INNOVATIVE MASHUP DEVELOPMENT: UTILIZING VISUAL INFORMATION ABOUT APPLICATION PROGRAMMING INTERFACES

Zoltán Harsányi, Martin Repta and Viera Rozinajová

Slovak University of Technology, Faculty of Informatics and Information Technologies,
Ilkovičova 3, 842 16 Bratislava, Slovakia

Abstract: In this paper we deal with design and creation of web mashups which represent one of the important Web 2.0 application approaches. The main goal of this work is to describe an approach where a recommendation about existing and cooperating data sources is made to the user, to analyze various existing methods and tools for creating mashups, and to investigate possibilities of automating the process of their creation. Following this analysis, the design of a new tool for creating mashups is described, using the recommendation tool for suggesting APIs to the user. Although it has the basic features of existing tools, it offers some useful extra features, especially for the users with limited or none IT skills. We also describe how we experimented using these tools.

Keywords: Web 2.0, mashup, application programming interface, mashup development, recommendation

1 INTRODUCTION

Internet and related technologies have created interconnected world, in which we can easily share data, work together on different tasks, and create communities among the users based on common interests.

After introducing some new useful features of Web, paradigm called Web 2.0 arose. Web 2.0 brought a new collection of technologies, business strategies and social trends. The most significant technologies which came with Web 2.0 were blogs, wikis, tags, clouds, Ajax, RSS, etc. [3].

This paper focuses on a new concept of Web 2.0 called mashup. It means a way or method of creating Web sites or Web applications that combines information and services from multiple sources on Web. Web mashups combine content, presentation and application logic from different Web sources with aim of creating a new Web application or Web service [4]. It is easier and much quicker to create a mashup application than to code a new one using modules and libraries in a traditional way. This capability is one of Web 2.0's most important and valuable features [3]. Web mashups do not represent a technological revolution, but they are likely to play a big role in shaping the future of the Web.

A typical mashup application is for example HousingMaps (*http://www.housingmaps.com*) which gets sales and rental information about houses, flats, estates, etc. from a classified Web site called Craiglist and displays this information on maps, got from Google Maps. It is much more interactive then just watching a raw list of advertisements [6].

This article presents results of our research project concerned in searching new ways of mashup development support. Based on the visualization of relationships among application programming interfaces we recommend the suitable sources for mashup creation to the user. We have designed a tool for mashup creation which utilizes the proposed recommendation and at the same time it facilitates the easy way of mashup development. The paper is organized as follows. Section 2 describes preliminary notions and related works. Section 3 introduces our ideas concerning recommendation of the suitable application programming interfaces and presents a way of visualization of their relationships. The proposed tool for mashup creation is described in section 4. Experiments and their evaluation are included in the fifth section and section 6 concludes the paper.

2 RELATED WORKS

Due to the popularity of Web 2.0 technologies significant endeavors were enrolled in the area of searching for new methods and tools facilitating easier mashup development. These efforts can be recorded in academic as well in commercial environments. In this section we will try to follow up both directions: we outline some of the academic research results and describe also the most popular mashup development tools and other tools for facilitating easier mashup development, pros and cons of the tools and their main features.

Among research approaches a project called Damia [7] is designed for a large and changing number of data sources that make available both structured and unstructured information, and to enable users to complete complex integration tasks. Damia is focused on the data problem presented by situational applications. It uses a data flow model to represent a mashup, leaving a control flow aspects to the application.

Marmite [8] is an end-user programming tool for creating web-based mashups. Users can choose from a number of different activities for extracting data from Web sites and local or remote databases. Activities are organized in simple sequential data flow. Marmite provides run-time views of the data as they flow from one activity to the next.

Project Mashlight [9] is aimed to a lightweight service composition. Users can easily create their mashups using prepared widgets. Widgets allow to access heterogeneous services on the web. The framework provides a well defined conceptual model that defines both the notion of composable widget, and how to aggregate them through appropriate control and data flows. Mashlight provides a set of design-time tools for creating widgets and process mashups.

