日本のソフトウェア産業

鳥　居　昭　夫

第1節　ソフトウェアの重要性

情報化社会がシナリオどおりに進んでいくか否かは，ハードウェアが見込みどおりに進歩し，社会に装備されていくかどうかのみならず，同時に有用なソフトウェアがバランスよく蓄積されていくかどうかにかかっている。現在，ＩＮＳの実験が国民の注目のなかで始まったところであるが，このネットワークシステムのもとに提供されはじめた，または計画されている様々なサービスのうち，どのくらいが有用なサービスと認知されて生き残ることができるのであろうか。ＩＮＳの様な新しいインフラストラクチャーが社会に根をおろし市民権を得るためには，そのインフラストラクチャーの上で提供されるサービスに充分な需要が発生しなければならない。

一つのエピソードがこれをものがたっている。数年前，非常に注目を浴びて完全機械化スーパーストアが開店したことがあった。しかし，そのスーパーストアは程なく閉店の憂き目にあっている。開店してしばらくは次々と見学客がおとずれ，見掛けだけは盛況だったそうである。しかし，販売実績はそうではなかったらしい。そのスーパーストアが行なった機械化は，すべてボタンで商品を選択せねばならないシステムで，商品を手にとって選択することはできず，新奇性はあったもののけっして消費者にとって便利なものではなかったのであろう。もう一つの例は，最近になって横浜に開店したやはり大幅に機械化されたスーパーストアである（西友ストア能見台店)。客の数よりも，偵察に来る商売敵や見学に来る研究者などの背広姿のほうがずっと多い(開店1年間に3千件1万人）という状況は，倒産した完全機械化スーパーとかかわりはないということである。しかし，こちらのスーパーはただやみくもに機械化したわけではなく，様々な工夫が施してあり消費者の間での評判もよいそうである。例えば，計算機付きシッピングカート，ハムの自動スライサー，無菌冷蔵庫といったもので，150坪の売り場の活性化に投資された金額は通常の倍の10億円ということである。通常150坪の店舗の年商は3億円から5億円であるがこの西友ストア能見台店は9億円である，当初の目標は越えることができ，一応の成功を納めたと言ってよいであろうということである。しかし，投資に見合う売り上げでは決してないそうであるが。

両者の違いはどこに求めたらよいのであろうか。消費者の間で定着することのできた西友ストア能見台店はあくまで買い物をする立場での勝手のよさを第一に考えることによって成功したのであろう。失敗したスーパーストアが機械化にふみ切った理由は省力化無人化であり，西友ストアが機械化した狙いは「新技術を駆使することで従業員を単純労働から解放し，余った精力を知的労働，顧客へのサービスに向けよう」ということである。計算機付ショッピングカートも客への親切の一例であるし，ハムスライサーも機械を客の意志で扱って種類，枚数，厚さを選べる意外性と面白さを打ち出したものであった。言い換えれば，マン・マシン・インターフェイスに成功したのである。

ＩＮＳの場合にも，同様のことが言えるのではないか。この計画が，着実に進展して行くかどうかは，これからシステム化されていく階層において，ＩＮＳを基礎としたサービスにどれだけの需要を起こすことができるか，すなわちどれだけの価値あるサービスを提供できるかにかかっているだろう。これからシステム化されていく階層とは，一般事務においてこれまでＯＡ化になじまないとされていた多様で一つ一つの規模は小さい業務，および中小規模の事務所や商店，さらには一般家庭である。ハードウェア生産における技術進歩と大量生産による低コスト化によって，これらの階層においてもシステム化が進んでいくことが可能となった。また，これらの階層にもシステム化が進展して行かなければ，ネットワークとしての価値もあやぶまれるだろう。しかしこの階層において，新しいシステムが定着できるか否かは，どれだけユーザーにとって使い易いソフトウェアが開発されるかによる。現在すでにシステム化されている公務および企業活動においては，より低いコストによって旧来の業務をコンピュータが代替できたから普及してきた。しかし，これからさらにコンピュータが普及するためには，具体的な値としてコストが意識されておらず，これまで業務を必要とする人間が自ら行っていた作業に代替するサービスを提供せねばならない。利用者自身の作業に代替するためには，具体的な時間の節約や作業の容易さ，安全性，正確さ等を提供せねばならない。これらはソフトウェアの充実によってのみ可能となる。

