成果物間の関連に着目した開発プロセスモデル：PReP

田中 康[†,††] 飯田 元[†††] 松本 健一[††]

ソフトウェアプロセスの適切なモデル化は，開発のフレームワークを提供するとともに，継続的なプロセス改善のベースラインを提供する意味においても重要な課題である．しかし，従来のプロセスのモデルは，変化を続ける現実の開発活動を適切にモデル化することが困難なために，当初定義したプロセスモデルが利用されず形骸化してしまう問題が起こっていた．そこで我々は，継続的なプロセス改善への適用に有効なプロセスモデルとして，成果物間の関連に着目したプロセスのモデル化方法「PReP（Product Relationship Process）」モデルを開発した．実際の開発プロジェクトに適用した結果，PReP モデルは，現実の開発活動のモデル化，理解の容易性，モデル化の柔軟性，そして再利用性にすぐれていることを確認した．PReP モデルを使用することにより，開発活動の実行とプロセスの継続的改善に効果的に利用できるプロセスモデルを定義することができる．

PReP: Product-based Modeling Method for Software Process

Yasushi Tanaka,[†,††] Hajimu Iida[†††] and Ken-ichi Matsumoto[††]

Appropriate modeling of software is a critical issue because it not only provides a framework for development process but it also provides the baseline for continuous process improvement. However, since the conventional process modeling cannot adequately describe actual development activities. This has given rise to the problem of people not using the define process models and letting them become empty shells. This prompted us to develop a product-based process modeling method PReP (Product Relationship Process). PReP describes actual development activities, provides the framework for development, and it is also more effective in providing a baseline for a process improvement program. When applied to an actual development project, we were able to establish that the PReP Model was excellent in modeling actual development activities and is easy to understand. The results also showed that PReP had a high degree of flexibility and reusability. With the PReP Model, we can define a process model which can be used effectively for executing development activities and for improving processes.

1. はじめに

近年，ソフトウェア開発の生産性および品質の向上を実現するために，多くのソフトウェア開発組織で CMM [1] および CMMI [2] 等に基づいた開発プロセスの改善活動が行われている．ソニー株式会社においても CMM および CMMI を利用したプロセス改善に取り組んでいる．

プロセス改善を行うためには，プロセスの外在化，すなわちモデル化が必要である．プロセスをモデル化することにより，改善のためのベースラインを定義することができる．そして，定義したプロセスを実際のソフトウェア開発に適用し，その結果を評価することにより，プロセス上の問題の特定と，改善方法の検討を行うことができる．さらに，定義したプロセスは，開発プロジェクトの実行と管理において，計画策定や進捗管理に利用される．このように，プロセスの改善と開発プロジェクトの実施において，プロセスモデルは重要な要素である．

CMM および CMMI では，組織のベストプラクティスから定義したプロセスを「組織の標準プロセス（以下「標準プロセス」）」と呼んでいる．ソニー株式会社においても，2000 年 4 月より CMM を参照モデルとしたソフトウェアプロセスの改善活動を進めてきている．その活動の中で，標準プロセスの定義と利用に取り組んでいるが，定義した組織の標準プロセスが利用されない，プロジェクトの計画策定や進捗管理に利用

† ソニー株式会社
Sony Corporation
†† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
††† 奈良先端科学技術大学院大学情報科学センター
Information Technology Center, Nara Institute of Science and Technology

可能な開発プロセスモデルを定義できないといった問題をかかえていた．このため，開発現場の要員にとっても記述が容易で，しかも形骸化することなく繰り返し利用されるようなソフトウェアプロセス記述を得られる，新たなプロセス記述モデルが必要であるとの認識に至った．

近年のソフトウェアプロセスに関する研究活動を調査した結果，奈良先端科学技術大学院大学（以下，NAIST と略記）において行われている，プロセスモデルに対するコンポーネント化とパターン化（プロセスパターン）の試み[3),4)] に注目した．プロセスパターンとは，ソフトウェアプロセスモデルに対してパターン化による再利用を進めようという試みであり，Gamma らによるソフトウェアデザインパターンの手法[5)] 同様，Alexander が建築分野で提案したデザインパターンの概念[6)] をソフトウェアプロセス分野に応用したものである．これらの概念を現場レベルで迅速に適用・検証を行うことを 1 つの目的として，NAIST の社会人学生としての立場でプロセスモデルにおけるパターン化の研究を開始した．

当初の活動として，XML 形式で記述されたインタフェースに基づいて定義されたタスクを一種のソフトウェアコンポーネントとして扱うことで，タスク自体の再利用や，タスクどうしの組合せをパターンとして定義し，再利用することを試みた．これらの成果はタスク指向のプロセスモデル Daibutsu-Den [4),7)] としてまとめることができたが，一方，並行して企業内で進めていた組織標準プロセス定義の試みを通じて，LSI 等ハードウェアと密着した組み込みソフトウェア開発を行っている現場のエンジニアにとって，本モデルは難解であり，適用が困難であるとの意見があらかじめ得られた．さらに，タスク指向のモデルよりもむしろ成果物指向による記述が直観的で親しみやすく再利用にも有望であるとの見通しが得られたため，成果物指向の観点からプロセスモデルの見直しを行い，再構成を行った．このようにして得られたプロセスモデルが，本論文において提案する PReP モデルである．Daibutsu-Den モデルにおけるタスクの定義方法等は PReP モデルに継承される一方，プロセスパターンは成果物間の依存関係として再整理されている．

本論文では，まずプロセス改善とプロセスモデルの関係を整理し，プロセスモデルに対する要件を定義する．次に，プロセスのモデル化のアプローチ方法を整理したうえで，成果物の関連によるプロセスモデル（以下「PReP モデル：Product Relationship Process モデル」と呼ぶ）とその利用方法を提案する．最後に，実際の開発プロジェクトへの適用結果から，PReP モデルの有効性をプロセスモデルに対する要件と照らして考察する．

2. プロセス改善とプロセスモデル

2.1 プロセスモデルの要件

プロセス改善とは，組織の開発活動でのベストプラクティスをプロセスの形式で定義し，定義したプロセスを個々の開発プロジェクトに適用し，その実施結果から適用したプロセスを評価し改善する継続的な活動である．Humphrey らは，プロセスモデルはプロセス改善における重要な要素であり，次のように利用されるとしている[8)]．

1. プロセスに関する効果的なコミュニケーションの実現

プロセスの利用者，プロセス定義者，プロジェクト管理者，そして開発者間のプロセスの理解と実行に関する基礎を提供し，個人の開発活動を支援する．

2. プロセスの再利用の促進

プロセスの定義には非常に大きなコストがかかる一方で，個々の開発プロジェクトには自分たちのプロセスを独自に開発する時間および作業者の十分な割当てがなく，再利用による効率の向上が必要である．

