Behavior Research Methods
2009, 41 (1), 1-12
doi:10.3758/BRM.41.1.12

Timing accuracy of Web experiments: A case study using the WebExp software package

Frank Keller, Subahshini Gunasekharan, Neil Mayo, and Martin Corley
University of Edinburgh, Edinburgh, Scotland

Although Internet-based experiments are gaining in popularity, most studies rely on directly evaluating participants' responses rather than response times. In the present article, we present two experiments that demonstrate the feasibility of collecting response latency data over the World-Wide Web using WebExp—a software package designed to run psychological experiments over the Internet. Experiment 1 uses WebExp to collect measurements for known time intervals (generated using keyboard repetition). The resulting measurements are found to be accurate across platforms and load conditions. In Experiment 2, we use WebExp to replicate a lab-based self-paced reading study from the psycholinguistic literature. The data of the Web-based replication correlate significantly with those of the original study and show the same main effects and interactions. We conclude that WebExp can be used to obtain reliable response time data, at least for the self-paced reading paradigm.

1. INTRODUCTION

Over the past 10 years, it has become more and more common to use the World-Wide Web to collect data for psychological research. This development has been driven by the advantages of Web-based experimentation, which include the possibility of recruiting large numbers of participants, access to diverse participant populations, and cost-effectiveness (see, e.g., Birnbaum, 2000). In psycholinguistics, the Web has the added advantage of making it easy to work on languages for which lab-based samples are difficult to obtain. This has led to an increased interest in Web experimentation among psycholinguists (e.g., Featherston, 2005; Keller & Alexopoulou, 2001).

The majority of Web experiments are designed to collect data in the form of ratings, response accuracies, or response frequencies. Such data can be gathered straightforwardly using purpose-built Web applications that can be implemented using HTML forms and CGI scripts. This process is made easier by the availability of tools that automatically generate experiments of this type (Naumann, Brunstein, & Krems, 2007; Rademacher & Lippke, 2007; Reips & Neuhaus, 2002). Although this approach to Web-based data collection has yielded a large number of useful results, psychologists often rely on time-critical experiments to test their hypotheses. To realize such time-critical experiments, the stimulus presentation has to be timed precisely, and response times have to be measured accurately. In a laboratory setting, millisecond accuracy can be achieved using specialized software and purpose-built hardware (see, e.g., Forster & Forster, 2003; MacInnes & Taylor, 2001; McKinney, MacCormac, & Welsh-Bohmer, 1999; Myors, 1999).

It is less straightforward for one to achieve comparable accuracy for Web-based experiments, which can be expected to yield time measurements with high variance. This is partly due to the nature of Web experimentation: Participants are self-selected and may participate multiple times or drop out of the experiment. There is reduced control over the experimental environment (noise, distractions), and limited interaction between participant and experimenter (Reips, 2002). Although some attempts can be made to reduce noise that is due to human factors (see section 4.2 for an example), technological considerations are equally important. Participants taking part in Web experiments are likely to use a range of different platforms to run the experiment, and these can differ in computer hardware (processor type, clock speed), input and output devices (screen refresh rate, keyboard and mouse latency), and software (operating system, Web browser). All of these factors can affect timing accuracy. Another potential problem is the processor load on the participant's computer; if the processor is running a number of other applications concurrently with the experiment, then this can be expected to affect timing. Network latency can also be a problem if the experiment relies on frequent network access—for example, to download large audio or movie files.

In order to reduce the effect of the diverse hardware and software used by participants, previous work has relied on platform-independent client-side technology to deliver Web experiments, in particular Flash (Reimers & Stewart, 2007) and Java (Eichstaedt, 2001). This approach can be expected to give the experimenter an increased amount of control over stimulus timing and response time measurements. Encouraging results have been reported for Flash, which has been found to measure response times for a

F. Keller, frank.keller@ed.ac.uk

binary choice task with similar accuracy over the Web, in the lab, and as compared with a native implementation (Reimers & Stewart, 2007). Using Java, Eichstaedt found that timing accuracy deteriorated on computer systems with less powerful hardware and increased system loads. He suggested a way of addressing this problem, using a filter that detects inaccurate time measurements (see section 2.2). For stimulus presentation, the situation is reversed: Research has shown that less powerful hardware leads to a degradation of timing accuracy for Flash, but not for Java (Schmidt, 2001).

This article further investigates the timing accuracy that can be achieved in Web experiments. We focus on the measurement of response times using WebExp, a Java-based experimental software package. Our initial evaluation of the software follows previous work by Eichstaedt (2001) and Schmidt (2001) by measuring artificially generated time intervals of known duration. But, in order to fully investigate the feasibility of measuring timed responses, we also conduct a Web-based replication of a full-fledged reading time study from the psycholinguistic literature. In contrast with previous evaluations (see, e.g., Reimers & Stewart, 2007), our replication involves a multifactorial design with several analysis regions; thus, it provides a realistic case study for the usefulness of Web-based response time measurements. To our knowledge, no comparable replication has been attempted before (although Web-based response time measurements have been reported in previous psycholinguistic studies by Corley & Scheepers, 2002, and by East, 2005).

The present article is structured as follows. In section 2, we introduce the WebExp software package on which we base our evaluation, with particular focus on timing issues. In section 3, we present our first experiment, which evaluates the accuracy of time measurements of known intervals. Section 4 then presents a full replication of a study from the psycholinguistic literature (Sturt, Pickering, & Crocker, 1999). Conclusions and future work are presented in section 5.

2. THE WebExp PACKAGE

This section provides an overview of the WebExp package for Web-based psychological experiments. We describe WebExp's client-server model, explain how experiments are specified, and give an overview of the main features of the software. This is followed by a more detailed description of WebExp's timing component, whose evaluation is the main focus of this article.[1]

2.1. Architecture and Features

Client–server model. WebExp is implemented as a client–server system consisting of two separate programs that interact with one another. The WebExp server is an application that runs on the Web server that hosts the experiment. It waits for clients to connect to it and request experiments, whereupon it provides (serves up) the experimental files and manages the results that are returned. The WebExp client is implemented as an applet that runs in the browser of the participant. It connects to the server to receive the specification of the experiment, administers it to the participant, and returns the results to the server. The HTML for the experimental Web page does not have to be hosted on the same server as does WebExp.

WebExp is written in Java and uses XML as the description language for system configuration, experiment specification, and reporting of results. Java is designed to be platform independent and is particularly suited to running across the Internet. The WebExp server can run on Windows, Linux, and Mac OS, and anyone with a Java-capable browser can run the experimental client. As a data-description language, XML is standardized, flexible, and supports simple validation of the data.

Experiment specification. In WebExp, experiments are designed around a sequential stimulus–response model. An experiment consists of a timeline that is broken down into stages (e.g., instructions, practice, actual experiment). Each stage in turn consists of a sequence of slides, each of which involves the presentation of stimuli and the collection of responses (a range of options are implemented both for stimulus presentation and for response collection, detailed below). This approach makes it easy to design new experiments in existing paradigms (a range of examples is distributed with WebExp). It is also straightforward for one to implement entirely new experimental paradigms without the experimenter having to know anything about the code behind the system. All that is required is an understanding of the XML language used to specify experiments. This language is introduced in detail in the WebExp manual (Mayo, Corley, & Keller, 2005) and is reasonably intuitive for nontechnical users, in particular if they are familiar with HTML for Web-page design.

