Programmers' experiences with working in the restricted-view mode as indications of parafoveal processing differences.

Pavel A. Orlov
1) School of Computing
University of Eastern Finland
2) Engineering Graphics and Design Dept.
Peter the Great Saint-Petersburg
Polytechnic University
paul.a.orlov@gmail.com

Roman Bednarik
School of Computing
University of Eastern Finland
roman.bednarik@uef.fi

Liudmila Orlova
Emperor Alexander I St. Petersburg
State Transport University
mila.e.orlova@gmail.com

Abstract
Understanding of programmers' attention provides benefits for developing comprehension models, bug-prediction models, for increasing software productivity, and facilitating programming education activities. Here we conduct a gaze-contingent study involving a real-time restriction of the viewing area and compare professionals' and novices' verbal feedback after the parafoveal view was restricted during source-code comprehension. Such information provides clues about the differences in parafoveal processing during programming.

We recorded the participants' verbal feedback and divided their answers into seven topics and types. Then we analysed the differences between the answers given by the experts and the novices. We compared the proportion of utterances used to comment upon a certain topic by each group. This allowed to identify the relative importance of a particular topic. Some topics turned out equally important for both the experts and the novices: 1. analysis of the working process, 2. personal evaluation of the source code, and 3. the use of life-hacks. The experts and the novices used a different proportion of utterances to comment upon the experimental conditions and visualizing, which was unexpected. The restriction of the extrafoveal area evoked a more emotional response from the expert programmers. At the same time, the novices perceived the restriction of semantic information in the extrafoveal area in a less emotional way. We suggest that the explanation is that the experts have increased expectations for the information to be obtained from the attentional objects located in the extrafoveal area. The masking of parafoveal objects makes them less easily available, which runs counter to the experts' expectations and, therefore, produces a stronger emotional feedback. This illustrates that it is a common thing for expert programmers to use the parafoveal information during source code comprehension.

1. Introduction
1.1. Background
The program-code comprehension is a high-level cognitive task. At the same time, it involves a number of low-level processes – e.g., perception, attention and memory (Hoc, Green, Samurcay, & Gilmore, 1990). One of the areas of the research of programmers behavior is advancing the understanding of the links between the low-level and high-level processing, such as the links between eye-movements and program comprehension.

The role of attention has been greatly investigated in programming. Programmers attend to various elements of the source code in different order than in reading, forming strategies of task solving (Crosby & Stelovsky, 1990). The understanding of programmers' behaviour as an attentional strategy provides benefits in a number of ways: developing bug-prediction models, enhancing teaching and learning, and increasing software productivity (Fritz, Begel, Müller, Yigit-elliott, & Züger, 2014; Busjahn et al., 2014; von Mayrhauser & Vans, 1997). Hence, the importance of the research into the process of attention allocation during source-code comprehension arises.

The common way to identify programmers' attention is to link it with visual attention that can be measured by gaze-fixation patterns (Bednarik, 2007). Human eyes are permanently moving. Visual in-

formation is mostly processed during fixation phase, when eye-movements are very slow (from 0 to 40 grad./sec.) (Yarbus, 1965). During fixations, vision samples the maximum clarity of information from the foveal area, which makes up 2 degrees of the visual field (Bridgeman, Van der Heijden, & Velichkovsky, 1994). Then fixations are terminated by the ballistic eye-movements that are called saccades (Yarbus, 1965). After a saccade, a new image stabilizes on the retina for the next fixation. The eye-tracking studies of source-code comprehension rest on an assumption that the foveal area plays a pivotal role in localizing attention, thus linking the direction of gaze and the focus of mind (Reichle, 2006; Bednarik, 2007).

Evidence from the domains of reading and visual studies indeed suggests that the foveal area is among the most important factors of visual attention, but not the only one (Rayner, White, Kambe, Miller, & Liversedge, 2003; Engbert & Kliegl, 2003). Visual information is obtained from an operational area that exceeds the foveal area (Gippenreyter, 1978). For instance, the perceptual span in the reading of natural-language left-to-right texts achieves up to 14–15 letters to the right of the fixation point, which extends beyond the 2 degrees area of fovea (Rayner et al., 2003; Angele et al., 2015). The operational area is utilized to plan the next fixation during a process called saccadic planning. Therefore, human attention shifts from the foveal object during fixation (Rayner, Castelhano, & Yang, 2009).

