iMashup: Assisting End-User Programming for the Service-Oriented Web

Xuanzhe Liu *
Institute of Software, Peking University
Beijing, China, 100871,
liuxzh@sei.pku.edu.cn

Qi Zhao
Institute of Software,Peking University
Beijing, China, 100871,
zhaoqi06@sei.pku.edu.cn

Gang Huang
Institute of Software,Peking University
Beijing, China, 100871,
huanggang@sei.pku.edu.cn

Zhi Jin
Institute of Software,Peking University
Beijing, China, 100871,
zhijin@sei.pku.edu.cn

Hong Mei
Institute of Software,Peking University
Beijing, China, 100871,
meih@pku.edu.cn

ABSTRACT

The Web is currently moving towards a platform with rich services. A notable trend is that end-users create mashups by composing services with short, iterative development life cycles as well as updating with evolving needs. However, the large number of services and the high complexity of composition constraints make manual composition extremely difficult. Addressing this issue, we have developed an approach to assisting the end-users to build mashups in a simple and fast fashion. A tag-based model provides end-users a quick and intuitive insight of services. End-users simply describe their desired goals with tags. Interacting with a service repository, our approach employs a planning approach to suggest services that end-users might want to involve in the final outputs, including some additional interesting or relevant ones to induce more potential composition opportunities. End-users are allowed to iteratively modify, adjust or refine their goals. We have implemented our approach with a tool called iMashup.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques—*User interfaces*; D.2.6 [**Software Engineering**]: Programming Environments—*Interactive environments*; H.3 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

General Terms

DESIGN, HUMAN FACTORS

*corresponding author

ASE'10, September 20–24, 2010, Antwerp, Belgium.

Keywords

End-user software engineering,service composition, mashups

1. INTRODUCTION

The Web is now evolving from a primarily publication platform to a participatory platform with rich available services[1]. Spurred by Services Computing and Web 2.0 paradigm, the Web is further moving towards an extensible development platform, where data and services are mixed and published in innovative ways, along with lightweight programming and development technologies[8]. A noteworthy trend is the rapid growing community of casual end-users creating so-called "web mashups" that combine existing web services and data sources[7]. The end-users' desire of creating such mashups is exactly strong. As of February 2010, Programmableweb.com has published more than 4,700 web mashups developed by a number of end-users. In other words, there are three mashups deployed every day on average, which is a rather rapid growth in any metric. Several mashup tools have been developed, such as Yahoo! Pipes[9], IBM DAMIA[3] and Google Mashup Editor[2], provide Graphical User Interfaces for end-users. However, due to the ever-increasing number of web services, a key problem remained is not trivial because end-users expect to have a simple, quick and native representation of the service inputs and outputs rather than the hard-to-understand complex specifications and protocols (e.g., the XML encoding), and to figure out all the intermediate steps needed to generate the desired mashups automatically. Most current tools emphasize low-level data processing or programming techniques that are beyond the ability of average web users, rather than processing existing web-based services. Moreover, many end-users may have a general wish to know what they are trying to achieve, but not know the specifics of what they want or what is possible. It means that the process of designing and developing mashups requires not only the attention to the individual services, but also much broader perspective on the evolving collections of services that can potentially incorporate with current ones.

Addressing the problem above, we propose an approach with a tool, called iMashup, that will let end-users create these kinds of mashups interactively. Rather than the lower-

level service interface details, iMashup associates widely-used tags to services that are simple and close to the end-users native representation, so as to lower their entry barrier. Once the user specifies a tag as a goal, iMashup interacts with a service repository (called *Service Community*[4])and performs the planning approach as a two-phase forward-backward search stage to retrieve all possible solutions. These solutions not only contain the desired outputs, but also some additional interesting or relevant ones to induce more potential composition opportunities. iMashup allows end-users to visually revise the planned recommendations, and iteratively refine their goals until the final outputs satisfy their requirements. The main contributions of this paper are as follows:

- An approach for reducing the end-users efforts while reusing existing services and retrieving potentially relevant services by providing possible composition suggestions;
- A service model for providing end-users a quick and intuitive insight of services and lowering their entry barrier from the underlying processing details;
- A technique for planning not only the desired outputs of the end-users but also some additional interesting or relevant ones to induce more potential composition opportunities;
- A web-browser based tool for the approach to aiding rapid end-user development of mashups.

