Smartphone | Context Awareness | Service Linking

Technology Reports

The "BLOCCO" Service Linking System, Enabling Combination of Services through User Configuration

There are various services that implement context awareness and creation of mashups that can be customized to meet diversifying user needs, but most of these are difficult for users to operate or are limited in what they can accomplish as they are. For this reason, we have developed the BLOCCO[*1] *service linking system, which enables application-mushups by linking applications like playing with building blocks. BLOCCO has been developed jointly with GClue Inc.*

Service & Solution Development Department

Hiroaki Hagino
Kunihiro Fujii
Junko Murakami
Mirai Hara

1. Introduction

There are several services available that allow users to customize functions to suite their varying needs. In particular, context aware services[1], which allow service behaviors to be set according to conditions in addition to allowing simple customizations to behaviors, are increasing [2][3]. Mashup[*2][4] services, which allow services to be combined as a form of customization, are also appearing [5]-[8]. However, in most cases, these services are difficult for users to operate, or as they are, are limited in accomplishing what the user desires.

For this reason, we have developed BLOCCO[9], which enables users to do a form of application-mashups by linking applications like playing with building blocks.

BLOCCO is an Android™[*3] application which runs on the Android OS[10] and was developed jointly by NTT DOCOMO and GClue Inc. Using BLOCCO, users are able to combine multiple Android applications, freely configuring their behavior by detecting occurrences of events in one application and using them to trigger execution of services, and by passing parameters between applications and processing them in various ways. BLOCCO monitors the state of other applications, mediates passing of parameters between applications, and implements automatic execution of applications according to user configuration.

In this article, we give an overview of the BLOCCO service linking system and describe its effectiveness through comparison with other related technologies.

2. The "BLOCCO" Service Linking System

BLOCCO is an Android application that runs on the Android OS, and can be downloaded from the Android marketplace.

2.1 BLOCCO Features

With BLOCCO, multiple Android applications are combined by configuring a service execution scenario called a plan. BLOCCO and plans configured with BLOCCO have the following two features.

1) Event-driven Programming[*4]

Preset conditions such as "when xxx

*1 **BLOCCO**: Blocco is a trademark of GClue, Inc.
*2 **Mashup**: To create and provide a service by combining the content and services from several other, different services.
*3 **Android™**: An open source platform targeted mainly at mobile terminals or promoted by Google Inc., in the United States. Android™ is a trademark or registered trademark of Google Inc., in the United States.

occurs...” are detected and when these conditions are satisfied, a service is executed. In addition to running services, BLOCCO can allow plans to be configured from sets of action conditions and run operations, and the user can assign various Android applications to each action condition or run operation.

2) Parameter Linking between Services

For an application assigned to a run operation, parameters can be passed in from other applications. The user can also configure this type of action in a plan.

Examples of the main screens in BLOCCO are shown in **Figure 1**. BLOCCO uses plug-ins, which is a general name for applications that can be assigned to action conditions or run operations, and applications that can provide parameters to other applications assigned to a run operation. Plug-ins assigned to action conditions are called event plug-ins, those assigned to run operations are called action plug-ins, and those that provide parameters to action plug-ins are called input plug-ins.

Communication between each of the types of plug-in in BLOCCO is implemented using a protocol defined with the intent[*5] mechanism provided by the Android OS.

