Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Visual Languages and Computing
17 (2006) 187–202
Journal of Visual Languages & Computing
www.elsevier.com/locate/jvlc

Impact of high-intensity negotiated-style interruptions on end-user debugging

T.J. Robertson, Joseph Lawrance*, Margaret Burnett

Oregon State University, Corvallis, OR, USA

Received 10 June 2004; received in revised form 17 August 2005; accepted 16 September 2005

Abstract

Extending our previous work [T. Robertson, S. Prabhakararao, M. Burnett, C. Cook, J. Ruthruff, L. Beckwith, A. Phalgune, Impact of interruption style on end-user debugging, ACM Conference on Human Factors in Computing Systems (2004)], we delve deeper into the question of which interruption style best supports end-user debugging. Previously, we found no advantages of immediate-style interruptions (which force the user to divert attention to the interruption at hand) over negotiated-style interruptions (which notify users without actually preventing them from working) in supporting end-user debugging. In this study, we altered our negotiated-style interruptions [A. Wilson, M. Burnett, L. Beckwith, O. Granatir, L. Casburn, C. Cook, M. Durham, G. Rothermel, Harnessing curiosity to increase correctness in end-user programming, Proceedings of the CHI 2003 (2003), 305–312] (which were shown to help end-user debuggers learn about and use debugging features of our programming language) such that they were more intense (larger, blinking, and/or accompanied by text).

Keywords: Interruptions; End-user software engineering; Debugging; Surprise-Explain-Reward; Spreadsheets

1. Introduction

Unlike most research on end-user programming, which focuses on the creation of new programs, our research team has been focusing on supporting aspects of software

*Corresponding author. Tel.: +1 541 760 9374.
E-mail addresses: tjjuggle@yahoo.com (T.J. Robertson), lawrance@eecs.oregonstate.edu (J. Lawrance), burnett@eecs.oregonstate.edu (M. Burnett).

1045-926X/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2005.09.002

engineering for end-user programmers. Recently, our group and other researchers have begun to focus on helping end-user programmers debug their programs. Most systems that aim to help end-user debug programs have to notify the users of information relevant to debugging (e.g., [1–4]). The mechanism for communicating such information can have a significant impact on the systems' effectiveness. We have been investigating the appropriateness of various methods of communicating this information to users.

In [5], we presented our findings from a study comparing the effects on end users of two styles of interruptions. We studied the effects on the end users in terms of their ability to learn the debugging features of the software, their ability to fix bugs, and their ability to accurately rate how successful they were at fixing bugs. The two debugging styles compared were *negotiated-style interruptions* and *immediate-style interruptions*, according to McFarlane's [6] classification of interruptions. Negotiated-style interruptions are interruptions that inform users of a pending message, but do not force them to acknowledge the message immediately. For example, in our study, a red circle appearing around possibly incorrect values in spreadsheet cells serves as a negotiated-style interruption, thereby letting the user know that they should check that cell for errors. Immediate-style interruptions are interruptions that must be acknowledged by the user. In our study, we used modal pop-up "ok" boxes that displayed messages.

Our study in [5] concluded that negotiated-style interruptions were superior to immediate-style interruptions in all aspects studied (participants' learning of debugging features, fixing bugs, and predicting their performance). Additionally, analysis of the actions in which the different participants engaged suggests that the immediate-style interruptions may have encouraged over-reliance on shallow debugging strategies that had low short-term memory requirements.

This paper extends [5] in that we delve deeper into the question of how to interrupt end-user programmers in order to help them better debug their programs. We have already shown that negotiated-style interruptions were superior to immediate-style interruptions in every metric we used. Now we wish to explore how the intensity of the negotiated-style interruptions affects end-user programmers. In this paper, we present the results of a study in which we compare the previous participants with a new group of participants. The new participants had much more intense negotiated-style interruptions, i.e., the interruptions had more intense characteristics such as being larger, blinking, and being accompanied by text. We will refer to these more intense forms of negotiated-style interruptions as *high-intensity* negotiated-style interruptions. In contrast, our original negotiated-style interruptions (such as the red circle around cell values) will be referred to as *low-intensity* negotiated-style interruptions.

Because negotiated-style interruptions can range from barely noticeable to extremely hard to ignore, we felt that it was not sufficient to simply endorse negotiated-style interruptions (as in [5]). This study will, therefore, compare the higher intensity negotiated-style interruptions with both the lower intensity negotiated-style interruptions and the immediate-style interruptions. In doing so, we hope to paint a more complete picture of the possible effects on users of negotiated-style interruptions.

For this study, our research question is as follows.

> What are the effects on end-user programmers (in terms of learning debugging features, debugging productivity, and ability to predict debugging performance) of high-intensity negotiated-style interruptions as compared to both low-intensity negotiated-style interruptions and immediate-style interruptions?

2. Related work

Researchers have recently begun to study aspects of end-user programming beyond just the programming stage; examples include [1,3,7,8].

