. When $n = 0$, the composed invocation failure probability is 0 since the task is not executed. In this paper, we assume that the values of $n$ (maximum looping times) and $l_i$ are provided by the system designer (e.g., the designer can simulate the service flow for a lot of times and record that approximate values).

- **Parallel.** All tasks are executed at the same time in the parallel structure, where each branch has an execution probability of 1. A parallel structure is counted as a failure if any of the parallel branches fail in execution. The invocation failure probability of the parallel structure is calculated by:

$$p = 1 - \prod_{i=1}^{n} (1 - p_i), \tag{15}$$

where $p_i$ is the probability that the $i^{th}$ parallel branch will fail.

These basic compositional structures can be nested and combined in an arbitrary way. For calculating the aggregated failure probability of a service flow, we decompose the service flow to the basic compositional structures hierarchically. As shown in Figure 3, firstly, the failure probabilities of the basic compositional structures $T_1$ and $T_2$ are calculated by using the corresponding formulas introduced above. Then, the failure probability of $T_3$ is calculated by employing the failure probabilities of $t_5$ and $T_2$. Finally, the failure probability of the whole service flow can be obtained by using Equation 12 and the failure probabilities of $t_1$, $T_1$, $T_3$, and $t_6$.

By the above approach, we obtain the aggregated failure probability of the service flow. To predict the reliability of the service flow, we adopt the commonly used exponential reliability function [17]:

$$R(t) = e^{-\gamma \times t}, \tag{16}$$

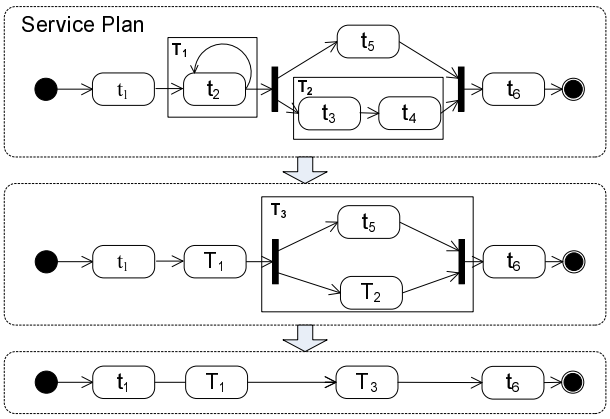



**Figure 3: Failure Probability Composition**

where $\gamma$ (*failure-rate*) is the number of failures of the service flow during a certain time duration, and $t$ is the time period for which the reliability is to be calculated. The value of $\gamma$ can be calculated by $p \times f$, where $p$ is the composed failure probability of the service flow $f$ is the execution frequency of the service flow (e.g., number of executions per hour). Therefore, we obtain the following equation for the reliability prediction for a service flow:

$$R(t) = e^{-p \times f \times t}. \tag{17}$$

By the above approach, designers of the service-oriented systems are able to predict reliabilities of the systems early at the architecture design phase. Moreover, after the system released, the prediction of system reliability can be dynamically updated when the performance of the service components is changed.

# 4. EXPERIMENTS

## 4.1 Experimental Setup

Our real-world Web service performance dataset (published at *www.wsdream.net*) is employed for experiments. This dataset inlcudes 100 publicly available Web services located in more than 20 countries and 150 distributed computers from Planet-Lab [8]. The service users (distributed computer from Planet-Lab) observe, collect, and contribute the failure data of the selected Web services to our centralized server, which is implemented by JDK, Eclipse, Axis2[4], Apache Tomcat, Apache HTTP Server, and MySQL. Each service user executes about 100 invocations on each selected Web service and the invocation failures are recorded. The failure probability of a Web service obtained by a service user can thus be obtained. Failure probabilities by all the 100 Web services observed by all the 150 service users can be presented as a $150 \times 100$ failure probability matrix.