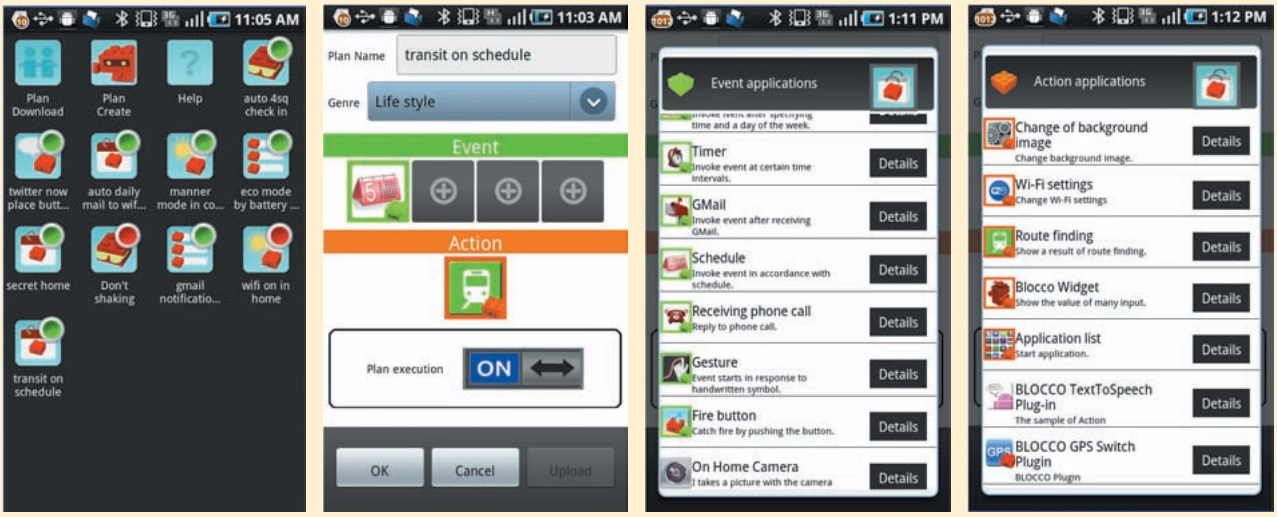
Next, we present some example plans that are possible using BLOCCO.

- Example plan 1:

  With this plan, if for example, an item called “Meet for drinks” is entered in the scheduler application for between 7 and 10 pm at the location “Roppongi”, then one hour before hand, the plan automatically initiates a search for a subway route from the current location and arriving at Roppongi at 7 pm.
- Example plan 2:

  With this plan, the user launches the handwriting gesture application[*6] and draws a circle when arriving at home, and the plan automatically posts the message “At home now” to Twitter[*7].

Example plan 1 is almost the same as functionality provided by NTT DOCOMO’s i-concier service. The plug-in assignments for example plan 1 are shown in **Figure 2**. The scheduler application is assigned as an event plug-in and configured with the condition “one hour before the registered item.” A subway route search application is assigned as the action plug-in. The arrival station and time



(a) Plan list screen (b) Plan configuration screen (c) Event plug-in list (d) Action plug-in list

Figure 1 BLOCCO screen shots

*4 **Event-driven programming**: To create a program based on detecting the occurrence of particular events and using this to trigger execution of sections of the program. With BLOCCO, it refers to how users create scenarios for running services.

*5 **Intent**: A mechanism provided by the Android OS for programs to exchange parameters. Used between components within an application and between applications to exchange information.

*6 **Handwriting gesture application**: An application for recognizing handwritten input. In this article, it refers to an application that allows the user to pre-register simple handwritten gestures such as a circle, and then for handwritten input from the user, determines which of the shapes it represents.

parameters required by the route-search application are obtained from the scheduler application, which is an event plug-in, but it can also act as an input plug-in to provide the parameters. The departure station parameter is provided by a GPS application. By using BLOCCO in this way, an application can be run automatically when any type of event occurs, by pre-configuring a plan with an action plug-in. In this way, by simply changing the combination of plug-in applications, various responses can be configured, such as automatically searching for a route to the event when an e-mail arrives from a friend inviting the user to an event.

The plug-in assignments for example plan 2 are shown in **Figure 3**. For example plan 2, a handwriting gesture application is used as the event plug-in, and a Twitter client application is the action plug-in. In this example, BLOCCO not only runs an application automatically, it also provides semi-automatic execution with a user-operation shortcut.

With BLOCCO, the trigger for a plan is raised by an event plug-in. In this way, if an application that monitors the state of an Android terminal is configured as an event plug-in, the plan can run applications automatically, and if an application that issues an execution trigger due to user operation is configured as an event plug-in, it can be used as a shortcut, for semiautomatic execution.

Thus, BLOCCO provides an environment that allows users to combine the applications installed on an Android terminal flexibly, creating a variety of plans. This is particularly useful for users that need to perform every-day operations automatically or semi-automatically.

