レビュー重視と品質・生産性の関係分析

- 品質と生産性の向上は両立するか -

Relational analysis among Review improvement, Quality and Productivity

- Is it possible to balance Quality and Productivity? -

日本電気株式会社　ソフトウェアエンジニアリング本部
NEC Corporation, Software Engineering Division
○丸山　志保　　柳田　礼子 1)
○Shiho Maruyama　　Reiko Yanagida1)

Abstract　Improvement of quality and productivity is an eternal theme in software development. In our company, software development data is collected and accumulated variously in organization crossing way. By analyzing factors influencing quality and productivity, it's reflected to companywide improvement activities toward improving quality and productivity. A relation between a review, quality and productivity was analyzed using 527 cases of project data which has ended in fiscal year 2015. This paper clarified the way to make improvement of quality and productivity compatible, based on the actual result of data analysis.

1.　はじめに

ソフトウェア開発において、品質と生産性の向上は永遠の課題である。

弊社では、ソフトウェア開発を行っている組織からプロジェクトデータを収集・蓄積し、組織横断的にさまざまなデータ分析を行い、品質や生産性への影響要因を分析することにより、品質・生産性の向上へ向けた全社的な改善活動に結びつけてきている。

本論文では、2015 年度に終了した 527 件のプロジェクトデータを使用し、レビューと品質および生産性の関係を分析することにより、ソフトウェア開発での品質と生産性の関係を明らかにする。さらに、品質・生産性向上を両立させる方法について報告する。

本論文の構成は、2 章で前提とする品質管理技法と開発プロセス、品質と生産性の判定方法、分析対象データと項目を説明し、3 章で弊社の過去の分析結果の一例紹介と今回の分析の経緯を説明する。4 章でレビューと品質および生産性の関係を分析し、そこから得た結果を 5 章で報告する。最後に今後の課題を 6 章で述べる。

2.　分析の準備

2.1　品質会計と開発プロセス

はじめに、データ分析にあたり前提とする品質管理技法と開発プロセスについて説明する。弊社では、ソフトウェア開発の標準的な品質管理手法として品質会計を適用している[1]。品質会計の前提とする開発プロセスを図 1 に示す。開発プロセスはV字モデルである。

基本設計からコーディングまでを「上工程」といい、単体テストからシステムテストまでを「テスト工程」という。

日本電気株式会社　ソフトウェアエンジニアリング本部
Software Engineering Division, NEC Corporation

東京都港区芝 4-14-1　　Tel: 03-3798-9552　e-mail:s-maruyama@cp.jp.nec.com
4-14-1, Shiba, Minato-ku, Tokyo Japan

1) 日本電気株式会社　ソフトウェアエンジニアリング本部　マネージャー
Manager, Software Engineering Division, NEC Corporation

【キーワード：】品質会計、レビュー工数、生産性、出荷後品質

品質会計は、その開発で作り込まれるであろうバグ数を予測し、それを開発の早期に摘出するとともに、全件を摘出したとき出荷するという考え方に基づく品質管理手法であり、「レビューでの早期バグ摘出」と「的確なテスト完了判断」を特徴としている。

弊社社内では、上工程レビューで全体の80%のバグ摘出を目標とすることを推奨している。この上工程レビューで摘出したバグの割合を上工程バグ摘出率と呼び、以下の式により算出する。

