チュートリアル

チュートリアルシリーズ『ソフトウェア工学の実証的アプローチ』第 3 回

ソフトウェアプロセスのモデリングとそれに基づく管理手法

伏田 享平　飯田 元

ソフトウェア開発プロセスのモデリング (ソフトウェア開発に関する諸作業の実施に関する約束事等を明示的に記述すること) は，プロセス実行中の定量的なプロジェクト管理や，CMMI 等に代表される定性的プロセス評価の枠組みにとって，とても重要である．本稿ではソフトウェアプロセスに関する最新の研究動向を踏まえつつ，ソフトウェアプロセスのモデリングと，プロジェクト管理やプロセス改善への応用について解説する．また，ライフサイクルプロセスやプロセスの評価・改善に関連する主要な標準規格や参照モデルについても紹介する．

Software process modeling (i.e. describing explicit rules/constraints for the execution of various activities in software development) is very important to form a basis of quantitative management, qualitative evaluation, and improvement of software process. This paper overviews research activities in software process modeling and its application to process management and improvement basing the latest research direction. We also briefly introduce standards and reference models related to process management and process improvement.

1 はじめに

ソフトウェアの品質維持や開発期間とコストの制御を目的として，定量的なプロジェクト管理手法が重要視されている．特に近年は，実施中の開発プロセスの状態を随時評価する「インプロセス分析」を効果的に行うことでプロジェクトの早期に問題を検出し対処することが求められている．一方，プロセス自体の改善を目的として，CMMI [7] 等に代表される定性的プロセス評価の枠組みが多数提案され，多くの組織で適用が進められている．これらの活動を系統だてて行ううえで，ソフトウェア開発プロセスのモデリング (ソフトウェア開発に関する諸作業の実施に関する約束事等を明示的に記述すること) は極めて重要な技術である．

ソフトウェアプロセスを主題とする国際会議 ICSSP[†1]では，アジャイルプロセスやビジネスプロセスなどのテーマが新たにとりあげられる一方で，ソフトウェアプロセスの「モデリング」「計測・分析」「管理」は，依然として主要テーマであり続けている．本稿では，これらのソフトウェアプロセスに関する最新の研究動向を踏まえつつ，特に重要と思われる以下の 2 つのトピックについて解説を試みる．

- ソフトウェアプロセスのモデリング：ソフトウェアプロセスを客観的に定義し表現するための手法およびツール．

An Introduction to Software Process Modeling and Management.

Kyohei Fushida and Hajimu Iida, 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.

コンピュータソフトウェア, Vol.29, No.1 (2012), pp.61–74.
[解説論文] 2011 年 7 月 4 日受付.

†1 ICSP (International Conference on Software Process) は 2006 年まで開催されていた SPW (Software Process Workshop) と ProSim (Workshop on Process Simulation and Modeling) が合併して 2007 年より開催された．2011 年からは，対象をソフトウェアからシステムのプロセスにまで広げて，ICSSP (International Conference on Software and Systems Process) として開催されることとなった．

- プロセスモデルの管理や改善への応用：管理データの収集・可視化やプロセスの分析・評価を行うための手法およびツール．

以降，2 節でソフトウェアプロセスとプロセスモデルについて定義した後，3 節でソフトウェアプロセスを客観的に定義し表現するための手法およびツールについて紹介する．4 節では，ソフトウェアプロセスの評価・分析手法や，それらを用いてプロジェクト管理を支援するツールについて紹介する．5 節では，ソフトウェアプロセス技術に関する現状と今後の展望について著者らなりの考えを述べ，最後に 6 節で本稿をまとめる．

2 ソフトウェアプロセスのモデル化

本節では，本稿で扱うプロセスモデリングに関する解説に先立ち，ソフトウェアプロセスとプロセスモデルについて概説し，いくつかの概念について定義する．

2.1 ソフトウェアプロセスとは

「**ソフトウェアプロセス**」という語がソフトウェア工学における重要なトピックとしてとりあげられるようになっておよそ 30 年であるが，周辺の語彙も含め，その詳細な定義についてのコンセンサスが得られているとは未だに言い難い．

主要な国際規格や知識体系での「プロセス」の定義を示すならば，たとえば，ISO/IEC 12207 (Software Life Cycle Process; ソフトウェアライフサイクルプロセス) 等では用語定義部分に「互いに関連を持った “活動 (アクティビティ)” の集合であり，入力を出力に変換するもの」と簡単に説明されている.「アクティビティ」は複数の「タスク」により構成されるという記述もあり，作業の粒度による用語の使い分けも行われている．また，SWEBOK (Software Engineering Body of Knowledge; ソフトウェアエンジニアリング基礎知識体系) [17] では，ソフトウェア・エンジニアリング・プロセスという語に対して,「ソフトウェア・エンジニアリングに関連した全てのプロセスに対する一般的性質に関する議論」「1 つの組織内で実行されているアクティビティの現実の集合体」といった様々な解釈があることが示されている．PMBOK (Project Management Body of Knowledge; プロジェクトマネジメント基礎知識体系) [36] では，プロセスとは「事前に定められた一連のプロダクト，所産，またはサービスを生み出すために実行される，相互に関連したアクションとアクティビティの組み合わせ」と定義されている．

本稿では，上記の内容や後述する SPEM のメタモデル定義も参考とした上で，ソフトウェアプロセスを「ソフトウェア開発に関連して実施されるあらゆる**活動**の集合」と定義する．その上で，個々の活動を構成する基本要素を，1) **作業**，2) 作業の入出力としての**成果物**，3) 作業の実施主体としての**役割**とし，これら基本要素の個々の状態や，要素間に存在するあらゆる関連 (例えば，作業間の階層や実行順序・系列，作業と成果物との入出力関係，作業と役割との対応付け等) やプロセスの実行に関わる制約条件もモデル化の対象として想定する．ただし，モデル化の目的・視点によって，プロセスの範囲や取り扱うべき要素が限定されるという立場をとる．

2.2 ソフトウェアプロセスモデルの種類

ソフトウェアプロセスに対して厳密な理解や議論を行うためには，プロセス自体を何らかの視点に基づいて形式化・抽象化し，一定の規約に従って記述すること (**オーサリング**と呼ばれる) が必要である．**ソフトウェアプロセスモデル** (以降プロセスモデル) という言葉は記述されたソフトウェアプロセスを指し，記述のための規約をソフトウェアプロセスの**メタモデル**と呼ぶ．たとえば，先に示した PMBOK の定義に従えば，PMBOK で利用されている WBS(Work Breakdown Structure) やそれをもとに作成される線表類は，プロセスを「階層的に構成された成果物」と「それらに関連付けられた作業の実行順序」に着目して記述した，一種のプロセスモデルであるといえる．

プロセスモデルは参照のためのモデル (**参照モデル**) と，現実のプロセスを表現するためのモデル (本稿では**実施モデル**と呼ぶ) の 2 種類に分類できる．以下では参照モデルと実施モデルについて概説する．

参照モデル

参照モデルはソフトウェアプロセスに関する一般的な概念やプロセスのあるべき姿を示すためのモデルである．

参照モデルの代表例として，前述した ISO/IEC 12207(以降 SLCP と呼ぶ) がある．SLCP はソフトウェアライフサイクルプロセスの参照モデルを定義した国際規格である．ソフトウェアライフサイクルとは，ソフトウェアの構想から作成，廃棄に至るまでの一連の作業を含んだプロセスを指す．SLCP は，ソフトウェア製品およびサービスのライフサイクル中で実施すべき作業を列挙することで，ソフトウェアプロセス共通理解のための枠組みを提供している．さらに，プロセスの管理および改善についてもこの枠組みで規定されている．SLCP を日本のソフトウェア産業事情に適応させた「共通フレーム」と呼ばれるドキュメントが IPA/SEC から出版されている [59].