To study the failure probability prediction performance, we compare our prediction approach (named as *Hybrid* for ease of presentation) with four other approaches: user-mean (UMEAN), item-mean (IMEAN), user-based approach using PCC (UPCC) [2], and item-based approach using PCC (IPCC) [25]. UMEAN employs the average failure probability of the current service user on other Web services for the

[4]http://ws.apache.org/axis2

prediction, while IMEAN employs the average failure probability of the Web service observed by other service users for the prediction. UPCC only employs similar users for the failure probability prediction, while IPCC only employs similar Web services for the prediction.

The 150 service users are divided into training users and testing users in our experiments. Failure probabilities of the training users are stored in our centralized server to make prediction for the testing users. To make our experiment more realistic, we randomly remove entries of the training users to make the training matrix sparser. Since the testing users usually only employ a small number of Web services, we also randomly remove entries of the testing users. Different prediction approaches are employed for predicting the failure probabilities of the removed entries. The original values of the removed entries are used as the expected values to study the prediction accuracy.

The well-known Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics are employed to measure the prediction accuracy of different approaches. MAE is defined as:

$$MAE = \frac{\sum_{u,i} |p_{u,i} - \widehat{p}_{u,i}|}{N}, \tag{18}$$

where $p_{u,i}$ denotes the expected failure probability value of Web service $i$ observed by service user $u$, $\widehat{p}_{u,i}$ denotes the predicted failure probability value, and $N$ denotes the number of predicted values. RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{u,i}(p_{u,i} - \widehat{p}_{u,i})^2}{N}}, \tag{19}$$

where smaller RMSE (or MAE) values indicate better prediction accuracy.

## 4.2 Comparison of Prediction Accuracy

Table 1 shows both the MAE and RMSE results of different prediction approaches employing 10% and 20% density training matrix. In the third row of Table 1, the numbers 10, 20 and 30 represent different *given numbers*, which are the number of failure probabilities provided by the testing users.

Table 1 shows that our prediction approach obtains better prediction accuracy (smaller MAE and RMSE values) in all the experimental settings. Table 1 also shows that MAE and RMSE values of the *Hybrid* approach become smaller with the increase of given number from 10 to 30. This observation indicates that the prediction accuracy can be improved by providing more Web service failure probabilities. With the increase of the training user number from 100 to 140, the prediction accuracy also has significant enhancement, since larger training matrix provides more information for the missing value prediction. When the density of the training matrix is increased from 10% to 20%, the prediction accuracy is also enhanced, since denser training matrix provides more information for the missing value prediction.

From Table 1, we also observe that the item-based approaches (IMEAN, IPCC) outperform the user-based approaches (UMEAN, UPCC) in all the experimental settings. This observation indicates that similar Web services provide more useful information than the similar users for the missing value prediction.

## 4.3 Studies on Parameters

### 4.3.1 Given Number

*Given number* indicates the number of Web service failure probabilities given by the current testing user. To study the impact of the *given number* on the prediction results, we vary the given number from 5 to 50 with a step value of 5. We set *training user number = 100, Top-K = 10, training matrix density = 10% and 20%* in the experiments.

Figure 4 shows the experimental results, where Figure 4(a) and 4(b) employ the 10% density training matrix for the missing value prediction, while Figure 4(c) and 4(d) employ the 20% density training matrix. The experimental results of Figure 4 show that: (1) the prediction accuracy of the *Hybrid* approach is enhanced slightly. This observation indicates that by providing a small set of observed Web service failure probabilities, the service user can obtain good prediction accuracy for the remanding Web services; (2) the prediction performance of the *IMEAN* approach is not influenced by the change of *given number*, since it does not employ the given failure probabilities for the missing value prediction; and (3) our *Hybrid* approach outperforms other approaches consistently.

### 4.3.2 Training Matrix Density

The prediction accuracy is also influenced by the training matrix density. To study the impact of the training matrix density on the prediction results, we vary the density from 5% to 50% with a step value of 5%. We set *training user number = 100, Top-K = 10, given number = 10 and 20* in the experiments.