上工程バグ摘出率
＝上工程レビューで摘出したバグ数
／出荷前に摘出した総摘出バグ数
×100（%）

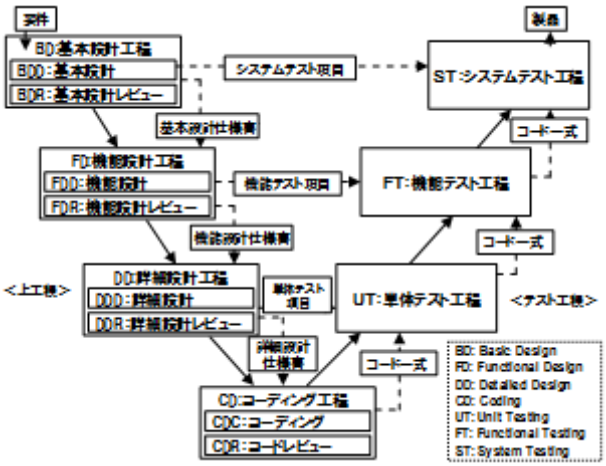

図１　開発プロセス

2.2　品質の判定方法

各プロジェクトが開発したソフトウェアの品質の良し悪しは、出荷後バグ基準値に対する出荷後バグ数で判定する。

出荷後バグ基準値とは出荷後の品質目標値であり事業領域毎に設定する。出荷後バグ基準値は以下の式により算出する。

出荷後バグ基準値＝出荷後 12 か月以内の顧客摘出バグ数(件)／開発規模(KLOC)

出荷後バグ実績値も同様の式で算出する。

出荷後バグ基準値の「達成」「未達」は、以下のように判定し、出荷後バグ基準値を「達成」したプロジェクトを「品質が良い」プロジェクトとする。

達成：出荷後バグ実績値≦出荷後バグ基準値
未達：出荷後バグ実績値＞出荷後バグ基準値

また、出荷後バグ基準達成率は出荷後バグ基準値を達成したプロジェクトの比率であり、以下の式により算出する。

出荷後バグ基準達成率
＝当該カテゴリの達成プロジェクト数／当該カテゴリの全プロジェクト数×100（%）

2.3　生産性の判定方法

生産性は開発規模および工数を使用し、以下の式で算出する。値が大きい方が「生産性が良い」と判定する。

生産性＝開発規模／工数（Line/人 H）

2.4　分析対象データと分類

開発の特性から対象プロジェクトデータを表 1 に示すとおりに分類して分析する。また、本分析に使用したデータ項目は表 2 に示すとおりである。

表１　開発特性によるプロジェクトデータの分類

分類	開発特性
汎用ＳW系 プロジェクト	特定の顧客に特化しない汎用ソフトウェア製品の開発を行うプロジェクト
ＳＩ系 プロジェクト	特定の顧客向けに開発を行うプロジェクト

表２　データ項目

No.	データ項目	単位	定義
1	開発規模	KLOC	開発規模（KLOC）＝新規規模（KLOC）＋修正規模（KLOC）
2	生産性	Line/人H	生産性（Line/人H）＝開発規模（Line）／総工数（人H）
3	全摘出バグ数/KLOC	件/KLOC	出荷前に摘出した全摘出バグ数（件）／開発規模（KLOC）． 全摘出バグ数/KLOC ＝上工程摘出バグ数/KLOC（No.4）＋テスト工程摘出バグ数/KLOC（No.5）
4	上工程摘出バグ数/KLOC	件/KLOC	全摘出バグ数/KLOC（No.3）のうち、上工程(設計レビュー,コードレビュー)で摘出されたバグ数/KLOC
5	テスト工程摘出バグ数/KLOC	件/KLOC	全摘出バグ数/KLOC（No.3）のうち、テスト工程(テスト作業,テスト仕様書レビュー)で摘出されたバグ数/KLOC
6	上工程バグ摘出率	%	上工程バグ摘出率 ＝上工程摘出バグ数/KLOC（No.4）／全摘出バグ数/KLOC（No.3）×100
7	テスト項目数/KLOC	項目/KLOC	テスト項目数（項目）/開発規模（KLOC）
8	全工数/KLOC	人H/KLOC	開発に費やした全工数（人H）/開発規模（KLOC）． 全工数/KLOC ＝ 上工程工数/KLOC（No.9)＋テスト工程工数/KLOC（No.10）
9	上工程工数/KLOC	人H/KLOC	上工程(基本設計～コーディング)に費やした工数（人H）/開発規模（KLOC）
10	テスト工程工数/KLOC	人H/KLOC	テスト工程(単体テスト～システムテスト)に費やした工数（人H）/開発規模（KLOC）
11	上工程レビュー工数/KLOC	人H/KLOC	設計およびコードに対するレビューに費やした工数（人H）/開発規模（KLOC）． 上工程レビュー工数は、上工程工数の内数である．

