Journal of Theoretical and Applied Information Technology
31st December 2011. Vol. 34 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE DESIGN OF BLOCK-BASED MASHUP TOOL FOR END-USERS MASHUP APPLICATIONS DEVELOPMENT

[1]RODZIAH LATIH, [2]AHMED PATEL, [3]ABDULLAH MOHD. ZIN
Center for Software Technology and Management,
Faculty of Information Science and Technology,
Universiti Kebangsaan Malaysia, 43600Bangi, Selangor, Malaysia
E-mail: [1]rodziah@ftsm.ukm.my , [2]apatel@ftsm.ukm.my , [3]amz@ftsm.ukm.my

ABSTRACT

A mashup application is a web application that combines contents from several sources into an integrated web experience A mashup tool is a software tool to assist users in developing web mashup application. Most of these tools are developed by employing end-users development approaches such as scripting, wire, widget, spreadsheet, and Programming by Demonstration. However, although these tools are based on end-users development approaches, they are still difficult to be used by most of end-users since they require end-users to have some programming background. In this paper, we present the design of Whip mashup tool. The aim of this mashup tool is to allow end-users to develop web mashup application easily by using Block-Based Software Development approach. This development approach enables end-users to develop software or web applications by combining several programming blocks together.

Keywords: *Web Mashup, Block-Based Software Development, End User Development, Web Services Aggregation*

1. INTRODUCTION

Over the past few years, mashup applications have received a significant attention as one of the most supportive web applications. Users develop mashup meta-application to track events like hurricane, crimes, stock market, etc. There are three main reasons for this new trend. The first one is the availability of tools that have enabled end-users to develop mashup applications. The second reason is the availability of Web technologies that support this activity. The third reason is the interest and acceptance of mashup application development by enterprise users. Organizations are beginning to realize that they can use mashup to coordinate their services with other existing services, either internal or external, as well as to provide new and interesting views of the data.

A mashup application is a web application that combines contents from several sources into an integrated web experience [23]. A mashup application aggregates multiple services with each services serving its own purposes into a new service that serve a new purpose [10]. For example, Kangaroo (*http://groups.csail.mit.edu/uid/kangaroo/*) is a mashup application that extends the capability of webmail systems (such as Gmail). This application helps to alleviate problems of sending e-mails to wrong recipients by automatically displaying pictures of the selected recipients while the user is composing an e-mail message. Pictures are taken from Google images and Facebook (*www.facebook.com*). This application can be a useful security feature for webmail systems that helps to prevent people from sending wrong information to the wrong recipient.

Mashup tools are software tools that assist users in developing mashup applications. There are a number of mashup tools currently in the market, for example, Yahoo Pipes (*http://pipes.yahoo.com*), Dapper (*http://open.dapper.net*), Intel Mashmaker [4], IBM Mashup Center (*www-01.ibm.com/software/info/mashup-center/*), Marmite [22], and Vegemite [9]. Different mashup tools targeted at different types of users; developer or non-developer. Developer is a group of users that have a skill in programming. Non-developers are end users that do not have programming skill however they may have knowledge about computers and Internet.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. PROBLEM STATEMENT AND PROPOSED SOLUTION

Studies by [14-15] show that although most of mashup tools are developed to support end user development, they are not easy to be used by end users since they require end users to have a knowledge of computer programming. In order to solve this problem, we propose a new mashup tool called Whip that is based on Block-based Software Development (BBSD) approach. In this approach, end users can develop software applications by combining several programming blocks together. BBSD is a simple concept that allows end users to develop application without computer programming background.

3. METHOD

The design of Whip presented in this paper is carried out in five stages as follows: study of the currently available mashup tools, understanding the concept of block-based programming approach, analyze the requirement specification, develop the system architecture, and finally present the system design.

4. REVIEW OF MASHUP TOOLS

There are two approaches for categorizing mashup tools. The first approach is to categorize them based on types of supports provided while the second approach is to categorize based on types of environment provided.

4.1 Types of Support

From this aspect, mashup tools can be categorized into automated, semi automated and scripting.

4.1.1 Automated Mashup Tools

This category of mashup tools is normally support the development of situational mashup applications. A situational application is an application that is developed rapidly to address an immediate need of an individual or small community [3]. It is created for a specific situation and it is utilized only for short periods of time while the situation exists. It is a Just-in-time solution but not necessarily short-lived.

