Building a Customizable User Interface Framework using Hyperlinks for Smart Devices

Mitsuko Sato, Eigo Okada, and Yukikazu Nakamoto

Graduate School of Applied Informatics, University of Hyogo
1-3-3, Higashi-Kawasaki-cho, Chuou-ku, Kobe 650-0044, Japan
nakamoto@ai.u-hyogo.ac.jp

Abstract. A new customizable user interface for smart devices based on hyperlink associability is presented. Although mobile devices should be easy to use, many current devices have complex and widely varying interfaces. The proposed framework, Hyrax, attempts to improve the menu structure and accessibility of functions while considering user preferences. In Hyrax, the user interface is constructed and customized using hyperlinks for access to application functions. We focus herein on the user interface of a phone and present the customizable menu structure of the phone using XLink defined in W3C and the External Function Interface (EFI) defined in the WAP Forum specifications. To implement the proposed framework, we have developed a design tool to customize the user interface with hyperlinks and a runtime environment, which manages the objects generated by the tool with the hyperlinks, to evaluate the framework.

1 Introduction

Smart devices such as information appliances and mobile devices should be convenient enough to use on a daily basis. However, many products have user interfaces that are difficult to use. For example, with current mobile phones, users are required to navigate a complex menu structure to access even simply functionalities. To overcome this inconvenience, customization mechanisms with greater flexibility are expected to be introduced to allow users to organize functionalities more easily and flexibly according to personal preferences.

Recently more sophisticated mobile phones have the ability to provide user-preferred functionalities. However, the flexibility of such functionalities is limited. Thus, the user interface should be made more customizable and flexible to better handle individual user preferences.

The fixed menu structures of smart devices force users to adapt to the user interface of each device. Mobile phone manufactures often manufacture different types of phones with different menu structures, and users may have difficulty adapting to the different interfaces. Ideally, the menu structure should be the same regardless of how often the user changes phones, that is, the user-preferred menu should be portable. In addition to portability of the menu structure among the same device class, portability among different device classes, for example,

phone and television, is expected. In this scenario, a user is able to interact with a television in the same menu as a mobile phone according to the user's personal preferences.

In the present paper, the menu structure contains menus for invoking programs as well as menus for calling a function inside the program. In this sense, the menu structure can be used to form an application program framework that determines the structure of applications. Thus, the customizability and flexibility of the menu structure affects the software structure of the device.

Some software vendors utilize XML in implementing phone functions primarily to improve the productivity of phone software [1–5]. Such vendors separate GUI functionalities implemented by XML from phone functionalities. Separating the GUI functionalities gives the following benefits. First, single phone hardware can be utilized for various phones by changing the GUI. Second, the GUI design of the phone can be performed not only by programmers but by web designers as well. Since the number of web designers is larger than the number of programmers, a productivity increase is expected.

In the present study, a customizable user interface framework based on hyperlinks is presented for use with smart devices, particularly smart phones. The proposed user interface framework, Hyrax, attempts to solve the problems of customizability and portability mentioned above. Hyperlinks are widely used throughout the World Wide Web, and can be regarded as a formulation of human associative memory. Thus, hyperlinks are well suited to the improvement of user interface functionality in smart devices. We herein focus on user customizability and framework, which enables customizability by hyperlink structure.

In a previous paper, we described the concept and the design of the hyperlink-based user interface framework [6]. In the present paper, we present design and implementation issues of the hyperlink-based customizable user interface framework.

Section 2 summarizes the requirements for the customizable user interface framework with respect to ease of use. We present a customizable user interface framework to meet these requirements in Sect. 2. To implement the proposed user framework, we have developed a design tool to customize the user interface with the hyperlinks and a runtime environment that manages objects generated by the tool with the hyperlinks. Section 4 and Section 5 present specifications and an evaluation of the design tool and the runtime environment, respectively. Conclusions are presented in Sect. 6.

2 Requirements of a Customizable User Interface Framework in Smart Phones

In the present study, we examine the requirements for a customizable user interface by examining examples of such a customizable menu structure of a smart phone. We can describe the customizability of the user interface as follows:

– To make a menu structure in the smart phone customizable.

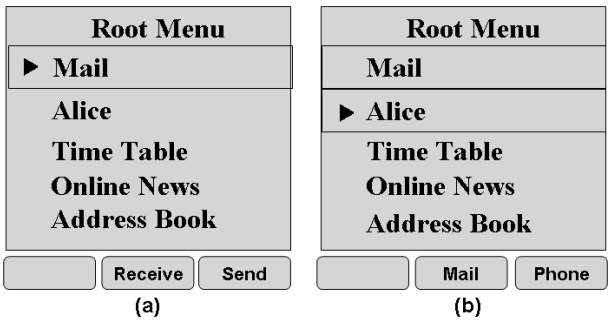

Fig. 1. Example of a customizable user interface

– To assign phone functionalities to programmable keys easily and flexibly.

