ソフトウェア工学 95－4
（1993．11．30）

管理者と作業者間の作業誘導を支援するソフトウェア工程管理システム

櫨山 淳雄[1]　　古宮 誠一[2]

情報処理振興事業協会 技術センター

大藤 倫昭　　有原 浩司

ニチメンデータシステム 株式会社

著者らはオブジェクトデータベースを中核とした総合的なソフトウェアプロジェクト管理システムの構築を目指している。本稿では、まずシステムの中核となるソフトウェアプロジェクト管理のためのプロセスモデルについて述べている。このモデルに基づいて立案された計画により、プロジェクト管理者が各作業者に作業指示を与え、作業者から報告される進捗情報に基づき進捗把握を行い、リスクを検出し、その影響を解析するまでの一連の作業を支援できるような枠組を提案している。このような枠組を実現するために、プロジェクト管理の視点から作業に対して引き起こされるアクションを検出し、アクションにともなって遷移する状態（ステータスと呼ぶ）を有限状態機械としてモデル化している。そしてプロセスモデルに基づいて作成された作業ネットワークで規定した関係を利用して、状態遷移をひき起こすアクションの起動時に適切な人に適切な情報を通知する枠組を提案している。また、提案した枠組のプロトタイプシステムについて述べている。

A Software Process Management System Supporting the Work Induction Between Managers and Developers

Atsuo Hazeyama　　Seiichi Komiya

Information-technology Promotion Agency, Japan (IPA)
Software Technology Center
Shuwashibakoen 3-chome BLDG.,
3-1-38 Shibakoen, Minato-ku, Tokyo 105, Japan
(hazeyama/komiya)@stc.ipa.go.jp

Noriaki Daitoh　　Hiroshi Arihara

Nichimen Data System

The authors aim at constructing an integrated software project management system with an object database. This paper describes a process model for software project management that forms the basis of the system. This paper also describes a framework to induce a series of works that a project manager must perform as follows: giving work instructions in accordance with the plan that have been made based on the process model to workers, progress ascertainment based on the progress reports sent from workers, analysis of the impacts by problems detected. In order to realize these facilities, the authors analyse actions triggered for a unit activity from the point of view of project management. Based on the analysis, they propose a finite state machine model involving the actions as well as the states changed by actions. When an action is triggered, whether the action is effective or not is checked, information accompanying the action is notified to appropriate persons by referring the relationships specified by the process model. This paper also shows a prototype system which has been developed based on the framework.

[1]日本電気株式会社から出向
[2]日立製作所から出向

1 はじめに

情報化社会の進展にともない、ソフトウェア開発に対するニーズが高まっている。しかも開発されるソフトウェアは大規模化、複雑化傾向にあり、このようなソフトウェアを高品質で、納期通りに、しかも予算の範囲内で作成することが難しい状況にある。このような状況のもと、近年ソフトウェアプロセスに関する研究開発が活発に行われており (例えば [4][5] 等)、ソフトウェアプロセスに基づいた開発環境の構築が行われている (例えば [10][16][17] 等)。大規模なソフトウェアの開発は、多人数により構成されるプロジェクトを組んで行われるのが一般的である。このようなプロジェクトを成功に導くためには、プロジェクトで行われる作業を定義するとともにその実施計画を立案し、その計画に基づいて作業者が作業を行い、その進捗を確認するといった管理指向の強いアプローチで行うのが現実的である。なぜならばソフトウェア開発では各工程での作業が実質的にいつ終了したのかがわからないという特徴をもつからである。このため、工程ごとの作業とその成果物を明確にし、各工程の作業の途中/終了時点でその作業を確認する方式を採ることが現実的だからである。しかも、ソフトウェア開発のような知的作業を要求される領域では、ある作業には特定のスキルが求められ、そのようなスキルを持ち得る人しか、そのような作業を行うことができない。結果として、1人の作業者が複数の作業を担当することが珍しくない。従って、このような作業者の制約も考慮したプロジェクト管理が求められ、このようなプロジェクト管理作業を強力に支援する技術の開発が望まれる。ところがこれまでに開発されているソフトウェアプロセスに基づいた開発環境は、作業者間の協調作業の側面を主に支援するもの [1] [17] や、プロジェクト管理の側面を支援しているもの [10][16] についても、先に述べたようなさまざまな制約を考慮した管理者と作業者間の動的な協調作業を支援しているものはほとんどない。

