JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 17-36 (2017)
DOI: 10.6688/JISE.2017.33.1.2

Retrieval of Web Service Components using UML Modeling and Term Expansion*

WEN-TIN LEE[1], SHANG-PIN MA[2] AND YAO-YU TSAI[2]
[1]*Department of Software Engineering and Management*
National Kaohsiung Normal University
Kaohsiung, 824 Taiwan
[2]*Department of Computer Science and Engineering*
National Taiwan Ocean University
Keelung, 202 Taiwan
E-mail: wtlee@nknu.edu.tw; albert@ntou.edu.tw; finalcai1215@gmail.com

Web service discovery plays a crucial role in the development of applications based on service-oriented architecture, due to the importance of identifying services capable of fulfilling the requirements of service requesters. In the field of software engineering, web service discovery can be applied to the problem of software component retrieval in order to facilitate the reuse of software components and speed up the development of new software projects. Unfortunately, most existing service discovery schemes are unable to perform this task effectively due to manual service queries, a lack of support for service queries using unified modeling language (UML), and the high degree of complexity associated with integration. This study proposes a novel approach to resolving this situation referred to as SCRUMTE (Service Component Retrieval with UML Modeling and Term Expansion). The proposed system provides three functions: 1) the generation of a lexically expanded term set for each candidate service in the service repository; 2) transformation of required UML models into WSDL (Web Services Description Language) documents for use in service queries; and 3) integration of a web service retrieval mechanism for the identification of services based on textual similarities and service integration. This system was designed to assist software developers in the search for service components according to their specific requirements. Experiment results demonstrate that the search precision of the SCRUMTE system far exceeds that of text-based approaches.

Keywords: service-oriented architecture, service discovery, UML, term expansion, term expasion

1. INTRODUCTION

Service-oriented computing (SOC) is an important trend in the field of software engineering [1]. In SOC, web services and service-oriented architectures (SOAs) act as fundamental elements in the provision of on-demand applications. Web services are self-described, self-contained, and platform-independent computational elements, which can be composed, published, and located using standard protocols for the construction of applications operating on a range of platforms. Among these technologies, web service discovery is particularly important in the development of SOA-based applications. An effective service discovery mechanism can help users to locate services capable of satisfying specific requirements [2], playing an important role in situations where service

Received December 8, 2015; revised March 6 & March 24, 2016; accepted April 5, 2016.
Communicated by Chang-Shing Lee.
* This research was sponsored by Ministry of Science and Technology in Taiwan under grants MOST 104-2221-E-017-014 and MOST 104-2221-E-019-001.

composition technology is needed to assist in the locating and binding of component services [3, 4]. Over the past decade, numerous researchers have addressed the issue of service discovery using information retrieval mechanisms [5], applying techniques of the semantic web [6, 7], or utilizing hybrid solutions [8]. Within the current paradigm, service requesters must issue queries that include all criteria used in the search of web services, such as service capability or quality of service (QoS). It should be noted that the methods used in service discovery are similar to those applied by software engineers to deal with the problem of software component retrieval [9]. Documents in WSDL [10] (Web Services Description Language) can be generated automatically or semi-automatically to describe the functional specifications of software components in the asset library of an organization or in a public API (application programming interface) repository. Thus, we posit that the mechanisms used in service discovery could also be applied to the retrieval of software components. Effective mechanisms for the retrieval of service components could facilitate and expand the re-use of software components and thereby speed up the development of new software projects.

Unfortunately, most existing service discovery schemes are unable to perform these tasks effectively due to the following problems:

- **Manual service query process** The service components currently available are able to deal with only a portion of the enormous number of functions designed into a system. Unfortunately, software developers are unaware of the number of service queries that should or could be made. Issuing a large number of service queries manually increases the likelihood of errors and consumes a great deal of time.
- **Lack of support for service queries using Unified Modeling Language (UML)**[1] Software developers today are expected to have a working knowledge of UML for the analysis and design of software systems. In fact, most of the content of service queries is embedded in developed UML models. However, as mentioned, software developers must perform searches manually to find reusable service components, even for systems that have undergone thorough analysis and design. Automatic or semi-automatic ways to generate service queries from UML models is required to avoid unnecessary manual efforts.
- **High integration complexity** Conventional service discovery mechanisms tend to spread service operations throughout multiple service components, and using a large number of service components in a software project greatly increases the complexity of integration.

In this study, we addressed these issues by identifying three core requirements in the construction of an effective method for the discovery of service components: (1) We eliminated the need for manual interventions in the discovery of service components; (2) We employed existing UML models in the discovery of service components; (3) We sought to minimize the number of service components retrieved. The resulting system is referred to as the Service Component Retrieval with UML-based Modeling and Term Expansion (SCRUMTE), which prov ides two key features: (1) transformation of UML models into WSDL documents for service queries; and (2) a novel mechanism to facilitate the discovery of web service components by minimizing the number of components while satisfying the requirements of users through the application of lexical similarity. Our

[1] http://www.uml.org/

objective was to decrease the cost of software development by enabling the retrieval of reusable service components. Fig. 1 presents an overview of the SCRUMTE system, which performs two fundamental types of activity: system activities implemented by the SCRUMTE engine, and user activities conducted by the developers of software systems. System activities include the parsing of WSDL documents, WordNet-based term expansion, the transformation from UML to WSDL, and service ranking based on the Vector Space Model (VSM). User activities include the development of UML models and the selection of services from a list of services generated for the user.

The remainder of this paper is organized as follows: Section 2 presents a review of existing research in this field. In Section 3 we detail the proposed approach. Empirical evaluations as to the feasibility of the proposed system are presented in Section 4. Conclusions are drawn in Section 5.

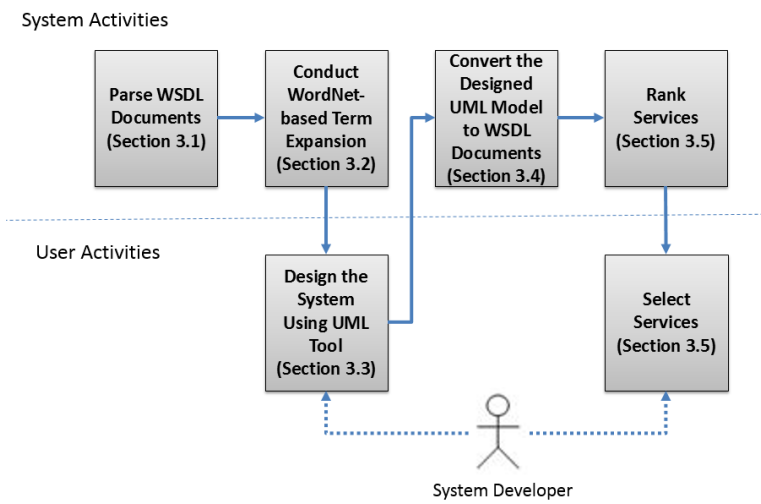

Fig. 1. Overview of proposed SCRUMTE system.