3.　分析の経緯

3.1　過去の分析結果[2]

図２は、過去の分析結果の一例である。弊社で分析に使用している各データ項目について、出荷後バグ基準値を「達成」したプロジェクト(以降、達成 PJ と呼ぶ)と「未達」のプロジェクト(以降、未達 PJ と呼ぶ)の中央値を、全体の中央値を１としたときの相対値で比較したものである。達成 PJ と未達 PJ を Wilcoxon 検定で検定し、５％水準で有意差のあるデータ項目(黄色網掛けのデータ項目)に注目して分析する。

達成PJと未達PJの中央値（SI系）
（全体の中央値を1としたときの相対値）

	単位	達成	未達
開発規模	Line	0.81	2.67
開発期間	日	0.94	1.24
生産性	Line/人H	0.91	1.25
全摘出バグ	件/KL	0.86	1.15
上工程バグ	件/KL	0.90	1.18
テスト工程バグ	件/KL	0.70	1.28
上工程バグ摘出率	%	1.03	0.98
レビュー工数	人H/KL	1.14	1.04
テスト項目数	件/KL	1.01	1.03
全工数	人H/KL	1.10	0.80
上工程工数	人H/KL	1.10	0.91
テスト工程工数	人H/KL	1.18	0.78
ドキュメント量	頁/KL	1.05	1.10

達成PJと未達PJの中央値（汎用SW系）
（全体の中央値を1としたときの相対値）

	単位	達成	未達
開発規模	Line	1.01	1.84
開発期間	日	0.98	1.11
生産性	Line/人H	0.99	1.10
全摘出バグ	件/KL	0.99	1.28
上工程バグ	件/KL	0.95	1.33
テスト工程バグ	件/KL	0.92	1.18
上工程バグ摘出率	%	1.00	1.03
レビュー工数	人H/KL	1.05	1.16
テスト項目数	件/KL	0.87	1.16
全工数	人H/KL	1.02	0.92
上工程工数	人H/KL	0.99	1.05
テスト工程工数	人H/KL	1.06	0.91
ドキュメント量	頁/KL	0.98	1.11

※表の黄色は、Wilcoxon検定で5%水準で有意差あり

図２　出荷後バグ基準値 達成 PJ と未達 PJ の中央値の比較
[2]より引用(一部表記変更)

ＳＩ系プロジェクト(以降、ＳＩ系と呼ぶ)、汎用ＳＷ系プロジェクト(以降、汎用ＳＷ系と呼ぶ)共に、達成 PJ より未達 PJ の方が、全摘出バグ数、上工程バグ数、テスト工程バグ数が多いことがわかる。

また、ＳＩ系の達成 PJ と未達 PJ を比較すると、未達 PJ はテスト工程工数が少ないが、テスト工程バグ数が多く、結果として上工程バグ摘出率が低い。

さらに工程毎の分析により出荷後品質に影響を与えるキーファクターを探索したところ、ＳＩ系は、基本設計工程と機能設計工程で摘出バグ数が多く、かつ、レビュー工数が少ない場合に、最も出荷後バグ基準達成率が低いという結果を得た[2]。

以上をまとめると、出荷後バグ基準が未達となるプロジェクトの特徴は以下の通りである。
・出荷前に摘出したバグ数が多い。
・基本設計工程と機能設計工程で、レビュー工数が少なく摘出バグ数が多い。
上記より、出荷後品質向上のための方向性として、以下が考えられる。
・設計内容そのものを改善し、出荷前に摘出されるバグ数を減らす。
・基本設計工程と機能設計工程でレビューに時間をかけてバグを摘出し、テスト工程へバグを持ち越さないようにする。(上工程バグ摘出率の向上と同義)

3.2　今回の分析のねらい

過去の分析結果から、設計内容そのものの改善は大きな課題として取り組むとともに、現在のしくみの中ですぐに取り組める内容としては、設計工程(上工程)のレビューでより多くのバグを摘出することによりテスト工程へバグを持ち越さない、すなわち上工程バグ摘出率の向上である。今回は、上工程バグ摘出率に対する品質・生産性の関係を分析する。