2.2 User Operation Support

Generally, most services allowing

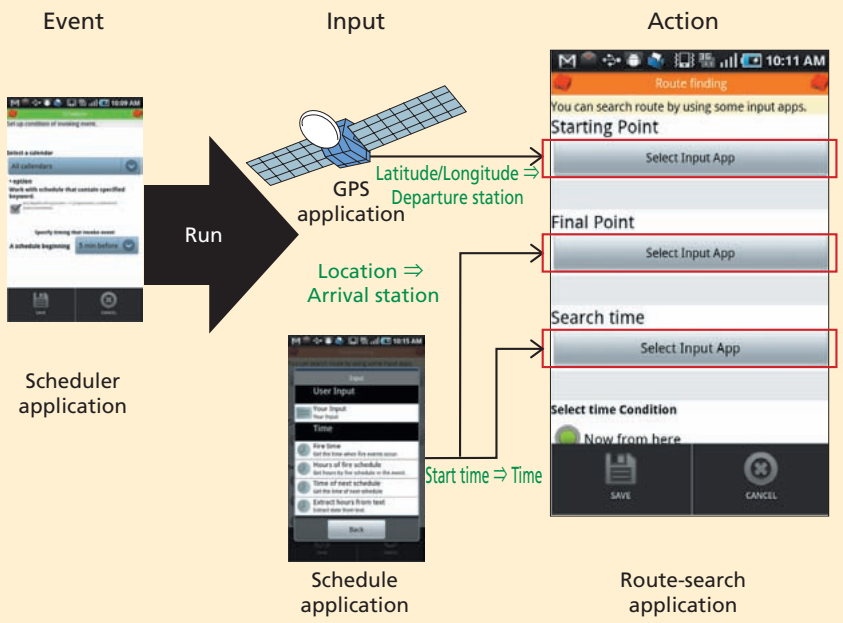


Figure 2 Example plan 1 plug-in assignment

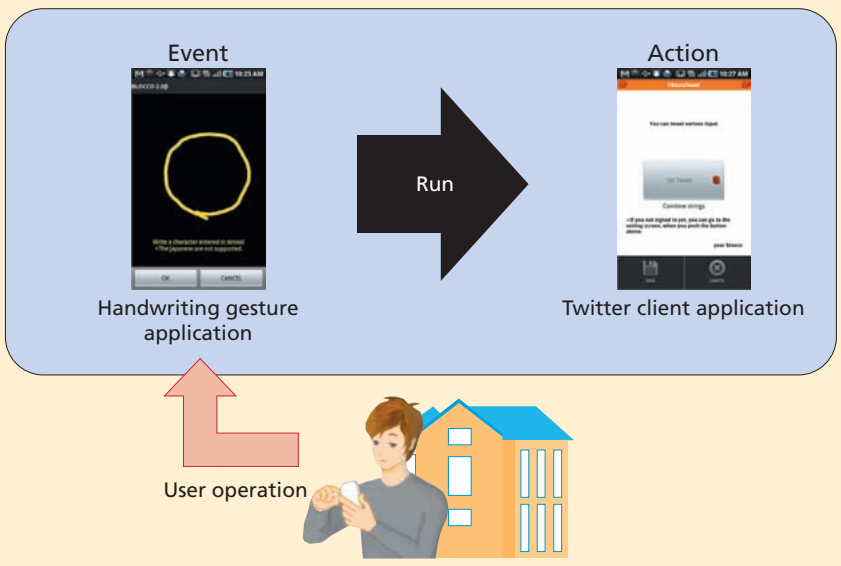


Figure 3 Example plan 2 plug-in assignment

*7 **Twitter**: A registered trademark of Twitter Inc. in the United States and other countries.

configuration of scenarios like plans in BLOCCO are for PCs, and many of them provide graphical display of parameter flow and relationships among components making up the scenarios [10]. However, BLOCCO presumes operation by a user on an Android terminal, so only a limited amount of information that can be displayed on the screen. With BLOCCO, particularly when the user is configuring a plan, the required plug-ins must be selected from among many applications and these must be configured, keeping the flow of parameters between them in mind, using the limited information displayed on the screen. Because of this, BLOCCO provides user support for configuring plans using the methods shown below. The screen transitions when configuring example plan 1 from Section 2.1 are shown in **Figure 4**.