2. RELATED WORKS

Semantic Similarity Retrieval Models (SSRM) [11] is a mechanism of query expansion based on WordNet [12], comprising three phases: term re-weighting, term expansion, and determining the degree of similarity among documents. The re-weighting phase involves the adjustment of weights for each query term, in accordance with its lexical similarity to other terms in the same query. For instance, in a query containing the terms "news", "theater", and "cinema", the last two terms have a lexical relationship. As a result, these terms are assigned higher weights, whereas the weight of term "news" remains unchanged. In the term expansion phase, queries are expanded using synonymous terms, and terms are augmented using hyponyms and hypernyms with weights assigned according to the distance between the original term and the augmented term. Only the terms (hypernyms/hyponyms) that possess a degree of similarity higher than a given threshold T are added to the query. In the assessment of document similarity phase,

SSRM is used to quantify the degree of similarity between expanded queries and the document in question. The weight assigned to each term in the document is then calculated using term frequency-inverse document frequency (TF-IDF). In this research, we enhance SSRM to do term expansion for service descriptions.

Levi and Arsanjani [13] proposed an approach based in a goal-model that included criteria for the decomposition of systems into modules. High-level business goals are decomposed into sub-goals with explicit dependencies, and a set of services is required to achieve each of the sub-goals. This method makes it possible to map the architecture of a business into the architecture of component-based software (CBSA). In this approach, the emphasis is on identifying high-level enterprise components rather than retrieving existing software components.

Service-Oriented Modeling and Architecture (SOMA) [14] involves the establishment of a software development life cycle aimed at the analysis, design, implementation, and deployment of projects based on a service-oriented architecture (SOA). SOMA is a fractal method based on two principles: (1) process execution in self-similar scope and (2) iterative and incremental processes. The first principle makes it possible to adapt the application of SOMA to projects of any size, without the need to alter the means of implementation. The second principle stipulates that software development be conducted in an iterative and incremental manner similar to unified process. SOMA is a high-level life cycle model rather than an operational methodology.

Bauer and Müller [15] employed the Model-Driven Architecture (MDA) approach for web service choreography and demonstrate a mapping of platform independent models based on UML sequence diagrams to a platform-dependent model based on the Business Process Execution Language for Web Services (BPEL4WS). Although this approach also utilizes UML sequence diagrams, SCRUMTE further adapts the sequence diagram to discover service components.

Skogan *et al*. [16] utilized standard UML constructs with a minimal set of extensions for web services. This approach involves the use of a standard class diagram and an activity diagram to model the composition of services, whereupon web service discovery is performed by searching through the WSDL of web services. The user is able to specify preferences for the language used in composition and for the execution engine used for composite web services. This approach supports two executable composition languages, BPEL4WS and WorkSCo (Workflow with Separation of Concerns). The proposed SCRUMTE system is similar to this approach; however, the developer is not required to provide a model of the services. SCRUMTE makes it possible for developers to retrieve candidate reusable service components, as long as they are able to conduct the analysis and design of software systems using UML.

Wu and Ibrahim [17] used a UML sequence diagram to model the workflow of web service composition by describing the interactions between various web services and stipulating the sequence of service execution. Sequence diagrams are parsed to enable the extraction of information needed for composition, including the name of the web service, the name of the operation, inputs, outputs, preconditions and post-conditions. A tree-based optimization algorithm is used to formulate an optimal solution for the composition based on multiple QoS attributes and the sequence of tasks in the UML sequence dia-gram. This approach focuses on QoS-driven service discovery whereas the aim of SCRUMTE is to find service components with interfaces that are not perfectly compati-

ble with the service queries extracted from UML models.

Spanoudakis and Zisman [18] proposed a UML-based framework to assist with the development of service-based systems. The framework makes uses of a query language to specify the structural, behavioral, and quality properties that services should satisfy, and provide a query processor to match the queries against service registries. The framework assumes services are specified by WSDL to describe their interfaces, BPEL4WS to represent their behaviors, and an XML document to assert their quality. The query processor simultaneously considers signature distances, behavioral distances, and soft constraint distances. This method needs the user to annotate a lot of information in UML models and to provide additional query documents. Compared to this approach, SCRUMTE provides a simpler way to let users develop UML models directly and generate possible service queries automatically.

Zille *et al*. [19] developed a matching mechanism using UML-based rich service description language (RSDL) to enable the automatic discovery of services. RSDL comprises a description of operation signatures in WSDL, a semantic description of each operation using UML-based visual contracts (VC), and specific service protocols in which UML sequence diagrams are used for service requests and UML state charts are used for service offers. SCRUMTE is similar to this approach; however, SCRUMTE eliminates the need to patch semantic annotations on WSDLs while retaining the ability to locate required services using the proposed term expansion mechanism.

3. PROPOSED APPROACH: SERVICE COMPONENT RETRIEVAL WITH UML-BASED MODELING AND TERM EXPANSION

In this section, we begin by outlining the proposed methods of parsing WSDL and performing term expansion. Next, we introduce the use of two UML models, use case diagrams, and sequence diagrams in the analysis and design of a software system. The UML model is then converted into a WSDL document in the form of a service query based on the proposed mapping rules. Finally, the mechanisms proposed for the discovery of web services based on lexical similarity and the minimization of components are applied in the retrieval of service components.

We also provide an example implementation involving the development of a movie information system (MIS). The aim is to illustrate how the SCRUMTE system could help developers to identify service components in an organizational asset library or a public component repository that may be applicable in the development of the software system.

3.1 Parsing of WSDL Documents

Before any other steps can be taken, the WSDL terms used for web services must be preprocessed in two steps, as follows:

- Splitting combined terms: The names of operations and input/output parameters are usually specified by naming rules, such as those in Pascal, Hungary, and Camel. The splitting of combined terms enables the retrieval of separate terms in accordance with naming conventions and removes useless words (such as “a”, “this”, and “type”).

