ソフトウエア工学 93－8
（1993．7．8）

ソフトウェアプロセスに関する研究の現状

落水 浩一郎
北陸先端科学技術大学院大学 情報科学研究科

概　要

ソフトウェアプロセスに関する研究成果の現状を、ソフトウェア工学分野におけるに他の主要な成果と関連づけつつまとめる。話題には以下のようなものが含まれる。(1) プロセス成熟度モデル、(2) ソフトウェアプロセスモデルのレベル、(3) プロセスアーキテクチャ、(4) ツール統合と CASE データベース、(5) 統合環境、(6) グループウェア。

Survey of the Research on Software Process

Koichiro OCHIMIZU

School of Information Science, JAIST

Reserch efforts on Software Process are surveyed, including (1) Process Maturity Model, (2) Level of Software Process Model, (3) Process Architecture, (4) Tool Integration, (5) Integrated Software Environment, (6) Groupeware.
Technical implication of software process to other major technical results in software engineering field is also discussed.

1 はじめに

ソフトウェアプロセスに関する研究は、従来、個別に研究がなされてきた、開発パラダイムや開発方法論、プロジェクト管理や品質管理、ＣＡＳＥツールとソフトウェア開発環境等のソフトウェア工学関連の諸技術(図1[1])を包括して互に関連づけ、さらに理論的基盤を与えうる可能性を持つ新しい研究分野である。本稿の目的は、ソフトウェアプロセスに関する現在までの主な研究成果を報告しつつ、本分野の研究に関心を持つ諸氏に、一つの展望を与えることである。

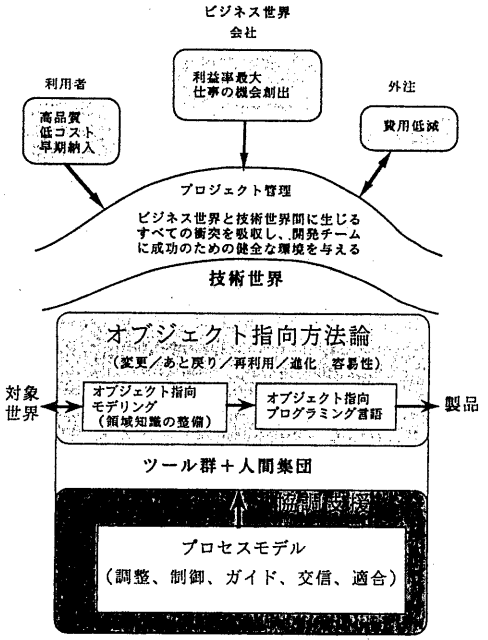

図１　ソフトウェア開発をとりまく環境

筆者は、ソフトウェアプロセスに関する研究の初期の段階から、ＩＣＳＥ，ＩＳＰＷ，ＩＣＳＰなどの国際会議や国際ワークショップに参加しつつ、ソフトウェアプロセスに関する研究の意義や課題(攻め方)を検討・考察してきた(時として、本研究の存在意義そのものに疑問を感じることさえあった)。しかし、W.S.Humphreyによる著作「Managing the Software Process」の訳書「ソフトウェアプロセス成熟度の改善」[1]を読了するにいたって、冒頭に述べたようにこの分野の研究の意義、必要性について確信を持つにいたった。そこでは、現在までのソフトウェア工学の成果と現状における問題点をふまえつつ今後我々が指向すべき方向を端的にまとめてある。

このような背景をふまえて、第2節、第3節では、「ソフトウェアプロセス成熟度の改善」の内容を簡単に紹介する。その枠組のもとで、第4節以降に、筆者がソフトウェア工学の技術的な面において、ここ2，3年の間に整理しえた知見[2, 3,4,5,6,7,8]を紹介する。紹介にあたっては個人的な研究の興味や見解はなるべく排除するようにした。

[1]図1におけるプロジェクト管理の定義は第4回ＪＳＰＷにおいて望月氏によってあたえられたものである。

2 プロセス成熟度モデル

品質の高いソフトウェアを経済的に作成する手段としての「銀の弾丸」[9]はなく、組織におけるソフトウェア開発手段の日常的な改善(プロセスの改善)が重要である。ソフトウェアプロセスを定義する目的は、作業のやりかたを改善することである。プロセスを秩序だてて考えることにより、問題を予測し、予防もしくは解決する手段を工夫することができる。問題の困難さがその解決に用いた手法の能力を超えた時に、エラーは発生する。このためには、ひとたび認知し対処したエラーは二度と起さないようプロセスを改善すべきであり、そのためには、プロセスを統計管理が可能である状態にしなければならない。このような立場を形式化したのが、W.S.Humphreyによる「プロセス成熟度のレベル」である。現在の組織のレベルをアセスメントにより認識した上で、段階的に上位の状態に移行することを推奨している。以下、図2に従ってその概要を紹介する。

