Study on the Mobile Mashup WebApp Development System

Yoon-Seop Chang*, Jae-Chul Kim*, Seong-Ho Lee*, Young-Jae Lim*
*ETRI(Electronics and Telecommunications Research Institute), Korea
ychang76@etri.re.kr, jckim@etri.re.kr, sholee@etri.re.kr, yjlim@etri.re.kr

***Abstract*— People now expect similar user experiences on mobile devices as those on PCs, and mobile mashups are also required as equivalents of existing web mashups. Well-organized systems and tools are required for easier development of mobile mashups. However, most of existing systems and tools are not easy because they are designed for experts not for common users. Tools for mobile mashups need to be differentiated for each user group. The design of a new development system for mobile mashup webapps was proposed considering these two user groups and it has been implemented successfully in this study. Users can mashup various useful mobile webapps using GUI tools on the web browser. The system also includes many useful built-in blocks for the usage by common users, and these blocks can be shared and reused among users. It enables easier development of mobile mashup webapps for both of experts and common users.**

***Keywords*— Mobile Mashup, WebApp, Development System, Experts, Common Users**

I. INTRODUCTION

Mashup is a process of integrating several data and functions from different sources [1]-[3] and it enables users to make their own web applications. The web provides an enormous amount of data as open formats (such as RSS and Atom) and functions with various interfaces (such as REST, Javascript and SOAP) for mashup [4]. The statistics on the site "programmable web" show there are registered 7,770 sites with published APIs at the time of writing.

Nowadays most people rarely stay in front of PCs, but they are always carrying network-accessible mobile devices with them. Emerging devices such as smartphones, pads (tablet-PCs), PDAs, net-books and so on have taped the latest technological breakthroughs in wireless networking and leading market trends [5]. Smartphones and pads have become necessities in their everyday life, and people use these devices to access the network wherever they are wandering. For instance, they search information immediately using their mobile devices instead of turning on the power of PCs. Smartphone sales had overtaken those of PCs for the first time in 2011 according to the research firm "Canalysis". "Cisco Visual Networking Index" also shows that global mobile data traffic grew 2.3-fold in 2011.

People now expect similar user experiences on mobile devices as those on PCs. In the same manner, mobile mashups are strongly required as equivalents of existing web mashups on PCs. While web mashups integrate open web resources, mobile mashups should also integrate device functionality and user's context information (such as location, schedules, contacts etc.) on mobile devices. As a result, mobile mashups are able to provide various useful context-aware mobile applications [6], [7]. There are already a number of mobile applications in Apple App Store and Google Play, and most of them came from mashup ideas.

There is an enormous amount of open resources such as feeds, open APIs and light-weight programming techniques (such as HTML, Javascript, CSS) that make it easier to develop mashup applications. Well-organized systems and tools are also required for easier development of mobile mashups. However, most of existing systems and tools are not easy because they are designed for experts, i.e. developers, not for common users. Tools for mobile mashups need to be differentiated for each user group by considering both of two user groups. Mobile devices have limitations such as screen size, user interactions and so on. Appropriate tools also should be prepared for each environment of PC and mobile devices.

This study dealt with issues on easier development environment for mobile mashups. The design of a new development system for mobile mashup was proposed and it has been implemented successfully. Challenges and results of this study will be presented in following sections.

II. MASHUP DEVELOPMENT ON THE PC AND MOBILE WEB ENVIRONMENT

There are many cases of studies, suggestions, implemented systems and final products of development system for web mashups on PCs compared with mobile mashups. "Mash-o-matic" is a utility which exploits superimposed information (SI) referenced to fragments of existing information, and it was implemented using the middleware and query processor for SI [8]. "MashMaker" is a tool to edit, query, manipulate, and visualize live data from web pages [9]. It exports useful queries as widgets, and all query tasks can be formulated through interactive browsing and exploration. "Marmite" extracts contents from web pages and integrates them in data flow manner [10]. It provides linked views of program and data, and it suggests available operators based on the data being processed. IBM "Sharable Code" takes its domain-specific language (DSL) defined mashups then generates codes for Ruby on Rails web applications [4]. The platform contains creator and community applications to

create and share mashups. Yahoo "Pipes" and Microsoft "Popfly" let users to make mashups using predefined components with drag-and-drop interfaces. IBM developed "QEDWiki (Quick and Easily Done Wiki)" and "DAMIA (Data Mashup Fabric for Intranet Applications)" for combining user interface components and data mashup.

