論　文

利益予測に基づくソフトウェアプロセス改善の試み

坂本　啓司[†,†††]　田中　敏文[†]　楠本　真二[††]　松本　健一[†††]
菊野　亨[††]

An Improvement of Software Process Based on Benefit Estimation

Keishi SAKAMOTO[†,†††], Toshifumi TANAKA[†], Shinji KUSUMOTO[††],
Ken-ichi MATSUMOTO[†††], and Tohru KIKUNO[††]

あらまし　ソフトウェア開発プロセスの改善は，通常，開発プロセスの現状把握と分析，分析結果に基づく改善策の作成と実行，に分けて実施される．これまでに多くの開発プロセス改善の提案があるが，それらはプロセスが具備すべき条件の評価に重点がおかれており，たとえ高い評価を受けても開発されるソフトウェアの品質が良くなる保証はない．また，提示される改善目標は抽象度が高く，開発現場に特化した改善計画を立てることは難しい．本論文では，これらの問題点を解決する一つのアプローチとして開発者指向のプロセス改善の枠組みを提案する．提案する枠組みでは，現状の開発プロセスを形式的に記述し，その定量的分析結果に基づいて実行可能な改善計画を開発者に提示する．更に，改善計画に基づいてプロセス改善を行うことで達成される工数の削減量を定量的に予測し，それらを具体的な利益の形で提示することで，開発者にプロセス改善の強い動機付けを与えるところに特徴がある．提案した枠組みをある組込みソフトウェアの開発を行う実際のプロジェクトに適用して評価実験を行った．その結果，そのプロジェクトにおいてほぼ予測に近い工数削減が達成されていることを確認できた．

キーワード　ソフトウェアプロセス改善，プロジェクト管理，メトリクス，ペトリネット

1. まえがき

ソフトウェアシステムの大規模化，複雑化に伴いソフトウェア開発における生産性，及び，品質向上の実現はソフトウェア工学における研究の主要な目標に位置づけられてきている．ソフトウェアの品質や生産性を向上させるためには，開発されたソフトウェアプロダクトだけでなく，その開発プロセスを対象として作業の改善を行うことが必要である[1]．開発プロセスの改善は，通常，開発プロセスの現状把握と分析，分析結果に基づく改善策の作成と実行，に分けて実施される．

これまでに，様々なプロセス品質評価手法が提案されている[3],[4]．Humphreyによって提案されたプロセス成熟度モデル[3]はソフトウェア開発組織が信頼性の高い安定したプロセスをどうすれば作っていけるのかをモデル化したものである．その目的はソフトウェア開発組織の状況をプロセスの観点から分析し，その組織に合ったプロセス改善の目標を設定することにある[9]．一方，ISO9000-3は二者間契約におけるソフトウェア品質保証システムの規格であり，品質の高いソフトウェアを開発するためのガイドラインがプロセス全体にわたって提示されている．また，各種の文書を用意するという管理的な側面が重視されている．

数多くのソフトウェア開発組織がプロセス成熟度モデルやISO9000-3による評価を受けてきている．しかし，これらはプロセスが具備すべき条件を評価するものであり，たとえ評価に合格しても開発されるソフトウェアそのものの品質が良くなる保証はない．また，改善策の作成と実行においては，改善目標の抽象度が高く，開発現場に特化した改善計画を立てることは難しい．更に，トップダウン的に改善計画が与えられる

† オムロン株式会社開発・生産センタ，草津市
Development & Production Division, OMRON Corporation, 2-2-1 Nishikusatsu, Kusatsu-shi, 525-0035 Japan
†† 大阪大学大学院基礎工学研究科，豊中市
Graduate School of Engineering Science, Osaka University, Toyonaka-shi, 560-8531 Japan
††† 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma-shi, 630-0101 Japan

ため，現場の開発者に十分な動機付けを与えるのが難しく，結果として現場への導入を困難なものとしてしまう．

本論文では，現場の開発者にプロセス改善の動機付けを与えることを目的として，開発者指向のプロセス改善のための枠組みを提案する．提案する枠組みでは，現状の開発プロセスを記述し，開発者にとって実行可能な改善計画を提示する．更に，改善計画に基づいてプロセス改善を行うことで得られる利益を定量的に予測して，開発者にプロセス改善の動機付けを与えるところに特徴がある．提案した枠組みを，ある企業での実際のソフトウェア開発プロジェクトに適用してその有効性を評価した．具体的には，ある組込みソフトウェア開発プロジェクトに対して，現状のプロセスの記述，問題点の把握，改善計画の作成，改善計画の実施による利益予測，利益予測の開発者への提示，改善計画の実行，実行結果の評価，からなる一連の手順を適用した．その結果，ほぼ予測に近い改善結果を確認できた．

