ソフトウェア開発プロセスの複雑さに着目したプロダクト品質の予測モデルの構築

尾花将輝[†]　花川典子[††]　飯田元[†]

ソフトウェア開発プロジェクトでは想定していた開発計画とプロジェクト終了後の開発実績には大きな隔たりがある．例えば，顧客の要望で機能要件が変更され，それに伴い変更作業プロセスが計画に追加される等である．予期せず追加されたプロセス（フラグメントプロセス）は開発プロセス全体を複雑化する．複雑になったプロセスはプロダクト品質に影響すると予想されるが，従来の研究ではプロセスとプロダクトの関係が明確に示されていない．そこで，本稿ではプロセスの複雑さに着目した PCPQ (Process Complexity - Product Quality)モデルを提案する．PCPQ モデルはフラグメントプロセスの追加によって複雑化したプロセスからリリース後のプロダクト品質を予測するモデルである．産業界における６つのプロジェクトを用いて PCPQ モデルを構築した．構築したモデルを用いて，他プロジェクトのリリース後のプロダクト品質を予測した結果，実測値に近い品質を予測できた．また，時系列変化するフラグメントプロセスの特徴に着目することで，リリース後の障害の種類を特定することの可能性についても検討した．

A process complexity based product quality model for software development

MASAKI OBANA[†]　NORIKO HANAKAWA[††]
HAJIMU IIDA[†]

In software development projects, large gaps between planned development processes and actual development exist. For example, processes of the requirement repeat the definitions by sudden demands of customers in a design process. We call the added process a fragment process. A development process becomes complicated by addition of such fragment processes, and product quality decreases. Therefore, we propose a PCPQ (Process Complexity - Product Quality) model with process complexity. A PCPQ model consists of two elements; process complexity, and product quality. Process complexity is measured by fragment processes. Product quality means a final specific gravity of post-release faults. Because a PCPQ model is constructed at each organization, post-release product quality can be predicted using the PCPQ model in the same organization. We constructed a PCPQ model using six industrial projects. After that, product qualities of other projects are predicted by the PCPQ model. As a result, a gap between the predicted product quality using the PCPQ model and real product quality is small. Moreover, we discuss relationships between characteristics of fragment process and faults after release.

1. はじめに

ソフトウェア開発プロジェクトにおいて，想定していた開発計画と実際の開発実績には大きな隔たりがある[1]. 例えば, 顧客の急な要望により計画にない作業の追加や，インクリメンタル開発プロセスで計画されたサブシステムの機能が全く異なる機能へと変更された等が挙げられる．開発計画に新たな作業が追加される事は開発現場では問題視されている．急な追加作業が発生することで作業進行が円滑に進まなくなり，結果としてプロダクトの品質低下が予測される．しかし，追加作業を行わなければ顧客にとって有益なシステムにならない．追加作業や計画変更を少なくするため，要件定義をより確実にする研究も行われているが[2][3], 顧客の満足度を向上させるために開発現場では細かな計画変更や追加作業は頻繁に発生する．追加作業や計画変更は避けられない事象としても，無節操に変更するのではなく，最終プロダクトに与える影響を考慮する必要がある．しかし，追加作業等により変更された開発プロセスがプロダクト品質へ与える影響が明確にはなっていない．そこで，本研究ではプロセスとプロダクト品質の関係を示す PCPQ モデル構築方法を提案する．PCPQ モデルはプロセスの複雑さの観点から最終的なプロダクト品質を予測する．PCPQ モデルの構築はプロジェクトの開発プロセスの複雑さを示した Process Complexity 値（以下 PC 値）と最終的なプロダクトの品質である障害比重の値を利用する．これを同一組織の複数プロジェクトにて計測し，各プロジェクトの最終的な PC 値と障害比重の値にてモデルを構築する．構築されたモデルを使って，同一組織内の他のプロジェクトの PC 値から最終プロダクトの品質を予測することができる．PC 値の計測方法として，実践的なプロジェクトで一般的に利用されている工程管理表を用いる[4]．工程管理表からプロジェクト当初に計画されたプロセス（以下ベースプロセス）と予期せず追加されたプロセス（以下フラグメントプロセス）を抽出する．PC 値のパラメータは抽出されたプロセスの同時実行プロセス数，開発グループ数及びプロセスの実行期間の値である．プロジェクトの進捗に伴い変化する工程管理表の構成管理より PC 値の変化をリアルタイムで計測できる．また障害比重の値は組織ごとに作成された障害管理表から求める．リリース後に発生した障害の重要度ランク（例えばシステムダウン等は最重要レベル

[†]奈良先端科学技術大学院大学
Nara Institute of Science and Technology.
[††] 阪南大学
Hannan University

等）とその障害件数から障害比重を求める．工程管理表や障害管理表は組織毎に記述方法や管理項目の粒度が異なる．しかし，PCPQ モデルは組織毎に作成するため，管理項目の粒度が異なったとしても構築できる汎用的なモデルである．PCPQ モデルを用いてプロジェクト実行中のPC値から最終プロダクトの品質を予測する事ができる．したがって，顧客の要件変更等の計画外の作業実行が最終プロダクト品質に与える影響を早期に判断し，計画の再設計の方針を決定することに利用できるモデルである．

２章ではプロセスメトリクスとプロジェクトリスク管理の関連研究を示し，３章ではプロセスの複雑さの計測手法と PCPQ モデルを提案する．４章では６つの実プロジェクトへの適用事例を示し，５章のプロセスの複雑さと障害発生について考察し，６章でまとめと今後の課題を述べる．