2. OVERVIEW OF APPROACH

The approach overview is shown in Figure 1, which is consisted of three main distinct components: a service repository named *Service Community*, the *Service Advisor*, and the *User Interface*. Our approach is consisted of four main phases, all of which can be iterated more than once. At the beginning phase, *Service Community* processes tags by a splitting-clustering technique, and associates tags with web services input/output messages. In the next phase, end-users can interact with iMashup, where a *tag cloud* will be displayed to end-users, and they can view all goals that can be obtained and processed in current application domain. End-users can select one or more tags from the tag cloud as the initial request, and *Service Advisor* will lookup *Service Community* and discover all services that can generate such tag at least once. Then, end-users may add one or more tags as the desired outputs, and *Service Advisor* will employ a graph-based planning technique to generate the possible relevant compositions outputs,and display ranked solutions on the user interface. Finally, end-users are allowed to investigate both individual services and the overall compositions, choose one of the generated composition solution that is best satisfying, submit to the *iMashup Runtime Engine* to interpret and call the associated services, and attain the immediate results with user interfaces.

2.1 Service Repository

To support our approach, we have developed a service repository called *Service Community*[4], to take charge of managing published web services and mashup applications. *Service Community* provides *Service Publication Interfaces* for both service providers and end-users to publish or share

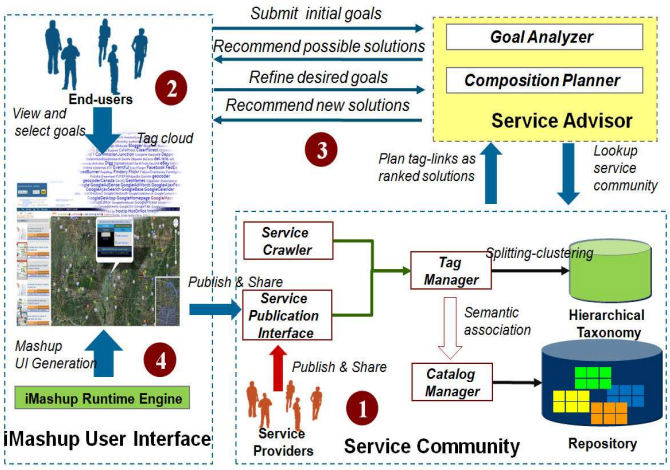

Figure 1: Overview of iMashup Approach

services and mashup applications, and collaborate for particular interests. It also exploits a *Service Crawler* to download available web services from existing registries and the published mashups on Programmableweb. With a series of splitting-clustering techniques[4], the *Tag Manager* extracts all tags on *Service Community*, from both service descriptions and user annotations, derives the tag hierarchies and presents the *tag cloud* to describe the capability of application domain. These tags are automatically associated with the service input and output semantics. The *catalog manager* keeps track of all services and mashups to facilitate the search of resources.

2.2 Service Advisor

Service Advisor is responsible for facilitating end-users to find the composition solutions relevant to their desired goals. Initially, *Service Advisor* looks up the *Service Community*, and creates several service sequences by identifying the tag-based semantics of inputs and outputs between services. It means that, according to the hierarchical tag taxonomy, once a tag contained in service operation A's outputs can be semantically associated with another tag in service operation B's inputs, a ***tag link*** will be created between the two service operations. Then *Service Advisor* uses all tag links to create a Directed Acyclic Graph (DAG). A node in the DAG represents a web service operation and an edge represents the data flow (described by tags) between the two web service operations. When *Goal Analyzer* receives and interprets end-users goals, *Composition Planner* will perform a graph-based planning process to generate all the paths that can associate with the desired output with current state, and those possible relevant ones that might be of end-user interests. End-users are allowed to choose a path and make invocation. Coordinating with iMashup *iMashup Runtime Engine* embedded in web browser, *Service Advisor* can immediately result in the changes of *user interface*(to be described in Section 5).

