解　説

プロジェクト管理のツール†

岩元莞二††　岡田正志†††

1. はじめに

ソフトウェアの開発は人間の創造力に頼る面が多く，その生産性はシステムの規模，新規性，ホストシステムの新しさ，開発環境，要員のスキル，開発手法などの多くの要因によって影響され，一律に生産性の基準値を与えることができない．また，生産性そのものを定義する，一般に認められた明確な尺度すら現存しない．このような人間活動の集積であるプロジェクトを成功させるために，プロジェクト管理者の果す役割は極めて大きい．特に，開発するソフトウェアの規模が大きくなるほどその重要性が増してくる．

プロジェクト管理の具体的作業として下記があるが管理者は何をなすべきかの決定と動機づけを行い，如何にすべきかは，それぞれの担当者が行う場合が多い．

（1）　プロジェクト計画　プロジェクト目標の設定，資源（人，金，物）計画，日程計画．

（2）　組織づくり　要員確保と責任分担の割当て，資源調達，開発環境の整備，要員教育訓練計画．

（3）　標準化　開発手法の導入，開発標準とプロジェクト運営標準の設定．

（4）　工程管理　詳細日程計画，進度の計測と制御．

（5）　予算管理　実績費用の計測と制御．

（6）　品質管理　各作業過程での品質管理と製品の品質管理．

（7）　コンフィギュレーション管理　システム構造の体系的管理．

（8）　プロジェクト評価　実績の評価および次期プロジェクトへ反映させるためのデータ収集と分析．

このように，ソフトウェアプロジェクトの管理は多くの問題を含んでいるので，プロジェクト管理の問題に対するアプローチも多種多様である．

プロジェクト計画のツールとしては PERT が知られているが，Woodgate[1] は PERT 的ネットワーク手法に基づく新しい計画システムを開発した．計画での見積り精度を高めるため，生産性要因分析をした例として，Daly[2]，Walston[3]，Wolverton[4] などがある．また，Cave と Salisbury[5] はソフトウェアのライフサイクル制御の観点から，ソフトウェア品質をコストとからめて定義，その定量的尺度を与えている．

組織化に関しては，IBM のチーフプログラマチーム，Weinberg の民主的プログラミングチームが良く知られており，組織に関する問題はそれぞれの企業により工夫されていると思われる．組織の問題と標準化の問題は自動化の対象になりにくい．

工程管理では，各工程の生産物の状態に関する情報を自動収集して進捗管理の助けとする段階のツールと，それらの情報および人手入力の情報から計画の達成状況を報告する段階のツールがある．これらのツールについてはあとで詳しく触れる．

予算管理は企業経営上不可欠であるが，ソフトウェアのコスト分析をするうえでは，既存の企業内予算管理システムは管理単位が大き過ぎる．ソフトウェア開発費の大部分を占めるのは人件費であるから，工程管理システムから得られる工数情報が，工程，プログラム，人，対応のコスト分析に利用できる．

効果的品質管理のためには最終製品の検査のみならず，開発過程での品質管理が重要であるが，これには各作業に対応した方法論やツールの開発が必要である．これまでのソフトウェア工学の研究は多くがこのために行われていると言えるだろう．

コンフィギュレーション管理に関しては，従来はプログラム類のバージョン管理のツールしかなかったが，最近は要求定義や設計のための機械処理可能な言語が開発されてきたので（たとえば，TRW 社の RSL，ミシガン大の PSL），要求定義や設計とプログラムを

---

† Tools for Software Project Management by Kanji IWAMOTO (Computer System Research Laboratory, Central Research Laboratories, Nippon, Electric Co., Ltd.) and Masashi OKADA (NEC Software, Ltd.).

†† 日本電気(株)中央研究所コンピュータシステム研究部
††† 日本電気ソフトウェア(株)第一システム技術部

関連づけてソフトウェアの構造を管理することができるようになりつつある.

