談話室

プログラミング言語とソフトウェア工学†

有　澤　誠††

長尾真先生の「ソフトウェア工学に対する思想」[1]を読み，たいへん興味深かった．このような議論が，情報処理誌上でこれからも盛んになることを期待し，筆者もこの小稿を投じてみることにした．

日本のソフトウェア工学の現状からみて，最も根本的な問題点のひとつは，ソフトウェアというものをかなりゆるく把えていることであると考える．「ソフトウェア」は「プログラム」である，というのは正しい．しかし，「プログラム」は必ずしも「ソフトウェア」であるとは限らないことに，もっと目を向けなければならない．必ずしも厳密とはいえないが，プログラムを書いた当人以外の不特定多数の人たちが使用するプログラムがソフトウェアである，として以下の議論をすすめていく．「プログラム」は「ソフトウェア」よりも広い概念であり，それだけ特徴づけ性格づけの条件がゆるい．(同様な例は，正規言語のクラスと文脈自由言語のクラスの対比である．もっとも，こんな例をひいたのでは，筆者のお里が知れてしまうかもしれない)．プログラムは，書いた当人の目的さえ果せば，一応は満足であることから，その機能は「部分正当性」を満たせばよいことが多い．ここで，「部分正当性」と書いたのは，停止性を考慮するかどうかだけでなく，入力ドメインの部分集合に対して正当性を満たすだけというほどの，広い意味に解釈してほしい．これに比べて，ソフトウェアであるためには，予期しうるどのような使われかたに対しても，理性的にふるまう必要があることから，その機能は「全域的正当性」を満たさなければならない．このことは，オペレーティング・システムやコンパイラなどのシステム・ソフトウェアだけでなく，統計計算や数値計算などの数学的ソフトウェアにも言えることである．

言い換えれば，プログラムはたしかに「アルゴリズム」+「データ構造」[2]であるが，ソフトウェアはそのほかに，注釈などの文書，アサーションなどの正当性保証および防衛的プログラミング・コード，さらに外部仕様などの文書を備えなければならないわけである．

プログラムを記述するために，現在我々が用いている言語（プログラミング言語）の多くは，プログラムの記述には便利にできていても，ソフトウェアの記述には不足している部分がある．我々がほんとうに「ソフトウェア」工学をめざすためには，そのための専用の言語，いわばソフトウェアリング言語 (softwaring languages) が必要なのではないか．「ソフトウェアリング」などという新語を発明したりすると，かの Dijkstra から「英語がなっていない」としかられるかもしれないけれども[3]，筆者の論点は明らかになると思う．

Dijkstra の「構造的」プログラミングの受けとられかたをみても，「構造的プログラミング」か「構造的ソフトウェアリング」かがはっきり区別されなかったように思われる．「プログラム」を書く手法を示しながら，こうすればよい「ソフトウェア」もできると説いたところで，現場からの反論が出ることは当然であろう．

現在の大学教育をみても，「ソフトウェア」を教えるはずの授業で，実際には「プログラム」を教えるところまででとどまっているケースが，筆者の身辺では少くないように見受ける．プログラムを書いた経験をもつ研究者は，ここ何年かの間にきわめて数が多くなった．しかし，ソフトウェアを生産した経験をもつ人の数は，その何割にも満たないであろう．プログラムを書いた経験だけを基にして，ソフトウェア一般を論じた場合には，時に錯覚がはいりこむことがあっても不思議ではない．

プログラムを書くためだけならば，自家用のプログラミング言語で十分ことたりる．自家用に特注しなくても，既存のものでも使い慣れさえすれば，たとえアセンブリ言語や機械語そのものであってもさしつかえない．ソフトウェアを作る段になってはじめて，使用する言語のもつ意味が大きくなる．プログラミング言

† Programming Languages and Software Engineering by Makoto ARISAWA (Computer Science Dept. Yamanashi University)

†† 山梨大学計算機科学科

語として評判のあまりよくないFortranが今もって哀えをあまりみせない理由は，プログラム・モジュールの蓄積性に優れ，かつ処理系が広範囲に得られることによる移植性が，すべてとはいわないまでもかなりの比重を占めていると思う．プログラミング言語としてのFortranの人気が哀えても，それに代りうるソフトウェアリング言語がなければ，問題は解決しない．ここで，Fortranをアセンブリ言語と読みかえても，ほぼ同様な主張がなりたつ．

プログラムを書く場合，その知的活動のかなり多くがアルゴリズムにむけられるであろう．プログラミング言語は，「アルゴリズム記述」に重点をおいて設計されてきたことになる．いっぽうソフトウェアを作る場合には，アルゴリズムもさることながら，モジュール間の機能の相互作用のほうにずっと重点が移ってくる．研究者の目からみれば，Hoareの文献[4]のようなプロセス間交信という立場でのプログラミング言語設計ということになるのであろう．また，Bachmanの文献[5]のような立場，すなわちデータの海を，プログラマがうまくかじをとって航海するという考えかたも，従来のプログラミング方法論にない発想を含んでいる．

ソフトウェアに対する哲学の確立と，ソフトウェアリング言語のあるべき姿の追求とは，最終的に同じゴールを目ざしているもののように考えられる．そして，個々のプログラムを書くための人間の頭脳のはたらき，すなわちアルゴリズムをデータ構造を介してプログラムに組立てる知的活動を，ひとつのソフトウェアを作るための組織体の一部としての人間の頭脳のはたらき，すなわち個々の機能をはたすプログラムを全体のわくぐみにうまくはめこめるように組立てる作業と，明確に区別してかかることが，このゴールへむかう第一歩であると考えられる．

参 考 文 献

1) 長尾 真：ソフトウェア工学に対する思想，情報処理 Vol, 19, No. 9, pp. 890～891 (1978).
2) N. Wirth: Algorithms+Data Structures=Programs. Prentice-Hall [片山卓也訳，科学技術出版社] (1976).
3) E.W. Dijkstra: On a Political Pamphlet from the Middle Ages. Software Engineering Notes 3-2, 14～16 (1978).
4) C. A. R. Hoare: Communicating Sequential Processes. Comm. ACM, Vol. 21, No. 8, pp. 666～677 (1978).
5) C. W. Backman: The programmer as navigator. Comm. ACM, Vol. 16, No. 11, pp. 653～658 (1973) [星守，西村恕彦共訳，bit 6～13, 27～35 (1974)].

(昭和53年10月4日受付)