2014 IEEE 8th International Symposium on Service Oriented System Engineering

A Community-Driven Workflow Recommendations and Reuse Infrastructure

Jia Zhang[1], Chris Lee[1], Sean Xiao[1], Petr Votava[2], Tsengdar J. Lee[3], Ramakrishna Nemani[2], Ian Foster[4]
[1]Carnegie Mellon University, USA
[2]NASA Ames Research Center, USA
[3]Science Mission Directorate, NASA Headquarters, USA
[4]University of Chicago and Argonne National Lab, USA
jia.zhang@sv.cmu.edu, chris.lee@sv.cmu.edu, petr.votava@nasa.gov, tsengdar.j.lee@nasa.gov, rama.nemani@nasa.gov, foster@mcs.anl.gov

***Abstract*—**NASA Earth Exchange (NEX) aims to provide a platform to enable and facilitate scientific collaboration and knowledge sharing in the Earth sciences, as current satellite measurements rapidly magnify the accumulation of more than 40 years of NASA datasets. One of the main objectives of NEX is to help Earth scientists leverage and reuse various data processing software modules developed by their peers, in order to quickly run value-added executable experiments (workflows). Toward this goal, this paper reports our efforts of leveraging social network analysis to intelligently extract hidden information from data processing workflows. By modeling Earth science workflow modules as social entities and their dependencies as social relationships, this research opens up new vistas for applying social science to facilitate software reuse and distributed workflow development. As a proof of concept, a prototyping system has been developed as a plug-in to the NEX workflow design and management system (VisTrails) to aid Earth scientists in discovering and reusing workflow modules and extending them to solve more complex science problems.

I. Introduction

NASA's Earth observing satellite measurements rapidly magnify the accumulation of more than 40 years of scientific datasets. To analyze such vast amounts of data, Earth scientists in many laboratories have developed highly specialized software components (tools) and processes. The software components are often composited as workflows in order to define more comprehensive data analysis processes and explore various approaches. Aiming to provide a novel platform to facilitate knowledge sharing, scientific collaboration, and direct access to NASA compute resources, NASA Earth Exchange (NEX) [1] seeks to connect the Earth science community to accelerate the rate of discovery in Earth sciences. It is a collaborative compute platform that improves the availability of Earth science data, models, analysis tools and scientific results.

In recent years, workflow has become a popular technique for scientists to define executable multi-step procedures [2]. To help NEX users develop workflows, NEX has adopted VisTrails [3] as its workflow management system and has been working with the VisTrails team to provide baseline implementation of a number of tools as reusable VisTrails modules. Such VisTrails modules will allow NEX users to make data and computational resources on NEX accessible through both VisTrails' graphical development interface [4] as well as command-line interface.

The ability to discover and reuse knowledge (sharable workflows or workflow components - throughout this paper, the term artifact will be used to refer to either one) is critical to the future advancement of science especially in this information age. Many data projects in other domains (such as Kepler [5] in phylogeny, caGrid [6] in life science, and Science Gateway [7] in chemistry) have adopted similar approaches to help domain scientists find related artifacts. However, as reported in our earlier study in the biomedical domain [8], the reusability of scientific artifacts at current time is very low.

Scientists often do not feel confident in using other researchers' tools and utilities. For example, many Earth science processes and models have known input formats; input data thus usually have to undergo preprocessing steps. This is where everyone reinvents the wheel with each researcher or a research group creating their own version of tools for data reprojection, subsetting, mosaicking and so on. One major reason is that researchers are often unaware of the existence of others' data preprocessing processes. Meanwhile, researchers often do not have time to fully document the processes and expose them to others in a standard way. These issues cannot be overcome by the existing workflow search technologies used in NEX and other data projects.

Therefore, this project aims to develop a proactive recommendation technology based on collective NEX user behaviors. In this way, we aim to promote and encourage process and workflow reuse within NEX. Particularly, we focus on leveraging peer scientists' best practices to support the recommendation of artifacts developed by others.

Our underlying theoretical foundation is rooted in the social cognitive theory [9], which declares people learn by watching what others do. Our fundamental hypothesis is that sharable artifacts have network properties, much like humans in social networks [10]. For instance, workflows reveal "who knows what" (by one module calling another). More generally, reusable artifacts form various types of social relationships (ties) based on their invocation

978-1-4799-3616-8/14 $31.00 © 2014 IEEE
DOI 10.1109/SOSE.2014.23

dependencies, and may be viewed as forming what organizational sociologists who use network analysis to study human interactions [11] call a "knowledge network."

In particular, we will tackle two research questions:

R1: What hidden knowledge may be extracted from usage history to help Earth scientists better understand existing artifacts and how to use them in a proper manner?

R2: Informed by insights derived from the computing contexts, how could such hidden knowledge be used to facilitate artifact reuse by Earth scientists?

As shown in Fig. 1, our study of the two research questions will provide answers to three technical questions aiming to assist NEX users during workflow development: 1) How to determine what topics interest the researcher? 2) How to find appropriate artifacts? and 3) How to advise the researcher in artifact reuse?

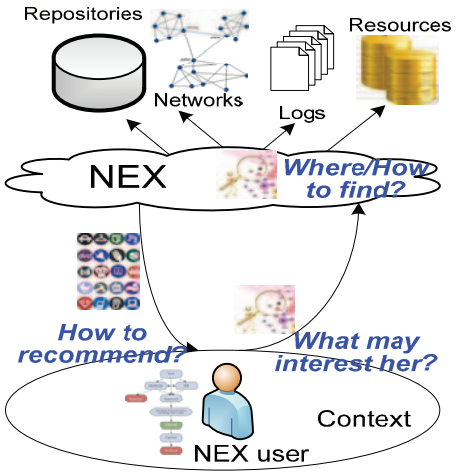


Fig. 1 Service-oriented workflow reuse.

In this paper, we report our efforts of leveraging social networking theory and analysis techniques [12, 13] to provide dynamic advice on artifact reuse to NEX users based on their surrounding contexts. As a proof of concept, we have designed and developed a plug-in to the VisTrails workflow design tool. When users develop workflows using VisTrails, our plug-in will proactively recommend most relevant sub-workflows to the users.

The remainder of the paper is organized as follows. In Section 2, we use a motivating example to explain the technical challenges. In Sections 3 and 4, we present our service social network and candidate service selection techniques, respectively. In Section 5 and 6, we present system implementation and preliminary experimental results, respectively. In Section 7, we discuss related work. In Section 8, we draw conclusions.