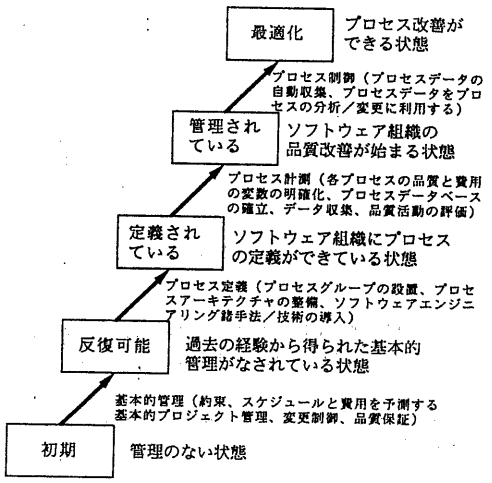

図２　プロセス成熟度のレベル

2.1 「初期」から「反復可能」へ

「初期」状態とは、正式な手順、費用見積り、プロジェクト計画をもたないままに作業を行なっており、

1. ツールとプロセスとはうまく統合されず

2. 変更制御は厳密でなく

3. 問題点や課題を知っていたり理解している上級管理者はほとんどいなく

4. 多くの危機的状況が先送りになったり、忘れられたりすることさえある

仕事を計画し追跡する公式の手順はあるかもしれないが、これが使われているかどうかを確かめる管理の仕組がなく、問題が発生したとき公式手順を放棄して、コーディングとテストに逆戻りする兆候をもつ。この状態を抜けだすには、スケジュールと費用に関する予測をおこなう基本的なプロジェクト管理をまず実施しなければならない。すなわち、約束、計画、変更制御、品質保証を充実する必要がある。このようにして「反復可能」の状態に移行できる。

2.2 「反復可能」から「定義されている」へ

「反復可能」状態とは、ある程度の統計管理(約束、計画、構成管理、品質管理にもとづく厳密なプロジェクト管理) はできる組織の状態である。ただし、このような管理は、今まで似たような仕事を行なってきた経験から得られたものであり、新たなチャレンジ(新しいツールや手法の導入、経験のない種類の製品開発、大きな組織変更) は大きなリスクをともなう。これを抜け出すには、

1. プロセスグループを設置する。プロセスグループは開発プロセスの定義、技術のニーズと利用機会の明確化、プロジェクトへの助言、四半期毎のプロセスの現状/実績の管理レビューをフルタイムでおこなう。

2. プロセスアーキテクチャを整備する。具体的には、開発サイクルを構造的に作業分解し、各作業を、前提条件、機能の記述、検証の手順、作業完了によって定義する。

3. 一連のソフトウェアエンジニアリングの手法と技法を導入する。すなわち、設計、ソフトウェアインスペクション、形式的設計手法、ライブラリ制御システム、テスト手法を導入する。プロトタイピングも新しい開発言語とともに考慮する。

必要がある。これらの努力を通じて、組織の状態は「定義された状態」になる。「定義された状態」において、はじめて、先進的技術を導入して役立てることができる。

2.3 「定義されている」から「管理されている」へ

「定義されている」状態は、プロセスを吟味し改善策を決定するための基礎固めができた状態であり、ソフトウェアの品質を左右する変量がまだ明確でない。定義されたプロセスは定性的なものである。これを改善するには、プロセスの定量的評価手段を持つ必要がある。すなわち、

1. プロセスの品質と費用を支配する変数を明確にするために、プロセス計測の最少の基本セットを確立し、各主要プロセス活動の費用と効果との相対関係を定量化する。

2. プロジェクトデータベースを確立し、プロセスの品質と生産性の分析を支援する。

3. プロセスデータを収集する。

4. 独立した品質保証グループが各プロジェクトの品質活動を評価し、品質計画を追跡する。

2.4 「管理されている」から「最適化」へ

「管理されている」状態においては、ソフトウェア組織の顕著な品質改善が始まる。ただ、収集されるデータは製品改良に直接関係するデータにかたよりがちである。しかし、例えば、ソフトウェアエラーの除去にあたって、その費用までを考慮にいれると、テスト法の改善にくわえて、設計やコードのインスペクションを充実させる方がより経済的であるという例からもわかるように、収集されるデータをプロセスの変更のためにつかうほうが効果が大きい。すなわち、プロセスの最適化が品質と生産性により多く寄与する。