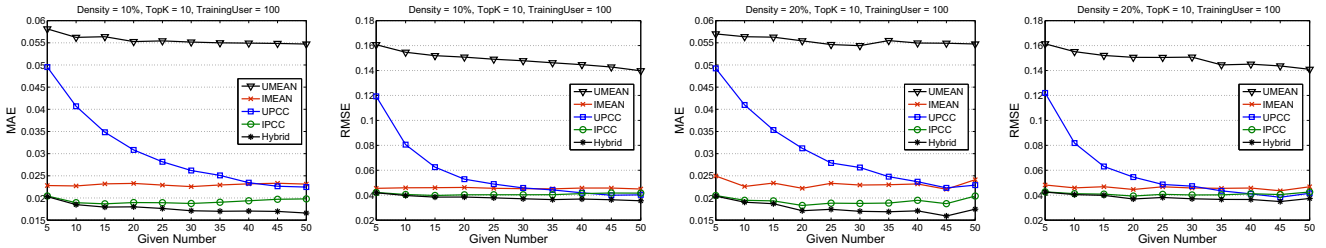
Figure 5 shows the experimental results, where Figure 5(a) and 5(b) employ given number of 10, and Figure 5(c) and 5(d) employ given number of 20. The experimental results of Figure 5 show that: (1) the prediction accuracy of *Hybrid* is significantly enhanced when the matrix density is increased from 5% to 10%. With the increase of matrix density, the speed of accuracy enhancement slows down. This observation indicates that the prediction accuracy will be enhanced by collecting more past failure data from different service users to make the training matrix denser, especially when the matrix is sparse; (2) the performance of *UMEAN* is not influenced by the density of training matrix, since it does not employ the information of the training matrix for the missing value prediction; and (3) the *Hybrid* approach still outperforms other approaches consistently.

### 4.3.3 Training User Number

To study the impact of the training user number on the prediction results, we vary the training user number from 20 to 140 with a step value of 20. We set *Top-K = 10* in the experiments. Figure 6 shows the experimental results under different experimental settings, where Figure 6(a) employs *given number = 10, matrix density = 10%*, Figure 6(b) employs *given number = 20, matrix density = 20%*, Figure 6(c) employs *given number = 30, matrix density = 30%*, and Figure 6(d) employs *given number = 40, matrix density = 40%*. The experimental results of Figure 6 show that: (1) the prediction accuracy of *Hybrid* is enhanced stably with the increase of training user number under all the experimental settings. This observation indicates that by encouraging more service users to contribute their Web service past failure data, the prediction accuracy can be improved stably. On the other hand, if we are able to provide accurate Web service failure probability prediction for the service users,

