Applications of Mobile Application Interface Description Language MAIDL

Prach Chaisatien, Korawit Prutsachainimmit, and Takehiro Tokuda

Department of Computer Science, Tokyo Institute of Technology
Meguro, Tokyo 152-8552, Japan
{prach,korawit,tokuda}@tt.cs.titech.ac.jp

Abstract. Developments of mobile mashup applications have a rapid growth in the recent years. We present a development of Mobile Application Interface Description Language (MAIDL) and its applications. The language enables the development of mobile mashup applications with less programming efforts. Using our description language, composers are able to reuse existent mobile applications, Web services, and Web applications as the components to create a mashup mobile application or a Tethered Web service on a mobile device (TeWS). We demonstrate the further application of a TeWS to deliver a cooperative mashup via a functionality exchange between an Android and an iOS device.

Keywords: Mobile mashup application, description language, tethered Web service, mobile Web server.

1 Introduction

A composition of Web information and mobile devices unique features has recently become an important development trend. In this paper, we approach a development of an XML-based description language to compose mobile mashup applications and Tethered Web services on a mobile device (TeWS). Components in the mashup execution are derived from a combination of existent mobile applications, JavaScript-based Web automations and Restful Web service consumptions. The composition method applied a workflow model which later translated into a script in description language called Mobile Application Interface Description Language (MAIDL). Finally, a mobile application or a TeWS is generated from the MAIDL script as an output. Furthermore, a complex mashup example is provided to demonstrate applications of the generated TeWS between mobile devices.

To integrate various functionalities to a mashup component, developers are having no alternative but to study a very specific programming language API. Divided by its target platform, mobile applications are generally created as mobile Web pages and native language applications. The major drawback when creating a multiplatform mobile Web application is that it tends to employ fewer amounts of mobile devices useful features. Moreover, mobile software developments using the devices native programming language require more explicit knowledge. In the term of data flows and Web-enabled information reuses, current approaches do not allow applications be developed as rapid as the Web-based ones do.

Code, which is generated from MAIDL, is in a procedural paradigm rather than declarative [1], since the control part mainly consists of procedures that are passing parameters and synchronizing processes in the mashup runtime environment. For this reason, we proposed automatic code generation algorithms, which assist composers in creating mobile mashup applications. In this research, we applied partial information extraction [2] and the final output is not limited to mobile application as traditional methods are [3]. A TeWS can be generated and later consumed by other clients. Later in an example, we show how the TeWS is applied to a platform-independent communication between devices.

2 Overview

2.1 Objective

Explore a mobile mashup model. The topics discussed in section 1 show that a mashup model for the mobile mashup application is not concretely defined. We aim to find an optimal mashup model which leads to a better solution in creating mashup applications for mobile devices.

Deliver reusability. Our mashup components include existent mobile applications and Web information. Developing mashup applications with low-level API, such as creating an image recognition component with a new algorithm, is beyond our research scope.

Enable fast prototyping. Mashup applications can be created from a Web-based software generation tool. Composers are allowed to generate source code, compile, and test it immediately after the composition model is correctly prepared. Methods called *Mashup Output Context Transformation* and *Mashup Process Scheduling Algorithm* would assist composers by automatically managing foreground and background runtime behaviors of the mashup components.

Demonstrate a Tethered Web service on mobile devices. A mashup application in our approach can be created as a mobile application to run on a device or as a TeWS. Functionality exchanges and interactive collaborations between devices can be derived from our approach, and these are unique features and contributions which do not appear in other approaches. In order to run the most flexible configuration on mobile devices (such as third party mobile applications and embedded server modules), we use the Android open source platform [5] as our mashup runtime environment.

2.2 Overview of MAIDL and Its Abstract Model Composition

The general concept of MAIDL (shown in Fig. 1) is to provide data flows between mashup components for its execution and output. The components consist of:

