ソフトウェア工学 109－6
（1996．5．17）

分散並行開発支援環境ＰＲＩＭＥの開発と実践

青山　幹雄*　　吉松　朋聖†
新潟工科大学*　富士通株式会社†

ソフトウェアプロセス支援環境ＰＲＩＭＥ(Process Information ManagEr)の開発と大規模分散並行開発への適用経験を報告する．『全ての作業をＰＲＩＭＥの中に』の理念に基づき，プロセス，プロダクト，プロジェクトの視点からソフトウェア開発のマクロ構造モデルを提案する．このモデルに基づき，ＰＲＩＭＥの３階層クライアント／サーバアーキテクチャ，実現，評価を述べる．さらに，本環境を実践した経験から，プロセス支援環境のあり方と今後の課題を議論する．とくに，開発者の視点に立った柔軟なプロセス支援のために，『あそび』の概念と管理水準の設定，プロセスの非線型性の概念を提案する．

Development and Experience of A Process-Centered Software Engineering Environment PRIME

Mikio Aoyama*　Tomokiyo Yoshimatsu†
Niigata Institute of Technology*　Fujitsu Limited†

This article reports on the development and experience of a process-centered software engineering environment *PRIME (Process Information ManagEr)*. The basic concept of PRIME is to provide a common vehicle for individual developers, on which we can seamlessly ride along with software processes. Modeling the software development activities from the process, product and project points of view created a three-tier client/server architecture of PRIME. Practice of the PRIME in the development of a family of large-scale communications software is discussed. Among them, *levels of management*, *play mechanism* and *non-linearity* are proposed.

1．はじめに

ソフトウェアプロセス支援環境ＰＲＩＭＥ(PRocess Information ManagEr)の開発と大規模分散並行開発への適用経験を報告する．

プロセスプログラミングの提案以来ソフトウェアプロセスについての研究，開発が行われている．特に，ソフトウェア開発の主体は人であることから，一人一人の作業を支援するプロセスガイドやプロセスに基づくプロジェクト管理機能などを提供するプロセス支援環境ＰＳＥＥ(Process-centered Software Engineering Environment)が開発されている[Fink94, Garg94, Pene93]．プロセス支援環境は，ＣＡＳＥ環境を統合する枠組みとしても期待されている．さらに，開発者一人一人にパーソナルコンピュータ（ＰＣ）やワークステーション（ＷＳ）が普及した結果，プロセス支援環境を実践する素地が整ってきた．しかし，大規模プロジェクトでのＰＳＥＥの開発，適用評価については報告が少ない．

一方，筆者らの従事する大規模通信ソフトウェアの開発では，分散並行開発を行っている[Aoya95]．分散並行開発では，プロセスの実行と管理が複雑であることからＰＳＥＥの活用が望まれている．

このような背景に基づきプロセス支援環境ＰＲＩＭＥを開発した．ＰＲＩＭＥは，試行，評価を経て 1993 年から大規模通信ソフトウェアの分散並行開発に適用している．本稿では，ＰＲＩＭＥの開発理念，アーキテクチャ，実践経験とその教訓を述べる．

2．開発理念：すべての作業をＰＲＩＭＥに

ＰＲＩＭＥ開発における基本理念は『すべての作業をＰＲＩＭＥの中に』である．これは，開発プロセス，ツール，開発情報と資源の統合を通して，ＰＲＩＭＥが全ての開発，管理を支援する共通プラットフォームの役割を担うことを意味する．一方，ＰＲＩＭＥの実現はできるだけ市販のソフトウェアを組み合せられるよう，オープンな環境を目指している．

3．アプローチ

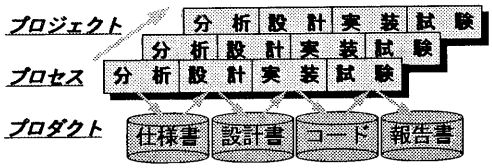
3.1 実現の考え方

ＰＲＩＭＥの開発，適用にあたってはプロセスを中心とする開発のモデル化を行い，そのモデル上で実行すべき共通開発プロセスを定めるなどの開発全体の見直しを行った．

筆者らは，ソフトウェア開発を，図－１に示す３つの側面から捉えている[Aoya90]．

ソフトウェア開発
＝プロセス＋プロダクト＋プロジェクト

(1) プロセス：一つの開発ライフサイクルにわたり作業を実行する側面．
(2) プロダクト：ライフサイクルに沿った各プロセスの入出力（成果物）の側面．
(3)プロジェクト:複数チームが複数版にわたりプロセスの実行を行う側面．