Instead of explicitly describing every slide in an experiment that contains a sequence of similar slides, WebExp makes it possible to describe just a single template together with the number of repetitions required. WebExp will then import different stimuli into each slide according to the experimenter's specification. For example, to implement a Stroop experiment, the experimenter only has to define the layout of each item slide, and can separately define the words and colors that are to be used in each slide.

The use of templates therefore achieves a level of abstraction in the specification of experiments by separating the form and content of experimental items. To reinforce this separation, lists of stimuli are specified in separate resource files, whereas the specification of the experiment describes the template, plus an import statement for the required resource files. Advantages to this approach are that it simplifies experiment descriptions while making it clear what the stimuli are and how they relate to one another; stimuli can also be grouped into blocks and rearranged (e.g., randomized) without affecting the structure of the experiment.

Another abstraction available to the experiment designer is the use of globals, which are global values that can be defined once in an external file and imported into the experiment specification wherever needed. This makes it possible to affect the overall design of an experiment by changing a single value in a file that contains all such variables—for example, to change the color or width of all stimulus elements simultaneously, or to change the text used in a prompt or on a button. In addition, an experiment can have several versions, each of which uses the same experimental structure but imports its own resources. For example, this can be used to build an experiment that provides stimuli in different languages according to the version requested. The version feature also allows for swift prototyping of a new variation on an existing experiment.

Features. WebExp gives the experimenter full control over the visual properties of the experiment. Font properties and background and foreground colors can be specified for each stimulus, or default properties for the whole experiment can be provided that may be overridden by individual slides or stimuli. WebExp provides a flexible visual layout manager that gives control over both horizontal and vertical alignment and positioning. The experimenter can specify proportional row heights and column widths and then position the visual stimuli with respect to this grid.

WebExp supports a range of stimulus types, including text, images, animations, and audio files. There is also a variety of response types: Questionnaire-type input can be collected through text boxes, selection lists, and radio buttons. Simple keystrokes (for keyboard and mouse) and click buttons are also supported. Furthermore, WebExp is designed in a modular fashion that makes it easy for a skilled Java programmer to add new stimulus or input types if required. The experimenter can make inputs mandatory if required and can place restrictions on the length and type of textual input (e.g., only allow letters or numbers). During the experiment, WebExp will not only display the stimuli, but will also output a progress indicator that tells the participant which slide in which stage of the experiment they are viewing currently.

The software makes it possible to randomize sections of the experiment or to reorder slides by hand (without having to change their sequence in the experiment specification file). Any reorderings are recorded in the results file. WebExp currently supports free randomization and blocked randomization; due to its modular design, adding new randomization schemes is straightforward.

WebExp allows the experimenter to use responses that have already been collected in later sections of the experiment. This makes it possible, for instance, to implement magnitude estimation experiments (Lodge, 1981) in which the participant first specifies a reference rating, which will be displayed as a standard of comparison for subsequent ratings.

The software supports comprehensive logging of experimental activity, recording potential and actual problems. The WebExp server generates log files that record attempts of clients to connect and describe the data transfer. The WebExp client produces log messages on the Java console and also sends back log files to the server at the end of the experiment. These logs provide valuable information both for experiment development (debugging) and while the experiment is running, since they reveal information about the participants (IP address, browser, operating system). They also make it easy to identify failed, incomplete, or multiple participation by a given participant.

The WebExp server is designed to administer multiple experiments by multiple experimenters. It provides automated administration of all the relevant files and enforces the separation of each experimenter's experimental descriptions and results. The server design also ensures that participant data is secure from remote access.

2.2. Timing

WebExp's timing component uses Java's `System.currentTimeMillis()` to measure response time; `currentTimeMillis` is a native method that makes a direct call to the operating system through Java native interface to capture time. Java 1.5 introduced a new native method, `System.nanoTime()`, that can provide greater accuracy, but this is still only as precise as the underlying system timer. So, in order to avoid giving a spurious impression of nanosecond accuracy, we have continued to use the former method.[2]

As for determining the accuracy of time measured, WebExp uses a timing filter similar to Eichstaedt's (2001) inaccurate timing filter. WebExp has a control thread that measures time of known duration. This control thread is run parallel to the main thread that controls the experiment and measures response time. If the time measured by the control thread is inaccurate, then the response time measured at the same period by the main thread is discarded. Hence, the control thread is used to identify whether the response time measured by the main thread is accurate. However, instead of filtering out and discarding all inaccurate measurements, as was implemented by Eichstaedt, WebExp's timing filter retains all time measurements and assigns a filter value to each. Experimenters can discard any of these readings at their own discretion.

The WebExp timing filter schedules a Java `TimerTask` that should provide a tick every 2 msec by use of the `sleep()` method, and compares the time elapsed in ticks to the time elapsed as measured with calls to `System.currentTimeMillis()`. Experimenters should be aware that both of these method calls have their limitations. The use of `sleep()` does not guarantee that a thread will be run after the given delay, but only makes sure that a thread is available to be woken after the delay. `System.currentTimeMillis()` provides a reading whose granularity depends on the underlying operating system. This issue will be investigated empirically in Experiment 1 below.

Apart from measuring response times, WebExp also provides functionality for stimulus timing. It makes use of the `java.util.concurrent` package available since Java 1.5 to control the slide elements at various levels—from the automatic advancement of a slide to the automatic presentation or withdrawal of stimuli—and the execution of other events, such as changing the color or content of a stimulus.

```
<component>
    <start>5000</start>
    <name>my sound</name>
    <type>audio</type>
    <content>bell</content>
</component>

<component>
    <name>my counter</name>
    <type>counter</type>
<bgcolor>red</bgcolor>
<fgcolor>yellow</fgcolor>
    <startnum>5</startnum>
    <endnum>0</endnum>
    <increment>1</increment>
    <interval>1000</interval>
</component>
```

Figure 1. Two examples for timed WebExp stimuli. The first component plays an audio stimulus after a delay of 5,000 msec. The second generates a counter that counts down from 5 to 0 in increments of 1,000 msec.

Events can also be scheduled so that they occur repeatedly. In this case, the `java.util.concurrent.ScheduledThreadPoolExecutor` concurrent task scheduler is used. When scheduling repeated events, there is a choice between `scheduleAtFixedRate()` and `scheduleWithFixedDelay()`; WebExp uses the latter, assuming that the relative timing of events is more important than the overall timing. This means if there is a delay in performing an event, subsequent events do not get bunched up with one another but are shown relative to the delayed event. This is less suitable for timers expected to maintain accuracy, but should provide a better visual experience for showing a repeated sequence of images, for example. Figure 1 gives two examples for the specification of timed stimuli in WebExp.

