Using Mobile Agents in User Interfaces Functionality

Cristina POPÎRLAN, Claudiu-Ionuţ POPÎRLAN

Faculty of Mathematics and Computer Science,
Department of Computer Science,
University of Craiova, Romania
cristina_popirlan@yahoo.com, popirlan@gmail.com

Abstract. Adapting graphical user interfaces (GUI) to meet higher level of usability for users is one of the most interesting questions of today's mobile computation.
In this paper, we present a solution based on mobile agents. Mobile agents 'learn' users' habits, cooperate with other agents and construct the GUI in order to meet the users' expectations. Mobile agents move from host to host and are able to 'learn' about GUI usability by observing multiple users using the GUI. The result is an adaptable GUI platform that strives to be more usable. We show the application of this approach by implementing a simple business application.
Keywords: mobile agents, graphical user interface, interface language, java swing, mobile technologies
Math. Subjects Classification 2000: 68T40, 68T05

1 INTRODUCTION

Adapting graphical user interfaces (GUIs) to meet usability is one of the most challenging questions in the user interfaces area. Main problems are raised from the fact that the usability is hard to measure and analyse, and that measured data are often not available to multiple instances of the program.

Some solutions aim to collect metrics on web GUI usability [2] so the data could be used to analyse the usability, and more advanced approaches [1] try to predict user behaviour and to propose measures that could increase GUI's usability. The idea of this work is to transparently predict user behaviour and to adapt accordingly graphical user interface by using mobile agent systems [3]. Agents are highly mobile, autonomous and intelligent. They can cooperate with other intelligent agents in order to exchange information and maximise performance.

Our prototype adapts user interface using mobile agents [1], [2] that process user interface definition described in Extensible User-interface Language (XUL) [10]. XUL interpretation to a standard Java Swing interface is done by jXUL platform [4], [6]. Agents automatically adapt the interface definition to the clients' interface, making user interface dynamic and multiple interface implementations unnecessary.

2 GENERATING USER INTERFACES WITH MOBILE AGENTS

In our prototype we use eXtensible User Interface Language (XUL) and Mobile Agents in order to create user interface. A mobile agent [1], [2], is a program that executes autonomously on a set of network hosts on behalf of an individual or organization. The agent visits the network hosts to execute parts of its program and may interact with other agents residing on that host or elsewhere [5], while working towards a goal. During their lifetime agents travel to different hosts that can have distinct user interface possibilities. Agents typically possess several (or all) of the following characteristics:

- *Goal oriented*: they are in charge of achieving a list of goals (agenda);
- *Autonomous*: they are independent entities that pursue certain objectives, and decide how and when to achieve them;
- *Communicative/collaborative*: to achieve their goal they can cooperate;
- *Adaptive/learning*: agents "learn" from their experience and modify their behaviour respectively;
- *Persistent*: agent's state (should) persist until all the goals are achieved
- *Reactive*: they react to their environment which also could change their behaviour;

Many agents are meant to be used as intelligent electronic gophers – automated errand boys. Tell them what you want them to do – search the Internet for information on a topic, or assemble and order a computer according to your desired specifications – and they will do it and let you know when they have finished. Some agents are used as *Personal Agents* that store user preferences, certificates, policies or perform actions on the behalf of the user (e.g. enforcing security policies [9]).

3 USING MOBILE AGENTS TO IMPROVE USABILITY

Mobile Agents are particularly suitable for adapting user interfaces and learning [1], [8]. Agents are autonomous, communicative, they work towards their goal, and can decide of their actions based on the environment and external factors [3], [9]. Mobile agents endorse "push" technology – agents can travel to any host or user without prior invitation. They can provide transparent resolution of many environment errors (e.g. network errors). In our prototype we use mobile agents to create Swing interfaces, and we plan to use this technology to improve user interface usability for various user devices (e.g. HTML clients, WAP clients, etc.). We created specialised mobile agents that learn user behaviour. These agents examine usage data in order to predict next probable user action. They exchange data, learn from user actions and keep in mind user's preferences. Furthermore, re-designed user interface could be pushed to

users (using mobile agents) at any time since GUI designers can also learn from the usage data. We present a sample application for invoice composition and manipulation. The application has basic options, such as opening, saving, closing and printing an invoice, adding items and taxes and selecting a customer. Sample application is mobile and it communicates with other agents and application instances. User interface is adapted to user's preferences and the application has interactive help that guides users to achieve goals.