もちろんこれまでもソフトウェアは重要であったが，ハードウェアに比べ，それほど意識されてはいなかった。ソフトウェアの増大しつつある重要性は，情報処理過程における費用のうちソフトウェアの開発，保守にかかる費用の割合が増大してきたことをみてゆくことによっても明白になる。コンピュータが姿を現し始めた1955年にはソフトウェア開発のための費用は総費用の１割，保守費用[1]が５％を占めるに過ぎなかったが，しだいに開発・保守費用は相対的な比重を増しはじめた。B. W. Boehm [1976] は1985年には開発費用が４割，保守費用が５割を占め，ハードウェアが占める費用は１割にも満たないと予測した。実際には米国でもそれほどには逆転してはいないようであるが，ソフトウェアの費用はハードウェアの費用を上回っている（B. W. Boehm〔1981〕）。辻 [1985] は日本におけるハードウェアとソフトウェア費用の比率を表１のよ

表 1 ハードウェアとソフトウェアの費用比率

	56 年	57 年	伸 び
ハードウェア	52.9%	51.4%	9%
ソフトウェア	47.1%	48.6%	16%

うに推定した。

世界のコンピュータ産業の6割強を占める巨人ＩＢＭはハードウェアの供給者であるが，1984年においてその売上高の18.5%が（保守費用を含む）ソフトウェアで占められている。しかもソフトウェアプロダクツだけをみると，売上は年々40%程度の率で上昇している[2)]。

ソフトウェア関係費用の比重が高まってきた理由はハードウェアの生産性の上昇に比べ，ソフトウェアの生産性が依然として低いことにある。ハードウェアの生産性の伸びの様子は様々に報告されている。たとえば，バイトあたりのメモリーの価格は15年で1/200まで低下し，1命令実行あたりの価格は5年で1/10のペースで，ＬＳＩの集積度は5年で8倍のペースであがってきている。次節ではソフトウェアの生産性が高まらない理由を分析する。

第2節 ソフトウェア生産の特質

ソフトウェアの生産性が高まらない理由は，言うまでもなく人手の手作業に頼らざるをえないということにある。その他にソフトウェア生産に固有の理由として，以下の諸点があげられよう。

① 生産物が個性的，属人的である。

Zelkowitz et al. によるとソフトウェアのライフサイクルにおいて開発の比重は1/3にすぎず，2/3が保守に投入される[3)]。開発した以外の人間が保守を行う時には，プログラムの解読という非常に困難な作業をせねばならない。これが，保守費用を押し上げる要因ともなる。プログラムの標準化・構造化が進み，ドキュメンテーションが充実されれば，費用は節約されると主張されているが，標準化・構造化はなかなか成功しない。またドキュメンテーションの作成には，多大な費用がかかることが知られている。

② 製品規模が大きくなると規模あたりの生産性は低下する。

ソフトウェア生産の費用曲線としては COCOMO (Constructing Cost Model) という経験式が紹介されている (B. W. Boehm [1981])。すなわち，ソフトウェア生産のために投入せねばならない費用は，

$$MM = 2.4\ (KDSI)^{1.05}$$

MM：投入労働量（単位：人×月）

KDSI：生産されるソフトウェアの命令数（単位：千）

という式で近似される。投入労働量の製品規模に対する弾力性は1.05である。すなわち，開発されるシステムが大きくなるにつれ，生産性が低下してゆく。さらに，労働投入の時間的配分は一定のパターンをとらねばならないことも知られている。効率的時間配分は Rayleigh 分布で近似される。したがって，開発に要する期間も，生産ソフトウェアの大きさによって，下限が定まる。すなわち，一定のマンパワーを投入すれば生産が可能というわけではなく，生産計画の効率的な設定が困難である。最小開発期間は，投入労働量の関数，したがってソフトウェアの規模の関数であり，

$$TDEY = 2.5\ (MM)^{0.38}$$

TDEV：開発期間（単位：月）

という式で与えられる。

③ 使用して始めて品質が分かる。

注3からわかるとおり，プログラムのテストは開発コストの半分近くを占める。しかも修正を要するプログラムの不良箇所を開発時のテストにおいて全て見つけ出すことはできない。品質には，開発時に評価できるシステム開発品質と，システム運用時に初めて評価できるシステム運用品質とがあり，開発時には前者しか評価できない[4]。

④ ソフトウェア作成における生産性の個人差が大きい

ソフトウェア生産時においては，個人の能力に，一般の生産における個人間の生産性格差に比べはるかに大きい格差が生ずる。日本情報処理協会の中山隆夫氏によれば，この差は最大で以下の水準にまで広がる。

コーディング	25：1
プログラムサイズ	5：1
デバッグ	26：1
機械使用時間	11：1
実行速度	13：1

大規模なソフトウェアは分業によって製作される，個人間の生産性があまりに異なると，協調がとりにくく全体の生産性を阻害する原因となる。また中山氏によれば個人の貢献度を評価するにあたっても，優秀な人材の処遇を難しくする（下田 [1983]）。

⑤ ソフトウェア作成のノウハウはあくまで個人に蓄積される。

生産されるプログラムはいわばノウハウの固まりである。①で挙げたソフトウェアの属人的性格とあいまって，得られたノウハウを作業者間で共有し，生産性の向上を目指すことが難しい。

