解　説

ソフトウェアマネジメント概説†

東　基衛††　細谷僚一†††　高橋宗雄††††

1. はじめに

日本の産業の競争力の源泉として，高品質の製品を低いコストで生み出す生産技術，管理技術が指摘されている．ソフトウェアの品質の重要性が注目されるようになって，ソフトウェアの分野でも日本の管理技術にアメリカやヨーロッパから高い関心が寄せられている．

ソフトウェアマネジメントは，ソフトウェア工学，経営工学及び人間要素工学 (Human Factors Engineering：組織論，行動科学，人間工学など) などの境界領域に属すると考えられるが，専門の研究者はまだ少なく，また発表される論文も各種の専門誌に分散されるためその実態はなかなか分かりにくい．

ソフトウェアマネジメントに関する最初のまとまった文献としては，リーファ (D. J. Reifer) のチュートリアル「ソフトウェアマネジメント」[1] をあげることができる．その後ベーム (B. W. Boehm) のコストモデルに関する名著「Software Engineering Economics」[2] が発行され注目を集めた．しかし最近の文献は，日本，海外共に品質管理に関するものが圧倒的に多い．

ソフトウェアマネジメントの体系に関する文献はほとんど見あたらない．しかし，なんらかの体系化を行うことは，この分野の今後の発展のためにも重要なことと思われる．そこで本稿では，ソフトウェアマネジメントの体系化を試みるとともに，本特集の他のところでふれられていない人間要素，組織，標準化及び外注管理について述べる．

† Survey on Software Management by Motoei AZUMA (Waseda University), Ryoichi HOSOYA (NTT) and Muneo TAKAHASHI (Toin University of Yokohama).
†† 早稲田大学理工学部工業経営学科
††† NTT 情報通信網研究所
†††† 桐蔭学園横浜大学工学部制御システム工学科

2. ソフトウェアマネジメントの概念

2.1 ソフトウェアマネジメントの目的

ソフトウェアマネジメントは，ソフトウェアに関する開発，保守などの作業を効率よく行うための管理技術であるといえる．ドラッカ (P. F. Drucker) は，彼の著「マネジメント」[3] においてマネジメントの課題として次の三つをあげている．

（1） 組織の目的と使命を果たす．

（2） 仕事の生産性を上げ，働く人間をして何事かを達成させる．

（3） 社会的インパクトと社会的責任の管理．

これらの課題は，ソフトウェアマネジメントに対しても適切であるといえよう．すなわち，ソフトウェアマネジメントの目的は，直接的には，

（a） 必要な時期までに，与えられた資源を有効に使用して，必要かつ十分な品質のソフトウェア製品の開発及び保守を行う．

すなわち，原価管理，納期管理及び品質管理であるということができる．

しかし，これだけでは十分ではない．一般にソフトウェアの部門は，一部門，たとえばソフトウェア部，の中に多くのプロジェクトが並行して進行していることが多い．そこでマネジメントとしては，単なるプロジェクト管理以上に要員の育成や，満足感に留意しなければならない．

そこで第二の目的として，

（b） ソフトウェア技術者が，仕事を通じて向上し，満足感が得られるように，生産性，品質向上の目標を自ら設定し，その達成に向かって自主的に努力してゆくように導くとともに，その環境を整備する．

をあげることができる．

さらに，第三の目的としては，

（c） ソフトウェア製品の与える社会的なイン

パクトを配慮し，その結果に責任をもって対処する．

ことがあげられよう．

2.2 ソフトウェアマネジメント発展の経緯

管理は古代エジプトでピラミッドの建設が行われた昔から，戦争や建設など，多くの人が集まって何かを成し遂げようとするときに必要とされてきた．工業における管理の歴史はこれらよりはずっと遅く，産業革命以後のことであるといっても過言ではない．しかし，管理に関する技術の研究がされるようになったのはそれよりもさらに遅く，1911年に出版されたテーラー (F. W. Tayler) の「科学的管理法」が最初であるとされている．その後ギルブレス (Gilbreth) の動作時間研究やメイヨー (Mayo) の行動科学，フォード (H. Ford) の生産方式，これにシューハート (Shewhart) の統計的品質管理なども加わって経営工学として結実する．経営工学は，一般的には Industrial Engineering の訳とされるが，類似の分野に Industrial Management 及び Busniess などがあげられる．

ソフトウェアの開発は，当初は規模が小さかったこともあって，プログラミングとデバッグの2工程しかなかった．その後，大規模リアルタイムシステムの急速な発展により，一つのソフトウェアの開発に多勢の技術者の参加が必要になるとともに，工程も，設計やシステムテストなど多くの工程に分けて作業を進めることが必要になった．このためプロジェクト制が普及し，プロジェクト管理技術の発展をみた．

一方コンピュータの急速な普及は，深刻な技術者の不足を惹き起こし，生産性の向上が重要な課題となった．この結果標準化や生産技術が注目されるようになった．また，重要な分野でのコンピュータの利用は，品質管理への関心となって現われた．これらの管理問題は，管理技術へのニーズとなり，経営工学の分野での成果の応用が試みられた．これらは，クスマノ (Cusmano) 氏が Japan's Software Factories[4] という図書に示したように，一般にソフトウェア工場アプローチといわれる．

アメリカで始まったプロジェクト管理技術と日本で始まったソフトウェア工場制による品質管理，及び標準化と再利用可能部品の利用を中心とする生産管理などは，ソフトウェアマネジメントを代表する主要な技術といえよう．

2.3 ソフトウェアマネジメント技術の目標とアプローチ

ソフトウェアマネジメントの技術目標は，図-1のように，標準化及び制度化，自動化及び効率化，ならびに計量化及び可視化の三つの軸で現わすことができる．しかし，これらは常にソフトウェア技術者がこれらをどう受けとめるか，また技術者の能力をどう発揮させ，向上させるかという人間要素を考慮しながら実現しなければならない．これらを統合したのがシステム化であり，最終目標といってもよい．