一方, プロジェクト進行の巨視的予測に, Putnum[6] と Basili[7] は Rayleigh 曲線, $y=2Kat\cdot\exp(-a\cdot t^2)$ を利用, 実際との比較を行っている. ここで, y は時間における資源消費量, Kはプロジェクトの資源総消費量, $a=1/t_d^2$ で, t_d は y が最大になる時間である.

以上述べたように, プロジェクト管理者が心を配るべき事柄は多様で, 開発の各段階の作業を支援するツールがそのままプロジェクト管理に利用できる場合が少なくないが, 以下ではプロジェクト管理固有の問題として, プロジェクト計画と工程管理を中心に支援ツールをいくつか紹介し, 最後に, 筆者らが開発中のソフトウェア開発・保守システムにおけるプロジェクト管理サブシステムの概要を述べる.

2. プロジェクト計画のツール

プロジェクト計画の代表的ツールとしては PERT が知られており, 主要計算機には応用プログラムとして PERT が具備されている. 後述の SIMON, SEF, ソフトウェア・ファクトリなどでもプロジェクト計画を支援する PERT 型のツールが組込まれている. ここでは, 筆者の知る範囲で最も進んでいると思われる米国インターナショナルズ社の多重プロジェクトの計画システム[1]を紹介しよう.

同社では何百人かのプログラマ, アナリストがいて多くのプロジェクトが同時に進行している. そこで, プロジェクトの優先度を考慮したうえで, 複数のプロジェクトにどのように資源を割付け, 誰がいつどの仕事をやるべきか, 予想達成日はいつか, 目標達成のためにどの位費用がかかるか, などの計画の意志決定を支援する会話型システムを開発した.

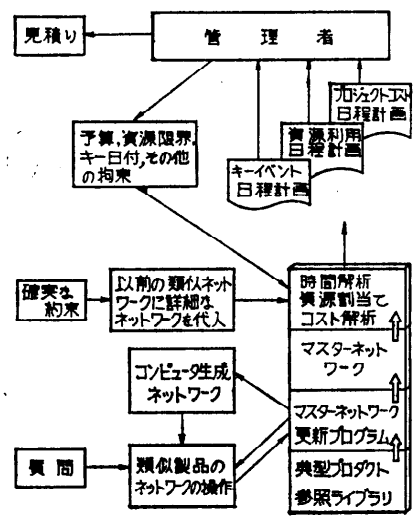

図-1 多重プロジェクト計画システム

システムの概要を図-1に示す. 日程計画のアルゴリズムは PERT と同様のネットワーク手法を用い, 多重プロジェクトのスケジュールはマスターネットワークとして管理される. 個々のプロジェクト対応のサブネットワークはネットワークリファレンスライブラリから生成される. マスターネットワークに含まれる情報は, イベント, アクティビティ, プロジェクト間関係, 外部拘束条件, プロジェクト間の相対的優先順位, 実行時間, 資源, コスト, などである. システムの出力は, キーイベントの日付け, リソース利用スケジュール, プロジェクト見積り費用, デジタルプロッタによるネットワーク図などである. プロジェクトはいくつかの優先度グループに分類しておき, 解析はプロジェクト作業の連続性条件を加味して, 優先度考慮の時間限定, リソース限定の2種を選択できるようになっている. 最適スケジュールの決定は最も難しい問題であるが, ここでは蓄積してある過去のパターンを利用した発見的手法をとっている.

3. 工程管理のツール

以下述べるものは, 最後の一つを除いて, プロジェクト計画, 工程管理, 資源管理, 費用管理, コンフィギュレーション管理を含めてソフトウェア開発全般を支援するシステムであるが, そのうち, 工程管理の機能を中心にその特徴を紹介する.

(1) SIMON[8]

米国 MITRE 社のシステムで, ソフトウェアの構造またはフェーズに関連してプロジェクトを管理するのを支援する. このシステムを利用することにより下記の質問に答えることができる.

・予想以上に大きく, 複雑, あるいは誤りがちになっている部分はないか,

・テストに適した順序でつくられているか,

・システムのどの部分がドキュメントされていないか, テストされていないか, コンパイル未了か,

・テストは何%が完了しているか.