以降，**2.** では，改善対象プロジェクトの概要と改善目標について述べる．**3.** では，提案するプロセス改善の枠組みについて紹介する．**4.** では，提案したプロセス改善の枠組みの適用について述べる．**5.** では，適用結果に対する考察と枠組みに対する開発者からの意見をまとめる．最後に，**6.** でまとめと今後の課題について述べる．

2. 準　　備

2.1 対象プロジェクト

プロセス改善の対象となるプロジェクトは，ある一連の組込みソフトウェア開発プロジェクトの一つである．各プロジェクトでは，既存ソフトウェアに対してユーザの要求に応じて一部の機能だけを変更した類似ソフトウェアの開発がなされている．通常，これらのプロジェクトは少なくとも3年間続けられる．プロセス改善を始めた時点で，二つのプロジェクト（PR_1 と PR_2）が既に終了しており，次のプロジェクト PR_3 が4か月後に始まる状況にあった．これら三つのプロジェクトの特徴を次の(1)～(7)にまとめる．

(1) 開発工数：プロジェクト当り 20～50 人月（平均 30 人月）

(2) 開発期間：プロジェクト当り 3～7 か月（平均 5 か月）

(3) プロジェクトメンバ：ほぼ同じメンバがすべてのプロジェクトを行う．

(4) 開発環境：UNIX ワークステーション

(5) プログラミング言語：C 言語

(6) 動作環境：専用ハードウェアとリアルタイムモニタ

(7) 開発プロセス：ウォーターフォールモデル

開発プロセスは，構想設計（CD），機能設計（FD），構造設計（SD），モジュール設計（MD），プログラミング（PG），単体テスト（UT），結合テスト（IT），機能テスト（FT），受入れテスト（VT）の9工程から構成される．なお，UT と IT では開発者によるホワイトボックステストが，FT では開発者によるブラックボックステストが，VT ではテスト組織によるブラックボックステストが，それぞれ実施される．

プロジェクトで開発されるシステムの構造を簡単に説明する．システムはアプリケーション部と基本部の二つの部分から構成される．アプリケーション部は七つのモジュール（A, B, C, D, E, F, G）で，基本部は二つのモジュール（リアルタイムモニタと H）で，それぞれ構成される．これらのモジュールの中で，リアルタイムモニタはすべてのプロジェクトで共通に用いられ，変更はほとんど行われない．モジュール B, C, D, E, F, G は一部が変更され，A と H は大部分が変更される．したがって，モジュール A と H の他のモジュールとの結合テストが各プロジェクトにおける重要な工程の一つとなっている．

2.2 改善目標

事業としての視点からは，開発コストを最小に抑えることが各プロジェクトに求められる．つまり，開発コストに重大な影響を与える問題を識別し，それを解決することが望まれる(注1)．この数年間コスト削減を目標として，管理を行ってきたが，その効果は現れていなかった．したがって，**4.** で述べるケーススタディでは，コストの削減を第一の目標と設定した．

技術的な視点からは，開発工程の遅れによるプロセスの混乱に着目した．既に述べたように，開発プロセスはウォーターフォールモデルに従っており，理想的には各工程は連続して実行されるはずである．しかし，これまでに継続して収集してきた開発データを分析すると，他の工程と並列して行われている工程が数多く存在することが確認された．この原因をつきとめ，対

(注1)：品質と納期も重要であるが，現状では特に問題が発生していないため，今回の改善における目標とはしていない．

策を立てることがもう一つの目標である．

3. プロセス改善の枠組み

提案する枠組みでは，ソフトウェア開発プロセスの改善のためのガイドラインを提供する．類似の手法として Process Characterization がある [5]．

Process Characterization では，まず，現状のプロセスを “As Is”map として記述し，対策を立てるべき問題点を明らかにする．次に，問題点に対する改善項目を明らかにして，優先度を付ける．更に，プロセスの改善度合を計測するためのメトリクスを定義する．また，改善されたプロセスを “Should Be” map として記述する．しかし，プロセス改善を行う上で最も重要な開発者への動機付けは考慮されていない．