ソフトウェアプロセスの評価，改善のためのモデルを定義した ISO/IEC 15504 では，開発組織のプロセス能力を評価するための様々な測定項目が定められているが，そのパート 5 では，SLCP に基づくプロセスのアセスメント例が記されている†2.

ISO/IEC 15504 自体に影響を与えた CMM，更にその後継である CMMI などは組織の持つソフトウェアプロセス実施能力や成熟度を評価するためのフレームワークである．これらの定義書中ではプロセスの分野や活動といった用語を用いてプロセスの概念整理を行っており，これもプロセス参照モデルの 1 つと位置づけられる．さらに，CMMI に基づいたプロセスの評価・改善のためのフレームワークである IDEAL [22] やプロセス評価手順規格 SCAMPI [40] などは，プロセス改善のために実行すべきメタプロセスの参照モデルと捉えることができる．

参照モデルは具体的な作業順序や手法ではなく，むしろ特定の手法によらない一般的で共通の概念を示すものであるため，モデル化の手法が議論の対象となることは少ない．

†2 現在，ISO/IEC 15504 規格 は，ISO/IEC 33000 シリーズとして再編作業が進められている．

実施モデル

実施モデルはプロセスを実行するうえでの指示書，あるいは，実行されたプロセスの説明書として用いるためのモデルであり，作業順序や特定の開発手法・管理手法に依存した内容などについての具体的な記述が要求される．また，プロジェクトごとの特性に応じたカスタマイズ (**テーラリング**と呼ばれる) が行われることもあるため，記述の容易性や可読性が重要である．以降では特に断りのない限り，主に実施プロセスのモデリングについて扱う．

2.3 参照モデルと実施モデルの関係

図 1 に前節で示した参照モデルと実施モデルの関係を整理して示す．

ある開発プロジェクトにおける実施モデル (プロジェクト標準プロセス) を作成することを考える．通常，開発組織では組織標準プロセス (これも実施モデルの一種と考えることができる) が用意されている．組織標準プロセスは，国際規格などの参照モデルをもとに，プロセスエンジニア (一般的には，プロセスエンジニアグループと呼ばれる，組織標準プロセスを管理する部署に所属する専門家) によって作成される．この作成作業のことをオーサリングと呼ぶ．

このようにして作成された組織標準プロセスを各プロジェクトで実施する場合，プロジェクトの予算や納期といった制約や目的にあわせた調整が必要である．このようにプロセスを特定の目的や制約にあわせて変更，適応させる作業のことを特にテーラリングと呼ぶ．

個々のプロジェクトでの開発は，基本的にはこのよ

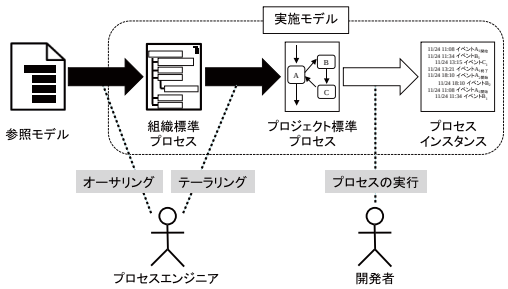

図 1 参照モデルと実施モデル

うにして作成・調整されたプロジェクト標準プロセスに従って行われる．このプロジェクト標準プロセスの実行結果を，**プロセスインスタンス**と呼ぶ[†3]．理想的には，プロジェクト標準プロセスとプロセスインスタンスは一致するはずである．だが現実には，プロジェクトの状況に応じてプロジェクト標準プロセスから逸脱した開発が行われることもある．記述の厳密さと実行時の柔軟さは相反する性質であり，このような逸脱を (現実的に) ある程度許容しつつも管理・制御する手法は，プロセスモデリングにおける重要な課題の 1 つである．

2.4 ソフトウェアプロセスの分析と評価

プロセスモデルを作成する大きな目的の 1 つは，プロセスを分析，管理し，より組織の状況に沿った形に改善することである．ここで**ソフトウェアプロセスの分析**とは，実施モデルをもとに開発プロセスの実施状況を明らかにすることである．分析にあたっては CMMI のように専門家がプロセスの実施状況をもとに定性的な分析を行う場合もあれば，プロセスモデルに基づき測定した値を用いて定量的に行う場合もある．また，**ソフトウェアプロセスの評価**とは，分析結果をもとに分析対象のプロセスモデルの品質や善し悪しについて，決定を下すことを指す．プロセス評価の結果によっては，さらにプロセス改善など次のアクションをとる．

評価の枠組みとしては，Basili らが提唱する GQM (Goal - Question - Metrics) アプローチがある [3]．GQM アプローチでは，プロジェクトの管理目的を定め，次に定めた管理目的の達成度を測るための定量的指標との関連付けを行い，実践する．また，GQM と類似の概念としては，McGarry らにより提案され，ISO/IEC 15939 として規格化されたソフトウェア計測プロセスである PSM (Practical Software Measurement) [23] がある．ISO/IEC 15939 は，ソフトウェアの測定プロセスに関する国際規格で，プロジェクト管理や品質保証など様々な情報ニーズを満たすために実施すべき測定や分析，解釈のためのフレームワークが示されている．ソフトウェアプロセスの定量的な評価にあたっては，このようなアプローチ[†4]を参照し分析を行う．

4 節ではソフトウェアプロセスの分析，評価方法に関して具体例を挙げて説明するとともに，分析を支援する手法について紹介する．

3 プロセスモデリング

本節ではプロセスモデリングのための具体的記法，モデルの記述 (オーサリングやテーラリング) の支援環境等について紹介する．

3.1 モデリング記法

過去の研究では，ソフトウェアプロセスを手続き的プログラムとして表現する試み [32] に始まり，成果物間の変換規則として捉えるもの [19]，プロセス代数に基づいて作業間の情報交換や同期に着目するもの [50] など，様々な記法 (パラダイム) が試されてきた．これらの試みは人間の知的活動プロセスを特定の処理モデルに基づいて表現するという意味においては非常に興味深く野心的なものであった．このようなモデリング記法同士の比較・検討も行われ [56]，以下の様な視点が主要な評価項目となっている．

記述の形式性の度合い 記述された内容が計算機によって解釈可能であり，一貫性や振る舞いの形式的な検証，部分的な自動実行などの処理が可能かどうか

記述の粒度 記述がプロセスのおおよその流れを表すのか，より具体的な個人作業まで記述するのか，また，成果物の状態や内容についてどこまで詳細に記述するのか

記述の柔軟性 試行錯誤や例外的な作業の実施，再計画による手順の変更などにより，本来の予定とは異なったプロセスの実施を記述できるか，あるいは記述からの逸脱をどのように許容するのか

†3 SWEBOK での解釈のように，「プロセスの一般的な性質に関する議論」と「一つの組織内で実行された現実の集合」とを明確に区別して議論する場合，後者がプロセスインスタンスに相当する．