この質問から想定されるように, SIMON はプログラムのコンパイル結果, プログラムの静的あるいは動的解析, テスト履歴によってプログラムの状態を自動的に解析してプロジェクトの管理に役立てている. 従

って，プログラミングの段階以降の工程についての管理情報の自動収集が特徴であるが，その他の計画，ジョブ間の関係，資源の限界等の情報は他のシステムと同じく人手によって入力する．

（2） ソフトウェア・ファクトリ[9]

SDC 社が開発したソフトウェア開発を支援するトータルシステムで，要求定義，設計の段階からこのシステムを利用してソフトウェアの開発を行うようになっている．従って，開発上の情報がプロジェクト制御データベースに格納され，この中の情報とマシンアクセスの履歴から，プロジェクト管理サブシステム IMPACT (Integrated Management, Project Analysis and Control Technique) が，作業の計画，監視，制御の支援とコンフィギュレーション管理の支援のための機能を提供する．上記データベースにはドキュメント，プログラム，装置情報，要求，設計変更，アクティビティ，プロダクトおよび人手入力の各種管理情報を格納する．これらの情報を解析して以下の種類の報告書を生成する．

・資源割付け報告（スケジュールと費用情報），

・コンフィギュレーションインデックスと状況報告（引き渡し可能なモジュール，到達マイルストーンなど），

・コンフィギュレーション要約（モジュール，仕様への索引，変更の累積），

・修正索引報告（設計変更，計画されたマイルストーン，承認），

・モジュール実行要約（コンパイルとテストの履歴）．

（3） SEF[10]

Softech 社開発の SEF (Software Engineering Facility) は，最近のソフトウェアエンジニアリングの効果的成果を利用して，設計，プログラミング，テストの作業をできるだけコンピュータで支援してやろうというもので，各種の解析，設計，仕様化のツールがデータベース内にソフトウェアの構造にそって結果を記録し，その情報を用いてインタフェースの無矛盾性の検査を行うようになっている．これらの情報を利用してシステム要約や傾向，クロスリファレンスなどのプロジェクト管理用の報告書を生成している．プロジェクト管理用の特別なデータを入力させず，ソフトウェア開発データベースから得られる情報を最大利用した例である．したがって，スケジューリングを直接に扱うツールはない．ソフトウェア開発の可視性をあげることにより，プロジェクト管理がし易くなるようにするという思想である．

（4） 協同システム開発のプロジェクト管理プログラム[11]

協同システム開発社の「プログラム生産技術」プロジェクトの一つとしてプロジェクト管理プログラムが開発されている．このシステムは SEF とは対照的にプロジェクト管理のみに的を絞ったもので，その他のソフトウェア開発ツールとは独立している，機能としてはプロジェクト計画，進行管理，プロジェクト履歴データの収集など，費用管理も含めて狭義のプロジェクト管理の機能を一通り揃えている．このうち，プロジェクト進行管理のための機能としては，作業状況報告書，例外報告書，バーチャート報告書，ネットワーク図表，マイルストーン報告書，アクティビティ費用報告書，組織費用報告書，などが出力される．作業状況報告書ではアクティビティ単位の完了，進行中，開始日，残日数，完了度合を表示する．

以上，工程管理機能を中心に代表的なシステムについて述べたが，このほかに TRW 社の SREP や富士通の SDSS[12]，あるいは東芝の T-SWB[13] など，統合的なソフトウェア開発支援システムは多かれ少なかれプロジェクト管理の機能を有している．しかしながら，その取り組み方には開発技術を中心に支援して副産物としてプロジェクト管理の機能をもっているものや，当初からプロジェクト管理を中心機能と設定しているものがあり，それぞれ相異がある．これらの傾向を示すために，管理技術指向の度合と開発技術指向の度合を二つの軸として，紹介したシステムと後述の SDMS の相対的位置づけを図-2 で示す．この定義は主観的なものであるが，開発効率化，品質向上，コンフィギュレーション管理の容易さが開発の多くの過程で支援されるものほど開発技術の指向性が高く，費用，日程，資源の計画・管理の機能が豊富で経営的色彩が強いほど管理技術の指向性が高いとした．理想的