Figure 1 shows an example of a menu structure of a smart phone. The large area is the main screen of the phone, and the two smaller areas are soft keys. The main screen shows the user's preferred menu. We assume that the interface has three programmable keys: two soft keys and one 'select' key, which is used for the default action. The menu shown in Fig. 1(a) is the root menu and contains five items: Mail, Alice, Time Table, Online News, and Address Book. When the first item, Mail, is selected, the programmable keys have two functions: receiving mails of the mail program, which is assigned to the center 'select' key, and changesending mails, which is assigned to the soft key on the right. Since a user often receives mails, the user assigns this function to one of the programmable 'select' keys on the Mail item in the root menu. This means that the user can initiate the two actions with only one click from the Mail menu. For the user, it is convenient to issue commands with one click, without having to navigate through multiple menus. In Fig. 1, the item 'Alice' appears because the user communicates with Alice frequently. Since the user communicates with Alice mainly by mail, the 'Mail' operation is assigned to the select key (Fig. 1(b)). Since the user also frequently accesses a timetable, an online news service, and an address book, the items Time Table, Online News, and Address Book appear in the root menu.

The framework for the above described customizable menu structure, or user interface, has the following requirements:

F1: Flexible structure and linking between menu items

An item that the user clicks is a functional item or a data item. Clicking a functional item invokes a function, and clicking a data item specifies either actual data or a directory to the data. A menu may contain both functional items and data items. In the example menu shown in Fig. 1(a), Mail is a functional item that invokes the mail function and Alice, Time Table, Online

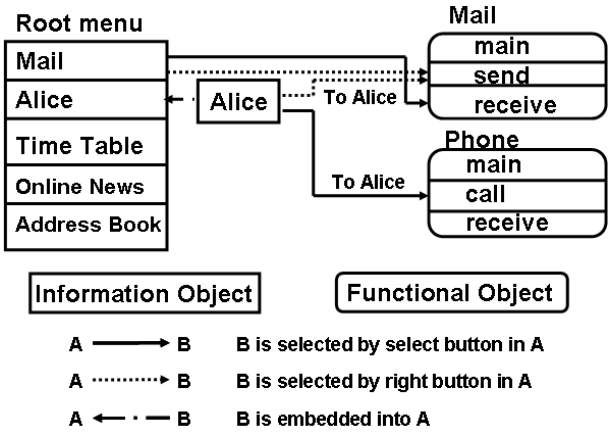

Fig. 2. Object structure for Fig. 1

News, and Address Book are data items, each of which has links to the related information.

F2: Enabling multiple operations from one item
A single menu item may have multiple operations. The example menu shown in Fig. 1(a) indicates that when a user chooses to execute Mail program or send a mail to the 'team mate', the user wants to perform the operation by a single click. Moreover, multiple operation functionality for a single item is preferable because the display size of smart devices, particularly smart phones, is limited.

F3: Enabling a customizable menu structure in the usage time
It is preferable for the user to be able to customize the user interface of the smart phone while using the phone. It is because access frequencies to menu items of functional objects and information items in information objects changes in the usage time.

3 Hyperlink-based Customizable User Interface Framework

In Hyrax, menus are linked by hyperlinks and users select functions by traversing the hyperlink. The menu structure in the user interface functions is constructed in the form of an object structure linking functional and information objects as follows.

Functional objects: Functional objects are program entities that correspond to expected functions and are implemented by sets of functions or methods.

Information objects: Information objects include menu items and links to other information objects or functional objects and are implemented in extensible markup language (XML).

Requirement **F1** is satisfied by implementing an object structure. An object structure provides menu structure for application programs and enables a consistent and unified user interface for the programs. Figure 2 shows an object structure to realize menus in the user interface shown in Fig. 1. The information object of the root menu contains four items: Mail, Alice, Time Table, Online News, and Address Book. Operations assigned to soft keys, or selection operations for items, are represented as links between objects in an object structure. The first item, Mail, has two ending objects, which correspond to the two operations 'receive' and 'send' mail functions. The Alice information object has two links to Send Mail and Phone Call and is embedded in the root menu. An embedded object is displayed simultaneously when an object that has a link to the embedded object, the root menu in this case, is displayed.