3. プロセス改善の支援

正確で理解しやすく，拡張と再利用が可能なプロセスモデルは，プロセスの学習のための有効な手段であり，プロセス改善のための重要な要素となる．

4. プロジェクトのプロセスの管理

プロジェクトを効果的に管理するためには，計画に関する明確な理解とプロジェクトの状況に対する正確な把握が要求される．プロセスモデルは，正確にプロジェクトの状況を把握するための基準と手段を提供する基礎となる．

さらに Humphrey らは，上記に示したプロセスの利用から，プロセスモデルが満足すべき要件として下記の 3 項目を定義している[8)]．

(1) 現実に行われている，または行われるべき活動をモデル化できること
(2) プロセスのモデル化と改善を行うために十分であるとともに，柔軟で理解が容易であること
(3) 必要とする粒度でのプロセスの洗練が可能であること

定義された標準プロセスが形骸化する大きな原因としては，従来のプロセスの定義方法が，Humphrey の示した上記の要件を満足していないことが考えられる．そこで，上記要件を満足するプロセスモデルについて

表 1 プロセスモデルの抽象度とモデル
Table 1 Abstraction level and modeling method of a process model.

抽象度	プロセスモデルの抽象度	モデルの利用	モデルの例
高	**Universal レベル**	プロセス概念の理解	Waterfall model, Spiral model 等のタスク観点でのモデル
	Worldly レベル	実際の開発活動の理解・改善・管理・再利用	EPMs[8]
低	**Atomic レベル**	特定の作業手順の記述 作業の自動化	プログラミング言語，ETVX[15]，ペトリネット等のタスク観点でのモデル

改めて検討した．まず，プロセスモデルを抽象度の観点から分析し，次にプロセス改善に適したモデルを特定した．

2.2 プロセスモデルの抽象度

Humphrey らは，ソフトウェアプロセスモデルの抽象度を，**表 1** に示すように,「Universal」,「Worldly」,「Atomic」の 3 段階のレベルで定義している[9]．Universal レベルはプロセスの概念を提供する抽象度の高いモデル化のレベルである．Universal レベルでのプロセスモデルとしては，“Waterfall model”[10),11)]，“Spiral model”[12]，“Iterative enhancement”[13] 等があり，これらのモデルは，プロセスを概観するために有効である．

一方，抽象度が低く特定の作業手順をモデル化するレベルは Atomic レベルと呼ばれている．Atomic レベルは作業手順を詳細にモデル化し，標準化された手順を提供する．Atomic レベルには，プログラミング言語やペトリネット，状態遷移図，形式文法等を用いたモデル化の方法があり，作業手順の記述のほか，作業の自動化のためのモデルとして有効である．標準プロセスの定義においても，Atomic レベルのプロセスモデルは，局所的なプロセスの記述として標準作業手順の定義に有効である．

Universal レベルと Atomic レベルとの間の抽象度で，実際の開発活動をモデル化するレベルは Worldly レベルと呼ばれている．Worldly レベルは，開発活動の実体をモデル化するレベルで，開発プロジェクト全体で行われる活動の順序を示し，それぞれの活動の開始・実行の条件とその結果を定義する．Worldly レベルのモデルは，開発プロジェクトの計画を詳細に立て，進捗管理の基準となるモデルを提供するために有効であり，さらに，プロセス改善の検討や，プロセスの再利用にも適している．2.1 節で示した要件を満足するには，Worldly レベルを中心としたプロセスモデルが必要であると考えられ，このような要件を満足するプロセスモデルの定義方法を，プロセスの記述視点から検討する．

2.3 プロセスモデルの記述視点

プロセスモデルの記述は，タスク（作業）に着目したモデルと，作成される成果物に着目したモデルの 2 通りの方法に主に分類される[14]．

2.3.1 タスク観点のプロセスモデル

プロセスをモデル化する従来の方法の多くは，プロセスをタスクの観点から定義するものである．タスク観点によるプロセスモデルとは，意味を持つひとまとまりの作業を「タスク」として定義し，定義したタスク間の時間的順序関係を記述するものである．表 1 のモデルの例に示したように，Universal レベルのプロセスをモデル化する Waterfall model，Spiral model 等はタスク観点でのプロセスモデルである．また，Atomic レベルをモデル化するプログラミング言語や ETVX[15]，およびペトリネット等を用いたモデルもタスク観点でのプロセスモデルである．

前節において，改善のためのプロセスモデルとしては，Worldly レベルを中心としたプロセスモデルが必要であることを述べた．Worldly レベルのプロセスをタスク観点で記述することも原理的には可能であるが適切ではない．なぜなら，プロジェクト全体で行われる実際の活動をタスクの時間的な順序関係としてモデル化してしまうと，複数のタスクが関係し合った非常に複雑なモデルとなってしまうからである．また，現実の開発活動においては，状況の変化に応じてタスクの順序の見直しがしばしば行われるため，定義したプロセスモデルが，すぐに現実の活動と乖離してしまい，結局は利用されなくなる[9]．

2.3.2 成果物観点でのプロセスモデル

タスク観点と異なるプロセスのモデルとして，成果物観点によるモデルがある．成果物とは,「作業の成果

としての実体であり，振舞いを持つもの」である．たとえば要件や仕様，実行コードのほかに，検証済みのテスト環境等も成果物と考えられる．成果物観点でのモデルでは，プロセスは成果物の集合とその要素間の利用関係として定義される．タスク観点によるモデルでは，タスク間の時間的順序関係が状況に応じて変化しうるのに対して，成果物観点のモデルでは，適用技術の変更や開発規模の大幅な変更による管理方法の変更等がない限り，成果物間の関係は開発のライフサイクルを通して変化しない．

Worldly レベルでのプロセスモデルには，成果物観点でのモデルが有効である[8),9)]．しかし，成果物観点でのモデルの提案は非常に少ない．Humphrey らは，Jackson System Development[16)] でのエンティティ指向の考え方をプロセスモデルに適用した「Entity Process Models」[8)]（以下「EPMs」）を提案している．EPMs は，開発ライフサイクル中で作成される各成果物の状態遷移の詳細なモデルであるが，1 つ 1 つの成果物の状態遷移の記述に力点が置かれており，各成果物間の関連や制約を記述すると非常に複雑なモデルとなる．そのため，プロジェクト全体の見通しが悪く，ライフサイクルを表現しにくく，標準プロセスの定義のためのプロセスモデルとしては冗長でかつ難解なモデルであるといえる．このように，我々の目的に合致した Worldly レベルでのプロセスモデルがなかったため，Worldly レベルを中心とした成果物観点でのモデルとして，新たに PReP モデルを開発した．次章では，PReP モデルの構成およびモデル化の方法，そしてその利用方法に関して述べる．

3. PReP モデル

PReP モデルは，Worldly レベルのプロセスモデルを中心とした成果物観点のモデルである．対象とする開発プロジェクトは，第 1 筆者の所属する組織で開発している組み込み系のソフトウェア開発を想定しており，プロジェクトの規模は約 100 名程度，開発期間は半年から約 1 年程度を想定している．