標準化は，コンピュータに限らず全ての工業製品の生産技術にとって重要な手段である．標準化には大別して，部品の製品の標準化と，作業方法や作業手順などのようなプロセスの標準化がある．ソフトウェアの場合，製品の標準化は，製品自体より，インタフェースのように製品の規格に関するものであり，また，部品の標準化は再利用による生産性の向上を目指すために重要視される．一方作業標準は伝統的な経営工学の中心的な課題の一つとされてきたが，知的活動であるソフトウェア開発の場合には，文書化や，ソフトウェアツールと結び付けて行われることが多い．これらの各標準は互いに関連をもっているため，独立して推進してもそれほど効果は大きくないため，統合的な標準化が推進されている．1970年代の終わりまでには，コンピュータメーカ各社ともに統合的な標準化を支援するに至っている．

標準化には，その影響範囲によって，国際標準，国内標準，企業内標準及びプロジェクト内標準などがある．上位レベルの標準が存在する場合には下位レベルの標準はそれに準拠するように設定するべきであるのはいうまでもない．筆者の関

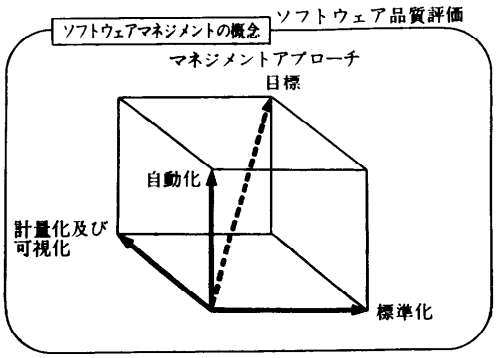


図-1 ソフトウェアマネジメントの概念

与する ISO/IEC JTC 1/SC 7 では，Software Engineering の標準化を担当しているが，最近では品質管理，構成管理など，ソフトウェアマネジメントの標準化に関する新規提案が多い．

第二の目標は，自動化とそれに基づく効率化である．管理の重要な原則の一つに例外管理があるといわれる．これは，管理のための意思決定の規則がはっきりしているものはできるだけ下位のレベルの管理者，または担当者自身による自主管理にまかせ，本人は例外的な事項の意思決定に専念することを意味する．ソフトウェア開発の自動化の分野では CASE ツールが注目を集めるなど，多くの成果が得られているが，ソフトウェアマネジメントの分野ではまだ市場にでているものは少ない．これは，実際には遅れているというよりは，他のアプリケーション分野の情報システムと同様に企業固有の特性に依存すること，及び汎用性のある分野については，データベース，スプレッドシートなど汎用のソフトウェアが利用できることによるものといえよう．

第三の目標は，可視化と計量化である．マネジメントレポートは，一般に1ないし2ページにとどめるべきであるといわれる．忙しい管理者にとって分厚い報告書は時間の浪費である．状況を的確に把握し，意思決定の結果を予測することによって適切な意思決定を行うためには，作業量，品質など管理対象項目をできるかぎり計量化し，またグラフなどを用いて一目で状況が把握できるようにすることが重要である．ソフトウェアの管理項目を計量化する技術は，ソフトウェアメトリックスといわれ，最近研究が盛んな分野であるが，まだまだ未解決の部分のほうが多いフロンティアであるといえよう．

人間要素は，ソフトウェア技術者の能力に関するものであり，上記3目標と深く関係する．たとえば過度の制度化は，創造性や改善工夫の芽を摘む恐れがあるし，また，管理ツールによるソフトウェア管理の自動化は，管理者による監視強化や干渉と捉え管理者とソフトウェア技術者の間の信頼関係を損なう恐れがある．

3. ソフトウェアマネジメントの技術分野と枠組み

ソフトウェアマネジメントは，**図-2** のようなシステムモデルで表わすことができる．すなわち，ソフトウェア開発，保守などのソフトウェアエンジニアリングプロセスは，ソフトウェア部品，仕様など前プロセスの製品，資源及びソフトウェア技術などを入力し，中間製品または最終製品を出力する一連のテクニカルプロセスからなるマンマシンシステムであると考えられる．

ソフトウェアマネジメントは，テクニカルプロセスまたはその入出力製品に関する情報を入力し，制御信号または管理指令情報を出力するマンマシンシステムであるといえよう．

ソフトウェアマネジメントは，管理対象によっても識別することができる．管理対象としての資源は，いわゆる 4M すなわち，Man, Money, Machine 及び Material である．前3者は，それぞれ要員管理，原価管理，設備（Workstation など）管理に対応するが，Material はソフトウェアでは情報に置き換えて文書（仕様書）管理，及び再利用可能なソフトウェア部品の管理がこれに対応するものと考えられる．またプロセスの管理には，工程管理，日程管理，納期管理などが対応する．製品，中間製品の管理に対応するのは品質管

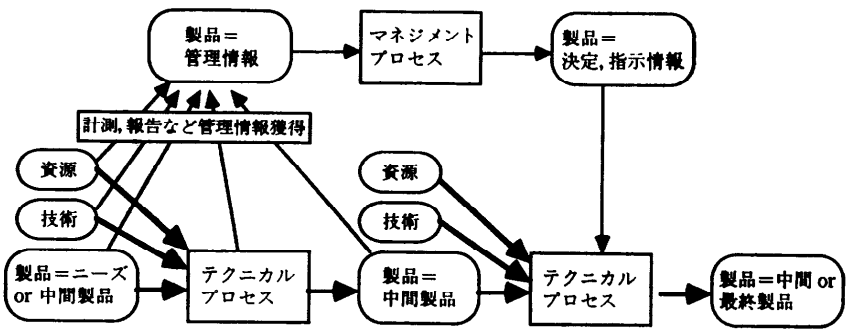


図-2 ソフトウェアマネジメントプロセスモデル

理及び構成管理であるといってもよいだろう.

