ELSHOFF82

Elshoff, J.L., Marcotty, M.; Improving Computer Program Readability to Aid Modification; CACM 25, 8 (August 1982) pp. 512-521.

This paper, although it is probably applicable in other areas, is aimed mainly at commercial data processing environments where it has been estimated that 75% of all programmer/analysts' time is spent on existing program modification. The authors argue that program readability is extremely important because: Programs are models of some part of the real world and as this world changes, programs must be modified to keep pace. Also as new software is developed, the inventory of programs to be maintained grows so that the amount of program modification is not expected to decrease. The modification process always starts with a program and its documentation, which quite often means just a listing of the program's source code.

Further, in a study of commercial programming practices cited by the authors, it was found that most programs were poorly written, large, extremely difficult to read, and more complex than necessary and that programming language usage was poor and inconsistent.

The authors put forth their thesis that modifying a program simply to improve its readability is generally a worthwhile endeavor. They introduce various transformations that can be applied to a program to increase its readability. They take the point of view of a programmer who is knowledgeable in the application area of the program but who is seeing the particular program for the first time.

The following is a summary of the transformations to be applied (often in several passes) to a program to increase its readability. Program transformations: These consist of adding comments to the source code and reformatting it. The recommended style of commenting and program formatting resemble closely the commenting and formatting conventions that are taught to first year computer science students. Readability transformations: These are the changes that actually reorganize the code of a program. The authors list thirteen of these transformations. All are fairly simple and do not require any deep undestanding of the program. However, the ideas is that as a programmer makes these changes, he/she will gradually gain a good understanding of it.

The transformations include modifying IF statements, making loops obvious, making loop termination explicit, creating procedures under certain circumstances, introducing status variables to track execution, etc. To give one concrete example, another transformation is adding ELSE clauses to IF statements. The authors state that adding an ELSE clause (even if it is null) removes ambiguity from an IF statement because of the optional nature of the ELSE clause.

The paper concludes by recommending that the step of modifying a program to make it readable be added to the modification cycle which would then consist of: The user requests that a program be changed. The program is made readable (if need be). Specifications for the change are written and the cost is estimated. It is decided that the changes are worth being made. The program is changed.

In summary, this paper puts forth some good ideas on improving the quality of code of existing programs. It is oriented towards the practical side of programming and is relevant because it acknowledges the current state of data processing software and its long expected lifetime and then presents a simple method of improving the quality of that software.