これに対して、我々はデータベースを中核とした総合的なソフトウェアプロジェクト管理システムの構築を目指している。これまでにプロジェクト管理の基盤となるプロセスモデルを提案した。そしてリソースの日程上の制約を考慮した場合、このモデルをベースに、作業の遅延がプロジェクト全体にどのように影響を及ぼすかを指摘する影響波及解析に適用し、その有効性を示した [6][7]。また、工程管理の視点に対して、管理者が計画に基づいて行う作業指示から、進捗把握に基づくリスクの早期発見, リスクを早期に回避するために採る意思決定フェーズまでの作業を計算機支援により誘導するための枠組を提案した [8][9]。本稿では、これらのモデルについて説明するとともに、モデルに基づいたプロトタイプシステムを試作したので、その枠組を明らかにする。

以下本稿の構成を示す。2章ではソフトウェアプロジェクトの全体プロセスを記述するためのプロセスモデルについて概観する。3章では、工程管理における管理者と作業者間の動的作業誘導、すなわち、ソフトウェア開発の工程管理における作業指示から、作業者により報告される進捗情報に基づいたリスクの早期発見までの過程で、管理者が意思決定をするために必要な情報を知らせたり、管理者が採った意思決定に基づき必要な人に必要な情報を伝えるための枠組について述べる。4章で今回開発したプロトタイプについて述べ、最後にまとめと今後の課題について述べる。

2 ソフトウェアプロジェクト管理のためのプロセスモデル

本章では、システムの基盤となるソフトウェアプロジェクト管理のためのプロセスモデルについて述べる。本システムが対象とするソフトウェア開発プロジェクトは、最終的に開発されるソフトウェアシステムを作り出すために、複数人の作業者がツールやマシンを使いながら次々と多くの中間生成物 (プロダクト) を生成する過程であると考える。このことは概念的にはSADT[13]と類似の表記を使って図1のようなプロセスモデルとして表現できる。

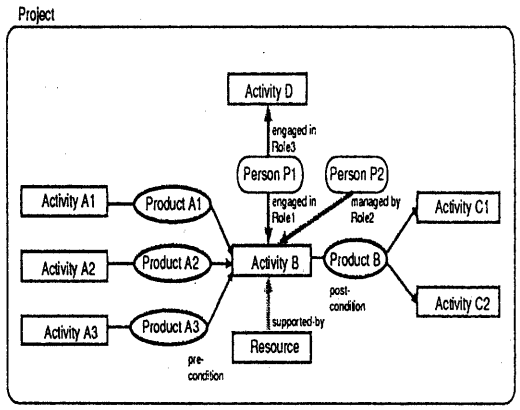

図 1: ソフトウェアプロジェクト管理のためのプロセスモデル

工程管理のための主要なオブジェクトとして、作業 (アクティビティ)、プロダクト、リソース、役割、プロ

ジェクトの5つを考える。プロセスモデルを構築するために行った考察等の詳細については [6][7] を参照されたい。

- アクティビティ (作業) : 割り当てられたプロダクトを作成するために行われる作業のこと。計画開始/終了日、実績開始/終了日、見積り工数、進捗データ、進捗状況 (ステータス) などプロジェクト管理に必要なデータを属性として持つ。
- プロダクト : アクティビティで作り出されるべきプロダクト。プロジェクト管理の視点からはアクティビティの進捗を定量的に計測する対象である。
- リソース : 作業を実行するための実行主体である。リソースには、人と、ツールやマシンといった非人間系のリソースがある。ここで重要なことは前節で述べたようにリソースの日程上の制約を表現するために、リソースの利用可能期間に関するカレンダ情報を持つことである。
- 役割 : プロジェクトという組織においては、さまざまな役割形態がある。たとえば、プロジェクト管理者、設計者、プログラマ、品質保証技術者などがある。