**Table 1: Prediction Performance Comparison**

| Metrics | Methods | Training Users = 100 | | | | | | Training Users = 140 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Matrix Density=10% | | | Matrix Density=20% | | | Matrix Density=10% | | | Matrix Density=20% | | |
| | | 10 | 20 | 30 | 10 | 20 | 30 | 10 | 20 | 30 | 10 | 20 | 30 |
| **MAE** | UMEAN | 0.057 | 0.056 | 0.055 | 0.056 | 0.055 | 0.056 | 0.052 | 0.048 | 0.050 | 0.051 | 0.050 | 0.050 |
| | IMEAN | 0.024 | 0.023 | 0.024 | 0.023 | 0.023 | 0.023 | 0.017 | 0.016 | 0.016 | 0.016 | 0.016 | 0.016 |
| | UPCC | 0.049 | 0.041 | 0.038 | 0.043 | 0.035 | 0.029 | 0.041 | 0.035 | 0.032 | 0.039 | 0.030 | 0.025 |
| | IPCC | 0.028 | 0.027 | 0.027 | 0.026 | 0.025 | 0.025 | 0.019 | 0.019 | 0.019 | 0.017 | 0.016 | 0.016 |
| | **Hybrid** | **0.024** | **0.021** | **0.021** | **0.021** | **0.020** | **0.020** | **0.016** | **0.014** | **0.014** | **0.014** | **0.012** | **0.012** |
| **RMSE** | UMEAN | 0.155 | 0.151 | 0.147 | 0.155 | 0.151 | 0.147 | 0.146 | 0.144 | 0.139 | 0.148 | 0.143 | 0.139 |
| | IMEAN | 0.057 | 0.054 | 0.055 | 0.046 | 0.046 | 0.046 | 0.034 | 0.033 | 0.033 | 0.027 | 0.026 | 0.026 |
| | UPCC | 0.107 | 0.072 | 0.064 | 0.080 | 0.055 | 0.045 | 0.091 | 0.057 | 0.047 | 0.073 | 0.042 | 0.034 |
| | IPCC | 0.060 | 0.056 | 0.055 | 0.045 | 0.043 | 0.043 | 0.034 | 0.032 | 0.031 | 0.024 | 0.023 | 0.022 |
| | **Hybrid** | **0.055** | **0.050** | **0.049** | **0.042** | **0.039** | **0.039** | **0.030** | **0.027** | **0.026** | **0.022** | **0.020** | **0.020** |




(a) MAE of Density 10% (b) RMSE of Density 10% (c) MAE of Density 20% (d) RMSE of Density 20%

**Figure 4: Impact of the Given Number**

more service users will be willing to contribute their observed Web service failure data to exchange for more accurate missing value prediction service; and (2) the *Hybrid* approach also outperforms other approaches consistently.

### 4.3.4 Top-K

To study the impact of the *Top-K* parameter on the prediction results of the *Hybrid* approach, we vary the value of *Top-K* from 2 to 20 with a step of 2. Figure 7(a) and 7(b) show MAE and RMSE results under the experimental settings of *training matrix density = 50%, given number = 10, 20, and 30, train user number = 100*. Figure 7(c) and 7(d) show MAE and RMSE values under the experimental settings of *training matrix density = 10%, 20% and 30%, given number = 50, training user number = 100*.

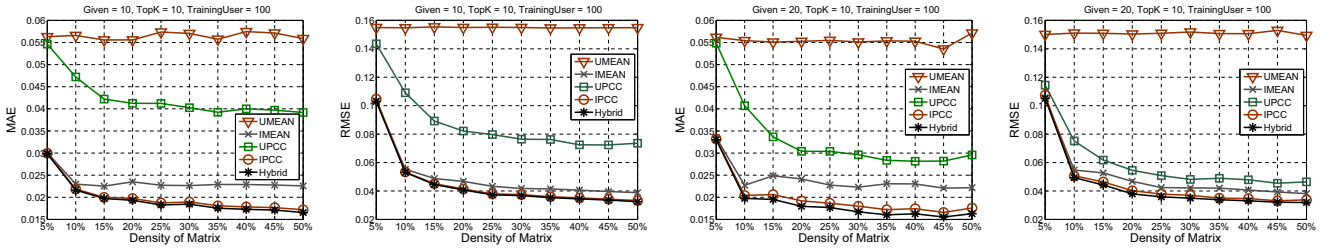
Figure 7 shows that (1) the prediction accuracy stops to enhance when the *Top-K* is larger than a certain value (e.g., 6 for given 10, 10 for given 30, and 12 for given 50). (2) The prediction performance will not decrease with the increase of the *Top-K* value, since we exclude dissimilar users with negative PCC values from the *Top-K* set. This observation indicates that by using our enhanced *Top-K* algorithm, we can simply set the value of *Top-K* to a large value for obtaining better prediction performance.

### 4.3.5 Parameter $\lambda$

The experimental results of Table 1 show that item-based approaches outperform the user-based approach in our Web service failure probability dataset, indicating that different datasets may inherit different data distribution and correlation characteristics. Our prediction approach is adaptable to different datasets by providing a parameter $\lambda$. 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 based on the value of $\lambda$. To study the impact of $\lambda$ on the prediction result of our approach, we vary the value of $\lambda$ from 0 to 1 with a step value of 0.1. We also set *Top-K = 10* and *training users =100* in the experiments.