以上に述べた枠組みと,前述の目標を組み合わせるとソフトウェアマネジメントの技術の枠組みをより詳細化することができる.たとえば標準化と組み合わせて標準原価設定,原価管理自動化,原価計量化(コストメトリックス)などがその技術分野の枠組みとなる.

しかしながら,これらの枠の全てが均等の重みをもっていたり,技術として均等に発展しているわけではない.したがって本特集では,現在比較的多くの関心が寄せられ,研究開発が行われている分野を中心に整理してみた.

4. ソフトウェアマネジメントの実際

4.1 人間要素

ソフトウェアは主に人間によって作成される.したがって,ソフトウェアマネジメントの諸問題を解決するためには,その人間的側面を解明することが必須である.しかし,本質的には人間そのものを対象としているため,成果が見えにくく研究がなかなか進まないのが現状である.

(1) 心理学的側面

大規模システムを効率よく作成するためのソフトウェア工学的な立場からは,ソフトウェアは開発グループ全体に属していて個々の担当者に属するものではない,という考え方が望ましい.しかし,グループの構成員である意志をもった人間の立場からみると,担当者個人の作成部分がシステムに埋没してしまうと達成感がなくなり,やりがいが得られないことになる.人間のこのような心理的影響は個人や集団の行動特性として現われ,一般に人間要素と呼ばれる.

ソフトウェアプロセスの測定における担当者の心理的影響も人間要素の典型的な問題の一つである.すなわち,プロセスを測定するというバイアスがかかることによって,担当者はそれを意識した行動をとるため,測定値がくるってしまい役に立つデータを得ることが難しくなる.このことから,ソフトウェア開発における各種の測定は無意味なものと思われがちである.

このような人間要素に関する問題は,人文科学や社会科学においては古くから研究されているが,ソフトウェアの分野ではまだ蓄積が少ない.ソフトウェア開発における人間要素の研究としては,ワインバーグ (G. M. Weinberg) の研究がよく知られている[5].彼は,ソフトウェア開発現場での日常的な人間のふるまいを観察し,そこから一般的な原則や教訓を引き出そうと試みた.たとえば,プログラマは正式のドキュメントのほかに,インフォーマルな場での情報交換で意外に多くの知識を得ているという観察から,もっとオープンにコミュニケーションできるような環境を用意することが望ましい,という経験則を導いている.

ワインバーグの研究はいまでもしばしば引用されており,ソフトウェアにおける人の問題を扱った古典である.以後,人間要素に関する理解不足がソフトウェア開発のボトルネックになっているという認識のもとに人間要素の研究が盛んになりソフトウェアマネジメントやソフトウェア開発環境との関係が見直されている.これらの研究の変遷は,シュナイダーマン (B. Schneiderman) のサーベイから知ることができる[6].

(2) ソフトウェア技術者の教育

ソフトウェア需要は西暦2000年には約34兆円にも達し,このソフトウェア開発に必要なソフトウェア技術者は,ソフトウェアの生産性が現状のままであれば,約100万人不足するといわれている[7].ソフトウェア生産人口の増大のためには,当然プログラマ数の増加が必要であるが,単に数を増やすだけでは不十分である.ソフトウェア技術者には最低限の素質が必要であると同時に,業務知識,システム知識,ソフトウェア知識が必要である.これら三つの知識を全てもった専門家は少なく,二つ以上に通じた専門家をシステムエンジニア (SE) と呼んでいる.

ソフトウェアの需要増大にともない,プロジェクトが直面している問題に対して,それぞれの専門的知見で処理できる SE の育成強化がますます重要となるが,これからの SE は従来とは違った能力が求められる.従来は,既存業務の機械化を中心とする合理化システムの構築が課題であり,そこで必要な SE の能力は,コンピュータ技術の知識と対象業務の知識だけでよかった.これに対して,今後はコンピュータと通信ネットワークを用いた戦略情報システム (SIS: Strategic Information System) の構築に重点が移ると予想されるため,コンピュータと通信ネットワークを用いた

新しい業務を創造する能力が求められる．

SE を取り巻く状況がこのように変化するに従い，次の教育が必要となる．一つは，従来の文系理工系の枠を超えた情報システム工学の教育である．これは，組織と情報の関係をシステム論的に捉えて教育するものである．もう一つは，社会人のための大学院教育である．これは，一度企業にでて実務を経験した人を対象として教育するものであり，社会の変化に即応した教育が期待できる．同時に，いったん実務を経験して問題意識を明確にした上で再教育を受けることは，新しい技法やコンセプトの創出能力を養う上でも効果的である．最近，このような認識の下に，総合技術研修所（企業内大学院）を設立している企業が見受けられる．また，イギリスやドイツなどでは工科系の場合，企業で1～2年の経験を経て大学院に入ることが奨励されている．

次に，大学と企業における教育の役割について考えてみよう．従来，日本の企業は大学教育にほとんど期待してこなかった．特に情報系企業はこの傾向が強く，学生に対してポテンシャルを求めるだけで，スキルは自社内で教育するというのが人材採用の一般的傾向である．しかし，社会の高度情報化が急速に進行するにともない，将来の情報産業に対する危機感が強く意識されるようになり，大学などにおける情報系専門教育の改善が提案されるなど，その様相に変化のきざしもみえる[8]．