1) Presentation of Plug-in List

BLOCCO automatically detects whether applications that can be used as plug-ins are installed. This allows a list of plug-ins to be displayed when the user is configuring a plan. The list of plug-ins is shown to the user in the following situations.

- When an event plug-in or action plug-in is to be assigned in a plan
- When an input plug-in is to be assigned to a parameter of an action plan

2) Parameter Passing Configuration

One of the strengths of BLOCCO is that connections linking services are parameter-to-parameter, rather than application-to-application. For example, in example plan 1 in Section 2.1, the "departure station," "arrival station" and "time" parameters for the route search application are assigned to the "latitude/longitude," "location" and "start time" values from the GPS and scheduler applications. Thus, in BLOCCO, the user can make transition directly from the parameters displayed on the action plug-in screen to a configuration screen for an input plug-in. When the user selects an action plug-in parameter, the input plug-

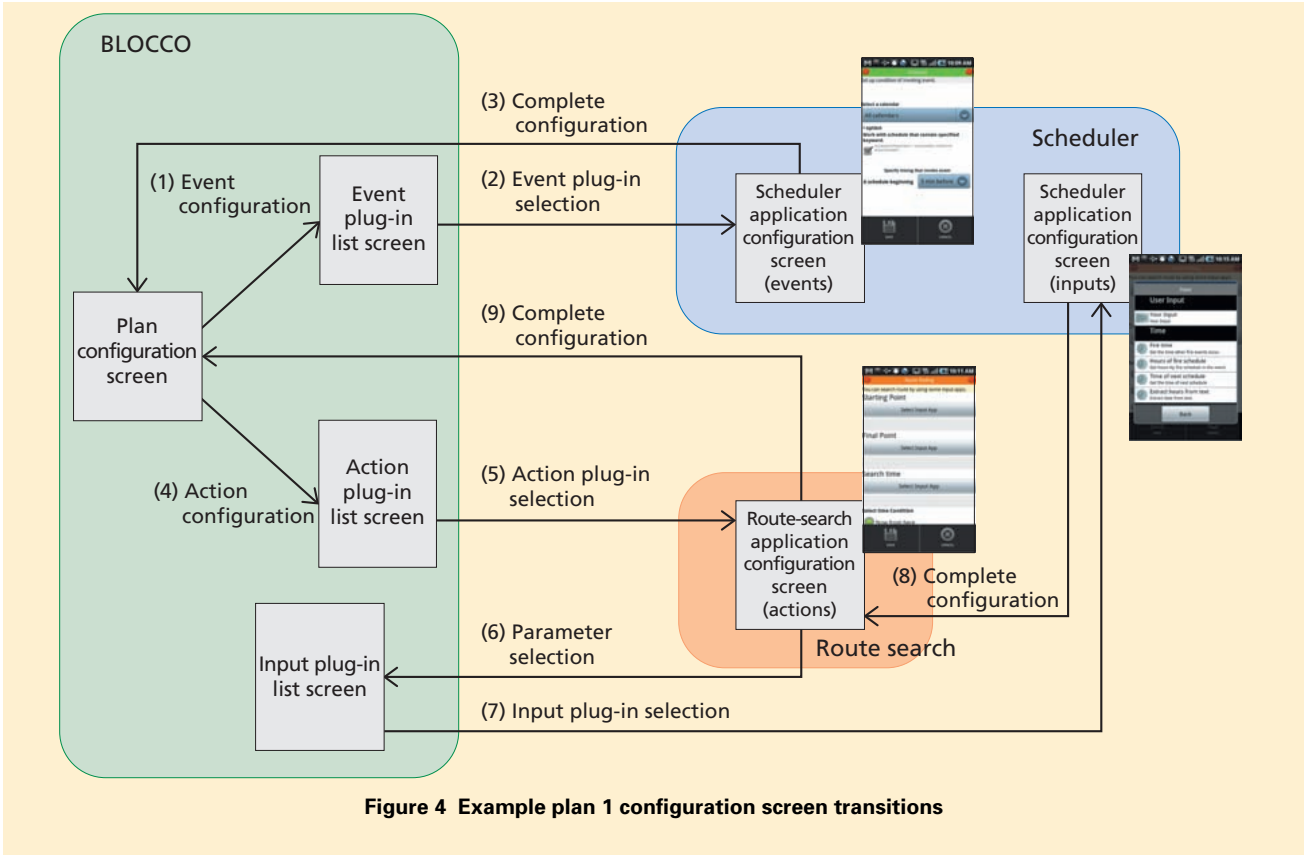


Figure 4 Example plan 1 configuration screen transitions

in list as described above is displayed.