II. Strategy

In this section, we briefly introduce our strategy of building and testing a recommendation engine on top of existing techniques: VisTrails and ProgrammableWeb.

A. Plug-in to VisTrails

VisTrails is a popular open-source scientific workflow management system that provides support for data-oriented simulations, exploration and visualization [14]. It provides a visual programming language, so that domain scientists can drag and drop reconfigurable, predefined building blocks to visually define executable data processing workflows. Workflow components can be linked together through their input/output ports, to define execution order between them. VisTrails also supports web service technology, where workflow components can be specified as external software applications wrapped as web services with standard interfaces.

Known for its facility supporting data provenance, VisTrails has been adopted at NEX for scientists to design and execute data analysis workflows through VisTrails' graphical development interface [4]. Therefore, our strategy is to build our technique on top of the VisTrails, to design and develop a plug-in to VisTrails as a seamless workflow recommendation engine. With the formed partnership, this project will make sure that it aligns with the VisTrails to ensure the sustainability of the plug-in.

B. ProgrammableWeb as Testbed

As explained earlier, this project aims to design the next-generation of tools for scientific collaboration. One significant challenge is an inherent lack of user data. Our strategy is two-fold. First, we select a reusable software repository as our testbed to evaluate our technique and plug-in. Second, studying an existing repository will help us decide features useful for workflow recommendation to build next-generation workflow repository.

Based on our earlier study [15], we have decided to use ProgrammableWeb as our testbed, which is a repository of APIs (reusable software services) and mashups (service composition). We draw a natural analogy between the concepts in ProgrammableWeb and our study domain. An *API*, or Web 2.0 API, refers to a web application that can be remotely invoked (programmable) through its standard interface (e.g., REST or WSDL). Such web services are analogous to reusable scientific components that can be shared and recommended to peers. A *mashup* refers to a web application comprising published APIs in ProgrammableWeb. In other words, a mashup is a synergistic composition of APIs for a value-added service. Thus, a mashup is analogous to a scientific workflow that is typically and intentionally comprised of multiple existing software modules. Throughout the paper, we will use two

sets of terms interchangeably: (1) workflow and mashup; (2) module, service, and API.

By modeling the APIs and mashups as a social network, we can analyze their usage history and use it to formulate recommendations. Due to the natural mappings between workflow/module and mashup/API, we believe that similar analysis will hold true for scientific workflows.

Another reason why we chose ProgrammableWeb as our testbed is its rapid growth. Since its inception in late 2005, the number of APIs and mashups published at ProgrammableWeb has been increasing rapidly. Up to September 17, 2013, 9985 APIs and 7185 mashups have been published. Such rapid growth will help us evaluate the effectiveness and efficiency of our approach.

III. Service Social Network

A. Network Construction

All services and workflows are modeled as social nodes in a multi-modal, multi-relational, and multi-featured network called Service Social Network (SSN): a graph SSN = (V, E) where V is a finite set of nodes, and E is a finite set of edges. An SSN shows as a multiplex network. Multiple types of nodes co-exist: e.g., services and workflows. Edges may be classified according to the nature of the relationships that they represent, for example, usage or authorship. Each edge type may further carry an influential coefficient with respect to a given criterion, e.g., user ranking.

We start by modeling the "use" relationship between a workflow and a service. If a workflow invokes a service, a social edge is identified between them. A workflow-service network is thus established based on their inclusive relationships. Fig. 2 illustrates a segment of a constructed SSN, where an edge exists between a workflow (blue nodes) and a service (red node) if the service is used in the workflow. A workflow may uses multiple services; and a service may be used by multiple workflows. Therefore, there is a many-to-many relationship between them. Fig. 2 illustrates that a service is socially connected to many workflows based on the "use" relationship. From a workflow repository, the constructed SSN may be disconnected, comprising connected subgraphs and isolated nodes.

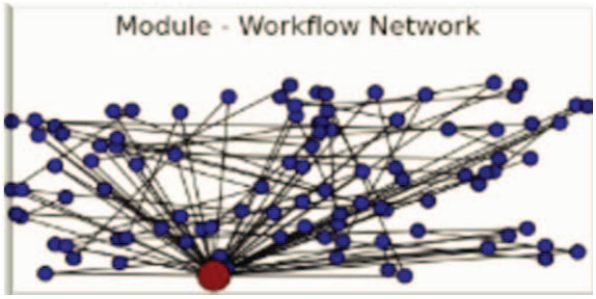


Fig. 2 Module-Workflow network.

The SSN can be formalized as a matrix Q that describes the involvement relationships between *m* workflows and *n* services:

$Q = [q_{ij}], 0 \le i \le m, 0 \le j \le n$, where:

$q_{ij} = 1$ if workflow *i* calls service *j*.

Based on the relation *Q*, another relation *S* can be retrieved from matrix calculation:

$S = Q^T \bullet Q = [s_{ij}], 0 \le i, j \le n$, where:

s_{ij} = the number of workflows where both services *i* and *j* are used; s_{ii} = number of workflows where service *i* is used.

Relation *S* represents a "social network" among services based on their collaboration relationship. The semantic meanings of its comprising edges are: if two services are used in the same workflow, a social tie is established between them. Note that rich context information may be carried by the edges in the network as labels. For example, an edge between two services may be labeled with the corresponding workflow that uses them. If two services are used together in multiple workflows, multiple edges can be created with proprietary labels. By analyzing the profile of a specific workflow, we may understand under which conditions the two services can be used together. Such a collaboration relationship maps to the association rules in social network analysis.

Collaboration ties can be further classified into various levels. The more workflows in which two services are used together, the tighter collaboration tie between them. In the SSN, a tight tie implies that multiple edges exist between a pair of service nodes. In contrast to direct collaboration relationship between two services where there exists edges between them, an indirect collaboration relationship is defined when a path exists between two service nodes. The collaboration tie can be thus calculated as follows. Assume that a path exists in SSN between services s_{i1} and s_{ik} with *(ik-2)* intermediate service hops:

$$ct(s_{i1}, s_{ik}) = \mathrm{ct} < s_{i1}, s_{i2}, s_{i3}, \dots, s_{ik} > = \sum_{j=1}^{k-1} \alpha^j ct(s_{ij,} s_{ij+1}),$$

Where $\alpha \in (0,1)$ is a coefficient to weigh the collaboration relationship of an edge.