そこで，我々はプロセス改善の手順に，開発者への動機付けを含めるべきであると判断した．具体的には，改善計画の実施によって生じる利益とその計画の妥当性を開発者に定量的なデータを用いて示し，更に，(単に計画を提示するだけでなく) 改善作業そのものを開発者自身と協力して実施することを導入した．

提案する枠組みは次の 6 ステップから構成される．(1) 現状プロセスの記述，(2) 現状プロセスの分析，(3) 改善計画の作成，(4) 利益予測，(5) 改善計画の実行，(6) 改善計画の評価．本手法の特徴は次の(a)～(c)にまとめられる．

(a) 現状のプロセスと改善されたプロセスを記述する．

(b) 改善計画を導入した場合の利益予測を行う．

(c) 改善活動を開発者と協調して行う．

なお，実施においては，プロセス改善活動の中心的な役割を果たす SEPG (Software Engineering Process Group) を設立した．

以降では各ステップでの作業内容について詳細に説明する．なお，各ステップでは，SEPG が中心となって作業を行い，必要に応じて開発者と共同して作業を行う．

Step1 (現状プロセスの記述)：ソフトウェア開発プロセスは，相互に関係し合った多くの作業から構成されている．したがって，開発プロセスの現状を正確に把握するためには，プロセスを形式的に記述する必要がある．

提案する枠組みでは，現状のプロセスをペトリネットを用いて記述する [7]．開発作業をトランジションで，プロダクトをトークンで表す．プレースは次の作業実行の待ち状態を表す．

Step2 (現状プロセスの分析)：現状のプロセスから収集したデータに基づいて，コスト，品質，納期に関する問題点を明らかにする．例えば，多くのコストを要した作業，品質に影響を与えた作業 (フォールトを多く作り込んだ作業) を識別する．更に，開発者との議論を通じて，その原因を徹底的に究明する．

Step3 (改善計画の作成)：Step2 で発見した問題点に対する改善計画を作成する．改善計画は，改善されたプロセスの記述とプロセスを構成する各作業に対する詳細な対策から構成される．改善されたプロセスの記述にもペトリネットを用いる．詳細な対策はすべて文書化される．

Step3 では，ソフトウェア工学の技術 (例えば，設計手法，レビュー技法，テスト手法) や過去のプロセス改善で得られた経験を改善計画に導入することを試みる．導入においては，対象となるプロセスへの導入可能性の検討を開発者と十分に行い，同意を得ることが重要である．

Step4 (利益予測)：利益予測では，改善計画の導入前と導入後における効果 (改善計画を導入することによる工数の削減量等) を定量的に評価する．複数の改善計画がある場合には，予測結果から最も適切なものを採用する．予測には様々な方法が考えられるが，過去の類似プロジェクトで収集したデータを用いて，改善プロセスのシミュレーションを行うことは，最も効果的な手法である．

Step5 (改善計画の実行)：Step4 で採用した改善計画を，改善対象プロジェクトへ適用し，実行する．SEPG は改善計画の実行状況を観察し，問題が発生すれば援助を行う．

Step6 (改善計画の評価)：対象プロジェクトが終了した後で，改善計画の評価を行う．例えば，Step4 で予測した利益が実際に達成されているかどうかを検証する．

4. ケーススタディ

4.1 プロセス改善の進捗

ここでは提案した枠組みのケーススタディの進捗概要について述べる．筆者のうちの 2 名が SEPG として参加し，プロセス改善作業に 10 人月の工数を要した．

1994 年 4 月に最初の全体ミーティングを開催した．そこではすべてのプロジェクトメンバ (管理者，開発者 (うち 1 名が開発リーダ)) に対して，プロセス改

善の必要性を説明した．プロジェクト管理者はプロセス改善の実施を承認した．

Step1 は 1994 年 4 月から 7 月にかけて行われた．その結果，プログラミング工程とテスト工程に多くの工数が費やされていることを確認した．引き続き Step2～Step4 を実施し，これらの工程における問題点を分析し，改善計画を作成した．更に，改善計画による利益予測を実施した．