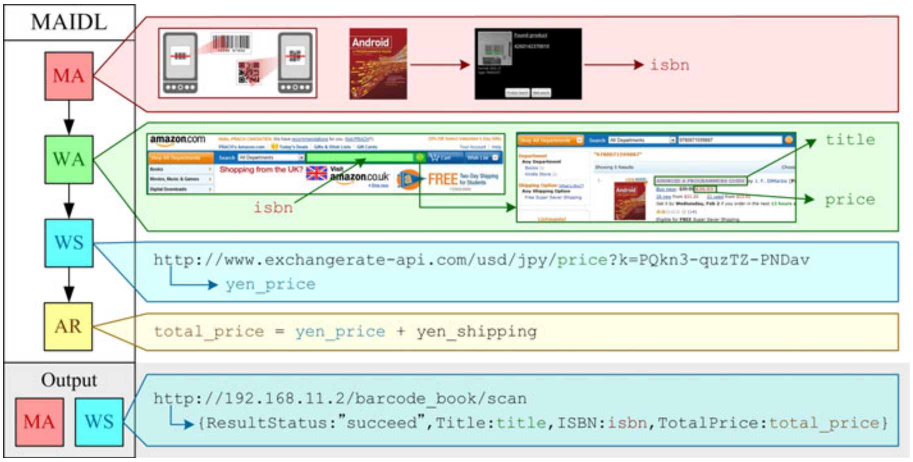
MAIDL
MA
WA
WS
AR
Output
MA
WS
isbn
title
price
isbn
http://www.exchangerate-api.com/usd/jpy/price?k=PQkn3-quzTZ-PNDav
yen_price
total_price = yen_price + yen_shipping
http://192.168.11.2/barcode_book/scan
{ResultStatus:"succeed",Title:title,ISBN:isbn,TotalPrice:total_price}

Fig. 1. Overview of MAIDL and its abstract model composition

1. Web Application Component (WA). A part of a Web page or a query through form in an HTML document can be reused through a WA component. Composers are provided with a tool to annotate tags and specify execution commands. JavaScript code will be generated according to the specification and execute automatically in the runtime environment on the mobile device.
2. Web Service Component (WS). Connections to REST Web services are applicable to our mashup composition. Composers specify a URL, a query path and a query expression (such as XPath or JSON dot notation) to access a part of the whole data.
3. Mobile Application Component (MA). A part of mashup execution can be derived from a mobile application. Our method allows an application which implemented Intent and Service [4] messaging protocol to be integrated.
4. Arithmetic Component (AR). A mathematical operation between parameters from one or more components can be performed through an Arithmatic Component. The operation includes addition, subtraction, division, multiplication, summation, comparison, array merge and GPS distance calculation from 2 pairs of GPS coordinates.

3 Cooperative Mashup

To demonstrate functionality exchanges and a cooperative mashup application, we created a mashup application using our approach. It requires interaction between 2 or more mobile devices. In this way, the application created in a TeWS output context can be deployed on an Android mobile phone. On the other hand, the iOS device [6] is manually programmed to consume the TeWS on the Android phone.

In this mashup application, geolocation of 2 devices are used as a data to find a list of restaurants located near the middle point between each devices GPS

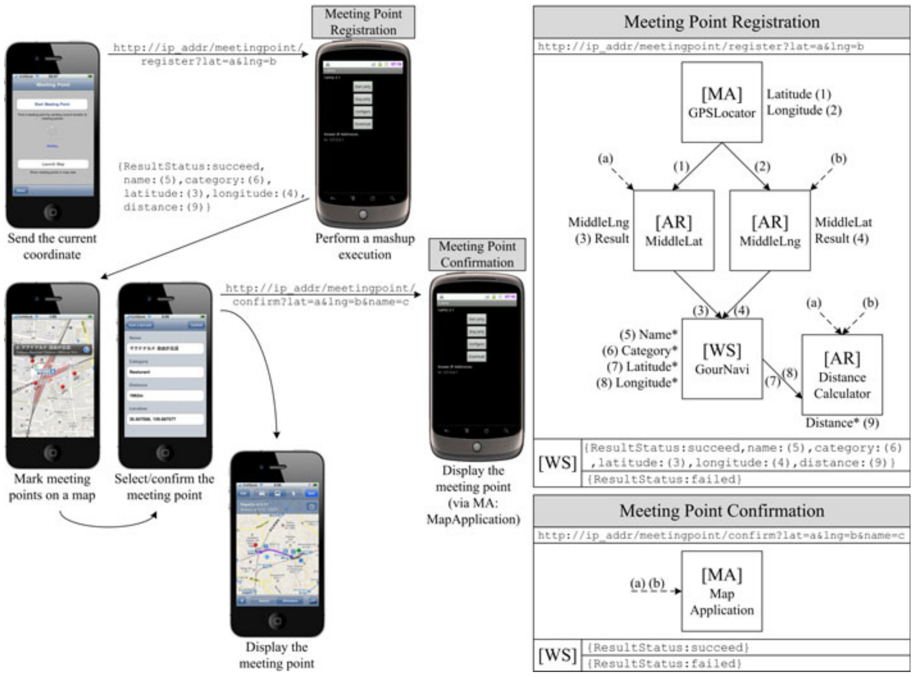
Meeting Point Registration
http://ip_addr/meetingpoint/register?lat=a&lng=b
Send the current coordinate
Perform a mashup execution
{ResultStatus:succeed, name:(5),category:(6), latitude:(3),longitude:(4), distance:(9)}
http://ip_addr/meetingpoint/confirm?lat=a&lng=b&name=c
Meeting Point Confirmation
Mark meeting points on a map
Select/confirm the meeting point
Display the meeting point
Display the meeting point (via MA: MapApplication)
Meeting Point Registration
http://ip_addr/meetingpoint/register?lat=a&lng=b
[MA] GPSLocator
Latitude (1)
Longitude (2)
(a)
(1)
(2)
(b)
MiddleLng (3) Result
[AR] MiddleLat
[AR] MiddleLng
MiddleLat Result (4)
(3)
(4)
(a)
(b)
(5) Name*
(6) Category*
(7) Latitude*
(8) Longitude*
[WS] GourNavi
(7)
(8)
[AR] Distance Calculator
Distance* (9)
[WS]
{ResultStatus:succeed,name:(5),category:(6),latitude:(3),longitude:(4),distance:(9)}
{ResultStatus:failed}
Meeting Point Confirmation
http://ip_addr/meetingpoint/confirm?lat=a&lng=b&name=c
(a),(b)
[MA] Map Application
[WS]
{ResultStatus:succeed}
{ResultStatus:failed}