  ソフトウェア開発プロジェクトの工程が製造業における組み立てラインの工程などと異なることは、ソフトウェア開発プロジェクトでは、1人の作業者が同時に複数の作業を担い、しかも、作業ごとに異った役割を担って関与することが珍しくないということである。1人の作業者に役割の異なる作業を同時に割り当てることを可能とするためには、役割という概念を擬人化してとらえる必要があろう。ゆえに、役割を1つのオブジェクトととらえる。
- プロジェクト : プロジェクトで行われるアクティビティやプロジェクトの遂行に必要なリソースはプロジェクトとの間にプロジェクトを構成するという関連で規定される。またプロジェクトは、プロジェクト期間やプロジェクト予算、プロジェクトの対象領域の性質を表す属性を持つ。

これらの管理対象は、さまざまな関係により関連づけられている。

3 プロセスモデルと有限状態機械モデルに基づく作業誘導機構

本章では、工程管理におけるプロジェクト管理者と作業者間の情報通知による作業誘導をはかるための枠組について述べる。

ソフトウェア開発において行われるさまざまな作業 (仕様書を作成する、テストを実施する等) をプロジェクト管理の視点から見ると、計画時点で管理者により作業が定義され、各作業に対して作業者や予定実施期間が割り当てられる。そして、管理者はプロジェクト遂行時に作業者に作業指示を与える。一方作業者は、割り当てられた作業を行い、作業の進捗報告や終了時に承認を得るために管理者に報告を行う。プロジェクト管理では、このようにさまざまなアクションが引き起こされ、アクション起動時に各種データの設定/更新が行われる。設定/更新された情報は必要な人に適切なタイミングで通知されなければならない [8][9]。

そこで、1つの作業に対して管理者や作業者が情報の受渡しを行う際に起動するアクションと、アクションにともなって遷移する状態 (これを「ステータス」という作業の属性として表現する) を有限状態機械としてモデル化する。図 2に各作業に対して行われるアクションと、アクションによって遷移するステータス値を示す。図 2で丸はステータス値を、矢印はステータスの遷移を、矢印の上に起動されるアクションを表している。表 1に図 2で示された各アクションとその意味を示す。図は OMT[12] の動的モデルの表記に従った。さらにアクションを誰が起動し得るか、それぞれのアクションが起動されるタイミングでどのような情報が受け渡されるかを図 3にその一部を示す。実線がアクションの起動を、破線がアクションの起動に基づいて通知される情報の流れを示す。

管理者や作業者により各作業に対して引き起こされたアクションに基づき、システムはデータベースを更新し、必要なデータを算出し、作業ネットワークに規定された関係を参照して必要なデータを必要な人に適切なタイミングで通知する。何らかの理由 (突然の出張や研修等) で、ある作業が遂行できなくなり、スケジュールに変動があったとする。この時、これによって影響を受ける作業を指摘し、管理者に通知する。このことを実現するために、システムは引き起こされたアクションを契機として一連の処理を行う。また、ある作業を終える場合、その作業の作業者は作業を終了するというアクションを起こす。システムは作業の終了というイベントを受け、消費期間や遅れの有無、その影響波及を算出し、それらのデータとともにプロジェクト管理者に作業の終了を通知する。プロジェクト管理者はその作業の終了を承認する意味で承認というアクションを起こすと、システムはそのアクションを状態遷移のためのイベントとして受け取り、作業者に対

