情報システムのユーザーズ・ガイド(Ⅱ)

――プロジェクト管理――

昭和57年3月

JIPDEC

(財) 日本情報処理開発協会

この報告書は，日本自転車振興会から競輪収入の一部である機械工業振興資金の補助を受けて昭和５６年度に実施した「情報システムの有効利用方法体系化に関する調査研究」の成果をとりまとめたものであります。

はじめに

わが国におけるコンピュータ利用の進展には著しいものがあり，処理の高度化，多様化と相俟って，情報システムは規模の拡大，機能の拡充が着実に図られている現状にある。

しかしながら，現状のシステムを有効にかつ効率的に利用するといったニーズは情報処理部門にとどまらず，経営者・管理者からも聞かれており，その実施にあたってのツール，手法などの整備・体系化が急務とされている。

本事業は，このような観点からコンピュータ・ユーザが情報システムの有効利用を推進する際のツール，手法，用語・概念について調査研究し，情報処理全般にわたる資源の有効利用とユーザの利便を狙いとした標準的な手引書を作成することにある。

本年度は，昨年度実施した費用対効果分析と稼動分析・予測方法に引続き，ソフトウェア開発におけるプロジェクト管理をテーマに取上げ，コンピュータ・ユーザの実情を踏まえながら，プロジェクトの計画，実施，評価の各フェーズにおける作業とその考え方，具体的な計画・管理・評価手法，組織と要員管理のあり方などを明らかにした。

最後に，調査研究にあたってご協力を頂いた委員及び関係各位に深く感謝の意を表します。本書がわが国のコンピュータ・ユーザにおける情報処理システムの有効利用推進の一助となることができれば幸いである。

昭和５７年３月

情報処理ユーザーズ・ガイド委員会

（五十音順，敬称略）

委員長	道下忠行	東海大学工学部経営工学科教授
委員	上田周治	日本鋼管㈱情報システム部計画室企画調整班長
〃	大橋有弘	行政管理庁行政管理局主査
〃	紙谷進	日本電気㈱情報処理営業支援本部応用プログラム部主任
〃	菅野孝男	日本タイムシェア㈱ コンサルティング部主任技師
〃	瀬野浩	行政管理庁行政管理局副管理官
〃	瀬谷重治	日本電信電話公社データ通信本部第三データ部調査役
〃	忠祐治	建設省大臣官房政策課情報管理室係長
〃	原沢正宏	ファコム・ハイタック㈱システム第一部第二課長
〃	本間仁史	横須賀電気通信研究所データ処理部データ蓄積方式研究室研究専門調査員
〃	前川征弘	外務省大臣官房電子計算機室
〃	松平和也	㈱日本システミックス取締役
〃	松村貴章	三井情報開発㈱社会システム事業本部業務室長
〃	小島利文	(財)日本情報処理開発協会常務理事

情報処理ユーザーズ・ガイド専門委員会

（五十音順，敬称略）

主査	瀬野　浩	行政管理庁行政管理局副管理官
委員	上田周治	日本鋼管㈱情報システム部計画室企画調整班長
〃	大橋有弘	行政管理庁行政管理局主査
〃	菅野孝男	日本タイムシェア㈱コンサルティング部主任技師
〃	瀬谷重信	日本電信電話公社データ通信本部第三データ部調査役
〃	本間仁史	横須賀電気通信研究所データ処理部データ蓄積方式研究室研究専門調査員
〃	松村貴章	三井情報開発㈱社会システム事業本部業務室長

目　　次

はしがき

この報告書は，昭和５５年度に行われた「情報システムのユーザーズ・ガイド(I) ― 費用対効果分析及び稼動分析・予測方法 ― 」につづいて，昭和５６年度に行った「プロジェクト管理」に関する調査研究をまとめたものであり，「ユーサーズ・ガイド(Ⅱ)」である。

コンピュータが普及し，利用目的が多様化すれば，一方でソフトウェアの大型化・高度化・複雑化の傾向が強くなり，そのための管理も難しくなってくる。

ソフトウェアの開発の管理目的は，あらかじめ定められている内容（目的・品質）のものを，所定期日までに所定資源（経費・人員）で達成させるとともに，その結果を次期プロジェクトに反映させることにある。そのために，この研究では問題をプロジェクトの計画（経費・日程など）の立て方，実施の方法（工程管理，品質管理，費用管理，外注管理など），評価方法（指標の作成，フィードバックなど）及び推進方法（組織，要員管理など）に区分し，それぞれの分野の専門家を集め，本委員会，専門委員会を設け，アンケート調査，ヒアリング調査を併用しながら作業を続けてきた。

今，この経過を振返ってみれば，各委員は多忙を極めた１年であり，各委員の貴重な知識・経験を生かして作成されたから，本報告書の価値は相当なものであると自負している。関係各位にとっても実務に益するところ大で，貴重な参考資料になるものと思われる。

しかしながら，この分野は研究範囲が広く，奥行きも深いものであり，今後一層の研究を必要とし，もし，読者諸賢から御叱責，御教導をたまわれば幸せである。なお，この研究のために惜しみない協力を頂いた関係各位，特に「調査対象」各位には心からの謝辞を捧げたいと思う。

情報処理ユーザーズ・ガイド委員会

委員長　道下忠行

概　　要

本報告書は，最近，大型化・高度化・複雑化するソフトウェアの開発に関し，これをプロジェクトとして管理するための方法について，とりまとめたものである。

報告書の構成は，次のとおりである。

第Ⅰ部　プロジェクト管理の方法

第1章　プロジェクト管理の概要

第2章　プロジェクトの計画

第3章　プロジェクトの実施

第4章　プロジェクトの評価

第5章　プロジェクト管理ツール

第Ⅱ部　プロジェクト組織の編成と要員管理

第Ⅲ部　マルチ・プロジェクトの管理

すなわち，プロジェクト管理の体系を計画，実施，評価の3つのフェーズでとらえ，第Ⅰ部では，その概念と実態調査の結果の概要を明らかにするとともに，具体的なプロジェクト管理の方法論を展開している。

また，方法論と併わせプロジェクト管理のツールについても紹介している。

第Ⅱ部では，プロジェクト管理の方法に関連し，プロジェクト管理全体を通じるテーマとして，プロジェクト組織の編成と要員管理についてとりまとめている。

第Ⅲ部では，個別のプロジェクトを総合して管理する，いわゆるマルチ・プロジェクトの管理に関し，全体的なシステムの評価という観点から，その方法についてA社の方法を紹介している。

それぞれの内容を要約すると，次のとおりである。

第Ⅰ部　プロジェクト管理の方法

(1)　プロジェクト管理の概要

一般的なプロジェクトの概念，定義を明らかにするとともに，対象をソフトウェア開発のプロジェクトに定め，プロジェクト管理の基本概念と体系ならびに効果をとりまとめた。

また，今回実施したプロジェクト管理の実態調査の結果から，プロジェクト管理全般及び具体的な方法に関し，現状と課題の展望を試み，プロジェクトの計画，実施，評価，組織編成，要員管理，ツール等の導入部を成している。

(2) プロジェクトの計画

プロジェクト管理における計画の意義と実態を明らかにし，ソフトウェアの開発サイクルと開発工程区分を踏まえつつ，対象となる計画の範囲を具体的に示すとともに，開発資源と費用の見積り，日程計画の作成のために必要な手法，方法を詳細に具体例を加えて記述している。

(3) プロジェクトの実施

プロジェクトの実施段階での作業を，工程管理，品質管理，費用管理，外注管理に分け，その基本的な考え方，これに必要な方法，具体的な図表を明示するとともに管理上のポイント，留意事項について詳述している。

また，プロジェクトを円滑に実施するための各種ドキュメントの標準化や変更管理を対象とするコンフィグレーション管理についても紹介を行っている。

(4) プロジェクトの評価

プロジェクト評価の対象を，プロジェクト終了段階における最終評価に定め，プロジェクト全体の評価（見積りと実績，日程計画と実績，バグ原因の分析，指標の評価，当初のユーザ要求と開発後の評価）とフィードバック（基準生産性の見直し，チーム編成及び外注の見直し等）の2つの局面について方法と管理図表を明らかにしている。

(5) プロジェクト管理ツール

プロジェクト管理ツールとしてPMP（Project Management Pro-

gram），PCMP（Project Cost Management Program）について述べるとともに，ソフトウェアの開発技法であるが，その体系の中にプロジェクト管理の技法を包含するものとして，PRIDE（Profitable Information by Design through Phased Planning and Control），SPDS（Standard Procedure to Develop System）SDEM（Software Development Engineering Methodology）について，プロジェクト管理の見地から紹介を行っている。

第Ⅱ部　プロジェクト組織の編成と要員管理

実態調査の結果を踏まえ，プロジェクト組織の要件，構成要素と機能・役割，組織の形態（機能別組織，マトリクス組織，タスクフォース組織，委員会組織）とその構造比較を明らかにし，プロジェクト組織及び開発作業を担当するチーム編成のあり方を述べるとともに，要員管理の重点課題（マネージャの養成，メンバの管理，コミュニケーション，モチベーション）に関して，組織管理論と実践的な方法論の2つの見地から総合的なアプローチを行っている。

第Ⅲ部　マルチ・プロジェクトの管理

マルチ・プロジェクトの管理として，プロジェクトの計画，実施の段階を通じ，全社的なシステムの評価，コントロールという観点からプロジェクトの内容，効果等をチェック，検定する方法に関し，A社の方法論をとり上げ，体系，要領，図表を織り込みながら具体的な紹介を行っている。

第Ⅰ部　プロジェクト管理の方法

1. プロジェクト管理の概要

本章では，プロジェクト及びプロジェクト管理の一般的な概念を明確にするとともに，本ガイドの対象プロジェクトであるソフトウェア開発プロジェクトに関し，アンケート調査結果などから現状における問題点を明らかにする。

1.1 プロジェクトの概念

1.1.1 プロジェクトの定義

現在，我々が所属する企業や組織，あるいは社会や国家では，ソフトウェア開発を初めとしてプロジェクトという言葉は数限りなく利用されている。ことにプラント建設，工場建設，コンビナート建設，港湾施設建設，新幹線建設，通信施設建設などの建設業務は言うまでもなく，宇宙開発，海洋開発，新製品開発，及び企業内改善活動，装置保守作業，市場開拓などの業務にもプロジェクトという言葉が使われている。これらのプロジェクトの中には，一企業内のみで実施するものだけでなく，多数の企業が連合して取り組むものも少なくない。ソフトウェア開発プロジェクトは一企業内で実施する場合が多いが，中には１つのプロジェクトを数企業が分担し，共同して開発するものもある。例えば，超ＬＳＩの開発や最近開始されたスーパ・コンピュータの開発などは通産省やコンピュータ・メーカの共同研究開発プロジェクトである。

以上のようなケースは，プロジェクトとして典型的なものであるが，この他に，例えば，市場調査を特別に限定された期間だけ臨時組織で行う市場調査プロジェクトとか，全社的にＯＤ（Organizational Development）を推進するため，特別チームを組織したり，特定の製品について大幅なコスト低減を遂行するため，設計，生産技術，資材，原価などのスペシャリストを集めてプロジェクト組織を編成したりする場合も，プロジェクトとして扱われる。

このように，プロジェクトという言葉は，何の抵抗もなく種々の業務につけられるようになっているが，一般には次のように定義されている。

「ある特定の目的をもって実施されるソフトウェア開発や建設工事あるいは新製品開発など，一度限りの性格を持つ仕事又は業務のこと。」

これ以外にも，プログラム（例えばアポロ計画のような）をプロジェクトの上位概念として位置づけたＮＡＳＡ（米航空宇宙局：National Aeronautics and Space Administration）の定義などがあるが，本ガイドでは，上記の定義をもってプロジェクトを考えていくことにする。

1.1.2　プロジェクトの特徴

プロジェクトによって生産を出される製品特に大きな違いがあっても，プロジェクトには，その全てに共通するいくつかの基本的な特徴がある。その特徴とは次のようなものである。

① プロジェクトは複雑な活動である

プロジェクトとは，特定の結果を特定の時期に，計画した予算内で生み出すことを意図したものであり，その活動はユニークなものであっていわゆる定常的な業務ではない。

② プロジェクトは特定の結果を生み出すプロセスである。

プロジェクトは，新製品，新しいシステム，あるいはその他の特定の結果を生み出すのに必要な全体プロセスとみなすことができる。

③ プロジェクトにはライフサイクルがある。

プロジェクトには明確な開始時点と終了時点があり，時間で表わすことができる。例えばソフトウェア開発のライフサイクルの例としては，"システム化要求→システム分析→システム設計→プログラム設計→プログラミング→システム・テスト"がある。

④ プロジェクトの特徴はライフサイクルの各段階ごとに変わる。

ライフサイクルの各段階では，それぞれ異なった中間結果が得られる。

従って，プロジェクトに含まれる要員，スキル，組織，資源などもライフサイクルごとに異なったものになる。

⑤ プロジェクトを完成させるための時間と費用に関する不確実性はプロジェクトの進行に伴って減少する。

特定の結果を得るためには，そのための時間と費用を切り離して考えることはできない。しかし，これらの不確実性はプロジェクトの開始時点が最も高く，終了時点ではなくなっている。この特徴から言えることは，終了時点とトータル・コストをできるだけ迅速にしかも正確に予測できるような計画と管理のシステムが必要であるということである。

⑥ プロジェクトの変更，遅延を取り戻すための費用は，プロジェクトの終了時点に近づくほど急激に増加する。

プロジェクト・ライフサイクルのどこかの段階で予定が遅れると，それを取り戻すためには費用がかかり，しかもその費用は，プロジェクトの進行に伴って次第に高くなるのが普通である。

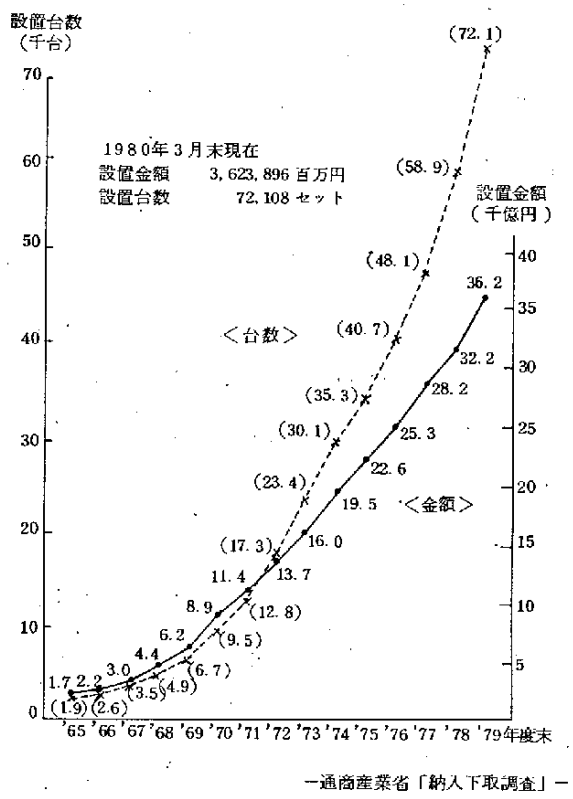
1.1.3 プロジェクトの種類と本ガイドの対象プロジェクト

これまでに述べたようにプロジェクトには様々なものがあるが，本書では次のような理由からソフトウェア開発プロジェクトをその対象として，その管理の方法について述べることにする。

① コンピュータの革新的な発展によって，様々な分野において，コンピュータが導入され，数多くのソフトウェアが開発されている。（図1－1参照）

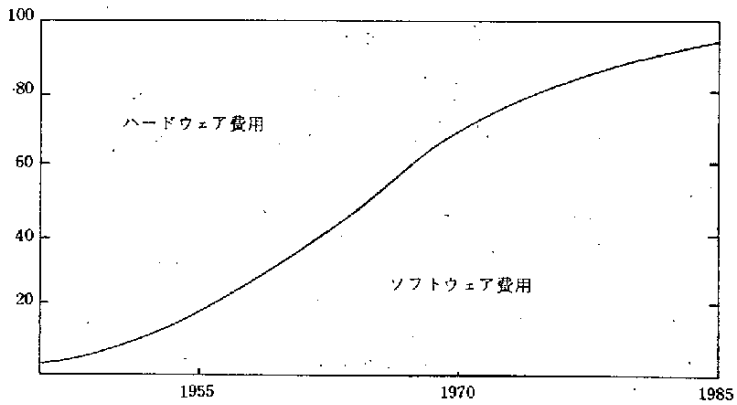
② ハードウェアのコストパフォーマンス向上に比べると，ソフトウェア開発技術の発達は非常に遅く，未だに手工業の域を出ていない。従って，人件費の高騰に伴い，システム開発時におけるソフトウェア開発費の占める割合が年々増加している。（図1－2参照）

③ ソフトウェア開発プロジェクトの管理を効果的・効率的に行うための



－通商産業省「納入下取調査」－

図１－１　コンピュータの設置状況の推移



－ＴＲＷ社のB.W.Boehm－

図１－２　ハードウェア費用とソフトウェア費用の比率の推移

手法・技術が，十分とは言えない状況にあり，多くのユーザにおいて，有効なガイドが嘱望されている。

なお，本ガイドでは，ソフトウェア開発のうち，計画，実施，評価を対象にするものとし，保守については，その対象外とする。また，１つの組織の中で複数のプロジェクトがあり，それらが密接に関係している場合には，マルチ・プロジェクトと呼ばれ，この場合の管理もプロジェクト管理と呼ばれるが，マルチ・プロジェクトの管理は，プロジェクト・マネージャが行う管理とは異なり，主として「プロジェクト管理部」といった別な組織で行われる管理であるので，第Ⅰ部では対象外とし，第Ⅲ部で事例として紹介する。

つまり，本ガイドの第Ⅰ部ではプロジェクト・マネージャが１つのプロジェクトを計画，実施，評価する場合の考え方や方法論について述べることにする。

1.2　プロジェクト管理

1.2.1　プロジェクト管理の基本概念

プロジェクト管理の目的は，最終製品について望ましい水準を達成するとともに，プロジェクトをスケジュール通りに，しかも，予算内で確実に完了するために，プロジェクトを適切に管理することだといえる。プロジェクト管理のアプローチはこの目的を達成するものであり，その基礎となっている主要な概念は次の２つである。

・ただひとりの責任者（プロジェクト・マネージャ）

・統合的な計画と管理

この２つの概念は相互に補い合うものであり，プロジェクト管理を効果的に実施するためには，これらの概念を同時に適用しなければならない。２つの概念を同時に使わないで，一方だけの概念を適用したのでは，意図

した結果は得られない。

(1) プロジェクト・マネージャ

プロジェクト・マネージャは，プロジェクト管理を実際に行う直接的な責任者であり，プロジェクト管理の最も基本的な概念である。従って，プロジェクト・マネージャの組織内における位置付けや要求される資質，スキル，あるいは責任や権限等についての問題は重要な問題である。しかし，これらについては第5部で詳述することにし，ここでは，プロジェクト・マネージャの主な責務を明らかにするにとどめる。

プロジェクト・マネージャの責務は多岐に亘るが，主なものをあげると次のようになる。

① プロジェクト・コントロール

- 全てのプロジェクト活動が，正しく行われるための費用の見積りや日程計画の立案を行う。
- 実績と計画の差異をできるだけ早く明らかにし，その原因の究明と対策を速やかに行う。
- プロジェクトの終了時には適切な評価を行い，次のプロジェクトに役立つ資料を整備する。

② 情報の伝達や意見の調整

- プロジェクトに関するユーザやトップなどの要求事項を完全に理解し，プロジェクト・メンバに周知させる。
- プロジェクト全体の目標，実行方法，予算，スケジュール等について，関連部門の責任者に周知させる。
- プロジェクトの特定の作業とか，活動についてプロジェクト・メンバ又は他部門との間に対立や意見の相違がでてきた場合，これを裁定し，解決する。
- 支援してくれる全ての部門に対し，連絡を密にする。
- プロジェクトの問題領域と現況について，上位階層の管理者と常

にコミュニケーションを保つ。

③ そ の 他

(2) 総合的な計画と管理

プロジェクト管理を最も有効に適用するには，プロジェクトの計画と管理を徹底的に行なうための方法と手続きを整えておく必要がある。さらにプロジェクト・マネージャがこれらの方法と手続きを実際に活用することが肝要である。

プロジェクトの管理とは，所有，独裁といった古い意味での管理ではない。それは，目的と目標を設定して実施すべき作業を定義し，必要な資源と利用できる資源を考慮して作業項目とそのスケージュールを決め，確立した，きちんとした手続きにもとづいて，進捗状況と達成度を測定することによって成り立つ管理である。

プロジェクト・マネージャは，計画が適切かつ有効なものであることを保証し，適切な管理上の処置が確実にとれるように進捗状況を監視する。計画が最適なものであれば，プロジェクト・マネージャはチーム・メンバ間のインタフェースを調整することができ，対立や問題の解決に力を貸したり，未解決の問題を適切な階層で意思決定してもらうように提示することができる。

また統合的な計画と管理を行うためには，①プログラム及びドキュメント ②時期 ③資金，人的資源といった情報をプロジェクト実施のあらゆる段階について収集し，将来計画を継続的に修正するとともに，実際の結果と計画を比較し，あらゆる情報を相互に関連づけて分析し，プロジェクトの終了時点における全体時間と費用を予測することが重要である。
本ガイドの第2章以下では，プロジェクトの計画，実施，評価の各段階における管理の方法を説明している。

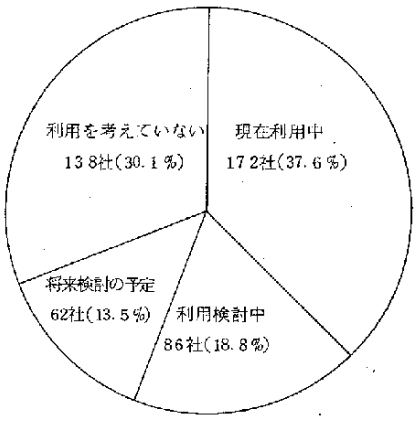
1.2.2　プロジェクト管理の必要性

従来の管理方法によらず，いわゆるプロジェクト管理を実施すれば，組織構造が複雑になったり，人間に関する微妙な問題を生ずることも多い。しかし，それにもかかわらずプロジェクト管理へのアプローチが行われるようになってきているのは，それを必要とする強い力が働いているとみなすことができる。

具体的には次のようなことが考えられる。

① プロジェクト規模の拡大とリスクの増加

バンキング・システム，本支店ネットワーク・システム等のオンライン・システムや大規模データベース・システムの開発など，開発されるソフトウェアの規模が拡大化の傾向にあり（図１－３，表１－２参照），それに伴って当然リスクも増大することになる。予想されるリスクを巧みに回避し，膨大な資金とマンパワーを運用し，初期の目標をスケジュール通りに，経済的に実施しようとすれば，機能部門の管理者がインフォーマルにプロジェクト活動を調整するだけでは不十分であり，正しいプロジェクト管理が実施されなければならない。



—コンピュートピア，1980／3—

図１－３　民間企業における商用ＤＢＭＳの利用状況

表１－２　全産業・オンライン・システム形態の現況と５年後の予定

（多重回答）

システム形態		実回答者数	データ収集システム	メッセージ交換システム	照合応答システム（ファイル更新なし）	取引処理システム（ファイル更新あり）	リモート・ジョブ・エントリー（リモート・バッチ・システム）	タイム・シェアリング・システム	その他	のべ回答者数
現在	社数	627	405	185	397	406	216	180	8	1,797
	%	100.0	64.6	29.5	63.3	64.8	34.4	28.7	1.3	286.6
5年後	社数	549	320	256	375	404	317	298	8	1,978
	%	100.0	58.3	46.6	68.3	73.6	57.7	54.3	1.5	360.3

―コンピュータ白書，1981―

② プロジェクト内容の複雑化

プロジェクトは，単に規模が大きくなっているだけでなく，内容自体が非常に複雑になってきている。これはオペレーティング・システム，データベース管理システム，言語処理プログラム，各種ユーティリティ・プログラムなどの基本ソフトウェアがますます複雑になっている他，通信装置，周辺装置，端末なども高度化・複雑化してきているためである。これらのものはユーザにとって使い易くなっているものの，その反面，機能の全てを知ろうとすれば，複雑多岐に亘る場合がほとんどである。

これに加えて，ユーザのニーズも多様化してきている。このような状況のもとで，高い品質のソフトウェアを開発するためには，強固な方針

構想，計画のもとに，複雑な業務が整然と遂行されるための特別な管理方式が適用されなければならない。

③　資源の制約

マンパワー，資金，資材などプロジェクト遂行に投入される資源は有限である。資源に余裕があれば，プロジェクト管理は楽である。制約された資源をいかに有効に活用し，目標とする品質，機能，信頼性などを持ったソフトウェアを生産するかということはプロジェクト管理の大きな課題の１つである。

企業が実施するプロジェクトであれば，資源面での限界が現実となった場合，企業にとって致命的な危機をもたらすこともあり得る。また，ナショナル・プロジェクトであれば，国家予算を大幅に超過することによって重大な問題に到ることにもなりかねない。

ちなみに，アメリカにおいてＰＥＲＴ（Program Evaluation and Review Technique Time）が開発された背景には，予算及び工程の計画と実際にあまりにも大きな差があったためといわれている。

④　不確定要素

ソフトウェアの開発に係る環境は，建設プロジェクトのように天候や土質といった自然条件のようなものに左右されることはあまりない。しかし，要員の資質に頼るところが大きいために，要員の病気，けがなどはプロジェクトに非常に大きな影響を与える。また，ソフトウェアの開発の場合，ユーザのニーズを最初に１００％は握することが難しいことも多く，開発途中でのニーズの変更も多い。これが進捗に与える影響は大きい。これらの要素はいずれも予想することはほとんど不可能なものであり，これらの事態に柔軟に対処し，影響を最少にするためには，正しいプロジェクト管理のもとに総合的に管理することが必要となる。

⑤　時間の制約

国にとっても，企業にとっても，タイミングは戦略上重要なファクタ

である。タイミングよくプロジェクトを完成させようとすれば，許される時間的資源は厳しいのが常であり，貴重な時間をいかに有効に利用するかは管理上大きな問題となる。この時間の管理についても，通常のルーチン・ワーク的な管理ではなく，プロジェクトのための特別な方法が必要になる。

1.2.3 プロジェクト管理の効果

本ガイドの第2章以降で示しているプロジェクト管理の方法は，機能部門を管理するために考えられた手続と方法を使って，機能部門の管理者がインフォーマルにプロジェクト活動を調整するという管理方法に比較すると，次のような特徴と効果を有している。

（特　　徴）

① プロジェクトの実施は，達成可能な技術，日程，費用の目標にもとづいて行われる。

② プロジェクトの目標が現実に達成できるようにプロジェクトを計画し，スケジュールを決めて厳密に管理される。

（効　　果）

① 工程の確保

正しい工数見積り及びWBS（Work Break Down Structure）に基づく必要な作業の設定と，入手可能な資源を勘案したスケジュールによって，各工程ごとに厳密にその進捗が管理されるので，ほぼ当初の予定通りにプロジェクトを完成させることができる。もし，不測の事態が起きても，事故に対処し，その影響を最小限に止めることができる。

② 品質の保証

適切な品質管理技術の駆使，及びコンフィグレーション管理によって，ソフトウェアの品質を高めることができる他，要求変更等に対しても柔軟に対処でき，品質の低下を防ぐことができる。

③　費用の抑制

的確な費用の見積りと，その定期的な管理によって費用の無意味な増大を抑制し，当初の予算内での終了を可能にする。また，終了時の適切な評価は，次のプロジェクトに対し，非常に有効な見積り資料を提供することになる。

1.3　プロジェクト管理の実態

一般コンピュータ・ユーザ，及びソフトウェア・ハウスにおけるプロジェクト管理の現状を，アンケート調査及びヒアリング調査から①プロジェクト管理全般的な問題，②具体的な方法の２つの観点から浮き彫りにし，その実態と問題点を整理してみた。以下でその概要を紹介する。

1.3.1　プロジェクト管理全般

プロジェクト管理全般については，図１－４のような結果が得られている。

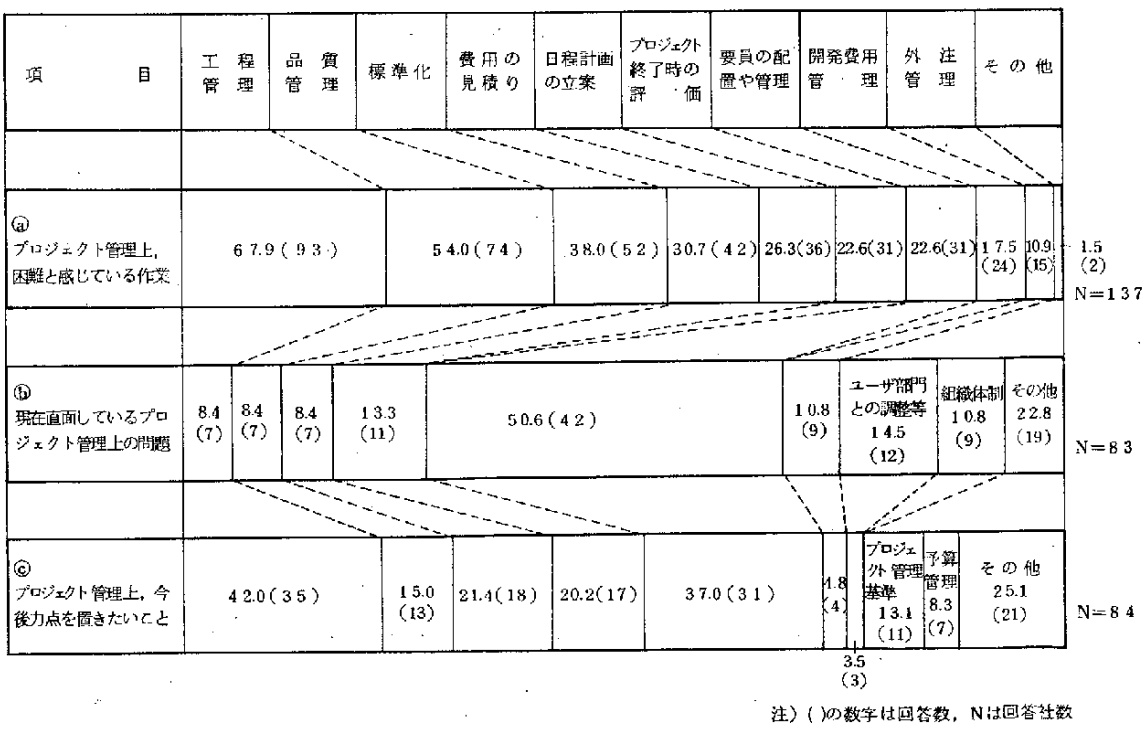


図１－４　プロジェクト管理全般についての問題

この結果から，次のようなことがいえる。

① 工程管理は技術的にも難しいと思い，今後研究していこうとする姿勢がうかがえるが，困難となっている原因はⓑの解答から容易に予想できる。つまり，1つは要員のスキル，モラールの問題であり，1つはユーザの要求の変更等の問題であり，さらにもう1つは，工数見積りの不正確さの問題である。

② ⓐ，ⓑ，ⓒの中で費用に関する問題は，ⓑの見積りに関するもののみであるが，これは，費用の増大の大部分の原因は，工程の遅延によるものであり，人件費以外の費用の増大が問題になることはあまりないという，ソフトウェア開発プロジェクトの特徴があらわれているといえる。

③ 要員管理は技術的には問題はないと考えられているが，現実には非常に難しい問題としてとらえられ，スキルやモラールの向上のため，様々な試みがなされている。

④ 組織に合った標準化は，なかなか難しいが，一応整備していこうという姿勢がみられる。

⑤ 品質管理は，特に困難と考えられているが，実際どうすればよいかということについては，うまい解決策がなく，ひたすら要員のスキルに頼っているのが実情である。

1.3.2 プロジェクト管理の具体的な方法

ここでは計画，実施，評価，各レベルの現状における問題を総括して紹介するとともに，組織や要員の問題に対する対処の仕方，及びツールの利用状況について述べる。なお，個々の具体的な内容については，第2章以降で随時紹介する。

(1) 計画レベル

プロジェクト全体の費用の見積りは，担当者の過去の経験による推定というはなはだ不確実な方法にたよっているところが半数以上にのぼっ

ている。しかし，半面，見積り基準値を作り，それによって見積っているところも少なくない。具体的には，技術者レベルの区分，プログラムの難易度の区分，処理形態，作業項目単位などに応じた標準値が作られている。

また，ＳＥの工数算定についても，総工数，総ステップ数，総プログラム本数を基準として求めている例が見られる。

また，日程計画は，過去の経験に照らして作成される場合がほとんどであり，ターゲットに合わせて大工程ごとに，ある割合によって割り振られている。しかし，その割合についてはあまり明確な数字は得られていない。

(2) 実施レベル

実施レベルには含まれる管理の内容が多いので，それぞれ個別にその現状を示す。

① 工程管理

工程管理の具体的な方法としては，ガント・チャートの利用が５０％で最も多く，次いでネットワーク図も比較的多く用いられているようである。

また，工期遅延の原因としては次の３つが代表的なものであるといえる。

ⓐ 見積りの不正確

ⓑ ユーザ・ニーズの不正確な把握

ⓒ ユーザからの要求の変更

② 品質管理

ソフトウェアの品質を向上させる方策としては，ウォーク・スルーを実施しているところが４０％程度あったが，全体としては，ソフトウェア生産技術（構造化設計，テスト・ツール等）と要員のスキルに頼っているところが多いようである。しかし，品質の問題の大部分はテ

ストの不十分さをあげているものが多く，生産技術としてのテスト技術の未成熟さをうかがわせている。

③ 費用管理

費用管理の具体的な方法としては，部門ごとに，また費目ごとに集計しているのが半数以上あるが，もし，この方法のみの場合，これは今回の調査で¼近くの企業が予算管理を行っていないという結果とともに，プロジェクトとして発足させ運用する意味が全く失なわれているといえる。

また，費用の増大の原因としては，工期遅延と全く同じ結果が出ており，その主な原因が工期の遅延にあることを裏付けている。

④ 外注管理

プログラムを外注する場合，最も問題となるのは納期と品質である。納期の問題については定期的な報告を義務付けているのが最も多い。また納期が遅れた原因としては，質の低い要員のアサインをあげているのが最も多いが，これはいわば外注先選定の誤り，あるいは外注単価にも問題があると思われる。また品質の問題については，レビュー作業の時に社員が必ず参加するという方法で，要求事項の実現の確認，品質のチェックを行っている。

さらに検収時の問題については，要求した機能が満足されているかどうかのテストに苦慮しているところが８０％近くに達しており，ここにもテスト技術の未完成さが，はっきりあらわれている。

(3) 評価レベル

プロジェクトの終了時の評価については，半数近くしか「日程計画と実際の対比」あるいは「予算と費用との対比」を行っていない。これは「特に分析，評価を行っていない」が⅓以上あることとともに，プロジェクト管理が十分，地に足のついたものになっていないことを表わしている。また，プログラムの生産がプログラマの資質に大きく影響されるこ

とは明らかであるはずなのに，個人生産性の分析を行っているのは１０％に満たない。これでは，要員の効果的な配置や教育効果の確認などは全くできず，見積りが不正確になるのも当然と思われる。

(4) 組織，要員の問題

組織，特にチーム編成の方式に関してはチーフ・プログラマ・チーム方式と一般的な階層化された組織がともに４０％程度と最も高かった。チーフ・プログラマ・チームはＩＰＴ（Improved PogrammingTechnique）そのままの方式を採用するというよりも，各企業の実情に合わせた方式にしている例が多く，単に数人でチームを組み，そのリーダにある程度の権限を与えているというのも多いようである。

要員のスキルやモラールの向上については各企業ともかなり苦労している様子であり，実に様々な意見が寄せられた。特にモラールの向上は，ソフトウェアの開発が，ほとんど人間によるものであり，一人一人の能力を十分に発揮させる必要のあることから，様々な工夫がなされている。代表的な意見として次のようなものがある。

① チーム編成時において，人間関係，スキルを考慮している。

② 能力，経験その他により仕事の与え方を工夫している。

③ 仕事の内容，目的について理解を深めさせている。

④ ミーティングの充実によってコミュニケーションをよくしている。

⑤ ローテーションを積極的に推進している。

(5) ツールの利用状況

プロジェクト管理を目的とするツールは，ほとんど使われておらず，ツールとして解答があったのは，ＰＲＩＤＥ（Profitable infomation by Design through Phased Planning Control）（日本システミックス）ＳＤＥＭ（Softnare Development Engineering Methodology）（富士通）＜特に進捗管理＞などであった。その他テスト支援ツールの解答も何件かあったが，本ガイドの対象外なので，詳細は省

略する。また自社開発のツールも数件あった。

(6) そ の 他

その他として標準化についての調査結果の概念を紹介する。標準化も様々な種類があるが，最もよく作成されているのはドキュメント標準であり，実に９０％近くの企業で実施している。また，コーディング上の技法及び形式上の基準及びソフトウェア・モジュールに対する命令法等の形式標準を用意している企業はいずれも７０％を越えている。一方，テストの内容や方法に関する基準を備えているところは３０％程度しかなく，ここでも「テストの難しさ」が表われているようである。

2. プロジェクト計画

2.1 プロジェクト計画の意義と現状

2.1.1 プロジェクト計画の意義

ソフトウェア開発におけるプロジェクト管理のねらいは次の３大目標（ターゲット）に集約できる。

- 所定の機能，質の確保・保証
- 定められた期間内での完了
- 予算内での完成

これらの目標を効率的，効果的に達成するためには開発の基本となる最適なプロジェクト計画の策定が必須である。

プロジェクトの効率的推進の疎外要因あるいはプロジェクトの失敗の原因を整理してみると一般的には以下の事項を指摘できる。

① プロジェクト計画の甘さ
② 無計画な仕様，プログラム変更
③ プロジェクト管理技法の欠如
④ コミュニケーションの欠如
⑤ システム利用部門の不参加
⑥ トップ・マネジメント・サポートの欠如
⑦ 参加者のモチベーションの欠如
⑧ ドキュメントの不明確さ，欠如

また，米国連邦政府の１５０のプロジェクトの事例調査によれば５０％の事例が計画の不備に原因があるとされており，計画が管理されていないことになる失敗事例が３３％であったとされている。残り１７％が技術的な原因によるものであった。

今回のプロジェクト管理に関する実態調査結果から過去のプロジェクトの予算超過，工期遅れの原因をその大きさの順に指摘すると以下の通りで

ある。

① 見積り不足——主として工数見積りのミス

② 仕様の途中変更

③ ユーザ部門とのコミュケーション不足

④ 計画（予算，工期）の無理

以上の事例，要因はいずれもソフトウェア開発に要する資源（要員，ハード／ソフト，データ通信機器等）を最適に割り当てた計画づくりがなされておらず，かつ計画に沿った効率的なプロジェクトの運営（進捗管理，変更管理，品質管理，要員管理等）がなされていなかったことに起因するものとみてよい。

従って，プロジェクト管理において，プロジェクトを成功させるにはソフトウェアの開発環境にマッチした計画づくりが必須条件といえる。

2.1.2 プロジェクト計画の現状と課題
ー調査結果の分析・評価

今回の調査結果からプロジェクト管理における計画の実態，位置づけ，問題点等については握することができる。以下には主要な調査結果とそれらにもとづく今後の課題について述べることとする。

(1) プロジェクト管理における計画の位置づけ

プロジェクト管理全体の中で計画のフェーズがどのように位置づけられているかは計画の中での重要な業務である費用の見積り，日程計画の立案の実態を見ることでは握できる。前出の図１－４はプロジェクト管理上の３つの観点，①困難と感じている作業，②現在直面している問題点，③今後の重点事項から費用の見積り，日程計画の立案等の計画業務の位置づけを示したものである。いずれの観点からも計画の重要性は示されているが，特に現在直面しているプロジェクト管理上の問題点の中で日程計画を指摘している比率がきわめて高い。

(2) 予算オーバ，工期遅れのプロジェクトの理由

予算オーバ及び工期が遅れたプロジェクトの理由を示したのが図2－1である。これらの理由のうち計画に直接起因するものとして工数見積りの誤り，マシン・タイムの見積り誤り，無理な計画等をあげることができる。さらに間接的に計画に関連する大きな要因には仕様の途中変更，ユーザとのコミュニケーションの欠如があり，プロジェクト管理の疎外要因となっている。

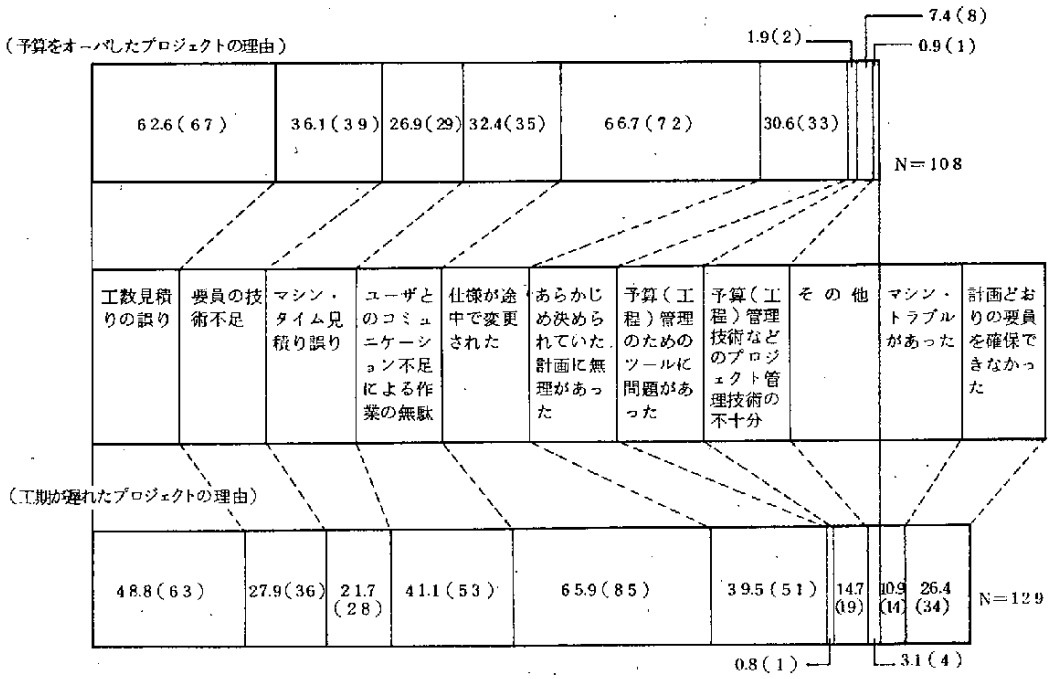
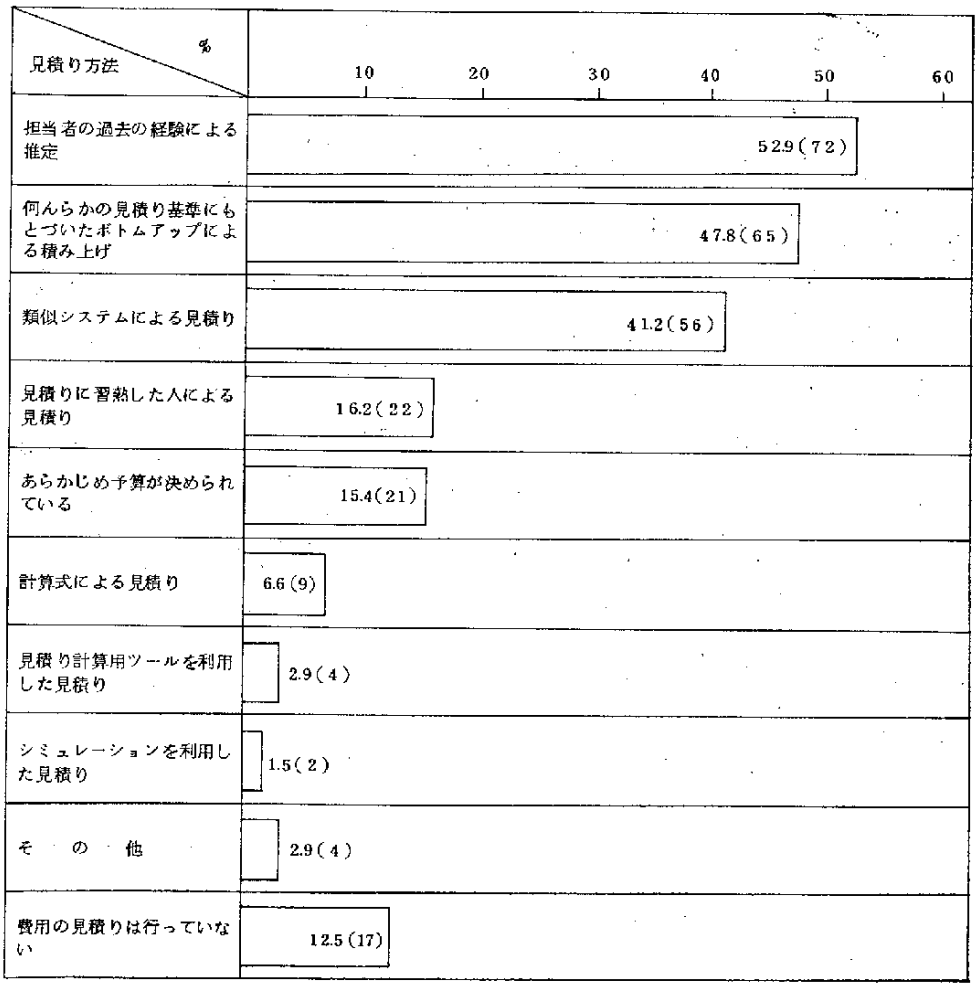


図2－1　予算オーバ及び工期が遅れたプロジェクトの理由

(3)　プロジェクト費用の見積り方法

プロジェクト費用の見積り方法には大きく，経験・類似プログラム等にもとづく伝統的見積り手法と科学的手法の２通りが考えられるが，見積り作業の実態は図２－２に示すとおりである。これによると代表的な見積り



N＝136

注）（）の数字は回答数，Nは回答社数

図２－２　プロジェクト全体の費用の見積り方法

手法は伝統的な方法が主流であり，次の３種に集約である。

① 担当者の過去の経験による推定（５３％）

② 見積り基準にもとづいたボトムアップによる積上げ（４８％）

③ 類似システムによる見積り（４１％）

しかしながら見積りの実態は方法論よりもむしろ見積りに習熟したベテランに依存していることも見逃すことのできない事実（２２件）である。また，あらかじめ予算が決められている（２１件）という事実はソフトウェア開発の現実の姿でもある。

(4) プログラミング作業の工数基準

プログラミング作業（プログラム設計からプログラムテストまで）における工数の見積りは通常，類似システムの事例，過去の経験等からプログラムの機能及び開発言語を加味してまず開発規模（開発ステップ数）を推定する。開発に必要な工数は，この開発ステップ数をもとに算出する。この場合，過去のプログラム開発から得られた実績データをもとに見積りの基準値（標準値）を設け工数の設定を行っているのが一般的である。

今回の調査結果によれば，表２－１に示すように基準値は各企業によって異なるが，開発言語種別ごとに以下のような観点から設定されている。

① 技術者のレベル ―― 上級（経験３年以上）
―― 中級（経験２年）
―― 初級（経験１年未満）

② プログラムの難易度 ―― 難
―― 中程度
―― 易

③ プログラムの形態 ―― バッチ処理
―― オンライン処理

表２－１　プログラミング作業の工数基準

（単位・ステップ/日）

区分＼言語の種類		アセンブラ		COBOL		FORTRAN		PL／1	
区分なし（一括）		30.6		64.9		30		40	
		20	50	38	100				
技術者のレベル	上級	85		112.5					
				110	115				
	中級	25		83.2					
				75	90				
	初級			66.7					
				50	80				
プログラムの難易	難	15		50		22		35.0	
								30	40
	中	27.5		62.5		28		50	
				60	65				
	易	40		75		34		50	
								40	60
プログラムの形態	バッチ	45		71.9		37.5		39.5	
		35	50	20	110	20	50	25	53
	オンライン	23.0		57.6		27.5		24.7	
		10	30	30.5	100			15	34.4
	オンライン（難）	16.7		25		25		25	
		10	25						

注）

平均値	

これらの以外にも技術者のレベル，プログラムの難易度，処理形態を加味せず，開発言語の種類（アセンブラ，COBOL，FORTRAN，PL/1）別に基準値を設ける非常に簡易化した方法もみられる。

また，言語別（COBOL），作業区分（システム設計，プログラム設計，プログラミング，テスト）ごとにそれぞれの基準値を設定している例もある。

(5) ＳＥ工数の算出方法

プログラミング段階での工数見積りには過去の実績データにもとづく見積り基準値により確度のある工数設定が行なわれている。これに比べＳＥ工数の見積りはシステム個々に要求条件が異なるところから統一的，定型的な算出方法はあまりみられず，過去のノウハウにより見積っているのが現状である。

調査結果によればＳＥ算出の方法は次の２つに類型できる。

① 開発総工数あるいはプログラム工数に各企業が過去の実績データ，経験から得た比率（ＳＥ工数比，標準比率）を乗じて求める方法。

② プログラム開発規模（本数等），プログラムの形態（オンラインプログラム，バッチプログラム）別にＳＥ工数の標準値が設定されておりそれを参考に見積る方法。

各企業，開発対象システムの性格によってＳＥ工数算出の方法は異なるが調査回答の手法例を示すと表２－２のとおりである。

表２－２　ＳＥ工数の算出法　(1)　－計算式－

算　　出　　式	備　　考
イ. $\frac{\text{開発総ステップ数}}{\text{月間作成ステップ数}}$ ×ＳＥ工数比率 （単位：人月）	・月間作成ステップ数 ９００ステップ ・ＳＥ工数比率 0.6～0.8
ロ. Σプログラム作成工数×標準比率	・標準比率 0.8～2.4 開発システムの規模，難易度によって決める。
ハ. システム開発規模の推定 ステップ数／6,０００　（単位：人月）	・システム分析以降
ニ. システム設計（総工数×１５％） プログラム設計（総工数×２０％） プログラム作成（総工数×５５％） 納入テスト（総工数×１０％）	
ホ. （基本設計＋機能設計） ＋（詳細設計＋コーディング＋デバッグ） × 0.1	
ヘ. 全体の開発ステップ／作業区分 （単位：人月）	
ト. プログラム本数／ ｛（ＤＥＦ生産性ＰＨ×期間）＋（基本設計生産性ＰＨ×期間）＋（詳細設計生産性ＰＨ×期間）＋（プログラミング生産性ＰＨ×期間）＋（テキスト生産性ＰＨ×期間）｝	・生産性は本数／人月
チ. 全体工数の４０％相当	

ＳＥ工数の算出法 (2) －数値－

数	値		備考
リ.			
	IMSを利用したDB/DCシステム	Ｓ．Ｅ工数	プログラム工数
	プログラム３０～５０本規模	10~20 人月	５～１０人月
	プログラム１５０本規模	６０人月	３０人月
	プログラム３００本規模	120~150人月	６０人月
ヌ.			
	区分 / S．E作業	バッチ・プログラム	オンライン・プログラム
	システム設計	0.3~0.6 人月/Kステップ	0.6人月/Kステップ
	プログラム設計	0.1~0.3 人月/Kステップ	〃
	テスト	0.2 人月/Kステップ	0.5人月/Kステップ
ル。			
	バッチ系	７～８本／月	
	データベース系	４～５本／月	
	オンライン系	３～４本／月	

(6) 計算機使用時間（マシン・タイム）の見積り

マシン・タイムの見積り方法もプログラミング工数，ＳＥ工数の見積りと同じように基準値を設けて算出しているのが一般的といえる。調査結果によれば以下の主要なファクタからステップ当りのマシン・タイムを基準値として設定している。

① 言語の種類

・アセンブラ，COBOL，FORTRAN，PL／1

② プログラムの難易度（易，中，難の3区分）

③ プログラムの形態

・オンラインプログラム，リモートバッチプログラム，バッチプログラム

・新規作成プログラム，既存プログラムのメンテナンス

④ テスト種別

・単体テスト，結合テスト

マシンタイムは，これらの要素以外にも使用コンピュータ機種，デバック用ソフトウェアツールによっても効率が異なるのでこれらを加味した見積り基準値の設定が重要である。

(7) 日程計画の立案方法

現状におけるソフトウェアの開発環境は必ずしも開発資源に余裕がある状況にない。アンケート調査及びヒヤリング調査からも明らかにされている（図2－3）ように，特に

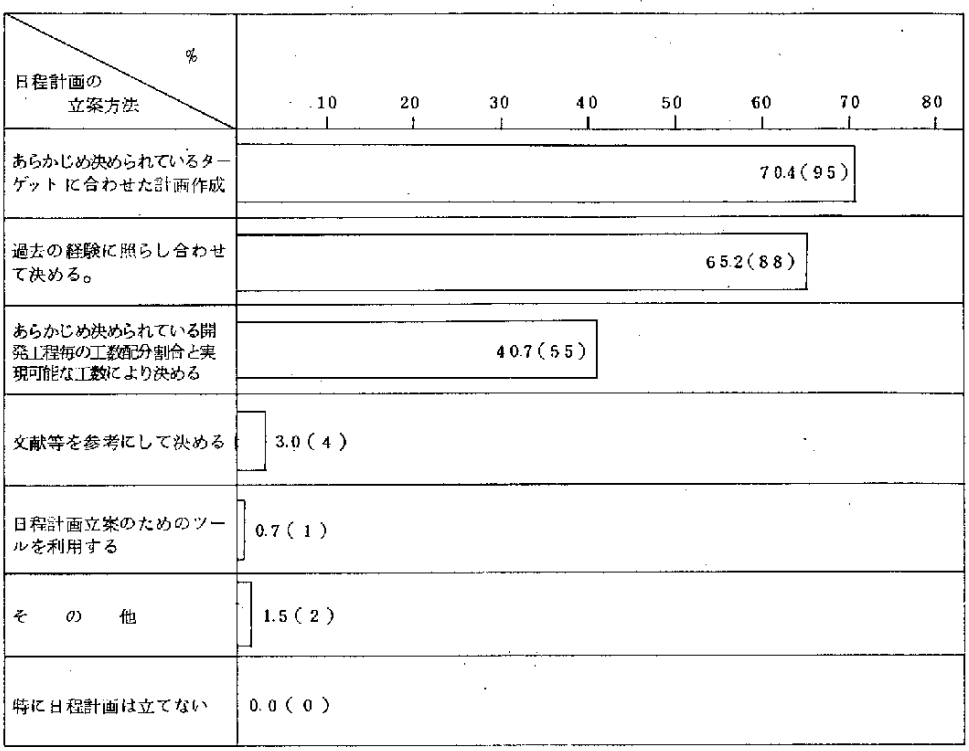
① ターゲット（システム完成時期）があらかじめ決められており，それに合わせた計画を作成している（70%，回答95社）のが現実の姿といえる。

従って，開発期間がすでに決まっている条件のもとでの日程計画は

② 各開発工程毎の工数配分比率をもとに実現可能な工数を各工程に割り当てている（41%，回答55社）のが実態である。

実際の運用にあたっては外注や残業等によって調整しているのが現況といえる。

開発工程毎の工数配分比率は各企業によってまた工程区分，開発システムの性質，規模等によって異なるが調査の過程で例示されたものを参考として以下にあげる。



N＝135　注）（）の数字は回答数，Nは回答社数

図２－３　日程計画の立案方法

（例１）	比率
システム設計	１５
プログラム設計	２０
プログラム作成	５５
テスト	１０

（例２）

	比率
基本設計	１５
詳細設計	２０
プログラミング	３０
システムテスト	３５

（例３）

設計工程	４０
コーディング	２０
テスト	４０

（例４）

設計工程	６０～７０
プログラミング・テスト段階	３０～４０

さらに①，②で示した日程計画立案の実態に加えて

③　過去の経験に照らし合わせて計画の作成を行っている（６５％，回答８８社）現実も大きな比重を占めている。

2.2　ソフトウェアの開発サイクルとプロジェクト計画

2.2.1　ソフトウェア開発の工程区分

ソフトウェアの開発計画の最適化をはかり効率的な計画の実施と管理を行うには基本条件としてソフトウェア開発工程の区分と各工程における作業内容を明確に規定しておくことが必須である。

実際にはソフトウェア開発の作業の流れを一義的に分割し，各工程を定義づけすることは困難であり，企業あるいは開発システムの性質によって大きな差異があるのが実態である。ソフトウェア開発における工程区分の

数はシステムの運用作業，保守作業の工程に比べて一般的に多く，かつ企業間のバラツキが大きいのが特徴である。また，各工程に対する名称も各企業ごと統一性がなくほとんど同じと思われる工程区分に異なる名称がつけられていたり，あるいは同じ名称でもその作業内容が異なる場合も多い。

ソフトウェアの開発工程区分は一般的に

① 企業の特性

② 開発システムの特性，規模

③ システムに求められる品質

④ 開発手段（ハードウェア，ソフトウェア等の生産設備）

⑤ 技術，管理，経験（生産技術，管理技術，要員のスキル等）によって決まると考えられる。

これらの諸要素をふまえて，プロジェクト管理の立場から開発工程の作業区分のための代表的な手法にＷＢＳおよびＰＰＰ手法がある。以下にはその概要を述べる。

(1) ＷＢＳ（Work Breakdown Structure：作業区分構成）

ＷＢＳとはプロジェクトの目標を達成するうえで必要となる作業を詳細化し，階層構造に表現したものである。

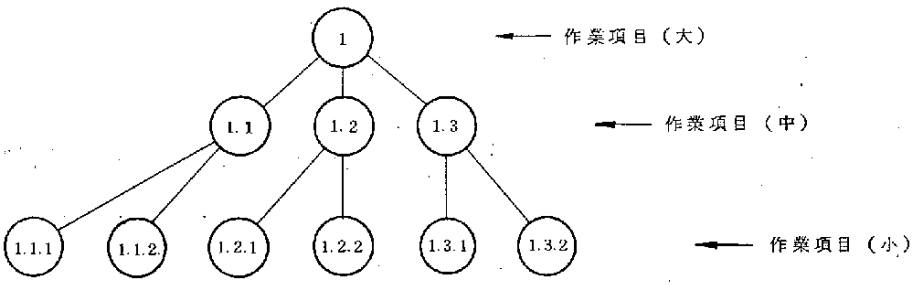


図２－４　ＷＢＳ の 例

ＷＢＳのねらいはプロジェクトを構成する作業を洩れなく洗い出すことである。従って，的確なＷＢＳを作ることによってプロジェクトの組織体制，作業工程，スケジュール，所要工数，必要とする専門知識，さらに予算などを明確にすることができる。それ故，プロジェクトの計画立案にあたってＷＢＳの作成はきわめて重要な作業といえる。

作業洩れのないＷＢＳを作成するために重要なことは次の２つである。

(i) 最終的に何を作るかを明確に定義しておくこと。

これはシステムを構成する必須要素のみならずデバック・ツール，移行ツール，各種エンドユーザ向けマニュアル，テスト・データ等のシステム作成の支援要素についても十分洗い出すことである。

(ii) プロジェクトの納期，新規性，対象範囲等のプロジェクトの特性および問題点，制約事項をは握すること。

プロジェクト管理者はプロジェクトの期間，開発の難易度，対象範囲等に関する項目を検討し，特殊な作業項目の洗い出しを行う必要がある。

プロジェクトが承認段階，プロジェクト計画段階，作業単位の計画段階へと進展，詳細化するにつれ，各作業段階で目的応じたＷＢＳが作成され漸次正確さを増し，最も詳細かつ正確なＷＢＳにもとづき，個別作業の指示やプロジェクト資源の割り当てを行うことになる。

(2) ＰＰＰ（Phased Project Planning）

ＰＰＰとはプロジェクトをいくつかの単位（フェーズ），例えば，要求定義，システム設計，プログラム設計，プログラム作成，テスト，評価等に分割してプロジェクトの計画を段階的に実施していく方法である。この計画技法は，まず最初に全体計画を立て，その計画に従いながらフェーズ毎の詳細計画を策定し，フェーズ毎に計画レビュを実施していくもので，その狙いは問題点や異常の早期発見のみならずプロジェクト見積りの正確と計画修正のフィードバックを小刻みに行うことにある。

ＰＰＰの基本的な視点は計画や見積り誤差のコントロール方法を重視することにある。当初の全体計画にはトップマネージャ，ユーザの合意形成の意義を持たせ，直接作業担当者がスケジューリングや資源配分を行うための計画はできる限り作業の直前に立て，誤差を少なくしようとする狙いである。

ＰＰＰ技法の見積り手順は次のとおりである。

(i) プロジェクトをフェーズに分割する。初期の全体見積りをもとに，各フォーズ毎に配分しておく，（プロジェクトの承認は全体見積りで行われる）

(ii) プロジェクトの進行に沿って，各フェーズの作業開始前，そのフェーズの見積りを再度行う。

(iii) 予じめフェーズ毎に配分されている当初の見積りをオーバーした場合は，必要に応じ計画変更等の処置をする。

このように，ＰＰＰ技法はプロジェクトの初期に１回だけ計画を行うのではなく，プロジェクトの各工程を通じて計画作業を行い，レビュを通じて，プロジェクト管理の精度向上を図ろうとするものである。

2.2.2 ソフトウェアの開発サイクル

ソフトウェアの開発工程はすでに2.2.1項で述べたように各企業の特性，その他の要素から決められるが基本的な開発手順は共通している。

コンピュータ・メーカ及び主要企業におけるソフトウェアの開発段階を整理，比較分析してみると図２－５に示すような基本的な開発サイクルを設定することができる。「システム化要求」工程から「システム設計」工程までが設計段階であり「プログラム設計」から「システム・テスト」工程までを開発実段階であり，これらがプロジェクト管理の対象範囲である。以下には各工程における作業内容を明確に規定する。