B. Network Analysis

After establishing the workflow-service networks, we calculate various metrics over them to comprehend the interaction patterns between services and workflows, as well as patterns among service usages. We adopt the tradition of calculation of *centrality and prestige* in social network analysis, including degree centrality (popularity),

betweenness centrality, and clique. Through *degree centrality* analysis, highly reused services can be identified based on the popularity of corresponding nodes. Through b*etweenness centrality* analysis, hinge services can be identified. Through *clique*, we can identify collaborative services, i.e., association rules among services.

If a service is used multiple times in multiple workflows, popularity can be considered as an annotation to be used for future purpose. Using the *k-path* betweenness algorithm, we can identify the key services, in the sense that they collaborate with other services in user queries.

Through these analyses, we can answer user queries regarding service query (usage) behaviors. For example, "how are different services used together in workflows?" and "in what types of workflows is a service usually used?" As another example on service-service relationship, we can answer questions such as "are there many services that collaborate with each other in workflows, and how?" and "what are the key services in these collaborations?" Such information will help to construct service recommendation engine.

IV. Candidate Service Generation

When services are published, their service providers usually also publish some metadata such as its category, descriptions, and development team information. Such metadata will help shorten the candidate lists, before network analysis is conducted that may become expensive in large-scale service networks. Our earlier work extends the Support Vector Machine (SVM)-based text classification technique to enhance service-oriented categorization [15]. In this project, we will apply the similar method to categorize user interests in addition to services to generate shortened candidate list.

C. Label Services and User Interests

TF-IDF (Term Frequency-Inverse Document Frequency) is a traditional metric that calculates the weight of every term comprised in a document. Applying the idea to service repositories, we can use TF-ICF (Term Frequency-Inverse Category Frequency) to weight every term contained in a category [15]. Based on our previous work [16], Equations (1-3) show how to calculate TF-ICF for a term ($\boldsymbol{t}$) in category ():

$$\mathbf{n\_c} = \sum_{c \in C} \mathbf{I}(\exists \boldsymbol{s} \in \boldsymbol{c}: \boldsymbol{t} \in \boldsymbol{s}) \quad (1)$$

$$\mathbf{n\_s} = |\{\mathbf{s}: \mathbf{s} \epsilon \mathbf{c}\}| \quad (2)$$

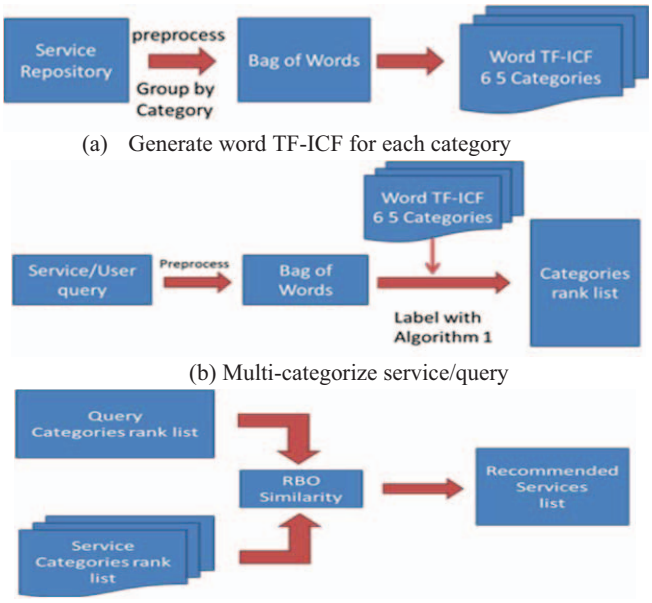
$$\text{tf–icf}_{t,c} = \frac{\text{num}(t,c)}{\text{n\_s}} \cdot (\alpha \cdot (1 - \frac{1}{\log \frac{|C|+1}{\text{n\_c}}}) + (1 - \alpha) \cdot \frac{\text{num}(t,c)}{\sum_{c \epsilon C} \text{num}(t,c)}) \quad (3)$$

where $\boldsymbol{num}(\boldsymbol{t}, \boldsymbol{c})$ represents the frequency of the term ($\boldsymbol{t}$) used in the category ($\boldsymbol{c}$), and C represents a collection of categories. It is divided by the number of services (n_s) in the category for normalization purposes. The frequency of the term is further adjusted by the distribution of terms over the corresponding category in a service repository. $\boldsymbol{\alpha}$ is a coefficient that can be adjusted in specific categories. Terms with high TF-ICF are keywords of the category. TF-ICF is pre-calculated for every term in each category, over the entire service repository.

Since a service may be relevant to more than one category, Algorithm 1 illustrates how TF-ICF is used to label a service (metadata) over multiple categories. After metadata is tokenized, we sum up TF-ICF for all comprising terms regarding each category, and rank the categories based on the summation. The top-K category list will be associated with the service. Labeling (categorizing) user interests follow the similar procedure as that of services.

Algorithm: **Label_Service**(metadata)
Loop c **in** C
  $sum_c = \sum_i \text{tf–icf}_{t_i,c}, \{t_i\} : \textit{all terms in metadata}$
$sortedSum = sort(sum)$;
$topKCategories = \emptyset$;
Loop i **from** 0 **to** $K-1$
 If sortedSum[i] == 0 **Then Break;**
 EndIf
End Loop
 $categoryList.append(sortedSum[i].category)$
End Loop
return $topKCategories()$;

D. Generate Candidate List

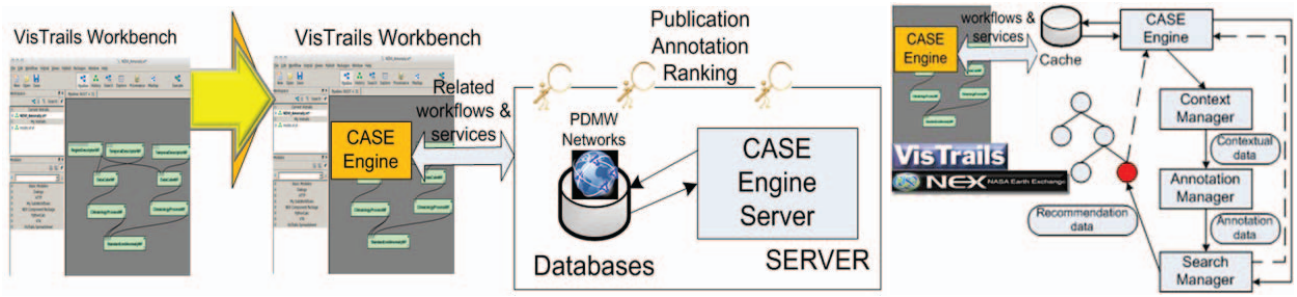


(a) Generate word TF-ICF for each category

(b) Multi-categorize service/query

(c) Use RBO similarity to generate candidates

Fig. 3 Generate candidate list



(a) (b)

Fig. 4 Workflow recommendation plug-in to VisTrails.