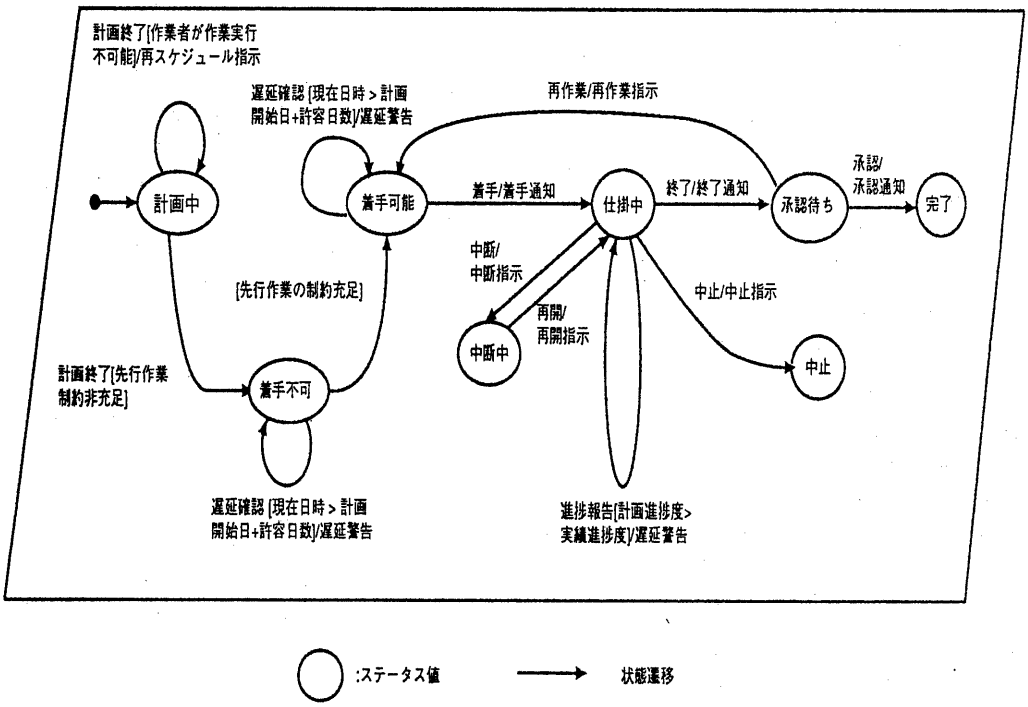

図 2: 作業に対して行われるアクションと状態遷移

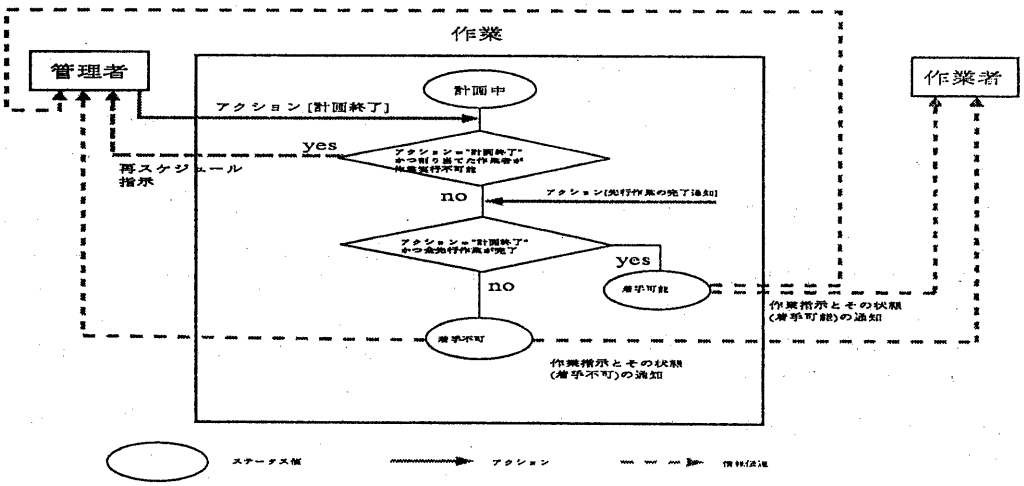

図 3: アクションとそれに基づく情報の流れ

表 1: アクションとその意味

アクション	意味
計画終了	管理者が作成した計画をプロジェクトの計画として確定し、各作業の担当者である作業者に通知するトリガー。ここで作業者の日程を考慮して、作業が割り当て可能か否かのチェックを行う。
着手	作業間の先行後続関係の制約充足により着手可能な作業に対して作業者が作業を開始するアクション。
終了	作業者が作業を終了した時に取るアクション。このアクションを契機としてシステムは管理者に作業の終了を通知する。
承認	終了と作業者により宣言された作業に対して、管理者が作業の承認を与えるアクション。このアクションが起動されると、システムは該当する作業の作業者に作業の完了を通知する。また、完了した作業の後続作業に関して、作業間の先行後続関係に基づき、着手可能か否かをチェックする。着手可能な場合には、作業を着手可能にし、担当作業者に対して作業の着手可能を通知する。
進捗報告	作業者が作業の途中進捗を報告するアクション。
中断	管理者が作業の中断を決定した時に起こすアクション。
再開	中断されていた作業を再開させるために管理者が起こすアクション。
再作業	終了と通知された作業に対してやり直しを求めるアクション。
中止	作業の取り辞めを指示するアクション。

