Human-Computer Interaction -- INTERACT'03
M. Rauterberg et al. (Eds.)
Published by IOS Press, (c) IFIP, 2003, pp. 256-263

User Interface Transformation Method for PC Remote Control with Small Mobile Devices

Hidehiko Okada & Toshiyuki Asahi

Internet Systems Research Laboratories, NEC Corporation

8916-47, Takayama-cho, Ikoma, Nara 630-0101, Japan

h-okada@cq.jp.nec.com, t-asahi@bx.jp.nec.com

Abstract: One problem with controlling remote PCs via small mobile devices such as cellular phones is the difference between PC GUIs and mobile device UIs (screen sizes, input devices etc.). To solve the problem, we propose a method for transforming PC GUIs into mobile device UIs. The essence of our method is to read GUI screens in order to obtain current PC screen information with a GUI accessibility method. The screen information data is transformed into email/HTML-formatted textual data so that users can browse the data with an emailer or a Web browser. Users can send their input operations to PCs by email replies or Web operations (hyperlink selection/form inputs). Because the PC screen data transferred to mobile devices are not images but textual data, our method requires less data transfer time and lower cost than other image-based methods. In addition, our method enables users to interact with PC applications more efficiently than others because annoying screen zoom in/out operations are not required.

Keywords: GUI, mobile device, screen reader, email, HTML

1 Introduction

PC remote control with small mobile devices such as cellular phones is a challenging topic of mobile/ ubiquitous computing. Enabling users to use data and functions stored in/served by their home/office PCs from anywhere with small mobile devices is beneficial because users can access the data/functions at any time they want without carrying heavy notebook PCs. For example, users can browse email messages, address books, schedules, documents, photos etc. stored in their home/office PCs by controlling GUI applications, installed in the PCs, that handle those data. Furthermore, users can control applications they want to keep running even while they are out.

Several systems have been proposed and developed for controlling remote PCs with cellular phones (Flexfirm, 2002; IBM Japan, 2002; Shizuki et al., 2002; Su et al., 2002) or PDAs (Harakan Software, 2001; Microsoft, 1999; Myers et al., 2001). The systems employing cellular phones enable users to remotely input GUI operations (e.g., clicks) and use PC applications without any modifications or extensions in the applications; however, there are two problems because PC screen data transferred to cellular phones are *image* data (small-sized and/or trimmed screen snapshots). The problems are as follows.

(1) Users often have to zoom PC screen snapshots in or out on their cellular phone screen because cellular phone screens are too small to clearly show the entire PC screen. These inconvenient operations make user-application interactions less efficient.
(2) Image data tends to be much larger than textual data, so PC screen data transfer time and cost greatly increase.

To solve these problems, this paper proposes a method for transforming PC GUIs into mobile device UIs. The essence of our method is to read GUI screens with a GUI accessibility method in order to obtain the data on the screens. We describe two systems employing the proposed method.

As Bouillon et al. (2002) says, two main approaches for achieving cross-platform UI are 1) transcoding dynamically performed at runtime (e.g., Bickmore et al., 1997; Bouillon et al., 2002) and 2) model-based UI reengineering/development (e.g., Eisenstein et al., 2001; Mori et al., 2003). Our method employs the runtime trancoding approach,

but the method is unique because it transcodes GUI of PC applications, not Webpages.

2 Proposed UI Transformation Method

We propose a UI transformation method shown in Figure 1 for using GUI applications in remote PCs with small mobile devices, including cellular phones. The method obtains logical and spatial information of the current PC screen (what windows and widgets are currently shown and where) by reading GUI screens. GUI screen reading is currently applied to accessibility applications for people with disabilities (Schwerdtfeger, 1991; Kochanek, 1994). For example, a system for blind users reads window titles, widget labels etc. on the current GUI screen aloud (Mynatt et al., 1994). On Microsoft Windows, the screen reading function can be implemented by using Microsoft Active Accessibility (Microsoft, 1997). The GUI screen information is called the off-screen model (OSM).