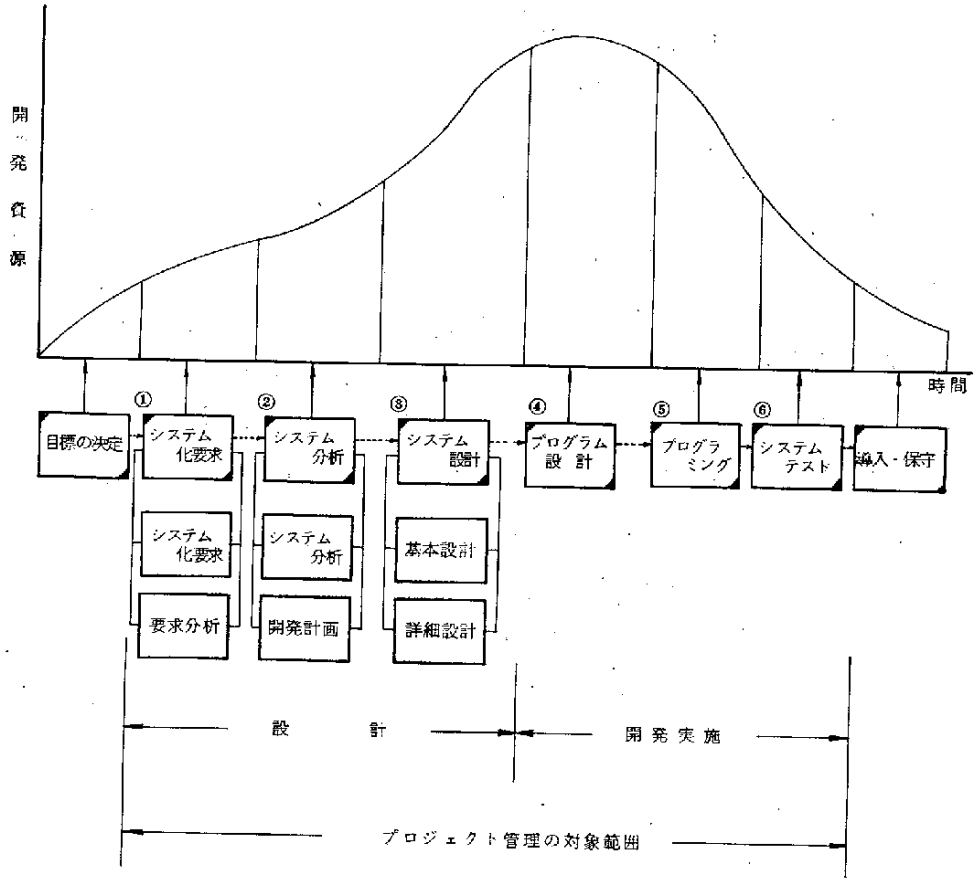


図２－５　ソフトウェア開発のプロセス

(1)　システム化要求工程

(i)　システム化要求段階

ユーザ部門がシステム化する業務の内容，範囲を機能面から検討し，ユーザ部門におけるシステムの要求案を作成する作業である。

具体的な検討内容は次のとおりである。

- ○ システム化の背景，必要性の整理
- ○ システム化の目的，効果の明確化
- ○ 対象業務範囲の明確化
- ○ 人手作業と機械作業全担
- ○ 関連資料帳票類の洗い出し
- ○ システム運用開始時期の確定
- ○ 制約条件の明確化
- ○ レビュ

(ii) 要求分析段階

ユーザ部門で検討した要求内容について，事務処理の流れ，帳票類の統廃合・標準化等システム化するための現状調査，分析を行い，要求事項の妥当性を総合的に分析する。

具体的な主要検討事項は次のとおりである。

- ○ 各要求内容の妥当性のチェック
- ○ 事務処理フローの調査とシステム化対象業務の範囲，処理内容の詮味
- ○ 帳票類の統廃合，標準化の実施
- ○ 経済性，システム化の効果，技術の観点からシステム可否の評価分析
- ○ レビュ

(2) システム分析工程

(i) システム分析段階

ユーザ部門の要求内容を，業務分析調査結果，類似システム，技術動向等をふまえて検討しかつシステム化の環境条件を充分勘案してシステム案を作成する。

具体的な検討事項を示すと以下のとおりである。

- ○ 類似システム，技術動向の調査
- ○ システムの代替案を種々の観点から作成
- ○ 環境条件（ハード，ソフト，制約条件）の明確化
- ○ システム代替案の評価，比較
- ○ システム案の作成
- ○ レビュ

(ii) 開発計画段階

ソフトウェアの設計を実際に行う場合に必要となる開発体制，開発工程，開発工数及びシステムの運用方法等の計画案を作成する。

具体的な検討事項は以下のとおりである。

- ○ 開発方針の作成
- ○ 開発計画の作成
- ○ テスト計画の作成
- ○ 移行計画の作成
- ○ 運用計画の作成
- ○ 訓練計画の作成
- ○ 開発経費の算出，配分
- ○ レビュ

(3) システム設計工程

(i) 基本設計段階

システム分析工程で明確にされたシステム案を開発予算，開発計画等の諸条件をもとに見直して，システムの基本的仕様を設計する。規定する対象範囲はシステムを構成するサブシステムの機能レベルまで設定する。

具体的な検討事項は以下のとおりである。

- ○ システム案の再検討
- ○ システム設計の基本方針の明確化

- システムの基本機能の明確化
- サブシステムの設計
- 人出力情報の基本設計
- コード・メッセージ・テーブル体系の確定
- 信頼性の基本設計
- ファイルの基本設計
- レビュ

(ii) 詳細設計段階

基本設計の内容をさらに詳細な機能単位に分割してシステムの機能構成，プログラム構成，さらにプログラム作成のための諸条件等システムの詳細な仕様設計を行う。

具体的な検討事項は以下のとおりである。

- プログラムの基本設計
- 入出力帳票類の詳細設計
- コード・メッセージ・テーブル仕様の作成
- システムテスト仕様の明確化
- レビュ

(4) プログラム設計

詳細仕様に基づき，プログラミングの方針，プログラムの構造，プログラム相互のインタフェース及び処理手順等を明確化して仕様化する。あわせてプログラムテストのための仕様を作成する。

具体的な検討内容は以下のとおりである。

- プログラミングの方法・手法の明確化
- プログラム構造，インタフェースの確定
- 各プログラムの処理内容，手順の明確化
- プログラムテスト仕様の設定
- レビュ

(5) プログラミング

プログラム仕様に基づいて，コーディングを行い，各プログラムが所定の機能を満足しているかの検査（単体デバック）を行う。

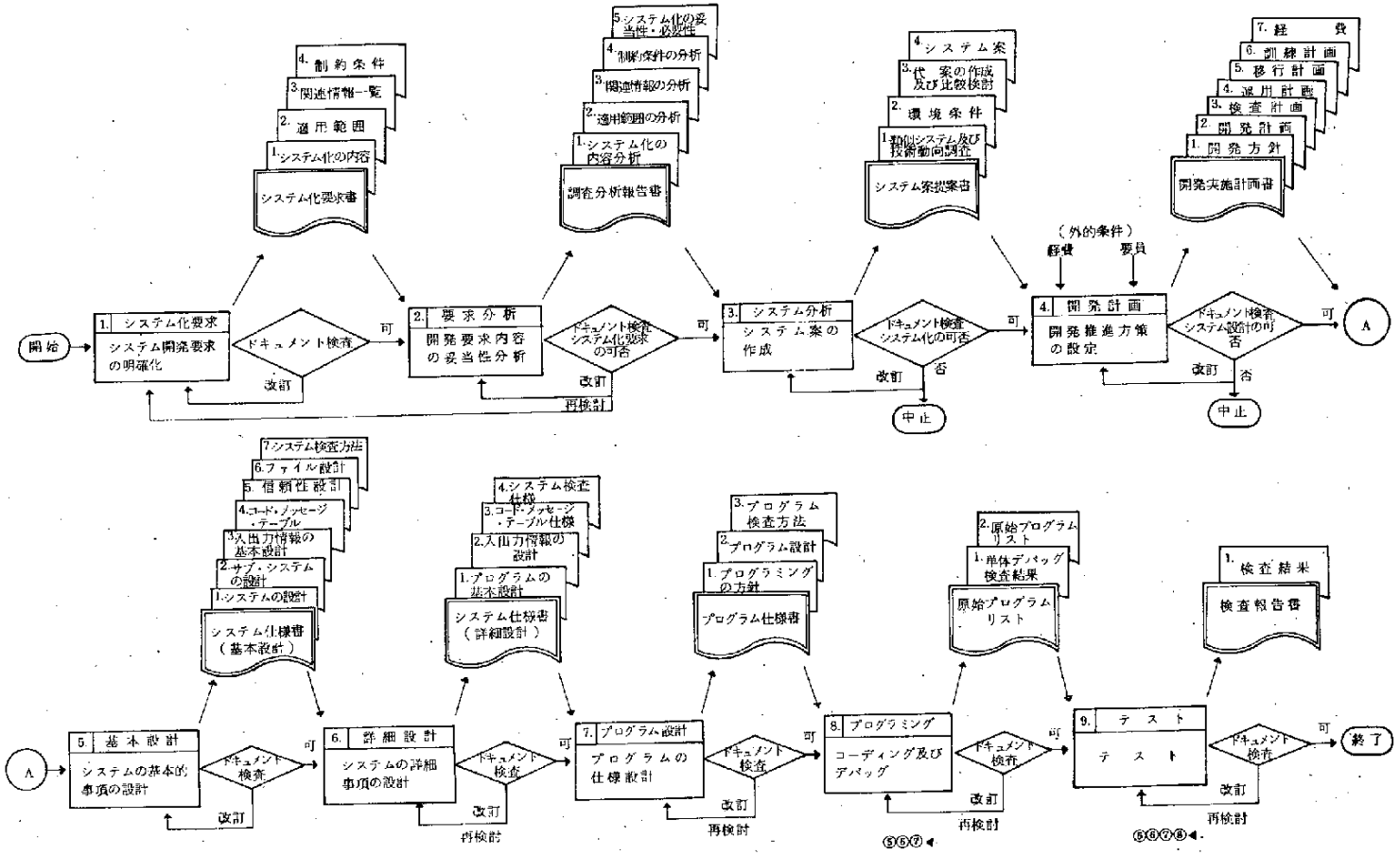
(6) システム・テスト

完成した単体プログラムを結合してシステム全体としての機能及びドキュメントとの対応をチェックし，総合的なテストを行う。

具体的な検討事項は次のとおりである。

- プログラムの結合検査
- システムとしての総合検査
- ドキュメントの検査
- レビュ

以上のような各作業工程の流れと各工程での生産物（各種ドキュメント）の例を図2－6に示す。



ソフトウェアの開発・管理の効率化
研究報告書（行政管理庁）

図２－６　工程区分と中間生産物の例

2.2.3　プロジェクト計画の内容

ソフトウェア開発工程の中で示した開発計画段階において計画の対象となる計画の内容は以後の各工程における作業の見積りである。これらの具体的内容には開発工数や工程の見積りを行う(1)開発資源の見積り計画の立案，(2)経費の算出，(3)テスト計画の立案，(4)移行計画の立案，(5)運用計画の立案，(6)訓練計画の立案等がある。プロジェクト計画が対象とする主要な計画対象業務は上記のうち(1)及び(2)であり，以下にはその内容を述べる。

(1)　資源見積り計画

システム分析工程で作成されたシステム案にもとづきソフトウェア開発工数の見積り，開発工程の設定等次のような開発スケジュール策定の基本的事項を明確にする。

(i)　開発工数の見積り

見積り基準，開発技法その他作業見積りの方法等をもとにシステム開発の各段階の所要工数，要員等システム開発資源の見積りを行う。

(ii)　開発体制の設定

ユーザ部門と開発部門とのコミュニケーション体制，プロジェクト・チームの結成，外部委託先との作業分担等開発要員の構成，および体制を明確にする。

(iii)　開発工程の設定

クリティカルパスの算出および開発工数，体制，要員，運用時期等の調整など各開発段階別のスケジュールを設定する。

(iv)　拡張計画の策定

将来の機能拡張，業務追加等，将来の拡張計画の策定を行う。

(2)　経費見積り

経費見積りの対象となる費用には，開発に要する経費（ハードウェア，ソフトウェア等）と運用に要する経費（レンタル料，保守費　消耗品費等）があり，見積り額及びその算出根拠を明確にして開発計画の予算配分を確定する。

2.3. 開発資源の見積り

2.3.1 開発工数の見積り

(1) 工数設定のねらい

プロジェクト計画における工数設定のねらいの第1は原価管理にある。プロジェクト別，部門別の原価管理にとどまらず採算性をは握したり，プロジェクトの日程，要員，費用に対する総合的な原価管理を行うには的確な工数設定が必須といえる。また，ソフトウェアの開発コストのうち人件費の比率が５０～７０％と大半を占めていることから，工数設定を欠くことは原価管理が無いにも等しい。

工数設定の第2のねらいは進捗管理にある。工程毎に設定された工数とその実績を管理することは作業の進捗管理のみならず工数設定の基準値，設定方法の評価にも役立つ。

さらに，第3のねらいとしてソフトウェアの品質理面からとらえることができる。ソフトウェアの品質を高めるにはそれぞれの工程における作業内容の信頼性が保証されて可能となる。投入工数とプログラムバグ数等から開発の各工程における品質向上のための要因評価を行うことが可能である。

(2) 工数の見積り方法

工数見積りの実際的，具体的な方法をあげると次のようなものがある。

(i) 経験による見積り方法

類似プログラムにより類推する。

(ii) 定量的分析による見積り方法

標準値，係数を設定して算出する。

(iii) 積上げ方式による見積方法

個々の作業見積りを集計

システム開発環境の実態は要員，マシーン等の開発資源に十分な余裕を持っている状況とはほど遠く，限られた資源のもとで限られた期日迄

に開発を行なわなければならないような厳しい状況にあると言ってよい。

これらの見積り方法はそれぞれが単独で使用されるよりむしろ1つの方法で得た見積り結果を他の方法でチェックするなど相互補完的な利用によって見積り精度を高めることができる。

これらの手法のうち多くの企業で使用されている実用的手法は(ii)に示す過去実績データに基づく標準工数値を設定して見積る方法である。この見積りの指標は開発工数に大きな影響を与える要因であるが，代表的なものとして次のような指標（パラメータ）がある。

(i) 開発ステップ数

(ii) 作業内容，工程

(iii) 技術的難易度

(iv) プログラムの種類（アプリケーションプログラム，システムプログラム等）

(v) プロジェクトチームの構成

(vi) プログラミング言語　設計技法，テスト技法等の開発技術

(vii) 計算機および使用形態（バッチ／ＴＳＳ等）などの開発設備

これらの指標のうち実際の見積作業では，(i)開発ステップ数，(ii)開発言語（COBOL，PL／1，FORTRAN，APL 等），(iii)難易度（ランクづけ）を指標として，工数，計算機使用時間，期間を見積り，開発時期を加味して計画を策定するのが標準的な方法である。

(3) 工数見積りの手順と実際

工数見積りの実態は2.1.2項の現状調査結果でも明らかにされているように過去の実積データから設定した工数見積りの基準値をもとに算出，あるいは担当者の過去の経験による推定，類似システムによる見積り等の方法がとられている。

一般的には工数見積りの作業手順は図2－7のような流れを踏んで行われる。以下にはその手順と作業内容を述べる。

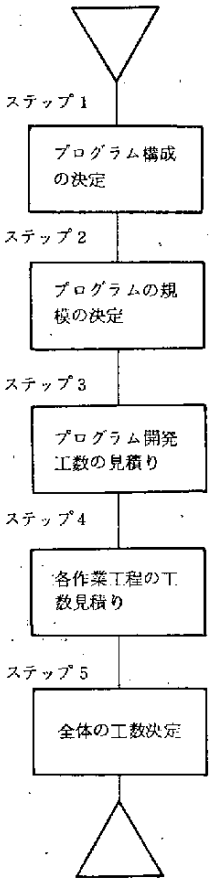


図２－７　工数見積り作業フロー

（ステップ１：プログラム構成の決定）

システムに要求される機能を満足するためにどのようなプログラム構成を採用すべきかを検討し，決定する。

（ステップ２：プログラムの規模の決定）

各プログラムの開発ステップ数の見積りには次のような方法がある。

（例１）　過去に開発した実績データおよび類似プログラムのステップ数をもとに推定する。（事例によるトップダウン方式）

（例２）　プログラムの機能単位に分解して開発言語別にステップ数を積上げて推定する。（積上げによるボトムアップ方式）

（例３）　標準値による見積り方法

例えば平均的なプログラムサイズを開発言語別に小形，中形，大形にクラス分けしてそれぞれの標準値を設定し，それに開発プログラム本数を乗じて開発ステップ数を求める。

（留意事項）

① 事例によるトップダウン方式，積上げによるボトムアップ方式等各種の方法を併用して妥当性を吟味すること。

② 開発規模の見積り誤差が納期，開発コストに直接影響を与えるためあらかじめ見積りの中に加味しておくことも大切である。（例えば見積り値を楽観値，悲観値，目標値の平均値として算出する方法もある。）

③ 見積りの方法はどれも使用してもプログラムの詳細が明確になった時点で絶えず見直しをすることが肝要である。

（ステップ３：プログラム開発工数の算出）

ステップ２で見積った個々のプログラムの開発工数を生産性指数テーブルの該当値から求める。

生産性指数テーブルには次のパラメータによって工数算出の基準値が過去の開発実績データをもとに設定されている。

① 開発言語――アセンブラ，COBOL，FORTRAN，PL／等

② プログラムの形態――バッチ処理，オンライン処理等

③ プログラムの難易度――難，中，易等

④ 技術者のレベル――上級，中級，初級

⑤ 開発範囲――システム設計からシステムテストまで，プログラム設計から結合テストまで，システム設計，プログラミング，テスト等

(注)

$$\text{プログラミング開発工数（個別：Pi）} = \frac{\text{プログラム開発ステップ}}{\text{生産性指数（基準値）}} \times \text{（難易度係数）}$$

（注）　生産性テーブルは難易度を含めて指数の設定をしている場合と，プログラムの新規性，複雑さの度合等から独立に難易度の係数テーブルを設けている場合もある。

なお，見積りは直接工数とあわせて間接工数（管理，サポート，テストツール，移行ツール等）も忘れてはならない。（作業易は２０～４０％の比率）

（計　算　例）

ステップ2の見積値

$$\text{プログラミング開発工数（Pi）} = \frac{900\text{ステップ}}{60\text{ステップ／人日}} = 15\text{人日}$$

〔プログラミング生産性テーブル〕

開発言語＼プログラム難易度	難	中	易
COBOL	50	60	75
FORTRAN	20	30	35
アセンブラ	15	30	40

単位ステップ／人日

$$\text{プログラミング総開発工数} = \sum_{i=1}^{n} \text{個別プログラムPiの工数}$$

（n＝プログラム総本数）

（留意事項）

① 見積りの見直し，実績データによる生産テーブルの見直しが見積り精度向上の必須条件である。

② 生産性指数テーブルは直接工数以外に間接工数を含めるかどうかでその値が異なる。

（ステップ４：各作業工程の見積り）

プログラミング工程以外の各作業工程の工数見積りをステップ３で算出した工数をもとに行く。

$$\text{各工程の工数}(Di) = \text{プログラミング総開発工数} \times \frac{\text{各工程の工数比}}{\text{プログラミング工数比}}$$

各作業工程の工数比はシステム化要求分析工程からシステムテスト工程迄の工程区分ごとの実績データをもとに設定したものである。

（計　算　例）

ステップ３で求めた総工数

$$\text{システム設計工程の工数} = 250\text{人日} \times \frac{16}{20} = 200\text{人日}$$

〔各工程の工数比（％）〕

基本設計	詳細設計	プログラム設計	プログラム作成	テスト
8	16	20	28	28

（ステップ５：全体の工数決定）

以上を総合してプロジェクト全体の工数決定を行う。

工数見積りにあたっての生産性テーブル及び作業工数比は各企業によって異なるが参考例を示すと表２－３～表２－６がその一例である。

表２－３．標準プログラミング・ステップ数コンピュータ使用比率

① プログラム １本の ステップ数	② 難易度	③ 標準プログラミング・ステップ数 〔プログラマ１人が１日(5時間)に書けるステップ数〕 コンパイラ	 アセンブラ	④ 工数当りコンピュータ使用比率（％）	⑤ ①の平均ステップ数	⑥ プログラム１本当りのコンピュータ使用時間（分）
２００未満	簡易	５０	４５	1.5	１５０	１５
	普通	４５	４０	1.8		２０
	高通	４０	３５	2.0		２５
２００以上	簡易	４５	４０	1.5	３００	３０
～	普通	４０	３５	1.8		４０
４００未満	高通	３５	３０	2.0		５０
４００以上	簡易	４０	３５	1.5	５００	６０
～	普通	３５	３０	1.8		８０
６００未満	高通	３０	２５	2.0		１００
６００以上	モジュール単位に分割し，上記３区を組み合わせる。				－	－

（注）1. プログラミングとはプログラム設計から単体テストまでをいう。
2. プログラムStep数はPROCEDUREのみをいう。
3. １日は５Ｈとして算出

表２－４　作業区分及び人工配分規準値

人工配分値 / 作業区分	過去のプロジェクトの実績値	現在見積りにおいて用いようとしている値
システム設計・基本設計	２０％	２０～２５％
機能構成設計	２４％	１０％
プログラム詳細設計		１５％
コーディング・デバック 単体テスト	３５％	２５～３０％
システムテスト	１０％	１５％
ドキュメント・管理	１１％	１０％

－ソフトウェア開発費用に関する調査研究（日本情報処理開発協会）－

表２－５　ソフトウェア開発工程別工数比（昭和５２年度）

%

分野 / 開発工程	科学技術計算分野	プロセス制御関係	事務処理分野	ベーシック・ソフトウェア	その他	平均
基本設計	12	14	7	8	8	10
システム設計	20	11	16	12	15	15
プログラム設計	30	21	21	20	16	22
プログラム作成	18	29	28	36	39	29
テスト	20	25	28	24	22	25
合計	100	100	100	100	100	100

表２－６　ソフトウェア開発工程別時間（昭和５２年度）

%

分野 / 開発工程	科学技術計算分野	プロセス制御関係	事務処理分野	ベーシック・ソフトウェア	その他	平均
基本設計	10	16	12	13	11	13
システム設計	18	15	19	16	16	17
プログラム設計	21	21	22	23	21	21
プログラム作成	27	25	21	25	32	24
テスト	24	23	26	23	20	24
合計	100	100	100	100	100	100

（ソフトウェア産業振興協会）―
―ソフトウェア産業の動態および将来ビジョンに関する調査報告書

2.3.2　計算機使用時間の見積り

計算機使用時間に余裕のある企業では，見積りを行っていないケースもみられるが，開発工数の設定と同様に算出の基準値を設定して計算機使用時間を見積るのが一般的であるといえる。

基準値は次のような単位で設定されている。

(1)　１ラン当り処理時間（ＪＯＢ実行時間，ＣＰＵ時間）

(2)　ステップ当りの処理時間

(3)　平均処理時間に一定の係数を乗じた値

(4)　プログラマ１人月あたりの処理時間

これらの基準値は次のようなプログラムの属性によってそれぞれ異なるものである。

(1)　プログラム言語の種類

　○　アセンブラ，COBOL，FORTRAN，PL／1等

(2)　プログラムの性格

　○　新規作成，プログラム・メンテナンス

(3)　テストの性格

　○　単体テスト，結合テスト

以上の各要素から設定された基準値をもとに計算機（ＣＰＵ）の利用率を加味して計算機時間を見積り，これを各工程に配分する。

その一例を次に示す。

$$\boxed{\text{全CPU時間(T)}} = \left(\begin{array}{c}\text{プログラミングから}\\ \text{テストまでの総工数}\end{array}\right) \times \left(\begin{array}{c}\text{平均}\\ \text{処理時間}\end{array}\right) \times \left(\begin{array}{c}\text{CPU}\\ \text{利用率}\end{array}\right)$$

$$\boxed{\text{各工程}\left(\begin{array}{c}\text{プログラミング}\\ \text{テスト}\end{array}\right)\text{のCPU時間}\ (\text{Pt})} = \begin{array}{c}\text{全CPU}\\ \text{時間}\end{array}_{(T)} \times \frac{\left(\begin{array}{c}\text{各工程のCPU時間比}\end{array}\right)}{100}$$

〔各工程のＣＰＵ時間比〕

基本設計	詳細設計	プログラム設計	プログラム作成	テスト
0	0	(5)	(50)	(55)

2.4　開発費用の見積り

2.4.1　見積り費用項目

費用見積り項目のうち大きなウエイトを占めているのが人件費と計算機使用料でありその構成は人件費６０～７０％，マシン使用料２０～３０％であり，人手に大きく依存しているソフトウェア開発の現実の姿が明確にあらわれている。

見積り対象費目を列挙すると主要なものは次のとおりである。

① 人件費
② 計算機使用料
③ データ作成費
④ 材料費，消耗品費
⑤ 旅費・交通費
⑥ 資料費，印刷費
⑦ 会議費
⑧ パンチ代
⑨ 技術料
⑩ 一般管理費

2.4.2　費用見積り手法

ソフトウェアの開発費用は前述のようにほとんどが人件費であり，これ

に計算機使用料と直接経費，一般管理費，技術料を加えたものであり，その見積り手法は次の２つに大別できる。

(1) 開発工数による費用見積り

既に2.3項で述べた見積り方法によって求められた各工程毎の開発工数（人時，人日，人月）に人件費単価を乗じて人件費を見積る方法である。

(2) 開発ステップ数による費用見積り

最終的なプログラムステップ数を見積り，ステップ当りの単価を乗じて人件費を算出する方法である。

これら２つの手法は併用されているところが多い。

一般管理費については総開発費の一定比率（およそ１０％）を計上しているケースが多く，技術料は人件費に加味しているところが多い。

2.5 日程計画の作成

2.5.1 日程計画作成の留意点

日程計画の実態は，図２－３で明らかなようにあらかじめターゲットが決められ，それに合わせて過去の実績データ等をもとに各工程の工数割当てが行われている状況にある。

日程計画を立案するうえで特に留意すべき事項には次のものがあげられる。

① 各作業工程とその作業内容を明確に規定すること。

② 各工程の生産物であるドキュメントについて記述内容，記述形式を明確に規定しておくこと。

③ 各工程の完了時にレビュー工程を設け，ドキュメントの審査，承認手続を規定して，工程の品質確保，各工程の完了のオーソライズを行うこと。

④ 定時的に日程管理が可能となるように管理帳票を規定しておくこと。

このような留意点を踏まえて，以下に日程計画の技法として，協同乳業

（株）の市川寧氏の提唱による，いつでも，どこでも，誰でもが使用可能なようにＰＥＲＴを大衆化したハンディ・パード（Handy PERT)について述べる。

2.5.2　ハンディ・パートの基本手順

ハンディ・パートの基本手順は，図２－７に示すとおりである。

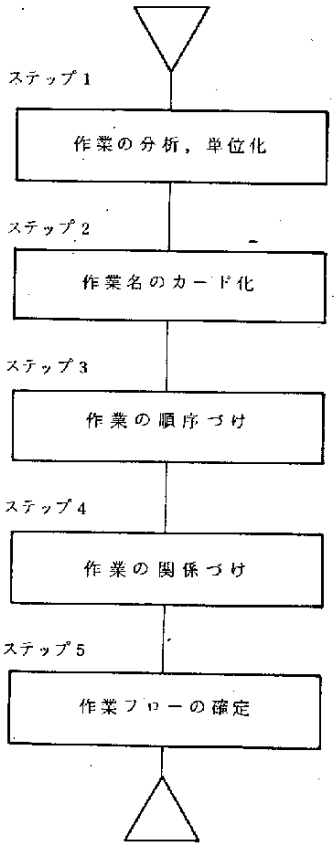


図２－７　ハンディパートの基本手順

以下には各作業手順（ステップ1～ステップ7）の内容を示す。

まず，作業メンバはプロジェクトの知識，経験を有する5～8名でチーチを構成することが集団思考の効果をあげる面で望ましい。プロジェクトを構成する作業名，所要時間などを記述するものとして図2－8に示すHP（ハンディ・パート）カードを用意する。

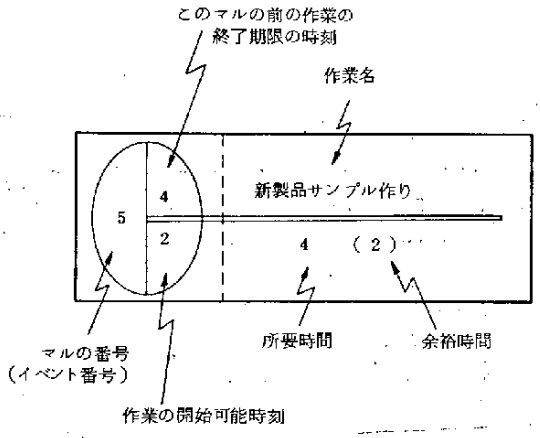


図2－8　HPカード

- ステップ1：作業の分析，単位化

  開発の対象となるプロジェクトを実施するうえで必要となる作業を洩れなく洗い出し，作業項目を整理する。作業内容が大まかすぎるものについては，さらに細かく分割する。

  また，逆に細かすぎるものについては一つにまとめるなどして，各作業項目，内容のレベルをそろえておく必要がある。

- ステップ2：作業名のカード化

  ステップ1で確定した作業名をHPカードに記す。（図2－8参照）

- ステップ3：作業の順序づけ

  作業名が記述されたＨＰカードを作業の性格（先行・後続作業，並行作業）を考慮しながら，作業手順に沿って配置する。

- ステップ4：作業の関係づけ

  作業相互の関係づけ，すなわち該当作業を開始するために終了していなければならない作業項目，あるいは該当作業終了後開始可能な作業項目などそれぞれの関連づけを行う。

- ステップ5：作業フローの確定

  作業の関係づけを確認した後，最初の作業から最終作業までの作業フローを矢線で示しネットワークが完成する。

2.5.3　ＰＥＲＴ計算

ＰＥＲＴをさらに有効に活用するためには，2.5.2項で作成したネットワーク図をもとに図２－９に示す手順でＰＥＲＴ計算を行うことが必要である。

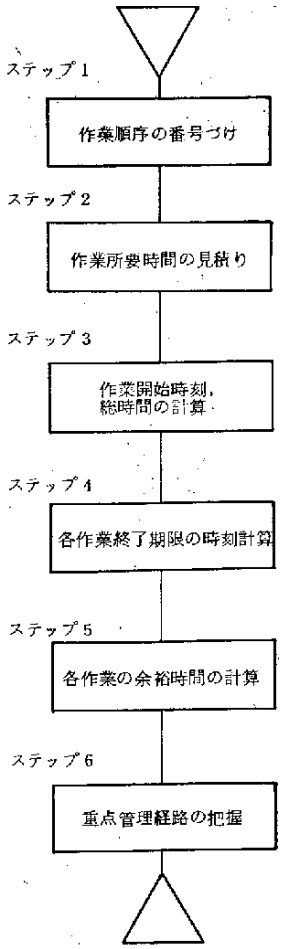


図２－９　ＰＥＲＴ計算手順

・　ステップ１：作業順序の番号づけ

最初の作業から始めて，順次右に向かって番号を入れる（図２－１０）番号は，左から右へ作業の先行，後続の順にしたがって，右のマルの番号が左のマルの番号より大きくなるように入れる。

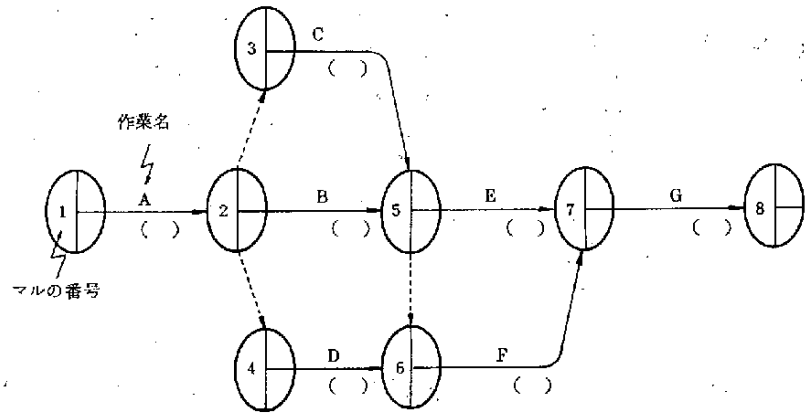


図２－１０　ステップ１：番号づけ

・　ステップ２：作業所要時間の見積り

各作業の所要時間を見積り，矢線の下に記入する（図２－１１）。点矢線は実態のない作業なので所要時間は「０」とする。

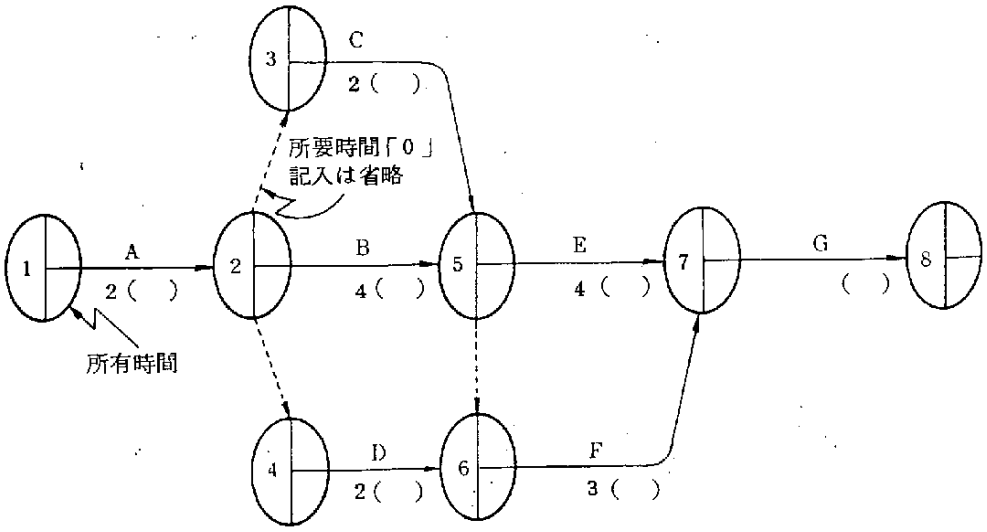


図２－１１　ステップ２：所要時間の見積り

- ステップ３：作業開始可能時刻，総所時間の計算

図上で，各作業の開始可能時刻を計算し，最後にプロジェクトの総所要時間を求める（図１－１２）。

まず，スタートのマルの右下の欄に「０」を入れ，それにつづく第１の作業の所要時間を加えたもの（０＋２＝２）を，つぎのマルの右下欄に記入する。最後のマルに至るまで順次この形で累加計算する。複数の作業が一つのマルにつながる合流点では，それぞれの流れについて累加計算した中での最大値を入れる。

最後のマルに入る数字が，そのプロジェクトの総所要時間である。最後のマルの右下欄に記入された「総所要時間」を，そっくりそのまま右上欄に転記する。

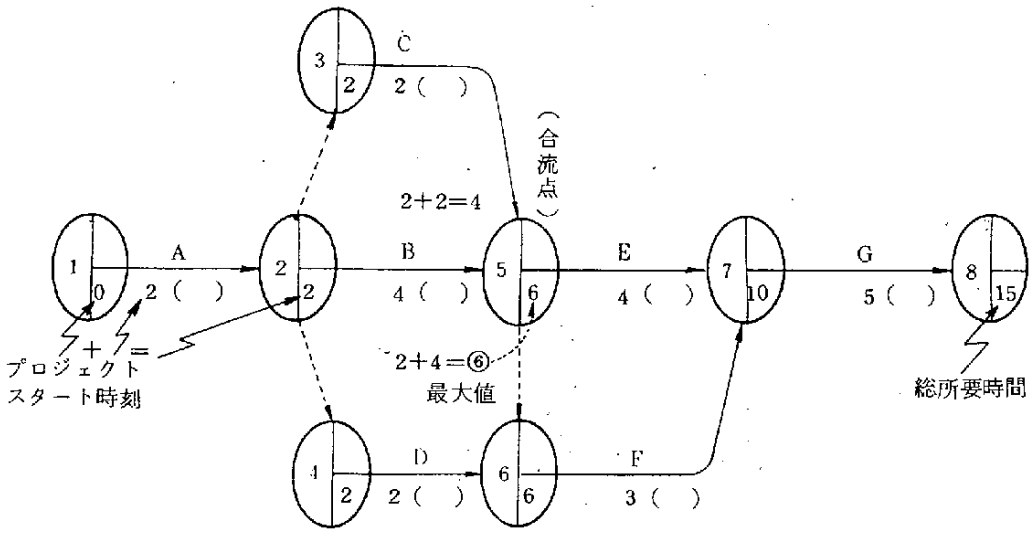


図２－１２　ステップ３：各作業の開始可能時刻，総所要時間の計算

- ステップ４：作業終了期限の時刻計算

こんどは，最後のマルを出発点として，作業の流れとは逆方向に（マルの右上欄の数字）－（そのマルの前にある作業の所要時間）→（その作業の前にあるマルの右上欄に入れる）といった操作をくり返し，そのマルに入る作業の終了期限の時刻を計算スタートのマルまでもどる（図２－１３）

ここで・の留意点は「複数の流れの逆合流点では、それぞれの流れの計算値の最小値をとる」ということである。

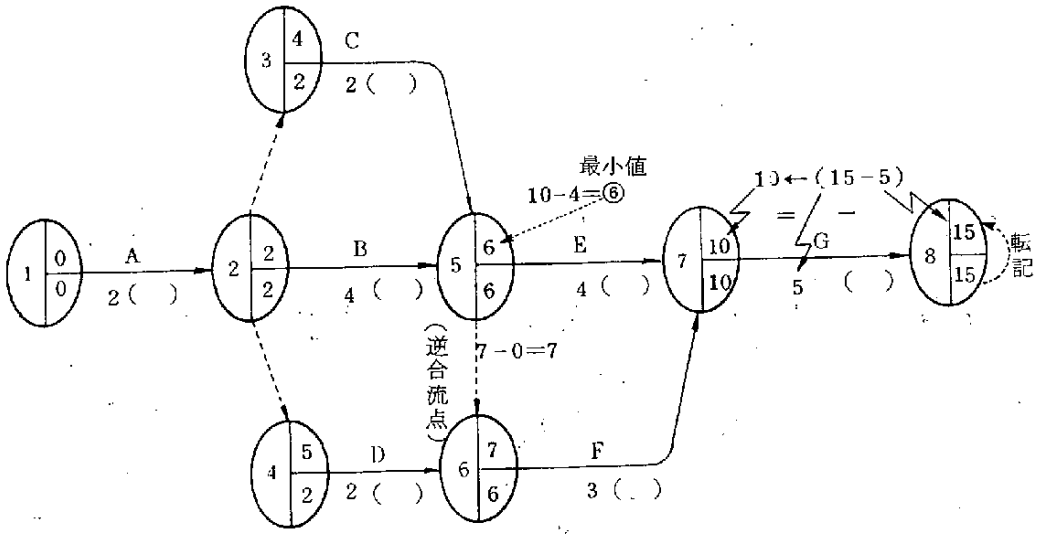


図２－１３　ステップ４：各作業の終了期限の時刻の計算

- ステップ５：作業余裕時間の計算

  各作業について，つぎの計算により余裕時間を求めカッコの中に記入（図２－１４）。

  （作業の終了期限の時刻）－｛（作業の開始可能時刻）＋（作業の所要時間）｝このようにして計算された余裕時間は，その作業のみの余裕時間を加えただけのユトリがその作業群にあると考えてはいけない。

- ステップ６：重点管理経路の把握

  重点管理経路をつかまえる（図２－１４）。

  スタートのマルから出発して，余裕時間が「０」の作業の流れを赤線で塗る（または，太めの線にする）。

  この赤線を引いた経路は，まったく遊びのない経路，いいかえれば，もしこの経路上にある各作業が一日でも遅れれば，それだけ工期も遅れるという性格のものである。

また，総所時間をもう少し短縮したいという場合，真先にチェックすべき経路ともいえる。そういった意味から，この経路を重点管理経路という。

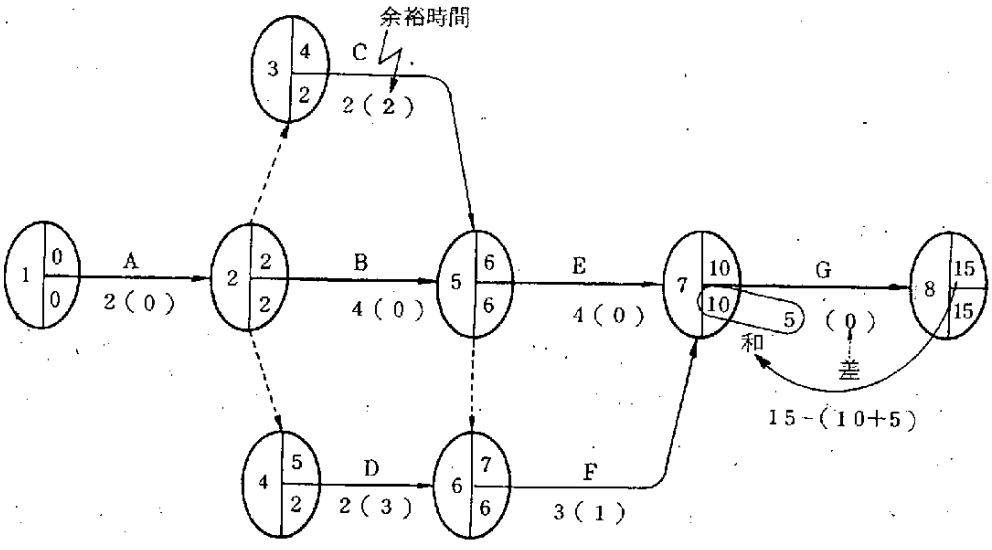


図２－１４　「余裕時間」の計算と「重点管理経路」

ＰＥＲＴ計算の結果，総所要時間が当初予定していた期限をオーバーした場合には，ネットワーク上の各作業を改めて見直し，

① 各作業の所要時間の算出は妥当かどうか。

② 直列作業で並列作業にもっていけるものはないか。

③ 余裕のある作業の流れから，余裕のない作業の流れの方に「資源」，（ヒト・モノ・カネ等）をふりむけることで短縮できないか。

といった検討を加えることによって，予定工期内に収める努力が必要となる。

また，計画が実施段階に入った時，適宜作業の進行状況をチェックし，予定工期に遅れないよう手を打っていく必要がある。

3. プロジェクトの実施

本章では，プロジェクトを実施する場合の工程管理，品質管理，費用管理，外注管理について，それぞれの基本的な考え方，及び管理の方法や留意点等を明らかにする。なお，各種標準化や変更管理を対象とするコンフィグレーション管理については品質管理の中で触れることにする。

3.1 工程管理

第1章でも述べたように，コンピュータ・ユーザにとって，工程管理はプロジェクト管理上，最も困難と感じている作業であり，又今後最も力点を置きたい作業として位置付けられている。ソフトウェア開発は，予定期間内に終了することは大変難かしく，予定の2倍近くかかってしまうことも珍らしくない。本節では工程管理を行うにあたっての前提条件，つまり基本的要件を初めとして，工程区分や各工程別の管理上のポイント，工程管理のための図表等について説明する。

3.1.1 工程管理の要件

工程管理を正しく行おうとすれば，次のような要件が満足されていなければならない。

① 工数見積りが正確に行われていること。

予算オーバ及び工期が遅れた理由として，工数や費用の見積りの甘さが原因と思われるものが非常に多い。このことからも，最初の見積りを正確にすることがいかに重要なことであるかが分かる。

② ユーザのニーズの吸収は正確に行うこと。

これは，工程管理に限らず，全ての管理の前提となることであり，ユーザのニーズに合わない，つまり，無駄なソフトウェアを開発するようなことは絶体に避けなければならない。

③ 作業工程が明確に規定されていること。

例えば，システム化要求，システム分析，システム設計，プログラム設計，プログラミング，システム・テストといったように，プロジェクトの開始から終了までの作業工程が明確にされており，かつ，各工程での作業の内容が示されていることが必要である。

④ 各工程の生産物が明確に規定されていること。

各工程の生産物，特にドキュメントに関しては，③の作業内容に合わせて，その記述内容，記述形成式が明確に規定されていることが必要である。各工程の終了の認定は，このドキュメントの検査によってなされる訳であるから，これらの規定は特に重要なものである。

⑤ レビュ工程が設けられていること。

各工程の終了を完璧にチェックするために，また，多少の修正を可能にするためにも，各工程の切れ目には必ずレビュ工程が設けられていなければならない。これは，問題が生じた場合のフィードバックの幅を少なくして，最終的な製品の品質を高めるうえにも極めて重要なものである。

⑥ 中間生産物の審査・承認手続が規定されていること。

各工程の終了を正式に認定するために，審査・承認手続が規定されていなければならない。作業担当者が自分一人でレビューし，自分一人で終了の認定をするような方式では，高品質の生産物を得ることはできない。G.J.Myers は，ソフトウェアといえども審査のためには特別な技術を必要とし，また，作成者と審査する者の間には不良摘出意欲に歴然たる差があることなどから，審査部署の必要性を力説しているが，これは傾聴に値する。

⑦ 工程管理用の図表が規定されており，かつ定期的な報告が義務付けられていること。

工程管理を正しく進めるためには，各工程の終了時にまとめてチェッ

クするのではなく，定期的（例えば毎週）に，作業担当者より進捗報告書を提出させ，ネットワーク図に書き込むというように，管理用の図や帳票が規定されていなければならない。

定期的に進捗をチェックすることは，問題の早期発見を可能にし，その対策も容易にする。なお，ここで重要なことは，「終了した割合」に重点を置くのではなく，「残された割合」と「残された時間」に重点を置くことである。

3.1.2 工程区分の明確化と中間生産物

他の産業のように，各ジョブ・ショップを仕掛品が流れていくのとは異なり，1つのプロジェクト・チームの中で，長い仕掛期間をかけて徐々に製品が完成されていくソフトウェア生産にとって，工程の進捗は握は大変難かしい。そこで，特に重要となることは，工程の区分とその各工程における作業内容を明確にすることであり，そのための作業手順が確立されていることである。（工程区分と中間生産物の例は第2章図2－6を参照されたい）

3.1.3 工程管理上の留意点

(1) 工程管理の基本的な考え方

工程管理を実際に行う場合に最も重要なことは，品質管理と組み合わされて実施しなければならないということである。品質管理を欠いた工程管理は，単に形式のみの管理にとどまり実質的価値は持たない。このような管理を行っていると，"昨日まで，スケジュール通りだったものが，今日になって突然1ヶ月遅れであることが分かった。"というようなことになりかねない。つまり，優れた工程管理を行うためには，中間生産物の品質をチェックするためのチェック・リスト等，各工程の品質を確認する手段が入っていなくてはならない。なお，品質管理そのもの

については次節で詳述する。

(2) 進捗度評価

工程管理のための方法としては，期限管理と進捗率管理の２つがある。前者は開発工程の途中に設けたマイルストーンに到着したかを管理するものであり，後者は，マイルストーン間のどの位置にあるかを管理するものである。

期限管理を行うためには，マイルストーンをきめ細かく規定しておく必要があり，少なくとも１ヶ月当り２個以上のマイルストーンを設定しておいた方がよい。また，期限管理のための帳票を用意し，それを用いて管理することになる。大規模なプロジェクトにおいては，マイルストーンの数が多くなり，ともすると期限オーバーを見過ごしてしまうことがある。このような状況では，期限管理システムを用意し，期限予告，期限オーバーの警告，期限に対する遅延率等が出力できるようにすることが必要である。なお，これらに関連する帳票については次項でその例を示すことにする。

また，進捗率管理を行うためには，マイルストーンとマイルストーンとの間の進捗度を表わす適切な指標が必要となる。

以下，期限管理をより科学的に行うための作業達成度の考え方の例を示す。

① 重み付け

作業をＷＢＳでブレイクダウンしていくとき，子ノード間の作業量の比重を示す重み（推定値）を与える。図３－１で，ノードｉの重みをWi で表わすと，

$$\sum_{i=1}^{n} Wi = 1 \quad \cdots\cdots\cdots\cdots\cdots\cdots〔\ 1\ 〕$$

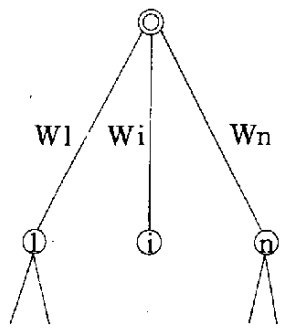


図３－１　重みつき作業階層構造

とする。

また，ノードｉにフェーズ重みベクトル（Ui１，Ui２……，Uik）を定義する。

これは作業区分間の作業量の比重を示し，Ｕｉｊ＝０はノードｉに 対する第ｊ作業区分の作業は存在しないことを意味する。ここでも

$$\sum_{j=1}^{k} Uij = 1 \quad \cdots\cdots\cdots\cdots\cdots\cdots〔\ 2\ 〕$$

② 終端ノードの作業達成度

終端ノードｉの第ｊ作業区分の達成度ｔｉｊは，達成マイルストーン数と設定マイルストーン数の比で定義する。ノードｉの達成度 tiは次式から求める。

$$\widehat{ti} = \sum_{j=1}^{k} Uij \cdot tij \quad \cdots\cdots\cdots\cdots〔\ 3\ 〕$$

③ 非終端ノードの作業達成度

非終端ノードの作業達成度は下位ノードの達成度の重み付き和で求められる。

$$\widehat{to} = \sum_{i=1}^{n} Wi \cdot ti \quad \cdots\cdots\cdots\cdots\cdots〔\ 4\ 〕$$

第ｊ作業区分の達成度は下位ノードの第ｊ作業区分の達成度から次式で求める。

$$toj = \frac{1}{\sum_{i} Wi \cdot Uij} \sum_{i} Wi \cdot Uij \cdot tij \quad \cdots\cdots\cdots〔\ 5\ 〕$$

ノード０の作業達成度は作業区分の達成度からも計算できる。

$$\widehat{to} = \sum_{j} Uoj \cdot toj \quad \cdots\cdots\cdots\cdots〔\ 5\ 〕$$

(4)式と(5)式から求められる$\widehat{to}$は等しくなければ ならないので重み間には次の制限が課せられる。

$$Uoj = \sum_{i} Wi \cdot Uij \quad for \quad 1 \leq j \leq k \quad \cdots\cdots〔6〕$$

(3) 各工程別管理上のポイント

ここでは，ソフトウェアの開発工程を，システム化要求，システム分析，システム設計，プログラム設計，プログラミング，システム・テストと考え，それぞれの工程における工程管理上のポイントを示す。

① システム化要求，システム分析工程

定義すべき項目の詳細リストを定めておき，その項目の埋まり具合で判断する。詳細リストの内容は，ユーザからみたソフトウェアの品質（例えば，合目的性，操作性，信頼性，互換性，性能，保守性等）に関する項目を網羅したものでなければならない。

② システム設計工程

システム化要求，システム分析と同様に，システム設計工程で定義すべき項目の詳細リストの埋まり具合で判断する。項目はハードウェアとのインタフェース，プログラム間のインタフェース及び，これらの間の機能分担を明確に規定するものではなくてはならない。

③ プログラム設計工程

まず外部仕様の決定状況は，ユーザーズ・マニュアルの作成ページ数により判定する。この前提として，ユーザーズ・マニュアルの記述内容が詳細に規定されていなければならない。また，内部仕様の決定状況を判断するための指標として，設計完となったモジュール数が使用できる。この場合も，内部仕様記述用ドキュメントの詳細な規定が用意されていることが必要である。

④ プログラミング工程

プログラム完（コンパイル・エラーが0となったもの）のソース・ステップ数を指標とする。この場合，プログラム全体だけでなく，モジュール単位に進捗率を管理する必要がある。

⑤ システム・テスト工程

テスト工程は，プログラムが設計仕様どおり動作することを確認し，また動作しない部分を摘出・修正する工程である。従って，テスト工程はソフトウェアの品質を確認し，向上するための工程でもある。

テスト作業を実施するためには，テスト項目の設定，テスト・データの作成，テスト・プログラムの作成，テスト・ジョブの作成等数多くの準備作業が必要である。この準備作業を十分考慮せずにテスト作業を開始すると，最初の１～２ヶ月は準備だけに費やされ，実際のテスト作業がなかなか進まないということになりかねない。このような準備作業はあらかじめ日程を定め，期限管理することが大切である。また，テストそのものの進捗の管理は，当初予定したテスト項目の完了度合によって行われる。

3.1.4 工程管理の図表

工程管理には様々な管理図や帳票類が利用されている。その利用状況は図３－２のとおり，ガント・チャートの利用が最も多く（５２.９％），ついでネットワーク図（３８.７％）である。

ここでは，ガント・チャート，マイルストーン・チャート，ネットワーク図，トレンド・チャートの４つの代表的な管理図について，その特徴と概要を説明するほか，マイルストーン予定通知書，プロジェクト進捗報告書など６つの帳票例を紹介する。

(1) ガント・チャート

ガント・チャートはバー・チャートとも呼ばれ，最も広く用いられているものである。ガント・チャートには次のような特徴がある。

① 計画が比較的簡単に行える。

② 見やすく，解りやすい。

③ 作業相互の関連がはっきりしない。

④　工程管理上の要点がはっきりしない。

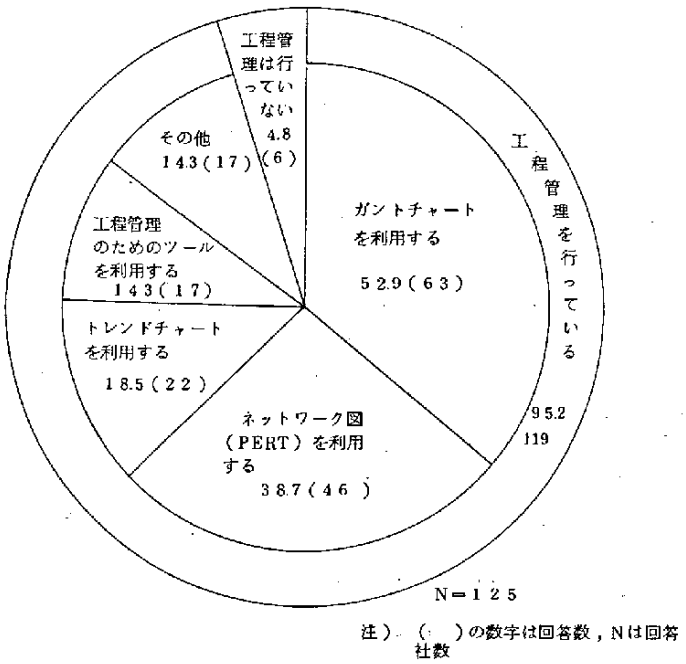


図３－２　工程管理の方法

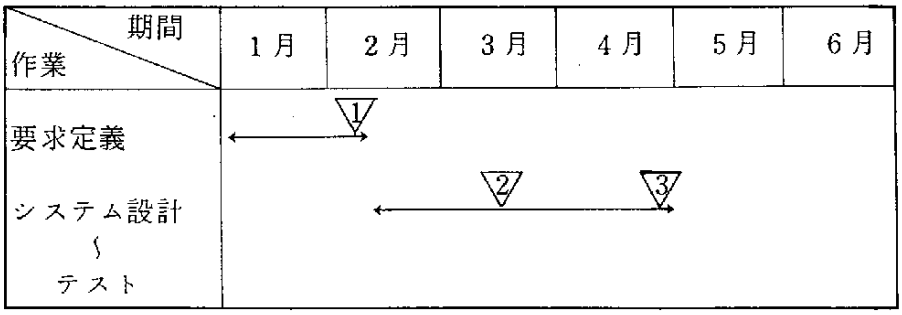
この方法を用いる場合の留意点は，１つのバーの長さをあまり長くならないようにすることである。１つのバーの長さは２週間から１ヶ月ぐらいにしておかないと，進捗の把握が極めて曖昧になってしまう。図３－３にガント・チャートの例を示す。

期間 / 要員	57／4	／5	／6	／7	／8
ＳＥ　Ａ	業務ａ ←	業務ｂ →			
ＳＥ　Ｂ		業務ｂ ←	業務Ｃ →		業務Ｅ →
プログラマＣ			業務Ｄ ←	業務Ｆ →	
プログラマＤ			業務Ｄ ←	業務Ｇ →	

図３－３　ガン・トチャートの例

(2)　マイルストーン・チャート

この方法は，ガント・チャートの欠点の一部を補うものであり，スケジュール上で重要な仕事が完了していなければならないポイントや，ある意思決定をしなければならないポイントをマイルストーンとして，ガント・チャート上に表わしたものである。これによって工程管理上の要点を明確にし進捗のは握を容易にすることができる。マイルストーン・チャートの例を図３－４に示す。



マイルストーン１：①問題仕様書作成　②ユーザによる確認

マイルストーン２：①基本設計作成　②ユーザによる承認

図３－４　マイルストーン・チャートの例

(3) ネットワーク図（ＰＥＲＴ図）

ネットワーク図は，正確な工程管理を行うためには最適なものであり，次のような特徴がある。

① 作業の順序，相互関係が明らかになる。

② 全体の工数は，各作業単位の工数の積み上げにより計算される。

③ 工程の途中で，これからの工数予測が簡単に行える。

④ ネットワーク図の作成にはかなりの時間がかかる。

ネットワーク図については，費用管理にも用いられるので，費用管理の中でも触れる。

ネットワーク図の例を図３－５に示す。

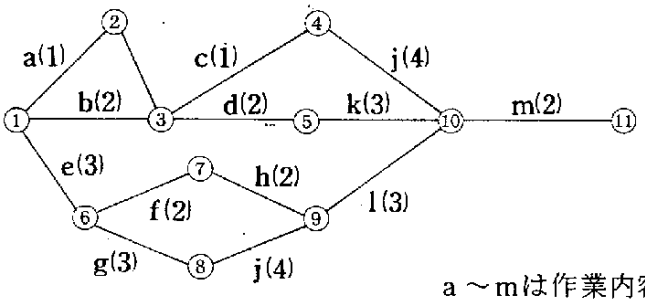


図３－５　ネットワーク図の例

(4) トレンド・チャート

トレンド・チャートは費用管理と工程管理を同時に行うものであり，図３－６に一例を示す。図からも明らかなように，計画と実績との差の累積された結果が各円の大きさとして表わされている。矢印の方向によって，この偏差の意味は次のようになる。

○ 右上方向は，スケジュールより遅れ，予算を超過している。

○ 左上方向は，スケジュールより進み，予算を超過している。

○ 左下方向は，スケジュールより進み，予算を下回っている。

○　右下方向は，スケジュールより遅れ，予算を下回っている。

トレンドチャートの特徴は次のとおりである。

①　プロジェクト全体としての費用，工程の管理には最適である。

②　スケジュールと費用の進み具合，ならびにその傾向がよくわかる。

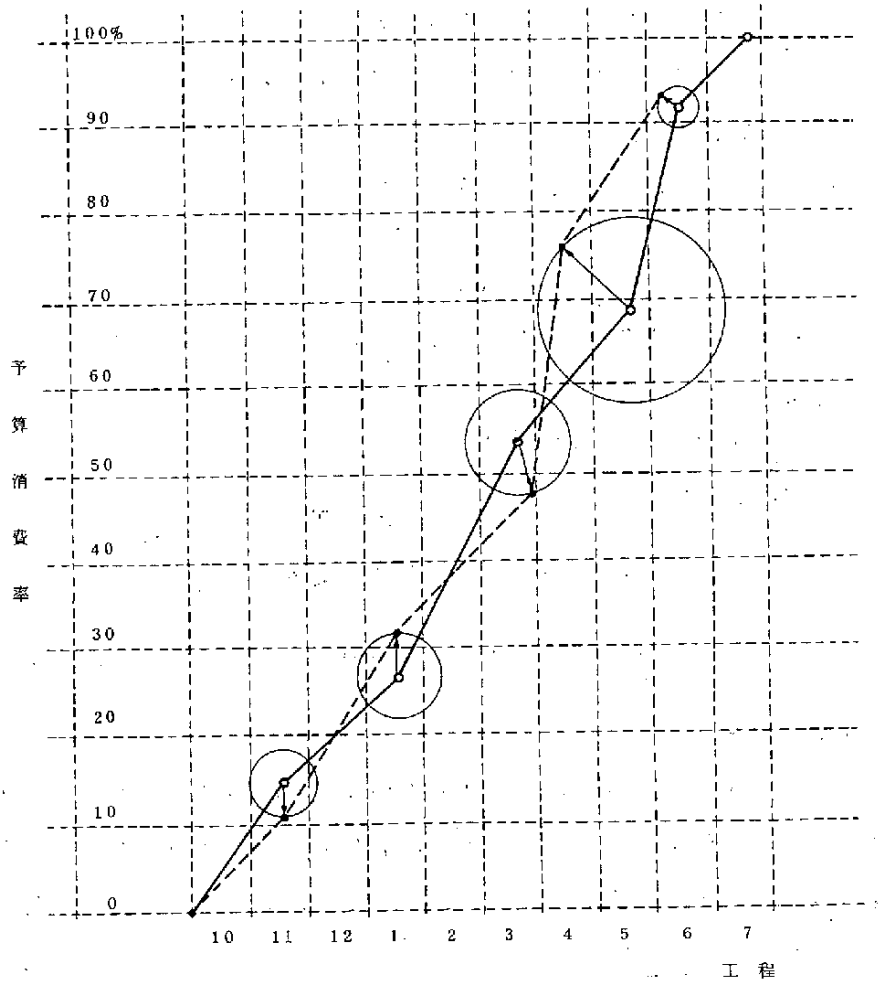


図3－6　トレンド・チャートの例

(5) 工程管理のための帳票

工程管理のためには図のみでなく，各種の帳票が使用されるのが一般的である。ここでは，次のような帳票の例を示す。

① マイルストーン予定通知書（表３－１）

② プロジェクト進捗報告書（表３－２）

③ 作業報告書（表３－３）

④ 問題処理表（表３－４）

⑤ 個人別進捗管理表（表３－５）

⑥ モジュール別進捗管理表（表３－６）

表３－１　マイルストーン予定通知書の例

５７.１.２０現在

責任者	マルルストーン名	コード	予定日	余裕	
A	モジュールAの仕様決定	D10A	1／25	＋ 5	
B	テストデータの作成	TESP−3	1／23	＋ 5	
C	マシンの調整	MA	1／10	−10	＊
D	モジュールC～Eの単体テスト	STC～STE	1／21	＋ 1	

（注）　＊印は予定をオーバーしていることを示す。

表３－２　プロジェクト進捗報告書のフォーマット例

（　　）プロジェクト進捗報告書

マネジャ	リーダ

グループ名	
対象期間	年　日

年　月　日

作業項目名	上段:目標 下段:実績	当月		累計		作業時間	
						当月	累計
			%		%		

コメント欄

表３－３　作業報告書のフォーマット例

個人別　作　業　報　告　書

年　　月　　週

プロジェクト名：

氏　　名：

リーダ	担　当

作業項目名		当　週		累　計		作業時間	
						当週	累計
	目標				%	H	H
	実績		%		%		
	目標						
	実績						
	目標						
	実績						
	目標						
	実績						
	目標						
	実績						
	目標						
	実績						

担当者・コメント欄
リーダ・マネジャ・コメント欄

記入要領：目標,実績の左欄は測定可能な枚数,ステップ数,チェック項目等を工程毎に設定する。
目標，実績の右欄の当週欄はその週の完了度，累計欄はその工程全体を１００％とした完成率を記入する。