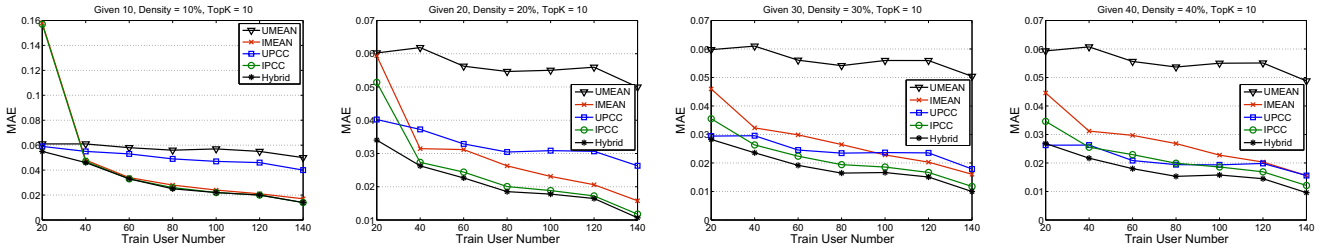
Figure 8(a) and 8(b) show the MAE and RMSE results of given number 10, 20, 30, 40 and 50 with 10% training matrix density, while Figure 8(c) and 8(d) show the experimental results with 20% training matrix density. Figure 8 shows that the value of $\lambda$ impacts the recommendation results significantly. Suitable $\lambda$ values will provide better prediction accuracy, since it enables properly combination of the user-based method and the item-based method.

Another interesting observation is that the optimal $\lambda$ value (the minimal MAE and RMSE values of the curves in Figure 8) shifts from left to right when the given number is increased from 10 to 50. For example, in Figure 8(c), the optimal $\lambda$ value of given 10 is 0.1, while the optimal $\lambda$ value of given 50 is 0.6. This observation indicates that the optimal $\lambda$ value is influenced by the *given number*. The similar Web services are more important than similar users when the given number is small. With the increasing of given number, the similar users become more important. This observation is reasonable, since with limited user-given Web service failure values, the UMEAN prediction method, which employ the mean of the user-given failure probabilities to predict the failure probability of other Web services, has higher probability to be inaccurate. This will influence the prediction performance of UPCC, which employs the UMEAN value for the missing value prediction as shown in Equation (6).




(a) MAE of Given 10 (b) RMSE of Given 10 (c) MAE of Given 20 (d) RMSE of Given 20

**Figure 5: Impact of the Training Matrix Density**




(a) Given 10 Density 10% (b) Given 20 Density 20% (c) Given 30 Density 30% (d) Given 40 Density 40%

