2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) | 978-1-6654-3786-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SANER53432.2022.00092

Towards a Fine-grained Analysis of Cognitive Load During Program Comprehension

Thierry Sorg, Amine Abbad-Andaloussi, Barbara Weber
Institute of Computer Science
University of St. Gallen
St. Gallen, Switzerland
[thierry.sorg, amine.abbad-andaloussi, barbara.weber]@unisg.ch

***Abstract*—Program comprehension is inherent to all software development activities. This task may require a high mental effort (or so-called "cognitive load"), which in turn can hinder the performance of developers. In the literature, several authors have investigated the ability of biosignals to estimate developers' cognitive load during program comprehension. While the majority of these studies provide estimates at the task level, we aim for a more fine-grained level of analysis allowing to pinpoint the critical parts of code that could be associated with cognitive load. We infer these critical parts solely from eye fixation features and investigate qualitatively their relationship with those perceived as challenging by users. Being able to pinpoint critical parts in the source-code, is a first stride towards a very handy approach providing targeted support to developers to prevent them from committing errors. Furthermore, such a lightweight approach can be adapted in online settings.**

***Index Terms*—Program comprehension, Eye-tracking, Cognitive load, Fine-grained analysis**

I. Introduction

Program comprehension is a fundamental task in software development. Well known development's activities like code review and debugging require attention and effort from developers [1], [2]. A better understanding of the difficulties they encounter during these tasks can help bringing them support.

Enhancing program comprehension has motivated several studies from early theory [3] to recent research [1], [2], [4]. If the use of bio measures in [5] allowed to predict difficulties in program comprehension at the task-level, the current trend in the literature is moving towards a more fine-grained analysis, aiming at identifying exactly where and when developers are challenged [6]–[9]. In the same vein, we propose a lightweight and non-intrusive approach denoting the critical Part(s) of the Code (PoCs) based on eye fixations (i.e., the periods the eyes remain still at a specific position of the stimulus [10]). Moreover, we investigate the relationship between the inferred critical PoCs and the ones highlighted by users as challenging. Our research questions can be formulated as follow: **RQ1.** *How can we infer the critical PoCs potentially related to cognitive load from eye-tracking features?* and **RQ2.** *What is the relationship between critical and highlighted PoCs?*

The Second author is supported by the International Postdoctoral Fellowship (IPF) Grant (Number: 1031574) from the University of St. Gallen, Switzerland

Our exploratory study is inspired by the experiment described in Fritz et al. [5]. We extended it for a fine-grained analysis by asking participants to highlight the parts they perceived as challenging in the code and by linking our eye-tracking measurements to the source code. These PoCs were compared with the critical PoCs inferred solely from fixation measures (i.e., fixation duration and count). The mapping between the participants' highlights and critical PoCs gave promising results. Overall, our contributions consist of **(1)** an approach to infer critical PoCs relating to user's cognitive load during program comprehension tasks; and **(2)** a set of visual insights suggesting that the PoCs perceived as challenging are related to those proposed by our approach.

By being able to pinpoint the PoCs susceptible to require mental processing, our approach can be seen as a first stride towards a very handy approach targeted to support developers in their working tasks. The remaining of this paper is structured as follows. Sect. II presents the background and related work. Sect. III describes the method used to collect and analyze the data. Sect. IV reports our findings. Sect. V opens a discussion on the findings of this work, its threats to validity and its potential impact. Sect. VI concludes the paper.

II. Background and Related Work

This section provides a background covering the key concepts discussed in this paper. Sect. II-A introduces the cognitive load theory and measurements. Sect. II-B presents the eye-mind hypothesis and motivates the use of fixation-based measures as indicators of cognitive load. Finally, Sect. II-C provides an overview of the studies investigating developers' cognitive load at a fine-grained level.