The area of source-code that can be captured by a single fixation correlates with the increase of the useful operational area. Neurophysiological studies show that visual attention can be focused on an area that is different from the orientation of gaze position (Engbert & Kliegl, 2003). There is also an evidence that attention switching could be indicated by micro-saccades (Engbert & Kliegl, 2003; Engbert, 2006; Martinez-Conde, Macknik, Troncoso, & Hubel, 2009). The micro-saccades are small involuntary saccades that take place during fixation (Engbert & Kliegl, 2003). There are still debates how this knowledge can be implemented in such a complex process as source-code comprehension (P. A. Orlov, 2015). In any case, the question about the role of extrafoveal information processing during source-code comprehension remains open.

If we limit the size of the useful perceptual area to fovea only and quantify how the programmer's behaviour changes compared to unrestricted vision, we may be able to understand the role of the parafoveal information. The studies of visual attention in a variety of other professional domains such as car driving, sport, and chess playing, show that experts have a wider perception span; see meta-analysis by Gegenfurtner and colleges (Gegenfurtner, Lehtinen, & Säljö, 2011) for more details. In the domain of computer programming, it was previously shown that partial restriction of the extrafoveal area influences programmers' performance (Bednarik & Tukiainen, 2005).

The emotional response of the expert programmers to working in the restricted-view mode was negative, while the novices perceived it in a neutral way (Bednarik & Tukiainen, 2007). Authors conducted informal interviews with subjects about the restricted viewing conditions after a debugging session. They provide only the summary of programmers feedbacks and one extreme situation where expert programmer declined to participate. Authors reported how the expert explained his decision: "I do not want to work with that" (Bednarik & Tukiainen, 2007). In this present study, we analyze the *working process* as well, however, our work differs from the previous in a number of ways: we conduct a detailed analysis of the post-comprehension self-reports on the working process, and we employ a real gaze-contingent environment, as opposed to the mouse-driven precomputed tool employed in the previous research.

1.2. Goals and aims

The motivation of our study is to uncover the underlying processing related to the utilization of the extrafoveal area during source-code comprehension. We focus on the personal experience accounts of the participants, in terms of information interpretation and affect, as proxies to understand programmers' behaviour.

In this work we evaluate how programmers experience the restriction of the extrafoveal area, how they feel in the restricted conditions, and what topics they provide feedback about. If experts' and novices'

experiences under the restricted visual conditions differ, we hypothesize, they utilize the extrafoveal area differently.

In sum, the goal of the study is to report and analyze how experts and novices reflect upon their working process when the extrafoveal area is restricted. To that end, we defined the following research questions:

- RQ1. Experts and novices reported on their source-code comprehension experiences in the restricted-view mode. What topics can be singled out in the experts' and the novices' reports?
- RQ2. What are the differences between the experts' and the novices' reports on working in the restricted-view mode?

2. Method

2.1. Design and Procedure

We interviewed the programmers about the nature of the experiment and their feelings about working in the given experimental conditions (Hoc et al., 1990). We employed methods from the qualitative research to obtain insights into the behaviour of expert and novice programmers (Biggerstaff, 2012). Textual analysis was based on self-reports collected from the interviews that followed the experiment. Self-report measurements are the most common methods used to ascertain subjects' affective state (Brave & Nass, 2003; Robinson & Clore, 2002; Salah, Hung, Aran, & Gunes, 2013). Discourse analysis was used to understand how the participants organise their thinking about their experience of source-code comprehension in the restricted-view mode (Starks & Trinidad, 2007).

2.2. Materials

We prepared five short programs in Java. Each program was compilable and runnable. The result of each program was a value shown in the output console. The first program consisted of two class definitions and definitions of two methods. A method was calling from a class object and the result of the formula contained was printed to the output. A similar logic, but with a *for*-loop was used in the second and third programs: objects were created in a loop and were collected into an array. Then in the forth and fifth programs we used a loop nested within a loop. An example listing is provided in Figure 1.