**Figure 6: Impact of the Train User Number**

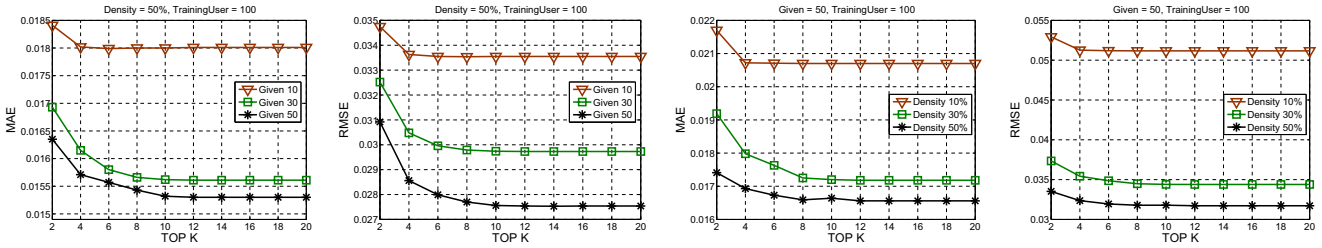
## 5. RELATED WORK AND DISCUSSION

A great deal of software reliability prediction models have been proposed for stand-alone software systems, such as Musa's execution time model [20], Putnam's model [22], Rome laboratory models [10], Jelinski's model [15], Littlewood's models [16], and so on. More and more reliability prediction approaches are also proposed for component-based systems and service-oriented systems [4, 7, 11, 13, 14, 31]. However, most of these previous approaches focus on system-level reliability compositional analysis and assume that the component reliabilities are known. Different from these approaches, our collaborative reliability prediction approach focuses not only on system-level reliability aggregation, but also on component-level Web service failure probability prediction.

A few approaches [6, 12, 23], which consider component-level reliability prediction, are mainly designed for traditional component-based systems. Goseva-Popstojanova et al. [12] calculate reliability risk of a component by the component complexity and the levels of its failures. Reussner et al. [23] compute 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. Cheung et al. [6] predict the component reliability at architecture design phase by exploiting behavioral models from sources of information available at design time. Different from work [6], which tries to predict reliability of a component which is not implemented yet, our work targets at predicting reliabilities of remote Web services which are implemented and provided by other organizations. Moreover, work [6] does not need to consider the influence of communication links since the components are usually invoked locally in traditionally component-based systems. Our approach addresses the influence of communication links by employing collaborative filtering for reliability prediction, which significantly enhances the reliability prediction accuracy for the service-oriented systems.

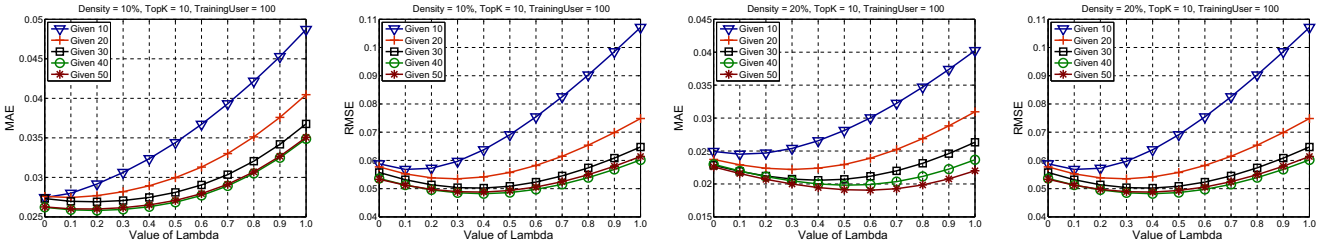
Collaborative filtering is the process of filtering for information using techniques involving collaboration among multiple agents. Various collaborative filtering techniques are widely adopted in recommender systems [3, 18]. The most studied approaches include user-based approaches [2] and item-based approaches [25]. In the environment of service computing [34], it is possible to collect and make use of the past failure data of Web services from different service users. To take advantages of these collected data, this paper applies the collaborative filtering techniques [2, 18, 25, 39] to attack the challenging research problem of reliability prediction of service-oriented systems. As far as we know, the collaborative reliability approach of this paper is different from all previous reliability prediction models.

In our current design, the collaborative reliability prediction approach mainly focuses on service-oriented systems, since the Web services are usually publicly accessible and are invoked by a lot of service users. Therefore, it is possible to obtain the past failure data of Web services from different service users. Our approach can further be extended to component-based systems, other distributed computing platforms, and cloud computing platforms, in case that the reusable components are employed by other users previously and the past failure data can be obtained from these users. Similar to the design of BitTorrent [1] where users can obtain faster download speed if the file (e.g., a movie) is popular and a lot of users are collaborating on downloading it, our approach can achieve better reliability prediction accuracy




(a) MAE of Density 50% (b) RMSE of Density 50% (c) MAE of Given 50 (d) RMSE of Given 50

**Figure 7: Impact of the Top-K**




(a) MAE of Density 10% (b) RMSE of Density 10% (c) MAE of Density 20% (d) RMSE of Density 20%

**Figure 8: Impact of the $\lambda$**

when a Web service is popular with a lot of users and past failure data.