†4 ここに挙げている国際規格やフレームワークも，ソフトウェアそのものやプロセスを評価するための参照モデルであることに留意されたい．

提案されたモデリング手法は協調作業支援や開発ツール間の動的な連携の制御，あるいは，事前計画のシミュレーション†5などへの応用が試みられた．近年は組織がプロセスを実施する際の手引書としてのプロセスモデルの役割が重要視されるようになり，SPEM のように実行や分析よりも記述の粒度や柔軟性を重視したプロセスモデルが考案されている．

本稿では伝統的記法としての WBS，線表 (ガントチャート) と，専用のプロセスモデリング記法である SPEM を紹介する．

WBS

WBS(Work Breakdown Structure) は PMBOK などでも言及されており，プロジェクト管理手法のためのツールとして記述・利用されているが，階層構造を持つソフトウェアプロセスの記述にも有用であり，開発プロジェクトのために記述された WBS はそれ自体が一つのプロセスモデルとみなすことができる．WBS では，まずプロジェクトの成果物の構成を分割し，最終的に末端成果物を作成するための一連の作業群 (ワークパッケージと呼ばれる) にまで分解することで，プロセス全体の階層構造を得る．各ワークパッケージにはそれを実施する人員 (リソース) が割り当てられる．

WBS 自体を規格化したものとしては米軍規格 MIL-STD-881(現 MIL-HDBK-881) があるが，一般にはより抽象的な概念ツールとして利用されており，細部の記法については様々なバリエーションが存在する．WBS により分解された作業群の実施順序や細かいスケジューリングは次に紹介するガントチャートなどを用いてより具体的に記述されることが多い．図 2 は著者らが開発した WBS オーサリングツール [49] による画面表示例である．この例では，ツールのサンプルデータとして添付されている組み込みソフトウェア向け開発プロセスの参照モデル [58] での WBS の全体構造や，タスク「システムテスト」についての入出力成果物や開始条件等の詳細説明が表示されている．

†5 プロセスシミュレーションに関する研究動向については，Zhang らの報告が詳しい [46].

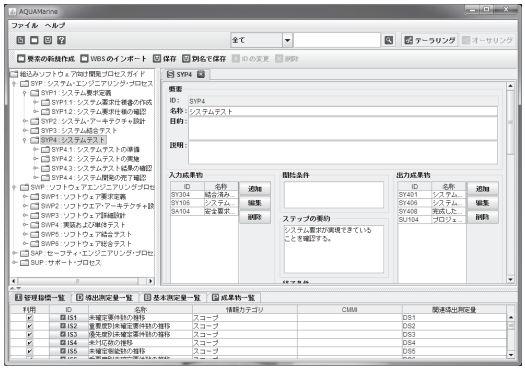

図 2 オーサリングツール上で表示させた WBS

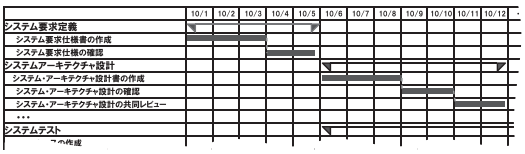

図 3 ガントチャートによるプロセス記述例

ガントチャート

ガントチャートもプロジェクト管理のためのツールとして伝統的に用いられてきた記法の 1 つであり，多くのバリエーションを持つ．図 3 はガントチャートの記述例である．ガントチャートでは個々の作業 (タスク) の実施時期を横向きの時間軸に沿った棒グラフの形で記述する．複数のタスクを縦軸方向に並べて記述することで，タスク間の順序関係や多重度を視覚的に確認することができる．ガントチャート自体は他のプロセスモデルをもとにした，より具体的な実施モデルの記述法と位置づけることができる．WBS のワークパッケージに対して，タスクの規模 (実施に必要な時間) やタスクを実行する人員 (リソース)，タスク間の順序的依存関係などの制約を指定することで，時間軸へのタスクの配置をある程度自動的に行うことができる (伝統的手法としてはクリティカルパス法や PERT 法が有名である)．Microsoft Project 等のプロジェクト管理ツールの多くは，WBS に対して追加の制約条件を指定することでガントチャートを自動生成する機能を持つ．タスクの数が膨大になるような大規模なプロジェクトプロセス記述のために，タスクのグループ

化・階層化などの記法を追加したものもある．

SPEM

SPEM (Software and Systems Process Engineering Metamodel) [30] はソフトウェアプロセスの多様な局面を表現するための記法を定めるものとして OMG(Open Management Group) によってまとめられているメタモデルである．ソフトウェアモデル化言語 UML(Unified Modeling Language) と同様に，記述モデルを定義するためのメタモデルを記述する枠組みである MOF(Meta Object Facility) を用いて定義されている．

SPEM ではプロセスモデルの核を作業，成果物，役割という 3 つの基本構成要素と捉え，その上に，様々なプロセスの形態を表現可能とするための拡張の仕組みを提供している．また，フェーズ，反復，といった，いくつかの典型的な概念要素についてはあらかじめ定義が行われ，アイコン画像も用意されている．

SPEM version 2.0 では様々な開発方法論について記述するための要素とそれを個々のプロジェクトで実施するための具体的なプロセスを記述するための要素とを明確に区別して定義している．方法論に関わるものとして，Work Product Definition，Role Definition，Task Definition 等の概念要素を持ち，プロセスに関わる概念要素としては Product Use，Role Use，Task Use などが定義されている．また，個々のプロセス (Process) は複数の Activity の集合により定義される．

SPEM を用いることで作業の順序や成果物間依存関係からプロセスにおける組織構造に至るまで幅広くソフトウェアプロセスを記述することができる．SPEM は「あるがままのプロセス」や,「あるべきプロセス」を説明的に記述するうえで豊富な記法を持つ一方で，記述に対する形式的な検証や記述に基づくプロセスの実行支援などは直接扱わない†6．

本稿では SPEM で定義されるメタモデルの詳細については省略し，図 4 に SPEM2.0 のアイコン表記を用いて記述したソフトウェアプロセスの例を示すにとどめる．ここでは UML2 クラス図に準ずる記法で要求分析作業とそれに関連する成果物とロールを明示し，さらに UML2 アクティビティ図に準ずる記法で作業の具体的な実施手順を明示したものを示している．ユースケース分析作業に関して，分析作業の主務者と補助者，必須および任意の入出力文書などの関連付けにより，WBS のワークパッケージに相当する情報が記述されている．UseCaseAnalysis ボックス内のアクティビティ図はより細かい作業 (step) への分解とその実行順序を示している．この他，WBS の構造に従って作業間の関連を示した図や，成果物間あるいは組織の階層構造を示すための記法など，UML に準じた記法が複数用意されている．

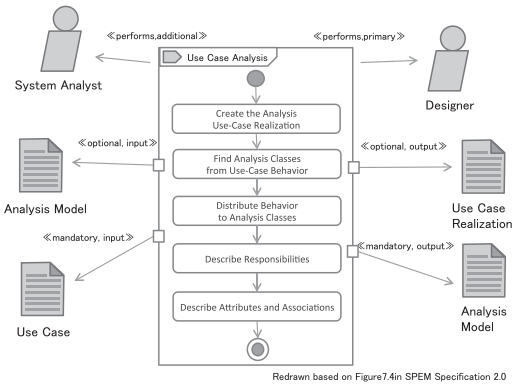

図 4　SPEM によるプロセス記述例

3.2　プロセスの記述と閲覧のための支援技術

前節で紹介したようなモデリング記法に従い，プロセスの記述や記述されたプロセスの閲覧を支援する環境がこれまでにいくつか提案されている．これらの環境は，目的にあったプロセスモデルを効率よく作成することと，記述されたプロセスを容易に参照し理解を深めて実践することを主な目的としている．