There were also several studies and experimental architectures of development system for mobile mashups. Maximilian [11] discussed challenges of mobile mashup such as minimizing communication, making possible quick reading, minimizing user inputs, and novel human interfaces etc. Cramer et al. [12] also discussed advantages and tradeoffs of using mobile mashup for research purposes. They mentioned risks of API changes, service downtime, limit of available contents and dependency on service strategies etc. Xu et al. [2] suggested architecture of SOA based mobile mashup platform which guarantees the quality of services by adopting self adaptive management on each service level. AT&T developed a speech mashup platform for multimodal mobile services [1]. The platform allocates speech processing resources on the server and combines those resources with web contents into multimodal mobile mashups. Peng et al. [13] suggested semantic-based mobile platform. The platform recommends possible links between components by semantic annotation of input, output parameters and by the probability of links and services in repository. "TELAR" is a mashup platform for mobile devices such as the Nokia Internet Tablets [6], [7]. On the server side, wrappers integrate data from web-based services. On the client side, the platform integrates context information of local sensors into context-aware mobile mashups. Yahoo provides "Pipes" website for iPhone, and the site has the list-centric interface which make it similar to native applications [14].

However, former cases, i.e. development systems for web mashups on PCs, were designed without any consideration of mobile mashups executed on mobile devices. All of them were aimed at the development and execution of mashups on PC web environment. Existing systems for mobile mashups in latter cases are not easy because those systems were not aimed at common users. Those systems were designed without any consideration for direct development of mobile mashups on mobile devices. Instead, they only dealt with issues on the development of mobile mashups in PC environment in which mobile devices only manage the execution of mobile mashup results. However, common users rarely stay in front of PCs as mentioned previously. Development environment for mobile mashups need to be differentiated for common users, and appropriate tools also should be prepared for each environment of PC and mobile devices.

III. A New Development System for Mobile Mashups

A. Challenges for easier development of mobile mashups

A new development system for mobile mashups has been proposed in this study. The system was designed and implemented considering following challenges. All these challenges are prerequisites of easier development environment for end-users.

- experts vs. common users
- easier procedure of mashup development
- mobile mashups as a form of webapp

Common users almost always bring mobile devices with them anywhere, and they use a number of mobile apps in daily life. As a result, they are getting more familiar with those apps and now have desires to make those apps by themselves using their ideas. The number of open APIs on the web continues to grow continuously, and mashup using these APIs is rather easy to grasp the concept. However, most of existing mashup systems and tools are aimed at the usage by experts instead of common users. As a result, it is difficult for common users to understand and to use those systems and tools. Tools for mobile mashups need to be differentiated for each user group. And mobile devices have limitations such as screen size, user interaction method and so on, so appropriate tools are also required for each environment of PC and mobile devices.

Procedure of mashup development consists of making an idea, searching open APIs, wrapping into blocks (or components), constructing workflow, authoring UIs and so on. Creating a new block is the task in which common users feel difficulties because it includes programming and experts also spends most of time for this during the mashup development. However, it may be rather easy to make mashups if there are already all necessary blocks. It is necessary to differentiate roles between experts and common users about the creation and usage of these blocks.

Most of mashup developments in existing systems are workflow-oriented, and users need to understand and predict inner execution flows among many blocks. It is difficult for common users who are not familiar with programming. If the mashup development begins with UI authoring, then it will become more intuitive for common users. The mobile mashup development system in this study takes this UI-oriented approach.

Nowadays a number of applications can be executed on the web browser, and there are several webapp markets such as Google's "Chrome Web Store" and "Open Web App Store" from Mozilla project. Webapp can be the most suitable form of mobile mashup application because mobile mashups utilize open web resources and should solve cross-platform problems. The advance of standard web technologies such as HTML5, CSS, and Javascript also promotes the adoption of mobile mashups as a form of webapp.