A. Cognitive Load: Theory and Measurements

The cognitive load theory (CLT) addresses the characteristics of the human working memory, which can be defined as a buffer that stores information temporally while they are processed or integrated with other information. CLT defines cognitive load (CL) as *"a multi-dimensional construct representing the load imposed on the working memory during* [the] *performance of a cognitive task"* [11]. CLT posits that the working memory has a limited capacity and can only hold and process a fixed number of items at a time [12]. When this

978-1-6654-3786-8/22/$31.00 ©2022 IEEE
DOI 10.1109/SANER53432.2022.00092

Authorized licensed use limited to: University of St. Gallen. Downloaded on July 28,2022 at 16:42:08 UTC from IEEE Xplore. Restrictions apply.

limit is approached, humans experience *cognitive overload*, which can hinder their performance and cause errors [11].

CL can be estimated using subjective, performance, behavioral and physiological measures [11]. Subjective measures (e.g., NASA-TLX [13]) capture humans' perception of CL. Performance measures (e.g., response time, answer correctness [11]) assess the extent to which CL affects the accomplishment of a given task. Behavioral measures (e.g., fixation duration, fixation count [10]) estimate CL from humans' voluntary responses to difficult tasks. Physiological measures (e.g., pupil size, electroencephalography EEG, for an overview see [14]) focus on involuntary changes in humans' physiology due to variations in their CL. While subjective and performance measures are typically collected post-hoc, behavioral and physiological measures can provide continuous measurements of CL [11]. In this paper, we focus on the behavioral measures of CL, i.e., fixation duration and count.

B. Eye-Mind hypothesis and fixation characteristics

The eye-mind hypothesis [15] postulates that the mind processes what the eyes are fixating. Accordingly, in the literature, long and recurrent fixations, measured with fixation duration and fixation count, have been correlated with memory load and processing effort [10], [16]. These measures have been also used in a number of studies investigating visual effort during program comprehension (see [16]).

Previous literature has classified fixations into different types, distinguishing the process of shallow information scanning and screening from deep mental processing. While the former is effortless and requires few mental resources, the latter is rather intensive and requires high mental resources. Glöckner and Herbold [17] reported that shallow processing manifests in fixations with a duration of less than 250 ms, while deep processing tends to happen during higher fixation durations (above 500 ms). Grounded in the eye-mind hypothesis [15] and supported by the findings of the aforementioned studies [10], [16], [17], we rely on fixation features to estimate developers' CL and investigate the potentially critical PoCs following the approach we propose in Sect. III.

C. Estimating cognitive load at a fine-grained level

Given their sensitivity to variations of CL within the task [11], continuous measures (e.g., behavioral and physiological) are needed for a fine-grained analysis of developers' CL. Behavioral measures have been widely used in the literature to investigate the difficulty experienced by developers when engaging with source-code (overview in [16], [18]). Since most of these studies rely on fixed stimuli (e.g. in form of images), their approach is not adequate for tracking developers' gazes directly on source-code artifacts. This requirement is important for a fine-grained analysis especially if it should apply to large pieces of software. iTrace was proposed, in this vein, to automate the mapping between fixations and the corresponding lines and columns in the source-code [19]. The tool was, for instance, used in [8] where the authors investigated the duration of fixations on identifiers with and without lexical inconsistencies. In their analysis, fixation duration served to compare pre-defined PoCs (i.e., identifiers) designed for this purpose. In contrast, in our work fixation features are used to discover critical PoCs. Moreover, behavioral measures were collected at a fine-grained level in [4]. However, their interpretation served to predict the overall task difficulty rather than pinpointing difficult parts of the code.