3.2.1　プロセス記述支援環境

ここではプロセスのオーサリングのためのツール例として，EPFC と Spearmint を紹介する．なお，両者の詳細な比較に関しては，Phongpaibul らの報告 [35] を別途参照されたい．

†6 SPEM のアクティビティ図による記述を BPMN (Business Process Modeling Notation) 等のワークフロー記述言語とのマッピングを行うことでワークフローエンジンによる実行を想定すると規格に記されているものの，その具体的実現は与えられていない．

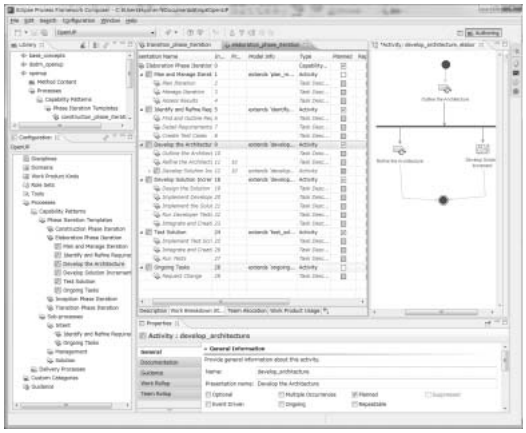

図 5 EPF Composer

Eclipse Foundation は，プロセスのオーサリングとテーラリングを支援する環境である EPF Composer(Eclipse Process Framework Composer) を開発している [43]．図 5 は EPF Composer のスクリーンショットである．図 5 ではあるプロセスのオーサリング画面を表示しており，プロセスの WBS や，SPEM で記述されたアクティビティが表示されている．

EPF Composer は SPEM2.0 に準拠しており，プロセスの目標や作成する成果物，ロールなど，プロセスを実施するのに必要な情報に関する記述を，個々の作業内容や開発ライフサイクルなどプロセスのコンテンツを表す情報と分離して扱うことができる．このため，既存プロセスの構成要素を再利用して新たなプロセスを構成したり，既存プロセスをプロジェクトにあわせてテーラリングしたりする際も，基底となる要素に影響を与えることなく特定のプロセスを修正することが容易である．

Spearmint は EPF Composer よりも早く Fraunhofer IESE において開発されたプロセスモデルのオーサリングツールである [4]．Speamint ではプロセスの記述には UML に類似した独自の図式言語を採用しており，4 つのビュー (成果物と作業・役割の関連を表したプロダクトフロービュー，成果物と作業の持つ階層構造をツリー形式で表示した構成ビュー，各要素の持つ属性が一覧表示する属性ビュー，プロセスを詳細なテキストで表したテキストビュー) に基づいたプロセスモデルの表示・閲覧が可能である点で特徴的である．

3.2.2 プロセスのテーラリング

テーラリングはオーサリングと類似の作業であるが，既存のプロセス資産を活用してプロジェクトごとに効率よく実施する必要がある．そのためプロジェクトの状況を入力としてテーラリングを支援する手法が研究されている．

Basili らは，プロジェクトの目的・制約にあわせてプロセスをテーラリングし，プロセスの改善を実現する手法と，それらに基づいたテーラリングツール環境 TAME を提案した [2]．TAME は計測ツールを統合・活用することで開発プロセスのテーラリングおよび改善活動を支援することができる．Park らは，テーラリング時にプロジェクトで必要なプロセスを，組織標準プロセスの中からニューラルネットワークを用いて自動的に絞り込むことでテーラリング作業を容易にする手法を提案している [34]．Huo らは，実際のプロジェクトのデータからプロセスパターンを抽出する手法を提案している．Huo らの手法を用いて抽出したパターンを用いることで，テーラリング時に実際にプロジェクトで採用するプロセスの選定がより容易に行える．

3.2.3 プロセスガイド

プロセスガイドとはある特定のプロセスに関してそのプロセスを実行するのに必要な参照情報を記述したものである [20]．プロセスガイドには少なくとも対象となるプロセスモデルが含まれている．これに加えて，このプロセスモデルを閲覧検索するためのツールや，プロセス実施のための参考情報が含まれる場合もある．

プロセスガイド自体は電子化されたものである必要はなく，印刷物の形態で提供される場合もある．しかし，検索などの利便性の観点から電子媒体で提供される方が望ましい．このように電子媒体で提供されるプロセスガイドを特に電子プロセスガイド (Electronic Process Guide, EPG) と呼ぶ．EPF Composer の閲覧モードや EPF Composer で外部保存されたプロセスモデルは典型的な電子プロセスガイドである．Spearmint も簡易 Web サーバ機能を持ち，作成した

プロセスモデルを電子プロセスガイドとして組織内で共有することができる．

特定のツールではない一般的なウェブブラウザで閲覧可能な電子プロセスガイドも数多く作成されている．たとえば，Boehm らは MBASE[6] という包括的なプロセスに関する解説を行う電子プロセスガイド[44] を公開している．また，Jeffery らは，EPG/ER[21] というプロセスに関する指針と知見をまとめた電子プロセスガイドを提案しており，中小企業のプロセス改善やプロセス革新での導入実績が報告されている．電子プロセスガイドの利用実態調査としては，中規模なソフトウェア開発組織における電子プロセスガイドの利用形態や導入にあたっての組織的要因の影響を Moe らが報告したものがある[26]．なお，その他のツールも含めた詳細な紹介は García らの記事[11] も参照されたい．

4 プロセスモデルに基づいた管理の実践

プロセスモデルを作成する大きな目的の 1 つは，プロセスを分析，管理し，より組織の状況に沿った形に改善することである．プロジェクト開始前に策定した標準プロセスと現在の実施状況 (プロセスインスタンス) を比較することで，プロセスが順調に進行しているか，作業に不整合や無駄がないかを確認でき，さらにモデルの改善につなげていくこともできる．近年では開発プロセスに関わる作業記録を自動的に収集する研究が数多く進められており，このような作業記録をもとにしたプロセス分析手法も提案されている．

本節では，4.1 でデータ収集や可視化，ソフトウェアプロセスの分析・管理を支援するツールについて紹介し，4.2 でリポジトリマイニング手法を中心にプロセスの分析・評価に関する研究を紹介する．

4.1 プロセス分析・管理を支援するツール

4.1.1 プロセスデータ収集ツール

ソフトウェア開発に関わる作業の記録を自動収集するシステムは数多く存在し，それらの多くは版管理システムやバグ管理システムなどといったソフトウェア開発支援ツール群と連携して動作する．ここでは，このような作業の記録やそれを利用した定量データの収集に関する取り組みを紹介する．

EASE プロジェクト†7で開発された EPM(Empirical Project Monitor)[55] は，版管理システム CVS†8，障害管理システム GNATS†9など，広く普及している開発支援ツール群が収集したデータを集計，分析し，グラフなどの形で可視化して利用者に提示するシステムである．EPM は既存開発支援ツール群と連携することで，開発者に作業負荷をかけることなく，インプロセスでのプロジェクトの状況を分析，可視化することができる．