図－１ ソフトウェア開発のモデル

3.2 ＰＲＩＭＥのアプローチ

ＰＲＩＭＥの開発理念を実現するために，次のアプローチを採った．

(1) 開発者個人の支援

開発者の視点から個人作業の支援を目標とした．一人一人の作業状況に応じてきめ細く支援する．

(2) 集団的活動の支援

個人からプロジェクト全体まで，開発者集団が円滑に開発を進めるための『社会基盤』を提供する．上記の３つの側面を社会基盤の観点から見ると，次の点を考慮する必要がある．

1) プロセス：プロセスは，全ての開発者の『行動原理』の役割を果たすべきだと考えている．プロセスの実行に沿ってその実行状況に関する情報を自然に蓄積し共有できる仕組みが必要である．さらに，工程毎に開発者が異なる場合が

多いので，工程間で動的に情報を共有できる仕組みも必要である．

2) プロダクト：プロセスの入出力であるプロダクトの情報もプロセスの実行に沿って自然に蓄積，共有できる仕組みが必要である．

3) プロジェクト：大規模プロジックトでは，階層的に開発の計画，実行，管理が行われる．この階層構造がソフトウェア開発の集団的活動の基本構造を形成する．このため，プロセスの実行と管理に関する情報をプロジェクト全体で共有できる仕組みが必要がある．

(3) 環境としての支援

スケーラビリティ：プロセスやプロジェクトの変化に対応し，かつ，異なる規模のプロジェクトを支援できること．

1) オープン性：ＰＲＩＭＥの情報を市販のＤＢＭＳやスプレッドシートで利用できること．

2) 分散性：遠隔地の開発拠点と分散並行開発を行っているので，広域分散開発環境上で利用できること．

3) 視覚性：作業の実行，管理が目で見て分かるように視覚的な表現が可能であること．

4．モデル

4.1 モデル化

ＰＲＩＭＥの実現に当たっては，プロセス，プロダクト，プロジェクトの３要素を中心に開発のモデル化を行った．

(1) 開発プロセスの標準化

開発プロセスを形式的に表現するため木構造チャートＹＡＣを拡張した表記法を開発した[Aoya90]．さらに実際の開発プロセスを分析し，標準開発プロセスとして定義した[Aoya94]．

(2) 開発プロダクトの標準化

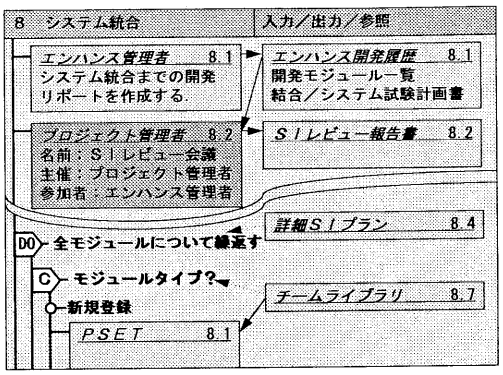
対象プロジェクトでは，主要なプロダクトは既に標準化されていたが，多くは紙で管理されていた．開発プロセスの標準化と合わせて，主要なドキュメントの電子化を行った．

(3) 開発プロジェクトの標準化

個々のプロジェクトの構造を整理し，共通の構造モデルを構築した．さらに，対象プロジェクトの形態に適する実行モデルを定義した．これは，プロセスの実行を管理するスケジューラの役割を果たす．