3.1 IMPLEMENTED TECHNOLOGY

We built a prototype that adapts user interfaces to user needs in transparent manner to both designer and user. We extended *eXtensible User Interface language* to support demarcation of design patterns. Using extended tags we are able to determine content and position of different design patterns within the user interface definition and therefore to adapt user interface according to user's preferences.

3.2 SPECIALISED AGENTS

Applications based on mobile agents typically consist of several agents that perform different tasks [11]. These agents are specialised to perform these tasks, and contain expert knowledge on how to achieve their goals. Agents learn and react to their environment and autonomously provide functionality to the system, application or other agents. We have created several specialised agents that work together and help the adaptation and learning process:

- *User Interface Agent*: serves as a bridge between user and mobile agents. This agent is capable of transforming user interfaces to meet capabilities of various user devices, and it uses XUL as user interface definition language. This agent is fully extendible and connectable to other agents;
- *Helper Agent* : using this agent, our prototype is able to learn. Helper analyses data that is being collected from all users and suggests the next most probable action;
- *Wanderer Agent* : this agent is a specialised agent that wanders trough all clients and collects usage data. Its goal is to exchange data between all Helper agents;
- *Personal Agent* : users can store their preferences (e.g. font sizes, colours, etc.) with this agent. Additionally, this agent can store other relevant data: accessibility preferences, preferences on usability patterns or user certificates;

In *Figure 1* we can observe the composition of the sample application and its network topology. While User Interface, Personal and Helper Agent are mainly static, Wanderer agent travels through network and distributes usage data and updates. Functionality of these agents will be described in following sections.

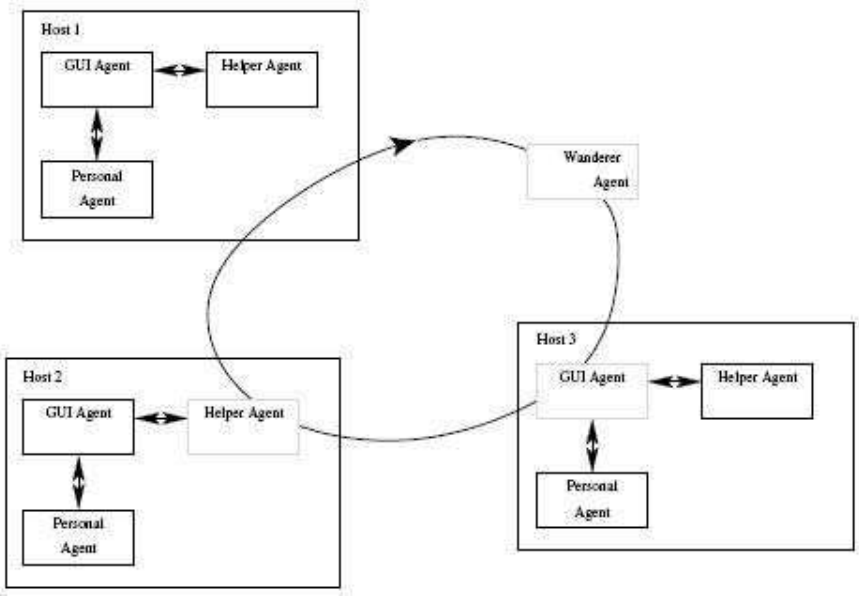

Figure 1. Sample application structure

4 SAMPLE APPLICATION

In our sample scenario, *Invoicing application* is an application based on mobile agents and is retrieved from network, from the nearest host. In *Figure 2* we can see that sample application has several windows:

- *Main window*: from this window, user can execute some of the options, e.g. "new invoice", "open invoice", etc;
- *Invoice window*: whether invoice is new or loaded from the database, this window fits into the main window and gives additional options, e.g. "save invoice", "print invoice", "edit items", etc;
- *Edit items window*: this window enables user to add or remove items from invoice or to print item details;