The graph-theoretic (McCabe) complexity was used to prepare programs with different complexity of understanding. The complexity of the programs corresponded to different McCabe complexity levels: 1, 1.33, 1.5, 1.66, and 2 (McCabe, 1976). The calculation was done using the Metric framework plugin for Eclipse (http://metrics.sourceforge.net/). The linguistic aspects, like names of variables, were also individual for each source code. The stimuli were presented randomly in the restricted-view mode.

```
class MyPointSM{

  public int cx, cy;

  MyPointSM(int x, int y){
    cx = x;
    cy = y;
  }

  public int getDist(int x, int y){
    int dist = 2 * (cy + y) + cx * x;
    return dist;
  }

}

public class Index{

  public static void main(String[] args){
    MyPointSM mObj = new MyPointSM(4, 1);
    System.out.println(mObj.getDist(2, 2));
  }
}
```

Figure 1 – An example of Java program used in the experiment.

The subjects, as per instructions, were to calculate the result of the program. The instructions were: 'Please, read the program and answer: What will be displayed in the console when the program is

finished? You can skip the current source code (press Skip button), but, please, use this option only as a last resort'. The original instruction was in the Russian language: here we present the translation. The subjects were to remain silent when solving the task. When the answers were ready, the subjects pressed the 'Answer' button to supply the answer in a new screen. At the beginning, each subject underwent training with five tasks to become familiar with the gaze-contingent environment and the experimental conditions. There were no time limits for either the warm-up trials or the test itself.

2.3. Participants and Apparatus

The experiment involved 13 male participants: 6 experts (aged 19-25) and 7 novices (aged 18-21). All subjects had normal or corrected-to-normal vision, according to their own report. All six experts had at least two years of professional experience as programmers in a software development company as a Senior Developer. All were graduates from different universities with a technical background. They were familiar with at least three programming languages. The main programming languages for them were object oriented languages such as C++, Java or PHP. Subjects' company projects corresponded with game development for Android platforms and back-end side web development.

The novices were students of the local polytechnic university. The novices did not learn Java at a professional level. They passed a university-level exam on Processing programming knowledge and obtained an average grade of 3.5 (M: 3.5, SD: 0.46, the maximum of the scale is 5). Participation in the experiment was voluntary. Their participation was not compensated and was based only on the subjects enthusiasm.

ScreenMasker, a gaze-contingent tool, was used to allow real-time restriction of the extrafoveal area (P. A. Orlov & Bednarik, 2015). We built a patterned texture picture to mask the extra-foveal area as semi-transparent (see Figure 2 A). The transparent window had a round shape and was 200px in size. The window's size corresponds to the 5 deg. viewing angle. Figure 2 B shows the transparency of the stencil: the area 0-2 deg. is fully transparent; the next 2-5 deg. are transparent by gradient; the area exceeding 5 deg. is fully restricted (Calvo & Lang, 2005). The area that was in clear view changed dynamically and it was contingent on the direction of gaze in real time. Thereby, we implemented the window-moving paradigm – that is, the subject could clearly see only through the stencil placed on the screen in the point of gaze-fixation position, like through the window (McConkie & Rayner, 1975).

The size of the window was determined from the physiological limit of the 2 degrees angle of the foveal vision. The viewing window size was smaller than the window presenting the code on a screen. To solve the task subjects should comprehend the whole program. Programmers had to point attention onto different source code elements and understand the coordination of methods calling and variable definitions.

After the tasks, we asked the subjects to describe their feelings and experience during the experiment in a free form, verbally. All self-reports were collected by a researcher in writing.

Eye movements were registered with an SMI RED250 eye-tracker. The stimuli were presented used a desktop PC with Intel(R) Core(TM) 2 Duo CPU E8400 @ 3.00G Hz, 3.25GB memory, NVIDIA GeForce 8800GT, running the Microsoft Windows 7 operating system. The graphics system contained High-performance monitor BENQ XL2411 with 24" screen size, 53cm x 30cm,1920 x 1080px with 144Hz refresh rate. Direct latency measurement of ScreenMasker environment was done with a Go-Pro HERO 3+ Black Edition camera (240fps with 848x480px screen size in WVGA mode), and an established measurement procedure (P. Orlov & Bednarik, 2014). The mean latency of the system was approx. 25-28 ms, well under the recommended limit of 80ms (Loschky & McConkie, 2005).

