Journal of Web Engineering, Vol. 15, No.5&6 (2016) 501-520
© Rinton Press

# EXPLORING WEB SERVICES FROM A NETWORK PERSPECTIVE USING MULTI-LEVEL VIEWS

MINGDONG TANG, FENFANG XIE, BUQING CAO, SAIXIA LYU, JIANXUN LIU

*Hunan University of Science and Technology, China*
*{tangmingdong, xiefragrance, buqingcao, saixialv, ljx529}@gmail.com*

Received February 28, 2015
Revised April 14, 2016

With the spread of service-oriented computing, more and more Web services and service-based applications emerged, and they naturally gave rise to the service ecosystem. The exploration of Web services and the service ecosystem has recently attracted considerable attention from researchers. Since service-based applications such as Mashups are usually developed via composing Web services, the collaboration relation of Web services in Mashups can be used to build Web service networks. To help understanding the characteristics of Web service networks, we propose a three-level view model. With the proposed model, Web service networks can be explored from different-level views such as service-level, tag-level and domain-level, with each level capturing different knowledge of the Web service networks. In the following paper, we firstly describe in detail the three-level view model for Web service networks. Then, based on the model, we present an experimental analysis on real Web service data and report the results. Finally, a visual analysis tool for exploring Web service networks using different-level views is presented.

*Key words*: Service network, Service composition, Web services, ProgrammableWeb, Mashup
*Communicated by*: M. Gaedke & Q. Li

## 1 Introduction

In the Web 2.0 age, numerous open services have been published through Application Programming Interfaces (API) on the Web by various software venders and IT enterprises. Meanwhile, more and more developers are throwing themselves into composing different services to develop innovative applications for value-added service provision. Mashup is among the most prevalent approaches to composing Web services to create new applications [1]. The term "Mashup" is borrowed from pop music, where it denotes remixing songs (or parts of songs) to create new derivative works. Similarly, Web-based Mashups are created by integrating existing services on the Web using their APIs, typically in a way that hides the details of the source applications to provide a seamless experience for the user. With the increase of the amount of Web services and Mashups, the Web service market grows rapidly in recent years. For example, in *ProgrammableWeb.com* (a popular Web service and mashup repository), there are more than 10,000 Web APIs and 7,000 Mashups according to the latest statistics. Web APIs (sometimes abbreviated as APIs) are typically referred to a class of light-weighted Web services such as RESTful services, which are gaining increasing popularity in the Web2.0 age. In this paper, we do not distinguish between Web APIs and Web services and thus will use the two concepts interchangeably.

The growth of Web services and Mashups gave rise to the so-called Web service ecosystem [2-3], which represents a system with large amount of interconnected units and complex interactions. The collaboration relation of services in Mashups has recently attracted much attention from researchers in the Service Computing field [4-5]. The benefits of investigating the collaboration relation between services are multi-fold. First, investigating the collaboration relation among Web services can tell us Web services' ability to be composed for Mashup creation. Second, being aware of collaboration relation among Web services can provide the guide for efficient Web service discovery and composition. Third, with the collaboration relation among Web services, a set of Web services can naturally be represented under the form of networks, and such kind of structures constitutes a convenient way to represent a set of Web services for visualization and analysis purposes [6].

Web service networks provide an effective way to structure and organize Web services. Though a few works has investigated the model of Web service networks and analysed their structures, there still lacks a clear way to understand the Web service networks. The understanding of the Web service ecosystem is also limited. With the rapid growth of Web services, these issues pose even more challenges. In order to better understand the structure of the Web service ecosystem, this paper presents a multi-level view model for exploring Web service networks. The model uses three-level views for depicting a Web service network, i.e., service-level view, tag-level view and domain-level view. Our idea is inspired by the operations of an electronic map, which usually contains multi-layers of information that provide rich functionality for users. When the map zooms in or zooms out, different layers of information will be exhibited, and thus the user can see the map from different detail levels. By this means, the user can efficiently locate places or routines that he/she is interested. In summary, we made the following contributions in this paper:

- *We propose a three-level view model for exploring Web service networks, which is built based on the collaboration relation between services. Several effective methods for constructing the three-level views were developed as well.*
- *We also propose a novel method for ranking tags of Web services based on their importance to a service, which can be used to refine categorization of Web services.*
- *We conduct an empirical study using a real Web service dataset, and analyze the constructed Web service network using different views. Not only the structural properties of the three views of the Web service network but also some interesting connection patterns are revealed through analysis.*
- *We develop a software tool to implement our proposed model, which can visualize a Web service network in a way like operating electronic maps.*

The rest of this paper is organized as follows. Section 2 surveys related work. Section 3 describes an emerging Web service ecosystem, by which we can build Web service networks. Section 4 presents the three-level view model for Web service networks, including the service-level view, the tag-level view and the domain-level view. Section 5 employs the proposed model to analyse the structures of a Web service network generated using real Web service data. Section 6 presents a visual analysis tool for Web service networks. Finally, Section 7 concludes this paper.

## 2 Related Work

As the number of Web services available increases, how to manage Web services for efficient discovery and composition has become a vital issue [7]. To address this issue, a few works propose to structure and organize Web services as a network, so that the relationships between Web services can be employed for efficient Web service discovery and composition. In the following, we survey related work in this area.

To structure Web services, two kinds of criteria were usually employed by previous works, i.e., similarity criteria and dependency criteria. By measuring similarity between Web services according to some similarity criteria, the Web services can be grouped into categories usually called communities, and thus automated Web services classification becomes feasible. One significant benefit of grouping similar Web services is that, it can help improving the performance of Web services substitution when a Web service fails in a composition process. The most popular approach to measuring similarity between Web services is via analyzing the description text of Web services such WSDL files. Depending on whether the Web services are described in syntax or semantics, many syntactic or semantic approaches have been proposed for accurately measuring similarity between Web services [8-11]. The dependency between Web services indicates how a Web service depends on another in fulfilling some functionality, in other words, the ability of the two services to be composed. In this regard, the dependency relation between Web services is similar to the collaboration relation.

There also have been a number of studies investigating the dependency relation among Web services. Aydogan et al. [12] defined the dependency relation between services' input and output, and proposed a service dependency graph model, based on which an efficient Web service composition algorithm was developed. Liu et al. [13] proposed a method to mine the dependency relation between two Web service operations, which computes the similarity between a source operation's output parameters and a target operation's input parameters and presumes that a high similarity indicates a dependency relation between the two Web service operations. In [14] the authors defined three composition network models according to the node types that can be parameters, operations or Web services. They used syntactic description information of Web services to build networks and used complex network theory to provide an analysis of the topological structure of Web services networks formed by a real-world data set. Cherifi et al. [15] used both syntactic and semantic Web services to build networks, and compared the topological properties of the two kinds of Web service networks. Feng et al. [16] also proposed a semantic approach to build Web service networks and discussed its use in Web service discovery. However, most of the above works considered only standard Web services which are typically well described in WSDL or other structured languages, and the dependency relation does not mean the real collaboration relation between Web services.