A mashup application is a type of situational application if it is developed for a specific situation [23]. For example, HomePriceRecords (*homepricerecords.com*) is a mashup application that combines home sales data with Google maps. This mashup application lets users check the price of the house for sale in the United States. 2RealEstateAuction (*www.2realestateauctions.com*) is a mashup application that let users see all real estates (residential, land, commercial and timeshares properties) in the United States that are currently being auctioned in eBay (*www.ebay.com*) by using Google maps.

Examples of mashup tool that support the development of situational mashup applications are MaxMash [18] and Automatic Mashup of Composite Application [2]. MaxMash composes selected features of networked application and generates the source code for mashups application that can integrate those features. On the other hand, Automatic Mashup of Composite Application is a framework that supports automatic creation of mashup which allows for composition of non web service based components such as portlets, web applications, native widgets, legacy systems, and Java Beans. Users' situation can be configured depend on user location and schedule [5], and it will help users to select the web source easily and quickly.

4.1.2 Semi-Automated Mashup Tools

Semi-automated mashup tools were developed to support end user development of mashup applications. Thus, more of mashup tools in this category were developed based on end-user development (EUD) paradigms. EUD paradigms provide programming capabilities for everyone by pushing on different aspects of computing technologies [13]. There are four EUD paradigms identified used in mashup tools as shown in Table 1: wiring, Programming By Demonstration (PBD), spreadsheet and widget.

Wiring or dataflow paradigm is where several selected modules or widgets that support particular functions (i.e. data retrieval, data presentation, etc.) are connect together. Such tools are Pipes and Marmite.

PBD is a EUD paradigm that let the users demonstrate the desired task going through several steps as actions that should be performed on the data. The system records these actions and concludes a generalized program that can be used upon new data. For example, in Vegemite users start with an empty table. Users demonstrate a series of actions on how to fill the spreadsheet-like table manually or copied from an existing source or

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

extracted from an existing web page using Vegemite direct-manipulation tool. These actions are recorded into scripts, which can be re-executed immediately for other rows in the table and used later to refresh the data in the table.

Spreadsheet paradigm is also another EUD paradigm that implemented in several mashup tools like Spreadsheet-based Web Mashups [8], Vegemite, Marmite, Mashmaker, and SpiderCharlotte [20]. In spreadsheet paradigm, a spreadsheet-like table is used to display the data. Cell in the table is referring by its column and row coordinates, and users manipulate their data directly using drag-and-drop fashion. For instance, Spreadsheet-based Web Mashups use spreadsheet-like table to display the query result from the Web. Users then can specify the constructing data views. Vegemite also use a spreadsheet-like table called VegeTable to display all the data accessed from the web.

Widget paradigm is a paradigm that has gained wide popularity because it allows end users to create mashup easily. Each widget represents a particular function. User just needs to select the widget and use it. However, this approach has several limitations such as the number of widgets increases to support more operations, locating the right widget for the task can be confusing and time consuming. On the other hand, the advantage of using widget is that it can promote customization where users can tailor their systems to match their personal work practice or preference [11]. Such tools that use widget paradigm are iGoogle and Netvibes.

4.1.3 Scripting Mashup Tools

This category of tools implies that the development of mashup applications requires some form of programming or coding. The coding process is simplified by using scripting languages rather than the normal programming languages. However, non-programmer end users will find that this approach is still difficult because it requires users to know basic program structure and the syntax of the language. An example of a mashup tool in this category is Web Mashup Scripting Language (WMSL) [17].

Table 1: EUD paradigms used in mashup tools.

EUP paradigm	Description	Mashup tool
Wiring paradigm	Wire together selected widgets.	Pipes
PBD	Users demonstrate desired task to be repeated by the tool.	Vegemite
Spreadsheet paradigm	Data is inserted into a spreadsheet-like table.	Spreadsheet-based Web Mashups
Widget paradigm	Each widget contains different data.	iGoogle, Netvibes

4.2 Types of Environment

In general, as shown in Table 2, there are three types of environment of mashup tools; visual mashup editor, browser extension application, and web portal mashup tools.

A visual mashup editor is a tool that lets users create a mashup application by manipulating program elements graphically rather than by specifying them textually. Yahoo Pipes is an example of a mashup tool that let users to develop mashup applications via a visual editor by selecting a few modules and wire them together. Yahoo Pipes editor consist of three panes; the library, canvas and debugger. The library pane shows a list of all the available modules and user's favorite Pipes from other users. Modules are grouped by functionalities; data source (like RSS Feeds, Yahoo Search, etc.), user's inputs field that can be filled in at runtime, operators (like sort, count and filter), URL (for building and manipulating URLs), strings (for handing strings data type) and date (for manipulating dates).