McFarlane [6] created a classification for interruptions in which he described four styles of interruptions, each with associated strengths and weaknesses. *Negotiated interruptions* alert users of pending notifications, but do not force them to acknowledge the notification immediately. *Immediate interruptions* prevent the user from continuing with their work until the interruption is dealt with. *Mediated interruptions* present information when the system decides it is an appropriate time to interrupt the user. *Scheduled interruptions* present interruptions at fixed time intervals. In [5], our group found negotiated-style interruptions to be better for assisting end-user debugging than immediate-style interruptions in all metrics we measured.

It has been shown, however, that the properties of negotiated-style interruptions can have significant effects on users. In a study of peripheral scrolling text displays [9], it was found that displays that scrolled continuously were more distracting than displays that only scrolled in new information when it was available. In addition, it was found that the information remembered by users was the same in either case. Still, in a study of animated displays, McCrickard et al. [10] found that animated displays can be used with minimal negative impact. McCrickard et al. attributed this difference from [9] to the facts that their primary task was less cognitively demanding, and their animations were slower than in [9].

Berlyne [11] explains that the *intensity variables* (traits such as size and color) will affect the amount of *arousal* (a psychological term which essentially refers to the amount of mental stimulation a person is experiencing). Thus, our high-intensity negotiated-style interruptions are likely to evoke high arousal in our participants. Furthermore, arousal research [12] shows that people have an optimal level of arousal, and they will adjust their activities to reach this optimal level. Exceeding this level can result in the user becoming desperate to get rid of the arousal evoking stimuli. Thus, though high-intensity interruptions are negotiated-style interruptions, it may be so hard to ignore them that they pull the user from their work just as an immediate-style interruption would. If this is the case, then we should expect their performance to suffer, just as the performance of the participants with immediate-style interruptions suffered in [5].

It has been shown that waiting until the user has completed certain tasks is less disruptive than interrupting them in the middle of the tasks [13]. Indeed, it was found that in a data entry task, an immediate-style interruption on screen was more distracting than a phone call or walk-in visitor (both of which can be considered negotiated-style interruptions, because the user can continue working for a brief period of time until they get to an appropriate stopping point) [14]. A potential problem with high-intensity negotiated-style interruptions is that they may disrupt the user (cause them to reach overly high arousal levels) at inopportune times.

On the other hand, new work [15] shows that software that receives data from a relatively simple set of sensors can determine the best times to interrupt users. So intelligent timing of such high-intensity interruptions may mitigate such a problem.

It is interesting to note that even low-intensity negotiated-style interruptions work on the principle of raising arousal in order to evoke action from the user. This study is intended to provide some data showing what the effects are at high-intensity levels.

3. Experiment

We conducted a controlled laboratory experiment with three groups of end-user participants. For brevity, throughout this paper we will refer to our different groups of participants as *high-intensity*, *low-intensity*, and *immediate-style* participants.

3.1. Design, procedures, and tasks

We replicated the experiment done in [5,16], except for the way in which interruptions were presented. In this study, we modified the environment from [16] so that the negotiated-style interruptions were more intense.

We reused data from [16] as our low-intensity group (16 participants) and data from [5] as our immediate-style group (22 participants), and added in a new group of 15 participants for our high-intensity group. As in earlier studies, participants responded to an invitation given to business majors who were at least sophomores and who had prior spreadsheet experience. This study, however, occurred during Fall term, rather than during summer term as the other two studies did.

Reusing prior groups and adding a new group was a necessary pragmatic design decision but it is not ideal, as it opens the door to validity questions. To guard against differences in our participants that might bias our results, we administered and analyzed a background questionnaire. No difference was found in the participants' confidence in their ability to create a spreadsheet. There was also no difference found in the participants' GPAs. Additionally, these participants were all business majors, as were most participants in previous studies. However, we uncovered two differences. First, the high-intensity participants had less professional spreadsheet experience than the low-intensity participants (Mann–Whitney $p = 0.0418$) and less total spreadsheet experience than the immediate-style participants (Mann–Whitney $p = 0.0370$). Second, the high-intensity participants tended to be of junior standing, whereas the other participants tended to have senior standing (Mann–Whitney $p = 0.0023$ for high-intensity vs. low-intensity participants and Mann–Whitney $p = 0.0295$ for high-intensity vs. immediate-style participants). To statistically account for potential impacts of these differences, we checked our results by analyzing main and interaction effects of background data using ANOVA. In particular, unless stated otherwise in Section 4, ANOVA showed no evidence to suggest that results of interest differed based on background, nor did ANOVA show evidence to suggest that the results of interest varied based upon the interaction of background data and interruption style.

We gave the participants a 25-min tutorial introducing them to the spreadsheet language Forms/3. After the tutorial, they were asked to debug two spreadsheets, Grades and Weekly Pay, with time limits of 35 and 22 min, respectively; see [16,17] for details of these spreadsheets. The time limits were to ensure that all participants worked on both spreadsheets, to avoid possible peer influences of some participants leaving early, and to ensure consistent treatment of all participants. Roughly half of the participants did Grades first and the other half did Weekly Pay first. This was done so as to evenly distribute learning effects over the problems.

The participants were given problem descriptions that described what the spreadsheet was to accomplish, and what individual cells were for. The problem description instructed participants to "test the spreadsheet thoroughly to ensure that it does not contain errors

and works according to the spreadsheet description. Also, if you encounter any errors in the spreadsheet, fix them''.

