

4.1-1

Dynamic Mashup Platform for Mobile Web Applications

Seongho Cho, Hyounghick Kim, Dongshin Jung, and Hoyeon Park
Samsung Electronics Co. Ltd., South Korea

Abstract-- Mashup enables users to create a new application based on a composition of contents retrieved from external services or applications. Even though several tools were previously developed, most of existing tools were mainly designed for PC applications which generally require complex interactions with a user. Therefore, conventional platforms are not appropriate for mobile applications with limited user's input methods. In this paper, we present a new platform for generating mashup services with user's preference in mobile devices.

I. INTRODUCTION

There have been considerable improvements on Web-oriented programming technologies, such as Flex [1], JavaScript [2], Prism [3], and Silverlight [4]. Using these programming technologies, various Web services provide open APIs (Application Programming Interfaces) to enable user-generated dynamic web applications. For example, Google maps provides open APIs, such as location management, event handling, object controlling and object overlapping functions, to manipulate items on the map. These open APIs can enable a user to unify multiple useful objects from multiple sites on a single web page. These Web-based applications are called as Web mashup services.

Currently, increasing number of Web sites provide open APIs to support user customized web pages. As the number of open APIs for web applications increases, the user flexibility for dynamic Web mashup services can be widened. Some of users collect the possibility of wiring APIs of multiple on a mashup matrix site [5] to provide mashup services information. And some of sites, like Microsoft Popfly [6] and Yahoo Pipes [7], also provide users to generate their own mashup services.

For enhanced user experience, the mashup recommendation service is a crucial problem. Especially, user preference considered mashup authoring engine is the important part of successful service deployment in mobile devices with limited user's input methods. Therefore, we provide a dynamic mashup platform for mobile devices and propose an authoring algorithm with the graph theoretic approach.

II. RELATED WORK

There are some activities to collect open API information to provide possible mashup combination to users. For example, the mashup matrix [5] provides the possibility of connectable mashup services to a user. However, this matrix just provides the 2-dimensional connectivity. If a user wants to combine multiple sources, several traverses are required to identify the feasibility of mashup services.

And some of sites, like Microsoft Popfly [6] and Yahoo Pipes [7], provide a composition tool to manipulate mashup contents. Even though these sites provide mashup creation tools, a user has to select each site and wire mapping among sites. Therefore, more simple and user-friendly mashup authoring service is important especially for mobile devices.

III. ASSUMPTIONS AND PROBLEM DEFINITION

First, we assume that mashup relations are already known. Some Web site can be gracefully connected or overlaid with other site. For example, regional information, such as weather, pictures, and news can be placed on the map. However, some Web site cannot be interconnected. Especially, all of these pair wise connectivity information has been investigated in [5]. Therefore, the possible connectivity among multiple sources can be combined by pair wise information. Second, we assume that user's preference can be learned by the user usage patterns or programmed by the user-specified settings. Therefore, the authoring engine can utilize user's preference information to provide a recommendation list of mashup services.

With this assumption, we define our mashup authoring engine in this paper. Considering user's preference, an authoring engine can recommend the possible mashup lists considering possible Web service overlay. The possibility of Web service overlay is determined by the provided open APIs which can allow the results of some Web services output results to be input parameters of APIs. The mashup wiring problem is to check the possibility of connecting multiple Web services. Solving the wiring problem considering user preference is the main component of mashup authoring engine.

IV. DYNAMIC MASHUP PLATFORM AND ALGORITHM

To Provide efficient mashup services on a mobile device, we devise the following mashup platform. From user input of target services or categories, an mashup engine generates recommendations of possible mashup services. Here, a mashup wiring problem should be solved considering user's preference. After user selection for the specific mashup service, the mashup engine retrieves required Web services and generates mashup results to a user.

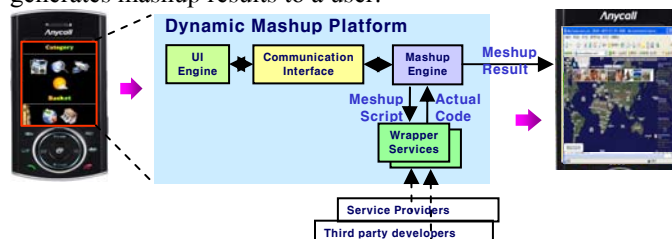


Fig. 1 Dynamic Mashup Platform

In this mashup platform, the key problem of mashup service generation is the solution for feasible recommendation for a user. To solve this mashup authoring problem, we model the mashup wiring problem after a graph model. An example of a typical mashup service wiring is shown in Figure 2. Nodes represents each Web service and edges are directed and connected when Web services can be wired over the previous Web services. Also, the weight of each edge reflects user's preference. Here, the weight is inverse-proportional to user's preference.