⑥ 陳腐化が激しい。ソフトウェア自身の技術革新によっても，ハードウェアの技術革新によ

っても陳腐化される。

⑦ 利用の共用化と作成者の権利の保護が相反する課題となる。

第3節 ソフトウェアクライシス

第1節で示したように今後はソフトウェアの重要性がますます高まる傾向にある。しかし，第2節で示したようにソフトウェア生産における生産性には，ソフトウェア生産の性質が反映され，急な伸びを見込むことはできない。もし今後に需要されるソフトウェア製品の規模が大きくなれば，第2節②で紹介した効果により，生産性の低下さへきたすかもしれない。したがって，ソフトウェア生産においては技術者不足からくる供給の不足が最大の問題となる。

通商産業省の予測によれば，今後のわが国におけるソフトウェア技術者に対する需要は年率28%で成長し，1990年には160万人が必要となると推定されている。一方，ソフトウェア技術者は最大で年率17%の率の増加しか見込めず，1990年における供給は100万人であり，60万人の技術者が不足すると予想されている。このギャップは，ソフトウェア・クライシス (Software Crisis) を形成する[5]。ソフトウェア・クライシスとは，今後に予想されるソフトウェア技術者の絶対的な不足を指す。この現象は，1967年の NATO 科学技術委員会で報告された。この危機の自覚によって，1970年代に入るとプログラムの構造化・モジュール化等，「ソフトウェア工学」と呼ばれるさまざまな形の生産性の上昇に対する工夫が研究され始めたのである[6]。

ソフトウェアは全ての産業に対して，投入財である。したがってソフトウェア生産の危機は単に情報処理産業の問題にとどまらない。ソフトウェアの費用の占める割合が上昇している以上，ほとんど全ての財の生産性の上昇はソフトウェアの生産性に影響されてしまう。ソフトウェア生産に伴う種々の困難を前提としつつ生産性を高める工夫としては，「ソフトウェア工学」で考えるような，標準化・構造化・モジュール化等の技術革新を指向する方向がある。しかし，日本における生産の現場においては，ソフトウェア生産の特質に起因する要因以外の，経済的，産業組織的要因によって生産性上昇が阻害される傾向が大きく，それをできるかぎり見きわめ対処することによって生産性の上昇をもたらす余地があると思われる。

具体的には，以下の項目が可能であると思われる。

① ソフトウェア技術者の労働環境を整備する

② ノウハウの蓄積が最も容易な場所にノウハウを蓄積する

③ ソフトウェアを汎用化し，共用化する

それぞれの項目はそのまま日本のソフトウェア生産がかかえている問題に直結している。すなわち，①の労働環境の整備について考えると，「ソフトウェア技術者の反乱」というソフトウェア産業がかかえている特異な労働条件の問題に，②のノウハウの蓄積の問題は下請け制に関する

問題につながり，③はそのままソフトウェア汎用化の遅れの問題となってあらわれている。

以下の諸節ではそれぞれを説明し，問題点を整理する。その前に，日本においては，ソフトウェアがどこで生産されているかを概観しておこう。日本においては，ソフトウェアを使用する場においてほとんどのソフトウェアが生産される。一般の産業の局面でどの程度ソフトウェアが生産されているかを示す統計は残念ながら存在しないが，日本興業銀行の調査によればシステムエンジニアやプログラマーの分布は，情報処理産業に約5万3千人，メーカーに約3万2千人，ユーザーに約11万8千人である。情報処理産業はソフトウェア生産の約26%しか担っていない[7]。

この情報処理産業は産業分類によるとさらに，プログラムの作成を行うソフトウェア業，受託計算処理業務を中心とする情報処理サービス業，データベースサービスを中心とする情報提供サービス業，その他の情報サービスに細分される。1983年度においては，情報処理産業全体で，2,148の事業所，127,978人の従業者を数え，10,953億円を売上げている。このうちソフトウェア業は，591の事業所，48,188の従業員数，売上高3,827億円を占める。売上は年率20.8%の割合で増加している。

第4節　ソフトウェア技術者の労働環境

この問題は2つの局面を持つ。一つは非情報産業内におけるソフトウェア技術者達が遭遇している問題であり，いま一つはソフトウェア業に従事している技術者達にとっての問題である。非情報産業内すなわちソフトウェアのユーザー企業における問題とは，技術者の企業内での人事的な処遇の問題であり，ソフトウェア業内の問題とは要員派遣の問題である。

