COGNITIVE SCIENCE
A Multidisciplinary Journal

Cognitive Science 44 (2020) e12893

ISSN: 1551-6709 online
DOI: 10.1111/cogs.12893

How Experts Adapt Their Gaze Behavior When Modeling a Task to Novices

Selina N. Emhardt,[a] Ellen M. Kok,[b] Halszka Jarodzka,[a]
Saskia Brand-Gruwel,[a,d] Christian Drumm,[c] Tamara van Gog[b]

[a]*Department of Educational Sciences, Open Universiteit Nederland (Open University of the Netherlands)*
[b]*Department of Education, Utrecht University*
[c]*Faculty of Business Studies, FH Aachen University of Applied Sciences*
[d]*Zuyd University of Applied Sciences*

Received 11 September 2019; received in revised form 26 June 2020; accepted 4 August 2020

Abstract

Domain experts regularly teach novice students how to perform a task. This often requires them to adjust their behavior to the less knowledgeable audience and, hence, to behave in a more didactic manner. Eye movement modeling examples (EMMEs) are a contemporary educational tool for displaying experts' (natural or didactic) problem-solving behavior as well as their eye movements to learners. While research on expert-novice communication mainly focused on experts' changes in explicit, verbal communication behavior, it is as yet unclear whether and how exactly experts adjust their nonverbal behavior. This study first investigated whether and how experts change their eye movements and mouse clicks (that are displayed in EMMEs) when they perform a task naturally versus teach a task didactically. Programming experts and novices initially debugged short computer codes in a natural manner. We first characterized experts' natural problem-solving behavior by contrasting it with that of novices. Then, we explored the changes in experts' behavior when being subsequently instructed to model their task solution didactically. Experts became more similar to novices on measures associated with experts' automatized processes (i.e., shorter fixation durations, fewer transitions between code and output per click on the run button when behaving didactically). This adaptation might make it easier for novices to follow or imitate the expert behavior. In contrast, experts became less similar to novices for measures associated with more strategic behavior (i.e., code reading linearity, clicks on run button) when behaving didactically.

Correspondence should be sent to Selina Emhardt, Faculty of Educational Sciences, Open University of the Netherlands, 6419 AT Heerlen, The Netherlands. E-mail: selina.emhardt@ou.nl

Keywords: Eye tracking; Expertise; Expert–novice communication; Programming; Didactic behavior; Eye movement modeling examples

1. Introduction

Imagine you just wrote your very first, small computer program, but something just does not work. After unsuccessfully trying to find and correct the error (the “bug”) for a while, you consult your programming teacher—an expert in the field. After figuring out the solution, the expert wants to demonstrate to you how to solve the problem. In all likelihood, he would adjust his problem-solving behavior when explaining the solution to you in a didactic manner.

However, when trying to communicate to novices how to perform a task, experts’ verbal communication has its limitations. For instance, it may be difficult if not impossible for experts to verbalize automated processes (Ericsson & Simon, 1980). Moreover, they may have difficulties in correctly assessing novices’ prior knowledge (Hinds, 1999). This might make it difficult for experts to adequately adjust their explanations to a novice audience (e.g., the expert might be referring to a part of the task with a term that is unfamiliar to the novice, who, consequently, will have difficulty to follow and learn from the expert’s explanation).

Recent technological advances offer a strategy for overcoming such limitations. It is possible nowadays to create videos of task demonstrations by not only recording an expert’s problem-solving behavior and verbal explanations but also recording and visualizing their eye movements overlaid on the information the expert was looking at (Van Gog, Jarodzka, Scheiter, Gerjets, & Paas, 2009). Especially for tasks that rely strongly on processing visuospatial information, visualizations of experts’ eye movements (e.g., superimposed dots, circles, or spotlights onto a screen recording, Jarodzka et al., 2012) can make cognitive processes that are hard to verbalize, visible to learners. This can guide the learners’ attention to the information the expert is referring to at the right moment, which would foster their understanding. Video examples with task demonstrations by experts, with the expert’s eye movements superimposed on the task, are known as “eye movement modeling examples” (EMMEs; Jarodkza et al., 2013). When EMMEs are combined with screen-recording videos (e.g., for programming education), learners can follow the model’s nonverbal behavior by observing the model’s eye movements and mouse clicks during task performance.

In some studies, EMMEs display experts’ behavior as it naturally occurs (e.g., Litchfield, Ball, Donovan, Manning, & Crawford, 2010; Nalanagula, Greenstein, & Gramopadhye, 2006; Seppänen & Gegenfurtner, 2012; Stein & Brennan, 2004). In most studies, however, the experts were instructed to behave didactically, displaying successful strategies and modeling familiar process as explicitly as possible (Litchfield & Ball, 2011; Mason, Pluchino, & Tornatora, 2015a,b; Mason, Scheiter, & Tornatora, 2017; Scheiter, Schubert, & Schüler, 2018, Van Gog et al., 2009; Van Marlen, Van Wermeskerken, Jarodzka, & Van Gog, 2016, 2018). Jarodzka et al. (2012) and Jarodzka, Van Gog, Dorr, Scheiter, and Gerjets (2013) presented the expert models with prompts that were designed to evoke a didactic recipient focus and, hence, keep the novice audience in mind. Such prompts have been found previously to alter explicit verbal communication patterns (Jucks, Schulte-Löbbert, & Bromme, 2007). However, it is unclear whether and how experts' less explicit, nonverbal behavior, which is visible, for instance, in EMMEs, would change when explaining a task didactically.

The question addressed in the present study is how experts' nonverbal behavior (i.e., mouse clicks and eye movements during code debugging) changes when moving from their natural problem-solving mode to a didactic teaching mode. To draw conclusions about the direction of this change (i.e., becoming more or less similar to novices), we also investigated characteristics of novices' natural nonverbal behavior. This study aims to broaden our understanding of experts' behavior during expert–novice communication, by extending it to experts' nonverbal behavior. This is not only relevant for understanding expertise and expert–novice communication but also for educational research on example-based learning.

1.1. Fostering expert–novice communication with EMMEs

The audience design theory (Clark & Murphy, 1982) states that speakers create utterances with the intention to be understood by specific recipients. During expert–novice communication, experts should adjust their utterances to novices' needs. Hence, their verbal communication patterns change to appeal to the common ground of the communication partners—their mutual knowledge, beliefs, suppositions, and assumptions (Clark, Schreuder, & Buttrick, 1983). Isaacs and Clark (1987) found that experts used more words and more understandable object descriptions when talking to novices than when talking to other experts. Similarly, experts also adapt written explanations according to the knowledge of the recipient: When giving an explanation to a layperson, experts tend to use fewer specialist terms, explain issues in more detail, and use more illustrative examples than when addressing an expert (Bromme, Jucks, & Runde, 2005). In the study of Jucks et al. (2007), medical experts expanded their texts and made more meaningful revisions when being instructed to focus on the knowledge of the recipient, for instance by asking whether the used terms were familiar to the reader or explained in enough detail.

Still, experts' verbal explanations might often not be sufficient to successfully guide the novices' attention. First, experts might have difficulty verbalizing their processes due to the nature of the task (e.g., tasks with a large visual component) or their highly automatized task processing (Ericsson & Simon, 1980; Persky & Robinson, 2017; Samuels & Flor, 1997). Second, experts might not always be able to correctly assess novices' prior knowledge ("the curse of expertise": Hinds, 1999). Consequently, experts' verbal explanations might sometimes be too abstract (Hinds, Patterson, & Pfeffer, 2001) or ambiguous to be understood by novice learners (Van Marlen, Van Wermeskerken, Jarodzka, & Van Gog, 2018). For instance, programming experts might refer to "commenting lines" when talking about the symbol of "#" during programming, which might not be a familiar term for a real novice audience.

The rationale behind EMME is that displaying a model's eye movements may help to overcome these limitations through different mechanisms (Krebs, Schüler, & Scheiter, 2019). First, EMME can support learning by synchronizing the learners' visual attention with that of the model and, hence, guide attention to the relevant information at the right point in time (Jarodzka et al., 2013). When verbal explanations are present in an EMME, the eye movement displays have the potential to disambiguate the model's verbal references (Van Marlen et al., 2018), which should help the learner to follow the expert's process.

Second, EMME can reveal a model's unverbalized, and thus usually covert cognitive processes to the learner. Eye movements contain information about a performer's or model's attentional and cognitive processes (Just & Carpenter, 1980; Rayner, 1978). For instance, fixations occur when the eyes rest relatively still on an object and indicate what is at the center of visual attention. This usually indicates which information a performer is currently processing. Shifts in the center of attention to another location (i.e., saccades) may sometimes be indicative of search behavior or comparing behavior. Analyzing experts' and novices' eye movements can therefore give us valuable insights about their underlying cognitive and attentional processes and the literature in this field is growing in the last years (for a recent, systematic overview on eye-tracking studies in programming research, see Obaidellah, Al Haek, & Cheng, 2018). As Busjahn et al. (2014) stated: "The observation of eye movements adds an objective source of information about programmer behavior [. . .] which can be used to facilitate the teaching and learning of programming" (p. 1).

1.2. Differences in experts' and novices' natural problem-solving behavior

There is an extensive body of literature on differences between experts' and novices' problem-solving skills and behavior (e.g., Ericsson, Hoffman, Kozbelt, & Williams, 2018). Typically, domain experts systematically outperform novices in terms of task-solving speed and correctness (during programming, see e.g., Soh et al., 2012). In the following sections, we briefly introduce how expertise-related differences in knowledge representation (i.e., script-based, automatized knowledge) and strategies (i.e., chunking and forward reasoning) might be reflected in experts' and novices' nonverbal behavior during problem-solving tasks in general and debugging tasks in particular.

Boshuizen and Schmidt (2008) designed a model for the development of expert knowledge structures, which focuses on medical expertise and problem-solving. According to this model, novices first acquire general concepts and detailed knowledge that is used to reason in chains of small steps. This reasoning process has to be actively monitored and is cognitively demanding. With experience, the concepts cluster together and the knowledge network becomes increasingly well structured—a process that is called knowledge encapsulation. With more experience, scripts for solving specific tasks develop out of the encapsulated networks. Using these scripts leads to a fast, automated, script-guided, theory-driven, less effortful, and goal-oriented information processing. Nivala, Hauser, Mottok, and Gruber (2016) argued that this theory could also be fruitful for programming and debugging expertise, because, like medical diagnosis, debugging aims at finding errors in a malfunctioning system. Programming experts might acquire a knowledge repertoire of stereotypical behavior sequences and "programming plans" (Soloway & Ehrlich, 1984).

