Adaptive Help for e-mail Users

Katerina Kabassi

Department of Informatics,
University of Piraeus
80, Karaoli and Dimitriou St.,
Piraeus 185 34, Greece
kkabassi@unipi.gr

Maria Virvou

Department of Informatics,
University of Piraeus
80, Karaoli and Dimitriou St.,
Piraeus 185 34, Greece
mvirvou@unipi.gr

Abstract

This paper describes a graphical user interface that provides intelligent help to the users of an e-mailing system. The graphical user interface is called Intelligent Mailer (I-Mailer). I-Mailer monitors users while they work; in case a user is believed to have made a mistake, the system intervenes automatically and offers advice. In order to provide individualised help, the system keeps information about each user interacting with the system. This information is maintained centrally on a server based on Web Services. The main characteristic of Web Services is their strong reliance on web standards when they interact with the applications that invoke them. The Server keeps user models for every individual interacting with the client application. In case the Server has not gathered enough information about a particular user, it consults a library of stereotypes and acquires some default assumptions about the user.

1 Introduction

As the number of users that use the Internet in order to process their e-mail increases, the insufficient ability of e-mail clients to satisfy the heterogeneous needs of many users becomes more apparent. Novice users may encounter difficulties due to lack of experience and expert users may face problems due to carelessness or possible tiredness. A remedy for the negative effects of the traditional ‘one-size-fits-all’ approach is to develop systems with an ability to adapt their behaviour to the goals, tasks, interests and other features of individual users and groups of users (Brusilovsky & Maybury, 2002).

Therefore, a lot of research energy has been put into the development of intelligent user interfaces that could help users organise their mailbox (Payne & Edwards 1997) or help them by performing tasks on their behalf (Lashkari, Metral & Maes, 1994). However, a main drawback of these approaches is that they do not aim at helping users perform the tasks by themselves and, therefore, learn from that experience. This goal however, is addressed by the research area of intelligent help systems. Examples of help systems are UC system (Wilensky et al., 2000), RESCUER (Virvou & du Boulay, 1999), CHORIS (Tyler, Schlossberg, Gargan Jr., Cook & Sullivan, 1991) and Office Assistant (Horvitz, Breese, Heckerman, Hovel & Rommelse, 1998).

For the purposes of helping users organise their mailbox by assisting them perform the tasks by themselves, we have developed a Graphical User Interface (GUI) that offers automatic assistance to users in problematic situations. The system developed is called Intelligent Mailer (I-Mailer). I-
Mailer constantly reasons about every user action and provides spontaneous advice in case an action contradicts the user’s hypothesised intentions. Hypotheses are generated based on a simulator of human error generations and a limited goal recognition mechanism that is used by the system to improve control. However, the main feature of the system is its ability to adapt its interaction to the needs and characteristics of each individual user.

Many researchers agree that the development of effective assistance systems should be based on a user model which keeps track of what the user is doing (e.g. Matthews, Pharr, Biswas & Neelakandan, 2000). However, one problem with user modelling is that the user model of each individual is usually device-dependent. This means that the user model is built and maintained on a particular device in a computer lab. This approach assumes that users always use the same device. However, in a real computer-lab environment users often receive and send messages from different devices. A solution to this problem may be achieved with the incorporation of a user model based on Web Services into the intelligent help system.

2 Intelligent Mailer

Intelligent Mailer (I-Mailer) is an Intelligent Graphical User Interface that works in a similar way as a standard e-mail client but it also incorporates intelligence. I-Mailer’s main aim is to help users while sending and receiving e-mail messages and help them organise their mailbox. Therefore, I-Mailer monitors users during their interaction with the system and reasons about all users’ actions. In case it suspects that a user may have been mistaken with respect to his/her hypothesised intentions, it provides spontaneous advice. Otherwise, the action is executed normally.

In particular, every time a user issues an action, I-Mailer reasons about it and categorises it in one of four categories, namely “expected”, “neutral”, “suspect”, “erroneous”. A command is categorised as expected if it is compatible with the user’s hypothesised goals. It is considered suspect if it contradicts the system’s hypotheses about the user’s goals and erroneous if the command is wrong with respect to the user interface formalities. The command is considered neutral if it cannot be assigned to one of the former categories. Finally, an action is categorised as erroneous if it is wrong with respect to the user interface formalities.

If the action is categorised as expected or neutral, it is executed normally. However, if the action is categorised as suspect or erroneous, the system tries to generate alternative actions that the user may have meant to issue instead of the one issued. Therefore, the action issued is transformed so that similar alternatives can be found, which would not be suspect or erroneous. However, since the transformed action has to fit better in the context of the user’s goals, the system reasons about every alternative action generated from the transforms. As a result, each transformed action is categorised in one of the four categories in a similar way as the actual command issued by the user. Finally, only expected actions are selected. In case, the system cannot find any action that is considered better than the action issued by the user, then the user’s action is executed normally.

