Journal of Visual Languages and Computing 19 (2008) 485–498

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Using scenario-based requirements to direct research on web macro tools ☆

Christopher Scaffidi [a,*], Allen Cypher [b,1], Sebastian Elbaum [c,2], Andhy Koesnandar [d], Brad Myers [e,3]

[a] Carnegie Mellon University, 4104 Wean Hall, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
[b] IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA
[c] Computer Science and Engineering Department, 261 Avery Hall, University of Nebraska, Lincoln, NE 68588, USA
[d] Computer Science and Engineering Department, University of Nebraska, Lincoln, NE 68588, USA
[e] Carnegie Mellon University, 3517 Newell-Simon Hall, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

ARTICLE INFO

Keywords:
End-user programming
Web macros
Scripting

ABSTRACT

Web macros automate the interactions of end users with web sites and related information systems. Though web macro recorders and players have grown in sophistication over the past decade, these tools cannot yet meet many tasks that people perform in daily life. Based on observations of browser users, we have compiled ten scenarios describing tasks that users would benefit from automating. Our analysis of these scenarios yields specific requirements that web macro tools should support if those tools are to be applicable to these real-life tasks. Our set of requirements constitutes a benchmark for evaluating tools, which we demonstrate by evaluating the Robofox, CoScripter, and iMacros tools.

1. Introduction

Information workers, webmasters, and other people often need to perform repetitive tasks in a web browser, which is tedious and error-prone. In response, researchers and companies have provided numerous tools intended to help users automate web browser tasks [1,4,5,10,11,12,13,18,22]. Typically, these tools follow the programming-by-example (PBE) paradigm: a recorder watches the user perform operations, determines the user's intent, and generates a "web macro" to represent that intent (generally as a sequence of steps). Later, a macro player executes the macro on new data. Applying PBE to web browser automation has seemed promising, since the paradigm had previously been successfully applied in other contexts, such as user interface design [14] and HyperCard programming [3].

Web macro tools offer several benefits. First, they offer significant time savings to users. In addition, when a procedure in a large web application is complex and hard to learn, then users who have mastered that difficult procedure can create a web macro as a teaching tool for other users, in order to encapsulate and communicate the steps required to perform the procedure. Finally, web application developers can use web macros to create automated test suites.

Given these promised benefits and the arrival of commercial web macro tools (e.g. [7]), it might seem that web macro tools should be in widespread use, particularly among information workers, whose work in

☆ Portions of this article previously appeared as C. Scaffidi, A. Cypher, S. Elbaum, A. Koesnandar, and B. Myers, Scenario-based requirements for web macro tools, in: VL/HCC'07: Proceedings of the 2007 IEEE Symposium on Visual Languages and Human-Centric Computing, 2007, pp. 197–204.

* Corresponding author. Tel.: +1 412 268 3564; fax: +1 412 268 5576.
E-mail addresses: cscaffid@cs.cmu.edu (C. Scaffidi), acypher@us.ibm.com (A. Cypher), elbaum@cse.unl.edu (S. Elbaum), akoesnan@cse.unl.edu (A. Koesnandar), bam@cs.cmu.edu (B. Myers).
[1] Tel.: +1 408 927 2513; fax: +1 408 927 2100.
[2] Tel.: +1 402 472 6748; fax: +1 402 472 7767.
[3] Tel.: +1 412 268 5150; fax: +1 412 268 1266.

1045-926X/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2008.04.002

web browsers is highly repetitive [21]. Moreover, users seem to be generally capable of understanding the process of recording and replaying macros, as 42% of information workers report that they or their subordinates recorded spreadsheet macros in the past 3 months, and 33% similarly report recording of word processor macros [20].

Despite these factors, web macro tools do not seem to be in widespread use. One reason is that the web context introduces new challenges that did not apply in traditional PBE. One such challenge is the frequent changes to web sites' structure, which can cause web macros to fail without warning. In addition, whereas many traditional macro tools only need to operate in one environment (such as HyperCard [3]), many office tasks that involve a web browser also involve other applications, such as spreadsheets. Automating these tasks requires a web macro tool to support inter-application integration.

