An Adaptive Multilevel Indexing Method for Disaster Service Discovery

Yan WU, ChunGang YAN, Lu LIU, ZhiJun DING and ChangJun JIANG, *Member, IEEE*

Abstract—With the globe facing various scales of natural disasters then and there, disaster recovery is one among the hottest research areas and the rescue and recovery services can be highly benefitted with the advancements of Information and Communications Technology (ICT). Enhanced rescue effect can be achieved through the dynamic networking of people, systems and procedures. A seamless integration of these elements along with the service-oriented systems can satisfy the mission objectives with the maximum effect. In disaster management systems, services from multiple sources are usually integrated and composed into a usable format in order to effectively drive the decision-making process. Therefore, a novel service indexing method is required to effectively discover desirable services from the large-scale disaster service repositories, comprising a huge number of services. With this in mind, this paper presents a novel multilevel indexing algorithm based on the equivalence theory in order to achieve effective service discovery in large-scale disaster service repositories. The performance and efficiency of the proposed model have been evaluated by both theoretical analysis and practical experiments. The experimental results proved that the proposed algorithm is more efficient for service discovery and composition than existing inverted index methods.

Index Terms—Service Computing, SOA, SOC, Disaster Management

1 Introduction

Natural and man-made disasters and their significant impacts on human lives are always being the focal points of discussion among the community of human and environmental activists. We are witnessing the natural calamities showing different dimensions in different geography, affecting the human community and it is always being a tough fight with the nature. According to the World Disaster Report 2012 [1], 15 million people were forced to be displaced due to natural and technological disasters.

The importance of ICT and its possible innovation in the disaster recovery has been highlighted in the newly published World Disaster Report 2013 [2]. According to which, "*the responsible use of technology offers concrete ways to make humanitarian assistance more effective, efficient and accountable and can, in turn, directly reduce vulnerability and strengthen resilience. Finding ways for advances in technology to serve the most vulnerable is a moral imperative; a responsibility, not a choice.*" "*Haiti in 2010 saw the first field deployment of many technologies with the potential to support disaster assessment and response.*"

In the 2010 Haiti Earthquake, between 217,000 and 230,000 people found dead, and 300,000 estimated to be injured, and 1,000,000 estimated to be homeless. Rescue efforts began in the immediate aftermath of the earthquake, supported momentously by the international communities. The factor lies behind the successful relief and recovery is the real-time monitoring of sensible data with an international boundary.

A rescue capability can be defined as the operation of integrated services to fulfil a rescue mission objective. For the provision of rescue capabilities, a defined set of disaster services need to be integrated in an effective workflow in order to fulfil the objective requirements. In more particular, the operations of sensors should be integrated to achieve a joint functionality to fulfil a wide range of search and rescue requirements. Generally, in an earthquake rescue, the search and rescue teams will initially search for the survivals struggling within the destructed buildings, and save them accordingly. In the meantime, medical assistance should be on its prompt action along with immediate transportation facilities to hospitals. Apart from these basic necessities, there are other services to be considered including power supply, road services, sufficient food and water availability, communication, transporting goods, and secondary disaster recovery services, such as, monitor and alarm devices. On the whole, disaster recovery services cumulate a set of duties functioning towards the rescue objectives with a defined workflow. The fundamental architecture of the Service Oriented Architecture (SOA) can be effectively utilized in order to enable the above said recovery services optimally and with a timely approach. SOA usually stores the metadata in its repositories and its novel service composition method can be used to search and retrieve the required services and also to dynamically compose them into a joint-rescue capability. These factors suggest that more efforts are required to devise efficient solutions to ultimately solve the service composition problem. Therefore, a new service discovery and retrieval model is re-

- *Y. Wu, C. Yan, Z. Ding, and C. Jiang are with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, 201804, China. E-mail: vw_@163.com; yanchungang@tongji.edu.cn; dingzj@tongji.edu.cn; cjjiang@tongji.edu.cn.*
- *L. Liu is with the Department of Computing and Mathematics, University of Derby, Derby, DE22 1GB, UK E-mail: l.liu@derby.ac.uk.*
- *Y. Wu is also with the School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang, Jiangsu Province, 212013, China.*

quired to address and to overcome the existing issues and challenges faced by disaster management systems.

In this paper, we propose a novel adaptive multilevel index model to efficiently manage and to retrieve services in a large-scale repository and to facilitate service discovery and composition. The proposed model includes a multilevel index and functionalities to support management operations, i.e., addition, deletion, replacement and retrieval and adaptive deployment.

The contributions of this work are summarized as follows:

1. A novel multilevel index model without any redundancy has been proposed in this paper, based on the theory of the equivalence relations and quotient sets, which can reduce the service discovery and composition time for an effective disaster service management;
2. This model specifies how to retrieve, insert and delete services from the system;
3. This model includes a novel flexible deployment algorithm capable of adapting self-deployment according to different situations by using different types of indices;
4. Finally, the efficiency of the proposed model is proved in comparison with other existing indexing methods with our supporting experimental analysis and results.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 presents the fundamental architecture for dependable service composition. Section 4 introduces the proposed multilevel index model. Section 5 describes the enabling key functions and operations built in the proposed model. Section 6 discusses the flexible deployment algorithm of the proposed model. Section 7 reports the experimental results. Section 8 concludes the paper with an outline of our future research directions.

2 Related Work

SOA provides proper foundation architecture for disaster rescue. Technologies of Service-Oriented Computing (SOC), e.g. service composition, discovery and selection [3-10], offer effective methods for integration of rescue capability. Reliability and timeliness are the two important factors for rescue capability. In our previous works, reliability is studied [11, 12]. L. Liu *et al.* [11] proposed an evolutionary service oriented architecture to offer a reliable rescue capability by means of dynamically integrating businesses, systems and computing in heterogeneous environments. In [12], a reconfiguration algorithm enabling proactive self-diagnosing of the evolution is proposed to evaluate the impact of evolution, and the self-configuration capabilities in adopting to the changes. However, they were targeted only at reliability, ignoring timeliness.

Service discovery and composition are closely related with the timeliness of rescue capability that influences users' satisfactions. Recent studies [13-15] reveal the fact that the composition problem is very time-consuming. Narayanan and McIlraith [13] used Petri nets to analyze the complexities in the composition of the Semantic Web services described by DAML-S and identified this problem to be a Exp-Space-Time-hard in the worst case. Nam *et al*. [14] studied the computational complexities of the behavioral description-based Web service composition, and concluded that the composition problem of nondeterministic Web services with incomplete information, to be 2-EXP-complete. Our prior work also proved that it is co-NP-hard for a given group of services to decide the behavioral compatibility of the service composition [15].

Tang *et al*. [16] introduced a novel automatic Web service composition method based on the logical inference of Horn clauses and Petri nets. They first transformed a Web service composition problem into a logical inference problem of Horn clauses, based on the forward-chaining algorithm. They used the structural analysis techniques of the Petri nets in order to obtain the composite service. Since the service repository usually holds a large number of services and generates many operating rules, the Petri net of a Horn clause is usually set very large. In order to reduce the composition time, they proposed a method to select the candidate clauses during the arrival of new queries, which creates interference. The main drawback of this method is that it can be executed only after receiving the user requirements, not beforehand.

Wu and Khoury [17] proposed a tree-based search algorithm for Web service composition in a cloud computing platform. They first created a tree that represents all the possible composition solutions according to the user requirements, and then pruned the illegal branches, aiming to reduce the response time and to improve the performance, and finally used a heuristic algorithm to search for an optimal solution. This method has the disadvantage similar to that in [16]. Particularly, its optimization process cannot be executed before receiving user requirements.

Constantinescu *et al*. [18] proposed a type-compatible service composition method. They used a forward composition algorithm to obtain a solution. This method includes other irrelevant services, which leads to its downfall. Kwon *et al*. [19] proposed a two-phase composition method to overcome such a drawback of [18]. Its first phase is to generate a composition solution via a forward composition algorithm and the second phase is to eliminate useless services backward. In the forward phase, a service net is used to reduce the composition time. Let s_i denote a service, and ${}^\bullet s_i$ and $s_i{}^\bullet$ denote the input and output parameters of s_i, respectively. Services are the nodes of the net. If $s_i{}^\bullet \cap {}^\bullet s_j \neq \emptyset$, there exists a directed edge from s_i to s_j. If s_i is selected in the solution, only the services linked by s_i need to be retrieved, and its efficiency in solving the problem is determined. The service net narrows the search space for every step of service composition (except the first one), and is effective in improving the composition efficiency. Lee *et al*. [20] proposed a scalable and efficient Web service composition method based on a relational database. They used the service net as a basic data structure. The service net can be addressed to have two shortcomings. First, it does not take the consideration of facilitating the service discovery. Secondly, it is time-

consuming for service addition and deletion. For example, when a service s_i is added, service net requires the determination of, whether $s_j^{\bullet} \cap^{\bullet} s_i \neq \varnothing$ and $s_i^{\bullet} \cap^{\bullet} s_j \neq \varnothing$ for every existing service s_j.

Aversano *et al*. [21] proposed a backward composition method that composes services from the terminal state to the initial one. It includes two steps, as the horizontal and the vertical step. The former is to find a minimal set of services that can converge to a target state. The latter is to repeat the former until a stop condition is met. Clearly, according to the Binomial theorem [22], the time complexity of the horizontal step is 2^n in the worst case, where n is the number of services. In order to reduce its composition time, Li *et al*. [23] proposed an inverted index to manage the services. An inverted index strategy is highly efficient and it is used widely in many fields. For example, Google uses it to reduce the response time for user's queries [24, 25]. It is also adopted in the Web service storages. The inverted index [21] usually stores the services along with their corresponding parameters, and an index link is created between the service and its output parameter. This strategy is efficient and effective for the backward service composition methods, and also convenient for service addition and deletion. But it is not applicable to the forward composition method [17-19] which is simpler and more popular than the backward one. Note that, the time complexity of this method is $n(n\text{-}1)/2$ in the worst case, which is in contrast with the exponential time complexity of the backward method [21].