StagE プロジェクト†10では，開発時に得られる種々のデータを「ソフトウェアタグ」という形式で集約しユーザに提供することで，ソフトウェアの品質検証や問題発生時の対応の迅速化に取り組んでいる．ソフトウェアタグ運用支援技術として，定量的管理計画作成支援ツール AQUAMarine[49] をベースにした収集計画策定支援ツールや，プロジェクトのタグデータ収集を支援する CollectTag[45] などを開発している．

Johnson らのグループは，開発者の行動に関するデータを収集する環境 Hackystat を開発，公開している[24]．Hackystat は，開発者の利用するツールに「センサ」と呼ばれるデータ収集のためのプラグインを導入することで計測を行う．計測対象となるツールごとにセンサを開発することで，様々な開発環境から粒度の細かい作業のデータを収集できる．

SoftPit プロジェクト†11では，Münch らが提唱する Software Project Control Center という概念[29] に基づき，開発に関わる複数のステークホルダーが持つ種々の目的に応じた定量的プロジェクト管理を支援する環境の提供を行っている．SoftPit プロジェクトでは Specula というツールを開発している．Specula はプロジェクトで収集されたデータを一箇所に集約し，多種多様な分析のためのビューを提供している．

4.1.2 ソフトウェアプロセスの可視化

プロセス中で計測されたデータをプロセスの構造な

†7 http://www.empirical.jp/
†8 http://www.nongnu.org/cvs/
†9 http://www.gnu.org/software/gnats/
†10 http://www.stage-project.jp/
†11 http://www.soft-pit.de/

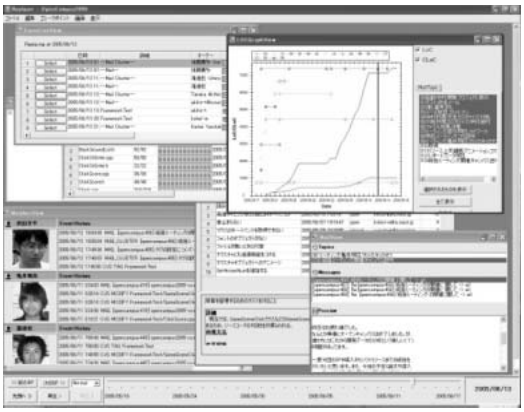

図 6 ProjectReplayer

ど，一定の基準に沿って整理し，可視化することで，ソフトウェアプロセスの分析者に対して有効な情報を提示し，分析を円滑に支援することができる [14].

前節で示したデータ収集ツールを利用した開発プロセスの可視化分析環境としては，ProjectReplayer [31] が挙げられる．図 6 に ProjectReplayer のスクリーンショットを示す．ProjectReplayer は，開発記録から編集作業の内容や E メールの情報を時系列に沿って表示することで，プロジェクトの振り返りを直観的に支援する．また Sarma らが開発した Tesseract [39] は，モジュール間や開発者間の関連と，プロジェクト実施中にやりとりされたメールやソースコードの修正に関わる情報を可視化する．CodeSaw [12] も同様に，開発者間のやりとりとソースコードの修正状況を時系列に可視化することで，分散開発プロジェクトの分析を支援する．

これらの可視化ツールは，収集ツールに記録された一次情報を可視化することで，開発プロセスの定性的な評価を支援するが，特定のプロセスモデルに従った可視化は行っていない．組織においてプロセスを確立し，継続的に改善していくためには，プロセスモデルに従った分析が必要となる．そこで，次節では実データを用いてプロセスモデルに基づきプロセス分析・評価を行う手法について紹介する．

4.2 実データを用いたプロセス分析・評価手法

開発ツールなどが自動収集したデータをプロセス分析に活用する手順はおおむね以下の通りとなる．

1. **プロセスインスタンスの抽出** データ群をマージして時系列順に並べたものとプロジェクト標準プロセスを照合し，標準プロセスの持つ作業構造に合致する作業実行系列 (プロセスインスタンス) 群を抽出する．
2. **プロセスメトリクスの計測** 各プロセスインスタンスについて，目的に応じて定義された定量的な値 (プロセスメトリクス) を計測する．
3. **プロセスインスタンスの評価** 計測したプロセスメトリクスを用いてプロセスインスタンスを評価し，何らかの判断を行う．

このように，プロセス分析に関する研究は大きく「プロセスインスタンスの抽出」と「プロセスメトリクスの計測，評価」に関する研究の 2 つに大別される．以降ではそれぞれのテーマについて研究動向を述べる．

4.2.1 プロセスインスタンスの抽出

Cook らは実際の開発記録からソフトウェアプロセスがどのように実行されたか，その実施モデルを抽出する手法を提案している [9] [8]．Cook らの手法では，ソフトウェアプロセスをペトリネットを拡張した形で定義し，実際の作業記録から確率論的な手法を用いて実施モデルを抽出する．Huo らの手法 [15] も Cook らの手法と同様の手順にプロセスインスタンスを抽出するが，より粒度の大きい単位で抽出が可能となる．Śliwerski らはソースコード管理システムとバグ管理システムの情報を利用し，ソースコード中にバグが混入した時期を特定する手法 (SZZ アルゴリズム) を提案している [42]．Zimmermann らはこの手法を用い，Eclipse の各プロダクトに含まれる，リリース前後のバグ数とプロダクトメトリクス†12との相関分析を行っている [47].

Morisaki らのマイクロプロセス分析手法では，プロセスインスタンスに加えて解釈のテンプレートとなるプロジェクト標準プロセスを与え比較することで，そのプロセスを特徴付ける値を抽出し，プロセス評価を行う [54] [27]． 図 7 にマイクロプロセス分析の概念図を示す．図 7 は単純な階層関係を持つプロセスモ

†12 ソフトウェアや関連ドキュメントなど，開発成果物を計測することで得られるメトリクス．

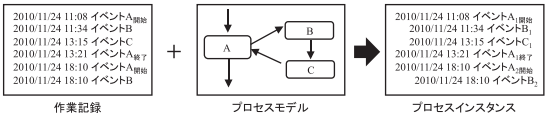

図 7　マイクロプロセス分析の概念図

デルを用いて作業記録の解釈を行う例を示している．作業 A は作業 B および作業 C の逐次実行に分解されている．この場合，各イベントについて,「作業 A の開始」「作業 B の実行」「作業 C の実行」「作業 A の終了」の 4 種類に対応付けた解釈を行っている．

作業報告などのドキュメントに基づいた分析では，実施されたプロセスを全て復元することは難しいが，自動収集された作業データがあれば自動的にプロセスインスタンス群を抽出することができる．このため，特にプロジェクトレベルで容易にプロセス分析に必要なプロセスインスタンスを抽出することが可能となる．

4.2.2　プロセスメトリクスの計測と活用

プロセスメトリクスとはプロセスインスタンスを測定対象としたメトリクスである．個々のプロセスインスタンスを測定して得られるプロセスメトリクスとしては，各種作業の進捗率や実施回数 (ソースコードの追加・削除行数，会議対の実施回数など)，作業の経過時間などが挙げられる [13] [25] [33]．一方で，プロセスインスタンスとプロジェクト標準プロセスを比較することで測定できるプロセスメトリクスもある．このように測定されたプロセスメトリクスとしては，作業の遵守度 [54] や作業の手戻り回数 [41]，開発者メトリクス [53] などが挙げられる．