In Web 2.0 age, more and more light-weight Web services (such as RESTful services and Web APIs) emerged and have constituted the majority of Web services on the Internet. Different of standard, heavy-weight Web services, these Web services are usually described with pure natural or semi-structured languages such as HTML. This indicates that the aforementioned works may be impractical to them. Moreover, there is an increasing interest for Web 2.0 users to annotate Web services with tags and Mashup them to develop new Internet-scale applications. These user-generated data, however, were seldom exploited by aforementioned works. Only a few works have addressed this issue and began to employ the user-generated data to infer relation between Web services and build Web service networks. Chen et al. [17] took the user-generated tags on Web services into consideration, and combined them with WSDL for accurate similarity measurement between Web services. [4, 18, 5] employed the Mashup data in *ProgrammableWeb.com* to infer the collaboration relation between Web services, and investigated both the static structure and dynamic evolution of Web service network. Our previous works [19, 20] have investigated both the similarity relation and collaboration relation between Web services by employing user-generated data.

Most of the previous studies on Web service networks focused only on the service level. However, with the exponential growth of Web services, analyzing and visualizing the Web service networks becomes a challenging issue, which the previous work may not address well. This work can deal with the above issue by using a three-level view model for Web service networks. The model not only can provide a better method for structuring and visualizing Web services but also can help users understanding the Web service ecosystem and discovering appropriate Web services in a more efficient way.

## 3 The Emerging Web Service Ecosystem

To investigate the emerging Web service ecosystem and explore the Web service networks, we turn to the largest online repository of Web services and Mashups, *ProgrammableWeb.com*. This Web service repository provides the most comprehensive listing of Mashups and Web APIs available, including information on which Mashups use which APIs. It also has a number of tags for annotating APIs and Mashups.

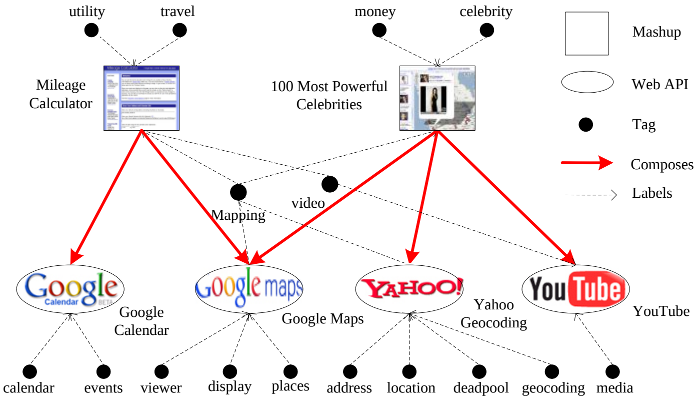


Figure 1 A slice of the Web service ecosystem

Figure 1 presents a slice of the Web service ecosystem represented by *ProgrammableWeb.com*. Let rectangles denote Mashups, ellipses denote APIs, and filled dots denote tags annotating APIs and Mashups. As can be seen from figure 1, there are two Mashups----*100 Most Powerful Celebrities* and *Mileage Calculator*, four APIs---- *Yahoo Geocoding*, *YouTube*, *Google Calendar and Google Maps*, and sixteen tags including *utility*, *travel*, *money* etc. Let a solid arrow connecting a Mashup with an API denote that the API is a component of the Mashup, and a dash arrow connecting a tag with a Mashup or an API denote that the Mashup or API is commented by the tag. Clearly, a Mashup is likely to use multiple APIs and two Mashups are likely to share the same APIs. We define that if two APIs have been composed to create at least one Mashup, there exists a collaboration relationship between them. Based on the collaboration relationships between Web services, a Web service network can be constructed.

TABLE I OVERVIEW OF THE WEB SERVICE DATA COLLECTED FROM PROGRAMMABLEWEB.COM

| | |
|---|---:|
| Number of Mashups | 6,970 |
| Number of API categories | 67 |
| Number of APIs | 9,135 |
| Number of tags labelling APIs | 1,727 |
| Number of APIs per category | 136.343 |
| Number of APIs per Mashup | 2.017 |
| Number of tags per API | 3.275 |
| Number of APIs used by Mashups | 1,193 |

TABLE II CATEGORIES OF WEB SERVICES

| Category | #APIs | Category | #APIs |
|---|---:|---|---:|
| 1-Tools | 624 | 35-Backend | 79 |
| 2-Internet | 549 | 36-News | 79 |
| 3-Social | 467 | 37-Weather | 74 |
| 4-Financial | 402 | 38-Real Estate | 73 |
| 5-Enterprise | 378 | 39-Entertainment | 71 |
| 6-Reference | 326 | 40-Blogging | 66 |
| 7-Mapping | 323 | 41-Recommendations | 65 |
| 8-Shopping | 313 | 42-Retail | 63 |
| 9-Science | 300 | 43-Food | 62 |
| 10-Government | 286 | 44-File Sharing | 60 |
| 11-Telephony | 274 | 45-Media Management | 55 |
| 12-Messaging | 269 | 46-Job Search | 52 |
| 13-Payment | 262 | 47-Chat | 51 |
| 14-Search | 237 | 48-Bookmarks | 41 |
| 15-Photos | 215 | 49-Feeds | 40 |
| 16-Other | 208 | 50-PIM | 33 |
| 17-Video | 201 | 51-Widgets | 32 |
| 18-Advertising | 195 | 52-Calendar | 30 |
| 19-Travel | 189 | 53-Answers | 30 |
| 20-Education | 188 | 54-Fax | 21 |
| 21-Music | 182 | 55-Dictionary | 21 |
| 22-Email | 176 | 56-Tagging | 17 |
| 23-Security | 173 | 57-Wiki | 13 |
| 24-Utility | 151 | 58-Media Search | 12 |
| 25-Transportation | 144 | 59-Politics | 12 |
| 26-Games | 141 | 60-Blog Search | 11 |
| 27-Project Management | 125 | 61-Goal Setting | 5 |
| 28-Medical | 114 | 62-Dating | 3 |
| 29-Sports | 102 | 63-Catalog | 2 |
| 30-Storage | 96 | 64-Auctions | 2 |
| 31-Events | 88 | 65-Other Search | 1 |
| 32-Office | 86 | 66-Specific website | |
| 33-Database | 82 | traffic ranking service | 1 |
| 34-Shipping | 81 | 67-Portal | 1 |

To explore the characteristics of real Web service ecosystem and Web services, we collected all Mashups and APIs as well as their description information from *ProgrammableWeb.com* in June, 2013. The numbers of Mashups and APIs are 6,970 and 9,135 respectively. The description information of each Mashup includes the name, URL, tags, APIs they invoked etc. The description information of each API includes the name, URL, category, tags, comments etc. Table I overviews the Web service data crawled from *ProgrammableWeb.com*. As it shows, only 1,193 APIs among 9135 APIs have been used by Mashups. The low use rate of APIs in this ecosystem indicates that the Web service ecosystem, though has been growing rapidly, is still in early development. At *ProgrammableWeb.com*, the 9135 APIs have been classified into 67 categories with each category representing a set of APIs in a specific domain. Table II lists all API categories by ranking them in descending order according to the number of APIs they have. We can see that, the category of *Tools* has the most APIs, followed by the API categories of *Internet* and *Social*.