Table 2: Three Types of Environment of Mashup tools

Types	Description	Mashup Tools
Visual editor	Editor with visual programming environment	Pipes
Browser extension	Plug-in application to web browser	Mashmaker, Marmite, useKit,Vegemite
Mashup portal	Web portal with a dashboard full of mashup widget	iGoogle, Netvibes

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A browser extension application mashup tool is a tool that is developed as a plug-in application to web browsers. By using this tool, users can develop a mashup while browsing the web. For example, Intel Mashmaker, an application that is a plug-in to Firefox web browser. Mashmaker let users develops mashup applications while browsing the web. It is equipped with a capability to suggest to users on what type of data that need to be integrated. Other browser extension mashup tools are useKit [16], Marmite and Vegemite.

Web portal mashup tools like iGoogle (*www.google.com/ig*), MyYahoo (*my.yahoo.com*), Windows Live (*www.live.com*), Netvibes (*www.netvibes.com*), PageFlakes (*www.pageflakes.com*) and Facebook are simple tools that provide users with a dashboard full of widgets ready to be mashed. Users can stream their emails from Gmail, get local and world weather information, and pull RSS from their favorite sites. In this type of mashup tool, data sources are aggregate into a single layout. However some of these data sources may themselves consist of multiple data sources integrated into a single list or view.

iGoogle allows users to add to their iGoogle page any number of widgets or RSS feeds that contain data from other sources and arrange these widgets by dragging them from one location on their page and dropping them onto another location. Each widget has its own settings that can control the amount of data gets displayed and the source of that data. Users can create their own widget and share any widget on their page with other users. However, iGoogle only support data sources from Google. Netvibes is one of the early AJAX based personal content aggregators. Netvibes allows users to create their own website within the Netvibe domain and customize both its content and appearance. Each widget has its own settings that can control its appearance, how much data it displays and where that data comes from. Netvibes also allow users to share with others their Netvibes pages.

5. BLOCK-BASED SOFTWARE DEVELOPMENT (BBSD) APPROACH

BBSD is an approach that allows users to develop application by combine several program blocks together [12] [1] [6]. The applications can be developed quickly and within estimated cost. The main aim of this approach is to allow the end users that lack of programming skill to develop the application by simply combining several selected blocks. BBSD approach adopts the idea of component-based development (CBD) approach where it simplifies the process of CBD to make it suitable for the end users.

An individual block is defined as a program component, a software package, or a module that support certain task or function. Unlike components in CBD, block is a single level of program component where a block cannot be a component of other blocks. However, blocks can be reused. Users are also allowed to customize the blocks before integrate those blocks to form a required application. The elements of BBSD approach are interfaces, attributes, behaviors and GUI elements [1] (Table 3). Interface is like a communication contract between the blocks. Attribute is a characteristic of the block. Behavior is a set of functions for the block to perform. GUI elements are features for building graphical user interfaces i.e. buttons, sliders, etc.

Table 3: Elements of BBSD Approach.

Elements	Description
Interface	A communication contract between the blocks.
Attributes	The characteristics of the block.
Behaviors	A set of functions for the block to perform.
GUI element	A group of features for building graphical user interfaces i.e. buttons, sliders, etc.

There are two types of developers in BBSD approach; block developer and application developer. Blocks developer is a people that developed the blocks. Application developer will use these blocks to develop the application. Therefore, in BBSD approach, end users also can be an application developer. The development processes of block-based applications are separated from development processes of the blocks. The blocks should already been developed and possibly used in other applications when the application development processes start. Figure 1 illustrate the development process model of BBSD approach [1].