PReP モデルは，**図 1** に示すように，構造モデル，工程モデル，そして作業モデルの 3 つのモデルで構成される．Worldly レベルでの成果物間の関連構造を「構造モデル」で表現し，Universal レベルでの開発ライフサイクル情報を構造モデルに追加したものが「工程モデル」となる．そして，Atomic レベルの作業手順情報は「作業モデル」として各成果物に関連付けられる．

PReP モデルでは，Atomic レベルである作業モデルの記述に，ETVX や IDEF0 および一般的なフロー図，あるいは Daibutsu-Den モデル[4)] における XML 形式による記述等の，従来からのプロセスモデルの適用を考えている．以下，PReP モデルの特徴である成果物間の関連構造のモデル化を中心に，構造モデルと工程モデルの記述方法とその利用に関して述べる．

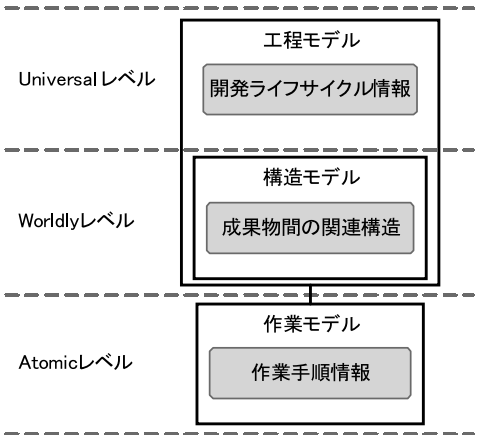


図 1 PReP モデルの構成とプロセスモデルの抽象度との関係
Fig. 1 Relationship between the structure of PReP model and the abstraction level of a process model.

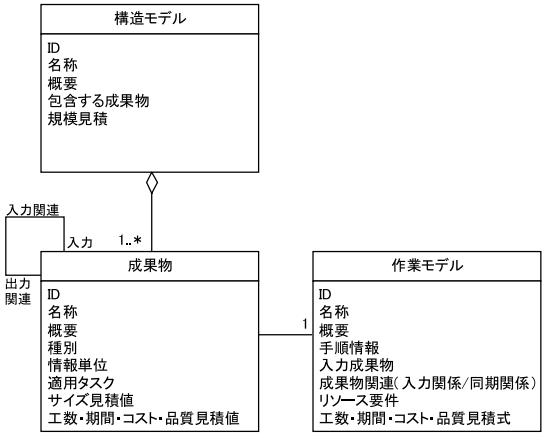


図 2 構造モデル
Fig. 2 Structure model.

3.1 構造モデルの記述

図 2 に示すように，構造モデルは，開発のライフサイクルで作成される成果物間の関連構造によってプロセスをモデル化する．成果物間の関連構造とは，開発プロセスへの最初の入力となる成果物から最終成果物が作られるまでの，個々の成果物間の入出力関係である．

成果物観点でのモデルは，開発プロジェクトで適用される技術の変更や開発規模の大きな変更等がない限

り，成果物間の関連が開発ライフサイクルを通して変動しないという特徴を持つ．そのため，定義したプロセスの再利用性が高く，標準プロセスの定義に適していると考えられる．一方，個々の成果物を作成するための手順として関連付けられる作業モデルには，プロジェクトごとに異なった方法を選択することが可能である．

図 2 に示すように，PReP モデルで定義される成果物は，成果物の属性として，成果物の規模の見積り値，工数・期間・コスト・品質の見積り値等の値を持つ．そして，成果物に関連付けられる作業モデルが，定義された手順に従って成果物を作成した場合の工数・期間・コスト・品質の見積り式を持つ．このことによって，選択した作業モデルによる成果物の見積り値の各値が決定される．構造モデル全体では，包含する作業成果物情報等のほかに，プロジェクトの特性を示す属性として開発規模の見積り値を持つ．

3.1.1 「成果物」の定義

成果物観点でのプロセスのモデルでは，開発プロセスから “適切な成果物” を定義することが重要である[8),9)]．PReP モデルでは，成果物を「各作業者に割り当てられた作業の成果としての実体であり，かつ振舞いを持つもの」として定義する．このことにより，プロセスモデルで定義する「成果物」の粒度が，プロジェクトが管理する「作業者」の粒度，すなわち「プロジェクトの管理粒度」に対応付けられることになる．たとえば，能力の高い作業者が複数の成果物からなるプロセスを引き受ける場合には，その作業者からの最終の成果物を管理することになる．また，通常の作業者が分担して作業を進める場合には，個々の作業者が受け持つ成果物を細かに管理することになる．なお，ここでの「作業者」は個人とは限らず，複数の作業者とその責任者から構成される場合もある．さらにプロジェクト外から成果物が入力される場合は，組織やプロジェクトが「作業者」となる．このようにプロセスモデルで扱う「成果物」の粒度を「プロジェクトの管理粒度」に対応させることは，実際に行われている開発活動の実体をモデル化する方法として適しているだけでなく，プロジェクトの管理粒度に合致した計画の作成や，定義したプロセスに従った開発活動の実行と進捗の監視と制御にも適していると考えられる．

また，実際の開発では，同種の成果物が何回かに分けて作成される場合がある．開発プロジェクトの規模が大きくなり開発期間が長くなる場合や，技術的リスクにより数回の検証が必要な場合は，たとえば最終成果物の第 1 版，第 2 版というように，同種の成果物が何回かに分けて作成される．このような場合，PReP モデルでは，作成するそれぞれの成果物を異なる成果物として定義する．

表 2 成果物の表記方法
Table 2 Notation for a product description.

表記	説明
通常成果物 成果物名称	1つの情報単位から構成される通常の成果物 成果物を示す図形と成果物の名称で構成
並列成果物 N 成果物名称	複数の独立した情報から構成される成果物 成果物を示す図形に複数個を示す「N」の表記を付加

3.1.2 成果物の表記

構造モデルの表記には，個々の成果物の表記と，成果物間の関係の表記がある．成果物の表記は，**表 2** に示すように，成果物の種別に対応して「通常成果物」と「並列成果物」の 2 通りの表記がある．それぞれの成果物には名称が付けられ，さらに，並列成果物には「N」の表記をする．

通常成果物とは，1 つの情報単位から構成される成果物である．一方，並列成果物とは，同種の独立した複数の情報から構成される成果物であり，たとえば機能単位，モジュール単位，ユースケース単位等，情報の単位ごとに独立して扱うことのできる情報の集合である．このような情報の単位は，成果物の種類（型）として，図 2 で示した成果物の属性の中の情報単位属性として定義される．並列成果物は，それぞれ個別の情報単位ごとに並行開発を行うことが可能であり，また，一部の独立した情報ができあがった時点で，それを入力とする成果物の開発に着手できる．

3.1.3 成果物の関連の表記

構造モデルの成果物間の関連は**表 3** に示すように「入力関係」,「同期関係」そして「入力と同期の合成関係」の 3 種類の関係がある．それぞれの定義は以下のとおりである．

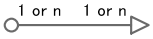
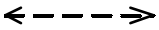
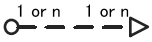
入力関係

始点にある成果物の情報に基づいて終点の成果物が作成される関係．終点にある成果物が始点にある成果物の情報を必要とする関係にあることを示し，表 3 に示すように実線の矢印で示す．複数の成果物からの線が煩雑になる場合は，関連する成果物を矩形によりまとめた表記も可能とする．

同期関係

始点にある成果物と，終点にある成果物とが相互に情報を必要とする関係．この関係にある成果物は，

表 3 成果物間の関連の表記方法
Table 3 Notation for describing relationships between each product.