Experts' script-based task processing can account for experts' faster and automatized information processing with minimal conscious effort (Samuels & Flor, 1997). In light of Just and Carpenter's (1980) eye-mind and immediacy assumptions, which assert that what is in the focus of attention is processed at that moment, a faster and less effortful information processing should result in shorter fixation durations during programming (as observed, e.g., in Bednarik, Myller, Sutinen, & Tukiainen, 2005; Bednarik & Tukiainen, 2005; Nivala et al., 2016; Sharif, Falcone, & Maletic, 2012). Furthermore, if experts apply knowledge-based programming scripts, they should process code not simply linearly (e.g., linewise from top to the bottom of the code, similar to regular text reading), but in a knowledge-driven and logical manner (e.g., Busjahn et al., 2015; Lin et al., 2015; Nanja & Cook, 1987).

Another difference between experts and novices lies in their problem-solving *strategies*. One strategy is to cluster or "chunk" task elements together to process them in larger units more effectively (Chase & Simon, 1973; Reingold, Charness, Pomplun, & Stampe, 2001; Reingold & Sheridan, 2011; Van Meeuwen et al., 2014). For instance, programming experts could process (chunk) frequently occurring code elements together (e.g., perceiving an expression like "for i in list_name": as one typical element to create a loop). Via this more holistic processing, a task that originally required detection in several parts can be accomplished by detecting a single unit. As a result, experts might not fixate on each (code) element individually, which could increase their saccade amplitudes (e.g., Busjahn et al., 2015). Additionally, chunking mechanisms allow experts to keep more information present in visual working memory (Bauhoff, Huff, & Schwan, 2012). A larger visual memory capacity might especially help to keep more relevant information activated without the need to revisit already processed information, such as new output, when running the code.

Another strategy that experts might apply is "forward reasoning" (Katz & Anderson, 1987; cf. Van Meeuwen et al., 2014). Programming experts seem to first spend time building a mental representation of the problem statement (e.g., the code) and only later direct their attention toward the outcome (e.g., erroneous output). In contrast, novices reason "backwards" from the goal by first inspecting the erroneous output and try to reason backward to the code in order to search the errors there. The effort-demanding means-end problem-solving strategy consists of continuously orienting toward the goal (the "end," here the output) and applying operators (the "means," here changes in the code) to identify the next problem-solving step based on the outcome of the previous changes (Simon, 1975). Novices' backward reasoning behavior should lead to a more trial-and-error-based problem-solving approach. In the protocol study by Nanja and Cook (1987), experts indeed first tried to understand the code and ran the code less frequently during debugging and, thus, showed a less trial-and-error, backward-directed approach. In eye-tracking studies, programming experts were found to spend more time initially scanning and trying to understand code before concentrating on specific code parts (Sharif et al., 2012). Longer scan times were furthermore linked to less overall time to find bugs (Uwano, Nakamura, Monden, & Matsumoto, 2006).

Altogether, there is a large body of expertise literature and theories that can be used as a basis for the comparison of experts' and novices' problem-solving behavior. However, little is known about the changes in experts' (nonverbal) problem-solving behavior when communicating their knowledge to novices.

1.3. Exploring experts' change in nonverbal problem-solving behavior when acting didactically

Previous studies have shown that experts adjust their verbal explanations to communicate their knowledge to a less knowledgeable audience (e.g., Bromme et al., 2005; Isaacs & Clark, 1987; Jucks et al., 2007; Nückles, Winter, Wittwer, Herbert, & Hübner, 2006; Persky & Robinson, 2017). However, it is yet unclear whether and how experts' nonverbal behavior (such as eye movements and clicks on the run button during programming) changes when modeling a task didactically. In this context, an important question is whether task performers are aware of their eye movements and can deliberately adapt their eye movements to task instructions or the social context, for instance to behave didactically. Humans seem to possess only limited awareness about their own visual behavior. For example, people experience difficulties distinguishing displays of their own and other task performers' eye movements (Foulsham & Kingstone, 2013; Van Wermeskerken, Litchfield, & Van Gog, 2018) and have difficulties recalling where they looked during a previous task (Clarke, Mahon, Irvine, & Hunt, 2016; Kok, Aizenman, Võ, & Wolfe, 2017; Võ, Aizenman, & Wolfe, 2016). Eye movements are difficult to change at will, as Hooge and Erkelens (1998) showed, for example, for the case of fixation durations. This limited awareness could make it difficult for task performers to adapt their eye movements at will.

While it is likely that the instruction to behave didactically will cause a change in experts' eye movement behavior, it is difficult to predict what these changes are. On the one hand, experts' cognitive processes could, for instance, slow down when behaving didactically, which might lead to a behavior that is more similar to novices' behavior. On the other hand, by exemplifying their substantially different problem-solving behavior to novices, experts' nonverbal behavior could also become (even) less similar to that of novices.

An exploratory investigation of experts' behavioral changes can contribute to the more general understanding of expert behavior during expert–novice communication. More specifically, we can fill the literature gap about experts' didactic, nonverbal behavior using eye-tracking technology. In the context of EMME research, this investigation might raise researchers' and practitioners' awareness about the influence of how the model is instructed on EMMEs.

1.4. Overview of the present study

Our first research question (Research Question 1) was how experts' and novices' natural, nonverbal behavior (i.e., mouse clicks and eye movements that are also displayed in screen-recording EMMEs) differs during debugging. This investigation was done in a hypothesis-driven manner and was based on established expertise theories and findings from prior empirical studies.

Based on the assumption that experts use more automatized and script-based knowledge (cf. Boshuizen & Schmidt, 2008) than novices, we expected that experts would perform shorter average fixations in the code area (Hypothesis 1) and process code less linearly (here approximately linewise, Hypothesis 2).

Based on the assumption that experts use more chunking strategies than novices (cf. Charness, Reingold, Pomplun, & Stampe, 2001; Chase & Simon, 1973), we expected that they should perform longer average saccade amplitudes in the code area, because not every element needs to be processed (Hypothesis 3). The accompanying increase in working memory capacity (cf. Bauhoff et al., 2012) should cause experts to require fewer transitions between newly created output and the code (Hypothesis 4).

Based on the assumption that experts first build a mental representation of the problem and reason (forward) from the code toward the output (cf. Katz & Anderson, 1987), we expected that experts would take longer until first testing the code (Hypothesis 5). In addition, experts should test the code less frequently than novices (indicating forward reasoning and at the same time less trial-and-error-based strategy Hypothesis 6).

Addressing Research Question 1 extends classic expertise research to the domain of programming and is necessary to subsequently draw conclusions about Research Question 2, which was whether and how (i.e., in which direction) experts change their nonverbal behavior (becoming more or less similar to novices) when modeling a task to novices didactically. Due to a lack of existing literature on this topic, we did not specify hypotheses, but analyze this question exploratorily, using the same measures as for Research Question 1.

2. Methods

2.1. Participants

After excluding two experts (one indication of missing expertise and one case of weak tracking ratio of 25.2%, $M_{\text{tracking ratio}} = 87.81\%$, $SD_{\text{tracking ratio}} = 8.37$), the remaining sample consisted of 22 experts (2 female, 20 male; $M_{\text{age}} = 29.82$, $SD_{\text{age}} = 7.14$) and 18 novices (7 female, 11 male; $M_{\text{age}} = 23.59$, $SD_{\text{age}} = 2.62$). All experts were employed as professional programmers and all novices were university students who had just participated in a programming introductory course. Experts reported on average 11.97 ($SD = 7.36$) years of programming experience, 4.16 ($SD = 3.28$) years of Python experience, and 27.14 ($SD = 11.54$) hours of weekly programming activities. Novices reported on average 0.99 ($SD = 1.65$) years of programming experience, 0.29 ($SD = 0.51$) years of Python experience, and 4.91 ($SD = 4.47$) hours of weekly programming activities.

Six experts stated that they had at least some experience in teaching programming. The other participants had no teaching experience. All except one participant were non-

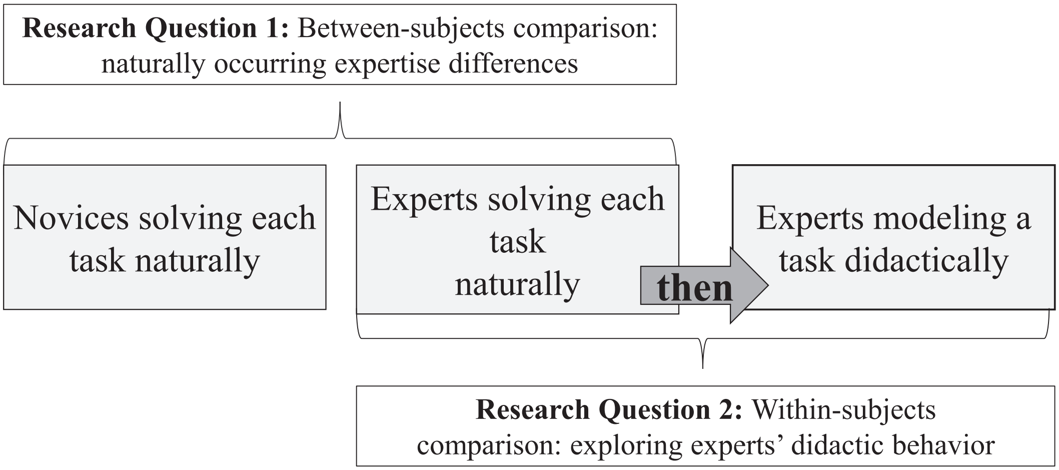

Research Question 1: Between-subjects comparison: naturally occurring expertise differences
Novices solving each task naturally
Experts solving each task naturally
then
Experts modeling a task didactically
Research Question 2: Within-subjects comparison: exploring experts' didactic behavior

Fig. 1. Schematic overview of the experimental design and the procedure.

native English speakers, who worked with English code material. The compensation for study participation was € 15 for the experts and € 10 for the novices.