8 月に 2 回目の全体ミーティングを開催した．そこでは，現状のプロセスの問題点，改善計画，改善計画による利益予測をプロジェクトメンバに説明した．管理者は改善計画の実行を承認した．

8 月から翌年の 3 月にかけて，改善計画に基づいて新しいプロジェクト PR_3 が実施された．プロジェクト終了後，改善結果の評価を行った．以降では，Step1～Step5 についての詳細内容について述べる．

4.2 現状プロセスの記述（Step1）

ソフトウェア開発プロセスを理解するため，既に開発を終了している二つのプロジェクト PR_1 と PR_2 の開発者と議論を重ね，図 1 に示すプロセス記述を作成した．FT&VT のトランジションの詳細については，図 2 に示す．

図 1 と図 2 の記述を作成した過程は次のとおりである．議論の最初，開発者は「何を実際に行ったか」というよりは，「何を行うべきであったか」という議論に終始した．これが，現状プロセスの理解に 3 か月もの期間を要した理由である．そのため，PR_1 と PR_2 で収集したデータ（例えば，各工程の実工数や各工程で発見されたフォールト数）を提示し，開発者が各工程で何をして，何をしなかったかということの事実確認を繰り返して行った．最終的に，図 1 と図 2 に示す現状プロセスの記述に成功した．

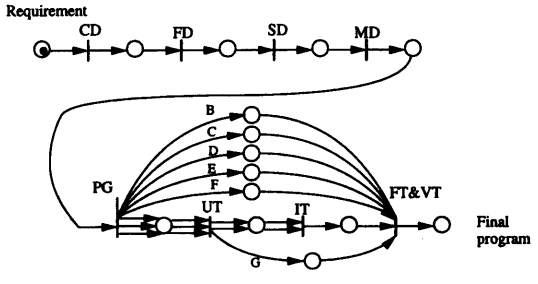

図 1　現状プロセスの記述
Fig. 1　Current process description.

図 1，図 2 からは，次の(1)～(3)が認識できる．

(1)　五つのモジュール B，C，D，E，F は PG 終了後すぐに，FT&VT を実施されている．

(2)　モジュール G は UT 終了後すぐに，FT&VT を実施されている．

(3)　FT と VT が並行して行われている．

4.3 現状プロセスの分析（Step2）

現状プロセスの記述に実際のデータを割り当てて分析した結果，ブラックボックステスト（FT と VT）に多くの工数を要していることを確認した．これは，納入直前の FT と VT が混乱した状況となっていることを意味する．実際に，PR_1 と PR_2 では，制御不能なほどに混乱した状態になり，そのためプロジェクトメンバは多くの時間を費やしていた．これを定量的に確認するために，ブラックボックステストに対して次の三つのメトリクス M_1，M_2，M_3 を導入した．

$$M_1 = \frac{\text{ブラックボックステストの工数}}{\text{全プロジェクト工数}}$$

$$M_2 = \frac{\text{ブラックボックステストで発見されたフォールト数}}{\text{コードレビュー以降に発見されたフォールト数}}$$

$$M_3 = \frac{VT\text{開始後の}\ FT\text{の工数}}{FT\text{の全工数}}$$

なお，M_2 の定義式で，「コードレビュー以降に発見された」は，「コードレビューとテストで発見された」ことを意味している．

M_1 と M_2 は，テストの効果を評価するために従来から用いていたメトリクスである．これまでに継続してブラックボックステストで発見されたフォールト数についてのデータを収集・分析していたため，これら

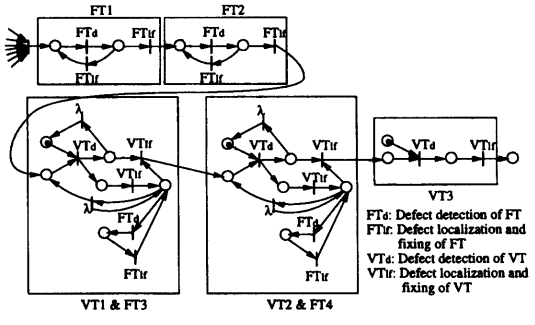

図 2　FT & VT の詳細記述
Fig. 2　Details of transition FT & VT.