2. 関連研究

ソフトウェア開発プロセスをメトリクスで計測する研究は多く実施されている．CMM はハンフリーが示すプロセス成熟度モデルであり，組織におけるプロセスの成熟に関する５レベルを提案している[5]．レベルを決定する際に様々なメトリクス（総合試験欠陥率，試験密度，レビュー密度等）値が収集される．また，坂本らは企業内でウォータフォール開発のプロセス改善を組織的に実施した．レビュー実施回数や指摘事項数を計測するメトリクスとして，プロセス改善度合いを計測した[6]．

また，プロセスモデリングについての研究も多く提案されている．Cugola らは容易にタスクを追加することができるプロセスモデル言語を提案している[7]．Fuggetta らはソフトウェア開発環境における問題点や様々な開発プロセスにおける問題を解決するためのツールを提案している[8]．さらに，Garcia らはプロセスモデルの保守性や変更容易性などを，GQM ベースのメトリクスで評価している[9]．修正作業の追加を変更容易性として着目している点は我々の提案するフラグメントプロセスと似ている．これらのプロセスモデリングはプロジェクトをシミュレートすることでプロセスの評価や改善をすることが目的である．我々の研究では，プロジェクト内でプロセスの複雑さの変化を計測し,プロセスとプロダクトとの関係を明確にすることが目的である．

一方，実践的ベストプラクティスの経験をもとに，複数のプロセスを統合して成長した実践的なラショナル統一プロセスが提案されている[10]．反復型開発，ユースケース駆動，リスク駆動等の考え方を取り入れ，アジャイルや.NET 等の最新技術テーマにも対応した実質的な統合プロセスである．管理の問題や規模の問題もあるが，体系的なプロセスの実践には有益な統合プロセスである[11]．ひとつのプロジェクトで複数のプロセスが混在するという考え方は我々のプロセスの複雑さの考えに似ている．しかし，プロセス体系であるラショナル統一プロセスと，プロジェクト実行中に細かなプロセスが発生するフラグメントプロセスは根本的に異なる発想である．さらに提案する PCPQ モデルはプロセスとプロダクト品質の関係まで言及する点が異なる．

プロセスとプロダクト品質の関係に言及した研究は，青木らがアジャイル開発の実証データにてプロセスとプロダクトの関係を導いた[12]．レビュー回数やテストケース回数等のプロセスメトリクス値とシステムテスト工程のフォールト数の関係を重回帰式にて求めた．本研究はプロセスとプロダクトの関係を明確にするという研究目的は同じであるが，我々はプロセスの複雑さを用いるのに対し，プロセスメトリクスを

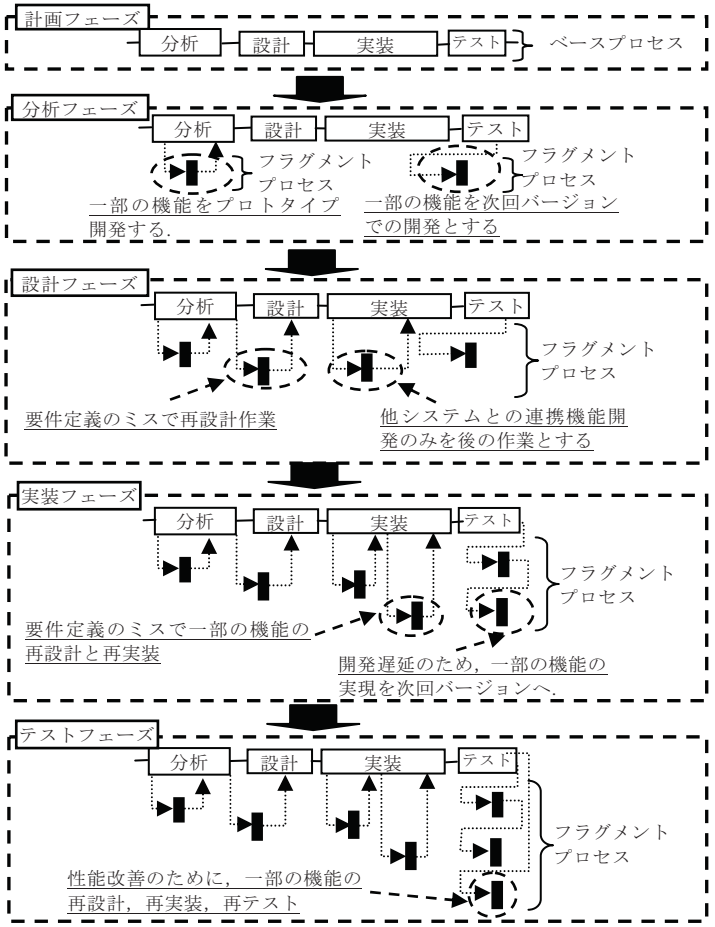


図 1　フラグメントプロセスの概念

Figure 1 A concept of fragment process

用いている点で異なる．また，プロセスメトリクスはプロジェクトが終了した時点で確定する値であるため，プロジェクト途中で計測できない点が本研究とは異なる．

3. Process Complexity の計測方法と PCPQ モデルの提案

PCPQ モデルはプロセスの複雑さから最終的なプロダクトの品質を予測するものである．本章では PCPQ モデルを構築するための Process Complexity 値（以下 PC 値）について説明した後，PCPQ モデルの提案と構築方法を述べる．

3.1 ベースプロセスとフラグメントプロセス

ソフトウェア開発において，開発全体をひとつのプロセスとして計画し実行することが多い．しかし，実際のプロジェクトでは計画工程で予定した開発プロセスでプロジェクトが終了することは稀である．例えばウォータフォール開発プロセスで開発計画を作成したとしても，顧客の要望によるプロトタイプの作成や，仕様変更，またテスト工程中のプログラム変更等により追加作業が随時発生し，プロジェクト全体としての開発プロセスは複雑化する．本稿ではプロジェクトの計画段階で決定された開発プロセスをベースプロセスと呼び，プロジェクト中に発生した計画外のプロセスをフラグメントプロセスと呼ぶ．フラグメントとは「断片」という意味であり，ベースプロセスに断片的に追加され，並行実行される様からフラグメントプロセスと命名した．