Fig. 3 shows how to generate candidate list. A user's query is assigned with a category rank list using Algorithm 1, which will be matched with service categories in the repository. TF-ICFs of all services are calculated (Fig. 3(a)). All services are multi-labeled using Algorithm 1 (Fig. 3(b)), so as to user interest when a user query comes. Fig. 3(c) shows how a candidate list is created using RBO similarity. The category list for user query is compared with all services' category list. Candidate services are ranked from most similar to least similar.

A score function is used to measure the relevance between user's query and the services in the repository. All services ranked with relevant score will be provided as a candidate list to enter service analysis phase.

The score function measures the similarity between two ranked lists. The ranking algorithm should handle non-conjointness, favor higher ranks, and be monotonic with increasing depth of evaluation. Such features allow us to model the issue as an indefinite ranking problem [17], whose rank-biased overlap (RBO) is calculated in Equation (2).

$$RBO\ (L,S,l,s) = \frac{1-p}{p}\left(\sum_{d=1}^{l}\frac{X_d}{d}p^d + \sum_{d=s+1}^{l}\frac{X_s(d-s)}{sd}p^d\right) + \left(\frac{X_l-X_s}{l}+\frac{X_s}{s}\right)p^l \quad (2)$$

L and S are two ranked lists with length l and s, respectively ($l \geq s$). p is the probability that the user intends to continue the comparison (e.g., $p = 0.98$). X_l represents the overlap on the seen lists at depth l.

V. System Implementation

Centered on the SSN, we have designed and developed a service search engine, as shown in Fig. 4. It comprises four major components as shown in Fig. 4(b): (1) data collection: Workflows and modules are collected incrementally from various data repositories. (2) annotation: Automatic annotation and statistical data generation creates additional knowledge to support module-oriented workflow composition. (3) search: Based on the SSN networks, this module supports relation-aware cross-workflow search function. (4) recommendation: This component provides guidance on workflow reuse during workflow development. Recommendations can be either passive (requested by users) or proactive (automatically delivered when a need is perceived). So we call it a Collection, Annotatoin, Search, rEcommendation (CASE) plugin.

The prototyping system is implemented as a plug-in to VisTrails. A local CASE engine will dynamically communicate with the central CASE engine at the server side to retrieve related modules and workflows. The service side CASE engine spawns several agents, each monitoring a data source (such as NEX workflow or data repositories). Any relevant event (e.g., publication of a new workflow) detected on the data sources will trigger a recalculation and subsequent changes to the SSN networks maintained at the server side. While a NEX user is building a workflow in VisTrails, our plug-in will observe the user's contextual environments, such as past workflows and the incomplete workflow.

As shown in Fig. 4(a), VisTrails provides a visual interface that displays the module composition in a workflow. Since our prototype is a plug-in to VisTrails, we provide a graphical interface to support workflow recommendation to NEX users, to keep consistent with VisTrails. For example, we allow NEX users browse PDMW networks and comprised (social) relationships.

A. System Architecture

Fig. 5 illustrates the internal architecture of our plugin. A Source Manager is in charge of loading historical usage information from external data sources, e.g., ProgrammableWeb. A Context Manager is in charge of modeling and managing the context of a user in the process of a workflow design. A Search Engine is connected to a typical keyword-based search engine to conduct syntactical match making.

External data sources are monitored by the Source Manager. When changes happen, the Parser module will filter the related information and input to the Social Analyzer that will alter the PDSW network. As shown in Fig. 5, Python-oriented graphical tools (PyQt and PyPlot)

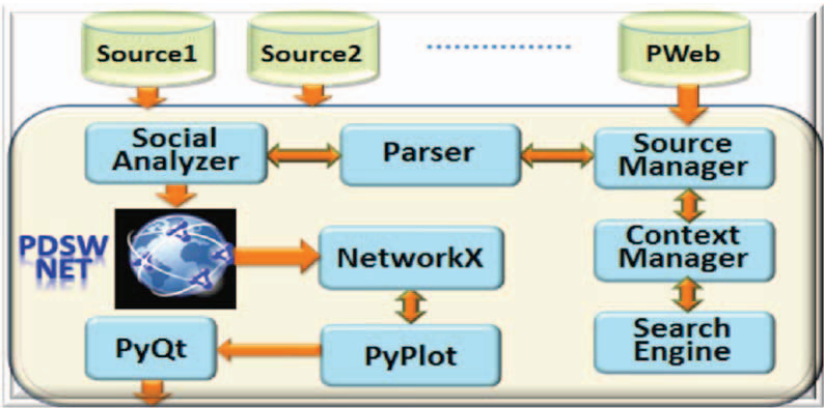


Fig. 5 System architecture.

are used to provide visualization to the service network.

As explained earlier, our recommendation engine is implemented as a plug-in to VisTrails. Thus, working with the VisTrails team, we have identified the least amount of four modules where changes are applied, as summarized in Table I. All plug-in code stays under the "package" directory, while corresponding menu items and their event handlers are initiated by overwriting the function menu_items in __init__.py file under the plugin package.

Table I. Changed VisTrails modules

Path	Function
packages/componentSearch/__init__.py	Initialize the plugin menu and callback
packages/componentSearch/component_search_form.py	The form for searching and displaying result
packages/componentSearch/networkx_graph.py	The form for drawing the graph of relationship among APIs or mashups
db/programmableweb/mongo_source.py	The API to retrieve data from database (e.g., MongoDB)

All functions in the plugin are triggered with the VisTrails' in-context GUI. After initializing a menu and the handler, one has to display the GUI components to present the features in the VisTrails GUI framework, which is constructed in pyQt. For example, when a search form is needed for a user to input keywords, an object containing an instance of QWidget class will be created as the frame of the VisTrail in-context GUI.

The graphical analysis is realized by integrating matplotlib with VisTrails' in-context GUI. Service networks are modeled using networkx package, and then plotted by matplotlib and Qt4 libraries. When analyzing the mashup-API relationships, if two APIs are shared in the same mashup, an edge will be created between the two API nodes.

B. Usage Scenarios

We have implemented our workflow recommendation engine as a plug-in to VisTrails. The following usage scenarios will briefly depict how our plug-in can be used in the process of a workflow design to facilitate productivity, as illustrated in Fig. 6.