プロセスメトリクスは，抽出したプロセスインスタンスを評価するのに用いられる．Ihara らはバグ管理システムにおけるバグの初期対応時間と滞留時間の関係について分析をしている [18]．あらかじめ規定された作業手順を遵守しているかによって，バグの修正作業に影響が出るか分析した研究もある [48]．

プロセスメトリクスはプロダクトメトリクスとともに，プロダクトの品質予測モデルを構築する際に用いられることもある．Moser らは複雑度メトリクスなどのプロダクトメトリクスと版管理システムの履歴を分析することで抽出できるメトリクス (コードの追加・削除行数など) を併用してバグ予測モデルを構築することで，プロダクトメトリクスのみで予測モデルを構築するよりもより良い精度が出ると報告している [28]．国内でも同様の分析が，ある企業の開発データを対象に纐纈らによって行われている [52]．

5　ソフトウェアプロセス技術の現況と今後の展望

本節では実際の現場におけるソフトウェアプロセス技術の利用状況と今後の研究の展望について，近年の研究動向も踏まえたうえで，著者らなりの考えを述べる．

5.1　構成管理ツールを利用したインプロセス分析

ソースコードを主な対象にファイルベースの版管理を自動的に行うツール (版管理システム) や要件管理システムのような，開発成果物を管理するための構成管理ツールは多くの現場で導入されてきている．特に版管理システムは構成管理における標準ツールといえ，このようなツールと連動して成果物の計測を行う環境も整備されている．また，個々の構成管理ツール間での連携を実現するため，OSLC(Open Service for Lifecycle Collaboration)[†13]という規格の策定が進められている．このような状況から，4.1 で示したようなインプロセス分析を行うためのツールを導入するための下地ができつつあるといえる．

構成管理ツールの利用増大傾向は，4.2 で紹介したようなプロセス分析手法を適用するための環境が整ってきているともいえる．特にソフトウェア開発の迅速性が求められている昨今，構成管理ツールに記録された膨大なデータを全て手作業で記録，分析することは現実的ではない．そのため，構成管理ツールが記録収集したデータの活用が期待できる．さらに，各ツールの記録を直接可視化して分析するだけではなく，複数のデータソースをプロセス全体の視点から関連付けて分析する「リポジトリマイニング」と呼ばれる分析手法も注目を集めている [51]．特にソフトウェアプロ

†13　http://open-services.net/

セスの分析に特化したリポジトリマイニングに関しては，Rubin [38] らが分析のための枠組みとツールを提案している．

このようにプロセス分析を行うための環境は十分に整いつつある．リポジトリマイニング技術を主体に，企業における分析の取り組みも報告されている．しかしながら，現状ではこのような分析に関する研究報告の多くは，オープンソースソフトウェア開発プロジェクトを対象にしたものであり，企業における取り組みが十分に浸透しているとは言い難い．また，その多くはソースコードの追加・削除行数など下流工程における開発プロセスを計測して得られたデータを利用したものが多い．これは，設計書作成など上流工程における作業を自動計測するための仕組み作りや，評価に有効な分析手法など，有効な管理を行うための環境が十分に整っていないためと考えられる．今後は開発プロセスを包括的に，また自動的に評価するための技術の発展が期待される．

5.2 開発形態の変化

ソフトウェアの開発形態は近年大きく変化している．従来のウォーターフォール型の開発に加えて，短期にかつ柔軟に開発するソフトウェアへの要求を取り込むためアジャイル開発手法を取り入れている開発組織が増えてきている．社団法人 日本情報システム・ユーザー協会 (JUAS) が行った調査では，一部の企業は今後アジャイル開発が増えていくことを予想している [60]．またオフショアに代表される分散開発が盛んになってきており，開発業務が地理的にも分散した複数の組織にまたがって行われることはもはや珍しくもない．

このような新しい開発形態を対象にした研究は，近年学術会議でも取り上げられている．たとえば分散開発を対象とした研究としては，Bird らによる地理的な位置関係と開発成果物の品質との関係に着目した研究がある [5]．またグローバルソフトウェア開発を専門に扱う国際会議 (ICGSE; International Conference on Global Software Engineering) も開催されている．ソフトウェアプロセスに関しても，今後はこのような文化的なギャップや地理的な関係を考慮に入れた開発プロセスのモデリングや評価が重要となってくると考える．

5.3 プロセスモデルの再利用

プロセスモデルを記述することで得られる恩恵の 1 つは，過去に利用され確立されてきたプロセスの再利用が可能となることである．開発組織において整備されている組織標準のプロセスなどはプロセス再利用の一例といえる．このようなプロセスの再利用に関してはモデリング記法でも考慮されており，SPEM でも「プロセスパッケージ」という概念が定義されている．さらに，ソフトウェア設計におけるデザインパターン [10] と同様に，ソフトウェアプロセスにおいてもそのベストプラクティスの集合である「プロセスパターン」の概念が提唱されている [1]．文献 [1] では，プロセスパターンとは「オブジェクト指向開発における技術や作業の集合」と定義されており，プログラミング工程でのパターンなどが定義されている．

プロセスモデルの再利用は，ソフトウェアプロダクトライン開発 [57] 手法の応用と考えることもできる．プロダクトライン開発とは，再利用のためのソフトウェア資産を整備しておき，それを利用して個々のソフトウェアを生産するアプローチである．このように既存のソフトウェア資産を利用して個々の要求に沿ったソフトウェアを生産することをバリエーション展開と呼ぶ．Rombach らは，プロダクトライン開発の手法に沿って開発プロセス自体をバリエーション展開する「プロセスライン」という考え方を提示している [37]．このようなバリエーション展開は，プロセスのテーラリングと類似の概念と捉えることができる．たとえば，プロセスラインの考えに基づきプロセスモデルをモデル駆動エンジニアリングのアプローチを応用してテーラリングする手法などが提案されている [16]．

このようにソフトウェアプロセスモデルの再利用に関する取り組みは古くから報告されているが，それが実務の現場で十分に活用されているとは言い難い．この原因の 1 つとして，再利用のための具体的な手順や指標が不明確なことが挙げられるであろう．たとえば，再利用可能なプロセスの部分集合をどのように評価すべきか，再利用に適した範囲はどの程度か，

新たな開発プロセスを作成する際にどのような基準で既存のプロセスを再利用すれば良いか，といったように実際に再利用を促すためには課題はまだ多い．今後の研究の方向性としては，記述されたプロセスモデルから再利用可能な箇所を抽出するための方法論や，抽出した箇所を定量的に評価することが可能なプロセスメトリクスの提案などが考えられる．

6 おわりに

本稿では，ソフトウェアプロセスの記述法，およびプロセス評価・分析手法について紹介した．本稿で紹介した手法は特定のプロセスモデルを対象としたものではなく，汎用的に利用できるものを挙げた．

本稿がプロセス評価，改善に取り組まれている実務者の方，およびソフトウェアプロセスに関する研究を始めようとしている方の一助になれば幸いである．

参 考 文 献

[1] Ambler, S. W.: *Process patterns: building large-scale systems using object technology*, Cambridge University Press, New York, NY, USA, 1998.

[2] Basili, V. R. and Rombach, H. D.: Tailoring the Software Process to Project Goals and Environments, in *Proc. 9th International Conference on Software Engineering* (*ICSE '87*), 1987, pp. 345–357.

