第 2 章

ソフトウェア危機の歴史

2.1 ソフトウェア危機とは

コンピュータの歴史を振り返ると，その誕生の前に既に長い歴史がある。文献(9)によれば，四つの大きな潮流があった。その第一はフランスの数学者 Pascal の加算機の発明から卓上計算機に至る四則演算の機械化の流れである。その第二は情報の符号化の流れである。穿孔カードの発明から Hollerith の統計機械を経て PCS (punched card system) に至る情報の符号化の技術は，今日のコンピュータの記号操作機能という大きな特徴を示している。第三はイギリスの発明家 Babbage に始まる計算手順の自動化の流れである。第四は真空管に至る電子技術の発達である。そして，1946 年に真空管を使った最初の電子計算機 ENIAC がペンシルバニア大学で開発された。

これらのうち，第三の計算手順の自動化が現在のプログラムの概念の始まりである。このプログラムの重要性は，Neumann の提案したプログラム内蔵方式に基づく最初のコンピュータ EDSAC-1 がケンブリッジ大学で 1949 年に開発されて以来，今日まで増大し続けている。この間 30 年あまり，ハードウェア技術はめざましい進歩をとげたが，ソフトウェア技術，なかでもソフトウェア生産技術は情報化社会の要求に応えられるレベルに至っていない。ある調査によると，世界中のコンピュータの計算能力を MIPS 値で表すと，1980 年代後半は年平均 80%の割合で増加している。しかしながら，コンピュータも「ソフトウェアがなければただの箱」といわれるように，これだけのコンピュータを活かして用いるにはそれに見合ったソフトウェアが必須である。

このような危機的状況を一般に「ソフトウェア危機」と呼ぶ。その歴史は古いが，ここでは，表2.1に示すように，1970年代，1980年代，1990年代の3つの時期に分け，各々をソフトウェアの「規模」，「量」，「質」の問題に対応付けて説明する。

表2.1 ソフトウェア危機の歴史

時 期	ソフトウェアに関する課題
1970年代	「規模」の問題： 100万ステップオーダのソフトウェアの作成方法
1980年代	「量」の問題： 情報処理技術者不足に伴うバックログの解消方法
1990年代	「質」の問題： 社会的影響大のソフトウェアの故障の防止方法 エンドユーザ向けインタフェースの実現方法

2.2 規模の問題

最初にソフトウェア危機が言われたのは，1970年頃である。当時はハードウェアが非常に高価であり，グロッシュの法則に従ってどんどん大型化されていった。それに伴って，次のようにソフトウェアも大規模化していった。

①IBMの汎用大型コンピュータのOSの例では，機械語命令数で1962年に189キロステップであったものが，1974年には3.7メガステップと20倍近く膨張している。(文献(10))

②米国の有人宇宙飛行プログラムの機械語命令数は，図2.1に示すように1970年頃に既に10メガステップを超えている。

③銀行オンラインシステムのソースプログラム命令数は，図2.2に示すように1980年代末に稼動し始めた第3次オンラインシステムでは既に10メガステップに達している。1995年に稼動予定の郵便貯金3次オンラインでは合計30メガステップと予想されている。