The inverted index can be easily modified to increase its adaptability to the forward composition method, with a simple change in the index link created between the service output and its corresponding input parameter. Clearly, the principle of the amended inverted index and the service net [19, 20] is almost the same, nevertheless. But it overcomes the service net issues of service addition and deletion. However, it also includes considerable redundancies, which is later explained in this paper. These redundancies can significantly cause considerable wastage of time during the process of service composition and discovery. In order to eliminate such redundancies effectively, an adaptive multilevel index model is proposed in this paper based on the equivalence relations and the quotient sets. Our prior work [26] presents some preliminary work relevant to the multilevel index theory. This paper comprehensively presents a multilevel index model, and introduces an adaptive deployment algorithm and four efficient operations to manage and maintain services.

3 Architecture for Dependable Disaster Service Composition

In our previous work [11], an evolutionary service oriented architecture has been proposed to offer a dependable rescue capability by means of dynamically integrating businesses, systems and computing in heterogeneous environments. For example, services for detecting survivors could be benefitted by employing lasers, visible spectra, heat (infrared), radar or sonar sensors and these heterogeneous sensor services can also be deployed on rescue helicopters, UAVs and on the search and rescue personnel. The sensing services allow interoperable service interfaces and enhance the quality of service (QoS) in order to deliver dependable capabilities in highly critical situations.

In order to satisfy different search and rescue requirements, different services need to be integrated to deliver a joint search and rescue mission. A dynamic network of services is modelled with a series of radar sensors, which provide the ongoing data and facilitate timely update. In our system, the information of Points of Interest (POIs) in a specified region can be obtained in real-time. A sequence of services (such as "Get map information", "Get radar reading", and "Display targets on map") are operated in a workflow in order to provide a regional surveillance service, as illustrated in Fig. 1(a). The service integration can be abstracted by using workflow patterns as shown in Fig. 1(b), where s_1 represents the service of 'getting map information', s_2 represents the service of 'getting radar reading' and s_3 represents the service of 'displaying targets on map'.

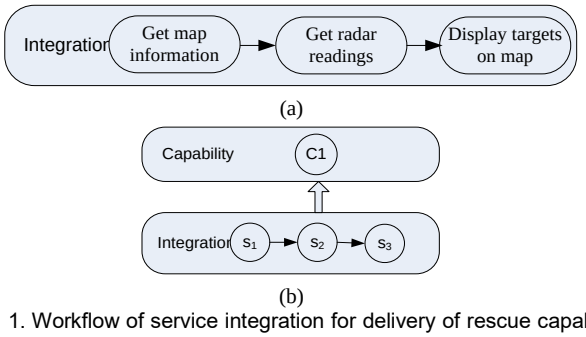


Fig. 1. Workflow of service integration for delivery of rescue capabilities.

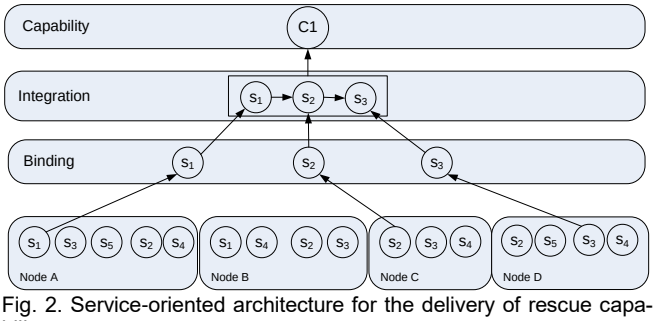


Fig. 2. Service-oriented architecture for the delivery of rescue capability.

Disaster monitoring sensor networks consist of potentially a large number of nodes with sensing and wireless communication capability. Each platform node can have one or more sensors supplying data through services. Fig. 2 illustrates the integration of services for the delivery of rescue capability on SOA [11]. In this architecture, each platform node has a number of services that can be integrated to form a higher level of functionality to deliver a rescue capability. For example, a rescue helicopter is a platform node offering services such as movement, metrological surveillance, situation surveillance, target surveillance and patient delivery services. Further, the metrological surveillance function can be combined with other services to form a higher-level weather service contributing to the search-and-rescue mission.

Redundant service binding is a technique to improve the reliability of the provision of rescue capabilities [11]. In order to deliver a reliable rescue capability, the necessary functionalities should be provided by multiple services, which are usually allocated to different peer nodes. The reconfiguration algorithm can switch to one of the backup services in case of any failures of the initial service. The Distributed Recovery Block scheme [27] is applied to minimize the recovery time of the integration process.

Apart from the failures of the services, rescue capability evolution could develop potential problems, affecting the reliability of provision of the rescue capabilities. Operational requirements often change during a rescue and search operation, in accordance with dynamically varying situations. The rescue capability should be flexible to adapt such dynamic requirements. A reconfiguration algorithm enabling proactive self-diagnosing of the evolution is used to evaluate the impact of evolution, and the self-configuration capabilities in adopting to the changes, were dealt in [12].

Reliability and timeliness are the two important factors affecting the effectiveness of a disaster relief effort. The previous works discussed earlier were providing a number of architectural solutions to the problem of dynamic changes, but most of the works were targeted only at reliability, ignoring timeliness. However, the benefits of the integration of services will not be realized unless an effective method is developed to manage all the sensor services and to discover the required services effectively from a large-scale sensor service repository, for dynamic service composition. To this extent, our researches are using SOA for disaster management and relief, without undermining the key factor 'timeliness' and developed our highly efficient multilevel index model.

In the proposal model, a service is represented by its input and output parameters and its related attributes. A service is defined as follows:

Definition 1. A service $s=({}^{\bullet}s, s^{\bullet}, O)$, where ${}^{\bullet}s$ is the set of input parameters, and $s^{\bullet}$ is the set of output parameters. O is a set of service attributes, e.g., QoS.

Service retrieval is a function that accepts a set of parameters and returns a set of services that can be invoked by this parameter set.

Definition 2. Service retrieval $\mathrm{Re}(A, S)=\{s \mid {}^{\bullet}s \subseteq A \wedge s \in S\}$, where A is a given parameter set and S is a service set.

Service retrieval is an operation that accepts a set of service parameters and rapidly returns a set of services that can be invoked for effective service composition to achieve an effective search and rescue mission.

Definition 3. Service discovery can be defined as $\mathrm{Dc}(A, B, S, L)=\{s \mid {}^{\bullet}s \subseteq A \wedge B \subseteq s^{\bullet} \wedge s \in S \wedge L(O)\}$, where A is a parameter set provided by a user, B is a parameter set required by the user, S is a service set, and $L(O)$ means that the discovered services must meet some conditions required by users.

Therefore, $\mathrm{Dc}(A, B, S, L)=\{s \mid \mathrm{Re}(A, S) \wedge B \subseteq s^{\bullet} \wedge L(O)\}$. It is clear that service retrieval is an essential part of dynamic service composition. If the time of service retrieval is reduced, the overall time of service composition will be reduced.

According to the definition of service retrieval, it will return a set of services with the same input and output parameters. The functionality of service retrieval defined in this paper is to narrow the search space for service discovery. For this reason, if $s=(\{a, b\}, \{c, d\}, O)$, it is written as $s{:}ab \rightarrow cd$ for conciseness. There is no ordered relationship among the elements before or after "$\rightarrow$". S denotes the set of all services in a repository, and $s \in S$, if no other setting is mentioned.

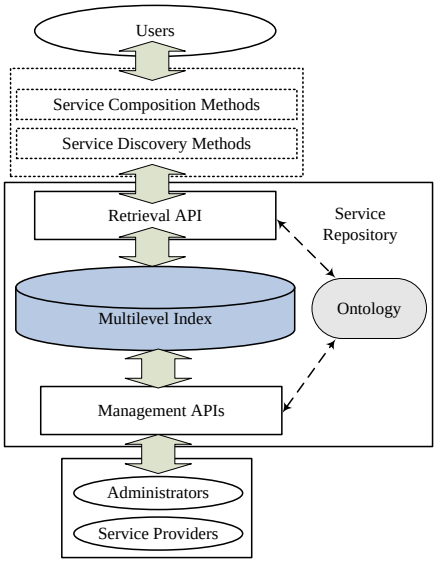


Fig. 3. A multilevel index model in a service repository

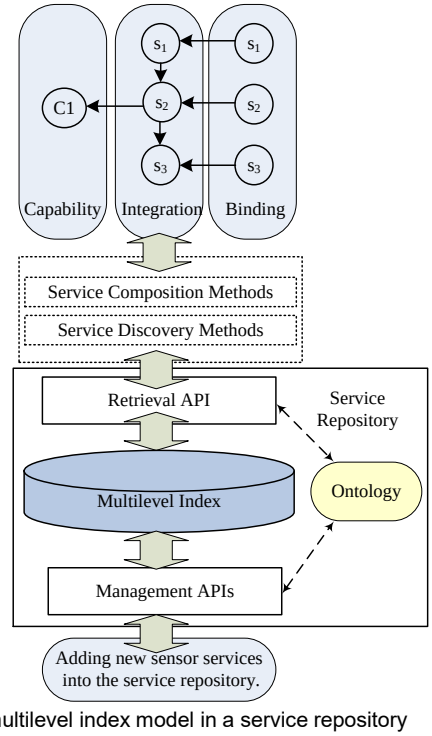


Fig. 4. A multilevel index model in a service repository

The proposed multilevel index model can be easily implemented in a service repository. Fig. 3 shows such a scenario. The multilevel index is the core of the proposed model, which is an efficient data structure for service re-

trieval, addition, deletion and replacement. Service retrieval is an operation that accepts a set of service parameters and returns a set of services that can be invoked according to the parameter set. It is commonly invoked by the service composition and the discovery methods, when a particular service needs to be identified from the service repository according to a parameter set. Since this retrieval returns a smaller service set rather than all services in a repository, the service composition or the discovery method uses it effectively and works on a smaller service sets to find the most suitable services, thereby reducing their respective processing time. Service addition, deletion and replacement are the three maintenance operations that can be used by the administrators or the service providers to maintain the services in a service repository. The proposed model needs an ontology to ensure that each service parameter has a unique identifier. Otherwise, it cannot work correctly.