表３－４　問題処理表のフォーマット例

問　題　処　理　表

宛　　先		発行プロジェクト	
表　　題		整　理　番　号	

問い合せ事項・要望・改善		発行日	・　・　　発行者印
回答欄		回答日	・　・　　回答印
状況チェック欄回答後の実施		チェック日	・　・　　回答印

回答希望日		緊急度	1　2　3　4　5
メモ欄			

マネジャ

表３－５　個人別進捗管理表の例

個人別進捗管理表　　山　口　一　夫

作業＼チェック日	1／10	2／1	2／20	3／10	3／30	4／20	5／10	6／20
プログラムAの仕様決定	30%	50%	70%	100%				
〃　B　〃	10%	40%	60%	80%	90%	100%		
〃　C　〃	10%	20%	50%	70%	100%			

（注）□は完了していなければならないチェック日を示す。

表３－６　モジュール別進捗管理表の例

モジュール＼工程	システム名：ABC				プログラム名：DATA ACCSESS			
	プログラム設計		コーディング		単体テスト		結合テスト	
FILE SERCH	予定 7／10	実績 7／10	予定 8／10	実績 8／12	予定 8／30	実績	予定 10／20	実績
FILE SET	7／15	7／10	8／12		8／30			
FILE READ	7／20	7／19	8／20		9／5			

3.2 品質管理

ソフトウェア開発における品質管理は，品質そのものの定義が明確でなかったり，品質に大きな影響を与えるテストの技法やツールが不十分であることなどから，十分に行われているとは言い難い状況である。しかしながら，ソフトウェアの品質の確保，高品質化に対する要求は，次のような事情によりますます強くなってきている。

- システムの大規模化
- オンライン・システムの普及
- 意思決定のためのシステムや会計情報システム，災害防止システム等誤りが許されない分野でのコンピュータ化の増大
- 上級プログラムなどソフトウェア技術者の育成，確保が困難

本節では，このような状況下で，より実効的な品質管理を行うための留意点や方法について述べることにする。

3.2.1 ソフトウェアの品質

ソフトウェアの品質の定義は必ずしも明確になっていないが，Boehmはソフトウェアの品質を表わす特性要因として，図3－7のような階層化した定義を行っている。この図において，高位レベルのものは品質特性を示し，下位レベルのものはそれぞれの構成要素を示している。

次に，図中の主な品質特性の意味を示す。

- Usability（使用性）

  プログラムが信頼でき，効率が良くかつ操作性が良いという度合。

- Maintainability(保守性)

  プログラムが，新しい要求を満足するためや欠陥を修正するために容易に更新できる度合

- Portability（移植性）

  プログラムが，現在のもの以外の計算機に移行しても正しく，容易に

動作することができる度合。

- Reliability（信頼性）

  プログラムが，意図された機能を満足に実行することができる度合。

- Efficiency（効率）

  プログラムが，資源を浪費することなく目的を果す度合。

- Human－Engineering（操作性）

  ユーザの時間やエネルギーを浪費させない度合。

- Testability（テストの容易性）

  プログラムのテストや性能の評価が行いやすい度合。

- Understandability（理解の容易性）

  プログラムの分かりやすさの度合。

- Modifiability（変更の容易性）

  プログラムの変更しやすさの度合。

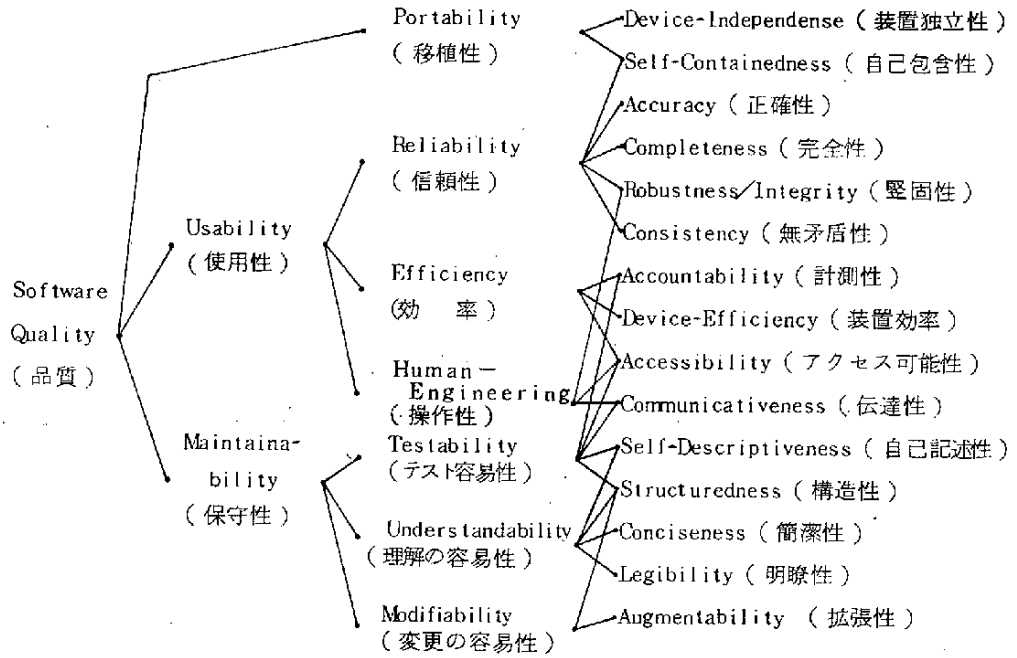


図３－７　ソフトウェアの品質特性

以上に示したような品質特性は，利用者の要求を満足する度合，あるいは，定められた品質基準を満足する度合でその良否が問われることになる。

また，一般的なユーザのニーズから考えると，図３－７に示した諸要素のうち，信頼性，効率，操作性は特に高いプライオリティを持っているといえる。

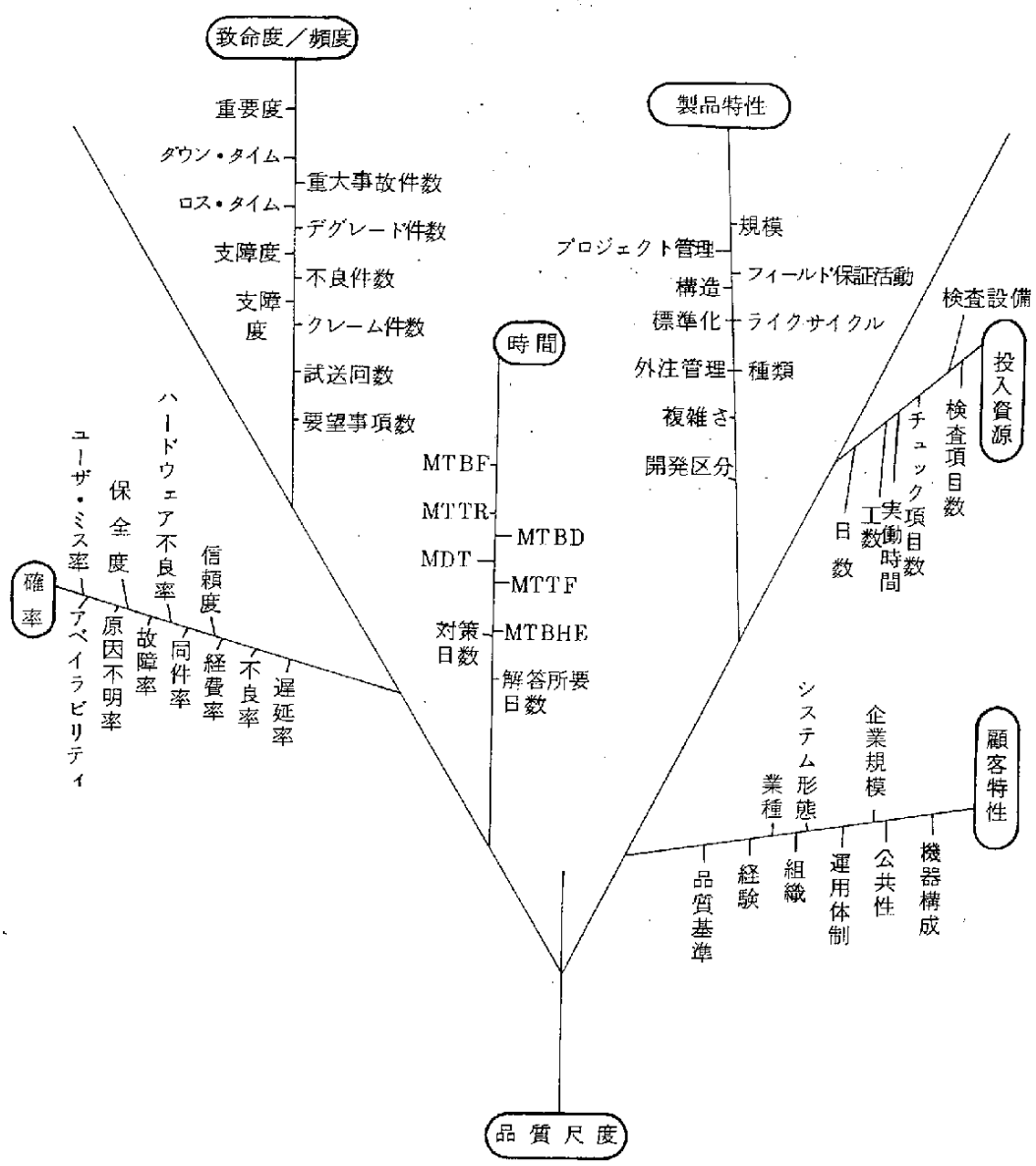
3.2.2　ソフトウェアの品質管理と信頼性の向上

ソフトウェアの品質を管理し，向上させるためにはどのように考えればよいのであろうか。以下，そのための基本的な考え方を示す。

(1)　品質評価測度の設定

ソフトウェアの品質を上げるためには，生産の各工程の中間生産物に対して評価測度を明確に設定しておくことが大切である。これによって初めて適切なレビュの実をあげることができる。現実に評価測度を考える際には，その製品が管理プログラム，言語処理プログラムなどの基本ソフトウェアなのか，あるいは個々のユーザの業務を処理するためのユーザーズ・プログラムなのかによって，きめ細かな特性の配慮を必要とする。また，それらに共通した諸問題，例えば，共通的なエラー・リカバリ（フール・プルーフ，フェイル・セーフ，フェイル・ソフトの全てを含む）などについても，矛盾のない対処，方策を吟味しておくことが大切である。

ソフトウェアにおける品質評価尺度因子に関する具体的な事例としては図３－８に示すものがあげられる。図左側の枝は，時間，頻度，確率といった区分にしたがっての品質尺度を示すものであり，右側の枝は，品質尺度を設定するに際して考慮すべき諸条件を示したものである。



「ソフトウェア工学におけるQAとQC」
（菅野文友）

図3－8　品質尺度トリー

(2) デザイン・レビュの重要性

ソフトウェア開発において最も大切な工程は仕様を明確にする設計段階である。この段階での生産物，つまり設計仕様書等をチーム全員，時にはチーム外の専門家や審査部門の技術者を交えて入念にレビュすることは製品であるソフトウェアの品質にとって極めて有効である。要は，デザイン・レビュによる「先憂後楽」こそ，コスト有効性の立場から信頼性向上を図る最良の方法であると言える。

デザイン・レビュ時の留意事項を表３－１に示し，その効果を表３－８に示す。さらに，ソフトウェアのデザイン・レビュにおける表現上，及び内容上の指摘事項を例示すると，それぞれ表３－９及び表３－１０に示すようなものがある。

表３－７　デザイン・レビュ時の留意事項

項番	項　目	理　由／内　容
1	動機づけ	デザイン・レビュを設計者が積極的に実施するためには，デザイン・レビュの効果を設計者が直接感ずることが貴重な要素となる。
2	計画的なデザイン・レビュの実施	審査者として，他システムのデザイン・レビュに参加する場合は，システム単位，プロジェクト単位の工数寄与率の低下をもたらすので，事前に工数計画が必要である。
3	手法の確立	デザイン・レビュの効果を上げるためには，十分に体系づけられた手法を取り入れて，ソストウェア生産に，より適切なデザイン・レビュを実施することが大切である。
4	規格化の推進	デザイン・レビュの早期実施を実現するための仕組を設定することが大事である。
5	技術者の育成	デザイン・レビュの実施を通して，ＯＪＴにより，技術力およびデザュに精通した技術者を育成するようすすめる。
6	人材の登録	専門技術者のリスト・アップが必要。これによって，構成メンバの選択が容易にできるようにする。
7	データバンクの確立	デザイン・レビュの効果を上げるための重要な要素となるフィールド・データを効果的に集約し，活用できる体制が必要である。
8	評価の確立	デザイン・レビュ結果の実用的な評価法を，確立する必要がある

表３－８　デザイン・レビュの効果

項番	区分	項目
1	信頼性	設計不良の早期撲滅による品質向上
2		設計者間の情報伝達の徹底（誤解にもとづく不良の撲滅）
3		他システムからの関連技術の反映，フィールド・データの反映
4		システムの問題点のは握と徹底
5	管理の強化	中間製品の品質のは握と改善
6		品質向上のための適切な指示
7		次工程への問題点のフィード・フォワード
8	削減・低減	ドキュメント試送回送の低減
9		検査工数の削減
１０		ドキュメント修正作業に必要とする設計工数の削減

表３－９　表現上の指摘事項の例

項番	区分	項目
1	基本不良	表現形式の規格不遵守
2		表現の不統一
3	表現不良	文字的誤り
4		文章曖昧
5	努力不足	誤字／脱字
6		誤記入
7		記述不足
8	技術不足	記載もれ
9		内容の誤り

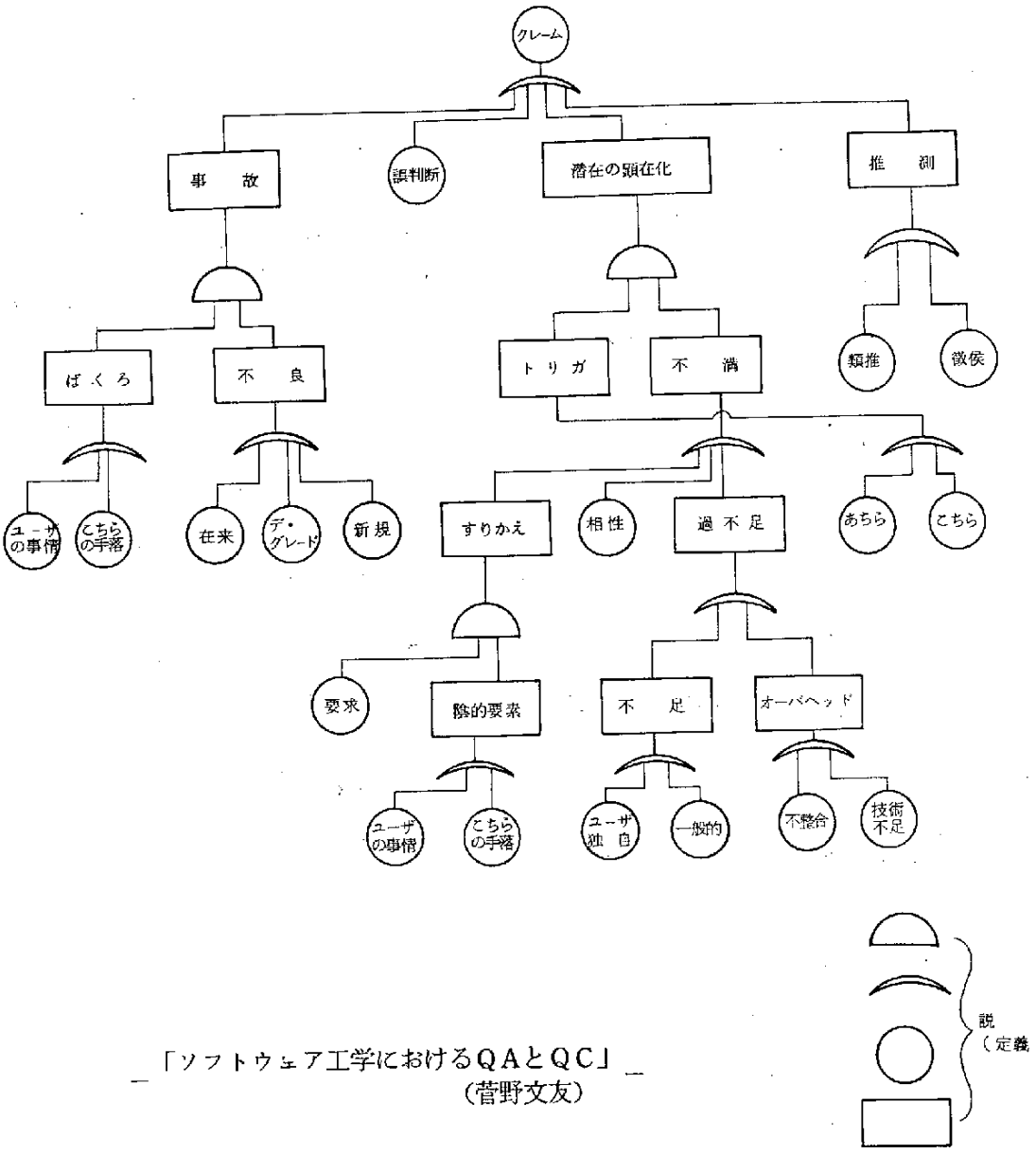
表３－１０　内容上の検討事項の例

項番	区　　分	項　　目
1	記　　述	曖昧，不明確
2		不適切（もっとよい表現がある）
3	仕　　様	機能欠落
4		システムと基本的に矛盾
5		仕様不適切
6		使用上の必要条件の検討不足
7	インタフェース	ハードウェアとのインタフェース不良
8		ほかのプログラムとインタフェース不良
9		同じプログ内でのインタフェース不良
10		ドキュメント間の矛盾
11	そ　の　他	全　般

(3)　運用後のフォロー

一般にユーザが自分の使いたいソフトウェアの機能や性能を，仕様として完全に明示することは難しい。従って，ある程度の要望は示せても，詳細な内容については開発者まかせになることが多い。しかし，実際に開発したソフトウェアを使ってみると，改めていろいろと潜在的な要望が顕在化してくるのが普通である。従って，製品のリリースに際しては，その後のユーザの声を適切に反映出来るよう，バージョン・アップ（機能の追加や性能向上の実現）とリビジョン・アップ（不具合の修正）の実施時期を計画しておかねばならない。ユーザの稼動実績のは握に際しては，受動的な態度では駄目で，能動的なアプローチを意識しなければならない。特に，ソフトウェアの場合には，ハードウェアのトラブルやユーザの運用形態などとの複雑な絡み合いがあり，表面的な事実の裏に隠された事実を見極めることはなかなか難しい。能動的なアプローチの

方法としては，信頼性工学の場で広く適用されている解析技法を大いに利用することが大切である。図３－９にクレームのＦＴＡを示す。



「ソフトウェア工学におけるＱＡとＱＣ」
（菅野文友）

図３－９　クレームのＦＴＡ

3.2.3　品質管理上の留意点

プログラムの品質に問題があった場合の原因として図３－１０に示すようなものがあげられている。この結果を踏まえて，品質管理上のポイントを示すと次のようなものがあげられる。

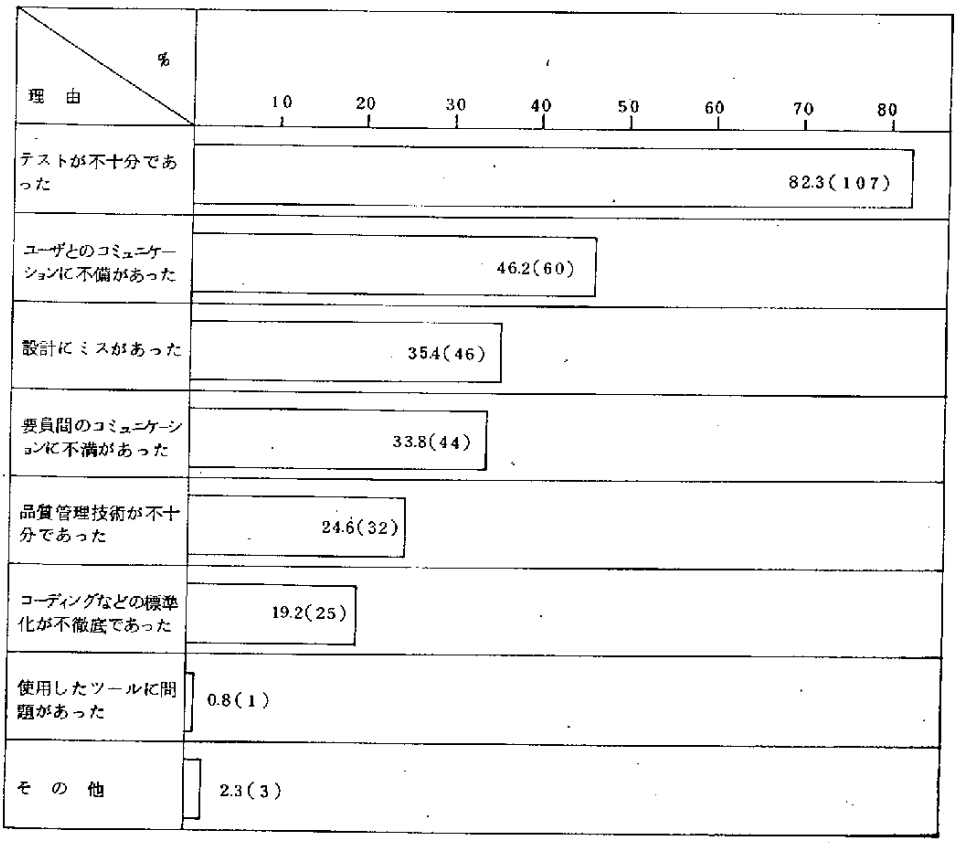


図３－１０　プログラムの品質が問題になった理由

① インタフェース部分は特に注意する。

物事は，インタフェース，結合部分，つなぎ目といったところに弱点が多い。ソフトウェアの場合も，マンとマシン，新しいプログラムと前に作成したプログラム，その他もろもろのインタフェースに注目することが，品質管理全般を通じての眼目である。

一般に，製品が複雑化してくるにつれて，保全性とともに拡張性が要求される。この拡張性を活用して，システムの増強，増大を実現していく場合，必然的にインタフェースも増加するが，それがややもするとシステムの弱点を招来することになる。このようなシステムの拡張に伴った新しいインタフェースの出現にも十分な注意が必要である。

② テスト時のエラー処理には細心の注意をはらう。

テスト時にエラーが生じた場合には，当然修正することになるが，その際，エラーの原因を的確には握しなければならない。どうしてもそのエラー箇所の修正だけに眼を奪われがちであり，ややもすると，その周囲のモジュール，あるいは，インタフェース部分などに影響を与えたことに気がつかない場合もある。従って，エラーの修正に際しては，単に表面的な当初のエラーだけに固執してはならない。例えば，ソフトウェアのエラーが，データの値による場合など，落ち着いてよく吟味しなければならなかなか分らない。このような場合には，プログラムの修正だけではなく，入力データの再確認を行わなければ，かえってエラーの再発や拡大を招くことになる。

③ 過去のエラー（バグ）記録を軽視してはならない。

どんな製品であっても，何から何まで全く新しい製品というものはあり得ない。部品なり，方式なり，構造なり，何らかの点で過去の技術を利用している。また，どのような製品であっても大なり小なりの欠陥を保有していたものであり，最初から無欠陥というものはほとんどない。

ソフトウェアの開発に際しても，何らかの意味で同類のプログラムに

ついて，過去のエラー（バク）の記録を丹念に調べ，「他山の石」として活用しなければならない。先人の経験や過去の技術の蓄積などを無駄にしてはならない。エラー（バグ）についての記録を整理しておくことは，極めて初歩的なことである。

④ テストにおける検査は限界点検を基本とする。

最終的な製品の検査にあたっては，検査項目として，限界ギリギリのところでの様相はもちろんのこと，限界を若干超過した場合の様子を吟味しておくことが大切である。限界までは線形に変化しても，限界を少し超えると急速に非線型の特性変化を示すこともよくある。ちょっとした確認の有無が，莫大な損失を左右する事態にもなりかねない。データ許容容量，数値の大きさ，精度等の限界を知ることは非常に重要なことである。

⑤ 要員間及びユーザとのコミュニケーションを充実する。

ソフトウェアの品質はユーザによって微妙に変化するのが普通である。例えば，精度よりも速度を要求するユーザもあれば，速度よりも精度を要求するユーザもある。このような要求の違いは当然プログラムの組み方に影響する。従って，ユーザとのコミュニケーションを充実して，ニーズを十分吸収する必要があることはいうまでもないが，要員に対して，それらのニーズを十分周知させなければならない。

⑥ 要員のモラールの向上を図る。

ソフトウェアの品質は，関連する要員のモラールにかなり影響される。従って，プロジェクト・マネージャを中心にして，チーム全体のモラールを向上させることは，製品の品質の向上に大きく寄与することになる。

⑦ 過去のプログラムを再利用する。

ソフトウェアはハードウェアと違い，サビ，摩滅などは無く，長期間使用されても，品質が落ちることはない。この性質を利用して，過去に作成したプログラムを出来るだけ再利用することこそ，生産性をあげる

と同時に，高い品質を約束する道である。このためには，再利用可能なプログラムの開発と，その適切な管理が重要である。具体的にはモジュール・データベースの構築などの方法が考えられる。

⑧ 各種標準化を徹底する。

ソフトウェアの開発は，２人以上の要員によって行われることが多い訳であるが，この際，品質を均質にするためには，全ての要員が同一の基準によって行動することが必要である。そのためには，ソフトウェア開発の全ての段階に亘っての作業を標準化しておかなければならない。標準化については，後でも多少ふれるが要員の資質の違いを吸収するためには標準化が必要不可欠なことである。

⑨ テスト・ツール，技法を活用する。

現在，数多くのテスト・ツールが開発されているし，ＵＮＩＸのような開発専用ＯＳともいえようなシステムも利用され初めている。これらのツール等を活用することは，ソフトウェアの品質にとって極めて有用なことである。

3.2.4 品質向上のための具体的方法

調査の結果によると，品質管理の具体的方法としては，ウォーク・スルーが最も多く（４６.２％），ついで，簡易言語・テスト支援ツールの利用（４１.０％）であった。（図３－１１参照）

ここでは，品質管理の具体的方法の中から，次の５つをとりあげ，その概要を説明する。

- ウォーク・スルー
- バグ管理図
- キャプチュア・リキャプチュア
- ゴンペルツ曲線によるバグ発生予測
- ＱＣサークル

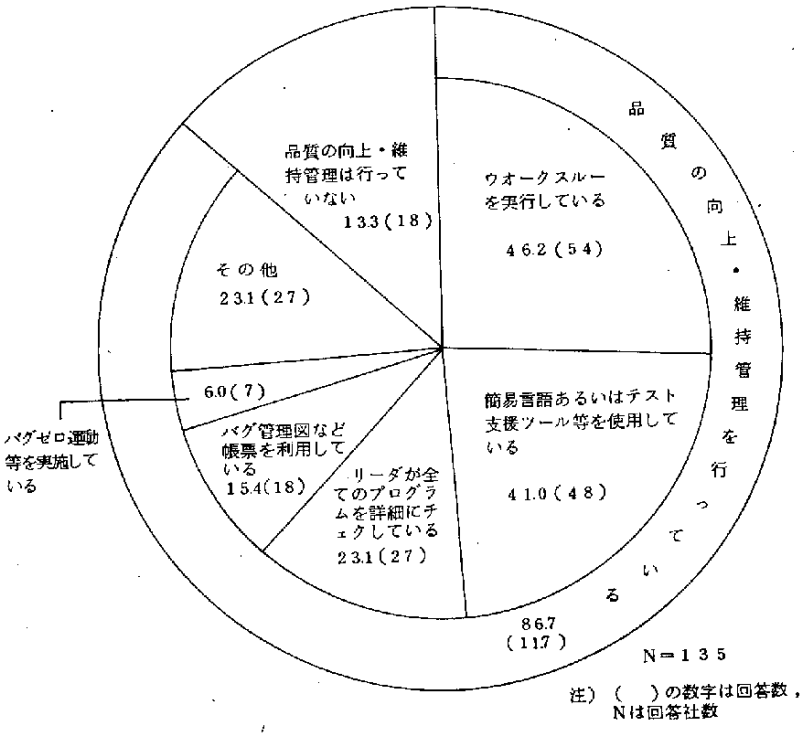


図３－１１　品質の向上・維持管理方法

(1)　ウォーク・スルー

これは，プロジェクトのメンバによって，開発者自身の作業（システム設計，プログラム設計，コーディングなど）を検討してもらう方法である。これらの検討は，エラーの早期発見に役立つばかりではなく，点検者にとっても，新しい方法や技術を習得することができる。正しい方法でウォーク・スルーを実施すれば，高い品質が得られるばかりでなく，要員のスキルの向上にも役立つ。

(2) バグ管理図

これは，デバグ工程において，チェック項目の確認予定／実績件数，およびバグ摘出件数と解決ステータスを管理する方法である。これによって，未解決バグがたまり出すという最も危険な徴候を早期には握できるばかりでなく，バグ摘出累計曲線の収束状況と，チェック項目の残数との相関を見つつ，チェック項目の追加，見直し，工数，計算機資源の追加投入等の対策を早期に打っていくことで，遅延を生じ初めたシステムテスト工程の完了日を確保することも可能となる。バグ管理図の例を図３－１２に示す。

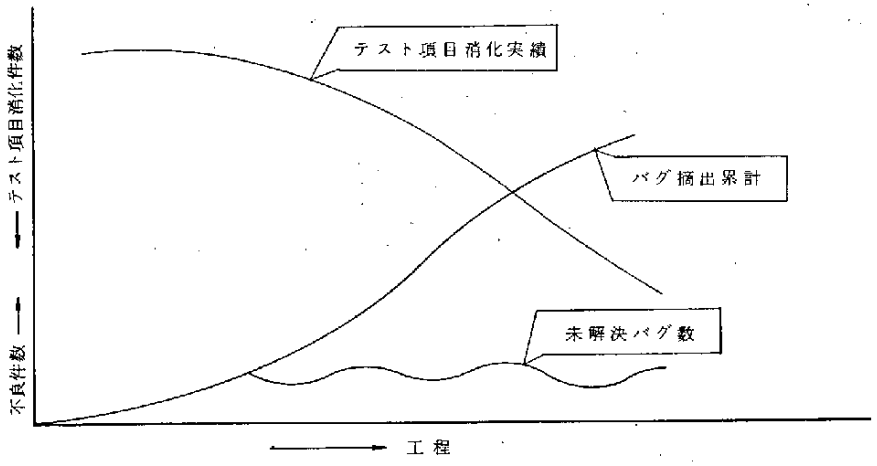


図３－１２　バグ管理図の例

(3) キャプチュア・リキャプチュア

キャプチュア・リキャプチュア法とは，もともと野生動物の頭数推定のためのものであり，捕えた動物に記号（標識）をつけてもとの場所に放し，その再補率によって総固体数を推定する方法である。この方法を用いて，ソフトウェアの潜在バグ数を推定することができる。具体的

には，次の手順による。

① 故意に作ったバグをプログラムに埋め込む。

② 埋め込まれたバグの存在を知らないプログラマあるいは検査グループがバグを摘出する。

③ その結果から総バグ数(N)を推定する。

$$N = Mn / m$$

M：埋め込みバグ件数

n：後で摘出された総バグ件数

m：後で摘出された埋め込みバグ件数

故に，その時点の埋め込みバグを除いた潜在バグ数(ℓ)は次のようになる。

$$\ell = N - M - (n - m)$$

(4) ゴンペルツ曲線によるバグ発生予測

一般にバグの発生は，成長曲線（例えばゴンペルツ曲線）をたどって進行することが知られている。よって，この曲線を用いて将来のバグ発生件数をある程度予測することができる。ゴンペルツ曲線を利用すれば次の式で予測できる。

$$N = K \cdot a^{bt}$$

N：tにおけるバグ数の予測

K：バグ総数（予測）

$0 < a,\ b < 1$：曲線の形状を決定する値

t：時間

ゴンペルツ曲線の例を図３－１３に示す。

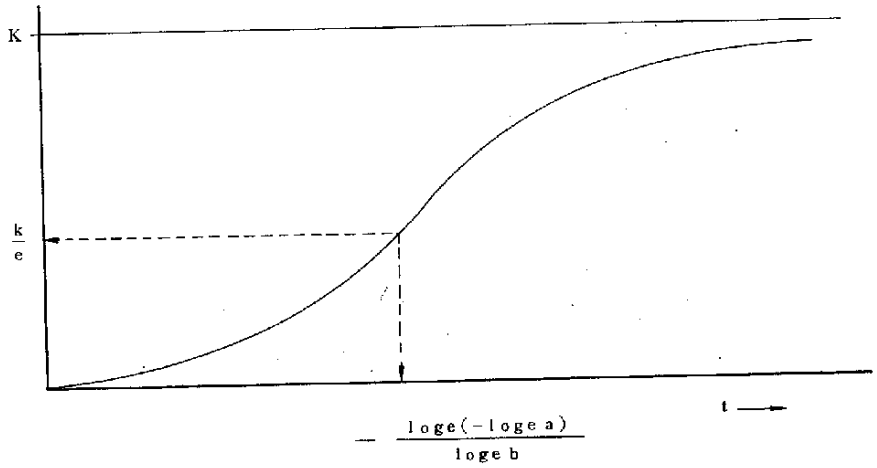


図３－１３　ゴンペルツ曲線

(5)　ＱＣサークル

ＱＣ（Quality Control）サークル活動は，一般の製造業においてはよくみられ，成果をあげているものである。この方法はソフトウェアの開発においても適用可能であり，次のように定義される。

「ＱＣサークル活動とは，データの記録・分類・蓄積にもとづき，作業の中で発生した“誤りの真の原因”を分析・追求し（Check），“予防・発見・除去の方策”を創意工夫で見つけ出し，これを活用し(Action)更に，蓄積されたノウハウにより“目標，標準，手段”を整え，計画を作り（Plan）“誤まりを作り込まない正しい作業”を実行（Do）ることである。」

3.2.5　コンフィグレーション管理

ここでいうコンフィグレーションとは，ドキュメントに記述されたシス

テムやプログラムに関する機能や特性である。したがって，コンフィグレーション管理とは，ソフトウェアの開発に係わるあらゆる技術情報の管理を意味する。コンフィグレーション管理を実施する際，特に重要な概念は，標準化と変更管理である。これらの概念は，ソフトウェアの品質に大きな影響を与えるものであるので整理しておく。

(1) 標 準 化

適切な標準あるいは基準が設定され，円滑に運用されれば，プロジェクト管理に次のような効果が期待できる。

① システムの開発費用を低減する。

② システムの開発期間の短縮を可能とする。

③ プロジェクトの参画者全員に対する情報伝達が統一的に行える。

④ 各種標準作業手続き，標準ドキュメントの作成により，正確な情報交換，仕事の均質化，容易な管理が行え，かつ，これにより開発技術の蓄積が行いやすくなる。

⑤ 要員の配置転換に際し，新しい参加者がスムーズに移行しやすい。

以上のように，ソフトウェア開発は，最も制御しにくいと言われる人間を扱うので，適切な標準の設定と運用は，プロジェクトの成否を決める大きな要因ということができる。

以下，具体的に標準化の対象を示す。

① 運用・管理標準

ⓐ プロジェクトの計画：工程や中間生産物及び各作業についての定義。

ⓑ プロジェクトの組織：チーム編成の方法や役割分担。

ⓒ プロジェクトの運用：プロジェクトメンバ間の情報伝達方法等。

ⓓ プロジェクトの管理：工程，予算，品質等の管理に際してのデータの収集法。

ⓔ 標準化の維持・管理：標準化の維持，運用方法の規定。

② 作業標準

ⓐ　設計方式：設計手順の基準。

ⓑ　設計技術：設計上の技術を整理統合したもので，要員全員が共通に使える。

ⓒ　コーディング：コーディング上の技法及び形式上の基準。

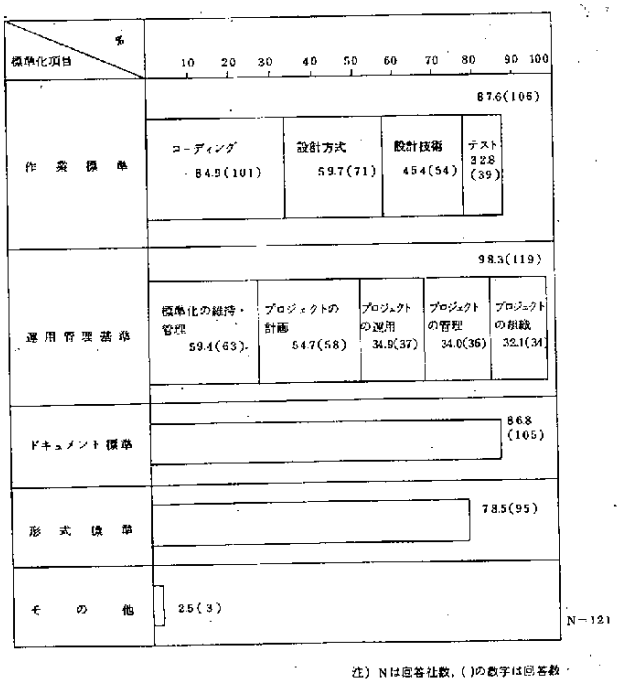
ⓓ　テスト：テストの内容や方法に関する基準。

③　その他

ⓐ　ドキュメント標準：ソフトウェアの設計上必要な文書やマニュアル等の説明書作成上の標準で，記述内容，記述方法，記述様式を規定したもの。

ⓑ　形式標準：ソフトウェア内で，一意に定めるべきソフトウェア・モジュールに対する命令法あるいは識別子，コードの付与方法及び入出力メッセージの形式等を規定したもの。

参考までに，標準化の実施状況を図３－１４に示す。これによると，大部分の企業において何らかの標準化が行われていることが分る。



注）Nは回答社数，（）の数字は回答数

図３－１４　標準化の状況

(2) 変更管理

変更管理の基本は，各種の変更の取扱い方とそれに対応したドキュメントの管理である。以下，それぞれについて，その基本的な考え方を示す。

(a) 仕様の変更

仕様の変更は，工期，品質に与える影響が極めて大きいために慎重に対処する必要がある。仕様の変更には2通りの場合が考えられ，1つは契約条件やユーザからの要求による変更で，安全性，実施の困難度，経済性などの理由によって，主に変更されるもので，もう1つは内部的な事由，つまり開発上の不手際等などによって止むを得ず変更するものである。前者の場合，それが外注（あるいは受注）したものであれば，当然契約自体が変更されることになる。そうでなくても工期の延長や費用の増加は避けられないのであるから，特別な場合を除いては開発途中における変更は極力止めた方がよい。しかし，実際問題として，小さな変更はよく起きることであり，この全てを受け入れないわけにもいかない。そこで，止むを得ず変更する場合は，2工程以前にさかのぼらなくてもよいものに限定すべきである。2工程以前にさかのぼるような変更は，工期，品質に与える影響が極めて大きいからである。

後者は，開発技術あるいはプロジェクト管理技術に起因する場合が多いので，特に次のようなことに十分留意しなければならない。

- 要員間のコミュニケーションを充実させ，情報の伝達を確実に行う。
- レビュは完全に行い，あいまいなままで次の工程に移るようなことは絶体にやめなければならない。
- 標準化の徹底を図り，少なくともあとで体裁だけを整えるような作業は極力無くすべきである。

(b) ドキュメント管理

ソフトウェア開発の場合，各種の仕様書あるいは設計書は，製品の全てを決定する極めて重要なものである。従って，プログラムとドキュメントが一致していないといったことは無いようにしなければならない。ドキュメント管理上の主なポイントは次のようなものである。

- ＩＢＭのＩＰＴ（Improved Programming Technologyies）におけるライブラリアンのように，あらゆるドキュメントを管理する専任の担当者がおければ最も良い。
- ドキュメントに対するコードの付与基準等を明確にして，識別を容易にしておく。
- 全てのドキュメントには改訂履歴表を入れておき，どんな小さな変更であっても必ず，改訂履歴表に記入し，承認を受けるようにする。

3.3 費用管理

プロジェクト管理における費用管理とは，適切な見積りと，それに対する実際支出の測定を行って，差異を明確にし，差異が生じている場合に適切な管理処置を施すことである。見積りについては第２章で詳述したので，ここでは支出の管理を中心に費用管理のポイントを示すことにする。

3.3.1 費用管理の要件

(1) 予算に対する実際支出の測定

直接労務費，計算機使用料，その他の費用を進行中の各作業ごとに示すために，週単位で，（少なくとも月単位で）報告が必要である。このデータは，費用管理のための図や帳票に記入され，予算と実際支出の差異を明確にするために使用される。前節で述べたトレンド・チャート等を利用すれば，スケジュールに対する実績進捗の情報も同時に示すこと

ができる。予算と実績の対比は，プロジェクト全体レベル，サブ・システム等の中間レベル，及び個々の作業レベルで必要で，これには，直接労務費，計算機使用料，カード等の消耗費，印刷費などが示される。

(2) 内約事項の記録と管理

内約事項として未だ記帳されていないものでも顕著なものは，費用管理のためには，予実対比費用報告のなかに入れておかねばならない。そうした内約事項の最終的な数値が正式記帳時に変わることがあるにしても，出来るだけ早い機会に明らかにし，報告する仕組みを作らなければならない。そうした内約事項を記録し管理しないと，特にハードウェアやソフトウェア・パッケージを購入する場合や，プログラムの一部または全部を外注するような場合にしばしば費用超過の原因となる。

(3) 賦課記帳の遅れ

賦課の遅れは，請求書についての疑議，管理上の誤まりの訂正，請求書の提出遅れ，その他これに類似した行為から起る。内約事項についての報告と管理をしっかり行えば，賦課遅れの多くはなくなるであろうが，それでも避けることのできない場合が残る。もし，相当程度予算を下廻る場合でも，その資金を再配分してしまう前に注意深く分析するようにし，あとになって忘れていた賦課事項が出て来て驚くようなことがないようにしなければならない。

(4) 支出の正当性の確認

マネージャは個々の作業にかけられる労務と予算の支出が正当であるかどうか，また，それが本当にその作業に貢献したものであるかどうかを絶えず確かめていかなければならない。プロジェクトの中の予算超過した仕事の費用を，まだ予算をそれほど使っていない他の作業に賦課することもよく見られるが，このような行為は，真の問題をおおいかくし，費用記録の分析結果をゆがめ，データを将来の見積りの資料として使えなくしてしまうことになる。

表３－１１　作業時間記録表の例

作業時間記録表

昭和　　年　　月　　日提出

所属＿＿＿＿＿＿　氏名＿＿＿＿＿＿＿＿

作業月日	作業時刻		プロジェクト・コード	直接作業時間	間接作業時間	備考
月　日						

また，実際に仕事を遂行している人達は，プロジェクトあるいは作業単位毎に，時間を正しく記録していくことの重要性を認識しないまま，作業時間記録表（図３－１１）などに単純に記入しているだけということがある。マネージャはこれらの行為の重要性を十分に説明しておかなければならない。さらに，内部監査にあたる担当者は，既定の方針や手続が実際に守られているかどうかを定期的に監査しなければならない。

3.3.2　費用管理の方法

調査の結果によると，プロジェクトの費用管理の方法としては，工程別予算実績対比表等の帳票を利用している例が多いようである。（図３－１５参照）

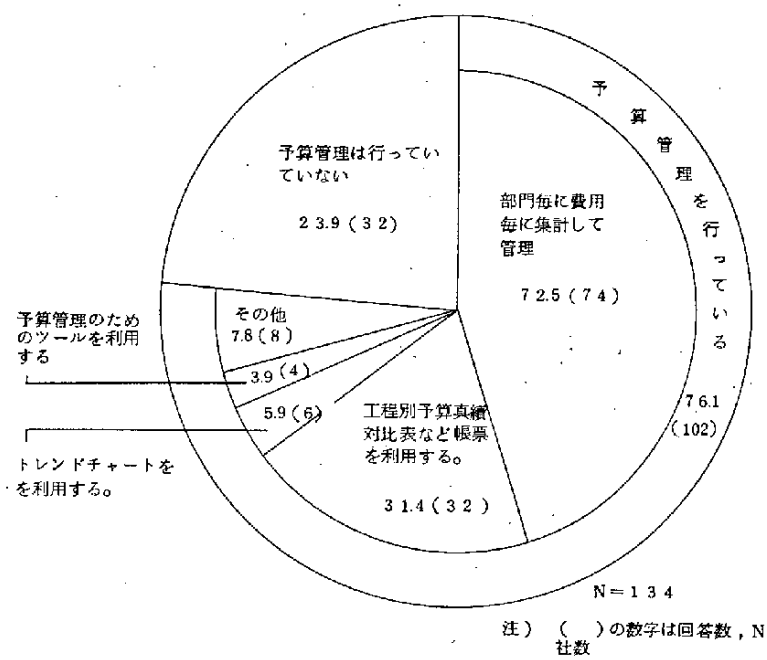


図３－１５　開発費用の管理方法

しかし，プロジェクト管理としての費用管理は，工程管理と同時に使用できるネットワーク図，つまり，ＰＥＲＴ／Cost を利用するのが最も良いと思われる。そこでここでは，ＰＥＲＴ／Cost を中心に費用管理の具体的な方法を示すことにする。

(1) ＰＥＲＴ／Cost

PERT／Time はネットワークをベースにした工程管理の手法であるが，ＰＥＲＴ／Cost は日程とコストの両者を総合した管理手法である。これは，プロジェクト・ネットワークの各アクティビティを遂行するために必要な日程とコストの見積りを行い，日程とコスト管理を総合して実施する方法である。図３－１６に示すような，PERT／Cost

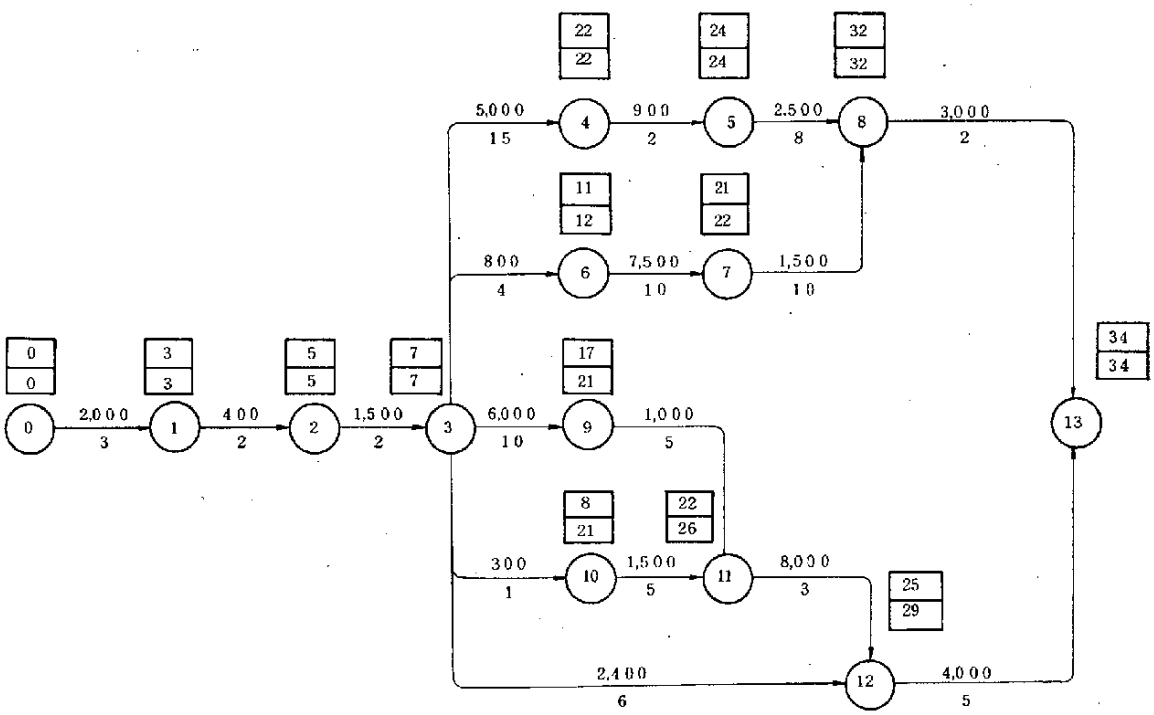


図３－１６　PERT／Cost ネットワーク

ネットワークを例に見てみよう。ネットワーク図からプロジェクト期間は３４日であり，コスト累計は48,300千円となる。

プロジェクト開始時点から終了時点に至る日程計画は，総工期３４日は変わらないにせよ，最早日程と最遅日程との間に中間日程計画はいろいろ考えられる。

一方，コスト累計の48,300千円は，最早日程計画でも最遅日程計画でも結果は同じである。ただし，コスト累計カーブを日程に関係づけて描けば，図３－１７のように，最早日程コスト累計カーブと最遅日程コ

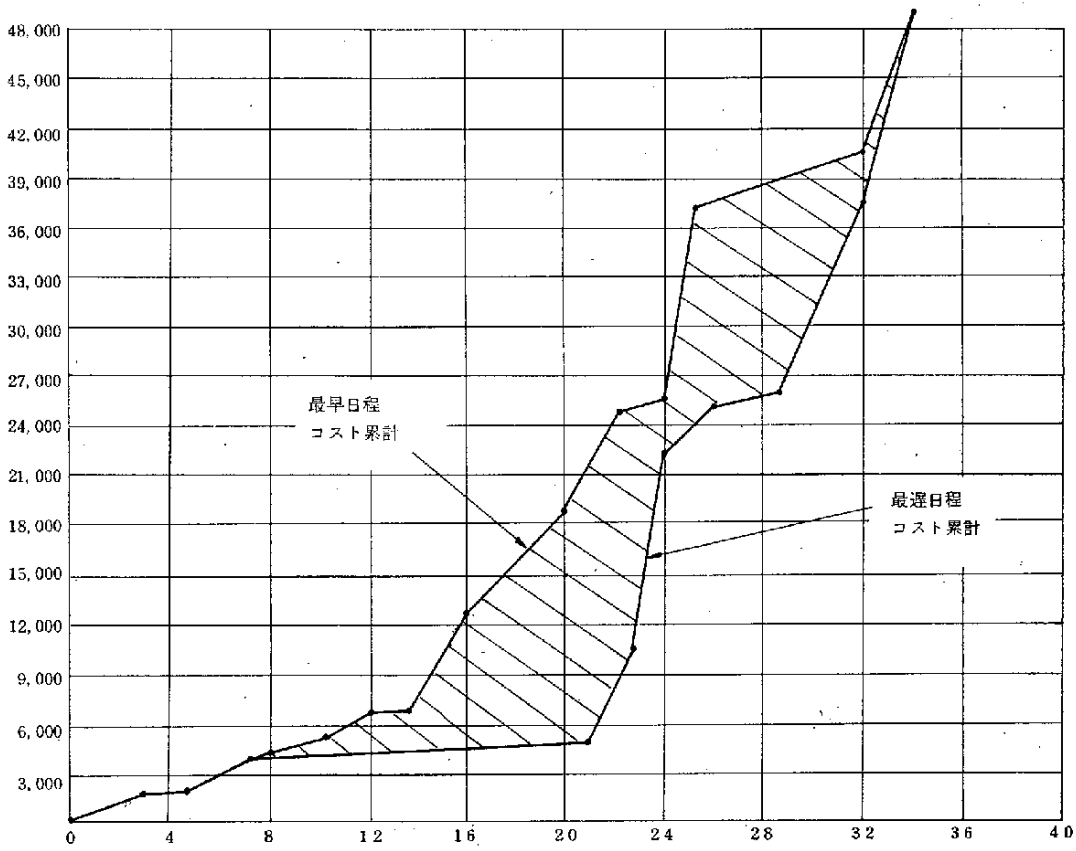


図３－１７　最早，最遅日程とコスト累計カーブ

スト累計カーブの２種類が描かれる。

ＰＥＲＴ／Ｃｏｓｔでは，日程をＰＥＲＴ／Ｔｉｍｅで管理し，一方のコストは，計画コスト累計カーブ，実績コスト累計カーブ及び完了作業に対する当初見積額などの比較に加え，フォローアップ時点におけるコストの将来予測などによって行なわれる。

次に，ＰＥＲＴ／Ｃｏｓｔにおけるコスト予測のやり方を示す。

図３－１８において，〔Ⅰ〕の部分はコスト累計カーブ，〔Ⅱ〕の部分はタイムスケール・ネットワークである。

このプロジェクトの日程は９ケ月，コストは６５,０００万円である。４月末の時点において，日程とコストの実績を調査したところが，次のことが分った。完了したアクティビティは，①→②，②→③，②→④である。４月末までに完了する計画であったが，まだ終了していないアクティビティは，③→⑤，③→⑥，②→⑤，④→⑦である。現在までに消費したコスト実績(A)は３５,０００万円である。現在までに，終了すべき仕事に計画したコストは２５,０００万円である。これらの事実により，一見，現在までの予算超過額は，

３５,０００－２５,０００＝１０,０００（万円）

であるかのようにみえるが，実際はそうではない。３５,０００万円の実績は，①→②，②→③，②→④のアクティビティを遂行した結果発生した費用である。一方，もともと，①→②，②→③，②→④の３アクティビティについて計画したコストＰは，１５,０００万円である。したがって，終了したアクティビティに関する限りその超過額は，

３５,０００－１５,０００＝２０,０００（万円）　となる。

ここまでのデータから，プロジェクト終了時点におけるコスト予測を行うことができる。まず，コスト・インデックスを計算する。

$$CI\ (\text{Cost Index}) = \frac{P}{A} = \frac{15,000}{35,000} = 0.42$$

このような傾向が今後も続くと想定すると，プロジェクトのトータル

・コスト65.000万円は，次のようになると予想できる。

- $Q' = \frac{Q}{CI} = \frac{65,000}{0.42} = 150,000$（万円）
- 超過予想額 $Q' - Q = 85.000$（万円）
- 日程の遅延は約3ヶ月
- 現時点までの業務遂行率 $\frac{P}{Q} = \frac{15,000}{65,000} = 0.23$

  23％
- 終了業務の超過額 35,000 − 15,000 = 20,000（万円）

この例でも分るように，コスト・インデックスはコスト面での警報を出し，早期の対策を図るためのものである。重要なことは，現時点から納期（終了予定時期）までの間に，予想されるコスト超過をおさえ，当初の予算を維持するために，どんなコスト・ダウン対策を実行するか，そして日程の遅れ傾向をいかに挽回し，期限内にプロジェクトを終了することができるかにある。図3－19は，コスト・インデックスの傾向を追跡し，プロジェクトの日程とコストを総合して管理するための資料である。

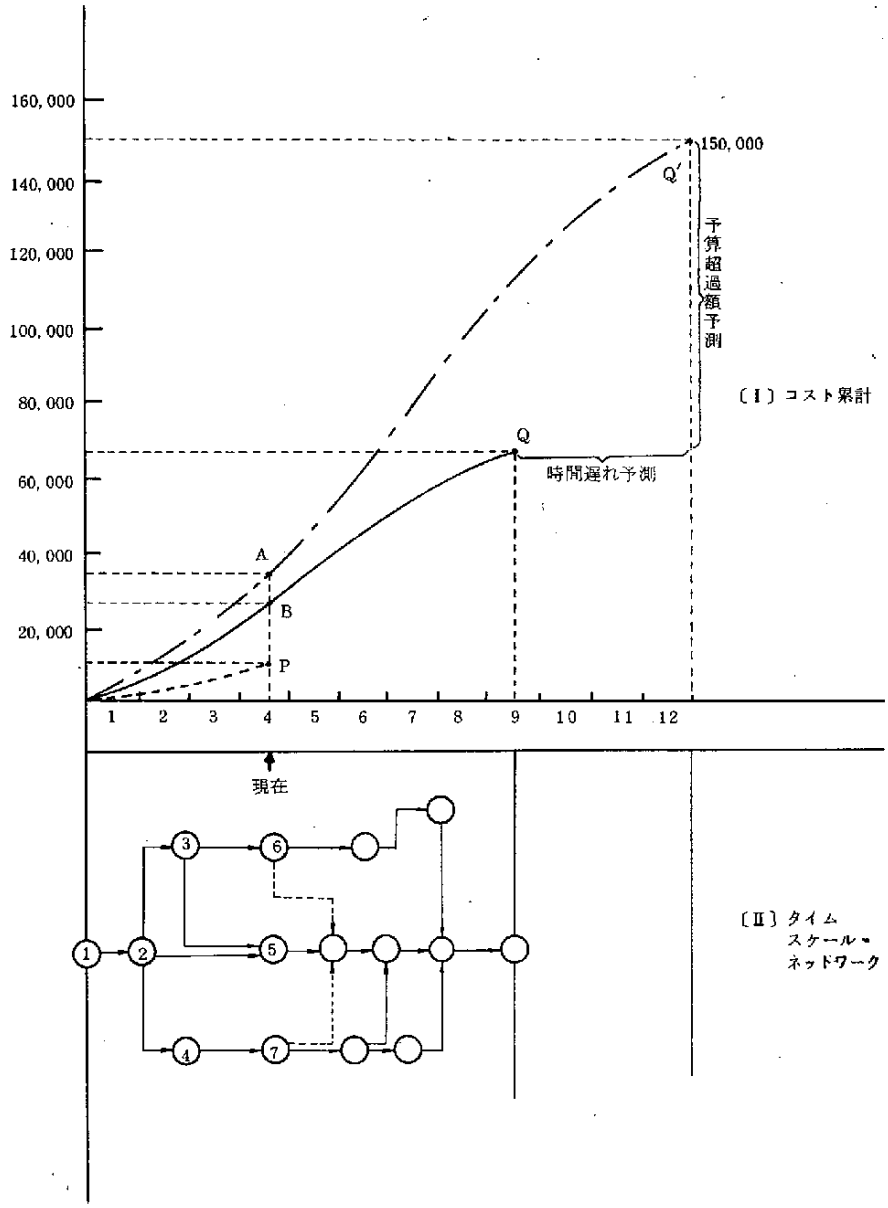


図３－１８　PERT／COST概念図

フォローアップ期日 / 管理項目	月　日	月　日	月　日	月　日	月　日
コスト・インデックス	0.5	0.6	0.8	0.7	0.9
日程進捗	−4	−3	−2	−3	−1
総予算（万円）	25,000	25,000	25,000	25,000	25,000
予想額（万円）	30,000	32,000	30,000	30,000	29,000
超過予想額（万円）	5,000	7,000	5,000	5,000	4,000

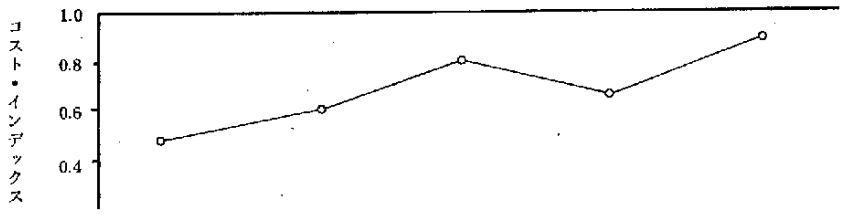


図3－19　コスト・インデックスを利用した日程とコストのフォローアップ

以上述べたように，ＰＥＲＴ／Costは，日程とコストを関連させ，管理していく手法であるが，John S.Baumgartner は両者の関係を指数化する方法を提案している。

この指数は，ステータス・インデックス（ＳＩ）と呼ばれ，次の式により計算される。

$$SI = \frac{\text{開始時点からの経過時間}}{\text{開始時点からの経過時間} - \text{PERTスラック}} \times \frac{\text{当初予算}}{\text{実績費用}}$$

また，ＳＩは，次のような概念としてとらえることもできる。

$$SI = \frac{\text{Output}}{\text{Input}} = \frac{\text{進度}}{\text{投入資源}} = \frac{\dfrac{\text{実績進度}}{\text{計画進度}}}{\dfrac{\text{実績費用}}{\text{当初予算}}}$$

このＳＩを定期的にフォローし，ＳＩを1.0に近づけるように管理す

ることによって，日程遅延防止と予算超過を防ぐことができる。

(2) トレンド・チャート

トレンド・チャートは，グラフにより日程と費用両方の管理を行う方法であるが，工程管理の中で述べてあるので詳細については省略する。

(3) 費用管理のための帳票

費用管理にも各種の帳票が用いられるが，ここでは，次の3種の例を示す。

① 費目別管理表（表3－12）

② 機能あるいはプロジェクト分割構成単位の管理表（表3－13）

③ 個人別工数管理表（表3－14）

表3－12 費目別管理表

プロジェクト名（　　）			（　）月（　）日～（　）月（　）日			
費目 ＼	当該期間		現在までの累積		完了時	
人件費	予定	実績	予定	実績	予定	予想
計算機費						
出張費						
消耗品費						
TOTAL						

表３－１３　機能あるいはプロジェクト分割構成単位の管理表

プロジェクト名（　　）			工　程　名（　　）			
サブシステム名	当該期間		累　　計		完　了　時	
A	予定	実績	予定	実績	予定	予想
B						
C						
TOTAL						

表３－１４　個人別工数管理表

プロジェクト名（　　）			担当者名（　　）				報告年月日（　　）			
作業＼月	５月		６月		７月			現在までの累計		完了時
A	予 20	実 21	予 30	実 35	予	実		予 120	実 119	予想 300
B	30	29	30	20				130	200	250
C	40	38	40	45				150	130	250
合　計										

3.4　外注管理

本節では，ソフトウェアの一部あるいは全部を外部のソフトウェア会社等へ外注する場合の考え方や，外注した際の管理上のポイントを示すことにする。

3.4.1　外注する場合の要件

(1)　外注の必要性

ソフトウェアの外注には，種々の形態が考えられるが，どのような外注も，その必要性，つまり目的は次の２つに集約できる。

- 専門技術の利用
- 負荷のバランス

①　専門技術の利用

一般の企業において，実用的なプログラムを作ることはそれほど易しいことではない。しかも，何人もの人間が何ケ月も費やして開発を行わなければならないプログラムのような場合には，その失敗が大幅な損失をもたらすことになり，そのリスクは大きい。そのような場合には，ソフトウェア会社等の外部の専門機関を利用する方が，はるかに合理的である。一般にソフトウェア会社等は，一応基礎技術が確立されており，長い間の経験の蓄積や専門的な創造力があり，品質，費用の両面からみても，自社で開発するより得策と思われるからである。

また，データベースやオンライン・システム，あるいはCAD/CAMなどのシステムを開発する場合などは，技術，経験の差によって効率や信頼性には雲泥の差が生じる。従って，技術的にみて専門的な分野のソフトウェアを開発するような場合には，その分野を専門としているソフトウェア会社等に外注した方が，高い品質のシステムを構築することができる。

②　負荷のバランス

どんなに計画的にソフトウェアの開発を行っている企業においても，不測の事故や設計変更，期限の繰り上げ，トップからの緊急な要求などで，ソフトウェア開発部門の負荷は変化するものである。また，ソフトウェア開発の場合，多人数を投入した方が効率的に進められる工程もある。このように，必要とされる要員の数が時期によって変化す

