Adaptive User Interfaces Based on Mobile Agents: Monitoring the Behavior of Users in a Wireless Environment*

Nikola Mitrović	Jose Alberto Royo	Eduardo Mena
IIS Department	IEC Department	IIS Department
University of Zaragoza	University of Zaragoza	University of Zaragoza
Maria de Luna 1	Maria de Luna 1	Maria de Luna 1
50018 Zaragoza, Spain	50018 Zaragoza, Spain	50018 Zaragoza, Spain
mitrovic@prometeo.cps.unizar.es	joalroyo@unizar.es	emena@unizar.es

Abstract

Adapting user interfaces to meet users' context and preferences is one of the most challenging questions of mobile computing. In the mobile world, every application must be aware of the different user devices, application platforms and contexts where it should execute; the alternative is reimplementing the same user interface to meet different device capabilities.

This paper presents a proposal based on mobile agents that transparently adapts user interfaces to the corresponding device capabilities and monitors the user behavior using an *indirect user interface generation* mechanism. User behavior patterns are used to predict users' next most probable action. We apply this approach to a software retrieval service to show its feasibility.

Keywords: Adaptive user interfaces, mobile computing, mobile agents and multiagent systems

1 Introduction

Adapting graphical user interfaces (GUIs) to the user device capabilities is one of the most challenging issues in mobile computing as devices have different processing power and GUI capabilities. Another important challenge is to adapt the GUI to the user preferences, which requires user behavior analysis and reuse of knowledge.

In order to create adaptive user interfaces, researchers use an *abstract user interface definition language* as a common ground. There are many abstract user interface definition languages: XUL [3], UIML [1], XIML [6], XForms [20], etc. These languages are designed to provide an abstraction layer that would provide a basis for contemporary user interfaces.

From the architectural perspective, some researchers use a client-server architecture [7] for generating user interfaces, some provide tools that create separate interfaces for different platforms [15] and some focus on mobile agent technology [13, 10].

In this paper we present the *ADaptive User Interface System (ADUS)*, a system designed for indirect generation of user interfaces. We have adopted a mobile agent architecture and an abstract user interface definition language – XUL (eXtensible User interface definition Language [3]). The developed prototype adapts a user interface definition to Java AWT, Java Swing, HTML and WML clients and supports limited plasticity [19]. In addition, our architecture monitors user behavior to acquire knowledge about the user and reuse it in future execution. When creating the user interface, our system considers user preferences, contextual information, monitors user behavior and abstracts device properties. Next most proba-

*Supported by the CICYT project TIN2004-07999-C02.

ble user action is derived from behavioral data and this is advertised to the user.

The rest of this paper is as follows. In Section 2 we give an overview of the technology behind our proposal. In Section 3 we detail our solution to generate adaptive GUIs and monitor the user behavior. In Section 4 we show the feasibility of our proposal by applying it to a multiagent system. Section 5 gives an overview of the related work. Finally conclusions and future work are presented in Section 6.

2 Generating User Interfaces with Mobile Agents

In our prototype we use XUL and mobile agents in order to specify user interfaces. We have adopted this approach [14] as it enables the description of a flexible user interface that is able to adapt and move through the network; it also enables the analysis of the user behavior.

2.1 XUL: eXtensible User-interface Language

The eXtensible User interface Language [3, 9] is designed for cross-platform user interface definition. This language is part of the Mozilla project [16].

XUL lacks the abstraction layer of interface definition, and is restricted to window-based user interface. It is capable of referencing Cascading Style Sheets (CSS) to define the layout of elements. User actions, property access and functionality can be stored in JavaScript (ECMAscript) [5] files. Although there are several similar UI definition languages, we found XUL to be a suitable open source solution for our purpose.

2.2 Mobile Agents

A mobile agent is a program that executes autonomously on a set of network hosts on behalf of an individual or organization [12]. Mobile agents execute in contexts denominated *places*. A mobile agent is able to pause its execution, travel from one place to another, and once there it resumes its execution.