As our approach is based on recommendation of suitable application programming interface (API), now we briefly mention the systems offering some sort of recommendation:

Recommendation in mashup development has been analyzed in project called MashupAdvisior [5]. MashupAdvisior provides design-time assistance to mashup developers. To generate recommendations the MashupAdvisior calculates co-occurrence patterns between inputs which have been added by creator and outputs of data sources in own repository. Similarity between used and suggested APIs is based on domain independent thesaurus and domain dependent ontologies. This principle strongly depends on semantic similarity of names of inputs and outputs.

In recent years, many industrial players created several tools and frameworks. One of the most popular projects is Yahoo Pipes (*http://pipes.yahoo.com/*) which offers a graphical interface to arrange different machine-accessible data sources in a graph, where nodes represent data sources and arcs model the data flow. Yahoo Pipes provides a pure visual drag-and-drop Ajax editor. However, it allows combining only data sources like RSS, Atom or XML.

Current situation in the area of mashup development tools is not very favorable for end-users without IT knowledge. There is a lack of tools, which can be used for creating basic mashups or personalized sites without the need of using different and sometimes advanced types of tools. Moreover these tools are not suitable for processing different types of Web APIs or are not able to link different APIs together to form personalized or specialized end-user site or Web application. These aspects motivated us to come up with new approach in this area.

3 PROVIDING USEFUL INFORMATION FOR MASHUP DEVELOPMENT

3.1 Application programming interface

Mashups are generally created using application programming interfaces. API can be described as an application interface, which allows user to interact with data, services and resources of the application from outside. In essence, a program's API defines the proper way for a developer to request services from that program. It may be considered as a set of instructions and standards that govern the way an application operates. Many companies make APIs available to the public – thus allowing other software developers to design new products that incorporate its service.

Web APIs are typically defined as a group of messages sent via HTTP along with a well defined structure of responses [1]. They are most often expressed through XML or JSON notation.

In connection with mashup development, APIs are very important. They are the key parts which form mashup application.

The number of data providers and publicly available API (data sources) constantly increases. Portal *http://programmableweb.com* gathers information about existing APIs and also about completed mashup applications. Nowadays (April 2011) there are described more than 3000 API and more than 5700 mashups. This may seem to

the users forming mashup applications as a very beneficial fact. On the other hand, for non programmers and common PC users, the increasing number of data sources can be confusing. We have analyzed these sources in order to get an overview of them and to find out some of their useful features.

3.2 Suggestion of the suitable sources

The main purpose of this work is to describe an approach where a recommendation about existing and cooperating data sources is made to the user. System builds a network of cooperating APIs. The relationships among data sources are visualized as a planar graph. The recommendations can be integrated into existing system through our web service which returns relationships in JSON format. We created a mashup framework integrating this service, where end users can easily create new mashups and find new ideas in our graph.

Our idea is to take advantage of existing mashup applications and their used APIs from portal programmableweb.com. Each API has many information attributes such as name, category, users rating, supported data formats, protocols and tags. We used these attributes together with information about created mashups. Each described mashup application has information about used API. We get these data and store in the local database.

While searching in our dataset of data sources and based on our statistical computing we found out that only approximately 30% of described APIs were used in existing mashups. Then, we discovered that only about 15% of APIs were used more than three times. People create mainly small mashups (in 90% consisting from 1 or 2 API) because often they do not know, which other API could be used with their selected API. We found out that there are several APIs which are used constantly and are very popular.

3.3 Process of recommendation

Recommendation systems are built on the principle of personalized information filtering and predicting whether the predicted product (article, movie) would be evaluated as a good one by a specific user (prediction problem). Often the whole set of elements that might be of interest to a specific user is predicted (Top-N recommendation problem). In recent years various approaches for recommendation have been introduced. Among them, collaborative filtering (CF) is probably the most successful and the most common form of recommendation [2].

Our tool uses collaborative recommendation based on existing mashup dataset. Our recommendation is partly model-based and partly based on the element.

Model-based collaborative filtering analyzes historical information to identify relations between different items - such as: purchase of an item (or a set of items) often leads to purchase of another item (or a set of items), and then uses these relations to determine the recommended items. These systems are usually very fast, but tend to require significant amount of time to build a model [2].

Item-based collaborative filtering has been developed to address the scalability and performance problems in concern of user-based recommendation algorithm.

