プログラミングと文章技法

玉井哲雄(東京大学)

3つのメタファー

ソフトウェア開発のメタファーとして，「書く」「作る」「育てる」のいずれが適切か，という議論がある．あるいは「記述」「工学」「育成」のいずれか，と言いかえてもよい．

歴史的にはまず「書く」があった．確かにプログラミングはプログラミング言語という人工言語による記述作業であり，一種の言語表現行為であるといえよう．プログラムをキーボードで入力し，あるいは視覚的プログラミング言語を用いてマウスによる図形操作でプログラムを作成する現在でも，「プログラムを書く」という表現は生き残っている．

バベッジの解析エンジンのためのプログラムを作り，世界最初のプログラマといわれるエイダ・バイロン(ラブリス夫人)は，その作業を編み物を「編む」ようだと言ったという．また，1960 年代から 70 年代のプログラマは「プログラムを組む」という表現をよく使った．これはもしかすると「活字を組む」作業の連想ではないだろうか．とにかく，これら「編む」や「組む」というメタファーも，「書く」に近い発想である．

これに対して，「工学」として製品を「作る」のがソフトウェア開発だとする見方は，「書く」メタファーへのアンチテーゼとして出てきたものと言えるだろう．言語による記述とはあくまでも個人的な営みである．しかし工業製品としてのソフトウェアは，確立された方法論に基づき，組織によって系統的に開発されなければならない．すなわち従来の機械工学や電気工学と同じようなエンジニアリングであるべきだ，というわけである．こうして 1960 年代末に「ソフトウェア工学」と命名された分野が誕生した．

しかし，ソフトウェア「工学」が伝統的な工学に追いつこうと努力を重ねるうちに，やはりソフトウェアというしろものは，機械やトランジスタとはだいぶ違うのではないか，という疑念も生じてきた．ソフトウェアは抽象的であり，物理的な実体がないという特徴を持つ点で機械やトランジスタと違うということは，初めから分っていた．その上で，ソフトウェアには手を加えやすい，後から機能を追加することが日常茶飯事である，という性質のあることが強く意識されるようになる．機能追加ならよいが，もともと潜んでいる欠陥(バグ)を，運用に入ってから気づいて取り除くということもこれまた普通のことで，他の工業製品では考えられないことだが，このようなバグの修正も「保守」と呼ぶ．しかしこれも，ソフトウェアに手を加えやすいという性質の一面を示すものともいえる．

そこでソフトウェアは，全体の設計を一度にすませて丸ごと開発し提供するというよりは，少しずつ手を加え，機能を高め，使いやすくなるように「育て」ていくというメタファーが有効ではないか，という議論が生れてくる． そうして成長した大きなシステムの内部は，入り組んだ構造となる．それにさらに手を加えていくには，中がどう仕切られその間がどうつながっているか，どこに何が置いてあるか，というような生活的な知識が必要となる．そこでソフトウェアを家や巣に見立て，そこに「住む」あるいは「棲む」という

メタファーを考えることもできる．これも家の修理，改築，増築を念頭においているという意味で，「育てる」に近い発想である．

筆者は昨年「ソフトウェア工学の基礎」という本を出し，日頃ソフトウェア工学を研究し教育している立場ゆえ，「工学」メタファーがもっとも適切といいそうなものであるが，実はこの 3 つのメタファーはいずれもある面の真実を表していると思う．中でも一番古い「書く」というメタファーは，いまだにその意義を失っていないと考えるので，ここからは話をそこに絞ってみたい．

文章読本

プログラミングが文章を書くという行為に似ているとすれば，よいプログラムを書くにはよい文章を書く技術が参考になるに違いない．こういう考えは昔からあり，有名な文章作成法の本，ストランク&ホワイト著"The Elements ofStyle"(初版は 1959 年．現在第 4 版が出版されている)を下敷きとして，カーニハンとプローガの"The Elements of Programming Style"が書かれたことは，よく知られている．