Figure 3 shows a description of the hyperlink structure in Fig. 2. Links between objects are implemented using XLink [7] because XLink has strong and flexible link mechanisms. The `itemList` tag denotes a functional or an information object, and the `item` tag denotes an item contained in the object. The `dest` tag denotes a remote object that is linked to an object defined by an `item` tag. An object defined in an `item` tag or a `dest` tag has labels with the `xlink:label` attribute[1] In the arc definition `xlink:type="arc"` in the `go` tag of (3), the `xlink:from` and `xlink:to` attributes have labels of the starting object and the ending object, respectively. The arc definition specifies a link from the starting object to the ending object.

Function invocation: A link is defined between a menu item and a function invoked from that item and is implemented using the External Functionality Interface (EFI) scheme [8]. In the EFI, a program can access functionalities inside a device through the uniform resource identifier (URI) naming scheme.

Multiple links between objects: As shown in Fig. 2, the Mail item has multiple links to remote objects. Multiple links enable Requirement **F2** to be satisfied. To implement a multiple-link object, we utilize an *extended* link in XLink. The extended link enables a link that associates an arbitrary number of linked objects. A multiple link is described as follows. Starting and ending objects are declared with the `xlink:type="locator"` and the `xlink:label` attribute. Multiple tags with `arc` values define links between the starting objects and the ending objects. For example, there are two `go` tags in (3) as multiple links. One is a link to invoke the main service in the mail server, and the other is a link to invoke the Send Mail service in the mail server with Team Mate parameter. Multiple arc definitions realize multiple links in the object structure.

[1] In this paper, an attribute in the XLink namespace has the `xlink:` prefix, e.g. `xlink:type`.

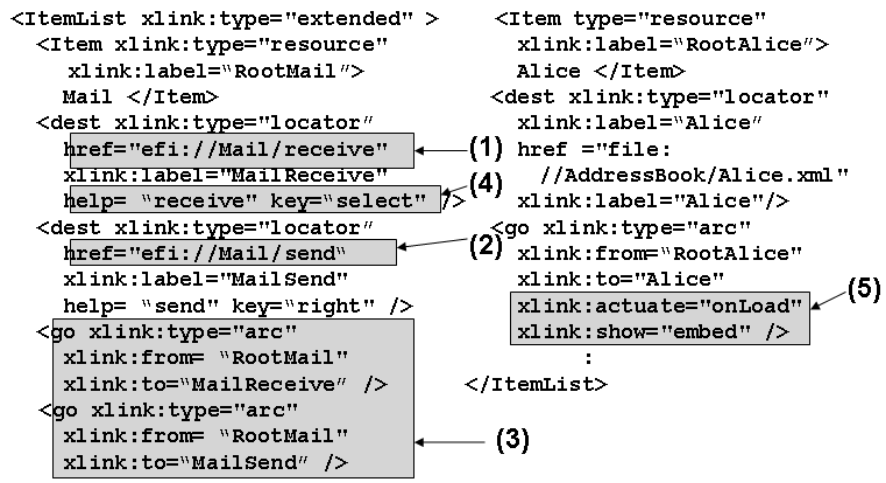

Fig. 3. XLink description for Fig. 2

Next, we consider how to deal with multiple implementation-dependent links. In Hyrax, we introduce `key` and `help` attributes for (4) in Fig. 3 to denote a key that is assigned to a link object and a string, shown at the bottom of the main screen to display its functionality.

- `key` attribute: A `key` attribute specifies a programmable key when the linked object is traversed. The value of the attribute is "left", "select," or "right," which denotes the left-hand soft key, the 'select' key of the pointing device, and the right-hand soft key, respectively.
- `help` attribute: A `help` attribute must be used with a `key` attribute. The value of the attribute is displayed at the soft key display area at the bottom of the phone screen to indicate the meaning of the programmable key.

4 User Interface Design Tool: Hyrax Builder

We have developed the Hyrax Builder to design and customize the user interface in a smart phone. The builder manages an object structure and reconfigures the structure by changing the links between functional objects and information objects manually on a PC, according to the preferences of the user. The builder generates object descriptions in the XML format from the object structure with hyperlinks. The main display of the builder is shown in Fig. 4 and contains the following panes:

Main pane (1): The main pane shows an image of the menu of the smart phone. We provide two types of menu layouts: a list layout and a picture layout. The two layouts can be exchanged with the layout change item in the menu. The main pane in Fig. 4 shows the picture layout.