Our method generates an OSM by reading the current PC screen, encodes the OSM into textual data, and then sends the textual OSM to a mobile device. The user browses the OSM on the mobile device screen and recognizes what windows and widgets are shown in the current PC screen. After the PC receives the operations input by the user with the mobile device, mouse/keyboard events necessary for performing the user operations are generated. For example, in a case where the user inputs an operation to select a menu item in the OSM, a mouse click event for selecting the menu item is generated in the PC - the mouse pointer is automatically moved to the menu item (the location of the menu item is obtained by the preceding screen reading) and a mouse click event is generated. The method then monitors screen changes and rereads the screen to generate an OSM for the new screen.

Basically, the method enables PC remote control through the cycle of reading a PC screen, encoding and sending an OSM, receiving user inputs and generating mouse/keyboard events. However, it must be remembered that small mobile devices, especially cellular phones, have much smaller screens than PCs.

Due to the large difference in screen sizes, displaying *all* the information on the current PC screen on the mobile device screen will make it difficult for users to read and search, even if the PC screen information is encoded to textual data. Therefore, the method does not read the entire desktop screen but reads only a limited (focused) area of the screen and generates an OSM for that area only. To determine which area of the current PC screen should be read, we need to define the "if-then" rules. Figure 2 shows the rules, which are specialized here for Microsoft Windows, but the concept of screen filtering is applicable to other GUI OSs.

The "if" parts define states of the current PC screen, whereas the "then" parts define areas of the

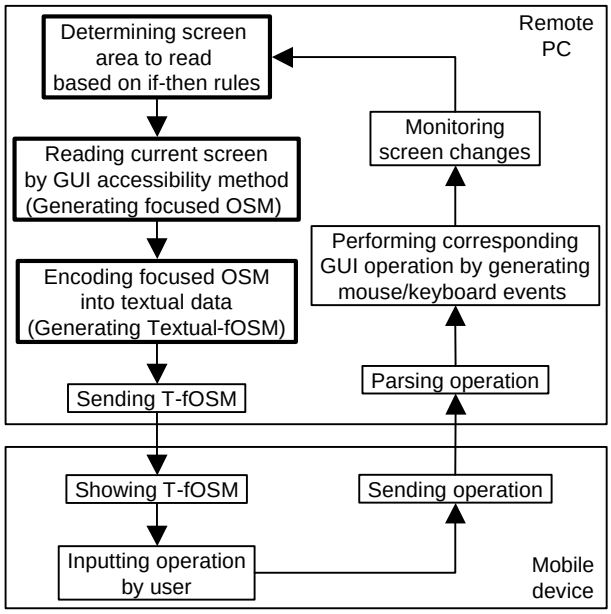


Figure 1: Proposed UI transformation method.

```
if (the login prompt is shown (the user is not yet logged in))
{
   read the login prompt window (RULE1)
}
else {
   if (a window is active) {
      if (a submenu of the active window is shown) {
         read items in the submenu (RULE2)
      }
      else {
         if (a list widget is selected by the last operation) {
            read items in the list widget (RULE3)
         }
         else {
            read widgets on the active window (RULE4)
         }
      }
   }
   else {
      if (a submenu in the Start menu is shown) {
          read items in the submenu (RULE5)
      }
      else {
         read the Start button and Taskbar icons of inactive
         windows (RULE6)
      }
   }
}
```

Figure 2: Rules for determining screen reading area.

screen to be read. For example, if a submenu of the active window or a submenu in the Start menu is shown (by the user's last operation of selecting a parent menu item or the Start button), our method reads items in the submenu (RULE2 and Rule5). In cases where two or more windows (including dialog boxes) are shown and a window is active, the method dose not read any inactive window (RULE2-RULE4). Specific examples are shown in the next section.

As described above, the method does not read the entire PC screen but reads only a focused area in the screen and sends OSM of the area. Thus, the amount of PC screen data shown in mobile device screens is reduced. In addition, the amount of data transfer is also reduced so that the method requires less data transfer time and cost. We name the partial PC screen information as a focused OSM (fOSM). "T-fOSM" in Figure 1 is the textual fOSM.