[3] Basili, V. R. and Weiss, D. M.: A Methodology for Collecting Valid Software Engineering Data, *IEEE Trans. Softw. Eng.*, Vol. 10, No. 3(1984), pp. 728–738.

[4] Becker-Kornstaedt, U., Hamann, D., Kempkens, R., Rösch, P., Verlage, M., Webby, R. and Zettel, J.: Support for the Process Engineer: The Spearmint Approach to Software Process Definition and Process Guidance, in *Proc. 11th International Conference on Advanced Information Systems Engineering* (*CAiSE'99*), 1999, pp. 119–133.

[5] Bird, C., Nagappan, N., Devanbu, P., Gall, H. C. and Murphy, B.: Does distributed development affect software quality? An empirical case study of Windows Vista, in *Proc. 31st International Conference on Software Engineering* (*ICSE'09*), 2009, pp. 518–528.

[6] Boehm, B. and Port, D.: Conceptual Modeling Challenges for Model-Based Architecting and Software Engineering (MBASE), *Conceptual Modeling*, Goos, G., Hartmanis, J., van Leeuwen, J., Chen, P., Akoka, J., Kangassalu, H. and Thalheim, B.(eds.), Lecture Notes in Computer Science, Vol. 1565, Springer Berlin / Heidelberg, 1999, pp. 24–43.

[7] Carnegie Mellon University: Capability Maturity Model Integration (CMMI), http://www.sei.cmu.edu/cmmi/.

[8] Cook, J., Du, Z., Liu, C. and Wolf, A.: Discovering models of behavior for concurrent workflows, *Computers in Industry*, Vol. 53, No. 3(2004), pp. 297–319.

[9] Cook, J. and Wolf, A.: Discovering Models of Software Processes from Event-Based Data, *ACM Trans. Softw. Eng. Methodol.*, Vol. 7, No. 3(1998), pp. 215–249.

[10] Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, Boston, MA, USA, 1995.

[11] García, F., Vizcaino, A. and Ebert, C.: Process Management Tools, *IEEE Softw.*, Vol. 28, No. 2(2011), pp. 15–18.

[12] Gilbert, E. and Karahalios, K.: CodeSaw: A Social Visualization of Distributed Software Development, *Human-Computer Interaction - INTERACT 2007*, Baranauskas, C., Palanque, P., Abascal, J. and Barbosa, S. D. J.(eds.), Lecture Notes in Computer Science, Vol. 4663, Springer Berlin Heidelberg, 2007, pp. 303–316.

[13] Graves, T. L., Karr, A. F., Marron, J. S. and Siy, H.: Predicting Fault Incidence Using Software Change History, *IEEE Trans. Softw. Eng.*, Vol. 26, No. 7(2000), pp. 653–661.

[14] Hanson, K. T.: Project visualization for software, *IEEE Softw.*, Vol. 23, No. 4(2006), pp. 84–92.

[15] Huo, M., Zhang, H. and Jeffery, R.: A Systematic Approach to Process Enactment Analysis as Input to Software Process Improvement or Tailoring, in *Proc. 13rd Asia Pacific Software Engineering Conference* (*APSEC'06*), 2006, pp. 401–410.

[16] Hurtado Alegría, J. A., Bastarrica, M. C., Quispe, A. and Ochoa, S. F.: An MDE approach to software process tailoring, in *Proc. 2011 International Conference on Software and Systems Process* (*ICSSP2011*), 2011, pp. 43–52.

[17] IEEE Computer Society: *Guide to the Software Engineering Body of Knowledge (SWEBOK) 2004 Version*, IEEE, 2004.

[18] Ihara, A., Ohira, M. and Matsumoto, K.: An Analysis Method for Improving A Bug Modification Process in Open Source Development, in *Proc. 10th International Workshop on Principles of Software Evolution* (*IWPSE'09*), Amsterdam,The Netherland, 2009, pp. 135–143.

[19] Katayama, T.: A hierarchical and functional software process description and its enaction, in *Proc. 11th International Conference on Software Engineering* (*ICSE'89*), 1989, pp. 343–352.

[20] Kellner, M. I., Becker-Kornstaedt, U., Riddle, W. E., Tomal, J. and Verlage, M.: Process Guides: Effective Guidance for Process Participants, in

Proc. 5th International Conference on the Software Process, 1998, pp. 11–25.

[21] Kurniawati, F. and Jeffery, R.: The use and effects of an electronic process guide and experience repository: a longitudinal study, *Information and Software Technology*, Vol. 48, No. 7(2006), pp. 566–577.

[22] McFeele, B.: IDEAL: A User's Guide for Software Process Improvement, Technical Report CMU/SEI-96-HB-00, Software Engineering Institute, Carnegie Mellon University, 1996.

[23] McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J. and Hall, F.: *Practical Software Measurement: Objective Information for Decision Makers*, Addison Wesley Professional, 2001.

[24] M.Johnson, P., Kou, H., Agustin, J., Chan, C., Moore, C., Miglani, J., Zhen, S., and Doane, W. E. J.: Beyond the personal software process: metrics collection and analysis for the differently disciplined, in *Proc. 25th International Conference on Software Engineering* (*ICSE'03*), 2003, pp. 641–646.

[25] Mockus, A., Fielding, R. T. and Herbsleb, J. D.: Two case studies of open source software development: Apache and Mozilla, *ACM Trans. Softw. Eng. Methodol.*, Vol. 11, No. 3(2002), pp. 309–346.

[26] Moe, N. B. and Dybå, T.: The Use of an Electronic Process Guide in a Medium-sized Software Development Company, *Software Process: Improvement and Practice*, Vol. 11, No. 1(2006), pp. 21–34.

[27] Morisaki, S., Matsumura, T., Ohkura, K., Fushida, K., Kawaguchi, S. and Iida, H.: Fine-Grained Software Process Analysis to Ongoing Distributed Software Development, in *Proc. 1st Workshop on Measurement-based Cockpits for Distributed Software and Systems Engineering Projects* (*SOFTPIT 2007*), Munich, Germany, 2007, pp. 26–30.

[28] Moser, R., Pedrycz, W. and Succi, G.: A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction, in *Proc. 30th International Conference on Software Engineering* (*ICSE'08*), 2008, pp. 181–190.

[29] Münch, J. and Heidrich, J.: Software project control centers: concepts and approaches, *Journal of Systems and Software*, Vol. 70, No. 1-2(2004), pp. 3–19.

[30] Object Management Group: Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0, http://www.omg.org/spec/SPEM/2.0/.

[31] Ohkura, K., Goto, K., Hanakawa, N., Kawaguchi, S. and Iida, H.: Project Replayer with Email Analysis - Revealing Contexts in Software Development, in *Proc. 13th Asia Pacific Software Engineering Conference* (*APSEC'06*), 2006, pp. 453–460.

[32] Osterweil, L.: Software processes are software too, in *Proc. 9th international conference on Software Engineering* (*ICSE'87*), 1987, pp. 2–13.

[33] Ostrand, T. J. and Weyuker, E. J.: Predicting the Location and Number of Faults in Large Software Systems, *IEEE Trans. Softw. Eng.*, Vol. 31, No. 4(2005), pp. 340–355.