2.4. Analysis of Self-reports

Analysis of program summaries is often used for the insights into subject's mental models of computer programs (Good, 1999; Good & Brna, 2003; Hughes, Buckley, Exton, & O'Carroll, 2005). One open question for this type of approach is related to the issue of replicability (Byckling, Kuittinen, Nevalainen, & Sajaniemi, 2004). To solve this problem Good suggested an analysis scheme to probe the comprehen-

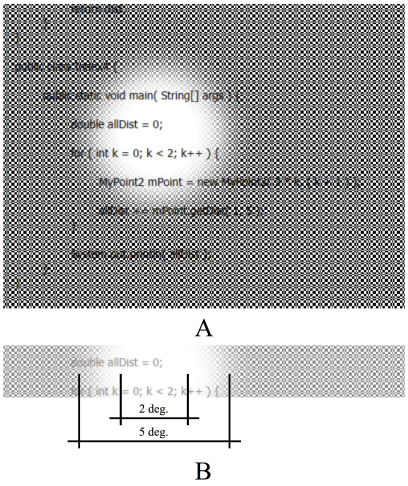

Figure 2 – The sample of the stimuli displayed using ScreenMasker. A: the sample of the masking stencil applied to restrict the visual zone to 5 deg. The text outside the stencil was not readable. B: dimensions of the 5 deg. stencil

sion of computer programming languages (Good, 1999).

The Good's scheme is based on two types of classifications of summaries: information types classifier and object description categories. Nevertheless, both classifications focus on the utterances that subject used and they miss *how* the utterances were used, and, the previous work does not analyze the emotional state of subjects. That is why we generate topics of discussion from the content of the self-reports and we did not use a-priori categories of analysis.

After the experiment sessions, the third author sampled the programmers' self-reports by the topics that they reported about. We analysed the self-reports to break them down by categories of topics: what the subjects were talking about, what was interesting for them, and what seemed important to them. For the analysis, we took the ratio of utterances in each category to the total number of utterances by subject.

The utterances were placed into *Analysis of the working process* category if a subject reflected on his strategy of task solving and patterns he used, for example: *'I tried to summarize constant values in advance to use one instead of several when calculating the formula'*.

The category *Personal evaluation of the program* includes utterances used by a subject to express an opinion about the program, for instance: *'Who produced this inefficient source code here?'*

Evaluation of the experimental conditions category: here the subjects reported on the gaze-contingent mode of working. For example, one subject said: *'It was cool to control the spot by gaze'*.

Viewing conditions category: here the subjects reported on their feelings about the view being restricted, the system's lag, or the psychological pressure created by the environment. For example, one expert reported: *'I had psychological pressure due to the restricted-view mode. It was distracting me'*.

For the *Emotional state* category we collected all the utterances corresponding to the latent emotional condition of a programmer. For example, one expert reported: *'I was tired at the end of the experiment'*.

The *Life-hacks* category contained the utterances that described any additional methods of solving the task. Life-hacks could include, for example, repetition of values in a low voice in order not to forget them; using fingers for calculations and for 'additional memory'.

We determined six topics and furthermore, we singled out the category of emotionally charged lexicon – e.g., abusive language. As a result, we obtained seven categories from the content of the self-reports, which are shown in Table 1.

Table 1 – Topics of utterance with explanations

Topic of utterance	Explanation
Analysis of the working process	Subject's reflection on his strategy of task solving and patterns he used
Viewing conditions	Subject's feelings about the view being restricted, the system's lag, or the psychological pressure created by the environment
Personal evaluation of the program	Subject's expressions and opinions about the program
Emotional state	All the utterances corresponding to the latent emotional condition of a programmer
Evaluation of the experimental conditions	Gaze-contingent mode of working
Life-hacks	Additional methods of solving the task
Emotionally-charged language	Abusive language used by the subjects in their reports

Then third and first authors independently coded self-report's utterances into these categories from every subject. As the reliability results indicated good inter-observer agreement (Kappa = .781).