A limitation of the proposed method is that users cannot input operations for widgets that are actually shown in the current PC screen but not shown in the mobile device screen (e.g., widgets on an inactive window). We consider, however, that ease of reading and searching PC screen information on a small mobile screen is more important for users than high interaction flexibility due to the large gap in screen sizes.

3 System Implementation

This section describes how our PC remote control systems employ the proposed UI transformation method. The two systems are different with respect to their T-fOSM encoding and sending methods. One encodes T-fOSM as email messages and sends it via a SMTP server. The other encodes T-fOSM as HTML data and sends it via a HTTP server.

3.1 Email-based System

Figure 3 shows the configuration of our email-based system. Any email-enabled mobile devices can be used for this system. In contrast, other systems (Flexfirm, 2002; IBM Japan, 2002; Shizuki et al., 2002; Su et al., 2002) require Java-enabled devices because the systems use client Java applets. Consequently, older model mobile devices that are email-enabled but not Java-enabled cannot be used for other systems. The following shows examples of UI transformation by our email-based system.

In a case where the submenu in the Start menu (Figure 4(a)) is shown by the user's last Start button selection, the system reads items in the submenu (following RULE5 in Figure 2), encodes the obtained fOSM into an email message as shown in

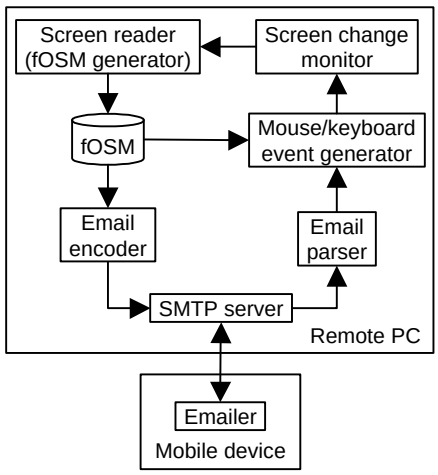


Figure 3: Configuration of our email-based PC remote control system.

(a) (b)

(X@Y means the email address of the mobile device.)

Figure 4: Example #1 of UI transformation by our email-based PC remote control system.

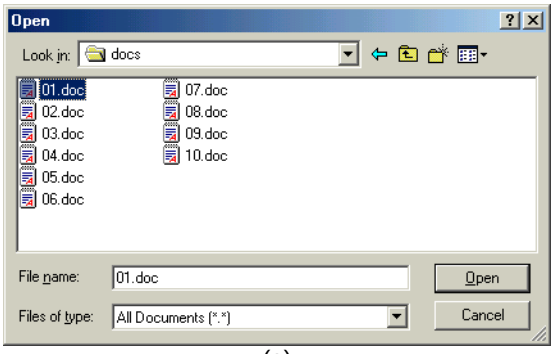
Figure 4(b), and sends the message to the mobile device via email. The user reads the received email and sends a reply message to select an item. For example, the user simply writes the number "4" in the reply message and sends it to the PC to select the menu item "Programs." The system receives the reply from the mobile device, detects that the user has selected "Programs" by parsing the message, and then clicks on the item by generating mouse events. The result of this operation is that the "Programs" submenu is shown on the PC screen. In the same manner as described above, the system next reads items in the new submenu.

In another case, where the "Open" dialog box (Figure 5(a)) is the active window, the system reads widgets on the window (following RULE4 in Figure 2) and encodes the obtained fOSM as shown in Figure 5(b). It should be noted that filenames in the listbox are not included in the message in Figure 5(b)

where the #2 item “List” corresponds to the filename listbox in Figure 5(a). The items in a list widget are read after the user has selected the list widget (see RULE3 in Figure 2). RULE3 is defined to restrict the amount of T-fOSM for cases where an active window has list widgets - there are cases where list widgets contain many items. In such cases, the amount of T-fOSM for an active window becomes too large for the user to easily read and search the T-fOSM on the mobile device screen. RULE3 is designed to prevent this problem. Items in list widgets are shown on the mobile device screen only when the user needs to see the items. Figure 5(c) shows the message sent to the mobile device after the user selects the filename listbox in Figure 5(a), i.e., after the user sends a message (as a reply to the message in Figure 5(b)) in which the user writes “2.” For cases where too many items are contained in a list widget, the system reads and sends only M items at a time, where M is a predefined threshold (e.g., M = 20).