When a scheduled event executes in the slide—such as the presentation or withdrawal of a stimulus—the onset time is recorded, along with the name of the component affected, the action invoked, and the filter value. The actual accuracy of repeated events can then be assessed by the experimenter with reference to these values. Similarly, the timing component also measures response times when an event in the slide is triggered by user interaction, such as a buttonpress or keyboard input. Examples for the timing information recorded in the WebExp output are given in Figure 2.

3. EXPERIMENT 1
Measuring a Known Interval

In order to obtain an estimate of the accuracy of WebExp's timing component, we carried out an experiment in which WebExp was used to measure the duration of an interval of known duration. The aim was to find out whether system configuration and system load have an influence on timing; we tested two different system configuration and two load conditions, thus providing a partial replication of Eichstaedt's (2001) studies on time measurements using Java. Another aim of this experiment was to determine whether WebExp's timing filter is able to detect inaccurate time measurements.

This timing filter adopts a similar approach to that of Eichstaedt's (2001) inaccurate timing filter (see section 2.2): A control thread runs parallel to the main thread and measures a known time interval (a scheduled task) repeatedly. It keeps count of the number of times the interval was repeated while the main thread controls the experiment and measures response time. The accuracy of the response time measurement is then determined by calculating the number of missed intervals (MIs). If there are too many MIs in a given measurement, the experimenter can decide to discard the measurement.

To implement this approach, the duration *D* of the intervals to be counted has to be set in advance. Therefore, before conducting experiments to gauge the effectiveness of WebExp's timing component and filter, we carried out an experiment to determine a suitable value for *D*. This was to ensure that the control thread would work properly on different operating systems. The results show that a *D* of 40 msec is adequate for Linux platforms, whereas an adequate *D* for Windows platforms is 50 msec. A detailed description of the experiment to determine *D* can be found in Gunasekharan (2007).

3.1. Method

When a key is pressed, the operating system generates a signal that indicates a keystroke. If the key is held down, then it waits for a specified time (the keyboard delay d) before it generates another keystroke signal. After that, it generates more signals, with a specified interval between them (the keyboard repetition rate r), until the key is released. If we assume that n is the number of keystroke signals generated, then we can compute the expected total time (t_e) elapsed between pressing the key and releasing it as

$$t_e = d + r(n - 1). \tag{1}$$

Keystroke repetition can therefore be employed to generate an interval of known duration. We designed a keystroke repetition experiment with WebExp, consisting of the presentation of a single slide that had a text input box capable of accepting a maximum of 31 characters. Once the experiment slide was successfully loaded, a key was pressed and held, which meant that one keystroke and 30 repetitions were generated ($n = 30$). WebExp records the time of the first keystroke and the time at which the maximum character limit is reached. This interval—the observed total time (t_o)—should in theory be equal to the expected total time (t_e). If we set the keyboard delay to be identical to the repetition rate ($d = r$) to simplify Equation 1 and divide by n, then we get t_a, the observed mean time per keystroke:

$$t_a = \frac{t_o}{n}. \tag{2}$$

In the present study, we tested whether t_a is equal to r, as predicted. We experimented with a range of keyboard repetition rates under both Linux and Windows. For the Linux platform, the repetition rates used were 10, 20, 25, 50, and 100 msec. For the Windows platform, we used repetition rates of 300, 500, 700, and 1,000 msec. (Note that the minimum key repetition rate on Linux is 10 msec, and that for Windows is 300 msec.) For each repetition rate, the experi-

```
<stimulus_timings name="null">
    <!-- the time when the counter was shown (in this case it took
    about 10s to get to this slide) -->
    <event action="present" stimulus="my sound">
        <time>10007</time>
        <filter>1</filter>
    </event>
    <!-- the time when the sound was played (note it is about 5s after
    the counter starts) -->
    <event action="present" stimulus="my sound">
        <time>15059</time>
        <filter>1</filter>
    </event>
</stimulus_timings>

<display.CounterStimulus name="my counter">
    <countTime>
        <time>15051</time>
        <filter>1</filter>
    </countTime>
</display.CounterStimulus>
```

Figure 2. Partial timing results for the two stimuli in Figure 1, as output by WebExp. The first block shows the times (relative to the start of the overall experiment) of the events, such as when a stimulus is presented or withdrawn. The second block shows the last count time (relative to start of the overall experiment) recorded by the counter. Note that not all events are recorded, since the counter can continue indefinitely.

ment was run 10 times. The information about missed intervals produced by the timing filter was also recorded to determine whether it could be used to identify inaccurate measurements.

Two different computer/operating system/browser combinations were employed for this experiment, as are detailed in Table 1. Both configurations were tested under two different loads: low and high. Under low load, any known background applications such as virus scan, firewall, and instant messaging were disabled. Only the browser and the Java Virtual Machine were enabled. For the high-load condition, additional applications were run simultaneously with the experiment (see Table 1 for a list of the applications run).[3]

3.2. Results and Discussion

Table 2 shows the observed mean times per keystroke (t_a) as measured by the keystroke repetition experiment on the Linux platform. We observe that in the low-load condition, the mean time measured by WebExp is very close to the expected time for all keyboard repetition rates. Under high load, we observe a slight increase in the mean time measured (up to around 1 msec), and the variance of the measurements also increases (as can be seen from the larger confidence intervals). Overall, however, it seems that the additional applications run under high load did not disrupt WebExp's timing measurements when conducted on a sufficiently powerful computer. This contrasts with Eichstaedt's (2001) findings, which indicate that a heavily loaded Java Virtual Machine can have considerable impact on time measurements.

Under low load, WebExp's inaccurate timing filter did not report any missed intervals; hence, none of the measurements had to be removed. However, it was observed that the control thread of the filter froze when the experiments were run under high load. At first it was suspected that this might be due to the scheduled task's duration of 40 msec, but changing this value did not make any difference. (These problems seem to be operating system specific; they did not arise under Windows—see below.)

Table 3 shows the observed mean times per keystroke (t_a) as measured by the keystroke repetition experiment on the Windows platform. Under low load, t_a closely matched the expected time for the repetition rates of 300, 700, and 1,000 msec. However, the measurement for the smallest repetition rate of 300 msec overestimates the expected time by about 13 msec. In the high-load condition, the same pattern was found: The measurement was too high for the 300-msec repetition rate (by 22 msec), whereas the measurements for the other repetition rates were close to the expected values. This confirms the result that high load is not a problem for WebExp if it is run on a sufficiently powerful computer.

The MI values returned by WebExp across the different runs for each experiment under low load again indicated that no measurements needed to be removed. However, for the experiments conducted with a keyboard repeat rate of 300 msec under high load, two measurements had high

Table 1
Configurations Used for the Linux and Windows Platforms, and Applications Run for the High-Load Condition

	Linux	Windows
Computer	Dell Desktop	Dell Inspiron 5150 Laptop
Operating system	Linux Fedora Core 3	Windows XP Professional (SP2)
Browser	Mozilla Firefox 1.5.0.4	Internet Explorer 6
Processor	Intel Pentium CPU 3.00 GHz	Mobile Intel Pentium CPU 3.06 GHz
Keyboard	Hewlett Packard	Standard 101/102 key
Java version	Sun Java 1.5.0_04	Sun Java 1.5.0_05
	Applications Run for High-Load Condition	
	GAIM Internet Messaging	Yahoo Messenger
	Google Video	MSN Messenger
	BBC Radio	Zone Alarm
		Norton Antivirus
		Google Video

MI values. Removing these readings resulted in a lower standard deviation and a lower time difference from the expected time, as is indicated in Table 3.