- Word normalization and stemming: Normalization is used to transform a morphological word into its original form to ensure the inclusion of words that are suitable to the process of WordNet-based expansion. Stemming is the process of reducing inflected (or sometimes derived) words to their root form. In this study, we employed Porter stemming [20] to eliminate differences among inflectional morphemes.

Following the completion of these two steps, the extracted terms are categorized in an XML-encoded document as <*operation*>, <*input*>, and <*output*>. For instance, the input parameter "*movieType*" is transformed into lower case and divided into the two terms "*movie*" and "*type*". "*Type*" is a designated stop word; therefore, SCRUMTE adds only the term "*movie*" to the <*input*>.

3.2 SSRM-Enhanced Term Expansion

To mitigate linguistic imprecision between a service description and a user query, we planned to do term expansion for service descriptions. We sought to leverage the hierarchical structure of the general thesaurus WordNet in the identification and organization of as many matching terms as possible. WordNet is a large lexical and domain-independent database of English and was applied in a lot of service discovery methodologies [21, 22]. Currently, we make use of WordNet solely to do domain-independent term expansion. Ontologies in various domains [23] could be integrated into SCRUMTE easily using the same method described below.

The proposed term expansion method is based on SSRM and WordNet to find out expanded terms according to the degrees of similarity among terms [24]. Many methods have been proposed for the measurement of similarity between two terms. In this study, we adopted the edge counting method proposed by Li *et al*. [25], which is strongly correlated to human judgements related to similarity and has been widely applied in the past decade. Eq. (1) is used to compute the degree of similarity according to the length and depth of the path, as follows:

$$sim(w_1, w_2) = f(l) * f(h) = e^{-\alpha 1} * \frac{e^{-\beta h} - e^{\beta h}}{e^{-\beta h} + e^{\beta h}} \rightarrow [0..1], \tag{1}$$

where l is the shortest path length between w_1 and w_2, and h is the depth of the similar word, which is derived by counting the number of levels between the similar word and the top of the hierarchical structure. In this research, we followed the suggestions provided in [7] in setting the values of α and β at 0.2 and 0.6 respectively. For example, the l and h between price and value are *2* and *5*. The *sim(price,value)* is *0.8175*.

The proposed lexical expansion mechanism was inspired by SSRM, which uses only nouns and noun phrases from WordNet in the expansion of query terms. However, nouns are insufficient to provide a complete description of service capability. Service operations are usually labelled using verbs as well as nouns, such as "*cancelOrder*" and "*deleteOrder*". When applying SSRM, it may appear that these two services possess similar capabilities, because the verbs "*cancel*" and "*delete*" cannot be included in the expanded query. Thus, the proposed method also extracts verbs from the name of operations to facilitate the calculation of similarity. Only nouns are extracted from the

input and output elements because the elements used in the description of web services are usually pure data. The proposed lexical expansion method also differs from SSRM by not imposing a limit on the number of senses taken into account during expansion.

When a WSDL for a published service is registered in the service repository, term expansion is used to extract term tokens in WSDL. Each term token is then processed using WordNet to obtain the following expanded terms: synonyms, hypernyms (words that are more generic or more abstract than a given word), and hyponyms (a word that is more specific or less abstract than a given word). The resulting lexical relationship with query terms and similarity scores must exceed the threshold specified in Eq. (1). In this study, the threshold for the extraction of highly relevant terms was set at 0.8. The principle of lexical expansion is presented in Fig. 2.

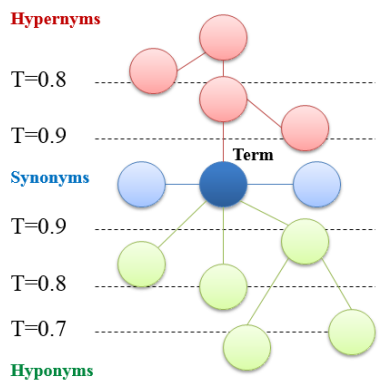

Fig. 2. Synonyms, Hypernyms, and Hyponyms.

During lexical expansion, every term is assigned a weight calculated as follows:

$$q_j = q_i \times sim^{lex}(i, j) \tag{2}$$

where $sim^{lex}(i, j)$ is the same as in Eq. (1) and q_i refers to the weight initially applied to the term in the TF-IDF calculation. Note that i represents the term belonging to the original query and j stands for a synonym, a hypernym, or a hyponym, which is expanded by i in accordance with the proposed query expansion process. For example, the word "price" is expanded to include the word "*value*". If q_{price} were 1, then the q_{value} would be 0.8175, indicating the similarity between "*price*" and "*value*".

To illustrate the differences between the original SSRM and our expansion approach, another example, airline reservation, was prepared. In the airline reservation example, we have two service queries and one service (considering the service operation name only for simplification):

Service 1: *bookAirlineTicket*
Query 1: *reserve airway ticket*
Query 2: *cancel airway ticket*

By conducting the term expansion mechanism of SCRUMTE and the original SSRM, the expanded tokens for the service name are as follows. It is noted that SCRUMTE expanded both nouns/noun phrases and verbs whereas SSRM expanded only nouns/noun phrases.

- SCRUMTE: {(*book*, 1), (*reserve*, 0.946), (*request*, 0.833), (*bespeak*, 0.815), (*airline*, 1), (*airway*, 0.901), (*ticket*, 1)}
- SSRM: {(*book*, 1), (*airline*, 1), (*airway*, 0.901), (*ticket*, 1)}

The tokens for Query 1 and Query 2 are shown as follows:

- Query 1: {*reserve*, *airway*, *ticket*}
- Query 2: {*cancel*, *airway*, *ticket*}

Next, we used these two expanded token sets to calculate the similarity between Service 1 and Query 1 as well as Service 1 and Query 2 based on Eq. (3), which is described in Section 3.5. Note that we did not consider TF-IDF here for the sake of simplicity. The similarity calculation results are shown in Table 1:

Table 1. Similarity calculation results.

The similarity between Service 1 and Query 1	SCRUMTE	0.999
	SSRM	0.815
The similarity between Service 1 and Query 2	SCRUMTE	0.815
	SSRM	0.815

It is obvious that these two queries sought for opposite services and Service 1 could only satisfy Query 1. SCRUMTE-based similarities were able to provide more appropriate information whereas SSRM-based could cause incorrect service retrieval.