Input plug-ins also have various settings in BLOCCO (**Figure 5**). In the example in the figure, the location, "Roppongi," is taken from the scheduler application, and that information is used to generate the string, "I'm at Roppongi," which is passed to the Twitter client application. For this case, there is an input plug-in that retrieves a parameter from another input plug-in, uses it to generate a string, and provides it to another plug-in as a parameter. In other words, the string-generating and scheduler applications are connected as a double input plug-in for the Twitter client, which is an action plug-in. In this type of case, the string generator application could also be configured as an input plug-in for a parameter of the Twitter client application, and the scheduler application could be configured as an input plug-in for a parameter of the string generator. With this type of process, configuration of complex plans can be done on the limited screen area of the Android terminal.

3. Discussion Regarding BLOCCO

3.1 The Service Linking Concept

BLOCCO has been designed and developed based on a service linking framework proposed by NTT DOCOMO. This service linking framework takes into consideration a combination of the three concepts of user customization, mashups, and context awareness[*8], and their related technologies. The service linking framework is an environment and technology that allows users, rather than developers, to combine services freely (user customization) and to control the behavior of services based on parameters that are passed between services (mashups, context awareness).

3.2 Related Technologies

There are various mechanisms that allow users to create mashups from services, including Plagger[5], IntelligentPad[*9][6], Yahoo! Pipes[*10][7], and Accelerators[8]. There are also context awareness services that allow user cus-

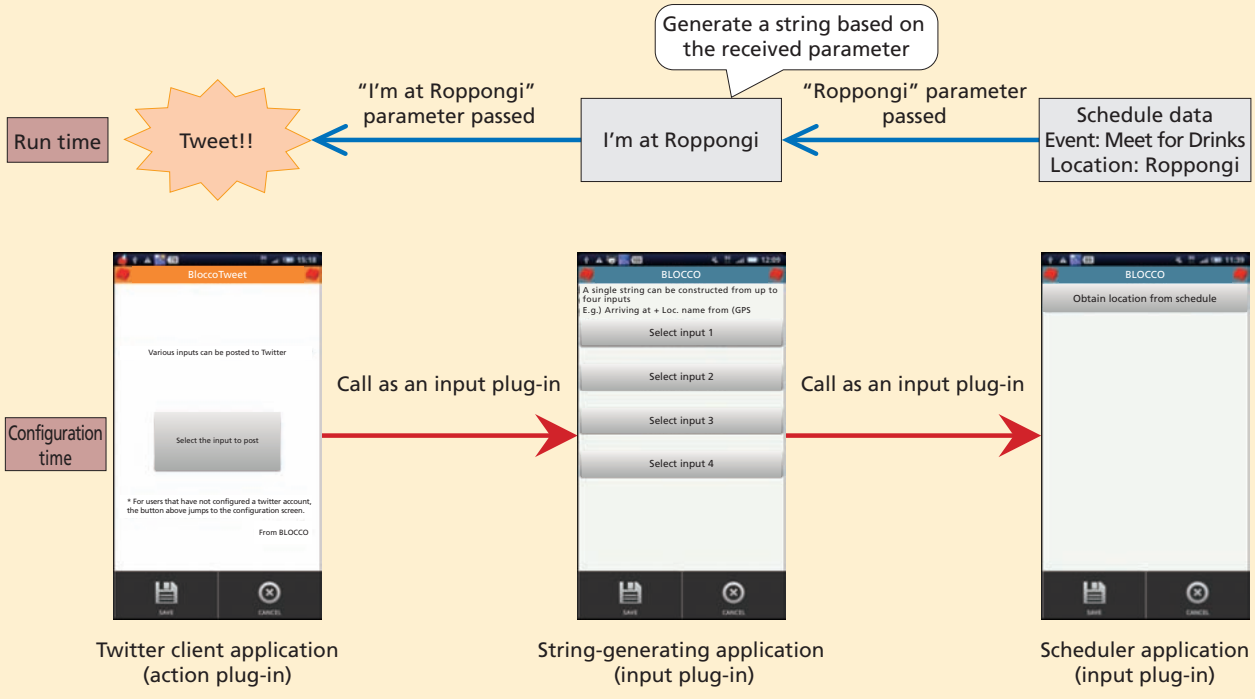


Figure 5 Input plug-in linking example

*8 **Context awareness**: A service that makes decisions or changes behavior based on information related to the user or service is called context aware, and this concept and type of behavior are called context awareness.