表記	説明
入力関係 	始点にある成果物の情報に基づいて終点の成果物が作成されることを示す．つまり，終点にある成果物が，始点にある成果物の情報を必要とする関係にあることを示す．
同期関係 	始点にある成果物と，終点にある成果物とが相互に情報を必要とする関係にあることを示す．この関係にある成果物は，お互いに情報を交換しながら並行に作成される．
入力と同期の合成関係 	始点にある成果物の情報に基づいて終点の成果物が作成され，かつそれぞれが同期を取って作成されるため，始点にある成果物が完成しなくとも終点にある成果物が作成開始できるが，始点にある成果物が完成しないと終点にある成果物も完成しないことを示す．

お互いに情報を交換しながら並行に作成される．表 3 に示すように点線の双方向矢印で示す．

入力と同期の合成関係

始点にある成果物の情報に基づいて終点の成果物が作成され，かつそれぞれが同期をとって作成される関係．始点にある成果物が完成しなくても終点にある成果物の作成を開始できるが，始点にある成果物が完成しないと終点にある成果物も完成しないことを示す．表 3 に示すように点線の矢印で示す．

多重度の修飾子

両端が並列成果物の場合には，表 3 に示すように，関連の両端に「1」または「n」の修飾を付けることができる．この修飾子は，両端の各並列成果物の要素どうしの対応関係における多重度を表す．

以上の表記ルールによって，構造モデルでの成果物とその関連の記述を行う．次に，構造モデルを開発プロジェクトのリスクや制約を考慮して，ライフサイクルの戦略を検討するための工程モデルの定義方法と定義時のルールについて述べる．

3.2 工程モデル

工程モデルは，構造モデルで記述した成果物の関連に対して，開発のライフサイクルを定義するモデルである．ライフサイクルは，たとえば，要件開発，概要設計といった複数の工程の系列として構成される．各工程は，一連のプロセスまたは一連のプロセスの反復として定義される．プロジェクトのリスクや制約，前提条件等の違いによっては，同じ構造モデルから異なった工程モデルが得られることとなる．たとえば，要件定義におけるリスクの異なるプロジェクトでは，要件定義の工程内の反復回数が異なった工程モデルが考えられる．

3.2.1 工程モデルの表記要素

PReP モデルでは，構造モデルに対して，工程，マイルストーン，マイルストーン成果物，反復を決定することによりライフサイクルを定義する．工程モデルを構成するこれら概念を下記に示す．

工程

マイルストーン成果物に対して，リスク低減の観点から段階的に完成度や内容の確認をとりながら開発を進めていく際の区分け．

マイルストーン

次の工程の開始について利害関係者との間で合意を形成する場（会議体）．マイルストーンでは，主にマイルストーン成果物が次工程の開始条件を満足しているかが評価される．

マイルストーン成果物

開発プロジェクトにとって重要であり，進捗と内容の評価や承認等を行うべき成果物．PReP モデルが成果物に着目したプロセスモデルである特徴から，ライフサイクルを定義する際に構造モデル上で特定する成果物である．

反復

各工程の実施方法として，工程内の反復を定義．反復終了時の成果物（主にマイルストーン成果物）の完成度や反復の目的等が定義される．

たとえば，**図 3** において，工程 1 では 2 回の反復が定義されているが，この場合，工程 1 で定義される一連の作業が 2 回繰り返される．各反復の終了条件は，主にマイルストーン成果物の完成度によって定義される．

3.2.2 工程モデルの展開手順と規則

構造モデルでは，成果物間の入出力の関連が記述されているため，作成された成果物が入力成果物の入力となるようなループ構造は許されない．ループが生じる場合は，異なる粒度の成果物記述されている等，主に成果物の特定に問題があると考えられる．成果物を

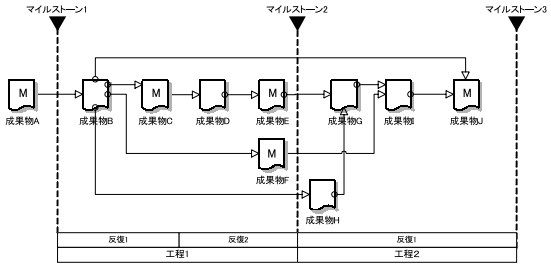


図 3 工程モデルへの展開例
Fig. 3 Example of a lifecycle model.

作成するうえでの繰返しは，反復として実現される．

工程モデルでは，工程の設定がライフサイクル決定の基本となる．構造モデルに対して，マイルストーン成果物を特定したうえで，成果物の開発段階の区切りを定義することで工程を設定する．各工程では反復が定義される．具体的な手順と規則は次のようになる．

手順 1：マイルストーン成果物の特定

構造モデルで定義された成果物から，マイルストーンとなる成果物を特定する．図 3 の例では,「M」の記号のついた成果物（A，C，E，F，J）がマイルストーン成果物として特定されたものである．このマイルストーン成果物の特定は，手順 2 に先立って行うが，並行して行う場合もある．

手順 2：工程とマイルストーンの設定

定義したマイルストーン成果物に対して工程を設定し，工程の終端をマイルストーンとして定義する．マイルストーンの位置は，以下の規則によって定める．

規則 2-a：工程に対してマイルストーン成果物が 1 つであれば，その直後がマイルストーンとなる（図 3 の例では成果物 A と成果物 J の直後）．

規則 2-b：工程に対してマイルストーン成果物が複数である場合は，最も下流に位置するマイルストーン成果物の直後がマイルストーンとなる（図 3 の例ではマイルストーン成果物 E と F の直後）．

また，マイルストーン成果物以外の成果物がどの工程に含まれるかは次の規則に従う．

規則 2-c：マイルストーン成果物の直接の入力成果物（図 3 の例での成果物 B，D，I）は，マイルストーン成果物と同一の工程に含まれる．

規則 2-d：マイルストーン成果物の直接の入力成果物でない成果物（図 3 の例での成果物 G，H）は，それらの出力となる成果物と同じ工程に含む．これは，当該成果物の進捗をその出力となる成果物の側で考慮する必要があるためである．図 3 の例では，G は I の直接の入力であるので I と同じ工程に含まれ，H は G の直接の入力であるので G と同じ工程に含まれる．