3.3 System Design Using UML Tools

In this stage, developers can use UML design tools, such as StarUML[2], ArgoUML[3], MagicDraw[4], or Visual Paradigm[5], to develop use case diagrams and sequence diagrams for the modeling of the software system. In this study, we opted for StarUML to provide examples. Developers tend to draw use case diagrams to analyze the main business functions of the system and sequence diagrams to model the interactions among the user interface, system controller, and core modules. Use case diagrams and sequence diagrams can serve as service queries in the search for appropriate service components. It should be noted that XMI (XML Metadata Interchange)[6] is commonly used as an interchange format for UML models and XMI documents can be exported automatically using UML design tools. We therefore assumed that most developers would follow the above process in the design of the system, using UML to generate a corresponding XMI document. The document would be a record of all UML models, including use case

[2] http://staruml.io/
[3] http://argouml.tigris.org/
[4] http://www.nomagic.com/products/magicdraw.html
[5] http://www.visual-paradigm.com/
[6] http://www.omg.org/spec/XMI/

diagrams and sequence diagrams. In the next sub-sections, we describe the guidelines used in the application of the proposed system approach.

3.3.1 Development of use case diagrams

Use case diagrams are generally employed to capture the usage requirements of a system. In these diagrams, each use case represents a business function, such as searching for a movie, reserving a hotel room, or planning a route. Fig. 3 presents an example of a typical use case. In this study, we assumed that the user would create a sequence diagram corresponding to each use case. For instance, a developer designing a movie information system may decide that the system requires a search function for movies. Thus, a use case referred to as "*Search Movie Info*" would be added to the use case diagram and a corresponding sequence diagram would be created to describe interactions between key participants in realizing the functionality of movie search.

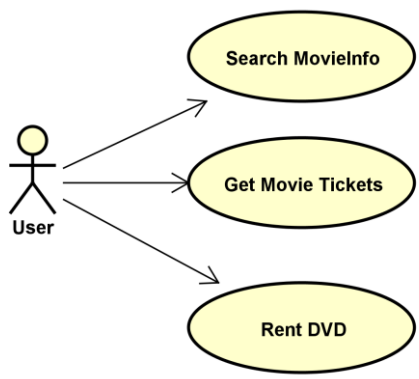

Fig. 3. Example of use case diagram.

3.3.2 Development of sequence diagrams

The most important element modelled in a sequence diagram is the role of the classifier, *i.e*., the participants in any collaborative actions. In this study, we assumed that the developer would use three proposed stereotypes to facilitate service discovery: systemView, systemController, and systemModel. These stereotypes are based on the MVC (Model-View-Controller) pattern, a software architecture in which the representtation of information is distinct from the way it interacts with users. Fig. 4 presents an example of a typical sequence diagram. The semantics used in the three stereotypes are described as follows:

- systemView: a user interface, such as the HTML pages or GUIs found in conventional stand-alone applications, web applications, or mobile appli- cations on the client side.
- systemController: a system flow controller, such as PHP script, Servlet components, or node.js pages, which is used to coordinate the user interface at the front-end with functionalities and data at the back-end.
- systemModel: a core business logic or important source of data in the system, such as a back-end data provider or API.

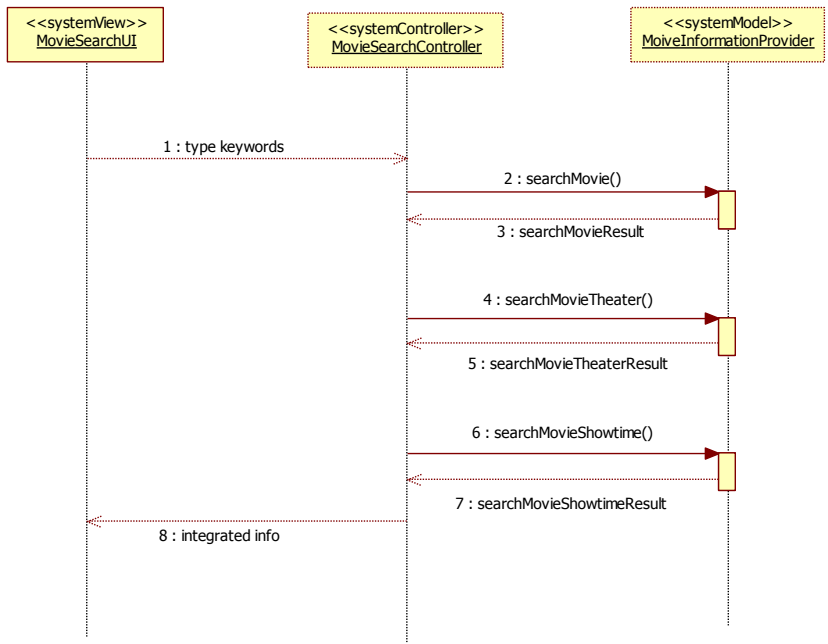

Fig. 4. Example of typical sequence diagram.

The message (*i.e.*, stimulus) represents the interaction between the roles of two classifiers. Because systemView is irrelevant to core functions and core data, we captured only two types of messages that are relevant to the retrieval of web services:

- From systemController to systemModel: This message is generated when the Controller calls the function of the Model. Note that the input parameters conveyed by the message should be also modeled. In StarUML, the input parameters can be specified as an "Argument" for the message.
- From systemModel to systemController: The message indicates that movement of output data when the Model returns results to the Controller.

Taking the movie information system as an example, the developer creates a sequence diagram for the use case "Search Movie Info" and then specifies three classifier roles: Movie Search UI, Movie Search Controller, and Movie Information Provider, by annotating three specific stereotypes. The messages "searchMovie" and "seachResult" are then inserted between the Controller and the Model, and the input parameter is specified as "movieType".

3.4 Conversion of Designed UML Model into WSDL Document

In this phase, the SCRUMTE system maps each sequence diagram into a WSDL document in accordance with the stipulated mapping rules.

Table 2. Structure of WSDL document.