M_1, M_2 の値が出荷後の品質や生産性と強い相関をもつことが確認されている．一方，M_3 は二つの工程 FT と VT における制御不能状態を評価するために新しく導入した．

評価結果を表 1 の PR_1, PR_2 の列に示す．これらの値と開発者へのインタビューから，以下の問題点(1)〜(3)を確認した．

(1) M_1：開発工数の約半分がブラックボックステストに費やされている．これは，納入直前の数か月に徹底してブラックボックステストを実施していたことと一致している．

(2) M_2：ブラックボックステストでほとんどのフォールトが発見されている．この事実から，プロジェクト PR_1 と PR_2 では，開発効率が非常に低く，プロダクトの品質も悪いことを示唆する．

(3) M_3：VT が開発者によるテストが終了する前に始められている．管理者と開発者への納期のプレッシャーによるあせりが現れていると考えられる．

そこで SEPG は PG とテスト工程（UT, IT, FT, VT）における問題が明らかになるまで，開発者に対して以下の質問 Q_1, Q_2, Q_3 を繰り返し行った．

Q_1：ブラックボックステストで発見されたフォールトは，どの工程で作り込まれたのか．

Q_2：ソースコードのレビューをどのように行ったか．

Q_3：ホワイトボックステストをどのように行ったか．

その結果，以下にまとめる原因と思われる事実が明らかになった．

(1) ほとんどのフォールトは PG で作り込まれているが，コードレビューではほとんど発見されていない．これはレビュー方法が適切でなかったためである．

(2) 更に，ホワイトボックステストでは，ほとんどフォールトが発見されていない．これはホワイトボックステストが明確に実施されていなかったことを意味する．

(3) 開発者はコードレビューとホワイトボックステストの必用性を認識しているが，納期に対する強いプレッシャーが，これらの作業の実施を妨げていた．

表 1 FT と VT に対するメトリクス
Table 1 Metrics for FT and VT.

	PR_1	PR_2	PR_3
M_1	47%	49%	44%
M_2	96%	94%	51%
M_3	45%	34%	0%

4.4 改善計画の作成（Step3）

明らかになった問題点の解消を目指して改善計画を作成した．改善計画は改善されたプロセスの記述と記述上の各作業に対する詳細な指示から構成される．

図 3 に改善されたプロセスの記述を示す．図 3 において，IT_1 は 6 個のモジュール B, C, D, E, F, G の結合テストに，IT_2 はモジュール A とモジュール組（B, C, D, E, F, G）の結合テストに，IT_3 はモジュール H とモジュール組（B, C, D, E, F, G）の結合テストに，それぞれ対応している．各作業に対する指示は次のとおりである．

(1) より実用的で効率の良いコードレビューを実施する．

(1.1) レビューを 2 人で行う [2]（1 人がコードの説明をし，もう 1 人がその内容を確認する）．

(1.2) 1 時間当り 100 行から 200 行をレビューする．

(1.3) 新規作成された部分と変更された部分は必ずレビューする．

(2) UT に単体テストツールを導入するとともに，フォールト報告書の提出を求める．

(2.1) 新規作成された部分と変更された部分は必ずツールを用いてテストする．

(2.2) UT で発見・除去されたフォールトは必ず記録する．

(3) ソフトウェアの構造に合わせて IT を実施する．具体的には以下のように IT を実施する．

(3.1) IT (IT_1, IT_2, IT_3) を改善プロセスに記述された順番に行う．

(3.2) 各 IT でのテスト項目は，テストを実施する前に決定する．

(3.3) IT で発見・除去されたフォールトは必ず記録する（**2.2** と同様）．

(4) コードレビューと UT のスケジュールを制御

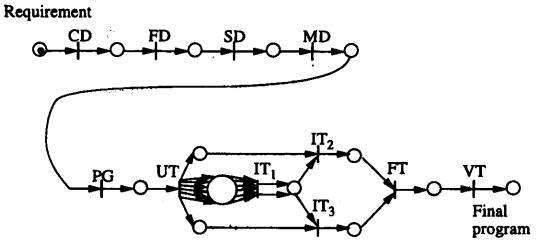

図 3 改善されたプロセス記述
Fig. 3 Improved process description.