We assume a mobile agent architecture because software agents can easily adapt their behavior to different contexts. Mobile agents are able to arrive at the user device and show their GUIs to the user in order to interact with her/him. Mobile agents can be hosted by platforms that support different models of user interfaces or have different processing capabilities. Agents are autonomous, and can handle network errors (unreachable hosts, etc.) independently. Also, they can move to the target device instead of accessing such a target device remotely. Agents can be sent to a home computer supporting Java. Also, an agent can play the role of a proxy server for a wireless device, such as mobile telephone or a web terminal; in that case it should produce the adequate GUI (WML or HTML, respectively, for the previously cited devices).

In general, agents do not, by themselves, constitute a complete application. Instead, they form one by collaborating with other agents.

2.3 Predicting the User Behavior

Predicting the user behavior is a difficult task. There are several approaches such as: 1) WebTango[8], that collects user interface metrics for a web site; 2) Predictions of the user behavior based on Markov chains [4]; and 3) Longest Repeating Subsequence (LRS) [17] and Information Scent [2] that perform data mining seeking to analyze navigation path based on server logs, similarity of pages, linking structure and user goals.

In our approach the user agent (see Section 3.1) monitors the user behavior and could make use of any of the above approaches to predict the user behavior.

3 ADUS: ADaptive User interface System

The architecture of the system (shown in Figure 1) is based on the use of the Client/Intercept/Server model [18] and the incorporation of modules and agents both at the wireless devices and at intermediary elements (also called *proxies*) located at the fixed network.

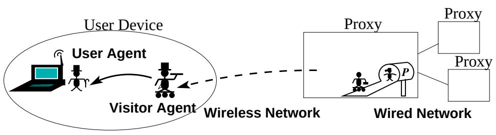


Figure 1: Motivating architecture.

ADUS is our proposal to create adaptive user interfaces for *visiting agents* using *user agents* to customize such user interfaces according to user preferences and device capabilities, and allow the monitoring of user interactions.

From the point of view of GUI generation, user interface rendering is a complex task. Development of contextual GUIs for mobile applications has the following problems:

- *Adaptation of user interfaces*: The visitor agent must adapt the creation of the user interface to the user preferences and user device capabilities. For example, the user can prefer thumbnails rather than full size images. Visitor agents are not aware of users' context, device capabilities, plasticity [19], etc. In addition, user preferences could change during the execution of the application.

- *Monitoring user interfaces*: If the user behavior is monitored, the user agent could use the data of previous executions to automatically assign initial values to the GUI of future visitor agents that request the same information [14].

3.1 Indirect User Interface Generation

In [14] we can find a comparison of the different approaches to perform an indirect user interface generation. In the following we explain the multiagent solution used in our system (see Figure 2):

- *The visitor agent*: It is a mobile agent that brings a service requested by the user to the user device. This agent is able to generate a XUL [9] specification of the GUI that it needs to interact with the user. Such a XUL specification is sent to the user agent on the user device.

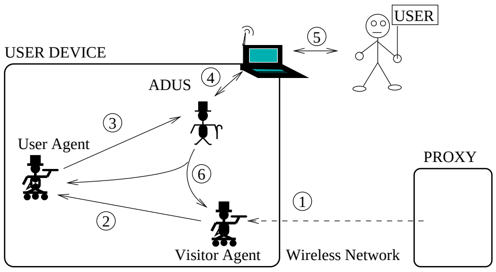


Figure 2: Indirect generation of GUIs

- *The user agent*: It is a highly specialized personalization agent that is responsible for storing as much information about the user computer, and the user herself/himself, as possible. For example, it knows the look and feel preferences of the user, the kind of GUI preferred by the user or imposed by the user device or the operating system. The main goals of this agent are: 1) To proxy the generation of user interfaces, 2) To help the user to use the services of the visitor agent, 3) To modify the GUI specification of the visitor agent according to the user preferences and device capabilities, 4) To create an ADUS agent initialized with the GUI specification, and 5) To monitor user interactions by receiving such an information from the ADUS agent.

- *The ADUS agent*: The main features of this agent are: 1) To adapt the user interface to the user preferences and device capabilities, following the user agent suggestions, 2) To generate GUIs for different devices according to XUL specifications, and 3) To handle the GUI events and to communicate them to the visitor agent as well as to the user agent.

