Software Science, Cobol Analyzer, Software Engineering, Modularity, Effort Estimation

This report studies the applicability of the Theory of Software Science to COBOL Programs. A COBOL analyzer was written to produce the most important Software Science metrics. The analyzer was used to study several COBOL Databases. The analyzer has been distributed to AIRMICS and to several other organizations including Ohio State University, Jet Propulsion Labs, CINCOM INC., and the University of Trondheim, Norway.
Using the analyzer we investigated the Software Science Effort Estimator and compared this with estimator based on lines of code and on "cyclometric complexity." Our experiments show that the software science effort estimator works reasonably well only if a program is properly modularized and if inter-module factors are included. Published reports on our findings are available on request.

We also studied the language level hypothesis of software science in a COBOL environment. While the study shows that the mean language level for COBOL falls between those for FORTRAN and PL/I, the study also shows that language level is heavily dependent on the size of the program and that it fluctuates too much to be considered "constant" as the Halstead theory postulates.
Final Report

Systematic Study of COBOL Programming

S. D. Conte
V. Y. Shen

Computer Sciences Department
Purdue University
W. Lafayette, IN 47907

Period covered by report: 09/09/79-03/08/81
ARO project number: P-16400-A-EL
Contract number: DAAG29-79-C-0172

Scientific Personnel Supported by this Project:

Alka Agrawal
Dennis Cok (M.Sc. 1980)
Stephen M. Thebault
Scott N. Woodfield (Ph.D. 1980)
1. The Problem Studied

The purpose of this research was to study the applicability of the theory of Software Science in managing the development of COBOL programs. Since Software Science was a theory in its infancy, we hoped that through this study we could obtain evidences supporting the theory or make refinements to the theory.

2. Summary of Important Results

During the contract period we have developed an analyzer for COBOL programs, gained additional insight in modularized program development, and discovered the limitations on the language level hypothesis of Software Science. These three accomplishments are described below:

2.1. The COBOL Analyzer

The necessary tool for systematically studying COBOL programs is an automatic analyzer. It was completed in early 1980. We used it to analyze a variety of programs and the preliminary results were reported in [1]. The major findings were that the Software Science length equation works as well in COBOL programs as in other languages previously analyzed, that the effort equation appeared to work for one single-programmer, single-module program; and that the language level equation did not provide stable results. To date the analyzer has been distributed to the following institutions for research purposes:

AIRMICS
The Ohio State University
Jet Propulsion Laboratory
Cincom, Inc.

The University of Trondheim, Norway

Mr. Wilhem Ottes of the University of Trondheim has conducted an elaborate experiment using the COBOL analyzer. He wrote a report which generally confirmed our findings [2].
2.2. Modularity Factors in Effort Estimation

Effective software management requires accurate estimation of the programming effort. The "lines of code" measure, the "cyclomatic complexity" measure, and the "effort" measure of Software Science have all been proposed to be important factors in programming effort estimation. We have conducted experiments which showed that neither measure was adequate in estimating effort for single-programmer, multi-module programs unless the modularization and module-interconnection factors were included [9]. This important extension of the Software Science theory has yielded two publications [4,5]. A third report on the research has been submitted [6].

2.3. The Language Level

A hypothesis of Software Science is that the power of a programming language can be quantified by the "language level" equation under certain assumptions. Previously reported studies show that the equation yields numbers that rank languages such as PL/1, Algol, FORTRAN, and assembly language in the desired order. However, the numbers reported were based on hand-analysis of small sets of programs. The development of automatic analyzers enabled us to analyze hundreds of programs written in COBOL, FORTRAN, Pascal, PL/1, and assembly language. The language level based on Software Science theory does not remain constant. It appears to have a strong dependency on program length. The results of our analysis are reported in [7].

3. List of Publications