可能にする．管理者は開発者に以下のことを要求する．

（4.1） コードレビューとUTにおける，各モジュールごとの進捗と発見されたフォールト数を記録する．

（4.2） （4.1）の記録を定期的に管理者に報告する．

前述のように，現状プロセスの把握には3か月を要した．その作業によって現状プロセスを完全に把握することができたため，改善計画の作成には2週間しかかからなかった．

4.5 利益予測（Step4）

2.2で述べたように，事業としては主に開発コストに焦点をあてている．したがって，予測される工数の削減で改善計画による利益を評価する．

SEPGは改善計画を PR_1 に適用した場合の総工数を予測した．以降，実際の PR_1 を PR_1^O，改善計画が適用された PR_1 を PR_1^I と表す．PR_1^I に対する予測では，次の(A1)～(A5)を仮定した．このうち，(A2)～(A5)の数字は，文献[8]から引用した．

（A1） PR_1^I のPGとテスト工程で発見される総フォールト数は，PR_1^O のPGとテスト工程で発見された総フォールト数と等しい．

（A2） PGのコードレビューにおいて，総フォールトの52.6%が発見される．

（A3） ホワイトボックステスト（UTとIT）で総フォールトの26.4%が発見される．

（A4） FTでは，総フォールトの15.9%が発見される．

（A5） VTでは，総フォールトの5.1%が発見される．

次に，各工程で必要とされる工数を算出するために，以下の式を用いた．

$$E_i = E_{im} + E_{id} + E_{ir} + E_{io}$$

$$E_{im} = \alpha_i \times s$$

$$E_{id} = \beta_i \times n_i$$

$$E_{ir} = \gamma_i \times n_i$$

$$E_{io} = \delta_i$$

ここで，E_i は開発プロセス中の i 工程 (i = PG, UT, IT, FT, VT) における総工数を表す．E_{im} は i 工程において，プロダクトを作成するために必用な工数を表す（ただし，i がUT，IT，FT，UTの場合は，これらの工程でプロダクトを作成しないため，$E_{im} = 0$ である）．E_{id} はフォールト発見に要する工数を，E_{ir} はフォールト修正に要する工数を，E_{io} は環境設定，文書作成等の工数を，それぞれ表す．また，s はプロダクトのサイズを，n_i は i 工程で発見されたフォールト数を，ぞれぞれ表す．

γ_i の値は PR_1^O での実際の値を用いた．一方，α_i，β_i，δ_i の値は，いくつかの過去のプロジェクトのデータを参考に，プロジェクトメンバとの議論を重ねて決定した．

表2に PR_1^O での各工程ごとに発見されたフォールト数と工数（人日）を示す．一方，表3には PR_1^I での予測結果を示す．

表2と表3から，提案した改善計画を適用することで，工数について14.7% ($1 - 1083/1270 = 0.147$) の削減が予測される．更に，同様の方法を PR_2 に適用した場合，11.3%の工数削減が予測された．

結果として，改善計画を実施することで約10%の工

表2 PR_1^O でのフォールト数と工数
Table 2 Defect and effort of PR_1^O.

工程	CD, FD, SD MD	PG	UT, IT	FT	VT	他	合計
フォールト数（分布（%））	-	14 (4.5)	0 (0)	219 (69.7)	81 (25.8)	-	314 (100.0)
工数（人日）（分布（%））	399 (31.4)	134 (10.5)	111 (8.8)	448 (35.3)	151 (11.9)	27 (2.1)	1270 (100.0)

表3 PR_1^I でのフォールト数と工数の予測
Table 3 Estimation of defect and effort of PR_1^I.

工程	CD, FD, SD MD	PG	UT, IT	FT	VT	他	合計
フォールト数（分布（%））	-	165 (52.6)	83 (26.4)	50 (15.9)	16 (5.1)	-	314 (100.0)
工数（人日）（分布（%））	399 (36.8)	155 (14.3)	199 (18.4)	227 (21.0)	76 (7.0)	27 (2.5)	1083 (100.0)

数削減が達成できると判断した．

4.6 改善計画の実行（Step5）

提案した改善計画に基づいて新規プロジェクト PR_3 を実行するための全体ミーティングを開いた．そこでは PR_1，PR_2 で明らかになった問題点，改善計画とそれを適用した場合の利益予測について説明を行い，管理者から承認を受けた．プロジェクトメンバは多くの会合を経て，改善計画の内容を完全に理解するのと同時に，プロセス改善の動機付けが十分になされた．