Physiological measures have been used for the fine-grained analyses of cognitive load in several recent works (overview in [14]). Couceiro et al. [6] proposed an approach allowing to spot the critical lines of code based on gaze data. The authors used a spatial and temporal clustering of gaze points and claimed that the clusters with a high density correspond to the areas where developers might have been challenged. Plots of heart rate variability (HRV) and pupillography were used to support the claim qualitatively. Alternatively, our approach is based on fixations (rather than gazes), which are filtered at a specific threshold (cf. Sect. III-B). Our focus on fixations is grounded in the eye-mind hypothesis [15] and supported by Glöckner's and Herbold's [17] distinction between shallow information scanning and deep mental processing (cf. Sect. II-B). Hijazi et al. [7] used pupillography, heart rate variability (HRV) and eye-tracking to infer the critical code fragments, while Fakhoury et al. [8] investigated whether near-infrared spectroscopy (fNIRS) and eye-tracking can infer the critical parts in the code. Both studies asked participants to highlight the parts that have challenged their comprehension of the source-code after completing a set of tasks. The obtained highlights were used as a ground truth against which the robustness of CL models and measures is compared. Similarly, Medeiros et al. [9] investigated the ability of EEG measures and eye-tracking to differentiate the critical parts of the source-code. The approaches in [8] and [9] rely on intrusive measures whose application can be challenging in workplaces. As for the approach in [7], it includes HRV features and thus requires a minimum window length which goes beyond what is required for linking PoCs with CL at a fine-grained level. For example, in [20], the authors have shown that with a window length of less than 1 minute, the HRV analysis of all parameters is affected negatively. Moreover, the approaches described in [6], [7], [9] have not linked the physiological data with the source code and can thus not be used for large software systems where developers navigate between different source code files and make use of scrolling. In turn, this paper aims to provide a lightweight approach for the fine-grained analysis of CL solely based on eye-tracking linked to the source code artifact.

III. Method

This section describes our method to collect and analyze the data. Sect. III-A presents the data collection procedure, while Sect. III-B explains the key steps of the analysis.

A. Data collection

1) Acquisition protocol: We designed our experiment with the same stimuli (code snippets) as Fritz et al. [5]. The protocol is divided into 8 randomly presented blocks containing a

Authorized licensed use limited to: University of St. Gallen. Downloaded on July 28,2022 at 16:42:08 UTC from IEEE Xplore. Restrictions apply.

baseline, a task, and a set of questionnaires. The baseline consists of a two minutes fish tank video to watch and is directly followed by the task during which the participant has to read a source code snippet written in C# and respond to a related question. Afterwards, and as an extension of [5], the participant is asked to rate the difficulty and then to highlight the challenging parts of the given snippet.

2) Participants: Ten subjects registered for the experiment. Two did not pass the calibration process. The remaining 8 participants are 3 PhD students (2 with a Computer Science background and one with an Electrical Engineering background), 4 postdoctoral researchers (with a Computer Science background) and 1 software engineer. The participants had different levels of proficiency in programming. On a 4-points scale (i.e., none, novice, core, pro/master), 1 participant was *novice*, while 7 participants had a sufficient level for *core* programming. Regardless of their proficiency levels, all participants knew object-oriented and procedural languages and were exposed to at least Python, Java or C# during the last 6 months. We collected data from 8 participants on 8 different tasks for a total of 64 samples. Two participants had to be excluded since they have not been correctly recorded. Thus, the analysis has covered 48 samples.

3) Instrumentation: We used a Tobii X3-120 eye-tracker mounted on a laptop with a 1920x1080 screen to capture gaze data and to forward it to two applications: iMotions (Version 8.2[1]) and iTrace [19]. iMotions runs the experiment protocol and is used to derive fixations from the gaze data. iTrace is used to convert the participant's gaze data into line and column positions in the IDE (here, Eclipse[2]), providing a linkage between the gaze data and the source code.

4) Preprocessing: In this part, we synchronized the data recorded by iMotions and iTrace using a common identifier appended to gaze points before being forwarded to both applications. As a result, we derived a new dataset where each gaze point (obtained from iMotions) was linked with the gazed line and column in the source-code (obtained from iTrace). Following that, we used the IV-T algorithm [10] to group gazes into fixations. Then, we computed the centroid of the gazes within each fixation to derive the corresponding line and column of code that was fixated by the participant.

B. Analysis approach

To identify the critical PoCs (i.e., RQ1), we filter the fixations in our dataset, keeping only those with a duration above 250 ms, which we call "long fixations" (following the insights in [17], cf. Sect. II-B). Then, we (1) look at the cumulation of long fixations over the different lines of code as it gives insights on the relevance of the different lines using the statistical distribution of the time spent on them. Also, we (2) search for long fixations that are close enough (in space) to form clusters, showing that a PoC has received a higher count of fixations of long duration. Based on the identification of the lines with high cumulative fixation duration and the presence of clusters of long fixations we infer the critical PoCs.