Element	Description
<types>	A container for data type definitions used by the web service
<message>	A typed definition of the data being communicated
<portType>	A set of operations supported by one or more endpoints
<binding>	A protocol and data format specification for a particular port type

A formal WSDL document contains information pertaining to a web service, as shown in Table 2. The goal of SCRUMTE is the retrieval of service components based on the degree of similarity between the request and the functionality of the candidate service; therefore only information related to <*type*>, <*message*>, and <*operation*> of <*portType*> is captured. The <binding> information, which describes the implementation of information services, is not covered in this study. In addition, each <*portType*>, which includes multiple service operations, represents an individual service component.

In this study, we used WSDL as the service query language for two reasons: (1) WSDL can be used to capture and organize important service-related information as well as service queries; and (2) WSDL is a de facto specification accepted by the public, which means that WSDL-based service queries can be submitted to other service match-making systems, thereby maximizing interoperability.

Table 3. Mapping rules for conversion from XMI to WSDL.

WSDL		XMI
types		The types of I/O parameter included in the messages between systemController and systemModel
message		1. The parameters included in the message from the systemController to the systemModel. (input message) 2. The name of message from systemModel to systemController. (output message)
portType	operation name	The method name of the message from systemController to systemModel.
	input	Refer to the input message
	output	Refer to the output message

Table 3 lists the set of mapping rules devised for transforming the UML model (in the XMI format) into a WSDL document to serve as a service query. In other words, SCRUMTE automatically generates a WSDL document representing a single service query according to the mapping rules. In the example in Fig. 5, the input parameter "*movieType*" was converted to a "*type*" element and embedded in an input message in WSDL, whereas the message "*movieSearchResult*" was transformed into an output message. The names of operations in WSDL are extracted from the name of methods used as input messages in the UML sequence diagram.

[7] http://staruml.sourceforge.net/en/

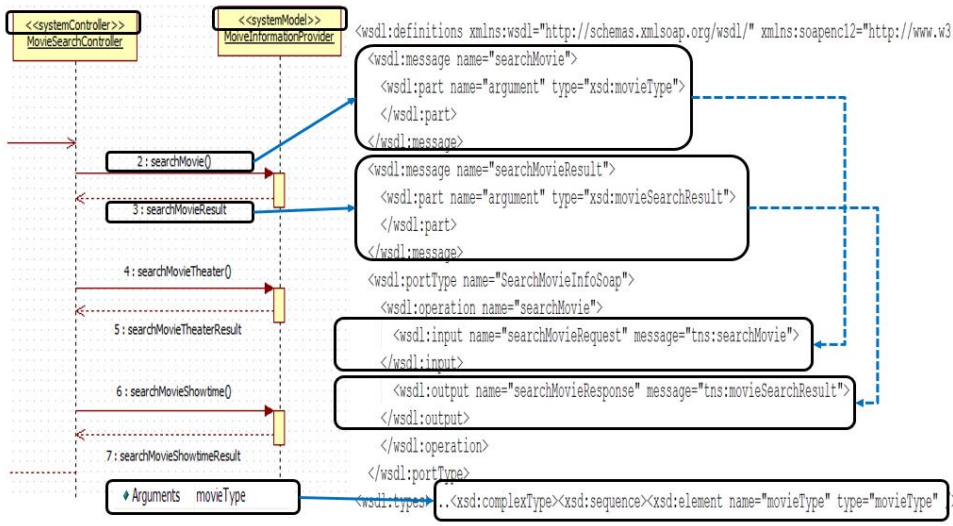

Fig. 5. Illustration of mapping from UML to WSDL.

3.5 Ranking and Selection of Services

The final step of the SCRUMTE system involves the ranking and selection of services. Two coring mechanisms were designed: VSM-based similarity computation (VSM: Vector Space Model) and the scheme for minimization NoSC (Number of Service Components).

The principal goal of SCRUMTE is the retrieval of services from the organizational service asset repository based on the WSDL-based service queries. SCRUMTE calculates the degree of similarity between the request and candidate services according to the cosine of the angle between the vector *Qt*, which represents the service query *SQ*, and the vector OPt_k, which represents the *k*th lexically-expanded web service operation ($SOP^{LX}{}_k$).

$$Sim(SQ, SOP^{LX}{}_k) = \frac{\overrightarrow{Qt} \bullet \overrightarrow{OPt_k}}{\left|\overrightarrow{Qt}\right| \left|\overrightarrow{OPt_k}\right|} \rightarrow [0..1], \tag{3}$$

where *Qt* comprises an operation part, an input part, and an output part, including terms derived from the WSDL-based query. OPt_k is similar to *Qt*.

The proposed similarity calculation method enables the SCRUMTE system to retrieve a set of services with the Top-K similarity scores for each request extracted from the UML model. Furthermore, each set of the services is ranked according to the similarity score before returning results to the users.

In addition to lexical similarity, we also sought to minimize the number of service components required to satisfy the requests in the UML model in order to reduce the complexity of integration in system development. Following the service discovery phase, the system developer obtains a set of the ranked services for each service query. Meanwhile, SCRUMTE elicits the Top-K services for each query and filters out services

with similarity below a given threshold. All service combinations of the Top-K retrieved services are then prepared for each query. Finally, SCRUMTE calculates the score of all service combinations according to the combination of components, as follows:

$$CS_i = \frac{\sum_{j=1}^{N} Sim(SQ, SOP^{LX}{}_j)}{N} \times cos\frac{\left(s_i^c - 1\right)\pi}{2N} \rightarrow [0..1], \tag{4}$$

where CS_i is the ith service combination. Sim(SQ, $SOP^{LX}{}_j$) is the same as Eq. (3) and N is the number of service queries. S_i^c is the number of components in a given combination.

For example, under the assumption that a combination includes three services (S_1, S_2, and S_3) with similarity scores of "0.866", "0.692", and "0.952" for the original queries generated from the UML model. Including these services in the same service component would produce a combination score of 0.836 (the average of the similarity scores). If these services belonged to two service components, then the combination score would be decreased to 0.724 (0.836× 0.866). If all of the services were provided by different components, then the combination score would be only 0.418 (0.836× 0.5). Clearly, the proposed method enables the prioritization of service combinations that include services from a smaller number of components.

SCRUMTE only recommends combinations with a component combination score exceeding the given threshold, which makes it possible for the developer to select all of the services in a recommended combination in order to minimize the number of service components.

After the selection of service operations using the proposed service discovery method or service component number minimization mechanism, the reuse rate is calculated for the service query embedded in the UML model. The reuse rate in Eq. (5) is calculated according to the proportion of selected services to service queries.

$$R = S_r \big/ N, \tag{5}$$

where R is the rate of service reuse, S_r is the number of the service operations selected by the developer, and N is the total number of service queries.