All user actions were recorded in electronic transcripts for later review. After each problem, participants were given a questionnaire in which they rated how well they thought they had debugged the spreadsheet. At the end of the study, every participant answered questions that tested their understanding of the debugging feature that pertained to our high-intensity interruptions.

3.2. Assertions and Surprise-Explain-Reward

This study was conducted using the research spreadsheet language Forms/3 [18]. Among the features of Forms/3 are assertions on spreadsheet cells, which past empirical work has shown that end users can use effectively [17,19]. Although spreadsheet users can debug and find errors through means other than assertions, this study focuses on assertions in that we intensified only those interruptions that related to assertions. Therefore, the results and discussion focus on the interruptions' impacts on participants' responses to assertions.

Assertions in Forms/3 are represented as allowable ranges for a cells' value. When a cell's value falls outside the allowable range, the assertion is violated; such a violation is called a *value violation*. Forms/3 draws red circles (shown as gray circles in this paper) around value violations. When a user enters an assertion on a cell, this *user-entered assertion* is propagated through the dataflow chain of the spreadsheet, generating new assertions called *system-generated assertions*, and these too are propagated.

Assertion propagation is done as follows. Just as traditional spreadsheets compute the value for a cell based on its formula, Forms/3 computes an assertion for a cell based on its formula. Forms/3 can compute an assertion for a cell if all the cells referenced in its formula also have assertions. When a system-generated assertion for a cell conflicts with the user-entered assertion for that cell, Forms/3 draws a red circle (shown as a gray circle in this paper) around the two conflicting assertions; such a conflict is termed an *assertion conflict*.

For example, in Fig. 1, when the user entered an assertion for input_temp, Forms/3 computed a system-generated assertion for output_temp. When the user subsequently entered an assertion "0 to 100" for output_temp, Forms/3 circled the assertion conflict since the user-entered assertion disagrees with the cell's system-generated assertion. Forms/3 circled the values "200" and "33.3333" because they violate their cells' assertions.

Besides assertions, participants had other debugging devices available. Forms/3 includes a "testedness" indicator that measures testing progress for an entire spreadsheet as participants check off cell values as correct. If a participant decided a cell's value was correct, the participant could check it off using the checkbox in the upper-right corner of the cell (see the checkbox in Fig. 1's output_temp cell). As participants checked off previously untested values in a spreadsheet, the border colors of each checked-off cell changed from red to a bluer hue to indicate increased testedness; also, the "testedness" indicator showed increased progress toward 100% testedness. If participants wanted help conjuring up more test inputs, participants could push a *Help-Me-Test* button to automatically generate values that had not yet been tested [19].

Surprise-Explain-Reward is a strategy we use to introduce users to features that may benefit them, such as assertions. A vehicle for this strategy is the *Help-Me-Test* button designed to help users test their spreadsheets. When a user clicks the Help-Me-Test button,

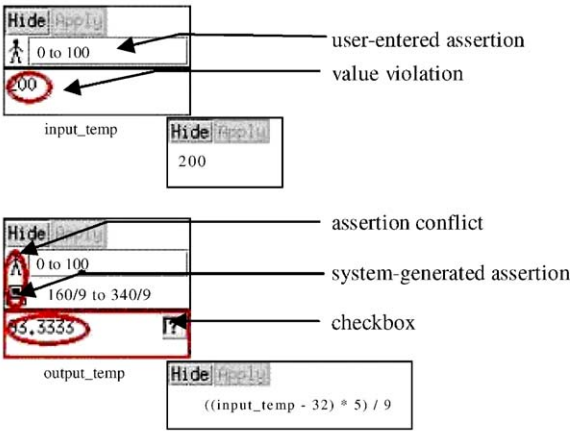

Fig. 1. Assertion examples.

Forms/3 not only generates values for input cells, but also creates a (usually incorrect) "guessed" assertion to place on these cells. These guessed assertions, termed *HMT assertions* (because they are generated by Help-Me-Test), are intended to *surprise* the user into becoming curious about assertions. They can satisfy their curiosity using tool-tips, which *explain* the benefits and syntax of assertions. If the user accepts an HMT assertion (either as guessed or after editing it), Forms/3 *rewards* the user with the benefits of assertions by circling value violations or assertion conflicts that may occur, which helps ensure that cell values, user-entered assertions, HMT assertions and system-generated assertions are consistent. All surprises are communicated via interruptions.

3.3. High-intensity negotiated-style interruptions

In [16], the mechanisms for communicating surprises to the user came in the form of low-intensity interruptions. Surprise-Explain-Reward with these low-intensity interruptions was found to be beneficial to end-user programmers [16]. In [5], the low-intensity interruptions were accompanied with immediate-style interruptions in the form of pop-up "ok" dialog boxes that contained the same text as the tool-tip.

In this study, we substituted high-intensity interruptions for the low-intensity interruptions and removed the experimental immediate-style interruptions. Recall that by intensity, we refer to properties of the interruption. Fig. 2 shows each property that was manipulated.