4.2 標準開発プロセスモデル

標準開発プロセスをＹＰＬを用いて定義した[Aoya94]．図－２にシステム統合プロセスの例を示す．

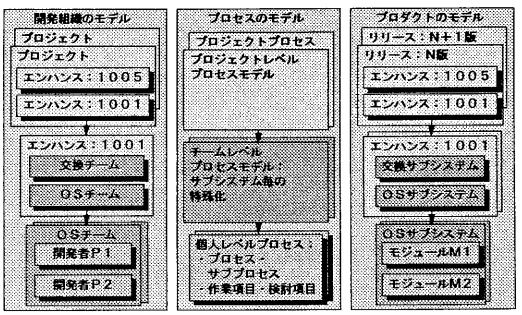


図－２ 標準開発プロセスの例

4.3 開発プロジェクトモデル

(1)構造モデル

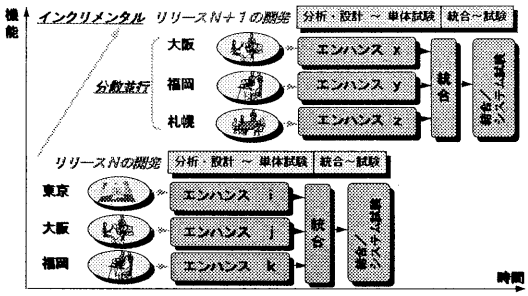
プロジェクトの構造を整理し，プロセスとプロダクトの視点から図－３に示す３階層でモデル化した．ここで，エンハンスとは幾つかの機能の集まりであり，論理的な概念である．



図－３ 開発プロジェクトモデル

(2)実行モデル：分散並行開発モデル

対象プロジェクトは，図－４に示すように，プロジェクトが並行して繰り返し実行される分散並行開発を採った[Aoya90]．



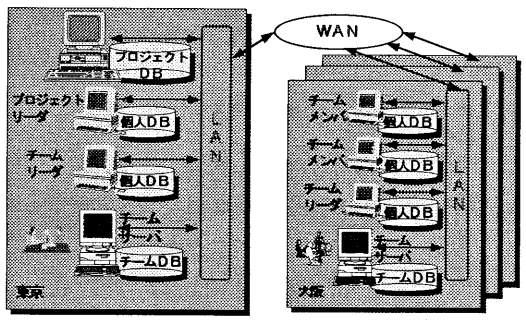
図－4 分散並行開発の動的モデル

5．アーキテクチャ

5.1 アーキテクチャ

(1) オペレーションアーキテクチャ

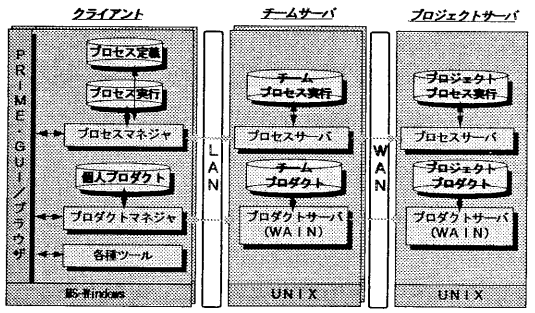
図－5に示すように個人，チーム，プロジェクトの3階層から成るクライアント／サーバアーキテクチャを採る．



図－5 オペレーションアーキテクチャ

(2) ソフトウェアアーキテクチャ

ソフトウェアアーキテクチャを図－6に示す．プロダクトはWAIN(Wide-Area Information Network)[Aoya96]で管理する．



図－6 ソフトウェアアーキテクチャ

5.2 実　現

ＰＲＩＭＥの実現では，次の理由からプロトタイピングによるインクリメンタルな方法を用いた．

(1) 変化への対応

開発途中でＰＲＩＭＥの出力帳票やユーザインタフェースなどが利用者の要求や使い方によって変わりうる．実際，ＰＲＩＭＥの開発中や適用後も多数の機能追加要求があり，プロトタイピングは有効であったと評価している．しかし，プロトタイピングのためには，ある単位，例えばサブシステム単位で，システムの開発ができるアーキテクチャを採ることが必要である．

(2) リスクの低減

ＰＲＩＭＥは，数百台のクライアントと数台のサーバから成る大規模広域クライアント／サーバシステムである．設計にあたっては，機能面のみならず性能，信頼性，通信トラヒックなど考慮すべき条件が多く，かつ，それらの関係が複雑である．この種のシステムの開発方法論は確立されているとは言えず，ある程度試行錯誤的な方法を採らざるを得なかった．開発リスクを低減する目的でプロトタイピングを行ったが，予想以上の開発期間を要した．