Fig. 2. Mashup models and screenshots of *Meeting Point*

coordinates (via the *GourNavi* Web service [7]). Fig. 2 shows 2 mashup models and mashup applications, *Meeting Point Registration* and *Meeting Point Confirmation*, which communicate between devices via TeWS in separated contexts.

For the internal runtime and the connection performance, the application on the iOS side was presumably lightweight. Since this is a cooperative mashup application for 2 devices with a handshaking-like protocol, multiple connections are not considered as a performance factor. Overall performance of this mashup application depends on the performance of *GourNavi* Web service. All other components work in native code. In usability and interaction test, if we assumed that 2 devices are connected using global IP addresses and are placed outdoors, the interactions between 2 devices might be interrupted by signal loss. Both sides must have a timeout configuration and a reconnection arrangement in the case of failure execution.

4 Evaluation

To deliver smooth interactions between devices of a mashup application in the context of TeWS, the behavior of running process, network latency and usage scenario has to be observed. Since MAIDL files contain information about each component and its runtime behavior, an alternative application of MAIDL for performance measurement can be considered. MAIDL files also contain a concrete description of the output message sent via TeWS. Applications on the client side might be generated or adapt themselves according to the description. A good example for the combination of a TeWS and a desktop-based Web application is to exchange multiple data from a mobile phone to automatically fill in personal information in an HTML form. The desktop Web application first observes the applicable TeWS on the device and connects to it. In addition, the result from our usability evaluation of MAIDL can be interpret that MAIDL might not perform well when mashup applications are composed by novice composers because of its complexity. Expert users are able to use MAIDL without confusion and may apply it to external libraries. However, both groups expectations are met. Composers in both groups rated that the approach delivered 75% subjective rating for creating mashup applications.

5 Conclusion

In this research, we proposed a fast-paced mashup development using MAIDL. The composition enables integration of annotated parts of Web pages, connections to Web services and the use of existent mobile applications. The output can be designated for a single device, as a normal mobile application, or for multiple devices, as a Tethered Web service. In the mashup example, we demonstrated how a mashup application works in a Tethered Web service context to deliver functionality exchange and cooperative application between devices. Our future work is to enable mobile mashups in the context of a Web application on a mobile device. To support a higher interactivity to run on desktop computers, the process control and the composition method might be different from the contexts we have observed.

References

1. Gruhn, V., Schäfer, C.: An Architecture Description Language for Mobile Distributed Systems. In: Oquendo, F., Warboys, B.C., Morrison, R. (eds.) EWSA 2004. LNCS, vol. 3047, pp. 212–218. Springer, Heidelberg (2004)
2. Guo, J., Chaisatien, P., Han, H., Noro, T., Tokuda, T.: Partial Information Extraction Approach to Lightweight Integration on the Web. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 372–383. Springer, Heidelberg (2010)
3. Kaltofen, S., Milrad, M., Kurti, A.: A Cross-Platform Software System to Create and Deploy Mobile Mashups. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 518–521. Springer, Heidelberg (2010)
4. Android Intents, `http://developer.android.com/guide/topics/intents/`
5. Android Developers, `http://developer.android.com/index.html`
6. iOS Technology Overview, `http://developer.apple.com/technologies/ios/`
7. Gourmet Navigator API, `http://api.gnavi.co.jp/api/manual.htm`