ることも多く，かつ社内での調達には限界がある。この能力的なアンバランスを解決するには，外注を活用するのが最も良い。つまり，外注は自社の能力を補充する手段と考えられるのである。

(2) 要員派遣と業務委託

外注には，現在要員派遣と業務委託の2つの形態がある。前者は自社に外部の社員を派遣させる方式であって，派遣された社員は，自社の社員と同様に管理され，いわゆる外注管理の対象とはならない。後者は，外注管理の対象となるものであり，まとまった業務を一括して発注する方式で，1～2本のプログラムからプロジェクト全体に至るまでその内容は様々である。ただ現在では，特別な場合を除いて，後者の方が一般的になっており，今まで派遣形態が多かった計算機メーカの外注方式も，ほとんど業務委託の形式がとられるようになってきている。

3.4.2 外注先の選定と発注

ここでは，外注先を選定する場合，及び発注する場合の考え方を示す。

(1) 外注先の適切な選択

外注先の選定は，外注の目的や方針を具体的に実現するための最も基礎的な第一歩である。外注先の優劣が，自社の生産活動に大きな影響を与えるのであるから，その選定にあたっては，外注利用の目的を具体的に吟味し，その目的に添った選定の方針が必要である。

一般に，外注先を選択する場合の原則となる方針には次のようなものがある。

① 要求する品質を十分に満足する開発技術と開発要員が備わっていること。

② 要求する品質を保証できる管理活動がなされていること。できれば，各種の標準化がなされていることが望ましい。

③　必要とする専門技術について，十分な経験を有していること。

④　コミュニケーションを容易にするためにも，自社とあまり離れていないこと。

⑤　財務体質が良いこと。

⑥　労使関係が円滑であること。

⑦　技術上，取引上の機密が保持できること。

(2)　要求内容の明確化

外注先が決まり，業務を発注する場合，最も重要なことは，要求内容を明確にするということである。これまでの場合，要求内容が不明確なまま発注し，相手も甘んじてそれを受け入れている例も多かった。しかし，これでは正確に見積ることは難しく，結果的に工期の延長，費用の増大を招いてしまう。従って，もし，要求内容が明確になっていないような場合には，それがはっきりするまで発注は控えるか，要求を明確にする作業のみを独立させて発注するようにした方がよい。

また，ソフトウェアの仕様は，当初の設計が充分であっても，開発途中で，要求が変わることもある。この場合，対応策は発注先と協議して決められることになるが，大幅な変更を強制させたりすると，結果的に工期や品質に影響を与えることになる。このような場合は，再契約をするか，無理をせず，当初の仕様をバージョン1とし，後から発生した要求を加味したものをバージョン2に位置づけて次期プロジェクトとすべきであろう。

(3)　適性な価格の維持

ソフトウェアの外注にあたっては，品質，価格，納期の3要素をそれぞれ十分満たすように契約することになるが，なかでも，価格については，単に金額の決定だけではなく，支払条件など，直接に経理面とつながる問題であるだけに，自社のみならず，外注先にとっても非常に関心が高い。価格の契約は，自社及び外注先の双方にとって公正妥当なもの

でなければならない。しかし，実際には価格決定のための要因が非常に複雑で，微妙なものが多いだけに，各社とも苦労しているのが実情である。

参考として，表３－１５に，（社）ソフトウェア産業振興協会の価格予想値を示す。

(4) 作業実施計画書の確認

作業を開始する前に，外注先の作業実施計画書を確認しておく必要がある。これは，納期や納品物件等の自社の希望を外注先に確認させると同時に，要員やスケジュールなど外注先の作業計画を確認するためのものである。作業実施計画書には，次のような項目が記述される。

① 開発するプログラムの概要
② 開発担当要員（学歴，業務歴，業務上の役割）
③ 開発計算機（ＯＳ，言語，自社の計算機と異なる場合には，移植の方法）
④ 作業スケジュール（マイルストーンの明確になったもの）
⑤ 月別，作業内容別，担当者別工数配分表
⑥ 納期，納品物件
⑦ 作業場所（住所，電話番号）
⑧ その他

表３－１５　ソフトウェア産業振興協会の価格予想値

工程＼単価	A	B
1. 調査分析	1,320千円以上	1,584千円以上
2. システム設計		
2－1 基本設計	1,320 〃	1,584 〃
2－2 詳細設計	1,056 〃	1,267 〃
3. プログラミング		
3－1 プログラム設計	880 〃	1,056 〃
3－2 プログラム作成	704 〃	845 〃
4. 総合テスト	1,056 〃	1,267 〃
5. 検収		
5－1 マニュアル作成	704 〃	845 〃
5－2 検査立会	880 〃	1,056 〃
6. 保守	（別途）	（別途）

注　①　いずれも１人月当りの作業料金とし、１カ月を２０日間稼動としている。

②　基準となる工程を「プログラム設計」とし、ここを基準料金としたが、これは情報処理技術者試験第１種合格相当の技術者１人月の標準的な作業量および内容を想定したものである。

③　単価Bは、特別な専門知識を有する学者や専門家を必要とする場合、特定ノウハウ等を使用する場合、特定技術者の指名ある場合、等に適用するものである。

④　本表に示した単価は、昭和５７年４月１日から適用のものであり、単価は原則として毎年改訂される。

3.4.3　外注管理上の留意点

調査の結果によると，外注プログラムの管理方法，及び，外注プログラムの納期遅れの理由は，それぞれ図３－２０，図３－２１に示すようなことがあげられている。ここでは，これらの結果を参考にして，納期すなわち工程と品質を管理する場合のポイントについて考える。

当然のことながら，外注した場合，そのソフトウェアの開発は外注先で行われる。自社が介入できるのはある点的な時間にすぎない。従って，上記２つの管理は，この点的な時間の中で同時に行わなければならない。この

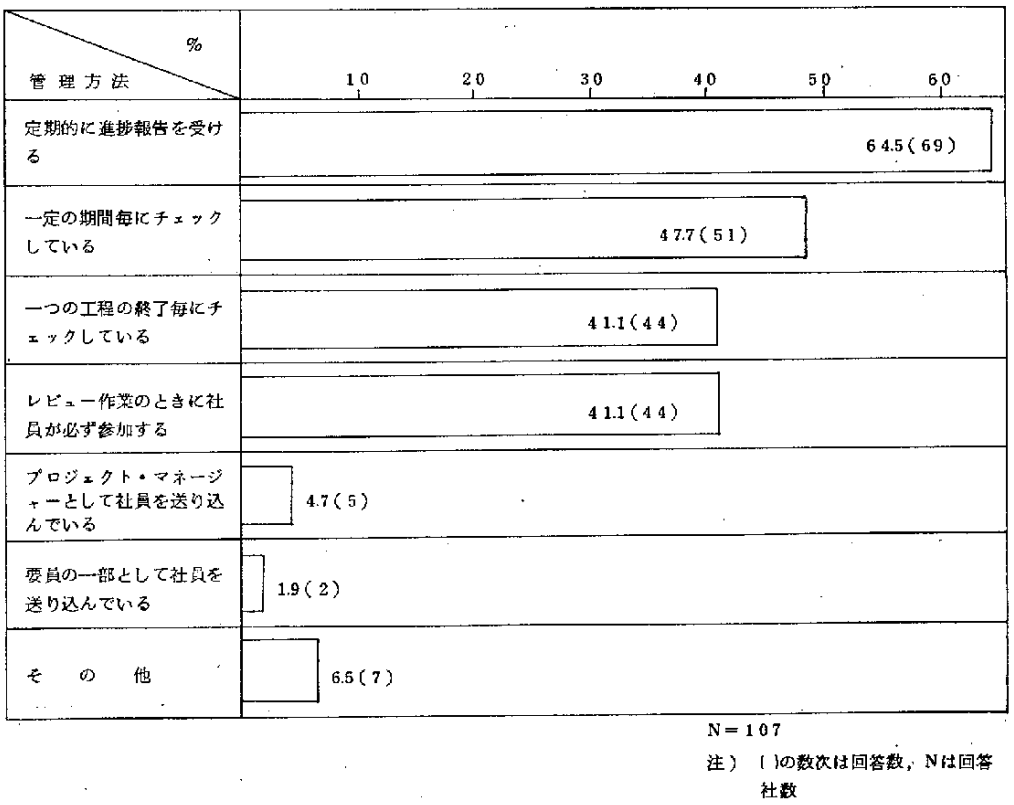


図３－２０　外注プログラムの管理方法

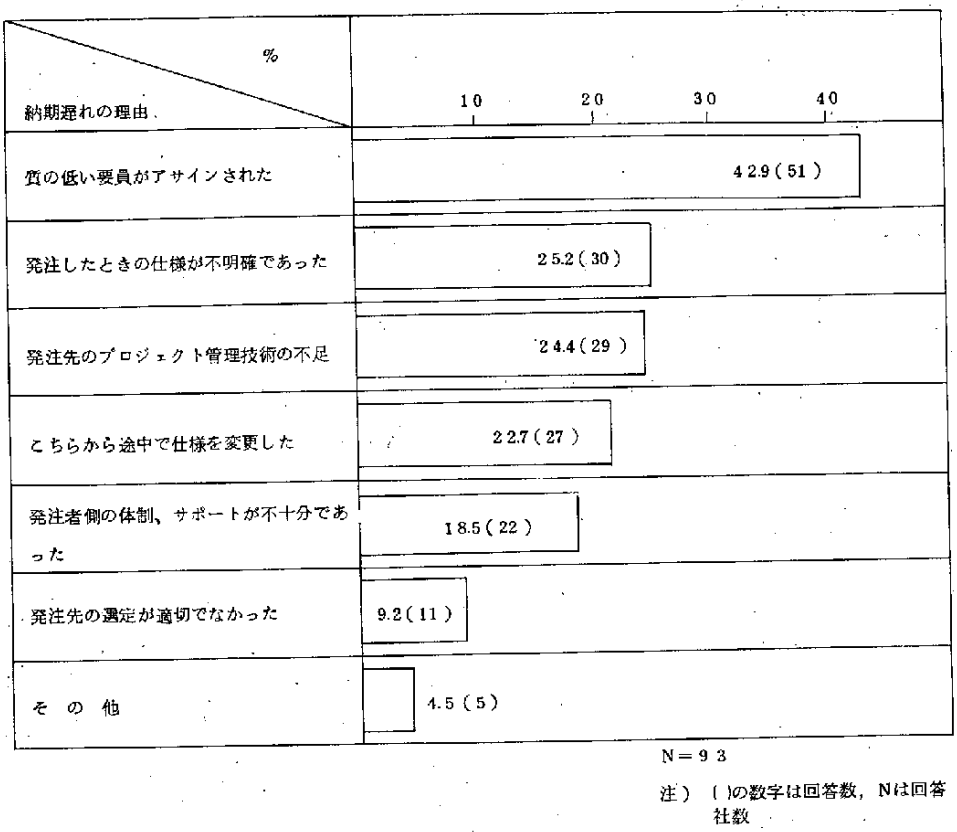


図3－21　外注プログラムの納期遅れの理由

こと踏まえたうえで，外注先の作業を管理する場合のポイントを示すと次のようになる。

① 外注先のスケジュール上に必らずレビュ工程を入れさせ，レビュ時には社員が必らず参加させるようにしなければならない。中間生産物のドキュメントを受け取って読むだけではどうしても十分なチェックができないばかりでなく，チェックの結果が出るのが遅れ，工程に支障を

きたすことも多い。

② システム化要求及びシステム分析工程においては，自社の要求を外注先に十分理解させなければならない。

また，この段階で，契約価格内での様々なトレード・オフについて十分検討し，実現する機能等について，全体のコンセンサスを得ておかなければならない。

③ システム設計工程においては，自社の要求がどのように実現されようとしているかを見極めなければならない。要求定義段階で，ある程度のコンセンサスが得られていても，この段階で，誤解やシステムに対する理解度の違いが顕在化することも少なくない。

④ プログラム設計工程においては，保守について考えなければならない。外注したプログラムは，納品されて自社のプログラムになるわけであるが，その後の保守（拡張，修正，改良等）は，エラーの修正を除いて，改めて外注しない限り，自社で行うことになる。この場合，プログラム設計書は最も有用な資料となる。従って，その記述が必要十分なものであることを確認すると同時に，変更に伴うリビジョン・アップも確認しておいた方が良い。

⑤ システムテスト工程は，外注先にとっても要求の実現度合を確認する重要な工程である。一般には，総合テストに入る前に，あらゆる可能性に対応した検査仕様書を作成し，これによりテストが行われる。従って，検査仕様書上に機能的な抜けがないかどうかを確認しておくことは重要なことである。

⑥ 作業上の管理の対象となるものは，中間生産物であり，作業そのものではない。作業そのもの，つまり開発の技法やプロジェクトの管理方法等については，特別な場合を除いて，外注先の自主性を尊重しなければならない。

3.4.4　検収時の留意点

調査の結果によると，検収時の問題点として，「全ての機能をテストするためのテスト・データの作成」が，際立って多くなっている。（図３－２２参照）

実際問題として，検収時にエラー処理を含む全ての機能についてテストすることは決して容易ではない。従って，検収時には主な機能のテストと効率テスト等を中心に行うようにし，その他の機能については，外注先に検査成績書を提出させ，それを確認する方法をとった方が効率的である。このようなことも含めて，以下，検収時のポイントについて述べる。

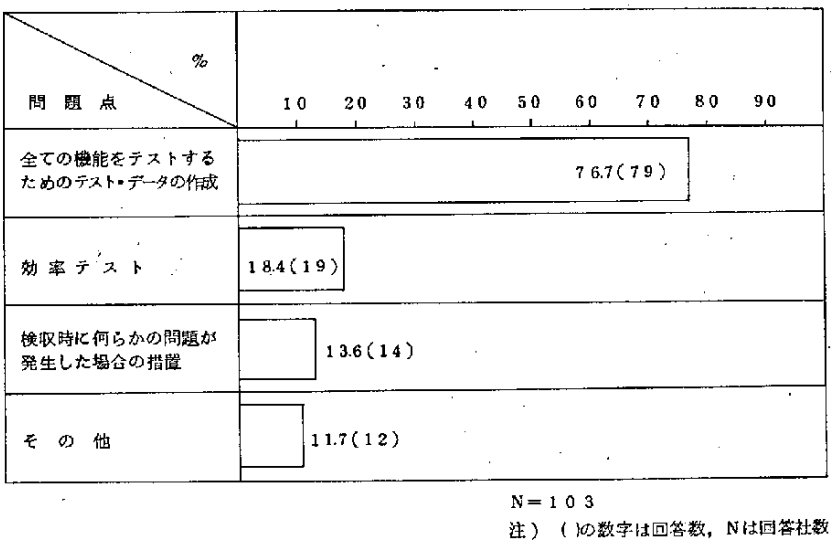


図３－２２　プログラム検収時の問題点

(1)　検収のためのチェック・リストの作成

検収にあたって行わなければならないことは，検収のためのチェック

・リストの作成である。これは，検収を効率的かつ網羅的に行うためには必要不可欠なことである。チェック・リストには，次のような項目が含まれる。

① 納品物件の種類や量（磁気テープの本数等）

② プログラムの機能に関するチェック・リスト及びそのテストの方法やデータの種類や量

③ マニュアルを中心として，納品ドキュメントの記載確認事項

実際の検収は，このチェック・リストに添って行われ，製品が評価されることになる。

以下，チェック・リストに盛り込まれる内容の中で，特に重要と思われることについて述べることにする。

(2) 検査成績書の確認

検収時には，本来，検査仕様書に盛り込まれるような，プログラムの全てのブランチを含むあらゆる可能性について実際に実行すべきであるが，事実上，それはできないことが多い。従って，あまり重要でないところは検査成績書をもって承認することになるので，検査成績書は出力リスト等も含めて十分にチェックしなければならない。

(3) 実際の稼動環境における性能テスト

外注した場合，プログラムが実際に稼動する環境は，テスト環境とは異なる場合が多い。従って，検収時には，そのプログラムが実際に稼動する環境（計算機システム，ＯＳ，データ等）の上でテストし，その主な機能の他に性能についても確認しなければならない。オンライン・システムのような場合は，レスポンス・タイムやターン・アラウンド・タイム等について，当初からある程度の要求を出される場合が多く，システムテスト工程で性能についてもチェックされるが，一般には機能のチェックが中心で，性能のチェックはあまりなされない。そこで，検収時にその性能についてチェックし，確認しておくことは，プログラムを使

用していく上で重要なことである。

(4) マニュアル等のドキュメントの検査

納品ドキュメントのうち，マニュアルは最も重要なものである。検収にあたっては，マニュアルの記載事項の抜けをチェックし，実際のテストはマニュアルに従って行うことになる。ここで，マニュアルの記載内容のチェックが行われ，もし，誤まりや不明なところがあれば，ただちに修正しなければならない。

(5) 保守体制の確認

一般に，ソフトウェア会社等で，プログラムを受注した場合，納品した後一年程度の保証期間を設定するのが普通である。この間に発生したエラーは無償で修正させられるが，このことに関しては，契約書に銘記しなければならないことはいうまでもない。

また，その一年の間，どのような保守体制がとられるのかも確認しておく必要がある。具体的には，連絡可能な担当者名，電話番号，ディフィカルティ・レポートの書き方，必要な添付資料の種類（リスト，コンソール・シート，ディスク・ダンプ等）等である。

4. プロジェクトの評価

4.1 評価の考え方と対象

4.1.1 目標設定と評価

マネジメント・サイクルとして，一般に言われているPLAN（計画）－DO（実施）－SEE（評価）の考え方は，プロジェクト管理の基本であり，本ガイドにおいて各々のフェーズはプロジェクトの計画，プロジェクトの実施，プロジェクトの評価として，とらえられている。

その中で，プロジェクト評価の視点を考えた場合次の2つに大別できる。

① プロジェクトの開発途上における管理・統制のための評価（工程毎のレビュなど）

② プロジェクトの最終評価

また，一般的に評価尺度としては過去のデータの積み上げにより導き出されたもの，あるいは達成すべき目標値があるが，ソフトウェア開発においても評価尺度をあらかじめ設定し，それらを達成するための基準値を設定することにより管理・統制あるいは評価して行くことが重要である。この考え方は，プロジェクト全体のみならず，プロジェクトを細分化したサブ・システムや各工程などについても適用されなければならない（図4－1）。

上記2つの評価のうち開発途上における評価については第3章（プロジェクトの実施）で設計やテストのレビュ工程という観点から述べたので，本章においては最終段階で必要となるプロジェクト全体の評価について述べる。プロジェクト全体の評価にあたっては，各工程で評価され得ないものを評価し，次のPLAN（計画）に継続して行くことが大切である。

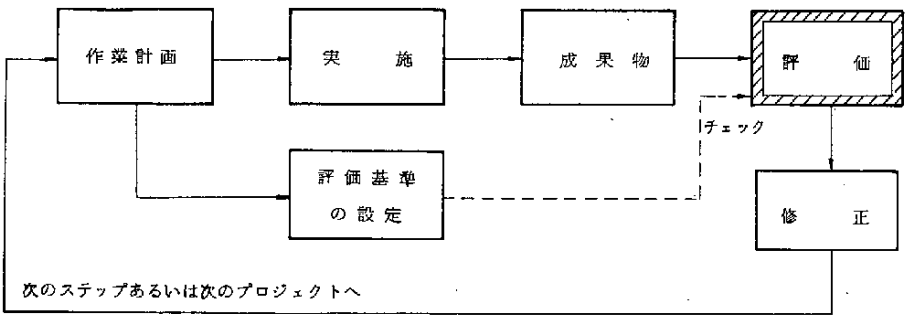


図４－１　作業の基本フロー

4.1.2　データの収集

プロジェクトの評価は，できる限り客観的なデータあるいは情報にもとづき行われなければならない。特に，計画通り進行しなかった場合には，弁解を行ったり，責任の転嫁を行ったりしがちであるが，謙虚に結果を評価する姿勢が必要である。また，良好な結果を得た場合についても，過大評価とならないよう，見積り等で過剰な点がなかったか分析することも重要である。

プロジェクト評価用データには，①開発段階で収集すべきデータと②プロジェクト完了後に収集するデータの２つの面がある。プロジェクト全体の評価は，それらを総合的にみることにより行われる。特に，開発段階で収集するデータはプロジェクト全体を細部に亘り分析する場合に必要となるものであり，プロジェクトの計画・実施における管理・統制が妥当であったか否かを判断する際の基礎となる。プロジェクトの実施において，製造や修正作業などに追われ，この種の管理データが収集されず，経験や勘にたよってしまった場合には，プロジェクト管理上の適切な行動が遅れるばかりでなく，評価の段階でも困難が生ずる。データの収集は各段階においてタイムリーに取得しないと失われてしまうことが多く，プロジェクト終了後に必要なデータが散逸して無いということも起りかねない。

そのためには，データの収集方法やルールをプロジェクト計画の段階あるいは作業計画の中に明確にしておかねばならない。とかく，作業者は，作業と直接関係のない資料の作成を敬遠する傾向がある。従って，出来るだけ簡易な形で必要十分なデータの収集に努めるべきであり，また，作業者への動機付けも重要となろう。可能ならば，一連の作業の流れの中でデータが自動的には握されてゆくことが望まれる。

具体的には開発段階で収集すべきデータとしては，各工程別，要員別などの工数計画／実績，工程計画／実績，計算機使用時間予定／実績，各種ドキュメント，レビュ結果，テスト計画／結果などの積み上げられたデータであり，あるいは開発段階の評価に従ってとられた実際の処置（アクション）などである。また，プロジェクト完了後に収集すべきデータとしては，システムの性能，信頼性，機能，保全性，操作性などの品質に関する評価データ，各種ドキュメント，経済的効果，能率向上，開発効率など各種の項目があげられる。

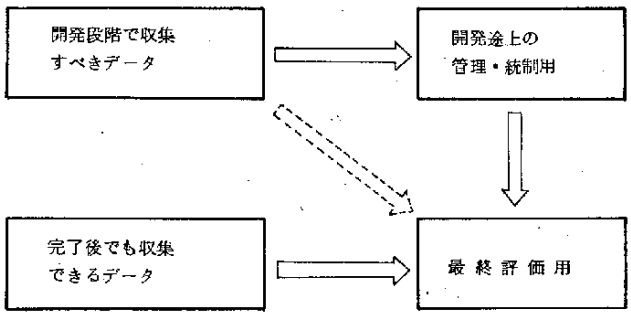


図４－２　データの収集

4.1.3　評価の対象

以上のような観点からプロジェクト全体の評価の対象としては次のものがあげられる。

1つは，実施されたプロジェクトの計画／実績，内容，効率等の評価であり，いま1つは，それらの評価から将来のプロジェクトにおける計画・実施へフィードバックすべき点としての各プロジェクトで共通的な基本課題である。具体的には次の項目があげられる。

(1) 実施プロジェクトの評価

① 見積りと実績の評価

② 日程計画と実績の評価

③ バグ原因の分析

④ 各種指標の評価

⑤ 当初の要求と実際の評価

(2) 評価のフィードバック

① 基準生産性の見直し

② チーム編成及び外注の見直し

③ 開発方法論の評価

④ 開発ツールの評価

今回の調査で，実際にユーザがどのような点についてプロジェクトの評価を実施しているかを質問した結果を図4－3に示す。

この中では，「日程計画と実際の対比」，「予算と費用の対比」が非常に高く，やはり納期と費用（予算）がプロジェクト管理の2大重要項目であるということを示している。その他としては「見積り時の基準生産性の見直し」，「工程別工数分布の分析」，「バグ原因の分析」などが比較的多いが全体の中では20％程度に留まっている。また，「特に分析は行っていない」というところも21.9％とかなり多い。

以上の様な結果からみる限りでは，プロジェクト終了後の評価は余り積極的に実施されていないといえよう。その理由としては，

① 各ユーザは，多量のソフトウェア開発要求を抱えているために余裕がない。

② プロジェクト管理の有効なツールがない。

③ プロジェクト開発の標準がある程度確立されており大きな予実の差異が余りない。

④ 評価の標準化が困難

などがあげられよう。

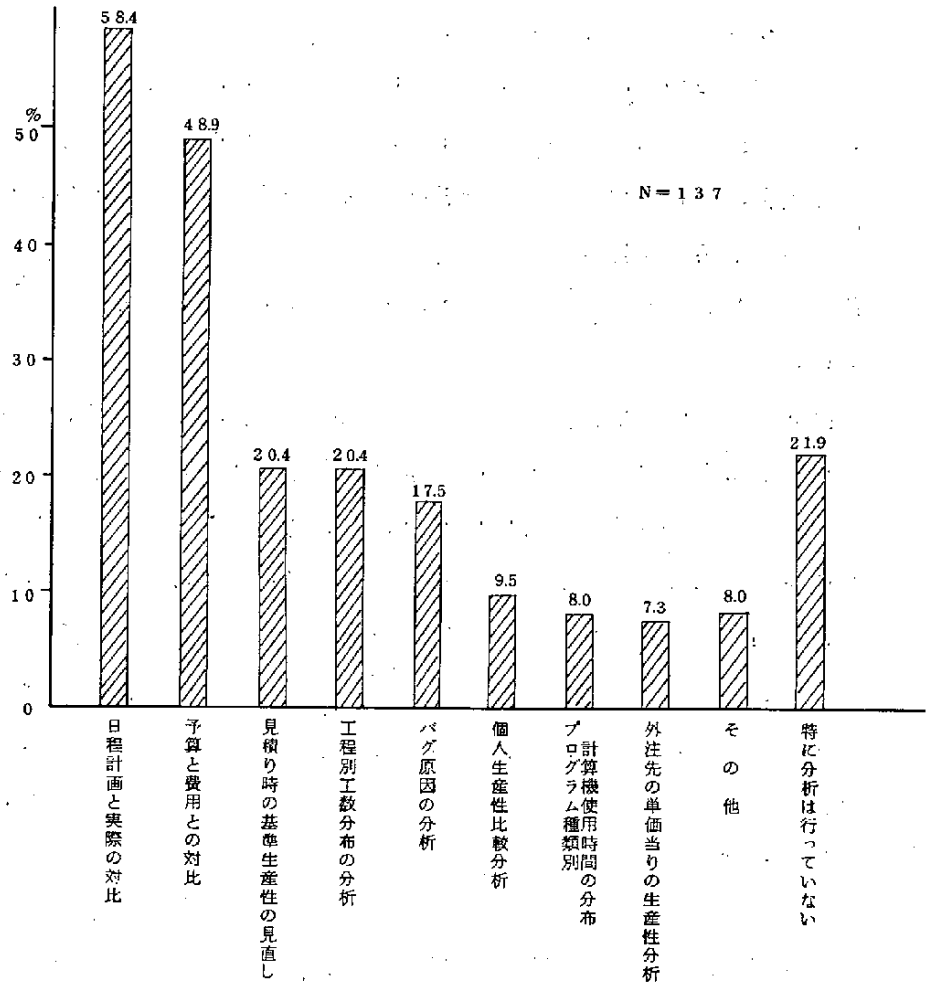


図4－3　プロジェクト終了時の分析評価

4.2 プロジェクト全体の評価

4.2.1 見積りと実績の評価

プロジェクト管理の中で最も重要視されるべきものは費用と納期であることは前述した調査結果からも明らかである。これらは，プロジェクト計画の段階で設定され，プロジェクト全体を通して管理・統制される。

ここでは，その中で費用の見積りと実績に対する評価として主な項目について述べる。

(1) 開発規模・工数の見積りと実績

プログラマの生産性とシステムの開発規模の見積りはソフトウェア・コストを推定する上での2大要素であることは自明のことである。しかし，現在，これらに対する特効薬的な手法はなく実際の場では経験と勘にたよっているのが現実である。従って，プロジェクト完了後，計画と実績を対比してみた場合に大きな差がでてくるということはかなりの確率で存在する。

図4－4は開発規模（ソースコード行数）の見積り・実績対比を示した例であるが，20～30%の誤差の範囲に70%程度が納っている。しかし，残りの約30%は問題となる程に見積りをオーバーしており，2倍近くに膨張してしまっている。一方，開発工数の見積り・実績対比の例を図4－5に示すが，開発規模の場合と異なり10%の誤差の範囲に集中している。開発規模（ステップ数／単位工数）で除して所要工数を求めるのが普通一般の方法であることを考えると，上記の差異は規模見積りの誤差を生産性の向上（ベテランの投入，既存ソフトの流用etc）で補完したということになろう。

今回の調査においても見積りの不備による費用オーバーをあげているユーザが多かったが，規模の見積り，工数の見積りの誤差に対して，特に異常値を示したものについて，その原因を種々の観点から分析・検討を加え，今後の見積り精度向上に反映してゆく必要があろう。また，見積

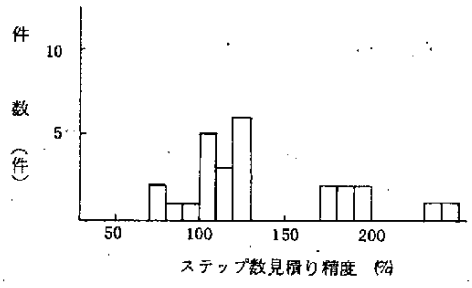


図４－４　ソフトウェア開発の見積り精度の例

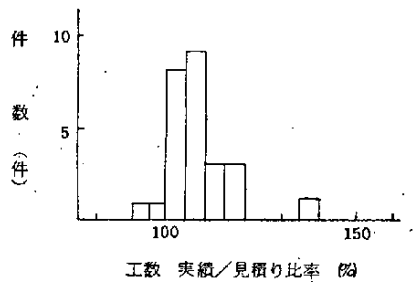


図４－５　開発工数見積り精度の例

りオーバー時に有効な生産性向上対策についても指針やチェックリストとして記録に留める。図４－６，表４－２は見積り計画／実績対比表の例である。

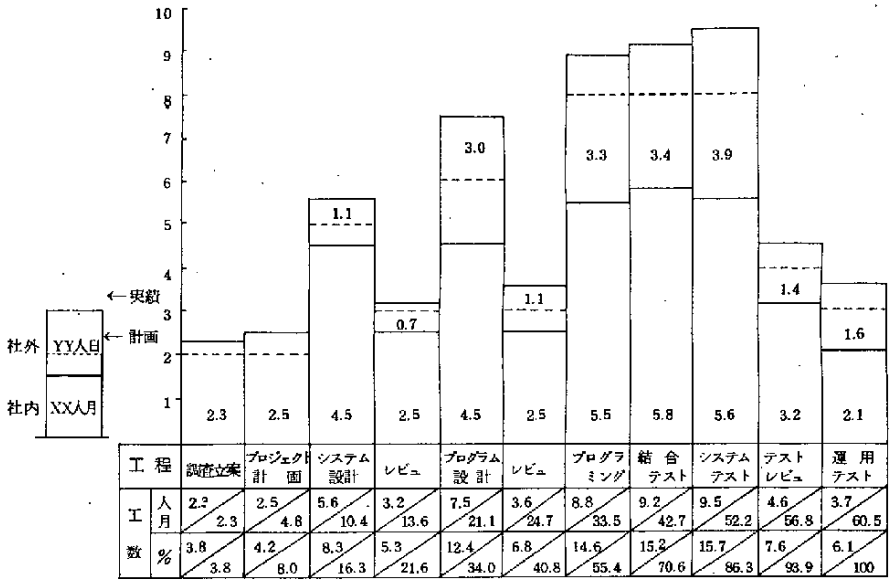


図４－６　工数計画／実績対比図の例

表４－１　工程別工数計画実績比較表の例
業務名　：　A

工数（人月） / 工程	社内		メーカ		外注		合計		所要期間（月）			
	計画	実績	計画	実績	計画	実績	計画	実績	計画	実績	増減	累積
調査・立案	2.0	2.5					2.0	2.5	1.0	1.0	0	0
プロジェクト計画	2.0	2.6					2.0	2.6	1.0	1.5	＋0.5	＋0.5
システム設計	2.0	1.5	4.0	4.8			6.0	6.3	2.0	2.5	＋0.5	＋1.0
システム設計レビュ	2.0	1.5	2.0	2.5	3.0	2.5	7.0	6.5	0.5	0.5	0	＋1.0
プログラム設計	3.0	2.5	3.0	3.5	4.0	3.2	10.0	9.2	2.5	2.0	－0.5	＋0.5
合計（人月）	35.0	40.8	20.0	16.5	35.0	38.5	90.0	95.8	12.0	12.5	＋0.5	＋0.5

表４－２　業務別（サブ・システム別）工数計画実績表の例

業務別工数 / 工程	A業務		B業務		合計	
	計画	実績	計画	実績	計画	実績
調査・立案	2.0	2.5	1.0	1.0	3.0	3.5
プロジェクト計画	2.0	2.6	2.0	2.5	4.0	5.1
システム設計	6.0	6.3	4.0	4.5	10.0	10.8
システム設計レビュ	7.0	6.5	6.0	6.5	13.0	13.0
プログラム設計	10.0	9.2	8.0	8.4	18.0	17.6
合計（人月）	90.0	95.8	85.0	90.0	175.0	185.8

また，見積りの誤差に影響する。要因としては次のものがあげられる。
図４－７は工数予測値ズレの原因別割合を示した例である。

①　記述言語の影響

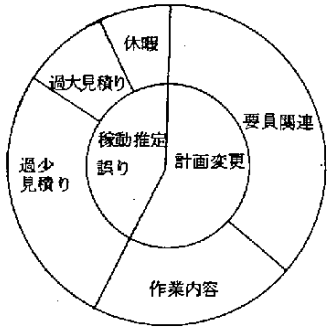
②　プログラム規模

③　開発経験不足

④　難易度の考慮の有無

⑤　契約形態からくる影響

⑥　その他（プログラム製造方式，プログラム構造など）



（備考）　各原因の詳細内容は以下のとおり
要員関連：不足／余剰
作業内容：繰上げ／繰延べ，作業追加／削除
過少見積り：不測作業の出現，勤務時間の延長
過大見積り：作業の立上り不足，予想外に順調

図４－７　工数予測値ズレの原因別割合の例

さらに，工数計画／実績の対比を工程別，プログラム種別別，プログラム規模別等で分析することにより，計画見積りと実績の差異を細部に亘り検討でき，次のプロジェクトにおける見積りや工数配分に生かせる。表４－３は，システム開発における工程別工数比の例である。

表４－３　システム開発における工数比の例

開発プログラム例		開発年時	設　計	コーディング	テスト	記　　事
○○研	オペレーティングシステム(A)	1975	28	10	62	高級言語，アセンブラ 340 ks
	オペレーティングシステム(B)	1976	29	11	60	高級言語，アセンブラ 370 ks
	言語処理プログラム　(A)	1977	29	20	51	高級言語，アセンブラ 900 ks
	言語処理プログラム　(B)	1977	35	9	56	高級言語 110 ks
	言語処理プログラム　(C)	1978	44	26	30	高級言語 50 ks
参考	C. Weissman	1973	40	20	40	詳細不明
	M. Zelkowitz	1978	35	20	45	

(2) 計算機時間の見積りと実績

最近では計算機のハードウェアの低価格化により次第に計算機使用時間のコストに占める割合が低下してきている。また，デザイン・レビュやウォークスルーなどの重視によりマシン・デバッグを能率化しようとする動きもみられる。しかし，計算機のコストが無視できない場合，あるいは，計算機時間に余裕がない場合にはプロジェクト管理の面から計算機時間の見積り・配分を計画し使用実績をは握しておくことが必要である。ただ，計算機コストのウエイトが小さい場合には，管理や評価のための工数が増えることのないよう自動的に計算機の使用実績がとれるようなツールの開発・利用，あるいは大まかなは握に留めるなどの工夫もあろう。

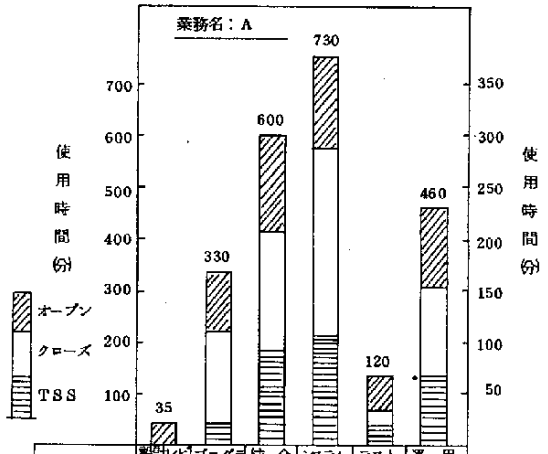
計算機時間の評価についてはコストは握という面のみでなく，工程計画での計算機使用時間の確保，配分の面からも工程別にどの程度のマシン時間が必要か，あるいは，計算機の使用形態（表４－４）によるデバッグの効率などとの関連で分析・は握しておくことも考えられる。表４－５，表４－６は，計算機の使用実績表の例である。

表４－４　デバッグシステムの利用技術

運用区分	利用形態				主な利用分野		計算機の利用場所
					制御プログラム	処理プログラム	
共同利用	仮想計算機方式	ローカルバッチ処理（オープン利用）			○	△*	センタ
	リモートデバッグ方式	会話処理			△*	○	端末
		バッチ処理	リモートバッチ処理				
			ローカルバッチ処理	クローズ利用			センタ
				オープン利用			
占有利用	ローカルバッチ処理（オープン利用）				○	○	センタ

＊△：効率的ではない利用分野

表４－５　工程別計算機使用実績表の例



工程		設計レビューまで	プログラミング	結合テスト	システムテスト	テストレビュー	運用テスト
使用時間	オープン		120	180	180	60	150
	クローズ	35	180	240	340	30	180
	TSS		30	180	210	30	130

表４－６　月間計算機使用実績の例

単位：分

業務名＼年月		52			53			合計	
		10	11	12	1	2	3		
A業務	オープン								
	クローズ			570	450	380	150	1,550	3,232
	TSS			610	520	450	102	1,682	
B業務	オープン	180	150	210	180	120		840	
	クローズ	350	310	180	190	100		1,130	2,405
	TSS	85	95	105	85	65		435	
合計	オープン	180	150	210	180	120		840	
	クローズ	350	310	750	640	480	150	2,680	5,137
	TSS	85	95	215	605	515	102	1,617	
	計	615	555	1,175	1,425	1,115	252	5,137	

（注）測定単位　・オープン …… 使用時間
・クローズ …… メモリタイム
・TSS …… セションタイム

(3) 費用の見積りと実績

以上述べた工数，計算機使用時間，その他資材などを含めた総費用に対して，当初の計画と実績の差異について，工程別，サブシステム別などでまとめ，原因分析し今後の見積り方法，見積り基準などに生かして行かねばならない。その場合に，純粋に見積り方法に誤りがあったのか，要員の技術力不足だったのか，ユーザの仕様変更のためかなど，今後の具体的な改善に結びつくような形でまとめる。

4.2.2 日程計画と実績の評価

費用に対する予実対比と同程度に重視される点として納期の問題があることは前述した通りである。プロジェクト計画で予定された工程が実績とどのように差異を生じたかについて対比し，その遅れや工数の増減について分析を行う。

一般に，日程計画が予定通り実施されるケースは稀なことであり，実施段階の進捗管理により差異を早期に発見し，予実差を予測し適切なアクション（ベテランの投入，補助員の増加，マシン・タイムの割当上の便宜を図る　など）を起し，遅れを修正し納期の確保に努めているのが現状であろう。傾向としては，結果として工数の増加につながることが多いが，単なる工数の追加で解決しない場合も多い。

工期が遅れる原因の例としては，次の様なものがあげられる。

① 無理な計画
② 計画要員不足
③ 教育不足・要員能力不足
④ 品質不良による手戻り
⑤ 仕 様 変 更
⑥ マシン・トラブル
⑦ 見積りの不備

⑧　管理技術の不足

⑨　管理ツールの不備など

日程計画の実績の対比は，各工程別，サブ・システム別，要員別などで表わす。また，工数との関連で計画実績表として一つの図表で表わし易くする工夫もされている。これらの総合的な評価としては，遅れの原因，対処法，今後の問題点などの形で考察を加える。表４－７，表４－８は計画／実績対比の例である。

表４－７　開発工程の計画実績線表の例

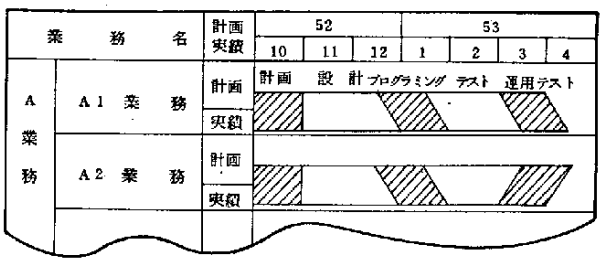
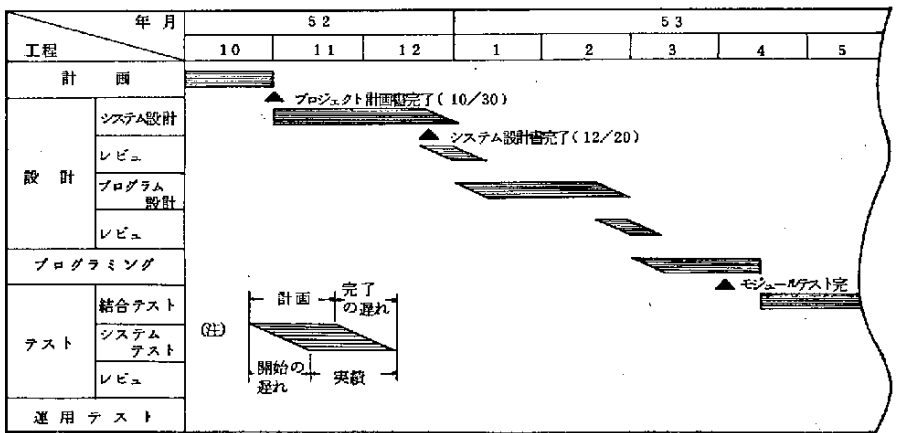


表４－８　日程計画／実績線表の例



実績工程		計画	システム設計	レビュ	プログラム設計	レビュ	プログラミング	結合テスト
総期間	(月)	1.0	2.0	0.5	1.5	0.5	1.0	1.5
10.0(月)	(%)	10.0	20.0	5.0	15.0	5.0	10.0	15
総工数	(人月)	2.5	8.5	3.2	13.2	6.5	12.5	16.2
85.3(人)	(%)	3.0	10.0	3.8	15.5	7.6	14.7	19.0
要員数		2	4	6	8	10	10	10
生産物		・調査立案書(10/15) ・プロジェクト計画書(10／30)	・システム設計書(12／20) ・テスト計画書(12／25)	・レビュ報告書(1／15)	・プログラム設計書(2／26) ・テスト手順書(2／27) ・システムテスト仕様書(2／20)	・レビュ報告書(3／15)	・プログラムリスト ・モジュールテスト成績書(4／12)	・結合テスト成績書(5／30)

4.2.3　バグ原因の分析

ソフトウェアに内在するバグをいかに減少させるか，早期に発見し修正するかについては製造されるソフトウェア製品の品質管理・保証として重要である。この問題については，3.2節に具体的に述べられているように設計からテスト工程に至るまで各種の方法が考えられ実施されている。最近では，バグを発見するという考えから，バグが入り込むのをいかに防止するかという考え方に重点が置かれるようになった。例えば，プログラム設計，製造工程の出来るだけ前の段階でエラーを発見すること，あるいはジェネレータ等の利用によりトータルな開発工数等の減少をはかり，生産性向上と品質向上を目指した方法があげられる。この様な動きの背景には，過去のソフトウェア開発におけるバグの発生原因を統計的に分析し，その結果を開発手法へ反映してきたことが大きな要因としてあげられる。

製造工程におけるバグの管理としては，主に工程毎の基準値による量的な把握が主体となると考えられるが，最終的な評価におけるバグの原因分析としては，それらを集約した形でプロジェクト全体を通してみることとなる。

即ち，具体的にはバグの発生傾向を工程別，サブ・システム別，要員別，原因別等に分類し分析することとなる。また，発生したバグの修正に要した工数が各々どの程度のものであったかなどについても分析することにより，最も効率的な開発のあり方について示唆を与えてくれる。各工程で目標としたテストが十分に行われたか，不十分な点は何か，人間的な要員か開発ツールの問題か，外注プログラムに多いか，単純なものが多いかなど種々の観点からバグの原因分析がなされよう。一般的に利用されているものとしては次のものがあげられる。

① 人間的要素による分析

注意不足か検討粗漏か

② 不良現象からの分析

デグレード，修正不十分，ドキュメント不良，制限解除不良　外囲条件変化など

③　工程からの分析

計画工程，設計工程，製造工程，検査工程など

また，バグの発生防止策としては表４－９のようなことが考えられ，事後保全から予防保全的な方法が重視されている。

図４－８，表４－９，表４－１０はエラー原因分析の例であり，表４－１１，図４－９，図４－１０はエラー修正工数に重点を置いた分析の例である。

この様なバグの発生原因分析により，ソフトウェア開発の重要な課題が明らかとなることが多い。例えば，設計工程のエラーが製造工程のエラーより多い場合には設計品質が低いことを示し，設計技法や改善や設計レビュの方法について見直しが必要となる。また，製造工程のエラーがプログラミングからシステムテストに進んでも収束しない場合については，テスト項目の設定数や抽出法に問題があるかもしれない。また，人的要因でケアレスミスなどが多い場合には，教育・訓練あるいは周知方法やコミュニケーションに改善が必要である等種々の原因が見つかるであろう。これらを次の開発に生かして同じような誤りのないよう修正して生かねばならない。

当然のことながらバグの原因分類等については作業標準やドキュメント標準として予じめ明記され，その分析・利用法も明確にしておかねばならない。データは実施段階におけるバグ管理図表から自動的に集計・図表化されることが望まれる。

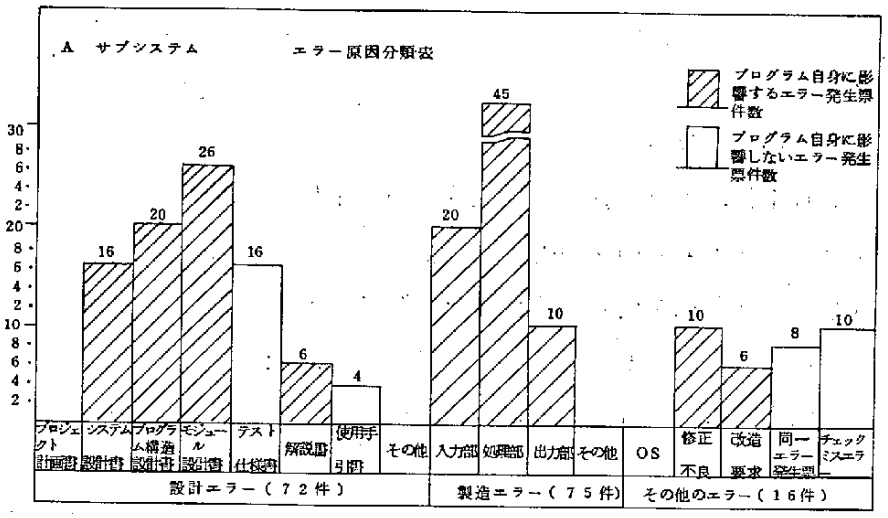


図４－８　サブ・システム別エラー原因分類表の例

表４－９　プログラム構造からみた分析の例

分　　類	定　量　化　項　目　例
データ構造参照	ビット変数の数 ポインタ変数の数 構造体のレベルの深さなど
データ処理	ステップ数 機械語記述文の数 ポインタ修飾の数 データのグローバル参照数
制御構造	GO TO 文の数 DO 文の数 パ　ス　数
インタフェース	入口・出口の数 外部手続きの fan in，fan out 数 ブロックのネストの深さ

表4－10　バグ原因の分析例

バグ原因	現象	① インタフェース設定の誤り …… (19%) ② 条件ぬけ，判定誤り …… (17) ③ ディグレード …… (13) ④ データ移送の誤り …… (9) ⑤ 記述誤り …… (7) ⑥ その他 …… (35)
	人的要因	① 確認不十分 …… (41%) ② ケアレスミス …… (26) ③ 思い込み …… (13) ④ 修正もれ …… (6) ⑤ 独断 …… (4) ⑥ その他 …… (10)

表4－11 サブ・システム別エラー件数・修正工数の例

工程＼サブ・システム名		A サブ・システム		B サブ・システム		全体	
		合計	エラー1件当り	合計	エラー1件当り	合計	エラー1件当り
プログラミング	エラー件数	103		110		213	
	工数（人・時）	418	4.1	560	5.1	978	4.6
	マシン（分）	408	4.0	380	3.5	788	3.7
	所要日数（日）	655	0.7	75.5	0.7	141	0.7
テスト 結合テスト	エラー件数	38		32		70	
	工数（人・時）	205	5.4	206	6.5	411	5.9
	マシン（分）	365	9.6	350	11.0	715	10.3
	所要日数（日）	395	1.1	30.5	1.0	70	1.0
テスト システムテスト	エラー件数	19		35		54	
	工数（人・時）	185	9.8	380	10.9	565	10.5
	マシン（分）	560	29.5	1,080	30.9	1,640	30.4
	所要日数（日）	55.5	3.0	110.5	3.2	166	3.1
運用テスト	エラー件数	3		5		8	
	工数（人・時）	65.5	21.9	130.5	26.1	196	24.5
	マシン（分）	195	65	370.5	74.1	565.5	70.7
	所要日数（日）	9.5	3.2	20.5	4.1	30	3.8
プログラミング＋テスト全体	エラー件数	163		182		345	
	工数（人・時）	873.5	5.4	1,276.5	7.1	2,150	6.3
	マシン（分）	1,528	9.4	2,180.5	12.0	3,780.5	10.8
	所要日数（日）	170	1.1	237	1.3	407	1.2

（注）・工数は修正に要した直接時間
・所要日数は，エラーが完全に修正され正常動作が確認されるまでの所要日数（ターンアラウンド）を示す（半日単位）。

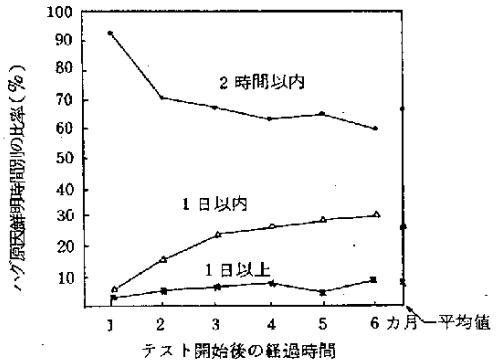


図４－９　バグ原因解明時間の推移の例

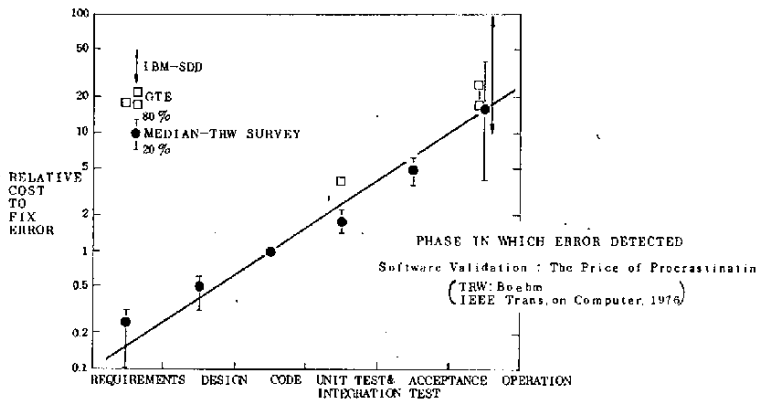


図４－１０　工程別エラー修正工数例

4.2.4　指標の評価

ソフトウェア開発における指標としては，大別すると①生産性指標，②品質指標，③その他があげられる。これらはプロジェクト開発の計画時あるいは工程別計画時に設定される。これらの中には，過去の開発経験から積み上げられてきた目標値としての指標がある。

最終的なプロジェクトの評価の段階においては，この様な指標がプロジェクトの計画と実績で対比されることが必要となり，ユーザの要求を満足すべき製品となっているか，あるいはプロジェクト計画の達成度，計画・実施の妥当性，指標の有効性などの評価につながる。

(1)　生産性指標

生産性の指標としては，ステップ／人月，計算機時間／人月，計算機時間／KS などのプログラム生産性，ドキュメント生産性，工数消化指

数などがある（表４－１２）。生産性に対しては，プロジェクト計画（第２章）でも述べているように，対象期間をどこに設定するか，ステップ数には何をとるか，異機種の計算機がある場合の計算機時間の換算比は何をとるか，あるいはドキュメントのサイズは何を基準とするかなどあらかじめ明示されていなければならない。これらの指標は計画時に使われた指標と同じもので実績対比する。

表４－１２　生産性指標

生産性指標	計　算　式
プログラム生産性	ステップ数／人月
	計算機時間／人月 計算機時間／Ｋステップ
ドキュメント生産性	ドキュメント枚数／人月

(注)　ステップ数，計算機時間，ドキュメント枚数，人月は計画時に予じめ設定。

(2)　品　質　指　標

品質指標としては表４－１３に示すように，エラーに関するもの，テストに関するもの，及びドキュメントに関するものがあるが，これらは，品質管理や進捗管理にとって重要なポイントとなる点である。詳しくは前述の第２，３章を参照のこと。これらは，主に各工程毎に設定され管理・評価されることとなるが，更にプロジェクト開発から運用に移行する時の大きな評価項目でもある。

例えば，最近，保守・運用のウエイトが重くなっていることを考えると，たとえプログラム自体の品質が良好であっても，保守性からみてドキュメント不良（不正確，要求条件と不一致，プログラムと不一致，ドキュメント標準と不一致など）が多い場合には評価を下げねばならない。これは，設計，製造，テスト，保守という工程が分離してゆくためには，重要な条件である。

表4－13　品質指標

品質指標	計算式
エラー率	エラー発生件数 / ステップ数
テスト項目設定率	テスト項目設定数 / ステップ数
テスト項目消化率	実施テスト項目数 / テスト項目設定数
ドキュメント不良率	ドキュメント不良件数 / ステップ数

4.2.5　当初の要求と実際の評価

ソフトウェア開発プロジェクトの開発効率の向上というプロジェクト・マネージャの要求とは直接的には関係はないが，プロジェクトの最終的な評価として重点な点に当初ユーザが期待していた要求に対して，どの程度満足しているか，あるいは開発されたシステムによりどの程度効果があがっているかについては重要な評価項目となろう。

現在ソフトウェア開発プロジェクトにおいて，大きな問題の1つにユーザとのコミュニケーション不足，あるいは仕様変更による費用の超過，納期の遅れがある。これには，ユーザの要求を十分には握できないということが原因しており，企業内開発でも，ソフトウェア会社における場合でも共通的に見られる。

従って，導入効果を含めたユーザの満足度について調査し分析評価しておくことも必要である。もし，必要ならば，直接ユーザに，アンケートやヒアリング等を実施する方法もあろう。

表4－14　業務評価の例

〔プロジェクト基本計画書にうたったシステム開発による効果について，システム稼動後の評価を行う。〕

評価項目	開発前	効果目標	評価	今後の指針
配送通知のターンアラウンド	・郵送又は電話連絡 ・伝票パンチ ・処理 各支店からのデータがそろわないと処理できないため遅れが多い ・ターンアラウンド 2～5日	・オンラインデータギャザ ・OMR伝票 ・オンライン集信，夜間自動送信 ・ターンアラウンド 集信：即時 配信：翌朝	・ターンアラウンドは目標を達成 ・OMR記入ミスが多い（約3％） ・エンドユーザは非常にメリットを感じている	・ピーク時のターンアラウンド評価（10／末） ・使用性の評価（10／末） ・拡張性の評価（12／末） ・保守性の評価（12／末）
在庫削減	・不良在庫額1500万円	・在庫50％削減	—	12月／末の棚卸し結果により1月／末評価報告

4.3　評価のフィードバック

前節で述べたプロジェクト全体の評価は，プロジェクトの計画と実績の対比を基に，主に目標の達成度という観点からの評価である。プロジェクトが単発的なもので継続性がないならば，それで終ってしまうこととなる。しかしソフトウェアの開発においてはサイクル性があり，次のプロジェクトでは，過去の開発経験が非常に有効であることがいわれており，1つの開発を足場として次の開発へ発展させてゆく必要性が叫ばれている。また，今後の企業

環境においては，低成長時代を迎え全社的な企業活動の効率化が必要となろう。そのためソフトウェア開発部門においても一層の生産性向上が求められている。

プロジェクトの評価は，単に人的な蓄積として継承されるのみでは十分に生かし切れない。既ち，客観的なデータあるいはドキュメントとして残し，連絡会あるいは，プロジェクト管理部門等の組織を通じ関連者へ周知，あるいは教育部門への反映等により初めて活用され開発効率の向上に結びつくものである。

4.3.1 基準生産性の見直し

ソフトウェアの生産性を評価する場合，種々の問題に遭遇する。それらは本質的な問題を含んでいるものと思われ，その解消には長時間を要しよう。このため，実際には開発環境に既した現実的な前提の下に簡便な方法を用いることにより，正確性の点では若干の難点はあるが，大筋をは握できる程度の評価を実施しているのが現状である。ここでは，評価上の問題点に少しふれ，生産性の見直しについて述べる。

(1) 生産性評価の問題点

ソフトウェア開発における生産物として何を選択し，どのように測定するかは基本的な問題点であるにもかかわらず統一的な見解がない。少なくともドキュメントとプログラムがあるという共通認識はあるが前者については直接の生産物として扱っていない場合が多い。

一方，プログラムを生産物とする場合に生産量の単位（尺度）に関する問題が顕在化する。一般には，プログラムの尺度としてソース・コード（行またはステップ数）を用いるものが最も普通であるがその場合にも次のような点が問題となる。

① 処理方式によるステップ数の相違

アルゴリズムによる差。ステップ数が大きいものが評価が高いとは

いい切れない。

② 記述言語による相違

アセンブラ，コンパイラ，プリコンパイラなど

③ 生産量に含めるステップ数の範囲

コメント文の扱い。行数，ステートメント数，命令数，オブジェクトコード数など

④ ツールやテストデータ等の処理

生産物としての認知，ステップ換算など

⑤ 改造におけるステップ数の計数

既存ソフトウェアの流用・追加・削除等の換算

⑥ そ の 他

要員レベル（上級，中級，初級など），プログラムの種類（オンライン－バッチ，業務－システムなど），ＯＳの機能の差による生産性の相違

また，開発工数の考え方にも次のような曖昧さが存在する。

① 開発工数に含める作業の明確化

評価・研究活動，技術協力・教育活動，マニュアル・運転説明書作成，共通業務などの取捨

② 工数の計数範囲の規定

持ち帰り作業の取捨・換算

③ 標準工数の設定

人年・人月・人時の標準

さらに，生産物の完成度としての品質差をどのように生産性に反映させるかという問題もある。

(2) 生産性の見直し

以上のような問題点を考慮し，どのような前提条件で生産性を評価してゆくのかに対しては，一定の基準を設け，個々のプロジェクトでの差

異を共通的に評価可能なよう処置されねばならない。それにより，管理的な基準生産性の確保が可能となろう。

このような基準値に影響を与えると考えられる開発環境の変化に対しては新しく基準生産性を見直し，開発効率の向上に見合ったものを設定していくことが，更に一層の生産性の向上につながるものと期待される。

4.3.2 チーム編成及び外注の見直し

(1) チーム編成

プロジェクト管理における組織体制・要員管理については第Ⅱ部で詳しく述べているが，ソフトウェア開発においては人的資源が最も重要であることは明白である。要員の能力を最大限に引き出せるチーム編成づくりが望まれており，各種の開発組織が提唱され実施されている。

要員の評価は，そのような最適なチーム編成作りを実現していく上で必須のものであり，その能力，適性，経験実績などを総合的に評価し次のプロジェクトに生かさねばならない。現実的には，明白な形で個人的な評価がなされることは困難なためプロジェクト・リーダの判断に任されている場合も多い。しかし，ソフトウェア開発における担当者による格差は生産性で8倍，品質（バグ数）で10倍という報告もあり，個人別生産性，個人別バグ発生原因比較分析などにより，チーム編成や作業標準化の効果について評価することが望まれる。

また，プロジェクト全体の評価においてチームの編成上に問題がなかったか，それは組織体制そのものの問題点か，不適切な要員配置によるものか，技術力不足によるものか等分析し評価を行う。

(2) 外　　注

前章3.4節で述べられているように各種の必要性により種々の形態で外注が活用されている。外注の場合においてもプロジェクト管理上，特に例外となるものではないと考えられるが，主に生産物（プログラム

とドキュメント）に関する評価が中心となり，工程計画／実績と品質などが重要となる。プロジェクト全体の評価の中でも，自社内，外注を対比した形で計画／実績の比較分析を行ない評価する必要がある（図４－６，表４－１参照）。

今後，ソフトウェア需要の増大，保守の増大に伴い外注は増加する方向にあるとも言われており，積極的に活用していこうとする動きもみられる。その場合には，外注に関する管理・評価も自社内開発と同様に重要となり，外注先別評価，外注形態別評価等が必要となろう。

4.3.3 開発方法論の評価

(1) 開発方法論（技法）

ソフトウェア開発の生産性向上に関する問題意識から開発理念，開発方法論が各種提唱され，それに立脚して具体的な技法（手法），開発ツールが研究・開発されてきた。開発方法論は開発技法と密接な関係にある。大別すると分析技法，設計技法，プログラミング技法があるが，更に管理技法，テスト技法を入れる例もみられる。また，これらを開発段階の広範囲で適用できるよう総合的な技法として各社から提供されているものもある。

(2) 方法論の見直し

プロジェクトの評価から発生してくる問題意識を発展させてゆくと開発方法論の評価につながる。これは，図４－１１に示すような開発の問題点を解決するものとして，開発技法・方法，管理技法・方法論，開発作業標準，開発の基本理念が相互に関連して支えていることに因る（図４－１２）。ソフトウェア開発の問題点個々の重要性の比重は，技術的，社会的背景により除々に変っているため，効果的な開発方法の模索は常に意識しながら総合的，長期的に改善してゆくことが重要である。

例えば，現在ソフトウェアの保守費用はますます増大し，トータルコ

ストの中に占める比率が非常に高くなってきている。この様な状況の中で初期開発のみの開発効率を向上させてもその経済効果は薄い。むしろ保守費が増大する原因を分析し，保守性に重点を置いた設計技法，あるいはドキュメント記述法等の改善をはからねばならない。また，ソフトウェアの品質をみた場合に製造不良に因るものより，設計不良に因るものにウエイトがあるならば，より投資効率のよい工程の改善に力点を移して行かねばならない。あるいは，外部のソフトウェア会社のレベルが高まり自社開発より品質が高く，総合的なコストダウンが図れるならば，外注の積極的利用を考える等々があげられる。