4.　今回の分析

対象データ 527 件（汎用ＳW系：117 件、ＳＩ系：410 件）を使用して分析した結果を説明する。品質会計で上工程バグ摘出率 80%を推奨していることと対象データの分布から、上工程バグ摘出率を「70%未満」「70%以上 80%未満」「80%以上」の３つの層に分けて比較分析をする。各グラフは、「70%未満」の層の値を 100 とし、各層の値を相対値で表したものである。

4.1　出荷後バグ基準達成率の比較

図 3 は、対象データ 527 件を上工程バグ摘出率の 3 層別に出荷後バグ基準達成率を算出し、上工程バグ摘出率 70%未満の層の達成率を 100 として比較したグラフである。100 より大きい場合、上工程バグ摘出率 70%未満の層よりも出荷後バグ基準達成率が高く、「品質が良い」ことを示す。

上工程バグ摘出率 70%～80%の層は 100 より大きく、80%以上の層はさらに大きい。これは、上工程バグ摘出率が高い層ほど、出荷後バグ基準達成率が高くなることを示している。すなわち、上工程でより多くバグを摘出するほど「品質が良い」といえる。

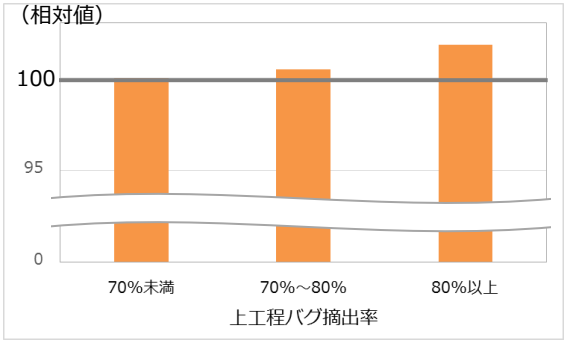

図 3　出荷後バグ基準達成率

4.2　工数の比較

次に、開発の特性毎に工数の比較分析結果を説明する。

図 4 は、汎用ＳW系のプロジェクトデータ 117 件を、上工程バグ摘出率の 3 層別に、上工程レビュー工数を除いた上工程工数、上工程レビュー工数、テスト工程工数の中央値を積み上げ、上工程バグ摘出率 70%未満の層の値を 100 として比較したグラフである。100 より小さい場合、工数が少なく、「生産性が良い」ことを示す。

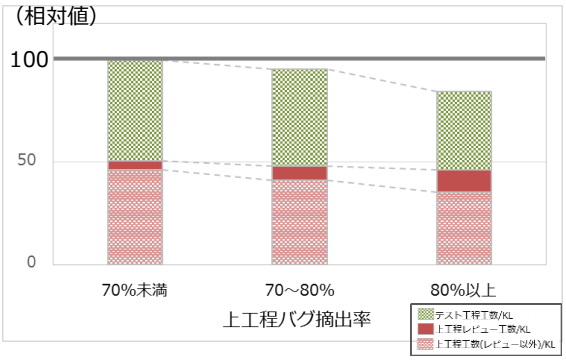

図 4　単位規模あたりの全工数〔汎用ＳW系〕

図 4 では、全工数は上工程バグ摘出率が上がるにつれて減少している。すなわち、上工程バグ摘出率が上がるにつれて生産性が良くなっていることを示す。その理由は、工数の種類別でみると上工程バグ摘出率が高い層ほど上工程レビュー工数が増加しているが、上工程レビュー工数を除いた上工程工数およびテスト工程工数が減少しているためである。

これは上工程でレビューに時間をかけて早期にバグを摘出することにより、上工程内で次工程に持ち越すバグおよびテスト工程に持ち越すバグが抑えられることから、バグ修正にかかる後戻りが減ったためと考えられる。