(3) オープン化

ＰＲＩＭＥの実現にあたっては，できる限りオープンなインタフェースを使い，市販の開発素材を活用する方針を採った．例えば，ＤＢＭＳは，ＸＢＡＳＥとＳＱＬを利用できる製品を使用した．グラフ出力はスプレッドシートを利用した．ＧＵＩは，市販のＧＵＩ構築ツールを利用した．この結果，短期間で効率的なプロトタイプの開発と変更が可能であった．

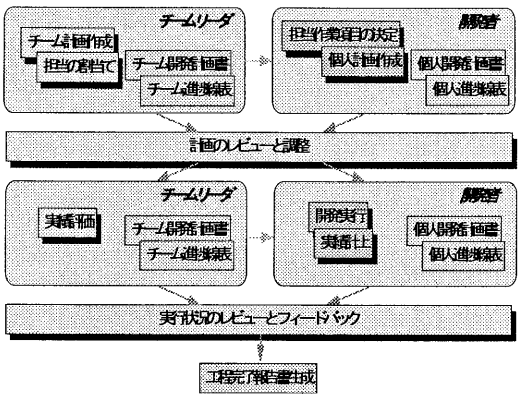
6．適 用と評価

6.1 適　用

ＰＲＩＭＥは1993年初めより試行を行い，同年6月より筆者らが従事する大規模通信ソフトウェア開発で適用している．さらに，同年 12月より関連する大規模通信ソフトウェア開発でも適用した．1994年には，いくつかの中小規模プロジェクトでも適用を開始した．

6.2 適用プロセスの例

設計工程におけるＰＲＩＭＥの適用例を図－７に示す．計画作成と指示はトップダウンで実績の収集はボトムアップとなる．開発者は，リーダの作成したチームの開発計画をウィンドウ上で参照しながら自己の開発計画を作成する．



図－７　適用例

しかし，工程毎に，かつ開発者毎に作業内容が異なるので，リーダと開発者は作業内容や計画のレビューと調整を共同で行う．

設計に着手すると，一つの設計作業項目が完了する毎に実績として情報が収集される．工程完了時には，実績のまとめとして個人やチーム毎に『工程完了報告書』が生成される．個人の情報は個人のＤＢに，チームの情報はチームサーバのＤＢに蓄積される．このように，ライフサイクルを通して必要な情報を一貫して収集，管理，共有できる．

6.3 ＰＳＥＥアーキテクチャの評価

ＰＳＥＥのアーキテクチャについての基本的なコンセンサスはまだ得られていない．ＰＲＩＭＥは個人，チーム，プロジェクトといった開発に係わる要素間の情報管理を中心とするアーキテクチャを採っている点でＭＡＲＶＥＬなどに近いと思われる[Pene93]．しかし，ＰＲＩＭＥでは，クライアント／サーバアーキテクチャ上で分散情報管理を行う点で，これらのシステムと異なる．

一方，分散処理環境上でＰＳＥＥを実現した例としてSynerVisonがあるが，大規模開発の構造的な支援や広域分散処理環境の支援が明らかでない．ＰＲＩＭＥは，広域分散処理環境上の大規模開発の支援を目的として階層的なクライアント／サーバアーキテクチャを採った．

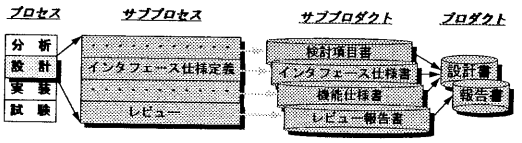
クライアント／サーバ間の機能分散は，ＭＶＣ(Model-View-Controller)フレームワークに基づくアーキテクチャをとった．個人支援の重要性を考慮すると，クライアント側で多くの支援機能を提供する必要があるので，ＰＲＩＭＥのアーキテクチャでは，クライアントの機能分担が大きくなった．

7．実践の教訓

ＰＲＩＭＥの実践にあたって，実行と管理の両面から次のような工夫をした．

7.1 プロセスの実行と管理

(1) プロセスの粒度　開発プロセスの実行，管理の粒度を一般的に規定することは困難である．図－８に示すように，ＰＲＩＭＥでは，各工程の作業をもう一段詳細なレベルで捉えることとした．