The proposed model can provide supports for disaster recovery in varying dimensions. Fig. 4 shows some examples. The process of integration of disaster services invokes the service composition and the discovery methods; meanwhile, the service composition and the discovery methods invoke the service retrieval in order to reduce their processing time. Upon identifying new disaster services (e.g. new sensing services), they should be added into the service repository quickly and efficiently. The addition operation could fulfill this task very well which is desirable in a search and rescue mission.

4 Multilevel Index

Service retrieval should return a service set for the service discovery and composition in such a way that it minimizes the number of searches. In the process of service retrieval, reducing the retrieval time is very crucial, thereby improving the retrieval efficiency. Information with more redundancy in a service set often consumes more time and causes unwanted delay in the retrieval process, which is undesirable especially in a disaster rescue. In our proposed model, such a redundancy is eliminated to the maximum extent. Our multilevel indexing scheme is explained as follows.

4.1. The First and Second Level Indices

Definition 4. Relation R_1 is defined on S: $\forall s_i, s_j \in S$, $s_i\ R_1\ s_j \Leftrightarrow ({}^{\bullet}s_i={}^{\bullet}s_j)\wedge(s_i{}^{\bullet}=s_j{}^{\bullet})$.

Theorem 1. R_1 is an equivalence relation on S.

Proof: (1) $\forall s\in S$, ${}^{\bullet}s={}^{\bullet}s\wedge s{}^{\bullet}=s{}^{\bullet} \Rightarrow s\ R_1\ s \Rightarrow R_1$ on S is reflexive [28]; (2) $\forall s_i, s_j\in S$, $s_i\ R_1\ s_j \Rightarrow {}^{\bullet}s_i={}^{\bullet}s_j\wedge s_i{}^{\bullet}=s_j{}^{\bullet} \Rightarrow {}^{\bullet}s_j={}^{\bullet}s_i\wedge s_j{}^{\bullet}=s_i{}^{\bullet} \Rightarrow s_j\ R_1\ s_i \Rightarrow R_1$ on S is symmetric [28]; (3) $\forall s_i, s_j, s_k\in S$, $(s_i\ R_1\ s_j)\wedge(s_j\ R_1\ s_k) \Rightarrow ({}^{\bullet}s_i={}^{\bullet}s_j\wedge s_i{}^{\bullet}=s_j{}^{\bullet})\wedge({}^{\bullet}s_j={}^{\bullet}s_k\wedge s_j{}^{\bullet}=s_k{}^{\bullet}) \Rightarrow {}^{\bullet}s_i={}^{\bullet}s_k\wedge s_i{}^{\bullet}=s_k{}^{\bullet} \Rightarrow s_i\ R_1\ s_k \Rightarrow R_1$ on S is transitive [28]. From (1)-(3), R_1 is an equivalence relation on S.□

An equivalence relation E on a set W, divides W into a family of disjoint subsets called the quotient set [28] of W induced by E. Each subset is called an equivalence class [28].

Procedure 1: Partition S according to R_1.

According to Theorem 1, R_1 as an equivalence relation on S can divide S, resulting in a quotient set, i.e., a family of disjoint service subsets.

Definition 5. The quotient set induced by R_1 is denoted as P_1. An equivalence class contained in P_1 is called a similar class, denoted as C_s.

According to R_1, a service set S is divided into many subsets, and each subset is called a similar class that contains one or more services with the same input and output parameters. Therefore, each similar class C_s has a unique pair of parameter sets, denoted as ${}^{\bullet}C_s$ and $C_s{}^{\bullet}$ where ${}^{\bullet}C_s={}^{\bullet}s$ and $C_s{}^{\bullet}=s{}^{\bullet}$, $\forall s\in C_s$.

The First Level Index (L_1I): There is an index between a service s and a similar class C_s, if $s\in C_s$.

Relation R_1 and Procedure 1 constructs L_1I as shown in the right side of Fig. 5. The definition of quotient set Q induced by an equivalence relation E on a set W, ensures every element contained in W is classified into a unique equivalence class and also states that any two different equivalence classes are disjointed. Therefore, L_1I ensures that all services are indexed only once, implying that neither service being omitted nor service being indexed twice. In other words, L_1I has the integrity and contains no redundancy.

It is observed that without the function L_1I, all the services in the repository are retrieved for computing Re(A, S). But with the presence of the function L_1I, only the similar classes are retrieved, instead of all the services in the repository. Since the similar class count is always smaller, when compared with the total number of services in the repository, L_1I reduces the retrieval time greatly. For example, given a parameter set $A=\{a\}$, and $S=\{s_1{:}a\rightarrow b, s_2{:}a\rightarrow b, s_3{:}c\rightarrow d, s_4{:}c\rightarrow d\}$, 4 services need to be retrieved to compute Re(A, S) without L_1I. Including the function L_1I, since that $C_{s1}=\{s_1, s_2\}$ with ${}^{\bullet}C_{s1}=\{a\}$ and $C_{s1}{}^{\bullet}=\{b\}$, and $C_{s2}=\{s_3, s_4\}$ with ${}^{\bullet}C_{s2}=\{c\}$ and $C_{s2}{}^{\bullet}=\{d\}$, only 2 similar classes need to be retrieved to compute Re(A, S).

Definition 6. Relation R_2 is defined on P_1: $\forall C_{s1}, C_{s2}\in P_1$, $C_{s1}\ R_2\ C_{s2} \Leftrightarrow {}^{\bullet}C_{s1}={}^{\bullet}C_{s2}$.

Theorem 2. R_2 is an equivalence relation on P_1.

This theorem can be proved in a similar way to Theorem 1.

Procedure 2: Partition P_1 according to R_2.

According to Theorem 2, R_2 as an equivalence relation on P_1 can divide P_1, resulting in a quotient set, i.e., a family of disjoint similar class subsets.

Definition 7. The quotient set induced by R_2 is denoted as R_2. An equivalence class contained in R_2 is called an input-similar class, denoted as X_{is}.

According to R_2, P_1 is divided into many subsets, and each subset is called an input-similar class that contains one or more similar classes with the same input parameters, which means, each similar class C_s contained in the same input-similar class X_{is} has the same ${}^{\bullet}C_s$. Therefore, each input-similar class X_{is} contained in R_2 has a unique parameter set, denoted as ${}^{\bullet}X_{is}$ where ${}^{\bullet}X_{is}={}^{\bullet}C_s$, $\forall C_s\in X_{is}$.

The Second Level Index (L_2I): There is an index between a similar class C_s and an input-similar class X_{is}, if $C_s\in X_{is}$.

Relation R_2 and Procedure 2 constructs L_2I as shown in Fig. 5. Since Re(A, S) focuses only on ${}^{\bullet}s$, with the function L_2I, instead of all similar classes, only all of the input-similar classes need to be retrieved. Since the input-

similar class count is smaller than the similar class count, the retrieval time is further reduced. Relation R_2 and Procedure 2 ensure that L_2I has the integrity and contains no redundancy.

The service retrieval is related only to the service input parameters. Therefore, after the construction of L_2I, L_1I has no effect on the service retrieval. But, it can improve the efficiency of the service discovery and composition and hence L_1I is preserved. Service retrieval can find a set of similar classes. If there were no L_1I, service discovery $Dc(A, B, S, L)$ would compute $B \subseteq s^{\bullet}$ for every single service. But with the aid of L_1I, service discovery needs to compute $B \subseteq C_s^{\bullet}$ only for similar classes. Since the number of similar classes is smaller than the total number of services in general, the service discovery time can be reduced. This is the reason why the first level index is preserved.

4.2. The Third and Fourth Level Indices

After developing L_1I and L_2I, a method to speed up $Re(A, S)$ is created and an inverted index is used between the input-similar classes and the parameters, so that every parameter points to those input-similar classes whose ${}^{\bullet}X_{is}$ contains this parameter. Then, while computing $Re(A, S)$, those input-similar classes pointed by the parameters contained in A need to be retrieved. However, this method cannot completely get rid of the information redundancy. For example, consider $A=\{a, b\}$, ${}^{\bullet}X_{is1}=\{a, c\}$, and ${}^{\bullet}X_{is2}=\{a, b\}$. In the inverted index method, a points to both X_{is1} and X_{is2}, and b points to X_{is2}. Both X_{is1} and X_{is2} are retrieved by a, and X_{is2} is retrieved by b. Clearly, X_{is1} is retrieved twice, thus creating redundancy. In order to avoid such redundancy, two new levels of indices are needed.

First, a concept of a key is required. A parameter in ${}^{\bullet}X_{is}$ is designated as a key of X_{is}, denoted as $\eta(X_{is})$. Each class of X_{is} is forced to pick only one key if $|{}^{\bullet}X_{is}| \geq 1$. Different input-similar classes may have the same or different keys. Next, a new level of index is delivered, and then the determination of the key for an input-similar class is discussed.