4. EXPERIMENTAL EVALUATIONS

In this section, we describe a system prototype using SCRUMTE and the experiments used for evaluation.

4.1 System Prototype

To verify the efficacy of the proposed system, we implemented a prototype referred to as the SCRUMTE engine, which was implemented using Java (for backend functionalities) as well as Java Servlet technology and client-side Web technology (HTML, CSS, and JavaScript). The SCRUMTE engine also utilizes a variety of external APIs for the realization of all required features. The adopted APIs include the following:

- JDOM[8]: This was used for the efficient parsing of WSDL documents for the extraction of necessary elements, such as operation name, input name with included elements, and output name with included elements.
- Terrier IR Platform[9]: Terrier APIs were used for term tokenization, the removal of stop words, and stemming (using PorterStemmer API of Terrier).
- JWI[10] (the MIT Java Wordnet Interface) and JWNL[11] (Java WordNet Library): These APIs were used to expand term tokens using WordNet. We also used APIs to search for synonyms, hypernyms, and hyponyms, to find the shortest path between two terms, and to determine the depth from the root to the target term.

Processing in the SCRUMTE engine includes four steps. The developer first uses a UML design tool for the modeling of a system and to export XMI documents. The developer uploads the XMI document to the system, which triggers the service discovery process. The system then converts the XMI document into WSDL documents. The application of mapping rules means that the resulting WSDL document contains only service interface information, such as service operation names, input messages, and output messages. Finally, the resulting WSDL documents are used as service queries for the retrieval of relevant services, the results of which can be browsed by the developer for the selection of services used in the development of the planned software system.

4.2 UML Models for Campus Information System

We used the aforementioned movie information system (MIS) and a campus information system (CIS) for university students as testbeds on which to verify the performance of the proposed system. We used the StarUML software to design UML models, including a use case diagram and multiple sequence diagrams, which were then exported as XMI documents. CIS includes two categories of functionality: browsing living information and browsing course information. For the living information, CIS can let users search for the phone numbers and location information of nearby shops, restaurants, hospitals, or clinics. For the course information, CIS can provide the course timetable, course materials and grades, as well as the location of classrooms. The resulting use case diagram of CIS is presented in Fig. 6, and a representative sequence diagrams of CIS for the "Browse course information" use case is presented in Fig. 7.

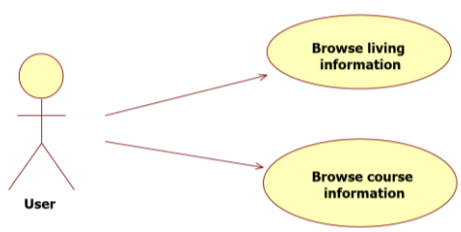

Fig. 6. Diagram of use case for campus information system.

[8] http://www.jdom.org/
[9] http://terrier.org
[10] http://projects.csail.mit.edu/jwi/
[11] http://sourceforge.net/projects/jwordnet/

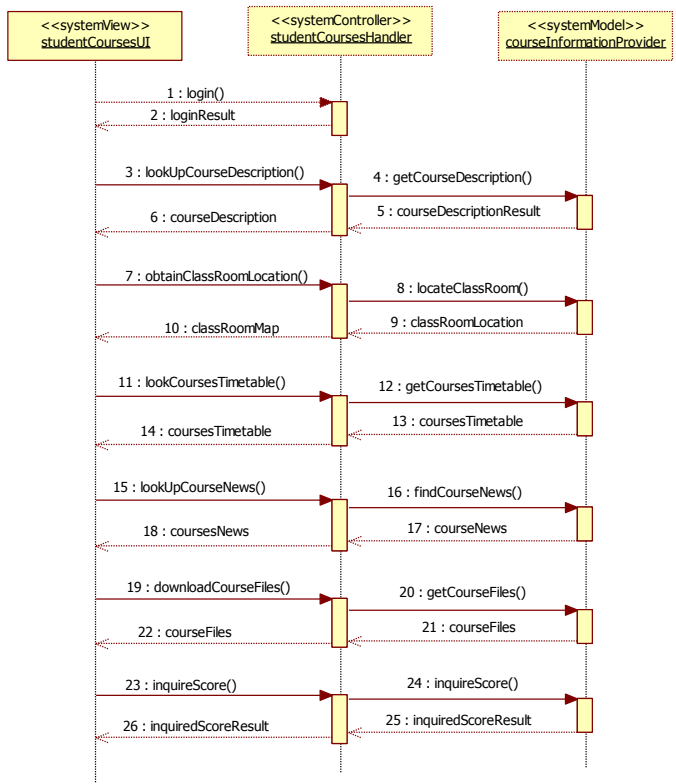

Fig. 7. Sequence diagram for "Browse course information" use case.

4.3 Experiment Setup and Analysis

To verify the feasibility and effectiveness of the proposed system, we created a control system comprising a traditional IR-based service retrieval system utilizing functional requirements as queries and retrieving services according to VSM-based similarity. We then prepared the functional requirements of two testbed systems as inputs for the IR-based system. Functional requirements for MIS were as follows:

- Ability to search for available movie tickets.
- Ability to search for movie-related information, such as the location of a movie theater and the scheduling of movies.
- Ability to search for movie DVDs.

Functional requirements for CIS were as follows:

- Ability to search for nearby shops, restaurants, hospitals, or clinics.
- Ability to search for the phone numbers of selected shops, restaurants, hospitals, or clinics.

- Ability to search for the location information, such as the addresses of restaurants or hospitals, of selected shops, restaurants, hospitals, or clinics.
- Ability to search for the course schedule of students.
- Ability to search for course information, such as materials and grades.
- Ability to display the location of classrooms.

For the experiments, we prepared 180 services (*i.e.*, service operations) as the basis of the search. Some of these services were extracted from OWLS-TC v4[12] and some were extracted from previous projects conducted in our lab. To enable an objective comparison between the proposed SCRUMTE system and the IR-based system, we used Precision (derived as the fraction of retrieved services deemed acceptable by the user), and Recall (derived as the fraction of acceptable services retrieved by the system), as our comparative indicators. The definitions of Top-K precision ($P^K(q_i)$) and Top-K recall ($R^K(q_i)$) are presented in Eqs. (6) and (7).

$$P^k(q_i) = \frac{\left|Rel(q_i) \cap Rank^k(q_i)\right|}{\left|Rank^k(q_i)\right|}, \tag{6}$$

$$R^k(q_i) = \frac{\left|Rel(q_i) \cap Rank^k(q_i)\right|}{\left|Rel(q_i)\right|}, \tag{7}$$

where $Rel(q_i)$ is the set of the relevant services pertaining to a given query q_i and $\text{Rank}^K(q)$ represents the set of Top-K web services related to query q_i.