In Section 4.2 we show an example of cooperation among the above agents. Thus, application developers define user interfaces using XUL only once and this specification is rendered transparently on various platforms.

3.2 The Learning Process

We would like to stress the relevance for the system of monitoring the user interaction with visitor agents. By knowing the user reactions and data entered to those services, the user agent can store such data locally and apply different artificial intelligence techniques to extract knowledge about the user behavior. For example, in the context of a currency converter service that executes on the user device, the user agent could set (in the XUL specification of the visitor agent) US Dollars and Euros as the initial and target currencies, respectively, if that was the selection of the user during the last execution of that service. Even if the user selects now another configuration, the user agent could learn and improve its behavior for the next time. Thus, the customization of GUIs can become really useful for the user, as the user agent is able to monitor, store and analyze her/his interactions with all the GUIs/applications.

4 Empirical Evaluation

In this section we adapt a multiagent application, the *Software Retrieval Service (SRS)* [11] and compare its performance with and without using the ADUS approach. The SRS tries to solve some of the most frequent tasks of a computer user: to search, download and install a new software.

We first explain the agents that take place in the SRS architecture, second we describe how the ADUS approach is applied to the SRS, and then we show the empirical results of our comparison tests.

4.1 The Software Retrieval Service: Multiagent Architecture

In this section we briefly present the SRS [11]. This service is situated on a concrete server of the wired network that we call *proxy*[1]. The SRS incorporates one agent place on the user device called the *User place*, and another on the proxy, called the *Software place* (see Fig. 3). In the following we summarized the main steps of the SRS functionality:

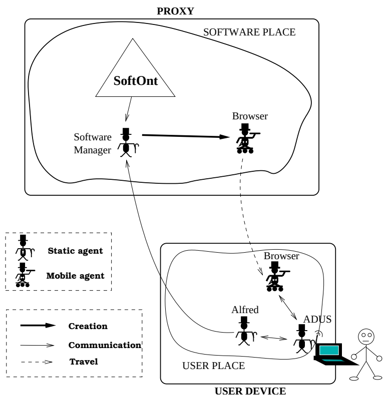


Figure 3: Main architecture for the SRS

1. The user communicates with *Alfred* the need of getting a new piece of software. The Alfred agent is an efficient majordomo that serves the user and is on charge of storing as much information about the user computer, and the user her/himself, as possible. Alfred communicates with the *Software Manager* agent, at the proxy providing the user device with coverage.

2. The Software Manager is capable to obtain customized metadata about the needed software, according to the needs expressed by Alfred (on behalf of the user), by consulting a software ontology (*SoftOnt*). Then the Software Manager creates and provides the *Browser* agent with a catalog of the available software that fulfil the specified requirements.

3. The Browser travels to the user device and presents its customized software catalog to the user in order to interact with him. As result of this interaction process,

[1] It provides connectivity and services to wireless users.

a piece of software will be selected by the user, and later it will be downloaded and installed.

Working in this way, the Browser agent directly generates its GUI on the user device without knowing the user preferences and the user device capabilities.

4.2 Using ADUS with the Software Retrieval Service

By applying the definitions of Section 3.1, in the SRS Alfred plays the role of user agent and the Browser agent behaves as a visitor agent that arrives to the user device with the purpose of creating a GUI. Then an ADUS agent is needed to intermediate in the GUI generation. The ADUS agent interacts with the SRS as follows:

1. Instead of generating the GUI by itself, the Browser agent sends to Alfred the XUL specification of the GUI it needs.

2. Alfred modifies the XUL specification of the GUI according to the user preferences and user device capabilities. In the example, the size and location of "split panes" is set by Alfred (see Figure 4).

3. Alfred delegates the generation of the GUI on the ADUS agent, who creates the GUI needed by the Browser to interact with the user and listens to the GUI events. In the example, the ADUS agent generates a Java Swing GUI supported by the PDA of the user (see Fig. 5).

4. GUI events and data received by the ADUS agent as result of user interaction are communicated to Alfred and the Browser agent for further processing. Alfred stores such data to predict future user actions, and the Browser reacts to the selections or data entered by the user by generating a new GUI according to such user's actions.