The user can input two or more consecutive GUI operations on the PC screen with a single reply message. For example, the user can type “03.doc” in the “File name:” field and click on the “Open” button in Figure 5(a) by sending the message shown in Figure 6. When the user inputs such multiple consecutive operations with a single message, the user should not forget that the target widget for another operation needs to be still shown on the PC screen after the last operation was completed. If the target widget for an operation vanishes (e.g., the parent window of the target widget is closed) because of the last operation, the system cannot find the widget to operate. In such cases, the system sends a message informing the user that the target widget could not be found. The message also contains the T-fOSM for the current screen; therefore the user can continue remote interactions, even after such widget-loss errors have occurred.

The next example demonstrates a case where an editor opens a document. Figure 7(a) shows a document-editing window. Using the copy-and-paste method, our email-based system obtains textual data from application data shown in an application window, i.e., by copying the application data into the



(a)

```
To: X@Y
Subject: Open
---
1: Look in:
 docs
2: List
 01.doc[selected]
3: File name:
 01.doc
4: Files of type:
 All Documents (*.*)
5: Open
6: Cancel
7: Go To Last Folder Visited
8: Up One Level
9: Create New Folder
10: View Menu
```

(b)

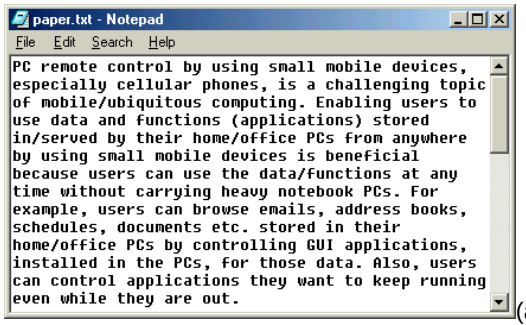
```
To: X@Y
Subject: List
---
1: 01.doc[selected]
2: 02.doc
3: 03.doc
4: 04.doc
5: 05.doc
6: 06.doc
7: 07.doc
8: 08.doc
9: 09.doc
10: 10.doc
11: Back
```

(c)

Figure 5: Example #2 of UI transformation by our email-based PC remote control system.

```
3
[03.doc]
5
```

Figure 6: Message example for “multiple consecutive operations by a single message”.



(a)

```
To: X@Y
Subject: Paper.txt - Notepad
---
1: File(>)
2: Edit(>)
3: Search(>)
4: Help(>)
---
PC remote control by using
small mobile devices, especi
ally cellular phones, is a
challenging topic of mobile/
ubiquitous computing. Enabli
ng users to use data and fun
```

(b)

Figure 7: Example #3 of UI transformation by our email-based PC remote control system.

buffer (the Windows Clipboard) and extracting textual data from the Clipboard. The data is output to an email message and sent to the mobile device so that the user can read the textual data in the document (Figure 7(b)). A limitation of the email-based system is that the user cannot see images displayed on the PC screen (e.g., photos shown by a image browser) because the system handles textual data only. In contrast, our Web-based system described in the next subsection can handle both textual and image data.

To support users' routine tasks, operation macro recording and executing functions have been developed for the system. Our operation macro recorder records a sequence of a user's GUI operations. The operation macros can be remotely activated as follows. The user first sends an email message to a predefined address for operation macros. When the system receives the message, it returns a numbered list of available macros. The user then selects a macro and sends the selected macro's corresponding number. After the system completes the recorded sequence of operations in the selected macro, the system sends the completion message.