図－８ サブプロセスとサブプロダクト

1)サブプロセス：プロセスの観点から各工程をもう一段階詳細化した作業単位．

2)サブプロダクト：サブプロセス毎の成果物．

各工程毎は10個程度の『標準サブプロセス』に分割した．さらに，チーム毎に必要であればサブプロセスを追加できるようにした．サブプロセスは進捗と対応しており，サブプロセスの実行が自動的に進捗報告となる．

(2)プロセス管理：管理水準の設定と『あそび』

各工程やサブプロセスの実行では，重複や手戻りを許容することとした．実際，プロセスの実行は一様に進まないので，逐次的で硬直的な実行の制約や表現は開発の実態に合わない．むしろ，ある管理水準を設定し，それより詳細な

レベルでは実行順序などを規定しない方が無理がないと思われる．これは，機構設計におけるあそびの概念と対応している[Aoya94]．また，この方策はプロセスの実行，管理の仕組みを損わない．重要なのは妥当な管理水準を規定し，分かることと分からないことを明確にすることである．プロジェクト管理でしばしば問題となるのは，管理水準を規定していないため，何が管理できて，何ができないかを区別できない点にある．

(3) 共通／個人プロセスによるプロセス相互作用の隠蔽

個人プロセスの詳細な管理は現実的ではない．一方，管理の観点からは，プロセスの進捗などの動的な挙動を把握できる必要がある．さらに，開発者やチーム間でのプロセスの相互作用を防ぐ必要もある．従って，開発者やチーム間で相互に影響を及ぼすプロセスの抽象度を基に，他に影響を及ぼすもう一つの管理水準を設定した．この水準のプロセスを『共通プロセス』とし，それより詳細なプロセスを『個人プロセス』とした．これは，情報隠蔽の考え方をプロセスに応用したものである．

(4) 実行の強制

ＰＲＩＭＥでは，プロセス定義はあくまで参考であり，プロセス定義に沿った開発作業の実行を強制しない．これは，次の２つの理由による．

1)クライアントにＰＣを使用するため基盤環境の弱さからプロセス定義の強制がかえって生産性を損なう．

2)開発者の多くが長年同一プロジェクトに従事してきたことからプロセスの理解が進み，むしろ，強制を必要としない状況にあった．

しかし，プロセスの実行と管理が乖離すると実態が把握できなくなる．このため，サブプロセスの実行と進捗を一体化することにより，開発の実態に沿った情報の収集を可能とした．

7.2 分散並行開発の支援

適用プロジェクトでは，複数の開発チームが複数の拠点に分散し，並行して開発する分散並行開発を行っているので，次の点に留意した．

(1) 分散開発の支援

分散開発では，次の問題に対して効果を狙った．

1)分散開発拠点での開発作業の水準維持

分散した各開発拠点では，開発プロセスが分化し，開発作業が不均質となる．ＰＲＩＭＥを通して，開発作業の品質を確保する．特に，プロセスに基づく共通の尺度で開発を評価できる点が重要である．実際，ＰＲＩＭＥの運用形態はチーム毎に異なる．詳細なプロセスまで指示や管理を行っているチームもあれば，管理水準を満たすレベルでの適用，管理に留めるチームもある．しかし，ＰＲＩＭＥで定めた管理水準を満たせば，それ以上の適用方法は各チームに任せた．

2)チーム間の情報共有

ＰＲＩＭＥを通して設計情報を共有したり作業の実行状況をリアルタイムに把握できる効果は大きい．例えば，試験工程では，毎日の定例で，ＰＲＩＭＥのデータを用いて試験の進捗や品質の評価を行っている．

(2) 並行開発の支援

並行開発の問題は，次のように，プロセスとプロダクトの相互干渉と同期にある．

1)プロセス相互作用：あるチームの進捗遅れが並行開発中の他のチームや後続の開発に影響する．

2)プロダクト相互作用：ある機能の変更が関連する機能を損なったり，実行条件に影響する．

ＰＲＩＭＥは，開発プロセス全体にわたって，プロセスとプロダクトの情報を開発の進捗に沿って蓄積できるので，プロセスとプロダクトの観点から他のチームのプロセスの実行状況や対象とするプロダクト情報を相互に，かつタイムリーに参照できる．例えば，ソフトウェアの統合プロセスでは，複数のチームが更新を行う『競合モジュールリスト』を生成し，更新の順序制御を支援する．