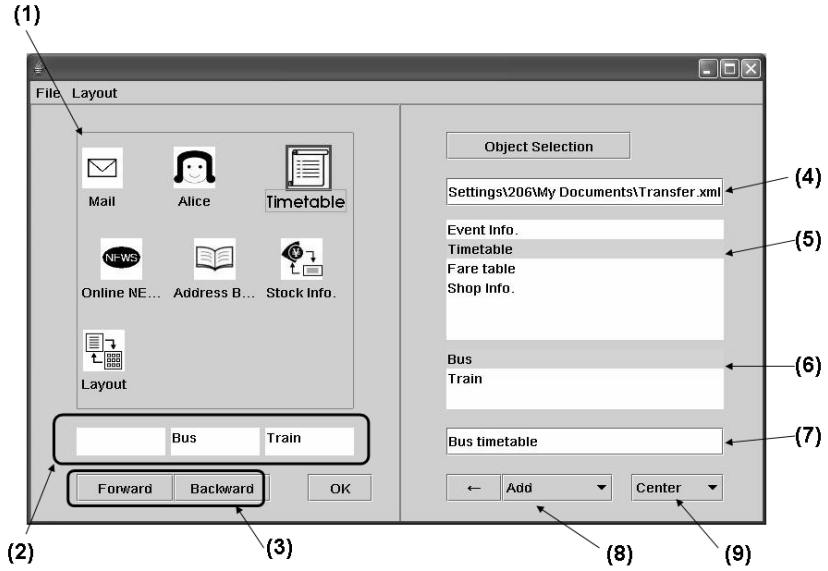

Fig. 4. Hyrax Builder

Programmable key pane (2): Help messages for the three programmable keys are displayed in the programmable pane.

Operation pane (3): The user can change the order or location of an item in the main pane using the operation pane.

Object selection pane (4): The user can select an object description file, which represents an information object and a functional object, to appear in the main pane.

Item pane (5): An object description file selected in the object selection pane may contain several items. In an object pane, a list of the items is shown.

Destination pane (6): Destinations linked from the item selected in the item pane are shown in the destination pane. The destination includes the items linked from an information object and the provided method names contained in a functional object.

Parameter pane (7): In this pane, the parameters needed for a method invocation are specified.

Set buttons (8) and (9): An item selected in the item pane is set to the selected programmable key in pane (2) with the parameters specified in pane (7).

5 Hyrax Runtime Environment

5.1 Functionalities

The Hyrax runtime environment provides the program execution environment and the framework for a customizable user interface on a target machine. The

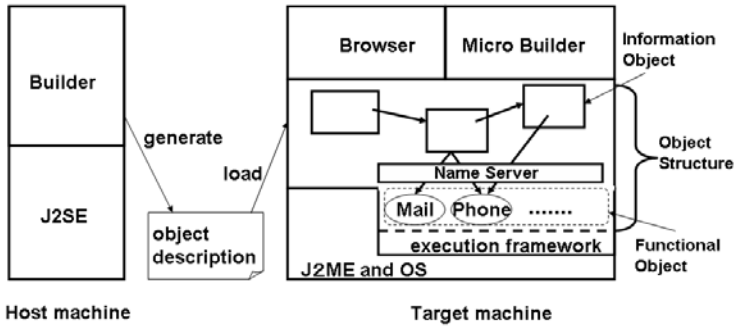

Fig. 5. Architecture of Builder and Hyrax runtime environment

Hyrax runtime environment loads the object description files and provides such a user interface. Figure 5 shows the runtime environment of Hyrax, which includes a micro browser, a builder, and an execution environment.

Browser: The browser loads the object descriptions generated by the builder and provides functionalities such as XML browsing and XML document management libraries, the application programming interface (API) of which invokes the XML parser and builds the data structure representing the XML document. The browser handles the manipulation and invocation of objects in the object structure.

Micro Builder: The Micro Builder is a runtime subset version of the Hyrax Builder and manages an object structure and reconfigures the structure by changing the links between functional objects and information objects manually or automatically, according to the preference of the user while using the phone. As an example of automatic reconfiguration, if an object is accessed frequently for communication, the object may be moved closer to an entry object for easier access. The micro builder is required to satisfy Requirement **F3**.

Hyrax Execution Framework: In Hyrax, a functional object has the following requirements. First, a functional object should have a main method that starts the execution of a program. Other methods can be exported from the functional object (from the Server in EFI terminology) and used in the specific service in the EFI scheme.

5.2 Implementation and Evaluation

We have implemented the Hyrax runtime environment, with the exception of the micro builder, in the Java programming language on a PC emulator, which is contained in the J2ME Wireless Toolkit provided by Sun Microsystems, and a smart phone Nokia 6680 as a target machine. The Java runtime environments of

Fig. 6. Picture of sample user interface of Hyrax on a smart phone

both are CLDC 1.0 and MIDP 2.0 [9]. The parser is based on kXML2 (version 2.2.2)[2]. Photographs of a sample program on the Hyrax runtime environment on the target machine are shown in Fig. 6.