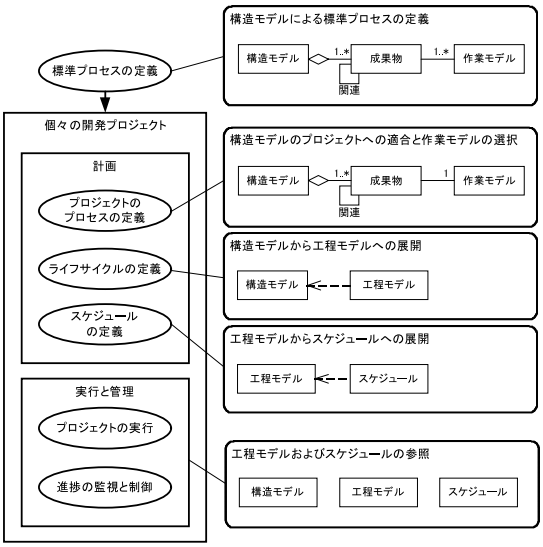


図 4 PReP モデルの利用
Fig. 4 Applying PReP model for project management.

手順 3：反復の設定

各工程の実施方法として，工程内の反復回数を設定する．図 3 の例では，工程 1 に 2 回の反復が定義されている．反復の回数は，プロジェクトごとに異なり，そのつど設定される．

3.2.3 プロジェクト管理・支援等のプロセス

PReP モデルは開発プロセスだけではなく，プロジェクト管理プロセスや，プロジェクト支援プロセス等にも適用が可能である．これらのプロセスの多くは，開発のライフサイクルには載らず，主に条件駆動型のプロセスとして構造モデルを使用して定義される．プロセスが駆動する条件としては，変更管理のプロセスのように，設定した条件を満たすイベントが発生した場合に駆動するプロセスと，進捗管理や確認会議のように，あらかじめ設定した日時で駆動するプロセスがある．

3.3 モデルの利用

標準プロセスの定義と個々のプロジェクトでの計画作成，およびプロジェクトの実行と管理への PReP モデルの利用を**図 4** に示す．以下，図 4 に沿って PReP モデルの主な利用方法に関して述べる．

3.3.1 標準プロセスの定義

標準プロセスを定義する際には，構造モデルを使用する．標準プロセスは，適用する技術の違いや，新規開発または派生開発といった開発の種別ごとに，いくつかのプロセスのパターンとして定義される．また，

表 4 成果物の関連のパターンによるスケジュール展開方法
Table 4 Scheduling by the relationships patterns of products.

関連のパターン	スケジュールへの展開パターン	ガントチャートでの表現
順次開発型 成果物A 成果物B	Aが完成されてからでないとBの作成を開始できない.	A B
分散順次開発型 N 1 1 N 成果物A 成果物B	AとBがそれぞれ複数の独立した情報から構成されており，独立したAの情報からBの要素が作成される．そのため，複数の作業者によって分散して開発を進めることが可能である.	N=3 で分散した場合 A1 B1 A2 B2 A3 B3
並行開発型 N 成果物A 成果物B	Aが独立した複数の情報から構成されているため，Aのうち独立した部分ができた段階でBの作成に取り掛かることができる.	A a B
並行開発型（終了基準あり） N 成果物A 成果物B	Aが完成しなくともBが作成開始できるが，Aが完成しないとBも完成しない.	A a B

プロジェクト全体のプロセスとして定義されるほかに，要件定義プロセス，検証プロセスのように，ある目的を持ったサブプロセスとして定義，利用されることも考えられる．

構造モデルでは，個々の成果物に作業モデルが関連付けられる．標準プロセスを定義する際，成果物を作成する方法が複数ある場合には，1つの成果物に対して複数の作業モデルが関連付けられる．この場合，個々のプロジェクトでは，複数の作業モデルの候補からプロジェクトに最も適した作業モデルを選択することになる．

3.3.2 プロジェクトのプロセスの定義

開発プロジェクトの計画を作成する場合，標準プロセスのプロジェクトへの適合を行いプロジェクト固有のプロセスを定義する．まず，標準プロセスからプロジェクトの特性や要件に最も合致した構造モデルを雛形として選択する．成果物に複数の作業モデルの選択肢がある場合は，プロジェクトに適用する作業モデルを選択する．また，プロジェクトの特性および要件に合わせて構造モデルの修正を行う．

PReP モデルでは，個々の成果物を作成するために必要な情報の入力となる成果物に過不足がないかを調べていくことによって，構造モデルの妥当性の確認を行うことができ，修正やプロジェクトへの適合を行った際に成果物の漏れや不整合がないか等の確認ができる．

3.3.3 ライフサイクルの定義

定義したプロジェクトのプロセスをもとに工程モデルを定める．工程モデルを定義することによって，スケジュールを決定するために必要な情報が決定される．作成した工程モデルは，プロジェクトの実行時に作業者によって参照され利用される．また，プロジェクトの実績は定義したプロジェクトのライフサイクルとともに管理され，プロジェクト終了後，プロジェクト実績の評価と標準プロセスの改善のために利用される．

3.3.4 スケジュールの定義

定義したプロジェクトの工程モデルに対して，個々の成果物の担当者を定め，カレンダ情報を付加することによって，スケジュールを決定することができる．工程モデルからスケジュールを定義する際に，成果物間の関連から，スケジュール展開時の入力成果物と出力側の成果物との作成開始タイミングを特定することが可能である．**表 4** に示すように，成果物の関連の種別に対応して「順次開発型」,「分散順次開発型」,「並行開発型」,「終了基準のある並行開発型」の 4 種類の展開方法が定義できる．

順次開発型

1つの情報単位から構成される通常成果物間の入出力関係では，入力成果物の完成を待ってから出力成果物の作成が開始される．スケジュールを決定する

際は，表 4 に示すように，成果物 A の作成が終了したのち，成果物 B の作成が開始される．

分散順次開発型

入力および出力成果物がともに並列成果物である場合は，入力成果物の 1 つの情報単位の作成が終わった段階で，出力成果物の作成を開始することができる．それぞれの情報単位ごとに複数の作業者を割り当てた分散作業が可能であり，スケジュールを決定する際には，作業者への分散を決定する必要がある．1 人の担当者に複数の情報単位を割り当てる場合は，その担当者において，割り当てられた情報に対する順次開発型となる．

並行開発型

入力成果物が並列成果物で，出力成果物が通常成果物の場合は，入力成果物の 1 情報単位が作成された時点で，出力成果物の作成を開始することができる．スケジュールを決定する際は，表 4 の並行開発型でのガントチャートの例に示すように，並行作業での作成開始時期「a」の定義をスケジュール調整の段階で行う必要がある．