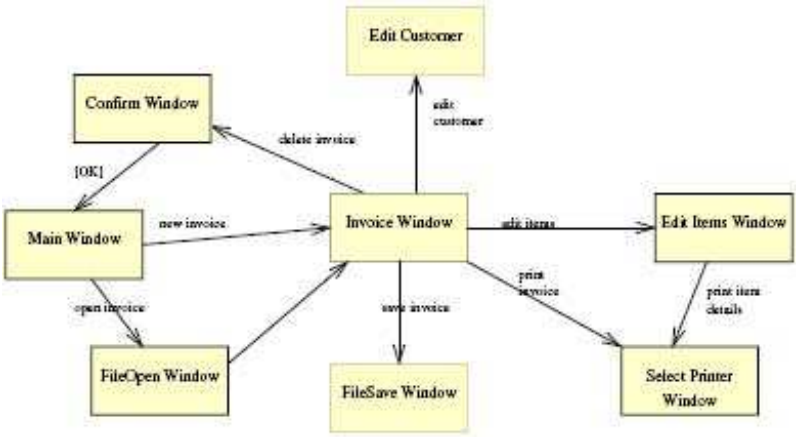

Figure 2. Application structure

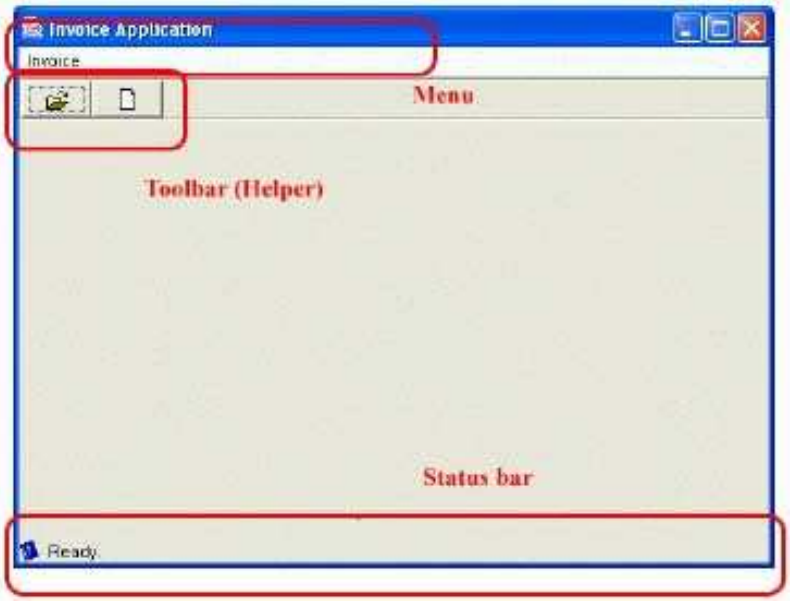

Figure 3. Main window: basic layout and patterns.

Mobile agents (in our case, specialised agents) autonomously collect usage data from all users and "learn" user habits using LRS(Longest Repeating Subsequence) [8] model. Helper agents then predict the next action to be performed by the user, and display available actions in order of probability in the specialised toolbar (see *Figure 3*). Additionally, agents cooperate and adjust user interface to suite user's preference.

In *Figure 3* we can observe that user interface consists of several design patterns: *Menu, Status bar and Toolbar*. As we mentioned earlier, this toolbar

(*Helper toolbar*) serves as front-end of the Helper Agent to users, giving a hint of the next most probable action. Action that is predicted as the most probable next action will be the first action in the Helper toolbar, and the last one will be the action with least chances to be selected by the user. User can click the icon that appears on the Helper toolbar to perform the action. We decided to use Helper toolbar to display prediction instead of modifying the user interface itself. Modifying the user interface in run-time could be very confusing for the users.

4.1 INTERACTION BETWEEN SPECIALISED AGENTS

Here is how the training process goes. When the application is created, it has no usage data. Helper agent's repository and Wanderer's repository are empty. In *Figure 4* you can see interaction and processes map of the application. In this case the Helper toolbar should be empty, as no prediction can be made. It is expected that the application designer will provide some basic training to Helper agent so the application could have some initial predictions.