*9 **IntelligentPad**: A registered trademark of the Sapporo Electronics and Industries Cultivation Foundation.

*10 **Yahoo! Pipes**: Yahoo! and Yahoo! Pipes are trademarks or registered trademarks of Yahoo! Inc.

tomization, including Toggle Setting[3] and Locale[2]. A comparison of these technologies is shown in **Table 1**.

Of these, Locale is particularly close to BLOCCO in concept. Locale is an Android application that allows settings in an Android terminal to be changed automatically, and it features freedom of configuration and automatic execution. As an example, the user can have the Android terminal switch Wi-Fi®*11 on or off when it enters a particular area. As with BLOCCO, the user can combine these types of settings freely. Locale also has an interface that allows other Android applications to be plugged in as "Events," which trigger configuration changes, or "Settings" which can be configured, so the behavior of various services can be modified and it is not limited to terminal settings. This concept allows multiple services to be combined through event-driven programming. However, although it allows the state of one service to act as a trigger for another service, it does not allow parameter passing between services as BLOCCO does.

3.3 BLOCCO User-convenience

As mentioned in the explanation of linking technologies in Section 3.2, there have been context aware services with user customization, and also mashup services, but there has been no context aware service allowing combining of services (mashups) through user customization. This is an area where BLOCCO provides better convenience and flexibility than other related technologies like Locale.

Recently, many applications have also been developed and offered on smartphones, for services that were originally offered on the Web. As such, enabling links between applications should provide the best chances with the least development cost for encompassing the most content and services in the industry. Thus, while earlier mashups were for Web services and content, BLOCCO is for Android applications. Since BLOCCO itself is an Android application, its range of influence may seem to be limited to the within the Android terminal, but BLOCCO's service linking concept is able to combine any type of service or content, so it constitutes a new platform.

Table 1 Comparison of BLOCCO and related technologies

	Context awareness	Mashup	Simplicity	Flexibility	Ease of installation	Features
Plaggar	△	○	×	◎	◎	A mechanism for connecting Web services together. Knowledge of perl is required for configuration
IntelligentPad	×	◎	○	△	×	A service allowing users to create mashups by cutting and pasting Web content. Content must conform to a particular specification
Yahoo! Pipes	△	○	△	△	◎	A service allowing data obtained through RSS to be processed in various ways. Easy configuration using a GUI
Accelerators	×	○	△	×	◎	A service allowing a single parameter to be passed from one Web content item to another. Special knowledge is required for configuration
Toggle Setting	◎	×	◎	×	—	An application allowing the user to create sets of terminal settings for each situation and then change them with a single action
Locale	◎	○	○	△	○	An application allowing detailed configuration of terminal settings together with conditions and operations. Can take other applications as plug-ins
BLOCCO	◎	◎	△	○	△	A service that can combine with other applications, creating sets of conditions and actions for running. Parameters passed between applications at run time can also be configured

perl : An interpreted programming language. Generally well suited for processing string.
RSS (RFD Site Summary) : A method of encoding metadata such as Web page summaries that is generally used for notifying about Web page updates.

*11 **Wi-Fi®**: A registered trademark of the Wi-Fi Alliance.

3.4 Issues with BLOCCO

As described in Section 3.3, BLOCCO is able to pass parameters between services, in contrast to Locale. This allows users to create scenarios more flexibly. However, there is generally a trade-off with the complexity of user operation when functionality becomes more flexible or sophisticated. For example, comparing Toggle Setting and Locale, both are able to change Android terminal settings, but Locale provides more flexibility in creating scenarios and because of this, user operations are necessarily more complex. BLOCCO is even more flexible and sophisticated than Locale, so operations required of users are even more complex. BLOCCO is expected to be used by users capable of complex configuration, but to develop it as a new platform, it will be important in the future to prepare an environment that is easy to use for users that are not yet accustomed to its operations and settings.

