Computer Analysis of User Interfaces

Antonio C. Siochi

TR 89-34

COMPUTER ANALYSIS OF USER INTERFACES

Antonio C. Siochi

Department of Computer Science
Virginia Tech
Blacksburg, VA 24061

ABSTRACT

Interface evaluation is a necessary phase in the production of quality user interfaces. The usual evaluation techniques involve formal experiments or observation, and can be invasive. One non-invasive method that can be used at user sites is to record all user input and system output to a file. This transcript is then algorithmically analyzed to determine interface problems. A new technique analyzes these transcripts by searching for maximal repeating patterns (MRPs), on the hypothesis that repeated sequences of user actions indicate interesting user behavior, and therefore may show problems in the interface. The technique was tested by using it to evaluate the human-computer interface of a large and complex image processing system in active use. Results showed MRPs were useful in detecting specific problems within the interface.

> "The null hypothesis can always be rejected. It just depends on how much money and time you are willing to spend in order to do so."
>
> - Anonymous

COMPUTER-BASED INTERFACE EVALUATION

The production of quality human-computer interfaces requires a thorough understanding of the user that can involve evaluating the system by observing the user working with the system through field studies or formal human factors experiments. Such methods traditionally involve use of techniques such as videotape, protocol analysis, and critical incident analysis. These methods may be invasive — certainly they require time-consuming analyses; for example, manually analyzing one hour of videotape can require a day or more [2]. An alternative approach is to record all user input and system output onto a file, i.e., log user sessions. Such transcripts can be collected automatically and over a long period of time. An important benefit of this method is that data can be collected inexpensively from user sites, thus providing the evaluator with transcripts which reflect the context of the job in which the software system is used. Unfortunately this method also produces voluminous amounts of data. There is therefore a need for tools and techniques that allow an interface evaluator to extract potential usability problems from such data. This research hypothesizes that repetition of user actions is a significant indicator of potential user interface problems. It has resulted in a repetition detection algorithm and an interactive tool constructed to support detection of these usage patterns.

THE REPETITION HYPOTHESIS

Assuming that user behavior at a computer is purposeful, then the user carries out a sequence of tasks to achieve some goal (the Rationality Principle [1]). It is therefore reasonable to assume that *repeated* sequences of actions in the transcript indicate a task, rather than random sequences of user actions. Moreover, the more repetitions of a sequence that exist, the greater the likelihood that the sequence actually represents a task. Repetition is also interesting of itself. Each action performed by a user takes a certain amount of time and involves a chance for errors. Repeating such actions increases both the user's performance time and the likelihood of errors. By detecting frequently repeated actions and

providing macros for them, it may be possible to reduce user performance times and user errors.

When errors are involved in the repeated actions, it may be the case that users are having problems with a particular task. Users may be trying variations of a command (e.g., changing the order of command arguments) in an attempt to make it work. Such repetitions might indicate problems with, for example, the command syntax or help system of the interface. Our hypothesis is therefore that *repeated sequences of user actions are interesting sequences and may indicate problems with the user interface.* The ability to detect such repeating patterns should be useful to interface evaluators.

MAXIMAL REPEATING PATTERNS

A session transcript can be modelled as a string of characters where each character corresponds to a command. The problem of detecting repeating sequences of command strings in the transcript is equivalent to detecting repeated substrings of characters in a string. Since our interest is in sequences of user actions, then only substrings of length at least two are considered.

Detecting repeated substrings presents some difficulties. Consider the string "abcdabcdxab". Which repeating substrings should be reported? It is more efficient to report just the substring "abcd," rather than, for example, "abc" and "bc" as well, since any substring of a repeating substring must also repeat. It is doubtful that reporting "abc" and "bc" in addition to "abcd" yields more information than reporting just "abcd" as the repeating substring. Note, however, that the substring "ab" also occurs independently of the substring "abcd," i.e., after the "x." Such substrings should also be reported since they may represent a task that occurs in more than one context, e.g., as part of a larger repeating pattern "abcd," or independently of that pattern.

THE REPETITION DETECTION ALGORITHM

The algorithm for detecting repetition is based on a data structure known as a position tree, which records positions of each repeating substring in the string. Maximal repeating patterns are extracted from this data structure by traversing the tree and deleting those repeating substrings which occur within the longest repeating substring, except for those which occur independently of the longest one. This is easily done by comparing the positions at which the substrings occur. This process is repeated from the longest repeating substring to the next longest remaining substring, until no more substrings can be deleted. Precise time complexities are not yet available for this algorithm, but in practice it takes only a few seconds on a Mac II to scan a string thousands of characters in length.

INTERACTIVE TOOL SUPPORT FOR MRP DETECTION

Initial implementations of this algorithm which were run on a session file representing approximately three months of one user's activity extracted hundreds of maximal repeating patterns. This led to an interactive version of the algorithm which allowed management of this list of maximal repeating patterns, and for the extraction of patterns from this list by specific attributes such as length of the pattern. Other information such as the number of patterns by length and occurrence were provided as an aid to deciding which patterns to investigate further.

RESULTS

This MRP technique was used in the evaluation of a large real world image processing system. Transcripts of seventeen users, totaling 17,086 command lines, were analyzed over an eight hour period. The average number of MRPs per user was 207, and average length of the longest MRP was 16. Some of the MRPs were noteworthy on the basis of violating expected usage patterns of commands. For example, ten commands were found in

MRPs consisting of consecutive invocations of the same command, possibly indicating that a user needs to perform the same command on several objects, or that the user is "fine-tuning" a single object. For example, a user may have a list of files that need to be converted from one format to another, or a user may be debugging a macro. In the first case, a possible remedy could be to allow an arbitrary number of arguments for each such command. The second case demands a closer study of the nature of the "fine-tuning." Since 82% of the sample users exhibit this MRP type, this indicates a problem inherent in the interface design, rather than a collection of user idiosyncrasies.

Another type of MRP that was detected consisted of consecutive lines where no commands were entered, indicating anomalous use of the command line terminator. This MRP type may be due to factors such as poor keyboard design, defective keyboards, or long response times. Since 42% of users exhibited this MRP, and because of the long experience designers have with keyboards, system response time is the more likely cause.

In summary, results demonstrate that the MRP technique was useful for finding specific problems in the interface. Although MRPs did not show the root cause of a problem, much less indicate solutions, they did identify specific, real problems. It is therefore reasonable to expect that this technique would work for other command line-based systems. While the MRP technique does not directly show general problems with an interface, detailed problems are also important, and consequently the technique could be part of a larger interface evaluation methodology.

This preliminary evidence suggests that the repetition hypothesis is justified and that the MRP technique is convincingly useful. Several extensions to the technique are being considered, including its use in conjunction with standard videotaping methods where repetition analysis of the session file could be used to identify interesting sections of the videotape, thereby eliminating the need to review the entire videotape.

REFERENCES

1. Card, S. K., Moran, T. P., and Newell, A. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum, Assoc., New Jersey, 1983.

2. Mackay, W. E.et al. Video: Data for Studying Human-Computer Interaction. In Proceedings of *CHI'88 Conference on Human Factors in Computing Systems* (Washington, D. C., May 15-19). ACM, New York, 1988, pp. 133-137.