In order to evaluate the Hyrax user interface framework in terms of overhead, we develop a program that shows menus and invokes programs directly without the XML document, which is called a direct implementation version. We have measured two response times: the first display time and the redisplay time of the Hyrax browser and the direct implementation version.

First display time : the period of time starting from when the object description of a menu in the memory has been processed and ending when the menu is initially displayed. The result is shown in Fig. 7(a) [3] . The display time and the increasing ratio of the display time in the Hyrax browser are larger than those in the direct implementation version. This is because the browser parses the object description by kXML parser, and links to objects from the menu items, and places the items into the form during the time.

Redisplay time: the period of time for redisplaying the display using the generated form. The result is shown in Fig. 7(b). The redisplay time and the increasing ratio of the response time in the Hyrax browser are also larger because a form in the Hyrax browser has more information. This amount of time is not considered to be lengthy in ordinary usage.

Since the Hyrax browser caches a form generated in the first display time and displays in the redisplay time, the menu display time in actual usage becomes the average of the first display time and the redisplay time.

[2] http://kxml.sourceforge.net/kxml2/

[3] In the current implementation, in order to avoid asking for user permission each file access, the browser loads all of the object description into memory. However, this implementation results in larger memory consumption at the start-up time. In order to reduce the memory consumption, we can modify the Hyrax browser, which reads object description files on-demand. The additional file access overhead are 176, 270, and 325 ms for 4, 8 and 12 items, respectively.

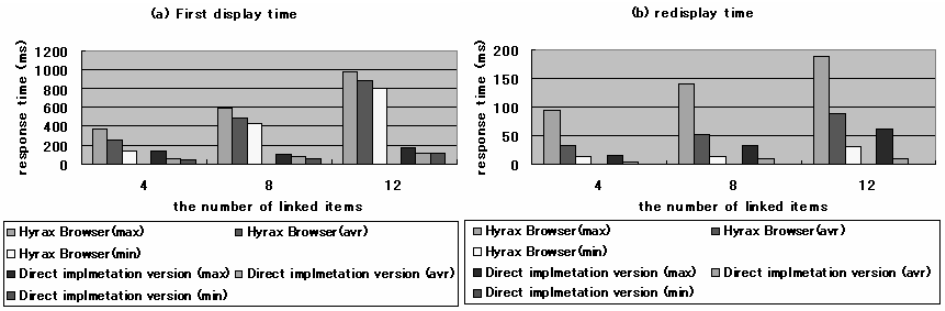

Fig. 7. Evaluation of the display time

6 Conclusions and Remarks

Hyrax is a hyperlink-based customizable user interface framework for smart devices. The architecture and implementation of Hyrax were presented in the present paper. The user interface realized by Hyrax is readily customizable according to user preferences. We have developed a design tool to customize the user interface with hyperlinks and a runtime environment that manages objects generated by the tool with hyperlinks, and evaluated the framework. We have demonstrated that the features of XLink in W3C and EFI in WAP enable user customization.

In the future, we will examine the customization pattern of the user interface using the builder. Based on this result, we will be able to prepare a template to customize the user interface more easily. Moreover, we should implement the micro builder and evaluate the usability of the customizability.

References

1. Access: Netfront Dynamic Menu (2005) http://www.access-sys-eu.com/fileadmin /user_upload/PDF_documentation ASE_NFDM_2005-02-24.pdf.
2. Microsoft: (Windows Automotive 5.0 Datasheet) http://www.microsoft.com /windows/embedded/windowsautomotive/about.mspx.
3. Qualcomm: (uiOne) http://brew.qualcomm.com/brew/en/about/uione.html.
4. UIEvolution: (UIEngine) http://www.uievolution.com/products/uiengine.html.
5. Acrodea: (VIVID UI) http://www.acrodea.co.jp/en/product_vividui.html.
6. Nakamoto, Y., Sato, M.: Design of A Hyperlink-based Software Architecture for Smart Devices. In: Proc. 9th IEEE International Symposium on Object and component-oriented Real-time distributed Computing. (2006) 261–268
7. W3C: XML Linking Language (XLink) Version 1.0. (2001)
8. Open Mobile Alliance: External Functionality Interface Framework, Candidate Version 1.1 9-Jun-2004. (2004)
9. Riggs, R., Taivalsaari, A., Huopaniemi, J., Patel, M., VanPeursem, J., Uotila, A.: Programming Wireless Devices with the Java2 Platform, Micro Edition. 2nd edn. Addison-Wesley (2003)