非情報産業内のソフトウェア技術者達の人事的処遇の問題は，企業内で情報部門が他の部署と協調をとりにくいことから生じる。企業は自社内の情報管理部門が肥大化することを嫌う[8]。次節に示すように日本では，ソフトウェア生産の外注比率が年々高くなっているが，なぜ下請けや外注に出すのかという要因のなかにも，コストに対する経済的効果のほかにソフト部門を増大させたくないという項目が入ってくる[9]。また，現在ＶＡＮが普及し始めているが，そのＶＡＮの普及を促進する要因となっているものの一つにも，同じ人事管理上の要因があげられよう。企業がＶＡＮに依存している部分に代えて，相当するシステムを企業内部にもつことは可能である。なぜ，企業はそのようなシステムを内部に持たないのであろうか。ＶＡＮによって企業はシステムの核心のみを自営で持ち，他の部分はＶＡＮに委託して情報管理部門の肥大化を防ぐことができるのである[10),11)]。

このような情報部門の孤立の原因の一つは日本の人事管理システムにある。日本のホワイトカラーはゼネラリストたるべく，いくつかの職種をローテーションしていく過程で昇進していく。このローテーションにおいて情報部門は特別の位置を占めてしまう。前に見たようにソフトウェ

アの開発・保守という仕事においては作業者にスキルが蓄積されることを必要とするため，このローテーションの制度とうまく噛み合わない[12]。また，会社に対する貢献度の評価においても，けっして他の部門とうまくいっているというわけではない。情報部門の業務は販売業績に直接反映することが少なく，正当な評価を求めて企業外部の場で活動することが多くなってしまう[13]。さらに，第2節で示したように，能力に個人差が大きく不満がたまりやすい。これらの結果，転社する意志を持つソフトウェア技術者の割合が高くなる。

一方，ソフトウェア業内において発生しているソフトウェア技術者の問題は，非情報産業内の問題が波及した形で生じている。非情報産業企業内の情報部門の要員不足を解決するため，企業の多くはソフトウェア業からの要員派遣と外注に依存する。次節で説明するように，現在のソフトウェア業はそうした仕事の受注によってなりたっている。ソフトウェア技術者の絶対数は不足しているので，ソフトウェア業の中には人材派遣業となっている企業も多い。「コンピュータ利用状況調査」によると回答企業1,233社の，コンピュータ要員の平均数は30.3人であるが，そのうち60.7%にあたる18.4人が被派遣要員であった。ソフトウェア要員にかぎっても52.4%が派遣された技術者であった[14]。派遣期間は1年から2年の間が最も多く（25%），5年以上派遣されている技術者も11.1%にのぼる。彼らは自社と派遣先に二重に帰属し，教育水準の不足および労働意欲の低下という問題を持っている。また，管理者の養成も難しく，若い人材の採用も難しい[15]。ソフトウェア業で初めて株式を上場したＣＳＫという会社は，「人材派遣サービス」の先駆であり，数多くのソフトウェア企業が追随するようになった「大川商法」で知られている。

外注された仕事を受注する際にも，ソフトウェア業はレベルの低い仕事を受注することが多い。なぜなら，発注する企業もシステムのアプリケーションの開発においては，ノウハウの蓄積が大切であることを気付いている。したがって，ノウハウの蓄積の効果が顕著なシステムの設計等の付加価値の高い仕事は自社内の要員を当てる。そして，ルーチンワークの性格の強いプログラムのコーディングをもっぱら外注するのである。このコーディングという仕事は，それだけに知的作業の性格が薄く，ノウハウの蓄積や作業に対する熟練の可能性も小さい[16]。この効果は企業が技術者に対して投下する教育費の差，およびプログラマーの平均給与の差にあらわれている。先の「コンピュータ利用状況調査」において，一人あたりのコンピュータ関連教育費は全産業の情報部門平均で年間26,500円なのに対し，情報処理サービス・ソフトウェア業の平均は17,700円にすぎないことが示されている。また，全産業のプログラマーの平均給与は192,500円であるのに対し，情報処理サービス・ソフトウェア業における平均給与は164,400円にすぎない。付加価値が低い仕事しか与えられないから，給与も低く，教育の必要度も低いのである。

第5節 ソフトウェア生産における外注の利用

前節の最後にソフトウェア生産における外注制の一つの問題点を指摘した。すなわち，プログラムの生産性の向上に必要なノウハウは，ソフトウェア産業において蓄積されるのではなく，ソフトウェアのユーザー企業に蓄積されるのである。これはソフトウェア産業の技術力の蓄積を著しく阻害し，独自の商品開発力などの産業としての自立性を損なってしまう。その結果またソフトウェアハウスの受注作業への依存をますます深める要因ともなる。

「コンピュータ利用状況調査」によれば，1983年度の全産業のコンピュータ部門の運用経費のうち，14.5%が外注費にあてられている。この割合は年々約20%の割合で増える傾向にある。一方，同じ時期のユーザー企業におけるコンピュータ部門要員増は約8.8%にすぎない。ユーザー企業に置けるソフトウェア開発の自給率は減少方向をたどっている[17]。