To study the use rate of Web APIs, for each API we counted the number of Mashups that have used it. Table III presents the top-10 most frequently used APIs. We can see that *Google Maps* is the most popular Web API since it has been used by more than 2400 Mashups. By summing up the use times of all APIs, we obtain the total use times of all APIs, i.e., 20,250. The percentage of the use times of the top-10 APIs over all is 6190/20250=30.568%, which is quite high since there are totally 9135 Web services and the top-10 APIs are only a small fraction of them. This result indicates that the distribution of use rate of APIs is quite heterogeneous. This may be caused by that users usually believe that they can learn from the historical usage of the services and tend to build new Mashups by reusing popular ones. Consequently, the popular APIs get even more popular.

TABLE III THE TOP-10 MOST USED APIS

| No. | API | #Mashups | No. | API | #Mashup |
|---|---|---|---|---|---|
| 1 | Google Maps | 2,417 | 6 | Facebook | 385 |
| 2 | Twitter | 757 | 7 | Twilio | 349 |
| 3 | YouTube | 651 | 8 | Last.fm | 225 |
| 4 | Flickr | 605 | 9 | eBay | 216 |
| 5 | Amazon Product Advertising | 406 | 10 | Google Search | 179 |

TABLE IV TOP-10 MOST POPULAR TAGS

| No. | Tag | #APIs | No. | Tag | #APIs |
|---|---|---|---|---|---|
| 1 | Deadpool | 1,088 | 6 | Mobile | 558 |
| 2 | Social | 993 | 7 | Reference | 506 |
| 3 | Tools | 829 | 8 | Search | 489 |
| 4 | Internet | 793 | 9 | Shopping | 465 |
| 5 | Enterprise | 579 | 10 | Mapping | 461 |

Figure 2 shows the cumulative distribution of the use times of APIs (plotted on log-log axes). The *X* axis (*x*) denotes the use times, and the *Y* axis denotes the number of APIs which have been used by at least *x* Mashups. Evidently, the distribution exactly follows a power-law distribution.

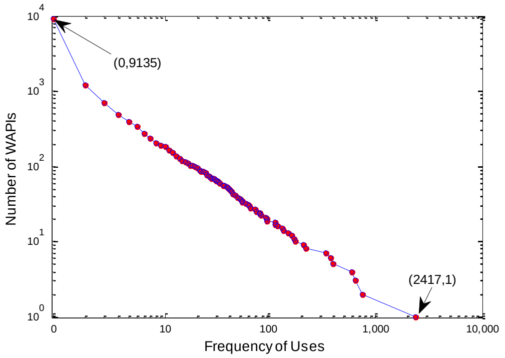


Figure 2 The cumulative distribution of use times of APIs

To facilitate understanding and promote discovery of APIs, a few tags are usually employed for annotating the functionality or other aspects of an API. The use times of a tag can reflect its popularity degree too. Thus, we also counted the usage times of each tag on the collected Web service data. Table IV shows the top-10 most popular tags. As it shows, the tag with maximum use times is *Deadpool*, followed by *Social*, *Tools*, ect.

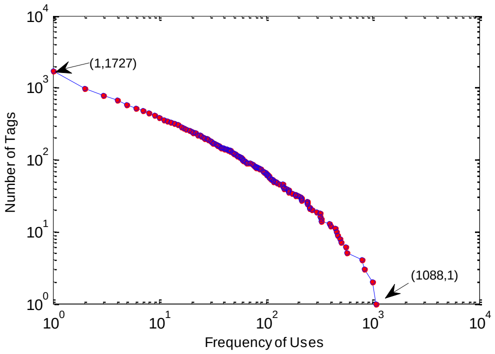


Figure 3 The cumulative distribution of use times of tags

Figure 3 shows the cumulative distribution of use times of tags (again, plotted on log-log axes). From the figure we can observe that the distribution also follows a power-law distribution to some extent.

The above discussion presents the general characteristics and statistics of the Web service ecosystem represented by *programmableWeb.com*. In this following, we focus on the Web service networks formed by the collaboration relation between Web services. Generally speaking, the more Mashups two Web services concur in, the stronger is the collaboration relation between the two Web services. Because of the large and ever-increasing number of Web services, how to visualize the Web service network in a better way is a challenging issue.

## 4 The Three-level View Model of Web Service Networks

In this section, we propose a three-level view model for analysing Web service networks which are formed by the collaboration relation between Web services. The bottom-level view of a Web service network is itself, which is named as the service-level view. This view is formed by the concrete services and their collaboration relationships. The middle-level view of a Web service network is named as the tag-level view, which is formed by tags that annotate the functionality or other aspects of Web services. The top-level view of a Web service network is named as by the domain-level graph, which is formed by service domains or categories. Both the domain-level view and the tag-level view can be seen as an abstraction of the service-level view. The tag-level view is actually a refinement of the domain-level view. With this model, the Web service ecosystem and the Web service network can be understood more clearly from different levels. Specifically, our proposed three-level view model will answer the following questions:

- *What Web services can be composed for creating Mashup applications?*
- *What kinds or what domains of Web services are usually employed for creating Mashup applications?*
- *Are there any interesting composition patterns for Web services?*

Before introducing the proposed three-level view model, for the convenience of description, we define the following notation that will be used throughout this paper.

- $A=\{A_1,A_2,\dots,A_m\}$ *represents a set of Web services;*

- *$M$={$M_1$,$M_2$,…,$M_n$} represents a set of Mashups;*
- *$T$={$T_1$,$T_2$,…,$T_l$} represents a set of tags;*
- *$D$={$D_1$,$D_2$,…,$D_k$} represents a set of Web service domains or categories, which satisfy $A$=$D_1 \cup D_2 \cup \ldots \cup D_k$;*
- *$ES$=($D$,$A$,$M$,$T$,$E$) represents a Web service ecosystem, where $E$ is the set of relationships among Web services, Mashups, tags, and service domains;*
- *$M(A_i)$ represents a subset of Mashups that invoke Web service $A_i$;*
- *$T(A_i)$ represents a subset of tags that annotate Web service $A_i$;*
- *$D(A_i)$ represents a Web service domain that contains Web service $A_i$.*

Next, we will describe the definitions of the above three-level views for exploring Web service networks and their construction methods.

### *4.1. The Service-Level View*

The service-level view of a Web service network can be denoted by graph $G_A$=($A$,$E_A$,$W_A$), where the vertex set $A$ represents the set of Web services, the edge set $E_A$ represents the set of collaboration relationships between Web services, and $W_A$ is the set of weights on the edges indicating the strength of composition relationships. This graph is intended to answers the follow questions: What Web services can be composed for creating Mashups? And how to measure strength of the collaboration relationships between Web services?