4. Results

4.1. Learning results

In this section, we will examine the effects of high-intensity interruptions on users' ability to learn the debugging features of the software. We examine this because much of the research in end-user programming focuses on guiding users to find bugs [3,4]. This necessarily involves teaching them about the debugging features of the software.

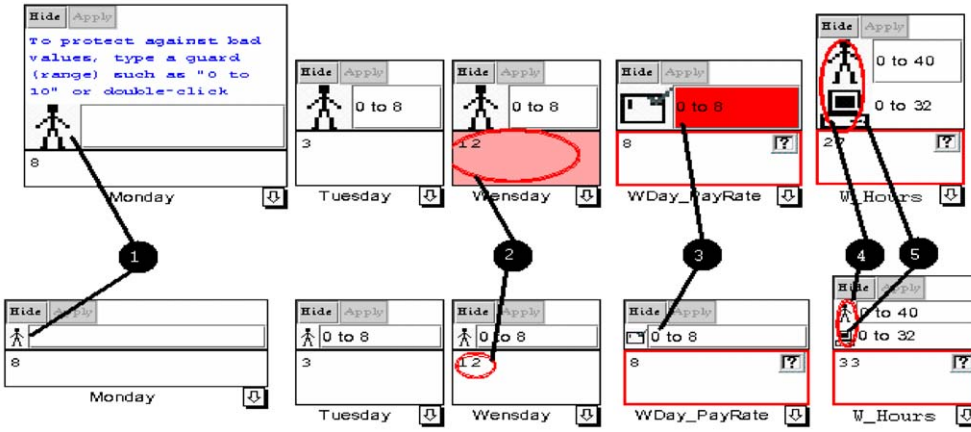

Fig. 2. What the high-intensity interruptions look like (top row) vs. low-intensity interruptions (bottom row).

	Event	Low intensity	High intensity
1	When the user shows interest in assertions (by clicking the button to make an assertion visible, or deleting an HMT assertion)	Stick figure next to guard	Large stick figure and explanation
2	When there are value violations	Red circle around value	Blinking red circle and cell background
3	When Help-Me-Test generates assertion(s)	Help icon with assertion range	Large icon with blinking background on assertion range
4	When there is an assertion conflict	Red circle around assertion icons	Blinking red circle around enlarged icons
5	When system-generated assertion(s) are created	Computer icon with assertion range	Larger icon with temporarily blinking background on assertion range

We analyzed the participants' post-session questionnaire scores for questions that test participants' understanding of assertions. In [5], it was found that immediate-style participants answered fewer questions correctly, and furthermore, the questions that posed a problem for them were ones that tested their understanding of advanced assertion features (specifically propagation of assertions through dataflow chains). The deficit in understanding advanced features is important, because those features are the ones that will aid users by automatically identifying software errors.

To test for differences in comprehension between the high-intensity vs. low-intensity and immediate-style participants, we formulated the following null hypotheses.

Hypothesis H1. There will be no difference in the high-intensity and low-intensity participants' comprehension of assertions.

Hypothesis H2. There will be no difference in the high-intensity and immediate-style participants' comprehension of assertions.

As discussed in Section 2, the high-intensity interruptions may affect users similar to the immediate-style participants. If so, they should result in a lower comprehension of assertion features (just as they did the immediate-style participants).

Indeed, the high-intensity participants did perform very similar to the immediate-style participants. The high-intensity participants correctly answered 43% of assertion-related questions, compared to 67% for the low-intensity participants, and 46% for the immediate-style participants. Thus, we cannot reject Hypothesis H2 (Mann–Whitney $p = 0.6933$). Just as with the immediate-style participants, the high-intensity participants were found to have significantly lower comprehension than the low-intensity participants (Mann–Whitney $p = 0.0080$). Therefore, we reject Hypothesis H1.

Observe from Fig. 3 and Table 1 that the high-intensity participants' performance in answering assertion-related questions mirrored that of the immediate-style participants. As in [5], they had trouble understanding the advanced features of assertions, including propagation of assertions and HMT assertions. The lack of knowledge about HMT assertions is interesting, because in this study, the high-intensity participants had HMT assertions that blinked (presumably increasing their awareness of them). As we will address in Section 5, the blinking may have caused them to care less about learning the reason behind assertions existence, and more about getting them to go away.

4.1.1. First assertion time

Surprise-Explain-Reward's goal of teaching users how to use the debugging features of software hinges on its ability to draw the users' attention to those features. To evaluate the

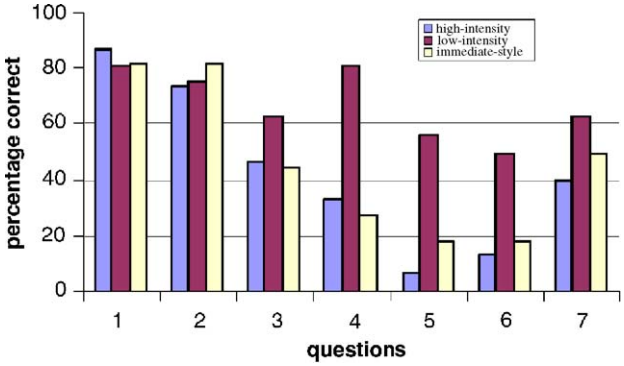

Fig. 3. Participants' scores on assertion-related comprehension questions.