これからの大学教育では，情報化社会の成熟を見通し，そこで活躍する指導的立場の人材の育成が責務となる．また，情報工学以外の専攻の学生に対する情報工学の基本についての教育も今後重要となる．いわゆる情報リテラシの基礎教育である．一方，企業では基礎理論よりも社会の変化や実現技術の進歩に追従することに重点をおいた実務型の教育カリキュラムを実施することが望まれる．教育方法についても，知識重視の教育より問題発見能力と解決能力の教育を重視すべきである．すでに一部の企業では，これらの能力を養うため，各人別の教育カリキュラムを設定し，OJT を通して実践を行っているところがある．

（3）　ソフトウェア技術者の適性と診断

プログラミングの個人差に端を発する適性の問題も人間的要素の一つである．一般に適性とは職種と資質の組合せで定義される概念である．資質は知識とは別のものであり，知識の活用・獲得能力である．知識の習得は教育で効果を上げることができるが，資質は先天的なところがあり，研究も未発達である．今後，認知科学的視点からの解明が望まれる．

プログラミングにおける個人差の例としては，表-1 の実験データがよく知られている[9]．これによると，個人差はコーディング・デバッグ時間で平均24倍の開きがある．このような開きはソフトウェアに対する適性が大きく影響している．ソフトウェアに対する適性とは何か，ということは明確には分かっていないが，個人の資質が問題，機械，人間などの対象によって異なって発揮されることに起因するのではないかと考えられている．浅井らはこのことに着目して，ソフトウェア技術者の適性は単に資質だけでなく，個人が自己の資質をどのように発揮するかについても考慮し，多面的に捉えなければならないとしている[10]．ソフトウェア技術者に必要とされる資質については，これまでに分析力，表現力・調整力などいくつかの要因があげられている（表-2）．

ソフトウェアに対する適性には個人差があるので，要員を適切に選抜するためには，個人差を評価するためのなんらかの手段が必要である．このための手法が適性検査である．ソフトウェア技術者の適性検査については，IBM 方式をはじめとして多くの検査法が開発されている．しかし，これらはプログラマの適性検査に関するものであり，SE の適性検査に利用できるものはほとんどない．最近，SE の育成強化の観点から，SE 向けの適性検査の必要性が主張され始めている[11]．

4.2　組織とチーム

ソフトウェア開発を効率的に進めるためには，個々の構成員の責任と担務を明確にしたプロジェクト組織が必須である．チームは組織の最小単位であり，大規模プロジェクトではその中に，分析

表-1　プログラミング効率の個人差の例（一部）

項目 ＼ 測定値		最低	最高	比
デバッグ時間	代数問題	170	6	28 : 1
	迷路問題	26	1	26 : 1
コーディング時間	代数問題	111	7	16 : 1
	迷路問題	50	2	25 : 1

表-2 職種・対象別資質（例）

職種	対象	資質
プログラマ	機械	判断力 記憶力 計算力 分析力 想像力 統合化力 表現力
S E	機械	分析力 想像力 統合化力 表現力
	人間	理解力 調整力 説得力
	業務	思考力 計画力 組織力 管理力 評価力

チーム，プログラミングチーム，テストチームなどの多数のチームを包含している．

（1） プロジェクト組織

プロジェクト組織として，ソフトウェア工場と最近注目されているクリーンルーム方式について述べる．

座席予約システム，バンキングシステム，電子交換システムなどで用いられるソフトウェアは大規模なものが多く，これを多数の人間で分担して作成する場合の方法論は，小さなプログラムを前提としたそれに比べて，管理的側面を重視したものが要求される．この管理的側面として，以下のものがある．

第1は，その組織に最も適した方法論を過去の経験に基づいて確立することである．第2は，この方法論をプロジェクトのメンバ全員に履行させるようにすることである．同じ方法論を用いることによって，開発者以外の人でもそのソフトウェアを理解することができる．これがソフトウェア生産の工業化の第一歩である．第3は，これらの方法論を普及・改善するような支援組織・管理体制を整備することである．特に，組織は自らの方法論の改善活動を継続することがきわめて重要である．以上の三つの項目を満たした開発環境で大規模ソフトウェアを生産する組織をソフトウェア工場[4),12)]という．

ソフトウェア工場の標準的なパターンを図-3に示す．ソフトウェア工場では複数のプロジェクトが並行して進められる．おのおののプロジェクトは要求定義から保守にいたる各工程を実施する．これは，ハードウェアに例えれば生産ラインに相当する．それぞれのプロジェクトに対しては，マトリックス方式と呼ばれる管理体制が望ましいとされている．マトリックス方式とは，プロジェクトに属する担当者を縦と横からマトリックスで管理する方式である[13)]．縦のラインはプロジェクトマネージャの権限によって支配され，担当者はプロジェクトの遂行に関して，これに従わなければならない．一方，担当者はおのおの専門の技術をもっており，その技術の内容について，自己の専門性が横のラインによって支配される．

一般に組織というものは，縦のラインは強力であるが，横のラインは結びつきが弱い傾向にある．しかし，ソフトウェア生産に必要な専門技術が適切に分類・蓄積され，マトリックスの横のラインがしっかりしていることは，ソフトウェア生産の工業化にとってきわめて重要なことである．このマトリックス方式を実際のソフトウェア開発でいかに運営していくかがソフトウェアマネジメントの最も重要なポイントである．

クリーンルーム方式は，ソフトウェア開発の現場で行われる作業を形式的記述法を用いて，できるだけ厳密に規定することにより，ウォータフォールモデルの非厳密性に起因する問題を解決しようとするものである．クリーンルームという言葉は元来，大規模集積回路の製造でよく知られて