しかしながら，1970年当時は，このような大規模なソフトウェアを開発するための技術が無かったため，早晩ソフトウェアがハードウェアの大型化に対応

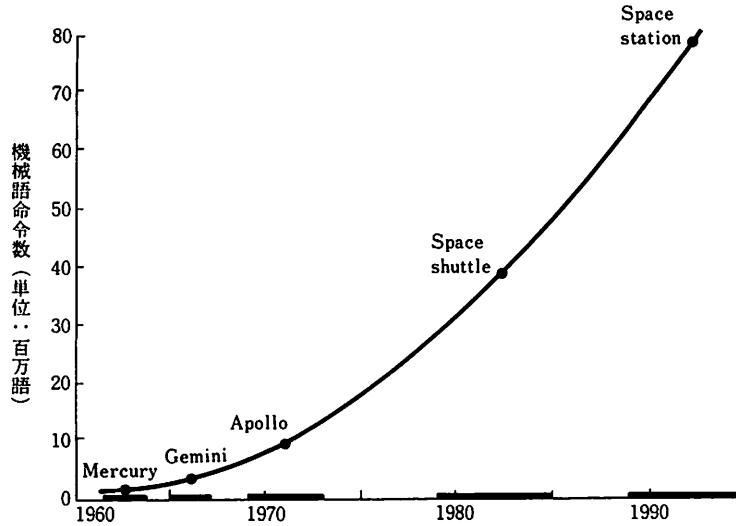

図2.1　ソフトウェア規模の増大：米国の有人宇宙飛行プログラム
（文献(11)より引用，© 1987 by IEEE）

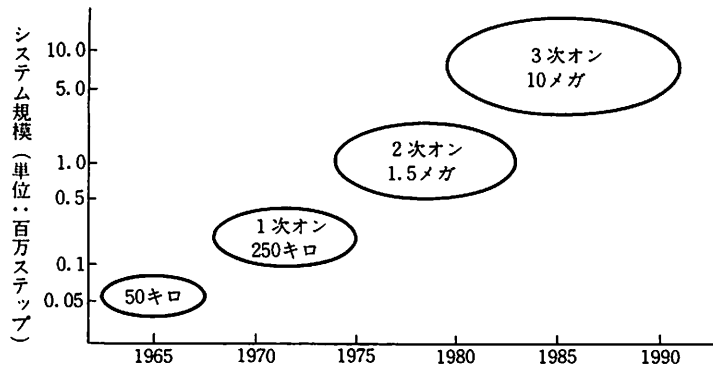

図2.2　銀行オンラインシステムの規模の変遷(文献(12)より引用)

できなくなるのではないかという心配があった。そのため70年代は，2つのアプローチがとられた。その第一は，「構造化プログラミング」に代表されるプログラミング方法論と言語に関する研究である。プログラムの記述容易性よりも理解容易性の方が重要視され，良い構造のプログラムを作るための構造化設計

法や構造化プログラミング技法とその言語が開発され，普及していった。

その第二は，「ソフトウェア工学」に代表される研究であり，ソフトウェア開発工程のあらゆる局面，すなわち，要求定義，設計，製造，検査，保守運用を対象に方法論，技法，ツールの開発が行われてきた。しかし，解決すべき課題の多くが80年代に持ち越された。

2.3 量の問題

規模の問題の後に量の問題が続いた。コンピュータシステムの普及と共に，開発すべきソフトウェアの量が増大し，情報処理技術者の不足が第二のソフトウェア危機をもたらした。通産省では，1990年には情報処理技術者が60万人不足（需要：160万人，供給：100万人）という推定に基づき，1985年には∑プロジェクトを発足した。さらに2000年には97万人不足（需要：215万人，供給：118万人）という推定が1987年に出されている。この現象は，かつての電話交換手の場合と同じく，“一億総プログラマ化”の不安をかり立てている。このような情報処理技術者の不足は，必然的にその平均的能力の低下という二次的な問題をも招いている。開発すべきソフトウェアがより複雑化し，情報処理技術者により高い能力が要求されるにもかかわらず，その平均的能力が低下している現象を筆者はセマンティクギャップになぞらえて「教育的ギャップの拡大」と呼んでいる。

そのため，ソフトウェア工学の分野では，開発者の作業効率向上や個人差の縮小を目的とした統合的ソフトウェア開発支援環境や部品化再利用を目的としたパッケージ化技術が研究され始めた。さらに，ソフトウェア基礎論の分野では，従来の手続き的プログラミングに代わる様々の新しいプログラミングパラダイムの研究が行われてきたが，未だ実用的成果をあげるに至っていない。

一方，官庁では情報処理技術者養成のために，通産省の情報処理技術者試験制度，労働省のハイテクカレッジ構想，通産省と労働省共同の地域ソフトウェアセンター構想，文部省の情報処理教育拡大策などの施策が取られている。