過去の事例の一部を用いたフラグメントプロセスの概念を図１に示す．計画フェーズでは分析，設計，実装，テストの単純なプロセスであった．多くの場合，計画時にはシステムに関する情報が少ない（分析されていない）ため，細かなプロセスというよりも大まかなプロセス（マクロプロセス）[13]で計画することが多い．しかし，プロジェクトが進行すると情報量が多くなり，プロセスも複雑に変化する．図１の場合，分析時には当初予定していなかったプロトタイプ開発が顧客の要望により追加された．また分析結果から期間とコスト制約により全ての機能の実装が難しいことが分かり，一部の機能を次バージョンへ移行した．さらに，設計フェーズでは要件定義ミスを発見し要件定義のやり直し作業や，他システムの連携設計では他システム開発進捗の遅れにより連携部分の実装を後にする等により計画は変更された．実装フェーズでは要件定義ミスや設計ミスによるやり直し作業が発生し，テストフェーズでは性能テスト結果に基づき設計のやり直し作業が発生した例が図１である．

このように顧客の要望変更，他システム関連などにより新たに発生した細かなプロセスがフラグメントプロセスである．フラグメントプロセスには簡単な仕様書の修正作業からひとつの機能を設計・実装・テストなどの大きな作業等，様々な粒度が存在するが，ここでは当初計画にない全ての一連の作業をフラグメントプロセスとする．

3.2 PC 値の計測方法

Process Complexity を定義する際の方針を以下に示す．

(1) あらゆる工程管理表から容易に抽出できる基本的な要素のみを利用する．これにより従来の定量的プロセス管理手法[14] の対象である CMM レベル５以上の組織である必要がなくなる．
(2) 抽出できる要素のうち，プロセス実行の複雑さに影響する要素に焦点を絞る．
(3) ひとつのプロセスフラグメントの複雑さは，複数要素がお互いに相乗的に影響するので，抽出された要素を乗算して求める．
(4) 複数のフラグメントプロセスが同時実行する場合，プロセス全体の複雑さは同時実行するフラグメントプロセスの複雑さの総和とする．
(5) 同一組織の工程管理表に記載された項目やその粒度は同一のものと仮定する．

上記の方針に基づき，提案するプロセスの複雑さの PC 値を以下に示す．

$$PC_{(t)} = \sum_{i=1}^{N_{(t)}} (Num\_dev_{(t)i} \times L_{(t)i} \times term_{(t)i}) \quad \cdots (1)$$

$PC_{(t)}$: 時刻 t のプロセスの複雑さ（時刻 0〜t までの終了プロセスも含む）
$N_{(t)}$: 時刻 t のベースプロセスを含む全プロセス総数（時刻 0〜t までの終了プロセスも含む）．
$Num\_dev_{(t)i}$: 時刻 t の i 番目のプロセスにかかわる開発者グループ数．
$L_{(t)i}$: 時刻 t の i 番目のプロセスの同時実行プロセス数．
$term_{(t)i}$: 時刻 t の i 番目のプロセスの実行期間の全体開発期間に対する割合．つまり，1.0 に近い場合はプロセスの実行期間がほぼ開発期間全体にわたる．

提案する PC 値のパラメータは，$Num\_dev_{(t)i}$ が時刻 t の i 番目のプロセスにかかわる開発者グループ数であり，$L_{(t)i}$ は時刻 t の i 番目のプロセスと同時実行するプロセス数，$term_{(t)i}$ は時刻 t の i 番目のプロセスのプロジェクト全体の開発期間に対する実行期間の割合を示す．工程管理表から抽出できる要素のうち，$Num\_dev_{(t)i}$ の開発者グループ数は一つの作業を複数人で担当することで工数が増える（つまり，プロセスが複雑になる）という観点から選択した．$L_{(t)i}$ の同時実行するプロセス数と $term_{(t)i}$ の実行期間は作業間の同期(コミュニケーション)をとることの難しさより選択した．選択された３要素はお互いに相乗的に影響を与える値であるので，それぞれを乗算した．

各要素を詳しく説明する．開発者グループ数 $Num\_dev_{(t)i}$ のグループ粒度はプロセスの粒度に依存する．例えば，プロセスに含まれる作業の粒度が日単位であると $Num\_dev_{(t)i}$ のグループ粒度は人単位が相当と考える．つまり，日単位の作業計画で管理するのは開発者毎の管理が適当である．同様に，プロセスに含まれる作業粒度が週単位であると，$Num\_dev_{(t)i}$ のグループ粒度はチーム単位が相当である．組織の工程管理表の管理単位ごとに $Num\_dev_{(t)i}$ のグループ粒度が決定される．一方，$term_{(t)i}$ はプロセスのプロジェクト全体に与える時間的影響量を示す．もし，あるプロセスの実行時間が非常に長い，つまりプロジェクトの全開発期間に相当する場合，そのプロセスは

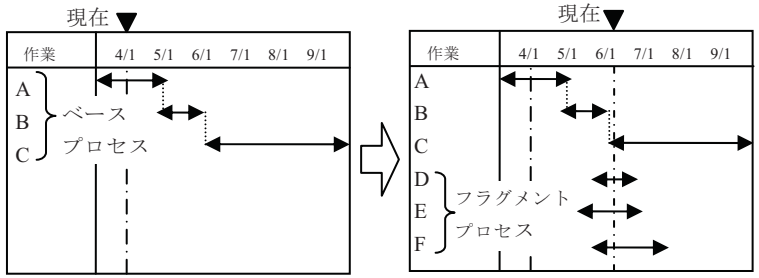


図 2 工程管理表からのプロセスフラグメント抽出例

Figure 2 Extracting fragment process from configuration of a workflow management table