To summarize, the results in this experiment indicate that WebExp is capable of accurately measuring the length of a time interval of known duration on a Linux platform. The accuracy obtained on the Windows platform is also good, although reduced for the shortest time interval investigated (300 msec, the shortest possible repetition rate under Windows). We also found that Eichstaedt's (2001) inaccurate timing filter was only useful in detecting inaccurate measurements under Windows for short intervals under high load.

4. EXPERIMENT 2 Measuring Reading Times

Experiment 1 showed that WebExp is able to measure the duration of a known time interval reliably, which raises the question of whether it can also be used to run realistic, time-critical psychological experiments. The present experiment aims to test this question by using WebExp to replicate a reading time experiment for which there are published and generally accepted results.

This experiment used the self-paced reading (SPR) paradigm introduced by Just, Carpenter, and Woolley (1982). This paradigm allows readers to use buttonpresses to control the exposure duration for each word of a text they read. The latencies of the buttonpresses depend on the properties of the words being read and correlate with the time course of the cognitive processes during reading and text comprehension. In their original study, Just et al. investigated three different variants of the SPR paradigm: cumulative window, moving window, and stationary window. However, a range of studies (Garnsey, Pearlmutter, Myers, & Lotocky, 1997; Just et al., 1982; Thornton, MacDonald, & Arnold, 2000) have shown that the moving-window paradigm most closely resembles natural reading, in the sense of replicating the results of eyetracking data for reading.

In a moving-window SPR experiment, a text is first displayed as a series of dashes on the screen, with each dash representing a character in the text. When the participant makes the first buttonpress, this causes the first word to appear, replacing the dashes that correspond to that word. Subsequent buttonpresses cause the previous word to be replaced by dashes while the current word is shown. Only one word is visible at any given time, thus creating the impression of a moving window of words on the screen. Figure 3 shows an example of a sentence presented using moving-window SPR.

For the purpose of evaluating WebExp's timing component, we replicated a published lab-based SPR experiment from the well-known sentence processing study of Sturt et al. (1999). The results include both small and large reading time effects; hence, a replication makes it possible to obtain a realistic estimate of the accuracy of WebExp timing.

4.1. Original Experiment

The original study by Sturt et al. (1999) investigated reanalysis in human sentence processing. Reanalysis happens when the sentence processor initially assumes a syntactic structure that later turns out to be incorrect and has to be revised. Sturt et al. reported two self-paced reading experiments to determine whether the type of structural change required has an effect on the degree of difficulty of the reanalysis. We replicated their Experiment 1.

Theories of sentence processing predict greater reanalysis difficulty for NP/Z ambiguities, such as Example (1c) below, than for NP/S ambiguities, such as Example (1a), when factors such as plausibility and verb bias are controlled for.[4] Sturt et al. (1999) hypothesized that it is harder to process a main clause after an initial subordinate clause (as in [1c]) than to process a comple-

Table 2
Mean Time (t_a) and 95% Confidence Interval (CI) for One Keystroke on the Linux Platform

	Low Load				High Load			
Keyboard Rate	Before MI Removal		After MI Removal		Before MI Removal		After MI Removal	
(msec)	*M*	CI	*M*	CI	*M*	CI	*M*	CI
10	10.06	0.19	–	–	10.26	0.19	–	–
20	20.00	0.08	–	–	20.22	0.11	–	–
25	25.00	0.06	–	–	25.33	0.17	–	–
50	50.02	0.05	–	–	50.21	0.08	–	–
100	99.99	0.01	–	–	101.17	2.16	–	–

Table 3
Mean Time (t_a) and 95% Confidence Interval (CI) for One Keystroke on the Windows Platform

	Low Load				High Load			
	Before MI Removal		After MI Removal		Before MI Removal		After MI Removal	
Keyboard Rate (msec)	*M*	CI	*M*	CI	*M*	CI	*M*	CI
300	312.66	0.20	–	–	321.56	13	312.65	0.22
500	500.73	0.35	–	–	500.20	0.25	–	–
700	705.15	3.79	–	–	703.59	0.32	–	–
1,000	1,001.10	0.54	–	–	996.62	7.69	–	–

ment clause after a main verb (as in [1a]), due to delay associated with the computation of the semantic relation between the subordinate clause and the main verb. They argued that this might even be the case for unambiguous constructions. To test this argument, they compared locally ambiguous items such as (1a) and (1c) with unambiguous controls, such as (1b) and (1d). They used region-by-region noncumulative self-paced reading in their experiments.

Sturt et al.'s (1999) experiment used 24 participants and 32 experimental items, such as those in Example (1) below. The items were pretested for verb bias and plausibility. They were divided into four lists, each containing one condition per item, distributed according to a Latin square design. An equal number of participants was assigned to each list. They read the items in random order; they were interspersed with 96 fillers and preceded by 5 practice items.

(1) a. NP/S ambiguous: The faithful employees / understood the technical contract / would be changed / very soon.

b. NP/S unambiguous: The faithful employees / understood that the technical contract / would be changed / very soon.

c. NP/Z ambiguous: Because the employees / negotiated the technical contract / would be changed / very soon.

d. NP/Z unambiguous: Because the employees / negotiated, the technical contract / would be changed / very soon.

```
1  --- --------- ---------- --- ---------

2  The --------- ---------- --- ---------

3  --- employees ---------- --- ---------

4  --- --------- understood --- ---------

5  --- --------- ---------- the ---------

6  --- --------- ---------- --- contract.
```

Figure 3. Illustration of the moving-window self-paced reading paradigm, with time unfolding from Position 1 to Position 6.

The experiment was run using PsyScope (Cohen, MacWhinney, Flatt, & Provost, 1993) on a Macintosh-compatible computer, and a PsyScope buttonbox was used to collect reading times. The experiment was conducted in a laboratory, with participants seated in soundproof booths. The participants used the buttonbox to move between sentence regions and to answer comprehension questions, which were presented for 50% of the sentences.

4.2. Method

We implemented Sturt et al.'s (1999) experiment using WebExp, following the original design as closely as possible. The experiment was run remotely over the World-Wide Web, hosted on a Web server of the University of Edinburgh, and listed on the Language Experiments Portal.[5] The experiment ran for 8 months (September 2006 to April 2007).[6]

Materials. Thirty-two experimental materials that were identical to Sturt et al.'s (1999) original items were used. In addition to the experimental items, 86 fillers from the original study were also reused, along with the 5 practice items.