ソフトウェア産業の受注作業依存に対しては，日本のコンピュータメーカーのバンドリング政策も原因となっている。日本のメーカーはユーザーからコンピュータの契約を取る際に，ソフトウェアも一括して受注するケースが多い。しかし，メーカーにもそれだけのソフトウェア要員はいないから，ソフトウェア産業へ外注に回す場合が多いのである。情報処理振興事業協会の1984年度の調査によると，メーカーに対する売上はソフトウェア業の販売額の41.7%を占めている[18]。

日本のソフトウェア業と米国のソフトウェアハウスとの最も大きな違いは，コンピュータメーカーとの関係である。日本のソフトウェアハウスの上位50社のほとんどの企業は直接であれ，間接であれ，コンピュータメーカー6社のどれかと資本関係があるか，結びつきが強い。たとえば，富士通はソフトウェアハウス30社と結びつきが強く，この30社は富士通のユーザー向けアプリケーションプログラムの開発運用のほとんどを担当している。日本のメーカーは，大型機のハードウェアでの競争力こそついているが，ソフトウェア，サービス，サポートなどの面はまだ弱い。これらは，中小システムハウス，ソフトウェアハウスがやっている。現実にミニコンは61%がシステムハウスに再販売用として売られている（1981年）。一方，米国のソフトウェアハウスはメーカーへの対抗意識が強く独立し，専門専業化している[19]。

米国のソフトウェア産業も60年代までは軍の下請けソフトウェアハウスが乱立している状態だった。それが，1969年にＩＢＭが反トラスト法で提訴され，アンバンドリング政策をとることになって以来，70年代の半ばごろより汎用ソフトウェアの開発，販売で力をつけはじめたのである。日本のコンピュータメーカーはハードウェアだけでなくソフトウェアのすべての面倒をみる。最近さらに国産コンピュータメーカーはソフトウェア子会社の増強にしのぎを削る一方，ソフトウェアハウスの系列化による下請け企業グループの強化に競っている。

ソフトウェア産業内の企業が以上のように部分的作業の受注に依存している状態は，日本のソ

フトウェア生産の生産性向上の面からみて，決して望ましい状態ではない。発注側の企業が望むのはあくまでも徹底的なコストダウンである。もし，この制度が評価を受けるとしてもコストダウンでという側面においてあろう。コストダウンは，発注企業内での内製化および他のソフトウェア業者との競争によってもたらされよう。しかし，このコストダウンの効果はあくまで短期的な効果でしかない。

より長期的な立場にたてば，ソフトウェア産業内にノウハウがどれだけ蓄積されたかがソフトウェア生産の効率を左右する。すでにある程度技術が成熟した産業と異なり，ソフトウェア業に現在必要なのは，ごく初期的な技術の蓄積である。ソフトウェア産業は未だ成熟していない産業であり，長期的な技術の蓄積を目指すためには受注請負制は足かせになっているようである。

第6節　ソフトウェア汎用化の問題

前に見たとおり，ソフトウェアを開発，生産しているのはソフトウェア産業やコンピュータメーカーだけではなく，コンピュータのユーザーがかなりの部分を占めている。例えば昭和55年度では都道府県市区町村だけで，保有しているソフトウェアの数は45万本にもおよび，毎年約8万本の新しいソフトウェアが開発されている。そのうち79.5%が自己開発である。隣の市が開発したソフトウェアが有償利用することが，ほとんどないのである[20]。中小企業を対象にしたソフトウェアの取得方法の調査でも，メーカーやディーラーのプログラムをそのまま利用した企業の割合は，導入時で16.2%，業務拡大時で6.1%にすぎない。これに一部修正使用を含めても，導入時で37.3%業務拡大時で20.6%である。また，コンピュータ運用経費に占めるプログラム購入費の割合は0.6%にすぎない（1984）。

もし，ソフトウェアの汎用化をすすめることによって，ソフトウェアの開発投資の重複を避けることができたなら，プログラマー不足に対して，一つの解決となるであろう。表2は汎用プログラムシェアの推移と，他国との比較を表にしたものである。ソフトウェア市場の広さは日本では米国の1/5である。日本のソフトウェアの内製比率が欧米並であれば，この差はもう少し小さいであろう。さらに，米国では半数以上のプログラムが汎用プログラムであるから，わずか1割

表2　汎用プログラムのシェアとその推移

国 （年　度）	日　本 1980	日　本 1981	日　本 1982	日　本 1983	米　国 1982	欧　州 1982
ソフトウェア売上高：A	1,959 億円	2,723 億円	3,802 億円	5,421 億円	99 億ドル	39 億ドル
うち汎用プログラム：B	81	163	313	593	58	18
B／A（%）	4.1	6.0	8.2	10.9	58.6	46.2