3.2 Web-based System

Figure 8 shows the configuration of our Web-based system. The system works with a HTTP server via the CGI interface. Any Web-enabled mobile devices, even older model ones that are not Java-enabled, can be used for this system. The following shows examples of UI transformation by the Web-based system.

In the case where the submenu in the Start menu (Figure 4(a)) is shown, the system reads items in the submenu (following RULE5 in Figure 2) and encodes the obtained fOSM into HTML-formatted data. The formatted T-fOSM can then be browsed with the Web browser in the mobile device (Figure 9). As this example shows, menu items shown on the PC screen are transformed to Web hyperlinks (i.e., <a> tags in the HTML-formatted T-fOSM). The user reads the browser screen and selects a hyperlink such as the "Programs" hyperlink, then the system receives the hyperlink selection and clicks on the "Programs" menu item that in turn brings up the "Programs" submenu. The Web-based system next reads items in the new submenu.

In another case where the "Open" dialog box (Figure 5(a)) is the active window, the system reads widgets on the window (following RULE4 in Figure 2) and encodes the obtained fOSM into HTML-formatted data. The formatted T-fOSM can then be browsed with the Web browser (Figure 10). The

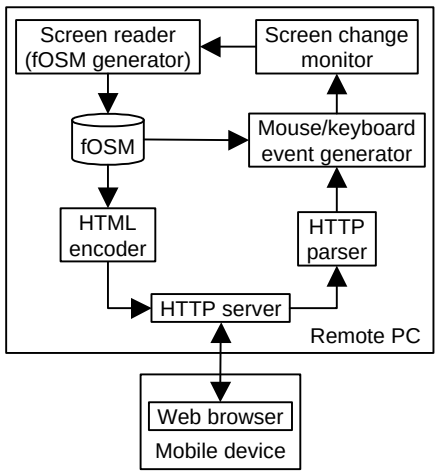
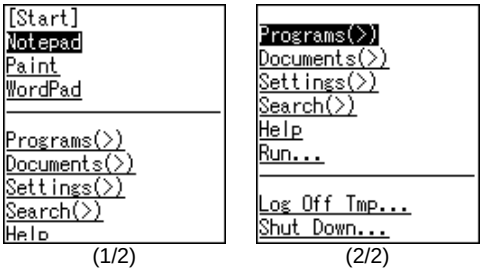
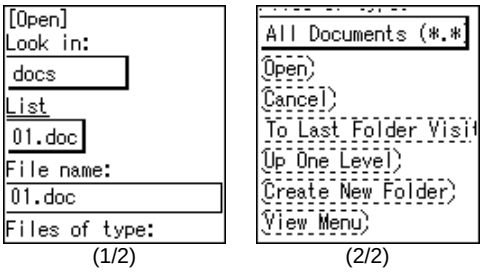


Figure 8: Configuration of our Web-based PC remote control system.



(1/2) (2/2)

Figure 9: Example #1 of UI transformation by our Web-based PC remote control system.



(1/2) (2/2)

Figure 10: Example #2 of UI transformation by our Web-based PC remote control system.

filename listbox is treated in the same way by both the Web-based system and the email-based system. The user reads the browser screen and makes Web form inputs. For instance, the user types a filename in the "File name:" field (Figure 10, 1/2) and selects the "Open" button (Figure 10, 2/2) on the browser screen. The system receives the form inputs and performs corresponding GUI operations on the PC screen by generating keyboard and mouse events.

As these examples show, the user can input remote GUI operations to the PC by Web operations (hyperlink selections and form inputs). GUI widgets on the PC screen (e.g., buttons, listboxes, checkboxes and radio buttons) are transformed into

Web form objects that look similar, thus the Web-based system enables the user to input remote operations more intuitively than the email-based one.

