# Sensing and Supporting Code Difficulty

Xiaozhe Yao[1]

[1]University of Zurich

October 27, 2019

In modern software engineering, the maintenance usually consumes 70% of the total life cycle cost of a software product (Buse & Weimer, 2010) while the most time-consuming task in the maintenance is reading source code. I am confident that the percentage of maintenance in the total life cycle will keep growing in the next decades, therefore, maintain a highly-readable source code is significant to any enterprise software products. (Buse & Weimer, 2010) conducted a survey of 120 human annotators on 100 code snippets, presented a novel descriptive model of software readability and revealed significant correlations between their metric for readability and external notions of quality. (Siegmund et al., 2017) contrasted bottom-up and semantic cues comprehensions by fMRI, and found that comprehension based on semantic cues leads to a lower activation intensity as compared to bottom-up comprehension and leads to neural efficiency. (Barnett, Bird, Brunet, & Lahiri, 2015) proposed a tool to investigate changesets for review. They conducted both quantitative evaluation and qualitative user study and showed that it can be used widely, at least at Microsoft and most developers agree with the proposed decomposition.

In the past, we usually focus on text features to measure the code readability, i.e. line length, number of keywords/parentheses etc. However, some source code, even with extremely easy-to-understand text features, is still hard to understand. For example, the famous *Fast Inverse Square Root Method* appeared in the Quake III engine:

```
float InvSqrt (float x)
{
float xhalf = 0.5f*x;
int i = *(int*)&x;
i = 0x5f3759df - (i>>1); // seems simple, but what's this?
x = *(float*)&i;
return x*(1.5f - xhalf*x*x);
}
```

From the aspects of text features, the code above seems quite simple to understand, but from the aspects of human programmer, it may hard to understand. Therefore, the measurements from pure text features are not stable on some occasions. To address this problem, (Buse & Weimer, 2010) uses annotators to acquire a human-aspect readability measurement. **However, the model operates as a binary classifier,** though it succeeded in classifying snippets as "readable" or "not-readable" in more than 80% of the cases, it would be better if the model could point out what problems exist in the snippets and help developers to improve their code readability.

(Buse & Weimer, 2010) uses annotators to acquire a human-aspect readability measurement. **However, the model operates as a binary classifier,** though it succeeded in classifying snippets as "readable" or "not-readable" in more than 80% of the cases, it would be better if the model could point out what problems exist in the snippets and help developers to improve their code readability.

fMRI is a pretty new technique for the software engineering research, and it has a huge potential in the evaluation of new tools, software, etc. fMRI has a tight connection with EEG(Electroencephalography) since EEG has a higher temporal resolution while fMRI has a higher spatial resolution. Therefore, combining EEG and fMRI could provide complementary Spatio-temporal information for brain activity study. With the combination, we evaluate something hard to measure with only surveys, such as the task difficulties (Fritz, Begel, Müller, Yigit-Elliott, & Züger, 2014).

To help developers better understand and review code, (Barnett, Bird, Brunet, & Lahiri, 2015) introduced ClusterChange an automatic decomposition of changesets. Inspired by (Siegmund et al., 2017), we could use fMRI to evaluate if this type of toolkit (like CodeBubbles, ClusterChange, etc.) could reduce the activation intensity.

# References


Barnett, M., Bird, C., Brunet, J., & Lahiri, S. K. (2015). Helping Developers Help Themselves: Automatic Decomposition of Code Review Changesets. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. IEEE. https://doi.org/10.1109/icse.2015.35

Buse, R. P. L., & Weimer, W. R. (2010). Learning a Metric for Code Readability. *IEEE Transactions on Software Engineering*, *36*(4), 546–558. https://doi.org/10.1109/tse.2009.70

Fritz, T., Begel, A., Müller, S. C., Yigit-Elliott, S., & Züger, M. (2014). Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. ACM Press. https://doi.org/10.1145/2568225.2568266

Siegmund, J., Peitek, N., Parnin, C., Apel, S., Hofmeister, J., Kästner, C., ... Brechmann, A. (2017). Measuring neural efficiency of program comprehension. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017*. ACM Press. https://doi.org/10.1145/3106237.3106268