Response Paper: Eye Tracking

[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.