In our prototype, in order to avoid that visitor agents creates GUIs directly, only the ADUS agent has the necessary (Java) permissions to create windows and widgets.

```
<panel name="panelPrincipal" columns="1" weightx="1"
   scrollable="false">
   <splitpane name="splitPanel1"
      orientation="horizontal" divider="200"
      weightx="1">
    <panel scrollable="true">
    <label name="graphPanel" icon="/GUIs/browsing.gif"
      action="click(X,Y,clickCount,popupTrigger,id)"
      weightx="1" weighty="1"/>
    </panel>
    <panel name="panelPrograms" columns="1" gap="0"
      weightx="1" weighty="1">
    <splitpane name="splitPanel2" orientation="vertical"
      divider="50" weightx="1" weighty="1">
     <tree name="programs" selection="single" weightx="1">
     </tree>
     <splitpane name="splitPanel3" orientation="vertical"
       divider="50" weightx="1" weighty="1">
     <textarea name="programDescription" wrap="true" text=" "
       weightx="1" weighty="1"/>
     <list name="programAttributes" selection="multiple"
       weightx="1">
     </list>
     </splitpane>
    </splitpane>
    <label name="download"/>
    </panel>
   </splitpane>
</panel>
```

Figure 4: (Partial) XUL description of the GUI of the Browser agent

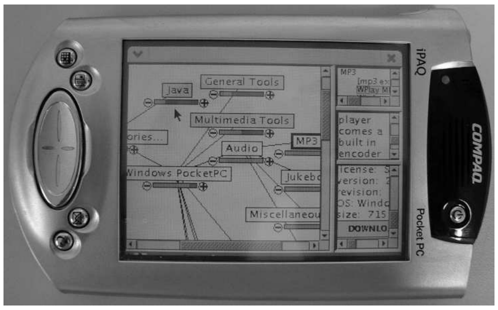


Figure 5: The Browser agent GUI

4.2.1 The learning process in the SRS

Behavior analysis and learning are provided by the user agent, Alfred, which treats user preferences and predicts the user behavior following the patterns stored (e.g. when expanding the nodes of the software catalog). This agent makes the necessary decisions that are later reflected on the user interface.

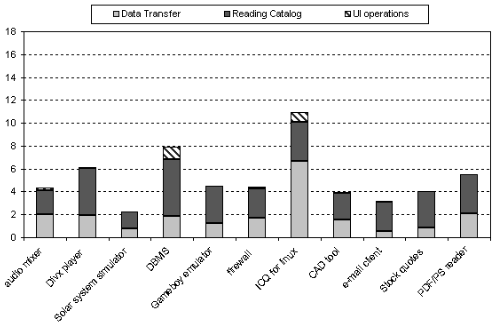
In our prototype Alfred uses the Longest Repeating Subsequence (LRS) model [17] to predict the user behavior. The application needs to be trained, as predictions are based on past execution of the service. Once users start using the application, Alfred collects the necessary data (the system monitors the actions executed by the user) to try to predict the next hop, i.e., Alfred stores which are the methods that have been triggered on each widget. In addition, Alfred is able to establish relationships among the initial set of keywords entered by the user and the piece of software finally selected, so it can show these pieces of software in a toolbar allowing a direct download; usually the pieces of software that appears in the toolbar are software upgrades or software that the user installed some time ago. The reason to show this software is to provide help in upgrading software and compatibility with older versions.

However, learning is not just limited to data collected by Alfred from a single user: it collects the usage data (for the same application) and creates one unique usage log that is sent to the proxy as default knowledge. This is the knowledge of Alfred in the first execution of a service by a new user. This process could lead to less efficient initial predictions, because each user can follow different browsing pattern. However, this also helps in increasing overall user expertise, since users with more expertise could influence predictions by supplying better sequences to naive users. Therefore, Alfred provides better predictions for users with less expertise and show them how to use application more efficiently.

