Homework 02 - Literature Search

Andrew Carter and Beryl Egerter

February 6, 2013

Below are short summaries of how various articles can be related to our research topic.

1. This paper is an overview of how concepts are relevant in program comprehension. It talks about both location of code concepts and how people learn from the code. It relates to our proposed study because we are interested in the process of program comprehension. [10]
2. This is a summary of six different program comprehension models and areas where they might not be as useful. This is useful information for our study as an explanation of theories of program comprehension that exist. [15]
3. This article talks about control structure diagrams, a specific method that can be used to aid program comprehension. This gives us a look at a solution to code comprehension. [6]
4. This article analyzes how programmers maintain unfamiliar code in an industrial setting. This gives a view of how program comprehension works in an industrial setting as opposed to an academic setting that we are doing our study in. [14]
5. This article proposes a course on code comprehension methods. Models used or concepts that are planned to be emphasized will likely be informative to our study. [2]
6. This article gives a theory of the code comprehension process, which may or may not be different than other theories mentioned in other papers we will read. [3]
7. This paper summarizes some terminology of reverse engineering or "program comprehension" as well as giving reasons why it is a difficult field. The paper may be useful orienting us to read other research in this field. [9]
8. This study looks at the effect of indentation in program comprehension. This will be good to keep in mind for our study as it brings up a number of small changes that could affect task completion over different problems. [7]

9. This study talks about the differences between syntactical and semantic knowledge, and include modularity tests. Probably more useful for the ideas it sparks rather than the results. [11]

10. This study discusses some techniques that expert programmers use to comprehend code, including stereotypical functions and chunks. This paper is useful because it gives us a technique to look for. [12]

11. This paper describes beacons (which are akin to chunks) and how program comprehension is affected by these beacons. This paper expands on one of the techniques described in the previous paper, and is useful for the same reason. [16]

12. This proceedings describes several ways for programmers to navigate through code, and claims that it has an effect on comprehension. Also it points out that there exists questions are easier to answer witha yes then a no, we should remember that if we have any questions in the study. [8]

13. This is a nice paper because the introduction describes in detail a lot of theories and even has a section on research implications (namely the reliance on novice programmers for experiments). Overall this is useful because it sums up many discussions from other papers. [13]

14. This paper focuses on how programmers interact with beacons, most notably they use eye tracking software to track where the participants eyes go, and show a fixation of expert programmers on beacons. This is useful as it can give Colleen an excuse to obtain eye tracking software, and also may indicate that if a user stops on a line of code for a second or so it may be an indication of synthesis into a beacon rather than a pause trying to work it out. [1]

15. This paper focuses on debugging rather than comprehension. We feel that this task is more similar to what would be asked of a programmer in a work environment. This paper is useful because they talk about strategies novices use to debug. [4]

16. This paper is very similar to the last paper, however this focuses on non-programmers. What might be interesting is whether experts differ from non-programmers or novices when they approach code with unfamiliar syntax or semantics or if their programming knowledge carries over. As such this provides a good baseline for what people would do without experience. [5]

References

[1] C. Aschwanden and M. Crosby, *Code scanning patterns in program comprehension*, (2006).

[2] III Austin, M.A. and M.H. Samadzadeh, *Software comprehension/maintenance: an introductory course*, Systems Engineering, 2005. ICSEng 2005. 18th International Conference on, aug. 2005, pp. 414 – 419.

[3] Ruven Brooks, *Towards a theory of the comprehension of computer programs*, International Journal of Man-Machine Studies **18** (1983), no. 6, 543 – 554.

[4] S. Fitzgerald, G. Lewandowski, R. McCauley, L. Murphy, B. Simon, L. Thomas, and C. Zander, *Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers*, Computer Science Education **18** (2008), no. 2, 93–116.

[5] P. Gross and C. Kelleher, *Non-programmers identifying functionality in unfamiliar code: strategies and barriers*, Journal of Visual Languages & Computing **21** (2010), no. 5, 263–276.

[6] D. Hendrix, II Cross, J.H., and S. Maghsoodloo, *The effectiveness of control structure diagrams in source code comprehension activities*, Software Engineering, IEEE Transactions on **28** (2002), no. 5, 463 –477.

[7] Richard J. Miara, Joyce A. Musselman, Juan A. Navarro, and Ben Shneiderman, *Program indentation and comprehensibility*, Commun. ACM **26** (1983), no. 11, 861–867.

[8] R. Mosemann and S. Wiedenbeck, *Navigation and comprehension of programs by novice programmers*, Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on, 2001, pp. 79 –88.

[9] Michael L. Nelson, *A survey of reverse engineering and program comprehension*, CoRR **abs/cs/0503068** (2005).

[10] V. Rajlich and N. Wilde, *The role of concepts in program comprehension*, Program Comprehension, 2002. Proceedings. 10th International Workshop on, 2002, pp. 271 – 278.

[11] Ben Schneiderman and Richard Mayer, *Syntactic/semantic interactions in programmer behavior: A model and experimental results*, International Journal of Computer and Information Sciences **8** (1979), no. 3, 219–237.

[12] Elliot Soloway and Kate Ehrlich, *Empirical studies of programming knowledge*, IEEE Transactions on Software Engineering **10** (1984), no. 5, 595–609.

[13] M.-A. Storey, *Theories, methods and tools in program comprehension: past, present and future*, (2005), 181 – 191.

[14] A. von Mayrhauser and A.M. Vans, *Industrial experience with an integrated code comprehension model*, Software Engineering Journal **10** (1995), no. 5, 171 –182.

[15] ———, *Program comprehension during software maintenance and evolution*, Computer **28** (1995), no. 8, 44 –55.

[16] Susan Wiedenbeck, *The initial stage of program comprehension*, Int. J. Man-Machine Studies **35** (1991), 517–540.