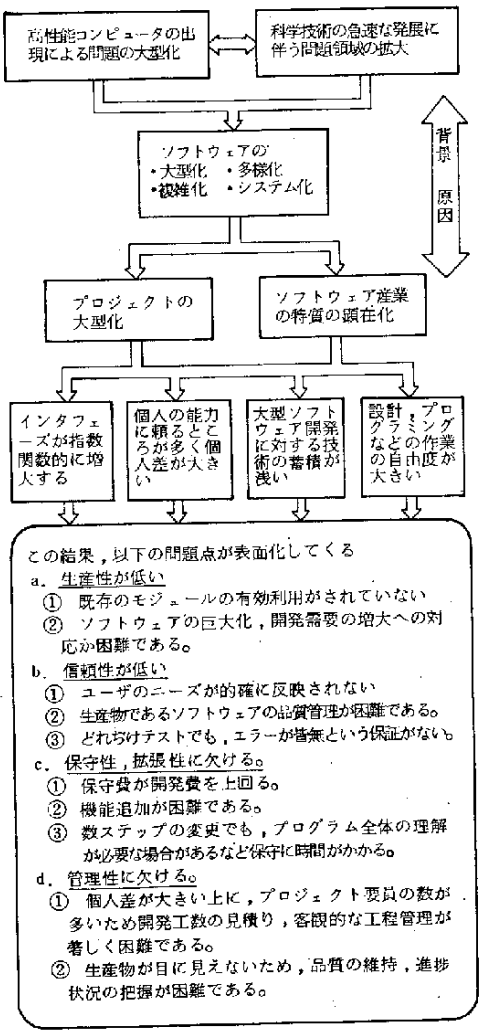


図４－１１　ソフトウェア開発の問題点の背景と原因

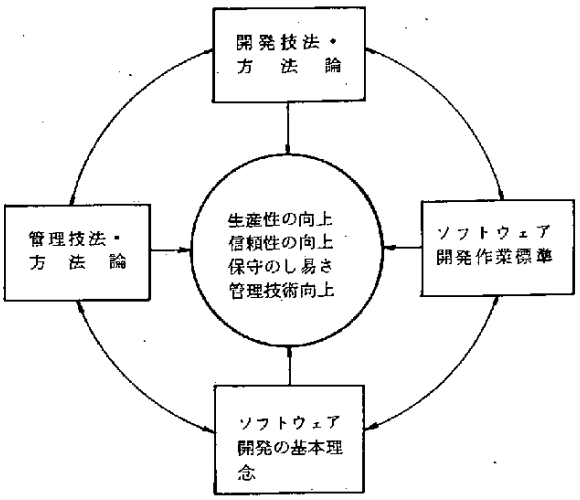


図４－１２　開発方法論の位置づけの例

4.3.4　開発ツールの評価

ソフトウェア開発ツールとしては，メーカやソフトウェア会社などから提供されている汎用パッケージや自社内開発の各種ツールがある。これらは開発の種々の段階で適用されるが，特に新しく導入されたツールについては，プロジェクト終了時等に評価し，それらが使われるであろう後発プロジェクトへ反映してゆく必要がある。

開発ツールの評価には２つの観点が考えられる。１つは開発ツールが目的としている生産性向上の効果に対する評価であり，いま１つは開発ツール自体の性能や機能に対する評価である。前者についてはツールの導入の検討の中で期待される効果として提示され，また試行的な利用の中で評価された基準や目標として示されよう。現実のプロジェクトでは必ずしもある特定の開発ツールの導入による効果か否か明白には握され得ないこと，あるいは開発メンバの習熟度にもよるため，ある一定の期間を設定し評価することとなろう。

また，後者については，開発中に実際に利用してみた後の問題点や効果

などを利用メンバの意見や収集データからまとめ後発プロジェクトへのアドバイスとすると共に，必要ならば改造，再教育などのアクションをとることが必要である。また，ツールのマニュアルへの反映は，即座に行われなければならない。

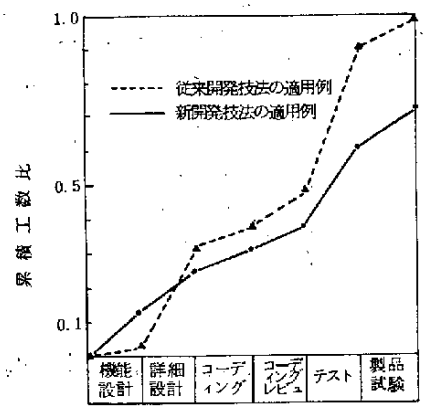


図４－１３. 生産技法の適用例

表４－１５　構造化言語の効果例

比較項目			比 (SIMPLE/SYSL)
プログラム特性	バグ件数		0.78
	ソースプログラム規模	宣言文	1.47
		実行文	0.82
		小計	0.95
作業時間	コーディング完了までの時間	仕様書の理解，GFの作成	0.79
		コーディング	1.05
		小計	0.90
	デバッグ時間	結果の確認，バグ原因の究明	0.71
		修正方法の検討	1.23
		修正箇所のコーディング	0.74
		小計	0.74
	合計		0.87

表4－16　開発ツールの評価例

技法／ツール	適用工程	効　　果	問題点／留意事項
汎用モジュールテスト支援ツール	MD ～ OT	・ドライバ／スタブの生成不要 ・TSSデバック可能（いつでも使える）	・テスト仕様書の作成が難しい ・再テスト時に一々手で入力しなければならない→バッチテスト要望

5. プロジェクト管理ツール

5.1 概　　要

ソフトウェアの開発に関するプロジェクト管理のツールは，ソフトウェアの開発を支援するものと密接な関連がある場合が多く，開発支援技法がプロジェクト管理のツールとして利用されている例がみられる。

プロジェクト管理ツールとして製品化されているものは少く，その利用も極めて限られており，技法や文献がよく利活用されている。

ここでは，現時点でプロジェクト管理のツールとしてメーカ，ソフトウェア・ハウス等から提供されているものとして，PMP（Project Management Program），PCMP（Project Cost Management Program）についてヒアリングを実施したので，これを中心にその体系と機能を述べるとともに，PRIDE（Profitable Information by Design through Phased planning and Control）などソフトウェアの開発支援技法について，プロジェクト管理の観点からその内容の紹介と展望を行う。

5.2 プロジェクト管理ツールの利用の現状

今回実施した「プロジェクト管理に関する実態調査」から，プロジェクト管理に関するツールの利用状況をみると，次のとおり，ほとんど利用されていない現状である。

表5－1　プロジェクト管理ツールの利用

利用ツールの内容	件　数	比　率
見積り計算用ツール	4	2.9(%)
日程計画立案のツール	1	0.7
予算管理のツール	4	3.9
工程管理のツール	17	14.3

また，これらツールに起因するトラブルもほとんどなく，①費用見積りあるいは費用管理のためのツールに問題があり，予算をオーバしたプロジェクト（2件，1.9％），②工程管理のためのツールに問題があり，工期が遅れたプロジェクト（1件，0.8％）といったものは稀なケースとなっている。

実態調査において，プロジェクト管理のツールとして回答があったものをみると，次のとおりである。

（メーカ，ソフトウェア・ハウスの提供するツール）

① ROT−11（Routine for Online Testing−11）（日本ユニバック）

② NUPS（Nippon UNIVAC Planning System）（日本ユニバック）

③ TPNS（Teleprocessing Network Simulater）（日本アイ・ビー・エム）

④ SDEM（Software Development Enineering Methodology）（富士通）

⑤ STEPS（Standardized Technology and Engineering for Programming Support）（日本電気）

⑥ MUDS（M series Unit Debug Simulater）（日立製作所）

⑦ PRIDE（Profitable information by Design through Phased Planning Control）（日本システミックス）

⑧ SPADE（Source Program And Document Editors）（三井情報開発）

⑨ T−CAT（Test Coverage Analyzing Tool）（ソフトウェア・リサーチ・アソシエイツ）

（自社開発のツール）

① プロジェクト別工数把握システム

② 開発進度管理システム

③ 管理システム（プロジェクト管理用計画の作成，実施，評価）

④ SPORTY（作業進捗管理，工数集計）

⑤ PSK （プログラム進捗管理）

⑥ EDIT01（ＴＳＳによるプログラム，データ等の作成・変更作業の管理）

このように，メーカ，ソフトウェア・ハウス提供のツールには，プロジェクト管理ツールというよりもソフトウェアの開発支援技法が，プロジェクト管理に利用されており，自社開発のツールとして，独自にプロジェクト管理ツールを作成，利用している現状である。

プロジェクト管理の今後の課題のうち，ツールに関するものをみると，次のとおり，将来この分野に重点を置いて，ツール及び技法を開発，整備しようとするものが多い。

① 工程管理，障害管理，変更管理，ドキュメント管理を統一データベース化し，全作業のシステム化を図る。

② 進捗状況，生産性のは握等のプロジェクト管理をサポートするためのツール及び管理データを自動収集するためのシステム環境を作る。

③ プロジェクト管理用のツール（パソコンなどを利用した安くて簡単なもの）を開発し，利用したい。

④ プロジェクト管理支援ツールの採用

⑤ 全社レベルでのプロジェクト管理方式の確立

以上に見てきたように，プロジェクト管理のツールの利用は，非常に少く，メーカ，ソフトウェア・ハウス提供のソフトウェアの開発支援技法を利用するほか，自社で独自に開発したツールを特定のユーザが利用しているに過ぎないというのが実態で，今後この分野に重点を置いて，ツールの開発，整備を図るという状況である。

5.3 プロジェクト管理のツール

5.3.1 PMP（Project Management Program）

PMPは，協同システム開発株式会社の委託を受け，(株)メルコム・オキタックシステムズが開発したソフトウェア開発プロジェクト向けの工程管理支援ツールである。

その対象範囲は，プロジェクトの計画段階から，実行，完了に至るまでの工程管理を包含し，プロジェクトをネットワークの形で管理し，管理者に必要な情報をユーザの実情に合わせ，Visible に提供することができる。

すなわち，計画段階では，経営者及びプロジェクト・マネージャ等がそのプロジェクト開発計画の適性を判断するのに必要な報告書類の作成を行い，実行段階では，工程の的確な状況は握ができる作業状況報告書や，予定工数／実績工数，予定計算機使用時間／実績計算機使用時間を表示した報告書を作成し，完了段階では，開発プロジェクト結果を総合的に判断できるプロジェクト効率報告書及び費用評価報告書を作成する。

(1) 機能の概要

PMPは，基本処理，資源割当処理，ネットワーク準備処理，費用評価処理の4つの機能を有している。その概要は，次のとおりである。

表5－2　PMPの機能

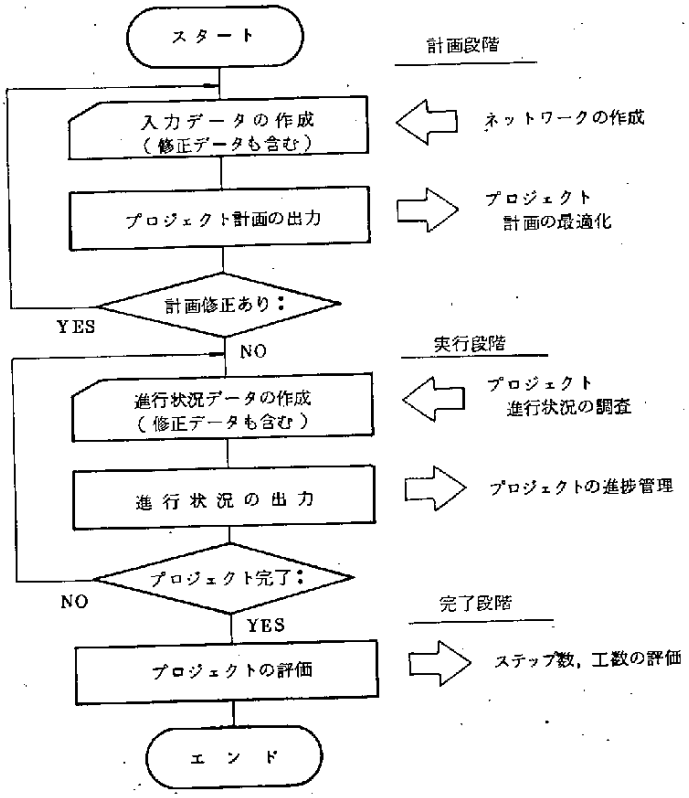
機　能	内　容
基本処理	PMPの全体を管理するとともに，ネットワークの作成，時間分析（作業の開始日，完了日，余裕等の計算），ファイル管理などを行う。
資源割当処理	ソフトウェア開発に必要な人員，計算機費用等をムダなく有効に利用できるように，プロジェクトをスケジュールする。また複数のプロジェクトについても資源の割当てが可能である。
ネットワーク準備処理	サブネットのライブラリ登録，ネットワーク図表の作成を行う。

費　　用 評 価 処 理	プロジェクトの各工程に必要な経費を，部門別に集計してプロジェクト全体の見積り費用や実際費用を計算する。

－メルコム・オキタックシステムズ社作成資料－

(2) 利　用　方　法

ＰＭＰの利用手順は，次のとおり，初期の計画段階，作業に入る実行段階，プロジェクト完了後のプロジェクトの評価の３段階に分けることができる。



－メルコム・オキタックシステムズ社作成資料－

図５－１　ＰＭＰ利用の流れ

計画段階の手順としては，①ネットワークの作成，②共通データの作成，③ネットワーク・データの作成，④PMPの実行とスケジュール評価の4つの作業がある。

① ネットワークの作成

ネットワークの作成にあたっては，プロジェクトの作業工程からアクティビティを決定し，アクティビティ間の前後関係を結んで，1つのサブネットを作成する。次にサブネットを結んで1つのネットワークを作成する。

その事例を示すと，図5－2のとおりである。

PMPでは，サブネットは500まで，アクティビティは5,000までという制限がある。

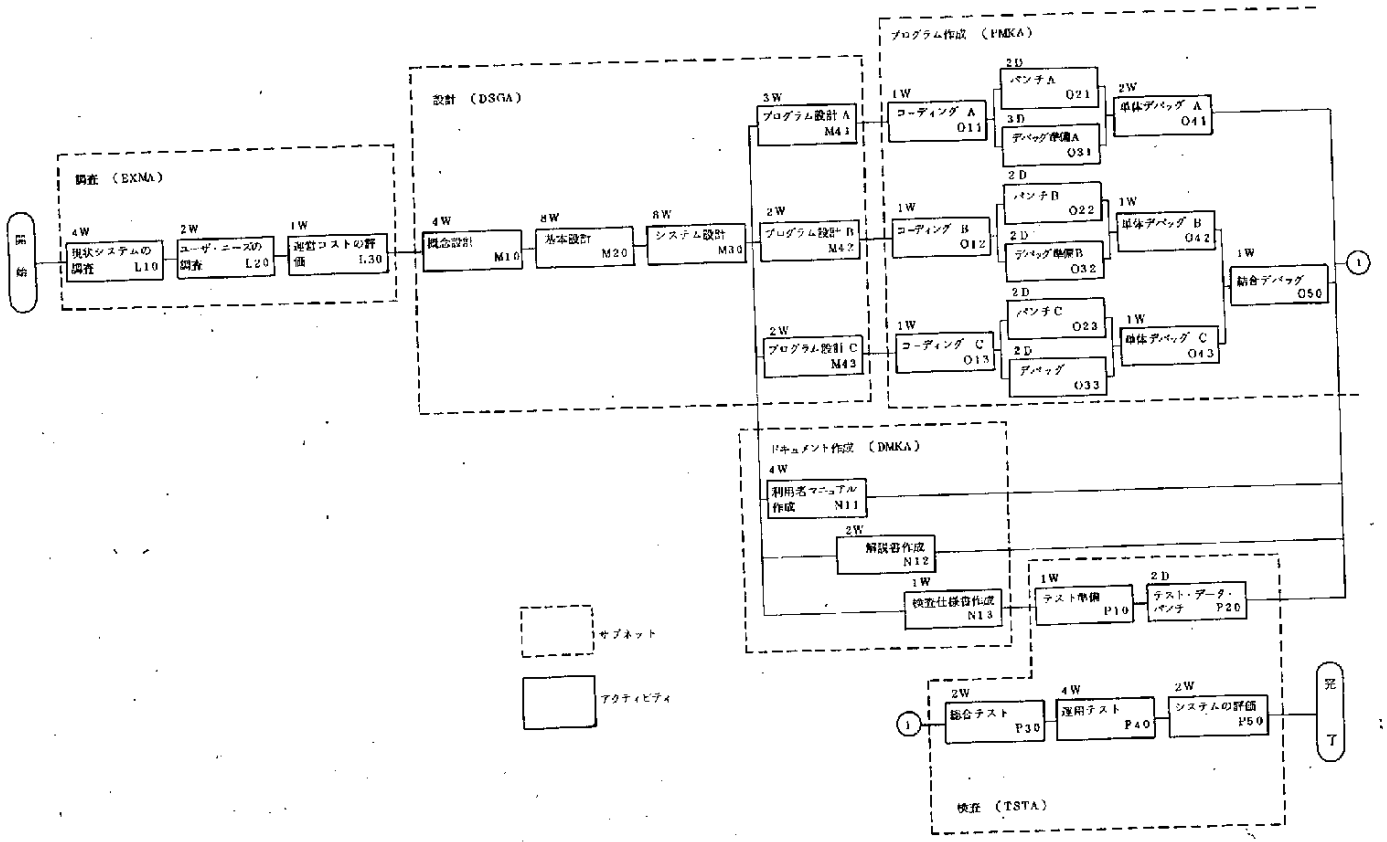
② 共通データの作成

共通データは，プロジェクトの作業時間，休日，利用する資源，作成する報告書などを指定するものである。それらは，週間作業日指定カード，カレンダカード，資源定義カード，資源グループ定義カード，報告書定義カード，報告書形成カードとしてマスタファイルの共通領域に入力される。

③ ネットワーク・データの作成

ネットワーク・データは，ネットワークの詳細情報を明示するもので，各アクティビティに対して作業に必要な時間，資源，休日等を設定し，その後アクティビティやサブネットの前後関係を指定する。

その内容は，ネットワーク定義カード，組織表定義カード，サブネット定義カード，境界条件定義カード，アクティビティ定義カード，アクティビティ継続カード，前後関係定義カード，サブアクティビティ定義カードとして入力される。



―メルコム・オキタックシステムズ社作成資料 ―

図5－2　プロジェクトのネットワーク図

また，特に重要な作業を行うアクティビティは，マイルストーンと呼ばれ,プロジェクト全体の主要な管理目標となるが，作業の重要度に応じた管理を行うようになっている。

④　PMPの実行とスケジュール評価

共通データやネットワーク・データをもとに，ネットワークの構造をチェックして，ネットワークの時間分析，資源山積み処理を行い，それらの報告書を出力する。報告書の内容は，ネットワークを構成するサブネットのスケジュール一覧表，マイルストーン報告書，アクティビティの集計報告書，資源利用報告書，ネットワーク図表などである。（表５－３参照）

こうして作成されたプロジェクト計画について，プロジェクト・マネージャ，作業担当者が全工程にわたり，チェックを行う。

実行段階の手順としては，①進行状況データの作成，②進捗管理の作業がある。

①　進行状況データの作成

個々のアクティビティ毎に各作業担当者から，進行状況の報告を受け，これを入力する。アクティビティが現在進行中であれば，実際の作業開始日や作業達成率，完了までの見積り時間などが指定され，完了した場合は，完了日付が指定される。（表５－４参照）

②　進捗管理

進行状況データをもとにして，計画とのズレを検討する。ズレが生じた場合は，できるだけ余裕のあるアクティビティに負荷をかけ，ズレの影響をなくすようにするが，クリティカルパスにズレが生じた場合は，まだ実施されていない作業との関連を考慮して修正計画をたてる。

完了段階の作業は，プロジェクトの評価である。PMPは，開発コストの評価という観点から，①プロジェクトの効率報告書と②作業工

程，計算機使用の予定／実績報告書を出力することができる。

① プロジェクトの効率報告書

この報告書では，開発されるプログラムのステップ数，作業工数，計算機使用時間，ステップ数／工数，ステップ数／計算機時間，計算機時間／工数などが出力される。（表５－５参照）

② 作業工程，計算機使用の予定／実績報告書

この報告書では，月別，工程別に工数や計算機時間の予定値，実績値が出力され，将来のソフトウェア開発の工数見積りの資料とすることができる。（表５－６，５－７参照）

(3) 対象機種及びオペレーティング・システム

ＰＭＰは，次のシステムで利用が可能である。

表５－８　対象計算機システム

対　象　機　能	対　象　ＯＳ
HITAC M−160Ⅱ	VOS 2
HITAC M−180	VOS 3
FACOM M−160 M−180ⅡAD	OS Ⅳ／F4
ACOS 77−700	ACOS 6
COSMO−700Ⅱ	UTS／VS

−メルコム・オキタックシステムズ社作成資料−

表5－3　スケジュール報告書

ラン デート　01FEB79

＊＊＊　スケジュール　キウコクショ　＊＊＊

ネットワーク　PROJ オウヨウ ソフト プログラム カイハツ　(テスト ケース 2)
ショヨウ シカン ダンイ　ヒ
ソート コウモク　サイソウ カイシ

アクティビティ コード	アクティビティ セツメイ	ショヨウ シカン	CW	P R	サイソウ カイシ	サイソウ カンリョウ	サイテ カイシ	サイテ カンリョウ	ヒマ ヨユウ	シュウヨウ ヨユウ	スラック ヒダリ
L10	ケンキュウ システム ノ テイギ	20	11	C	28MAR77	22APR77	28MAR77	22APR77	0	0	
L20	ユーザー ニーズ ノ テイギ	10	11	C	25APR77	11MAY77	25APR77	11MAY77	0	0	
L30	ウンエイ コスト ノ テイギ	5	11	C	13MAY77	19MAY77	13MAY77	19MAY77	0	0	
M10	カイハツ セツケイ	20	11	C	19MAY77	15JUN77	19MAY77	15JUN77	0	0	S1 19MAY77
M20	キホン セツケイ	40	11	C	16JUN77	10AUG77	16JUN77	10AUG77	0	0	
M30	システム セツケイ	40	11	C	11AUG77	07OCT77	11AUG77	07OCT77	0	0	
N11	リヨウシャ マニュアル サクセイ	20	11		11OCT77	08NOV77	28OCT77	28NOV77	13	13	
M41	プログラム セツケイ A	15	11		11OCT77	31OCT77	11OCT77	31OCT77	0	0	F1 31OCT77
M42	プログラム セツケイ B	10	11		11OCT77	24OCT77	19OCT77	01NOV77	6	0	
M43	プログラム セツケイ C	10	11		11OCT77	24OCT77	19OCT77	01NOV77	6	0	
N12	トウニュウ カイセツショ サクセイ	10	11		18OCT77	31OCT77	14NOV77	28NOV77	18	18	
N13	ウンテン ショクショ サクセイ	10	11		25OCT77	08NOV77	02NOV77	16NOV77	6	0	
O12	コーディング B	5	11		25OCT77	31OCT77	02NOV77	09NOV77	6	0	
O13	コーディング C	5	11		25OCT77	31OCT77	02NOV77	09NOV77	6	0	
O11	コーディング A	5	11		01NOV77	08NOV77	01NOV77	08NOV77	0	0	
O22	パンチ B	2	11		01NOV77	02NOV77	10NOV77	11NOV77	6	0	
O23	パンチ C	2	11		01NOV77	02NOV77	10NOV77	11NOV77	6	0	
O32	デバッグ シュウセイ B	2	11		01NOV77	02NOV77	10NOV77	11NOV77	6	0	
O33	デバッグ シュウセイ C	2	11		01NOV77	02NOV77	10NOV77	11NOV77	6	0	
O42	タンタイ デバッグ B	5	11		04NOV77	10NOV77	14NOV77	18NOV77	6	0	
O43	タンタイ デバッグ C	5	11		04NOV77	10NOV77	14NOV77	18NOV77	6	0	
O21	パンチ A	2	11		09NOV77	10NOV77	10NOV77	11NOV77	1	1	
O31	デバッグ シュウセイ A	3	11		09NOV77	11NOV77	09NOV77	11NOV77	0	0	
P10	テスト シュウセイ	5	11		09NOV77	15NOV77	17NOV77	24NOV77	6	0	
O50	ウワコウ デバッグ	5	11		11NOV77	17NOV77	21NOV77	28NOV77	6	6	S1 11NOV77
O41	タンタイ デバッグ A	10	11		14NOV77	28NOV77	14NOV77	28NOV77	0	0	
P20	テスト データ パンチ	2	11		16NOV77	17NOV77	25NOV77	28NOV77	6	6	
P30	ソウコウ テスト	10	11		29NOV77	12DEC77	29NOV77	12DEC77	0	0	S1 29NOV77
P40	ウンヨウ テスト	20	11		13DEC77	13JAN78	13DEC77	13JAN78	0	0	
P50	システム ノ ヒョウカ	10	11		17JAN78	30JAN78	17JAN78	30JAN78	0	0	

―メルコム・オキタックシステムズ 社作成資料―

表5－4　作業状況報告書

ﾗﾝ ﾃﾞｰﾄ　01FEB79

＊＊＊　ｻｷﾞｮｳ　ｼﾞｮｳｷｮｳ　ﾎｳｺｸｼｮ　＊＊＊

ﾈｯﾄﾜｰｸ　　PROJ ｷｮｳﾖｳ ｿﾌﾄ ﾌﾟﾛｸﾞﾗﾑ ｶｲﾊﾂ　(ﾃｽﾄ ｹｰｽ 2)
ｼｮﾖｳ ｼﾞｶﾝ ﾀﾝｲ　H
ｿｰﾄ ｺｳﾓｸ　　ｹｲｿｳ ｶｲｼ

ｱｸﾃｨﾋﾞﾃｨ ｺｰﾄﾞ	ｱｸﾃｨﾋﾞﾃｨ ﾒｲｼｮｳ	ｼｮﾖｳ CW ｼﾞｶﾝ	P R	ｹｲｿｳ ｶｲｼ	ｹｲｿｳ ｶﾝﾘｮｳ	ﾃﾞｰﾀ ﾋｿﾞｳ	ﾉｺﾘ ｼﾞｶﾝ	W.D (%)
L10	ﾗﾝｼﾞｮｳ ｼｽﾃﾑ ﾉ ﾁｮｳｻ	20 11	C	28MAR77	22APR77		0	100
L20	ﾕｰｻﾞｰ ﾆｰｽﾞ ﾉ ﾁｮｳｻ	10 11	C	25APR77	11MAY77		0	100
L30	ｳﾝｴｲ ｺｽﾄ ﾉ ﾁｮｳｻ	5 11	C	13MAY77	19MAY77		0	100
M10	ｶｲｲﾝ ｾｯｹｲ	20 11	C	19MAY77	15JUN77		0	100
M20	ｷｶｲ ｾｯｹｲ	40 11	C	16JUN77	10AUG77		0	100
M30	ｼｽﾃﾑ ｾｯｹｲ	40 11	C	11AUG77	07OCT77		0	100
N11	ﾘﾖｳｼｬ ﾏﾆｭｱﾙ ｻｸｾｲ	20 11		11OCT77	08NOV77			0
M41	ﾌﾟﾛｸﾞﾗﾑ ｾｯｹｲ A	15 11		11OCT77	31OCT77			0
M42	ﾌﾟﾛｸﾞﾗﾑ ｾｯｹｲ B	10 11		11OCT77	24OCT77			0
M43	ﾌﾟﾛｸﾞﾗﾑ ｾｯｹｲ C	10 11		11OCT77	24OCT77			0
N12	ﾄｳﾆｭｳ ｶｲｾﾂｼｮ ｻｸｾｲ	10 11		18OCT77	31OCT77			0
N13	ｹﾝｻ ｼｮｳｼｮ ｻｸｾｲ	10 11		25OCT77	08NOV77			0
012	ｺｰﾃﾞｨﾝｸﾞ B	5 11		25OCT77	31OCT77			0
013	ｺｰﾃﾞｨﾝｸﾞ C	5 11		25OCT77	31OCT77			0
011	ｺｰﾃﾞｨﾝｸﾞ A	5 11		01NOV77	08NOV77			0
022	ﾊﾟﾝﾁ B	2 11		01NOV77	02NOV77			0
023	ﾊﾟﾝﾁ C	2 11		01NOV77	02NOV77			0
032	ﾃﾞﾊﾞｯｸﾞ ｼｭｳﾘｮｳ B	2 11		01NOV77	02NOV77			0
033	ﾃﾞﾊﾞｯｸﾞ ｼｭｳﾘｮｳ C	2 11		01NOV77	02NOV77			0
042	ﾀﾝﾀｲ ﾃﾞﾊﾞｯｸﾞ B	5 11		04NOV77	10NOV77			0
043	ﾀﾝﾀｲ ﾃﾞﾊﾞｯｸﾞ C	5 11		04NOV77	10NOV77			0
021	ﾊﾟﾝﾁ A	2 11		09NOV77	10NOV77			0
031	ﾃﾞﾊﾞｯｸﾞ ｼｭｳﾘｮｳ A	3 11		09NOV77	11NOV77			0
P10	ﾃｽﾄ ｼｭｳﾘｮｳ	5 11		09NOV77	15NOV77			0
D50	ｸﾘｺﾐ ﾃﾞﾊﾞｯｸﾞ	5 11		11NOV77	17NOV77			0
041	ﾀﾝﾀｲ ﾃﾞﾊﾞｯｸﾞ A	10 11		14NOV77	28NOV77			0
P20	ﾃｽﾄ ﾃﾞｰﾀ ﾊﾟﾝﾁ	2 11		16NOV77	17NOV77			0
P30	ｿｳｺｰｳ ﾃｽﾄ	10 11		29NOV77	12DEC77			0
P40	ｳﾝﾖｳ ﾃｽﾄ	20 11		13DEC77	13JAN78			0
P50	ｼｽﾃﾑ ﾉ ﾋｮｳｶ	10 11		17JAN78	30JAN78			0

－メルコム・オキタックシステムズ社作成資料－

表５－５　プロジェクト効率報告書

＊＊＊　ﾌﾟﾛｼﾞｪｸﾄ ｺｳﾘﾂ ﾎｳｺｸｼｮ　＊＊＊

ﾈｯﾄﾜｰｸ　PRO2　ｵｳﾖｳ ｿﾌﾄ ﾌﾟﾛｸﾞﾗﾑ ｶｲﾊﾂ　(ﾃｽﾄ ｹｰｽ 2)
ｻﾌﾞﾈｯﾄ　DMKA　DSGA　EXMA　PMKA　TSTA
ｼｽﾃﾑ　MELC　ANA1

ｻﾌﾞﾈｯﾄ	ｻﾌﾞﾈｯﾄ ﾉ ｾﾂﾒｲ	ﾌﾟﾛｸﾞﾗﾑ ｽﾃｯﾌﾟ	ｷｶｲﾖｳ ｺｽｳ	ｹｲｶﾝﾄ ｼｭｳﾘﾖｳ	ｽﾃｯﾌﾟ / ｺｽｳ	ｽﾃｯﾌﾟ / ｹｲｶﾝﾄ	ｹｲｶﾝﾄ / ｺｽｳ
DMKA	ﾄｰｷｭﾒﾝﾄ ｻｸｾｲ	0	30	0	0.0	0.0	0.00
DSGA	ｼｽﾃﾑ ﾉ ｾｯｹｲ	5710	25	0	228.4	0.0	0.00
EXMA	ｼｮｳｶﾝ ﾌﾟﾛｸﾞﾗﾑ	4250	15	0	283.3	0.0	0.00
TSTA	ｼｽﾃﾑ ﾉ ﾃｽﾄ	0	0	390	0.0	0.0	0.00
	ｹｲ	9960	70	390	142.3	25.5	5.56

表５－６　月別工数報告書

＊＊＊　ﾂｷﾍﾞﾂ ｺｳｽｳ ﾎｳｺｸｼｮ　＊＊＊

ﾈｯﾄﾜｰｸ　PRO2　ｵｳﾖｳ ｿﾌﾄ ﾌﾟﾛｸﾞﾗﾑ ｶｲﾊﾂ　(ﾃｽﾄ ｹｰｽ 2)
ｻﾌﾞﾈｯﾄ　DMKA　DSGA　EXMA　PMKA　TSTA
ｼｽﾃﾑ　ANA1　ｼｽﾃﾑ ｱﾅﾘｽﾄ-1

ｷｶﾝ	ﾖﾃｲ	ｼﾞｯｾｷ	ﾖｼﾝﾘｮｳ
MAR77	4	4	100
APR77	20	11	55
MAY77	15	9	60
JUN77	33	16	48
JUL77	42	0	0
AUG77	46	0	0
SEP77	40	0	0
OCT77	30	20	67
NOV77	17	10	59
DEC77	21	0	0
JAN78	17	0	0
ｹｲ	285	70	25

表５－７　月別計算機報告書

＊＊＊　ﾂｷﾍﾞﾂ ｹｲｻﾝｷ ﾎｳｺｸｼｮ　＊＊＊

ﾈｯﾄﾜｰｸ　PRO2　ｵｳﾖｳ ｿﾌﾄ ﾌﾟﾛｸﾞﾗﾑ ｶｲﾊﾂ　(ﾃｽﾄ ｹｰｽ 2)
ｻﾌﾞﾈｯﾄ　DMKA　DSGA　EXMA　PMKA　TSTA
ｼｽﾃﾑ　MELC　MELCOM-COSMO

ｷｶﾝ	ﾖﾃｲ	ｼﾞｯｾｷ	ﾖｼﾝﾘｮｳ
NOV77	20	40	200
DEC77	285	350	123
JAN78	0	0	0
ｹｲ	305	390	128

－メルコム・オキタックシステムズ社作成資料－

5.3.2 ＰＣＭＰ（Project Cost Managemet Program）

ＰＣＭＰは，情報処理振興事業協会の委託を受け，日本電子開発株式会社が開発した原価管理を中心とするプロジェクトの計画・管理のツールである。

その特色は，プロジェクトの原価管理を，原価計算機能，見積り支援機能，要員管理支援機能としてとらえ，これをプロジェクト群管理，階層管理，多次元管理の考え方から総合的に管理する点にある。

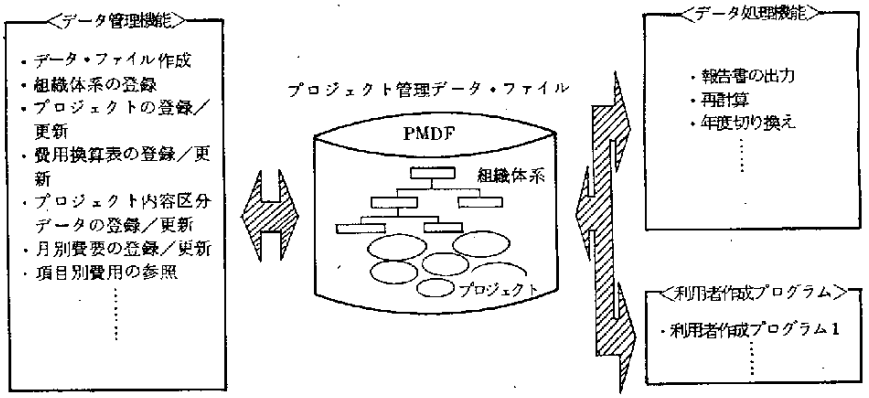
プロジェクト群管理は，プロジェクト相互の関連をは握し，プロジェクト相互間を共存的に管理しようという考え方であり，階層管理は，部長，課長，プロジェクト・リーダといったいくつかの管理レベルで，各レベルに適合した計画，管理をしようという考え方である。また，多次元管理とは，日程，費用，要員等の複数管理要素によって，目的に応じた多面的な計画，管理を行うという考え方である。

こうした考え方のもとに，ＰＣＭＰはデータを一元管理するとともに，ユーザの作成によるプログラムの利活用など拡張性をもたせている。

(1) 機 能 の 概 要

ＰＣＭＰの機能は，図５－３のとおり，プロジェクト管理データ・ファイルのもとに，次のような機能を有している。

① 導入処理機能
② 総合的なデータ管理機能
③ 画面との対話形式によるデータ入出力機能
④ プロジェクト別原価計算の即時計算機能
⑤ 部門別原価計算機能
⑥ 単位変更に伴う再計算機能
⑦ 報告書出力機能
⑧ 費用の年度切り替え機能
⑨ そ の 他

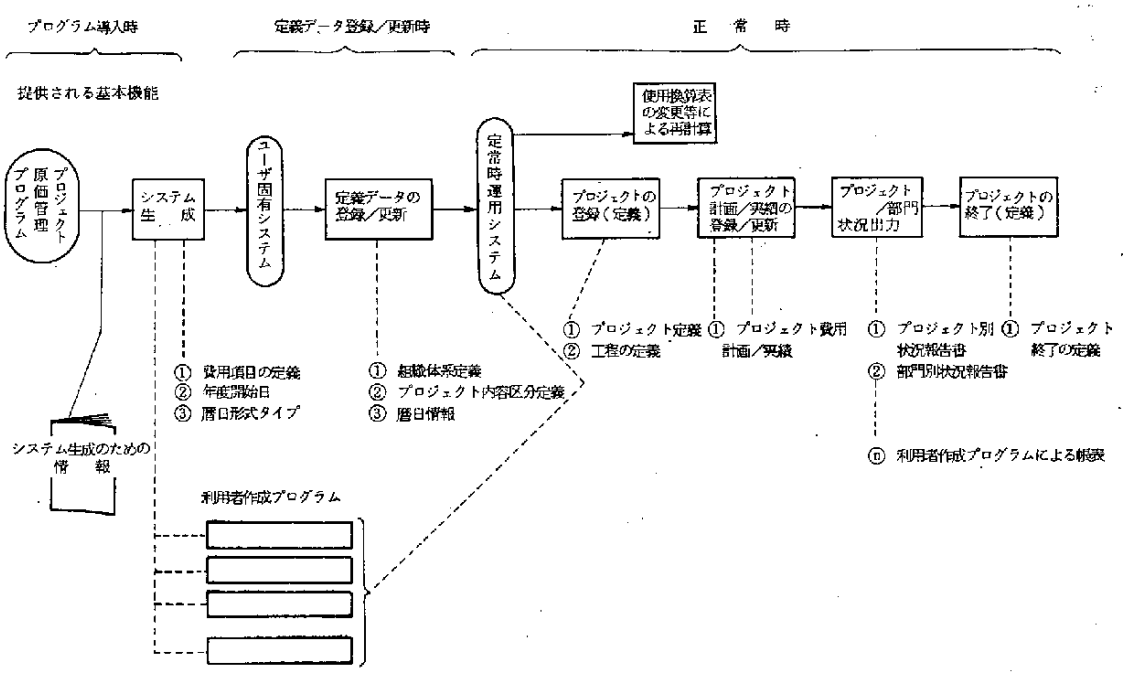


－日本電子開発社作成資料－

図５－３　ＰＣＭＰの機能の概要

(2) 利　用　方　法

ＰＣＭＰの利用の流れは，図５－４に示すとおりである。



－日本電子開発社作成資料－

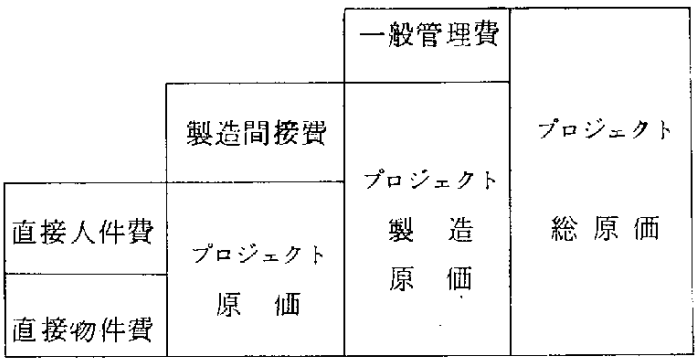
図５－４　ＰＣＭＰの運用関連図

プログラム導入時は，導入時に決まる基本的な条件，例えば費用項目，年度開始日，暦日形式タイプなどを与え，ユーザの原価管理システムへ組み込みを行う。

定義データの登録／更新時は，変更の少ない組織体系，プロジェクトの内容区分，暦日といった共通データを登録する。

定常時は，プロジェクトの登録，計画／実績の登録／更新，管理を行うものである。

プログラム導入時における費用項目の定義にあたっては，プロジェクト原価を次のようにとらえ，ユーザがそれぞれ費用項目を設定する。



— 日本電子開発社作成資料 —

図５－５　プロジェクト原価のとらえ方

定常時には，①プロジェクトの定義／更新，②プロジェクト工程定義／更新，③項目別費用，④月別費用（プロジェクト別，部門別），⑤部門状況報告書，⑥プロジェクト別状況報告書，⑦プロジェクト終了定義について画面出力が可能である。

このうち，プロジェクト別状況報告書及び部門状況報告書の出力例を示すと，表５－９，表５－１０，表５－１１のとおりである。

これらは，共通ルーチンとして与えられたものであるが，ユーザが独

自に利用プログラムを作成し，例えば，①プロジェクト内容区分コード別の原価及び売上分析，②プロジェクト要員の検索，③見積り基準値の作成等を行うことができる。

(3) 対象機種及びオペレーティング・システム

ＰＣＭＰの稼動可能な機種及びＯＳは，次のとおりである。

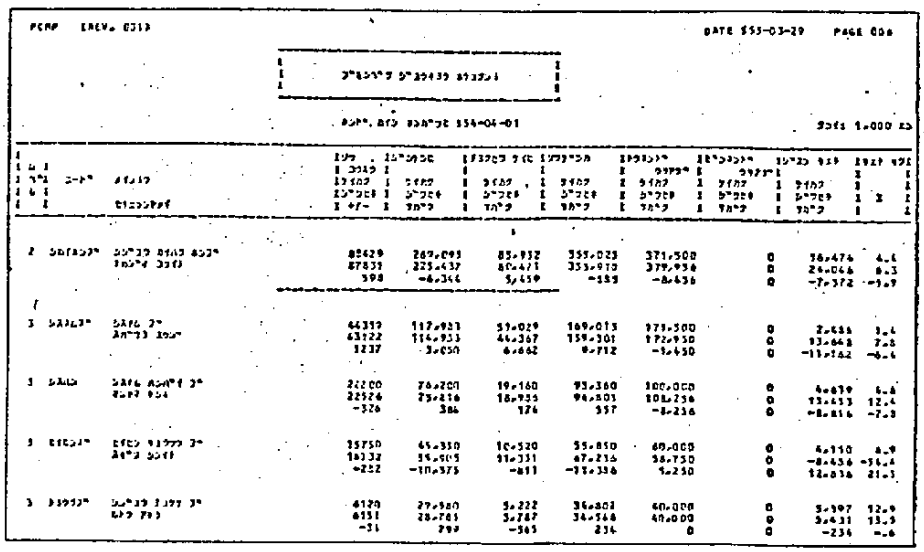
・ＮＥＡＣシステム１００モデル４０，８０

・ＮＥＡＣシステム１５０

・ＯＳ ＩＴＯＳ－４

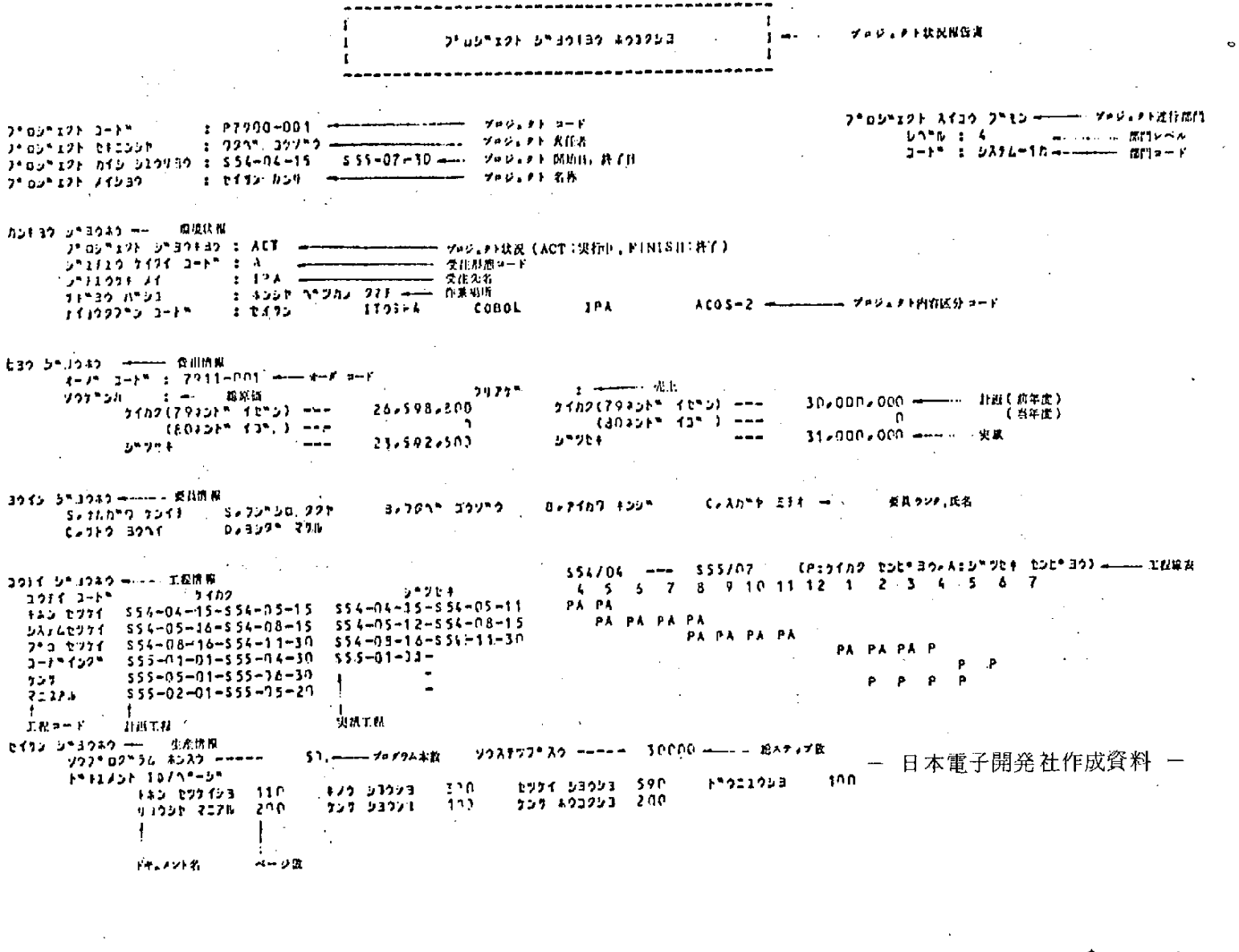
なお，ＰＣＭＰはCOBOL及び共通はサブルーチン仕様などを採用しているため，上記以外の他機種への変換も比較的容易である。

表５－１１　部門別状況報告書



－日本電子開発社作成資料－

表５－９　プロジェクト別状況報告書



― 日本電子開発社作成資料 ―

表５－１０　プロジェクト別状況報告書

*** ﾌﾟﾛｼﾞｪｸﾄ ｼﾞｮｳｷｮｳ ﾎｳｺｸｼｮ *** ——— プロジェクト別状況

ﾌﾟﾛｼﾞｪｸﾄ ｺｰﾄﾞ　：　P7900-001 ——— プロジェクト コード
ﾌﾟﾛｼﾞｪｸﾄ ﾋﾄｺﾝﾀｸﾄ：ﾌｸﾀﾞ ｺｳｿﾞｳ ——— プロジェクト 責任者
ﾌﾟﾛｼﾞｪｸﾄ ﾒｲｼｮｳ　：ｾｲｻﾝ ｶﾝﾘ ——— プロジェクト 名称
ﾎﾝﾄﾞ ｶｲｼ ﾈﾝｶﾞｯﾋﾟ：7904-01 ——— 年度開始月

ﾋｮｳｼﾞｭﾝｺｰﾄﾞ		79- 4	79- 5	79- 6	79- 7	79- 8	79- 9	79-10	79-11	79-12	80- 1	80- 2	80- 3	ﾄｳｼﾞﾝﾄﾞｳｲ ｺﾞｳｹｲ	ﾋﾞﾝｼﾞﾝﾄﾞｳｲ ｺﾞｳｹｲ
ｼﾞﾝｼﾞﾝﾋ	ｹｲｶｸ	2726000	2043000	2043000	1536000	1686000	1329000	1329000	1839000		1951000	1951000	1951000	20534000	
	ｼﾞｯｾｷ	2730000	2213000	2309000	1758000	1584000	1228000	1323000	1553000		1831000	2138000		18676000	
ｲﾝｼｮｳﾋﾝｶｳﾋ	ｹｲｶｸ	370800	285000	285000	250000	250000	215000	215000	265000		302200	302200	302200	3036400	
	ｼﾞｯｾｷ	358000	311600	337600	262800	228800	197600	214600	224400		282600	334000		2751800	
ｴｲﾀﾞﾝｼｭｳﾋ	ｹｲｶｸ	108500	79500	79500	65500	65500	51500	51500	74500		76400	76400	76400	802200	
	ｼﾞｯｾｷ	106400	86300	92000	58300	61600	47600	51300	60600		71700	83700		729500	
ｼｭｾﾝｼｭｳﾋ	ｹｲｶｸ														
	ｼﾞｯｾｷ														
ｺｳｺｸﾋ	ｹｲｶｸ	2200				1000			2000		6000	10000	10000	31200	
	ｼﾞｯｾｷ	1500				500			600		4000	9000		15600	
ｱﾙﾊﾞｲﾄﾋ	ｹｲｶｸ														
	ｼﾞｯｾｷ														
ｶｰﾄﾞ ﾘﾖｳﾋ	ｹｲｶｸ										200000	200000	150000	550000	
	ｼﾞｯｾｷ										180000	220000		400000	
ｶﾞｲｼﾞｭｸﾋ	ｹｲｶｸ														
	ｼﾞｯｾｷ														
ｼｮｸｲ ｲﾝｼｮｸﾋ	ｹｲｶｸ	2000	2000	2000	2000	15000	2000	2000	25000		10000	20000	5000	87000	
	ｼﾞｯｾｷ	1000	2500	1600	3000	20000	1500	3000	31000		5000	24000		92600	
ｼｶﾞｲ ｲﾝｼｮｸﾋ	ｹｲｶｸ		5000			24000			36000					65000	
	ｼﾞｯｾｷ		3000			14000			30000					47000	
ｿﾉﾀ	ｹｲｶｸ	1000	2000	1000	1000	3000	1000	1000	3000		380000	500000	600000	1493000	
	ｼﾞｯｾｷ	1000	5000			3000	1000		2000		350000	520000		880000	
ｼｭｳﾆｭｳ ｺﾞｳｹｲ	ｹｲｶｸ	3210500	2416500	2410500	1914503	2044500	1598503	1592500	2241500		2923600	3057600	3092600	26598300	
ｹ	ｼﾞｯｾｷ	3197900	2624400	2739000	2172100	1911900	1475700	1591900	1906600		2724300	3328700		23592500	
ｾｲｻﾝ ｺﾞｳｹｲ	ｹｲｶｸ					15000000							15000000	30000000	
ｹ	ｼﾞｯｾｷ					15500000							15500000	31000000	

— 日本電子開発社作成資料

5.4 ソフトウェア開発支援技法

実態調査においてみられるとおり，ソフトウェアの開発支援技法の中のプロジェクト管理機能を利用している事例がある。

ここでは，開発支援技法の主要なものについて，プロジェクト管理の機能に関し，概要の紹介を行う。

5.4.1 PRIDE

（Profitable Information by Design through Phased Planning and Control）

PRIDEは，アメリカのMBA社（Milt Bryce and Associates Inc.）が開発し，株式会式日本システミックスが販売，サポートを行っている情報システムの設計の方法論である。

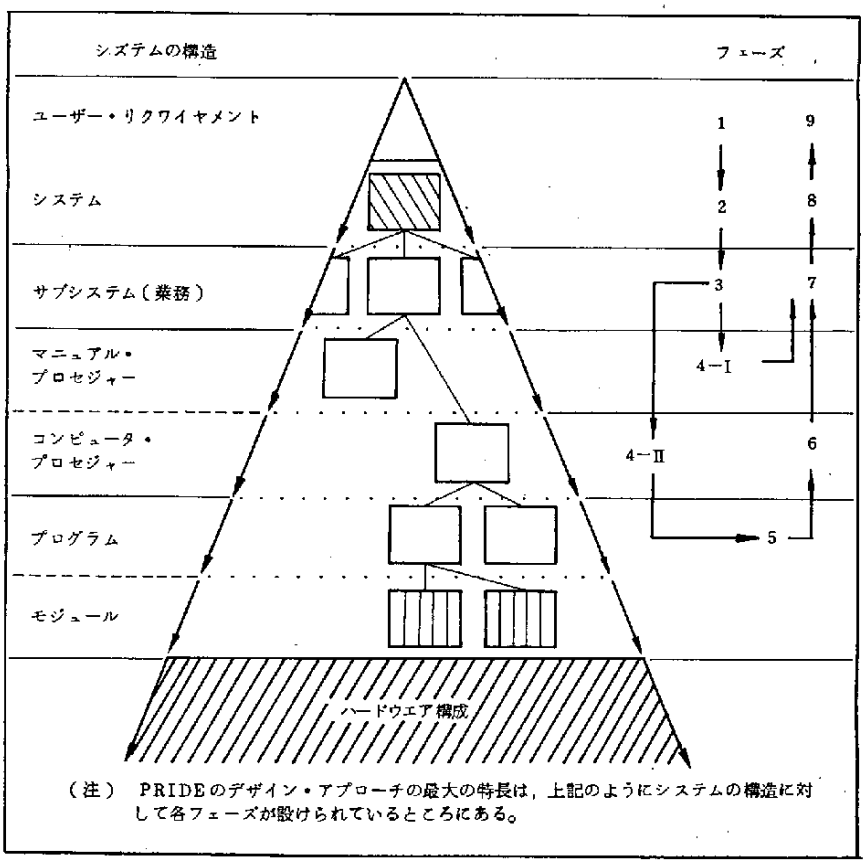
情報システムの設計の方法論ではあるが，PRIDEは，次のような技法の体系を有しており，プロジェクト管理の技法をとり込んでいる。

① システム構造化設計技法
② データ管理技法
③ プロジェクト管理技法
④ ドキュメンテーション技法
⑤ コミュニケーション技法

すなわち，プロジェクト管理は，システムの構造化設計とデータ管理をベースとして，仕事量の見積り，日程のスケジューリング，進捗管理・調整などをこれら技法と関連づけて，三位一体で管理を行う技法である。

(1) システム構造化のアプローチ

PRIDEでは，システムの構造を図5－6のとおり定義し，これをもとに図5－7のとおり，9つのフェーズに分けて設計を行うようになっている。



－日本システミックス社作成資料－

図５－６　システムの構造とフェーズ

(2) データ管理

PRIDEでは，システムの構造を設計すると，これにデータの関係づけを行う。

データとシステムの関係づけは，図５－８のとおりで，それぞれどこで，どのようにデータが使われているかを管理する。このように相互の関係づけを管理しておくと，いずれかのコンポーネントに変更があって

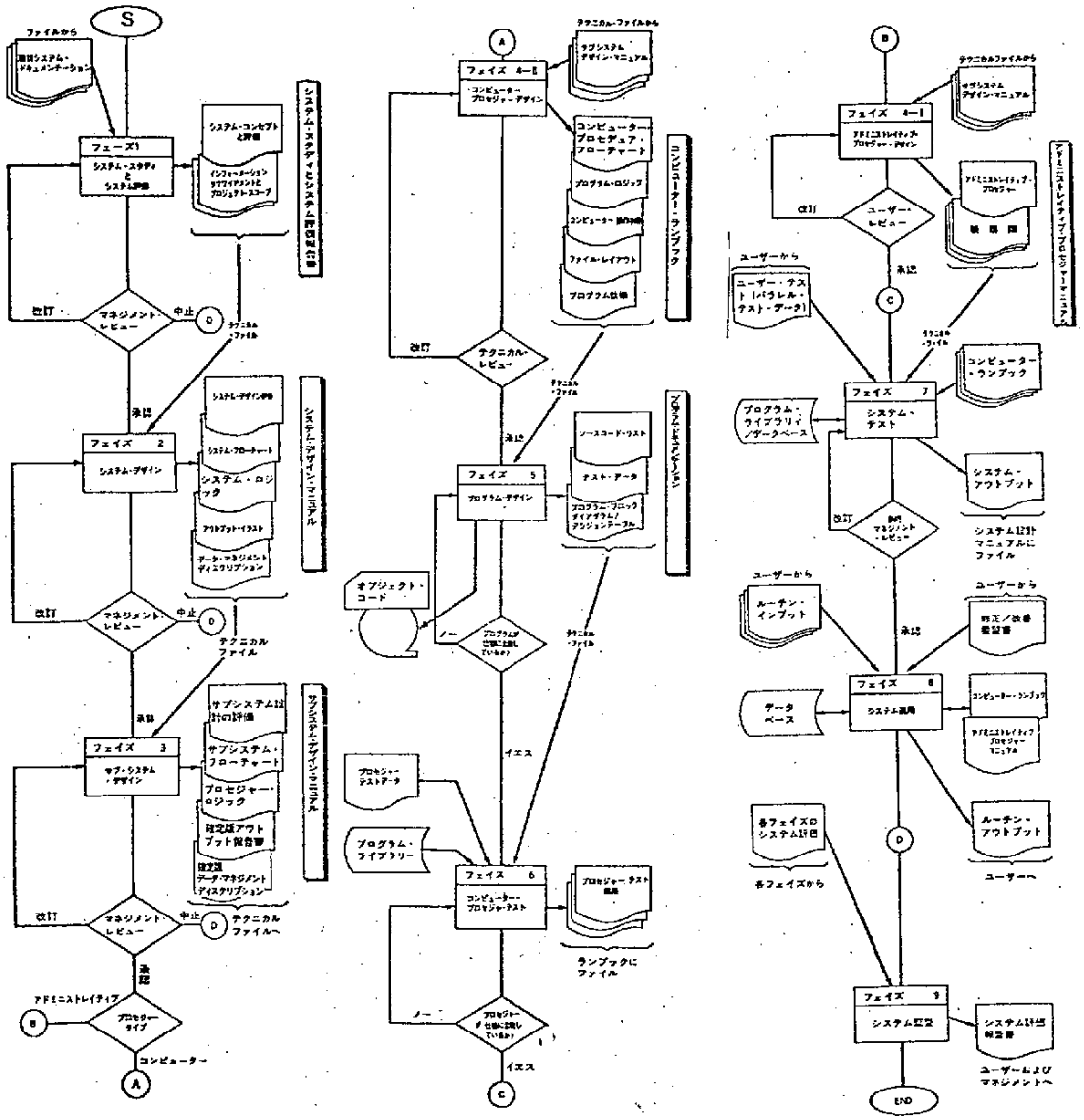
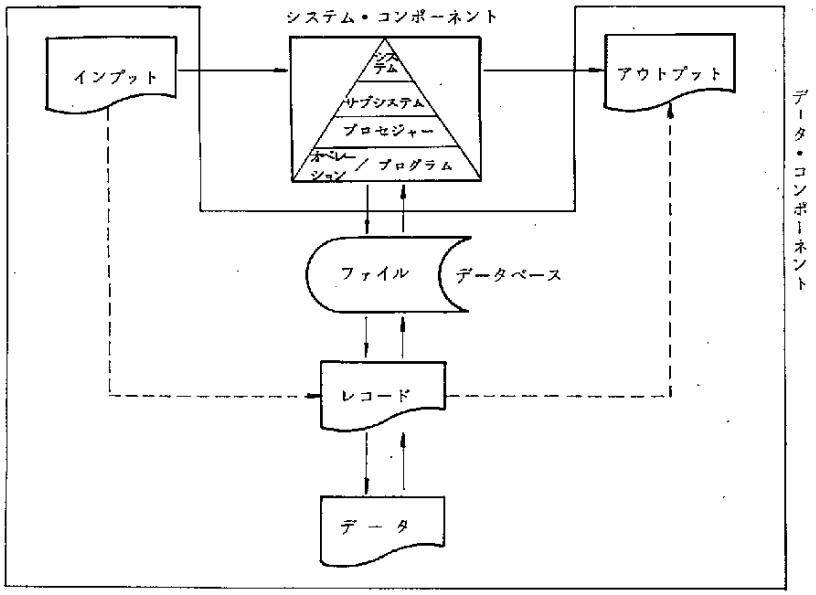


図5－7　PRIDEのフェーズ・アプローチ

－日本システミックス社作成資料－



－日本システミックス社作成資料－

図５－８　システム・コンポーネントとデータ・コンポーネントの関係づけ

も，その影響範囲と度合いを知ることができる。

(3)　プロジェクト管理

PRIDEでは，システム構造を明確に定義しているので，フェーズ・アプローチに示されているように，どの時点で何を行うかが明らかにされている。したがって，管理対象となる設計・開発作業に必要なリソースと，その適用の仕方が規定されている。

PRIDEのプロジェクト管理は，次の４つのアクティビティによって構成されている。

①　見　積　り

②　スケジューリング

③　実績報告

④　コントロール

見積りでは，プロジェクト全体の作業量及びコストと，次のフェーズの作業量及びコストをそれぞれ分けて見積る。この時，作業量を延べ時間でなく，直接投入時間で見積り，正味の作業量をは握する。見積りの結果から，おおよその投入人員と納期を設定し，延べ日数をは握して日程計画を作成する。これがスケジューリングである。見積りとスケジューリングによって計画が完成し，人員，日程，コストが明らかになる。

実績報告は，プロジェクトの実行結果を報告するもので，これにもとづき，問題をは握して適切なアクションを取るのがコントロールである。

このように，システムの構造をユーザのリフワイアメントにもとづきシステムーサブ・システムープロセジャ・プログラムーモジュールというように定義し，これを構築するために，フェーズに従ってどのようにアプローチするかを詳細に決めることになり，各フェーズでの作業を詳細に分割する。この方法によって作業に必要な職種別の工数，コストの見積りが可能となり，この作業の見積りと実行結果を対比することにより，プロジェクトの管理が容易になる。この結果を利用することにより，更に作業標準時間・コスト標準の見積り精度を高めることができ，新しい改善手順を作り出す契機となる。

PRIDEのプロジェクト管理は，要約すれば，ユーザがプロジェクトのの開発作業を見積り，これを管理するために，システムの構造をタスク・レベルまで詳細化する方法論であると言えよう。

5.4.2 HIPACE — SPDS

(High Productive Application Creation and Engineering —Standard Procedure to Develop System)

HIPACE−SPDSは，日立製作所が開発したシステムの計画・設計・開発及びこれらの作業の管理に関する標準手順である。

SPDSは，①システム開発の工程に沿った手順，②プロジェクト管理手

順，③システム管理手順から構成されており，プロジェクト管理手順は，システム開発の作業がSPDSが提供される手順に従って行われることを前提として作成されている。この意味では，先に述べたPRIDEと同じ考え方に立つ技法である。