To study the relationship between the inferred critical PoCs and those highlighted by the participants (i.e., RQ2), we look for a potential alignment between both. Then, if the alignment is not found, we investigate whether there are other relationships between the highlighted PoCs and the non-aligned critical PoCs. We do this by looking at (1) the evolution of fixations over time, which provides indications on participants' visual patterns, and (2) the dependencies in the source-code.

IV. Findings

This section reports our findings. We present representative results to support our explanations, while a listing of all results for each participant is available in an online appendix.[3] Sect. IV-A presents the insights on the identification of critical PoC (addressing RQ1). Sect. IV-B investigates the relationship between highlighted and critical PoCs (addressing RQ2).

A. Inferring critical PoCs

The data represented in Fig. 1 corresponds to one participant (ID 10.05) performing one task (b_11). Fig. 1a depicts a histogram showing the cumulation of long fixations on each line of code, while Fig. 1b depicts a plot showing the lines and columns fixated on the source-code. Fig. 1a shows that Lines 16, 15 and 10 are those with high cumulative fixation duration. In Fig. 1b, the same lines expose clusters of long fixations. Overall, RQ1 can be answered positively, i.e., using clusters of long fixations it is possible to infer critical PoCs.

B. Relationship between highlighted and critical PoCs

In Sect. IV-A, a set of critical PoCs have been identified. From Fig. 1, one can notice that the critical PoC corresponding to Line 16 (cf. Fig. 1b) aligns with the PoC highlighted (in red) by the participant (cf. Fig. 1c). For this task, this observation holds also when considering the data of all participants (cf. Fig. 2), which, in turn, supports the suggested alignment.

The fixations positions plots depicted in Fig. 1b and Fig. 2b show other clusters of long fixations (of different densities) referring to critical PoCs that have not been highlighted by the participants. To investigate this observation, we first looked at the evolution of fixations over time, plotted against the line of code axis (for Participant 10.05, Task b_11) as shown in Fig. 1d. In this figure, one can notice back and forth transitions between the highlighted PoC and the other critical PoCs. In the literature, transitions between AOIs (Areas of Interest or PoCs in our context) were denoted as visual associations and signs of cognitive integration [21], which itself, requires mental effort and thus could influence users' CL [21], [22].

Visual associations could be also observed when looking at the evolution of fixations over time for all participants (whose fixations are labeled with different colors) while conducting Tasks b_11 and b_6 as shown in Figs. 2d and 3d respectively. Looking at the corresponding fixation plots (c.f., Fig. 2b and 3b), one can see that some critical PoCs have not been

[1]See https://imotions.com/release/

[2]See https://www.eclipse.org

[3]See https://doi.org/10.5281/zenodo.5837711

Authorized licensed use limited to: University of St. Gallen. Downloaded on July 28,2022 at 16:42:08 UTC from IEEE Xplore. Restrictions apply.

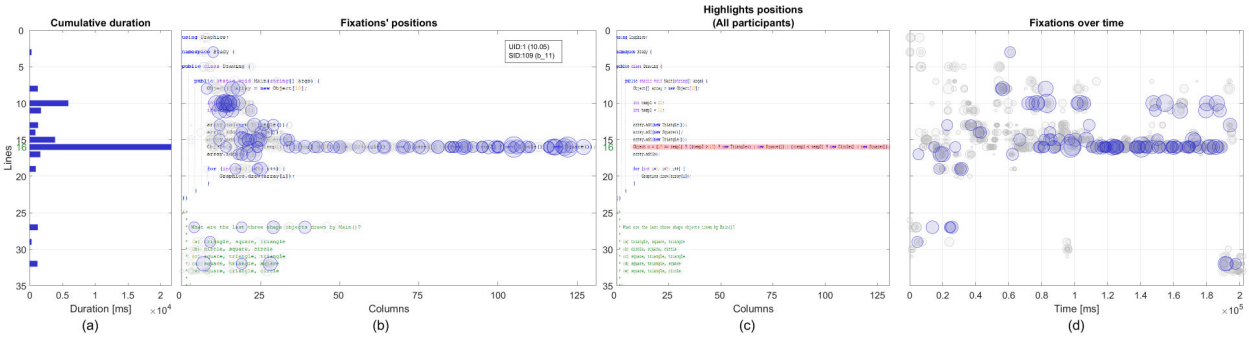


Fig. 1: Plots of fixations and highlights (see respective titles) for Participant 10.05 conducting Task b_11. In (a) only long fixations (above 250ms) are considered. In (b) and (d), short fixations (below 250ms) are marked in gray while long fixations are marked in blue.