B. Approaches in this study for problem solving

As mentioned before, user groups are divided into two groups, i.e. experts and common users, and tools in the system were implemented individually for each user group as shown in Figure 1. Mashup tools for experts take the PC based form because it should provide whole functionality such as searching open APIs, making blocks, making webapps, other managements and so on. These mashup tools were designed to be executed on PC web browsers and these have richer user interfaces than those on mobile devices for common users.

Mashup tools for common users are executed on mobile devices, i.e. smartphones and pads. These tools provide limited functionality compared with those for experts. These

tools provide functions for making webapps and partial management. It is because of the limited programming skills of common users and the limited hardware of mobile devices.

Figure 1. The mobile mashup development system which was designed for both of experts and common users.

This study suggested that experts create previously plenty of built-in blocks using the tools on the PC environment, and then common users just make mashups using these blocks without creating any additional blocks. Open APIs in top 10 categories cover over 90% of total usage in mashup development as shown in Figure 2. It will be very useful if built-in blocks are ready for these open APIs. The system of this study already has many built-in blocks over 30 blocks and continues adding additional blocks.

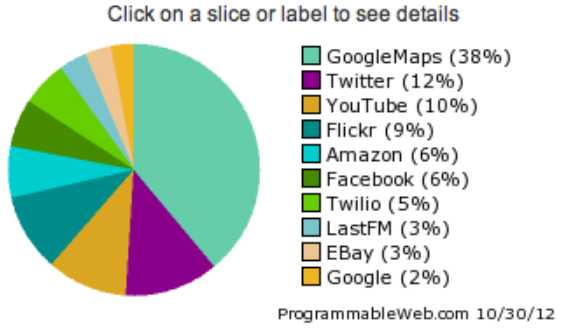

Figure 2. Open APIs in top 10 categories cover over 90% of total usage.

If the mashup development begins with UI authoring, then it will become more intuitive for common users. This study adopted UI-oriented approach instead of workflow-oriented approach for easier development of mashups. For instance, creating a new mashup begins with the edit of page layout for block arrangement, and then user arranges blocks into the page by drag-and-drop. User can preview the appearance of each mashup and HTML codes of the mashup can be generated automatically behind the block builder. All blocks are posted into the workflow canvas coincidentally during the above procedure, and the edit of mashup workflow is finally carried out. Creating a new block also begins with drag-and-drop UI controls to the canvas in the block builder, and HTML codes of each block can be generated automatically.

The results from this system take the form of mobile webapp instead of nativeapp or hybridapp, so those applications from the system can be executed on various web browsers in mobile devices. Webapp may be the most suitable form of mobile mashup application because mobile mashups utilize open web resources and should solve cross-platform problems. HTML5, CSS, Javascript and jQuery Mobile library were adopted in this study for easier development of mashup applications as the form of mobile webapp.

C. System Architecture of PC based mashup tools

At the time of writing, the implementation of PC based mashup tools for experts has been finished, and works for mobile device based tools are going on now. The design of mobile device based tools has been finished and implemented results will be shown at the beginning of next year. Figure 3 shows the architecture of implemented system which consists of three components. One is the mobile mashup engine, the other is mobile mashup authoring interface, and another is the mobile runtime environment.

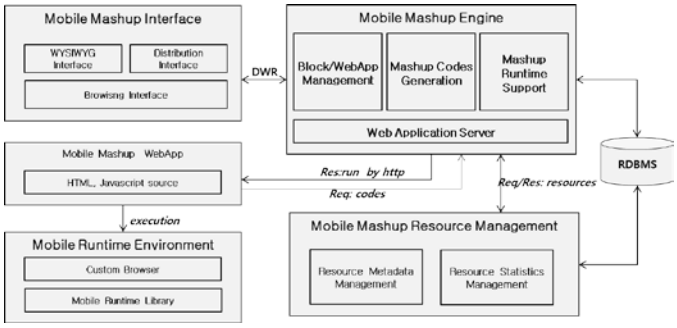

Figure 3. The architecture of mobile mashup webapp development system.