して作業が承認されたことを通知するとともに、その作業の後続作業にメッセージを伝える。後続作業はそのすべての先行作業が完了していれば制約が充足され、「着手可能」状態になり、担当作業者に着手可能なことが通知される。それぞれのアクションが行われた時点の日付情報を履歴として記録する。プロジェクトの管理上重要な実績開始/終了日は、それぞれステータスを「仕掛中」「完了」にした時点での日付が記録される。

このようなステータスデータと作業ネットワークをデータベース化し、その変更が起きたタイミングでどのような処理を行うのかを手続きとして記述することにより、管理者や作業者がその役割に応じて起こしたアクションから、システムはその内容を監視し、必要な処理を行うことができる。

4 プロトタイプ

前章までで述べた枠組に対するプロトタイプを開発したので、その仕組みを明らかにする。

4.1 システムプロトタイプの機能概要

プロトタイプは、大きく分けてプロジェクトの計画を立案する機能と、プロジェクトの進捗を管理する進捗管理機能から構成される。

計画立案では、プロジェクトのプロセスモデルを定義するとともに、プロジェクトの遂行に必要なリソース等を定義する。そして、定義されたプロセスモデルに基づき各作業に作業者や日程を割り当てる。これらの一連の作業を行うためのモジュールがある。それらはプロジェクトデータ定義ツール、プロセスモデラ、スケジューラというツールを用いて行う。

また、進捗管理に関しては、プロジェクト遂行時に時々刻々と変化する進捗や状態に関連する情報を管理者と作業者間で交わすコミュニケーション支援と、作業の遅延による影響がプロジェクト全体にどのように波及するかを把握する機能を提供している。前者は、管理者と作業者間のメッセージ通信により行い、後者のために影響波及解析ツールを用いる。これらのツールが全体でどのように連係しているかについては、次節で述べる。

4.2 システムプロトタイプのアーキテクチャ

システムプロトタイプのアーキテクチャを図4に示す。作業誘導は、サーバ/クライアント方式によるメッセージ通信により実現している。本アーキテクチャは、Ben-Shaul らにより提案された複数利用者によるソフトウェア開発環境の一般的なアーキテクチャ[2]に類似している。彼らが提案したアーキテクチャをプロジェクト管理という複数人により構成される1つの実例への適用と考えることができる。また同じようなアーキテクチャは MELMAC 環境[3]でも見られる。

- 管理者用モニタ (サーバ)

  プロジェクトデータ定義ツール、プロセスモデラ、スケジューラを用いて管理者が作成した計画に基づく作業指示をクライアントである作業者に伝えるため、あるいは影響波及解析ツールに対してメッセージを送信する。管理者と作業者間で