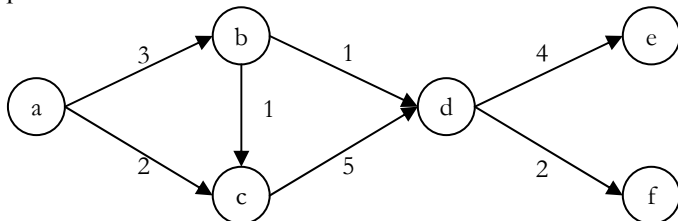


Fig. 2 An Example of Graph Representation

From the above graph model, we obtain a list of possible mashup services by solving the minimum spanning tree problem. This problem has well-known solutions, Kruskal's and Prim's algorithms [8]. Here, we modify Prim's algorithm to reflect user's category selection. Figure 3 and 4 shows the modified minimum spanning tree and relaxation algorithms. We adopt the virtual node concept into Prim's algorithm to represent the category of user selected input.

1. Initialization for the root node n_1
 - remove edges to n_1
2. Make a virtual node, v_i
 - Add edges to n_{i-1} and from v_i
 - Add edges to v_i and from n_{i+1}
3. Make a minimum spanning tree, T from the root node, n_1
4. Do Relaxation for virtual node v_i
5. Repeat 2 to 4 process from each Node n_i
6. Return the minimum spanning tree, T

Fig. 3 Modified Minimum Spanning Tree Algorithm

From the user input, the root node is selected for the mashup service. If the user requests specific categories, virtual nodes for each category are created and the edges for the virtual node and neighboring node are connected. The edge is created when the service which is included in the categories can be connected. Also, the weight for the edge is selected by the minimum weight from the original edges. After the virtual node creation, a minimum spanning tree, T , is generated from the root node, n_1 . The minimum weighted node in the same category substitutes the virtual node. To remove the cycle by the virtual node, we do the relaxation for the virtual node. And this procedure is repeated until the user requirements are satisfied.

The relaxation algorithm is shown in Figure 4. The cycle existence is checked for each virtual node using a simple cycle detection mechanism. If there exists a cycle, an edge which has the highest weight is removed from the cycle.

1. Add a minimum weighted edge, e_{ij}
2. If there is a cycle
 - Then,
 - If $w(e_{ij}) < w(l_{ij})$
 - Select an edge with a maximum weight on the link l_{ij} , remove it, goto 1
 - Else, drop e_{ij}
 - Else, goto 1

Fig. 4 Relaxation Algorithm

V. IMPLEMENTATION

With an assumption of pre-acquired user's preference, we implement a mashup platform to demonstrate the possibility of mashup services on the mobile phone. We devise several services, such as Google Maps [9], Yahoo Maps [10], Yahoo Weather [11], and a local PIMS (Personal Information Management System) in a device. During the authoring phase, a user selects his/her preferable services, such as Google maps and Yahoo weather. Or, a user can select categories to connect the services. Then the authoring engine shows the possible list of mashup services considering user's preference. Figure 5 shows the screen shot of the authoring procedure of our implementation on a mobile phone. In this case, a user selects three categories, maps, PIMS and weather, then the final result is shown with a mashup of Google Maps and Yahoo Weather.



Fig. 5 Mobile Screen Capture of Implemented Mashup Service

VI. CONCLUSION

In this paper, we propose a dynamic mashup platform for mobile devices. Especially, we model the mashup service after the graph and provide possible combinations of Web services considering user's preference. Also, we implement the dynamic mashup application on the mobile device to show the feasibility of our proposed dynamic mashup platform.

REFERENCES

- [1] Adobe Flex, <http://www.adobe.com/products/flex>, online link.
- [2] JavaScript, <http://www.javascript.com>, online link.
- [3] Prism, <http://wiki.mozilla.org/Prism>, online link.
- [4] Microsoft Silverlight, <http://silverlight.net>, online link.
- [5] Mashup matrix, <http://www.programmableweb.com/matrix>, online link.
- [6] Microsoft Popfly, <http://www.popfly.com>, online link.
- [7] Yahoo Pipes, <http://pipes.yahoo.com>, online link.
- [8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, Introduction to Algorithms, Second Edition, the MIT press.
- [9] Google Maps, <http://maps.google.com>, online link.
- [10] Yahoo Maps, <http://maps.yahoo.com>, online link.
- [11] Yahoo Weather, <http://weather.yahoo.com>, online link.