「管理されている」状態から「最適化」状態に移行するにあたっては、

1. プロセスデータを自動的に収集する。

2. プロセスデータをプロセスを分析し改善するために利用する

ことが重要である。このような成熟へのステップの中で注意すべきことは、「規律」と「規格化」のちがいの認識である。ソフトウェア組織に規律性を導入することと、「規格化」し強制することは異なる。

3 ソフトウェアプロセスモデルのレベル

ここで若干の用語の定義をする。必ずしもコミュニティ全体で合意されているわけではないが、W.S.Humphrey

によれば[1]、ソフトウェアプロセスとは、ソフトウェアの生産および進化の中で使用されている活動、手法、および慣習の集合。ソフトウェアプロセスアーキテクチャとは、プロジェクト固有のソフトウェアプロセスを定義するための枠組。ソフトウェアプロセスモデルとは、ソフトウェアプロセスアーキテクチャをある特定の目的をもって具体化したものとある。さらにソフトウェアプロセスモデルは、U (Universal) モデル、W (World) モデル、A (Atomic) モデルの3レベルにわけて定義される。

3.1 Uモデル

ソフトウェアプロセスの方針レベルの表現である。例えば、「すべての作業は、ベースラインに統合される前にインスペクションを受けること」というように、開発の方針を定める。Royce によるウォータフォールモデル [10] や Boehm によるスパイラルモデル [11] などのように、従来、開発パラダイムと呼ばれていたものがこのレベルに相当する。

3.2 Wモデル

上記方針を実現するための、作業レベルの手順を定める。例えば、どの時点で品質保証レビューを行い、その結果指摘された問題をどのように扱うべきかを、手順で定義する。

3.3 Aモデル

Wモデルレベルの作業手順の要素を「作業活動 (アクティビティ)」と呼ぶことにすると、Aモデルは、作業活動の標準作業定義やツール利用法を指定する。例えば、コードインスペクション標準では、レビューすべきコード、時期、使用すべき手法、作成すべき報告書、許容できる効率限界を規定する。

4 U, Wモデル構築の基礎としてのオブジェクト指向方法論

ソフトウェア開発方法論は、ソフトウェア開発に携わる人々が共通の言語で話し、サブゴールとその達成手段を共有することを可能にする。Uモデル構築の基礎として、データを中心に設計することにより比較的長期間にわたってシステムの安定性を保証しうるオブジェクト指向分析/設計法 [12] は、以下に述べるこのましい特徴を持つ。

すなわち、図3に示すように、オブジェクト指向方法論は、**変更容易性、あと戻り容易性、再利用容易性、進化容易性**等の各面で従来の方法論よりすぐれている。すなわち、データとそれに付随する操作の型定義による一体化は、オブジェクト設計レベルでのデータ構造の変更を容易にする。また、モデルの構造とプログラムの構造の間にシンプルな対応関係があるので、構成管理機構と連動させることにより、ソフトウェア開発における「作業の手戻り」や対象世界の変化にともなうソフトウェアの「進化」を容易にする。すなわち、あいまいなものを徐々に具体化するソフトウェア開発作業においては、作業のあともどりは本来自然であるという認識に基づいて、業務世界のモデル化の構造と計算機上での実現の構造をすべてオブジェクトという単一の枠組で定義することにより、対応構造をシンプルにしている。最後に、クラス階層を上位に向かって構築することよりえられる有用なスーパークラス [8] は、あらかじめ計画された進化 (時間軸上の変化に対する共通性の定義) や再利用 (類似の対象領域における共通性の定義) の基礎を与える。財産としてのスーパクラスは、モデリングにおいては発見のための思考を節約させ、プログラミングにおいては、再コーディングの労力を節約させる。