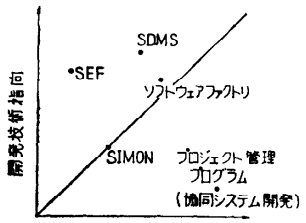

図-2 プロジェクト管理システムの技術指向性

には両方の指向性が高いものが望まれるが，開発のツールは人間の作業の一部を助けるものであり，一方すべての作業が管理対象となることから，両方の機能を完全に満たすようにするため大がかりな使いにくいシステムになったのでは得策ではない．両者の分担を明確にして余分な入力コストなしに共通化できるものを共用し，それぞれに使い易いツールを作るほうが勝っているだろう．もっと積極的には開発技術の革新によって管理ポイントを少なくする方向にもって行くことが望まれる．

次章では，筆者らが開発中のソフトウェア開発／保守システム SDMS における進捗管理機能を少し詳しく述べることで，ソフトウェア開発における管理ツールのあり方を考える上での参考にしたい．

4. SDMS のプロジェクト管理機能

SDMS はソフトウェア開発のライフサイクルを支援する統合システムであり，そこではプロジェクト管理固有の機能をもつサブシステムを含んでいる．SDMS のその他のサブシステムは主に開発のツール群を提供するが，副産物として開発の状況や中間製品，製品の状態に関する報告がプロジェクト管理にも利用できる．プロジェクト管理者は，作業の進捗と実績工数をプロジェクト管理サブシステムで知り，作業品質，遅れの原因，進捗の予測あるいはコンフィギュレーション管理については他サブシステムの報告書を分析して具体的対応策を指示する．他サブシステムのプロジェクト管理対応の機能の概要を下記に示す．

①設計サブシステム　高水準の設計記述言語とそのプロセッサによる設計品質の自動チェックと設計解析情報の自動生成，モジュール間インタフェースの静的チェック，設計書一覧，担当者別または期間単位の設計書一覧．

②テストサブシステム　テストケース被覆性の計測を通してのプログラム信頼性の推定，テスト実施状況報告．

③品質管理サブシステム　設計品質とプログラム品質に関するエラーカテゴリ別の統計．

④プロダクト管理サブシステム　ドキュメントを含めたソフトウェアコンフィギュレーション管理．

一方，プロジェクト管理サブシステムは作業工数実績管理，マシン使用状況統計データの自動収集，予算管理，進捗管理の機能を提供する．2番目の機能以外は人手によるデータの入力を必要とするが，このような

シンコウ　ジョウキョウ　イチラン

サブシステム レベル	サブシステム メイショウ	クブン	チョウサブンセキ	ヨウキュウブンセキ	シリョウサクセイ
01	SSMDB	カイシ	78/09/22	78/09/29	78/09/29
		カンリョウ	78/10/01	78/10/08	
		ヨテイ	78/10/01	78/10/10	79/02/01
		コウスウ	110.75	80.50	70.00
02	SSMDB-INPUT	カイシ	78/09/22	78/09/29	78/10/05
		カンリョウ	78/10/01	78/10/10	
		ヨテイ	78/10/01	78/10/10	79/02/01
		コウスウ	20.50	30.25	40.00
03	SSMDB-INPUT-PARAM	カイシ	68/09/25	78/09/29	78/10/05
		カンリョウ	68/10/01	78/10/10	
		ヨテイ	68/10/01	78/10/10	79/02/01
		コウスウ	12.50	18.75	15.50
03	SSMDB-INPUT-DATA	カイシ	78/09/25	78/09/29	78/10/05
		カンリョウ	78/10/01	78/10/08	
		ヨテイ	78/10/01	78/10/10	79/02/01

（a）　進行状況一覧

シンコウド　イチラン

サブシステム レベル	サブシステム メイショウ	クブン	シンコウド	チョウサブンセキ	シリョウサクセイ
01	SSMDB	ヨテイ	67.9	100.0	39.5
		ジッセキ	60.6	100.0	31.5
02	SSMDB-INPUT	ヨテイ	65.0	100.0	50.0
		ジッセキ	64.5	100.0	45.0
03	SSMDB-INPUT-PARAM	ヨテイ	65.0	100.0	50.0
		ジッセキ	66.0	100.0	60.0
03	SSMDB-INPUT-	ヨテイ	65.0	100.0	50.0