Definition 8. Relation R_3 is defined on R_2: $\forall X_{is1}, X_{is2} \in R_2$, $X_{is1}\ R_3\ X_{is2} \Leftrightarrow \eta(X_{is1})=\eta(X_{is2})$.

Theorem 3. R_3 is an equivalence relation on R_2.

This theorem can be proved in a similar way to Theorem 1.

Procedure 3: Partition R_2 according to R_3.

According to Theorem 3, R_3 as an equivalence relation on R_2 can divide R_2, resulting in a quotient set, i.e., a family of disjoint input-similar class subsets.

Definition 9. The quotient set induced by R_3 is denoted as P_3. An equivalence class contained in P_3 is called a key class, denoted as C_k.

The Third Level Index (L_3I): There is an index between an input-similar class X_{is} and a key class C_k, if $X_{is} \in C_k$.

Relation R_3 and Procedure 3 constructs L_3I, as shown in Fig. 5 and ensures that L_3I has the integrity and contains no redundancy.

According to R_3, R_2 is divided into many subsets, and each subset is called a key class that contains one or more input-similar classes with the same key, which means, each input-similar class X_{is} contained in the same key class C_k has the same key. Therefore, each key class C_k contained in P_3 has a unique key, denoted as $\eta(C_k)$. Then $\eta(C_k)=\eta(X_{is})$, $\forall X_{is} \in C_k$.

The function f: $A \rightarrow B$ is a bijection *iff* 1) $\forall x, y \in A$, $f(x)=f(y) \Rightarrow x=y$, and 2) $\forall z \in B$, $\exists x \in A$, such that $f(x)=z$. From the above analysis, it is clear that a key class has a unique key and a key identifies a unique key class. K denotes the set of keys of all key classes. Thus P_3 and K form a bijection $f_k: P_3 \rightarrow K$. Given $C_k \in P_3$ and $k \in K$, $f_k(C_k)=k$, if $\eta(C_k)=k$.

The Fourth Level Index (L_4I): There is an index between a key class C_k and a key $k \in K$, if $f_k(C_k)=k$.

The definition of bijection ensures that L_4I has the integrity and contains no redundancy.

After building L_3I and L_4I, the information redundancy contained in the prior example can be eliminated. For example, consider $A=\{a, b, c\}$, ${}^{\bullet}X_{is1}=\{a, c\}$, ${}^{\bullet}X_{is2}=\{a, b\}$, ${}^{\bullet}X_{is3}=\{b, c\}$, and ${}^{\bullet}X_{is4}=\{d\}$. If $\eta(X_{is1})=a$, $\eta(X_{is2})=a$, $\eta(X_{is3})=b$, and $\eta(X_{is4})=d$, $C_{k1}=\{X_{is1}, X_{is2}\}$, $C_{k2}=\{X_{is3}\}$, $C_{k3}=\{X_{is4}\}$, $\eta(C_{k1})=a$, $\eta(C_{k2})=b$, $\eta(C_{k3})=d$, and $K=\{a, b, d\}$. When computing $Re(A, S)$, only a and b in A are the keys, whereas c is not a key. Therefore, only C_{k1} and C_{k2} are retrieved and they are retrieved only once.

L_1I-L_4I constructs a multilevel index for the services. Fig. 5 shows the resultant structure. Since each level index has the integrity and contains no redundancy, the proposed model also ensures the integrity avoiding redundancy as a whole.

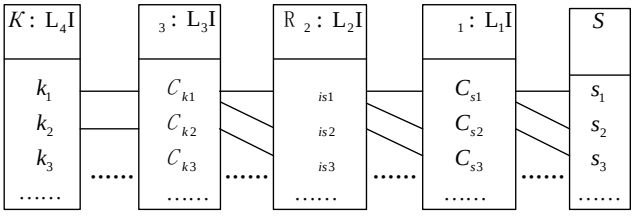


Fig. 5. The Multilevel Index Model

4.3. The Selection of a Key

A question arises as how to determine the keys of the input-similar classes. This determination is assigned as an automatic functionality in the multilevel index modeler to facilitate ease use to the service designer. The selection of the key is determined by the cardinality of C_k, denoted as $|C_k|$, in the proposed model automatically.

Theorem 4. Assume that $|R_2|=n$, $|P_3|=m$, the probability of every X_{is} being retrieved is equal, $|C_{ki}|$ is same, i=1, 2,…, m, and t denotes the time of retrieving X_{is} from R_2. When $m=\sqrt{n}$, $t=\sqrt{n}$ is minimal.

Proof: Key classes and input-similar classes constitute a two-level index. The average time to find X_{is} from R_2 without an index is $n/2$. The average time to find X_{is} with a two-level index is, $t=\frac{m}{2}+\frac{n}{2m}$. Since $\frac{dt}{dm}=\frac{1}{2}-\frac{n}{2m^2}$, solving $\frac{dt}{dm}=0$ leads to $m=\sqrt{n}$. Therefore, when $m=\sqrt{n}$, t is minimal, since $\frac{d^2t}{dm^2}=\frac{n}{m^3}=\frac{1}{\sqrt{n}}>0$. □

Theorem 4 suggests that $|C_k|$ should be as close to $\sqrt{|\Re_2|}$ as possible. For example, $R_2=\{C_{k1}, C_{k2}\}$, $|C_{k1}|=2$,

$\eta(C_{k1})=a$, $|C_{k2}|=1$, $\eta(C_{k2})=b$. An input-similar class X_{is} with its $\bullet X_{is}=\{a, b\}$ is added. If the key of X_{is} is designated to a, $|(\sqrt{|\Re_2|}-|C_{k1}|)|+|(\sqrt{|\Re_2|}-|C_{k2}|)|=|\sqrt{2}-3|+|\sqrt{2}-1|=2$. If its key is designated to b, $|(\sqrt{|\Re_2|}-|C_{k1}|)|+|(\sqrt{|\Re_2|}-|C_{k2}|)|=|\sqrt{2}-2|+|\sqrt{2}-2|\approx 1.17<2$. Therefore, its key should be designated to b in which, $|C_k|$ is as close to $\sqrt{|\Re_2|}$ as possible.

Theorem 4 presents a basic principle for the selection of keys. The operation "addition" is performed based on this principle, which is introduced in the next section.

5 Operations of the Index

This section first introduces the functions implied in the proposed model, and then, based on them, four operations, i.e., retrieval, addition, deletion and replacement have been illustrated.

5.1. The Functions in the Multilevel Index Model

Given a bijection $f:A\rightarrow B$, f^{-1} denotes its invertible function. A function $f:A\rightarrow B$ is a surjection, if and only if $\forall y\in B$, $\exists x\in A$ such that $f(x)=y$. If $f:A\rightarrow B$ is a surjection, and $C\subseteq B$, the preimage of C under f, is defined by $f^{-p}(C) = \{e \mid e\in A \wedge f(e)\in C\}$.

If a set W is partitioned by an equivalence relation E, into a quotient set Q, $f:W\rightarrow Q$ forms a surjection according to the partition relation. Then the functions can be embedded into the proposed model as shown in Fig. 6.

S is partitioned into multiple similar classes by R_1. According to Procedure 1, S and the set of all similar classes P_1, form a surjection, denoted as $f_1:S\rightarrow P_1$. P_1 is further partitioned into multiple input-similar classes by R_2. According to Procedure 2, P_1 and the set of all input-similar classes R_2, form a surjection denoted as $f_2:P_1\rightarrow R_2$. R_2 is partitioned into multiple key classes by R_3. According to Procedure 3, R_2 and the set of all key classes P_3, form a surjection, denoted as $f_3:R_2\rightarrow P_3$. The function $f_k:P_3\rightarrow K$ is a bijection.

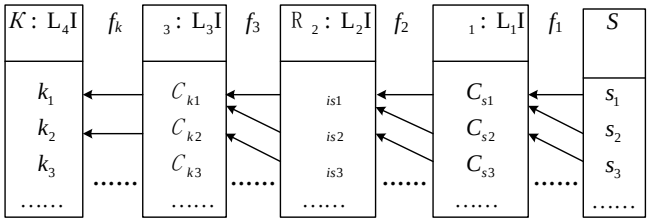


Fig. 6. The Multilevel Index Model with Functions

Next, the methodologies for operating the services in the proposed model are presented, along with the introduction of the four operations.

5.2. Retrieval

Retrieval is an important operation of the proposed model. It can be invoked by service discovery and composition methods and return service sets according to retrieval requirements. Based on the multilevel index, the retireval operation reduces their corresponding retrieval time effectively. The computation of Re(A, S) involves six steps, as shown in the following algorithm.

Algorithm 1. Retrieval.

Input A
$K=A\cap K$;
$X=\{C_k \mid C_k\in f_k^{-p}(K)\}$;
$C=\{X_{is} \mid X_{is}\in f_3^{-p}(X)\wedge \bullet X_{is}\subseteq A\}$;
$X=f_2^{-p}(C)$;
$\mathrm{Re}(A, S)=f_1^{-p}(X)$;

Theorem 5. Algorithm 1 to compute Re(A, S) is correct.