Journal of Theoretical and Applied Information Technology
31st December 2011. Vol. 34 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

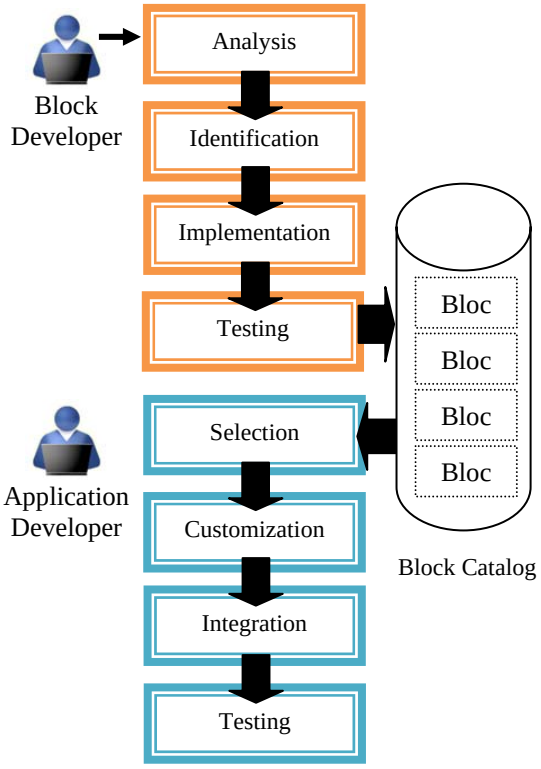

Figure 1: BBSD Approach Model.

Figure 2 shows the comparison between traditional approach in mashup development with BBSD approach. In traditional approach, developing mashup involve five processes which are data retrieval, data cleaning, data modeling, data integration and data visualization [19, 23]. While using BBSD approach, these processes will be simplified where users just need to select the block from the block catalog, customized it if needed, integrate the blocks and display. Blocks integration is also an option where user can integrate the block with other block if required. For example, user can integrate the 'searching' block with 'RSS feed' block or integrate photo from Flickr block with Google map block or just let those blocks function separately.

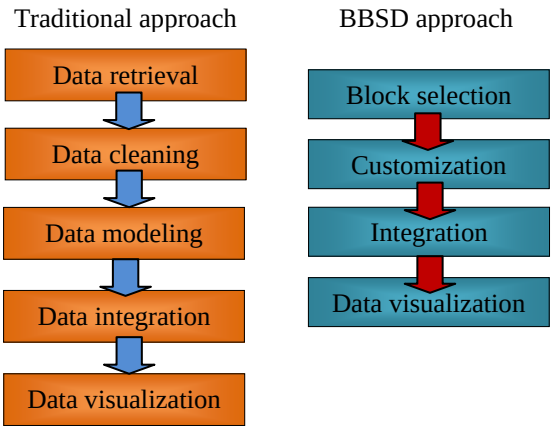

Figure 2: Comparison of traditional approach in developing mashup with BBSD approach.

The implementation of BBSD approach offer advantages such are:

- Existing blocks are made possible to be reused.
- Application can be developed adaptive to individual or specialized user.
- BBD approach allows for rapid software development and therefore decrease the overall development cost.
- Programming skills are not required.
- Application that developed using BBSD approach is flexible and extensible.
- Software development tool that support BBSD approach can be a general purpose software development tool and thus many types of applications can be developed.

6. REQUIREMENT SPECIFICATION OF WHIP

Whip is a mashup tool that is based on Block-based Software Development (BBSD) approach. The design of Whip is guided by the requirements as shown in Table 4.

7. WHIP ARCHITECTURE

Figure 3 shows the architecture of Whip. Whip is a client-side application. Upon request, Whip application is sent to the user's computer by the web server. The user's web browser executes the script and display the output. The data sources composition to make a mashup application is taken place at the client side. The data source can be accessed from internal and external sources.

Whip also supports web services access from multi-platform like Google, Yahoo, etc. Web

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

services like Google search, Yahoo search, Google map, Yahoo map, Google News, etc. are available as predefined blocks in blocks catalog.

8. DESIGN OF WHIP

The design of Whip is done in two stages: the design of Whip mashup tools and the design of blocks. The design of Whip mashup tools involves the use case analysis, interaction analysis and user interface design.

8.1 Use Case Diagram

Basic functionalities that required by Whip are:

- Create the website with default blocks. A new user can create a website with default blocks like Google search, Youtube channel, Map Location, CNN News, etc.
- Search blocks in blocks catalog. Block catalog is where all the pre-defined blocks are listed. Users can search the block based on the block's name.
- Add block into user's canvas. Users add blocks into user's canvas by selecting blocks from a list of pre-defined blocks in block catalog. The selected blocks will appeared on the display area of user's canvas.
- Delete block from user's canvas. The displayed blocks on the user's canvas can be deleted. Users select the block to be deleted and choose the delete function.
- Customize the selected block. The block can be customized. For example in CNN news block, rather than display the title, description, publication date and item link, user just can opt to display the title and the description only.

The use case as shown in Figure 4 illustrates those functionalities supported in Whip.