Naturally, when designing a workflow, the user has some general ideas about its purpose. These ideas can be written into a short description of the experiment. We can conduct some simple natural language processing procedures to identify keywords that mainly represent the intended aims. The set of resulting keywords will be used as a query into the search engine. For example, for a user interested in analyzing the efficiency of roads in a certain area, a keyword may be identified as "map." The built-in API search module in our plug in will look up related APIs from the API repository, and return with a collection of candidates, as shown in Fig. 6.

After the user picks up an API from the list, our plug-in will dynamically generate a network of all related APIs that are usually used together in some existing mashups. As shown in Fig. 6, the selected node is colored red and enlarged. A node is blue most of the time, and is enlarged if it has already been added to the project. If the user decides to take one recommended API, the process will iterate and more recommendations will be provided.

Alternatively, if the user decides to reuse a mashup, our plug-in will recommend related mashups, at the current time, based on the historical usage that some mashups use mutual APIs (in other words, they have similar goals).

VI. Experiments and Evaluations

A. Environmental Setup

As explained in earlier section, ProgrammableWeb is used as our testbed. ProgrammableWeb publishes a set of REST interfaces that can be leveraged to retrieve APIs and mashups in their repository. Note that one must first register for a developer key in order to access the REST interfaces.

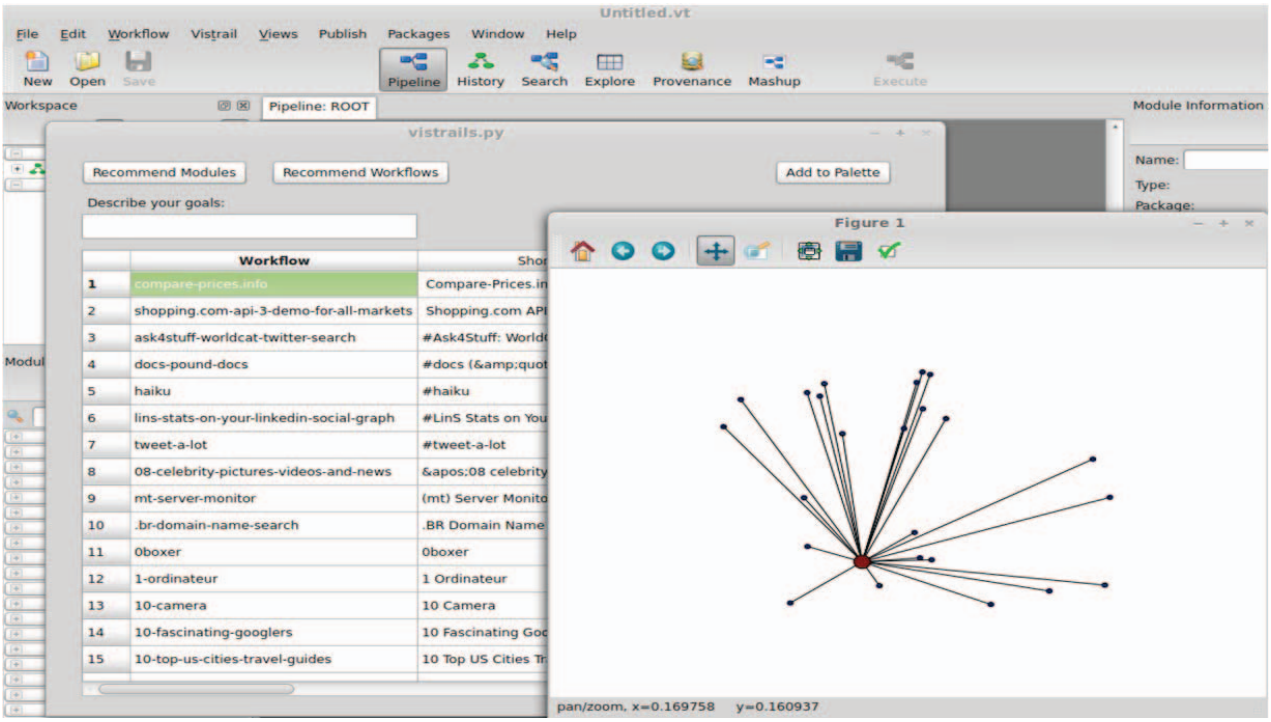


Fig. 6 Usage history-supported recommendation.

We have studied ProgrammableWeb since 2011 [15].

The metadata of all APIs can be retrieved through REST calls, divided by pages. The results are in either XML or JSON format. However, we found that an API may have multiple entries (e.g., multiple versions). Our solution is to first identify all duplicate entries, then use the API name to query ProgrammableWeb to get the latest API and discard other entries. A similar approach is applied to retrieve all mashups from ProgrammableWeb. In the metadata of a mashup, a field "apis" records the APIs used by the mashup. Such information helps us to identify all API-mashup connections (i.e., edges) of the service network. Derived networks are generated subsequently. Currently, all networks are stored in MongoDB, which is a nosql database that possesses a hybrid *collection* concept that can represent various types of nodes and edges in our network.

B. Experimental Setup

We have carefully designed a set of experiments to evaluate the effectiveness of our approach. Specifically, we focus on examining under which circumstances our method complements the existing search methods and enhances service discovery and recommendation. Our experiments are designed as follows.

Since ProgrammableWeb contains a set of predefined categories, we study the APIs in corresponding categories separately. For each category, all APIs published in the category are gathered to form a corpus. At this stage, the tags of APIs are considered and accumulated. A vector is constructed to represent every category.

We examined the relevance of each API in the corresponding category. Based on the built category vector, the relevance of each comprised API is calculated as follows. A vector is first built for every API. The similarity between the API and the category is calculated using the dot product. Thus, the top m (e.g., $\sqrt{N}$) APIs are identified based on their usage of the popular keywords in the category. Our assumption is that, the APIs tagged with popular keywords are more likely to be searched by users, and consequently more likely to be used by mashups in the past. Such past usage history can be leveraged by our presented method to refine API recommendation.

Now we simulate user requests: all contained tags of an API form a valid user request. Thus, we simulate a collection of reasonable user requests for a category from its comprised APIs. For each user request, we again use its contained tags to build a vector. It is used to conduct keyword search with all other APIs in the category. For each API under examination, its tags and descriptions are used to construct a vector for the API. The similarity (dot product) between the constructed API vector and the user request vector is calculated and used to decide its ranking in the recommendation list.

