[redacted]

Seminar Advanced Software Engineering
Response Paper 2

The first paper “Using (Bio)Metrics to Predict Code Quality Online” [1] investigated whether changing patterns in measured biometrics of a developer can indicate the possibility of quality concerns in cods produced during this time. They were able to successfully train a Classifier with this biometric data to identify possible quality concerns which were later identified as such during a code review. The second paper “Recognizing Developers’ Emotion while Programming” [2] researched if the emotions a developer experiences while coding can be identified via biometrics of the developer. Using machine learning they were able to do this. Additionally, they identified the least amount of information the classifier requires to identify these emotions. Surprisingly a minimal set of sensors seems to be sufficient for the task. The third paper “Combining Biometric Data with Focused Document Types Classifies a Success of Program Comprehension” [3] tried to infer code comprehension success from (bio) metrics using machine learning. They could also successfully train a classifier.

Simply put, all three papers investigated if it is possible to infer from biometrics what a developer feels and thinks while coding using machine learning (or as I like to call it: advanced statistics). As it is with statistics, they show correlation, not causation so we now might know that the patterns in some biometric might change when the developer is about to write bad code, but we do not know why she writes bad code in the first place. Sure, we know from a lot of other research under what circumstances developers perform good or bad but why do for example brainwaves change to a specific pattern when something bad is about to happen. The third paper answers a small fraction of this question by investigating directly the relation between code comprehension and biometrics. While the link between specific patterns on biometrics is still only correlational, the causality between “the coder does only badly understand what the code does he is about to change” and “the coder produced a bug” is quite obvious.

This is interesting for the proposed applications of the findings in each paper. The first paper describes, that online biometric monitoring could be used to help developers produce less bugs by telling developers where they might have created a bug. Assuming that the change in patterns indicating “possible quality concern” and “bad code comprehension” correlate, the developer would already know that he understands the just changed code badly and should check it again or ask a colleague. If they then do so is more dependent on the company culture. It could be more interesting to use the collected biometric data to make code reviews more efficient. Code reviewers could for example get a list of locations which they should review in deep while other locations can only be overlooked. However, this approach could be negatively influenced by overly confident developers *thinking* they had understood what they did. For sure, there is a lot research to do until *FitBits* improve code quality on a large scale and as mentioned in the first paper, moral and ethical concerns are also to consider.

The setting of the studies in the second and third paper are very artificial, and both were conducted with students hence on-site studies with professional developers similar to the study in the first paper should be conducted to confirm the results. Also, of interest could be, whether senior developers have the same coping strategies as the students in the second paper.

References

[1] MÜLLER, Sebastian C.; FRITZ, Thomas. Using (bio) metrics to predict code quality online. In: *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016. S. 452-463.

[2] GIRARDI, Daniela, et al. Recognizing Developers' Emotions while Programming. *arXiv preprint arXiv:2001.09177*, 2020.

[3] ISHIDA, Toyomi; UWANO, Hidetake; IKUTANI, Yoshiharu. Combining Biometric Data with Focused Document Types Classifies a Success of Program Comprehension. In: *Proceedings of the 28th International Conference on Program Comprehension*. 2020. S. 366-3

Feedback Provided by Lecturers

Points: 3 out of 3

- Overall very good with interesting ideas, questions and insights
- Try to add a bit more detail when making an assumption, i.e. that the causality between the coder not understanding what the code does and the coder producing a bug is "quite obvious" - there might be other reasons for the bug
- The summary of each paper is good and quite to the point, but a bit more detail would be helpful (especially for the third paper)
- For the 3rd paper: please pick a full research paper, i.e. generally 8 - 12 pages

Short Response Sensing Reading
[redacted]

Thomas Fritz, Andrew Begel, Sebastian C. Müller, Serap Yigit-Elliott, and Manuela Züger. 2014. Using psycho-physiological measures to assess task difficulty in software development. In 36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - 7 June, 2014. 402–413.

After reading the first assigned paper, I was really interested in reading the 3rd paper mentioned above, since it has been cited quite a few times in the initial paper.