Proof: Let $P=\mathrm{Re}(A, S)=\{s \mid \bullet s\subseteq A\wedge s\in S\}$ and S' denotes the set obtained by Algorithm 1. (1) $\forall s\in S'\Rightarrow f_1(s)=C_s \Rightarrow f_2(C_s)=X_{is} \Rightarrow \bullet s=\bullet C_s=\bullet X_{is}\subseteq A \Rightarrow s\in P \Rightarrow S'\subseteq P$. (2) Given $\forall s\in P$, $f_1(s)=C_s$, $f_2(C_s)=X_{is}$, $f_3(X_{is})=C_k$, and $f_k(C_k)=k$. $\bullet s=\bullet C_s=\bullet X_{is}\subseteq A$. Since $\eta(X_{is})\in K$ and $\eta(X_{is})\in A$, we have $\eta(C_k)=\eta(X_{is})\in K$. Since f_k is a bijection, $f_k(C_k)\in K\wedge C_k\in X$. Since $\bullet X_{is}\subseteq A$ and f_3 is a surjection, $f_3(X_{is})=C_k\wedge X_{is}\in C$. Since f_2 is a surjection, $f_2(C_s)=X_{is}\wedge C_s\in X$. Since f_1 is a surjection, $f_1(s)=C_s\wedge s\in S'$. Thus, $s\in S'$ and $P\subseteq S'$. By (1) and (2), $P=S'$. Therefore, this algorithm is correct. □

Since our model is designed based on the complete index for all the services in the repository and accurate key matching, the recall of service search will always be 1. The objective of the study is not going to improve the recall and accuracy of service search in large-scale service repositories, but to develop a novel indexing module to speed up the search process and reduce the searching time. Recall the example in Section IV.B, given $A=\{b, c\}$ instead of $A=\{a, b, c\}$ and the input-similar class X_{is1} with $\bullet X_{is1}=\{a, c\}$ and $\eta(X_{is1})=a$, X_{is1} will not be retrieved for Re(A, S) since c is not a key.

If a data structure without any index, called a sequential structure is used to compute Re(A, S), the number of services needed to be retrieved is $|S|$ in any case. For an inverted index structure [23], the number of services needed to be retrieved is 0 in the best case and $|A|\times|S|$ in the worst case, whereas our proposed model needs to retrieve 0 service in the best case but $|S|$ services in the worst case.

5.3. Addition

Theorem 4 gives a basic principle for selecting the key, which is $|C_k|$ should be as close to $\sqrt{|\Re_2|}$ as possible. Based on this principle, a best-fit method is given to insert a service into the multilevel index. When a service s is inserted into the multilevel index, the first step is to identify the input parameters which are used as keys contained in K, and then the appropriate key k is selected, such that $|f_k^{-1}(k)|$ is close to $\sqrt{|\Re_2|}$. If there are more than one key available with the same cardinality, a key is selected randomly. If there is no key contained in $\bullet s$, a parameter is randomly selected from $\bullet s$, as a new key. Unfortunately, this method may make some key classes such that their cardinalities are much bigger than $\sqrt{|\Re_2|}$ when some parameters are used as input parameters by most services. To avoid this, a preset threshold value is presented to control the cardinality of a key class, with a condition that, in a service, for a key k with, $|f_k^{-1}(k)|\geq\sqrt{|\Re_2|}$, it cannot be selected as the key of s. The addition algorithm is illustrated as follows.

Algorithm 2. Addition.

Input a service s;
$K=\bullet s\cap K$;
if($K==\varnothing$){randomly select a $k\in\bullet s$ as a key of s;}
else
{$X=\{C_k \mid C_k\in f_k^{-p}(K)\}$;
find X_{is}:$X_{is}\in f_3^{-p}(X)\wedge\bullet X_{is}==\bullet s$;
if(X_{is} exists){select $k=(f_3(X_{is})$ as a key of s;}
else
{$K_{include}=\{k \mid k\in K\wedge |f_k^{-1}(k)| < \sqrt{|\Re_2|}\}$;
$K_{close}=\{k \mid k\in K_{included}\wedge |(|f_k^{-1}(k)| - \sqrt{|\Re_2|})| \le |(|f_k^{-1}(k')| - \sqrt{|\Re_2|})|, k'\in K_{included}\}$;
if($K_{close}\neq\varnothing$){randomly select a $k\in K_{close}$ as a key of s;}
else
{if($(\bullet s-K)\neq\varnothing$){randomly select a $k\in(\bullet s-K)$ as a key of s;}
else
{$K_{excluded}=K-K_{included}$;
$K_{close}=\{k \mid k\in K_{excluded}\wedge |(|f_k^{-1}(k)| - \sqrt{|\Re_2|})| \le |(|f_k^{-1}(k')| - \sqrt{|\Re_2|})|, k'\in K_{excluded}\}$;
randomly select a $k\in K_{close}$ as a key of s;}}}
if($k\notin K$){add k into K; create C_k:$C_k\in P_3\wedge\eta(C_k)=k$;}
find X_{is}:$\bullet X_{is}=\bullet s\wedge X_{is}\in f_3^{-p}(\{f_k^{-1}(k)\})$;
if(X_{is} does not exist){create X_{is}:$X_{is}\in R_2\wedge\bullet X_{is}=\bullet s$;}
find C_s:$C_s\bullet=s\bullet\wedge C_s\in f_2^{-p}(\{X_{is}\})$;
if(C_s does not exist){create C_s:$C_s\in P_1\wedge\bullet C_s=\bullet s\wedge C_s\bullet=s\bullet$;}
$C_s=C_s\cup\{s\}$;

The proposed method was built upon the inverted index which inherits all the advantages of the inverted index. The efficiency of the proposed index is higher than the inverted one, even in the worst key selection case, because it eliminates the redundancy residing in the inverted index.

5.4. Deletion and Replacement

Deletion is to delete a service from the multilevel index. It is a reverse process of addition. The deletion algorithm is as follows.

Algorithm 3. Deletion.

Input a service s;
$S'=\mathrm{Re}(\bullet s, S)$;
if($s\in S'$)
{$C_s=f_1(s)$;$C_s=C_s-\{s\}$;
if($|C_s|==0$)
{$X_{is}=f_2(C_s)$;$X_{is}=X_{is}-\{C_s\}$;
if($|X_{is}|==0$)
{$C_k=f_3(X_{is})$;$C_k=C_k-\{X_{is}\}$;
if($|C_k|==0$)
{$k=f_k(C_k)$;$P_3=P_3-\{C_k\}$;$K=K-\{k\}$;}}}}

The replacement operation can be implemented by the deletion and addition operations. Replacement requires two inputs as s and s', where s is the original service, and s' is the service to replace s. The replacement algorithm is as follows.

Algorithm 4. Replacement.

Input s and s';
Deletion(s);
Addition(s');

---

6 Flexible Deployment

The proposed multilevel index is designed for large-scale complex disaster service repositories, with four-level indexing to optimize the service discovery process. However, the four-level indexing method is not always suitable for smaller scale service repositories, since smaller repositories usually comprise only a few services with similar input and output. In response to this issue, a flexible deployment algorithm has been developed which is dynamically adjustable with the number of indexing levels, according to the characteristics of individual service repositories. This deployment algorithm is aimed at satisfying a variety of scales of service repositories showing various characteristics.

6.1. Deployment of L_4I-L_2I

According to the definition of P_1, L_1I reduces the redundancy caused by the services with the same input and output parameter sets. If there is no or few services sharing the same input and output parameter sets, the L_1I would have no or little effect on reducing the service retrieval count and time. Furthermore, it might also cause an extra overload. In this case, the multilevel index model can be deployed with the index level of L_4I-L_2I, as shown in Fig. 7.

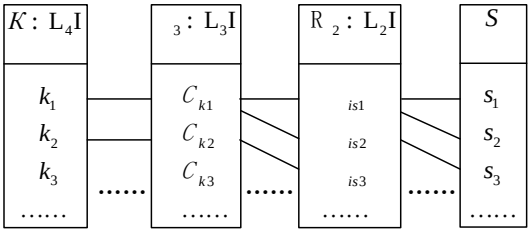


Fig. 7. The index model for L_4I-L_2I.

The operations are introduced as below for the deployment of L_4I-L_2I.

Algorithm 5. Retrieval process of the index model for L_4I-L_2I.

Input A;
$K=A\cap K$;
$X=\{C_k \mid C_k\in f_k^{-p}(K)\}$;
$C=\{X_{is} \mid X_{is}\in f_3^{-p}(X)\wedge\bullet X_{is}\subseteq A\}$;
$\mathrm{Re}(A, S)=f_2^{-p}(C)$.

Note that, f_2 is not the function from S to P_1, but it is the function from S to R_2 according to the relation $\bullet s=\bullet X_{is}$. The correctness of the retrieval can be proved in a similar way to Theorem 5.

The addition operation is shown as follows.

Algorithm 6. Addition operation of the index model for L_4I-L_2I.

Input a service s;
$K=\bullet s\cap K$;

if(K==∅){randomly select a $k\in{}^{\bullet}s$ as a key of s;}
else
{X={C_k | $C_k\in f_k{}^{p}(K)$};
find X_{is}:$X_{is}\in f_3{}^{p}(X)\wedge{}^{\bullet}X_{is}={}^{\bullet}s$;
if(X_{is} exists){select k=($f_3(X_{is})$ as a key of s;}
else
{$K_{include}$={k | $k\in K\wedge$ | $f_k{}^{-1}(k)$ | < $\sqrt{|\Re_2|}$ };
K_{close}={k | $k\in K_{included}\wedge$ | (| $f_k{}^{-1}(k)$ | - $\sqrt{|\Re_2|}$) | ≤ | (| $f_k{}^{-1}(k')$ | - $\sqrt{|\Re_2|}$) |, $k'\in K_{included}$};
if($K_{close}\neq\emptyset$){randomly select a $k\in K_{close}$ as a key of s;}
else
{if((${}^{\bullet}s$-K)≠∅){randomly select a $k\in({}^{\bullet}s$-K) as a key of s;}
else
{$K_{excluded}=K-K_{included}$;
K_{close}={k | $k\in K_{excluded}\wedge$ | (| $f_k{}^{-1}(k)$ | - $\sqrt{|\Re_2|}$) | ≤ | (| $f_k{}^{-1}(k')$ | - $\sqrt{|\Re_2|}$) |, $k'\in K_{excluded}$};
randomly select a $k\in K_{close}$ as a key of s;}}}}
if($k\notin K$){add k into K; create C_k:$C_k\in P_3\wedge\eta(C_k)=k$;}
find X_{is}:${}^{\bullet}X_{is}={}^{\bullet}s\wedge X_{is}\in f_3{}^{p}(\{f_k{}^{-1}(k)\})$;
if(X_{is} does not exist){create X_{is}:$X_{is}\in R_2\wedge{}^{\bullet}X_{is}={}^{\bullet}s$;}
$X_{is}=X_{is}\cup\{s\}$;

The deletion operation is shown as follows.