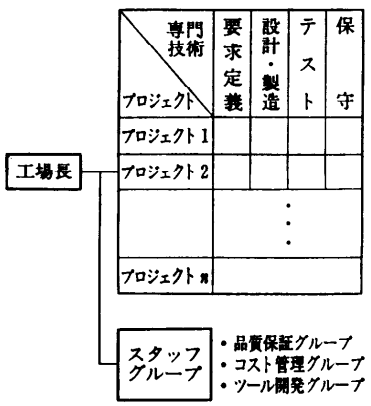


図-3 ソフトウェア工場の例

いる無塵室のことである．ミルズ (H. D. Mills) は，この大規模集積回路の製造工程の不良とソフトウェア開発におけるエラーとの類似性に着目し，ソフトウェア開発においても，よく管理された標準的な方法で作業することが人為的誤りを少なくすることができるという考え方からクリーンルーム方式を提案した[14]．

クリーンルーム方式は，基本的にはウォータフォールモデルの枠組みと同じであるが，次の2点がウォータフォールモデルと相違する特徴である．

一つは，要素定義からコーディングにいたるまでの各工程において，作業の完了を確認するためのドキュメントあるいはソースコードの査読があることである．この査読は，従来のレビュに比べて厳密性が要求され，形式的な仕様・設計記述法によって書かれたドキュメントを対象として第三者が検証することによって行われる．

もう一つは，コーディング工程でのプログラマによるデバッグが禁止されていることである．コーディング工程でデバッグがないため，査読されたソースコードはただちにテスト工程に移される．これは，一見危険に思えるが，それまでの各工程で査読が確実に行われていれば，ソースコードにエラーが残存している確率はきわめて小さいので問題はない．むしろ，プログラマに心理的緊張感を与えて誤りを防止する，という副次的効果が期待できる．

クリーンルーム方式は，高信頼性が要求されるソフトウェアの開発に適している．しかし，この方式は，要求定義からコーディングまでの各工程で査読を確実に実施することがキーポイントであるため，形式的な仕様・設計記述の困難な分野では効果が期待できないという見方がある．また，この方式は査読者の力量に大きく依存するため，査読のできる人材の確保も問題となる．いずれにしろ，まだ使用実績がそれほどなく，今後の評価を経ないとその有効性については論じられない．

（2） チーム編成

ソフトウェア開発の作業は，複雑な相互関係をもつシステム的な作業であり，プロジェクトメンバ間のコミュニケーションの労力は多大なものとなる．実際，大規模システムの多くの開発経験から，多人数の人手だけに頼ったプロジェクトの進め方はコストが高くつき効率的でない．プロジェクトメンバ相互のコミュニケーションを複雑にしないためには，小人数のチームを組んで仕事をすることが効果的であるといわれており，チームをいかに編成するかがソフトウェアマネジメントにとって重要である．

最初にチームの概念として登場したのは，IBM社のチーフ・プログラマ・チームである[15]．それ以降，他にもいくつかのチーム編成が提案され，チーム編成が品質・生産性に貢献することが認識されるようになった．日本では，古くから日本的管理社会に整合する階層的チーム編成を採っているところが多い．これは，一人のチームリーダの下に数人の担当者がいるという構成である．

チーフ・プログラマ・チームは，トップダウンプログラミングを組織的にチームメンバ間で展開するため，ベーカー (F. T. Baker) らによって提案されたチーム編成である．これは上級の専門家によって率いられたチームであり，チーフ・プログラマ，バックアップ・プログラマ，ライブラリアン及び数人の補助プログラマから構成される．

チーフ・プログラマはチームを管理し，要求定義，設計，重要なモジュールのコーディングなど全ての技術的な面にわたって，創造的及び意志決定的な作業を行う．バックアップ・プログラマはチーフ・プログラマと同等の能力を有し，チーフ・プログラマを補佐する．技術的な補佐に対して，事務的な補佐をする担当者も必要であり，この役目を果たすのがライブラリアンである．補助プログラマはチーフ・プログラマの手足となって働き，個々のモジュールのコーディングとテストを主な作業とする．

この方式は説得性があり，IBM社が担当したニューヨークタイムズ社のソフトウェア開発では実際に効果のあったことが報告されている．しかし，その後この方式については問題点がいくつか指摘され，批判的な意見も多い[16]．たとえば，大規模なプロジェクトに適用しようとしてメンバをさらに加えると，小人数であることによって可能であったメンバ相互のコミュニケーションの単純さが失われ，従来のやり方に戻ってしまう危険性がある．この問題は，チームの数を増やせば解決できそうに思えるが，チーフ・プログラマになれる人材が少なく，現実にはチームの数を増やすこ

とは困難な場合が多い．また，補助プログラマにとっては，いつも下請け専門では士気が上がらず，人材の浪費にもなりかねない．

チーフ・プログラマ・チームのような管理的・集中形のチーム構造に対して，これの分業化を進めた専門家チーム[17]とかエゴレスチーム[4]なども提案されている．これらは，チーフ・プログラマの人材不足や補助プログラマの不満を解消することをねらいとしたものである．

4.3 管理の標準化

ソフトウェア企業にとって，ユーザの満足が得られるソフトウェアを長期的に供給することが，その企業を発展させるために不可欠である．このためには，ソフトウェアそのものの品質だけでなく，それを生産するソフトウェアプロセスの品質（組織の技術力）についても，将来を見通して新しい技術の導入や要員育成などを継続的に実施し，改善していく必要がある．

しかし，既存のプロセスに新しい技術を導入することは，それの有効性が定着していないため現場サイドの抵抗が大きく簡単ではない．このことは，これまでソフトウェア工学の成果を採り入れた新しい手法が，必ずしもソフトウェア開発の現場で有効に利用されていないことからも理解できる．実務的な観点からは，ソフトウェア開発を一連のプロセスとみなして，その改善策を検討するアプローチが有効であり確実性が高い．すなわち，プロセスの入力である開発工数などの資源，出力であるプログラムなどの生産物，パラメータであるプロセス固有の環境特性を定量的に分析することにより，開発技法・ツールの効果や問題点が客観的に明らかになり，現場サイドに対し説得力をもつことができる．