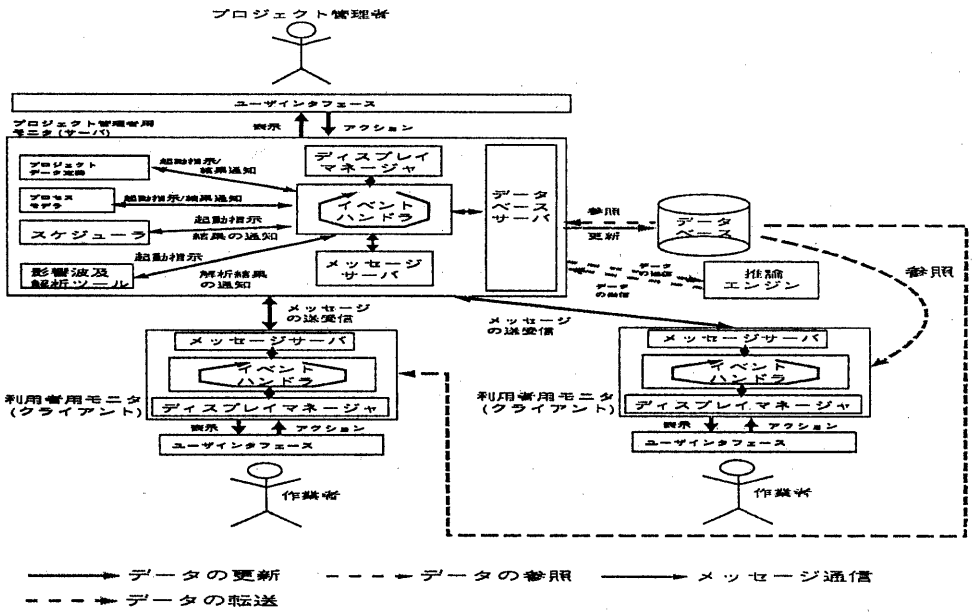

図 4: プロトタイプのアーキテクチャ

のメッセージの送受信はメッセージサーバを介して行う。また、サーバは作業の遅延警告や再スケジュール勧告を行うための推論エンジンとの間でデータの送受信を行う。さらに、プロジェクトのデータを管理するデータベースサーバとしての役割も担う。

本システムでは、プロジェクト管理者や複数の作業者が、非同期に作業指示を出したり、作業進捗を報告したり、承認を求めるためのアクションを起こしうる。それぞれのアクションに対応してデータベースが更新されるが、この時更新結果がプロジェクトの然るべき人に通知され、そのユーザインタフェース上に反映されなければならない。このようなデータベースの内容と表示情報の一貫性は、ディスプレイマネージャにより維持管理される。ディスプレイマネージャは、データベースのロック機能とメッセージ通信を用いて一貫性を維持している。

- 利用者用モニタ (クライアント)

  管理者モニタからの作業指示に関するメッセージの受信、作業者から発せられるアクション (作業の着手、進捗報告、終了 (承認依頼)) を管理者用モニタへ通知するメッセージの送出を行うメッセージサーバ並びに、メッセージの内容に基づきユーザインタフェースに表示する情報の一貫性を維持するディスプレイマネージャから構成される。

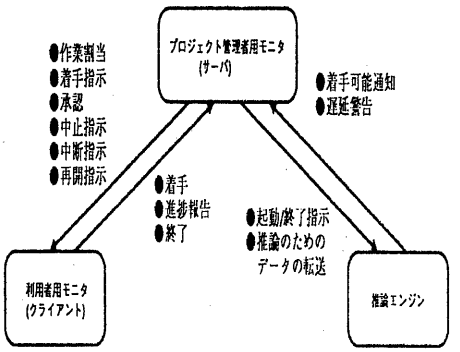

図 5: サーバクライアント推論エンジン間でのメッセージのやり取り

- 推論エンジン

  サーバからプロジェクトに関する情報を取得し、プロジェクトモデルとして記述された制約情報をもとに着手可能な作業についてサーバに通知したり、内部に保持している時間情報との比較により、作業の遅延警告をサーバに通知する。

3者間でのメッセージのやり取りを図5に示す。

本システムは、UNIXワークステーション上 (SUN SparcStation 2) で、データベースにオブジェクトデータベース Versant[15] を、ユーザインタフェースに XView を用いて実現している。プロジェクト管理者用ユーザインタフェースの画面イメージを図6に示す。

5 まとめ

本稿では、ソフトウェアプロジェクト管理における管理者と作業者間の作業誘導を計算機支援により支援するための枠組を提案した。すなわち、プロジェクト管理の側面から起動されるアクションとそれにより遷移する状態 (ステータス) を有限状態機械モデルとして提案した。状態を遷移させるトリガーとなる各アクション起動時には、さまざまな制約がチェックされ、然るべき人に適切な情報が送られる機構を提案した。各アクション起動時に通知されるべき情報の通知先は、プロセスモデルで規定される作業間の関連、作業とリソースの関連を利用している。そして、作業誘導に必要なコミュニケーション支援のためのアーキテクチャを示した。アーキテクチャとしてサーバクライアント方式を採用した。