Algorithm 7. Deletion operation of the index model for L_4I-L_2I.

Input a service s;
S'=Re(${}^{\bullet}s$, S);
if($s\in S'$)
{$X_{is}=f_2(s)$;$X_{is}=X_{is}$-{s};
if(| X_{is} | ==0)
{$C_k=f_3(X_{is})$;$C_k=C_k$-{X_{is}};
if(| C_k | ==0)
{$k=f_k(C_k)$;$P_3=P_3$-{C_k};$K=K$-{k};}}}

Since the replacement operation is composed of deletion and addition, this replacement algorithm for L_4I-L_2I is the same as Algorithm 4.

6.2. Deployment of L_4I-L_3I

According to the definition of R_2, L_2I reduces the redundancy caused by the services with the same input parameter set. If there are no or few services sharing the same input parameter set, the L_2I would have no or little effect on reducing the service retrieval count and time. Similarly to the function L_1I, it might also cause an extra overload. In this case, the multilevel index model can be deployed with the index level of L_4I-L_3I. It is the lightest deployment as shown in Fig. 8.

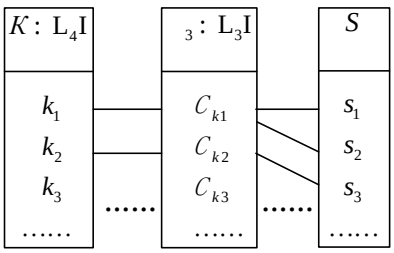


Fig. 8. The index model for L_4I-L_3I.

The f_3 embedded on the above index is not the function from R_2 to P_3, but it is the function from S to P_3 according to the relation $\eta(s)=\eta(X_{is})$. Here, a simplified retrieval operation is illustrated as follows.

Algorithm 8. Retrieval process of the index model for L_4I-L_3I.

Input A
$K=A\cap K$;
X={C_k | $C_k\in f_k{}^{p}(K)$};
Re(A, S)=$f_3{}^{p}(X)$;

Its correctness can be proved in a similar way to Theorem 5.

The addition operation is simplified as follows.

Algorithm 9. Addition operation of the index model for L_4I-L_3I.

Input a service s;
$K={}^{\bullet}s\cap K$;
if(K==∅){randomly select a $k\in{}^{\bullet}s$ as a key of s;}
else
{X={C_k | $C_k\in f_k{}^{p}(K)$};
$K_{include}$={k | $k\in K\wedge$ | $f_k{}^{-1}(k)$ | < $\sqrt{|S|}$ };
K_{close}={k | $k\in K_{included}\wedge$ | (| $f_k{}^{-1}(k)$ | - $\sqrt{|S|}$) | ≤ | (| $f_k{}^{-1}(k')$ | - $\sqrt{|S|}$) |, $k'\in K_{included}$};
if($K_{close}\neq\emptyset$){randomly select a $k\in K_{close}$ as a key of s;}
else
{if((${}^{\bullet}s$-K)≠∅){randomly select a $k\in({}^{\bullet}s$-K) as a key of s;}
else
{$K_{excluded}=K-K_{included}$;
K_{close}={k | $k\in K_{excluded}\wedge$ | (| $f_k{}^{-1}(k)$ | - $\sqrt{|S|}$) | ≤ | (| $f_k{}^{-1}(k')$ | - $\sqrt{|S|}$) |, $k'\in K_{excluded}$};
randomly select a $k\in K_{close}$ as a key of s;}}}
if($k\notin K$){add k into K; create C_k:$C_k\in P_3\wedge\eta(C_k)=k$;}
$C_k=f^{-1}(k)$;
$C_k=C_k\cup\{s\}$;

The deletion operation is simplified as follows.

Algorithm 10. Deletion operation of the index model for L_4I-L_3I.

Input a service s;
S'=Re(${}^{\bullet}s$, S);
if($s\in S'$)
{$C_k=f_3(s)$;$C_k=C_k$-{s};
if(| C_k | ==0)
{$k=f_k(C_k)$;$P_3=P_3$-{C_k};$K=K$-{k};}}

6.3. Flexible Deployment Algorithm

In this section, the facts and criterion in selecting the different deployments will be introduced.

First, the criterion about L_1I is discussed. α=| S |, β=| P_1 |, and $\lambda=\beta/\alpha$, for a non-empty service repository, $0<\lambda\leq1$. The mean time for a retrieval, without L_1I, can be defined as $T=\alpha\cdot t$, where t denotes the mean time between the service and the service under comparison. Including L_1I, the mean time for a retrieval can be defined as

$T_1=\beta(t+\Delta t_1)$, where Δt_1 denotes the mean of the overload time induced by L_1I. Then the efficiency function of L_1I can be defined as $E_1=T/T_1=1/(\lambda(1+\Delta t_1/t))$ and E_1 is dependent on the deployment environment. λ is usually decided by the service set, and t and Δt_1 are decided based on the implemental conditions, including hardware and software environments. $E_1>1$ implies that L_1I can improve the retrieval efficiency and L_4I-L_1I is the index level should be deployed here. Otherwise, L_4I-L_2I and L_4I-L_3I should be considered.

About L_2I, given $\gamma=|R_2|$ and $\lambda'=\gamma/\alpha$, the mean time for a retrieval can be defined as $T_2=\gamma(t+\Delta t_2)$ with the function L_2I, where Δt_2 denotes the mean of the overload time induced by L_2I. For the same reason as above, the efficiency function of L_2I can be defined as $E_2=T/T_2=1/(\lambda'(1+\Delta t_2/t))$. γ is decided by the service set, and t and Δt_2 are decided based on hardware and software. If $E_2>1$, L_2I should be implanted, otherwise, L_4I-L_3I should be considered.

So, the efficiency functions E_1 and E_2 are the key factors to be considered, while selecting the index levels. For a large-scale service repository, a subset of them can be selected as an experimental sample to evaluate E_1 and E_2. The following algorithm can be used to adaptively determine the deployment model, for a given service set S.

Algorithm 11. Selection of the deployment model.

Input a service set S;
If(E_1>1){deploying L_4I-L_1I;}
Else{if(E_2>1){deploying L_4I-L_2I;}
Else{deploying L_4I-L_3I;}}

Different deployment models can be transformed into each other efficiently by using the addition operation discussed in section V.C and the flexible deployment algorithm. The efficiency of transformation mainly relies on efficiency of the addition operation, which will be evaluated in the next section.

7 Experimental Results

In order to evaluate the proposed index model, an openly available test set [20, 29-31], ICEBE05 [32], is used in our experiments. The test set is comprised of 18 subsets that simulated various scenarios in the service composition. Each subset includes 11 composition queries. The contents of the test set are detailed in TABLE 1.

TABLE 1. EIGHTEEN SERVICE COMPOSITION TEST SETS

No.	Name	Service Count	No.	Name	Service Count
1	1-20-4	2156	10	2-20-4	3356
2	1-20-16	2156	11	2-20-16	6712
3	1-20-32	2156	12	2-20-32	3356
4	1-50-4	2656	13	2-50-4	5356
5	1-50-16	2656	14	2-50-16	5356
6	1-50-32	2656	15	2-50-32	5356
7	1-100-4	4156	16	2-100-4	8356
8	1-100-16	4156	17	2-100-16	8356
9	1-100-32	4156	18	2-100-32	8356

Since there is no service sharing the same input parameter sets in this test set, namely $\lambda=1$ and $\lambda'=1$, $E_1<1$ and $E_2<1$. The deployment of L_4I-L_3I is selected according to Algorithm 11. All the services in the testing set are stored in the proposed index and in the inverted index. The composition efficiencies of both of the indices are compared with a similar composition method.

Recall that service composition methods can be classified into two broad categories, the forward methods and the backward methods. A composition request can be denoted as $Q=(Q_p, Q_r)$, where Q_p is a parameter set provided by the user, and Q_r is a parameter set required by the user. The forward methods are used [16, 19, 20, 30, 33] to compose the composition result from Q_p to Q_r. The backward methods are converse in nature. The proposed model is effective only for the forward composition methods. Since the condition of the forward composition is ${}^\bullet s\subseteq A$, and thus it can be used to narrow the search space, while the condition of the backward composition is $A\cap s^\bullet\neq\varnothing$. Therefore, L_3I and L_4I are invalid for it.

One of forward composition methods, called breadth-first composition method, is used in our experiments, following Kwon *et al* [19]. This method exhibits the advantage of finding parallel services to the largest degree, with a similar time complexity of the depth-first method. Also, the breadth-first composition method can obtain the same composition results for a given composition request, regardless of the service storage structures. Therefore, this method is not affected by randomness and is fair to compare the performances of different storage structure. For example, $S=\{s_1:a\rightarrow b;\ s_2:c\rightarrow d;\ s_3:b\rightarrow f\}$, and $Q_p=\{a, c\}$ and $Q_r=\{f\}$. In the initial state, the parameter set available is $A=Q_p=\{a, c\}$. First, the breadth-first method is used to find all the services that can be invoked under A, i.e., $S'=\{s\,|\,{}^\bullet s\subseteq A\wedge s\subseteq S\}=\mathrm{Re}(A, S)$. As an outcome, s_1 and s_2 are identified. A is expanded to $A=A\cup s_1{}^\bullet\cup s_2{}^\bullet=\{a, c, b, d\}$, and $S=S-S'$. If $Q_r\vee A$ and $S'\neq\varnothing$, repeat the above process until $Q_r\subseteq A$ or $S'=\varnothing$. If $Q_r\subseteq A$, the composition request can be satisfied. In the above example, the composition result is "$s_1\,||\,s_2, s_3$", where "$s_1\,||\,s_2$" means that s_1 and s_2 can be invoked in parallel.