8.2 Activity Diagram

Whip support both type of developer; application developer and block developer. Block developer will develop a mashup block and placed it on blocks catalog. Application developer then can select these blocks to make a mashup application. Each block is a separate web services or web data. The overall process of mashup development using Whip is shown in Figure 8 activity diagram.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 4: Requirement Specification for Whip

Requirement	Description
R1: ***Developed for the end users.***	Mashup tool is developed for the end users; a group of users that may not have a computer knowledge and skill. Thus it must provide an environment that allows end users to make mashup quickly and effortlessly. This can be done by remove the coding phase in the mashup development process and BBSD approach support this aim by let the users develop the mashup by select the block, customize, integrate and display.
R2: ***User's preference.***	To promote freedom and flexibility in EUD, users should be given a list of options to choose that suitable to their requirements. For instance, options of data sources, functionalities, etc.
R3: ***User's learning curve.***	Mashup tools should be developed exhibit a gently-sloped of learning curve [7]. Therefore users with very little programming experience can quickly learn and familiar with the process and capable to build the mashup on their own.
R4: ***Data integration and aggregation.***	Generally there are two types of mashup based on its process; mashup by integration and mashup by aggregation [21]. Mashup by integration is involving 'cross data' where one resource becomes the input for processing by another. Most of this type of mashup use API to integrate like WikiCrimes (www.wikicrimes.com). While mashup by aggregation just simply a collection of web content that live side by side within the aggregator like in iGoogle, NetVibes, MyYahoo and PageFlakes. Therefore, develop mashup by aggregation is easier and require no programming skill compare to develop mashup by integration [15].
R5: ***Web technologies.***	Mashup is an emerging technology on the Web 2.0 like Representational State Transfer (REST) Web services, RSS or Atom feeds, XML data formats, etc. Therefore, mashup tools should compatible with these latest technologies.
R6: ***Rapid development.***	Mashup application must be developed rapidly. Iterative and collaborative development technique can shorten the traditional edit-compile-test-run development life cycle.
R7: ***Availability of Internet connection.***	Mashup is a Web-based application and Internet connection should available during the development and implementation.
R8: ***Availability of API.***	APIs are used to glue together contents, functionalities and presentation to make mashup. Most of websites nowadays provide their own API.
R9: ***Availability of data sources.***	Data sources should available during the mashup development and implementation. It requires updates for mashup tool from time to time, by block developer. If any data source is no longer providing information, the block developer is responsible to figure out the replacement data source, in order to provide the same outcome as the previous unavailable source.
R10: ***Web browser for runtime environment.***	Mashup can be assembled at server side and client side. Both approaches require Web browser to display the output.
R11: ***Consistencies of data models.***	Not all websites use the same structured data model. Some websites may use unstructured data while others use structured data. Data integration can be done if only the data model is consistence.
R12: ***Sharing and Reuse.***	The developed web mashup application should can be shared and reused across users so that other user also can benefit from that mashup.

Journal of Theoretical and Applied Information Technology
31st December 2011. Vol. 34 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

