

Familiar: automating repetition in common applications

Gordon W. Paynter

Department of Computer Science, The University of Waikato, New Zealand

ABSTRACT

Computers reputedly excel at repetitive problems, yet many users find themselves performing the same actions over and over again. Non-programmers have little choice but to perform iterative tasks by hand. Programming by demonstration is an end-user programming technique that lets the user *teach* the computer a program by showing it examples of what they want done, much as they might teach another human. This paper describes Familiar, an agent that helps users to automate iterative tasks by demonstration using common applications and existing technology. An evaluation shows that users are able to automate iterative tasks with Familiar, but found that they prefer other techniques in many situations if they have the choice.

1 Introduction

Computer users often encounter problems in common applications like spreadsheets and word processors that can only be solved by repeating an action or set of actions. Tasks that are quickly and easily accomplished once become tedious and time-consuming when they have to be repeated tens or hundreds of times. A common solution is to write a program or script to automate tasks like as these, but this approach is unavailable to non-programmers, who form the majority of users. Even experienced programmers face difficulties writing such a script: it will require debugging and testing, and may have to duplicate complex functionality that is easily available to a human through the application's user interface.

Programming by demonstration (PBD) is a potential solution to iterative problems [3]. PBD interfaces let users generate programs by demonstrating the task they want to perform. The PBD system observes the user's demonstration and attempts to learn and complete their task. Few demonstrational interfaces are in general use.

Familiar is a PBD agent with several unique features. It works with a set of existing, unmodified applications like spreadsheets and graphical operating systems, and uses existing technology to monitor user actions and control other applications. An evaluation showed that users were able and willing to use Familiar to automate repetitive tasks, though they preferred other tools and techniques in many situations.

The paper is arranged as follows: Section 2 discusses two existing demonstrational techniques for completing iterative tasks in existing applications. The Familiar PBD system is described in Section 3, and an example of its use is given. Section 4 summarises the user evaluation, the participant's reactions, and the Familiar implementation, with particular emphasis on using AppleScript as a basis for PBD. Finally, several other demonstrational interfaces are reviewed and compared to Familiar.

2 Programming by demonstration

The simplest and most widely-used PBD interface is the macro recorder. A typical macro recorder lets the user record a series of keystrokes and mouse actions, and play them back at a later time. Macros create efficiencies by aggregating a series of commands into a single keystroke, but are little help with tasks that vary from one iteration to the next.

Eager is a more sophisticated PBD system made up of an interface agent and a modified version of the HyperCard multimedia environment [2]. The Eager agent records the user's actions in the application as a sequence of high-level events, and searches it for simple iterative cycles. When a cycle is detected, Eager predicts the user's subsequent commands and displays the predictions in the application interface using a novel technique called *anticipation highlighting*. When the user sees the predictions are routinely correct they can invoke the agent directly and instruct it to execute the next iteration or the remainder of the task.

The learning in Eager is more sophisticated than the simple mimicry performed by a macro recorder. Eager is active the entire time that the HyperCard application is in use, continuously searching for repetition without cues from the user. An important aspect of the search is that low-level events (like keypresses and mouse movements) are generalised into more meaningful high-level commands (like *Save As* and *Open*). Once Eager has identified a cycle of events, it attempts to predict exactly the actions the user will perform next. It can recognise and extend patterns that span iterations of the task. Eager uses a complex model of the application to determine what parameter values need to be predicted for each command, and how they should be predicted.

Eager was a prototype system, and is unsuitable for everyday use. It is application-independent in theory, but in practice was only demonstrated with HyperCard because it requires changes to the operating system and application, and a large and complex model of the applications data.¹ The application model specifies how every data element can be predicted from other data elements. In a complex application like HyperCard, there are so many potential prediction relationships that it is difficult and expensive to describe them all. Inevitably, there are gaps in the model and hence in the agent's ability to learn. Another problem is that Eager has little tolerance for noise in the event trace—the user has to perform the demonstration without error or Eager is unlikely to recognise and extrapolate the repetition.

3 Familiar

Familiar is a PBD system that addresses many of Eager's shortcomings. It works with existing applications but does not require modifications to the program or operating system, nor does it need external application models to guide its inferencing. Further, Familiar tolerates many types of mistake in the user's demonstration. These improvements come at a cost: Eager's sophisticated user interface is not reproduced; instead the agent communicates with the user in simple, English-like sentences.

¹A spreadsheet implementation was “much less complete” (see [3], page 209).