1973

On Program Volume and Program Modularization

Rudolf Bayer

Report Number:

73-105

ON PROGRAM VOLUME AND PROGRAM MODULARIZATION

Rudolf Bayer
Purdue University
and
Technische Universitat
Munich, Germany

CSD TR 105

September 1973

105

ON PROGRAM VOLUME AND PROGRAM MODULARIZATION

Rudolf Bayer

September 10, 1973

Following [1] and [2] we define for a program p:
η_1 = the number of distinct operators in p,
η_2 = the number of distinct operands in p,
N_1 = the total number of occurrences of operators in p,
N_2 = the total number of occurrences of operands in p,
$\eta_p = \eta_1 + \eta_2$, the number of distinct basic objects in p,
$N_p = N_1 + N_2$, the length of the program p.

Then the volume V_p of a program p is defined to be:

$$V_p = N_p \log_2 \eta_p \quad [\text{bits}].$$

Program modularization shall be the writing of a program in the form of several nearly independent, only losely connected pieces. The volume of a program p written as a finite collection M of modules m shall be

$$V_p^M = \sum_{m \in M} V_m = \sum_{m \in M} N_m \log_2 \eta_m$$

A modularization C of p shall be called <u>complete</u> iff $|C| > 1$ and

$$\sum_{c \in C} N_c = N_p \text{ and } \sum_{c \in C} \eta_c = \eta_p \quad .$$

Then clearly $V_p^C < V_p$ since $V_p^C = \sum_{c \in C} N_c \log_2 \eta_c < N_p \log_2 \eta_p = V_p$.

Realistic modularizations M of p will generally not be complete for at least two possible hazards:

H1) $\sum_{m \in M} \eta_m > \eta_p$ because of some operators and operands being used in several modules.

H2) $\sum_{m \in M} N_m > N_p$ because of certain necessary additions to the code, e.g., loading base registers, transmitting parameters, etc., when control must be transfered from one module to another.

Although the length of a program may in many cases increase due to modularization, its volume may still decrease. We call a modularization M of p <u>good</u>, whenever $V_p^M < V_p$.

Guidelines for good Modularizations:

In trying to obtain good modularizations one should avoid hazards H1) and H2) as far as possible by following certain guidelines. These guidelines have intuitively been known and used for a long time:

G1) Minimize $\sum_{m \in M} \eta_m$: make the interface between modules, i.e. the set of objects common to several modules - like common global variables, parameters, common procedures, etc. - as simple and small as possible.

G2) Make code expansion due to modularization as small as possible by providing appropriate hardware and software support. This category covers features like:

a) use of base registers,

b) simple parameter passing mechanisms,

c) efficient procedure calls, etc.

Definition:

A modularization O of p is called optimal (over the programming language L), if O miminizes the volume of p, i.e. if for any other modularization M of p (written in L) we have:

$$V_p^O \leq V_p^M .$$

Conclusion:

If the effort E to write a program p is a monotonically increasing function of the volume of p, then an optimal modularization of p minimizes the effort to write p.

NOTE:

An optimally modularized program in a higher level programming language need not give rise to an optimally modularized program in machine language.

Example: Assume we have a program p according to model A of [1] with

$$\eta_p = 180$$

$$N_p = 1040$$

$$V_p = N_p \log_2 \eta_p = 7791$$

a) Incomplete modularization M:

Assume we write the program in 5 modules of equal size with an incompleteness factor of $\approx 10\%$, i.e. using $\eta_i = 40$ and

$$N_i = 230,\ i = 1,2,\dots,5.$$

Then the volume of the modularized program is:

$$V_p^M = 5 \cdot 230 \ \log_2 40 = 6120$$

b) Complete modularization C:

In a complete modularization C of p of 5 equally sized modules we would have:

$$V_p^C = 5 \cdot 208 \log_2 36 = 5377$$

Comparison of Effort: Let E_p, E_p^M, E_p^C be the effort to write program p in the unmodularized form, and according to modularizations M and C resp. Then the following table gives a relative comparison of these quantities under the two assumptions, that they are proportional to the volume or to the square of the volume.

	If effort proportional to volume	If effort proportional to square of volume
Improvement of E_p^M over E_p	21.4 %	38.3 %
Improvement of E_p^c over E_p	31.0 %	52.4 %
Degradation of E_p from E_p^M	27.3 %	62.1 %
Degradation of E_p from E_p^C	44.9 %	110.0 %

REFERENCES

[1] Halstead, M. H.: Natural Laws Controlling Algorithm Structure? ACM-SIGPLAN Notices 7,2, Feb. 1972, pp. 19-26.

[2] Halstead, M. H., Bayer, R.: Algorithm Dynamics. Proceedings of the ACM Annual Conference 1973, pp. 126-135, Atlanta.