The mobile mashup engine was implemented on the server side using Java and MySQL. The engine's main role is to generate mashup codes, and manage mashup blocks and webapps in the system. The mobile mashup authoring interface was implemented on the client side using HTML5(HTML, CSS, Javascript) and it can be executed on web browsers such as Chrome, Safari and Internet Explore 9 etc. This interface is a kind of user client which provides front-end tools for mashup development. It interacts with mashup engine on the server through DWR (Direct Web Remoting) interface which enable the interaction between Java on the server and Javascript on the web browser. User can make mashups by drag-and-drop and WYSWYG interfaces. The mobile runtime environment is a kind of custom web browser with additional runtime library. This mobile runtime library enables mobile mashup webapps to access the functionality of mobile devices such as camera, gallery and so on.

Figure 4 shows components of the mobile mashup engine. The layout code generator analyzes layout information from webapp metadata then generates integrated HTML codes. It optimizes layout for mobile environment by adopting jQuery Mobile library. Workflow code generator analyzes workflow information from webapp metadata, and it generates

Javascript codes which controls logical execution of the mashup. Core and common libraries on the server provide functions such as Ajax call, proxy generation, data validation, lifecycle management and so on. These libraries also support the mashup of REST and Javascript open APIs, and it will be extended for RSS/Atom, SOAP and so on.

Figure 5 shows components of the mobile mashup authoring interface. Block builder of the client includes UI forms for metadata input, Javascript editor for block's logic, and HTML editor which supports jQuery Mobile controls. Webapp builder includes page layout editor and workflow editor. The client also provides menus to inquiry, preview, share blocks and webapps in the system.

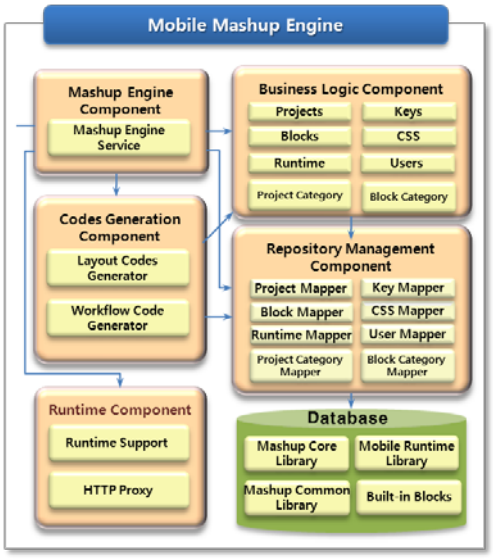

Figure 4. Components of the mobile mashup engine.

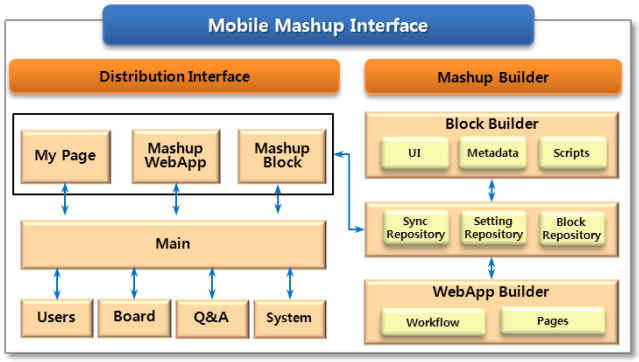

Figure 5. Components of the mobile mashup interface.

IV. Implemented Results and Mashup Examples

A. Implemented Results of the System

The user client on the web browser was implemented using HTML, Javascript and CSS without any plug-in extension such as Flash, ActiveX and Java Applet. As a result, the web client can be accessible on various web browsers. The client supports Google Chrome, Apple Safari and Microsoft Internet Explorer 9 at this time. The client consists of three parts; block builder, webapp builder, and the management menu for blocks and webapps,

The management menu for blocks and webapps consists of "My Page", "WebApps" and "Blocks". "My Page" shows lists of blocks and webapps of each user (Figure 6). "WebApps" and "Blocks" each shows list of blocks and webapps open to all users in the system. Users can inquiry with categories or can search by keywords for blocks and webapps. Users can share their blocks and webapps each other among users. Signed keys for open API usage are also managed and users can also create, modify and delete their own CSS themes.