3. Results

Novice developers skipped 8% trials, while expert participants did not skip any task. As we expected, experts performed better than novices. On average, experts solved 4 from 5 tasks correctly (M:4, SD: 0.71) when novices solved 1.33 (M:1.33, SD:2.23).

The mean completion time was about 150 seconds per task. We calculated the total number of words; the experts were more vocal than the novices. On average, their self-reports contained 150 words (SD: 92.04), while the novices' ones contained only 67 words (SD: 17.15). A two sample t-test shows that the means differ (t = 2.361, df = 6.483, p = .053).

3.1. Analysis of the working process.

A two sample t-test shows that there is no differences between the means of utterances percentage between Novices and Experts (t = .921, p = 0.383). On average, the programmers spent 50% of the utterances in their feedback to discuss the working process (Experts: M: 56.37, SD: 17.03. Novices: M: 45.18, SD: 25.25).

3.2. Personal evaluation of the program

There were no differences in means of utterance percentage (t = .298, p = 0.771). On average, the programmers spent 9% of their utterances to give their personal evaluation of the program (Experts: M: 9.997, SD: 13.744. Novices: M: 7.81, SD: 12.709).

3.3. Evaluation of the experimental conditions

The experts spent about 5% of their utterances to discuss this topic (M: 5.42, SD: 10.24). Novices reported nothing here: not a single novice commented upon the experimental conditions in his feedback.

3.4. Viewing conditions

A two sample t-test shows that there is no differences in means of utterances percentage (t = -1.755, p = .123). 12% of utterances used by the experts (SD = 8.21) and 25% of those used by the novices (SD = 17.07) can be attributed to this category. We did not separate utterances on positive or negative here.

3.5. Emotional state

A two sample t-test shows (t = -0.993, p = .358) that there is no differences in means of utterances percentage (Experts: M: 4.982, SD: 9.121. Novices: M: 15.205, SD: 23.759).

3.6. Emotionally-charged language

We found that on average 1.5% of the utterances used by the experts fall into this category. Typically, these are charged with negative emotions directed at the gaze restricted environment. The novices did not use abusive language at all.

3.7. Life-hacks

We found that both groups of subjects talk about their life-hacks in a similar way, Two Sample t-test shows (t = .305, p = .766) that there is no differences in means of utterances percentage (Experts: M: 4.787, SD: 6.198. Novices: M: 3.824, SD: 5.168).

Figure 3 shows a summary of the utterances' percentage used by reported topics. Subjects from both groups reported more about their working process. We can see that professional developers reported less about they emotional state directly and about viewing conditions.

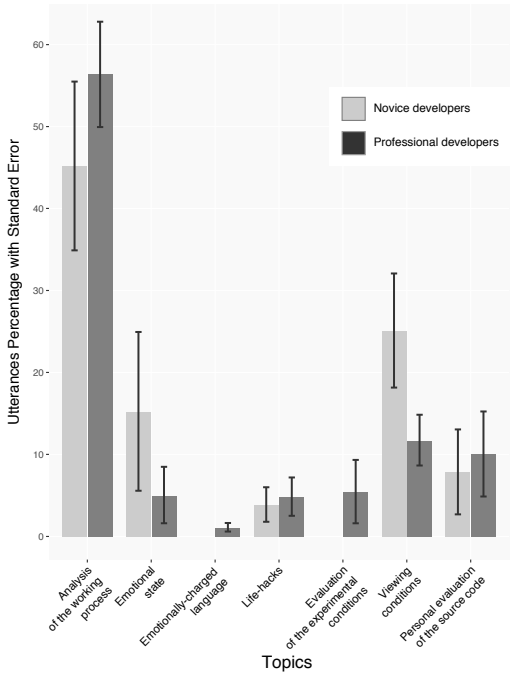

Figure 3 – A summary of utterances percentage for topics by experts and novices.