Since each user request will correspond to the API used to build the request, we further examine the historical usage of the API: 1) whether it is used by other mashups and how many; and 2) whether it is used together with other APIs in some mashups and how many. The more connections found, the more likely that our method can better adjust the

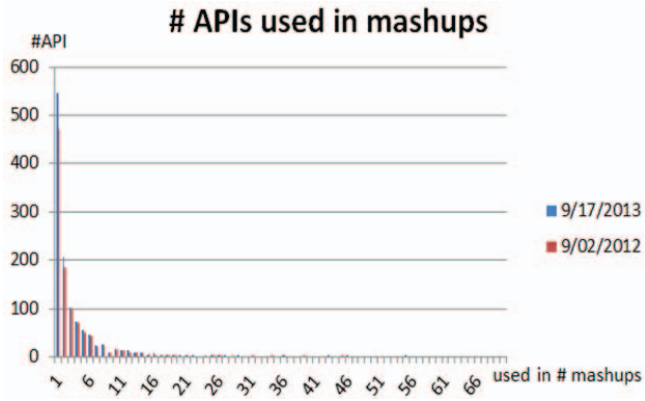


Fig. 7 Overall reuse status.

recommendation.

Our assumption is that a category with more APIs will attract more attention and thus imply more historical usage data. We have thus focused on the following top 10 categories, in a descendent order: Tools (726), Internet (609), Social (501), Financial (454), Enterprise (439), Reference (345), Mapping (341), Shopping (337), Government (314), and Science (313).

C. Experimental Results and Analysis

Our study reveals that current API reusability is very low. Fig. 7 shows the API reuse status at ProgrammableWeb on September 17, 2013. Most mashups call one to two APIs; only a very small number of mashups call more than three APIs (380 mashups as 5% of the total of 7188 mashups). 8694 APIs (87%) are not used in any mashups. Fig. 7 compares the status in 2013 and 2012: the enhancement is not significant. On September 2, 2012, 7142 APIs were published, 5993 APIs (83%) were not used by any mashup.

Table II summarized our experimental results on testing the effectiveness of our approach. We measure three numbers for each category: total number of APIs; *reused* indicating the number of times the APIs are contained in the 'apis' field of at least one mashup; *collaborated* indicating the number of mashups containing at least two APIs. The top 10 categories with the most APIs are listed. Take category *Tools* in Table II as an example. It contains 726 APIs; 46 APIs are used by some mashups (6.34%); 37 APIs are paired with other APIs (5.1%). According to our recommendation algorithm, the rankings of these APIs will be raised and highlighted. Consider the category *Shopping* as another example. About 25% of the APIs are reused, and over 20% of them have collaboration relationship. Such historical data usage analysis will help to proactively recommend such modules. As shown in Table II, similar findings were supported by other categories.

As revealed by Table II, although many more APIs are published to ProgrammableWeb, the reuse of existing APIs remains quite low. The highest record is less than 25% in the *Mapping* category. On one hand, this fact strengthens the necessity of our study and the usefulness of our technique. On the other hand, this fact has resulted insufficient dataset for our technique to leverage. Besides, it appears that although 'Tools' contains a much larger number of APIs (726), the reuse of its APIs (6.43%) lags behind many other categories.

Table II. Experiments on testbed

Snapshot on 9/23/2013

Category	#APIs	Reused	Collaborated
Tools	726	46 (6.34%)	37 (5.1%)
Internet	608	84 (13.8%)	77 (12.66%)
Social	501	95 (18.96%)	89 (17.76%)
Financial	454	18 (3.96%)	13 (2.86%)
Enterprise	439	19 (4.32%)	18 (4.10%)
Reference	345	58 (16.81%)	46 (13.33%)
Mapping	341	83 (24.34%)	72 (21.11%)
Shopping	337	71 (21.06%)	64 (18.99%)
Government	314	32 (10.19%)	24 (7.64%)
Science	313	8 (2.55%)	7 (2.23%)

Snapshot on 9/2/2012

Category	#APIs	Reused	Collaborated
Internet	470	80 (17.02%)	73 (15.53%)
Tools	467	37 (7.92%)	28 (5.99%)
Social	405	83 (20.49%)	81 (20%)
Financial	282	13 (4.60%)	9 (3.19%)
Enterprise	282	16 (5.67%)	15 (5.31%)
Reference	272	51 (18.75%)	39 (14.33%)
Mapping	264	79 (29.92%)	68 (25.75%)
Shopping	253	64 (25.29%)	57 (22.52%)
Telephony	233	24 (10.30%)	22 (9.44%)
Government	227	26 (11.45%)	18 (7.92%)

Nevertheless, our experiments have demonstrated that our method may complement the keyword-based search method to enhance service and workflow discovery. As more past usage history becomes available, our method becomes more effective. In addition, our method can help find services that are relevant but assigned to other categories, i.e., cross-category search.

Fig. 8 further examines the correlation between collaboration and reuse status, for the top 10 categories at ProgrammableWeb based on the number of APIs contained. For each category, we focused on its most “popular” 25 APIs, based upon their similarity with the test cases (detailed can be found in Section VI.B). Similarity is measured by taking the dot product of the description vector of a particular API with the category vector that contains the descriptions of all APIs in the category. The rationale is that, we focus on the APIs that more likely to be found by users. In other words, we intend to establish a benchmark for our study. For the selected most related APIs, we