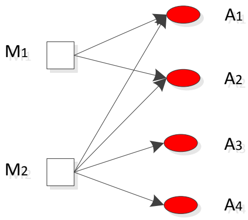


Figure 4 A toy example of the Mashup-service network

To measure the strength of the composition relationship between Web services, the following are some heuristics that can be exploited:

- *Generally, the more Mashups that two services concur in, the stronger is the collaboration relationship between them, i.e., the greater is the ability of the two services to be composed.*
- *Given two Web services (e.g., $A_1$ and $A_2$) that concur in two Mashups $M_1$ and $M_2$, if $M_2$ contains more Web services than $M_1$ (as shown in Figure 4), then the contribution of $M_2$ to the collaboration relation between the two Web services should be less than that of $M_1$.*
- *The strength of the collaboration relationship between two Web services can also be reflected by the ratio of the number of Mashups that invoked both of them over the number of Mashups that invoked either of them. A high ratio indicates that the ability of the two services to be composed is high, and vice versa.*

Having the above heuristics in mind, we developed the following formula to calculate the strength of the collaboration relationship between two Web services $A_i$ and $A_j$:

$$W_A(A_i, A_j) = \frac{\sum_{m\in(M(A_i)\cap M(A_j))} \frac{2}{k(m)}}{(|M(A_i)\cup M(A_j)|)^{\alpha}} \quad (1)$$

where $M(A_i)$ represents the set of Mashups which have invoked Web service $A_i$, $M(A_j)$ represents the set of Mashups which have invoked Web service $A_j$, $M(A_i)\cap M(A_j)$ represents the subset of Mashups which have invoked both $A_i$ and $A_j$, $M(A_i)\cup M(A_j)$ represents the subset of Mashups which have invoked either $A_i$ or $A_j$, and $k(m)$ represents the number of Web services used by a Mashup $m$. The parameter $0\leqslant\alpha\leqslant1$ is used to adjust the influence of the number of Mashups invoking either $A_i$ or $A_j$ on their collaboration relationship's strength; the larger the value of $\alpha$, the greater the influence, and vice versa.

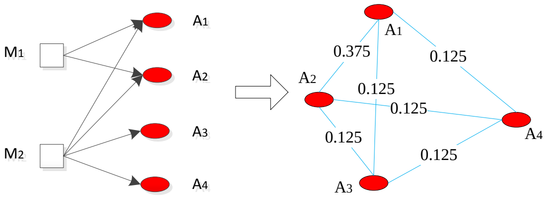


Figure 5 Illustration of the service view construction

Figure 5 is an example illustrating the construction of a service-level graph by measuring the collaboration relationships between given Web services. Suppose that Mashup $M_1$ is composed of two Web services $A_1$ and $A_2$, and Mashup $M_2$ is composed of four Web services $A_1$, $A_2$, $A_3$ and $A_4$. Let $\alpha$=1, then, with Formula (1), the strength of the collaboration relationships, i.e., the weights on the links between the above Web services can be calculated as:

$$W_A(A_1, A_2) = \frac{1+1/4}{4} = \frac{3}{8} = 0.375$$

$$W_A(A_1, A_3) = \frac{1/2}{4} = \frac{1}{8} = 0.125$$

$$W_A(A_2, A_3) = \frac{1/2}{4} = \frac{1}{8} = 0.125$$

$$W_A(A_3, A_4) = \frac{1/2}{4} = \frac{1}{8} = 0.125$$

The right diagram in Figure 5 shows the derived service-level graph.

The service-level graph can be transformed into an $n\times n$ matrix, denote by $AA$, where $n$ is the number of services. That is, $AA=[aa_{ij}]$, $0\leqslant i,j\leqslant 1$, where $aa_{ij}=W_A(A_i,A_j)$. If $A_i$ and $A_j$ concur in at least one Mashup, we have $aa_{ij}>0$, otherwise $aa_{ij}=0$.

### *4.2. The Domain-Level View*

The domain-level view is the top-level view of the Web service network in our proposed model. Different from the service-level view, it captures the collaboration relation between two abstract types of Web services, instead of two concrete Web services. It answers the question: What kinds of Web services are more likely to be combined to create new applications? The domain-level view can also be represented using an undirected weighted graph $G_D$=($D$, $E_D$, $W_D$), where the vertex set $D$ represents the set of Web service categories, the edge set $E_D$ represents the set of connections between Web service categories, and $W_D$ is the set of weights on the edges which indicate the strength of connections.

The domain-level view can be constructed from the service-level view by applying the rule that if there exist services in two domains that have concurred in at least one Mashups, i.e., have direct connections in the service-level graph, the two service domains shall be connected using a link. For example, given two service domains $D_i$ and $D_j$, an edge $E_D(i,j)$ will be generated between them if only there exist services $A_i \in D_i$ and $A_j \in D_j$, have concurred in at least one Mashups. Suppose there are two different service domains $D_i$ and $D_j$, the weight on the edge ($D_i$,$D_j$) between $D_i$ and $D_j$, i.e., $W_D(i,j)$, is calculated as follows:

$$W_D(D_i, D_j) = \frac{\sum_{a\in D_i \wedge b\in D_j} W_A(a,b)}{\sum_{a\in D_i} W_A(a,*) + \sum_{b\in D_j} W_A(b,*) - \sum_{a\in D_i \wedge b\in D_j} W_A(a,b)} \tag{2}$$

where the numerator $\sum_{a\in Di \wedge b\in Dj} W_A(a,b)$ is the sum of the weights on the edges connecting services in $D_i$ with services in $D_j$; the denominator is actually the sum of the weights on the edges which connect services in $D_i$ or $D_j$ with other arbitrary services. Specifically, we use $\sum_{a\in Di} W_A(a,*)$ to represent the sum of the weights on the edges connecting services in $D_i$ (e.g., $a$) with arbitrary services, and $\sum_{b\in Dj} W_A(b,*)$ to represent the sum of the weights on the edges connecting services in $D_j$ (e.g., $b$) with arbitrary services. Please note that Eq. (2) is not only available for two different service domains. It can also be used for measure the weight on the link connecting a domain with itself.

To illustrate the construction of the domain-level view, Figure 6 provides an example. Suppose that Web services $A_1$, $A_2$ and $A_3$ belong to service domain $D_1$, $A_4$ and $A_5$ belong to service domain $D_2$, and $A_6$ belongs to service domain $D_3$. With Eq. (2), the connection weights between $D_1$ and $D_2$ is calculated as follows: $W_D(D_1,D_2)$= (0.667 + 0.250 + 0.333) / (0.667 + 0.250 + 0.333 + 0.333 + 0.500 + 0.250 + 0.333 + 0.500) = 0.395. Likewise, the connection weights between the other service domains can be obtained. Figure 6 shows the results.