4. Discussion

The goal of the current study was to discover how professionals and novices report on working in the restricted-view mode. The experimental conditions did not allow programmers to process restricted extrafoveal objects even when they paid attention to them. Even though restricting parafoveal information is not practical for everyday programming, it is one of the methods to gain important understanding of the role of extrafoveal processing. The outcome of these investigations is to obtain implications on the development of future IDEs. For example, if we learned that programmers employed the extrafoveal area intensively during a certain stage of program development, this areas should be distinctively presented in the user interface of development environment to respond with its heightened role. Design of programming languages can be informed by the way programmers interact with the various structures of the source code. An additional motivation is educational: new knowledge about the expert programmers' attention can be used for developing novel teaching methods for students. The attentional process is latent from the direct measurement systems. More over, programmers can direct their visual attention at the extrafoveal objects that are at different locations that the direction of overt visual attention focus.

The analysis of self-reports allowed to single out six categories of topics and one emotional charged state. The professional programmers were found to produce more voluble reports compared to the novices. This was expected and accords with the professionals' familiarity with professional conversations about source code at their workplaces (RQ1).

One of the interesting findings is that both groups of programmers spent about 50% of utterances in their reports to comment upon the working process – in particular, to describe the steps of task solving.

We expected a different outcome, with the strategies description yielding a larger proportion of the professionals' words, which would be a typical scenario (Koenemann & Robertson, 1991; Aschwanden & Crosby, 2006; Bednarik, 2012). However, the professionals decided to describe the experimental conditions instead. They reported on how they dealt with source code in a different way. Conversely, the novices reported nothing on the experimental conditions (RQ2).

The programs was evaluated by both groups using a similar number of utterances. The professionals discussed possible optimisation, while the novices were speaking about syntax recollection. For example, professionals suggests some optimisations like keeping all functionality in a one place and not dividing it into several methods or loops. We can defined it as a higher level of understanding. Novices focus more at the low level of understanding: they talk about general syntax and variables naming.

While discussing the visual aspects, both groups reported the restricted-view mode as disturbing for them. Bednarik and Tukiainen (2007) showed that novices were not disturbed much by the restriction of view. This differs from the findings presented here. The inconsistency may be due to the different experimental conditions: the previous study used a mouse-controlled window-moving tool that should be more invasive, while the present work applied a real-time gaze-contingent tool.

The description of emotional state and life-hacks took the same share of utterances in both groups. But the professionals used more emotionally loaded utterances, including: '*It was awesome; It was tough; It distracted me quite a deal; I am psychologically crushed*'. No one expert expressed affinity with working under restricted-viewing condition. Moreover the negative emotionally-charged language was found only in some reports by the experts (RQ2). These results are in agreement with the previous findings that showed the professionals to universally dislike the restriction of viewing conditions (Bednarik & Tukiainen, 2007).

We propose that the increased emotions indicate that expert programmers in fact heavily and routinely rely on the use of parafoveal information. Visual information is obtained from the extrafoveal area especially when human attention shifts from the foveal object during a fixation (Rayner et al., 2009). Humans emotionally react when they expects to attend some object and do not manage to access it easily (Calvo & Lang, 2005). Because experts are generally better aware of the relevance of the information attended, their expectations of the attentional objects are higher then for novices. They are also better at prediction of the next object of attention before switching the gaze position on it. When that object is masked and the information about it not easily accessible, the increased emotional feedback is both likely because of the mismatch with expectations and also an evidence of the importance of extrafoveal information.

The results have implications in educational purposes and on the development of future IDEs, for example if programmers use the extrafoveal area intensively this area should be better presented and more easily accessible in the user interfaces of the development environments.

As a limitation of the experimental setup, we are aware that our results may be due to a different explanation: novices felt more restricted to express themselves because they were students and were affiliated with the university. We minimized such kind of risk in our experiment conditions: in our experiment subjects were free to express their self as the like without any restriction; subjects were instructed that their results of task solving do not influence their study process or diploma defence in future.

Green (1977) showed that upstream calculation is more beneficial for the experimental conditions then downstream evaluation of the program that was used here. Upstream evaluation is ecologically valid, being a common task in program development and debugging (T. R. G. Green, 1977; T. R. Green, Petre, & Bellamy, 1991). A further study with more focus on experiment design is therefore suggested.