Table 1
Categorizations of the comprehension questions

Question number	Question content
q1, q2	Ability to recognize user-entered assertions and values being outside these ranges (shown as red circles in the environment).
q5, q6	Comprehension of the computer-guessed HMT assertions.
q3, q4, q7	Comprehension of assertions' propagation through the dataflow chains formed by formulas, including conflicts between user-entered and system-generated assertions that could arise.

Table 2
Average time until users entered their first assertion on both tasks, as well as after they started Help-Me-Test for the first time (on the first task)

Average time to first assertion	1st task	2nd task	After Help-Me-Test (1st task)
High intensity	7:33	6:25	4:42
Low intensity	13:26	5:54	3:44
Immediate	7:10	5:01	4:04

Times after Help-Me-Test were calculated by considering only participants who used Help-Me-Test and then assertions.

ability of high-intensity interruptions to draw users to use assertions, we examined the average time that it took for participants to enter assertions under three different conditions. The three conditions were: first task, second task, and after they started Help-Me-Test for the first time (on the first task).

Hypothesis H3. The time when participants are first enticed to enter assertions will not differ among interruption styles.

Times for the first and second tasks were calculated by taking the difference between the time participants entered their first assertion and the time they made their first action. As Table 2 shows, the immediate-style participants had the fastest time to enter an assertion on the first task. On the first task, the immediate-style participants were significantly faster at creating their first assertion than the low-intensity participants (Mann–Whitney $p = 0.0288$).[1] The high-intensity participants appeared to perform just slightly slower than the immediate-style participants, but they were not quite significantly faster than the low-intensity participants at entering their first assertion on the first task (Mann–Whitney $p = 0.0580$).

On the first task, we also checked for differences between the first time participants used Help-Me-Test and the first time they entered an assertion. This is because Help-Me-Test is Surprise-Explain-Reward's gateway to the interruptions that we manipulated in this study. The delays between starting Help-Me-Test and entering their first assertion were not significantly different for any of the participants (all Mann–Whitney $p > 0.60$).

On the second task, the time differences were not very pronounced, and there were no significant differences in any participants' times.

4.1.2. Conjuring up accurate assertions

As part of assessing how well participants learned how to use assertions, we examined how accurate they were at creating correct assertions. This is an important metric, as the goal of Surprise-Explain-Reward is to teach users how to properly use debugging features. Additionally, creating correct assertions is crucial because when they propagate (regardless of whether users understand how they propagate) through the dataflow chain, the red circles and the propagated assertions will only be correct if the original was correct.

[1]There is another way of scoring the immediate-style participants' first assertion time, and that is because they had to hit an "experiment start" button, which could be considered to be the first action they took. If that scoring method is used, 8:34 s is their start time (as was reported in [5]). The alternative scoring method for the immediate-style subjects, however, is inconsistent with the way first assertion times were graded for the rest of the participants.

Hypothesis H4. There will be no difference between the percentage of assertions created correctly by high-intensity participants vs. the low-intensity and immediate-style participants.

As can be seen in Table 3, the high-intensity participants appeared to be slightly less accurate than the low-intensity participants and immediate-style participants (which have approximately equivalent accuracy at creating assertions). However, none of the participants were significantly different from each other in terms of assertion accuracy. Thus, we cannot reject Hypothesis H4.

4.2. Debugging productivity

In [5], we found that the participants with low-intensity negotiated-style interruptions were significantly more effective at fixing bugs than the immediate-style participants. The significant difference was found to be on the second task (after the learning phase of the first task in which there was no significant difference).

Again, we propose that the high-intensity interruptions will affect participants similar to the way that immediate-style interruptions affected participants. Thus, we expect high-intensity participants to perform roughly the same as immediate-style participants of [5]. To investigate, we use the following null hypotheses.

Hypothesis H5. There will be no difference in the high-intensity and low-intensity participants' debugging productivity on the first task.

Hypothesis H6. There will be no difference in the high-intensity and immediate-style participants' debugging productivity on the first task.

Hypothesis H7. There will be no difference in the high-intensity and low-intensity participants' debugging productivity on the second task.

Hypothesis H8. There will be no difference in the high-intensity and immediate-style participants' debugging productivity on the second task.

Table 4 shows the debugging productivity of all three groups of participants and the Mann–Whitney p-values comparing the high-intensity participants with the other two groups of participants.

On the second task, the high-intensity participants performed significantly worse than the low-intensity participants (Mann–Whitney $p = 0.007$) at fixing bugs. Two-way ANOVA validated this result with a main effect for interruption-style, $F_{2,47} = 5.0350$, $p = 0.01$. There was also an interaction effect, $F_{2,47} = 7.1965$, $p = 0.002$, indicating that the effect of interruption style also depends on total spreadsheet experience, but this

Table 3
Percentage of correct assertions that participants created and the percentage of cells with correct assertions

	Percent of correct assertions
High intensity	0.877
Low intensity	0.918
Immediate	0.912

Table 4
Productivity at fixing bugs for both tasks

Interruption style	Total bugs fixed	Bugs per minute—1st task	Bugs per minute—2nd task
High intensity ($n = 15$)	10.73	0.193	0.198
Low intensity ($n = 16$)	13	0.202 ($p = 0.827$)	**0.263 ($p = 0.007$)**
Immediate ($n = 22$)	11.18	0.205 ($p = 0.543$)	0.206 ($p = 0.3989$)[a]

p-values given are for low intensity and immediate style compared to high intensity with Mann–Whitney.