3.5 BLOCCO Extensibility

The general utility of Web technology and Web content is behind the spread of mashup technologies for users, as introduced in Section 3.2. Since there is a large collection of this generally useful technology, it makes existing content available for various other uses, and user mashups are one such use. However, among the user mashup technologies described in Section 3.2, IntelligentPad, which supports only content that conforms to its own particular specification, is advanced and very useful in concept and technology, but in spite of this it has not become particularly popular. Rather than general purpose Web content, BLOCCO handles applications, and it can only work with applications that have particular interfaces defined by BLOCCO for plug-ins, not all existing applications. Because of this, a certain amount of development overhead relative to ordinary applications is unavoidable when developing an application that supports BLOCCO. In order to minimize this unavoidable development overhead, we have published the BLOCCO plug-in protocol interface specifications and a BLOCCO Software Development Kit (SDK)[*12] on the Web[9]. A screen shot of the SDK is shown in **Figure 6**. Using this SDK, a sample of the interface source code for communicating with BLOCCO can be generated automatically, greatly reducing the overhead required to develop a plug-in.

In addition to its concept of linking individual applications, the specifications and development environment needed to develop plug-ins for BLOCCO have been made public, so BLOCCO is also very extensible.

4. Conclusion

In this article, we have described an overview of the BLOCCO service linking system, and shown its superiority as a platform by demonstrating its usefulness and extensibility in comparison with other related technologies. In the future, we plan to study mechanisms to help users that have difficulty with configuration to use BLOCCO as well, such as a mechanism allowing plans created by one user to be shared with other users.

REFERENCES

[1] A.K. Dey and G. D. Abowd: "Toward a Better Understanding of Context and

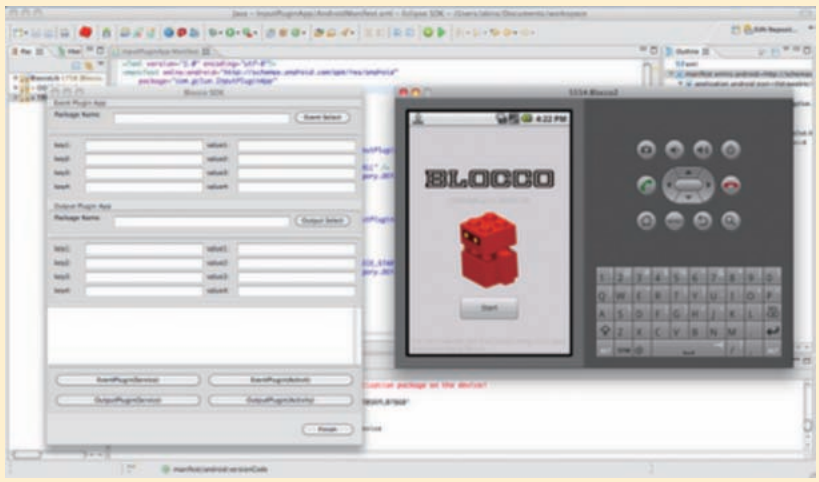


Figure 6 BLOCCO SDK

*12 **SDK**: A tool or set of tools used for software development.

Context-Awareness," Proc. of the CHI2000, 2000.
[2] "'Toggle settings' Configuration control app —Control tower for Android!— I androidnavi; Toggle Setting," Jan. 2010 (Website).
[3] Locale for Android Web page.
[4] T. O'Reilly: "What Is Web 2.0—O'Reilly Media," Sep. 2005.
[5] Plagger Web page.
[6] Y. Tanaka: "The Meme Media Architecture: IntelligentPad and its Applications," NII Information Processing, Vol.38, No.9, pp.222-231, Mar. 1997 (in Japanese).
[7] Yahoo! Inc.: "Pipes: Rewire the Web; Yahoo! Pipes."
[8] Microsoft Corp.: "Internet Explorer 8 Readiness Toolkit; Accelerators."
[9] BLOCCO Web page.
[10] Android Web page.