Note that, the above breadth-first process is one of the basic methods and it exhibits a degree of complexity during its implementation. For example, Kwon *et al*. [19] used a backward phase to prune the redundancy services, and Tang *et al*. [16] considered the behavioral-constraint compatibility of services into account. However, this does not imply that the proposed model is either ineffective or impractical. Since, service retrieval is the basic operation of service discovery and composition; it can be easily integrated with these methods. Our intention is to evaluate the effectiveness of the multilevel index model in narrowing the search space for service retrieval in a service composition process, but not to implement an advanced and complex service composition method. In this advent, the breadth-first method is well fitted to our experiments, particularly with our performance comparisons and evaluations.

In a service composition, a service should be selected from a service set to judge whether it meets a given requirement. If it is not in our interest, the next service is selected and tested; if so, proceed to the next step. The

number of services tested in the selection process, is a primary indication of the efficiency of the composition method. This primary indicator is called as the retrieval count. Fig. 9 shows the total retrieval count of 198 composition queries with a breadth-first method under two inverted and multilevel structures. The proposed multilevel index reduces 68.4% of retrievals than the inverted index.

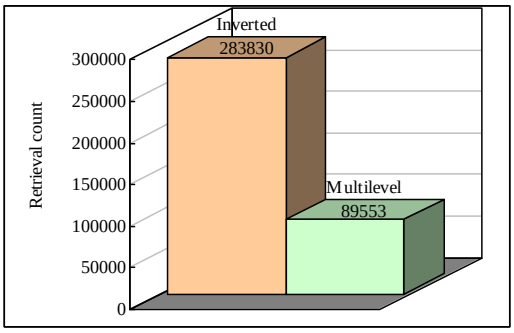


Fig. 9. Total retrieval counts of a breadth-first method under inverted and multilevel indices.

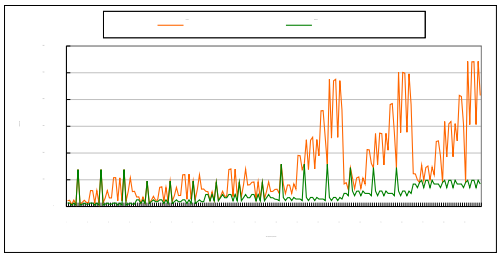


Fig. 10. Retrieval counts of a breadth-first method under inverted and multilevel indices for 198 composition requirements.

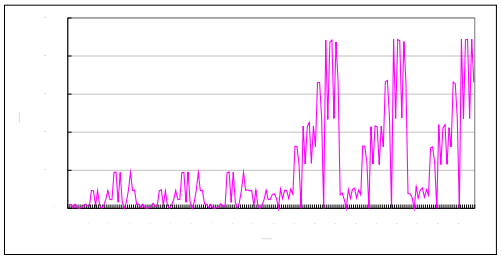


Fig. 11. Differences of retrieval counts of a breadth-first method between the inverted index and the multilevel index.

Fig. 10 shows the retrieval counts for all the 198 composition requirements in the test set. Fig. 11 shows the retrieval count differences between the inverted index and the multilevel index. The whole curve in Fig. 11 is no less than 0, which implies that the retrieval count under the multilevel index is no more than the count of the inverted index model. These results also agree with the previous theoretical analysis. There are some big fluctuations at some points in the curve of the inverted index model, which means the inverted index is not stable, namely, for some special requests, users must wait for a longer time than the response time of other requests. Clearly, the multilevel index is more stable than the inverted index. Another advantage of the multilevel index is that its increasing tendency is less than the inverted index, which means that it has more advantages in a larger service repository. Therefore, the multilevel index is very suitable for large-scale service repositories. Fig. 11 shows this advantage very clearly.

Since the proposed index is more complicated than the inverted index, it could have some overhead. Therefore, composition time is an important and also a practical indicator. Fig. 12 shows the total composition time of 198 composition queries with a breadth-first method, under both the inverted index and the multilevel index. The proposed multilevel index reduces 62.2% of the composition time than the inverted index. In a search and rescue mission during a disaster relief effort, time means lives, which is very crucial. From the obtained results, we can see that the proposed model can significantly reduce the composition time to deliver a joint search and rescue capability. The 62.2% reduction of the composition time also reflects in more than 60% of savings in the computation-related electricity power, and thus reducing the electricity cost. In the environment of sensor networks, reduced power consumption is also highly desirable, especially for those devices assisted power by batteries.

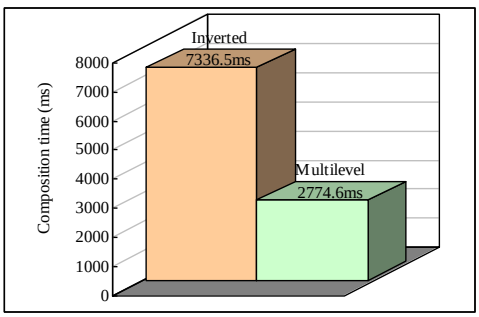


Fig. 12. Total composition time of a breadth-first method under the inverted and multilevel indices.

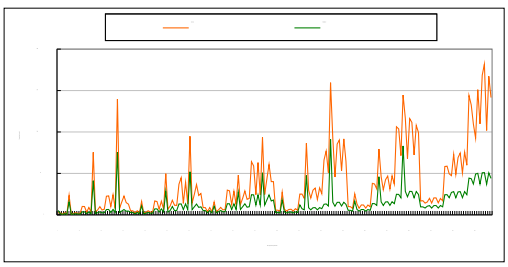


Fig. 13. Composition time of a breadth-first method under the inverted and multilevel indices for 198 composition requirements.

Fig. 13 shows the composition time for all the 198 composition queries. Fig. 14 shows the difference in the composition time, between the inverted index and the multilevel index. In Fig. 14, it can be observed that, only the first point of the curve is slightly smaller than 0, which implies that the proposed index has overwhelming

advantages. With the observation of an increasing tendency of the curve in Fig. 14, it can be concluded that with the increasing number of services, the proposed index exploits more benefits.

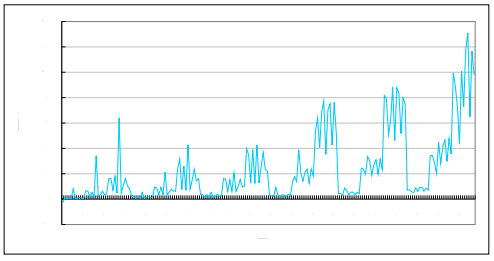


Fig. 14. Differences of composition time of a breadth-first method between the inverted index and the multilevel index.

When new services are identified in a disaster rescue scenario, they should be inserted quickly into a service repository and prepared to be invoked by other services or users. The efficiency of this operation highly depends on the efficiency of the addition algorithm. To this end, the addition operation of the multilevel index is tested, and proved its efficiency with our experiment results. The addition operation costs 1289.8ms to insert 81464 services. The deletion operation is less complex than the addition operation, and the replacement operation can be implemented by the addition and the deletion operations, respectively.

Above experiments are tested in a real-time mode. All composition and addition requests are given in real time, and they can be responded promptly, in less than 0.1 second. That means the proposed method is efficient and suitable for the dynamic disaster rescue environment.

8 Conclusion

This work proposes a multilevel index model to store and to manage the services for large-scale service repositories. Based on the theory of the equivalence relations and quotient sets, the four-level indices are developed to construct the multilevel index model. Four different operations are developed in the proposed model, in order to manage and to maintain the service repositories. The theoretical analysis and the experimental results indicate that the proposed multilevel index is more efficient for service discovery and composition than the existing inverted index method, especially for the repositories with an increasing number of services. The proposed model can be deployed adaptively and flexibly according to the characteristics of service repositories. Our experiments also prove the efficiencies of the developed operations in the index model. We also add that, in the advent of drastically expanding services, the proposed model provides a highly desirable storage structure for large-scale service repositories.

As a future work, the proposed model will be further tested in more complex and larger scale environments, in order to evaluate the influence of the potential factors those could affect the overall performance. The potential factors include the number of services included in a repository, the number of input parameters included in services and the number of services which use the same parameter as their inputs. The formal question resolved by the proposed method is how to quickly find out all subsets for a given set from a large volume of data set. Any question that can be abstracted to such a problem can use the proposed multilevel index to speed its process. Therefore, another future work is to evaluate this method for other applications.

Acknowledgment

This work is partially supported by the National Natural Science Funds of P. R. China under Grants No. 91218301, 61173016, 61332008, 90818023 and 61173042, HongKong, Macao and Taiwan Science & Technology Cooperation Program of China (2013DFM10100), Natural Science Foundation of Jiangsu Province of China (BK20130528), the Sino-UK Higher Education Research Partnership for PhD Studies and Visiting Research Fellow Program of Tongji University. Lu Liu and Changjun Jiang are the corresponding authors

References

[1] (2012). *World Disaster Report 2012*. Available: http://worlddisastersreport.org/en/data/index.html

[2] (2013). *World Disaster Report 2013*. Available: http://worlddisastersreport.org/en/

[3] W. Tan, Y. Fan, and M. Zhou, "A Petri Net-Based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language," *IEEE Transactions on Automation Science and Engineering,* vol. 6, pp. 94-106, 2009.