終了基準のある並行開発型

入力成果物と出力の成果物が並行開発型でさらに同期関係にある場合は，出力成果物の完了が入力成果物の完了条件となる．スケジュールを決定する際は，「並行開発型」と同様に，表 4 のガントチャートの例に示した，並行作業での作成開始時期「a」の定義をスケジュール調整の段階で行う必要がある．

このように構造モデルによって定義されたプロジェクトのプロセス，工程モデルによって定義されたプロジェクトのライフサイクル，そしてライフサイクルから定義されるスケジュールに従って，プロジェクトの実行と進捗が管理される．

3.3.5 プロジェクトの実行と管理

図 4 に示すように，定義したプロジェクトのプロセス，ライフサイクル，スケジュールをもとにプロジェクトの実行と進捗が制御される．プロジェクト実行時には，スケジュールに加えて，割り当てられた作業のプロセス上の位置付けや目標を知るためにライフサイクルが参照される．さらに，担当作業の標準手順等の情報を得るためにプロジェクトのプロセスが参照される．プロジェクトの進捗は，スケジュールと比較され監視される．そして，計画に対して是正処置が必要になった場合，作業者の割当ての見直し，またはライフサイクルの見直しによってスケジュールの変更が検討される．

構造モデルによって定義されたプロジェクトのプロセス，工程モデルによって定義されたプロジェクトのライフサイクル，そしてライフサイクルから定義されるスケジュールは，プロジェクトの実行と進捗の監視において，それぞれ独立した情報として取り扱うことが可能である．たとえば，構造モデルは開発の適用技術や規模等の大きな変更がない限り変動はない．一方，工程モデルは，定義した構造モデルに対して，開発プロジェクトのリスクを低減するように変更される．そのため，開発ライフサイクルの途中でリスクが変動した場合や，計画の見直しが必要となった場合には必要に応じて変更される．この場合でも，構造モデルには変動がないため，工程，マイルストーン，および反復の計画の見直しのみで再計画を行うことができる．さらに，開発プロジェクトの作業者の状況が変化した場合や，開発期間の見直しの必要性が生じた場合，作業者の割当ての見直しや，分散開発，並行開発の見直しを行うことによって，同一の工程モデルに対してスケジュールの変更を行うことができる．

以上に述べたように，標準プロセスの定義と個々のプロジェクトでの計画作成およびプロジェクトの実行と管理に PReP モデルを適用することができる．次に，PReP モデルを，実際の開発プロジェクトへ適用した結果と，結果から得られたモデルの有効性に関して考察を行う．

4. 適用結果と有効性の考察

PReP モデルの有効性を，Humphrey の提案するプロセスモデルの要件と比較し考察を行う．評価の範囲を，プロセスモデルの記述が行われるプロジェクト計画の策定段階（プロジェクトのプロセスの定義，ライフサイクルの定義，スケジュールの定義）とした．見積りのための属性と見積り式は，実際のプロジェクトの実績値を集めたうえで定義されるものであるため，評価の範囲には含まなかった．

適用対象組織はソニー株式会社の LSI 開発組織である．LSI 設計にはハードウェアの設計と組み込みソフトウェアの設計がある．今日のハードウェア設計は，設計記述言語を用いて論理回路を設計する論理設計と，LSI チップ上の物理的な回路パターンを設計するレイアウト設計とに分かれる．論理設計工程はソフトウェア開発とプロセス上同等の性質を持つため，評価実験では，ソフトウェアプロセスとして記述し，評価に用いた．

4.1 適用プロジェクトの概要

適用プロジェクトは**表 5** に示すように組み込みソフトウェア設計が 13 事例，論理設計が 6 事例である．

表 5 各事例における作業内訳および再利用結果
Table 5 Time required for modeling processes and the results of reuse in each example.

開発分野	開発種別	事例番号	作業者数（延人）	作業時間（延 h）	元になったプロセス	再利用ノード数/全ノード数	再利用率（%）
組み込みソフトウェア開発	派生 LSI 開発（3 事例）	1	6	24.0	―	15（全ノード数）	―
		2	4	6.0	事例 1	15/16	94
		3	4	6.5	事例 1	15/16	94
	新規 LSI 開発（10 事例）	4	5	7.5	事例 1	15/20	75
		5	4	7.0	事例 4	19/20	95
		6	1	3.0	―	24	―
		7	1	0.5	事例 6	24	100
		8	1	1.0	―	11	―
		9	1	0.5	事例 8	11	100
		10	1	1.0	―	26	―
		11	1	0.5	事例 10	26	100
		12	1	1.0	―	24	―
		13	1	0.5	事例 13	24	100
LSI 論理設計	新規 LSI 開発（6 事例）	14	5	10.5	―	29（全ノード数）	―
		15	5	10.5	事例 14	29/29	100
		16	3	5.0	事例 14	17/17	100
		17	2	6.0	事例 14	29/42	69
		18	2	3.0	事例 14	29/36	81
		19	2	3.0	事例 14	29/36	81

組み込みソフトウェア設計には，既存開発資産の再利用が多くを占める派生 LSI 開発の 3 事例と，新規の LSI 開発の 10 事例がある．事例 1 から事例 5，および事例 14，16 は，プロセス記述評価のために試験的な適用を行った．さらに実際のプロジェクトへの適用を進めた結果が，事例 6〜13，および事例 17〜19 である．また，定義したプロセスの再利用の関係を表 5 の「元になったプロセス定義」に示した．たとえば，事例 2，3 は，事例 1 で作成したプロセスモデルを再利用している．

4.2 PReP モデルの適用結果

適用 8 事例のうち，事例 1 での PReP モデルの適用結果を示す．**図 5** は構造モデルによって定義したプロセス事例である．各成果物の名称は，企業内情報のため一般的な名称に変更してある．図 5 の例に示すように，4 種類の外部入力から，基本仕様関連の成果物，個別の機能仕様，そして検証関連の成果物と最終成果物を出力するまでのプロセスを，構造モデルを用いてモデル化することができた．

図 6 には，工程モデルを用いて開発ライフサイクルの定義を行った例を示す．工程モデルでは，マイルストーンの定義，工程の定義，工程内の反復の定義を行った．図 6 に示すように，マイルストーンとして，「基本仕様判定会議」等の会議体が定義された．これらの会議は実際に行われている会議体であり，そこではマイルストーン成果物として定義した成果物の確認と承認が行われており，実際に行われている活動を工程モデルのマイルストーンとして対応付けることができた．

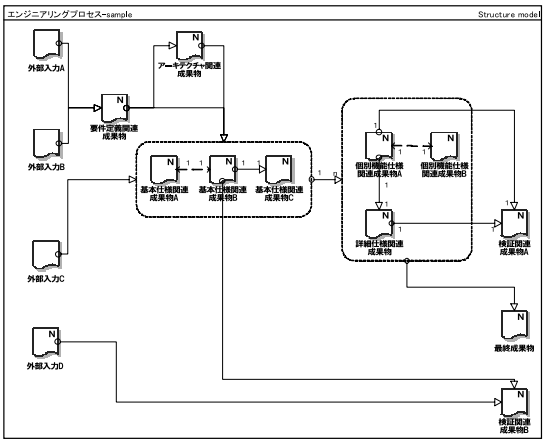


図 5 LSI 組み込みソフトウェア設計での構造モデルの記述例
Fig. 5 Structure model example for a firmware development process for LSI.