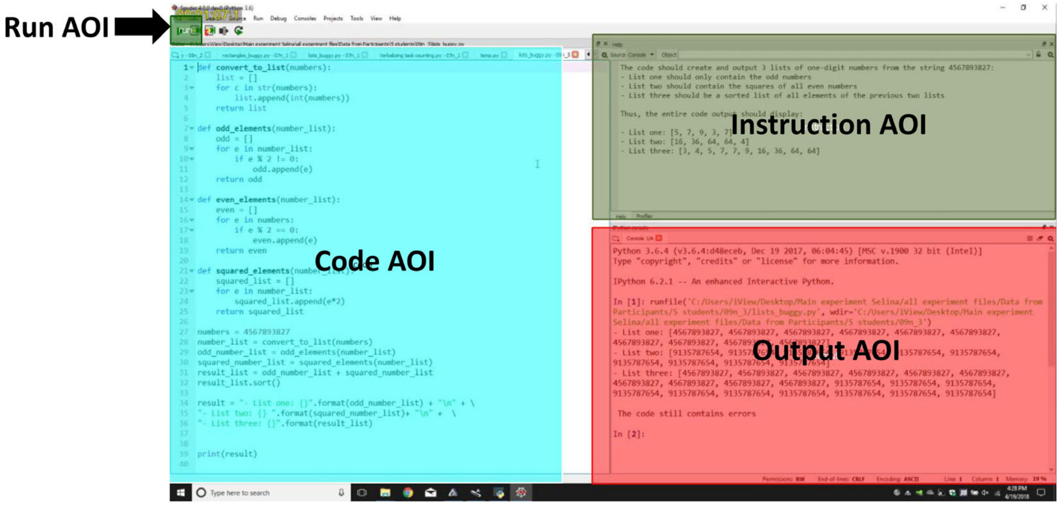
2.2. Design

Research Question 1 concerned a between-subjects comparison of novices' and experts' natural debugging behavior, whereas Research Question 2 concerned a within-subject comparison of experts' natural and didactic behavior. The study design is visualized in Fig. 1.

2.3. Materials

2.3.1. Code material

All code snippets were in the programming language Python and were presented in English, the standard language for computer code. The instruction as well as the erroneous codes and their solutions can be found in the Data S1 (Task 1, Task 2, and Task 3). Task 1, "printing rectangles," was based on Fitzgerald et al. (2008) and consisted of 43 lines of code. Task 2, "printing S,"[1] consisted of 38, and Task 3, "list manipulation," consisted of 39 lines. Each code snippet contained four nonsyntactic bugs, which means that they prevent the program from running as intended but that compilers and interpreters are not able to detect them (Gould, 1975, p. 152). The bugs can be categorized as misplaced, malformed, and missing statements ("bugs"). Misplaced statements are code components that appear in the wrong place of the program. Malformed statements are components that were formulated incorrectly but appear at the right place of the program. Missing statements are statements in which a required component was omitted (Fitzgerald et al., 2008; Johnson, Soloway, Cutler, & Draper, 1983).



```
def convert_to_list(numbers):
    list = []
    for c in str(numbers):
        list.append(int(numbers))
    return list

def odd_elements(number_list):
    odd = []
    for e in number_list:
        if e % 2 != 0:
            odd.append(e)
    return odd

def even_elements(number_list):
    even = []
    for e in numbers:
        if e % 2 == 0:
            even.append(e)
    return even

def squared_elements(number_list):
    squared_list = []
    for e in number_list:
        squared_list.append(e*2)
    return squared_list

numbers = 4567893827
number_list = convert_to_list(numbers)
odd_number_list = odd_elements(number_list)
squared_number_list = squared_elements(number_list)
result_list = odd_number_list + squared_number_list
result_list.sort()

result = "- List one: {}".format(odd_number_list) + "\n" + \
"- List two: {} ".format(squared_number_list)+ "\n" +  \
"- List three: {}".format(result_list)


print(result)
```

```
The code should create and output 3 lists of one-digit numbers from the string 4567893827:
- List one should only contain the odd numbers
- List two should contain the squares of all even numbers
- List three should be a sorted list of all elements of the previous two lists

Thus, the entire code output should display:

- List one: [5, 7, 9, 3, 7]
- List two: [16, 36, 64, 64, 4]
- List three: [3, 4, 5, 7, 7, 9, 16, 36, 64, 64]
```

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 6.2.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/iView/Desktop/Main experiment Selina/all experiment files/Data from
Participants/5 students/09n_3/lists_buggy.py', wdir='C:/Users/iView/Desktop/Main experiment
Selina/all experiment files/Data from Participants/5 students/09n_3')
- List one: [4567893827, 4567893827, 4567893827, 4567893827, 4567893827, 4567893827,
4567893827, 4567893827, 4567893827, 4567893827]
- List two: [9135787654, 9135787654, 9135787654, 9135787654, 9135787654, 9135787654,
9135787654, 9135787654, 9135787654, 9135787654]
- List three: [4567893827, 4567893827, 4567893827, 4567893827, 4567893827, 4567893827,
4567893827, 4567893827, 4567893827, 4567893827, 9135787654, 9135787654, 9135787654,
9135787654, 9135787654, 9135787654, 9135787654, 9135787654, 9135787654, 9135787654]

 The code still contains errors

In [2]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 1 Column: 1 Memory: 19 %

4:28 PM 4/19/2018

Fig. 2. Screenshot of the programming environment with superimposed areas of interest (AOIs).

2.3.2. *Integrated development environment (IDE)*

The Python code snippets were presented in the standard Python programming IDE Spyder 4.0, which was adjusted for the experiment (i.e., the participants could not scroll and change the size of the main areas). The interface of the IDE consisted of four areas of interest (AOIs): the code area (Editor, screen coverage: 38.7%), the output area (Console, coverage: 26.9%), the task instruction area (coverage = 19.3%), and the run button (coverage = 0.2%). Fig. 2 shows the division of the screen into the AOIs.

2.4. *Apparatus*

The study was created in the SMI Experiment Center software (version 3.7; SensoMotoric Instruments GmbH, Teltow, Germany) and was presented on a 15.6-inch monitor with a resolution of 1,920 × 1,080 pixels. Eye movements of both groups were recorded binocularly at 250 Hz using an SMI 250 RED (also MI RED-m) infrared remote mobile eye tracker (SensoMotoric Instruments GmbH) with a forehead rest. We used a velocity-based event detection algorithm from SMI BeGaze with a threshold of 80 ms (cf. Bednarik et al., 2005).

2.5. *Procedure*

All participants sat in front of a screen with an approximate viewing distance of 60 cm. Novices were tested one by one in an empty classroom of their university. Experts were tested in their offices. First, the participants answered demographic questions and read a short introduction to their task to debug short Python code snippets. They were informed that the program codes included several bugs, but that each bug could be fixed by changing at most one line of code. All participants then inspected a screenshot of the IDE with short descriptions of the relevant IDE areas. Prior to the main experiment, participants debugged one short exemplary code snippet in the IDE and practice thinking out loud in their native language while debugging.

Then, all participants debugged two out of the three code snippets in random order while thinking out loud, each with a time limit of 20 min (based on pilot trials). During this time, they could freely inspect, run, and manipulate the code. The output provided feedback on whether the code still contained errors or not.

After debugging each code snippet, the group of experts saw an exemplary 15-second dynamic display of another person's eye movements while looking at code and were subsequently instructed to create an instructional EMME for the previously solved task, again with the instruction to think aloud. The experts were instructed to behave didactically using the following instructions and prompts that were adapted from Jarodzka et al. (2012) (translated from German to English):

Imagine a student who has very little experience in programming and debugging asks you: "What are the key ways to fix the bugs in this code?"

[For the creation of the instructional video,] you should consider the following criteria:

1. It is important that the student knows the meaning of all terms.
2. For this student, the debugging process is explained in an understandable way.
3. For this student, the debugging process is explained in detail.
4. All the information the student needs is included.
5. All mentioned information is important for the student.

The experts repeated this same procedure for two debugging tasks. Four experts indicated that they had spare time and hence agreed to solve the third debugging task. These data were included in the analysis.

2.6. *Data analysis*

The performance of one expert solving one item was excluded, because he indicated confusion during the task. As for performance, a bug was categorized as fixed when the changes in the code resulted in the correct functioning of the code. As for the eye movement data, fixation durations (H1), linearity measures (H2), saccade amplitudes (H3), and transitions between AOI (H4) were determined. For the mouse click data, the time to first use of the run button (H5) and the total use of the run button (H6) were determined. For the analysis of fixation durations in the code area, fixations within the code area that deviated more than 3 *SD*s from each person's mean duration were excluded (1.89% of all data); we then calculated the average fixation duration per person per item. For measuring code reading linearity, we used a simplified linearity measure. First, we selected all fixations and their px-based coordinates within the code area. If the subsequent fixation was in the same line or moved one line down (between 0 and 33px on the y-axis), this change was classified as linear downward fixation behavior (compare "vertical next text" measure (%) of forward saccades that either stay on the same line or moved one line down; Busjahn et al., 2015). Larger downward movements and all upward movements were categorized as nonlinear fixation behavior (compare "vertical next text" and "regression rate" measure of Busjahn et al., 2015). The relative frequency of linear fixation behavior was calculated per person per item. For the analysis of saccade amplitudes in the code area, we included saccades that had a start and end point within the code, based on the pixel coordinates of the AOIs. Saccade amplitudes in degrees of visual angle (°) were provided by SMI BeGaze software. Erroneous recordings that indicated saccade amplitudes larger than 30 (i.e., the AOI size) were excluded from the analysis. We then calculated the average saccade amplitude per person per item for further analysis, but one datapoint of one person for one item was excluded as outlier from further analysis based on the visual inspection. This outlier deviated 8.68 *SD*s from the mean of the natural (instruction absent) condition. The average number of transitions between the code and the output area per click on the run button was calculated based on the transition matrix provided by SMI BeGaze per person per item. The time until the participants first ran the code (first click on the run button) and the total code running frequency (sum of all clicks on the run button) were retrieved from BeGaze per person per item.