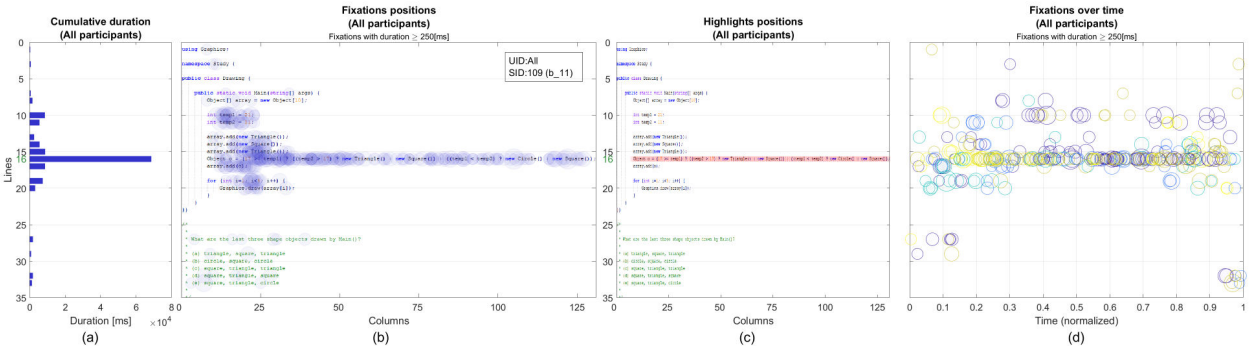


Fig. 2: Plots of fixations and highlights (see respective titles) for all participants conducting Task b_11. In (d) the colors refer to different participants. Only long fixations are shown in these plots.

highlighted by the participants (c.f., Fig. 2c and 3c). However, these critical PoCs share dependencies with the highlighted PoCs. A similar remark can be made at a more fine-grained level. In Fig. 3c, Line 23 was one of the most highlighted lines by participants. 3a shows, in contrast, that Line 24 has the highest cumulative fixation duration. Because of the dependencies shared by these lines, it seems plausible that the long fixations in Line 24 are associated with cognitive integration. Recalling RQ2, the results either expose the relationship between critical and highlighted PoCs as a direct mapping or explain it through dependencies in the code.

V. Discussion

The findings reported in Sect. IV suggest a relationship between the critical and highlighted PoCs, either as a direct alignment between the two types of PoCs or an indirect link due to source-code dependencies (cf. Sect. IV-B).

As mentioned in Sect. II-C, highlights of challenging PoCs were used in others studies as well. Fakhoury et al. [8] investigated the relationship between highlighted PoCs and those inferred as mentally demanding based on an fNIRS measure and fixation coordinates. Similar to our findings, the authors found several PoCs that have not been highlighted but were suggested by the fNIRS measure as being mentally demanding. Cognitive integration (underlying the visual associations between co-dependent PoCs [21]) might explain the cognitive load associated with the critical PoCs that have not been highlighted (as seen in our study and in [8]). In the literature, cognitive integration has been identified as a complexity factor in problem solving tasks [22] and a potential indicator of users' performance [21]. As this factor is likely to be present in code comprehension tasks as well, it is crucial for our fine-grained analysis approach to incorporate it. This extension is envisaged as part of our future work (cf. Sect. VI).

Our study is subject to a number of threats. In particular, our sample size is relatively small, the snippets adopted from [5] do not cover a large set of complexity patterns and our insights are rather exploratory. Another threat is associated with the interpretation of critical PoCs. With our approach, we can identify critical PoCs, i.e., areas that received many long fixations. This, however, does not necessarily imply that these PoCs will contain errors, since a relationship between CL and errors holds in particular when the level of CL is very high and reaches the overload region [11]. These threats will be addressed in our future work (cf. Sect. VI).