図 6 で示した工程モデルのうち，最初の工程である「基本仕様設計工程」のスケジュールへの出力例を **図 7** に示す．この例では Microsoft Visio で作成した工程モデルから，成果物の属性，成果物間の関連のデータを XML 形式で出力し，そのデータを Microsoft Project で読み込める形式に変換した．そして，Microsoft Project 上で作業者の割当て，カレンダ情報の付加，分散作業設定，並行作業での作成開始時期の定義等を行った．

上記で示したように，PReP モデルを使用して，プロセスの定義，ライフサイクルの定義を行い，ライフ

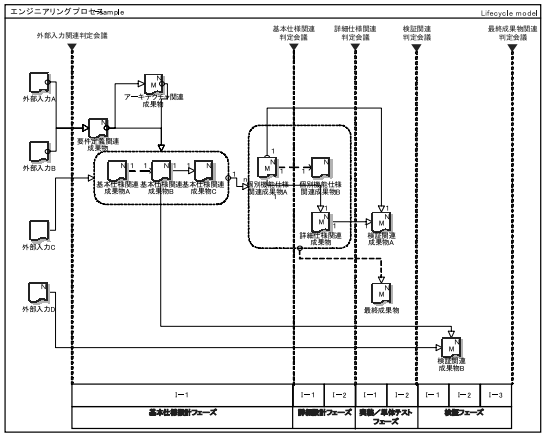


図 6　LSI 組み込みソフトウェア設計での工程モデルの例
Fig. 6　Lifecycle model example for a firmware development process for LSI.

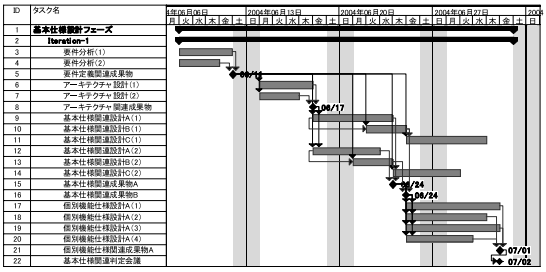


図 7　LSI 組み込みソフトウェア設計でのスケジュールへの出力例
Fig. 7　Schedule Example for a firmware development process for LSI.

サイクル情報からスケジュールの基本情報を出力し，スケジュールを定義することができた．事例 1 と同様にして適用を行った 19 事例の結果から，PReP モデルの有効性を次に考察する．

4.3　PReP モデルの有効性の考察

適用事例を通じて，PReP モデルを用いてプロジェクトのプロセスの定義，ライフサイクルの定義，スケジュールの出力が行えることを確認した．また，有効性の評価として，これらの事例における作業者数と作業時間，プロセスモデルのノード数（同等機能の部分の比較），再利用ノード数を調べた．さらに，従来のタスク観点でのモデルに対する PReP モデルの有意性を検証するために，PReP モデルの利用者 34 名を対象に，Humphrey によるプロセスモデルに対する 4 つの利用の観点に基づいた 5 段階評価の調査を行った．

4.3.1　現実の開発活動のモデル化

事例 1 では，以前に 2 名の担当者が開発プロジェクトと並行して，従来のタスク観点でのプロセスモデルの定義を試みていた．しかし，実際の開発活動と適切に対応付けされたプロセスモデルを定義することができず，開発プロジェクト全体の概観を示す抽象度の高い Waterfall model を定義するに終わっていた．それに対し，PReP モデルによるプロセスモデルでは，図 5 の適用結果で示したように，担当者に割り当てられた実際の成果物をもとにプロセスモデルを定義することができた．実際のモデル化の作業では，まず，各担当者に割り当てられた成果物を洗い出した．洗い出された成果物の粒度はプロジェクトの管理粒度に合わせて定義した．抽出した成果物間の関連を記述することによって構造モデルを作成することができた．定義したプロセスモデルは，現実の開発活動で実際に作成された成果物によってモデル化されており，Worldly レベルのプロセスモデルであるといえる．

また，適用事例の結果から，プロセス定義作業の延べ時間は 0.5 時間から 24 時間であった．従来モデルによる事例 1 のプロセス記述を試みた際，2 名の担当者が 3 カ月を要して記述を完了できなかった事実と比較しても，非常に短時間でモデル化が可能となったといえよう．特に，実際の適用事例 9，11，13 では，定義したプロセスの再利用により 0.5 時間程度で定義を行うことができた．

4.3.2　モデルの利用観点からの評価

Humphrey のプロセスモデルに対する 4 つの利用観点に基づいた設問による 5 段階評価を，従来のタスク観点でのモデルと PReP モデルそれぞれに対して行った．設問は，1）コミュニケーション：定義したプロセスが開発要員間でプロセスを理解し議論をする助けになるか，2）再利用：定義したプロセスが次回プロジェクトの計画や見積り時に再利用可能であるか，3）改善支援：定義したプロセスがプロセス改善の助けになるか，4）プロジェクト管理：定義したプロセスがプロジェクト計画の作成や，進捗管理に役立つか，の 4 問である．評価は PReP モデルの利用者 34 名により行われた．

提案方法と従来法との比較に同じ被験者による 5 段階評価を用いたため，評価結果に対して Wilcoxon の符号付順位和検定[17] によるノンパラメトリックな検定を行った結果，**表 6** に示すように 4 項目すべての利用観点において PReP モデルによる記述が従来のタスク観点での記述に比べて優れている（有意水準 5%）との評価を得た．

個別のコメントとしては，「プロジェクトの全体のプロセスを見通し良く検討することが可能になった」，「担当している成果物のプロセス上の位置づけが明確になった」，「定義したプロセスを再利用する意味があ

表 6 被験者 34 名による提案方法と従来方法との比較
Table 6 Comparisons between the proposed method and the conventional method by the 34 test cases.