しかない日本に比べ，その市場規模は25倍にもなる。また，日本市場における汎用プログラムのうち，売上20位以内に入っている国産プログラム位以内に入っている国産プログラムの数は6本に過ぎない。活発に流通しているのは米国製を中心とした外国製ばかりである。逆に日本で作られたソフトウェアで，世界に使われている汎用プログラムは皆無である。このように現在日本ではソフトウェアの汎用化が遅れているため，重複の多い多種多様なソフトウェアが数多く散在し，ソフトウェア危機をさらに増幅している。

日本のソフトウェア産業が汎用の普及が可能なソフトウェアを生産していないということの説明としてよく挙げられるのは，日本人の独創性の無さや，無形の技術に対して低い評価しか与えない国民性，米国に対する時間的な遅れである。このうち，米国に対しての時間的な遅れの要因は，このような標準化における問題においては無視できないかもしれない。一番先に発表されたものが，標準となった場合，後発のものがそれをつき崩すのは，性能によほどの差があるか，ないしは付帯する条件に特別な事由が無いかぎりすこぶる難しい[21]。

日本が世界に誇る鉄鋼，自動車，造船，エレクトロニクス等の業界で開発，使用されてるアプリケーションソフトウェアは一流と折り紙が付けられている(露口)。創造力がなければ，ここまではこなかったであろうし，これらの産業の進歩を支えてきたことからもそれは立証されよう。汎用ソフトウェアをわざわざ作らなかった理由は，「すでに作られたものがあるのに，無理に創造して作る必要は無い」ということにすぎないのかもしれない。しかし，「日本はただコンピュータの普及がアメリカに比べて遅れをとってしまったから，標準化をアメリカに頼らざるを得なかった」と言ってよいのであろうか。

Brooks [1975] によると，特定ユーザーが専用にプログラムを作成する場合に比べ，ソフトウェアを製品化するよう整えるだけで3倍，他のＯＳやハードウェアに対して汎用性を持たせるのにさらに3倍のコストがかかる[22]。ユーザー企業がソフトウェア開発の一部を外注する時には，コストをなるべく小さくしようという動機によることが多い。したがって，必要最小限のコストで作成するために，汎用性を極力節約した非常に狭い機能を持ったソフトウェアを外注することになる。そのため，ソフトウェア業内に汎用ソフトウェア作成のノウハウが蓄積されることは難しい。ユーザー企業の外注依存体制が，ここでも影響を及ぼしている。汎用に耐えるだけの生産力がソフトウェア産業に形成されていなければならないが，ソフトウェア業の受注依存性がそれを妨げている[23]。

日本産業の成長に大きな役割を果たしたとされている下請け制が，ソフトウェア産業においては生産物の標準化の妨げになり，それ自身の産業としての基盤の充実には役立たなかったのかもしれない。

第7節　ソフトウェア生産と下請け制

これまで日本のソフトウェア生産体制の特徴ともいえる発注受注システムにおいて，つとめて下請け制という言葉を避けてきた。下請け制が単なる外注生産と異なる点は，発注者の市場支配力である[24]。しかし，ソフトウェア市場においては常に需要超過であるから，通常の産業におけるような市場支配力に起因する問題は顕在化していないと思われる。

しかし，すでに前3節においてみてきたように，この生産体制はソフトウェア産業が産業としての生産力を増強していく際の障害となっているのではないだろうか。一般機械部品製造における下請け制においては，下請け側の技術革新によって商品の小型化軽量化が支えられ，コストダウンは完成品の国際競争力の源泉になったと評価されている。これに対し，ソフトウェア産業の発注受注システムにおいては，産業の労働環境の面から，そして固有の技術革新の立ち後れの面から問題が発生している。発生の形が，一般産業とは異なっているとはいえ，下請けという制度の持つ一つの側面であると考えられるのではないだろうか。

ところで，ソフトウェア産業をとりまく環境は非常なスピードで変化しつつある。ユーザー企業も経済財としてのソフトウェアの価値に気付きはじめ，システム部門が独立しソフトウェア業界に汎用ソフトウェアを中心に参入し始める動きが出ている[25]。また，ソフトウェア開発の時点で，ソフトウェア・パッケージとして商品化し販売することを前提とする企業も出現している[26]。こうした市場の自然なメカニズムにより，汎用プログラムがユーザー側が供給する傾向は今後ますます強くなるだろう。たしかに，一般的にすぐれたプログラムはユーザーに近いところで生産されるから，一つ一つの製品を見るかぎり優秀なプログラムをもたらすかも知れない。しかし，ソフトウェア環境のより基礎的なレベルを充実させるという意味で汎用性があり，共通の資産たりえるという意味で市場性のあるソフトウェアが生産できるようになるかどうかは疑問である。