All following analyses were conducted in R (R Core Team, 2015) with a significance level of $a = 0.05$. To answer both research questions, we compared different linear mixed-effect regression models. Such models are flexible in processing datasets with unbalanced designs, in which not all participants solve the same items and, thus, do not require averaging the data for each person over all items (Baayen, Davidson, & Bates, 2008). All models included an intercept (β0) and effects of item and participant as random intercepts, which take into account the nonindependence of the data on a person and item level (Barr, 2008). In the final analysis, we compared this model with a model that additionally included the fixed effect (β1) of expertise (naturally behaving experts vs. novices, RQ1) or instruction (absent: naturally behaving experts vs. present: didactically behaving experts, RQ2). This allowed for analyzing if and in what direction the factor levels served as significant predictor for each outcome variable. Furthermore, the marginal R^2_{m} describes the proportion of variance explained by the included fixed factor alone and the conditional R^2_{c} indicates the explained variance by the full model, including random effects (Nakagawa & Schielzeth, 2012). The models were analyzed using the lme4 package (Bates, Mächler, Bolker, & Walker, 2015) for R (R Core Team, 2015). Model assumptions were checked graphically prior to the analysis. The quantile–quantile (qq) plots indicated possible violations for the assumption of normally distributed model residuals for some of the models. We subsequently performed Shapiro–Wilk tests to further test the normality of the model residuals. Table 2 in Data S1 provides the detailed results of these tests. For those models that yielded significant test results, we visually confirmed that a log transformation improved the qq plots. Table 3 in Data S1 shows the quantile–-quantile plots of the model residuals before and after log transformations for the critical models.

As additional analysis and to explore if the groups behaved more or less homogeneously in the different conditions, the variances of observations for each measure were compared between the conditions (naturally behaving experts vs. novices or naturally behaving vs. didactically behaving experts) using a Levene's test. For this analysis, data from one expert who only solved a task naturally had to be excluded.

3. Results

We first analyzed the general performance of the participants in all conditions. Novices took on average 19.06 (*SD* = 2.50) minutes to fix 1.03 (*SD* = 1.43) out of four bugs, whereas naturally behaving experts took 7.60 (*SD* = 3.93) minutes to fix 3.95 (*SD* = 0.21) per code. Didactically behaving experts took on average 5.47 (*SD* = 1.66) minutes to explain the task.

Table 1 displays the regression confidents with standard errors for all subsequent models.

3.1. Comparison of naturally behaving experts and novices (Research Question 1)

The analyses in this section always consist of a comparison of the statistical models with and without the (between-subjects) factor expertise (naturally behaving experts vs. novices) as an independent variable on the outcome measures. Fig. 3 illustrates the results.

3.1.1. Fixation durations in code area

For this variable, expertise was a predictor that significantly improved the model fit; $\chi^2(1) = 38.19$, $p < .001$, $R^2_{\mathrm{m}} = 0.61$, $R^2_{\mathrm{c}} = 0.97^2$. Naturally behaving experts showed shorter fixations in the code area than novices; $M_{\mathrm{experts}} = 293.18$ ms, $SD_{\mathrm{experts}} = 55.09$ ms, $M_{\mathrm{novices}} = 483.93$ ms, $SD_{\mathrm{novices}} = 92.89$ ms. Experts had a significantly smaller variance in average fixation durations than novices; $F(1, 79) = 7.46$, $p = .008$.

3.1.2. Code reading linearity

For this variable, expertise was a predictor that significantly improved the model fit; $\chi^2(1) = 22.79$, $p < .001$, $R^2_{\mathrm{m}} = 0.38$, $R^2_{\mathrm{c}} = 0.76$. Naturally behaving experts showed a higher reading linearity than novices; $M_{\mathrm{experts}} = 33.91\%$, $SD_{\mathrm{experts}} = 3.21$, $M_{\mathrm{novices}} = 28.53$, $SD_{\mathrm{novices}} = 3.77$. The variance of average code reading linearity did not differ between the two conditions; $F(1, 79) = 1.93$, $p = .169$.

3.1.3. Saccade amplitudes in code area

For this variable, expertise was not a predictor that significantly improved the model fit; $\chi^2(1) = 1.00$, $p = .318$, $R^2_{\mathrm{m}} = 0.02$, $R^2_{\mathrm{c}} = 0.89$, $M_{\mathrm{experts}} = 0.87°$, $SD_{\mathrm{experts}} = 0.36°$, $M_{\mathrm{novices}} = 0.97°$, $SD_{\mathrm{novices}} = 0.26°$. The variance of the average saccade amplitude did not differ between the two conditions; $F(1, 78) = 0.81$, $p = .372$.

3.1.4. Log-transformed amount of transitions per running the code

For this variable, expertise was a predictor that significantly improved the model; $\chi^2(1) = 4.35$, $p = .037$, $R^2_{\mathrm{m}} = 0.06$, $R^2_{\mathrm{c}} = 0.17$. Naturally behaving experts performed fewer transitions per code run than novices; $M_{\mathrm{experts}} = 5.12$, $SD_{\mathrm{experts}} = 2.87$, $M_{\mathrm{novices}} = 6.54$, $SD_{\mathrm{novices}} = 3.66$. The variance of average amount of transitions per running of the code did not differ between the two conditions; $F(1, 79) = 0.87$, $p = .355$.

Table 1
Regression coefficients β0 and β1 with standard errors for all model comparisons

	Research Question 1: Effect of Expertise (naturally behaving experts vs. novices)		Research Question 2: Effect of Instruction (naturally vs. didactically behaving experts)		Model Comparison of Novices and Didactically Behaving Experts	
	β0($SE_{\beta 0}$)	β1($SE_{\beta 1}$)	β0($SE_{\beta 0}$)	β1($SE_{\beta 1}$)	β0($SE_{\beta 0}$)	β1($SE_{\beta 1}$)
Fixation durations in code area in ms	291.57 (15.62)	186.69*** (23.31)	314.24 (13.41)	−23.39*** (4.62)	308.83 (17.47)	169.33*** (25.74)
Code reading linearity in %	0.34 (0.01)	−0.05*** (0.01)	0.35 (0.01)	−0.01* (0.01)	—	—
Saccade amplitudes in code area in degrees of visual angle	0.87 (0.07)	0.10 (0.10)	−0.11 (0.09)	−0.11** (0.04)	−0.12 (0.09)	0.06 (0.12)
Number of transitions per running the code	1.49 (0.08)	0.26* (0.12)	1.84 (0.16)	−0.36*** (0.10)	1.83 (0.13)	-0.08 (0.14)
Time until first running the code in min	−0.45 (0.23)	0.35 (0.33)	−0.53 (0.21)	0.09 (0.20)	—	—
Code running frequency	2.15 (0.13)	0.82*** (0.15)	1.67 (0.08)	0.47*** (0.09)	—	—

β0 is the intercept and β1 is the regression coefficient for either the effect of expertise (Research Question 1) or the effect of expert instruction (Research Question 2). We also report the results for the additional comparison between novices and didactically behaving experts for cases when naturally behaving experts differed from novices in Research Question 1, but became more similar to novices when behaving didactically in Research Question 2.

*(p < .05), **(p < .01), and ***(p < .001) indicate a significant contribution of β1 to the prediction quality of the model in comparison to the null model without β1.

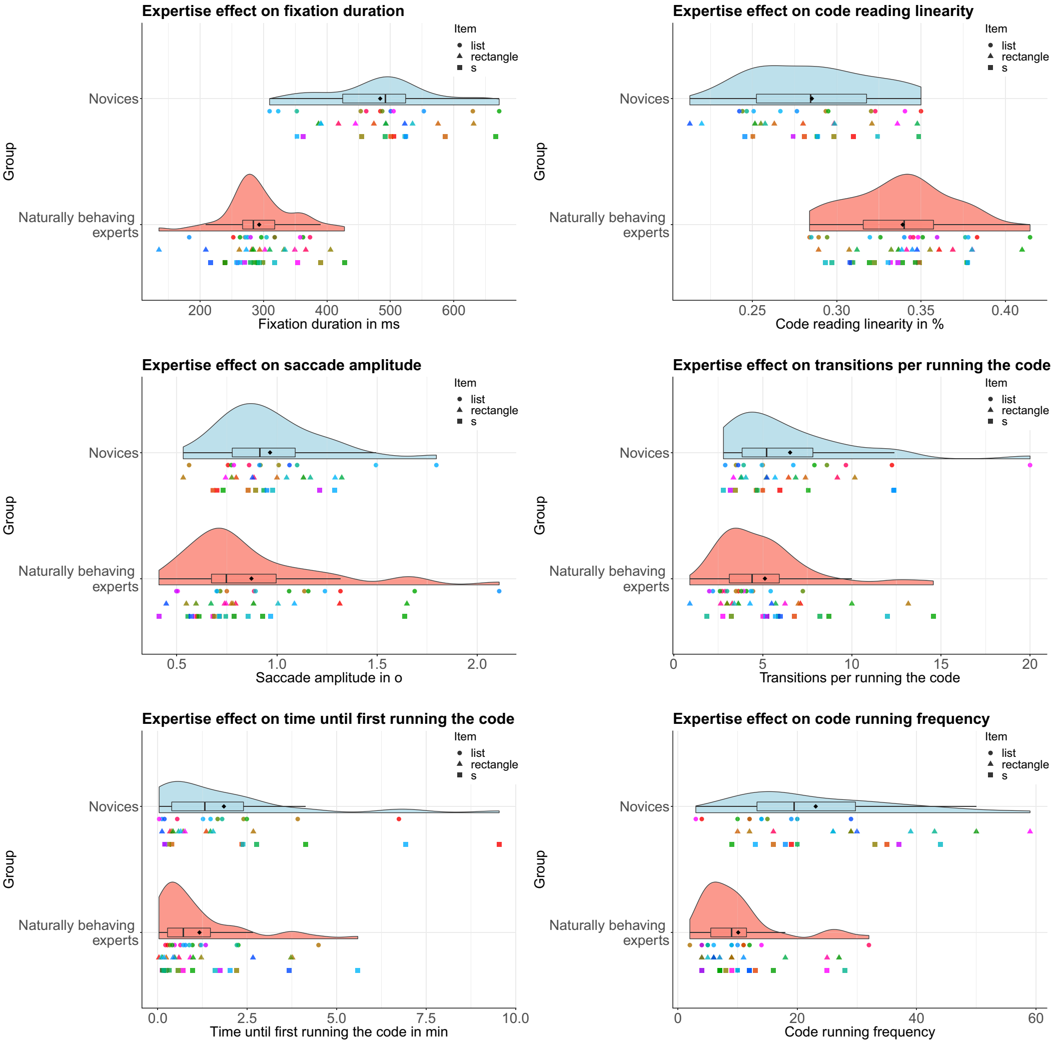


Fig. 3. Raincloud plots (Allen, Poggiali, Whitaker, Marshall, & Kievit, 2018) for each measure of Research Question 1. Each dot shows the means for each person (distinguished by color) and item (distinguished by shape) for novices (top) and experts (bottom). Additionally, the density distribution and boxplots for these data points are displayed for both groups.