An example of a user’s interaction with the system is the following: In an attempt to reorganise his/her mailbox, the user deleted the contents of the folder ‘Fifth’. Then s/he accidentally tried to delete the folder ‘Fifty’ as well. However, the system found the particular action “suspect”, because the user had previously stored in that folder a lot of e-mail messages and his/her action would have resulted in losing valuable data. Therefore, the system asked the user whether s/he really meant to delete the folder ‘Fifth’ instead of “Fifty” for two main reasons:
1. The folder ‘Fifth’ had been emptied whereas the folder ‘Fifty’ had not.
2. ‘Fifth’ was very similar to ‘Fifty’, therefore, there could have been a mistake.

3 Web Services for User Modelling

Web services introduced a new model on the Web in which information exchange is conducted much more conveniently, reliably and easily. Web Services interact with the applications that invoke them, using web standards such as WSDL (Web Service Definition Language) (Christensen, Curbera, Meredith & Weerawarana, 2001), SOAP (Simple Object Access Protocol) (Box et al., 2000) and UDDI (Universal Description, Discovery and Integration) (UDDI, 2001). Basing user modelling on web standards has the advantage of enabling the dynamic integration of applications distributed over the Internet, independently of their underlying platforms.

Until recently, user modelling through the Internet has been mainly based on a client-server model. Examples of applications or projects that follow a client-server architecture are Casper (Smyth, Bradley & Rafter, 2002), PACF (Machado, Martins & Paiva, 1999) and Adaptive Information Server (Billsus & Pazzani, 2000). However, in this approach, the developer has to create his/her own communication protocol and the clients may experience problems in receiving data from the Server. For example, if a user works both at home and at work, his/her user model may not function as expected, because the client at work may be behind a firewall that does not allow the user modelling server’s port to pass through. On the other hand, Web Services, follow the XML protocol for sharing data, thus making this data readable via virtually every machine. In addition, Web Services rely on the Hypertext Transfer Protocol and thus gain the advantage of being able to flow through most security systems (Firewalls, Proxy Servers, etc).

The operation of Web Services in I-Mailer is quite simple. I-Mailer incorporates a Server that stores and updates individual user models through Web Services. This Server is called Web Service User Model (WebSUM). The application makes a request to WebSUM and WebSUM returns a string containing the response to the application’s request. Based on this, I-Mailer sends the username and password of the user to WebSUM and WebSUM is responsible for finding the user model and sending this information to the client that requested it. The user model is updated with information gathered locally, through the user’s interaction with the application. In particular, the information, which is acquired locally, is sent to WebSUM so that the user model is updated there. In this way, WebSUM keeps track of intentions and possible confusions of each individual user. This information is available to the application irrespective of the computer where it is running.

Every time the user interacts with I-Mailer, the system requests information from the Web Server that contains the particular user model. WebSUM retrieves the request and returns the result to the client application. WebSUM tries to find the information requested in the individual history of the user. However, there are cases where WebSUM does not have adequate information about the user. In such cases, it consults a library of stereotypes that it maintains. Stereotypes are used to provide default assumptions about users until the user model acquires sufficient information about each individual user. Indeed as Rich (1989) points out a stereotype represents information that enables the system to make a large number of plausible inferences on the basis of a substantially smaller number of observations; these inferences must, however, be treated as defaults, which can be overridden by specific observations. Therefore, stereotypes are used in WebSUM only for capturing the initial impression of a user.
A stereotype is activated during the first interaction of the user with the system. The user has to answer some questions about his/her believed level of expertise, his/her previous experience, his/her knowledge in related topics such as file-store manipulation etc. The users are classified into one of three major classes according to their level of expertise, namely, novice, intermediate and expert. Furthermore, after the user has executed a satisfactory number of commands, the system can also draw inferences about the user’s proneness in making mistakes due to carelessness. Therefore, there are two more stereotypes that divide users into two groups, careless and careful. After a stereotype has been activated, the system makes some default assumptions about users’ possible errors and can provide some kind of advice.

In the beginning, information is acquired only by the stereotype. However, the system is also constantly collecting information about a particular user’s behaviour and errors and informs the individual user model of the user. As the system collects more and more evidence about a user, information is acquired in part by the stereotype and in part from the individual user model. The percentage of information acquired by the stereotype diminishes as the percentage of acquisition by the individual user model increases.

In case a conflict appears, the system always lays more weight on the information acquired from the individual user model. For example, if the stereotype supports that a user’s most common error is in the use of a certain command and the individual user model supports that the particular user’s most common error is due to a misconception concerning the structure of folders, then the system will favour the view proposed by the individual user model.

4 Conclusions

In this paper we described I-Mailer, an intelligent graphical user interface for an e-mailing program. I-Mailer reasons about every user action and produces advice in cases where the user is having problems with his/her interaction with the system. In order to identify when the user needs help, the system makes hypotheses about every user’s intentions. Hypotheses generation is based on a limited goal recognition mechanism.

For the provision of intelligent and individualised help, I-Mailer depends on its WebSUM user modelling component. This component allows for centralised maintenance of information about the user, allowing its access through the Internet from virtually anywhere. The main characteristic of Web Services is that they interact with the applications that invoke them, using web standards. Basing user modelling on web standards has the advantage of enabling the dynamic integration of applications distributed over the Internet, independently of their underlying platforms.

5 References