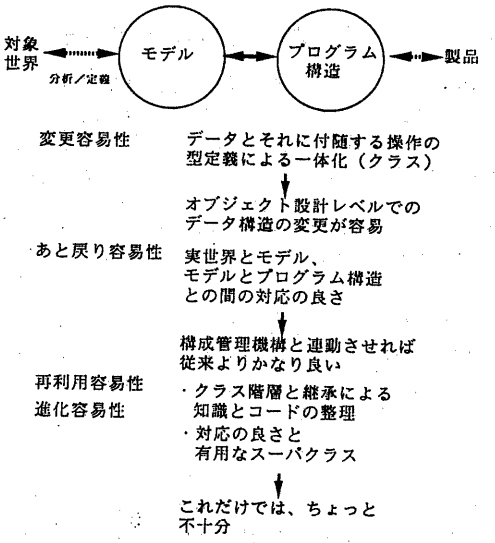

図3 オブジェクト指向 分析／設計／プログラミング方法論の特長

Wモデルの整備は今後の課題であろう。すなわち、オブジェクト指向方法論と一口にいっても、その詳細な作業手順は、対象システムの特徴によって、図4に示すようにシステム設計のポイントは異なってくる [12]。すなわち、図4における◎印のついた部分を、過去の設計事例を再利用しつつ重点的に定義し、さらにプロセッサやデータベースに割りつけていく作業プロセスは十分に定義される必要がある。