大きくプロジェクト全体に影響を及ぼす. 反対にプロセスの実行時間が 1 日程度なら，そのプロセスはプロジェクトへの影響が小さいと考える．つまり，多くのプロセスが発生し，各プロセスの実行期間が長く，各プロセスが並行実行された場合，PC 値は大きくなる．逆に，プロセスの発生が少なく，各プロセスの実行期間が短く，各プロセスが並行実行されない場合，プロジェクトへの影響は小さいと考える

3.3 フラグメントプロセスの抽出方法と PC 値の計測例

PC 値は工程管理表に記載されたベースプロセスとフラグメントプロセスに基づき計測する．図２に工程管理表の例からフラグメントプロセスの抽出方法を示す．表の横軸は日付，縦軸は作業（プロセス）を示す．図２の左の表は計画段階にて作成されたベースプロセスだけを示し，右の表は５月１日に予期せぬ要件が追加され，ベースプロセスに３つのフラグメントプロセスが追加された工程管理表を示す．つまり，図２の A，B，C はベースプロセスであり，D，E，F はフラグメントプロセスを示している．この時の関わっている開発者はグループ単位とし，ベースプロセスが SE，PG，顧客の３グループ，フラグメントプロセスは PG が関わっていたとする．

図２の工程管理表から６月１日にはベースプロセスと３つのフラグメントプロセスが同時実行することとなる．つまり，プロジェクト開始時と現段階の工程管理表の差分からフラグメントプロセスの抽出や PC 値のパラメータの値を抽出することができる．本提案は工程管理表さえ存在すればアジャイル開発プロセス等の様々な開発プロジェクトでも利用可能である．同時にプロジェクト計画当初のベースプロセスの複雑さも計測することが可能である．

次に PC 値の計測例を示す．図２の例では，180 日のウォータフォールプロセス（ベースプロセス）があったとする．このプロセスに追加プロセスとして，フラグメントプロセス D，E，F が 30 日，45 日，60 日の追加作業が発生したと仮定する．この段階でのプロセスインスタンス数はベースプロセス１つとフラグメントプロセスが３つなり $N_{(t)}=4$ となる．ベースプロセスを $i=1$，追加されたフラグメントプロセスを $i=2,3,4$ とすると，時刻 t が６月１日の PC 値は以下の計算となる．

① $i=1$(ベースプロセス)
- (i) $Num\_dev_{(t)\_1}=3$
- (ii) $L_{(t)\_1}=4$
- (iii) $term_{(t)\_1}=180/180=1.0$

② $i=2$(図中の D のフラグメントプロセス)
- (i) $Num\_dev_{(t)\_2}=1$
- (ii) $L_{(t)\_2}=4$
- (iii) $term_{(t)\_2}=30/180=0.166$

③ $i=3$(図中の E のフラグメントプロセス)
- (i) $Num\_dev_{(t)\_3}=1$
- (ii) $L_{(t)\_3}=4$
- (iii) $term_{(t)\_3}=45/180=0.25$

④ $i=4$(図中の D のフラグメントプロセス)
- (i) $Num\_dev_{(t)\_4}=1$
- (ii) $L_{(t)\_4}=4$
- (iii) $term_{(t)\_4}=60/180=0.333$

従って，ベースプロセスに３つのフラグメントプロセスが追加されたプロセス複雑さは，$PC_{(t)}=12.000+0.664+1.000+1.332=14.996$ となる．

3.4 PCPQ モデルの提案

PC 値はプロセスの複雑さを計測したものである．これにより，追加されたフラグメントプロセスがプロセス全体にどの程度影響しているかを判断できる．しかし，PC 値だけではプロダクトの品質を予測することはできない．そこで， PCPQ(Process Complexity - Product Quality)モデルを提案する．PCPQ モデルは，最終的な PC 値と障害比重の値から構築することで，今後のプロジェクトの最終的なプロダクト品質を予測するモデルである．最終的な PC 値とはプロジェクト終了時の PC 値のことであり，障害比重とはリリース後に発生した障害の数とその重要度に基づいたプロダクト品質である．PCPQ モデルでは，X 軸に最終的な PC 値，Y 軸に障害比重の値とし，過去の複数プロジェクトの２つの値をプロットする．これにより， PC 値と障害比重の関係を明確にでき，モデル構築に利用しなかったプロジェクトの PC 値からリリース後の障害を予測するモデルである．また，工程管理表やリリース後の障害管理方法が組織ごとに異なるため，組織ごとの PCPQ モデルを作成する必要がある．例えば，ある組織の管理者は日毎の工程管理表を作成するが，他の組織の管理者は週単位の工程管理表を作成する. 提案する PC 値のプロセスの粒度は工程管理表の粒度に依存するので，組織ごとの PCPQ モデルとなる．同時に障害管理方法も組織ごとに異なる．特に，障

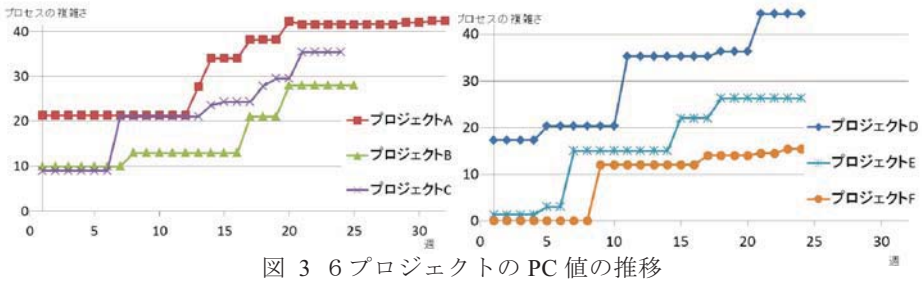


図 3 6プロジェクトの PC 値の推移

Figure 3 Changes of values of process complexities (moment version) of 6 projects