またハードウェアが安くなり，ユーザーの身近にパソコン・ＴＳＳ端末・オンライン端末が普及したことによって，手軽に小規模のプログラムを作成し，使用する階層が出現している。情報処理の分散化にともなう，ソフトウェア生産の分散化である[27]。さらに，メインフレームの時代はほぼ終わった言われ，コンピュータの購入価格が安くなるにともない，情報処理部門の担当者の役割も小さくなり，前記の分散化を支援する方向に変わってきている。これらの動きによって，第4節で分析したようなユーザー企業における問題は自然に解消されていくかもしれない[28]。

さて，ソフトウェア産業がこれから発展を遂げて行く為には，ある程度ベーシックなレベルでの標準化を，どこかの時点で行なっていく必要がある。自由競争でこの標準化を行なうことも考えられる。しかし，自由競争の万能性がこの分野で発揮されるとは考えにくい。なぜなら，先にも挙げたように，一度あるシステムを導入してしまったら新しいシステムを導入するためにはか

なりのコストが掛かる。そのため，競争の結果最後に残ったものが最適なものとは限らない。たしかに，多少の時間でも早く発表できたということによる社会的便益まで含めると，残った製品を最適なものとみなせるという考え方もできる。それにしても，そのシステムを開発した者の手に渡る開発料は標準システムとなった時点で膨大なものとなるだろうが，はたしてそれが妥当な開発利益と一致するであろうか。しかし，なんらかの利益の保証がないかぎり標準となるようなシステムを開発することは難しいだろう。この開発は，巨大企業のみが可能である。

注

1) ここでの保守費用はソフトウェアに対する保守費用である。B. W. Boehm [1981] は保守費用の分布を以下のように推定している。

41.8%	ユーザーのための拡張	6.2%	ハードウェア，ＯＳの変更
17.4%	データ，ファイルの変更	5.5%	ドキュメントの更新
12.4%	事故対応	4.0%	コードの改善
9.3%	デバッグ	3.4%	その他

2) ＩＢＭの製品別売上高構成は1984年において以下のとおりである（栗田 [1985]）

25.7%	プロセッサ	11.5%	プログラムプロダクト
25.5%	周辺機器	7.5%	保守サービス
21.4%	ワークステーション，オフィスシステム		

3) ソフトウェアのライフサイクルにおける費用の分布は Zelkowitz et al. によると以下のとおりである。注6も参照されたい。

開発	33%	保守	67%
要求分析	3%		
仕様決定	3%		
設計	5%		
コーディング	7%		
モジュールテスト	8%		
結合テスト	7%		

4) 風間 [1985] による。システム開発時品質とシステム運用時品質は以下のように定義されている。

$$\text{システム開発時品質}=\frac{\text{発生不良数}}{\text{システム規模}}$$

$$\text{システム運用時品質}=\frac{\text{累積運用発生不良件数}}{\text{システム規模}\times\text{期間}}$$

5) 『コンピュータ白書』[1985]

6) プログラムの構造化とは，プログラムを作成する際に構造が見通せるような明快な書き方をすることであり，モジュール化とはプログラムの各部分を独立した部品化し，汎用性を持たせるとともに保守を容易にする工夫である。

7) 興銀調査 [1985]

8) この部分の現状は下田 [1983] に詳しい。

9) 山田 [1985]

10) 青井 [1984]

11) 例えば，花王石鹸は以前，独自のネットワークシステムを運用し，パッケージを他社にも販売してい

ることで有名だった。しかし，日本経済新聞，「花王石鹸の情報ネットワーク再編」(昭和59年6月8日)によると，システムのほとんどの部分をＶＡＮに依存することになった。その理由は，核となる販社情報ネットワークにおいて，発注データの催促や，土日の処理，ハード・ソフトウェアの能力の限界など保守管理がけっこう面倒となったので，基本的な運営機能は花王が握ったまま外部移管することによって30%以上のコスト削減が期待できること，そして限られたコンピュータ要員を他へ振り向ける余裕を作ることにあったということである。

12) 古村 [1985] によると「丸井では前者員が小売業におけるゼネラリストとしての知識と経験を経て，個々の適性にあった職種を担当することをベースとしている。コンピュータのプログラマーがキャリアパスに組み込まれることよりは，全社員が回収業務を経験することがはるかに優先度が高いという考え方をしている」

13) 下田 [1983]

14) 『コンピュータ白書』，第2章，「コンピュータ利用状況調査」

15) 下田 [1983]

16) コーディングという仕事は，「顧客が用意する設計図に従ってその処理手順を機械が解読できるような独特の記号に直すだけの次元の低い仕事である。決して「責任や誇りのもてる仕事ではない」(下田)。