5. Conclusion

In our study we check the hypothesis that if professionals use extrafovel area of vision they also self-reports about programming or the comprehension in different way. We show that experts not only talk more, but that the quality of their self reports was different.

This paper argues that professionals express themselves in a more negative emotional and nervous way. We conclude that it is the unavailable extrafoveal objects of the source code that evoked negative emotional feedback from the professionals. This study found that generally both groups report on the working process, life-hacks, and source code features in the same proportion. This study raised important questions about the nature of programmers' reflection about the restricted-view mode. If the debate is to be moved forward, a better understanding of the role of extrafoveal information processing needs to be developed by means of the quantitative methods of eye-movement analysis.

6. References

Angele, B., Schotter, E. R., Slattery, T. J., Tenenbaum, T. L., Bicknell, K., & Rayner, K. (2015). Do successor effects in reading reflect lexical parafoveal processing? Evidence from corpus-based and experimental eye movement data. *Journal of Memory and Language*, *79-80*, 76–96.

Aschwanden, C., & Crosby, M. (2006). Code Scanning Patterns in Program Comprehension. In *Proceedings of the 39th hawaii international conference on system sciences.*

Bednarik, R. (2007). *Methods to Analyze Visual Attention Strategies: Applications in the Studies of Programming* (Unpublished doctoral dissertation). University of Joensuu, Joensuu.

Bednarik, R. (2012, feb). Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human-Computer Studies*, *70*(2), 143–155.

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 - chi '05* (pp. 1204–1207). New York, New York, USA: ACM Press.

Bednarik, R., & Tukiainen, M. (2007, may). Validating the Restricted Focus Viewer: a study using eye-movement tracking. *Behavior research methods*, *39*(2), 274–282.

Biggerstaff, D. (2012). *Qualitative Research Methods in Psychology*. InTech.

Brave, S., & Nass, C. (2003). Emotion in human–computer interaction. *Human-Computer Interaction*, 53–68.

Bridgeman, B., Van der Heijden, A. H. C., & Velichkovsky, B. M. (1994, feb). A theory of visual stability across saccadic eye movements. *Behavioral and Brain Sciences*, *17*(02), 247–258.

Busjahn, T., Schulte, C., Sharif, B., Simon, Begel, A., Hansen, M., ... Antropova, M. (2014). Eye Tracking in Computing Education Categories and Subject Descriptors. In *Icer '14 proceedings of the tenth annual conference on international computing education research* (pp. 3–10). Glasgow, Scotland, United Kingdom: ACM New York, NY, USA ©2014.

Byckling, P., Kuittinen, M., Nevalainen, S., & Sajaniemi, J. (2004). An Inter-Rater Reliability Analysis of Good's Program Summary Analysis Scheme. *Ppig '04*(April), 170–184.

Calvo, M. G., & Lang, P. J. (2005). Parafoveal semantic processing of emotional visual scenes. *Journal of experimental psychology. Human perception and performance*, *31*(3), 502–519.

Crosby, M. E., & Stelovsky, J. (1990). How Do We Read Algorithms ? A Case Study. *Computer*, *23*(1), 24–35.

Engbert, R. (2006, jan). Microsaccades: A microcosm for research on oculomotor control, attention, and visual perception. *Progress in brain research*, *154*(June 2005), 177–92.

Engbert, R., & Kliegl, R. (2003). Microsaccades uncover the orientation of covert attention. *Vision Research*, *43*(9), 1035–1045.

Fritz, T., Begel, A., Müller, S. C., Yigit-elliott, S., & Züger, M. (2014). Using Psycho-Physiological Measures to Assess Task Difficulty in Software Development Categories and Subject Descriptors. In *Proceedings of the 36th international conference on software engineering* (pp. 402–413).

Gegenfurtner, A., Lehtinen, E., & Säljö, R. (2011). Expertise Differences in the Comprehension of Visualizations: A Meta-Analysis of Eye-Tracking Research in Professional Domains. *Educational Psychology Review*, *23*(4), 523–552.

Gippenreyter, Y. (1978). *Movements of the human eye [Dvijenia chelovecheskogo glaza]*. Moscow: Moscow State University.