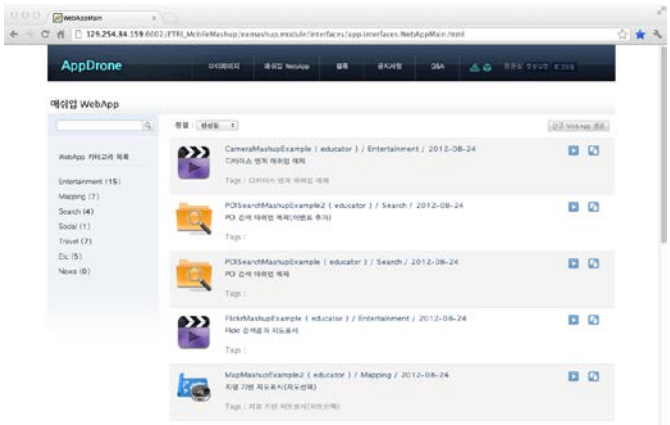

Figure 6. Management menu for blocks and webapps.

Blocks in the system consist of three main resources; xml metadata, HTML codes, and Javascript codes. Block builder also consists of four editors for these resources (Figure 7). XML metadata includes the whole information of each block. This information can be used during edit on the webapp builder and also used by server components for mashup codes generation. HTML codes includes UIs layout of each displayable block. HTML codes can be generated automatically by drag-and-drop jQuery Mobile controls into the block builder. Users can also write their own HTML codes into the HTML editor for more complicated and customized contents. Javascript codes of each block contain the execution logic of operations. Skeleton of these script codes can be generated from the block metadata and this reduces much of time and costs during the block editing.

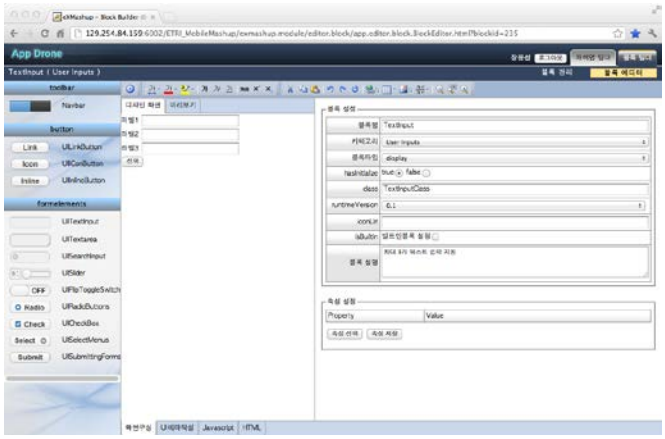

Figure 7. Block builder of the mobile mashup web client.

When all necessary blocks are ready, then mobile mashup webapp can be built from these blocks using the webapp builder on the web client. The webapp builder consists of layout editor and workflow editor (Figure 8).

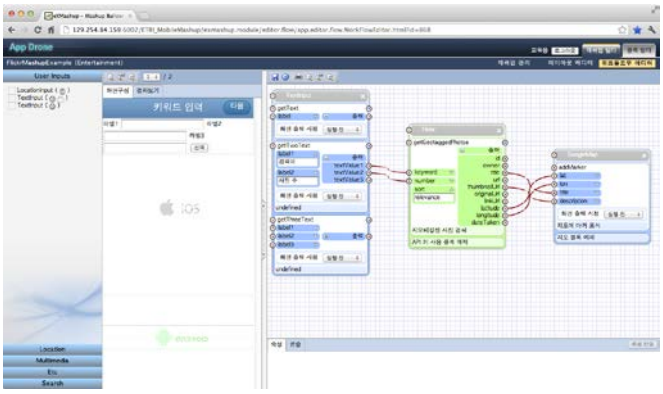

Figure 8. WebApp builder of the mobile mashup web client.

Mobile mashup webapps from the system take the multi-pages structure using jQuery Mobile library. The layout editor on the webapp builder supports the edit of multiple pages; pages creation and modification, toggle and edit of header and footer, setting navigation buttons and its transition, placement of grid cell for layout, applying CSS themes to contents, header and footer.

After finishing the layout edit for all pages, users can construct workflow for actual execution of the mashup using the workflow editor. Users drag and drop blocks into cells in pages, then those blocks are also posted into the workflow canvas. Users can complete their mashup webapp by linking input and output parameters of operations among related blocks. Block's properties, signed keys for open APIs and event-listener mapping between blocks also can be configured in this workflow canvas. Mashup workflows in this study also include conditional branch blocks, so users can make more dynamic mashup webapps using these blocks.