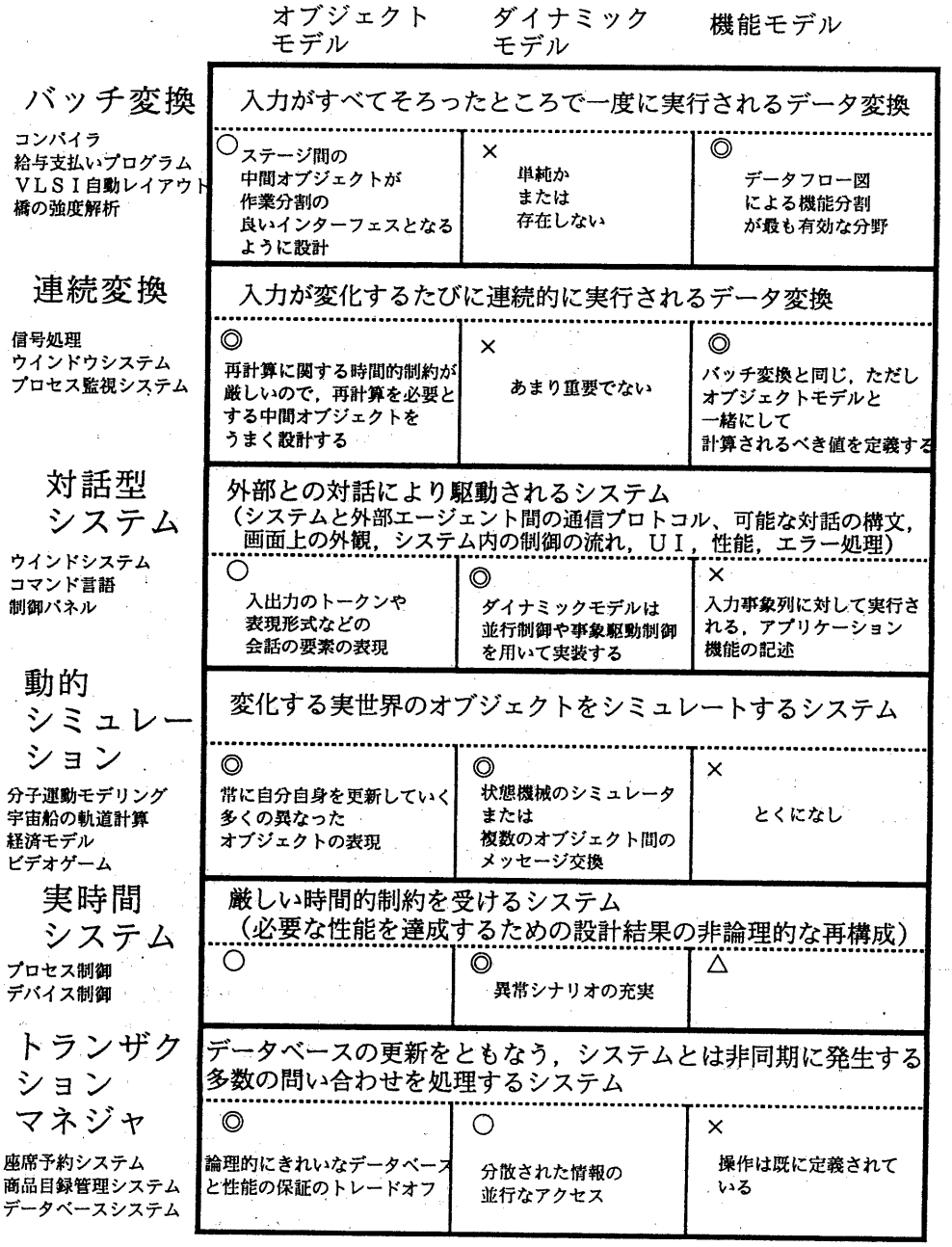

図４　モデル定義の比重とポイント

5　Aモデルにおける知識の整備

ソフトウェア開発方法論の記述は当然のことながら一般的であり、使いこなすには、対象分野に依存する知識や開発経験に依存する知識を整備する必要がある。すなわち、開発方法論はいくつかの基本アクティビティからなり、その中にはスキルの高い人間を割り当てるべき高度なアクティビティがある。一方、開発コストと人的資源の量を考える時、常に最適解をプロジェクト立案の段階で設計するわけにはいかない。一般に、質を作りこむアクティビティには評価基準が附随しておりこれがいわゆるノウハウの素であると考えられる。設計方法論の教科書に書かれている評価基準の一般的記述は開発の最前線の人間に有効ではなく、このような一般的記述を、対象分野に依存する言葉に翻訳する必要がある。対象分野や開発経験に依存する知識を獲得し[13]、開発方法論の枠組に張り付けることは負荷の軽減と危険の予防をはかる意味で重要である。

6　プロセスアーキテクチャ

多くの主要なソフトウェアプロセス問題は品質、製品技術(機能充足性、性能満足度)、要求の不安定さ(未知の要求、不安定な要求、誤解された要求)、複雑さに関係している。これらの問題に対処する一つの手段はプロセスアーキテクチャの整備である。それらの問題に対する解(方針)をUモデルで表現し、WモデルやAモデルで詳細化するにあたって統一的な記法で表現できることが望ましい。文献[1]においては、図5に示すような基本ユニットセルを分解・合成することで種々のレベルのプロセスの記述を可能にする手段を定義している。

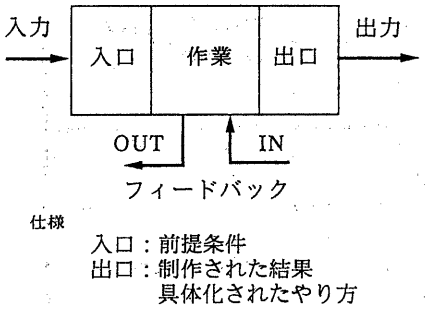

図5　基本ユニットセル

7　ツール統合機構とCASEデータベース

Wモデルのレベルで定義される種々の作業の支援はCASEツール単独では可能でない。作業毎のツールや、作業グループ毎に選択されたツールキットを利用するのが普通である。このとき、ツール統合の機構が必要となる。「ツール統合」の概念は1970年代中期に出現した。すなわち、ツール機能をツールフラグメント群の合成機能としてとらえ、ツールフラグメントの再利用を促進しようとする技術である。下流CASEツール統合機構としては、ツールフラグメント間のインタフェースをパイトストリームとして定義しコマンド行におけるパイプとフィルタの機構を利用してそのつど必要なツール機能を統合するUNIXのアプローチが代表的であり、生産現場における多くの成功実績がある。

今後、技術開発の目標となるのは、開発工程全体にわたって作業の連続性を保証するツール統合機構である。現在、初期のプロトタイプとしてPCTE等の製品があらわれている[14]。PCTEはソフトウェアデザインオブジェクトを管理するオブジェクト管理システム(ERAモデル表現)およびツールプロセスの管理とプロセス間通信の支援機構等を有する。1983年に開発が開始され、1986年にはPDSとしてのPCTE1.4が配布され、セキュリティ管理機構が強化されたPCTE+が1989年、C言語との結合が可能なECMA PCTEが1991年に順次発表されている[15]。現在C++との結合が進行中である。以上については、よくまとまった解説がある[16]。

すでに市販されている種々のCASEツールのいくつかは、現在、PCTEとの結合の作業が進行中であり、数年以内には、限られた範囲(会社の研究所、大学等)で利用実績が積み重ねられるものと予想される。

ツール統合機構の設計の中で最も重要なのは、データベースの設計である。文書化体系や設計方法論に対応する中間生成物(ソフトウェアオブジェクト)とその品質基準やソフトウェアオブジェクト間の依存関係を定義することにより開発の目標と基準を設定する。また、開発進行時の構成管理によりチーム構成員の活動を調整(Coordination)する。分析や設計に関わる情報構造モデル(デザインモデル)を応用分野毎に確立すること、および、Top-Downに認識されていく情報をCASEデータベースに投入していけば、データベース内部ではデザインモデル(スキーマ)に従って情報が整理され、定義の不備な点をデータベース自身が通知してくれるようなCASEデータベース環境が必要である。