SPDSのプロジェクト管理手順は，プロジェクトの計画，実施，評価の各段階を網羅しており，方法論として次のような方法をとっている。

① フェーズド・アプローチによるプロジェクトの推進

フェーズド・アプローチとは，プロジェクトを管理可能ないくつかの単位（フェーズ及びステップ）に分割し，その単位毎に作業計画と作業レビュを着実に実施しようとするものである。

② 計画プロジェクトと開発プロジェクトの分離

システム計画作業とシステム開発作業を各々別プロジェクトとして実施することにより，プロジェクト計画自体の充実を図るとともにシステム計画の作業が完了しないうちに，システム開発作業が開始されることがないように配慮している。

③ プロジェクト・イニシェーション業務の定型化

SPDSでは，プロジェクト発足の準備作業をプロジェクト・イニシェーションと呼び，その手順を定型化し，トップ・マネジメントの有効な意思決定，メンバに対する主旨の徹底，ユーザや関係者に対する協力事項の明確化を図ることにしている。

このような方法論をとることにより，プロジェクトの計画段階の充実を図ることに重点が置かれている。

(1) プロジェクト計画

SPDSにおけるプロジェクト管理手順においては，プロジェクト計画段階を重視していることは既に述べたとおりであるが，そのための方法論として次のような方法論をとっている。

①プロジェクト作業の構造的は握（WBS＝Work Breakdown

Structure）

②フェーズド・プロジェクト・プランニング

③資源見積り

④フェーズ別プロジェクト組織と責任／権限の明確化

その利用手順は，次のとおりである。

① プロジェクト作業の構造的は握

WBSとは，プロジェクトを構成する作業を詳細化して，階層構造として表現したものである。

SPDSでは，WBSを概略WBS，プロジェクトWBS，フェーズWBS，ステップWBSの4つのレベルに分け，上位WBSから下位WBSへと詳細化を行うとともに相互のチェックを行う。

次にWBSが作成されると，これを作業アクティビティ・ネットワークとして作業の実施順序を表現する。これらの作業にあたり，SPDSでは，WBS表（表5－12参照）と作業アクティビティ・ネットワーク図（図5－9参照）がワークシートとして用意されている。

② フェーズド・プロジェクト・プランニング

フェーズド・プロジェクト・プランニングとは，プロジェクトの計画を段階的に実施していく方法で，プロジェクトをいくつかの単位（フェーズ）に分割する。

その方法は，プロジェクトをいくつかのフェーズに分け，初期の全体見積りを立て，それをフェーズ毎に配分する。プロジェクトの進行に沿って，各フェーズの作業開始前にそのフェーズの見積りを再度行う。予じめ，フェーズ毎に配分されている当初の見積りを超過する場合はその旨異常報告を出し，必要であれば計画変更を行う。このようにプロジェクトの資源の見積りを何回か繰り返すことにより精度向上を図ることができる。

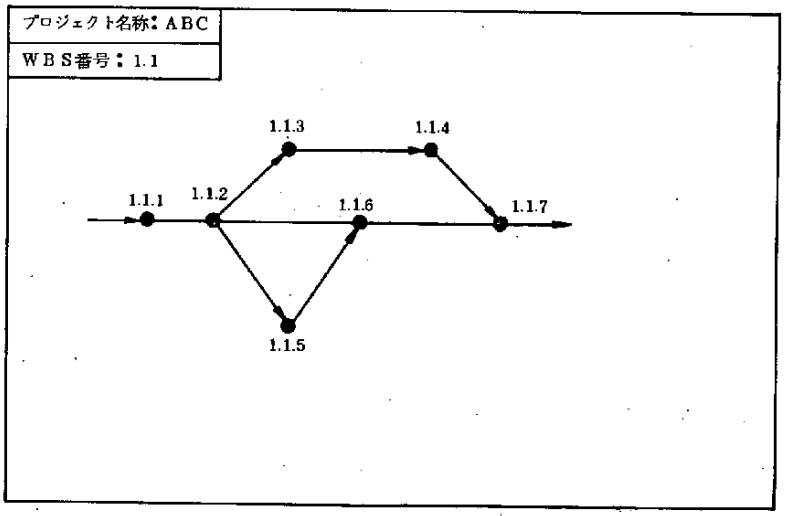
表５－１２　ＷＢＳ表

項番	最終成果物	大項目	中項目	小項目	所要日数	所要職能 (人) (人口) 所要工数	備考

ＷＢＳ番号	作業名	繰返し指定	繰返し決定要因
1.1.1	ユーザの定義	I	――
1.2.1	ユーザニーズの洗出し	M	関連ユーザ数

I：くり返しなし
M：複数くり返される

―日立製作所作成資料―



―日立製作所作成資料―

図５－９　作業アクティビティ・ネットワーク図

③ 資源見積り

プロジェクトの資源としては，工数，マシン時間，設備，技術の4つがあり，表5－13のとおり各計画段階でより詳細に見積りを行う。

このうち，工数の見積りの方法として，表5－14のとおり，3通りの方法を使用する。

表5－13 資源見積りの段階

計画段階	資源見積のための入力情報	見積るべき資源				
		工数		マシン時間	設備	技術
		人	時間			
プロジェクトイニシエーション	・プロジェクト成果定義 ・概略WBS	・ユーザや関係者を含めた全体の概略人員	・概略WBSによる大日程	・概略マシン時間総量	・プロジェクト実施に必要な固定設備	・想定される問題点の指摘
プロジェクト計画	・最終成果物定義 ・プロジェクトWBS ・フェーズ別組織体制	・最終成果物とWBSから算出した職能別総所要工数（人日） ・総所要工数のフェーズ別配分		・総開発量から算出したマシン時間量 ・マシン時間量のフェーズ別配分	・可変設備についてフェーズ別の所要量	・必要となる技術と必要となる時期
フェーズイニシエーション	・フェーズWBS	・フェーズWBSによるフェーズ内総所要日数（人・日） ・フェーズ総所要工数のステップ別配分 ・ステップ別要員の割合て		・フェーズ別マシン時間所要量 ・ステップ別マシン時間配分	・フェーズ別可変設備所要量 ・ステップ別可変設備割付	――
ステップイニシエーション	・フェーズ作業スケジュール	・作業別要員割当て	・所要別作業所要時間	・作業別マシン時間所要量	・作業別設備割付	――

―日立製作所作成資料―

表５－１４　見積り方式

タイプ	見積方式	次元	見積のためのインプット	使用段階
Ⅰ	プロジェクトの特性から標準値により見積る	人・月	概略ＷＢＳ プロジェクト成果定義書	プロジェクト イニシェーション
Ⅱ	最終成果物から開発作業量を見積り，標準値により工数を求める。	人・日	最終成果物定義書 プロジェクトＷＢＳ	プロジェクト計画
Ⅲ	詳細なＷＢＳに，メンバを割当て，所要日数を求め積み上げる。	作業者数×日数	フェーズＷＢＳ ステップＷＢＳ	フェーズ イニシェーション ステップ イニシェーション

－日立製作所作成資料－

④　フェーズ別プロジェクト組織と責任／権限の明確化

ＳＰＤＳでは，フェーズ別，ステップ別に組織を計画し，これに対応したＲＭＣ（Responsibility Matrix Chart）を作成し，責任と権限を明確化することにしている。

プロジェクト組織の基本形は，次のとおりである。

- プロジェクト・マネージャ
- プロジェクト管理スタッフ
- プロジェクト会議
- グループ・リーダ（フェーズの責任者）
- チーム・リーダ（ステップ単位の責任者）

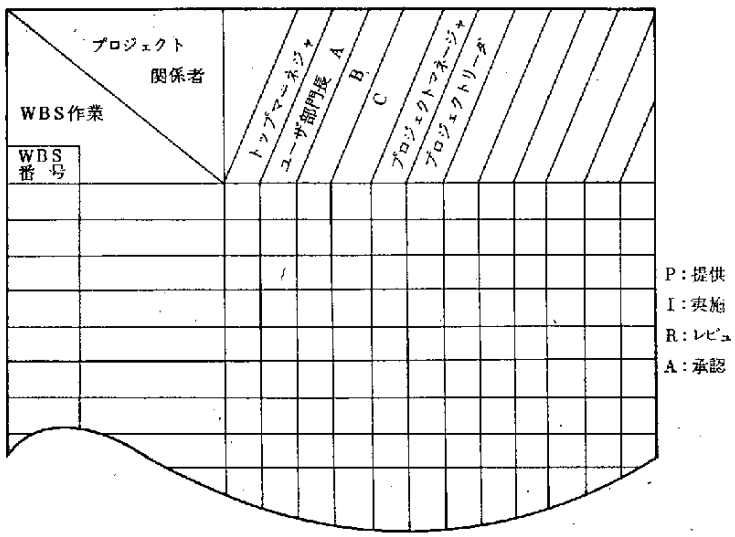
ＲＭＣのワーク・シートは，表５－１５のとおりである。

(2)　プロジェクトの管理計画

プロジェクトを計画どおりに進行させるために，プロジェクトを運営管理するための仕掛や体制，管理基準をあらかじめ計画する必要がある。

ＳＰＤＳでは，管理の仕掛として，コントロール・ルームの計画，プロジェクト・ファイルの計画，意志決定期間の計画を，また管理基準として，マイルストーン計画，レビュ計画を作成することとしている。

表5－15　　ＲＭＣ



－日立製作所作成資料－

プロジェクトの進行状況を管理するために，マイルストーン・チャート，作業アクティビティ・ネットワーク，ＡＴ（Available Time），プロジェクト・コスト／パフォーマンス，成果物構成管理（成果物の構成と属性）をそれぞれディスプレイで示し，これらをもとに意思決定やレビュを行うこととしている。

レビュ計画は，プロジェクト・レビュ，マイルストーン・レビュ，完了レビュの３つから成り，マネージャやグループ・リーダが実施することとなっている。

(3)　プロジェクトの運営管理

プロジェクトの計画が完了すると，プロジェクトの実施が行われるが，これを適切に管理する必要がある。

ＳＰＤＳでは，プロジェクト全体に関する管理作業を，プロジェクトの評価と制御と呼び，個別作業の管理をイニシエーション／ターミネーシ

ョンと呼んでいる。

プロジェクトの評価が，プロジェクトの完了段階だけでなく，実施段階で行われ対策を講じるというのが大きな特色である。

すなわち，イニシェーション／ターミネーションで発見された異常が，プロジェクトの評価と制御の作業により対策を指示することとなっている。

プロジェクトの評価にあたっては，プロジェクト・レビュ，マイルストーン・レビュ，完了レビュを通じて行われ，ＳＰＤＳが標準的に用意している管理レポートとしては，表５－１６のとおりである。

表５－16　管理レポート

項番	管理レポート名称	目　　的	主たる内容
1	プロジェクト状況管理表	・全ての作業の進捗状況，消費資源の計画／実績を一元管理する。	・ＷＢＳ別／週別，完成度，累積工数，累積レビュ回数
2	成果物別状況管理表	・引渡し可能な成果物毎に，プロジェクトをは握する。	・製品別プロジェクトとスケジュール，工数，ステップ数，メモリ・サイズ，マシン時間の予実績を示す。
3	マイルストーン管理表	・スケジュールの管理 ・マイルストーンの管理	・ＷＢＳ別に着手日，完了日とマイルストーンを図示する。
4	成果物構成管理表	・成果物の構成を管理し，変更制御	・成果物の各要素の関連
5	可処分残余資源管理表	・使用可能な残余資源の管理	・残余の工数，日数，マシン時間を示す。
6	コスト／パフォーマンス管理表	・目標達成率と資源消費の傾向と管理	・左記をグラフ表示する。
7	問題点管理表	・発生した問題点のフォロー状況を管理	・問題点発生の累積と未解決残をグラフ表示

－日立製作所作成資料－

プロジェクトの制御は，異常や問題の解決を図りながら個々の作業を統制したり，計画自体の修正を行うことをいうが，主として，成果物全

体の整合性や成果物全体の要求達成度の観点から行うこれらの活動は，コンフィギュレーション管理と呼ばれているものである。

(4) プロジェクト・ターミネーション

プロジェクトの完了に伴い実施する作業をプロジェクト・ターミネーションと呼び，プロジェクト完了レビュ，最終成果物の引渡し，プロジェクト完了報告が行われる。

プロジェクトの完了レビュは，フェーズ毎の完了レビュが既に実施段階で行われているので，最終的に成果を引渡すことが可能かどうかという点から行われる。

このようにＳＰＤＳのプロジェクト管理手順は，計画と実施段階ではとんどの作業が行われ，プロジェクト・ターミネーションでは最終チェックが行われるという手順になっている。

5.4.3 ＳＤＥＭ

（Software Development Engineering Methodology）

ＳＤＥＭは，富士通株式会社が開発したソフトウェアの開発及びプロジェクト管理のための開発作業標準である。

マニュアルとしては，次のものが用意されている。

① FACOM SDEM 概説書
② FACOM SDEM ソフトウェア開発作業標準ハンドブック
③ FACOM SDEM ソフトウェア開発作業標準解説書
④ FACOM SDEM 導入と適用の手引

ＳＤＥＭは，ソフトウェアの開発が中心となっているが，表５－１７のとおり，システム環境（運用・移行，ハードウェア，設備，障害対策），ソフトウェア開発（入出力，業務処理，ファイル，テスト），管理（技術管理・教育・訓練・プロジェクト管理）の３つの局面について，実施すべき作業の標準を設定している。

すなわち，ソフトウェアの開発作業とプロジェクトの管理が不即不離の関係で進められているという手順になっている。

プロジェクト管理に即していえば，プロジェクトの計画で，効果，資源工数，体制，条件，進捗管理方法を明らかにし，ソフトウェアの設計，プログラミング，テスティングの段階で，進捗管理，予実管理，作業環境整備，変更管理等が行われ，保守・システム評価の段階でプロジェクトの評価が行われる。

そのために，作業の項目と作業の内容，これに必要なドキュメントが用意されている。

とくに，システム設計後，プログラム設計後，テスト終了後それぞれレビュが設けられ，次の段階へ進むべきか否かの判断決定が行われるようになっている点が特色である。

ドキュメントとしては，プロジェクト完了報告書にその記載内容及び例が示されており，ＳＤＥＭの作業工程に従って，計画と実績の対比が可能となるように示されている。

プロジェクトの評価に結びつく，開発実績については，次のような図表の例が用意されている。

① プロジェクト開発実績総括表
② 工程計画／実績線表
③ プログラム生産物，ドキュメント生産物
④ 工程別工数計画実績比較表
⑤ 業務別（サブ・システム別）工数計画実績表
⑥ 工数（計画－実績）対比図
⑦ 月間計算機使用実績表
⑧ 工程別計算機使用実績表
⑨ 資材経費表
⑩ 開発指標

表５－１７　工程区分別開発作業標準

工程区分			システム環境	ソフトウェア開発	管理（とくにプロジェクト管理）
大工程	中工程	小工程			
計画（PN）	調査立案（SP）	調査立案（SP）	動向・技術調査	ユーザ要件分析 新システム構想立案	開発計画案 プロジェクト計画工程へ進むか否かの決定
	プロジェクト計画（PP）	プロジェクト実行計画（BP）	ハードウェア構成の概略設定等 設備計画	概念設計	投資効果分析 開発計画（概略） 資源計画（概略） 開発可否の決定
	プロジェクト計画（PP）	プロジェクト実行計画（DP）	移行計画 運用計画等 設備実行計画	データ量のは握 ファイル量のは握等	推進体制と責任分担の明確化 開発環境条件の設定 進捗管理方法の設定 開発及び資源実行計画（内外注区分を含む） プロジェクト・チームの編成 仕様凍結手続き／変更手続の制定
設計（DN）	システム設計（SD）	初期設計（ID）	運用設計 移行設計 設備計画実施等	入出力項目の決定 コード設計 サブ・システムへの分割 マスタ・ファイルのデータ項目の決定書	進捗管理，予実管理，作業環境整備，変更管理 外注要員の手配
	システム設計（SD）	論理設計（LD）	運用管理システムの設計 移行システムの設計 設備計画実施〔継続〕	入出力形式の決定 システム方式の初期設計 プログラムの分割 データ論理設計等	進捗管理，予実管理，作業環境整備，変更管理 外注要員の手配
	システム設計レビュ（SDR）	システム設計レビュ（SDR）	使い易さの検討	製品目標を実現する方法の確認 テスト計画，ツールなどの見直し	実行計画の見直しとプログラム設計のための体制の確立 プログラム設計工程へ進むか否かの決定
	プログラム設計（PD）	プログラム構造設計（PS）	運用管理プログラムの設計 移行に必要なプログラムの設計 テストマシンの確保 設備計画実施〔継続〕	システム方式の詳細設計 モジュールへの分割 データ物理設計 結合テスト仕様設計 テスト実施計画等	進捗管理，予実管理，作業環境整備，変更管理 消耗品，媒体手配 実行計画の見直し モジュール単位の詳細スケジュールの作成 外注契約手続

工程区分			システム環境	ソフトウェア開発	管理（とくにプロジェクト管理）
大工程	中工程	小工程			
設計（DN）	プログラム設計（PD）	モジュール設計（MD）	運用管理プログラムの設計〔継続〕 移行に必要なプログラムの設計〔継続〕 ハードウェアの保守体制の確立 ハードウェアの搬入及び現場調整 設備計画実施〔継続〕等	モジュール内処理手順詳細設計 モジュール・テスト仕様設計 結合テスト仕様設計〔継続〕 テスト実施計画〔継続〕	進捗管理，予実管理，作業環境整備，変更管理
	プログラム設計レビュ（PDR）	プログラム設計レビュ（PDR）		システム設計書とプログラム設計書の整合性のチェック テスト仕様書と統計書の整合性のチェック	実行計画の見直しと体制の検討 プログラミング工程へ進むか否かの決定
プログラミング（PG）	プログラミング（PG）	プログラミング（PG）	運用管理プログラムの作成 移行プログラムの作成 ハードウェアの搬入及び現場調整	コーディング コンパイル/アセンブル ファイルの初期設定 テスト・データの作成 モジュール・テスト等	進捗管理（テスト状況調査と問題点の摘出） 予実管理（工数/マシン使用実績管理，消耗品/カード・パンチ管理など） 作業管理（設計書の維持，管理，仕様変更の全員への徹底など） 労務管理（勤務形態申請，超勤限度管理） 作業目標管理（テスト内容，回数，要員勤務ローテーション，計算機時間割当） プログラム・エラー対策
テスティング（TG）	結合テスト（IT）	結合テスト（IT）	ハードウェア・モニター，シュミレータなどの設置	結合テスト	進捗管理，予実管理，作業環境整備，変更管理 労務管理〔継続〕 作業日程管理〔継続〕 プログラム・エラー対策〔継続〕
	テスティング・レビュ（TGR）	テスティング・レビュ（TGR）		ME工程で評価するために収集すべきデータ項目及び収集方法 製品の品質最終チェック及び評価	運用テスト開始決定の判断

工程区分			システム環境	ソフトウェア開発	管理（とくにプロジェクト管理）
大工程	中工程	小工程			
運用テスト（OT）	運用テスト（OT）	運用テスト（OT）	併行作業の立会い 移行作業等	運用テスト	新システムへの切替え日時の最終決定
保守システム評価（ME）	保守システム評価（NE）	保守システム評価（ME）	保守	評価用データの収集 製品目標達成度の長期にわたる評価	プロジェクト評価（本稼動後1年内に行う，短期評価，工程の途中で打ち切られたプロジェクトの評価も行う）

－富士通資料より作成－

⑪ テスト指標

- テスト指標一覧表
- エラー原因分類表
- 週間エラー検出件数推移表
- エラー修正工数（工程別）

⑫ 開発作業評価

- 全般的評価
- 新技法／ツールの導入評価

⑬ 業務評価

⑭ ハードウェア／ソフトウェア評価

⑮ プロジェクト・メンバのプロフィル及び所感

また，導入と適用の手引では，ＳＤＥＭの標準に加えユーザのシステムの特性を盛り込むことの必要性を認めるとともに，開発量（規模）の見積り，工程及び工数計画の立案，計画の詳細化，進捗管理のポイント，生産性指標，品質指標等の方法論，事例を紹介している。

第Ⅱ部　プロジェクト組織の編成と要員管理

1. プロジェクト管理の組織・要員に関する現状と課題

ソフトウェア開発という生産活動は，生産物が一種の工芸品であること，生産方法が手工業的であることにより，通常の生産活動からみれば人間的要素の濃い極めて労働集約的な生産方法の域を仲々脱しきれず，それがために発生する管理上の様々な問題に直面し，対処に苦慮しているというのが，大方のソフトウェア開発部門の実態であろう。もし“効率”不滅の法則というものがあるとするならば，ソフトウェア開発部門は世の中の非効率的な部分を一手に引き受けているかの感がある。それだけに今回の調査においても生々しい回答が多く寄せられている。

ここではソフトウェア開発という生産活動の特性及び実態調査結果の2つの視点から，プロジェクトの組織・要員に関する問題点を整理してみよう。

1.1 ソフトウェア開発の特性からみた課題

最近では“ソフトウェア・エンジニアリング”という概念で，ソフトウェア開発作業の近代化を図るべく各種の研究が行われているが，ソフトウェア開発作業の“前近代性”を体系的に整理した研究は意外に少ないようである。ここでは，松井氏による分析（巻末参考文献1参照）がよくまとまっているのでその概要を紹介することとする。

ソフトウェア開発作業の特徴と管理上の問題として，次の6項目があげられている。

① 人間的要素が中心の仕事である。

開発作業に係わる，わずかな方法論，ツールの存在は認められているものの，最終的には個々の要員のスキルに支配されざるを得ない。そのため，個々人の個性が反映したり，要員間の意志疎通を図らなければならない問題が生じる。

② 自由と統制に関し自己管理が必要である。

開発においては各個人の創意工夫が要求されるため自由な発想を助ける雰囲気が必要である反面，全体の整合性を確保するためにはある程度の統制も必要とされる。要員個々人はこの二律背反する要素に対し，自己を管理しなければならない。また，ソフトウェア開発は一種の問題解決型作業であるため，与えられた命題に対し自ら作業を細分し，設定しなければならない。

③ 専門知識にライフサイクルがある。

ソフトウェア関連技術は未だ日進月歩の状況にあり，ソフトウェア開発に要求される専門知識は時を経ずして陳腐化してしまう。そのため，組織的にも個人的にも，常に新しい技術・知識が学習・習得されなければならない。

④ 経験法則優位の環境

プログラム開発能力は経験度によって大きな差があることは多くの実績例・実験例が示すところである。そのため，養成に時間を要すること，特定個人が専門職人化することなど人事管理上の問題が発生する。この能力向上を個人的な問題に帰着させている限り根本問題は解決されない。組織的な養成計画及びチーム編成によるカバー等が必要とされる所以である。

⑤ プロフェッショナル・ワークが混在する仕事である。

開発作業全体には高度な頭脳労働を必要とするプロフェッショナル・ワークと比較的単純なオペレーション・ワークが混在しており，それらを一括して管理することは人事管理のみならず，進捗管理・品質管理の面でも非常に困難である。

⑥ 水平運動が必要である。

開発要員はマネージャに対してはスペシャリストであり，両者の間には必然的に多くの不平不満が発生する。また開発要員間の不平等間を完全に取り去ることは不可能であろう。にも拘らず，目的達成のためにこの不平等が障害にならないよう，配慮されなければならない。

1.2　実態調査にみる現状と問題点

今回の調査ではソフトウェア開発におけるプロジェクト管理固有の問題から，ソフトウェア開発に限定されない人事管理一般の問題に至る，様々な回答が寄せられており，前節で述べた問題点がより具体的な形であらわれている。

1.2.1　要員のモラール向上と人間関係円滑化

この問題に関しては，どの企業においても苦慮しているものだけに関心が高く，１７２件の記述回答を得た。これをグルーピングしてまとめると図１－１のようになる。

要員の配置・仕事の配分に対する配慮が最も多くなっており，その基本的な考え方は次のように集約できる。

① チーム構成員のバランス …… 勘案するファクタとしては，経験・経歴，スキル，年令，階層関係及び相性・性格等があげられている。

② 構成員の選定 …… 適性と本人の希望を勘案する。適正に関しては個人別の技術管理台帳（いわゆるスキルズ・インベントリ）があげられており，本人の希望に関しては，公式的な自己申告制度と非公式な日常的接触によるは握の２つがあげられている。

③ 仕事量の均一化 …… 仕事自体が質・量の両方において均等に発生するものではないため，特定個人に負荷がかかる傾向は避け難いようである。仕事の質による配分は，ある程度専門分野を持たせる方向と，様々な分野を広く経験させる方向の２つに別れている。

④ コミュニケーションの充実 …… この中味はミーティング等によるフォーマルなものと懇親会・リクリエーション等のインフォーマルなものとがほぼ同数となっている。また，よく言われているユーザ部門とのコミュニケーションの強化も配慮されている。

⑤ モチベーション …… これに関しては，コミュニケーションと同数の

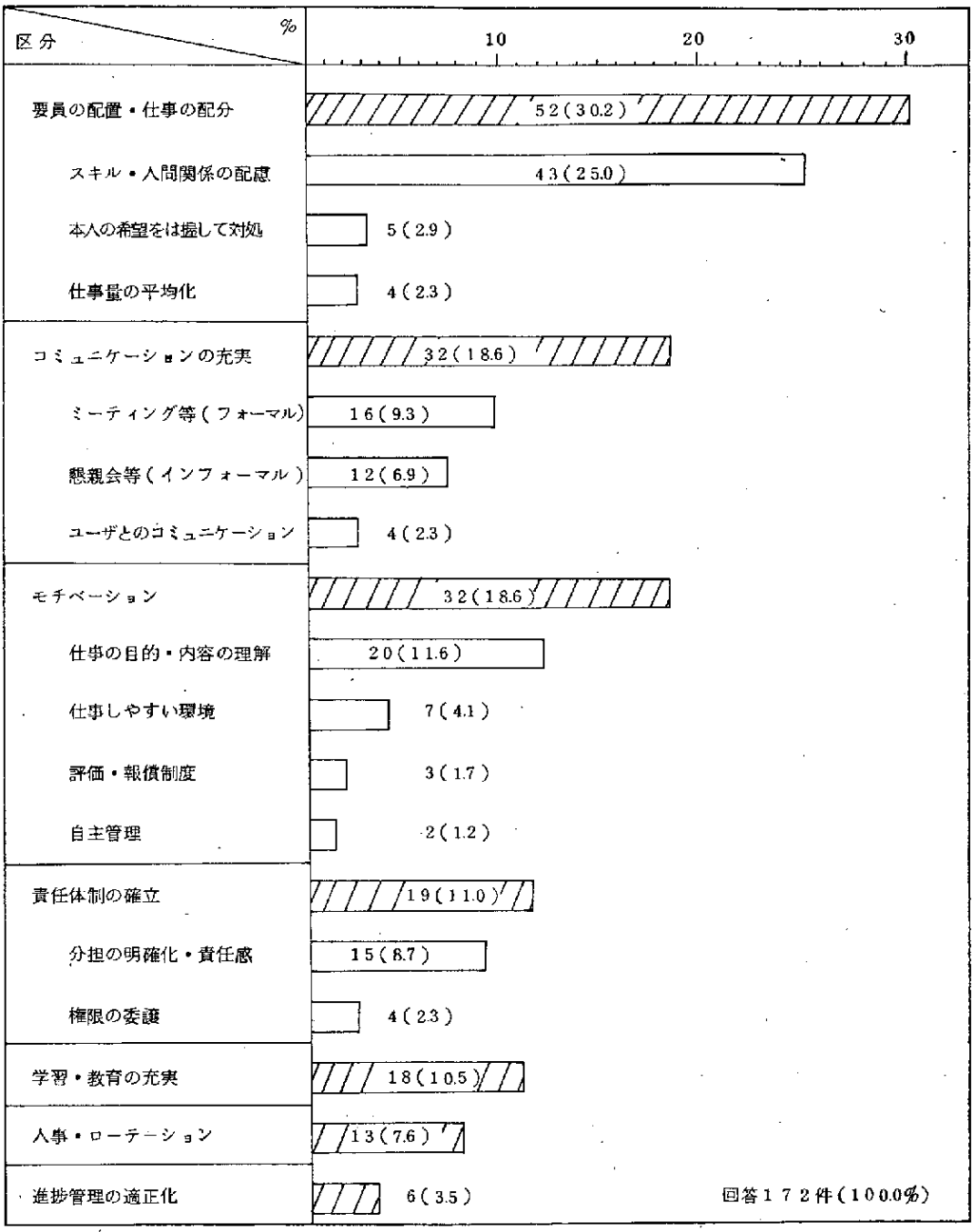


図1－1　要員のモラール向上と人間関係円滑化のための方策

回答が寄せられている。ここで注目されるのは動機づけの方法として，仕事の理解及び環境の整備が大半を占め，評価・報償制度という直接的なものは比較的少ない点である。

その外では学習・教育の充実が回答件数の1割と，相対的に少ない点が目につく。

1.2.2 直面している問題点

プロジェクト管理上，現在直面している問題に関して得られた記述回答123件のうち，要員管理に関するものが最も多く42件で，組織・体制に関するものが9件あり，合計では全体の41.5％を占めている。各々の内容は図1－2及び表1－1のとおりである。

要員不足は量的なものと質的なものが半々となっている。質に関してはコーディネータ，外注管理要員が特記されている。

ローテーションに関しては在籍期間の長期化による高令化，仕事の発生・日程等の制約によりローテーションが計画通り実施できないこと等が主な問題となっているようである。

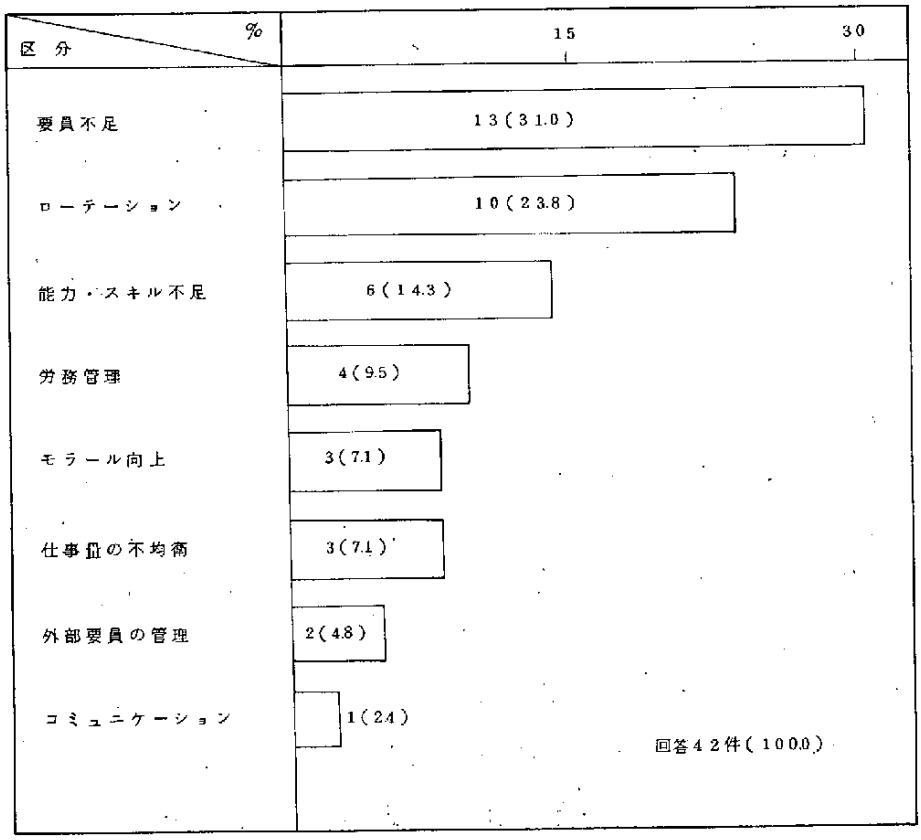
能力・スキル不足に関しては開発作業に追われる環境下で要員のスキル向上が極めて困難であるという記述が多い。

労務管理ではスケジュールの不適切さによる要員のオーバ・ロード，健康管理が主な関心事となっている。

その外では外部要員管理の困難さがあげられており，これはどこの部門においても今後の大きな課題となると思われる。

組織・体制に関しては大半の課題が要員管理に分類されたので9件にとどまっているが，レビュ体制，柔軟な開発体制，作業標準化による分担体制の確立等が注目できるであろう。

ちなみに，プロジェクト管理に関する問題点について，関東IBM研究会が実施した調査結果でも，図1－3でみるとおり，今回実施したアンケ



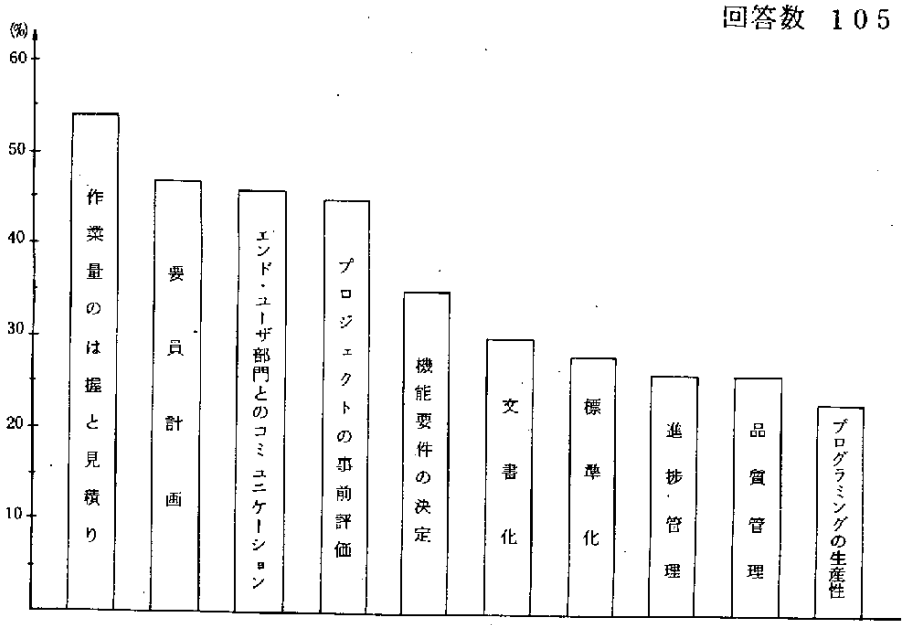
― 要員管理に関する事項 ―

図１－２　直面している問題点

表１－１　直面している問題

― 組織に関するもの ―

区　　分	件　　数
開発体制の不備 ◦重点フェーズでのレビュ体制不備 ◦階層組織の形成が困難 ◦要員配置が柔軟な組織の確立が困難 ◦予算・日程に合った体制が編成できない ◦管理・技術サポート機能が弱体	5
権限・責任等の不明確 ◦責任者の権限・責任範囲 ◦命　令　系　統	2
作業標準化不徹底による分担体制の不備	1
報告制度の不備	1



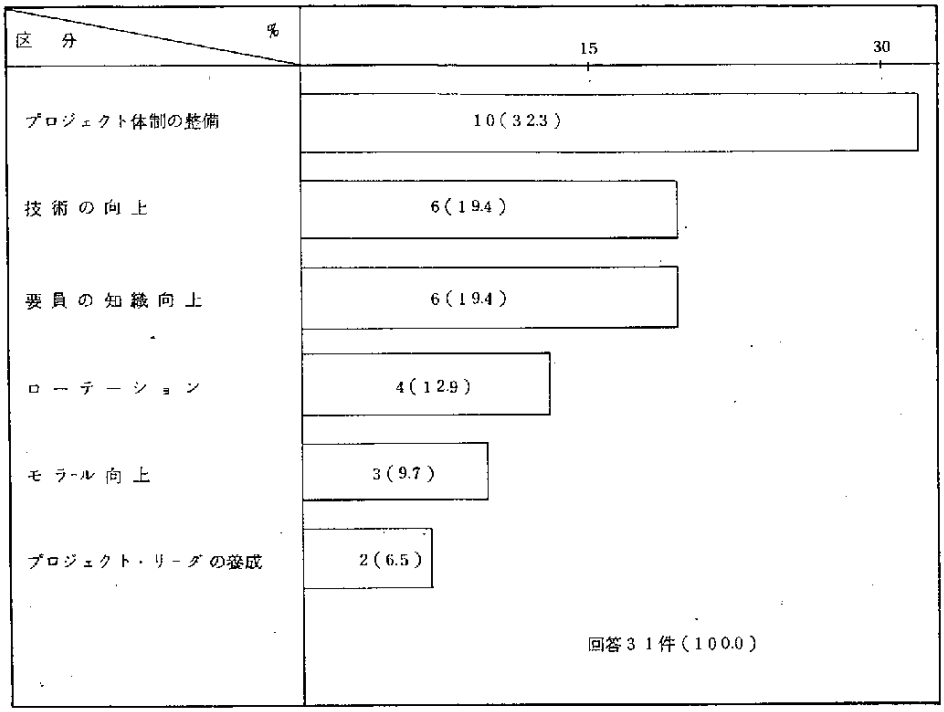
－（関東IBM研究会調査：１９７８年７月）－

図１－３　プロジェクト管理に関する問題点

ート調査結果とほぼ同様の傾向が認められる。工数見積が第１位となっているが，組織・要員に関する事項である要員計画及びユーザ部門とのコミュニケーションが第２，３位を占め，各々回答部門の半数が問題に直面しているという結果となっている。

1.2.3　今後特に力点を置くべき課題

プロジェクト管理上の今後の課題としては１６０件の様々な記述が得られたが，そのうち組織・要員管理に関するものは３１件（１９.４％）で，工程管理の３５件（２１.９％）に次いで２位となっている。その内容は図１－４のとおりである。



－組織・要員に関する事項－

図1－4　今後の課題

プロジェクト体制の整備では柔軟な組織作りに関するものが半数を占めており，既存組織の壁の中で，タスクフォース型のチーム体制を確立し，運営することが困難であることを伺わせている。その外では，チーム全体の統制とメンバの自主性の調和，複数のプロジェクトを兼務できる組織の確立等が目立った記述である。

技術の向上にはＯＪＴに関するものが2件ある。ソフトウェアの“可視化”技術（ソフトウェアの開発工程，負荷，プロダクツ等をプロジェクト・マネージャが的確には握する技術）という興味ある回答がある。

要員の知識・質向上については，個人の知識・質ではなく組織としての

向上が必要であるとして，要員の“職力”アップ，新人の早期戦力化等が具体的方策としてあげられているのが注目される。

要員のモラル向上では，ユーザ意識のエゴイズムからの脱却，システム開発の阻止要因は組織上の問題より，むしろ各メンバの意識にあることの啓発等が主な内容となっている。

1.2.4 組織・要員管理の課題への対応状況

以上の，組織・要員管理に関する現在の対策，直面している問題及び今後の課題を軸とし各々について，①要員・体制の整備，②コミュニケーション，③モチベーション，④責任権限，⑤教育・訓練，⑥ローテーションに関する回答の比率をプロットすると図１－５のようになる。自由記述内容を適宜グループ分けしていること，各々母数が違っていることからあまり厳密な分析とはなり得ないが，各々の矢印の方向から概ね次のような傾向が読みとれるのではないであろうか。

① 要員・体制の整備 …… 現在・将来に亘っての永遠の課題である。

② コミュニケーション … 既に様々な対策がとられており，当面はあまり問題となっておらず，今後特に対策は考えられていない。

③ モチベーション …… 一応の対策はとられているが，今後力を入れなければならない問題である。

④ 責任・権限 …… 当面の課題としては意識されているが，将来的にはあまり力を入れる傾向にない。但し，責任体制・権限の明確化の実質的内容が組織体制の整備の中に入っていることも考えられる。

⑤ 教育・訓練 …… 現在のところ，体系的・組織的な教育訓練はあまり行われていないが，当面スキル不足という問題に直面しており，今後は力を入れるべきであると自覚されている。

⑥ ローテーション …… 現在のところ仲々対処が困難であり，当面の大きな問題とされているが，今後の対策となると，教育・訓練に比べて

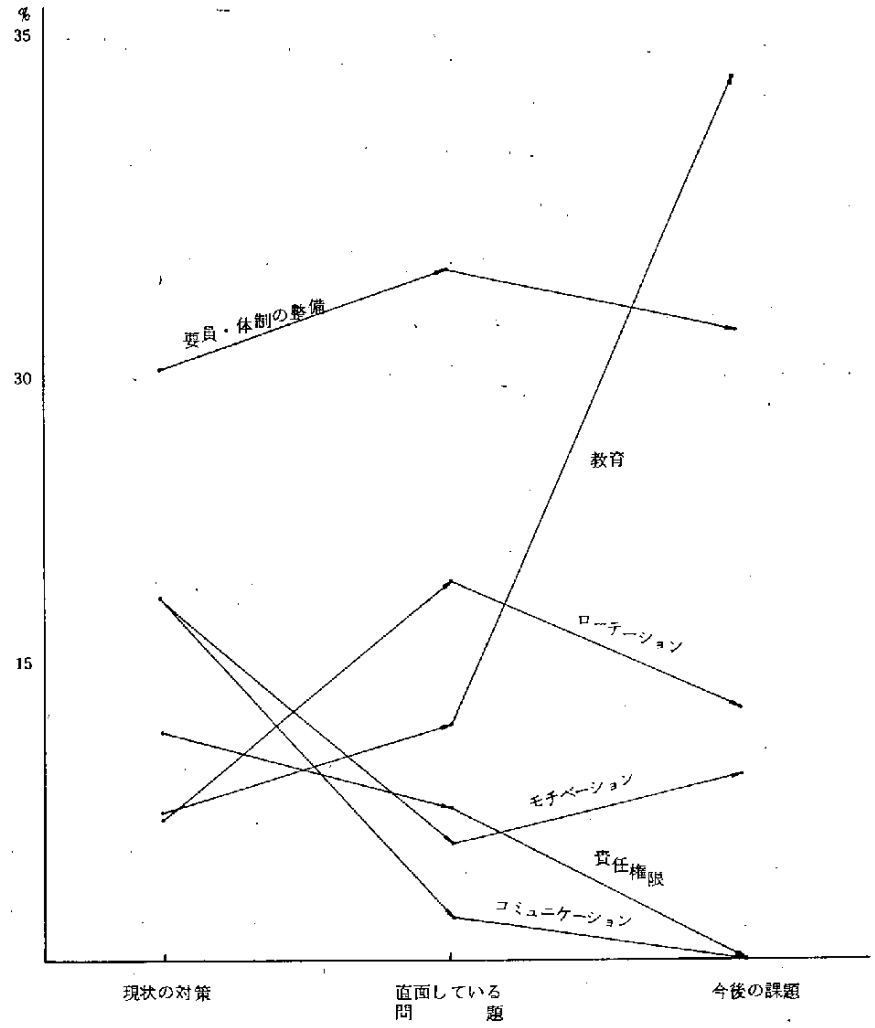


図１－５　組織・要員管理の課題への対応状況

特に適当な解決策がないこともあって，今後力を入れるべき事項の中ではウエイトが低くなっている。

以上，本節においてはアンケート調査結果から，プロジェクト管理における組織・要員管理に関する問題点をみたが，総括的にみると，どの企業においても難問が山積しており，その解決に苦慮している状況にあると言える。

しかしながら，このような現状に対して別のアンケート調査（「企業のＥＤＰマンの意識を探る」日経コンピュータ・５６年春季号）では問題は山積しているがコンピュータ関係技術者の士気は非常に高いことが報告されている。同調査の集約結果を参考に紹介しておく。

① 企業のＥＤＰ部門の士気は非常に高い。社内からの期待の高まりを強く感じ，自身の役割の重さを十分に自覚している。

② 今後の１０年は「大きな変化の時代」であり，小型コンピュータの導入，総合的な生産性向上など問題は山積していると認識されている。

③ 技術知識には大きな自信を持っている。一方，技術情報源の不足を強く感じている。

2. プロジェクト管理の組織体制

ここではプロジェクトの管理・開発を推進する組織体制に関し，組織の構成要素，編成方法，開発チーム（プログラミング・チーム）の構造の3点から検討する。

2.1 プロジェクト組織の構成

2.1.1 プロジェクト組織の要件

元々，プロジェクト組織は製品開発や大規模研究開発のための組織としてかなり以前から開発・提唱されてきたものであり，既に多くの実施事例があることは衆知のとおりである。そこで，本節では一般的にプロジェクト組織の要件として言われているもののうち，ソフトウェア開発のプロジェクト管理組織に係る部分を中心に検討してみよう。

(1) 背　　景

一般企業組織において機能別部門組織が限界に達した時点で採用されたものが，いわゆる事業部制であり，その特徴は職能部門を小規模に分割しそれを製品別等に再編成したものであった。しかしながらこの事業部制も，各事業部を横断的に見通し，異質の技術や製品を再認識することが困難であるという壁に突き当り，既存技術の見直し，組み合せ，総合化を図るための新しい組織形態が必要とされるに至った。これがプロジェクト組織という考え方の背景となっている。

このような事情はソフトウェア開発の組織においてもほぼ同様である。即ち，開発すべきシステムが複雑・高度化し，規模が大型化するに伴い，必要とされる技術が広範囲かつ高度なものとなり，ユーザ部門等関係部門も多くなり，従来の機能別組織では対処しきれない場面が度々発生するに至っているものである。この傾向は今後も増大するものと考えられる。

(2) プロジェクト組織の要件

プロジェクト組織の編成に当たってはあくまで与えられた目的達成に適合する課題中心の組織を設定すべきであり，具体的にはプロジェクト組織は以下の要件を充たしていなければならない。

① 組織構成が明快であること，即ち；
- 命令系統が統一されていること
- レベル相応の意思決定が確実になされること
- 各人の責任分担が明確であること
- 作業分担間のインタフェースが明らかであること

② 業績が正当に評価され得る組織であること

③ 管理のためのオーバーヘッドが少ないこと

④ プロジェクトの目的がメンバ全員に理解されていること

⑤ 安定性，適応力があること，即ち；
- メンバは割り込み作業や関連のない作業に煩わされないこと
- 必要とされる経験，知識，技術を持った要員が揃っていること

⑥ 従来の階層関係のみではなく，横の連絡，協力体制があること

⑦ メンバがその仕事を通じて技術力その他の面で成長し得る組織であること

以上のような要件を完全に充たす組織を実際に編成するのは困難であろうが，欠如部分にはそれ相応の対策が必要である。

このようなプロジェクト組織が機能し，威力を発揮するためには，その前提として①トップ・マネージメントの理解と承認，②全社的な認知，③プロジェクト・マネージャのリーダシップ確立等が必要であることは言うまでもない。

2.1.2 プロジェクト組織の構成要素と機能・役割

プロジェクトの成功の可否はプロジェクト組織の構造，技術力，マネージメント力及びチームワークに大きく懸っているが，本節ではプロジェクト組織の構造，即ち構成要素とその機能・役割について，ソフトウェア開発の場に即して検討してみたい。

構成要素はプロジェクトの種類，規模，開発工程によって異なるはずであるが，ここでは全体を通じて必要とされる構成員及びその機能をまとめることとする。

構成要素としては大きく，プロジェクト・マネージャ，開発チーム及び管理チームに分けられる。それらの関連及び主要機能を示すと図２－１のとおりとなる。個々の役割・機能は以下のとおりである。

(1) プロジェクト・マネージャ

プロジェクトの開始から終了までのほぼ全責任を負うことになるプロジェクト・マネージャの役割・機能は広範かつ高度である。実際の各工程において管理すべき事項は今までの第Ⅰ部において詳述されているので，ここでは必要とされる管理上の主な事項を次のように集約した。

① チームの編成

プロジェクトの目的達成に必要な要員の量及び技術力を確保することがプロジェクト・マネージャの任務の第一歩である。特に最近では必要とされる技術が広範囲に及び，かつ高度なものとなっているので，必要要員の確保は極めて困難であろう。更に，ユーザ部門からも当該業務に習熟した者を吸い上げる必要があり，抵抗があることも覚悟しなければならない。

② プロジェクト要員の管理

プロジェクト組織の要員は従来のライン部門から離れて編成されること，様々な立場の人の寄せ集め組織であること，スケジュール等の点でオーバーロードになりがちであること等からみて，その管理を適切

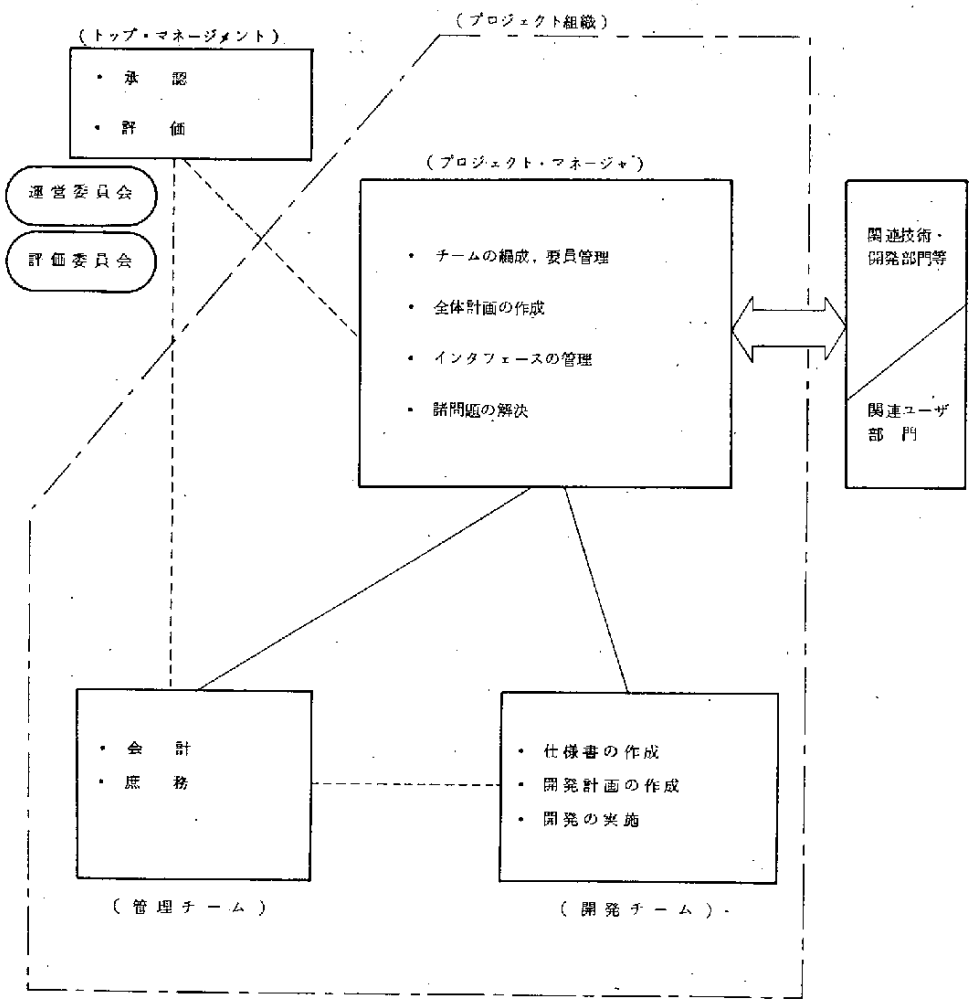


図２－１　プロジェクト組織の構成要素とその機能関連図

に行い，最終目的の達成まで，一定水準のモラールを維持し，円滑な人間関係を保つには極めて高度な組織力を必要とする。プロジェクトの要員管理に関しては，次章において詳細に記述することとする。

③ インタフェースの管理

プロジェクトは規模にもよるが，全体をタスク，サブ・タスクに分解してチームあるいはグループに分担させるのが普通であり，それらを統合し整合性を確保するためのインタフェース管理はプロジェクト・マネージャの最も重要な役割・機能の1つである。一般にこのインタフェースはシステム間，組織間及び個人間の3つに分けられる。

システム間のインタフェースは分割されたモジュール間の整合性を確保するものであり，管理の内容としてはモジュールの内容のみならず，各々の開発の進捗度合の調整も図る必要がある。

組織間のインタフェースは，プロジェクト内のチーム間の調整と，プロジェクト組織外の組織（管理部門，ユーザ部門等）との調整の2つがある。特に後者については，各部門には独自の職務，規律等があるためプロジェクト組織とは潜在的な衝突が存在することは止むを得ず，プロジェクト・マネージャはそれらが非生産的な利害論争に陥らないように配慮しなければならない。また，ライン部門，ユーザ部門には，誤解にもとづく被害者意識が往々にして発生することがあり，これら“外乱”をさばき，プロジェクトに悪影響を及ぼさないように配慮することも重要な任務である。

更に，目標期限内にプロジェクトの目的を達成するために，スケジュールと実際の進捗を対比しながら，必要なリソースを組織的に確保・投入しなければならない。必要に応じて現業部門等からの応援を受けるべく努力しなければならない。

④ 諸問題の解決

プロジェクトの開発過程で内外に発生する関連諸問題を即時，適切

に解決することが要求される。この問題は，プロジェクトは非定型業務であるため，各々特有の問題を抱えているのが実情であり，その都度個々に対処しなければならないだけに大変である。

(2) 開発チーム及び管理チーム

開発チームはプロジェクト・マネージャの指揮・監督の下に実際の開発にあたればよく，プロジェクト特有の問題はあまりない。むしろ，開発チームにプロジェクト遂行上の煩雑な問題を意識させないよう配慮することが，プロジェクト・マネージャの責任であろう。開発チームの編成・構造・役割・機能については 2.3 で詳述する。

管理チームの任務は，開発チームの管理業務負荷を極力軽減し，開発に専念できるようにすることに尽きる。

(3) トップ・マネージメント

プロジェクトのキック・オフに承認を与えプロジェクト・マネージャを選定し，全権を付与すれば任務は終了する。通常，プロジェクト組織の直接的な構成要素とされないが，先に述べたように，プロジェクト成功の鍵としてトップ・マネージメントの理解，重要フェーズでの参画は重要なファクタである。また，大規模かつ重要なプロジェクトにおいては，トップ・マネージメントの下に全体の運営を管理する運営委員会（Steering Committee）及び，フェーズ毎に評価を行い，次のフェーズへの移行に承認を与える評価委員会が設置されることもある。

2.1.3 開発工程と組織構成

プロジェクト組織は特定の作業開発のために必要な要員を配置する一時的な組織であることに特徴があり，流動性を持っている。そこでは要員を必要に応じて任命したり，解任したりすることが可能である。

ソフトウェア開発のプロジェクト組織においては，システム設計，プログラミング，テスト等の開発工程毎に必要とされる要員の質及びその人数

は自ずと変化するはずである。その点においても構造に柔軟性があるプロジェクト組織は有利であると言えよう。開発工程によるプロジェクト組織構成の変遷の典型例を図２－２に示す。本図から，開発工程による構成要素の異動，消長が読み取れるであろう。

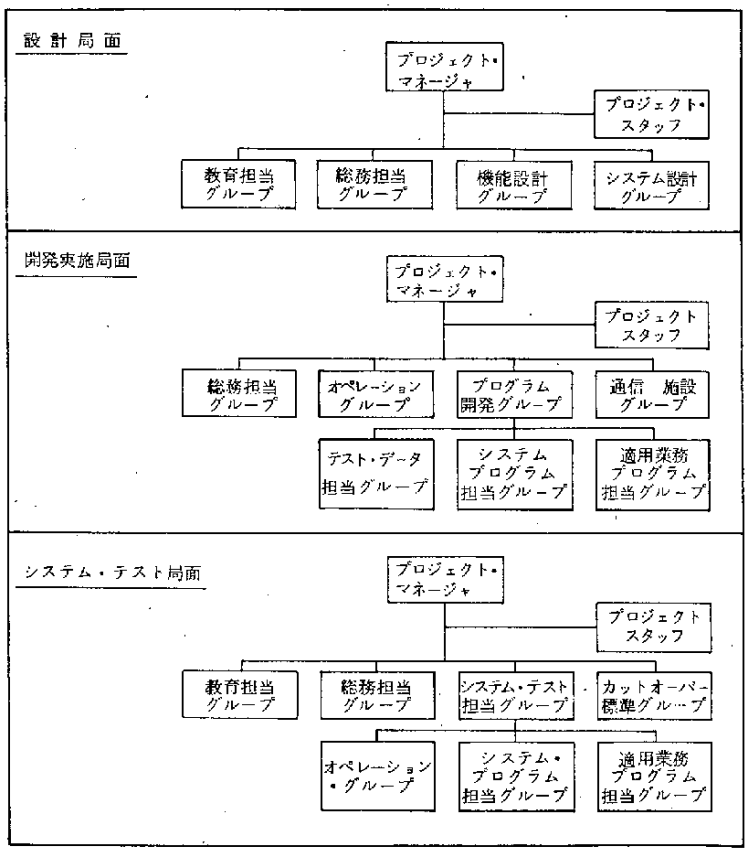


図２－２　システム開発工程とプロジェクト組織の構成の例

開発工程別の役割・分担の概要をまとめると表２－１のとおりとなる。

表２－１　開発工程別役割・分担

開発サイクル	プロジェクト・マネージャの役割	担当チーム	チームの役割
システム化要求	◦管理チーム※と開発チームのコミュニケーション・リンク確立 ◦ライン部門の管理者から情報収集 ◦プロジェクト・チーム全員に，プロジェクトの目的を理解させること ◦暫定的なプロジェクト計画の作成	管理チーム※	◦実行可能性分析，費用対効果分析等の評価 ◦目的達成上の問題点の摘出，理解
システム分析	◦メンバの参加意識，意欲の創成，維持	開発チーム	◦データ要素，インプットアウトプットの分析
システム設計	◦設計仕様書の内容の検検討，評価 ◦作業分担の決定	開発チーム	◦システム・フローの作成 ◦詳細開発計画の作成
プログラミング	◦開発の工程管理 ◦システム間のインタフェース	開発チーム	◦プログラムをモジュールに分解 ◦コーディング，デバック
テスト	◦モジュール結合，テストの監視	開発チーム	◦モジュールの結合，テストの実施
ドキュメンテーション	◦工程管理 ◦内容の検討・評価	開発チーム	◦利用手引書，解説書訓練資料等の作成

（注※　この場合の管理チームには，運営委員会，評価委員会を含む）

2.1.4　プロジェクト組織編成の手順

プロジェクト組織編成手順の概略を図２－３に示す。

① マネージャの指名 …… 既に述べたように，プロジェクト・マネージャはプロジェクトの成否の鍵となるものであるから，技術力は言うに及ばず，マネージメント能力，統卒力，人間性等を十分勘案し，当該プロジェクトに最適な人材を選定する必要がある。プロジェクト・マネージャに必要とされるスキル，資質については3.1で詳述する。

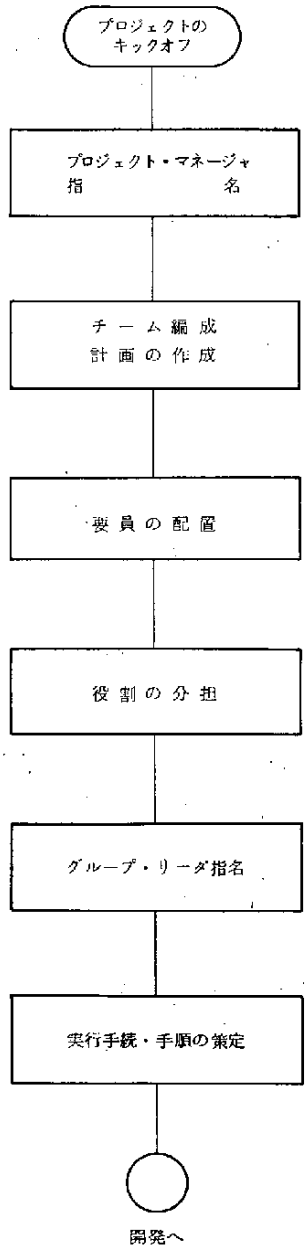


図２－３　プロジェクト組織編成の手順

② チームの編成計画作成 …… ここでのプロジェクト・マネージャの任務は以下のとおりである。

・見積書に記述されている要員計画の見直し
・開発作業の分解（いわゆるＷＢＳ）
・開発日程計画の作成
・必要要員数及び技術力の配置計画の作成
・ライン部門管理者との調整及び承認のとりつけ

③ 要員の配置 …… 社内のスキルズ・インベントリ，個人の希望等から候補者を選定し，所属部門長の了解を得た上で，面接ヒアリングにより配置を決定する。

④ 役割分担 …… 先のWBS等にもとづいてグループを編成する。この際，スキル，経験度等のみならず，グループ・メンバの人間関係にも細かな配慮が必要とされる。

⑤ グループ・リーダの指名 …… 適任者の選出とプロジェクト・マネージャとの責任分担の明確化がポイントとなる。

以上が完了すれば，必要な手続，手順を決め，実際の開発作業に入れるはずである。

2.2 プロジェクト組織の形態

前節ではプロジェクト組織一般に共通する編成方法について述べたが，ここではプロジェクト組織にはどのような形態があるか，またその特徴は何かについて，相互に比較しながら紹介する。

組織形態は通常，機能別組織，マトリクス組織，タスクフォース組織の３形態に分けられ，機能別組織には主として管理方法の違いで職制型と調整型の２つがある。プロジェクト組織としては非常に緩いものであるが，委員会組織も広い意味で一形態として考えてよいであろう。各々の特徴をまとめると表２－２のようになる。

表２－２．プロジェクト組織の形態

組織形態		専担マネージャの有無（○：有 ×：無）	プロジェクト・マネージャの権限（○：強 △：中 ×：弱）	既存組織からの分離度（○:完全 △:不完全 ×:分離せず）	適用プロジェクト	
					規模（○：大 △：中 ×：小）	複雑さ（○：複雑 ×：簡単）
機能別組織	職制型	×	△	×	×	×
	調整型	○	○	×	×	×
マトリクス組織		○	○	△	○	○
タスクフォース組織		○	○	○	△	○
委員会組織		×	×	×	×	×

2.2.1. 機能別組織 — 職制型

機能別組織は既存の組織をそのまま利用するもので，プロジェクト組織として特に新たな組織は設置されないが，プロジェクトは公式にオーソライズされ，ライン・マネージャが兼務でプロジェクト・マネージャに任命される。マネージャに対し，プロジェクト推進上必要な権限が付与され，関係部門は公式，非公式両面で協力が要請される。

プロジェクト・マネージャは全体計画及びタスクの決定，複数部門に関係する事項についての問題解決・調整・変更管理等を行う程度であり，作業の詳細定義，作業の実施，要員の配置・管理，工程管理等は各担当部門マネージャが行う。この組織形態の構造は図２－４のとおりである。

この組織形態の利点としては次のようなものがあげられる。

① 要員の配置に柔軟性がある。

プロジェクトの完了までには，要員のスキル，数が開発過程によって変動することは既に述べたところであるが，プロジェクトのために別途組織を編成すると，要員の異動は一定の手続，人事管理上の考慮及び関