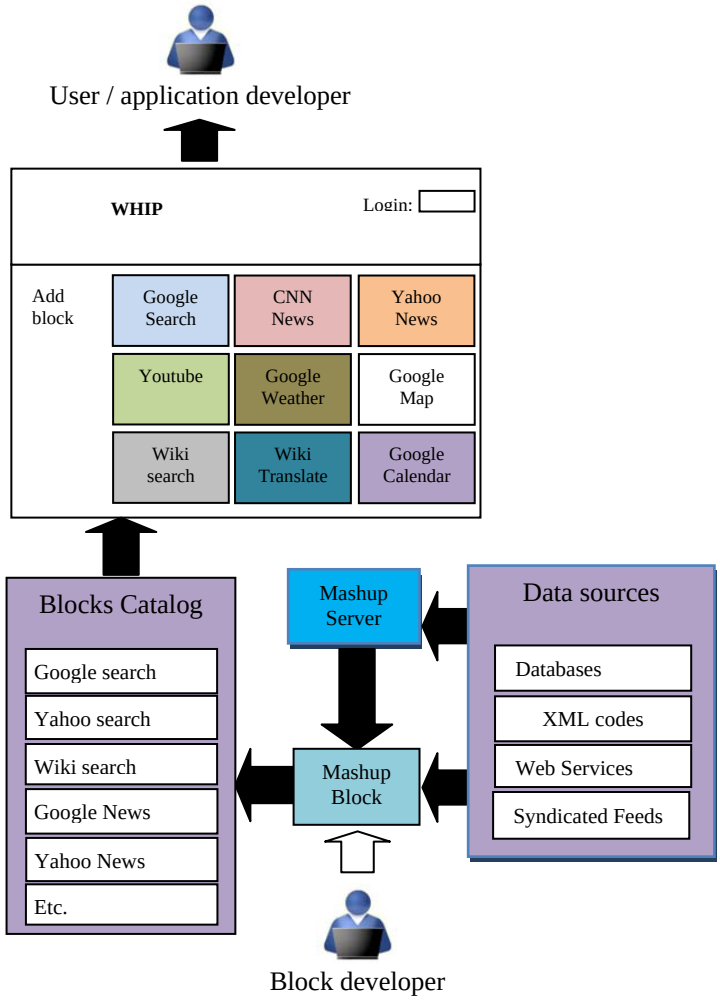

Figure 3: Whip Architecture

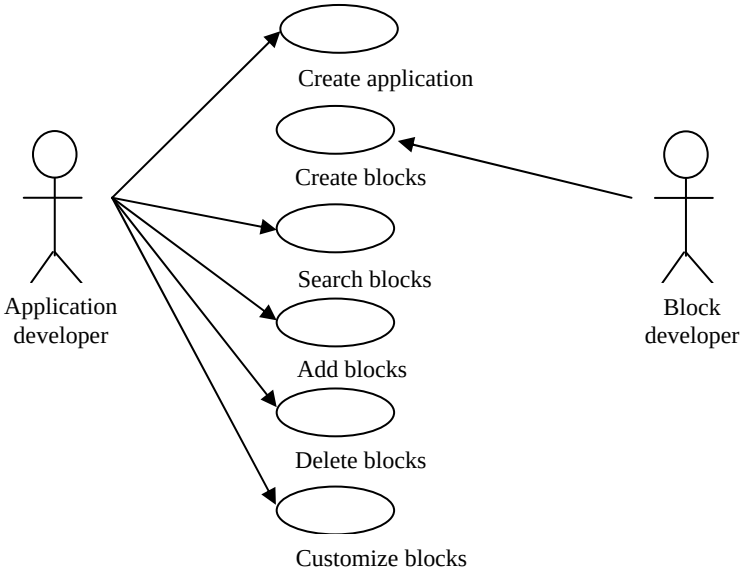

Figure 4: Use Case for Whip

Journal of Theoretical and Applied Information Technology
31st December 2011. Vol. 34 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