本システムを用いることにより以下のような効果が期待できる。

- 管理者が作業者に対して作業者が持つ日程上の制約を考慮した作業指示を出すことができる。

  ソフトウェア開発では1人の作業者が複数の作業を受け持っていることが一般的であり、管理者は全体スケジュールと個人スケジュールを照らし合わせながら作業指示を出す必要がある。本システムでは個人の日程情報を保持し、作業指示時にそれを制約としてチェックしている。多くの作業者を抱えるプロジェクト管理者には有効な機能であると考えられる。

- 単純な作業指示や報告等を、システムの介在により自動的に行うことにより、コミュニケーションの漏れや遅れを減らすことができる

  利用者はプロジェクト管理上重要と思われるアクションをユーザインタフェースから起動することにより、そのアクションに関与する必要な人に必要な情報を送ることができ、プロジェクト管理におけるコミュニケーションに関する1つの問題を回避することができる

本稿では、有限状態機械モデルに基づいた管理者と作業者間の作業誘導の枠組を提案し、その実現について述べた。しかし、まだ解決するべき課題は多い。

現状ではステータス情報等のデータはすべて手入力である。これでは管理のためのオーバーヘッドになりかねない。ツールを使って作業を行う場合には作業からツールを起動する枠組を提供できれば、ツールの起動/終了のタイミングでステータスを変更させたり、作業の終了をシステムから問い合わせることができるようになる。あるいは実プロダクトと関連づけることにより、ある程度の進捗データを自動的に収集できると考えられる。この点を改良し、CASEの単なる1ツールにとどまらず、プロセスモデルをベースとしたソフトウェア開発環境 ([10][16][17] 等) に発展させ、ソフトウェア開発管理の統合的な枠組にしていきたいと考えている。

また進捗管理に関して、現在は作業終了時にその作業の遅延の有無並びに遅延による後続作業への影響の有無をシステムが管理者に通知している。遅延の影響がプロジェクト全体へどのように波及するかについては、影響波及解析ツールにより把握することができる [7]。この時、影響波及解析結果に基づき影響が大きい場合には、管理者は対策を講じなければならない。いわゆる計画の見直しである。計画の見直しに関しては、代替案の提示とその実現可能性に関する検証が必要になる。複数の実施可能な計画を提示するために、我々はこれまでに GA (Genetic Algorithm) を用いた計画立案機能を実現している [14]。現状はこのように各機能を実現するツールを個別に開発したところである。今後はこれらを統合させた枠組を具体的に検討していきたいと考えている。

さらに、現在はアクションの起動時にさまざまな制約をチェックするに留まっているが、潜在的なリスクを回避するためには、作業進捗等を予測する機構を実現する必要があると考えている。

謝辞

本稿作成にあたり、熱心に議論して下さいました当プロジェクトのワーキング委員の方々 (日本電気株式会社 秋口課長、岩崎主任、岡田主任、岸課長、阪田主任、垂水主任) に感謝致します。また日本電気株式会社小山田課長からいただきましたコメントは有益でした。また、本稿の初期の版にコメントを下さいました方々にも感謝致します。

参考文献

[1] 鷲坂恒夫 松本吉弘, ソフトウェアエンジニアリングデータベース KyotoDB の設計と実現, 情報処理学会論文誌 Vol.33 No.11, pp.1402-1413 1992.11.