Another advantage of the Web-based system is that it can handle image data. Suppose image data is displayed in an image editor window (Figure 11). The system obtains the image by the same data copy-and-paste method as the email-based system obtains the textual application data shown in an application window, reduces the color depth of the image, splits and shrinks the image into smaller ones, and transcodes them into JPEG formats. The user can see the split and shrunk images with the mobile device's Web browser (Figure 12). The maximum color depth and the maximum width/height of split-and-shrunk images are predefined for the current system, but we will extend the system to adaptively set these parameters based on the capability of the mobile device accessing the PC at the time.

With the Web-based system, the user can browse the image of an active window/the entire desktop by the same method used for browsing images shown in an application window. This will be helpful when the user wants to see the images during remote interactions with this system. Furthermore, the Web-based system employs the same operation macro function as the email-based one; the user can browse a hyperlink list of available macros by accessing a predefined URL, and activate a macro by selecting the corresponding hyperlink in the list.

4 Example of Remote Task

An example of remote interaction sequence for a specific task is shown in Figures 13 and 14. In this example, a user remotely interacts with a PC emailer (Microsoft Outlook Express) by using the Web-based system in order to receive new incoming email. The user first accesses his/her remote PC by accessing a starting page whose URL is sent by the system to the mobile device via email. In the example, steps from initial access to user authentication completion are skipped. In addition, this example supposes that the initial state of the PC screen matches RULE6 in Figure 2; no active window (but two inactive windows "Untitled - Notepad" and "untitled - Paint") and no submenu in the Start menu are shown.

5 Discussion

In this section, we discuss remote tasks for which our systems are applicable, and future work.

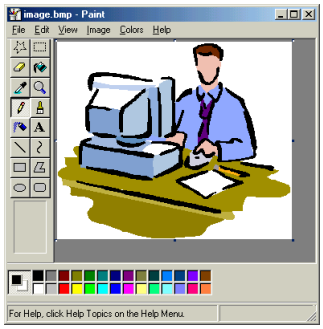
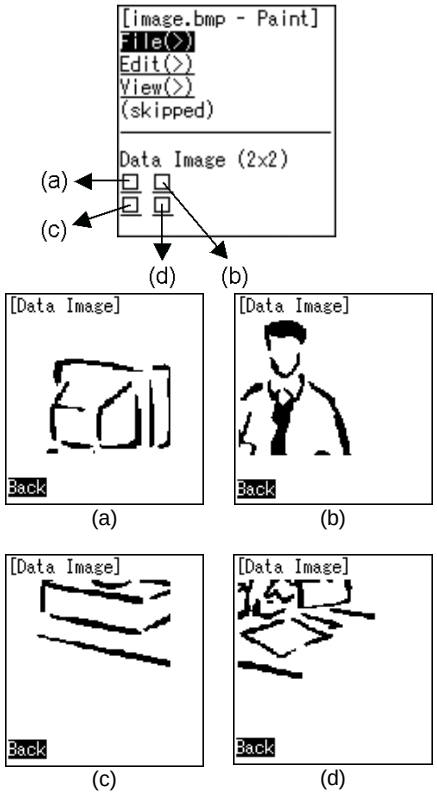


Figure 11: Application window that shows image data.



(Some widgets on the window in Figure 11 are omitted to save space)

Figure 12: Example #3 of UI transformation by our Web-based PC remote control system.

5.1 Applicable Remote Tasks

As the examples described in Sections 3 and 4 show, GUI operations that can be remotely performed with our systems are the left mouse clicks on GUI widgets and the textual string inputs to text input fields - some GUI operations such as drag-and-drops are not supported. In addition, some application-specific data is difficult to read and transform into email/HTML-formatted textual data. A typical example is a presentation slide shown in a window of a presentation slide editor (e.g., Microsoft