2.4　質の問題

量の問題が未解決のまま，さらに質の問題が重要になってきている。質の問題は，グローバル化の観点での信頼性の問題とパーソナル化の観点での使い勝手の問題が大きい。前者では，社会的に重要な役割をになうメガステップオーダの大規模ソフトウェアの信頼性がますます重要になってきている。1990年にソフトウェアが原因で生じたコンピュータシステムの事故の例をいくつか以下にあげるが，いずれも社会的影響の大きいものである。

●銀行オンラインシステムのダウン
●証券取引所売買システムのダウン
●水道局システムの料金請求誤り
●座席予約システムのダウン
●電話交換システムのダウン（米国）
●航空管制システムのダウン（英国）

また一方では，ハードウェアの急速な進歩と低廉化により，コンピュータは情報処理の専門家に限らず，幅広く一般の人達に利用され始めている。このような大衆化へ対応するためには，誰でも簡単に使えるソフトウェアの開発が必須である。さらにエンドユーザ自身が自分の使うソフトウェアを簡単に作れるようなエンドユーザコンピューティング技術が重要になっている。これらの課題が克服できなければ，ハードウェアが如何に安くなっても宝の持ち腐れになってしまうという意味で第三のソフトウェア危機と言える。

2.5　ソフトウェア生産技術の課題

(1)　情報化社会への対応

情報化社会に対応して，これら「規模」と「量」と「質」に関するソフトウェア生産技術への要求は急速に増大している。その典型的なものとして，以下の三つを挙げることができる。

●「大規模高信頼ソフトウェアを早く作って，長く利用できる」技術

●「使い勝手の良いソフトウェアを簡単に作って，どこでも利用できる」技術
●「機器制御用ソフトウェアをハードウェアの専門家が簡単に作れる」技術

第一の大規模高信頼ソフトウェアはグローバル化への対応である。最近のビジネス分野のSIS，産業分野のCIMの構築に見られるように，これらのコンピュータシステムとしては，広域ネットワークを前提に，従来のアプリケーションを統合し，分散コンピューティングやノウダウン，ノンストップのオンラインイントランザクション処理を実現する，より複雑で大規模かつ高信頼のソフトウェア開発が必要である。

第二の使い勝手の良いソフトウェアはパーソナル化への対応である。オフィスにおけるワークステーション，パソコン，ワープロなどの普及に伴って，情報処理の専門家ではない一般の人がエンドユーザとなるため，使い勝手の良いOAソフトやエンドユーザ自身が簡単にプログラムを作成できるツールが必須である。処理対象も数値データからマルチメディアデータへ拡大してくる。さらに分散システムへの移行に対応して，アプリケーションソフトウェアの移植性が重要になる。

第三の機器制御用ソフトウェアはインテリジェント化への対応である。工作機械や計測機器から自動車，家電に至るまでマイクロコンピュータが埋め込まれ，熟練技術者のノウハウのプログラム化による知能化が進んでいる。このような分野は多岐にわたるため，ハードウェアとソフトウェアの両方のわかる技術者を確保することは容易ではない。そのため，高性能でコンパクトな機器制御用ソフトウェアをハードウェアの専門家が簡単に作れる技術が必要とされている。

(2) ソフトウェア開発の問題点

ソフトウェアがあらゆる分野でコンピュータシステム普及のキーテクノロジーになってきている。しかしながら，現在のソフトウェア生産技術は，職人芸的生産技術および労働集約型生産形態から脱皮していないため，上記のようなソフトウェアの大規模化，量の増大及び機能の複雑化に対応できていない。「欲しいソフトウェアを早く，安く入手できる」ようにはなっていないが，自前で新規開発する場合は以下のような問題がある。

①開発する人がいない　←――　情報処理技術者の不足
②開発に時間がかかる　←――　バックログ増大
③開発コストが高い　←――　人件費増大
④保守コストが高い　←――　ソフト資産増大
⑤品質が不十分である　←――　大規模化と複雑化

次章では，このようなソフトウェア生産技術に関する課題の解決方法について述べる。