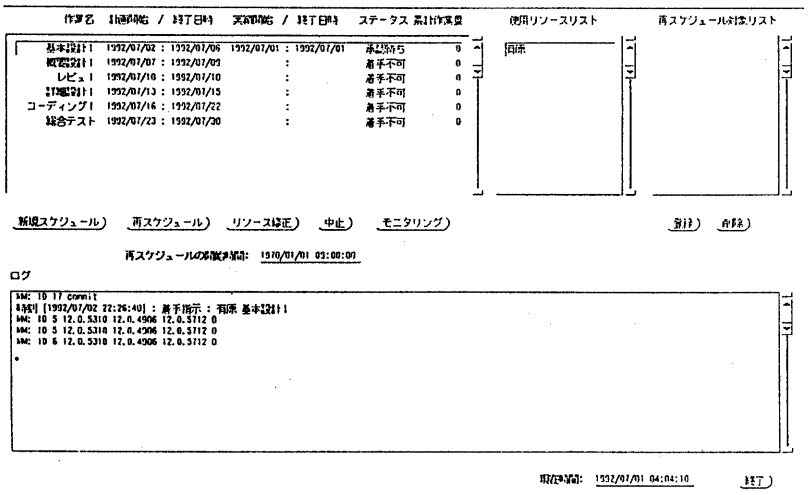

図 6: システムの画面イメージ (管理者用モニタ)

[2] Ben-Shaul,I., Kaiser, G.E., Heineman, G.T., An Architecture for Multi-User Software Development Environments, Proceedings of the 5th ACM SIGSOFT Symposium on Software Development Environments (SIGSOFT'92) Weber, H. (ed.), ACM Software Engineering Notes Vol.17, No.5, pp. 149-158, December 1992.

[3] Deiter,W., Gruhn,V., Managing Software Processes in the Environment MELMAC, Proceeding of the Fourth ACM SIGSOFT Synposium on Software Development Environments, page 193-205, Irvine 1990, ACM SIGSOFT Notes, 15(6).

[4] Dowson,M. (Ed.), Proceedings of the First International Conference on the Software Process - Manufacturing complex Systems-, IEEE Computer Society Press, October 1991.

[5] Proceedings of the Second International Conference on the Software Process -Continuous Software Process Improvement-, IEEE Computer Society Press, February 1993.

[6] Hazeyama, A., Komiya, S., A Process Model for Software Process Management, Proceedings of the Fourth International Conference on Software Engineering and Knowledge Engineering (SEKE'92) June 15-20 1992, IEEE Computer Society Press, pp.582-589.

[7] 櫻山 淳雄, ソフトウェアプロジェクト管理のためのデータモデルとその振舞いについて, 情報処理振興事業協会技術センター第 11 回技術発表会, pp.49-57, 平成 4 年 10 月 21 日.

[8] Hazeyama,A., Komiya,S., Software Process Management System Supporting the Cooperation Between Manager and Developers, Proceedings of the Fourth European Workshop on the Next Generation of CASE Tools (NGCT'93), Memoranda Informatica 93-32, Department of Computer Science University of Twente, pp.183-188, Paris France, June 7-8 1993.

[9] 櫻山 淳雄, 古宮 誠一, ソフトウェアプロジェクトの進捗管理に関する一方式, 情報処理学会グループウェア研究会, GW1-10 pp.75-82, 1993.4.28.

[10] Mi,P. Scacchi,W., Process Integration in CASE Environments, IEEE Software, March 1992, pp.45-53.

[11] Ould,M.A., Roberts,C., Defining formal Models of the Software Development Process, In Software Engineering Environments, Ellis Horwood Ltd, 1988, pp.13-26.

[12] Rumbaugh,J., Blaha,M., Premerlani,W., Eddy,F., Lorensen,W., Object-Oriented Modeling and Design, Prentice Hall, 1991 (邦訳: 羽生田栄一監訳, オブジェクト指向方法論 OMT プレンティスホール/トッパン).

[13] Ross,D.T., Structured Analysis(SA) for Communicating Ideas, IEEE Transactions on Software Engineering, Vol.3, No.1, 1977, pp.16-34.

[14] 澤部 直太, 古宮 誠一, 櫻山 淳雄, 東条 敏, 遺伝的アルゴリズムによるソフトウェア開発計画の立案, 電子情報通信学会 知能ソフトウェア工学研究会 KBSE93-4, pp.25-32, 1993 年 5 月 19 日.

[15] Versant System Reference Manual, Versant Object Technology, January 1992.

[16] Bourguignon,J.P., The EAST Eureka Project European Software Advanced Technology, in Software Engineering Environments - Research and Practice-, Bennett, K.H.(ed.) pp.5-16, Ellis Horwood, 1989.

[17] Process Weaver General Information Manual, Version PW1.0, Cap Gemini Innovation, 1992.