*User-collaboration* and *user-contribution* are major characteristics of Web 2.0. In our previous work, the idea of *user-collaboration* has been employed for Web service reliability evaluation [35, 37] and Web service recommendation [39]. In this paper, similar idea is employed to collect data from service users and to make reliability prediction of the service-oriented systems for the system designers. By our approach, the collected data from other users are employed for making reliability prediction for the current user without requiring real-world Web service invocations.

# 6. CONCLUSION AND FUTURE WORK

In this paper, we propose a collaborative reliability prediction approach for service-oriented systems. The main idea is to exploit the past failure data of the similar users for predicting Web service failure probabilities for the current user. The predicted failure probabilities of the elementary Web services are composed using different compositional structures to predict the reliability of the whole service-oriented system. The comprehensive experimental analysis shows the effectiveness of our collaborative reliability prediction approach.

When aggregating the failure probabilities of the service components, we assume service component failures are independent, In most cases, this assumption is reasonable, since Web services are usually deployed on separate severs of different organizations. The physical and electrical isolation ensures that Web service failures are independent. However, in some special cases, failures of Web services may have correlation (e.g., two Web services running on the same server, error propagation, etc.). We will address this Web service failure correlation problem in our future work.

Our on-going research also includes exploring failure correlation between different Web services, collecting more past failure data on more Web services, and investigating the optimal $\lambda$ value in different experimental settings.

# Acknowledgement

The work described in this paper was fully supported by a grant (Project No. CUHK4154/09E) from the Research Grants Council of the Hong Kong Special Administrative Region, China.

# 7. REFERENCES

[1] C. Bram. Incentives build robustness in bittorrent. In *Proc. First Workshop Economics of Peer-to-Peer Systems*, pages 1–5, 2003.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. 14th Annual Conf. Uncertainty in Artificial Intelligence (UAI'98)*, pages 43–52, 1998.

[3] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[4] J. Cardoso, J. Miller, A. Sheth, and J. Arnold. Modeling quality of service for workflows and web service processes. *Journal of Web Semantics*, 1:281–308, 2002.

[5] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *Proc. Third Symp. Operating Systems Design and Implementation*, pages 1–14, 1999.

[6] L. Cheung, R. Roshandel, N. Medvidovic, and L. Golubchik. Early prediction of software component

reliability. In *Proc. 30th Int'l Conf. Software Eng. (ICSE'08)*, pages 111–120, 2008.

[7] R. C. Cheung. A user-oriented software reliability model. *IEEE Trans. Software Engeering*, 6(2):118–125, 1980.

[8] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: An overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, July 2003.

[9] V. Cortellessa and V. Grassi. A modeling approach to analyze the impact of error propagation on reliability of component-based systems. In *Proc. 10th Int'l Symp. Component-Based Software Eng.*, pages 140–156, 2007.

[10] M. Friedman, P. Tran, and P. Goddard. *Reliability Techniques for Combined Hardware and Software Systems*. Rome Laboratory, RL-TR-92-15, 1992.

[11] S. S. Gokhale and K. S. Trivedi. Reliability prediction and sensitivity analysis based on software architecture. In *Proc. Int'l Symp. Software Reliability Eng. (ISSRE'02)*, pages 64–78, 2002.

[12] K. Goseva-Popstojanova, A. Hassan, W. Abdelmoez, D. E. M. Nassar, H. Ammar, and A. Mili. Architectural-level risk analysis using uml. *IEEE Trans. Software Engeering*, 29(10):946–960, 2003.

[13] K. Goseva-Popstojanova and K. S. Trivedi. Architecture-based approach to reliability assessment of software systems. *Performamce Evaluation*, 45(2-3):179–204, 2001.