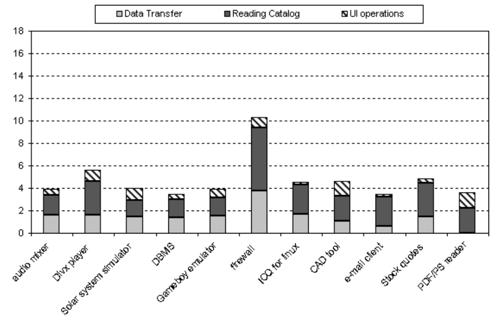
4.3 Performance Evaluation

In this section we present some performance results that explain the advantages of using the Adaptive User Interface System architecture. Testing users retrieved several pieces of software, first with the SRS that generates GUI's directly and then with the version that uses ADUS. Data were obtained after testing both methods by different kinds of end-users (46 users in total).

In Figure 6 we show that the communication cost due to the indirect GUI generation is very low in comparison with the time consumed by the browsing and data transfer tasks: only the UI operations have been increased slightly. However, the advantage of the ADUS approach is that the system automatically adapts the GUI of visitor agents to the user preferences and device capabilities and allows Alfred to monitor the user behavior.



(a)



(b)

Figure 6: Time-consuming tasks for SRS (a) without and (b) with ADUS

In Figure 7 we show how the prediction features of the SRS + ADUS help expert and naive users to find the wanted software. We detail the percentage of predictions that are just not followed by the user (prediction ignored), those that guide the user in a wrong direction (wrong prediction) and those that do show the user a quick way to find the wanted software (right prediction). Notice that the percentage of right predictions obtained in the tests is significant.

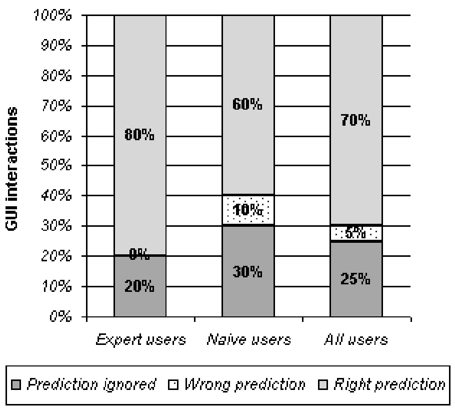


Figure 7: Predicting the user behavior

5 Related work

In this section we present several approaches that are related with the work presented in this paper.

5.1 Adaptive user interfaces

Various approaches to adapting user interfaces to device capabilities are present. Generally, the approaches are grouped into two categories: web applications and classic desktop applications.

In the context of web application, they are mostly oriented on how to transform web contents to various other formats that can be used on mobile devices —cHTML, WML, etc. However, different approaches exist. Microsoft in its .NET platform offers Mobile Web Forms. These forms are based on restricted set of intelligent components that, to our knowledge, cannot be extended with additional widgets. IBM's Transcoding Publisher [7] actually transforms web contents to variety of other formats, giving the user possibility of customization of the transformation parameters. However, the drawback of the approach is its ability to transform only web contents, and in a centralized fashion.

In the context of classic desktop applications, we would like to stress the approaches followed by adaptable XML-defined interfaces [1, 6, 3]. Without providing details, we mention some approaches: language-based, grammar-based, e.g., BNF, event-based, constraint-based, UAN (User Action Notation, in particular for direct manipulation) and widget-based. However, the XML-based efforts are the most interesting for us, since they provide flexibility and easy manipulation. Our architecture adopts a similar approach, but this transformation is done on-the-fly, transparently to the user.

5.2 Predicting User Behavior

Measuring user interfaces and predicting user behavior is based on several concepts. The basic idea is to collect user interface metrics for a web site [8]. Usually, collected data are used to perform traffic-based analysis (e.g., pages-per-visitor, visitors-per-page), time-based analysis (e.g., page view durations, click paths) or number of links and graphics on the web pages. Some approaches [8] tend to empirically validate metrics against expert ratings (e.g. PC Magazine TOP 100 web sites). This approach allows to monitor web interfaces, however our approach allows to monitor web interfaces and desktop applications, or any other kind of GUI.

Several approaches provide concrete methods to predict and simulate user behavior, the majority of them are based on Markov chains [4]. Predictions are based on the data from the usage logs. More advanced models, like Longest Repeating Subsequence (LRS) [17] or Information Scent [2] perform data mining seeking to analyze navigation path based on server logs, similarity of pages, linking structure and user goals. Our approach can follow any of these prediction methods: in our prototype LRS is used in order to work with any GUI generated by the ADUS agent.