このアプローチは最近，ハンフリー（W.S. Humphrey）によって提案され，注目を集めているプロセスの成熟度の概念[18]に従えば，レベル5の「最適化プロセス」の状態と一致している．レベル5の状態では，定量的なプロセス管理を支援するため，データが収集・分析され，これに基づいてプロセスの改善が継続的に実施される．このようなフィードバックループを維持するためには，プロセスの実行を通して得られるデータを標準化し，データベースに集中的に管理して，組織全体で分析する活動が必要である．これらの活動は通常，組織に設けられた管理グループあるいは委員会組織で実施され，傘下のプロジェクトに組織としての標準や管理指標のガイドラインを提供することにより，管理の標準化が行われる．以下，これらの組織レベルの管理活動について述べる．

（1） 作業標準

大規模ソフトウェアの開発は多数の開発者の共同作業により行われる．このため，各工程の作業内容と生産物を定め，これを作業の規範として，開発担当者に遵守させることが重要となる．この規範は作業標準と呼ばれ，その内容は**表-3** に示すように大別して，工程の定義と生産物のような管理要領的規定と設計ドキュメント作成法などの作業要領的規定に分けられる．作業標準はソフトウェア開発の手順，すなわちソフトウェアプロセスを規定したものであり，これに従って実行されるプロセスをとおして収集されたデータを分析し，その結果をフィードバックすることにより，プロセスの改善が行われる．

（2） データの標準化

表-4 はプロセス改善のための収集データの一例を示したものである．これらのデータはデータベースに一元的に蓄積・管理され，見積りモデルの精度向上，生産性阻害要因の特定，生産技法の改良などのプロセス改善にフィードバックされる．

データのフィードバックを有効に行うためには計測項目の定義及び計測法について標準を規定し，各プロセスで収集するデータを共通に扱えるようにすることが大切である．しかし，分析に対する要求は多種多様であり，収集データを標準化して固定的に決めてしまうと，要求に十分対応できないことになる．すなわち，何をなぜ分析するのか，というデータ分析の目的が先にあり，これに合わせて収集データは決定されるべきものであ

表-3 作業標準の規定内容

分類	項目	内容
管理規定	工程の定義と生産物 組織と運営方法 共通管理コード	工程区分，生産物 開発体制，議事録，連絡票 ソフトウェア名，特権命令コード
作業規定	ドキュメント作成基準 帳票と記入要領 コーディング規約	種類，形式，目次構成 障害処理票，設計用紙 コーディング技法

表-4 収集データの例

分類	収集データ	計測法		
		計測項目	計測単位	計測契機
プロジェクト特性	開発概要	ソフトウェア番号，開発年度，開発社	—	計画時
	開発特性	プログラム種別，開発環境，開発種別	—	計画時
	使用技法	設計法，ドキュメンテーション法，使用言語	—	工程単位
	仕様変更回数	機能仕様変更回数	回	工程単位
生産物	開発規模	新規/改造/再利用別プログラム規模	Ks	工程単位
	設計書枚数	機能設計書枚数，詳細設計書枚数	枚	週単位
	検討項目数	基本検討項目数，機能検討項目数	件	週単位
品質	レビュ指摘数	設計書レビュ指摘項目数	件	週単位
	摘出バグ数	単体試験バグ数，結合試験バグ数	件	週単位
	試験項目数	単体試験項目数，結合試験項目数	件	週単位
	残存バグ数	初期維持管理期間のバグ数	件	半年単位
リソース	稼働要員数	ランク別最大稼働要員数	人	工程単位
	開発工数	アクティビティ別ランク別作業工数	人時	週単位
	マシン時間	用途別マシン使用時間	時間	週単位
	ファイル量	用途別ファイル使用量	KB	工程単位
	開発期間	開発期間	日	工程単位
	保守工数	初期維持管理の作業工数	人時	半年単位

る．このような考え方に基づく体系的なデータ収集・分析の枠組みとして，GQM (Goal-Questions-Metrics) パラダイムがバシリー (V. R. Basili) らによって提案されている[19]．

4.4 外注管理

ソフトウェア需要の急速な増大にともなって，開発の全部あるいは一部を外注に頼ることが日常的となっている．ソフトウェアの外注においては，ソフトウェア開発のプロセスは基本的にはブラックボックスとみなせるため，これまで述べてきたようなプロセスに着目した管理とは違った側面の管理手法が必要とされる．これが，いわゆる外注管理である．外注管理をその流れに従って分類すると，外注計画，契約，進捗管理，受入検査，外注評価に分けることができる．これらはいずれも外注を成功させるための重要な管理作業であるがなかでも外注計画，受入検査及び外注評価の重要性が高い．そこで，以下これらを中心に述べる．

（1） 外注計画

表-5 は外注計画として実施しなければならない主な作業を示したものである．このうち，重要なものは仕様の確定と外注費用の見積りである．

ソフトウェア開発の発注において，仕様は外注費用の見積りや受入検査項目の選定などの基礎となるものである．これが明確でないと，外注費用を正確に見積もることが困難となり，納期や品質にまで影響して計画どおりの結果が得られなくなる．したがって，仕様をいかに詳細に規定できるかが，外注管理にとって重要な要因となる．

表-5 外注計画の主な作業

1. 仕様の確定
2. 実施時期の確定
3. 外注先の選定
4. 外注費用の見積り
5. 検査要綱の作成

しかし，契約時に開発内容が固まっていることは稀であるため，仕様を明確に規定することは困難である．このため，現実的にはプログラムの概略仕様，開発期間など，費用見積りに必要な最低限の内容に限定した基本仕様書などと呼ばれるもので契約を行い，それ以降共同で仕様を詳細化し開発に必要な仕様書を作成する場合が多い[20]．この仕様書は機能仕様書と呼ばれ，計画どおりの生産物を得るためと，作成された生産物を正確に確認するためのものであり，ソフトウェアプロセスにおける機能設計工程で作成される．