3. SERVICE DESCRIPTION MODEL

Our approach firstly proposes a simple-of-use model to enable end-users to reuse and compose web services in a *"data plus code"* fashion.

We formalize the service model by attaching semantic annotations with tags. Typically, a web service functionality is specified by its interface, including the *operation*, *input messages* and *output messages*. For the end-users, they mainly focus on the data that a service can consume and produce, while not paying so much attention to the interface specifications. For example, a user might look up the *HotelSearch* service with inputs {*"Hilton", "Antwerp"*}, and attain the outputs of hotel information as {*"Groenplaats 32, Belgium 2000","Zipcode?20133", "Tel?32-3-204-1212","Fax?32-3-204-8688"*}. In our approach, we describe the input and output messages by tags. Assume that a web service has two sets of parameters, ws^i={I_1,I_2,...} for request messages (as inputs) and ws^o={O_1,O_2,...} for response messages (as outputs). Let set $\boldsymbol{P}$ be the union of input and output parameters,$ws^i \cup ws^o$. For every parameter $p \in \boldsymbol{P}$, with the derived hierarchical tag-based taxonomy T, we define a mapping function $\Gamma : P \rightarrow T$, between service parameters and tags:

$$\Gamma(p) = \begin{cases} t & \text{if exists } t \in T \\ \varnothing & \text{otherwise} \end{cases} \quad (1)$$

Equation (1) indicates that, if there exists a tag $t \in T$ semantically equivalent to parameter p, we can replace p by t in terms of composition. From our investigation, the web service parameters are usually defined in form of *verb* plus *"noun"*, such as *"PostZipcodeRequest", "GetRestaurantInfoResponse"*, etc. So we extract the nouns in the parameters and bind them to a tag. In our approach, we bind a parameter with a *Feature Tag* F in T. From the inheritance relationship defined in last section, it means that all sub-tags {t_1,t_2,...} $\prec F$ can also be bound to p.

Associated with semantics, we then describe a web service ws as a two-tuple,<T^i,T^o>, where T^i and T^o respectively represents the set of tag-based descriptions for inputs that can be consumed and outputs that can be produced. Here, we constrain that $T^o \subseteq T^i$, which ensures all inputs can be consumed by the service. Obviously, to guarantee the constraints above, for $\forall t \in T^o$, $\exists t' \in T^i$ such that $t \prec t'$.

4. END-USER SERVICE COMPOSITION

4.1 Tag-based Composition Semantics

In terms of service composition, if a web service ws_1 can produce t_1 as its output and the service ws_2 can consume t_1 or its father tag t_2 as its input, we can conclude that ws_1 and ws_2 are composable. Then the tag-based service composition problem is defined simply as the result of creating a sequence of tags, namely ***Tag Link (TL)*** in the following.

DEFINITION 4.1. *We denote* **TL** *as* <T_1^i, T_1^o>, <T_2^i,T_2^o>, ..., <T_n^i,T_n^o> *for each* <T_k^i,T_k^o> *corresponds to a web service* ws_k. *If* $\forall k$, *the composition is valid when the following two preconditions are satisfied:*

1. $\forall t \in T_{k+1}^i, \exists t' \in T_k^o$, *such that t=f(t');*
2. $\mid T_k^o \mid \geq \mid T_{k+1}^i \mid$.

The above definition indicates the data dependencies between web services: subsequent web services may use the outputs produced by preceding web services as inputs.