3.1.5. Log-transformed time until first running of the code

For this variable, expertise was not a predictor that significantly improved the model fit; χ^2 (1) = 1.09, $p = .297$, $R^2_{\mathrm{m}} = 0.02$, $R^2_{\mathrm{c}} = 0.47$, $M_{\mathrm{experts}} = 1.09$ min, $SD_{\mathrm{experts}} = 1.15$ min, $M_{\mathrm{novices}} = 1.85$ min, $SD_{\mathrm{novices}} = 2.19$ min. The variance of average time until first running of the code did not differ between the two conditions; $F(1, 79) = 3.37$, $p = .070$.

3.1.6. Log-transformed code running frequency

For this variable, expertise was a predictor that significantly improved the model fit; $\chi^2(1) = 21.90$, $p < .001$, $R^2_{\mathrm{m}} = 0.30$, $R^2_{\mathrm{c}} = 0.50$. Naturally behaving experts ran the code less frequently than novices; $M_{\mathrm{experts}} = 10.11$, $SD_{\mathrm{experts}} = 6.96$, $M_{\mathrm{novices}} = 23.09$, $SD_{\mathrm{novices}} = 13.37$. Experts had a significantly smaller variance in (untransformed) average code running frequency than novices; $F(1, 79) = 13.01$, $p < .001$.

3.2. Experts' natural and didactic modeling behavior (Research Question 2)

The analyses in this section always consist of a comparison of the statistical models with and without the (within-subjects) factor instruction (absent: natural expert behavior vs. present: didactical expert behavior) as an independent variable on the outcome measures. Fig. 4 illustrates the results.

For some measures, we found a significant difference between experts' natural and didactic behavior. When the direction of this difference showed that the experts' behavior became more similar to novices' behavior on this variable, we additionally explored whether there was (still) a significant difference between didactically behaving experts and novices.

3.2.1. Average fixation durations in code area

For this variable, instruction was a predictor that significantly improved the model fit; $\chi^2(1) = 21.65$, $p < .001$, $R^2_{\mathrm{m}} = 0.03$, $R^2_{\mathrm{c}} = 0.89$. When behaving didactically, experts showed longer fixations in the code area than when behaving naturally; $M_{\mathrm{didactic}} = 312.33$ ms, $SD_{\mathrm{didactic}} = 69.34$ ms, $M_{\mathrm{natural}} = 293.18$ ms, $SD_{\mathrm{natural}} = 55.09$ ms. The variance of the average fixation durations did not differ between the two conditions; $F(1, 90) = 1.28$, $p = .261$. Since experts' behavior became more similar to novices' behavior when being instructed to behave didactically, we explored whether novices and didactically behaving experts still differed significantly in their average fixation durations. Indeed, didactically behaving experts still showed significantly shorter fixation durations than novices; $\chi^2(1) = 29.09$, $p < .001$, $R^2_{\mathrm{m}} = 0.52$, $R^2_{\mathrm{c}} = 0.97$.

3.2.2. Code reading linearity

For this variable, instruction was a predictor that significantly improved the model fit; $\chi^2(1) = 5.46$, $p = .020$, $R^2_{\mathrm{m}} = 0.03$, $R^2_{\mathrm{c}} = 0.51$. When behaving didactically, experts showed even more linear reading behavior than when behaving naturally; $M_{\mathrm{didactic}} = 35.34\%$, $SD_{\mathrm{didactic}} = 4.60\%$, $M_{\mathrm{natural}} = 33.91\%$, $SD_{\mathrm{natural}} = 3.21\%$. The code reading linearity did not differ between the two conditions; $F(1, 90) = 3.73$, $p = .057$.

3.2.3. Log-transformed saccade amplitudes in code area

For this variable, instruction was a predictor that significantly improved the model fit; $\chi^2(1) = 8.63$, $p = .003$, $R^2_{\mathrm{m}} = 0.02$, $R^2_{\mathrm{c}} = 0.83$. When behaving didactically, experts showed larger saccade amplitudes in the code area than when behaving naturally; $M_{\mathrm{didactic}} = 1.01°$, $SD_{\mathrm{didactic}} = 0.52°$, $M_{\mathrm{natural}} = 0.87°$, $SD_{\mathrm{natural}} = 0.36°$. The variance of the (untransformed)

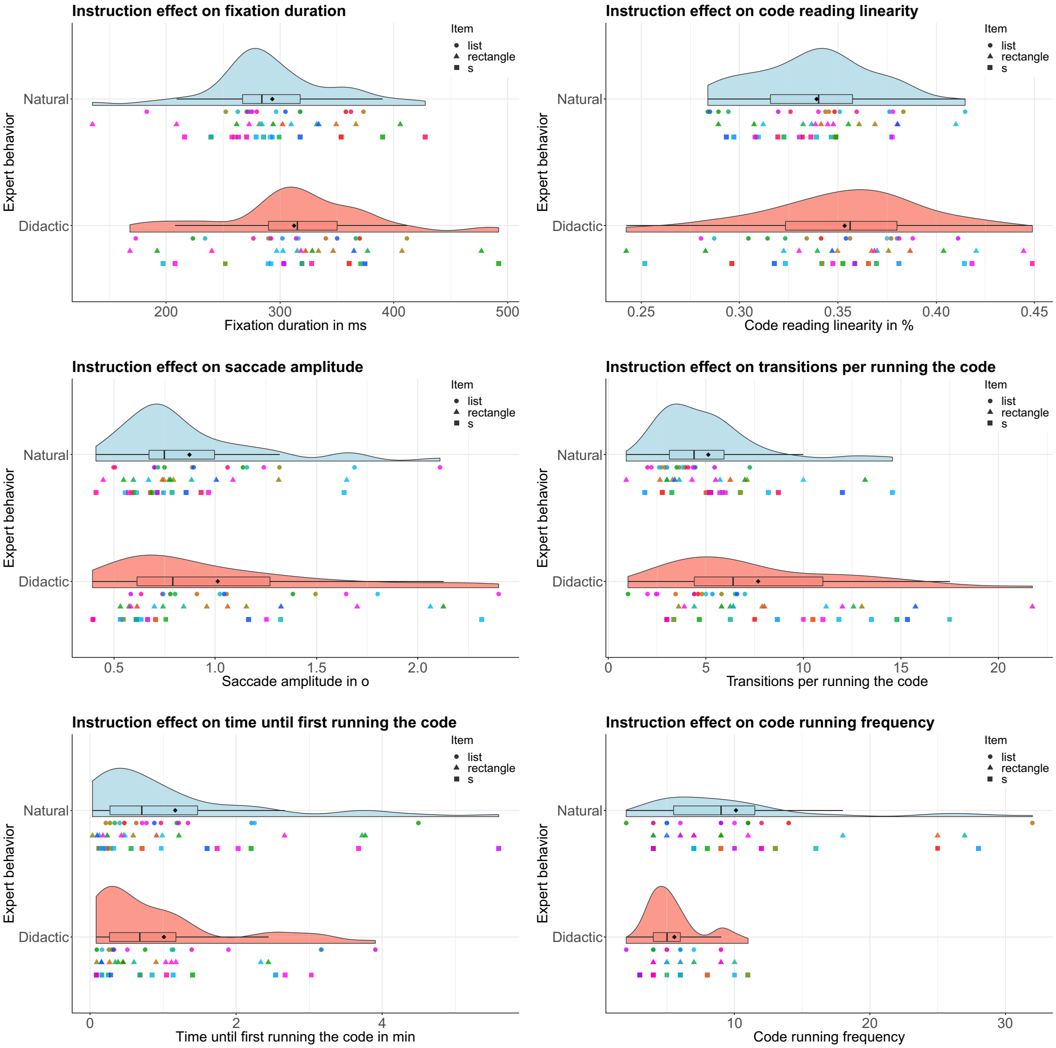


Fig. 4. Raincloud plots (Allen, Poggiali, Whitaker, Marshall, & Kievit, 2019) for each measure of Research Question 2. Each dot shows the means for each person (distinguished by color) and item (distinguished by shape) for the experts behaving naturally (top) and didactically (bottom). Additionally, the density distribution and boxplots for these data points are displayed for both groups.

average saccade amplitudes did not differ between the two conditions; $F(1,\ 88) = 3.78$, $p = .055$. Since experts' behavior became more similar to novices when they were instructed to behave didactically, we explored whether novices and didactically behaving experts still differed significantly for this measure. We found no effect of condition between didactically behaving experts and novices could be found for (log-transformed) saccade amplitude in code area; $\chi^2(1) = 0.27, p = .603, R^2_m = 0.01, R^2_c = 0.85$.

3.2.4. Log-transformed transitions per running the code

For this variable, instruction was a predictor that significantly improved the model fit; $\chi^2(1) = 11.76$, $p < .001$, $R^2_m = 0.09$, $R^2_c = 0.39$. When behaving didactically, experts showed more transitions between code and output area per running the code than when behaving naturally; $M_{\text{didactic}} = 7.68$, $SD_{\text{didactic}} = 4.67$, $M_{\text{natural}} = 5.12$, $SD_{\text{natural}} = 2.87$. Didactically behaving experts had a significantly larger variance in average transitions per running the code than naturally behaving experts; $F(1, 90) = 7.03$, $p = .009$. Since experts became more similar to novices when being instructed to behave didactically, we explored whether novices and didactically behaving experts still differed significantly for this measure. We could not find a significant effect of group between didactically behaving experts and novices for (log-transformed) saccade amplitudes in code area; $\chi^2(1) = 0.32$, $p = .571$, $R^2_m = 0.005$, $R^2_c = 0.28$.