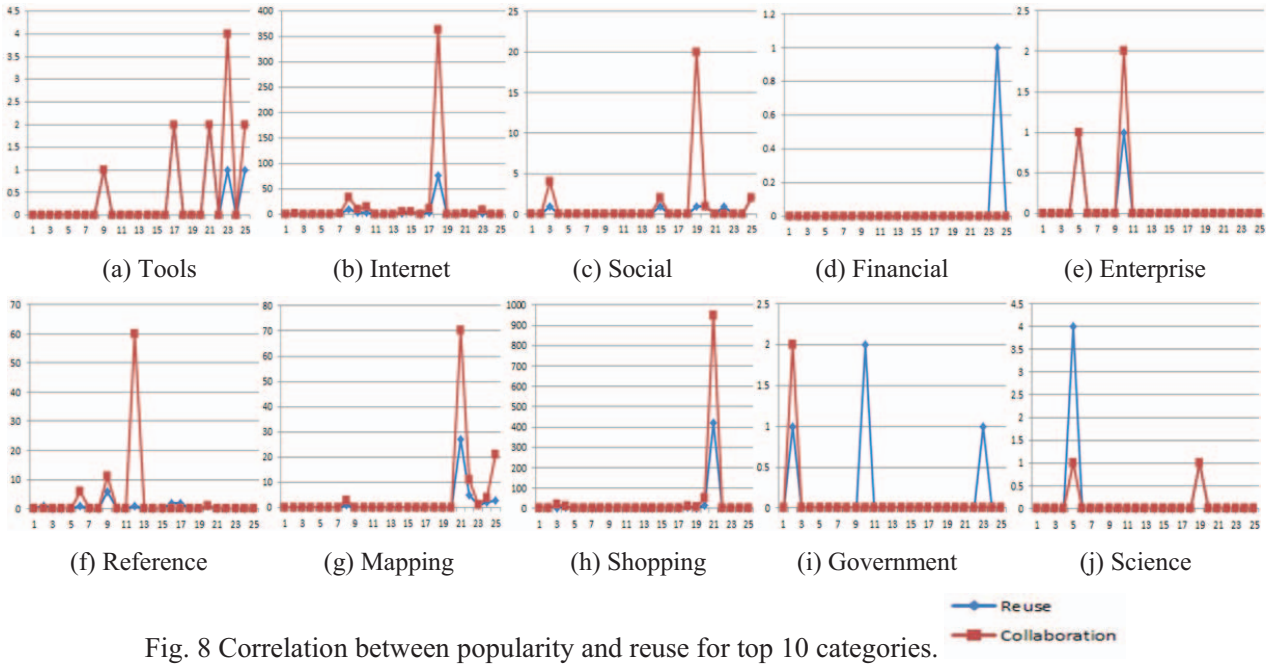


Fig. 8 Correlation between popularity and reuse for top 10 categories.

examined its reusability and collaboration history.

In Fig. 8, the horizontal axis represents the indices of the top 25 APIs using an arbitrary index, and the vertical axis represents the counts. "Reuse" (represented by diamond dots) means the number of mashups that use a particular API. "Collaboration" (represented by square dots) means the number of other APIs that are used in the same mashup as a particular API.

As shown in Fig. 8, it is noted that reuse is not too common and that popularity does not necessarily equate with "easy to find" ("similarity") using the description vector metric. Less than 15% of APIs are actually used by mashups, and less than 15% of APIs collaborate with other APIs. For example, in the category *Financial,* only one popular API is used by one mashup; and there is no collaboration relationship found in APIs. Only for some established categories such as *Tools*, *Internet*, and *Mapping*, it is more likely for easy-to-search APIs to be reused. For example, in category *Tools*, five most popular APIs (20%) collaborate with other APIs; two popular APIs (8%) are used by mashups.

This finding implies that using previous historical usage to help determine recommendations will likely to have effective results than using standard keyword search alone. Our finding also suggests that the two approaches can complement each other. It should be noted that whether or not the results are "good" requires a human to come in and evaluate them. In summary, our experiments prove that our techniques will be able to improve the effectiveness of service and workflow recommendation.

VII. Related Work

Workflow recommendation has been gaining more attention in the recent years. VisComplete [18] provides auto-complete suggestions by mining frequent patterns in existing pipelines. Leake et al. [19] propose a CASE-based approach to suggest the possible next step(s) aiding reuse of portions of prior workflows. In recent years, workflows allow reusable services as components. Thus, workflow recommendation largely relies on service discovery and recommendation.

Most work on service discovery and recommendation is based on either syntactical or semantic match making. Various information retrieval techniques, such as support vector machine (SVM) and term frequency-inverse document frequency (TF-IDF), have been used to build service search engines. The service descriptions of candidate services are analyzed against expected service queries based on syntactic match making process [20]. Our previous work enhances the SVM technique to facilitate and validate service categorization [15], a technique that automatically extracts semantic metadata from static WSDL service descriptions [21]. Other researchers focus on applying semantic technology to support service discovery [20]. Our earlier work yields a service semantics model empowered by a technique that automatically extracts semantic metadata from static WSDL service descriptions [21]. In addition, some other researchers take into consideration of QoS requirements to recommend services appropriately [22].

In our previous work, we have demonstrated that hidden usage history of services can be extracted from published scientific applications to facilitate capability reuse [23]. A

holistic set of service networks are established, and network analysis over them [24] opens a gateway to this exploration. We developed a pilot prototyping system [23] as a plug-in to the Taverna workflow workbench.

After our paper was published in 2011, recent years have witnessed more work of leveraging service usage history. Cao et al. [25] recommend Mashup services to users based on usage history and service network. User interests are also extracted from their Mashup service usage history. Huang et al. [26] conducted an empirical network analysis to quantitatively characterize the static structure and dynamic evolution of a Service-Mashup ecosystem at ProgrammableWeb. Zhou et al. [27] model services, associated with their attributes and entities, in a heterogeneous service network. A neighborhood random walk distance measure algorithm is proposed to provide service ranking. In contrast to related work, our CASE-enabled, fine-grained workflow/subworkflow recommendation mechanism will retrieve and rank related artifacts based on the scientist's interest and context—what we call "recommend-as-you-go."

Social relationships between service providers and service users have also been widely considered as important factors to help service discovery and recommendation. Xu et al. [28] establish a coupled matrix model to analyze multi-dimensional social relationships among potential users, topics, mashups, and services. A factorization algorithm is designed to predict unobserved relationships. Vollino and Becker [29] study service usage profiles to identify distinct groups of service users and their usage characterization in terms of service functionality. In contrast to their work, our research focusing on studying "social" relationships among services based on how they are interact with each other in the past workflows.

Some researchers also study change impact propagation during service evolution. For example, Oliva et al. [30] study the static dependency among workflows stored in a centralized repository, and derive change impact analysis for the workflows; Chinthaka et al. [31] analyze workflow revision history to track the evolution of targeted research. Their research results may help our future work on SSN network evolution and notification at runtime.

VIII. Conclusions

In this paper we have reported our efforts of building a historical usage-based workflow recommendation engine. The primary impetus behind the project is the movement to facilitate collaboration among scientists. To this end, the front-end of the workflow recommendation system takes the form of a VisTrails plugin, which allows it to be easily dropped into any VisTrails environment. As a result, our tool is seamlessly and transparently integrated into NEX users' familiar design environment while providing real-time artifact reuse guidance and recommendation. We have reported a prototyping system as a proof of concept.

We plan to continue our research in the following directions. We plan to enhance our recommendation techniques with semantics-based discovery approaches. We also plan to accumulate practice data to create benchmarks for the presented approach in this paper.

IX. Acknowledgement

This work is partially supported by National Aeronautics and Space Administration, under grant NASA NNX13AB38G. We thank Song Luan for his earlier involvement.

X. References

[1]. N.A.R. Center, "NASA Earth Exchange (NEX)", accessed, Available from: https://c3.nasa.gov/nex/.