Step 1	PC screen: no active window (but two inactive windows "Untitled - Notepad" and "untitled - Paint") and no submenu in the Start menu are shown. Rule: RULE6. Mobile device screen: Figure 14(1). User operation: selecting the “Start” hyperlink.
Step 2	PC screen: a submenu in the Start menu is shown. Rule: RULE5. Mobile device screen: Figure 14(2). User operation: selecting the “Outlook Express” hyperlink.
Step 3	PC screen: the main window of the emailer is shown and active. Rule: RULE4. Mobile device screen: Figure 14(3). User operation: selecting the "Tools(>)" hyperlink.
Step 4	PC screen: the submenu of the selected “Tools” menu item is shown. Rule: RULE2. Mobile device screen: Figure 14(4). User operation: selecting the “Send and Receive(>)” hyperlink.
Step 5	PC screen: the submenu of the selected "Send and Receive" menu item is shown. Rule: RULE2. Mobile device screen: Figure 14(5). User operation: selecting the "Receive All" hyperlink.
Step 6	PC screen: the “Logon - localhost” window is shown and active, where the string “localhost” in the window title depends on the POP3 server name. Rule: RULE4. Mobile device screen: Figure 14(6). User operation: Filling the POP3 password in the “password” field and select the “OK” button.
Step 7	PC screen: the received password is entered in the corresponding “password” field on the “Logon - localhost” window and the “OK” button on the same window is clicked on. As the result, new messages are received. The first message is shown in the message area of the main window. Rule: RULE4. Mobile device screen: Figure 14(7). Note that this figure shows the page that is scrolled down to the area in which the message is shown. The top of this page is similar to Figure 14(3). User operation: for example, selecting the next new message in the inbox.

Figure 13: Example of remote interaction with a PC emailer.

PowerPoint). Our Web-based system does not allow the user to edit objects in the slide; it only supports browsing of the slide via snapshot images, nor do our systems support user operations for widgets that are actually shown on the current PC screen but not shown on the mobile device screen due to the rule-based filtering. For example, user interactions using two or more windows simultaneously are not supported by our systems (this limitation depends on the design of screen filtering rules in Figure 2).

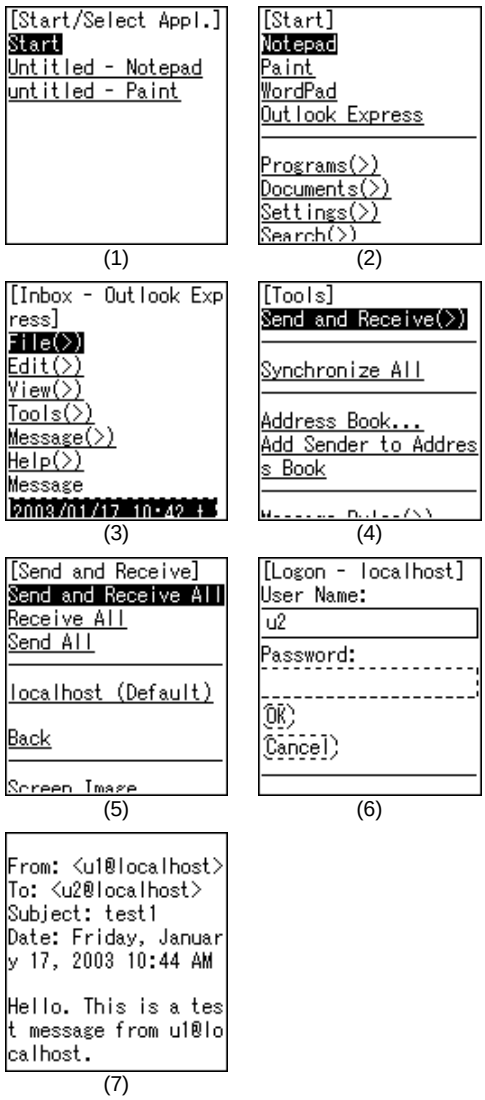


Figure 14: Example of mobile device screenshots for remote interaction with a PC emailer.

Therefore, it can be said that user remote tasks for which our systems are applicable are essentially those that can be completed by interactions with menus and dialog boxes. However, some actual remote tasks, such as those shown in Section 4, can be supported by our systems. For those tasks, our systems are superior to other image-based systems (Flexfirm, 2002; IBM Japan, 2002; Shizuki et al., 2002; Su et al., 2002) from the perspective of interaction efficiency and data transfer time and cost.