3.4 Our approach

Algorithm of computing recommendation has several phases. In the first phase we analyze the similarities between different elements. In the second phase we use this similarity to discover a set of recommended elements.

Original dataset contains also mashups which consist of only one API. We do not take into consideration these mashups in our model, as there are not any relations between data sources. After removing these mashups we have got model with 2251 mashups and 653 unique data sources. Our model can be described as binary matrix M where the rows are formed by particular mashups and columns are filled by particular APIs. It means we have matrix M with dimensions 653x2251. If mashup on row r is using API in column c the value of M_{rc} will be 1.

Algorithm has two different variants. The first variant uses model described above. The second variant uses model built from mashups with only two APIs. This model is smaller than the first one (653x1221). Basic rule for recommendation in both models is the fact that API "*a"* and API "*b"* were used together in a certain mashup. This can be considered as a basic assumption for the feasibility of future cooperation of these two APIs. The algorithm traverses whole model to find all data sources that were previously used with API "*a"*. When we get this dataset we apply second rule. The second rule is the level of minimal usage of this couple in the model. In practice when couple "*a"*, "*b"* was used in several mashups, there is higher probability that they can cooperate together. The third rule is about common output format and used protocol. Recommended data sources must work on common protocol and support common data format. When performing experiments we have got different results for precision and recall. While system based on the first model has precision only about 70%, for the second model (with two APIs in mashup) we got precision near 100%.

4 MASHUP DEVELOPMENT TOOL UTILIZING THE GRAPH OF API RELATIONS

4.1 Preliminary requirements for the tool

We have decided to design and create a general mashup development tool which could be used by end-user who is not Web developer but he wants to create some specific or personalized Web application (in this case mashup). We have in mind the exploitation of graphical information obtained by analysis of the available APIs. As we have mentioned in the previous sections, the number of accessible sources constantly grows and it is quite difficult for IT non-expert to cope with them. Therefore our tool is based on the recommendations described in section 3. Likewise the aim is to provide the user with an environment, where he can easily create his application, i.e. to offer very friendly and unexacting environment for building

mashups. Our target group is formed by the users who are not familiar with the advanced web application technologies. Therefore our main requirement was to create an environment which is easy to operate. We have stated some conditions that we have considered important when designing tool for non experienced users. Table 1 describes the main features of our development tool.

	Feature or parameter	Goal
1	type of tool	stand-alone application
2	platform	Web
3	installation	without installation of plug-ins
4	terms of usage	free license with registration request
5	target user group	end-users without the need of Web development skills or practices
6	development environment	interactive GUI for adding, combining and operating different Web sources
7	handling APIs	Web APIs are processed by the tool, users see the graphical equivalent of these APIs
8	overall impressions	allowing users to selects APIs interactively and easily, placing them on the defined layout, setting different parameters and thus creating mashups
9	purpose of tool	creating simple and general, personalized Web applications

Table 1: Preliminary features of our mashup development tool

Let's describe the defined criterias:

The tool must be a stand-alone application because end-users do not want to use frameworks or libraries which must be integrated into a single project.

Web is the most ideal platform for developing Web applications from Web sources. Mashups can be easily deployed, developed and viewed by users. Users do not want to be bothered by searching for and installing plug-ins or other equipments. That's the reason why we decided not to use any plug-ins. Furthermore using plug-ins limits the users in choosing their Web browser, as not all of them can use plug-ins.

The reason why we have chosen end-users without IT skills is that there are several tools, frameworks and libraries for creating mashups by developers, but only a few for ordinary people, – furthermore none of them is easy to use for non IT expert.

The development environment must be interactive, attractive and simple enough to attract people.

Our goal is to hide the back-end processes enough and keep the development environment as easy as possible.

4.2 Design of the tool

In this section we describe our approach to fulfill the defined criterias in our mashup development tool. We also bring in the motivation of our solutions.

Architecture of the tool is client – server. Server is responsible for handling user requests and communicating with Web APIs. Client is responsible for collecting user requests and displaying the outputs from the server. The tool uses a relational database layer for storing user specific data, mashup and API specific data. It is run by Web server. In the next section we describe the server side of the tool in more detail.