Sentences were divided into four regions as per the original study: The first region boundary was placed immediately before the occurrence of the first verb. The second region boundary was placed immediately before the first disambiguating word. The critical region, Region 3, contained two to four words, and the last region (the spillover region) contained a phrase of two to four words. The region boundaries are illustrated by the use of slashes in Example (1).

The items were divided into eight lists, each containing 16 items in one of the four conditions, distributed according to a Latin square. The decision was made to use eight lists rather than four (as in the original study) in order to shorten the experiment. It was estimated that the experiment would take about 50 min if four lists were used and that it could be expected to lead to an unacceptably high dropout rate (Reips, 2002). The number of fillers was also reduced to 21 per participant.

Procedure. Participants were first presented with a welcome page describing the experiment and the criteria for participation, including instructions and a description of the moving window technique with example. After reading the instructions, participants clicked a button at the bottom of the page to get to the experiment's Java applet. Upon successful initialization of the applet, a short demographic questionnaire was presented that asked for the name, age, occupation, sex, handedness, language region, and e-mail address of the participants. This was followed by the practice phase, in which 5 practice items were presented in random order. This phase was meant to familiarize participants with the experimental procedure. Then the actual experiment started, and WebExp logged response times from this phase onward. The 16 experimental items were presented in random order, with 21 randomly chosen filler items interspersed. No two experimental items appeared in sequence.

As in the original experiment, the sentences were presented using moving-window SPR, with region-by-region presentation. The first three regions appeared on one line, and the fourth region appeared on the next line. Participants were required to press a mouse button to move from region to region (previous work has shown that mouse buttons yield better response time measurements than keyboard keys; see Forster & Forster, 2003). As in the original study, comprehension questions followed 50% of the experimental and filler items. Participants were required to use the keyboard keys "Y" and "N" to answer the questions. The experiment was realized using the template feature of WebExp, with each sentence implemented as a sequence of slides (one per region). Figure 4 shows a screenshot of the experiment.

Participants. Forty-eight participants took part in the experiment. They were recruited through advertisements sent by e-mail. Participation was voluntary and unpaid, and was limited to native speakers of English who were at least 18 years of age. The data of 3 participants were discarded since they reported being nonnative speakers or less than 18 years old. The remaining data were screened using a series of criteria to determine whether participants had understood the instructions and completed the experiment in earnest. We eliminated the data of participants who provided fake personal details, who completed the experiment very quickly (generating unrealistically fast reading times), or who answered less than 80% of the comprehension questions correctly. Doing this eliminated 8 participants, leaving a total of 37. Five more participants were eliminated at random to balance the lists; the data of 32 participants (eight lists with 4 participants each) was retained for analysis.

Among these 32 participants, 12 were male and 20 were female. Twenty-seven were right-handed, and the remaining 5 were left-handed. Their ages ranged from 18 to 53 years, the mean being 27.3. Regarding language region, 17 participants reported being speakers of British English (we counted responses that stated Britain, U.K., Scotland, England, or Wales), 12 participants reported being speakers of North American English (responses were U.S. or Canada), and the remaining 3 participants just gave "English" as their language region. Twenty-two of the participants were Windows users, and the remaining 10 were Linux users. There were no Mac OS users in our sample.[7] (Note that there was a small number of Mac OS users in the original sample of 48 participants, which had the following distribution: 15 Linux users, 3 Mac OS users, and 30 Windows users.)

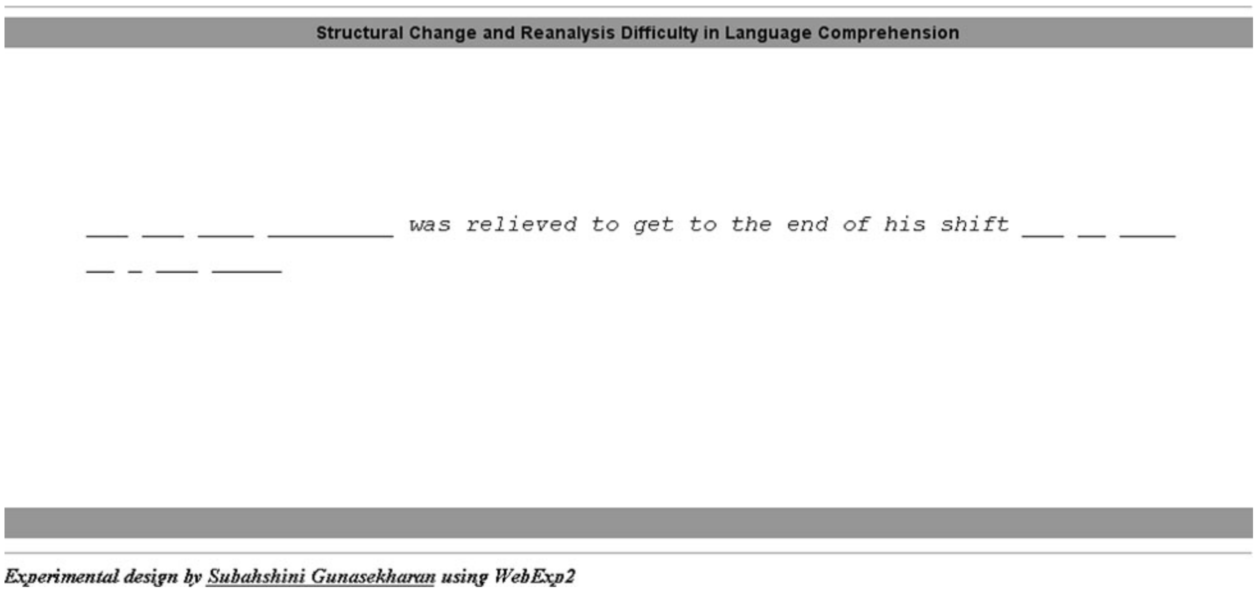
4.3. Results

The experimental data underwent the same outlier removal procedure as in the original study. All trials that included a region with a reading time that was either too short (250 msec or lower) or too long (8,000 msec or over) were removed. For the remaining data, all reading times over 2.5 *SD*s on either side of the participant mean for the region were replaced with the cutoff value.

Comprehension accuracy. Mean comprehension accuracy for the 16 experimental items that included questions was 96.1%; this is slightly higher than the mean comprehension accuracy of 92% in the original study. No significant differences were observed among the comprehension accuracy in the four experimental conditions. Mean comprehension accuracy of experimental items and fillers combined was 94.9%.

Mean reading times. To facilitate a comparison between the original and the replicated experiment, Figures 5 and 6 show the by-participant means graphically. (The by-item means match the by-participants means closely and are omitted for space reasons.) Note that the critical region of this experiment was Region 3, which is the part of the sentence in which we expect a difference between the NP/S and the NP/Z structure with respect to ambiguity resolution.

Correlation analysis. Referencing Shaughnessy and Zechmeister (1994), Krantz, Ballard, and Scher (1997) recommended the use of correlation analysis to determine the validity of a replicated psychological test. This analysis involves correlating the established measure (obtained from the original study) with the untested measure (obtained from the replicated study). To determine the validity of our WebExp experiment, we applied this approach to the reading times obtained for the critical region. The mean reading time for Region 3 of each sentence in the original study was correlated with the mean reading time for Region 3 of the same sentence in the replicated study. A significant correlation was obtained ($r = .423$, $N = 32$, $p < .001$).