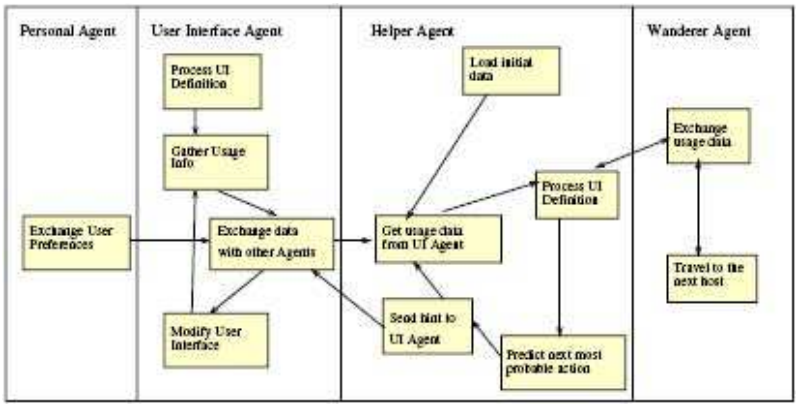

Figure 4. Agent processes and interaction.

When the application is requested from one of the hosts, the Wanderer agent is informed that one of the instances of application is about to depart. Wanderer registers this instance (so it can be visited or updated later), and loads the initial snapshot of usage data to *Helper Agent*. After being loaded with data, application moves to the destination host, and starts.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have presented autonomous and intelligent system based on mobile agents that transparently adapts user interface. We have constructed specialised agents that predict user behaviour and suggest actions to users in real-time. The main features of this approach are:

- We use and extend our previous work, a system for adapting user interfaces to various resources by using mobile agents;
- Specialised agents have been built:
  - *Helper Agent*: use predictions to improve application usability;
  - *Wanderer Agent*: exchanges data between application users, implements push service and is capable of pushing the re-designed user interface to all users;
  - *Personal Agent*: stores all user's preferences and cooperates with other agents in order to apply these preferences.
- User interface is modified at run-time.

Our future work will be focused on:

- Adapting of this concept to various resources (HTML, WAP, etc.). The potential problems, as we discussed, are in adapting usability patterns to various resources;
- Reduction of the space necessary for storing sequences;
- Definition of in-window tasks so the system could predict tasks within one window.

References

[1] **P. Braun , W. Rossak**- *Mobile Agents: Concepts, Mobility Models, & the Tracy Toolkit*, Elsevier Inc. (USA) and dpunkt.verlag (Germany), 2005

[2] **J. Baumann**- *Mobile Agents: Control Algorithms*, Lecture Notes in Computer Science, Springer

[3] **Claudiu Popîrlan, Cristina Popîrlan**, *Algorithms for Mobile Agents in Network using Tracy(Mobile Agent Toolkit)*, 5-th RoEduNet International Conference, Sibiu, 1-3 June 2006

[4] **C. Popîrlan**, *A Java Implementation of Modeling Results About Stratified Graphs*, Research Notes in Artificial Intelligence and Digital Communications, Vol.104, 4-nd Romanian Conference on Artificial Intelligence and Digital Communications, Craiova, June 2004, p.31-38

[5] **The Tracy web page**: http://wiki.tracy.informatik.uni-jena.de/mobileagents/tiki-index.php

[6] **Java Remote Method Invocation :** http://java.sun.com/products/jdk/rmi/

[7] **XIML (eXtensible Interface Markup Language):** http://www.ximl.org/

[8] **Distributed Objects & Components: Mobile Agents:** http://www.cetuslinks. org/oo_mobile_agents.html

[9] **Foundation for Intelligent Physical Agents:** http://www.fipa.org

[10] **Cheng, T.: XUL - Creating Localizable XML GUI:** http://www.mozilla.org/projects/intl/iuc15/paper/iuc15xul.html

[11] **The Mobile Agent List, University of Stuttgart:** http://mole.informatik.unistuttgart.de/mal/mal.html