All experiments were conducted using a desktop computer with the following configuration: Intel i7-930 2.8GHz with 8G RAM, 500G hard disk, and Windows 7 (64bit). Before conducting any experiments, we manually identified relevant services for each functional requirement and each extracted WSDL-based request to form a reference for evaluation. Service discovery was conducted using the SCRUMTE system and the IR-based system with the aim of calculating the Top-3, Top-5, Top-10, and Top-20 precision as well as Top-3, Top-5, Top-10, and Top-20 recall, by comparing retrieved services with services returned in the searches. The experiment results obtained from the two systems are presented in Figs. 8-11. From the experiment results, we can draw the following conclusions:

- Precision: SCRUMTE outperformed the IR-based system by 16%~58% with regard to MIS and CIS. This demonstrates the precision of the SCRUMTE system in the retrieval of appropriate services. Note that the requirements of CIS are more complex than those of MIS; therefore, the Top-K precision values for CIS were not as high as those obtained for MIS, when using either of the two methods. Nonetheless, SCRUMTE yielded good precision (89% top-3 precision).
- Recall: SCRUMTE outperformed the IR-based system by 32%~110%, particularly for the top-20 recall. Nearly all of the relevant services with top-20 rankings were successfully retrieved, thereby demonstrating the effectiveness of the proposed lexical expansion mechanism and method used in the calculation of similarity.

[12] http://projects.semwebcentral.org/projects/owls-tc/

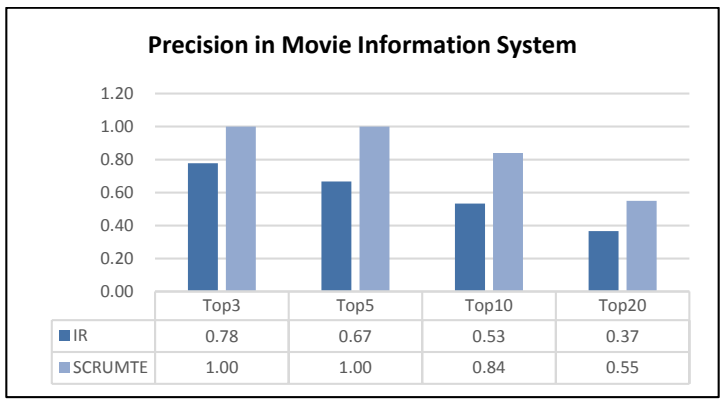

Fig. 8. Evaluation results of Top-K precision in movie information system.

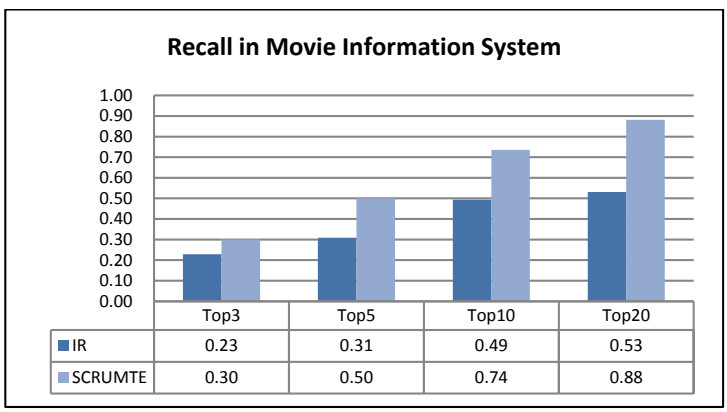

Fig. 9. Evaluation results of Top-K Recall in campus information system.

5. CONCLUSIONS

This paper reports a novel system using Service Component Retrieval with UML-based Modeling and Term Expansion (SCRUMTE) to assist in the retrieval of reusable service components for the development of software systems. Compared to existing service discovery techniques, this study makes the following contributions: 1) we provide a method for the translation of UML models into WSDL documents for use as service queries; and 2) we devised a mechanism to help software developers by facilitating web service discovery in the search for service components capable of meeting the requirements of software developers with regard to lexical similarity while minimizing the number of components. Experiment results demonstrate that the proposed SCRUMTE system is able to achieve a precision and recall superior to that of text-based service discovery schemes.

The SCRUMTE approach can be utilized in two ways: 1) when the developer has built UML models that represent system requirements, he/she can use SCRUMTE to retrieve appropriate service components that may fulfill parts of identified requirements,

from his/her organization's API library or public service repository. 2) the user can draw UML models intentionally based on his/her needs for web services and use SCRUMTE to acquire the information of candidate web services, without issuing numerous service queries manually. In conclusion, SCRUMTE facilitates the reuse of service components to speed up the development of new software projects and reduce overall costs.

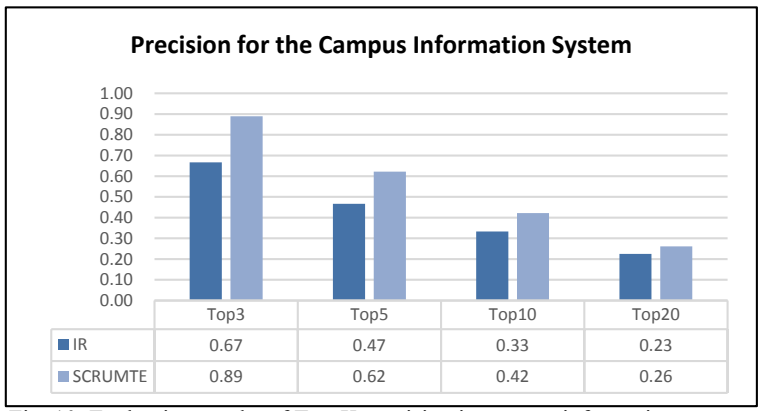

Fig. 10. Evaluation results of Top-K precision in campus information system.