しかし，これは英語の文章法の話をもとにした話である．日本人であるわれわれは，日本語の文章法を参考にしたほうがよいのではないか．筆者は，「文章読本」の類を昔からかなり読んできた．同じ文章読本という表題を持つ書物を，多くの人が書いている．たとえば，谷崎潤一郎，川端康成，三島由紀夫，中村真一郎，丸谷才一，井上ひさしといった作家達が出していて，これらをすべて愛読してきた．そこへ斎藤美奈子「文章読本さん江」（筑摩書房，2002 年）という本が登場した．これは世の文章読本を並べてまとめて料理してみせようという，人が考えなかった企みで作られた書である．そこで取り上げられるのは作家達の文章読本だけでなく，清水幾太郎「論文の書き方」，木下是雄「理科系の作文技術」，本多勝一「日本語の作文技術」のような，文学的な作品より論文や実用文をどう書くか，というものもかなり含まれる．これらもまた，筆者はかなり読んできた．さらに知人が書いたもの，したがってプログラミング畑に近い人によるこの手の本を挙げておくと，木村泉「ワープロ作文技術」，杉原厚吉「どう書くか―理科系のための論文作法」がある．

しかし，斎藤はさらにおびただしい文章読本・作文技術本を捜してきて，その徹底振りは驚嘆に値する．このように汗牛充棟の文章読本ものが共通に勧めるのは，

・簡明に書け
・起承転結を大事にせよ
・文のリズムを大事にせよ

といったあたりである．プログラミングにとっても簡明に書くこと，全体をきちんと構成することは，もちろん重要である．文のリズムに直接相当するものはあるいはプログラミングにないかもしれないが，モジュール分割の歯切れよさなどが思い浮かぶ．

また，文章の修練方法でだれもが挙げるのが

・名文を読め

である．よいプログラムを読むことがプログラミングの学習にとってきわめて有効であることは，多くの人が主張することであるが，それほど広く実践されてはいないようだ．すぐれたプログラムを鑑賞するには，それなりの読解力と審美眼を必要とする．しかし，プログラマが読まなければいけないのは，残念ながら名プログラムばかりではない．プログラムの保守のためには，およそ駄作・愚作と思われるものも，動いて役に立っている以上，読んで理解する必要が生じうる．その場合でも，読む力が高ければ作業の効率は増す．さらに審美眼があれば，そのプログラムをどう書き直すべきかという指針も自ずから生れてこよう．

文芸とプログラミング

ちょっとこじつけ臭いと思う読者もおられよう．しかし，プログラムは計算機が読むものでコンパイラに通りさえすればよい，というものではない．やはり自分自身も含めた人間が読むものでもある．それを強調したのが，ドナルド・クヌースの Literate Programming である．これは文芸的プログラミングと訳されたことがあったが，ここでいう literate は「人が読める」というほどの意味ではなかろうか．もちろんそれには，人が読んで楽しめる，鑑賞できる，というニュアンスが含まれているから，「文芸的」というのもあながちおかしくはないかもしれない．で，「文芸」であればまさに「文章読本」の世界ということになる．

クヌースには聖書についての著作もあるし，小説も書いている．しかし文学に造詣の深い計算機科学者はクヌースばかりではない．チューリング賞受賞者のトニー・ホアも，ジャクソン法で著名なマイケル・ジャクソンも，ともにオクスフォード大学の文学部の出身である．かつて筆者はマイケル・ジャクソンの著書を翻訳する機会があったが，イギリス文学が縦横に引用されるので苦労した．とくにすでに古典となっている文学作品の場合，書名や登場人物の表記には定訳があるだろうから，勝手なものをひねり出す訳にいかない．幸い，筆者の勤務先の同僚にディケンズ研究者がいて教えを乞うことができた．

別にプログラマに文学の素養が必要だと言いたいわけではない．しかし，プログラマが書かなければならないのはプログラムだけではない．仕様書，マニュアル，解説書，報告書など，書くことを要求される文書は山ほどある．それを書くのに，簡にして要をえた文章をものす力は，やはり必須のものだろう．