

Abstract of 'A Textual Journal for Telecommunication Services'

Olivier Curt

Laboratoire d'Intelligence Artificielle de Paris 5
UFR de Mathématiques et d'Informatique
Université René-Descartes
45, Rue des Saints-Pères
75006 Paris -France

Introduction

The significant increase of telecommunication services over the last few years combined with a widespread use of personal computers and workstations has opened up an incredible amount of application possibilities. Human factors and ergonomics have become a main concern for overcoming new kinds of problems arising along with the complexity and changes of these working environments.

On the request of the CNET and CNRS, we have been studying the interactions of an end-user with a system that bundles together as many as five different services : a word processor, an action historic, a macro-command recorder, a journal and a telecommunication tool (a French minitel emulator). One of our main concerns was to help the user with all their repetitive tasks by applying demonstrational techniques into the prototype. Repetitive tasks are such a burden for all potential users, from the experienced programmers to first timer computer users.

The heart of such a system is to define the invocation of these services while anticipating the upcoming end-user's actions. When trying to do so, we need to infer on the tasks performed by the user, we need to trace the data by storing them in a macro-command, keeping track of the user's past command history.

What our prototype does ?

Let's take the example of an end-user who once in his office near 9 AM, switches on his workstation, clicks on his french minitel emulator and searches for sports news in the journal "L'Equipe". If he is a big fan of sports, he could easily repeat these tasks five days a week for quite some time. Our prototype offers the opportunity to create a macro-command that performs these actions, so that the end-user would just have to select this macro-command in a list of macro-commands. Now let's consider our user has created a "Macro-Equipe" macro-command. He runs it on monday at 9 AM, on tuesday at 9:15 AM. Wednesday at around 9 AM (this is what we call the 'next use date') the prototype will propose the end-user to run the "Macro-Equipe" macro-command for him on . If the end-user accepts it, the system will propose a next run the following day at the same time and so on.

Macro-commands could run together the different services we have in our prototype. For example, the end-user could search for information in a Minitel service and save these information in our word processor via copy-paste in the appropriate positions.

How our prototype works ?

Macro-commands

In our prototype, the recording of macro-commands works on the begin / end principle, it means that the end-user needs to explicitly show to the system when a macro-command begins and ends. In between these two time-points, the system records the user's primitive actions.

Agenda and Journal

We need to store all the dates of use of the macro-commands, in order to be able to infer a 'next use date'. We called this data bases of dates an 'Agenda', because it regroups facts and dates. An agenda is associated with every macro-command recorded by the end-user.

The agenda of all the macro-commands can be shown on the screen using the "Journal", it is an in-line, array kind of presentation of all the macro-commands' name, the application it is related to as well as all the dates of uses. In fact, the journal gives an image of the end-user methodology, with all the past events stored. The journal is therefore a means of expression for the user and a source of observation for the system.

A graphical "Journal" is currently underway on another project at the LIAP-5. We aim to let the end-user find the most effective version (in-line or graphical) during an evaluation of our systems.

Programming by Demonstration (PBD)

PBD interfaces aim at helping the user work with customized environment by creating parametrized procedures by demonstration (here showing the first dates of use) in order to extend application capabilities and eliminate repetitive tasks. The context we are going to infer on is the time of use of macro-commands, to do so, we need a time-based system and the dates of all the macro-commands being used or in use.

When the agenda of a macro-command is enriched with a couple of dates, the system manager considers that an inference is possible. The system, then searches for constant or a logical sequence of numbers within the dates, trying to find a relevant 'next use date'. If the system finds one, it will record that date into a manager agenda (same presentation than the "Journal" of past actions). Because the system is a time-based one, the manager constantly searches in his agenda if a macro-command is about to start soon. A defined time (by default a minute but we need to define that value more accurately during our tests with end-users) before the start of a new macro-command, the system will ask the end-user if a run of this macro-command is desired now. This feedback on the system is very important because it is the end-user that will decide if the inference is right or wrong.

Wrong inferences

A problem occurs to any PBD system when the inference done by the system is denied by the end-user. That means the system found a logical "next use date" but it was a wrong one or that the end-user has changed his habit and does not need to run the macro-command as he used to.

This is crucial issue in any PBD system, two solutions are possible :

1- the system tries to solve the problem with the help of the end-user. It could be done by prompting several questions, trying to start a dialog with the end-user. For example : "Is there a wrong date in the agenda ?".

2- the system is on a "stand-by" mode, it means that he doesn't try any inference on the agenda of this macro-command, will stay asleep, waiting for the end-user to run this macro-command and give a new date.

The system aims at giving an accurate inference and to optimize the future propositions to the end-user. To do so, we need to find the most efficient interaction mode, cooperation between the machine and the user. We chose the second solution, asking too many questions could become a burden to the end-user and we don't know where is the limit between being an intelligent system (solving the problems alone) and trying to be guided by the end-user.

Conclusion

At the LIAP-5 laboratory, we have implemented prototypes using a generic architecture with programming by demonstration mechanisms. The major issues to define the best interaction mode between the system and the end-user (undo/redo, prompt, evaluation of an inference by the end-user).

The study of simple inferences, like inferencing on dates described in this article, helps us toward a better understanding on implementing effective cooperation between prototypes, said as intelligent, and the end-user.

Our ultimate goal is to bundle together all our project and to use AppleEvents so we could use real software (Word, Netscape, etc...) instead of our prototypes. But first of all, we need to do quantitative tests with users, finding how the user modifies his working methodology and behavior.