[14] V. Grassi and S. Patella. Reliability prediction for service-oriented computing environments. *IEEE Internet Computing*, 10(3):43–49, 2006.

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

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

[17] M. R. Lyu. *Handbook of Software Reliability Eng.* McGraw-Hill, New York, 1996.

[18] H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. In *Proc. 30th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'07)*, pages 39–46, 2007.

[19] R. Ma, Y. Wu, X. Meng, S. Liu, and L. Pan. Grid-enabled workflow management system based on bpel. *Int'l J of High Perf. Computing Applications*, 22(3):238–249, 2008.

[20] J. D. Musa, A. Iannino, and K. Okumoto. *Software reliability: measurement, prediction, application*. McGraw-Hill, New York, USA, 1990.

[21] Object Management Group (OMG). *Business Process Modeling Notation version 1.1*, January 2008.

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

[23] R. H. Reussner, H. W. Schmidt, and I. H. Poernomo. Reliability prediction for component-based software architectures. *Journal of System and Software*, 66(3):241–252, 2003.

[24] N. Salatge and J.-C. Fabre. Fault tolerance connectors for unreliable web services. In *Proc. 37th Int'l Conf. Dependable Systems and Networks (DSN'07)*, pages 51–60, 2007.

[25] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. 10th Int'l Conf. World Wide Web (WWW'01)*, pages 285–295, 2001.

[26] A. Scholz, C. Buckl, A. Kemper, A. Knoll, J. Heuer, and M. Winter. Ws-amuse - web service architecture for multimedia services. In *Proc. 30th Int'l Conf. Software Eng. (ICSE'08)*, pages 703–712, 2008.

[27] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating word of mouth. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*, 1995.

[28] M. H. ter Beek, S. Gnesi, N. Koch, and F. Mazzanti. Formal verification of an automotive scenario in service-oriented computing. In *Proc. 30th Int'l Conf. Software Eng. (ICSE'08)*, pages 613–622, 2008.

[29] W.-T. Tsai, X. Zhou, Y. Chen, and X. Bai. On testing and evaluating service-oriented software. *IEEE Computer*, 41(8):40–46, 2008.

[30] W.-L. Wang, D. Pan, and M.-H. Chen. Architecture-based software reliability modeling. *Journal of Systems and Software*, 79(1):132–146, 2006.

[31] S. M. Yacoub, B. Cukic, and H. H. Ammar. Scenario-based reliability analysis of component-based software. In *Proc. Int'l Symp. Software Reliability Eng. (ISSRE'99)*, pages 22–31, 1999.

[32] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. the Web*, 1(1):1–26, 2007.

[33] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Software Engeering*, 30(5):311–327, 2004.

[34] L.-J. Zhang, J. Zhang, and H. Cai. Services computing. In *Springer and Tsinghua University Press*, 2007.

[35] Z. Zheng and M. R. Lyu. A distributed replication strategy evaluation and selection framework for fault tolerant web services. In *Proc. 6th Int'l Conf. Web Services (ICWS'08)*, pages 145–152, 2008.

[36] Z. Zheng and M. R. Lyu. A qos-aware middleware for fault tolerant web services. In *Proc. Int'l Symp. Software Reliability Engineering (ISSRE'08)*, pages 97–106, 2008.

[37] Z. Zheng and M. R. Lyu. Ws-dream: A distributed reliability assessment mechanism for web services. In *Proc. 38th Int'l Conf. Dependable Systems and Networks (DSN'08)*, pages 392–397, 2008.

[38] Z. Zheng and M. R. Lyu. A qos-aware fault tolerant middleware for dependable service composition. In *Proc. 39th Int'l Conf. Dependable Systems and Networks (DSN'09)*, pages 239–248, 2009.

[39] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Wsrec: A collaborative filtering based web service recommender system. In *Proc. 7th Int'l Conf. Web Services (ICWS'09)*, pages 437–444, 2009.