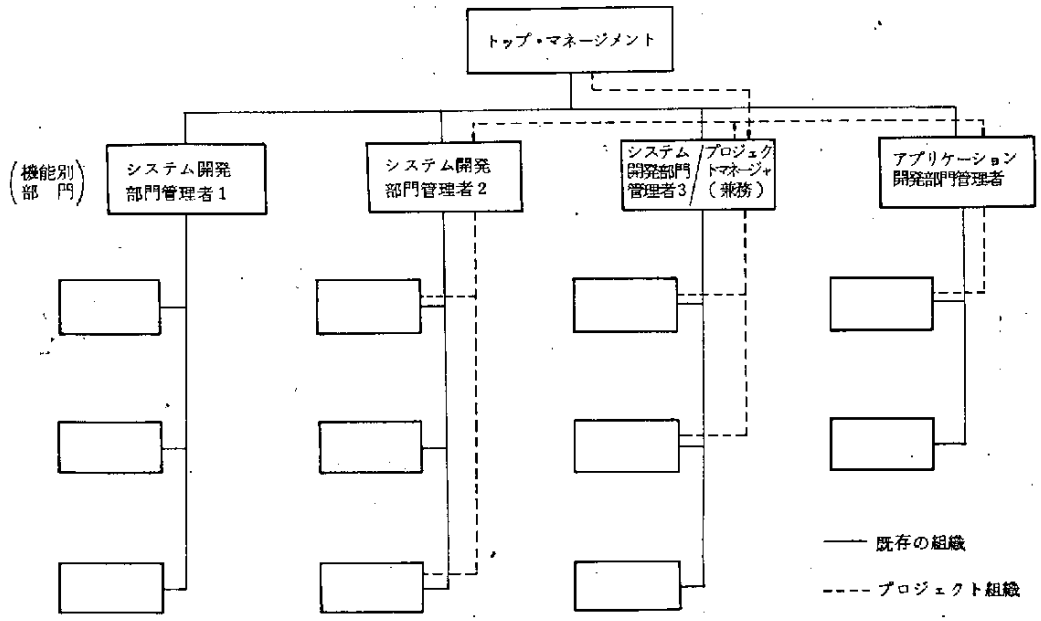


図２－４　機能別組織 － 職制型

連部門管理者の了解等が必要になる。機能別組織ではライン・マネージャの自己の判断によって自部門内要員の異動を決定することが可能であり，組織に融通性がある。

② 作業毎の実施が容易である。

タスクの割り当ては部門の技術，実績等の蓄積を勘案して行われるので，各部門が自分の作業を実施するのは比較的容易である。

③ 工程管理，要員管理が容易である。

各部門管理者は従来の階層組織の中で，必要な管理を行えばよく，その点で特に作業の工程管理，要員の管理が容易である。

機能別組織の欠点としては以下のようなことがあげられる。

① 管理・責任体制が不明確である。

作業が部門毎に分担され，開発の進捗に応じて横断的に移動していく

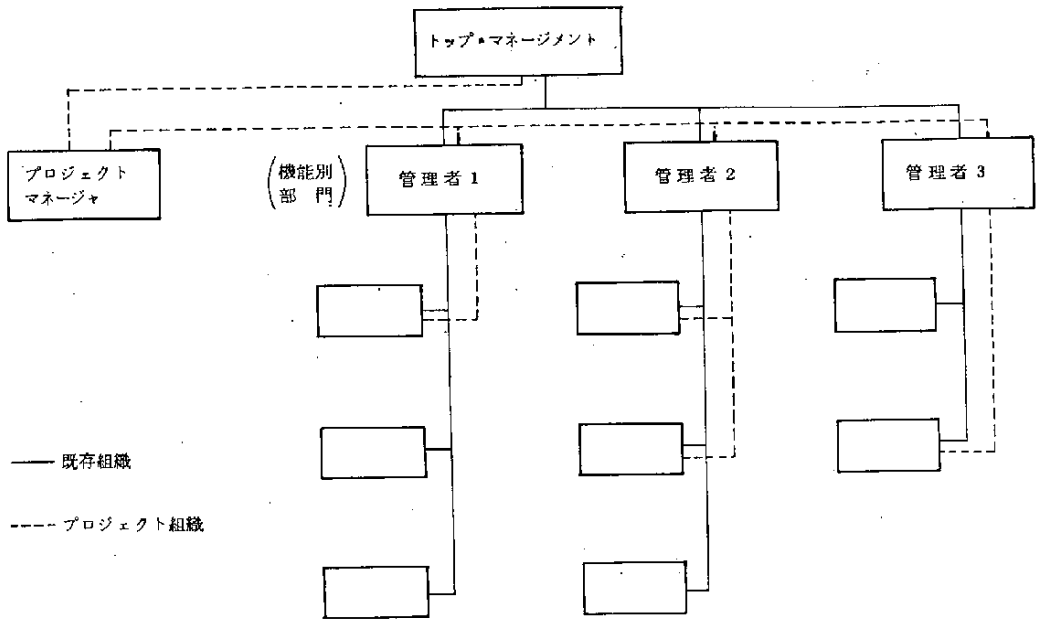


図 5.1 0　機能別組織－調整型

ため，プロジェクト全体の管理，責任体制が不明確になりがちである。特に，部門間にまたがる責任問題が発生した時は，機能部門管理者と兼務であるプロジェクト・マネージャでは対処が困難である。

② 意思決定を迅速に行うことが困難である

各部門はプロジェクトの一部分を担当しているだけであるため，全体に及ぶような意思決定を総合的な見地から行うことは困難であり，関係部門管理者総ての決裁・合議が必要とされるので時間を要す。

③ 機能部門本来の業務との区分けが困難である

プロジェクト・マネージャは機能部門の管理者を兼ねているが，ともすれば自部門の業務を優先しがちである。また，各機能部門においても，プロジェクトにアサインした要員を当該部門の業務から完全に"隔離"するのは事実上困難であろう。

2.2.2 機能別組織 — 調整型

この組織形態は機能別組織の一変型であり，作業の実施は各機能部門に任されるが，専任のプロジェクト・マネージャが設置される点が先の職制型と異なる。但し，このマネージャの権限は職制型に比べ限定されており，機能部門管理者には及ばず，各関係部門とのコミュニケーションによって，作業の完成及び統合に影響を与えるところに特徴がある。即ち，この組織形態のプロジェクト・マネージャはプロジェクト活動を直接的に管理するというよりはむしろ監視したり，進行を助けたりするのが主な任務であり，関与の仕方としてはミーティングの場において討論や問題の解決を促進したり，係争事項の調整を行うことが中心となる。本組織形態の構造は図２－５のとおりである。

調整型組織の得失は職制型のそれと類似するが，プロジェクト・マネージャが専任であるため，職制型における兼務から派生する問題に対応できる。但し，職制型に比べマネージャの権限が弱いという欠点がある。

2.2.3　マトリクス組織

マトリクス組織は機能組織を温存したまま，いわゆるプロジェクト組織のメリットを活用しようとする形態であり，製造・製品開発及び研究開発等の大規模プロジェクトにおいて採用され，実績をあげている企業も少なくない。米国ＮＡＳＡの宇宙開発プロジェクトがこの組織形態に拠って成功したことで，一躍その有効性が注目されるに至った経緯がある。

組織構造をシステム開発に沿って示すと，図２－６のとおりとなる。

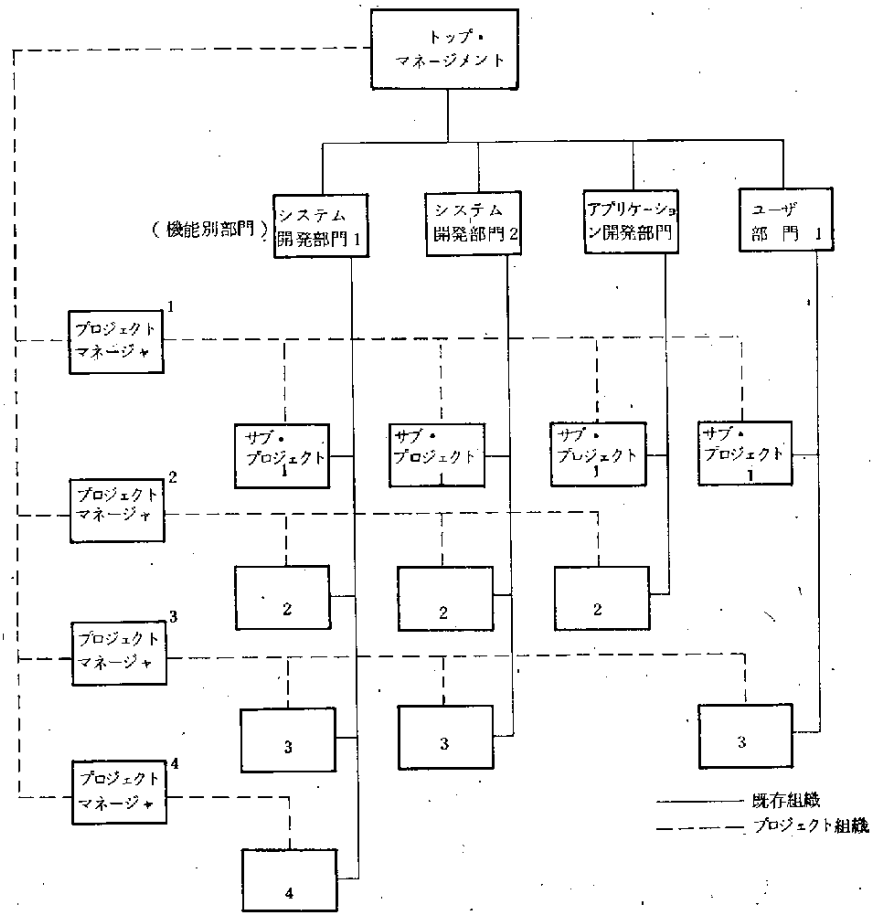


図２－６　マトリクス組織の構造

プロジェクト完成に向けて全面的な権限を持つマネージャが任命され，計画立案，スケジューリング，必要とされる全リソースの確保・維持・管理，総合調整及び開発実作業の管理等全責任を持たなければならない。但し，プロジェクトに配置された要員の人事上の管理は従来の職制上の管理者が行う。

プロジェクト・マネージャの権限を横断するような二重構造であるため，プロジェクト全体の運用・管理は組織論からみれば非常に難しいはずであるが，多くの企業で採用され，成功を収めているのは，この形態が実践的であること，機能的かつ合理的である等の理由によるものであると考えられる。

マトリクス組織の利点として次のような事項があげられる。

① 権限・責任体制が明確である。

マトリクス組織の構造は一見，機能別組織のそれと類似しているが，一番大きな差はマトリクス組織においてはプロジェクト・マネージャの負荷は絶大となるが，権限・責任体制は明確であるというメリットがある。

② 意思決定，インタフェース及び問題解決を迅速に行える。

プロジェクト・マネージャの権限が強大であること及び指揮系統が確立されていることから，意思決定，関連部門との調整及び問題への対処等が迅速に実行される。

③ 要員の配置，活用を効率的に行える。

プロジェクトにアサインされた要員は従前の機能部門に留ったまま作業に専念することになるので，割り当ての作業間の空白期間におけるスキルの習得，又は終了後における本来業務への復帰が容易であり，個々の要員の効率的活用が容易である。

④ 専門技術の蓄積が可能である。

プロジェクト完了までに習得した専門技術，ノウハウ，経験を各機能部

門に蓄積することができるので，それを他のプロジェクトに応用するというテクノロジー・トランスファが容易である。

マトリクス組織は以上のような利点を持つかわり，機能別組織形態を併存させているため，先にあげたような機能別組織に伴う欠点と類似するものを持つことになる。主として問題となる点は次の2つである。

① プロジェクト・マネージャと機能部門管理者の意見衝突が発生しがちである。

要員は機能別管理者の下に置かれたまま，プロジェクトの全体的視野から管理が実行されるので，両マネージャの間の意見の衝突はどうしても避け難い。この場合，機能別組織と比べて，プロジェクト・マネージャの権限が大きいだけに，機能別組織に比べれば，解決は容易である。

② 要員管理の二重構造によるモラールの低下を招く恐れがある。

プロジェクト要員は同時に従来の機能部門のスタッフでもあり，一時に2人の管理者の指揮命令下に置かれることになり，その不安定さがモラールの低下に結びつく危険がある。マトリクス組織の最も重大な欠点とされているものである。

2.2.4 タスクフォース組織

対象プロジェクトのために目的指向的に要員を集め，プロジェクト・マネージャはプロジェクトの管理，目的達成及び要員管理等に関する全権限を付与される。この組織形態は機能別組織と対極をなすものであり，通常，単に「プロジェクト組織」という場合，この形態を指すことが多い。ソフトウェア開発の場合のタスクフォース組織の代表例を図2－7に示す。

タスクフォース組織の利点は以下のとおりである。

① プロジェクトの遂行責任が明確である。

この組織形態は専任のプロジェクト・マネージャがプロジェクト遂行

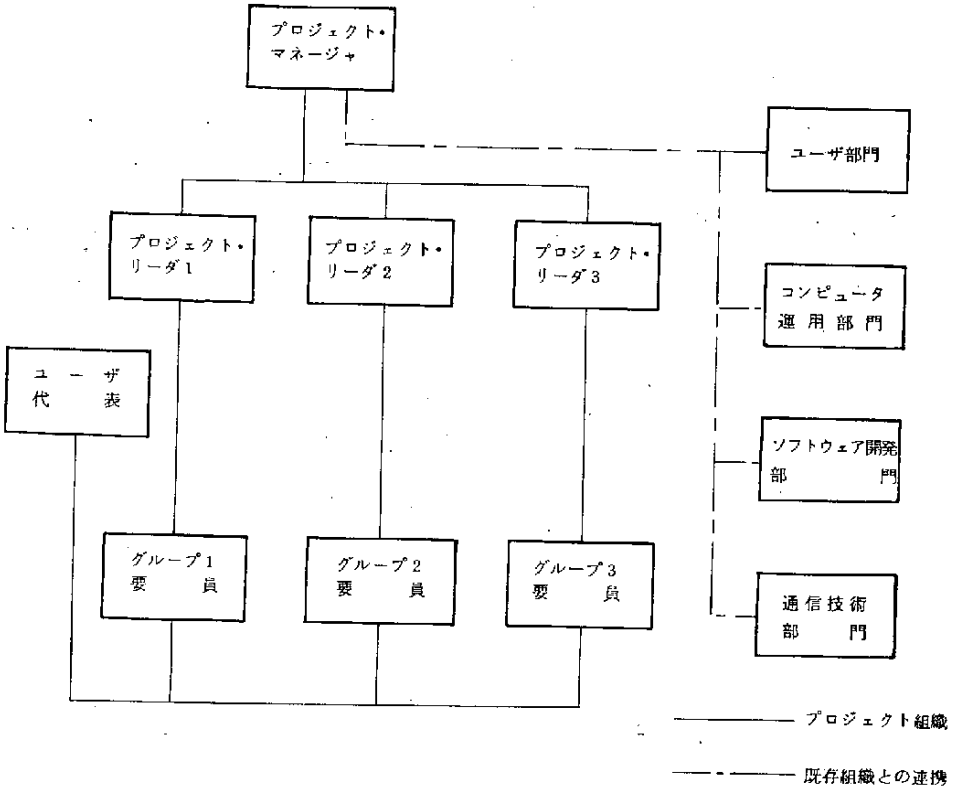


図2－7　タスクフォース組織の構造

上の関係者を統轄，管理する責任を負っている。即ち，プロジェクトの全責任は機能別組織のようにトップ・マネージメントにあるのではなく，プロジェクト・マネージャに帰着するのである。その点で，責任・権限の所在はプロジェクト組織形態中では最も明確であり，強固である。

② 組織の運営が容易である。

必要とされる権限，リソースは1人のマネージャに付与されるので，指揮命令系統が一元化され，要員も直接管理できるので，組織運営は他の組織形態と比べ容易である。特に，プロジェクト要員は従来所属していた組織から分離し，プロジェクトに専念できるので，機能別組織及びマトリクス組織のような管理の二重構造による弊害がないことは大きな

利点である。

③ 組織に柔軟性がある。

タスクフォース組織内の構造は従来の階層構造又は機能別構造にとらわれることなく，プロジェクトの進捗及び環境の変化に対応できる柔軟性がある。その結果として，縄張り争いに悩まされることもなく，新しい試みやアイデアを受け入れ易い素地ができるというメリットが派生する。

④ 意思決定等を迅速に行える。

プロジェクト進行過程の情報は直接かつ迅速にプロジェクト・マネージャに集中するので問題の解決，意思決定を円滑かつ迅速に行うことが可能である。更に，問題の発生を予測し，事前に対策を講ずることも可能となる。

⑤ 各種のインタフェース確保が容易に図られる。

プロジェクト組織外との接衝，調整及びプロジェクト作業間のインタフェース確保等はプロジェクト・マネージャの権限によって円滑に実施できる。

タスクフォース組織は以上のように多くの利点を有している反面，次のような欠点もある。

① 要員管理上，非効率となる

プロジェクトの開発サイクルによって，必要要員数が変動することは既に述べたとおりであるが，タスクフォース組織の場合はメンバが出身部局から離れて配置され，しかもプロジェクト期間中は固定的に配置されることが多いため，進捗状況によってメンバの作業担当に空白が生じ，要員活用上全体として非効率になる可能性が高い。更に，一組織が複数プロジェクトを抱えている場合，プロジェクト間の要員移動がスムーズにいかないという欠点もある。これらの要員配置上の非効率の背景には，プロジェクト・マネージャとしては出来る限り早い時期にプロジェ

クトを編成し，その容量をプロジェクト開発のピークに合わせ，かつピークが過ぎても不測の事態に備えて一度確保した要員を確保し続けようとするという事情がある。

② 適正能力を持った要員を配置することが困難である。

プロジェクトのメンバは従前の現業部門等から離れてしまうので，現業部門の管理者としては自部門の業務遂行上の必要から，プロジェクト遂行上の必要から，プロジェクト遂行上本当に必要とされるスキルを持った要員をタスクフォース組織に振り向けることに必ずしも積極的ではないという事情があることは避け難い。その結果，プロジェクト・マネージャにとって真に必要とされるだけの技術力，要員数を確保することは極めて困難となるのが普通である。

③ 技術のトランスファ及び標準化が困難である。

タスクフォース組織の場合，プロジェクトの目的に沿ってチームが編成され，その達成後は任務が終了し，組織としては解散されるので，プロジェクト期間中に習得された技術・ノウハウを組織的に蓄積することが困難である。このため，あるプロジェクトにおいて蓄積された優れた技術要素が他のプロジェクトに利活用されず，技術のトランスファ及び効率的活用が阻害されることになる。更に，プロジェクトが臨時的なものであるので作業，文書等の標準化が推進されることもあまり期待できない。

④ プロジェクト・メンバに不安感を与える恐れがある。

プロジェクト・メンバは自分の所属部門から一時的に離れて，当該プロジェクトに要求される技術，手続に従わざるを得ず，プロジェクト終了後出身母体に復帰する場合，空白期間のギャップを自ら埋めなければならず，場合によっては復帰場所そのものも確定していないという不安感がつきまとう。欧米諸国のように，専門技術の尊重及び契約による雇用形態が定着している場合には，自分のスキルを発揮できるプロジェク

トに参画し，そのプロジェクト終了と共に契約が解除されるということも行われているようであるが，終身雇用を前提とした日本的組織・風土においては，プロジェクト・メンバに人事・処遇上の不安を与える恐れは十分にあると言わざるを得ない。まして，一つのプロジェクト終了後，現業部門に復帰せず，絶えずプロジェクトを駆け巡ることになると，その不安感は更に増大することになろう。

⑤ 長期，大規模なプロジェクトの場合，組織維持が困難である。

以上述べたようにタスクフォース組織ではいわば一時的なプロジェクトに対し，各々の分野の専門技術を有する要員を一定期間固定的に割り当てるため，当該企業全体のバランスからしても，長期に亘って多くのメンバを配置することは困難であろう。その意味でタスクフォース組織は，その企業全体にとっての至上命令的な，重要かつ戦略的なプロジェクトの場合を除き，通常は大規模なものには適用せず，一般的に要員数は１０人程度が限度とされている。更に，本組織形態の場合，要員のモラールの維持が難しいので，大型プロジェクトの場合，投入人員を増やし期間を極力短かめにするというような配慮が必要とされる。

2.2 5. 委員会組織

今まで述べた４つの組織形態は，管理方法等の違いはあるがいずれも実際の作業に従事する要員が配置されるが，この委員会組織には開発作業に当たる要員は特に配置されないのが普通である。従って，この組織形態は戦略的事項に関するプロジェクト，即ち，当該機関にとって重要な政策的事項の決定，長期計画の策定，直面している重要問題への対処及び基本方針の確立等に適したものである。

また，上記４組織形態のいずれかによってプロジェクト開発を行う際，そのプロジェクトの発足までの事前調査，プロジェクトの評価等のための組織として採用されることもある。

この組織形態は特定問題に対する臨時的な特別委員会の形を採ることが多く，メンバは会議の時だけ参加するのみで，事務局的な組織は設置されるが，責任者が専任で配置されることは少ない。権限の強さは，指名される責任者及びメンバのその時点における職制及びその委員会組織に与えられた任務の重要性等によって大きく変わることになる。

実施部隊を有していないので，この委員会組織において検討され，提案される結論，方策等の実行をどのように保証するかが，ポイントとなる。

委員会組織は組織としては非常に緩いものであるため，編成・解散が容易であり，組織運営及び要員管理等の煩しい問題がないという利点がある。更に，関係部門の意見を広く吸い上げ，必要な調整を行う場としては有効であり，多く採用されている組織形態であるといえよう。

以上の各組織形態の利害得失をまとめると表２－３のようになる。

表2－3　プロジェクト組織の形態別利点と欠点

区分 / 形態	利点	欠点
機能別組織 職制型	◦要員配置の柔軟性 ◦作業遂行の容易性（技術，経験の蓄積） ◦工程管理，要員管理が容易	◦責任体制の不明確さ ◦意思決定の遅滞 ◦本来業務との区分が困難
調整型	◦専担のプロジェクト・マネージャによる調整機能が明確	◦権限が弱い
マトリクス組織	◦権限・責任体制が明確 ◦問題解決の迅速さ ◦要員配置の効率性 ◦技術の蓄積が可能	◦機能別部門の管理者との衝突 ◦要員の不安定感，モラール低下
タスクフォース組織	◦権限・責任制制が明確 ◦組織運営の容易性 ◦組織の柔軟性 ◦意思決定の迅速性 ◦インタフェース確保の容易性	◦要員配置の効率性 ◦スキル確保の困難さ ◦技術のトランスファ，標準化が困難 ◦要員の不安感 ◦長期・大規模なプロジェクトに不向きである
委員会組織	◦組織の編成，運用等が容易 ◦関連部門の調整が可能	◦権限，責任が不安定 ◦実施部隊が不在

2.3　開発チームの構造

前節では5つのプロジェクト組織について，組織全体の権限及び管理形態に着目し，それらの特徴を述べたので，ここではそれらの組織形態の下で実際のソフトウェア開発作業に当たる開発チームについて紹介する。

2.3.1　チーム構造とその特徴

ソフトウェア開発が担当者個々人のスキルや経験に依存している限り，生産効率の抜本的な向上は期待できず，更に対象ソフトウェアの大規模化や複雑化に伴って品質の確保も困難となってくる。開発チームの構造に関する理論及び実践が議論を呼んでいる背景には，このような事情があることは否定できない。即ち，ソフトウェア開発工程の機械化には限界があることを前提として，少しでも組織的・機能的な体制で対処することによって，生産効率の向上及び品質の確保を図ろうという気運が高まっているのである。ソフトウェア開発プロジェクトにおいても開発チームの編成は重要な問題であり，対象ソフトウェアの特性，メンバのスキル，マンパワーの容量及び環境条件等を勘案して適切なチーム編成を選定する必要がある。

開発チームは，その構造を政治の権力構造に擬して①独裁的・集権的チーム（いわゆるチーフ・プログラマ・チーム），②統制的・分権的チーム（階層チーム），③民主的・分権的チーム（エゴレス・チーム）に分類される。各々の特徴は以下のとおりである。

(1)　チーフ・プログラマ・チーム

Baker，Mills らによって提唱され，ＩＢＭのＩＰＴ（Improved Programming Technologies）の一環に組み込まれ，ソフトウェア開発管理技術として，有効性が認められているものである。

このチーム編成方法の根底にある考え方は作業分担を徹底することによって，生産効率の向上，工程管理の容易化及び品質の確保を図ろうとすることである。

チーム構成はチーフ・プログラマ１人，バックアップ・プログラマ１人及びライブラリアン１人を中核とし，作業工程毎に必要に応じ，システム・アナリスト，プログラマの支援を求める。チーム構成の例を図２－８に示す。

各構成員の任務は次のとおりである。

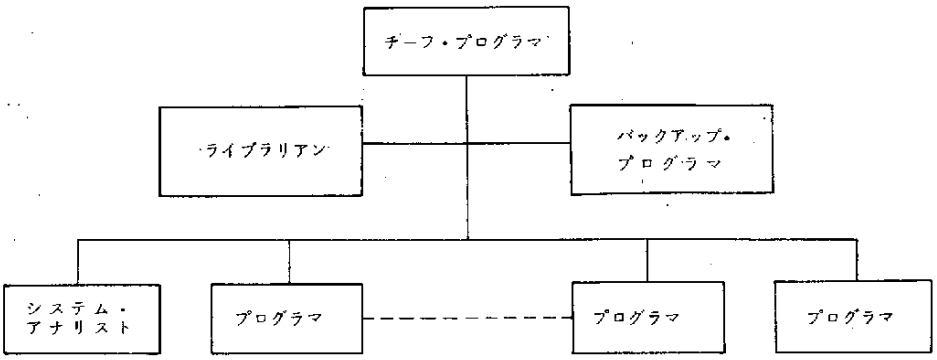


図２－８　チーフ・プログラマ・チームの編成

① チーフ・プログラマ

上級クラスのプログラマでチームが担当したプログラム作成の全責任を負う。プログラム設計を行うのみならず，重要なモジュールに関しては自らコーディングまでを行う。また，他に委ねたモジュールを含めて，全プログラムに目を通す。

チーフ・プログラマのもう一つの任務はチーム全体の管理である。即ち，チームの組織運営管理，要員管理といった一般的なマネージメントとプログラム開発の進捗度合，モジュール間のインタフェース及び品質等の技術管理の双方に責任を持っており，“独裁的”と称される所以である。

② バックアップ・プログラマ

チーフ・プログラマ・チームではチーフ・プログラマに技術及び管理面の責任が集中するので，主として技術面のサポート，即ちチーム・プログラマの指揮下でプログラムの設計，コーディングを行うとともに，チーフ・プログラマの見落し，ミス等を指摘する要員が必要となる。その意味ではバックアップ・プログラマはチーフ・プログラマと技術的にはほぼ同等の能力を有し，チーフ・プログラマの相談役又は代理を勤めることができなければならない。

③ ライブリアン

チーフ・プログラマの管理面をサポートする要員と考えてよく，開発支援ライブラリの管理を専担することによって，プログラマを作業に専念させるようにすることが主な任務である。具体的にはプログラム作成に必要なドキュメント類の整備・管理，ＪＣＬ，プログラム，テスト・データなどの集中管理（収集，整備，更新等），その他庶務事務等を受け持つ。

チーフ・プログラマ・チームの利点は以下のとおりである。

- 組織力によるプログラム開発 …… チーム構成員の作業分担，責任等が明確になり，インタフェース管理及び全体の統合が容易になる。
- 生産効率の向上が図れる …… ＩＰＴ技術（ＨＩＰＯ，トップダウン技術，構造化プログラミング，ストラクチュアド・ウォークスルー等）と有機的に結合させ，生産効率の向上を図ることが可能とされている。一人の"スーパーマン"プログラマと小人数のチーム・メンバという構成によって，指揮命令の一元化，コミュニケーションの円滑化，冗長性排除，事務作業の分離等も全体の生産効率向上に寄与している。
- 進捗，品質の管理が容易である …… チーフ・プログラマが直接作業に参画し，委ねた分も直接目を通すので，全体作業の進捗度及び品質の管理が容易に行える。

チーフ・プログラマ・チームの欠点としては以下のようなものがあげられる。

- チーフ・プログラマの能力への依存度が高い …… 先に述べたように，チーフ・プログラマは技術及び管理面両方の責任を有するので，かなりの能力が必要とされ，また負荷も大きい。それだけに適任者の選定が難しい。普段から意識的に養成しておくこと，一挙ではなく徐々に

導入する等の方策が必要であろう。

- チーフ・プログラマ以外のメンバのモラールを低下させる恐れがある …… チーフ・プログラマに全責任が集中することによって発生する“独裁”の弊害がメンバのモラール低下を招きかねない。具体的にはチーフ・プログラマの統制に対する反発，あるいは個々のメンバの功績，全体の功績が総べてチーフ・プログラマにすり替ってしまうのではないかという危惧等になって現われるであろう。
- 大規模なソフトウェア開発には適さない …… 小人数の編成を前提としているので，複雑・大規模なソフトウェアの開発には不適である。複数のチーフ・プログラマ・チームを編成する必要があるが，全体の作業の分割，チーフ・プログラマ間の調整（プロジェクト・マネージャの任務）等が新たな問題となる。

(2) スペシャリスト・チーム

チーフ・プログラマ・チームの変型と考えてよく，チームの責任はチーフ・プログラマにある。異なる点はチーフ・プログラマは全てのプログラムを作成し，外のメンバ（例えば，言語，論理，テスト，開発ツール，ドキュメンテーション等々の専門家）は各自の特殊専門能力に応じて作業を分担することである。即ち，必要とされる各分野の専門家を集めることによって，チーフ・プログラマの負荷を軽減し，その代りに，プログラミング作業に専念させようとする方法である。

本チーム編成の利点・欠点はチーフ・プログラマ・チームのそれと程度の差はあるが，ほぼ同様である。日本の環境では専門家がそれほど細分・特殊されていないこと，専門家を自在に集め，解散することが容易でないこと等があり，このスペシャリスト・チームの編成・運営は困難であると考えられる。

(3) 民主的チーム

チーフ・プログラマへの権限・責任集中とは対称的に，統制の不在，

メンバの平等を特徴としたものに民主的チームがある。これはWeinbergのEgoless Programmingの考え方にもとづくチーム編成であり，チームのメンバは平等の立場にあり，プログラムのモジュールは中核部の設計も分担し，相互に検討を重ねながらプログラムの仕様を決めていく方式をとる。このチームが適切に運営されるためには，メンバ間の階層関係があまりなく，意見が自由に出し合える状況が必要である。

チーフ・プログラマ・チームが強い組織力を持つのに比べ，このチーム編成は全体の統制力は弱いという特徴がある。チーム・リーダは正式には任命されず，開発工程毎にその局面に適任と考えられるメンバが一時的に非公式なリーダとなる。

民主的チームの長短はチーフ・プログラマ・チームのそれと丁度逆になると考えてよいであろう。要するに，ほぼ同等の能力を持った数人のプログラマが，中規模ではあるがロジック，内容が極めて困難なプログラムを，時期的には比較的ゆとりをもちながら開発するという場合に最適なチーム編成であると言えよう。強力なリーダ・シップが不在なことから，意見，見解が対立し統制がとれなくならないよう配慮されなければならない。

(4) 階層的チーム

チーム・リーダの下に上級プログラマ（通常，複数）を置き，その各上級プログラマの下に初中級プログラマ（複数）を配置する方式である。

階層構造が定着している日本の組織風土には馴じみ易く，特に意識せず事実上この編成方法が採られている例が多い。但し，この場合の階層は従来の単なる職制上の区分けではなく，プロジェクト全体の細分化と統合に見合ったものでなければならない。その点でこの組織は複数のチーフ・プログラマ・チームを有機的に統合したものとも考えられる。即ち，初中級プログラムが属する上級プログラマは分担された作業に関してチーフ・プログラマの役割りを果たすことになる。但し，チーフ・プ

ログラマとは違って，この上級プログラマの上位には，これら上級プログラマ（複数）を管理・統制するリーダがいるので全権限が付与され，全責任を負わされることはない。逆にリーダは全体の責任を負うが，チーフ・プログラマ・チームほど個々のメンバに対する管理・統制は直接的でなくなる。また，上級プログラマ同士及び初中級プログラマ同士の横のコミュニケーションも許される。

以上から，階層的チームはチーフ・プログラマ・チームの少人数による中規模までのソフトウェア開発という限界を脱し，むしろ大規模な開発に向いている反面，権限の分散に伴う全体の統制，調整等マネジメントの困難さが派生するという編成方法であると言えよう。

2.3 2. チーム編成の選択基準

ソフトウェアの開発に先立って，開発チームの構造を自由に選択できるという組織は少ないであろう。チーム編成の選択は，当該組織の特性・環境条件と対象ソフトウェアの特性の２つの視点から評価を行った上でなされなければならないが，ここでは後者のソフトウェアの特性とチーム構造の対応から選択基準を示す。

先ず，各々のチーム構造を管理構造及びコミュニケーション通路の点から図示すると図２－９のとおりとなる。

また，プログラミング作業の特性によるチーム構造の適否をまとめると表２－４のようになる。チーム編成方法の選択時の参考となるであろう。

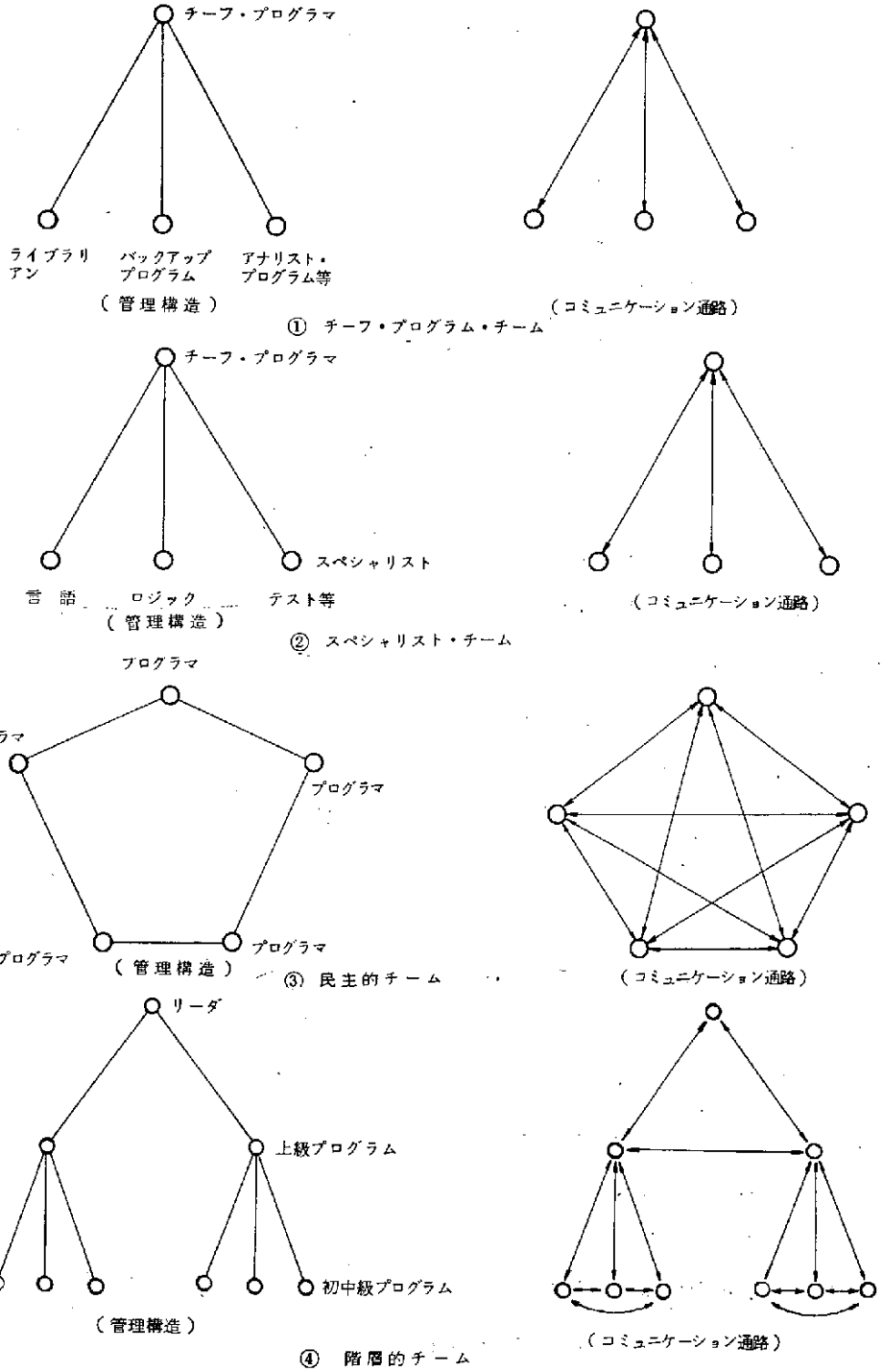


図２－９　プログラミング・チームの構造

表２－４　プログラミング作業の特性とチーム構造

プログラミング作業の特性 / チーム構造	困難性		規模		工期		柔軟性		信頼性		モラール	
	難	易	大	小	長	短	高	低	高	低	高	低
チーフ・プログラマ・チーム（専制的・集権的）		＊		＊		＊	＊		＊			＊
スペシャリスト・チーム（統制的・集権的）	＊			＊		＊		＊	＊			＊
民主的チーム（民主的・分権的）	＊			＊	＊			＊	＊		＊	
階層チーム（統制的・分権的）		＊	＊		＊		＊			＊		＊

2.3.3　チームの適正規模と効率性

プログラミング・チームはその規模が大きくなるに伴いコミュニケーションによるオーバーヘッドが増加し効率は低下する。また，図２－９で明らかのようにチームの構造によっても大きく変わる。ここでは構造の特徴が両極端であるチーフ・プログラマ・チームと民主的チームの２つについて，適正規模及びその効率を試算してみよう。

(1)　チーフ・プログラマ・チーム

一定期間，ｘ人の延有効稼動時間Ｆ(x)は次式で与えられる。

$$F(x) = H(x) - h(x) \quad \cdots\cdots (1)$$

Ｈ(x)：最大稼動時間

ｈ(x)：コミュニケーションに必要とされるオーバーヘッド時間

今，期間をａ，その間にチーフ・プログラマがメンバ１人とのコミュニケーションに費す時間比率をｋ（ｋ≦１）とすると，

$$H(x) = a\,x \quad \cdots\cdots (2)$$

$$h(x) = 2\,k\,a\,(x-1) \quad \cdots\cdots (3)$$

となり，Ｆ(x)は次のようになる。

$$F(x) = a\,\{\,x - 2\,k\,(x-1)\,\}$$

$$= a \{ (1 - 2k) x + 2k \} \quad \cdots\cdots\cdots\cdots \quad (4)$$

（図２－１０参照）

稼動効率Ｚは次式で与えられる。

$$Z = F(x) / H(x) = \frac{2k}{x} + 1 - 2k \quad \cdots\cdots\cdots\cdots \quad (5)$$

（図２－１１参照）

(4)及び(5)式から明らかになるようにチーフ・プログラマ・チームの場合，人数が増加する程，有効稼動時間は増加するが，稼動効率は低下する。但し，一般にｋは１よりかなり小さいので，その減少率は小さいと言える。

Ｆ(x)がｘの単純増加函数であるから人数の適正規模はこの式からは決まらないが，チーフ・プログラマの稼動には上限（その期間全部をコミュニケーションにかける場合）がある。ｘの最大値は次式から求められる。

$$a = ka(x - 1)$$

$$x = 1 + \frac{1}{k} \quad \cdots\cdots\cdots\cdots \quad (6)$$

今，仮りに１週間のプロジェクト（ａ＝４０時間）でコミュニケーション１つに要する時間を４時間（$K = \frac{1}{10}$）とすると，規模は１１人，有効稼動延時間は３６０時間，稼動効率は８１.８％となる。

以上は理論上の仮想値ではあるが，一般にチームの規模は１０人以下，ステップ数３～４万程度とされている。

(2) 民主的チーム

チーフ・プログラマ・チームの場合に比べ，人数の増加に伴ってコミュニケーションの数Ｃは増加し，それは次式により求められる。

$$C = \frac{x(x-1)}{2}$$

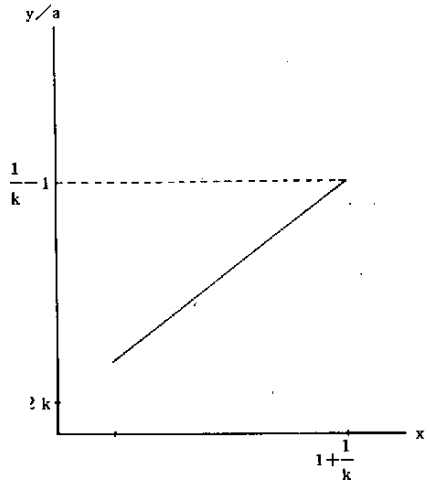


図2－10 チーフ・プログラマ・チームの有効稼動時間

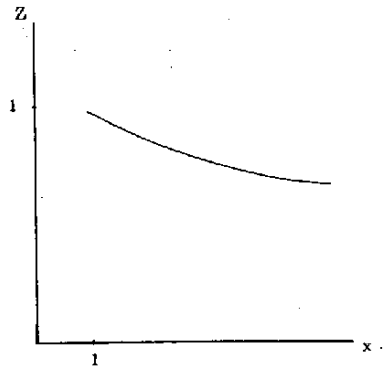


図2－11 チーフ・プログラマ・チームの稼動効率曲線

後は各々，チーフ・プログラマ・チームの式に準じて次のようになる。

$$F'(x) = H(x) - k(x)$$

$$H(x) = a\,x$$

$$h'(x) = k'a \times \frac{x(x-1)}{2}$$

$$F'(x) = \frac{a}{2}\{-k'x^2 + (2+k')x\} \quad \cdots\cdots (7)$$

$$Z = -\frac{k'}{2}x + 1 + \frac{k'}{2} \quad \cdots\cdots (8)$$

（図2－12参照）

(7)式から有効稼動延時間を最大にする人数X，その時の稼動効率Zは各々次のとおりとなる。

$$X = \frac{1}{k'} + \frac{1}{2} \quad \cdots\cdots (9)$$

$$Z = \frac{1}{4}k' + \frac{1}{2} \quad \cdots\cdots (10)$$ （図2－13参照）

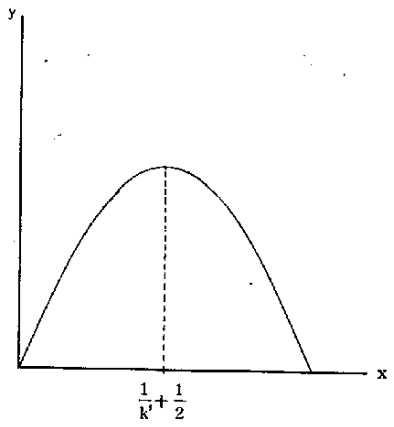


図２－１２　民主的チームの有効稼動時間

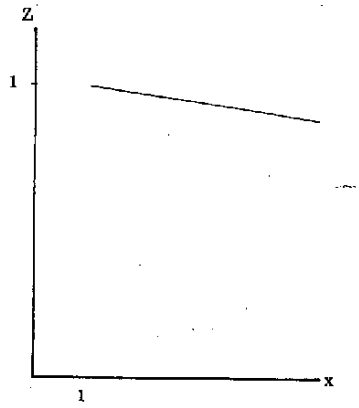


図２－１３　民主的チームの稼動効率直線

(9)及び(10)式に先のチーフ・プログラマの例と同じ$k'=\frac{1}{10}$とすると適正規模１０.５人，稼動効率は５２.５％となり，人数はほぼ同じで，稼動効率はかなり低下することが分かる。但し，両チーム構成においてコミュニケーションの質が異なるので$k=k'$とすることは無理であり，実際は$k<k'$となること，民主的チームの場合のコミュニケーション数はｘ人の間に同等に必要となることはなく，実際には$x(x-1)/2$より小さくなることを考慮しなければならない。

3. ソフトウェア開発プロジェクトにおける要員管理

ソフトウェア開発の生産管理の組織化，合理化を進めるべく，管理方法，組織体制及びいくつかの手法・ツールが工夫，提唱されていることは既に述べたところであるが，最終的にプログラムを作成するのは個々の要員であるという月並の結論に落ち着かざるを得ないのも事実である。

ソフトウェア開発における要員管理問題一般は，人事管理から技術管理，研修まで非常に幅広いものがあるが，本節ではプロジェクト管理の視点からポイントとなる事項に焦点を絞って検討してみることとする。

3.1 プロジェクト・マネージャの養成

プロジェクト・マネージャ養成の前提として，マネージャに要求される能力，資源を明確にしておく必要があろう。既に述べたように，プロジェクト・マネージャの役割・機能は非常に大きく，かつ困難であるため，必要される能力・資質も幅広く，高度なものとなる。

(1) 能　　力

プロジェクト・マネージャに要求される能力は，①問題は握能力，②技術能力，③マネージメント能力に分けられる。

① 問題は握能力

プロジェクト・マネージャの任務の大半はプロジェクト遂行上，内外に発生する問題の解決及び問題の予測とその対策にある。その意味で，プロジェクト・マネージャは問題の発掘，確定，予測等に関し，縦横な能力を発揮しなければならない。具体的には以下のようなものがあげられる。

- プログラム・アセスメント能力 …… ユーザ・ニーズの妥当性，問題の所在等を明らかにする能力
- プログラム・ファインディング能力 …… 問題の発生・所在を一早く

は握し，又は問題の発生を事前に察知する能力

- プロジェクト・フォーメーション能力 …… プロジェクト遂行上，環境条件，プロジェクトの内容及び問題の所在等が不明瞭な場合，これらを明確な形で定型化し，プロジェクトを具体化する能力

② 技術能力

技術能力は担当するプロジェクトによって異なってくるが，最新かつ高度な技術が幅広く必要とされる。即ち，技術動向を踏まえ，常に最新技術を吸収，適用できる能力，技術現状を前提に，必要とされる新たな技術を開発する能力及び必要な技術を既存技術から集積する能力等が要求されるであろう。

③ マネージメント能力

プロジェクト遂行上の全責任・権限が付与されているプロジェクト・マネージャに要求されるマネージメント能力は幅広く，高度なものである。このマネージメント能力は以下のように集約できる。

- 企画・管理能力 …… プロジェクトの方向づけ，各フェーズにおける状況変化への対応，判断・意思決定等を行う能力
- 指導・統卒力 …… 様々な分野からのメンバを統括し，調整，指導力によってプロジェクト遂行に向けて全体を統卒していく能力。
- メンバのモチベーション維持能力 …… 個々のメンバとの公式，非公式な接触を通じてプロジェクトの目的達成まで，モチベーションを維持する能力。そのためには，コミュニケーション能力，説得力及び個個の要員の適切な評価能力等が前提として必要とされる。
- 対外接衝能力 …… プロジェクト組織は通常，既存の機能別組織を横断する組織形態であり，プロジェクト遂行上，外部との連絡，調整が必要とされる局面は多く，これら"外乱"をさばく能力は重要な要素である。

以上の各能力に共通して幅広い知識，実務能力及び実践経験等が必要とされることは言うまでもない。

(2) 資　　質

プロジェクト・マネージャの資質（人間的側面）は①であげた能力を発揮させ，プロジェクトの目的を達成させるための根底となる人格形成，人間的成熟度という問題に帰着すると考えてよいであろう。その点では“望ましき経営者像”と大差ない。プロジェクト・マネージャに要求される資質を列記すると以下のとおりとなる。

- ◦やる気のあること（積極性）
- ◦発生した問題を迅速に処理すること（決断力）
- ◦内外の圧力に耐える精神力
- ◦事に当たっての冷静さ
- ◦柔軟な対応ができること
- ◦人間関係における積極性と抱擁力
- ◦協調性に富み，度量のあること
- ◦組織力があり，リーダ・シップがとれること
- ◦行動力のあること等々

なお，J. W. Hackney はプロジェクト・マネージャを個性が強く，プロジェクト業務の細部に亘って承知しており，何でも自分でやるタイプと自分では直接手を下さず，組織力，リーダ・シップを発揮して目的達成に統卒していくタイプの2つに分けている。前者は比較的規模の小さいプロジェクトのマネージャに適しており，後者は規模の大きい，内容の複雑なプロジェクトに向いているといえよう。

以上，プロジェクト・マネージャに要求される能力・資質を踏まえた上で，プロジェクト・マネージャの行動規範を5つの鉄則にまとめておこう。

① 誤った前提（リソース，他部門の協力，サポート等）に立ってプロジェクトを遂行しないこと。

② 致命的な報告機能を欠如しないこと。

③ メンバに任せた方が効果的な問題を自分で解決しようとしないこと。

④ 既に解決された問題，同じような問題に取り組まないこと。

⑤ 不必要な，歓迎されない口出し，手出しをしないこと。

(3) 養　　成

プロジェクト・マネージャには高い技術力，マネージメント能力，更には人間的資質が要求されるとならば，その養成は極めて困難であることは明らかである。幅広い知識及び様々な分野における実務経験が必要とされることからも，その養成期間は長期に亘らざるを得ない。養成期間は次のような段階に分けられるであろう。

◦第 1 期 …… 自分の専門技術・能力を習得する。

◦第 2 期 …… 専門外の技術・知識を広く習得し，基本的なマネージメント能力を身につける。

◦第 3 期 …… 実際のプロジェクトに，メンバとして参加し，実践的経験，問題解決の能力を培かう。

◦第 4 期 …… プロジェクト・マネージャのアシスタントとして，視野を広げプロジェクト・マネージャとしての修練を積み重ねる。

プロジェクト・マネージャ養成方法の基本的な考え方は，キャリアパスにもとづいたジョブ・ローテーションとＯＪＴ（On the Job Trainning）を中心とし，その補助としてのＯＦＦ－ＪＴの 2点である。

ジョブ・ローテーションは一般によく言われている割には実行が困難な問題であるが，有能なプロジェクト・マネージャを養成するためには，計画的かつ系統的なローテーションが必要とされる。

ＯＪＴについても，計画的，意図的に実施しないと，雑務的な補助仕事

に終始し教育的効果があがらないことになる点もよく指摘されているところである。

ＯＪＴによる実践的な訓練と併行して，ＯＦＦ－ＪＴとして，外部機関で実施される専門技術研修及びマネージメント研修等も必要であろう。アメリカでは宇宙開発プロジェクトのために必要なプロジェクト・マネージャの養成を図るための研修用シミュレーション（ゴダード宇宙センターのＧＲＥＭＥＸ，ＮＡＳＡのＰＬＡＮＥＴ等が有名である）が開発され，適用された例があるが，我が国ではプロジェクト・マネージャ養成を目的とした専門教育，シミュレーションによる訓練は未だあまり実施されるには至っていないようである。

3.2　プロジェクト・メンバの管理

プロジェクト組織におけるプロジェクト・マネージャの人事管理の内容・方法は，従前の機能別階層組織の管理者のそれとは異なるところがあり，その点が正にプロジェクト・メンバの管理を困難にしている点である。

(1)　要員管理の困難さ

プロジェクト組織における要員管理を難しくしている理由のおもなものは次のとおりである。

①　プロジェクト要員は，それまでに配属されていた組織から離れ，従来の階層関係とは必ずしも一致しないプロジェクト・マネージャに，いわば一時的に配置されるので，プロジェクトに参加したことによる処遇上の問題に不安を感ずる傾向にある。

②　従来の機能別組織に所属していた時と，仕事のやり方，管理のされ方，人間関係等がかなり異なるので，戸惑ったり，落ち着くまでに時間を必要とする。特に，先に述べたように機能別組織－職制型及びマトリクス組織においては，二人の管理者から管理されるので，要員の困惑，不安感は大きいものとなる恐れがある。

③ 各メンバは各々の専門分野を持っているが，プロジェクト・マネージャがそれら全部に関し，概念的にも，技術的にも習熟していることは稀であろう。その結果，要員との意思疎通に欠け，要員の士気に悪影響を与え，ひいては要員の評価，プロジェクトの遂行・管理上にも不都合なことが起こりかねない。

④ プロジェクト完成期限に向けて，各作業の日程計画，要員の配置がなされるはずであるが，一般に，当初計画通り進行しないことが多く，その結果，特定の要員又は一定期間の全要員に対し，オーバー・ロードを課さざるを得ないことになる。特に作業が細分され，それを各々の専門家が担当し，しかも各作業が密接に関係しているような場合，或る一部分の作業の遅れが全体に影響することが分かっていながら，他の要員を応援に当てることが困難であるという問題に直面する。

(2) 要員管理上の留意事項

プロジェクト・メンバの管理に関しては，実際の場面で様々な問題があるということは既に実態調査結果からも明らかであり，それら問題に個々に応えることは本書の範囲を越えると思われる。一言で言えば，3.1で述べたような，能力及び資質を有するプロジェクト・マネージャが適切な方法で管理するということに尽き，マネージャとして留意すべき点も明らかにしたので，ここではプロジェクト・メンバに対する仕事の割り当てについて留意事項をまとめてみる。

プロジェクト・メンバに作業を割当てる際に留意すべき原則は次のとおりである。

① 個々のメンバの能力と割当てる作業が対応していること。特定のメンバの能力が遊ぶことになったり，能力以上に割り当てることがないよう配慮しなければならない。

② 1人に割当てる作業が複数の場合，作業分担が便宜的であり，各作業に関連性が見当らないような状態は避けること。

③　②と背反することにもなるが，特定のメンバに作業負荷が集中しないようにしなければならない。

④　②及び③を前提とした上で，プロジェクトの作業のうち，最も重要な部分はプロジェクト・マネージャからみて最も信頼できる要員に割当てなければならない。

⑤　作業を割当てられた要員が，それを遂行するためには既存の能力・知識だけでなく，自らスキル向上が必要とされる部分が存在するよう工夫する。

⑥　⑤に関連するが，各要員が士気を維持し，意欲的に取り組むことができる様，配慮する必要がある。

⑦　作業の分担によってメンバ間の団結，人間関係に支障を来たすことがないようにする。

3.3　メンバ間のコミュニケーション

プロジェクトにおける要員管理上の方策として，コミュニケーションの充実が重要であり，広く実行されていることは今回の実態調査からも明らかである。

一般に，コミュニケーションの問題はフォーマルなものとインフォーマルなものに分けられ，視点としては能率と士気（モチベーション）の2つがあるが，日本的組織風土においては，インフォーマルなコミュニケーションを士気の点から重要視する傾向にある。

(1)　コミュニケーション・ネットワークの分類

プログラム開発チームのコミュニケーションの構造による効率性の差異については，先に述べたとおりであるが，同じような問題に関し社会心理学の立場からの研究・実験結果（バーベラス，リービッド）を紹介しておこう。

コミュニケーション・ネットワークの型を中心と周辺の分化程度によっ

て分け，それぞれの型によって，能率及び士気にどのような違いが発生するかを実験したもので，概要は図３－１及び表３－１のとおりである。

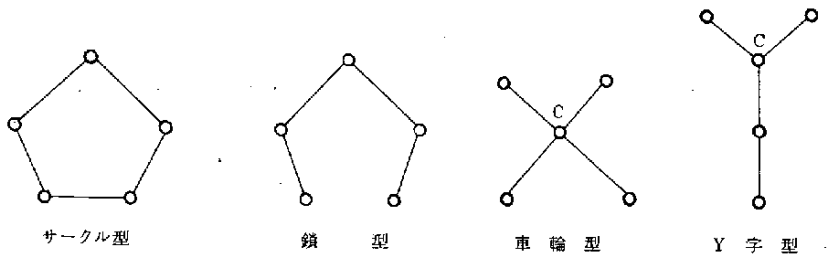


図３－１　コミュニケーション・ネットワークの型

表３－１　コミュニケーション・ネットワークの分類

		課題が単純な場合		課題が複雑な場合	
		士　気	能　率	士　気	能　率
分散型	サークル型 鎖　型	○	×	○	○
集中型	車輪型 Y字型	×	○	×	△ 中心人物Cの能力に大きく依存する

分散型は民主的チーム，集中型はチーフ・プログラマ・チームに各々，ほぼ対応しており，実験結果もプログラム開発チームの場合の議論におおよそ沿ったものとなっていると考えてよいであろう。

しかしながら，実際のプロジェクト組織においては上のようなフォーマルなコミュニケーションの構造にインフォーマルなコミュニケーションが錯綜し，複雑となることは言うまでもない。また，プロジェクト組織においては，プロジェクト・マネージャを中心として，図３－１に示したネットワークのうちのいくつかが結合され，それらネットワークの型は各々の

チームが分担した作業の量・質に応じて異なるものになるであろう（図3－2参照）。

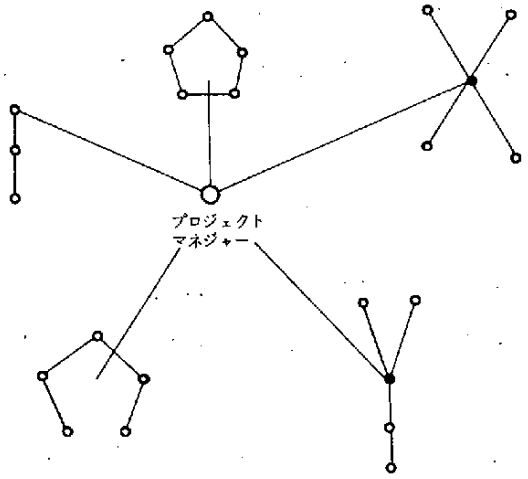


図3－2　プロジェクト組織におけるコミュニケーション

(2)　コミュニケーションの効率化

プロジェクトの大規模化に伴い作業をプロジェクトのメンバ又はチームに分担，割当てをしなければならないが，その分割に当たっては作業間のインタフェースを極力少なくするように配慮する。実際問題として，インタフェースが必要となることは避けられないが，その場合でも，プロジェクト・マネージャと要員間のコミュニケーションで解決ができるようにしておき，作業を分担した個々の要員又はチームは出来る限り，互に独立して作業を遂行できる方が効率的であり，ひいては工程管理の容易化，品質の確保にもつながるはずである。

全体の効率はコミュニケーションの数のみではなく，コミュニケーションのやり方によっても変わってくる。一般に，コミュニケーションを効果的・効率的に行うためには以下の5原則を守る必要がある。

① 情報は必要最少限のものを，必要な要員にのみ伝達すること。

② 特に重要な情報は文書で伝達すること。

③ 文書による伝達の場合はその内容が肝要で，かつ理解し易いものであること。

④ 誤解が生じないように伝達すること。

⑤ 会議による伝達の場合は，資料の事前配布，時間厳守，議事要旨の確認・文書化等を心懸けること。

3.4 モチベーション

プロジェクト・マネージャの任務のうち，最も重要なものは，プロジェクトの目的，即ち，可能な限り最良質のものを所定の期間，費用の範囲で完成することに向けて，プロジェクト・メンバの能力を引き出し，そしてメンバ各々を動機づけ，それを維持することにある。日本の経営組織においては，インフォーマルなコミュニケーションやファミリ・スピリットでカバーできる部分が多いのでモチベーションの問題を正面から取り組むことにはあまり熱心でないという傾向がある。しかしながら，プロジェクトの使命及びプロジェクト・マネージャの任務からみて，プロジェクト・メンバのモチベーションは避けて通れない問題である。

(1) 基本的な考え方

作業能率を最初に科学的に研究したF. W. Taylor の科学的管理法は古典的研究として著名であるが，その根本は作業環境の整備が要員の能率に大きく作用するというものであった。しかしながら，E. Maeyo によるホーソン工場の実験結果から，作業環境と作業能率は相関がないという結論が出され，その理論的説明として，個々の要員のモラールが作業能率を決める最大のファクタであり，それを高めるための動機づけ理論が注目されるに到った。

その後提唱されたこの動機づけに関する理論的研究として有名なD. McGregar のX－Y理論及び心理学者A. H. Maslow の欲求五段階説

の概要を紹介しておこう。

① X－Y理論

マグレガーは人間の心理，行動のパターンを伝統的な見方（X理論）と潜在能力を見る見方（Y理論）に分けた。X理論から見れば人間は本来，怠け者であり，自己中心的であるから組織の要請には無関心であり，マネージャの命令・干渉がないと働かず，直接的，物質的利益がないと動かないということになる。これに対し，Y理論から見れば人間は本来行動的であり，条件次第では責任を引き受けたり，積極的に働くものであり，その関心は必ずしも物質的利益のみではないということになる。

このX－Yは表面的には性悪説－性善説，本音－建て前とに対応させることができるであろう。しかしながら，人間はXが本音であり，Yは建て前にすぎないという説に対し，適切な動機づけがあれば人間はY理論的な行動をとるものであるというのがマグレガーの主張である。

Y理論を性善説と考えるとプロジェクト組織において，マネージャの要員管理は必要なく，放任しておけばよいということになりそうであるが，性善説は元々「人間は生来，善なるものであるが，社会の中で悪を身につけていくものである」という思想であり，マグレガーも，人間の潜在的能力に対する適切な管理，動機づけ等の働きかけによって，意欲を高め士気を上げることが可能であるとしている。

② 欲求五段階説

マズローは人間の欲求を5つに分け，それらの成長過程を明らかにした。即ち，人間の欲求は最も基本的な次元である，生命・生理的欲求から安全・安定の欲求，社会的欲求，自我の欲求及び自己実現の欲求までの五段階に分けられ，各々は各人の心理的成長につれて，高い次元に移行するとしている（図3－3参照）。

(2) モチベーションの診断・評価

1970年代半ばにJ. R. Hackman（イリノイ大学）とG. R. Oldham

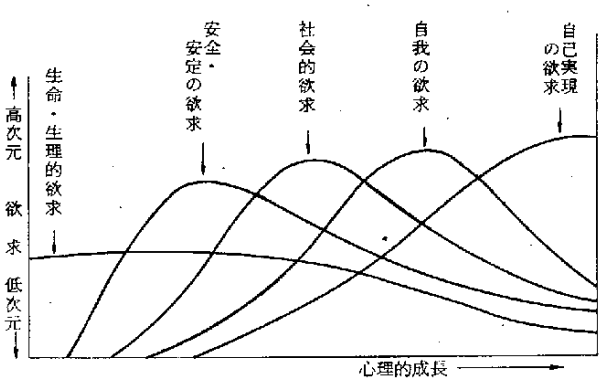


図３－３　マズローの欲求五段階説

（エール大学）によって開発された職務診断調査（Job Diagnostic Survey：ＪＤＳ）手法を，コンピュータ要員のモチベーション測定に適用した。J. D. Couger と R. A. Zawacki （コロラド大学）による調査結果を紹介しておこう。

ＪＤＳではヒューマン・モチベーション・モデルとして，作業と自分の経験，作業結果に対する責任及び作業に関する知識の３つの要素をあげており，これらが総べて適正に充足されれば高いモラール，高品質の作業パフォーマンス，高い満足度が得られ，欠席率及び退職率が低下するとしている。