害の重要度は組織によって判断が異なる．例えば，システムダウンが最も深刻な障害と判断する組織に対して，要件定義エラーが最も深刻な障害と判断する組織もある．つまり，工程管理表での管理方法や障害の重要度の管理方法も組織によって異なるので，本モデルは組織ごとに構築される．ただし PCPQ モデルを構築するには組織毎で工程管理表と障害管理表の管理方法を統一する必要がある．PCPQ モデルは新規のプロジェクト進行途中に，定期的に PC 値を計測し過去のプロジェクトから構築された PCPQ モデルを用いることで，リリース後の障害を予測するモデルである．PCPQ モデルは以下の手順で構築される．

Step1： 同一組織で終了した複数プロジェクトの工程管理表と障害管理表を準備する．

Step2： 各プロジェクトの最終的な PC 値を計測する．

Step3： 障害管理表からリリース後の障害の重要度を決定する．

Step4： 各プロジェクトの障害比重を計算する．

Step5： X 軸に PC 値, Y 軸に障害比重の値をプロジェクトごとにプロットし，そのプロットを結ぶ近似線を求める．

上記の手順で求められた近似線が PCPQ モデルとなる．次章にて実際のプロジェクトの工程管理表を用いた PCPQ モデルの構築を示す．

表 1 6プロジェクトの詳細

Table 1 Details of 6 projects.

プロジェクト名	開発期間	システム種類
A	9ヶ月	教育ポータルWebアプリケーションシステム
B	7ヶ月	教育ポータルWebアプリケーションシステム
C	6ヶ月	クライアントサーバの授業支援システム
D	6ヶ月	クライアントサーバの授業支援システム
E	6ヶ月	クライアントサーバの授業支援システム
F	6ヶ月	クライアントサーバの授業支援システム

4. PCPQ モデルの適用

4.1 6プロジェクトへの適用

PC 値を産業界の6プロジェクト（表1参照）にて計測した．これらのプロジェクトは教育系のアプリケーションを専門に開発する同一組織のプロジェクトである．6プロジェクトともに同一組織による大学の教育系システムの開発であるプロジェクト A とプロジェクト C は新規開発であるが，それ以外のプロジェクトは母体システムのバージョンアップであった．

4.2 6プロジェクトの PC 値の推移

図3に6プロジェクトの PC 値の推移を示す．最も高い PC 値はプロジェクト D の 44.4 であり，最も低い値はプロジェクト F の 15.4 である．したがって，プロジェクト D は多くのフラグメントプロセスを含む複雑なプロセスであることがわかり，プロジェクト F はフラグメントプロセスが少ないシンプルなプロセスであることがわかる．

また，プロジェクト A は新規開発システムであり，プロジェクト D に次いで PC 値が高いプロジェクトである．開発者へのインタビューによると，顧客が新しいシステムを早期にイメージすることが難しく，顧客の要求が頻繁に変更され，追加作業が多くなった．つまり，要求変更により追加作業が多くなり，フラグメントプロセスの増加につながったと考えられる．同様にプロジェクト C も新規開発システムであり，顧客はシンプルな GUI を要望したが，開発者がイメージできず，結果プロトタイプを作成するフラグメントプロセスが発生した．また，テスト工程ではネットワークエラーによるネットワークチューニングのフラグメントプロセスが発生する等により PC 値が増加した．それに対して，プロジェクト B はすでに稼働している母体システムのバージョンアップであった．顧客も母体システムを熟知し，追加する機能のイメージも把握しているため要求変更が頻繁にされることはなかった．結果として，プロジェクト B では 15 週目の周辺で一時的に PC 値が増加するが，全体として低い値を維持できたと考えられる．また，プロジェクト D は母体システムのバージョンアップであったが，PC 値は開発期間を通して常に高かった．開発者へのインタビューによれば，プロジェクト D は前バージョンのシステムに多くの障害が発生し，その障害対応とプロジェクト D で本来実施すべき開発作業の同時進行が行われた．つまり，計画された新規機能の開発を行いながら，多くの障害対応というフラグメントプロセスが発生したこととなる．結果として，プロジェクト D の PC 値は常に高い値を示した．プロジェクト E は母体システムのリファクタリングが主な開発であった．したがって，新たな機能開発を行わずリファクタリングに集中したため, PC 値が比較的低くなったと考えられる．その後プロジェクト F はプロジェクト E で実装予定であった機能の開発であったため，機能内容や顧客の意思をかなり熟知しており，また母体システムも安定していたため全体を通して PC 値は低い値を維持した．

4.3 ６プロジェクトによる PCPQ モデルの構築

６プロジェクトによる PCPQ モデルの構築例を以下に示す．

Step1： 同一組織内の終了した６つのプロジェクトの工程管理表と障害管理表を用意する．

Step2： ６プロジェクトの最終の PC 値を計測する．表２に最終的な PC 値を示す．

Step3： リリース後の障害管理表から，各障害の重要度を決定する．表２に 5 段階ランク（SS，S，A，B，C）の重要度を付加された障害の件数を示す．この組織では SS ランクはシステムダウンレベルの障害を意味し，S ランクは重要な機能の欠如や性能問題を意味する．A ランクは中程度の重要な機能での問題，B ランクはユーザインタフェースの問題，C ランクは，メッセージや名称の問題を意味する．各障害はリリース後の障害管理表で管理され，各障害の重要度は管理者と顧客にて決定された．