一方，外注費用の見積りについても，いまだ有効な方法が確立されておらず，発注者側と受注者側の見積り値に大きな差ができることが多い．現状の見積り方法としては，発注者側は過去のソフ

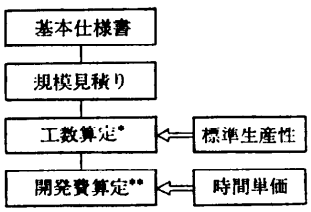
トウェア開発により得られたデータをもとに標準生産性を設定し，これを用いて見積もる方法が一般的である．これに対して，受注者側は作業分解構造 (WBS: Work Breakdown Structure) に基づいて標準作業時間を積み上げ，ボトムアップに見積もるところが多い．図-4 は発注者側の見積り手順の例である．標準生産性は生産性テーブルを利用して求める．生産性テーブルには，開発言語，プログラム種別，担当者レベルなどをパラメータとして生産性の標準値が過去の実績データをもとに設定されている．

(2) 受入検査

受入検査は外注先の作業が全て完了し，納入品が完成した段階で実施される検査であり，これに合格することが検収の前提条件となる重要な作業である．しかし，受入検査で全ての機能を検査することは，多量のテストデータを必要とするため容易ではない．したがって，受入検査では主な機能や性能について検査を行い，その他の機能については発注先に検査成績書の提出を義務づけて，それを確認する方法をとる場合が多い．

検査の合否はあらかじめ設定した品質基準を満たすか否かにより判定する．品質基準の例を表-6 に示す．品質基準はユーザに開放した場合の許容バグ率に基づいて設定する．たとえば，表-6 の B ランクバグの品質基準 Q は，ユーザに開放した場合の許容バグ率を 0.2 件/Ks とし，平均 250 ステップ走行する独立なテストプログラムを用意すると仮定して次の式から求めたものである．

$$\frac{0.2}{1000} \geqq \frac{Q}{250 \times 100}$$



* 工数＝規模/標準生産性

** 開発費 $= \sum_{i=1}^{n}$ 工数 i ×時間単価 i

ただし，i は開発要員のランク

図-4 外注費用の見積り

表-6 品質基準の例

バグランク	品質基準	バグ内容
A	0% 以下	重要機能のバグ
B	5% 以下	軽微なバグ
C	10% 以下	メッセージの不備

注) 品質基準＝バグ数/テストプログラム数

(3) 外注評価

ソフトウェア開発を継続して外注に頼る場合には，受入検査の結果及び開放後の品質などをもとに外注会社を評価し，その結果をフィードバックすることにより，外注のメリットを確実なものにしていく必要がある．たとえば，生産性，開放後の品質，計画と実績の差異などについて評価し，次回の外注計画の立案や外注会社の選定に役立てる．

このような評価を行うことにより，常に品質の良い安価なソフトウェアが取得できる仕組みが形成される．そして，評価の高い外注会社に継続的に発注を行うようにすることが，外注会社に対して開発意欲を向上させるインセンティブを与えるとともに，企業グループとしての信頼関係を継続する基盤となる．

5. 今後の展望

20 数年前，ソフトウェア工学の必要性が叫ばれてから，ソフトウェア開発に関するさまざまな技術が開発され，現実のソフトウェア開発の改善に貢献してきた．しかし，多数の人を最適に管理しながら，ソフトウェアの品質や生産性を上げるためのソフトウェアマネジメントの諸問題は，まだ十分に解決されていない．これは，ソフトウェア開発が多数の人が関係する協同作業であり，本質的には人間の問題に帰着するため，研究のアプローチが難しいからである．

人間要素の問題は，人間の思考過程の解明にまでおよぶ長期的な研究課題である．このため，現実的な観点からみて，人間要素の影響を取り除くか小さくして，ソフトウェアマネジメントを定量的に扱う技術の研究が，今後ますます重要となる．このような理由から，プロセスプログラミングとソフトウェアメトリックスが，ソフトウェアマネジメントの今後の方向を特徴づける主要技術であると予想される．

(1) プロセスプログラミング[21),22)]

ソフトウェア開発は，作業の遂行に多様な選択肢があるきわめて複雑なプロセスであるが，プロ

セスがあらかじめ用意された計画に従って定義されていれば秩序だったやり方で作業を選択できる．これをさらに進めて，個々のプロジェクトに合ったプロセスを設計し，アルゴリズミックに記述して，プロセスを自動的に実行する環境．いわゆるプロセスプログラミングが実現されれば，人間の行動特性の影響を小さくすることが期待できる．これは，定量的プロセス管理に必要なソフトウェアメトリックスの測定精度を高めるための重要な要因であり，プログラミング環境の研究とあいまって，研究が進むものと予想される．

一部には完全なプロセスプログラミングは困難であるという意見もあるが，次の時代のソフトウェアマネジメントを特徴づける重要なキーワードの一つであることは疑う余地がないであろう．AI 技術，ファジー理論，ニュラルネットワークなどの新しい技術の導入による今後の展開に期待したい．

（2） ソフトウェアメトリックス

プロセスプログラミングにより人間の関与する部分が少なくなればなるほど，プロセスにおける変動要因がそれだけ限定される．その結果，メトリックスによる測定値に誤差が入りにくくなるのでデータの収集と分析によるフィードバックが効果的に行われる．このことは，従来難しいといわれていた上流工程のメトリックスにもインパクトを与え，設計メトリックスとその自動計測法の研究が今後ますます加速されるであろう．また，これと並行してメトリックス指向の設計法の研究も進むものと予想される．