7.3 プロジェクト独立性

ＰＲＩＭＥは，複数の開発部門で適用されている．部門毎にプロセスや実行管理方法は異なるが，本質的な差はないと考えている．むしろ，開発リーダの嗜好，組織の習慣など開発組織の属人的，文化的要素が強い．このような組

織固有の特性を無視することはできないが，異なる開発モデルを支援することも難しい．

ＰＲＩＭＥの適用部門は比較的関連があったので，ＰＲＩＭＥの開発チームがリーダや開発者と一緒に仕様の調整や運用方法を検討した．この結果，部門毎のカスタマイズをせずに適用している．しかし，一部の機能は使用していない部門もある．

7.4 使われなかったツール

ＰＲＩＭＥで開発したツールが必ずしも全て使われたわけではない．例えば，個人のスケジュール管理機能のプロトタイプを開発した．しかし，個人が既に日常的に利用しているスケジュール管理と重複するなどの理由から本格システムの開発は取り止めた．

7.5 ＰＳＥＥのあり方

ＰＲＩＭＥの実践から，ＰＳＥＥを実用化するためには，現在提案されているＰＳＥＥにはない次の概念を考慮すべきである．

(1) 個人開発プロセスの支援機能

従来のプロジェクト管理では，管理単位がチーム毎に異なっていたり，チームやプロジェクなどのマクロな管理しかできなかった．ＰＳＥＥでは，個人の開発プロセスを支援すべきである．個人プロセスの支援を通して個人毎のプロセス改善を図るべきだと考えている．これは，ＰＳＰ(Personal Software Process) [Hump94]と共通するアプローチである．

(2) プロセスの非線型性の支援

プロセスの実行はウォータフォールモデルのように直線的に進まない[Perr94]．設計や実装上の考慮点は複数あるが各考慮点は一様に解決が進まないからである．さらに，後戻りも許容する必要がある．本稿では，この動的特性を『非線型』と呼ぶ．これに対して逐次的なプロセスの実行を『線型』と呼ぶ．多くのＰＳＥＥはマクロなプロセスの支援が中心であるため線型プロセスを基礎としている．しかし，個人プロセスの支援では，このようなＰＳＥＥは実体に沿わない．

(3) 個人プロセスの実行における『あそび』の許容

個人プロセスでは，非線型性のために，プロセスの実行は個人やその実行状態により異なる．画一的なガイドや強制は好ましくない．むしろ，管理水準以下では『あそび』を許容する支援方法が必要である．

8．今後の課題

8.1 ＰＲＩＭＥの評価と改善

ＰＲＩＭＥ適用の評価と改善を行う必要がある．

ＰＲＩＭＥの実現にあたっては，現行の基盤環境のさまざまな制約を受けている．さらに，開発プロセス支援の観点から見ると本質的な課題も残っており，今後，改善を図る必要がある．

特に，無干渉性(Non-Intrusiveness) の点で問題が指摘されている．ＰＳＥＥは開発者の通常の作業に干渉を及ぼすべきではないと考えている．ＰＲＩＭＥは，ＰＣ上でも動作する必要があるためＰＧＵＩで疑似マルチタスクを実現した．しかし，開発者への干渉は否めない．今後は，ウィンドウ環境を活かした干渉のない支援環境の実現を図る予定である．

8.2 ＰＲＩＭＥの活用とプロセス改善

筆者らの部門ではＰＲＩＭＥを基盤環境として日常的に使用している．今後は，プロセスの定量的評価，改善を通してＰＲＩＭＥを活用する仕組みを確立する必要がある．

8.3 ＰＳＥＥアーキテクチャの評価

一般の基幹業務プロセスを支援する分散環境として，ＳＡＰ　Ｒ／３ [Buck95]などが開発，実践されている．これらの環境を含むアーキテクチャの評価を行う必要がある．

8.4 ＰＳＥＥのあり方

ＰＳＥＥのあり方や実現については議論が多い．特に，人がかかわることに起因する問題は多様である．本稿でも提案した『あそび』や『管理水準』などの概念を活かして，人による多様性を受け入れるＰＳＥＥのあり方を研究する必要がある．このためには，日本的生産システム