8　統合環境

ソフトウェア開発環境とは、プロセスを確立する方法ではなく、効果的なプロセスをより一層効率的にするも

のである。プロセスモデルの確立が自動化のきっかけを作り、環境の充実がそれを実現する。代表的なアプローチは、80年代後期にL.Osterweilによって「ソフトウェアプロセスを記述することにより、開発工程全般にわたって、規範性と自動化の度合を高めうる」というプロセスプログラミングの接近である[17]。彼の提案に刺激されて、日常のソフトウェアエンジニアリングに関する諸活動を体系的に記述するための様々の(計算)モデルが提案され、そのモデルを実現するためのプロセス記述言語が設計された。その言語の実行環境がプロトタイプレベルで開発されつつある現状である。Wモデルレベルの記述法については多くの研究成果がある[18,19,20]。

プロセスモデルに基づく統合環境の開発はわれわれに**柔軟な標準化の手段を与える**ものと思われる。すなわち,

1. 組織やプロジェクトチームの特性に適合するプロセスモデルをまず定義し、
2. それに関係するツール、プロセス記述、データベーススキーマ、通信プリミィブ機構等をとりだし
3. カスタマイズした後に、
4. それらを統合する

すなわち、方法論や技術の進歩に応じて、対応する部分を抽出・変更することが可能になる。

これが実現されるためには、Adaptation機能を開発・充実させる必要がある。ここで、Adaptation機能とは、文書化体系、ソフトウェアプロジェクト構成員間の情報交換支援、資源割当てや開発スケジュール等のプロジェクト計画支援、危機状態の検出や対応等のプロジェクト監視支援等に関わる種々のデータベース機能やツールの集合(メタSDE)から、**ユーザ固有の作業形態に適合した開発環境**(specific SDE)を、**プロジェクトの規模**[21]、**対象分野の特性、チーム構成員の分散度、データの安全性や機密度、利用する計算機設備、コスト等のパラメータに応じて生成または選択抽出し合成する機能**である[4]。

以上述べてきた諸技術の成果は、以下のようなサービス機能を有する統合環境という形で、そのプロトタイプが世の中に提供されることを期待したい[5]。

(Coordination支援) 開発対象物に対する目標と基準の設定、および個々人の活動の調整

文書化体系や設計方法論に対応する中間生成物(ソフトウェアオブジェクト)の定義。依存関係、品質基準等の定義。および開発進行時の構成管理。

(Navigation支援) 質のよい中間生成物を作成するための手順の支援

開発方法論の枠組や詳細手順を与え、またソフトウェアオブジェクトの状態を変化させる操作(ツール機能)を統合化する役割をもつ。さらに熟練者の作業プロセスの再利用を促進することにより、初心者の教育に関する熟練者の負荷を軽減するねらいもある。

(Communication支援) ソフトウェアプロジェクト構成員間の情報交換支援

ソフトウェア開発時に生じる、合意形成、意志伝達、技術情報の問い合わせ等によって生じるチーム構成員間の情報交換の支援

(Control支援) 集団活動の制御

資源割当てや開発スケジュール等のプロジェクト計画支援。および危機状態の検出や対応等のプロジェクト監視支援

(Adaptation支援) 作業形態に適合した開発環境の提供

上記4つの機能を、開発形態を特徴づける種々のパラメータに応じて、ユーザ固有の作業形態に適合させる機能。

これらの相互関係を図示すると図6のようになる。

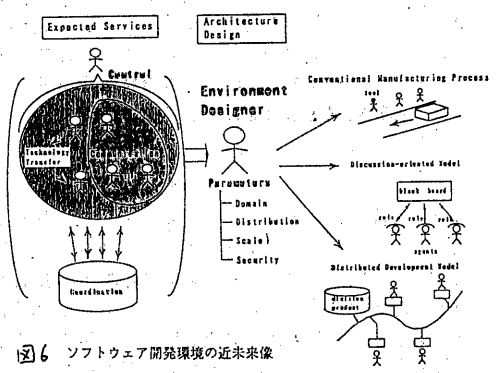

図6 ソフトウェア開発環境の近未来像

9 グループウェア

文献[1]で必ずしも十分議論されていないのが、プロジェクトチームや組織内におけるコミュニケーション支援の問題である。一般に、ソフトウェア開発は、必ずしもスキルレベルがそろっていない専門家集団の協調作業である。チームの構成員は、過去のシステム開発経験より得られた知識を再利用しつつ、CASEツールを使用して作業の自動化の度合を高めながら各自に課された課題を解決していく。割り当てられた仕事を、各自が首尾よく達成しつつ全体としては、共通のゴールを共有・理解してプロジェクトを成功に導くためには、