Step4： 障害比重を計測する．障害比重は障害件数と重要度の積算で求める．例えば，SS を５，S を４，A を３，B を２，C を１の定数に設定すると，プロジェクト A の障害比重は 5*4 + 4*4 + 3*6 +2*9 +1*11=83 となる（表２障害比重参照）．同様に，プロジェクト B は 66，プロジェクト C は 82，プロジェクト D は 88，プロジェクト E は 37，プロジェクト F は 28 となる．

Step5： 各プロジェクトの最終の PC 値と障害比重の関係をプロットする．図４に，６プロジェクトの２つの値の関係をプロットし，それらを結ぶ近似線を示す．今回 PCPQ モデルを構築した組織の近似線は以下の式となる．

$$y = -0.0105X^3 + 0.9069X^2 - 22.017X + 189.69 \quad \cdots（２）$$

x: 最終の PC 値

y: 障害比重の値

式(2)がこの組織の PCPQ モデルとなる．この時求めた近似線が有意かを判断する目安として決定係数を用いる．求めた PCPQ モデルの決定係数が小さすぎる場合，この組織での PCPQ モデルは有意性が低いという判断を支援する．

表 2 PCPQ モデル構築に用いる値

Table 2 Parameter values of a PCPQ model

プロジェクト名	最終的なPC値	障害比重	障害管理表による障害レベル				
			SS	S	A	B	C
A	43	83	4	4	6	9	11
B	28	66	2	1	6	10	13
C	35.4	82	2	3	19	1	1
D	44.4	88	10	6	4	1	0
E	26.3	37	3	3	2	1	2
F	15.4	28	0	0	7	3	1

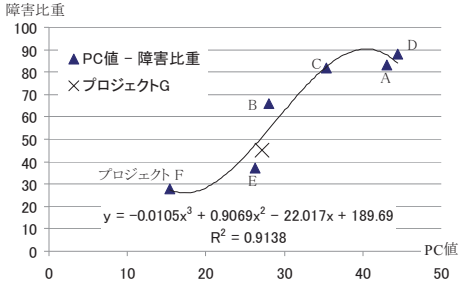


図 4 PC 値と障害比重の近似線

Figure 4 An approximate value curve of relation between process complexities and specific gravities of failures

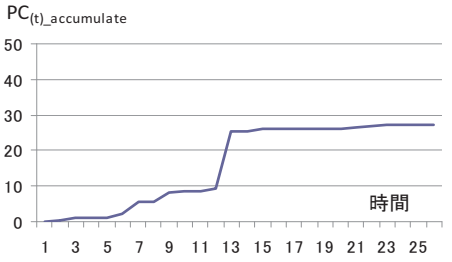


図 5 プロジェクト G の PC 値の変化

Figure 5 Change of process complexity of Project G

今回の組織での PCPQ モデルの決定係数は 0.9138 であり，有意性が高いと判断する．また，近似線の求め方は組織毎に求め方を変更する必要がある．この組織の PCPQ モデル式(2)は多項式が決定係数が高かったが，他組織では組織毎にあった近似線を検討する必要がある．つまり，組織毎で工程管理表や障害管理表の管理粒度の異なりがあるため，線形近似，対数近似，指数近似などの様々な近似線の中で最も優位性の高い近似線の求め方を利用する事で PCPQ モデル(2)を求める必要がある．

4.4 PCPQ モデルを利用したプロダクト品質の予測

図４の PCPQ モデルを用いて，他のプロジェクトのプロダクト品質の予測を行う．プロジェクト G は同一組織内で実施された e-learning システム開発のプロジェクトである．もちろん，プロジェクト G は PCPQ モデル構築で利用した 6 プロジェクトとは異なる独立したプロジェクトである．プロジェクト G の PC 値の推移を図５に示す．13 週目に，PC 値は 25.21 に急増した．新しい作業プロセスが計画に追加されたと考えられる．この値を式(2)の PCPQ モデルの x 値へ適用する．障害比重は 42.78 と計算される．したがって，プロジェクト G のリリース後の障害発生はプロジェクト E より多く，プロジェクト B より少ないことが予測された．

次に，プロジェクト G のリリース後の障害管理表より実際の障害比重を計測した．リリース後の障害件数は 20 件であり，SS と S ランクの障害は 0 件，A ランクが 8 件，B ランクが 9 件，C ランクが 3 件であった．これらから障害比重を計測した結果 45 となった．PCPQ モデルで予測した障害比重は 42.78 であり，近い値となった．図４に×でプロットした箇所が，プロジェクト G の最終 PC 値と障害比重の値の関係の位置である．つまり，プロジェクト G の 13 週目（全プロジェクト期間は 25 週）の時点で，リリース後のプロダクトの品質の予測が出来たことになる．もし，リリース後のプロ