Given we succeed in discerning the problematic PoCs contributing to high CL, we can then provide indications about the fragments requiring more attention during code review tasks. Moreover, this could form the basis for designing novel intervention mechanisms that provide an online and context-adaptive support to developers when high CL is detected.

VI. Conclusion and Future Work

This paper presents an approach to infer the critical PoCs

Authorized licensed use limited to: University of St. Gallen. Downloaded on July 28,2022 at 16:42:08 UTC from IEEE Xplore. Restrictions apply.

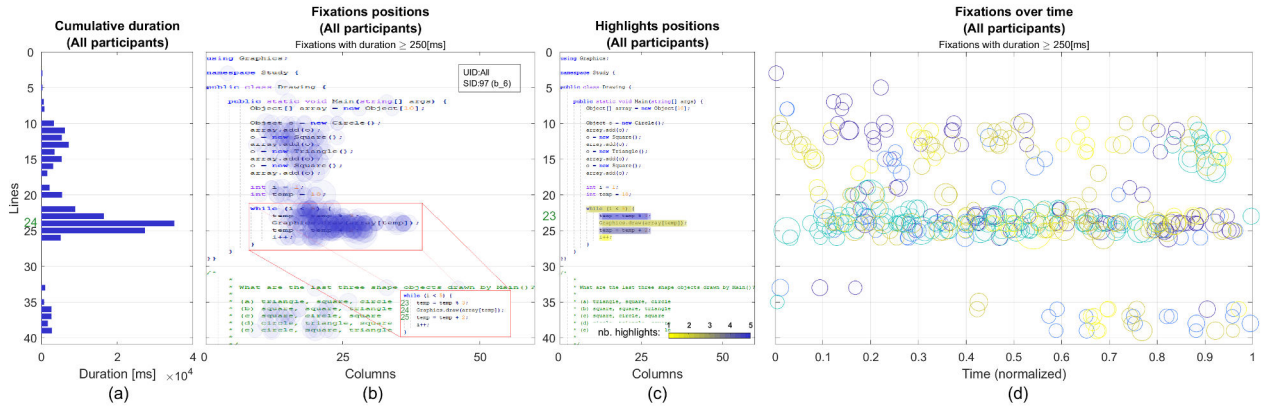


Fig. 3: Plots of fixations and highlights (see respective titles) for all participants conducting Task b_6. In (d) the colors refer to different participants. Only long fixations are shown in these plots.

based on temporal and spacial analyses of fixations. The findings of our exploratory study suggest a relationship between the identified critical PoCs and those perceived as being challenging by the participants. As future work, we are planning to enlarge our sample size, consider longer and more complex source-code and verify our findings quantitatively. Moreover, we are planning to extend our approach by incorporating metrics, indicating cognitive integration between PoCs (e.g., AOI run count [21]). Furthermore, we are envisaging to delve further into the analysis of critical PoCs and identify the ones contributing to high CL. Therein, an event-based analysis will be conducted to inspect physiological data concurring each long fixation and derive features indicating a potential increase in CL. Following these contributions, we are planning to investigate the support of our approach in code review tasks. Finally, we are studying the feasibility of moving our approach to online settings, allowing to provide a context-adaptive support at runtime.

REFERENCES

[1] R. Minelli, A. Mocci, and M. Lanza, "I know what you did last summer-an investigation of how developers spend their time," in *IEEE 23rd International Conference on Program Comprehension*, 2015, pp. 25–35.

[2] L. Gonçales, K. Farias, B. da Silva, and J. Fessler, "Measuring the cognitive load of software developers: A systematic mapping study," in *IEEE/ACM 27th International Conference on Program Comprehension*, 2019, pp. 42–52.

[3] R. Brooks, "Towards a theory of the comprehension of computer programs," no. 6, pp. 543–554.