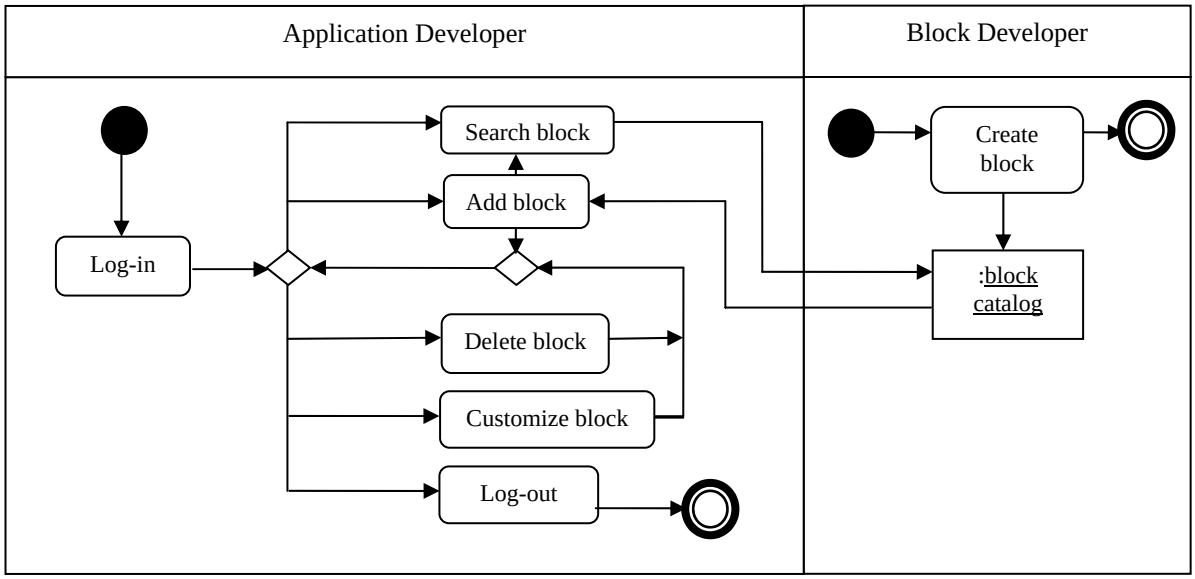

Figure 5: Whip Activity Diagram

8.3 User Interface Design

Whip user interface is simple where it just consists of two main panes; the menu and the canvas as shown in Figure 6. The menu consists of list of functions supported in Whip like add new block. The canvas is where the mashup blocks are integrate to form a mashup.

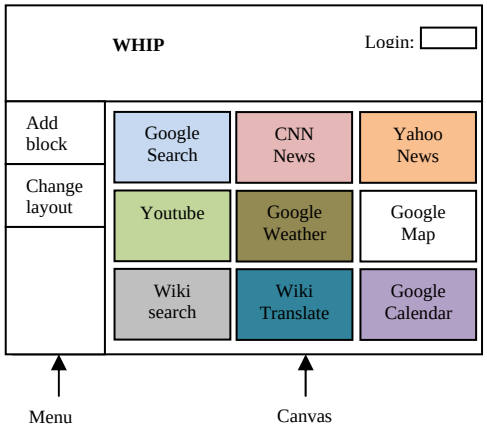

Figure 6: Whip user interface.

8.4 Block Definition

In Whip, blocks can be any web services or web data. Different blocks representing different web services and data. Examples of Whip's blocks are Google Search, Google News, Yahoo Search, Youtube Channel, Google Map, Music player, Wiki Translate, Wiki Search, and RSS feeds. The definition of block mashup is present in the following.

```
type block = (webServices, webData)

type blockCatalog = [Google Search, Google News, Yahoo Search, Youtube Channel, Google Map, Music player, Wiki Translate, Wiki Search, RSS feeds]
```

A list of blocks will form a mashup. The member of list must be of the same type however blocks in mashup can be of any types. Therefore, we can write mashup as a list of tuple as in the following.

```
type mashup = [(block)]
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

If user want to add a new block into mashup, that block must either a type of web service or web data.

```
mashup x = [(x)| x<-block]
```

Some blocks can be used to control other blocks like Google search can be used to search a keyword in CNN News or Google News.