B. Mashup Examples using the System

There are already many blocks in the system within categories such as location, multimedia, social, search, operators, user inputs and so on. These blocks also can be shared and modified among users in the system. As a result, users can make a number of mobile mashup examples using these blocks and the system. Some typical examples of mobile mashups using the system are presented below, and there are already many other mobile mashup webapp examples and demonstrations in the system.

Figure 9 shows an example of mobile webapp which mashups Flickr and Google Maps open APIs. "TextInput" block get a search keyword from user and pass it to the next "Flickr" block. "Flickr" block search geotagged photos using the keyword passed from previous block and pass it to the next. Finally, "Google Maps" block adds markers on the map using geotagged photos array from the previous block. This webapp example consists of two pages; one for user input and the other for map display. "Flickr" block is a kind of processing block which processes requests and responses for Flickr open API.

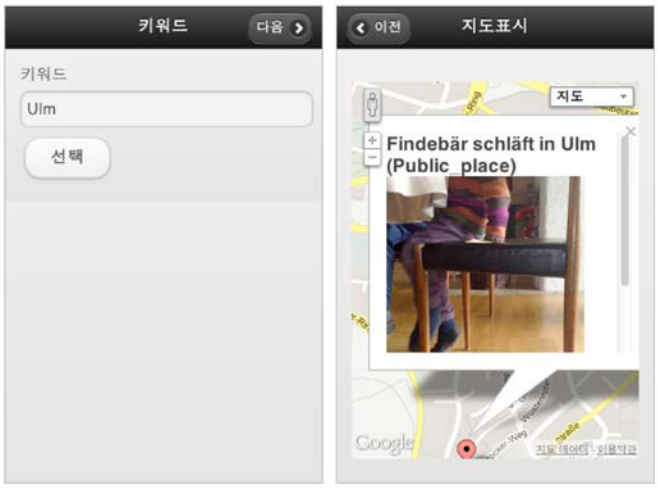

Figure 9. Mobile WebApp example which mashups Flickr and Google Maps Open APIs.

Other mobile mashup webapps can be made quickly and easily by replacing "Flickr" block or "Google Maps" block each to the other image block or map block without additional works. Appearance of the webapp also can be modified quickly and easily just using the layout editor on the webapp builder.

Next example is a mobile webapp which mashups Twitter open API, one of typical social services (Figure 10). "Twitter" block searches twits for the keyword input from users. This mashup webapp displays searched results on the "Listview" block. The workflow of this mashup includes "ListItems" block which is an example of operator block which manipulates items to the suitable format for output on the "Listview" block. In this example, "Twitter" block can be replaced with other social blocks like "Facebook" block. Display of searched results also can be changed quickly by replacing "Listview" block with other appropriate blocks.

Figure 10. Mobile WebApp example which mashups Twitter Open API.

Last example is a mashup webapp which mashups mobile device resources in addition to mashable web resources (Figure 11). The workflow of this webapp includes “Camera” block which accesses device’s camera functions by using mobile runtime library. Mobile runtime library has been implemented within the custom web browser developed in this study, and webapps using mobile devices functions can be executed on this custom browser. A number of mobile mashup webapps can be made by integrating device functions and user information on mobile devices. If major web browsers become to support the access of mobile device functions in standard manner, this custom web browser can be thrown away.

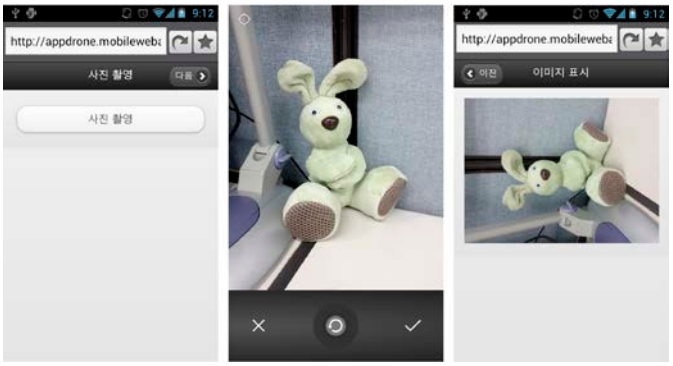

Figure 11. Mobile WebApp example which mashups functions of camera in smartphone.