実際の調査は要員に質問票を配布し，各質問事項に１～７のスコアをつけてもらい，同一作業に従事しているグループの平均点を算出する方法を採る。評価尺度は成長要求の強さ，社会的要求の強さ及び動機づけ可能性の３つである。調査結果から導き出された主な結論は以下のとおりである。

① プログラマ及びアナリストの成長要求は非常に高い。

特にプログラマは最新の技術 ― ハードウェア，オペレーティング・システム，言語等を使って仕事をしたいという要求が最高得点となっている。このことは，プログラマやアナリストが高度な個人的発達を求め

ており，彼等の職務がそれに応えなければ，モラールは低下するということを意味している。

② プログラマやアナリストの社会的要求は極めて低い。

プログラマやアナリストの社会的要求は，他の５００種の職業に比べて最低点であるという結論が出ている。Cougerは，これは彼等が人づき合いがきらいであるということより，意味のあるミーティングには積極的に参加するであろうが，実際には彼等の要求がうまく組織されず，能率的に運営されていないグループ活動に耐えられないためであるとしている。但し，留意しなければならないのは，プログラマの場合は社会的なかかわり合いが少なくても比較的問題はないが，システム・アナリストはユーザとの広範なかかわり合いによって仕事をしなければならない点である。この測定結果からすればシステム・アナリストはユーザとのかかわり合いを出来る限り避けようとするか，止むを得ない場合でも必要最少限のものにしようとすることになる。このことは，ユーザ要求の調査が不完全であるという理由の一部を説明するものではないか。

③ 今後，要員のモチベーションの上から問題となるのは，プログラム・メンテナンス及び開発スタッフとユーザとの間のインタフェースである。これら２点については日本においても既に現実的な課題になっているものであるが，要員のモチベーションの点からも大きな問題となることが実証されたことは注目しなければならない。

(3) 動機づけの原理

一般にモチベーションが「組織において人々に仕事を始めさせ，続けさせる各種の強制力の合成」（R. Dudin）であるとするならば，プロジェクト・マネージャが自分のリーダ・シップによって個々のメンバに方向を与え，彼等の士気を全体の最終目標に向け統合し，維持することということができよう。大幅な生産性向上をもたらすことができるのは，技術ではなく個々の要員であると言われているソフトウェア開発では，マネージャ

が如何にすれば要員を動機づけられるかは大きな問題である。

モチベーションは外的なものと内的なものに分けられ，前者は金銭等による報償，後者は要員の精神的，心理的ニーズの充足によるものである。Le Duc によればプログラマの場合「外的モチベーションは多分に信頼できない。内的モチベーション，即ち，プログラマを特徴づける本質的な属性の探求は若干の望みがある。そのためには，マネージャは仕事そのものによって動機づけられる雰囲気，達成と挑戦を高く評価する雰囲気を確立し，それを助長する必要がある。」ということになる。

内的モチベーションの高揚方策としては極めて抽象的ではあるが，外的モチベーションの扱いは，我が国においても実態調査結果において具体的報償制度があまり高く評価されていないことに符合しており興味深いところである。

内的モチベーションを高める要素は，J. C. Lillis によれば，「人々の心の中にある感情的刺激を受け易い欲望を理解すること」と「人々の衝に刺激を与えるために，人々との意思疎通を保つこと」の2つである。

動機づけの要因をもう少し具体化したものとして，行動科学の F. Herzberg は次の6項目をあげている。

① 達　成
② 承　認
③ 評　価
④ 成長の可能性
⑤ 仕事の内容そのもの
⑥ 昇　進

最後に，プロジェクト・マネージャが要員を動機づけ，モラールを高め，維持するために考慮すべき，より実践的な原則をまとめておこう。

① 目標，実情，条件をありのままに，明白に表明すること。

② ミーティング等において“私”ではなく“我々”という言葉を使って参加意識を刺激するごと。

③ 要員に対し信頼感を持ち，それを常に具体的な言動に表わすこと。

④ 評価・判断は公正であること。

⑤ 要員の反応を常にあらかじめ予測し，対策を持っていること。

⑥ 個々の要員の得手・不得手を見極めておくこと。

⑦ 競争意識を不必要にあおり立てないこと。

⑧ 人気取りに陥することなく，威厳をもって対処すること。

⑨ 要員の意見をフォーマル，インフォーマルの両面から，は握に努めること。

⑩ 個々の要員の裁量範囲を極力拡大し，責任を持たせること。

第Ⅱ部　マルチ・プロジェクトの管理

ソフト・ウェア開発におけるプロジェクト管理に関するものの定義はあるが，範囲に関してはあまり明確にされたものがない。単独のプロジェクト管理を意味する場合と，複数のプロジェクトの管理を意味する場合ではその内容は大きく異なるのは当然であるが，どちらもプロジェクト管理の範囲にある。

一般的には単独プロジェクトの管理を指していることが多いと思われる。詳細については，第Ⅰ部プロジェクト管理の方法に述べられているので，ここでは単独プロジェクトの管理だけではなく，組織全体の立場から管理目標を設定し，これに則った優先順位づけ，資源の配分などを行うための手続き，つまりマルチ・プロジェクト管理の視点からシステム開発計画及び稼動システム評価制度とシステム費用，効果のは握方法をA社の事例を通じて紹介する。

1　システム評価制度

1.1　マルチ・プロジェクトの管理と評価制度

組織規模がある程度大きくなると，組織全体としての計画管理は必須の条件となってくる。仮にこれを行わないとすれば，おそらく組織機能が低下して，このために生じる問題から生産性の低下を引き起す恐れがある。例えば作業負荷の格差増大，コンピュータのキャパシティ管理が後追い的になる等組織としての機能に問題が生じる。

システム部門のマネージャは，少くとも年1回（年度計画）又は2回（半期計画）組織全体の計画に参画して，他チーム（又はシステム）の進行状況等の情報交換が必要である。こうすれば，優先順位，資源配分等の調整，決定に関する情報を担当するプロジェクト管理に有効に活かし管理の柔軟性，総合性が発揮できる。

A社では全体の計画策定を以上のような目的，理由から年1回行い上期末に下期計画，予算の見直しを実施している。

一方個別的プロジェクトの管理は，システム評価制度を適用して行っている。

1.1.1　システム開発評価の目的

システム開発の評価は，開発過程の諸段階における計画・検討立案内容を組織機能にもとづいて検討し，プロジェクト運営管理面の充実強化を図るものである。

その主たる狙いは以下の通りである。

① システム開発過程における管理の諸段階を開発手順として明確にし，システム開発管理を強化充実する。

② システム開発の各段階において方針と計画及びそれにもとづく実績を組織的に承認・確認する。

③ 特に各開発ステップの計画時点に組織の意思を反映する。

④ システム開発による効果の最大化と費用の最少化を実現する。

ここでのポイントは，開発ステップを標準化することである。

即ち個別のプロジェクト管理の目的の多くは，ソフト製品を決めた納期にきちっと間に合わせることであって，工程管理の要素が大きいところである。このため開発ステップの標準化は絶対不可欠で，システム開発評価制度の土台になっている。

各ステップの間に，システム評価の概念図（図１－１）で示すよう評価のステップが設けられ原則として，前工程が終わらないと次工程へ進まないことになっている。これのもつ意味は，各ステップ毎にドキュメントを整理すること。構想立案段階又は基本設計段階は特にユーザ部署長の承認と，次ステップへの要望・意見を求めこれをシステム設計に反映すること。またこの段階でユーザ部署長，システム担当部長協議で開発を中止することもできる。といった内容が評価制度として組み入れられている。

システム稼動後のシステム維持に当たって詳細設計レベルのドキュメン

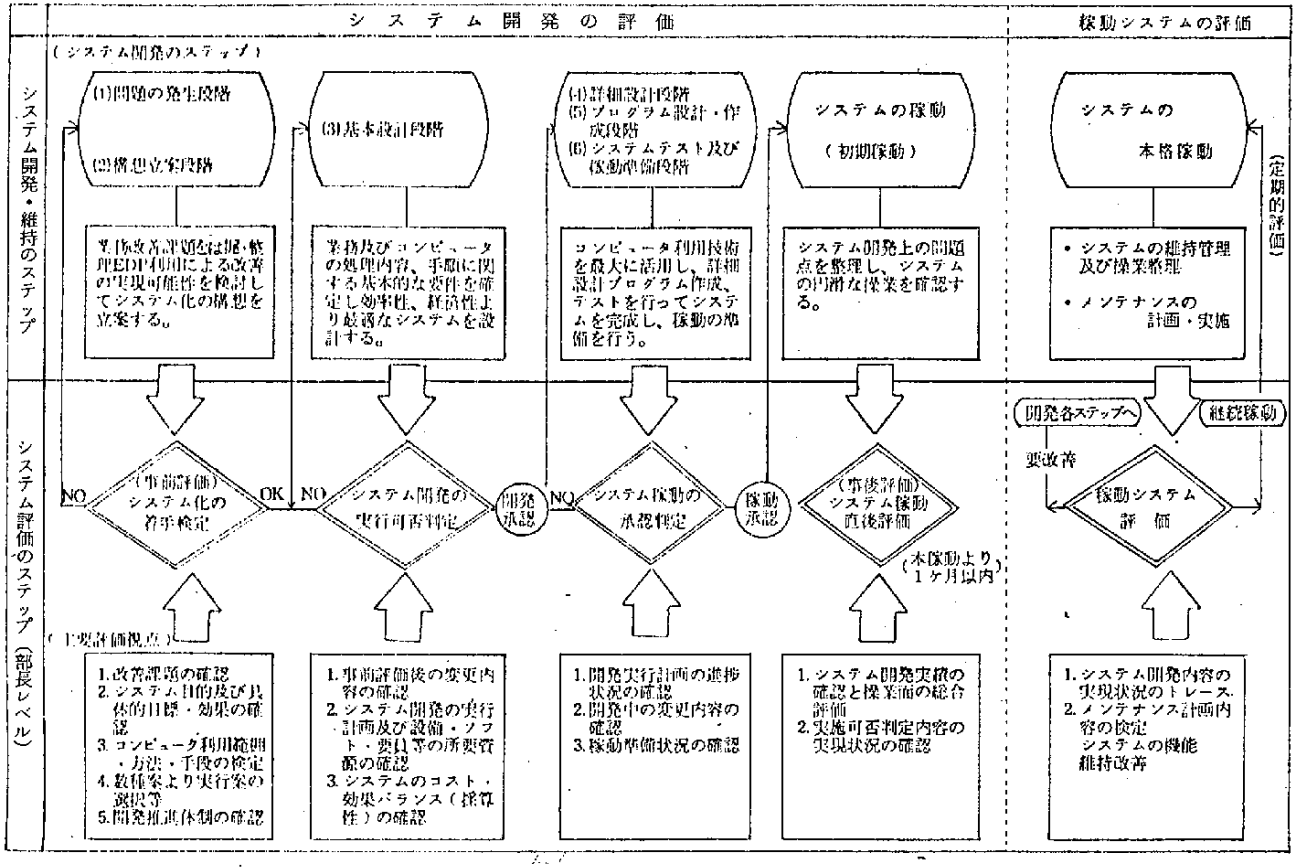


図１－１　システム評価の概念図

トは充分整備されているが，往々にしてマネージャが必要とするシステム管理に要するドキュメントが欠落しているケースがある。

しかしシステム開発評価制度を適用するようになってから，管理レベルに要するドキュメントの整備状態が良くなった。構想立案，基本設計段階は，開発するシステムの狙い，目的等と様々な前提条件，関連性その他必要な諸要件が記述されるもので，稼動以後の管理には非常に重要なドキュメントである。

これからのシステムは増々複雑化・多様化する上，管理レベルのシステム化が多くなってくるので，この段階のドキュメントの重要性は一層高まるものと思われる。

システム評価制度の適用実施は昭和５２年度からであるが，それ以前に開発して，このようなドキュメントが整備していないシステムについては５２年，５３年の２年間で整備するよう計画的にドキュメント整備運動を行ったので現在は殆んどのシステムに，このレベルのドキュメントが整備されている。

1.1.2　稼動システム評価の目的

稼動システムの評価は，現在稼動しているシステムの現状は握と機能維持，改善に関する評価であり，システム維持管理の充実強化を図るものである。

その主たる狙いは以下の通りである。

① 稼動システムの現有機能の確認

② 稼動システムの有効性の評価と継続承認

③ データ処理の効率向上とコスト低減

④ システム維持管理業目標の明確化と組織承認

なお，評価結果は利用部署と共同して全社的にオーソライズする。

ここでのポイントはシステム開発で設定され，ドキュメントに記述され

ているシステムの目的・諸前提条件が変っていないか，システムが果すべき機能は充分発揮しているか，使い勝手が悪くなっていないか等そのシステムが持つ目的，機能を定期的に要求面，サービス面両方を点検することにある。このため当然のことであるが，システムの目的，達成すべき目標が明確に設定されていないと稼動システム評価が不完全になってしまうので開発段階の責任は重大である。

稼動システム評価を定期的に実施することにはもう1つ別の大きな狙いがあり，その結果がシステム担当部署にとって大きなメリットになっている。それは年1回であるが定期的なユーザとの公式的交流の機会となって，現有システムに関する問題，今後のシステム開発に関すること等，双方の部長が話し合うことが出来るようになったことである。

従来は何か特別のことでも生じない限り本社部長クラスがシステムの会議に出席することはなかったのである。先にも述べたように，部長管理領域のシステム化を指向するような時代となってきた現在，各部長クラスが担当部署の業務の質的向上或いは効率化を考えることは必要であり，当然のことでもある。

1.1.3　システム評価制度の前提主要条件

システムの開発手順（ステップ）は従来より，開発のスケジューリング及び進捗管理等に運営されて来ているが，その運用実態は全体的に不統一で特にシステム開発管理の観点から見ると，ステップ区分内容にかなり曖昧な点が多い。今後，更にシステム開発を組織的に効率よく管理推進するために，各ステップの定義，なすべき事柄を基本的に整理し，単なる開発技術的ステップとしての観点のみではなく，共通の管理ステップとして位置付け，開発管理並びに「システム開発評価」の基盤としている。

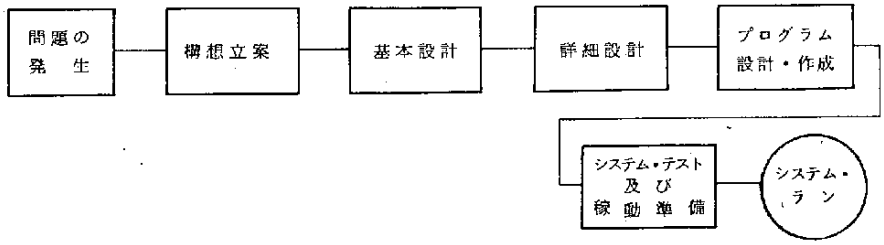


図１－２　システム開発手順

なお，図１－２に示す各開発ステップの定義と設定趣旨を以下で説明する。

(1)　問題発生段階

開発ステップの定義	設定趣旨
各業務部署における業務運営管理上の問題が認識され，システム化の必要性が提起される段階	問題の発生提起の段階であり提起された課題にもとづき，構想立案を行うために，情報システム部が組織的に参画し，問題解決を図る段階と区別するために設定

(2)　構想立案段階

開発ステップの段階	設定趣旨
問題の形成と解決方向を整理し，システム構想を立案する段階	システムが基本的に要求される諸要件を確認し，その対処方向を多角的に検討して，実現可能性を踏まえた最適なシステム構想を築くステップであり，基本設計に入

	いるにあたりシステムの方針を設定する段階として設定

(3) 基本設計段階

開発ステップの定義	設定趣旨
人的組織及びＥＤＰの相互機能に関する基本的内容の確立を図り，各々の効率性経済性を最大に発揮する実現可能なコンピュータ利用システムを設計する段階	システム化を実現するための要件を具体化し，効率性経済性を踏まえて業務及びコンピュータの処理内容手順等を確定する実行ベースでのシステムの基本設計段階である。 コンピュータ処理技術面をベースとした詳細設計段階と区別する。

(4) 詳細設計段階

開発ステップの定義	設定趣旨
業務及びコンピュータ処理の具体的な詳細実行内容を最終的に確認し，コンピュータの利用効率を基本として，技術的な視点より最適なデータ処理設計を行う段階	人とコンピュータ処理の各々の実際的処理内容を組織的に最終確定するためシステム運営上の例外処理，データ保護等を含めた実態を想定し，相互間の調整確認を行うシステム設計の最終段階である。併せてコンピュータ処理技術の最大活用を図りプログラム開発，データ処理，メンテナンス等の効率

	化を織り込んだプログラムの全体構造を設計確定する段階として設定

(5) プログラム作成・作成段階

開発ステップの定義	設定趣旨
プログラムの全体構造設計にもとづきプログラミング技法を中心として，個別プログラムの処理手順を作成する。これと共に，コーディング個別テストを経て効率的かつ正確なプログラムを作成する段階である。	詳細設計段階時に，確定されたコンピュータ利用に関する実行設計内容の単なるプログラム言語への転換ではなく，現実有効に機能するプログラムへ展開するため，プログラミング技術を最大に駆使して効率性，正確性，簡便性等を考慮したプログラムを構造的に設計具体化する段階として設定

(6) システム・テスト及び稼動準備

開発ステップの定義	設定趣旨
作成した個別プログラムについて，一貫したシステム・テストを実施し，設計した初期目的に対して正確に機能するコンピュータ利用システムの完成を図ると共に円滑なる本稼動実施のための諸準備作業を展開する段階である。	システム・テストを通して具体化されたコンピュータ利用システムの全体に対して正確性，効率性等を最終的に確認し，更にシステム運営管理上の諸問題をは握して本稼動の実施の諸条件を整備する段階として設定

1.2 期間活動計画策定要領

期間活動計画には，次の各計画が包含されている。

① 業務計画

システム開発・大型メンテナンスのスケジュールをシステム開発の各段階毎に線表で記入する。

② 設備計画兼操業計画策定

業務単位（コンピュータ処理単位）にＥＤＰ関係設備の使用する量を要求するため，システムの稼動時期に合せて各設備の使用計画，オペレーションのスケジュール，伝送のスケジュールを数量又は線表で記入する。

③ ソフトウェア外注作業要望

通常プログラミング作業は外注化しているので，工数×@で求めた金額（予算）で月単位の外注計画を記入する。

④ 本社電算機稼動時間予想データ投入

既稼動，新稼動システム総てのコンピュータ稼動時間を月単位に記入する。（既稼動システムについては，前期実績，年度実績をコンピュータよりアウトプットし，これを次期の予定時間の参考データとして用いる。新稼動する予定のシステムについては，プログラムの構造・ボリューム及びデータ量等類似したシステムを過去データから探してこれを参考にして予定データを作成し記入する）。

⑤ 要員計画

現状の要員数と今後システム開発等に要する要員数を係長以下男女別に人員数を記入，また外注要員（メンテナンス要員）の現状と計画を記入する。

⑥ その他

以上期間活動計画を構成する諸計画を個別に説明したが，各用紙をサンプルとして巻末につけたので参照されたい。

業務計画と要員計画については翌年度計画と向う４ヶ年の計５ヶ年分の計画をたてることにしている。毎年５ヶ年分の計画をたてる，いわゆるローリ

ング・プラン方式になっている。

しかし翌年度の計画が実行計画として具体的に求められ，予算面との斉合性が整ってないと問題になるので，業務計画を策定する前に，予算面，要員面で次年度計画立案の方針なり，大枠的なデータを夫々の経理・人事部と接衝し，予算，要員，設備，外注各計画立案の前提条件を設定してから計画立案作業を始めることにしている。従って当然，業務計画をまとめる過程では優先業務の選定・スケジュールの調整といった内容の計画立案業務が伴ってくる。これは前年末乃至当年始めに計画立案方針の設定一計画起案依頼の中で，次年度計画立案に当たっての具体的重点業務の範囲等に関し，部長方針と斉合性を整えた計画の指針といったものを前述の前提条件と併せて，開発担当部署に示す。

これらの作業は総て，企画調整を担当するスタッフの仕事である。但しスタッフは直接業務計画立案段階での業務間の調整はノータッチである。調整的な範囲は，業務優先順位の設定の考え方なり，前提条件となっている予算，要員数の調整といったところに止どめ，開発担当の範囲での調整分野には触れないことになっている。室内或いはスタッフ間との調整で整理がつかない場合は，諸計画調整の段階で部長裁定を求めるが，殆んどは開発担当とユーザとの接衝で整理出来ている。

次年度計画の内訳をみると前年度よりの継続分が約６５％，残り３５％が新年度着手分で予算面等に直接響く，プログラミング～システム立上がりの計画は大部分が継続業務で，予算・要員に関してかなり精度の高い見通しがある。また，新年度着手分でこのステップに到るのは比較的小規模のシステムでしかも下期に出てくるケースが多いので調整の余裕がある場合が多い。従って，システム開発計画において構想立案あるいはその前段のフィジビリティ・スタディ的段階で要員乃至時間の調整を事実上行い得る場合があるので極端な切落しを行うケースは１０年位続けているが発生したことはまだない。

次に期間活動計画の策定フローを図1－3に示す。なお，参考迄にA社のシステム担当部署の現状組織・機能を表1－1に示す。

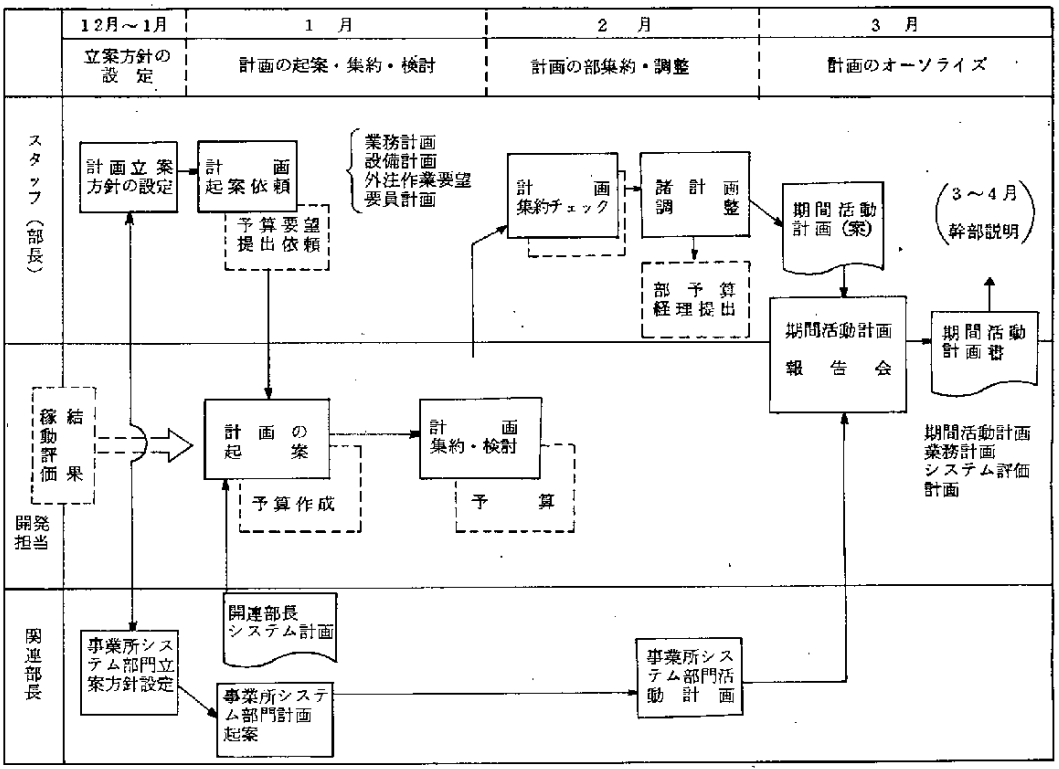


図1－3　期間活動計画の策定フロー

表１－１　システム担当部署の現状組織・機能

システム部　（合計　175名（38名））

全社業務機械化計画の立案・総合調整及び機械計算作業を行い各部活動の円滑な遂行に協力する。

- 計　画　室　（1）

  システム化計画の全社調整及び利用技術・設備についての研究開発教育指導並びに本社ＥＤＰＳの操業管理に関すること。

  - 企画調整班　（10（3））

    業務計画・システム評価・ＥＤＰ設備・外注・予算・教育（内外）等の企画，運営，調整

  - 技術開発班　（9（1））

    利用技術の調査・研究・標準化と汎用ソフトの開発（購入），維持管理

  - 機械計算班　（14（6））

    電算機の授業管理とＤＳサポート

- 本社システム室　（1）

  全社管理部門及び各事業部本社業務の新規システム開発並びにシステム維持管理に関すること。

  - Ａチーム　（21（5））
  - Ｂチーム　（19（4））
  - Ｃチーム　（15（4））
  - Ｄチーム　（9（1））

- Ｃ事業所システム室　（1）

  Ｃ事業所における電子計算機及びその関連技術の研究開発・新規システム開発，システム維持管理並びにＥＤＰ操業に関すること。

  - Ａチーム　（17（3））

Bチーム （13（1））

Cチーム （11（1））

Dチーム （16（2））

管理班 （5（2））

本社企画調整班と連繋して，業務計画・システム評価・ＥＤＰ設備・外注・予算・教育（内外）等の企画，運営，調整

機械計算班 （12（5））

本社技術開発班と機械計算班に連繋して，利用技術の調査・研究・標準化と汎用ソフトの開発（購入），維持管理及び電算機の操業管理とＯＳサポート

備考：① （）中の数は女性数で内数

② 各組織名称は仮称

③ Ｃ事業所システム室は，本社組織でＣ事業所に駐在

1.3 評価制度の体系

1.3.1 システム開発評価の体系

(1) 基本的な性格

システム開発ステップ（問題の発生，構想立案，基本設計，詳細設計，プログラム設計，作成，システム・テスト及び稼動準備の各開発ステップを設定）単位の計画検討立案内容に関し，各開発ステップの性格に応じた検定を行うことによって，組織的運用にもとづく開発プロジェクト運営管理面の強化を図るものである。当該評価は検討立案された内容に関し，開発プロジェクト終了時において最終検定確認を行う事後評価的な側面と，次開発ステップに入いるに際しての基本方針又は前提条件について検定確認を行うという事前評価的な側面の双方を内包するものである。なお，システム稼動直後（本稼動後６ヶ月以内のシステム稼動安定

時）に行う評価も含む。

(2) 検　定　者

検定者に関する基本的な考え方は自己検定である。

この場合の自己検定とは，システム開発に係る部署（システム化要望部署，情報システム部）が検定を行うことを意味する。

検定は職制活動に基くものであり，検定を第一次検定，関連セクション検定，最終検定の3段階に分けている。

○第一次検定 ―― 原則として，システム化要望部署，情報システム部のシステム化担当室長が行い，検定内容によってはチーム主査が代行する。

○関連セクション検定 ―― 情報システム部計画室が有する専門的見地にもとづき必要時に行う。

○最終検定 ―― 原則としてシステム化要望部署，情報システム部双方の部長が行い，検定内容によってはシステム化要望部署，情報システム部のシステム担当室長が代行する。

各種検定の最終検定者及び考え方は，以下の通りである。

なお，最終検定は原則として検定確認会議を以って行うものとする。

① システム化検討着手検定

システム部署として提起された課題を，組織的，本格的に構想立案段階で検討するか否かをシステム化検討着手検定として，情報システム部長が最終的に検定するものである。

② 構想立案検定

システム化の基本方針に関する検定である事を考慮し，システム化要望部署，情報システム部双方の部長を最終検定者とする。

③ 実施可否決定

システム実施可否に関し，最終的な決定・承認措置である事を考慮し，上記と同様の部長決定とする。

④ 稼動直後評価

主として稼動したシステムに関して当初（開発時）目標に対する達成度確認であると共に，システム開発評価の最終総括であることを考慮して上記と同様の部長評価とする。

⑤ その他

上記以外の基本設計検定，詳細設計検定，プログラム検定，稼動直前検定は主として設計内容の細目又はコンピュータ処理技術面，稼動準備事項に関する検定である事を考慮して，原則としてシステム化要望部署，情報システム部双方のシステム化担当室長を最終検定者とする。

但し内容によっては，チーム主査への権限委譲もありうる。

(3) 対象システム

新規開発システムを対象とする。

また，改善システムの内，改善内容が新規開発システムの構想立案段階，又は基本設計段階を経るものであり，従って管理単位として業務計画に計上されたもの又は計上されるべきものも対象とする。

上記を基本とし不明確な場合は情報システム部長，システム室長間で協議決定する。

(4) 検定の主要視点

① システム化検討着手検定

システム化の要望内容について，問題の整理状況，問題の解決方向としての，コンピュータ利用の有効性を判断し更に期間活動計画等を考慮し検討する。

② 構想立案検定

システムの必要性と目的の妥当性，優先順位を検定し，更にシステム目的を表現するために各種案の中から選択されたシステム素案の最適性及び開発順位の合理性に関し，重点的に検定する。

併せて今後のシステム開発を効率的に進めていくためにシステム開発推進体制についても検定する。

③　基本設計検定

システムの基本的内容，特に人的処理内容，コンピュータ処理内容の妥当性，有効性の検定と所要システム資源（設備・ソフト）が適切に対応しているかの検定を加えてコンピュータ利用システムの効率性と実現可能性を判定する。

④　実施可否決定（判定の視点）

○基本的視点

利用部署の業務運営，管理上の諸問題に対する解決方向を確認するため，①構想立案で採択した実行案の適否を再確認し，②構想立案にもとづいて具体化した開発実行設計としての基本設計内容の妥当性について確認する。

○システム化の基本方向・機能に関する確認

○コンピュータ利用に関する確認

○設備・ソフト等利用技術に関する確認

○システムの有効性・採算性に関する確認

○開発推進体制に関する確認

○予算措置，契約手続きに関する確認

○その他，予備検定で問題とされたシステムの基本要件に関する内容及びその対処方法の確認

なお，実行可否判定以降上記確認の諸点を変更する場合は，適時変更の可否について判定を受ける。

⑤　詳細設計検定

業務処理内容とコンピュータ詳細処理内容との対応を検討し，更にコンピュータ処理効率の観点から全体プログラム構造の適否に関し重点的に検定する。

⑥　プログラム検定

第１次検定……プログラム設計者

（検定視点）

○詳細設計内容とプログラムの正確な対応性

○プログラミング・ルールの遵守

○プログラム・メンテナンスの容易性

○プログラム・ステップ数の予実は握管理

最終検定……チーム主査

（検定視点）

○プログラム設計上の前提条件，基本方針の確認

○第一次検定内容の確認

○プログラム関係所要ドキュメントの整備状況の確認

○プログラム設計・作成の進捗管理

⑦　稼動直前検定

機械操作能率，機械スケジュール，Ｉ／Ｏタイミングとデータ授受，ファイル管理，作業用材料等のオペレーション関連事項を円滑に実行する観点より検討し，併せてシステム運用面に関する組織体制面の稼動準備状態についての検定を行う。

⑧　稼動直後評価

システムの実施可否決定段階で開発承認されたシステム開発目標の達成度を評価し，評価結果にもとづき以後の対応方向についても確認する。加えてシステム開発全過程における開発プロジェクト運営管理に関する総括的な評価を行う。

⑨　全開発ステップ共通の評価視点

各開発ステップ単位における開発進捗の予実対比評価及び次開発段階スケジュールの妥当性検定は全開発ステップを通じて共通的な評価の主要視点である。

1.3.2 稼動システム評価の体系

稼動システム評価は，システム開発以降稼動しているシステムの機能を定期的に点検し，機能維持・改善に関するシステムの管理諸点を組織的に確認するための評価制度である。

評価の概要は稼動システムの現状をは握してシステム開発時における設計，実行内容の実現状況を組織的に確認し，更に現在稼動しているシステムが，現実有効に機能しているかの評価を加えて次年度以降のシステムの管理方針を明確にすることである。

(1) 稼動システムの骨子

稼動システム評価は稼動しているシステムに対する日常管理の基本的な要点を部門共通の管理諸点として確認するものであり，下記諸点を骨子として組織的に実施する。なお，全体的な体系を図1－4に示す。

① システムの現状は握

システム開発時点の設計，実行内容を確認し，更に本稼動以降のシステム経過より開発内容の変更諸点をは握して，システムの現状実態を明らかにする。

更に，稼動システムの現有機能を確認する過程でシステムの環境変化に起因する現状の諸問題をは握する。併せて過去1年の維持管理実績（稼動時間，コスト及びメンテナンス実績）をは握し，システム評価の基礎データを整理する。

② システム評価

「システムの現状は握」で確認した稼動システムの現有機能を目標達成度，採算性の視点にたって評価し，この結果を総合的に判断してシステムの有効性を評価する。

i) 目標達成度評価

システム目的を具体化したシステム目標に対する達成度を評価し，システム開発時における設計・実行内容の実現状況を確認する。

ii) 採算性評価

システム目標に対するシステムの採算状況をは握して，コスト・効果バランスの適性を評価する。

iii) 有効性評価

目標達成度及び採算性の評価を基礎にして，システムが現実有効に機能しているかを総合的に評価し，次年度以降のシステムの稼動条件を下記の要領で整理する。

イ．次年度以降達成すべきシステム目標の確認・再設定

ロ．解決すべき基本課題の設定と解決方向の検討

ハ．今後の採算見通しの確認・総合評価（稼動継続可否の判定）

③ システムの改善方向の検討

システム評価の結果をもとにして「システムの現状は握」で抽出した現状諸問題より解決課題を選択し，優先度を決定して次年度以降のシステムの改善方向を検討する。

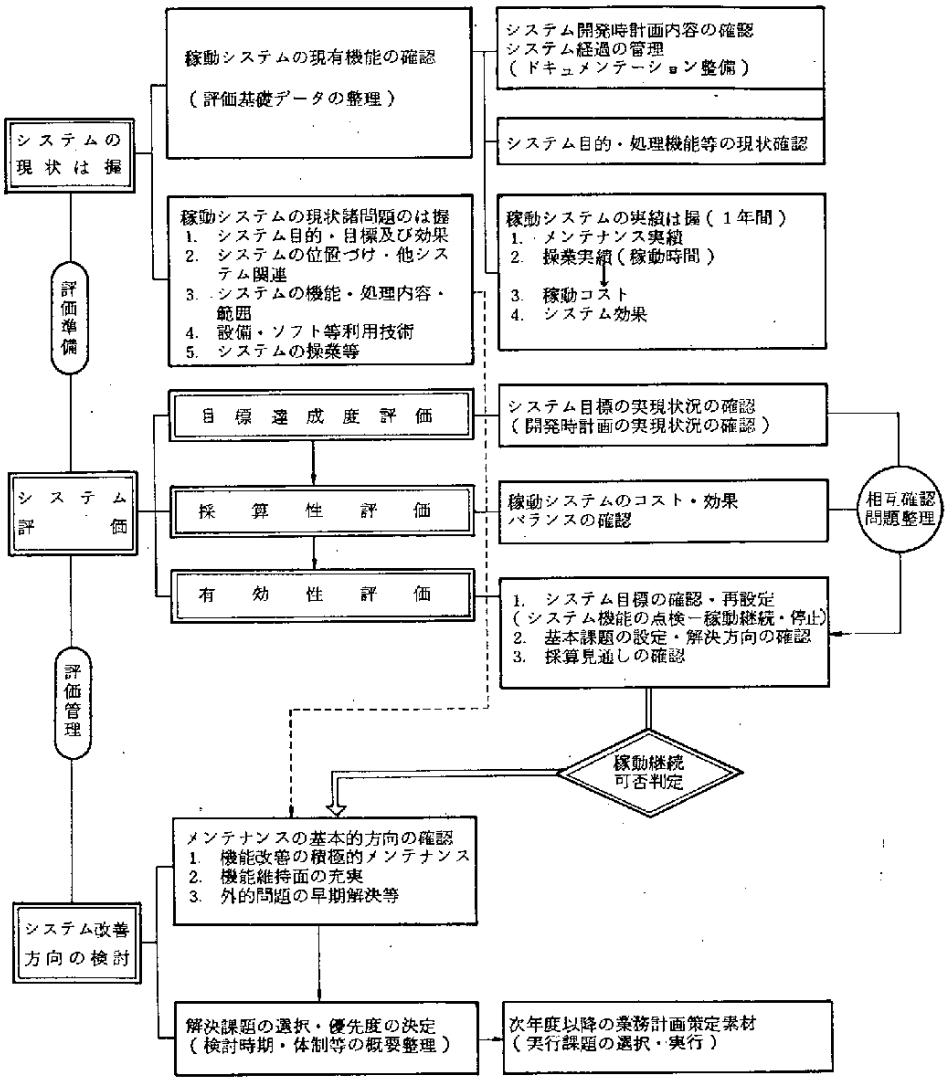


図1－4　稼動システム評価

表１－２　稼動評価シート

昭和　　年　　月　　日

年度稼動システム評価（　年度分）

（評価期間　　～　　）

業務コード

実施時期

開発費用　　千円

主管部署

情報システム部（　　室）

目的・目標

システム評価

目的、目標の達成度評価

採算性評価

総合判断

稼動費用

単位；千円

		電算機関係費用						維持管理労務費		計
		CPU費用		直接費用		CPO費用				
			金額		金額		金額		金額	金額
稼動費用	本番費用									
	維持管理費用									

効　果

定量効果

定性効果

前年度対比

	稼動費用 本番 (A)	稼動費用 維持管理 (B)	計 (C=A+B)	維持管理費用の割合 (B/C)	効果金額 (C)	利　益 (D−C)	費用対効果率 (D−C)
年度							
年度							
差				——			——

前年度課題の実現状況

今後の課題と対処方向（システムの機能、処理効率、維持管理、操業面等）

1.3.3　評価検定の確認運営要領

各開発段階共通の基本パターンは下図の通りである。

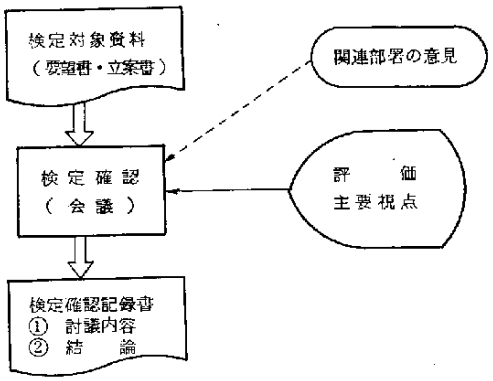


図1－5　評価検定の確認フロー

(1)　検定確認記録書

①　各開発段階終了時に実施される検定確認会議の結果内容については，組織的に整理確認していく事が必要であり，従って討議内容及び最終結論を検定確認記録書（統一フォーム）に明記する。

この検定確認記録書に記述された内容は特に開発段階において検討立案していく場合の重要な前提・指示条件を形成するものであり，よって次開発段階終了時に行われる検討確認に際しては検討立案済の内容が前開発段階終了時の検定確認会議で示された前提条件に適切に対応しているか否かが検定確認の重要なるキーポイントになる。

なお詳細設計及びプログラム設計・作成過程において適時に行う資料単位ごとの個別検定確認の結果内容は個別検定確認書（フリー・フ

ォーム）にまとめる。

② 検定確認の結果資料（検定確認記録書，個別検定確認書）は原則としてシステム開発主体（主として情報システム部のシステム化担当室）が記入配布保管等の措置を行う。

但し，稼動直前検定の１つである稼動引渡し検定に際しては電算機操業部署が上記措置を行う。

(2) 検定確認の対象資料

① 各開発段階別検定確認で使用する対象資料の記載内容は原則として「システム開発手順」にもとづく各開発段階毎に整理する。

② 問題の発生段階において，システム化要望部署より提起され，従ってシステム化検討着手検討の対象資料でもあるシステム化要望書についても統一フォームを用いる。

しかし，当該段階におけるシステム化要望に際し，その項目及びボリューム等は多様的にならざるを得ず，従って，実質的にはシステム化要望書は統一フォーム内容とフリー・フォームの別紙内容の混合形式が大部分と考えられる。

(1) なお，各開発段階における検定確認は原則として，開発評価の主要視点を踏まえて行う。

2 システム費用と効果

2.1 システム費用と効果のは握方法

2.1.1 費用と効果の体系

システムの採算性評価は，開発システムを対象としたシステム開発評価における「実施可否決定」及び稼動システムを対象とした稼動システム評価における「稼動継続可否判定」において，重要かつ大きな影響力を有す

るものである。

このシステムの採算性評価を有効裡に展開していくためには，採算性要因であるところのシステム費用並びにシステム効果に関する統一的なは握方法の確立が特に必要不可欠である。

上記の視点を踏まえ，これを「システムの費用・効果は握要領」ということでまとめており，システム開発過程及びシステム維持管理過程におけるシステムの費用・効果は握は，当内容にもとづき行っている。

システム費用・効果は握方法における基本要件として下図の通り費用・効果体系を形成している。

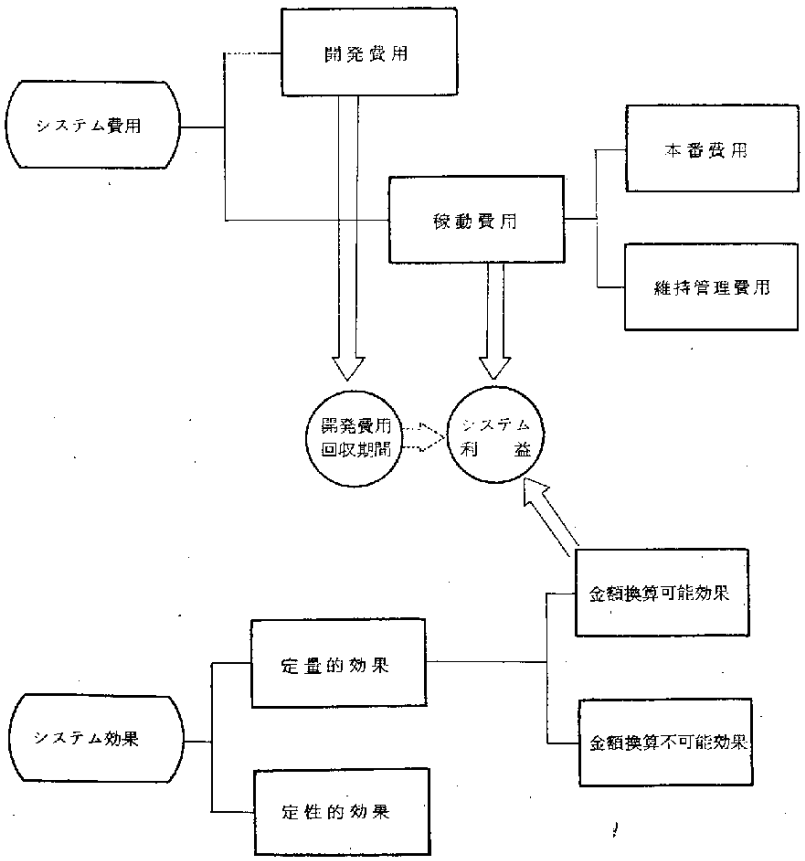


図２－１　費用・効果の体系

2.1.2 費用のは握方法

費用は，システム開発費用と稼動費用の2つに大別され，開発費は見積りと実績の両面からは握している。

また，稼動費用については，本番費用と維持管理費用の2つから構成されている。

以下では，それぞれのは握方法，費用項目，計算方法などを示す。

(1) 開発費用のは握

① 見積り開発費用のは握

i) は握のタイミング

構想立案段階で概要検討し，基本設計で検定する。なお，変更ある場合は詳細設計段階で見直しを行う。

ii) 費用は握・対象期間

構造立案段階～システム本稼動に至るシステム開発期間中に発生する費用

iii) は握単位

一時費用として一括は握

iv) 費用項目と計算方法

イ．社内労務費

労務費単位／人・月×開発人数×期間（月）

〔対象〕○システム担当：原則としてチーム主査以上は含まない。なお，当該システムの開発以外も兼任している場合は，比率を以って設定する。

○システム化要望部署：原則としてチーム主査以上は含まない。なお，専任のみ対象とする。

ロ．ソフトウェア外注費

・委託契約形式の場合

単価／1ステップ当たり×全プログラムのステップ数合計

・請負契約形式の場合

外注労務費単価／人・月×人数×期間（月）

ハ．開発用ソフトウェア費

開発負荷の軽減を主目的としたソフトウェア費（ソフトウェアパッケージ等の購入費など）

ニ．電算機処理費用

・CPU費用＝CPU単価／分×開発テスト分CPU時間／月×期間（月）

・その他費用＝CPO（Concurrent Peripheral Operation）単価／分×開発テスト分CPO時間／月×期間（月）

・オンライン費用（開発テスト期間中に発生する費用）

レンタルの場合：機器レンタル料／月×発生期間

機器買取の場合：本稼動時迄に生ずる減価償却費用及び保守料

ホ．システム化要望部署が費用管理する機器及び作業用材料費用

開発テスト期間中に発生する費用を対象とする。

・機器費用

レンタルの場合：機器レンタル／月×発生期間（月）

買取契約の場合：本稼動時迄に発生する減価償却費用及び保守料

・作業用材料費用：作業用材料単価×使用料

ヘ．端末機の導入諸経費用

ト．端末側の設備工事費用

② 実績開発費用のは握

i) は握タイミング

構想立案段階以降の各開発ステップ単位で実績は握し，累計する。

ii) 費用は握・対象期間
iii) は握単位
iv) 費用項目と計算方法

} 見積り開発費用の場合と同じ。

(2) 稼動費用のは握

① 稼動費用のは握

i) は握タイミング

（見積り）

構想立案段階で概要検討し，基本設計段階で確定，変更ある場合は詳細設計段階で見直しを行う。

（実　績）

システム本稼動以降毎月実績をは握し稼動システム評価時に集約する。

ii) 費用は握，対象期間

システム本稼動以降発生する費用

iii) 費用は握単位

月額費用（年間費用÷12ヶ月）としては握

iv) 費用項目と計算方法

○本番費用

イ．社内労務費（月額）

労務費単価／人・月×人数

〔対象〕コンピュータ利用システムの運営に専任という形で従事しているシステム化要望部署側の労務費

ロ．電算機処理費用（月額）（本番関係処理費用）

・CPU月額費用＝CPU単価／分×CPU時間　分／月

・オンライン月額費用

レンタルの場合：

機器・回線レンタル／月

買取りの場合：

減価償却費用／月＋保守料／月

・その他月額費用＝CPO単価／分×CPO時間　分／月

ハ．システム化要望部署側が費用管理する機器及び作業用材料費用

機器月額費用

レンタルの場合

機器レンタル／月

買取りの場合

減価償却費用／月＋保守料／月

ニ．本番費用（月額）＝イ＋ロ＋ハ

○維持管理費用

イ．電算機処理費（月額）（テスト関係処理費用）

CPU月額費用とその他月額費用は本番費用の場合と同じ。

なお，オンライン月額費用は一括本番関係処理費用として扱う。

ロ．システム維持管理社内労務費（月額）

労務費単価／人，月×維持管理人数

〔対象〕システム担当を対象とし原則としてチーム主査以上は含まない。なお，当該システムの維持管理以外も兼任している場合は比率を以って設定する。

ハ．システム維持管理外注費（月額）

外注労務費単価／人・月×維持管理人数

当該システムの維持管理以外も兼任している場合は比率を以って設定する。

ニ．維持管理費＝イ＋ロ＋ハ

ホ．稼動費用＝本番費用＋ニ．

2.1.3 効果のは握方法

効果のは握方法は以下に示す通りである。

i）は握タイミング

見積り

構想立案段階で概要検討し，基本設計段階で確定する。変更のある場

合は詳細設計段階で見直しを行う。

実　績

システム本稼動以降定期的には握（少なくとも稼動システム評価時には握すると共にシステム特性に応じ適時（月・半期・年）には握）

ii) は握対象期間

システム本稼動以降発生する効果

iii) は握単位

定量効果と定性効果に分けては握し，更に定量効果については全額換算可能効果と全額換算不可能効果別には握する。

定量効果については，月単位（年間定量効果÷12ヶ月）では握する。

iv) は握方法

イ．構想立案段階で設定されたシステム目的を踏まえ，具体的，定量的なシステム目標を構想立案段階にて概要設定し，基本設計段階にてこれを固めることになっている。このシステム目標が実現された時，発生してくる直接的あるいは間接的効果を極力定量化し，これを定量的効果としては握する。なお，定量化がどうしても不可能な場合は定性的効果として極力具体的には握する。

ロ．更に，定量的効果については，全額換算可能効果と全額換算不可能効果とに分けては握する。

全額換算可能効果＝効果量／月×単価

2.1.4　費用と効果の対比

(1)　システム利益のは握

費用と効果にもとづき，下記要領でシステム利益をは握する。

i) は握タイミング

○ 見積り費用・効果の対比

構想立案段階で概要検討し，基本設計段階で確定する。なお，変更

ある場合は詳細設計段階で見直しを行う。

○実績費用・効果の対比

システム本稼動以降定期的には握（少なくとも評価時には握すると共に，システム特性に応じ適時（月・半期・年）には握）

ii) は握対象期間

システム本稼動以降発生するシステム利益

iii) は握単位

月額単位（年間システム利益÷12ヶ月）

iv) 計算方法

システム利益（月額）＝全額換算可能効果（月額）

－稼動費用（月額）

システム利益≧0の場合

システム換算面では通常問題なし。

システム利益＜0の場合

全額面から観たシステム換算面では問題がある。但し，この場合全額換算不可能効果及び定性効果を加味し総合的に採算性を評価することになる。

なお，併せて投資効率という観点より

$$\frac{\text{全額換算可能効果}}{\text{稼動費用}} = \text{投資効率係数}$$ も握しておく。

(2) 開発費用の回収状況は握

開発費用は少なくともシステム・ライフ内にシステム利益によって回収される必要があるという観点に立ち，次の方式で開発費用の回収状況をは握する。但しこれは，システム利益≧0の場合のみ行う。

i) は握タイミング

費用と効果の対比と同じ

ii) は握対象期間

システム本稼動における開発費用の回収状況

iii) は握方法

○見積り費用・効果の対比

月額見積りシステム利益と一括費用としては握された見積り開発費用にもとづき，回収期間を設定，これと構想立案段階で概要検討し，更に基本設計段階で設定されたシステム・ライフを対比する。

イ．開発費用・回収期間＝ $\dfrac{\text{見積り開発費用}}{\text{見積りシステム利益（月額）}}$

ロ．開発費用・回収期間とシステム・ライフとの対比

- 開発費用回収期間≦システム・ライフの場合

  システムの採算面では通常問題なし。

- 開発費用回収期間＞システム・ライフの場合（a）

  金額面からみたシステム採算面では問題あり。但し，この場合全額換算不可能効果及び定性効果を加味して総合的に採算性を評価することになる。

○実績費用・効果の対比

月額実績システム利益と一括費用としては握済みの実績開発費用の残高をは握すると共に，これにもとづき実績開発費用の見込み回収年月と残存システム・ライフを対比する。

イ．今回稼動システム評価時・開発費用の回収残高＝前回稼動システム評価時・開発費用の回収残高－システム利益総額（前回稼動システム評価時迄に発生した月額システム利益の総額）

ロ．開発費用の見込み回収年

$$\frac{\text{今回稼動システム評価時開業費用の回収残高}}{\text{システム利益／円}}$$

- 開発費用の見込み回収年月≦残存システム・ライフの場合

  システム採算面では通常問題なし。

- 開発費用の見込み回収年月＞残存システム・ライフの場合
  見積り費用・効果の対比のロー(a)と同じ

昭和　　年度業務計画

室

①主管部署	業務 ②業務名	業務 ③サブシステム名	④区分	⑤業務概要	⑥開始年月	年度 上	年度 下	⑦年度 4	5	6	7	8	9	10	11	12	1	2	3	年度 上	年度 下	年度 上	年度 下	年以降	⑧実施年月	⑨関連部署

＜要領＞

（区分）－ 新：新規，継：継続，改：改善，維：維持管理，　　（スケジュール）－ 構想立案，基本設計，詳細設計，プログラム設計作成，システム・テスト，ドキュメンテーション他

昭和　　年　　月　　日

設備計画兼操業計画策定書（個別業務）

室

①業務名	サブ・システム名	②業務区分	③処理形態
		□新規 □改善 □維持管理 □その他（　）	□バッチ □IMS □その他（　）

④新規処理増加分

		時期及び増加分／月
a. CPU（分単位）		
b. CPO（分単位）	3800	
	3211	
c.（IMSのみ）増加トランザクション数		

⑤新規設備増加分

		使用開始時期及び増加分（減があれば▲で記入）
a. 外部記憶装置関連	専用ディスク・パック（データセット数及びMバイト数）	
	MSS保管データ量（データ・セット数及びMバイト数）	
	JOB・LIB（プログラム本数及びトラック数）	
	COPY・LIB（メンバ数及びトラック数）	

	端末機種名	台数	設置場所	設置時期
b. オンライン端末装置				

⑥操業関連

	周期	件数	備考（〆切時間等）
穿孔量			

作業日又は作業周期	（バッチ⟺，オンライン⟷，CPO時間帯⟺，伝送時間帯 ----◎）8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 1 2 3 4 5 6 7 8	ELAPS時間 昼（8時~18時）	ELAPS時間 夜（18時~8時）	伝送関連 区間	伝送関連 時間（分）	備考（伝送内容等）
作業スケジュール						

室長	課長	係長	係員

ソフトウェア外注作業要望書

①業務名　　②本番稼動時期　　年　　月　　③外注要員導入時期（開始）　　年　　月　（終了）　　年　　月

④委託範囲　SA　PA　PRG　　⑤委託作業量　　Step

導入理由

要求技術

⑥・処理形態　オンライン　バッチ

⑦・使用言語　PL／1　その他（具体的に）

⑧・その他アプリケーション

⑨検収予定（支払いベース）　　単位：千円

		上期							下期						
		4	5	6	7	8	9	計	10	11	12	1	2	3	計
当年度	プログラム														
	その他														
	計														

		上期	下期	次年度計	累計
次年度	プログラム				
	その他				
	計				

		上期	下期	次々年度計	累計
次々年度	プログラム				
	その他				
	計				

その他：IMSコントロール・コメント・パンチ等

室長	課長	係長	係員

長期要員計画表

昭和　　年　　月　　日

＿＿＿＿＿＿＿＿室

業務名称	昭和　年　月現在					年度								年度				年度				年度			年度		
						上期				下期																	
	社内				外注	社内			外注	社内			外注	社内			外注	社内			外注	社内			社内		
	係長	男子	女子	計		係長男子	女子	計		係長男子	女子	計		係長男子	女子	計		係長男子	女子	計		係長男子	女子	計	係長男子	女子	計

特記事項	

(パンチマニュアルNo.：951Y4)

カード No.	① Y 4	
起点年月	昭和 年 ③ 月	
主管部署	④ M 1	情報システム部
担当者	⑥ 氏名のイニシャル	氏名；
費目	⑦ H C P ⑩	cpu時間；u cpo時間；o
クローズド・オープン区分	⑪ 1	クローズド；1 オープン；2

昭和　年度　本社電算機CPU及びCPO関係稼働時間予想データ投入表（I）

	年間予算上期（実施予算） 下期
	下期実施予算

（昭和　年　月　日作成）

備考；	課長	係長

本番；P
テスト；T

業務コード 大区分	業務コード 区分	本番・テスト区分	4又は10月	5又は11月	6又は12月	7又は1月	8又は2月	9又は3月	計	備考
			本社電算機CPU及びCPO関係稼働時間予想　単位　分							
⑫	⑬ ⑭	⑳	㉘ ㉙ ㉚ ㉛ ㉜ ㉝ ㉞	㉟ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶	㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽	㊾ ㊿ 51 52 53 54 55	56 57 58 59 60 61 62	63 64 65 66 68 69		

一般管理費　予算

昭和　年度　本社電算機オンライン端末機及び回線等 費用予想データ投入表

年間予算上期(実施予算) 下期
上期実施予算

（パンチマニュアル№ 19465）

カード№	①	Y S
起点年月		昭和 ＿＿ 年 ③ 月
主管部署	④	M 1　主管部署名；情報システム部
担当者	⑥	氏名； 氏名のイニシャル
費目	⑦	6 2 2
クローズド・オープン区分	⑪	＿＿＿ クローズド；1 オープン；2
業務コード	⑬ ⑭	＿＿＿ オープンの場合記入不要

（昭和　年　月　日作成）

備考；	課長	係長

本番；P
テスト；T

クローズドの場合記入不要

単位 千円

区分	本番テスト分	費用負担先 負担番号	4又は10月	5又は11月	6又は12月	7又は1月	8又は2月	9又は3月	計	備考
20 21	22 23 24 25		28 29 30 31 32 33 34	35 36 37 38 39 40 41	42 43 44 45 46 47 48	49 50 51 52 53 54 55	56 57 58 59 60 61 62	63 64 65 66 67 68 69	70	
	9									
	9									
	9									
	9									
	9									
	9									
	9									
	9									
	9									
	9									

業務コード；９６　本社電算機費用配賦システム

情報システム部　各　室 ―― 情報システム部　電算室

備考：	課長	係長	係員	課長	係長	係員

本社電算機費用負担先マスター　新規投入表（兼）修正表

（パンチ・マニアルNO；96001）

情報システム部内業務開発・維持管理部署の各室にて記入

情報システム部内業務開発・維持管理部署
計　画　室；0　（第三開発室；3）
第一開発室；1
第二開発室；2
電　算　室；0

経理部管理室決定に基づき情報システム部電算室にて記入

「費用負担先」が2つ以上（最高6つまで）の場合，継続番号として1～2を記入すること。尚，この場合，「業務コード」（7～11欄）も重複して記入すること。

カードNO	マスター投入 年	マスター投入 月	業務コード 代表コード	業務コード 大区分	業務コード 小区分	業務名称（16桁）	本番稼働又は本番稼動予定 年	本番稼働又は本番稼動予定 月	業務主管部署 負担番号 投入	費用負担先 負担先(1) システム	負担先(1) 負担番号	負担先(1) 負担比率(%)	負担先(2) システム	負担先(2) 負担番号	負担先(2) 負担比率(%)	負担先(3) システム	負担先(3) 負担番号	負担先(3) 負担比率(%)	継続番号	備考	NK-DACS 修正SEQ・NO 取消し(1桁)	修正コード(1桁)	業務コード	SEQ・NO
1 2	3 4	5	6	7 8	9 10	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27	28 29	30 31	32 33 34 35 36	37	38 39 40 41	42 43 44 45	46	47 48 49 50	51 52 53 54	55	56 57 58 59	60 61 62 63 64	65		71	72	73 74 75 76 77 78 79 80	
M	1									9			9			9				（新規投入用切取り線）		D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1
M	1									9			9			9						D	9 6	M 1

費用や時間等を業務別に作表する場合，実質上同一業務内の大区分（例えば，パッチとＩＭＳオンライン）を集計する単位のコード。但し，それが1対1の場合でも大区分と同じコードを必ず重複記入すること。

㊟ 1. 新規投入時；「NK-DACS 修正SEQ・NO」10桁（71～80欄）は，パンチ対象外とするため，その部分を必ず「切取り線」に従って切り取るか又は「　」印で抹消して投入すること。

2. 修正時；① 「NK-DACS 修正SEQ・NO」10桁（71～80欄）は，絶対記入項目である。但し，上1桁の取消しサイン ＊（アスタリスク）は，全面取消し以外は何も記入しないこと。

② 上記以外は，訂正したい項目だけを修正する部分修正方式とする。

〔用語及び帳票例索引〕

参考文献

1. 知識労働者のマネジメント （松井　好「マネジメント'73.11月号」）
2. 意思決定支援システムの開発を管理するためのチーム・アプローチ （Auerbach Mgt Report，日本経営科学研究所）
3. プロジェクト管理のための体制 （　同　上　）
4. プロジェクト管理 ― そのプロセス （　〃　）
5. ソフトウェア製品生産の人間的側面 （
6. プロジェクト管理の実際 （鈴木徳太郎，日本能率協会）
7. ソフトウェア開発ハンドブック （コンピュータ・アプリケーションズ編，オーム社）
8. 効果的プログラム開発技法 （国友義久，近代科学者）
9. ソフトウェアの複合／構造化設計 （国友義久，伊藤武夫，近代科学社）
10. ソフトウェア・エンジニアリングに関する調査 （日本電子工業振興協会）
11. ワークデザイン （ジェラルド・ナドラー，建　社）
12. 情報処理心理学 （森本正昭，誠信書房）
13. 変りゆく情報システム部 （名和小太郎，企画センター）
14. "生産性"向上の課題 ― EDPリサーチレポート 1981.8.20 （日本能率協会）
15. プロジェクト，マネージャー教育訓練のための情報システムの開発研究 （運輸経済研究センター）
16. 管理者のリーダシップ （産業能率短期大学）
17. 部下の教育指導 （　同　上　）
18. プロジェクト・マネージメント （J.S. バウムガードナー，抜報堂）
19. プロジェクト・ハンドブック （須賀基嗣ほか，日刊工業新聞社）
20. マトリクス組織 （カンファレンス・ボード，日本能率協会）
21. マトリクス組織の編成と運営 （北野利信ほか，ダイヤモンド社）
22. 横断組織の設計 （J．ガルブレイス，ダイヤモンド社）
23. システムマネジメント （D.I．クリーランド，ダイヤモンド社）
24. 行動科学の展開 （P.C．ハーシーほか，日本生産性本部）

25. Managing High-Technology Programs & Projects （Rusell, D.A.）
26. Managing the System Development Process （Atkins, B.B.）
27. プログラミングプロジェクトの管理 （下田正昭ほか，近代科学社）
28. Management Methodology for Software Product （Gunther, R.C.）
29. ソフトウェア製品生産管理，一情報処理 Vol-21, №10 1980/10 （情報処理学会）
30. スケジェーリングの理論 （関根智明，日刊工業新聞社）
31. 外注管理，１９６６ （日本能率協会）
32. Project Control Manual （Hed, S.R.）
33. Caputure & Recapture 法による潜在バグの推定法とその応用，１９８１ （ソフトウェア工学研究会）
34. 情報処理サービス業の標準化に関する報告書，１９８１ （日本情報センター協会）
35. ソフトウェアの開発・管理の効率化研究結果報告書，1981. 3 （各省庁電子計算機利用効率化共同研究会）
36. プログラム生産技術の体系と評価 －通研実報，VOL 28, №12 （情報処理学会）
37. "特集：ソフトウェア製品生産管理" －情報処理，'80 VOL 21, №10 （同上）
38. FACOM SDEM 導入と適用の手引 －富士通マニュアル
39. "特集：プログラム作りの環境" －ビジコン，'81 VOL 18, №12
40. "特集：システム部門の効率化・省力化" －ビジコン，'81 VOL 18, №
41. ソフトウェア・エンジニアリング （菅野文友，日科技連）
42. HIPACE-SPDS －日立製作所マニュアル－

— 禁 無 断 転 載 —

昭和５７年３月発行

発行所　財団法人　日本情報処理開発協会
東京都港区芝公園３－５－８
機械振興会館内
TEL　(434)8211(代表)

閲覧（持出厳禁）	
付 No.	H - 36
受付年月日	
作 成 課	