Structural Change and Reanalysis Difficulty in Language Comprehension

___ ___ ____ _________ was relieved to get to the end of his shift ___ __ ____
__ _ ___ _____

Experimental design by Subahshini Gunasekharan using WebExp2

Figure 4. Screen shot of WebExp running Experiment 2.

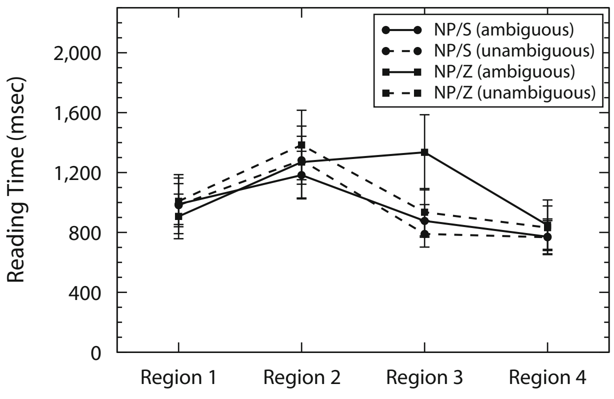


Figure 5. Mean reading times (in milliseconds) by participants in Experiment 2 (original), with 95% confidence intervals.

In order to be able to interpret this result, we need to establish what kind of *r* to expect for the validation. This can be estimated by resampling the original data. To do this, we randomly divided the participants from the original study into two halves and correlated the mean reading time for Region 3 of each sentence in the first half with the mean reading time for the same region in the same sentence in the second half. This procedure was repeated 10 times; the mean *r* obtained was .455. This value is close to the coefficient of .423 obtained when correlating the replicated original data with the replicated data, indicating that the *r* value obtained in the replication is in the acceptable range.

ANOVA. In order to establish whether our WebExp experiment was successful in replicating the pattern of results of the original study, we conducted a series of ANOVAs that mirrored the ones in the original article. Separate 2 × 2 ANOVAs were computed on the reading times for each region, treating either participants (F_1) or items (F_2) as the random factor.

An analysis of the second region in the original study revealed a main effect of ambiguity. This finding was replicated in our study [$F_1(1,31) = 8.414$, $MS_e = 218{,}823$, $p < .01$; $F_2(1,31) = 8.512$, $MS_e = 233{,}819$, $p < .01$]. In the original, there was also a nonsignificant tendency for NP/S constructions to be read more quickly than NP/Z constructions. This tendency was also observed in the replicated experiment [$F_1(1,31) = 5.788$, $MS_e = 79{,}695$, $p < .1$; $F_2(1,31) = 2.312$, $MS_e = 178{,}464$, $p = .139$]. However, in the replicated study, a marginal interaction was found [$F_1(1,31) = 4.181$, $MS_e = 158{,}576$, $p < .1$; $F_2(1,31) = 4.018$, $MS_e = 172{,}832$, $p = .054$]; such an interaction effect was not present in the original study.

In the critical third region, Sturt et al. (1999) found main effects of ambiguity and construction type, as well as a significant interaction of the two factors. In the replication, we obtained a significant main effect of construction type [$F_1(1,31) = 32.958$, $MS_e = 110{,}353$, $p < .001$; $F_2(1,31) = 31.578$, $MS_e = 107{,}855$, $p < .001$], a significant main effect of ambiguity [$F_1(1,31) = 25.674$, $MS_e = 147{,}612$, $p < .001$; $F_2(1,31) = 19.789$, $MS_e = 169{,}667$, $p < .001$], and a significant interaction [$F_1(1,31) = 19.278$, $MS_e = 151{,}092$, $p < .001$; $F_2(1,31) = 15.794$, $MS_e = 182{,}408$, $p < .001$].

In the final region, the ANOVA in the original study showed a nonsignificant tendency for NP/S constructions to be read more quickly than NP/Z constructions. In the replicated experiment, this effect reached significance [$F_1(1,31) = 11.762$, $MS_e = 118{,}186$, $p < .01$; $F_2(1,31) = 10.323$, $MS_e = 125{,}098$, $p < .01$]. There was no significant main effect of ambiguity [$F_1(1,31) = 2.150$, $MS_e = 329{,}856$, $p = .153$; $F_2(1,31) = 2.788$, $MS_e = 232{,}315$, $p = .105$] and no significant interaction [$F_1(1,31) = 2.850$, $MS_e = 158{,}662$, $p = .101$; $F_2(1,31) = 2.044$, $MS_e = 233{,}530$, $p = .163$]. These results were consistent with the original study.

Contrasts. In addition to running ANOVAs, Sturt et al. (1999) carried out a series of simple-effect analyses—that is, contrasts between pairs of means. They used one-way ANOVAs to carry out these tests (these are equivalent to paired *t* tests). They found that unambiguous sentences were read more quickly than were their ambiguous counterparts, both for NP/S and for NP/Z. In the replicated experiment, this difference was not significant for the NP/S construction (both *F*s < 1), but it was significant for the NP/Z construction [$F_1(1,31) = 29.079$, $MS_e = 229{,}509$, $p < .001$; $F_2(1,31) = 25.691$, $MS_e = 242{,}474$, $p < .001$]. Another set of simple-effects analyses in the original study showed that NP/S sentences were read more quickly than NP/Z sentences, not only in the ambiguous conditions, but also in the unambiguous condition. The result was replicated for the ambiguous condition [$F_1(1,31) = 44.827$, $MS_e = 145{,}665$, $p < .001$; $F_2(1,31) = 29.336$, $MS_e = 213{,}925$, $p < .001$], but it failed to reach significance for the unambiguous condition (both *F*s < 1).

A number of authors (e.g., Masson & Loftus, 2003) have argued that contrasts should be assessed on the basis of confidence intervals, rather than using null hypothesis significance testing. We followed this suggestion and computed a confidence interval for the contrasts between the means in Region 3, following the procedure proposed by Masson and Loftus. (Note that this confidence interval for contrasts is different from the confidence intervals for means reported elsewhere in this article.) We computed

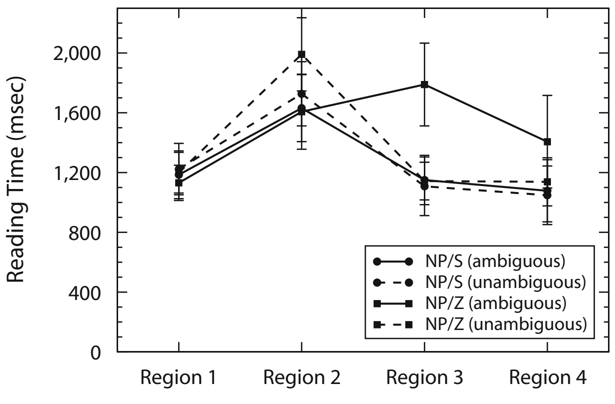


Figure 6. Mean reading times (in milliseconds) by participants in Experiment 2 (replication), with 95% confidence intervals.

the by-participants 95% confidence interval using the pooled MS_e of the two main effects and the interaction. For the original study, the confidence interval obtained was 145 msec [$t(69) = 1.995$, $MS_e = 63{,}038$]—that is, a difference between two means larger than this value can be regarded as significant. For the replicated experiment, the confidence interval was computed in the same way, resulting in a value of 183 msec [$t(93) = 1.986$, $MS_e = 136{,}352$].