評価基準		5 段階評価平均点	p 値
コミュニケーション	PReP	4.46	2.43e-005
	従来	3.18	
再利用	PReP	4.25	1.76e-006
	従来	2.21	
改善支援	PReP	3.90	2.31e-006
	従来	2.25	
プロジェクトプロセスの管理	PReP	4.18	5.32e-006
	従来	2.64	

る」,「プロセス改善や次回の計画に役立つので，成果物単位での規模と工数の計測を行いたい」等の報告を受けた．

4.3.3 モデル化の粒度

表 5 中，事例 16（LSI 論理設計への適用事例）は，能力の高い少人数の開発者による短期間の開発プロジェクトであった．事例 16 のプロセスモデルは，29 の成果物による事例 14 のプロセスモデルから中間成果物を省略したもので，17 の成果物による，より管理粒度の荒いプロセスである．このように PReP モデルは，プロジェクトの管理粒度に応じた粒度でプロセスをモデル化することができる方法であるといえる．

4.3.4 再利用性

表 5 に事例間の再利用の関係と再利用率を示した．同じ領域の開発プロジェクト間（事例 11 → 2・3，6 → 7，8 → 9，10 → 11，12 → 13，14 → 15・16）では，元となるプロセスモデルはそのまま利用されるか，もしくは成果物を追加して利用された．今回の適用事例では，追加された成果物は，ある成果物の検証用のプロトタイプであった．また，事例 4，5，17，18，19 では上流工程，テスト工程等のプロセスが追加され，共通するプロセス領域はそのまま再利用された．

以上の適用結果から，PReP モデルは現実の開発活動を適切にモデル化していること，プロセスモデルの理解が容易であること，管理粒度に合わせた粒度でのモデル化が可能であること，再利用性が高いことを確認することができた．2.1 節で議論したプロセスモデルの要件と比較して，PReP モデルはプロセスのモデル化の要件を満足する方法であると考えられる．

5. まとめと今後の課題

プロセスの改善を行うためには，標準プロセスを定義し利用することが必要である．しかし，これまで，定義したプロセスが効果的に利用されない問題をかかえていた．定義したプロセスが利用されない主要な原因が，プロセスのモデルにあると考え，PReP モデルを提案した．PReP モデルを実際の開発プロジェクトに適用した結果，Humphrey の提案するプロセスモデルの要件を満足することを確認した．

今後，PReP モデルをプロジェクト計画作成支援，プロジェクト実行時の作業の誘導と開発者間のコミュニケーション支援，そしてプロセス改善のためのプロセス実績の評価支援として，プロセス中心型開発環境[14),18)] へ適用することを計画している．また，PReP モデルで定義する見積り式とその属性を使った成果物の観点からの見積り方法や，プロセス改善に対する効果の検証は，現在，実際のプロジェクトの実績値を集めている段階であるため，今後の課題である．

謝辞 本研究および本論文の執筆にあたり，終始貴重なご指導を賜りました．また，奈良先端科学技術大学院大学の松本健一教授および飯田元助教授に深謝いたします．さらに，本研究の全般にわたり，貴重なご助言とご助力を賜りました，奈良先端科学技術大学院大学情報科学研究科産学連携研究員の大杉直樹氏に深謝いたします．

参 考 文 献

1) Paulk, M., Curtis, B., Chrissis, M. and Weber, C.: Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-024 (1993).
2) CMMI Product Team: CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1, CMU/SEI-2002-TR-011 /TR-012 (2002).
3) Iida, H.: Pattern-Oriented Approach to Software Process Evolution, *IWPSE'99*, pp.55–59 (1999).
4) Iida, H. and Tanaka, Y.: A Compositional Process Pattern Framework for Component-based Process Modeling Assistance, *Proc. 1st Workshop on Software Development Patterns* (*SDPP '02*), Technical Report TUM-I0213, Munich University of Technology, pp.57–64 (Dec. 2003).
5) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns — Elements of Reusable Object-Oriented Software*, Addison Wesley Professional Computing Series, Addison Wesley (1994). 本井田真一，吉田和樹（監訳）：デザインパターン，ソフトバンク (1995).

6) Alexander, C.: *The Timeless Way of Building*, Oxford University Press (1979). 平田翰那（訳）：時を超えた建設の道，鹿島出版会 (1994).
7) Iida, H., Tanaka, Y. and Matsumoto, K.: Daibutsu-den: A Component-based Framework for Organizational Process Asset Utilization, *Proc. Profes2002*, Lecture Notes in Computer Science 2559/2002, pp.207–219, Springer-Verlag, ISSN: 0302-9743 (2002).
8) Humphrey, W.S. and Kellner, M.I.: *Software Process Modeling: Principles of Entity Process Models*, CMU/SEI-89-TR-002 (1989).
9) Humphrey, W.S.: *Managing the Software Process*, Addison-Wesley, ISBN-0201180952 (1989). 藤野喜一（監訳）：ソフトウェアプロセス成熟度の改善，pp.268–274, 298–300, 日科技連 (1991).
10) Winston, W.R.: Managing the Development of Large Software Systems: Concepts and Techniques, *Proc. IEEE WESCON*, pp.1–9, IEEE (Aug. 1970).
11) Winston, W.R.: Managing the Development of Large Software System, *Proc. 9th International Conference on Software Engineering*, pp.328–338, IEEE (Aug. 1987).
12) Boehm, B.W.: A Spiral Model of Software Development and Enhancement, *ACM Software Engineering Notes*, Vol.11, No.4, pp.14–24 (Aug. 1986).
13) Basili, V.R. and Turner, A.J.: Iterative Enhancement: A Practical Technique for Software Development, *IEEE Trans.Softw.Eng.*, SE-1(4) (Dec. 1975).
14) 井上克郎，松本健一，飯田　元：ソフトウェアプロセス，p.27, 共立出版 (2000).
15) Radice, R.: A Programming process architecture, *IBM Systems Journal*, Vol.24, No.2 (1985).
16) Jackson, M.A.: System Development, Prentice-Hall International, Englewood Cliffs, NJ (1983).
17) 田中　豊，垂水共之（編）：Windows 版統計解析ハンドブックノンパラメトリック法，pp.60–63, 共立出版 (1999).
18) Kobialka, H.-U.: *Implementing Support for Software Processes in a Process-centered Software Engineering Environment*, GMD Research Series (1988).

(平成 16 年 9 月 2 日受付)
(平成 17 年 2 月 1 日採録)

田中　康（正会員）
昭和 57 年上智大学理工学部物理学科卒業．昭和 60 年ソニー株式会社入社．平成 14 年先端科学技術大学院大学情報科学研究科博士後期課程に社会人学生として入学．ヒューマンインタフェース研究を経て，ソフトウェアプロセス改善業務に携わり，ソフトウェアプロセスの研究に従事．ヒューマンインタフェース学会会員．

飯田　元（正会員）
昭和 63 年大阪大学基礎工学部情報工学科卒業．平成 3 年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．平成 7 年奈良先端科学技術大学院大学情報科学センター助教授．博士（工学）．ソフトウェアプロセスを中心としたソフトウェア工学の研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE，ACM 各会員．

松本　健一（正会員）
昭和 60 年大阪大学基礎工学部情報工学科卒業．平成元年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．平成 5 年奈良先端科学技術大学院大学助教授．平成 13 年同大学教授．工学博士．エンピリカルソフトウェア工学，特に，プロジェクトデータ収集/利用支援の研究に従事．電子情報通信学会，ACM 各会員，IEEE Senior Member.