1. データベース技術が提供する成果物の管理機構
2. ネットワーク技術が提供する情報交換機構
3. プロセスモデルが提供する
   (a) - プロジェクト構成員への仕事のガイドライン
   (b) - 作業局面に応じたCASEツール群の統合機構

等の要素技術を融合して、

(情報交換機能) 不完全で部分的な知識の統合

(プロセスの監視機能) 生成される情報群の正確な伝達と状況の把握

(データベースとの結合) 各自の作業分担における視野やコンテキストの設定

に関する有効な支援が実現されなければならない[6]。

ソフトウェア開発に特化されたグループウェアの形態は、分散・非同期タイプに属するものと思われるが、現状はまだ必ずしも充分とはいえない。たとえば、

1. ゴールを達成するために満たすべき制約をクレアしつつ中間生成物を作成していく**専門家達のコミュニケーション(協調)を支援**するためには話題に応じて気楽に情報を交換できるというような e-mail の効果[22]だけでは必ずしも十分ではなく、プロジェクト進行中に生じる異常状態を検知(できれば予測)し、それに対して適切な対応をとるためにはコミュニケーション支援単独ではなく成果物管理機構やプロセス監視機構等も考慮にいれてモデル化することが必要である

2. また、構造的思考のツールや構造的コミュニケーションの媒体としてハイパーテキストが注目をあつめている。しかし、問題点がまだよく理解されておらず、その結果、思考にあいまいさと若干の混乱があり、思考の推移が頻繁におこなわれるような状況下では、ある人の考えを要素分解して表現することは不自然であり、ハイパーテキストがカバーしにくい領域である。また、ハイパーメディアを通じて複数人の協調作業を支援するという立場からは以下のような問題がある[23]。

   ハイパーテキスト・ツールは、細かな粒子度で,なるべく分割された形でアイディアを表明することを利用者に要求する。このことは、逆に著者の大きな意図を不鮮明にする。また、作業内容に応じた適切な視野を設定したり必要な情報を得るためのコンテキストを与えるという点からは現状ではまだ不十分である。

以上のように、ソフトウェア開発の世界を対象とした時、現状技術がカバーする範囲はまだ少なく、ソフトウェア開発に特化された本格的なグループウェア(データベースと結合した情報交換機構)の出現が望まれる。

10 今後の動向

ソフトウェアエンジニアリングに関する研究は、ソフトウェアライフサイクルの概念の形成とその要素技術の確立からはじまり(７０年代)、上流工程の形式化と自動化による革新を経て(８０年代前半)、現在人間要因の分析と支援法の模索にまで、その対象領域が拡大されつつある(８０年代後半)。すなはち、７０年代においては、標準化による、生産性と信頼性の向上手段が追求され、８０年代前半においては、ユーザ要求への適合度や生産の自動化の度合を高める手段が追求され、さらに、８０年代後半から、品質を支配する主要因である人間の行動特性の分析や人間集団のコミュニケーションの支援法が追求されつつある。

このような流れの中で、現在までの一連の研究成果を統合して、さらに発展しうる可能性をもつソフトウェアプロセスに関する研究がめばえてきたことは意義あることである。

本稿でのべてきた研究内容の進展を把握しておくためには

1. ソフトウェア工学国際会議(ＩＣＳＥ)
2. ソフトウェアプロセス国際ワークショプ(ＩＳＰＷ)
3. ソフトウェアプロセス国際会議(ＩＣＳＰ)
4. ソフトウェア科学会 ソフトウェアプロセス研究会

における研究活動をフォローしておくことが必要である。

一方、ソフトウェアプロセスに関する研究グループは分散開発時代への移行を考慮にいれた研究対象の拡大が必要である。すなわち、２１世紀に向けて、ソフトウェア開発の形態は、各拠点に分散して存在する人的資源、知的財産を論理的に統合して実施する方向に動きつつある。最近のインターネットワーキング技術の出現と進歩により、上記の要請を満たすためのハードウェア基盤は整備されつつあるが、その上に構築されるべきソフトウェアやシステム化の技術はいまだ未成熟であり、以下のような基礎研究の推進が必要である。