[a]There was a mistake in [5], in that the 2nd task debugging performance for immediate-style participants was reported as 0.163. This was incorrect; the actual number is 0.263, but it does not change the statistical significance reported in that paper.

Table 5
Average number of each type of activity engaged in by the participants

Interruption style	Edit formula	Edit assertion	Use Help-Me-Test	Check off value	Avg. total
1st Task					
High intensity	12.07	11.27	9.20	20.27	52.81
Low intensity	17.81	11.19	18.00	30.31	77.31
Immediate style	12.73	12.68	17.50	23.86	66.77
High intensity vs. low intensity (Mann–Whitney)	$p = 0.0546$		$p = 0.2755$		
2nd Task					
High intensity	12.73	**16.80**	**17.40**	26.93	73.86
Low intensity	16.13	10.75	10.94	23.69	61.51
Immediate style	10.00	10.91	16.45	27.82	65.18
High intensity vs. low intensity (Mann–Whitney)	$p = 0.4048$	**$p = 0.0040$**	**$p = 0.0173$**		

Significant differences ($p < 0.05$) between the low-intensity and high-intensity participants are emphasized in bold (there were no interaction effects impacting any of these results).

interaction impacted only the high-intensity vs. immediate participants. Thus, we can safely reject H7.

4.2.1. How participants spent their time

In [5], we attempted to understand immediate-style participants' lower scores at fixing bugs by examining the types of actions they chose to do. We found significant differences in the actions they engaged in, and those differences suggested fundamental differences in their debugging strategies.

As can be seen from Table 5, there were significant differences in activities engaged in by high-intensity participants compared to low-intensity participants. On the second task, the high-intensity participants edited significantly more assertions and used Help-Me-Test significantly more than the low-intensity participants. This differs from the immediate-style

participants who did approximately the same number of assertion edits, but did significantly less formula edits than the low-intensity participants.

The more frequent use of debugging features during the second task (after the majority of learning effects have occurred) tells us that the high-intensity interruptions encouraged participants to make use of the debugging features. This is interesting because we have shown that they were less knowledgeable of the features and less effective at debugging compared to the low-intensity participants. We will address the more frequent use of assertions in Section 5. As for the more frequent use of Help-Me-Test, it is possible that an effect similar to what we believe occurred to the immediate-style participants of [5] occurred. Namely, the high arousal state that the high-intensity interruptions induced in the participants encouraged them to use tools such as Help-Me-Test which do not require as much mental effort, or short-term memory load. Indeed, there is psychological research showing that under high arousal, people's short-term memory can suffer [12].

4.3. Debugging self-assessment

With end-user programs being used for important calculations, it is important that these programs not go into use prematurely. In practice, end-user programmers are notorious for overestimating their ability to create correct software [20]. For this reason, we considered the participants' abilities to evaluate their accuracy at debugging spreadsheets.

After participants completed each problem, we administered a questionnaire that asked them to rate how confident they were that they had corrected all the bugs in the spreadsheet. Their choices ranged from 1 (''not confident'') to 5 (''very confident''). In [5], it was seen that the low-intensity participants' self-ratings were significant predictors of actual performance at fixing bugs whereas the high-intensity participants' scores were not. We now investigate how well the high-intensity participants were able to judge their debugging performance.

Hypothesis H9. The high-intensity participants' self-ratings will not be significantly correlated with their ability to fix bugs on a per problem basis.

Table 6 shows the regression analysis of the participants' ability to rate how well they corrected bugs for both problems. As the table shows the high-intensity participants' self-ratings were not significant predictors of their performance at fixing bugs (unlike the low-intensity participants). Thus, we cannot reject Hypothesis H9.

5. Discussion—agitated users

Theories on arousal suggest that if a users' arousal exceeds an optimum amount, the user will resort to avoidance strategies, in which they attempt to make the stimuli go away [12]. This is different from a more constructive state of curiosity, which might compel them to explore the surprise and learn how it works [21]. Rather, in such an overly aroused state, a user may be satisfied with any action that causes the arousing stimuli to go away. It seems likely that some of our high-intensity participants were in just such a state.

Table 6
Linear regression *p*-values for number of bugs fixed vs. belief in how successful they were at fixing all bugs

	Linear regression *p*-value
Grades	
High intensity	0.966392
Low intensity	**0.0410**
Immediate style	0.1194
Weekly pay	
High intensity	0.3383
Low intensity	**0.0190**
Immediate style	0.1711

As noted earlier, the high-intensity participants edited significantly more assertions in the second task than other participants. Indeed, the assertion edits on the second task reveals some interesting information about the high-intensity participants.