（b）　進行度一覧

図-3　SDMS の出力イメージ

な機能を加えた理由は，既存の PERT 型の管理システムではソフトウェアプロジェクトの特性である下記の要請に十分答えられないことにある．

①システム構造とフェーズ（開発工程）構造の2元構造による進度管理，工数実績管理および予算管理．

②トップダウン開発に応じた進度管理．

③定量的達成度の計算．

マシン使用状況報告を除いて，すべてシステム構造類似の作業階層構造とフェーズ構造の2元構造による報告書が出力される．以下，このシステムの最も特徴的な進捗管理機能について説明する．

（1）　システムの入力と出力

この機能の入力と出力は次の種類からなる．

・**入力**　作業階層構造とフェーズ構造の定義，個人別作業時間記録，マイルストーン承認，カレンダー．

・**出力**　進行状況一覧，現時点進行度（予定と実績）一覧，進行度バーチャート，作業ノード別担当者，予定超過警告，その他．

進行状況一覧と現時点進行度の出力イメージを図-3に示す．

（2） ソフトウェア開発作業の構造化

ソフトウェアの開発を規定する構造として，開発対象システムの階層構造と，要求分析，設計，プログラミング，テストなどの作業の内容で区分されるフェーズ構造がある．ここでは作業対象の階層構造に対応した作業階層構造と，作業の種類を表わすフェーズ構造によって開発作業を構造化する．言語システムの開発なら言語仕様も作業対象に含めるという点でシステム階層と作業階層は異なる．作業階層構造では上位の作業は下位の作業の和として表現する．フェーズ構造は各作業ノードに共通に定義し，上位作業のフェーズ i は下位作業のフェーズ i の和として定義する．作業が完成したかどうかを見る最小単位をマイルストーンと呼ぶ（通常用いられるマイルストーンより小さな単位である）．作業階層樹の終端ノードの各フェーズには個有のマイルストーンをいくつか設定する．以下，ノードは作業階層樹のノードを意味する．

（3） 作業達成度の計算

作業階層の兄弟ノード間には作業量の比重を示す重みを付与し，また各ノードに対応してフェーズ間の重みを示すフェーズ重みベクトルを定義する．終端ノード i の第 j 作業区分（フェーズ）の達成度 $t_{i,j}$ は，

$$t_{i,j}=\sum_k v_k \cdot m_k , \quad \sum_k v_k=1$$

で定義する．ここで，m_k は作業対象 i のフェーズ j に対する k 番目のマイルストーンが達成なら 1，未達成なら 0 を値とし，v_k はマイルストーンの重みを示す．

終端ノードの作業達成度 $\hat{t}_i$ は

$$\hat{t}_i=\sum_j u_{i,j} \cdot t_{i,j} , \quad \sum_j u_{i,j}=1$$

で求める．ここで，$u_{i,j}$ はフェーズの重みを示す．

非終端ノードの達成度 $\hat{t}$ は下位ノードの達成度から

$$\hat{t}=\sum_i w_i \cdot \hat{t}_i , \quad \sum_i w_i=1$$

で計算，w_i は子ノード間の重みである．このノードの第 j フェーズの達成度は下位ノードの第 j フェーズの達成度から

$$t_j=\frac{1}{\sum_i w_i \cdot u_{i,j}} \sum_i w_i \cdot u_{i,j} \cdot t_{i,j}$$