6 Conclusions and Future Work

This paper presents an architecture for adaptive user interface generation on wireless devices. We showed the enhancements that our system brings in contexts of mobility and adaptability without requiring any additional effort from the application developers. As summary, the main advantages of our approach are:

- Transparent adaptation of GUIs to different wireless device capabilities, contextual data and user preferences.
- Automatic monitoring of user interactions, which enables 1) the learning of user behavior patterns by using behavioral data mining/analysis, and 2) the development of intelligent user agents that anticipate future user actions.

A non-trivial multiagent application was used as sample testbed to show the advantages and feasibility of our approach.

As future work, we are improving the exploitation of user interaction data stored by the user agents.

References

[1] Abrams, M., Phanouriou, C., Batongbacal, A., William, S., and Shuster, J. Uiml: An appliance-independent xml user interface language. In *WWW8 / Computer Networks 31(11-16): 1695-1708* (1999).

[2] Chi, E., Pirolli, P., , and Pitkow, J. The scent of a site: A system for analyzing and predicting information scent, usage, and usability of a web site. In *ACM CHI 00 Conference on Human Factors in Computing Systems* (2000).

[3] Deakin, N. XUL tutorial, 2002. http://www.xulplanet.com/tutorials/xultu/.

[4] Deshpande, M., and Karypis, G. Selective markov models for predicting web-page accesses. Tech. rep., University of Minnesota Technical Report 00-056, 2000.

[5] ECMA. Ecmascript language specification, December 1999. http://www.ecma.ch/ecma1/stand/ecma-262.htm.

[6] (eXtensible Interface Markup Language), X., November 1999. http://www.ximl.org/.

[7] IBM. Ibm websphere transcoding publisher, 2001. http://www-3.ibm.com/software/webservers/transcoding/.

[8] Ivory, M., Sinha, R., , and Hearst, M. Empirically validated web page design metrics. In *SIGCHI* (2001).

[9] jXUL, 2002. http://jxul.sourceforge.net.

[10] Liu, H., Lieberman, H., and Selker, T. A model of textual affect sensing using real-world knowledge. In *2003 International Conference on Intelligent User Interfaces* (January 2003).

[11] Mena, E., Royo, J., Illarramendi, A., and ni, A. G. An agent-based approach for helping users of hand-held devices to browse software catalogs. In *Cooperative Information Agents VI, 6th International Workshop CIA 2002* (September 2002), Springer-Verlag LNAI, pp. 51–65.

[12] Milojicic, D. Mobile agent applications. *IEEE Concurrency 7(3)* (1999), 80–90.

[13] Mitrovic, N., and Mena, E. Adaptive user interface for mobile devices. In *Interactive Systems. Design, Specification, and Verification. 9th International Workshop DSV-IS 2002, Rostock (Germany)* (June 2002), Springer Verlag LNCS, pp. 47–61.

[14] Mitrovic, N., Royo, J., and Mena, E. Adus: Indirect generation of user interfaces on wireless devices. In *Seventh International Workshop Mobility in Databases and Distributed Systems (MDDS'2004), Zaragoza (Spain)* (Sept. 2004), IEEE CS.

[15] Molina, J., Melia, S., and Pastor, O. Just-ui: A user interface specification model. In *4th International Conference on Computer-Aided Design of User Interfaces CADUI 2002* (2002), C. Kolski and J. V. (ed.), Eds., Kluwer Academics Publisher, pp. 63–74.

[16] Mozilla. The mozilla project, 2000. http://www.mozilla.org.

[17] Pitkow, J., and Pirolli, P. Mining longest repeatable subsequences to predict world wide web surfing. In *2nd Usenix Symposium on Internet Technologies and Systems (USITS)* (1999).

[18] Pitoura, E., and Samaras, G. *Data Management for Mobile Computing*, vol. 10. Kluwer Academic Publishers, 1998.

[19] Thevenin, D., and Coutaz, J. Plasticity of user interfaces: Frame-work and research agenda. In *Proc of IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'99, Edinburgh* (August 1999).

[20] W3C. Xforms, 2000. www.xforms.org.