On the second task, all of the high-intensity participants except for two edited at least 13 assertions with a maximum of 25. By contrast, the low-intensity and immediate-style participants had an average of slightly less than 11 assertion edits each. What's remarkable is that the two high-intensity participants who did not perform double-digit numbers of assertion edits performed zero assertion edits (though they did use Help-Me-Test about as frequently as the other participants, so they were exposed to blinking HMT assertions). That is, the high-intensity participants appeared to be polarized in that they performed either many assertion edits, or zero assertion edits.

It is trivial to see how participants performing large numbers of assertion edits are compatible with the theory that they were experiencing an excess of arousal (and thus avoidance behavior). They found that editing assertions was a way to make the blinking HMT assertions go away. One of our high-intensity participants epitomized this avoidance behavior. This participant *deleted* 21 (out of 22) HMT assertions to stop the blinking. This means that 21 times the participant highlighted a blinking HMT assertion, deleted the text (e.g., "3 to 14"), and hit the "Apply" button. In doing so, this participant not only stopped the blinking, but also threw away potentially valuable debugging information. This participant clearly had little concept of the purpose of the assertions, but was very vigilant about deleting them. Indeed, this participant only got the simplest two questions (out of seven) correct on the post-session questionnaire. Thus, the participant seemed eager to take any action (no matter how counterproductive) that made the blinking go away. As this illustrates, high-intensity interruptions potentially have the serious drawback of causing users to respond with counterproductive behaviors. Other participants also edited assertions to stop the blinking.

But what about the participants with zero assertion edits? It was noticed during pilot tests and during the administration of the experiment that some participants found a way to hide all of the blinking HMT assertions by pressing a "Close HMT" button that appears whenever Help-Me-Test is invoked. It is possible that some participants learned to press Help-Me-Test to get new test values and then press the "Close HMT" button (which would leave the new test values in place) to make all the blinking HMT assertions go away. Unfortunately, pressing this button was not recorded in our electronic transcripts, so we have no way of knowing whether the two participants with zero assertions used this strategy.

6. Threats to validity

Here we address potential threats to the validity of our results.

Threats to internal validity are unintended factors that may have influenced our results. To provide a uniform treatment for participants, we used a tutorial script (though for the immediate-style participants, the script was read by a non-native English speaker) and written problem descriptions which were the same for all participants. We did have to rearrange the spreadsheet layouts slightly for the high-intensity participants so that the high-intensity interruption mechanisms (such as the larger icons) would fit onto the screen without overlapping with other spreadsheet cells. While rearranging them, however, we made every attempt to hold the relative locations of cells constant. On the Grades spreadsheet, unfortunately, high-intensity participants had to scroll down to see all the cells whereas the other participants did not. To ensure that our participants were uniform, we sent out an email to business students asking for a specific set of characteristics that were the same as for the earlier participants. Additionally, we asked the participants for background information so that we could analyze the data for differences. A possible source of bias was that the high-intensity study was done during fall term, whereas the other studies were done during the summer. The difference between fall term and summer school students may not be negligible (e.g., the summer school students may have been older than their fall term counterparts). Indeed, as mentioned earlier, we did find significant differences in seniority and participants' prior experience creating spreadsheets.

Construct validity refers to the question of whether the results are based on appropriate information. We used electronic transcripts to record user actions, so that there would be no question of what participants did (though as mentioned in the discussion, the transcripts did not record when participants pressed the "Close HMT" button). For most of our statistical data, we wrote tools to automatically harvest data from the transcripts for use in statistical comparisons. Additionally, we tended to use multiple choice questions rather than free response questions in order to avoid ambiguity in participants' responses. Sometimes, however, participants would write in a free response answer because they did not realize that the questions were multiple choice. This may have lead to some bias, because we had to choose the closest multiple choice answer to what they wrote.

External validity relates to how well the results can be generalized. About 50% of the participants in each group had at least some professional spreadsheet experience, and we discovered that the significant differences in our data were primarily contributed by these participants. This supports external validity because it implies that our results should be applicable to spreadsheet developers in the business world. However, because our study only lasted about 1.5 h, the results may not hold for long-term usage. Indeed, it has been found [22] that when people train on a system with immediate-style interruptions, they are better equipped to handle working with those immediate-style interruptions. Thus, it is possible that the effects of high-intensity/immediate-style interruptions may become less negative with time.

7. Conclusion

To further investigate the question of how interruption styles affect end-user programmers' abilities to effectively learn and use the debugging features of a programming language, we have extended [5] to examine user performance when

high-intensity negotiated-style interruptions are used. As expected, the high-intensity interruptions seemed to affect the users the same way as did immediate-style interruptions. However, we caution that these results should only be considered in light of the threats to validity discussed above.

Specifically, when compared to users with low-intensity interruptions:

- High-intensity interruptions impaired the users' ability to learn the debugging features of the language.
- Participants with high-intensity interruptions were less effective at debugging.
- Participants with high-intensity interruptions were not effective at judging their ability to debug.

Additionally, psychological theories, as well as user actions indicate that high-intensity interruptions may push the user into an uncomfortable state of mental arousal. This highly aroused mental state may have driven users to engage in any activity that would make the interruptions go away regardless of whether those actions were productive in terms of learning the debugging devices or improving the software.