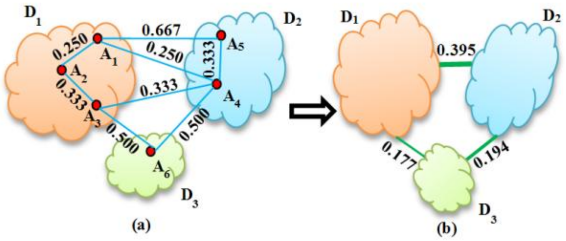


Figure 6 Illustration of the domain-level view construction

### 4.3. The Tag-Level View

The domain-level view can reveal some top-level characteristics of a Web service network. However, due to the limited number of Web service domains, the domain-level view may be too “rough” to depict the Web service network. For instance, at *ProgrammableWeb.com*, there are only 67 Web service domains (i.e., abstract types). Furthermore, it is unreasonable to say that a Web service belongs to only one domain. To refine the service domain-level view and allow a service belonging to multiple domains, we therefore introduce the tag-level view of the Web service network. A tag of a service can help identifying a category of services, and a service usually has a few tags, as indicated by *ProgrammableWeb.com*. We consider that different tags make different contributions to the identification of services in a Web service ecosystem. The difference of contribution of a tag is represented as a weight that can be treated as the degree of the tag in contribution to the identification of services.

The following discussions describe how to build the tag-level view based on the service-level view. The tag-level view can also be modelled as an undirected weighted graph $G_T=(T,E_T,W_T)$, where the vertex set $T$ represents the set of tags for annotating services, the edge set $E_T$ represents the set of edges representing links between service tags, and $W_T$ is the set of weights on the edges, indicating the strength of links between service tags. Two tags $T_i$ and $T_j$ in the tag-level graph are linked if only there are at least two services annotated by $T_i$ and $T_j$ respectively that have concurred in the same Mashup, i.e., have a direct link in the service-level graph. To measure the weights on edges in the tag-level graph, we propose a tag ranking method for services, which evaluates the degree of membership of a service to the sub-domain represented by a tag, or in other words, the importance weight of a tag to a service. This is done by exploiting various types of description information of services, such as service name, service category name and comments.

Suppose that service $A_i$ has $m$ tags $T_1,T_2,\ldots,T_m$, and its description information such as name, category name, and comments are denoted by $d_{i1}$, $d_{i2}$, $d_{i3}$ respectively, then the importance weight of a tag $T_j$ to $A_i$ is calculated as follows:

$$IM(T_j, A_i) = \frac{\sum_{k=1}^{3} f_i * (Occur(T_j, d_{ik}) + 1)}{\sum_{j=1}^{m} \sum_{k=1}^{3} f_i * (Occur(T_j, d_{ik}) + 1)} \tag{3}$$

where $Occur(T_j,d_{ik})$ is the occurrence number of $T_j$ in $d_{ik}$, and $f_1$, $f_2$, $f_3$ are the tunable weights which respectively represent the importance of the service's name, category name and comments. From our intuitions, a tag is more important to a service if it appears in the service's name or category name. Therefore, we rank $f_1$, $f_2$, $f_3$ as $f_1 \geq f_2 \geq f_3$. Based on the service-level graph, we use the following equation to calculate the weight of the edge between two tags $T_i$ and $T_j$ on the tag graph:

$$W_T(T_i, T_j) = \sum_{a \in A(T_i), b \in A(T_j)} (W_A(a,b) * IM(T_i, A_u) * IM(T_j, A_v)) \tag{4}$$

where $A(T_i)$ represents the subset of Web services with the tag $T_i$, and $A(T_j)$ represents the subset of Web services with the tag $T_j$. Figure 7 is an example illustrating how to use Equation (4) to calculate the weight of an edge between two tags and how to construct the tag-level graph of a Web service network. The left diagram of Figure 7 represents a simple service graph, which includes two Web services and five tags. Suppose that $WS_1$ have two tags $T_1$ and $T_2$, and $WS_2$ has three tags $T_3$, $T_4$ and $T_5$. The weights of edges between services as well as edges between services and tags are provided. With Eq. (4), we can calculate the weights of edges between these tags, as shown in the right diagram of

Figure 7. For instance, taking $T_1$ and $T_3$ into consideration, we have $W_T(T_1,T_3)$=0.26*(0.154*0.589)=0.022.

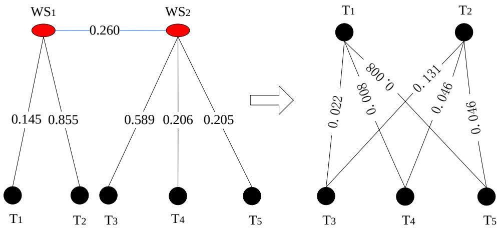


Figure 7 Illustration of the tag-level view construction

## 5 Empirical Analysis

In this section, we employ the proposed three-level view model to analyze the Web service network which is built using the real data collected from ProgrammableWeb.com. The general statistics, structural properties, and some vital patterns of each view of the Web service network are presented.

### 5.1. Analysis on the service-level view

We firstly employ the *ProgrammableWeb.com* dataset to construct the service-level view and measure the connection weights between every pair of services using Eq.(1). With the setting $\alpha$=1, Table V presents the top-10 API pairs that have the largest connection weights and their corresponding weight values.

TABLE V THE TOP-10 API PAIRS WITH THE LARGEST CONNECTION WEIGHTS

| **Web API** | **Web API** | **Connection Weight** |
|---|---|---|
| Twilio | Twilio SMS | 1.000 |
| Mendeley | PLoS Search | 0.365 |
| Travel Booking Engine | Travelport | 0.249 |
| Hoiio Voice | Hoiio SMS | 0.239 |
| Google Maps Flash | MapLarge | 0.236 |
| FanFeedr Sports News | FanSnap | 0.224 |
| HAMweather Aeris | Handset Detection | 0.224 |
| TelAPI | Zapier | 0.224 |
| PriceSpin | Primal | 0.224 |

Figure 8 uses a grid to visualize the collaboration relationships between the top-50 most popular services, with their weights being indicated by the darkness of cells, i.e., darker cells indicate larger weights. The numbers on the horizontal and vertical axes are IDs of the services, which are assigned to services according to their positions in the descending ranking of services based on their usage times. That is, the service with ID *x* indicates that it is the *x*-th frequently used service. From Figure 8 we can see, there are quite a few cells which are apparently darker than the other cells. For instance, the cell (12, 7) has the darkest shading, indicating that the corresponding two services *Twilio SMS* and *Twilio* have the largest connection weight according to Eq. (1).