[4] W. Tan, Y. Fan, M. Zhou, and Z. Tian, "Data-Driven Service Composition in Enterprise SOA Solutions: A Petri Net Approach," *IEEE Transactions on Automation Science and Engineering,* vol. 7, pp. 686-694, 2010.

[5] P. Xiong, Y. Fan, and M. Zhou, "A Petri Net Approach to Analysis and Composition of Web Services," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,* vol. 40, pp. 376-387, 2010.

[6] P. Sun, C. J. Jiang, and M. C. Zhou, "Interactive Web service composition based on Petri net," *Transactions of the Institute of Measurement and Control,* vol. 33, pp. 116-132, Feb 2011.

[7] G. Liu, C. Jiang, and M. Zhou, "Process Nets With Channels," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,* vol. 42, pp. 213-225, 2012.

[8] W. Tan and M. C. Zhou, Business and Scientific Workflows: A Web Service-Oriented Approach, 1 ed.: Wiley-IEEE Press, 2013.

[9] P. W. Wang, Z. J. Ding, C. J. Jiang, and M. C. Zhou, "Design and Implementation of a Web-Service-Based Public-Oriented Personalized Health Care Platform," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* pp. 1-17, 2013.

[10] Y. Wu, C. Yan, Z. Ding, G. Liu, P. Wang, C. Jiang, *et al.*, "A Novel Method for Calculating Service Reputation," *IEEE Transactions on Automation Science and Engineering,* vol. 10, pp. 634-642, 2013.

[11] L. Liu, N. Antonopoulos, J. Xu, D. Webster, and K. Wu, "Distributed service integration for disaster monitoring sensor systems," *IET Communications,* vol. 5, pp. 1777-1784, 2011.

[12] L. Lu, D. Webster, X. Jie, and W. Kaigui, "Enabling dynamic workflow for disaster monitoring and relief through service-oriented sensor networks," in *Proc. 2010 5th International ICST Conference on Communications and Networking in China (CHINACOM)*, 2010, pp. 1-7.

[13] S. Narayanan and S. A. McIlraith, "Simulation, verification and automated composition of web services," in *Proc. the 11th international conference on World Wide Web*, Honolulu, Hawaii, USA, 2002, pp. 77-88.

[14] W. Nam, H. Kil, and D. Lee, "On the computational complexity of behavioral description-based web service composition," *Theoretical Computer Science,* vol. 412, pp. 6736-6749, 2011.

[15] G. Liu, C. Jiang, M. Zhou, and P. Xiong, "Interactive Petri Nets," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, pp. 1-12, 2012.

[16] X. Tang, C. Jiang, and M. Zhou, "Automatic Web service composition based on Horn clauses and Petri nets," *Expert Systems with Applications,* vol. 38, pp. 13024-13031, 2011.

[17] C.-S. Wu and I. Khoury, "Tree-based Search Algorithm for Web Service Composition in SaaS," in *Proc. the Ninth International Conference on Information Technology: New Generations (ITNG)*, 2012, pp. 132-138.

[18] I. Constantinescu, B. Faltings, and W. Binder, "Large scale, type-compatible service composition," in *Proc. the 2004 IEEE International Conference on Web Services*, 2004, pp. 506-513.

[19] J. Kwon, H. Kim, D. Lee, and S. Lee, "Redundant-Free Web Services Composition Based on a Two-Phase Algorithm," in *Proc. the 2008 IEEE International Conference on Web Services*, 2008, pp. 361-368.

[20] D. Lee, J. Kwon, S. Lee, S. Park, and B. Hong, "Scalable and efficient web services composition based on a relational database," *Journal of Systems and Software,* vol. 84, pp. 2139-2155, Dec. 2011.

[21] L. Aversano, G. Canfora, and A. Ciampi, "An algorithm for Web service discovery through their composition," in *Proc. the 2004 IEEE International Conference on Web Services*, 2004, pp. 332-339.

[22] E. W. Weisstein. *Binomial theorem*. Available: http://mathworld.wolfram.com/BinomialTheorem.html

[23] K. Li, L. Ying, D. Shuiguang, and W. Zhaohui, "Inverted Indexing for Composition-Oriented Service Discovery," in *Proc. the 2007 IEEE International Conference on Web Services*, 2007, pp. 257-264.

[24] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems,* vol. 30, pp. 107-117, 1998.

[25] L. A. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture," *IEEE Micro,* vol. 23, pp. 22-28, 2003.

[26] Y. Wu, C. Yan, Z. Ding, P. Wang, C. Jiang, and M. Zhou, "A Relational Taxonomy of Services for Large Scale Service Repositories," in *Proc. the 19th IEEE International Conference on Web Services (ICWS)*, 2012, pp. 644-645.

[27] K. H. Kim and H. O. Welch, "Distributed execution of recovery blocks: an approach for uniform treatment of hardware and software faults in real-time applications," *Computers, IEEE Transactions on,* vol. 38, pp. 626-636, 1989.

[28] B. Kolman, R. C. Busby, and S. C. Ross, *Discrete Mathematical Structures*, 5th ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[29] S. C. Oh, D. Lee, and S. R. T. Kumara, "Web service planner (WSPR): An effective and scalable web service composition algorithm," *International Journal of Web Services Research,* vol. 4, pp. 1-22, Jan-Mar 2007.

[30] R. Hewett, P. Kijsanayothin, and B. Nguyen, "Scalable Optimized Composition of Web Services with Complexity Analysis," in *Proc. the 2009 IEEE International Conference on Web Services*, 2009, pp. 389-396.

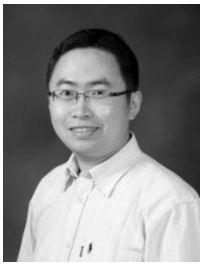
[31] S. C. Oh, D. Lee, and S. R. Kumara, "Effective Web Service Composition in Diverse and Large-Scale Service Networks," *IEEE Transactions on Services Computing,* vol. 1, pp. 15-32, 2008.

[32] ICEBE05. (2005). *Web Services Challenge 2005*. Available: http://www.comp.hkbu.edu.hk/~ctr/wschallenge/

[33] H. Mili, G. Tremblay, A. E. Caillot, R. B. Tamrout, and A. Obaid, "Web service composition as a function cover problem," in *Proc. The 2005 Montreal Conference on eTechnologies*, Montreal, Canada, 2005, pp. 61-71.

Yan WU received the M.S. degree from Shandong University of Science and Technology, Qingdao, China, in 2009, and the PhD degree from Tongji University, Shanghai, China, in 2014. He is currently a Lecturer with the School of Computer Science and Telecommunication Engineering in Jiangsu University, China. His research interests include formal methods, Service-oriented Computing, and Cloud Computing.

ChunGang YAN is currently a Professor with the Department of Computer Science and Technology, Tongji University, Shanghai, China. Her current research interests include Petri nets, formal method, workflow and service-oriented computing. She has published more than 30 publications. She has won several awards.



Lu LIU is the Professor of Distributed Computing in the University of Derby, Adjunct Professor in Jiangsu University and Visiting Research Fellow in Tongji University. Prof Liu received his PhD degree from University of Surrey (funded by DIF DTC) and MSc in Data Communication Systems from Brunel University. Prof Liu's research interests are in areas of cloud computing, service computing, peer-to-peer computing, virtual computing and system of systems engineering. Prof Liu has secured many research projects which are supported by UK research councils, BIS and RLTF as well as industrial research partners. Prof Liu has over 100 scientific publications in reputable journals, academic books and international conferences. He was recognized as a Promising Researcher by University of Derby in 2011 and received BCL Faculty Research Award in 2012. Prof Liu has chaired many international conference and workshops and has served as an Editorial Board member for several international computing journals. He is a member of IEEE and BCS.

ZhiJun DING received the M.S. degree from Shandong University of Science and Technology, Taian, China, in 2001 and the Ph.D. degree in computer science from Tongji University, Shanghai, China, in 2007. He is currently a Professor with the Department of Computer Science and Technology, Tongji University, Shanghai, China. His research interests are in Service Computing, Semantic Web, formal engineering, Petri nets, and

workflows. He has published more than 50 journal papers in domestic and international academic publications.

ChangJun JIANG received the Ph.D. degree from the Institute of Automation, Chinese Academy of Science, Beijing, China, in 1995. He is currently a Professor of Computer Science and Engineering and a Deputy President of Tongji University, Shanghai, China. He is also a Research Fellow in the City University of Hong Kong, Kowloon, Hong Kong and a Specialist of Information Area of Shanghai. His current research interests include concurrency theory, Petri nets, formal verification of software, cluster, grid technology, program testing, intelligent transportation systems, and service-oriented computing. He has been engaged in concurrency theory and concurrency processing. He has published more than 100 papers in domestic and international academic publications, including Science China, IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, IEEE TRANSACTIONS ON FUZZY SYSTEMS, International Journal of Computer Mathematics, International Journal of Computer Systems Science and Engineering, International Journal of Studies in Informatics and Control, and International Journal of Advanced System Science and Application. Furthermore, he has published four books (Science Publishing Foundation of the Chinese Academy of Science). He has taken in over 30 projects supported by the National Nature Science Foundation, National High Technology R&D Program, and National Basic Research Developing Program and other important and key projects at provincial or ministerial levels. Prof. Jiang has been the recipient of one international prize and seven national prizes in the field of science and technology. He was the recipient of the Advancement Awards of the Ministry of Education and National Excellent Doctoral Dissertation, and Excellent Youth Teacher. He is selected in the Encouragement Program for Teaching and Researching of Excellent Young Teachers by the Ministry of Education and the recipient of China National Funds for Distinguished Young Scientists.