5. 改善計画の評価

5.1 結　　果

表1に，PR_3 における M_1，M_2，M_3 の値も併せて示している．

まず，PR_3 における M_1 の値は期待したほど向上していない．これは，改善計画によって PR_3 の他の工程の工数も同様に削減されたからである．一方，M_2 の値は大幅に改善された．半数のフォールトがブラックボックステストの前に発見されている．これは，プロダクトの品質が前工程でかなり保証されるようになって，結果としてフォールト修正の工数が削減されることを意味する．最後に，M_3 は0となった．VT工程が始まる時点で，PR_3 は完全に制御されている．その結果として，プロダクトの開発と納入は期限どおりになされた．

5.2 利益評価

5.2.1 PR_1 と PR_3 の比較

PR_3 との比較対象として PR_1 を選んだ．機能と開発規模の点で，PR_1 は PR_2 よりも PR_3 に対する類似度が高かったからである．

表2と表4に PR_1 と PR_3 の各工程における工数とフォールト数を示す．テスト工程のデータを比較すると，PR_3 のテスト効率と品質が PR_1 に比べて非常に改善されていることがわかる．PR_3 のテスト工程での総工数は，PR_1 に比べて約10%削減されている．この結果は，**4.5** における利益予測の結果とほぼ等しい．

5.2.2 PR_3 と仮想的な PR_3 の比較

ここで，実際のプロジェクト PR_3 を（改善されているという意味で）PR_3^I と表し，改善計画が実現されていない仮想的なプロジェクト PR_3 を定義してそれを PR_3^O と表す．

改善計画によって，削減された工数の大きさを評価するために，PR_3^O の総工数を評価した．

評価にあたっては次の(1)～(3)を仮定した．

(1) PR_3^O の PG とテスト工程で発見される総フォールト数は PR_3^I の PG とテスト工程で発見された総フォールト数と等しい．

(2) PR_3^I の PG と UT で発見されたフォールトはすべて PR_3^O の FT で発見，修正される．

(3) コードレビューと UT は PR_3^O では実施されない．

γ_i の値に PR_3^I の実測値を用いること以外は，**4.5** で述べたものと同じ式を用いた．

表4と表5に PR_3^I と PR_3^O の各工程ごとのフォールト数と工数の値を示す．結果として，11.8% $(1 - 1051/1192 = 0.118)$ の工数削減が実現されている．この割合は **4.5** で述べた利益予測の結果とほぼ等しい．

表4　PR_3^I でのフォールト数と工数
Table 4　Defect and effort of PR_3^I.

工程	CD, FD, SD MD	PG	UT	IT	FT	VT	他	合計
フォールト数（分布（%））	-	141 (31.7)	78 (17.5)	83 (18.7)	93 (20.9)	50 (11.2)	- -	445 (100.0)
工数（人日）（分布（%））	303 (28.9)	75 (7.1)	5.4 (5.1)	127 (12.0)	215 (20.5)	244 (23.2)	33 (3.2)	1051 (100.0)

表5　PR_3^O でのフォールト数と工数の予測
Table 5　Estimation of defect and effort of PR_3^O.

工程	CD, FD, SD MD	PG	UT	IT	FT	VT	他	合計
フォールト数（分布（%））	-	0 (0)	0 (0)	83 (18.7)	312 (70.1)	50 (11.2)	- -	445 (100.0)
工数（人日）（分布（%））	303 (25.5)	59 (4.9)	0 (0)	127 (10.6)	425 (35.7)	244 (20.5)	33 (2.8)	1192 (100.0)

5.3 プロジェクトメンバへの動機付けの効果

これまでに述べてきたように，今回のプロセス改善を通して，改善のための動機付けがプロジェクトメンバに対して十分に行われた．その結果，プロジェクトメンバは協力的であり，改善作業に対して非常に良い印象をもっていた．具体的な事例を以下に示す．

（1） 管理者はプロセス改善の必要性を完全に理解していた．

（2） 多くの会合や議論を通じて理解し合うことで，SEPGはプロジェクトメンバと改善目標を共有することができた．

（3） プロジェクトメンバは，PR_3 を実施している間，改善が実際に進んでいることを実感できた（例えば，UTやITを改善することで，従来FTで発見されていたようなフォールトを早期に発見できているということを確認できた）．