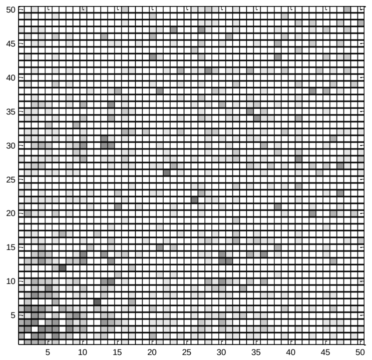


Figure 8 Visualization of the connection weights between the top-50 most popular APIs

The structure of the service-level graph is also shown in Figure 9. A bigger node is used to represent a Web service with a higher connection degree. To filter weak connections, we set a threshold 0.02 for the connection weight, such that the edges whose weights are smaller than 0.02 are not taken into account. We also analyzed the structural properties of the generated service-level graph, as shown in Table VI. We can see that it has a very small diameter and a high clustering coefficient, which indicate that the service graph has a relatively strong small-world property.

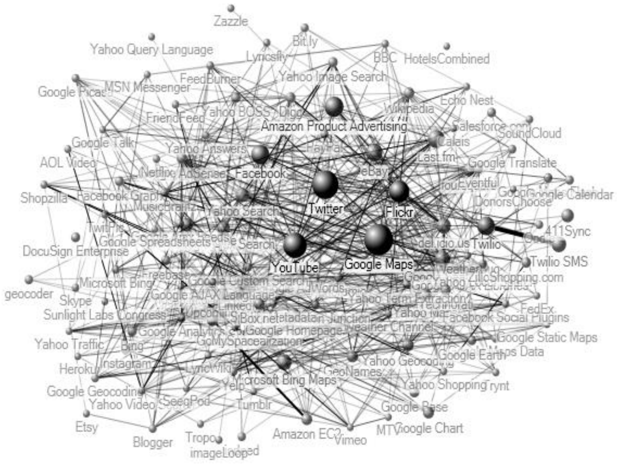


Figure 9 Overview of the service-level view

TABLE VI STRUCTURAL PROPERTIES OF THE SERVICE VIEW

| Property | Val<br>u<br>e |
|---|---|
| Vertices | 1,028 |
| Total Edges | 8,510 |
| Maximum Geodesic Distance (Diameter)* | 7.000 |
| Average Geodesic Distance | 2.610 |
| Graph Density* | 0.008 |
| Average Clustering Coefficient | 0.317 |
| Average Degree | 16.556 |

* Graph diameter is the length of the longest and the shortest hop path in the graph.
* Graph density is the number of edges (excluding self-links) presented in the graph divided by the maximal possible number of edges in the graph.

TABLE VII THE TOP-10 CATEGORY PAIRS WITH THE LARGEST CONNECTION WEIGHTS

| **API Category** | **API Category** | **Connection Weight** |
|---|---|---|
| Telephony | Messaging | 0.399 |
| Financial | Project Management | 0.270 |
| Financial | Office | 0.254 |
| Games | Widgets | 0.244 |
| Sports | Events | 0.199 |
| Entertainment | Recommendations | 0.191 |
| Reference | Education | 0.185 |
| Media Management | Tagging | 0.181 |
| Storage | Office | 0.162 |
| Advertising | Email | 0.159 |

### *5.2. Analysis on the domain-level view*

This section analyzes the Web service network from the domain-level view, i.e., the top-level view. The domain-level graph is an abstraction of the Web service network by treating each service domain or category as a node. With the domain-level graph, we can obtain the connection characteristics of various service domains.

According to the definition of the domain-level graph proposed in section III, we use Eq. (2) to calculate the connection weights between service domains. Table VII shows the top-10 largest connection weights between Web service categories (as specified in Table II). We can see, the largest connection weight occurs between the service categories *Telephony* and *Messaging*, which indicates that, services in the *Telephony* and *Messaging* domains are often composed together to create Mashups.

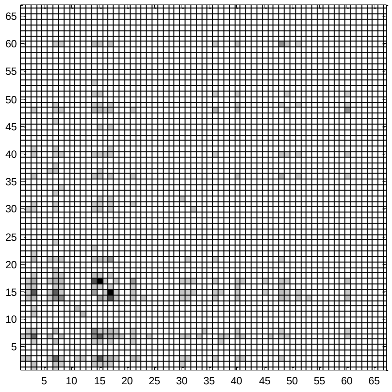


Figure 10 Visualization of the connection weights between all service domains (i.e., categories)

Figure 10 visualizes the connection weights among all service domains of Web services at *ProgrammableWeb.com*. The horizontal and vertical axes represent the ID of service domains which were specified in Table II. Again, let darker cells indicate greater connection weights. We can see, the cell (15, 17) or (17, 15) have the darkest shading, which indicates that there is strong connection between *Social* and *Mapping*. Likewise, the cell (3, 7) or (7, 3) is also relatively dark, indicating that there exists a strong connection between *Photos* and *Video.*

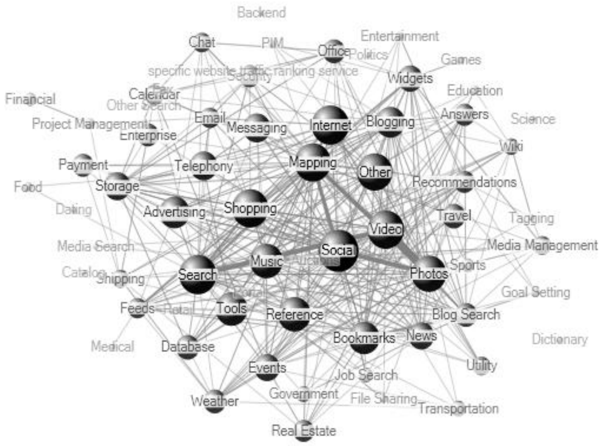


Figure 11 Overview of the domain-level view

TABLE VIII STRUCTURAL PROPERTIES OF THE DOMAIN VIEW

| Property | Value |
|---|---:|
| Vertices | 67.000 |
| Total Edges | 1,692.000 |
| Maximum Geodesic Distance (Diameter) | 3.000 |
| Average Geodesic Distance | 1.469 |
| Graph Density | 0.383 |
| Average Clustering Coefficient | 0.698 |
| Average Degree | 25.254 |

The structure of the domain-level graph is shown in Figure 11, where larger node indicates service domains with more connections. Again, to filter unnecessarily weak connections, we set a threshold 0.01 for the connection weight, the edges whose weights are smaller than 0.01 are not taken into account in the domain-level graph. As can be seen in Figure 11, there are mainly 10 dominant service domains in the Web service ecosystem, such as *Tools*, *Internet*, *Social*, *Financial*, *Enterprise*, *Reference*, *Mapping*, *Shopping*, *Science*, and *Government*, and there are significant connections between *Social* and *Mapping*, *Music*, *Video*, *Photos* etc.

The basic structural properties of the domain-level graph are presented in Table VIII. As expected, the domain-level graph has a high connection density, and thus has a very small diameter and average distance. It also has a high clustering coefficient.