These confidence intervals indicate that the reading times for ambiguous and unambiguous NP/S sentences did not differ significantly (in both the original study and the replication), whereas unambiguous NP/Z sentences were read significantly more quickly than ambiguous NP/Z sentences (in both the original and the replication). Also, ambiguous NP/S sentences were read significantly more quickly than ambiguous NP/Z sentences (in both the original study and the replication). Unambiguous NP/S sentences were read significantly faster than unambiguous NP/Z sentences in the original study, but not in the replication.

Missed intervals. We also performed an alternative form of outlier removal. Instead of removing all trials with short (less than 250 msec) and long (more than 8,000 msec) reading times as per the original study, we used WebExp's missed-interval measure to detect outliers. Reading times with high MI values (400, 300, 200, 100) were removed. After this, the remaining data were further processed as in the original study; all reading times over 2.5 *SD*s on either side of the mean were replaced with the cutoff value. The results provided no evidence for the effectiveness of the inaccurate timing filter; only a few data points were affected by the MI-based outlier removal, and the overall results (significant effects in the ANOVA) remained unchanged.

4.4. Discussion

The Web-based self-paced reading data collected in this replication experiment provided a close match with the data of the original study, which were collected under controlled lab conditions. We correlated the by-item means of the reading times for the critical region in the original experiment with those of the replication. The correlation coefficient obtained was close to the value that would be expected for a replication of the original experiment (as estimated by resampling the original data). This indicates that our replication was successful in generating reading time data that match those of a published study.

We were also able to demonstrate in a series of ANOVAs that the replication showed the same significant main effects and interactions as in the original experiment. This was true for both by-item and by-participant analyses, and in both the critical region and the other regions (less relevant for the experimental design). The results of the ANOVAs demonstrate that our replication showed all and only those effects that were present in the original study, strengthening our claim that it constitutes a full replication.

Furthermore, we computed a series of contrasts that compare per-condition means for the critical region. The original article reported four tests for contrasts, all of which were significant. In the replication, only two of these four contrasts were significant when the same method as in the original article was used (*t* tests between means). Arguably, however, this method is too liberal, since no correction for multiple testing is applied, thus increasing the risk of a Type I error. We therefore used an alternative way of assessing contrasts between means based on confidence intervals, as was suggested by Masson and Loftus (2003). When we applied this approach to both the original data and the replicated data, we found that the results agreed on three out of four contrasts (two significant, one nonsignificant), and disagreed on one contrast (significant in the original, but not in the replication). This provides further evidence that the replication matches the original closely.

The confidence intervals for contrasts are also interesting from the point of view of estimating the timing accuracy of the Web-based method we used to collect the reading-time data. The size of the confidence interval in the original study was 145 msec. This is the smallest significant difference that the original study could detect. In the replicated data, the corresponding value is 183 msec. Although this is larger than in the original, it is of a similar order of magnitude, thus demonstrating that even fairly small differences in reading time can be reliably measured using a Web-based experiment. At this point, it is important that we note that the replication used a smaller number of items per participant than the original (16 instead of 32) and increased the total number of participants (32 instead of 24) to compensate for this. As a result, any differences in the size of confidence interval for contrasts needs to be interpreted with caution, since it could also be due to a difference in statistical power between the experiments.

We made another observation regarding the replicated data. As Figures 5 and 6 show, the reading times in the replication were consistently higher than in the original. This may be due to the fact that our sample of participants was more diverse; it included speakers of both British and North American English and comprised a fairly wide age range (see section 4.2). Presumably, our participants came from various social and educational backgrounds (although we have no data on this). Sturt et al. (1999) stated that their participants were from the University of Glasgow, which presumably means that the sample included mainly Scottish undergraduate students. The reading speed of undergraduate participants can be expected to be faster than that of participants from a more balanced sample of the population.

Finally, as in Experiment 1, we found no discernible improvement of accuracy when using Eichstaedt's (2001) inaccurate timing filter to remove unreliable measurements (over and above standard outlier removal).

5. CONCLUSION

In this article, we introduced WebExp, a software package for running psychological experiments over the Internet. WebExp is implemented in Java and uses a client–server architecture that gives the experimenter maximal control over stimulus presentation and the collection of responses. The software is XML-based, which means that configuration files, experiment descriptions, and result files can be processed using standard tools. It includes support for a range of stimulus presentation modalities (including images and audio) and various response types. A strong point is WebExp's timing component, which includes both the timing of the presentation of stimuli, and the measurement of response times.

The work reported in the present article focused on the accuracy of the response times that WebExp can collect. In Experiment 1, we presented the results of a study that evaluates WebExp time measurements for known intervals (generated using keyboard repetition). The results indicate that the time measurements are remarkably accurate across two operating systems, two system configurations, and two load conditions.

Experiment 2 then provided a replication of an existing psycholinguistic study (Sturt et al., 1999) that used the self-paced reading methodology to measure reading times. The original study was conducted in a lab using dedicated experimental software and customized hardware for collecting response times. We reimplemented the original study as faithfully as possible using WebExp, using the same sentence materials. This resulted in a full replication of the original results, both in terms of a correlation between the reading times of the two studies, and in the main effects and interactions observed. We also managed to replicate the contrast between means reported in the original study, with one exception. Using a confidence interval approach, we estimated that the replicated study was able to detect significant differences between means of 183 msec. This result represents a considerable improvement over previous findings: For example, Corley and Scheepers (2002) were able to obtain significant response time effects using WebExp, but with a much larger effect size (around 500 msec).

Taken together, our results indicate that Web-based data collection can be used for measuring response times. In our view, this is a very promising line of research, in spite of potential problems faced by time measurement over the Web, including variation in hard and software, network latency, and load on the Java Virtual Machine of the participant. On the other hand, it is important that one not overgeneralize the present findings, since they are based on a single replication study. More research is needed to establish whether reliable time measurements can also be obtained across a range of experimental paradigms using WebExp.

Furthermore, future work should test the limits of Web-based timing by attempting to replicate studies that rely on smaller critical differences. For example, self-paced reading studies that measure response times on a word-by-word basis (rather than on a region-by-region basis as we did in the present article) typically report critical differences of 40–80 msec. It remains to be seen whether response time effects of this size can be measured reliably using Web-based methods.