```
mashup x y = [(x,y) | x<-block,
y<-block]
```

9. CONCLUSION

In this paper we present a design of Whip, a mashup tool that allows user to develop mashup through BBSD approach. BBSD approach is a combination idea of EUP paradigm and CBD approach where it let users to develop a mashup application by aggregate several blocks. The block is either a web service or web data retrieved from various sources. Through BBSD approach, users that lack of programming skill can easily develop a mashup for their own used or shared with other users. Besides the Whip architecture, and UML Use Case diagram to illustrate functionalities in Whip, we also present the activity diagram to demonstrate the process of making mashup using Whip. The design of blocks and requirements for developing Whip also presented.

REFERENCES

[1] Rokiah Bahari, Marini Abu Bakar, Norleyza Jailani, and Abdullah Mohd Zin, "A Technique to Identifying Blocks for Developing E-commerce Application " presented at the National Conference on Programming 2009 (Atur'09), Putrajaya, Malaysia, 2009.

[2] Michael Pierre Carlson, Anne H.H. Ngu, Rodion Podorozhny, and Liangzhao Zeng, "Automatic Mash Up of Composite Applications," in *International Conference on Service-Oriented Computing (ICSOC'08)*, 2008, pp. 317-330.

[3] Luba Cherbakov, Andy J.F. Bravery, and Aroop Pandya. (2007, April 19). *SOA Meets Situational Applications, Part 1: Changing Computing in the Enterprise*. Available: www.ibm.com/developerworks/webservices/library/ws-soa-situational1/

[4] Rob Ennals and Minos Garofalakis, "MashMaker: Mashups for the Masses," in *SIGMOD'07*, Beijing, China, 2007, pp. 1116-1118.

[5] Angus F.M Huang, Shin Bo Huang, Lee E.Y.F., and Stephen J.H. Yang, "Improving End User Programming with Situational Mashups in Web 2.0 Environment," in *IEEE International Symposium on Service-Oriented System Engineering 2008 (SOSE'08)*, Jhongli, 2008, pp. 62-67.

[6] Afizah Ismail, Marlinawati Djasmir, Nazlia Omar, and Abdullah Mohd Zin, "Designing and Implementing Blocks for Developing Educational Software for Children with Learning Disabilities," presented at the National Conference on Programming 2009 (ATUR'09), Putrajaya, Malaysia, 2009.

[7] Caitlin Kelleher and Randy Pausch, "Lowering the Barriers to Programming: a survey of programming environments and languages for novice programmers," *ACM Computing Surveys*, vol. 37, pp. 83-137, June 2005.

[8] Woralak Kongdenfha, Boualem Benatallah, Julien Vayssière, Régis Saint-Paul, and Fabio Casati, "Rapid development of spreadsheet-based web mashups," in *The 18th international conference on World wide web 2009 (WWW '09)*, Madrid, Spain, 2009, pp. 851-860.

[9] James Lin, Jeffrey Wong, Jeffrey Nichols, Allen Cypher, and Tessa A. Lau, "End-User Programming of Mashups with Vegemite," in *13th International conference on Intelligent User interfaces (IUI'09)*, Sanibel Island, Florida, USA, 2009, pp. 97-106.

[10] Giusy Di Lorenzo, Hakim Hacid, and Hye-young Paik, "Data Integration in Mashups," *SIGMOD Record*, vol. 38, p. 8, March 2009.

[11] Allan MacLean, Kathleen Carter, Lennart Loustrand, and Thomas P. Moran, "User-Tailorable Systems: Pressing the Issues with Buttons," in *SIGCHI Conference On Human Factors In Computing Systems: Empowering People*, Washington, United States, 1990, pp. 175 - 182.

[12] Siti Nor Hafizah Mohamad, Ahmed Patel, Yiqi Tew, Rodziah Latih, and Qais Qassim, "Principles and Dynamics of Block-based Programming Approach," in *2011 IEEE Symposium on Computers and Informatics (ISCI 2011)*, Kuala Lumpur, Malaysia, 2011, pp. 340-345.

[13] Brad A. Myers, Andrew J.Ko, and Margaret M. Burnett, "Invited Research Overview: End User Programming," in *CHI'06 Extended*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Abstracts on Human Factors in Computing, Montreal, Quebec, Canada, 2006, pp. 75-80.

[14] Abdallah Namoun, Tobias Nestler, and Antonella De Angeli, "End User Requirements for the Composable Web," in *ComposableWeb'10*, 2010, pp. 396-407.

[15] Ahmed Patel, Liu Na, Rodziah Latih, Christopher Wills, Zarina Shukur, and Rabia Mulla, "A Study of Mashup as a Software Application Development Techniquewith Examples from an End User Programming Perspective," *Journal of Computer Science*, vol. 6, pp. 1406-1415, 2010.

[16] Sven Rizzotti and Helmar Burkhart, "useKit - Lightweight Mashups for the Personalized Web," in *WWW2010*, Raleigh, North Carolina, 2010.

[17] Marwan Sabbouh, Jeff Higginson, Salim Semy, and Danny Gagne, "Web Mashup Scripting Language," in *16th International Conference on World Wide Web (WWW 2007)*, Banff, Alberta, Canada, 2007, pp. 1305-1306.

[18] Maxim Shevertalov and Spiros Mancoridis, "A Case Study on the Automatic Composition of Network Application Mashups," in *23rd IEEE/ACM International Conference on Automated Software Engineering* L'Aquila, 2008, pp. 359-362.

[19] Rattapoom Tuchinda, Pedro Szekely, and Craig A. Knoblock, "Building Mashups by Example," in *International Conference on Intelligent User Interfaces*, Gran Canaria, Spain, 2008, pp. 139-148.

[20] Guiling Wang, Shaohua Yang, and Yanbo Han, "A Spreadsheet-like Construct for Streamlining and Reusing Mashups," in *The 9th International Conference for Young Computer Scientists*, 2008.

[21] Kevin Wiliarty. (2008, July 5). *Educational Mashups 2*. Available: http://www.academiccommons.org/commons/review/educational-mashups-2

[22] Jeffrey Wong and Jason I. Hong, "Making mashups with Marmite: Towards End-User Programming for the web," in *SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*, New York, USA, 2007, pp. 1435-1444.

[23] Jin Yu, Boualem Benatallah, Fabio Casati, and Florian Daniel, "Understanding Mashup Development," *IEEE Internet Computing*, vol. 12, pp. 44-52, 2008.