1. 分散開発における、開発チームの構成法、作業分担法、プロダクトの管理法等に関するソフトウェア・プロセスモデル
2. 分散データベース、分散オペレーティングシステム、インターネットワーキング等の要素技術を前提として設計された環境アーキテクチャ
3. 上記アーキテクチャを駆使して、協調作業を行なう人々のための仕事の進め方(開発方法論)をオブジェクト指向分析/設計法に基づいて定義したもの。
4. 分散オブジェクト指向開発方法論に基づくソフトウェア開発を支援するためのソフトウェア開発環境

5. 分散開発におけるチーム構成員間の交信支援、プロジェクト管理支援を目標とするソフトウェア開発に特化されたグループウェア

（1）および（5）が新しいプロジェクト管理の基礎となり、（2）上に実装された（3）や（4）が分散環境の実現を推進するであろう。

また、ソフトウェア工学の基礎的な成果をとりいれつつ、会社独自の生産体制の整備・充実に力をそそいできた組織とそうでない組織の間には生産性において大きな開きがある。また、組織内においても作業の熟練者と初心者の間には生産性に大きな開きがあり、ソフトウェア開発に関わる人的資源の少なさからこの傾向はますます顕著になってきている。ソフトウェア開発に関する諸技術の共通性を括り出し(プロジェクト依存、組織依存、業種依存、汎用)、より共通性の高い基盤技術については社会的分業体制を整備する必要があろう。

参考文献

1. 藤野喜一監訳、「ソフトウェアプロセス成熟度の改善」、日科技連、1991.
2. 落水、春原、「ソフトウェア工学」、情報処理学会30年のあゆみ第7章、pp.205-211.
3. 落水、"ソフトウェア開発方法論とCASEツール", 情報処理学会第42回全国大会チュートリアル・セッション資料、1991年3月、pp.45-59.
4. C.Tully, "SDE Architecture Issues", Proceedings of the 6th International SOFTWARE PROCESS WORKSHOP, 1990, pp.31-36.
5. 落水,"CASEとは", 日本ソフトウエア科学会サマーチュトリアル資料、1991.
6. 落水、"ソフトウェア開発におけるグループウェアの役割"、ソフトウェア・ツール・シンポジウム'92、1992、pp.83-92.
7. 落水、" ソフトウェア開発環境の動向とAI技術への期待"、知能ソフトウェア工学の動向と展望シンポジウム、1992、pp.11-16.
8. 落水、[ソフトウェア工学実践の基礎ー分析・設計・プログラミング編」、日科技連出版会(近刊)
9. F.P.Brooks、"No silver bullet, essence and accidents of software engineering", IEEE Computer, April 1987.
10. W.W.Royce、"Managing the development of large software systems"、Proceedings of IEEE WESCON, Aug. 1970.
11. B.W.Boehm、"A spiral model of software deveopment and enhancement"、IEEE Computer,May 1988.
12. J. Rumbaugh et al.、"Object-Oriented Modeling and Design"、Prentice Hall,1991.
13. K.Ochimizu,T.Yamaguchi,"A Process Oriented Architecture with Knowledge Acquisition and Refinement Mechanisms on Software Process", 6th ISPW,1990,pp.145-147.
14. M.I. Thomas,"PCTE Interface: Supporting Tools in Software Engineering Environments", IEEE Software, Nov.,1989.
15. "Standard ECMA-149 Portable Common Tool Environment(PCTE) Abstract Specification",ECMA, 1990.
16. 鰺坂、沢田、満田、"Emeraude PCTE", コンピュータソフトウェア、Vol.10 No.2, 1993, pp.65-77.
17. L.Osterweil,"Software Processes are Software Too", 9thICSE,1987, pp.2-6.
18. L.Osterweil,R.Taylor,"The Architecture of the Arcadia-1 Process Centered Software Environment", 6th ISPW, 1990,pp.155-158.
19. G.E.Kaiser and P.H.Feiler,"An Architecture for Intelligent Assistance in Software Development", 9th ICSE, 1987,pp.180-188.
20. M.I.Kellner and H.D.Rombach, "Comparisons of Software Process Descriptions",Proceedings of the 6th ISPW, 1990, pp.7-18.
21. D. E.Perry and G.E.Kaiser,"Models of Software Development Environments", 10th International Conference on Software Engineering, 1988, pp.60-68.
22. Martha S. Feldman,"Constraints on Communication and Electronic Mail", CSCW'86, 1986, pp.73-90.
23. J.Conklin and M.Begeman,"gIBIS: A Hypertext Tool for Exploratory Policy Discussion", CSCW'88, 1988, pp.140-152.