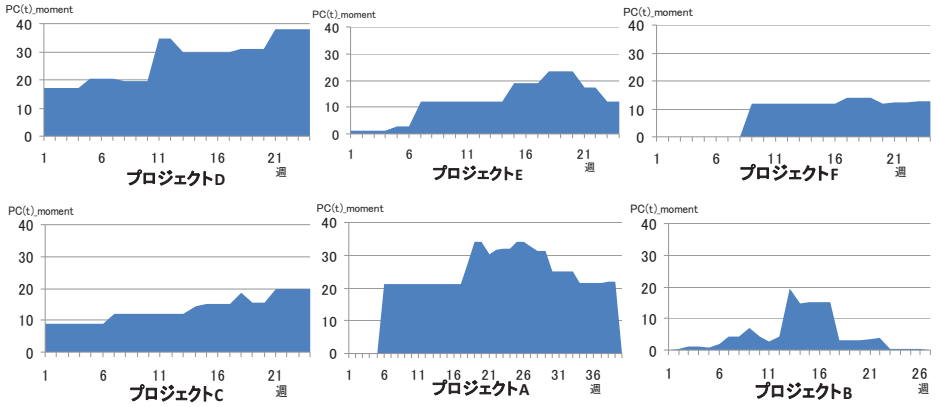


図 6 6プロジェクトの瞬間 PC 値の推移

Figure 6 Changes of values of process complexities (moment) of 6 projects

ダクト品質をプロジェクト E 程度に向上させたいならば, 13 週目に計画された追加作業の見直し等による計画の再設計を実施する等，管理作業の目安となる．

5. 考察

5.1 瞬間 PC 値

PCPQ モデルの PC 値はベースプロセスとフラグメントプロセスのすべての累積から計測した．これより，プロセスの観点からプロダクトの品質を予測し，プロジェクト全体の特徴を示すことができた．しかし，１つのプロジェクト進行中においてもプロセスの複雑さは随時変化する．例えば，前バージョンのシステムに障害が発生し障害対応作業が急遽追加された時期は一時的に PC 値が増加するが，障害対応が収束すると PC 値は元に戻る．このような PC 値の時系列変化を示す瞬間 PC 値を以下に示す．

$$PC_{(t)\_moment} = \sum_{i=1}^{N_{(t\_moment)}} (Num\_dev_{(t)i} \times L_{(t)i} \times term_{(t)i}) \quad \cdots (3)$$

$PC_{(t)\_moment}$: 時刻 t のプロセスの複雑さ（終了プロセスを含まない）

$N_{(t\_moment)}$: 時刻 t の稼働中のプロセス総数（終了プロセスを含まない）

基本的に $PC_{(t)}$ と同じ式であるが，パラメータ $N_{(t\_moment)}$ のプロセスの総数に終了したプロセスを含まない. つまり, 時刻 t に稼働しているプロセスだけを対象に計測する．これにより，時系列に変化する瞬間 PC 値を求めることができる．

瞬間 PC 値を前章の 6 プロジェクトに適用した結果を図 6 に示す．プロジェクト A とプロジェクト D の瞬間 PC 値はプロジェクト期間中において常に高い値を示す．つまり，プロジェクト A とプロジェクト D の開発期間中，多くのプロセスは並行実行されていることが分かる．また，プロジェクト A，B，E はプロジェクト終了に向けて瞬間 PC 値が減少している様子が見受けられるが，プロジェクト C，D は瞬間 PC 値が収束せずに増加傾向のままプロジェクト終了を迎えている事がわかる．通常はプロジェクト終了に向けて多くのプロセスが終了し，瞬間 PC 値が減少傾向を示すにも関わらず，瞬間 PC 値が増加傾向でプロジェクト終了を迎えた．つまり，プロジェクト中に追加作業等が収束せず納期を迎えたことが予想できる．表２のプロジェクト C，D の障害比重の値も他のプロジェクトに比べて大きく，瞬間 PC 値のプロジェクト終了時の減少傾向または増加傾向からも最終プロダクトの品質を予測することも可能と考える．

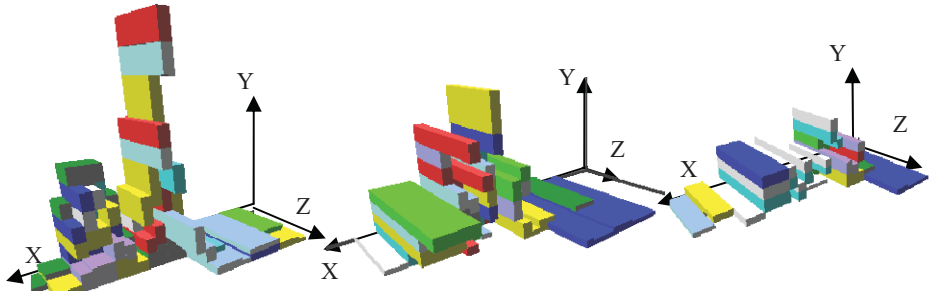
5.2 可視化ツール

瞬間 PC 値の推移を可視化するツールを作成した．図 7 にプロジェクト A，D，E の瞬間 PC 値の可視化結果を示す．可視化ではひとつのフラグメントプロセスをひとつのブロックとして表現する．ひとつのブロックが大きくなれば複雑なプロセスの可能性があり，逆に小さなブロックは複雑でないプロセスを示す．図 7 の X 軸は時刻を示し，Y 軸は式(3)の $L_{(t)}$ の値を示し，Z 軸は式(3)の $Nun\_dev_{(t)}$ を示す．つまり，X 軸方向に長いブロックは実行期間の長いプロセスであり，Y 軸方向に長いブロックは並行実行する他のプロセスが多いことを示し，Z 軸方向に長いブロックは多くの開発グループが関与するプロセスを意味する．また色は同じ作業を示しており，終了した作業が再度追加作業として追加された等も把握可能となっている．

可視化ツールからプロジェクト A の後半では多くのプロセスが発生したことが分かる．しかし，プロジェクト A は X 軸方向には短いブロックが多い．つまり，同時実行するプロセスは多いが，個々のプロセスは比較的すぐに終わる作業であることがわかる．また，プロジェクト A のブロックは Y 軸方向に多く積み重なっており，プロジェクト D,E に比べて並行実行するプロセスが多い．このように，プロジェクトごとのプロセスの複雑さの特徴を表現できるツールである．

5.3 瞬間 PC 値と障害の重要度

瞬間 PC 値と発生した障害の重要度の関係について考察する．プロジェクト A とプロジェクト D は両者とも累積の PC 値が高く，障害比重の値も大きい（表２参照）．似たようなプロジェクトの特徴を持つと考えられるが，図 7 に示すように，プロセスが複雑になった要因のプロセスの種類が異なる．プロジェクト A は細かなプロセスが多く発生したのに対し, プロジェクト D は比較的期間の長い大きなプロセスが発生した．特に，ひとつのプロセスが Z 軸方向に長いのが特徴である．つまり，ひとつのプロセスに複数の開発グループがかかわるプロセスが発生したと考えられる．ともに PC 値が大きいプロジェクトであるが，フラグメントプロセスの特徴が異なる．



プロジェクト A　　プロジェクト D　　プロジェクト E

図 7 瞬間 PC 値の可視化

Figure 7 Visualized images of changing process complexities.