3.2.5. Log-transformed time until first running the code

For this variable, instruction was not a predictor that significantly improved the model fit; $\chi^2(1) = 0.21$, $p = .655$, $R^2_m = 0.002$, $R^2_c = 0.27$, $M_{\text{didactic}} = 1.01$ min, $SD_{\text{didactic}} = 1.01$ min, $M_{\text{natural}} = 1.09$ min, $SD_{\text{natural}} = 1.15$ min. The variance of average time until first running the code did not differ between the two conditions; $F(1, 90) = 0.40$, $p = .529$.

3.2.6. Log-transformed code running frequency

For this variable, instruction was a predictor that significantly improved the model fit; $\chi^2(1) = 22.88, p < .001, R^2_m = 0.19, R^2_c = 0.37$. When behaving didactically, experts ran the code less often than when behaving naturally; $M_{\text{didactic}} = 5.56$, $SD_{\text{didactic}} = 2.09$, $M_{\text{natural}} = 10.11$, $SD_{\text{natural}} = 6.96$. Didactically behaving experts had a significantly smaller variance in average code running frequency than naturally behaving experts; $F(1, 90) = 14.83, p < .001$.

4. Discussion

This study investigated how experts adjust their nonverbal behavior (i.e., eye and mouse movements during code debugging) to model a problem-solving task didactically to novices. We first compared experts' and novices' naturally occurring nonverbal behavior during code debugging in a hypothesis-driven manner (Research Question 1). Subsequently, we explored whether and how experts change their nonverbal behavior when being instructed to model the tasks in a didactic manner (Research Question 2).

4.1. How did experts' and novices' nonverbal behavior differ?

The comparison of experts' and novices' natural debugging behavior was done in a hypothesis-driven manner (H1–H6). The findings for H1, H4, and H6 were in line with our expectations based on established expertise theories. First, experts performed shorter fixations in the code area than novices (H1). We associated this measure with experts' use of encapsulated knowledge structures and problem-solving scripts (cf. Boshuizen & Schmidt, 2008) that allow for automatized, faster information processing (cf. Ericsson, Krampe, & Tesch-Römer, 1993). This is in line with the observation that experts took, on average, less time to complete the debugging tasks, but it gives us additional, more fine-grained insight into their problem-solving processes (i.e., shorter time on task might also have occurred for other reasons). The finding that experts perform shorter fixations is, furthermore, in line with previous empirical findings from expertise research (e.g., from programming research, see Bednarik et al., 2005; Nivala et al., 2016; Orlov & Bednarik, 2017). Experts' shorter fixation duration is often seen as an indication for their faster, more efficient, and less effortful information processing (e.g., Nivala et al., 2016; Rayner, 2009).

Second, the hypothesis (H4) that experts would need fewer transitions between code and output when updating the output information was confirmed, which might be an indication that experts have more working memory capacity available, due to chunking mechanisms (cf. Bauhoff et al., 2012; Chase & Simon, 1973). Third, the finding that experts ran the code less frequently than novices (H6) was confirmed, which we connected to experts' forward-reasoning processes and, hence, their less trial-and-error-based problem-solving approach (cf. Katz & Anderson, 1987).

In contrast, the analyses regarding H2, H3, and H5 yielded unexpected results. We expected that experts would process code in a less "linear," approximately linewise, manner (H2), because they would use more top-down driven, script-based code processing approaches (Soloway & Ehrlich, 1984). However, experts showed significantly more linear code processing than novices. A possible explanation might be that for the relatively simple codes that we used, a linear debugging strategy might have been the most adequate method. At the same time, experts' more linear code processing might have also impacted the saccade amplitudes, explaining the findings regarding H3. That is, more linear behavior should automatically cause shorter saccades, because the relative frequency of short saccades from one line to the next is higher. This might have caused an effect opposite to what we expected in H3 that experts would show longer saccade amplitudes than novices in the code area because of their more holistic processing ability (e.g., Kundel, Nodine, Conant, & Weinstein, 2007). Finally, we could not confirm H5 that experts would take longer until first running the code. However, this does not necessarily mean that experts did not first engage in more forward reasoning by first building a mental representation of the problem. Instead, experts' overall faster processing ability (reflected in shorter time on task and fixation durations, H1) might have allowed the experts to build a mental representation of the code faster. In this case, the time until first running the code would not necessarily differ between the two expertise groups anymore.

Aside from the hypothesis-driven investigation of novices' and experts' natural, non-didactic problem-solving behavior, we explored variance differences for each measure between the two groups. Naturally behaving experts showed less variance for the measures fixation duration and code running frequency than novices. For all other measures, no variance differences were found. This observation is in line with previous observations that found that professional programmers often show very similar eye movement patterns

(Rodeghero & McMillan, 2015). In our study, experts' slightly more homogeneous debugging behavior could indicate experts' use of more similar problem-solving schemas that are stored in long-term memory. This finding might be an indication that it is worth displaying expert problem-solving behavior to novices, because it is more univocal compared to that of novices and could therefore be taught to them (cf. Van Meeuwen et al., 2014).

Taken together, our analysis of eye movement measures and mouse clicks allowed us to unravel the naturally occurring differences in experts' and novices' nonverbal behavior during the complex problem-solving task of code debugging. Aside from applying and extending previous expertise theories to our programming and code debugging, knowing how experts' and novices' nonverbal behavior usually differs during debugging is necessary to draw conclusions about the direction in which experts change their behavior when acting didactically (Research Question 2). More specifically, this helps us to answer the question whether experts' nonverbal behavior becomes more similar to novices' behavior or not.

4.2. How do experts change their nonverbal behavior to didactically model a task?

When comparing experts' natural and didactic nonverbal behavior, we found that didactically behaving experts showed longer fixation durations, processed the code more linearly, performed larger saccade amplitudes in the code area, performed more transitions between code and output when running the code, and ran the code less often than naturally behaving experts. No significant effect of instruction was found for the measure "time until first running the code."

Interestingly, concerning the direction of changes, experts became *more similar* to novices when behaving didactically on the measures "average fixation durations" and "amount of transitions per running the code," and *less similar* on the measures "click on the run button" and "linearity." Even though experts' saccade amplitudes became longer when behaving didactically, they still did not differ significantly from those of novices.

The average fixation durations in the code areas were shorter for naturally behaving experts than novices and although they became longer when experts behaved didactically, the difference remained significant. The amount of transitions per running the code was lower for naturally behaving experts than novices but increased for experts when they behaved didactically, to the extent that the difference with novices was no longer significant.

We associated experts' shorter fixation durations with a faster processing ability (Ericsson et al., 1993) and experts' fewer transitions per mouse click on the run button with an increased working memory capacity (Bauhoff et al., 2012). These two measures were hence associated with experts' superior processing abilities and capacities, which might not be easy to imitate by novices. A first interpretation of these results is that experts "slow down" their usual (natural) behavior when behaving didactically. However, the finding that experts also performed fewer fixations (of longer durations) in the code area when behaving didactically (at least numerically, as the difference in number of fixations was not statistically significant) could also indicate a more directed effort to focus longer on fewer, presumably more relevant, parts of the code to guide novices' attention. Future research using qualitative analysis of the processes experts engage in could confirm which of the two (slowing down or behaving in a more directed way) provides a more likely explanation.

Other measures might not reflect superior processing abilities, but instead advantageous, expert-specific strategies that could more easily be controlled and exemplified for novices, such as a linear code processing, or a more trial-and-error-based behavior (a higher code running frequency). For these measures, we found that experts' nonverbal behavior became more different from novices' behavior (i.e., even more linear code processing and even less clicks on the run button). In fact, experts seemed to exaggerate and exemplify their natural, nonverbal behavior, hence increasing the already existing difference between themselves and novices' behavior. It might be that these measures reflect more teachable strategies that can be illustrated and exemplified to novices to teach them. However, to what extent experts deliberately make these behavioral changes when behaving didactically or if the observed changes are involuntary effects of the changes in experts' cognitive processes when behaving didactically requires future investigation.

4.3. Limitations

When interpreting the results regarding experts' changes in nonverbal behavior when modeling a task didactically, two limitations should be kept in mind. First, in contrast to the condition in which the experts behaved naturally, experts also knew the code and its solution when acting didactically. This means that the knowledge about the task might have influenced results regarding experts' didactic behavior. Second, experts always behaved didactically after behaving naturally, which might cause order effects. However, we argue that it is not uncommon that experts know the answer of a problem-solving task when explaining it didactically to novices. Instead, the knowledge of the correct solution could even be needed for experts to be able to create truly didactic videos. We therefore think that our choice of conditions was necessary to evoke authentic natural and didactic behavior. To investigate the impact of order effects, future studies could ask a control group of experts to do the task twice, without acting didactically.

4.4. Educational relevance and future directions

In practice, experts often need to communicate their knowledge to novices. Until now, studies about experts' communication behavior with a less knowledgeable audience have mainly focused on changes in written or spoken verbal communication (e.g., Bromme et al., 2005; Isaacs & Clark, 1987; Jucks et al., 2007). We extend this research by investigating how didactic instructions influence nonverbal expert behavior (here mouse clicks and eye movements). We found that experts showed a substantial change in nonverbal behavior when modeling a task didactically: They would slow down some processes while at the same time exaggerating other strategies. One possible explanation for our findings is that experts become more similar to novices for measures that are affected by experts' automatized, superior processing and less similar to novices for measures that indicate their strategic behavior. However, especially experts' (numerically fewer and) longer fixations in the code area could also indicate a more directed effort to guide novices' attention to relevant code information. Whether experts can control these changes deliberately remains, however, an open question.

The finding that experts change their nonverbal behavior substantially when modeling a task didactically was a first and promising step to understand experts' behavior in a social, educational context. Our results have implications for the research on EMMEs as possible educational tool. De Koning and Jarodzka (2017) provide an overview of possible benefits of using EMMEs for attention guidance over regular screen recording videos. Regular screen recordings, especially in the field of programming education, often make use of the mouse cursor as a tool for attention guidance. In contrast to these videos, EMMEs can provide observers with more fine-graded information about the task performer's problem-solving behavior and can, consequently, support learning (see, e.g. in the field of programming education, see Bednarik, Schulte, Budde, Heinemann, & Vrzakova, 2018). Only a few studies directly compared eye movement and mouse-cursor displays as deictic tools and these studies found comparable effects of the two methods (Gallagher-Mitchell, Simms, & Litchfield, 2018; Velichkovsky, 1995). However, to the best of our knowledge, there is no research that compares learning from tutorial videos (e.g., modeling examples with voice-overs) with mouse and eye movement displays, and an interesting question would be if experts would adapt "mouse pointing" in a similar manner as they seem to adapt their eye movements when behaving didactically.