4.1.1 Goal-Driven Composition Model

The problem of tag-based goal driven composition can be described as that of generating several Tag Links that can produce the output satisfying the user's goal. A goal-driven planning problem for service composition can be transformed to the graph-planning model with AI planning techniques. We formally define such a problem as a planning problem as a seven-tuple:
Ψ=< $T, \varpi, S, r_0^i, g, \Theta(s), \tau$ >, where:

1. T is a set of hierarchical tags;
2. ϖ is a set of web services, $\forall ws \in \varpi$ is a two-tuple defined in last Section, in form of < T^i, T^o >;
3. S is a set of composition states,$\forall s \in S$ is a collection of tags in T;
4. $r_0^i \subseteq T$ is the initial input tags, and the initial state of the composition $s_0 = r_0^i$;
5. $g \subseteq T$ is the desired goal described by tags, and the final state of the composition $s_G \supseteq g$;
6. $\Theta(s)$ is a set of web services {ws_1,ws_2,...}, $\Theta(s) \subseteq \varpi$, such that $\forall ws_k, T_k^i \subseteq s$. It means that ws_k can be invoked in the state s by requesting tag $t_s \subseteq T_k^i$;
7. τ is the state transition function, $\tau(ws_k, s')$=s that transit the state s' to s, in condition that s=$s' \cup T_k^o$ and $ws_k \in \Theta(s')$. It means that the output tags of ws_k in $\Theta(s')$ are the subset of input tags of one web service $ws_l \notin \Theta(s')$;

DEFINITION 4.2. *Suppose that the initial input tags* r_0^i *and desired goal* g, *where* $g \subseteq T$ *and* $r_0^i \in T$. *The tag-based service composition problem is to find a finite sequence of services,* $ws_1, ws_2, ..., ws_n$ *such that:*

1. $\forall ws_k (k = 1, ..., n)$ *is a two-tuple* < T_k^i, T_k^o >;
2. ws_k *is invoked sequentially from 1 to n;*
3. $\forall ws_k$, $T_k^o \supseteq T_{k+1}^i$;
4. $g \subseteq (r_0^i \cup T_1^o \cup ... \cup T_n^o)$;
5. *the total cost* $\sum_{k=1}^{n} C(ws_k)$ *is minimized.*

4.1.2 Composition Planning Algorithms

As the size of the search space will be exponential to the size of tags, to address the intractable tag-based service composition, we propose a heuristic planning algorithm within polynomial-time complexity. When the end-user selects a tag from the tag cloud or input a keyword as the initial request r_0^i, the planning algorithm first computes the cost of achieving individual tags starting from r_0^i by conducting a *forward search*. Such a Depth-First step constructs all the possible *Tag Links* that can perform the final goal. Based on

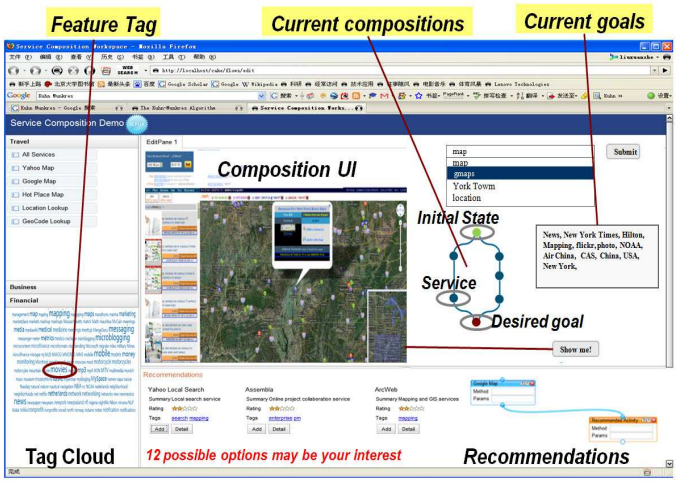

Figure 2: iMashup UI Interface

the results above, the planning algorithm then approximates the sequence of *Tag Links* that connects r_0^i to the final goal by a *regression search* step. The implementation details and complexity cost evaluation can be referred from[6].