で定義する．このノード全体の達成度は

$$\hat{t}=\sum_j u_{oj} \cdot t_j$$

でも計算できる．両方の値が等しくなるためには，$u_{oj}=\sum_i w_i \cdot u_{i,j}$ が成立しなければならない．

（4） トップダウン開発における進捗管理

トップダウン開発では開発が進むにつれて下位の作業構造が明らかになる．これに合せて作業階層構造を次のようにトップダウンに展開しながら進捗を管理することができる．

ステップ 0（初期設定）

フェーズ構造を決定，プロジェクト全体の作業量を見積り，ルートノードのフェーズ重みベクトルを設定．

ステップ 1（中間ノードの分解）

そのノードのレベルで行う実作業の対象を子の終端ノードとして定義，その重み，フェーズ重みベクトルおよびマイルストーンを定義．

ステップ 2（兄弟ノードの定義）

終端ノードの作業が進むと次レベルの作業が明確になる．次レベルの作業対象をその終端ノードの兄弟として割当て，ノード重み，フェーズ重みベクトル，マイルストーンを定義．ステップ 1 へ行く．

未定義の中間ノードの達成度を 0 として，任意の段階で，任意の作業対象と任意のフェーズについて達成度を見ることができる．同様に，期間，工数に関しても，任意の作業対象，任意のフェーズについて見ることができるので，生産性要因分析を高精度で行うことが可能になると思われる．

5. むすび

以上，プロジェクト管理のツールを紹介してきた．現実には，プロジェクト管理手法を標準化し，管理ガイドブックとワークシートを用いた方式が良く用いられていると思われるが，ここではツールをコンピュータ利用のツールと限定して言及しなかった．むしろ，どの程度コンピュータを利用できるか，の観点で述べてきた．

プロジェクト管理の入力データは，開発作業の結果を自動解析してある程度の自動収集が可能だが，作業時間，作業開始日／終了日などの重要なデータが人手に頼らざるを得ない現状にある．従って，正確なデータを収集するための制度，動機づけ，制度の運営法がこのようなツールを効果的に利用するために極めて重要である．適切な運営によって，ソフトウェア開発過程の可視性を高め管理効果をあげるとともに，集積されたデータをプロジェクトにまたがって分析して，将来プロジェクトに対する，客観的根拠のある基準日程，基準作業時間，基準コストを設定できるようになることを望む次第である．

参 考 文 献

1) Woodgate, H. S.: Management of Large-Scale Computer Program Production, Proceedings of NCC, pp. 277-283 (1977).
2) Daly, E. B.: Management of Software Development, IEEE Trans., Vol. SE-3, No.3, pp. 230-242 (1977).
3) Walston, C. E. and Felix, C. P.: A Method of Programming Measurement and Estimation, IBM Systems Journal, Vol. 16, No. 1, pp. 54-73 (1977).
4) Wolverton, R. W.: The Cost of Developing Large-Scale Software, IEEE Trans., Vol. c-23, No. 6, pp. 615-636 (1974).
5) Cave, W. C. and Salisbury, A. B.: Controlling the Software Life Cycle—The Project Management Task, IEEE Trans., Vol. SE-4, No. 4, pp. 326-334 (1978).
6) Putnum, L.: A Macro-Estimating Methodology for Software Development, IEEE Computer Society Compcon, pp. 138-143 (1976).
7) Basili, V. R. and Zelkowitz, M. V.: Analyzing Medium-Scale Software Development, Proceedings of International Conference on Software Engineering, pp. 116-123 (1978).
8) Clapp, J. A. and Sullivan, J. E.: Automated Monitoring of Software Quality, Proceeding of NCC, pp. 337-341 (1977).
9) Bratman, H, and Court, T.: The Software Factory, Computer, May, pp. 28-37 (1975).
10) Irvine, C. A. and Bracket, J. W.: Automated Software Engineering Through Structured Data Management, IEEE Trans. Vol. SE-3, No. 1, pp. 34-39 (1977).
11) 三井大三郎: プロジェクト管理支援ツール, ソフトウェアコンベンション'78報告書, pp. 109-116 (1978).
12) 竹内卓二, 落合隆: SDSSについて, ソフトウェア工学シンポジウム資料, (Janu. 1979).
13) 佐々木興亜, 竹沢国雄, 山本修一: ソフトウェア一貫生産ツール (T-SWB), ソフトウェア工学シンポジウム報告集, pp. 217-225 (1979).

(昭和54年5月17日受付)