C. Further Evaluation of User Experiences

The concept and results of this system have been introduced in the event “DevOn 2012” which is an annual event for both of developers and common users hosted by Daum Communication, one of major portal companies in Korea. There were positive responses for the system and useful feedbacks for mobile based system and tools. The system and more results of it will be shown again to public in another event “DevDay” and it is expected to become a good chance for further evaluation of user experiences using the system.

V. CONCLUSIONS

A new mobile mashup webapp development system has been designed and implemented for both of experts and common users. The system is available on various web browsers, and it provides GUI environment with simple interactions for easier development. Users can create useful mobile webapps that mashups open web resources and mobile device resources. There are already many useful built-in blocks for mashups and theses blocks can be shared easily among users in the system. This enables more efficient development of mobile mashups in the various fields.

PC based system has been implemented at the time of writing, and additional works for the mobile based system for smartphones and pads will be completed soon. Users will be able to make their own mobile mashup webapps on mobile devices directly and the system will adopt further easier interfaces for this.

ACKNOWLEDGMENT

This research is supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Contents Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2011.

REFERENCES

[1] G.D. Fabbrizio, T. Okken, and J.G. Wilpon, “A Speech Mashup Framework for Multimedia Mobile Services,” *Proceedings of the International Conference on Multimodal Interfaces/Workshop on Machine Learning for Multimodal Interfaces*, pp.71-78, November 2009.

[2] H.Y. Xu, M. N. Song, H. Chen, and J. D. Song, “Research on SOA based mobile mashup platform for telecom networks,” *Journal of China Universities of Posts and Telecommunications*, vol. 15, pp. 31-36, 2008.

[3] K. H. Cheung, K. Y. Yip, J. P. Townsend, and M. Scotch, “HCLS 2.0/3.0: Health care and life sciences data mashup using Web 2.0/3.0,” *Journal of Biomedical Informatics*, vol. 41, pp. 694-705, 2008.

[4] E. M. Maximilien, A. Ranabahu, and K. Gomadam, “An Online Platform for Web APIs and Service Mashups,” IEEE Internet Computing, vol. 12, pp. 32-43, 2008.

[5] Y.J. Kim and J.B. Sim, “Acceptance-Diffusion Strategies for Tablet-PCs: Focused on Acceptance Factors of Non-Users and Satisfaction Factors of Users,” ETRI Journal, vol. 34, pp. 245-255, 2012.

[6] A. Brodt, D. Nicklas, S. Sathish, and B. Mitschang, “Context-Aware Mashups for Mobile Devices,” *Lecture Notes in Computer Science*, vol. 5175, pp. 280-291, 2008.

[7] A. Brodt and D. Nicklas, “The TELAR mobile mashup platform for Nokia Internet Tablets,” *Proceedings of International Conference on Extending Database Technology*, pp. 700-704, March 2008.

[8] S. Murthy, D. Maier, and L. Delcambre, “Mash-o-matic,” *Proc. of ACM Symposium on Document Engineering*, pp. 205-214, August, 2006.

[9] R. Ennals and M. Garofalakis, “MashMaker: Mashups for the Masses,” *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 1116-1118, June 2007.

[10] J. Wong and J. I. Hong, “Making Mashups with Marmite: Towards End-User Programming for the Web,” *Proceedings of Conference on Human Factors in Computing Systems*, pp. 1435-1444, April 2007.

[11] E. M. Maxmimilien, “Mobile Mashups: Thoughts, Directions, and Challenges,” *Proceedings of IEEE International Conference on Semantic Computing*, pp. 597-600, August 2008.

[12] H. Cramer, M. Rost, and L. E. Holmquist, “Services as Materials: Using Mashups for Reasearch,” *Proceedings of International Conference on Ubiquitous Computing*, pp. 9-12, September 2011.

[13] Z. Peng, H. Chen, J. Rao, Y. Liu, L. Wang, and J. Chen, “Semantic-based Mobile Mashup Platform,” *Proceedings of International Semantic Web Conference*, November 2010.

[14] J. Trevor, “Doing the Mobile Mash,” *IEEE Computer*, vol. 41, pp. 104-106, 2008.