5. USER INTERFACE DESIGN

One of the core intents of our approach is to develop a user-friendly interface that can navigate end-users to discover desired services from a large service sets, provide useful recommendations to assist them in exploring new potential relevant compositions, and simplify the composition to shield the underlying details.As shown in Figure 2, the end-users interact with the iMashup systems via a web browser. A tag cloud will be first displayed to them, indicating all the relevant functionalities in current *Service Community*. Tags with larger fonts than others, are the *Feature Tags* corresponding to concepts such as *Weather, GeoInfo, News*, etc. To begin with, the end-users can select one or more tags from the tag cloud or directly search for several tags to describe their initial requests, and the tag cloud will display all relevant tags that can be achieved from the current tags. Then the end-users go on selecting one or more tags from the tag cloud, as their desired goal. *Service Advisor* recommends a set of optional composition solutions that can constitute the goal, and display them on the screen by sorting their ranking values in descending order. Each recommended solution not only has to contain all the tags of the goal, but also may bring new tags as the potential relevant composition goals. iMashup allows the end-users themselves to iteratively refine their goals. The end-users can flexibly remove tags, or add tags from the recommended solutions or even completely new ones. Each time the goal is modified, it will result in new recommendations generated from the planning. Meanwhile, iMashup will immediately retrieve all the possible tags in *Service Community* that can reach such goal by the planning algorithm, and synchronize the tag cloud view to display only relevant tags. Note that the tag cloud is updated only when at least one composition solution can be planned to satisfy the desired goal. In this way, end-users can iteratively refine their goals until the final planned outputs satisfy their requirements.

End-users are allowed to click each alternative solution, and iMashup will display a graphical view of the its composition topological structure with the related configuration parameters. The graphical representation consists of a set of vertices and edges, where each vertex stands for the web service associated with tags and each edge means the data flow between the web services. Therefore, the users do not have to understand the underlying details of individual web services. However, iMashup allows the end-users to have a deep insight, by double-clicking the vertex, and they will be navigated to the *"Service Component View"* mode of iMashup where they can view the detailed descriptions of the services. For space limited, the design details can be referred from our previous work[**?**]. Once the users regard that a (partial) composition solution is currently satisfying, they can immediately run it by clicking *"Show me!"*.The implementation details of remote service delegation can be referred from [5]

6. CONCLUSIONS AND FUTURE WORK

End-user generated mashups have proven great potential in exploring new functionalities for meeting emergent and transient problems[1].The main contribution of this paper is to show how to make use of pervasive tags, to lower the entry barrier and simplify the development tasks for the end-users. Our work can enable the quick rollout of personalized applications. We have described tag-based service composition approaches, and discussed the identified end-user issues based on its use in interactive and rapid prototyping.

7. REFERENCES

[1] S. Adams. Keynote: The future of end user programming. In W. Schäfer, M. B. Dwyer, and V. Gruhn, editors, *Proceedings of 30th International Conference on Software Engieering*. ACM, 2008.

[2] Google. Google mashup editor.

[3] IBM. Ibm damia.

[4] X. Liu, G. Huang, P. Wen, and H. Mei. Discovering homogeneous web services community in the user-centric web environment. *IEEE Transactions on Services Computing*, 2(2):167–181, April-June 2009.

[5] X. Liu, Y. Hui, W. Sun, and H. Liang. Towards service composition based on mashup. In *IEEE SERVICE CONGRESS*, pages 332–339, 2007.

[6] X. Liu, Q. Zhao, G. Huang, Z. Jin, and H. Mei. Interactive composition for service-oriented situational applications. Technical report, Institute of Software, Peking University, PRC, 2009.

[7] M. Shevertalov and S. Mancoridis. A case study on the automatic composition of network application mashups. In C. L. Williamson, M. E. Zurko, and et al, editors, *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE 2008)*, pages 359–362, L'Aquila,Italy, September 2008.

[8] D. E. Simmen, M. Altinel, V. Markl, S. Padmanabhan, and A. Singh. Damia: Data mashups for intranet applications. In *2008 ACM SIGMOD Conference*, pages 1171–1182, JUNE 2008.

[9] Yahoo. Yahoo! pipes.