17) 辻 [1985]

18) 北原・青木 [1984]

19) W. H. Davidson [1984]

20) 露口 [1983]

21) 性能に差があっても標準となれない例としては，8ビットマイクロコンピュータチップのシェア争いの例がある。8ビットマイクロコンピュータチップとして最初に登場したのは1974年のインテル8080である。このチップはもともと電卓用に設計されたため，マイクロコンピュータシステム用としてはあまり満足できるものではなかった。しかし，パーソナルコンピュータに登載されはじめてから爆発的に普及し，12年たった現在でも広く使われている（上方互換の Z80 を含む)。現在このチップが使用される場合に，その理由を尋ねられたらほとんどの人はただ単に良く知っているからと答えるだろう。その後1978年にモトローラ社から6809というチップが発表されている。おそらく8080を普段用いている人も6809の性能の方が数段上であることを認めるであろう。しかし，現在も6809は8080（Z80）程には普及していない。

22) 辻 [1985]

23) 汎用化・標準化を妨げているもう一つの要因としては，ＩＢＭのような巨人の不在もあげられるだろう。ＩＢＭ自身が米国内での汎用ソフトウェア供給のトップを占めている。さらになによりも，大型コンピュータの汎用ソフトウェアの8割がＩＢＭ製メインフレーム用である。米国のメインフレーム市場の63.8%をＩＢＭが占めていることを反映した数字である。一方，日本のメインフレーム市場のトップは日立であるが，シェアは32%にすぎない。

24) 植草 [1982]

25) 辻 [1985]

26) 高橋 [1985]

27) 津田 [1985] によれば，

オフィスの非定型業務に関しては，工場労働における熟練と同様に，日常事務処理とは異なり，そのソフトウェアを使用する人とプログラムを作る人とが同一でないと完全なものができないのであり，30歳代，40歳代の業務経験者が自身でＯＡ化を推進するという分散処理システムを必要としている。意志決定支援システムはこの方式で形成されるのであり，ここでは若年層ではなく中高年齢層のソフト業務への習熟という新しい労働の質の獲得が決め手になっている。これはオフィス労働の現在

のステップの特徴といってよいであろう。(p. 277)

28) 企業の電算機部門はシステムの拡張期に，処理効率と安全性を高めるため各部門の要求に答え，部外者のアクセスを拒絶した。この行動が，ＥＤＰ部門の孤立化を進めた面もある。しかし，分散処理の技術が進んだ今日ではその必要もなくなっている（根岸・梅沢 [1985]）。

文　　献

青井浩也 [1984],『ＶＡＮとは何か』, 日本経済新聞社

Boehm, B. W. [1976], "Software Engineering," *IEETrans. Computers*, Dec. 1976, pp. 1228—1241.

Boehm, B. W. [1981], *Software Engineering Economics*, Princeton-Hall.

Brooks Jr., F. P. [1975], *The Mythical Man-Month*, Addison-Wesley.

Davidson, W. H. [1984], *The Amazing Race*, John Wiley & Sons. Inc. (筑紫哲也監訳,『コンピュータウォーズ』, ダイヤモンド社, 1984年)

古村宏 [1985],「ソフトウェア開発外注事例Ｍ＆Ｃ 方式の特徴」,『データマネジメント』, 1985年４月

風間好尚 [1985],「ＮＥＣにおけるソフトウェアの評価見積方法」,『データマネジメント』, 1985年４月

北原正夫　青木良三 [1984],『コンピュータ業界』, 教育社

興銀調査 [1985] No. 224,「わが国情報処理産業の現状と問題点」, 日本興業銀行

栗田昭平 [1985],「超野心的企業に変身したＩＢＭ」,『bit』, 1985年７月

根岸正光　梅沢豊 [1985],「わが国企業におけるＯＡ化の動向」,（若杉・岡本編『技術革新と企業行動』, 東京大学出版会）

日本情報処理開発協会編,『コンピュータ白書』, コンピュータ・エージ社, 1985年２月

下田博次 [1983],『ソフト技術者の反乱』, 日本経済新聞社

高橋徹 [1985],「三井造船の開発したソフトウェア製品販売事例」,『データマネジメント』, 1985年４月

津田真澄 [1985],「現代の技術革新と人事労務」, 若杉・岡本編『技術革新と企業行動』, 東京大学出版会

辻淳二 [1985],『情報処理産業界』, 教育社

露口広 [1983],『ソフトウェア産業』, 日刊工業新聞社

山田純造 [1985],「ソフトウェア開発外注の現状と問題点」,『データマネジメント』, 1985年４月

植草益 [1982],『産業組織論』, 筑摩書房

Zelkowitz et al, *Principles of Software Engineering and Design*, Princeton Hall.