6. おわりに

本稿では，ソフトウェアマネジメントの体系化を試みるとともに，現在の技術の展望ならびに本特集の他の編で扱われていない，人間要素を取り上げて解説を試みた．ソフトウェアマネジメントは，まだきわめて未成熟であるが，研究のフロンティアとして，経営工学，品質管理，人間工学及び心理学など，従来の情報処理，ソフトウェア工学分野の研究者以外の関連分野の研究者の注目を集めて現在急速に発展しており，今後の技術の発達に期待するところが大きい．

参考文献

1) Reifer, D.: Tutorial: Software Management, IEEE Computer Society Press (1986).
2) Boehm, B. W.: Software Engineering Economics, Prentice-Hall (1981).
3) Drucker, P. F., 上田惇生訳：抄訳マネジメント，ダイヤモンド (1975).
4) Cusumano, M. A.: Japan's Software Factories: A Challenge to U. S. Management, New York, Oxford University Press (1991).
5) Weinberg, G. M.: The Psychology of Computer Programming, Van Nostrand Reinhold (1971).
6) Shneiderman, B.: Software Psychology, Winthrop (1980).
7) 2000年のソフトウェアの人材，産業構造審議会情報産業部会，通産省 (1987).
8) 野口正一，牛島和夫他：大学等における情報系専門教育の改善への提言，情報処理，Vol. 32, No. 10, pp. 1079-1092 (1991).
9) Sackman, H., Erikson, W. J. and Grant, E. E.: Experimental Studies Comparing Online and Offline Programming Performance, CACM, Vol. 11, No. 1, pp. 3-11 (1968).
10) 浅井正昭，外島 裕他：情報処理技術の適性に関する研究，日本教育心理学会第 30 回総会発表論文集，pp. 900-903 (1988).
11) 外島 裕，松田浩平：SE 適性検査と SE の職務能力評価の関連について，情報処理学会第 38 回全国大会 (1989).
12) Brantman, H. and Court, T.: The Software Factory, IEEE Computer, Vol. 8, No. 5, pp. 28-37 (1975).
13) Harden, W. R. and Gretsch, W. R.: Operating Dynamics of Matrix Management, NAECON '77 (1977).
14) Mills, H. D., Dyer, M. and Linger, R. C.: Cleanroom Software Engineering, IEEE Software, Vol. 4, No. 5, pp. 19-24 (1987).
15) Baker, F. T.: Chief Programmer Team Management of Production Programming, IBM System Journal, Vol. 11, No. 1, pp. 56-73 (1972).
16) McClure, C. L.: Managing Software Development and Maintenance, Van Nostrand Reinhold (1981). (渡辺純一，千田正彦訳：ソフトウェア開発・保守の管理，近代科学社).
17) Brooks, F. P.: The Mythical Man-Month, Addison-Wesley (1975). (山内正彌訳：ソフトウェア開発の神話，企画センター).
18) Humphrey, W. S.: Managing the Software Process, Addison-Wesley Publishing Co., Inc. (1989).
19) Basili, V. R. and Rombac, H. D.: The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Trans. on Software Engineering, Vol. 14, No. 6, pp. 758-773 (1988).
20) 森口繁一監修：ソフトウェア品質管理ガイドブック，日本規格協会 (1990).

21) Osterweil, M.: Software Processes Are Software Too, Proc. of the 9th International Conference on Software Engineering, pp. 2-13 (1987).
22) Boehm, B. W. and Belz, F.: Applying Process Programming to the Spiral Model, Proc. of the 4th International Software Process Workshop, pp. 46-56 (1988).

(平成4年2月20日受付)

東 基衛 (正会員)

1939年生. 1963年早稲田大学理工学部卒業. 同年日本電気(株)入社. 1987年同社を退社. 現在早稲田大学理工学部教授(工業経営学科). ソフトウェア工学 (特に品質管理, メトリクス, 要求定義, 標準化), ユーザインタフェースなどの研究に従事. 編著書は, 日経品質管理文献賞を受賞したソフトウェアの品質管理と生産技術 (1988, 日本規格協会) 及びソフトウェア品質管理ガイドブック (1990, 日本規格協会) の他, コンピュータソフトウェアの標準化 (1976, 日本経済新聞社), システム分析マニュアル (1991, 日本工業新聞社), ユーザインタフェースの設計 (1988, 日経マグロウヒル), ANSI/IEEE ソフトウェア規格集 (1988, 日本規格協会)など多数. 日本経営工学会, 日本経営情報学会, IEEE, ACM 各会員. ISO/IEC JTC 1/SC 7/WG 6 (主査), 情報処理学会情報規格調査会/SC 7 専門委員会 (委員長) 他各種委員会委員.

細谷 僚一 (正会員)

昭和19年生. 昭和41年東京大学工学部電気工学科卒業. 昭和43年同大学院修士課程修了. 同年日本電信電話公社電気通信研究所入社. 以来, 言語処理プログラム, オペレーティングシステム, ソフトウェア工学の研究に従事. 現在 NTT 情報通信網研究所. 著書「DIANA 入門/言語仕様/応用」(共著), 訳書「Ada 入門/和訳規約」(共訳) ほか. 電子情報通信学会, ACM, IEEE 各会員.

高橋 宗雄 (正会員)

昭和19年生. 昭和42年千葉大学工学部電気工学科卒業. 同年日本電信電話公社電気通信研究所入社. 平成3年桐蔭学園横浜大学工学部制御システム工学科助教授, 現在に至る. 工学博士 (九州大学). システム製造用言語, ソフトウェアメトリクス, 品質管理技術などの研究に従事. 昭和59年情報処理学会論文賞受賞. 電子情報通信学会, 日本ソフトウェア科学会, IEEE 各会員.