[4] K. Kevic, B. M. Walters, T. R. Shaffer, B. Sharif, D. C. Shepherd, and T. Fritz, "Eye gaze and interaction contexts for change tasks–observations and potential," *Journal of Systems and Software*, vol. 128, pp. 252–266, 2017.

[5] T. Fritz, A. Begel, S. C. Müller, S. Yigit-Elliott, and M. Züger, "Using psycho-physiological measures to assess task difficulty in software development," in *Proceedings of the 36th international conference on software engineering*, 2014, pp. 402–413.

[6] R. Couceiro, R. Barbosa, J. Durães, G. Duarte, J. Castelhano, C. Duarte, C. Teixeira, N. Laranjeiro, J. Medeiros, P. Carvalho *et al.*, "Spotting problematic code lines using nonintrusive programmers' biofeedback," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2019, pp. 93–103.

[7] H. Hijazi, R. Couceiro, J. Castelhano, P. De Carvalho, M. Castelo-Branco, and H. Madeira, "Intelligent biofeedback augmented content comprehension (tellback)," *IEEE Access*, vol. 9, pp. 28 393–28 406, 2021.

[8] S. Fakhoury, D. Roy, Y. Ma, V. Arnaoudova, and O. Adesope, "Measuring the impact of lexical and structural inconsistencies on developers' cognitive load during bug localization," *Empirical Software Engineering*, vol. 25, no. 3, pp. 2140–2178, 2020.

[9] J. Medeiros, R. Couceiro, G. Duarte, J. Durães, J. Castelhano, C. Duarte, M. Castelo-Branco, H. Madeira, P. de Carvalho, and C. Teixeira, "Can eeg be adopted as a neuroscience reference for assessing software programmers' cognitive load?" *Sensors*, vol. 21, no. 7, p. 2338, 2021.

[10] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. van de Weijer, *Eye Tracking: A comprehensive guide to methods and measures*. OUP Oxford, 2011.

[11] F. Chen, J. Zhou, Y. Wang, K. Yu, S. Z. Arshad, A. Khawaji, and D. Conway, *Robust multimodal cognitive load measurement*. Springer, 2016.

[12] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[13] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*. Elsevier, 1988, vol. 52, pp. 139–183.

[14] B. Weber, T. Fischer, and R. Riedl, "Brain and autonomic nervous system activity measurement in software engineering: A systematic literature review," *Journal of Systems and Software*, vol. 178, 2021.

[15] M. A. Just and P. A. Carpenter, "A theory of reading: From eye fixations to comprehension." *Psychological review*, vol. 87, no. 4, p. 329, 1980.

[16] U. Obaidellah, M. Al Haek, and P. C.-H. Cheng, "A survey on the usage of eye-tracking in computer programming," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–58, 2018.

[17] A. Glöckner and A.-K. Herbold, "Information processing in decisions under risk: Evidence for compensatory strategies based on automatic processes," *MPI collective goods preprint*, no. 2008/42, 2008.

[18] Z. Sharafi, Z. Soh, and Y.-G. Guéhéneuc, "A systematic literature review on the usage of eye-tracking in software engineering," *Information and Software Technology*, vol. 67, pp. 79–107, 2015.

[19] D. T. Guarnera, C. A. Bryant, A. Mishra, J. I. Maletic, and B. Sharif, "itrace: Eye tracking infrastructure for development environments," in *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, 2018, pp. 1–3.

[20] N. Bourdillon, L. Schmitt, S. Yazdani, J.-M. Vesin, and G. P. Millet, "Minimal window duration for accurate hrv recording in athletes," *Frontiers in neuroscience*, vol. 11, p. 456, 2017.

[21] P. Bera, P. Soffer, and J. Parsons, "Using eye tracking to expose cognitive processes in understanding conceptual models," *MIS Quarterly*, vol. 43, no. 4, pp. 1105–1126, 2019.

[22] S. Zugal, "Applying Cognitive Psychology for Improving the Creation, Understanding and Maintenance of Business Process Models," Ph.D. dissertation, University of Innsbruck, 2013.

Authorized licensed use limited to: University of St. Gallen. Downloaded on July 28,2022 at 16:42:08 UTC from IEEE Xplore. Restrictions apply.