[34] Park, S., Na, H., Park, S. and Sugumaran, V.: A semi-automated filtering technique for software process tailoring using neural network, *Expert Systems with Applications*, Vol. 30, No. 2(2006), pp. 179–189.

[35] Phongpaibul, M., Koolmanojwong, S., Lam, A. and Boehm, B. W.: Comparative Experiences with Electronic Process Guide Generator Tools, *Software Process Dynamics and Agility*, Wang, Q., Pfahl, D. and Raffo, D.(eds.), Lecture Notes in Computer Science, Vol. 4470, Springer Berlin / Heidelberg, 2007, pp. 61–72.

[36] Project Management Institute: プロジェクトマネジメント知識体系ガイド第 4 版, Project Management Institute, 2008.

[37] Rombach, D.: Integrated Software Process and Product Lines, *Unifying the Software Process Spectrum*, Li, M., Boehm, B. and Osterweil, L.(eds.), Lecture Notes in Computer Science, Vol. 3840, Springer Berlin Heidelberg, 2006, pp. 83–90.

[38] Rubin, V., Günther, C. W., van der Aalst, W. M. P., Kindler, E., van Dongen, B. F. and Schäfer, W.: Process Mining Framework for Software Processes, *Software Process Dynamics and Agility*, Wang, Q., Pfahl, D. and Raffo, D.(eds.), Lecture Notes in Computer Science, Vol. 4470, Springer Berlin / Heidelberg, 2007, pp. 169–181.

[39] Sarma, A., Maccherone, L., Wagstrom, P. and Herbsleb, J.: Tesseract: Interactive Visual Exploration of Socio-Technical Relationships in Software Development, in *Proc. 31st International Conference on Software Engineering* (*ICSE'09*), 2009, pp. 23–33.

[40] SCAMPI Upgrade Team: Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, Version 1.3: Method Definition Document, Technical Report CMU/SEI-2011-HB-001, Software Engineering Institute, 2011.

[41] Shihab, E., Ihara, A., Kamei, Y., Ibrahim, W. M., Ohira, M., Adams, B., Hassan, A. E. and Matsumoto, K.: Predicting Re-opened Bugs: A Case Study on the Eclipse Project, in *Proc. 17th Working Conference on Reverse Engineering* (*WCRE2010*), 2010, pp. 249–258.

[42] Śliwerski, J., Zimmermann, T. and Zeller, A.: When do changes induce fixes?, in *Proc. 2005 International Workshop on Mining Software Repositories* (*MSR2005*), 2005, pp. 24–28.

[43] The Eclipse Foundation and Eclipse Process Framework Project: Eclipse Process Framework Composer, http://www.eclipse.org/epf/.

[44] University of Southern California and Carnegie Mellon University: MBASE 577 Interactive Process Guide, http://sunset.usc.edu/research/MBASE/

EPG/home.html.
[45] Yamada, S., Ugumori, M. and Kusumoto, S.: A Software Tag Generation System to Realize Software Traceability, in *Proc. 17th Asia Pacific Software Engineering Conference* (*APSEC2010*), 2010, pp. 423–432.
[46] Zhang, H., Kitchenham, B. and Pfahl, D.: Software Process Simulation Modeling: an Extended Systematic Review, *New Modeling Concepts for Today's Software Processes*, Münch, J., Yang, Y. and Schäfer, W.(eds.), Lecture Notes in Computer Science, Vol. 6195, Berlin, Heidelberg, Springer Berlin Heidelberg, 2010, pp. 309–320.
[47] Zimmermann, T., Premraj, R. and Zeller, A.: Predicting Defects for Eclipse, in *Proc. Third International Workshop on Predictor Models in Software Engineering* (*PROMISE2007*), 2007.
[48] 伏田享平, 大前勇輝, 名倉正剛, 川口真司, 大蔵君治, 飯田元: バグ管理システムを対象としたアジャイルソフトウェア開発における保守プロセスの観察, 電子情報通信学会技術報告, ソフトウェアサイエンス研究会, Vol. SS2008-39, No. 362, 2008, pp. 1–6.
[49] 伏田享平, 高田純, 米光哲哉, 福地豊, 川口真司, 飯田元: AQUAMarine: 定量的管理計画立案システム, *SEC journal*, Vol. 5, No. 4(2009), pp. 244–251.
[50] 飯田元, 萩原剛志, 井上克郎, 鳥居宏次: ソフトウェア開発作業系列の形式的定義と誘導システムの生成, 情報処理学会論文誌, Vol. 34, No. 3(1993), pp. 523–532.
[51] 小林隆志, 林晋平: データマイニング技術を応用したソフトウェア構築・保守支援の研究動向, コンピュータソフトウェア, Vol. 27, No. 3(2010), pp. 13–23.
[52] 纐纈伸子, 川村真弥, 野村准一, 野中誠: プロセスおよびプロダクトメトリクスを用いた Fault-Prone クラス予測の適用事例, 情報処理学会研究報告. ソフトウェア工学研究会報告, Vol. 2010, No. 6, 2010, pp. 1–8.
[53] まつ本真佑, 亀井靖高, 門田暁人, 松本健一: 開発者メトリックスに基づくソフトウェア信頼性の分析, 電子情報通信学会論文誌 D, Vol. 93, No. 8(2010), pp. 1576–1589.
[54] 森崎修司, 松村知子, 大蔵君治, 伏田享平, 川口真司, 飯田元: エンピリカルデータを対象としたマイクロプロセス分析, 情報処理学会研究報告, ソフトウェア工学研究会, Vol. 2006, No. 125, 2006, pp. 9–15.
[55] 大平雅雄, 横森励士, 阪井誠, 岩村聡, 小野英治, 新海平, 横川智教: ソフトウェア開発プロジェクトのリアルタイム管理を目的とした支援システム, 電子情報通信学会論文誌 D-I, Vol. J88-D-I, No. 2(2005), pp. 228–239.
[56] 鈴木正人: 代表的なプロセス記述言語の特徴 : 共通例題による比較, 情報処理, Vol. 36, No. 5(1995), pp. 392–398.
[57] 吉村健太郎: 製品間を横断したソフトウェア共通化技術〜ソフトウェアプロダクトラインの最新動向〜, 情報処理, Vol. 48, No. 2(2007), pp. 171–176.
[58] 独立行政法人 情報処理推進機構ソフトウェア・エンジニアリング・センター (編): 改訂版 組込みソフトウェア向け開発プロセスガイド, 翔泳社, 2007.
[59] 独立行政法人 情報処理推進機構ソフトウェア・エンジニアリング・センター (編): 共通フレーム 2007 第 2 版〜経営者, 業務部門が参画するシステム開発および取引のために〜, オーム社, 2009.
[60] 経済産業省, 社団法人 日本情報システム・ユーザー協会: 企業 IT 動向調査 2011, 社団法人 日本情報システム・ユーザー協会, 2011.

伏田享平

2005 年大阪府立大学工学部電気電子システム工学科中退．2010 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年同大学情報科学研究科博士研究員．2011 年同特任助教．博士 (工学)．ソフトウェア工学，特にソフトウェアプロセス，ソフトウェアデザイン，リポジトリマイニングの研究に従事．情報処理学会，電子情報通信学会，IEEE 各会員．

飯田　元

1988 年大阪大学基礎工学部情報工学科卒業．1991 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．1995 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．2005 年同大学情報科学研究科教授．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．情報処理学会，電子情報通信学会，IEEE，ACM 各会員．