5.2 Future Work

Future work includes evaluating the effectiveness of the transformed UIs. By testing the systems with real users, we will determine whether users can easily understand fOSMs shown on the mobile device screen and complete their tasks. We will evaluate the usability of the email/Web-based systems based on user task performance measures such as the task time (not including data transfer time), the number of incorrect user operations (operations unnecessary for the task) and users' subjective satisfaction. We will enhance our method and systems based on the results.

We plan to develop methods for user/device adaptive UI transformations in future work. The methods detect user attribute values (e.g., frequent applications/tasks/interactions in his/her PC daily usage) and/or mobile device attribute values (e.g., screen sizes, input devices, available functions/data formats, network transfer speed) and adaptively transform PC GUIs to mobile device UIs based on the detected attribute values. Such methods will enhance the usability of our systems for various types of users and mobile devices.

In addition, security is an important issue for every remote control system. Our current systems filter out unauthorized access by methods that include user password authentication. Employing other advanced security methods will enhance our systems.

6 Conclusion

A UI transformation method was proposed for PC remote control with small mobile devices including cellular phones. The essence of our method is to read GUI screens in order to obtain current PC screen information by a GUI accessibility method. GUI screen information obtained by screen reading is encoded into textual data. We have developed two systems, email- and Web-based, that employ the proposed method, and these systems enable more efficient and lower cost remote PC use compared with other image-based systems.

References

Bickmore, T. W. & Schilit, B. N. (1997), Digestor: device-independent access to the World Wide Web, *Proceedings of 6th WWWC*, 655-663. http://www.fxpal.com/publications/PR/PR-97-149/FXPAL-PR-97-149.html

Bouillon, L. & Vanderdonckt, J. (2002), Retargeting Web pages to other computing platforms with VAQUITA, *Proceedings of WCRE2002*, 339-348.

Eisenstein, J., Vanderdonckt, J. & Puerta, A. (2001), Model-based user-Interface development techniques for mobile computing, *Proceedings of IUI2001*, 69-76.

Flexfirm, Inc. (2002), x-Remocon, http://www.flexfirm.co.jp/products/remocon/ (in Japanese).

Harakan Software (2001), PalmVNC: virtual network computing client for Palm platform, http://www.harakan.btinternet.co.uk/PalmVNC/.

IBM Japan (2002), Desktop On-Call Gateway, http://www-6.ibm.com/jp/pspjinfo/javadesk/news_dtocg.html (in Japanese).

Kochanek, D. (1994), Designing an OffScreen model for a GUI, *Lecture Notes in Computer Science*, 860, 89-95.

Microsoft Corporation (1997), Microsoft Active Accessibility, http://msdn.microsoft.com/library/en-us/msaa/msaastart_9w2t.asp.

Microsoft Corporation (1999), Windows 2000 Terminal Services, http://www.microsoft.com/windows2000/technologies/terminal/.

Mori, G., Paterno, F. & Santoro, C. (2003), Tool support for designing nomadic applications, *Proceedings of IUI2003*, 141-148.

Myers, B., Peck, C.H., Nichols, J., Kong, D. & Miller, R. (2001), Interacting at a distance using semantic snarfing, *Proceedings of Ubicomp2001*, 305-314.

Mynatt, E. & Weber, G. (1994), Nonvisual presentation of graphical user interfaces: contrasting two approaches, *Proceedings of CHI94*, 166-172.

Schwerdtfeger, R.S. (1991), Making the GUI talk, *BYTE*, December, 118-128.

Shizuki, B., Nakasu, M. & Tanaka, J. (2002), VNC-based access to remote computers from cellular phones, *CSN2002: Proceedings of the IASTED International Conference on Communication Systems and Networks*, 74-79.

Su, N.M., Sakane, Y., Tsukamoto,M. & Nishio, S. (2002), Rajicon: remote PC GUI operations via constricted mobile interfaces, *Proceedings of MOBICOM2002*, 251-262.