AUTHOR NOTE

We thank Patrick Sturt for making the original materials and data of Experiment 1 of Sturt et al. (1999) available. The development of WebExp was made possible by financial support from the Principal's e-Learning Fund and the Dean's Fund of the University of Edinburgh, as well as by the Psychology Network of the Higher Education Academy. Further support was provided by Mara Mather through a consulting contract with the Department of Psychology at the University of California at Santa Cruz. Address correspondence to F. Keller, University of Edinburgh, School of Informatics, 10 Crichton Street, Edinburgh EH8 9AB, Scotland (e-mail: frank.keller@ed.ac.uk).

REFERENCES

Birnbaum, M. H. (Ed.) (2000). *Psychological experiments on the Internet*. San Diego: Academic Press.

Cohen, J. D., MacWhinney, B., Flatt, M., & Provost, J. (1993). PsyScope: A new graphic interactive environment for designing psychology experiments. *Behavior Research Methods, Instruments, & Computers*, **25**, 257-271.

Corley, M., & Scheepers, C. (2002). Syntactic priming in English sentence production: Categorical and latency evidence from an Internet-based study. *Psychonomic Bulletin & Review*, **9**, 126-131.

East, H. (2005). *Models and mechanisms of sentence processing: Accounting for distance effects in temporarily ambiguous English sentences*. Unpublished doctoral dissertation, University of Cambridge.

Eichstaedt, J. (2001). An inaccurate-timing filter for reaction-time measurement by JAVA applets implementing Internet-based experiments. *Behavior Research Methods, Instruments, & Computers*, **33**, 179-186.

Featherston, S. (2005). Universals and grammaticality: Wh-constraints in German and English. *Linguistics*, **43**, 667-711.

Forster, K. I., & Forster, J. C. (2003). DMDX: A Windows display program with millisecond accuracy. *Behavior Research Methods, Instruments, & Computers*, **35**, 116-124.

Garnsey, S., Pearlmutter, N. J., Myers, E., & Lotocky, M. A. (1997). The contributions of verb bias and plausibility to the comprehension of temporarily ambiguous sentences. *Journal of Memory & Language*, **37**, 58-93.

Gunasekharan, S. (2007). *Evaluation of Web experiments*. Unpublished master's thesis, University of Edinburgh.

Just, M. A., Carpenter, P. A., & Woolley, J. D. (1982). Paradigms and processes in reading comprehension. *Journal of Experimental Psychology: General*, **111**, 228-238.

Keller, F., & Alexopoulou, T. (2001). Phonology competes with syntax: Experimental evidence for the interaction of word order and accent placement in the realization of information structure. *Cognition*, **79**, 301-372.

Krantz, J. H., Ballard, J., & Scher, J. (1997). Comparing the results of laboratory and World-Wide Web samples on the determinants of female attractiveness. *Behavior Research Methods, Instruments, & Computers*, **29**, 264-269.

Lodge, M. (1981). *Magnitude scaling: Quantitative measurement of opinions*. Beverly Hills, CA: Sage.

MacInnes, W., & Taylor, T. L. (2001). Millisecond timing on PCs and Macs. *Behavior Research Methods, Instruments, & Computers*, **33**, 174-178.

Masson, M. E. J., & Loftus, G. R. (2003). Using confidence intervals for graphically based data interpretation. *Canadian Journal of Experimental Psychology*, **57**, 203-220.

Mayo, N., Corley, M., & Keller, F. (2005). *WebExp2 experimenter's manual*. Retrieved February 24, 2008, from www.webexp.info.

McKinney, C. J., MacCormac, E. R., & Welsh-Bohmer, K. A. (1999). Hardware and software for tachistoscopy: How to make accurate measurements on any PC utilizing the Microsoft Windows operating system. *Behavior Research Methods, Instruments, & Computers*, **31**, 129-136.

Myors, B. (1999). Timing accuracy of PC programs running under DOS and Windows. *Behavior Research Methods, Instruments, & Computers*, **31**, 322-328.

Naumann, A., Brunstein, A., & Krems, J. F. (2007). DEWEX: A system for designing and conducting Web-based experiments. *Behavior Research Methods*, **39**, 248-258.

Rademacher, J. D. M., & Lippke, S. (2007). Dynamic online surveys and experiments with the free open-source software dynQuest. *Behavior Research Methods*, **39**, 415-426.

Reimers, S., & Stewart, N. (2007). Adobe Flash as a medium for online experimentation: A test of reaction time measurement capabilities. *Behavior Research Methods*, **39**, 365-370.

Reips, U.-D. (2002). Standards for Internet-based experimenting. *Experimental Psychology*, **49**, 243-256.

Reips, U.-D., & Neuhaus, C. (2002). WEXTOR: A Web-based tool for generating and visualizing experimental designs and procedures. *Behavior Research Methods, Instruments, & Computers*, **34**, 234-240.

Schmidt, W. C. (2001). Presentation accuracy of Web animation methods. *Behavior Research Methods, Instruments, & Computers*, **33**, 187-200.

Shaughnessy, J. J., & Zechmeister, E. B. (1994). *Research methods in psychology* (3rd ed.). New York: McGraw-Hill.

Sturt, P., Pickering, M. J., & Crocker, M. W. (1999). Structural change and reanalysis difficulty in language comprehension. *Journal of Memory & Language*, **40**, 136-150.

Thornton, R., MacDonald, M. C., & Arnold, J. E. (2000). The concomitant effects of phrase length and informational content in sentence comprehension. *Journal of Psycholinguistic Research*, **29**, 195-203.

NOTES

1. For more details regarding the software, including documentation and demonstration experiments, the reader is referred to www.webexp.info.

2. Eventually the Java real-time specification may allow the system to provide real-time guarantees, but in the meantime, users must accept the limitations inherent in running the Java Virtual Machine on a variety of systems.

3. We decided not to include the Mac OS platform in this experiment because of the following technical difficulty: There is no standard Sun Java implementation available for Mac OS, since Sun only ports its Java to Windows and Linux. Apple's port to Mac OS is unofficial and is not fully compatible with the Sun version of Java; we can therefore not guarantee that WebExp works under Mac OS as expected (and in certain cases it does not, in our experience). Furthermore, Experiment 2 shows that there are only a small number of Mac OS users, at least in our sample (3 out of 48 participants, or 6.25% in the unreduced sample).

4. NP/S ambiguities involve verbs that can take either a noun phrase (NP) or a sentence (S) as a complement. NP/Z ambiguities involve verbs that can take either a noun phrase or no complement.

5. www.language-experiments.org.

6. The experiment was left on the portal for such a long time to benefit from accidental visitors to the site. It is typically possible to complete a Web-based study much faster if targeted advertising is used (possibly along with financial incentives).

7. Recall that in Experiment 1, it was observed that the control thread of the inaccurate timing filter freezes under high load on the Linux platform. We screened the data of the Linux users in the present experiment and found that 6 of the participants showed MI values that are consistent with the control thread freezing. This suggests that these participants had additional background applications running. However, Experiment 1 also indicated that this problem only affects the control thread, not the main thread that measures the response times. We therefore decided against excluding these participants; the MI values of these participants are invalid, but Experiment 1 showed that MI values are only of limited use (and the present experiment confirms this).

(Manuscript received February 29, 2008;
revision accepted for publication June 24, 2008.)