In the context of EMME research, our present results generally imply that EMMEs of naturally and didactically behaving models will likely differ substantially with—until now—unknown effects on learning. While both types of model instructions have been used to create EMME (and shown to foster performance and learning), they have not yet been compared on their effectiveness for performance and learning. Future studies should therefore investigate effects of model instruction on how well students can follow and learn from EMME. On the one hand, novices might benefit from expert didactic attention guidance in EMMEs, because it is targeted toward the specific audience group and might be easier to understand and follow. On the other hand, observing an expert's natural viewing and problem-solving behavior could provide useful insights into the standards their performance should ultimately meet and stimulate more self-explanation behavior and deep processing. However, the effectiveness of (EMME) videos with different kinds of model instructions might also be dependent on the learners' level of prior knowledge, because learners with less prior knowledge might need more didactic guidance to follow the experts' behavior.

Acknowledgments

This research was funded by a grant from the Netherlands Initiative for Education Research (NRO-PROO # 405-17-301). We would like to thank Carlos Cordoba from the Spyder team on Github.com for helping us to adjust the Python IDE for our purposes.

Notes

1. Based on a task from https://www.w3resource.com/python-exercises/.
2. Mean fixation count (with *SD*) in the code area for all experimental groups: Novices: 1462.50 (329.71); naturally behaving experts: 882.96 (458.79); didactically behaving experts: 577.60 (227.86).

References

Allen, M., Poggiali, D., Whitaker, K., Marshall, T. R., & Kievit, R. A. (2019). Raincloud plots: A multi-platform tool for robust data visualization. *Wellcome Open Research*, *4*, 63. https://doi.org/10.12688/wellcomeopenres.15191.1

Baayen, R. H., Davidson, D. J., & Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, *59*(4), 390–412. https://doi.org/10.1016/j.jml.2007.12.005

Barr, D. J. (2008). Analyzing "visual world" eyetracking data using multilevel logistic regression. *Journal of Memory and Language*, *59*(4), 457–474. https://doi.org/10.1016/j.jml.2007.09.00.

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Lme4: Linear mixed-effects models using Lme4. *Journal of Statistical Software*, *67*, 1–48. https://doi.org/10.18637/jss.v067.i01

Bauhoff, V., Huff, M., & Schwan, S. (2012). Distance matters: Spatial contiguity effects as trade-off between gaze switches and memory load. *Applied Cognitive Psychology*, *26*(6), 863–871. https://doi.org/10.1002/acp.288.

Bednarik, R., Myller, N., Sutinen, E., & Tukiainen, M. (2005). Effects of experience on gaze behavior during program animation. In P. Romero, J. Good, S. Bryant, & E. A. Chaparro (Eds.), *Proceedings of the 17th Workshop of the Psychology of Programming Interest Group (PPIG)* (pp. 49–61). Sussex, UK: University of Sussex.

Bednarik, R., Schulte, C., Budde, L., Heinemann, B., & Vrzakova, H. (2018). Eye-Movement Modeling Examples in source code comprehension: A classroom study. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research* (pp. 22–25). New York: ACM. https://doi.org/10.1145/3279720.3279722

Bednarik, R., & Tukiainen, M. (2005). Effects of display blurring on the behavior of novices and experts during program debugging. In *CHI'05 Extended abstracts on human factors in computing systems* (pp. 1204–1207). https://doi.org/10.1145/1056808.1056877

Boshuizen, H. P., & Schmidt, H. G. (2008). The development of clinical reasoning expertise. In J. Higgs, M. Jones, S. Loftus, & N. Christensen (Eds.), *Clinical reasoning in the health professions* (3rd ed., pp. 113–121). Oxford, UK: Butterworth-Heinemann.

Bromme, R., Jucks, R., & Runde, A. (2005). Barriers and biases in computer-mediated expert-laypersoncommunication. In R. Bromme, F. W. Hesse, & H. Spada (Eds.), *Barriers, biases and opportunities of communication and cooperation with computers- and how they may be overcome* (pp. 89–118). New York: Springer. https://doi.org/10.1007/0-387-24319-4_5

Busjahn, T., Bednarik, R., Begel, A., Crosby, M., Paterson, J. H., Schulte, C., & Tamm, S. (2015). Eye movements in code reading: Relaxing the linear order. In *2015 IEEE 23rd International Conference on Program Comprehension* (pp. 255–265). New York: IEEE Press Piscataway. https://doi.org/10.1109/icpc.2015.36

Busjahn, T., Schulte, C., Sharif, B., Begel, A., Hansen, M., Bednarik, R., Orlov, P., Ihantola, P., Shchekotova, G., & Antropova, M. (2014). Eye tracking in computing education. In *Proceedings of the Tenth Annual Conference on International Computing Education Research* (pp. 3–10). New York, NY: ACM. https://doi.org/10.1145/2632320.2632344

Charness, N., Reingold, E. M., Pomplun, M., & Stampe, D. M. (2001). The perceptual aspect of skilled performance in chess: Evidence from eye movements. *Memory & Cognition*, *29*(8), 1146–1152. https://doi.org/10.3758/bf03206384

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology*, *4*(1), 55–81. https://doi.org/10.1016/0010-0285(73)90004-2

Clark, H. H., & Murphy, G. L. (1982). Audience design in meaning and reference. *Language and Comprehension*, *9*, 287–299. https://doi.org/10.1016/s0166-4115(09)60059-5

Clark, H. H., Schreuder, R., & Buttrick, S. (1983). Common ground at the understanding of demonstrative reference. *Journal of Verbal Learning and Verbal Behavior*, *22*(2), 245–258. https://doi.org/10.1016/s0022-5371(83)90189-5

Clarke, A. D. F., Mahon, A., Irvine, A., & Hunt, A. R. (2016). People are unable to recognize or report on their own eye movements. *The Quarterly Journal of Experimental Psychology*, *70*(11), 2251–2270. https://doi.org/10.1080/17470218.2016.1231208

De Koning, B. B. & Jarodzka, H. (2017). Attention guidance strategies for supporting learning from dynamic visualizations. In R. Lowe & R. Ploetzner (Eds.), *Learning from dynamic visualizations: Innovations in research and practice* (255–278). Cham, Switzerland: Springer.

Ericsson, K. A., Hoffman, R. R., Kozbelt, A., & Williams, A. M. (Eds.). (2018). *The Cambridge handbook of expertise and expert performance*. Cambridge, UK: Cambridge University Press. https://doi.org/10.1017/9781316480748

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the cquisition of expert performance. *Psychological Review*, *100*(3), 363–406. https://doi.org/10.1037/0033-295x.100.3.363

Ericsson, K. A., & Simon, H. A. (1980). Verbal reports as data. *Psychological Review*, *87*(3), 215–251. https://doi.org/10.1037/0033-295x.87.3.215

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, *18*(2), 93–116. https://doi.org/10.1080/08993400802114508

Foulsham, T., & Kingstone, A. (2013). Where have eye been? Observers can recognize their own fixations. *Perception*, *42*(10), 1085–1089. https://doi.org/10.1068/p7562

Gallagher-Mitchell, T., Simms, V., & Litchfield, D. (2018). Learning from where "eye" remotely look or point: Impact on number line estimation error in adults. *Quarterly Journal of Experimental Psychology*, *71*(7), 1526–1534. https://doi.org/10.1080/17470218.2017.1335335

Gould, J. D. (1975). Some psychological evidence on how people debug computer programs. *International Journal of Man-Machine Studies*, *7*(2), 151–182. https://doi.org/10.1016/s0020-7373(75)80005-8

Hinds, P. J. (1999). The curse of expertise: The effects of expertise and debiasing methods on prediction of novice performance. *Journal of Experimental Psychology: Applied*, *5*(2), 205–221. https://doi.org/10.1037/1076-898x.5.2.205

Hinds, P. J., Patterson, M., & Pfeffer, J. (2001). Bothered by abstraction: The effect of expertise on knowledge transfer and subsequent novice performance. *Journal of Applied Psychology*, *86*(6), 1232–1243. https://doi.org/10.1037/0021-9010.86.6.1232

Hooge, I. T. C., & Erkelens, C. J. (1998). Adjustment of fixation duration during visual search. *Vision Research*, *38*(9), 1295–1302. https://doi.org/10.1016/S0042-6989(97)00287-3

Isaacs, E. A., & Clark, H. H. (1987). References in conversation between experts and novices. *Journal of Experimental Psychology: General*, *116*(1), 26–37. https://doi.org/10.1037/0096-3445.116.1.26

Jarodzka, H., Balslev, T., Holmqvist, K., Nyström, M., Scheiter, K., Gerjets, P., & Eika, B. (2012). Conveying clinical reasoning based on visual observation via eye-movement modelling examples. *Instructional Science*, *40*(5), 813–827. https://doi.org/10.1007/s11251-012-9218-5

Jarodzka, H., Van Gog, T., Dorr, M., Scheiter, K., & Gerjets, P. (2013). Learning to see: Guiding students' attention via a model's eye movements fosters learning. *Learning and Instruction*, *25*, 62–70. https://doi.org/10.1016/j.learninstruc.2012.11.004

Johnson, W., Soloway, E., Cutler, B., & Draper, S. (1983). Bug catalogue: I (Technical Report. Vol. No. 286).New Haven, CT: Department of Computer Science, Yale University.