Good, J. (1999). *Programming Paradigms, Information Types and Graphical Representations: Empirical Investigations of Novice Program Comprehension.* (Unpublished doctoral dissertation). The University of Edinburgh.

Good, J., & Brna, P. (2003). Toward Authentic Measures of Program Comprehension. *Ppig '03*(April), 29–50.

Green, T. R., Petre, M., & Bellamy, R. (1991). Comprehensibility of visual and textual programs: A test of superlativism against the'match-mismatch'conjecture. *Empirical Studies of Programmers*, *91*(743), 121–146.

Green, T. R. G. (1977). Conditional program statements and their comprehensibility to professional programmers. *Journal of Occupational Psychology*, *50*(2), 93–109.

Hoc, J.-M., Green, T., Samurcay, R., & Gilmore, D. (1990). *Psychology of Programming*. London, England: Academic Press.

Hughes, C., Buckley, J., Exton, C., & O'Carroll, D. (2005). Towards a Framework for Characterising Concurrent Comprehension. *Computer Science Education*, *15*(1), 7–24.

Koenemann, J., & Robertson, S. (1991). Expert problem solving strategies for program comprehension. *Conference on Human factors In computing systems (CHI)*, 125–130.

Loschky, L. C., & McConkie, G. W. (2005). How late can you update? Detecting blur and transients in gaze-contingent multi-resolutional displays. In *Proceedings of the human factors and ergonomics society 49th annual meeting.* (pp. 1527–1530).

Martinez-Conde, S., Macknik, S. L., Troncoso, X. G., & Hubel, D. H. (2009). Microsaccades: a neurophysiological analysis. *Trends in Neurosciences*, *32*(9), 463–475.

McCabe, T. J. (1976). A Complexity Maesure. *IEEE Transactions of software engineering*, *SE-2*(4), 308–320.

McConkie, G. W., & Rayner, K. (1975). The span of the effective stimulus during a fixation in reading. *Perception & Psychophysics*, *17*(6), 578–586.

Orlov, P., & Bednarik, R. (2014). Low-cost latency measurement system for eye-mouse software. *Proceedings of the 8th Nordic Conference on Human-Computer Interaction Fun, Fast, Foundational - NordiCHI '14*, 1085–1088.

Orlov, P. A. (2015). Experts vs Novices in programming:" Who knows where to look? *Eye Movements in Programming: Models to Data*, 16–19.

Orlov, P. A., & Bednarik, R. (2015). ScreenMasker: An Open-source Gaze-contingent Screen Masking Environment. *Behavior Research Methods*.

Rayner, K., Castelhano, M. S., & Yang, J. (2009, sep). Eye movements and the perceptual span in older and younger readers. *Psychology and aging*, *24*(3), 755–60.

Rayner, K., White, S. J., Kambe, G., Miller, B., & Liversedge, S. P. (2003). On the processing of meaning from parafoveal vision during eye fixations in reading. *The mind's eye: Cognitive and applied aspects of eye movement research*, 213–234.

Reichle, E. D. (2006). Editorial: Computational models of eye-movement control during reading: Theories of the "eye-mind" link. *Cognitive Systems Research*, *7*(1), 2–3.

Robinson, M. D., & Clore, G. L. (2002). Belief and feeling: evidence for an accessibility model of emotional self-report. *Psychological bulletin*, *128*(6), 934–960.

Salah, A. A., Hung, H., Aran, O., & Gunes, H. (2013). Creative applications of human behavior understanding. In *Human behavior understanding* (pp. 1–14). Springer.

Starks, H., & Trinidad, S. B. (2007). Choose your method: a comparison of phenomenology, discourse analysis, and grounded theory. *Qualitative health research*, *17*(10), 1372–1380.

von Mayrhauser, A., & Vans, A. M. (1997). Program understanding behavior during debugging of large scale software. *Papers presented at the seventh workshop on Empirical studies of programmers - ESP '97*, 157–179.

Yarbus, A. L. (1965). *Eye movements and vision [Rol dvijenyi glaz v procecce zrenia].* Moscow: "Nauka".