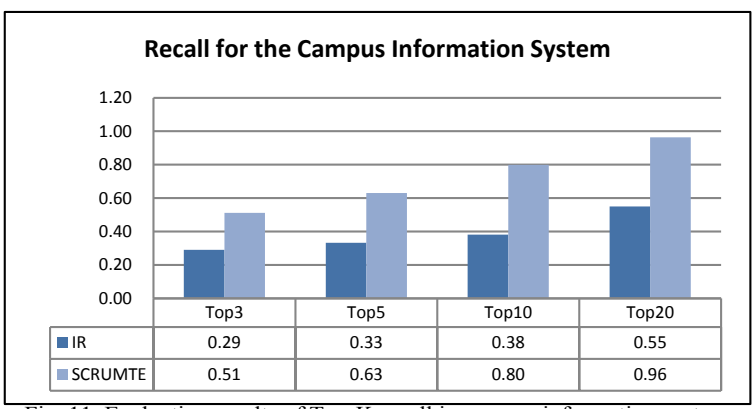

Fig. 11. Evaluation results of Top-K recall in campus information system.

REFERENCES

1. J. Lee, S.-P. Ma, and A. Liu, eds., *Service Life Cycle Tools and Technologies: Methods*, *Trends and Advances*, 2012, IGI Global: Hershey, PA, USA.
2. S.-P. Ma, C.-W. Lan, and C.-H. Li, "Contextual service discovery using term expansion and binding coverage analysis," *Future Generation Computer Systems*, Vol. 48, 2015, pp. 73-81.
3. S.-P. Ma, Y.-Y. Fanjiang, and J.-Y. Kuo, "Dynamic service composition using core service identification," *Journal of Information Science and Engineering*, Vol. 30, 2014, pp. 957-972.

4. J. Lee, *et al*., "Dynamic service composition: A discovery-based approach," *International Journal of Software Engineering and Knowledge Engineering*, Vol. 18, 2008, pp. 199-222.
5. J. Garofalakis, *et al*., "Contemporary web service discovery mechanisms," *Journal of Web Engineering*, Vol. 5, 2006, pp. 265-290.
6. K. Verma, *et al*., "METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of web services," *Information Technology and Management*, Vol. 6, 2005, pp. 17-39.
7. D. Martin, *et al*., "Bringing semantics to web services with OWL-S," *World Wide Web*, Vol. 10, 2007, pp. 243-277.
8. M. Klusch and P. Kapahnke, "The iSeM matchmaker: A flexible approach for adaptive hybrid semantic service selection," *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 15, 2012, pp. 1-14.
9. A. M. Zaremski and J. M. Wing, "Specification matching of software components," *ACM Transactions on Software Engineering and Methodology*, Vol. 6, 1997, pp. 333-369.
10. R. Chinnici, *et al*., *Web Services Description Language (WSDL) Version 2.0*., 2007, W3C.
11. G. Varelas, *et al*., "Semantic similarity methods in wordNet and their application to information retrieval on the web," in *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management*, 2005, pp. 10-16.
12. G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, Vol. 38, 1995, pp. 39-41.
13. K. Levi and A. Arsanjani, "A goal-driven approach to enterprise component identification and specification," *Communications of the ACM*, Vol. 45, 2002, pp. 45-52.
14. A. Arsanjani *et al*., "SOMA: A method for developing service-oriented solutions," *IBM Systems Journal*, Vol. 47, 2008, pp. 377-396.
15. B. Bauer and J. Müller, "MDA Applied: From sequence diagrams to web service choreography," in *Web Engineering*, N. Koch, P. Fraternali, and M. Wirsing, ed., 2004, Springer, Berlin Heidelberg, pp. 132-136.
16. D. Skogan, R. Gronmo, and I. Solheim, "Web service composition in UML," in *Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference*, 2004, pp. 47-57.
17. C. S. Wu and I. Khoury. "Web service composition: from UML to optimization," in *Proceedings of the 5th International Conference on Service Science and Innovation*, 2013, pp. 139-146.
18. G. Spanoudakis and A. Zisman, "Discovering services during service-based system design using UML," *IEEE Transactions on Software Engineering*, Vol. 36, 2010, pp. 371-389.
19. Z. Huma *et al*., "Towards an automatic service discovery for UML-based rich service descriptions," in *Model Driven Engineering Languages and Systems*, R. France *et al*., Ed., 2012, Springer, Berlin, Heidelberg, pp. 709-725.
20. M. Porter, "The porter stemming algorithm," http://www.tartarus.org/martin/PorterStemmer.
21. L. Baresi, M. Miraz, and P. Plebani, "A distributed architecture for efficient web

service discovery," *Service Oriented Computing and Applications*, 2015, pp. 1-17.
22. Z. Cong *et al*., "Service discovery acceleration with hierarchical clustering," *Information Systems Frontiers*, Vol. 17, 2015, pp. 799-808.
23. G. Brusa, M. L. Caliusco, and O. Chiotti, "Towards ontological engineering: a process for building a domain ontology from scratch in public administration," *Expert Systems*, Vol. 25, 2008, pp. 484-503.
24. G. Varelas, E. Voutsakis, P. Raftopoulou, E. G. Petrakis, and E. E. Milios, "Semantic similarity methods in wordnet and their application to information retrieval on the web," in *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management*, 2005, pp. 10-16.
25. Y. Li, Z. A. Bandar, and D. McLean, "An approach for measuring semantic similarity between words using multiple information sources," *IEEE Transactions on Knowledge and Data Engineering*,Vol. 15, 2003, pp. 871-882.

Wen-Tin Lee (李文廷) received his Ph.D. degree in Computer Science and Information Engineering from National Central University, Taiwan, in 2008. Lee is currently an Assistant Professor in the Department of Software Engineering and Manage-ment at National Kaohsiung Normal University. His research interests include software engineering, service-oriented computing and software process management.

Shang-Pin Ma (馬尚彬) received his Ph.D. degree in Computer Science and Information Engineering from National Central University, Taiwan, in 2007. Ma is currently an Associate Professor in the Department of Computer Science and Engineering at National Taiwan Ocean University. His research interests include service-oriented computing, software engineering, mobile computing, and semantic webs.

Yao-Yu Tsai (蔡燿宇) received his Bachelor (2011) and Master's (2013) degrees from the Computer Science and Engineering Department, National Taiwan Ocean University, Taiwan. His research interests include software engineering and service-oriented computing.