TABLE IX EXAMPLES OF THE IMPORTANCE WEIGHTS OF TAGS

| API | $T_1$ | $T_2$ | $T_3$ | Domain |
|---|---|---|---|---|
| Google Calendar | events<br>0.145 | calendar<br>0.855 | -<br>- | Calendar |
| Facebook Credits | payment<br>0.589 | games<br>0.205 | virtual<br>0.206 | Payment |

| | | | | |
|---|---|---|---|---|
| 360voice | games<br>0.493 | xbox<br>0.254 | social<br>0.253 | Games |

TABLE X THE TOP-10 TAG PAIRS WITH THE LARGEST CONNECTION WEIGHTS

| **Tag** | **Tag** | **Connection Weight** |
|---|---|---|
| social | photo | 0.772 |
| mapping | viewer | 0.587 |
| telephony | sms | 0.546 |
| mapping | display | 0.540 |
| mapping | places | 0.512 |
| sms | TTS | 0.508 |
| social | microblogging | 0.500 |
| deadpool | search | 0.497 |
| search | mapping | 0.437 |
| shopping | auction | 0.436 |

### *5.3. Analysis on the tag-level view*

According to the definition of the tag-level graph presented in section III, we use Eq. (3) to calculate the connection weights between service tags. We set $f_1$, $f_2$, $f_3$ in Eq. (3) to 0.450, 0.350, and 0.200 respectively. Thus the importance weights of all tags for each service are determined. For example, we computed the importance weights of tags for *WAPIs Google Calendar*, *Facebook Credits* and *360voice*, which are shown in Table IX. The importance weights of the tags *events* and *calendar* to *Google Calendar* are 0.145 and 0.855 respectively.

With the importance weights of tags, we calculate the connection weights between tags with Formula (4). Table X shows the top-10 largest connection weights between tags. We can see, the largest connection weight occurs between the tags *social* and *photo*, which indicates that, services concerning *social* and *photo* are often composed together to create Mashups.

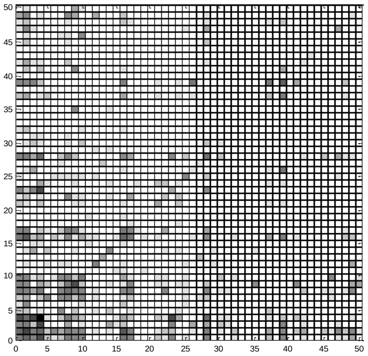


Figure 12 Visualization of the connection weights between the top-50 most popular tags

Figure 12 uses a grid to visualize the connection weights between the top-50 most popular tags, where the horizontal and vertical axes represent the ID of tags. Similarly, we let darker cells indicate larger connection weights. As can be seen from Figure 12, the cell (4, 4) has the deepest dark shading, indicating that services with the tag *Internet* are frequently composed.

TABLE XI THE PROPERTIES OF THE TAG-LEVEL VIEW

| **Property** | **Value** |
|---|---|
| Vertices | 525.000 |
| Total Edges | 25456.000 |
| Maximum Geodesic Distance (Diameter) | 4.000 |
| Average Geodesic Distance | 2.043 |
| Graph Density | 0.093 |
| Average Clustering Coefficient | 0.799 |
| Average Degree | 48.488 |

The structure of the tag-level graph is shown in Figure 13, where larger node indicates service tags with more connections. Likewise, to filter unnecessarily weak connections, we set a threshold 0.01 for the connection weight, the edges whose weights are smaller than 0.01 are not taken into account in the tag-level graph. The basic structural properties of the tag-level network are shown in Table XI.

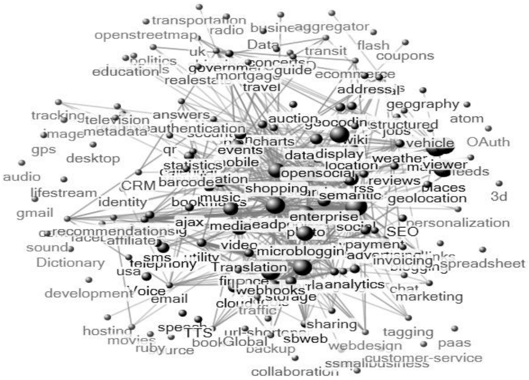


Figure 13 Overview of the tag-level view

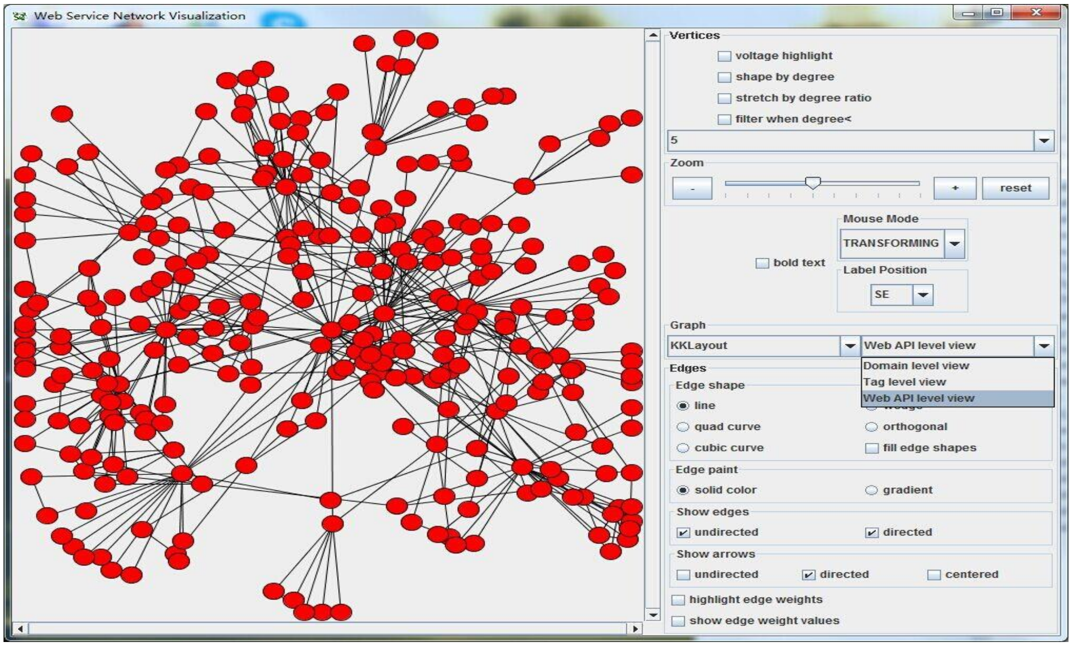
## 6 Visual Analysis Tool for Web Service Networks

In order to implement the above model for exploring Web service networks, we developed a visual analysis tool, as shown in Figure 14. It has the following three major functionalities.

Firstly, it can be employed to construct Web service networks. In this work, we use the data crawled from *ProgrammableWeb.com* as input to construct the Web service networks. However, our tool is developed for general purpose and thus can also use other Web services datasets to construct Web service networks.