What do these results tell the designers of end-user programming environments? Not only is it important to use negotiated-style interruptions when attempting to assist end-user programmers in learning and using debugging features, but the nature of the negotiated-style interruptions must also be considered. In particular, negotiated-style interruptions that are very intense should be avoided, as they can have similar effects on end users as immediate-style interruptions.

Acknowledgment

Dr. Douglas Derryberry in Oregon State University's Psychology Department provided help in finding relevant psychological research.

References

[1] A.J. Ko, B.A. Myers, Development and evaluation of a model of programming errors, Proceedings of the IEEE Symposium on Human-Centric Computing Languages and Environments (2003) 7–14.
[2] S. Mathan, K. Koedinger, Recasting the feedback debate: benefits of tutoring error detecting and correction skill, International Conference on Artificial Intelligent Education (2003).
[3] R. Miller, B. Myers, Outlier finding: focusing user attention on possible errors, in: Proceedings of the User Interface Software and Technology, ACM Press, New York, 2001, pp. 81–90.
[4] J. Ruthruff, E. Creswick, M. Burnett, C. Cook, S. Prabhakararao, M. Fisher II, M. Main, End-user software visualizations for fault localization, Proceedings of the ACM Symposium on Software Visualization (2003) 123–132.
[5] T. Robertson, S. Prabhakararao, M. Burnett, C. Cook, J. Ruthruff, L. Beckwith, A. Phalgune, Impact of interruption style on end-user debugging, ACM Conference on Human Factors in Computing Systems (2004).
[6] D.C. McFarlane, Comparison of four primary methods for coordinating the interruption of people in human–computer interaction, Human–Computer Interaction 17 (1) (2002) 63–139.
[7] A.J. Ko, B.A. Myers. Designing the whyline: a debugging interface for asking questions about program behaviour, Proceedings of the CHI 2004 (2004) 151–158.
[8] E. Wagner, H. Lieberman. Supporting user hypotheses in problem diagnosis on the web and elsewhere, in: Proceedings of the International Conference on Intelligent User Interfaces, Funchal, Madeira Island, January 2004, pp. 30–37.

[9] P.P. Maglio, C.S. Campbell, Tradeoffs in displaying peripheral information, in: Proceedings of the CHI 2000 Conference on Human Factors in Computing Systems, ACM Press, New York, 2000, pp. 241–248.
[10] D.S. McCrickard, R. Catrambone, J.T. Stasko, Evaluating animation in the periphery as a mechanism for maintaining awareness, in: Proceedings of the IFIP TC.13 Conference on Human–Computer Interaction, Tokyo, Japan, July 2001, pp. 148–156.
[11] D.E. Berlyne, Conflict, Arousal, and Curiosity, McGraw-Hill, New York, 1960 (pp. 170–174).
[12] H.I. Day, Advances in Intrinsic Motivation and Aesthetics, Plenum, New York, 1981.
[13] M. Czerwinski, E. Cutrell, E. Horvitz, Instant messaging: effects of relevance and time, in: S. Turner, P. Turner (Eds.), People and Computers XIV: Proceedings of the HCI 2000, vol. 2, British Computer Society, 2000, pp. 71–76.
[14] N.A. Storch, Does the User Interface Make Interruptions Disruptive? A Study of Interface Style and Form of Interruption (Report UCRL-JC-108993), Lawrence Livermore National Laboratory, Springfield, 1992.
[15] S. Hudson, J. Fogarty, C. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. Lee, J. Yang, Predicting human interruptibility with sensors: a wizard of Oz feasibility study, in: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '03), ACM Press, New York, 2003, pp. 257–264.
[16] A. Wilson, M. Burnett, L. Beckwith, O. Granatir, L. Casburn, C. Cook, M. Durham, G. Rothermel, Harnessing curiosity to increase correctness in end-user programming, Proceedings of the CHI 2003 (2003) 305–312.
[17] M. Burnett, C. Cook, O. Pendse, G. Rothermel, J. Summet, C. Wallace, End-user software engineering with assertions in the spreadsheet paradigm, in: Proceedings of the 25th International Conference on Software Engineering, 2003, pp. 93–103.
[18] M. Burnett, J. Atwood, R. Djang, H. Gottfried, J. Reichwein, S. Yang, Forms/3: a first-order visual language to explore the boundaries of the spreadsheet paradigm, Journal of Functional Programming 11 (2) (2001) 155–206.
[19] M. Fisher, M. Cao, G. Rothermel, C. Cook, M. Burnett, Automated test generation for spreadsheets, in: Proceedings 24th International Conference on Software Engineering, 2002, pp. 141–151.
[20] R. Panko, What we know about spreadsheet errors, Journal of End User Computing (1998).
[21] G. Loewenstein, The psychology of curiosity: a review and reinterpretation, Psychological Bulletin 116 (1) (1994) 75–98.
[22] S.M. Hess, M. Detweiler, Training to reduce the disruptive effects of interruptions, in: Proceedings of the HFES 38th Annual Meeting, Nashville, TN, 1994, pp. 1173–1177.