Jucks, R., Schulte-Löbbert, P., & Bromme, R. (2007). Supporting experts' written knowledge communication through reflective prompts on the use of specialist concepts. *Journal of Psychology*, *215*(4), 237–247. https://doi.org/10.1027/0044-3409.215.4.237

Just, M. A., & Carpenter, P. A. (1980). A theory of reading: From eye fixations to comprehension. *Psychological Review*, *87*(4), 329–354. https://doi.org/10.1037/0033-295x.87.4.329

Katz, I., & Anderson, J. (1987). Debugging: An analysis of bug-location strategies. *Human-Computer Interaction*, *3*(4), 351–399. https://doi.org/10.1207/s15327051hci0304_2

Kok, E. M., Aizenman, A. M., Võ, M. L. H., & Wolfe, J. M. (2017). Even if I showed you where you looked, remembering where you just looked is hard. *Journal of Vision*, *17*(12), 1–11. https://doi.org/10.1167/17.12.2

Krebs, M.-C., Schüler, A., & Scheiter, K. (2019). Just follow my eyes: The influence of model-observer similarity on Eye Movement Modeling Examples. *Learning and Instruction*, *61*, 126–137. https://doi.org/10.1016/j.learninstruc.2018.10.005

Kundel, H. L., Nodine, C. F., Conant, E. F., & Weinstein, S. P. (2007). Holistic component of image perception in mammogram interpretation: Gaze-tracking study. *Radiology*, *242*(2), 396–402. https://doi.org/10.1148/radiol.2422051997

Lin, Y. T., Wu, C. C., Hou, T. Y., Lin, Y. C., Yang, F. Y., & Chang, C. H. (2015). Tracking students' cognitive processes during program debugging—An eye-movement approach. *IEEE Transactions on Education*, *59*(3), 175–186. https://doi.org/10.1109/te.2015.2487341

Litchfield, D., & Ball, L. J. (2011). Rapid communication: Using another's gaze as an explicit aid to insight problem solving. *Quarterly Journal of Experimental Psychology*, *64*(4), 649–656. https://doi.org/10.1080/17470218.2011.558628

Litchfield, D., Ball, L. J., Donovan, T., Manning, D. J., & Crawford, T. (2010). Viewing another person's eye movements improves identification of pulmonary nodules in chest x-ray inspection. *Journal of Experimental Psychology: Applied*, *16*(3), 251. https://doi.org/10.1037/a0020082

Mason, L., Pluchino, P., & Tornatora, M. C. (2015). Eye-movement modeling of integrative reading of an illustrated text: Effects on processing and learning. *Contemporary Educational Psychology*, *41*, 172–187. https://doi.org/10.1016/j.cedpsych.2015.01.004

Mason, L., Pluchino, P., & Tornatora, M. C. (2015). Using eye-tracking technology as an indirect instruction tool to improve text and picture processing and learning. *British Journal of Educational Technology*, *47* (6), 1083–1095. https://doi.org/10.1111/bjet.12271

Mason, L., Scheiter, K., & Tornatora, M. C. (2017). Using eye movements to model the sequence of text-picture processing for multimedia comprehension. *Journal of Computer Assisted Learning*, *33*(5), 443–460. https://doi.org/10.1111/jcal.12191

Nakagawa, S., & Schielzeth, H. (2012). A general and simple method for obtaining R^2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution*, *4*(2), 133–142. https://doi.org/10.1111/j.2041-210x.2012.00261.x

Nalanagula, D., Greenstein, J. S., & Gramopadhye, A. K. (2006). Evaluation of the effect of feedforward training displays of search strategy on visual search performance. *International Journal of Industrial Ergonomics*, *36*(4), 289–300. https://doi.org/10.1016/j.ergon.2005.11.008

Nanja, M., & Cook, C. R. (1987). An analysis of the on-line debugging process. In G. M. Olsen, S. Sheppard, & E. Soloway (Eds.), *Empirical Studies of Programmers: Second Workshop* (pp. 172–184). Norwood, NJ: Ablex.

Nivala, M., Hauser, F., Mottok, J., & Gruber, H. (2016). Developing visual expertise in software engineering: An eye tracking study. In *2016 IEEE Global Engineering Education Conference (EDUCON)* (pp. 613–620). IEEE. https://doi.org/10.1109/educon.2016.7474614

Nückles, M., Winter, A., Wittwer, J., Herbert, M., & Hübner, S. (2006). How do experts adapt their explanations to a layperson's knowledge in asynchronous communication? An experimental study. *User Modeling and User-Adapted Interaction*, *16*(2), 87–127. https://doi.org/10.1007/s11257-006-9000-y

Obaidellah, U., Al Haek, M., & Cheng, P.-C.-H. (2018). A survey on the usage of eye-tracking in computer programming. *ACM Computing Surveys*, *51*(1), 1–58. https://doi.org/10.1145/3145904

Orlov, P. A., & Bednarik, R. (2017). The role of extrafoveal vision in source code comprehension. *Perception*, *46*(5), 541–565. https://doi.org/10.1177/0301006616675629

Persky, A. M., & Robinson, J. D. (2017). Moving from novice to expertise and its implications for instruction. *American Journal of Pharmaceutical Education*, *81*(9), 72–80. https://doi.org/10.5688/ajpe6065

R Core Team. (2015). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Available at: http://www.R-project.org/. Accessed March 6, 2019.

Rayner, K. (1978). Eye movements in reading and information processing. *Psychological Bulletin*, *85*(3), 618–660. https://doi.org/10.1037/0033-2909.85.3.618

Rayner, K. (2009). Eye movements and attention in reading, scene perception, and visual search. *The Quarterly Journal of Experimental Psychology*, *62*(8), 1457–1506. https://doi.org/10.1080/17470210902816461

Reingold, E. M., Charness, N., Pomplun, M., & Stampe, D. M. (2001). Visual span in expert chess players: Evidence from eye movements. *Psychological Science*, *12*(1), 48–55. https://doi.org/10.1111/1467-9280.00309

Reingold, E. M. & Sheridan, H. (2011). Eye movements and visual expertise in chess and medicine. In S. P. Liversedge, I. D. Gilchrist, & S. Everling (Eds.). *Oxford handbook on eye movements* (pp. 767–786). Oxford, UK: Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199539789.013.0029

Rodeghero, P., & McMillan, C. (2015). An empirical study on the patterns of eye movement during summarization tasks. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). https://doi.org/10.1109/esem.2015.7321188

Samuels, S. J., & Flor, R. F. (1997). The importance of automaticity for developing expertise in reading. *Reading & Writing Quarterly*, *13*(2), 107–121. https://doi.org/10.1080/1057356970130202

Scheiter, K., Schubert, C., & Schüler, A. (2017). Self-regulated learning from illustrated text: Eye movement modelling to support use and regulation of cognitive processes during learning from multimedia. *British Journal of Educational Psychology*, *88*(1), 80–94. https://doi.org/10.1111/bjep.12175

Seppänen, M., & Gegenfurtner, A. (2012). Seeing through a teacher's eyes improves students' imaging interpretation. *Medical Education*, *46*(11), 1113–1114. https://doi.org/10.1111/medu.12041

Sharif, B., Falcone, M., & Maletic, J. I. (2012). An eye-tracking study on the role of scan time in finding source code defects. In S. N. Spencer (Ed.). *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA'12)* (pp. 381–384). New York: ACM. https://doi.org/10.1145/2168556.2168642

Simon, H. A. (1975). The functional equivalence of problem solving skills. *Cognitive Psychology*, *7*(2), 268–288. https://doi.org/10.1016/0010-0285(75)90012-2

Soh, Z., Sharafi, Z., Van den Plas, B., Porras, G. C., Guéhéneuc, Y. G., & Antoniol, G. (2012). Professional status and expertise for UML class diagram comprehension: An empirical study. In *2012 20th IEEE International Conference on Program Comprehension (ICPC)* (pp. 163–172). IEEE. https://doi.org/10.1109/ICPC.2012.6240484

Soloway, E., & Ehrlich, K. (1984). Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering*, *SE10*, 595–609. https://doi.org/10.1109/tse.1984.5010283

Stein, R., & Brennan, S. E. (2004). Another person's eye gaze as a cue in solving programming problems. In *Proceedings, ICMI 2004, Sixth International Conference on Multimodal Interfaces* (pp. 9–15). New York: ACM. https://doi.org/10.1145/1027933.1027936

Uwano, H., Nakamura, M., Monden, A., & Matsumoto, K. I. (2006). Analyzing individual performance of source code review using reviewers' eye movement. In S. N. Spencer (Ed.). *Proceedings of the 2006 symposium on eye tracking research & applications* (pp. 133–140). New York: ACM. https://doi.org/10.1145/1117309.1117357

Van Gog, T., Jarodzka, H., Scheiter, K., Gerjets, P., & Paas, F. (2009). Attention guidance during example study via the model's eye movements. *Computers in Human Behavior*, *25*(3), 785–791. https://doi.org/10.1016/j.chb.2009.02.007

Van Marlen, T., Van Wermeskerken, M., Jarodzka, H., & Van Gog, T. (2016). Showing a model's eye movements in examples does not improve learning of problem-solving tasks. *Computers in Human Behavior*, *65*, 448–459. https://doi.org/10.1016/j.chb.2016.08.041

Van Marlen, T., Van Wermeskerken, M., Jarodzka, H., & Van Gog, T. (2018). Effectiveness of eye movement modeling examples in problem solving: The role of verbal ambiguity and prior knowledge. *Learning and Instruction*, *58*, 274–283. https://doi.org/10.1016/j.learninstruc.2018.07.005

Van Meeuwen, L. W., Jarodzka, H., Brand-Gruwel, S., Kirschner, P. A., de Bock, J. J. P. R., & Van Merriënboer, J. J. G. (2014). Identification of effective visual problem solving strategies in a complex visual domain. *Learning and Instruction*, *32*, 10–21. https://doi.org/10.1016/j.learninstruc.2014.01.004

Van Wermeskerken, M., Litchfield, D., & Van Gog, T. (2018). What am I looking at? Interpreting dynamic and static gaze displays. *Cognitive Science*, *42*(1), 220–252. https://doi.org/10.1111/cogs.12484

Velichkovsky, B. M. (1995). Communicating attention: Gaze position transfer in cooperative problem solving. *Pragmatics & Cognition*, *3*(2), 199–223. https://doi.org/10.1075/pc.3.2.02vel

Võ, M. L. H., Aizenman, A. M., & Wolfe, J. M. (2016). You think you know where you looked? You better look again. *Journal of Experimental Psychology: Human Perception and Performance*, *42*(10), 1477–1481. https://doi.org/10.1037/xhp000026

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article:

Data S1. Supplementary materials.