

An Approach to Understanding Program Comprehensibility Using Spatial Complexity, Concept Assignment and Typographical Style

Andrew Mohan, Nicolas Gold, Paul Layzell

Information Systems Group, Department of Computation

UMIST, UK

{a.mohan@postgrad.umist.ac.uk, n.e.gold@co.umist.ac.uk, p.layzell@umist.ac.uk}

1. The Problem

The majority of the lifetime cost of a software system is taken up by the activity of software maintenance [3]. The main reason for this is that to maintain existing code, the maintainer firstly requires a sufficient level of comprehension of it [5]. The ease with which a program can be comprehended, its comprehensibility, is dependent upon the ease to which the maintainer can gain access to the information contained within the source code itself.

The cost of program comprehension in software maintenance is dependent upon the programmer building their mental model, by searching for beacons in the code [5]. The ease of this search, or the comprehensibility of the program, is in turn strongly influenced by:

- The format the information is presented in
- The degree of difficulty in acquiring the information
- The amount of information to be acquired.

2. The Approach

The comprehensibility of a program is strongly influenced by its typographical style [4] (presentation of information), the number of concepts it contains [2] (amount of information to be acquired) and its spatial complexity [1] (degree of difficulty). These factors relate to how evolvable a program is through its comprehensibility.

The approach we applied was to capture these factors, relating them to changes in the source code of several COBOL programs over their evolution. This enabled us to begin to understand how difficult a program is to comprehend. Whereby this understanding is aimed at identifying how these factors have been changed through the application of maintenance and what relationship exists between them.

Initial results of applying this approach indicate that the programs studied had been continually changed to maintain their fitness in satisfying the purpose they were designed to fulfil. These changes have probably made the

process of comprehension more difficult. This was shown through an increase in the amount of information in the programs, as indicated by an increase in the number of concepts. In addition, by increasing the difficulty faced by a maintainer in acquiring the information contained within the programs, through an increase in the spatial complexity. Furthermore, our results also indicate that maintainers impose their own typographical style upon a program to aid their comprehension of that program.

3. Conclusions and Further Work

This paper has briefly presented an approach to identifying the comprehensibility of a program and initial results from its application. The results obtained so far, indicate that this approach is useful in modelling the comprehensibility of a program as it evolves. However further work is required to calibrate this approach to more accurately reflect comprehensibility and to identify at what point corrective action should be undertaken to maintain the quality of the program.

References

1. Douce, C.R., Layzell, P.J. & Buckley, J. (1999). Spatial measures of software complexity. PPIG-11 Annual Workshop, 47 January 1999, University of Leeds, UK, pp.36-45. www.ppig.org/workshops/11th-programme.html.
2. Gold, N.E. & Bennett, K.H. (2002). Hypothesis-Based Concept Assignment in Software Maintenance; IEE Proceedings Software, Vol. 149, No. 4, pp. 103-110.
3. Lientz, B.P. & Swanson, E.B. (1980). Software Maintenance Management; Addison-Wesley, Reading M.A.
4. Oman, P. & Cook, C. (1990). Typographic style is more than cosmetic; Communications of the ACM, Vol. 33, No. 5, pp. 506-520.
5. Wiedenbeck, S. (1991). The initial stages of program comprehension; International Journal of Man-Machine Studies, Vol. 35, No. 4, pp. 517-540.