The tool is designed using architectonical style MVC. It is implemented using object-oriented concepts. Each MVC layer consists of the following parts:

- controllers: layer consists of classes for handling user actions and responding on them (controller classes) and of special classes, which are used for mashup API handling and linking (component classes).
- models: this layer consists of a database layer and classes for working with each table of the database (retrieving data, storing and validating data, linking tables, etc.).
- views: this layer consists of classes responsible for generating the results of controller and component calls, it also consists of special classes (called helpers) which are responsible for generating specific type of output, i.e. rendering the output of each API and linking them.

The client side of the tool consists of 2 separate subsystems:

- Development module: responsible for developing mashups, the main part of the tool,
- Administrator part: responsible for user management, API management and for managing mashups.

The development environment of the tool is formed by 6 parts:

- menu bar: displays the basic operations with the tool,
- APIs list: displays the available mashup APIs for choosing and operating with them,
- mashups tab bar: displays created mashups of logged user,
- workspace: place, where users operates with APIs using components for their representation,
- mashup source: the source code of the generated mashup, mashup preview: preview of the mashup, how it looks like when it is deployed.

Users can create so called sections which are the containers for displaying the outputs of each APIs. These sections are created using development environment and users can define different parameters for them which are evaluated as HTML attributes in the final mashup.

After defining these sections, users can add mashup APIs into these sections. Mashups can be created in two ways:

- defining more sections with single mashup API inside them (using this approach users can create a kind of customized Web site which contains all of users favorite data sources)
- defining less sections with more mashup APIs inside them (this technique is used for linking the outputs of mashup APIs to create enriched data source or to get data from one API based on the inputs of another, i.e. displaying data from Web feed in Google Maps)

Our tool links mashup APIs based on these positions, e.g. if one section contains more than one API, the outputs of the APIs are chained. APIs can also be reordered and customized. Furthermore users can link two or more APIs into a single one. This can also be defined in mashup API's settings.

During the mashup creation process, users can use the recommendation tool for selecting and adding APIs into their mashups. The recommendation tool can be used in two different ways:

- Using the recommendation system in separate browser window or tab, hence using its graph and other outputs about the searched APIs.
- Using the development tool's recommendation graph in API's settings form. This graph is generated from the outputs of Web service from the recommendation tool. This graph does not provide all the features of the recommendation tool's graph, the purpose here is just to visualize the recommendation of choosing the API for linking. It means that the graph in the recommendation tool offers more information, e.g. by clicking on nodes of graph we can take a look at the API's details.
- One of the most significant limitations of the tool is that users can only work with APIs, which are allowed by system administrators of the tool. Furthermore, these APIs must be processed on the side of server, which can only be reached when the server has the necessary API classes on the sides of controller and view. To sum up, users can work only with APIs which have their API handlers on the side of server. This is due to different data structures or application logic of Web APIs. Users on the other hand have guaranty that the APIs they see are fully functional.

5 EVALUATION

Evaluation of the tool has been accomplished from different points of view.

From the functional side, the evaluation answers the question whether the system processes the APIs as it should. This functionality was verified by validating single components for handling APIs. Each API has its own handler component, designed for that specific API using its documentation. The capabilities of components were then evaluated using different parameters and analyzing the outputs of these components.

Components were also evaluated from the side of abilities of linking the generated results. This kind of evaluation method was quite tricky, because the correctness of the linking process could not be reached only by putting the outputs together, but the semantics of the output should also be considered. It means that if we use in search API (i.e. Wikipedia API) an input data from some kind of feed, we do not always get the result as we want. Evaluating the output of these APIs from the semantic point of view is a very difficult process. This kind of data evaluation can be one of the future directions, how to improve the tool.

The tools for API recommendation graph (described in chapter 4.2) have also been verified. The verification process consisted of comparing the graph of development tool with the graph of recommendation tool. The final graph of the development tool

must contain the same nodes for the same parameters. The evaluation proved the correctness of the given recommendation process.

The second type of evaluation consists of comparing the tool with other existing tools from the side of overall power. It means trying to create the same type or the same mashup and check the features and methods which are available by each of the tools, for example creating a mashup like HousingMaps. We tried to "simulate" mashups which were created by manual coding in our tool. The exact copy of these real mashups cannot be reached using our tool for two main reasons:

- manually coded mashups usually contain custom JavaScript and Ajax code for advanced data manipulation and interactivity, meanwhile mashups created by our tool do not
- manually coded mashups have their own, site specific design, specific using of CSS styles, customized images and other design technics.