協調作業がうまくいった一例として，コードレビューとUT工程で使用される進捗報告書の書式を開発者の1人が積極的に提案していた．

更に，次のような意見も寄せられた．

（1） 納期直前の混乱が実際になくなった．

（2） 多くのフォールトが早期に除去されるので，開発プロセスがより信頼性の高いものとなった．

（3） SEPGはプロジェクトメンバの一員であるかのごとく，プロセス改善に真剣に取り組んでいた．

6. むすび

本論文では，開発者指向のプロセス改善のための枠組みを提案した．提案する枠組みでは，現状の開発プロセスを形式的に記述し，開発者にとって実行可能な改善計画を提示する．更に，改善計画に基づいてプロセス改善を行ったときに得られる利益を定量的に予測する．この予測結果を提示することによって，開発者に対するプロセス改善の動機付けを与えるところに特徴がある．更に，提案した枠組みを実際の開発プロジェクトに適用し，有効性の評価を行った．その結果，利益予測で得られた結果とほぼ同じ10%の工数削減が達成された．

今後の課題としては以下のことが挙げられる．

（1） より多くのプロジェクトに対するプロセス改善の枠組みの適用と評価

（2） 実際のプロジェクトをより簡単に記述するための手法の開発

（3） 利益予測をより系統的に行う手法の開発

今回のケーススタディでは，テスト工程の改善を目的としていた．今後，(1)については，設計工程やプログラミング工程に対する改善（例えば，オブジェクト指向設計やCASEツールの導入）を検討している．また，(3)については，既に一般化確率ペトリネットに基づくプロジェクトシミュレータを開発しており，現場への適用について検討を始めている[6]．

文　献

[1] V.R. Basili and H.D. Rombach, "The TAME project: Towards improvement-oriented software environment," IEEE Trans. Software Eng., vol.14, no.6, pp.758–773, 1988.

[2] D.B. Bisant and J.R. Lyle, "A two-person inspection method to improve programming productivity," IEEE Trans. Software Eng., vol.15, no.10, pp.1294–1304, 1989.

[3] W.S. Humphrey, Managing the Software Process, Software Engineering Institute, Addison-Wesley, 1989.

[4] 飯塚悦功（編），ソフトウェアの品質保証，日本規格協会，1992.

[5] P. Koltun, "Software process assessment: A first step to improvement," Proc. ISQE92, pp.4B31–4B39, 1992.

[6] S. Kusumoto, O. Mizuno, T. Kikuno, Y. Takagi, and K. Sakamoto, "A new software project simulator based on generalized stochastic petri-net," Proc. 19th Int. Conf. on Software Engineering, pp.293–302, Boston, May 1997.

[7] J.L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice-Hall, 1981.

[8] P. Piwowarski, M. Ohba, and J. Caruso, "Coverage measurement experience during function test," Proc. ICSE15, pp.287–301, 1993.

[9] 山田　茂，高橋宗雄，ソフトウェアマネジメントモデル入門，共立出版，1993.

（平成12年1月12日受付，2月28日再受付）

坂本　啓司

昭44神戸大・工・電気卒．同年オムロン（株）入社．平10奈良先端科学技術大学院大学博士後期課程入学．ソフトウェアプロセス改善に関する研究に従事．情報処理学会会員．

田中　敏文

昭54京大・工・数理卒．昭56同大大学院修士課程了．同年オムロン（株）入社，現在に至る（平10～12サイバーキャッシュ（株）取締役）．

楠本　真二　（正員）

昭63阪大・基礎工・情報卒．平3同大大学院博士課程中退．同年同大・基礎工・情報・助手．平8同大講師．平11同大助教授．工博．ソフトウェアの生産性や品質の定量的評価，プロジェクト管理に関する研究に従事．情報処理学会，IEEE各会員．

松本　健一　（正員）

昭60阪大・基礎工・情報卒．平1同大大学院博士課程中退．同年同大・基礎工・情報・助手．平5奈良先端科学技術大学院大学・助教授．工博．ソフトウェア開発における計測環境，ソフトウェア品質保証の枠組に関する研究に従事．情報処理学会，ACM，IEEE各会員．

菊野　亨　（正員）

昭50同大大学院博士課程了．工博．同年広島大学工学部講師．同大助教授を経て，昭62大阪大学基礎工学部情報工学科助教授．平2同大教授．主にフォールトトレラントシステム，ソフトウェア開発プロセスの定量的評価に関する研究に従事．平5本会論文賞受賞．情報処理学会，ACM，IEEE等各会員．