Secondly, it can be employed to visualize the Web service networks from different-level views and in different styles. The tool fully implemented the three-level view model described in this paper. By default, the service-level view of the Web service network is exhibited. When the user chooses the domain-level view, the various Web service domains and the connections between them will be exhibited. Otherwise, when the user chooses the tag-level view, all service tags and the connections between them will be exhibited. The user can filter nodes or links with low connection degrees or weights to simplify the views. The user can employ the tool find the services, tags or service categories that have the highest connection weights. The user can also zoom in or zoom out the views to make them clearer. Moreover, the views can be displayed in different layouts in styles.

Thirdly, it can be employed to analyse and report the structural properties of the Web service network at different levels, such as these listed in Table VI, VIII and XI. It can also be employed to mining other kinds of relation between Web services such as the similarity relation and the potential collaboration relation.


Web Service Network Visualization

Vertices
□ voltage highlight
□ shape by degree
□ stretch by degree ratio
□ filter when degree<
5

Zoom
- + reset

□ bold text
Mouse Mode: TRANSFORMING
Label Position: SE

Graph
KKLayout
Web API level view
Domain level view
Tag level view
Web API level view

Edges
Edge shape
● line
○ quad curve
○ cubic curve
○ orthogonal
□ fill edge shapes
Edge paint
● solid color
○ gradient
Show edges
☑ undirected
☑ directed
Show arrows
□ undirected
☑ directed
□ centered
□ highlight edge weights
□ show edge weight values

Figure 14 A visual analysis tool based on the proposed three-level view model

## 7 Conclusion

In this paper, we proposed a three-level view model for exploring Web service networks which is built based on the collaboration relation between Web services. The model is composed of the service-level view, the tag-level view and the service domain-level view, which are intended to capture different characteristics of the Web service network. By employing the real Web service data collected from *ProgrammableWeb.com*, we provided an implementation of the proposed three-level view model, and analyzed the characteristics of the Web service ecosystem represented by *ProgrammableWeb.com*. Finally, a visual analysis tool for exploring and analysing Web service networks is presented.

Since the reuse rate of Web services is still very low, a deep investigation of Web service ecosystem is certainly helpful for promoting the understanding and reuse of Web services. The proposed model provides a sound way for the investigation of Web service ecosystem. It can help users find interesting service compositions or composition patterns more efficiently and easily. In the future work, we will explore more effective techniques for mining the relation between services, tags and service categories, and will conduct more experiments to evaluate the proposed three-level view model. In addition, we will consider improving the visual analysis tool by enriching its functionalities.

### Acknowledgement

The work described in this paper was supported in part by the National Natural Science Foundation of China under grant No. 61402168, No. 61572186 and No. 61572187, and in part by the Scientific Research Fund of Hunan Provincial Education Department of China under Grant 15K043.

### References

1. X. Liu, Y. Hui, W. Sun, and H. Liang, Towards service composition based on mashup, IEEE Congress on Services, pp.332-339, 2007.
2. A. P. Barros and M. Dumas, The rise of web service ecosystems, IT professional, Vol. 8, pp. 31-37, 2006.
3. K. Huang, Y. Fan, W. Tan, and X. Li, Service Recommendation in an Evolving Ecosystem: A Link Prediction Approach, Proceedings of the 2013 IEEE 20th International Conference on Web Services, June 28-July 3, 2013, Santa Clara, CA.
4. S. Yu and C. J. Woodard, Innovation in the programmable web: Characterizing the mashup ecosystem, Proceedings of the Service-Oriented Computing–ICSOC 2008 Workshops, 2009, Sydney, Australia.
5. K. Huang, Y. Fan, and W. Tan, An Empirical Study of Programmable Web: A Network Analysis on a Service-Mashup System, Proceedings of the 2012 IEEE 19th International Conference on Web Services, June 24-29, 2012, Honolulu, HI.
6. C. Cherifi and J.-F. Santucci, Analyzing Web Services Networks: A WS-NEXT Application, Ubiquitous Computing and Communication Journal, pp.60-77, 2011.
7. S. Wang, W. Su, X. Zhu, H. Zhang, A Hadoop-based approach for efficient web service management, Int. J. Web and Grid Services, Vol. 9, No. 1, pp.18-34, 2013.
8. X. Dong, A. Halevy, J. Madhavan, E. Nemes, J. Zhang. Similarity search for web services, Proceedings of the Thirtieth international conference on Very large data bases, 2004.
9. F. Liu,Y. Shi, J. Yu, T. Wang, J. Wu. Measuring Similarity of Web Services Based on WSDL. Proceedings of the 2010 IEEE 17th International Conference on Web Services, July 5-10, 2010, Miami, Florida.
10. P. Plebani, B. Pernici. URBE: Web Service Retrieval Based on Similarity Evaluation. IEEE Transactions on Knowledge and Data Engineering, Vol. 21 No.11, pp.1629-1642, 2009.
11. K.Elgazzar, A.E. Hassan, P. Martin. Clustering WSDL Documents to Bootstrap the Discovery of Web Services. Proceedings of the 2010 IEEE 17th International Conference on Web Services, July 5-10, 2010, Miami, Florida.
12. R. Aydogan and H. Zirtiloglu, A Graph-Based Web Service Composition Technique Using Ontological Information, Proceedings of the 2007 IEEE 14th International Conference on Web Services, July 9-13, 2007, Salt Lake City, UT.
13. X. Liu, G. Huang, H. Mei. Discovering Homogeneous Web Service Community in the User-Centric Web Environment. IEEE Transactions on Services Computing, Vol. 2 No.2, pp.167-181, 2009.
14. H. Kil, S.C. Oh, E. Elmacioglu, W. Nam, D. Lee: Graph Theoretic Topological Analysis of Web Service Networks, WWW, Vol. 12, No. 3, pp. 321-343, 2009.
15. C. Cherifi, V. Labatut, and J. F. Santucci, Web Services Dependency Networks Analysis, International Conference of New Media and Interactivity (NMI 2010), pp.115-12, 2010.

16. Z. Feng, B. Lan, Z. Zhang and S. Chen, A Study of Semantic Web Services Network, The Computer Journal, Vol. 58 No. 6, pp. 1293-1305, 2014.
17. L. Chen, Y. Wang, Q. Yu, Z. Zheng, and J. Wu, WT-LDA: User Tagging Augmented LDA for Web Service Clustering, Proceedings of the Service-Oriented Computing, December 2-5, 2013, Berlin, Germany.
18. J. Hwang, J. Altmann, and K. Kim, The structural evolution of the Web 2.0 service network, Online Information Review, Vol. 33 No.6, pp.1040-1057, 2009.
19. G. Wang，J. Liu, B. Cao, M. Tang. Mashup Service Classification and Recommendation based on Similarity Computing, Proceedings of the 2nd International Conference on Social Computing and Its Applications, November 1-3, 2012, Xiangtan , China.
20. B. Cao, J. Liu, M. Tang, Z. Zheng, Guangrong Wang. Mashup Service Recommendation based on User Interest and Social Network, Proceedings of the 2013 IEEE 20th International Conference on Web Services, June 28-July 3, 2013, Santa Clara, CA.