These two kinds of technologies, used in Web development, however require development skills from the user and our approach is to let non-IT users create mashups without the need of knowledge of these kinds of advanced technics.

This verification initiates another possible direction of improving the tool: one possible upgrade is to offer the mashup design customization to the user. The "hand-written" mashups are customized from the design point of view, but end-users with no IT experience do not have necessary skills, therefore automated graphical components could be used to improve and customize the design of created mashup.

6 CONCLUSION

In this paper we have presented one way of supporting mashup development. Our approach is based on visualized information about application programming interface and recommendation of the suitable sources for the designed mashup. We have also developed a tool for creating mashup, which is intended for the users with no experience in creating web applications.

As far as the type of API recommendation is concerned, our approach does not depend neither on tags, nor on semantic similarity of used inputs. We aggregate APIs from mashups that have been already created and we select only APIs which match rules described in our concept. If API matches all rules we add it to the list of recommended APIs. The precision of recommended API is very high, because similar combinations have been already used by different mashup creators. In comparison with the approach described in [5] where the system can suggest undiscovered relations which have not been used yet, our system has an advantage of independence on tags, as they can suffer from inconsistencies, typos, proliferation of synonyms and unstructured organization. Due to the drawbacks caused by the fact, that the tags are maintained by people, the systems based on tag creation can be in some sense unreliable. This is the main difference, where our approach can bring better results.

We have designed and created a tool for creating mashup applications which is based on the described API recommendation. The system provides really user-friendly environment which means the tool can be used by non-IT people - our target

user group. Mashup development environments, which we have analyzed beforehand, have either limited capabilities from the side of handling mashup APIs, or they are used for creating enterprise applications. Despite of the limitations of our solution, the designed tool can be treated as a fully functional mashup development tool.

ACKNOWLEDGEMENT

This publication is the partial result of the project "Adaptive Social Web and its Services for Information Retrieval" of the Scientific Grant Agency of Slovak Republic, grant No. VG1/0508/09, the Research & Development Operational Programme for the project "Research of methods for acquisition, analysis and personalized conveying of information and knowledge", ITMS 26240220039, co-funded by the ERDF, and the project "Cognitive Travelling across Digital World of Web and Libraries with the Support of personalized Services and Networks" of the Slovak Research and Development Agency under the contract No. APVV-0208-10.

7 REFERENCES

1. Orenstein, D.: QuickStudy: Application Programming Interface (API). In: Computerworld, vol.10 (2000)
2. Deshpande, M., Karypis, G.: Item-Based Top-N Recommendation Algorithms. In: ACM Transactions on Information Systems, vol. 22, pp. 143--177 (2004)
3. Murugesan, S: Understanding Web 2.0. University of Western Sydney, Australia, http://www.computer.org/portal/web/buildyourcareer/fa009 (2007)
4. Platt, M.: Web 2.0 in the Enterprise, MSDN Architecture Center, The architectural journal vol. 12, (2007)
5. Elmeleegy, H., Ivan, A., Akkiraju, R., Goodwin, R.: MashupAdvisor: A Recommendation Tool for Mashup Development. In: Proc. of 6th Int. Conference on Web Services, 2008, pages 337–344
6. Yu, J.; Benatallah, B.; Casati, F.; Daniel, F.: Understanding Mashup Development. IEEE Internet Computing, vol. 12 pp. 44--52 (2008)
7. Simmen, D. E., Altinel, M., Markl, V., Padmanabhan, S., Singh, A.: Damia: data mashups for intranet applications. Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 1171--1182 (2008)
8. Wong, J., Hong, J.: Making mashups with marmite: towards end-user programming for the web. In Proceedings of the SIGCHI conference on Human factors in computing systems, pp 1435--1444. ACM New York, NY, USA (2007)
9. Baresi, L., Guinea, S.: Consumer Mashups with Mashlight. In ServiceWave'10, pp 112--123 (2010)
10. Bartalos, P.: Effective automatic dynamic semantic web service composition. In Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 3, No. 1, 2011