そこで，表 2 の発生した障害の重要度に着目する．プロジェクト A の障害発生件数は多いが，SS クラス（最重要）の障害は 4 件で，B，C クラスの障害が 20 件と重要度の低い障害が多い．反対にプロジェクト D では SS クラスが 10 件で，B，C クラスがわずか 1 件である．すなわち，プロジェクト A では障害件数は多いが軽度な障害が主であり，プロジェクト D では障害件数は多くないが重要な障害が多く発生した．この障害の特徴を図 7 と比較すると，小さなプロセスが重なる場合，軽度な障害の発生が多く，大きなプロセス，特に複数グループが関わるプロセスが発生する場合，重大な障害が発生すると考えられる．このように， PC 値からは全体の障害しか予測できないが，瞬間 PC 値は発生する障害レベルの種類を予測できる可能性があると考える．

6. おわりに

本稿では，プロセスの複雑さとプロダクトの品質の関係を示す PCPQ モデルの構築方法を提案した. PCPQ モデルは複数プロジェクトの最終的な PC 値と障害比重の値を用意し，それらを近似線を求める事でモデルを構築する．PC 値は各組織の工程管理表に記述されているベースプロセスとフラグメントプロセスから計測される．また，障害比重は各組織で用いられている障害管理表の障害件数と重要度レベルから算出される．工程管理表の管理項目の粒度や障害の重要度レベルの決め方は組織ごとに異なるが，PCPQ モデルは組織毎に構築されるため，特定の管理方法に依存しない汎用的なモデル構築方法である．本提案を産業界における同一組織内の 7 つのプロジェクトに適用した．6 つのプロジェクトで PCPQ モデルを構築し，同一組織の別プロジェクトにおいてプロセスの複雑さからリリース後のプロダクトの品質を予測することができた．また，プロジェクト実施中のプロセスの複雑さの時系列変化を示す瞬間 PC 値も提案した．フラグメントプロセスの重なり方や大きさがビジュアルに確認できるツールを作成することで，リリース後に発生する障害の種類を予測できる可能性も示唆できた．今後は PCPQ モデルを更に評価するために，他の多くの組織で PCPQ モデルを構築し，組織ごとのプロセスとプロダクトの関係を調査する予定である．

謝辞　本研究の一部は文部科学省科学研究費基盤研究(C)22500027，および文部科学省科学研究費基盤研究(C)21500045 の補助を受けた．

参考文献

1) Walker Royce.: Software Project Management: A unified Framework, Addison-Wesley Professional, USA(1998).
2) 海谷治彦，北澤直幸，長田晃，海尻賢二 ：類似既存システムの情報を利用した要求獲得支援システムの開発と評価，電子情報通信学会論文誌, Vol.J93-D, No.10, pp.1836-1850(2010).
3) 中谷多哉子，藤野晃延 ：ロールに着目したビジネス領域における要求獲得手法 RODAN の提案，情報処理学会論文誌, Vol.48, No.8, pp.2534-2550(2007).
4) Guide to the Software Engineering Body of Knowledge (SWEBOK) , http://www.computer.org/portal/web/swebok
5) Humphrey Watts, S.: Managing the software process. Addison-Wesley Professional, USA(1989).
6) 坂本啓司，田中敏文，楠本真二，松本健一，菊野亨 ：利益予測に基づくソフトウェアプロセス改善の試み ，電子情報通信学会論文誌，Vol.J83-D1 ， No.7 ， pp.740-748(2000).
7) Cugola, G.: Tolerating Deviations in Process Support Systems via Flexible Enactment of Process Models , IEEE Transaction of Software Engineering, Vol.24, No.11, pp.982-1001(1998).
8) Fuggetta, A. and Ghezzi, C.: State of the art and open issues in process-centered software engineering environments, Journal of Systems and Software, Vol.26, No.1 pp.53-60(1994).
9) Garcia, F., Ruiz, F., Piattini, M.: Definition and empirical validation of metrics for software process models. Proceedings of the 5th International Conference Product Focused Software Process Improvement, pp.146-158(2004).
10) Kruchten, R.: The Rational Unified Process, Addison-Wesley Professional, USA(2000).
11) Manzoni, L.V., Price, R.T.: Identifying extensions required by RUP (rational unified process) to comply with CMM (capability maturity model) levels 2 and 3, IEEE Transactions of Software Engineering , Vol.29 , No.2, pp.181-192(2003).
12) 青木 俊樹, 山田 茂 :プロセスデータに基づくソフトウェア開発プロジェクトの品質指向型定量的評価法に関する考察 (不確実な状況における意思決定の理論と応用) 数理解析研究所講究録 ,1589 ,pp215-221(2008)
13) Obana, M. Hanakawa, N. Yoshida N. Iida, H.: Process Fragment Based Process Complexity with Workflow Management Tables. International Workshop on Empirical Software Engineering in Practice, pp.7-12(2010).
14) Bill,Curtis. Girish, V, Seshagiri. Donald, Reifer. Iraj, Hirmanpour and Gargi, Keeni.: The Case for Quantitative Process Management, IEEE Software, vol.25 , No.3, pp.24-28(2008).