[2]. C. Goble and D.D. Roure, *The Impact of Workflow Tools on Data-centric Research*, in *T. Hey, S. Tansley, and K. Tolle, eds., The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, Oct.* 2009. p. 137-145.

[3]. J. Freire, C.T. Silva, S.P. Callahan, E. Santos, and C.E. Scheidegger, "Managing Rapidly-Evolving Scientific Workflows ", *Lecture Notes in Computer Science*, May, 2006, 4145/2006: pp. 10–18.

[4]. L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. 2007: Springer.

[5]. "Kepler", accessed, Available from: https://kepler-project.org/.

[6]. caGrid, accessed, Available from: http://www.myexperiment.org/search?query=cabig&type=workflows.

[7]. "Science Gateway", accessed, Available from: http://www.sciencegateway.org/.

[8]. W. Tan, J. Zhang, and I. Foster, "Network Analysis of Scientific Workflows: a Gateway to Reuse", *IEEE Computer*, Sep., 2010, 43: pp. 54-61.

[9]. A. Bandura, *Social Foundations of Thought and Action: A Social Cognitive Theory*. Prentice-Hall Series in Social Learning Theory. 1985, Englewood Cliffs, NJ, USA: Prentice-Hall, Inc.

[10]. K. Carley, "A Theory of Group Stability", *American Sociological Review*, 1991, 56(3): pp. 331-354.

[11]. K. Carley, "On the Evolution of Social and Organizational Networks", *Research in the Sociology of Organizations*, 1999, 16: pp. 3-30.

[12]. P.J. Carrington, J. Scott, and S. Wasserman, *Models and Methods in Social Network Analysis*. 2005, Cambridge University Press: Cambridge.

[13]. S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. 1994: Cambridge University Press, Cambridge.

[14]. E. Santos, L. Lins, J. Ahrens, J. Freire, and C. Silva, "VisMashup: Streamlining the Creation of Custom Visualization

Applications", *IEEE Transactions on Visualization and Computer Graphics*, 2009, 15(6): pp. 1539-1546.

[15]. J. Zhang, J. Wang, P.C.K. Hung, Z. Li, N. Zhang, and K. He, "Leveraging Incrementally Enriched Domain Knowledge to Enhance Service Categorization", *International Journal of Web Services Research (JWSR)*, 2012, 9(3).

[16]. J. Zhang, J. Wang, P.C.K. Hung, Z. Li, J. Liu, and K. He, "Leveraging Incrementally Enriched Domain Knowledge to Enhance Service Categorization", *International Journal of Web Services Research (JWSR)*, 2012, 9(3): pp. 43-66.

[17]. W. Webber, A. Moffat, and J. Zobel, "A Similarity Measure for Indefinite Rankings", *ACM Transactions on Information Systems*, Nov., 2010, 28(4): pp. Article No: 20.

[18]. D. Koop, C.E. Scheidegger, S.P. Callahan, J. Freire, and C.T. Silva, "VisComplete: Automating Suggestions for Visualization Pipelines", *IEEE Transactions on Visualization and Computer Graphics*, 2008, 14: pp. 1691-1698.

[19]. E. Chinthaka, J. Ekanayake, D. Leake, and B. Plale, *CBR Based Workflow Composition Assistant*, in *2009 Congress on Services -- I*. 2009, IEEE Computer Society. p. 352-355.

[20]. A. Bouguettaya, S. Nepal, W. Sherchan, X. Zhou, J. Wu, S. Chen, D. Liu, L. Li, H. Wang, and X. Liu, "End-to-End Service Support for Mashups", *IEEE Transactions on Services Computing (TSC)*, Jul.-Sept., 2010, 3(3): pp. 250-263.

[21]. J. Zhang, R. Madduri, W. Tan, K. Deichl, J. Alexander, and I. Foster, "Toward Semantics Empowered Biomedical Web Services", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, Jul. 4-9, 2011, Washington DC, USA, pp. 371-378.

[22]. M. Picozzi, "Quality-Based Recommendations for Mashup Composition", in Proceedings of *International Conference on Web Engineering (ICWE)*, Jul. 5-9, 2010, Vienna, Austria, pp. 360-371.

[23]. J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri, "Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse", in Proceedings of *IEEE International Conference on Services Computing (SCC)*, Jul. 4-9, 2011, Washington DC, USA, pp. 48-55.

[24]. W. Tan, J. Zhang, and I. Foster, "Network Analysis of Scientific Workflows: A Gateway to Reuse", *IEEE Computer*, Sep., 2010: pp. 54-61.

[25]. B. Cao, J. Liu, M. Tang, Z. Zheng, and G. Wang, *Mashup Service Recommendation based on Usage History and Service Network*, in *International Journal of Web Services Research (JWSR)*. 2013: Santa Clara, CA, USA.

[26]. K.M. Huang, Y.S. Fan, and W. Tan, "An Empirical Study of ProgrammableWeb: A Network Analysis on a Service-Mashup System", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, Jun. 24-29, 2012, Hawaii, USA, pp. 552-559.

[27]. L.L. Yang Zhou, Chang-Shing Perng, Anca Sailer, Ignacio Silva-Lepe, and Zhiyuan Su, "Ranking Services by Service Network Structure and Service Attributes", in Proceedings of *IEEE 20th International Conference on Web Services (ICWS)*, Jun. 27-Jul. 2, 2013, Santa Clara, CA, USA, pp. 26-33.

[28]. W. Xu, J. Cao, L. Hu, J. Wang, and M. Li, "A Social-Aware Service Recommendation Approach for Mashup Creation", in Proceedings of *IEEE 20th International Conference on Web Services (ICWS)*, Jun. 27-Jul. 2, 2013, Santa Clara, CA, USA, pp. 107-114.

[29]. B. Vollino and K. Becker, "A Framework for Web Service Usage Profiles Discovery", in Proceedings of *IEEE 20th International Conference on Web Services (ICWS)*, Jun. 27-Jul. 2, 2013, Santa Clara, CA, USA, pp. 115-122.

[30]. G.A. Oliva, M.A. Gerosa, D. Milojicic, and V. Smith, "A Change Impact Analysis Approach for Workflow Repository Management", in Proceedings of *IEEE 20th International Conference on Web Services (ICWS)*, Jun. 27-Jul. 2, 2013, Santa Clara, CA, USA, pp. 308-315.

[31]. E. Chinthaka, R. Barga, B. Plale, and N. Araujo, *Workflow Evolution: Tracing Workflows Through Time*. 2009.