The contribution of this paper is a methodical characterization of the requirements that web macro tools must support in order to be useful for many real-world tasks. An additional contribution is to demonstrate that these requirements serve as a helpful benchmark for evaluating tools and identifying beneficial areas of work. The requirements that we have identified include support for triggering macros, using objects on web pages, adapting to site changes, reading and writing data outside of pages, transforming data, executing control structures, recovering from failure, and supporting macro maintenance.

To help ensure the validity of these requirements, we base them on a range of real-world tasks that should ideally be automatable with web macros. We selected these tasks because automating them would offer clear benefits to end users. For example, automating one time-consuming task was so desirable to one end user that he paid a professional PHP/Perl programmer to automate the task; open source programmers automated two other tasks to help people. Certain tasks are ones that we regularly perform, and we would like to automate these tasks to save ourselves time, but we have no suitable PBE macro tool. Finally, we have observed co-workers manually performing particular tasks, and automating these tasks would offer significant time savings.

We do not claim that our list of requirements is complete in the sense that satisfying them will necessarily make tools perfect for all imaginable tasks. Instead, by linking requirements to a diverse set of specific tasks, our benchmark indicates the wide range of real-world applicability that a tool would gain by satisfying certain requirements. In addition, the benchmark constitutes a seed that can grow as researchers contribute more scenarios where macros would be beneficial.

Section 2 presents related work. Section 3 uses a scenario format to describe tasks, and Section 4 analyzes scenarios to identify tool requirements. Section 5 demonstrates using requirements as a benchmark for evaluating the Robofox, CoScripter, and commercially available iMacros tools, thereby identifying areas for future work.

2. Related work

Many papers use scenarios to motivate and explain a PBE tool's features [1,4,5,10,11,12,13,18,22]. Generally, a paper first presents the scenario in a succinct form to motivate the work; later, the paper describes the scenario in some additional detail and discusses how to use a new tool to automate the scenario.

For example, a paper on the Turquoise web macro tool presents a scenario of combining clippings from web sites into a newspaper, and the paper also discusses a scenario of repeatedly submitting a web form in order to purchase sandwiches [13]. As another example, a paper on the Creo tool describes a scenario of reading a recipe on a web site, then copying each ingredient into a web form on another site to compute the recipe's nutritional value [4].

Such scenarios meet the intended purpose of motivating and demonstrating a new tool. However, they have two limitations.

First, each paper generally only mentions one or two scenarios and rarely describes any scenario details that are unsupported by the tool. Thus, such scenarios rarely highlight opportunities for extending the tool, they provide limited support for evaluating future tools, and they offer no uniform way to compare the capabilities of different tools.

Second, the "pedigree" of scenarios is rarely documented: that is, it is usually unclear whether each scenario was identified by observations of end users or if it is hypothetical. Consequently, it is difficult to determine if supporting the scenario will make the tool useful in practice.

In this paper, we specifically select a variety of scenarios that highlight opportunities for future work. In addition, we have documented the source of each scenario, providing traceability to help ensure that automating each scenario would give real benefit to users.

The identified requirements comprise a benchmark to measure tool improvement. Our intent is similar to that behind the Test Suite for Programming by Demonstration [16], a benchmark for traditional (non-web) PBE tools. Like that Test Suite, our benchmark illustrates the wide range of potential applications for tools and enables researchers to test tools with real-world tasks.

Our requirements can also be used to guide the design of future tools. In this sense, our intent is similar to that of user studies that have uncovered requirements for end-user programming tools other than web macro tools. Examples include a survey of programmers that influenced the design of the FlashLight tool for creating web applications [17], as well as field studies of system administrators that guided the design of the A1 tool for creating scripts that read and write data from servers [9]. Empirical studies have also preceded the design of programming tools for other particular populations such as children [15] and females [2]. These studies, like our work, are an application of the standard user-centered design process to the specific domain of end-user programming languages and environments.