[Aoya95]や建築[Alex85]などの関連分野でのプロセスについての考えや社会的，心理面からの考察[Wein71]から学ぶべき点も多いと思われる．

9．まとめ

「プログラミング」という言葉は多様な活動を包含している[Wein71]．さらに，ソフトウェア開発が人による集団的で社会的な活動であることから，その構造を閉じた世界として捉えることには困難がある[Aoya94]．これは，ソフトウェア開発の全ての活動をプロセスを用いて逐一規定したり制御することは困難であることを意味している．むしろ，プロセスはソフトウェア開発という一つの社会の『行動原理』と考えるべきである．ＰＲＩＭＥの使命は，全ての人にこの行動原理を提示することにより，詳細な規則を廃した柔軟で快適な生活環境を提供することにある．

ＰＲＩＭＥの重要な役割は，プロセスの改善を通して，ソフトウェア開発の生産性と品質を向上することである．このためには，まず，現行のプロセスとその実行過程を目に見える形に表現し，理解する必要がある．その意味で，筆者らの開発部門ではＰＲＩＭＥは基盤環境として重要な役割を果たしている．

本稿で報告したように，ＰＲＩＭＥの実践からＰＳＥＥのあり方についての幾つかの知見を得た．しかし，これは，開発プロセスを理解する第一歩にすぎない．今後，ＰＲＩＭＥの実践と収集データに基づき開発プロセスとその支援環境のあり方，開発プロセスの改善などについて理解を深める必要がある．

最後に，ＰＲＩＭＥの開発と実践にあたりご協力いただいた富士通(株)と関係各社の各位に感謝する．

参考文献

[Alex85] C. Alexander, *The Producion of Houses*, Oxford University Press [中村 博 (監訳), パターンランゲージによる住宅の建設, 鹿島出版会, 1991.

[Aoya90] M. Aoyama, "Distributed Concurrent Development of Software Systems: An Object-Oriented Process Model", *Proc. IEEE COMPSAC '90*, Nov. 1990, pp. 330-337.

[Aoya94] 青山幹雄, "People Meet Process: ソフトウェアプロセスとの出会いと対峙", ソフトウェア技術者協会 ソフトウェアシンポジウム '94 論文集, Jun. 1994, pp. 54-62.

[Aoya95] 青山幹雄, "ソフトウェアプロセス・リエンジニアリング", 情報処理, Vol. 36, No. 5, May 1995, pp. 442-447.

[Aoya96] 青山幹雄, 岩見泰夫, "大規模ソフトウェア分散並行開発における設計情報の共有と管理", 情報処理学会 1996 年情報学シンポジウム講演論文集, Jan. 1996, pp. 73-80.

[Buck95] R. Buck-Emden and J. Galimow, *Die Client/Server-Technologie des SAP-Systems R/3*, Addison-Wesley, 1995 [SAP R/3 システムのクライアント／サーバテクノロジー, アーバン・コネクションズ, 1995].

[Fink94] A. Finkelstein, et al. (eds), *Software Process Modelling and Technology*, John Wiley & Sons, 1994.

[Garg94] P. K. Garg and M. Jazareki, " Selected, Annotated Bibliography on Process-centered Software Engineering Environments", *ACM Software Eng. Notes*, Vol. 19, No. 2, Apr. 1994, pp. 18-21.

[Hymp94] W. S. Humphrey, " The Personal Processin Software Engineering", *Proc. 3rd Int'l Conf. on Software Process*, Oct. 1994, pp. 69-77.

[Pene93] M. H. Penedo and W. Riddle, "Process-sensitive SEE Architecture (PSEEA): Workshop Summary", *ACM Software Eng. Notes*, Vol. 18, No. 3, Jul. 1993, pp. A78-A94.

[Perr94] D. E. Perry, et al., " People, Organizations, and Process Improvement", *IEEE Software*, Vol. 11, No. 4, Jul. 1994, pp. 36-45.

[Wein71] G. M. Weinberg, *The Psychology of Computer Programming*, Van Nostrand Reinhold, 1971 [木村 泉ほか (訳) プログラミングの心理学, 技術評論社, 1994].