The 3rd paper answers the question that task difficulty can be measured using sensoric data, also providing a set of tools or techniques to record and elicit the sensory information. Many of the suggested techniques have also been used in the first assigned paper (e.g. EEG, EDA, BVP, and HR - however no eye-tracking since it might have been too expensive or due to regulatory burdens, even though the 3rd paper really benefited from the sensoric data of the eye-tracker, providing the best f-measure compared to EEG or EDA). I liked the approach of using a ML model to predict task difficulty and the therefore higher chance of committing lower quality code. Applications could be to highlight these parts of the code for review (even possible in a real-time). The first assigned paper even proved that just using the wristband results in similar classification quality compared to using the *BrainLink* headset, therefore making it feasible for real-world usage. With all the papers I've read, I wondered whether the ML classification performance could be improved since precision of ~65-70% is okay and recall of ~60% is not that great. Maybe scaling the training of such models using the massive amounts of anonymized Apple Watch / FitBit data could improve the ML performance…

Feedback Provided by Lecturers

Points: 2 out of 3

- Briefly summarize what the first two papers are about
- Go into more depth, why do you like the approach of using an ML model, and go beyond what is stated in the paper
- Refine English / grammar

Response Paper: Eye Tracking

[redacted]

[redacted]

The paper "**Improving Automated Source Code Summarization via an Eye-Tracking Study of Programmers**" (Paper 1) by Rodeghero et at. addresses the problem that there have not been any studies of how programmers read and understand source code specifically for the purpose of summarizing it. Rodeghero et al. conducted an eye-tracking study and used the results to write their own tool, which pulls keywords from a method to summarize it and then evaluated their method by comparing it to a different state-of-the-art approach. Their main limitation was the screen size and therefore the usage of only relatively small methods, since scrolling would disable the eye-tracking.
This is one particular challenge, which was later solved by Kevic et al. and described in the paper "**Tracing Software Developer's Eyes and Interactions for Change Tasks**" (Paper 2). Kevic et al. attempts to understand the detailed navigation behavior when changing code. They focus on doing their study in a realistic setting by using an open source project and its bugs. Due to the use of iTracker, they are able to let the developer use the full scope of the Eclipse IDE and are not limited to only showing one screen. iTracker associates the movement of the eyes directly with the code which is shown on the screen.

The results of the papers contradict each other in an important aspect. Paper 1 concludes that the developer spends significantly more time on the methods signature than the body of the method, when taking into account their different size. However, paper 2 states the opposite, that the developers spend much more time examining the body of the methods rather than the signature. This could be due to the fact that the purpose of the tasks conducted are completely different. The study of paper 1 was focused on solving realistic problems and therefore the understanding of the code was essential. In the study in paper 1 the main focus was the summery of the method and therefore it was not necessary to understand particular parts of the method, that might have been subject of investigation if a bug needed to be fixed. I find the argumentation of paper 2 more valid, since they focus on a realistic setting. They attempt to understand the entire process of understanding and changing code, as one would in the real world, rather than taking a look at how someone would summarize a relatively short method.

It is further interesting to point out how the usage of eye-tracking devices developed. A limitation that is stated in paper 1 is then resolved with iTracker in paper 2. This progress in research is important. It would now be nice to see the difference of results of paper 1 when more realistic setting would be adapted using the iTracker software from paper 2. When only summarizing one method that is taken out of its context, one takes a look at it from a different perspective. When a method is part of a class, the summarization process will include a look at the method as a contribution to the class. This way, different aspects of the method may be much more important than when the method is only summarized without context. Therefore, I find it important to take a look at summarization using iTracker. It would be interesting to see if the method signature would be more important as the body, like paper 1 suggest or when being part of a greater picture will make the participant focus more on the body, like suggested by paper 2.

Feedback Provided by Lecturers

Points: 3 out of 3

- Overall very good with interesting ideas, questions and insights
- Refine English/Grammar (e.g. “et al.”, “doing study” -> “conducting study”)