一方、図 5 はＳＩ系のプロジェクトデータ 410 件を、上工程バグ摘出率の 3 層別に、上工程レビュー工数を除いた上工程工数、上工程レビュー工数、テスト工程工数の中央値を積み上げ、上工程バグ摘出率 70%未満の層の値を 100 として比較したグラフである。

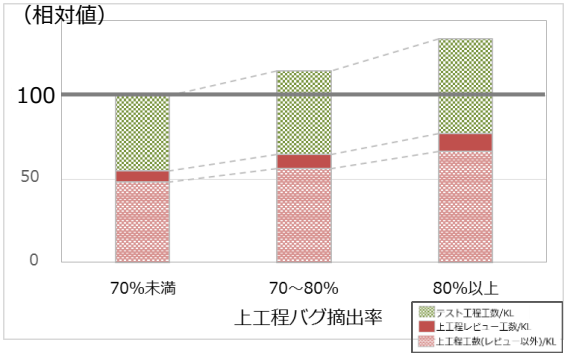

図 5　単位規模あたりの全工数〔ＳＩ系〕

上工程バグ摘出率が高くなるほど上工程レビュー工数が増加しているが、上工程レビュー工数を除いた上工程工数もテスト工程工数も増加しており、上工程バグ摘出率が高くなるほど生産性が良くなるとはいえない。

4.3　ＳＩ系プロジェクト特有の要因の考察

上工程でレビューに時間をかけてバグを摘出しているにも関わらず、汎用ＳW系プロジェクトとＳＩ系プロジェクトとでは生産性への表れ方が異なっていた。これは、ＳＩ系特有の要因による影響が考えられる。

ＳＩ系特有の要因の１つとして直接ソフトウェアを開発する以外の作業項目の工数があげられる。ＳＩ系には、移行・展開作業やユーザ支援、仕様調整等、顧客関連の対応作業が数多く存在し、プロジェクト毎に発生状況が異なる。一方、汎用ＳW系はＳＩ系で必要な顧客対応工数は不要であり、ソフトウェアを開発するための工数のみ必要である。

図 6 は、弊社で収集している工数の構造と定義を示したものである。工程ごとに設計やテスト、レビュー工数の他に、「管理工数」「顧客対応工数」を分離して計上する。

総工数					
上工程工数（BD、FD、DD、CD）			**テスト工程工数**（UT、FT、ST）		
	設計製造工数	設計およびコーディングに費やした工数 バグ対応工数を含む		テスト工数	テスト作業工数 バグ対応工数を含む
	上工程 レビュー工数	設計およびコードのレビュー工数 レビュー準備時間を含む		テスト工程 レビュー工数	テスト仕様書のレビュー工数 バグ修正に伴う設計書レビュー工数を含む
	管理工数	各管理作業工数（進捗管理、品質管理、費用管理、要員管理、リスク管理、外注管理等）		管理工数	各管理作業工数（進捗管理、品質管理、費用管理、要員管理、リスク管理、外注管理等）
	顧客対応工数	顧客関連対応工数（環境維持、移行・展開、ユーザ支援、技術支援、教育訓練等）		顧客対応工数	顧客関連対応工数（環境維持、移行・展開、ユーザ支援、技術支援、教育訓練等）

図 6　工数の構造と定義

汎用ＳW系の「管理工数」「顧客対応工数」データを確認したところ、どちらもほとんどデータが入力されていないことがわかった。

ＳＩ系の工数計上状況をヒアリングしたところ、「顧客対応工数」を適切に分離して計上しているプロジェクトは多くなかった。本来「顧客対応工数」に計上されるべき工数が他の工数に混在していると考えられる。

また、「管理工数」は開発のために必要な工数であり、生産性を測る工数に含めるべきであると考えるが、汎用ＳW系もＳＩ系も適切に分離せずに計上しているプロジェクトが多いことがわかった。

以上の課題を解決するため、以降においては、「管理工数」と「顧客対応工数」を除いた工数を使用して分析する。そのために、開発工数および開発生産性という用語を新たに定義して使用する。

・開発工数：全工数（表 2、8 項目目と対応）から、管理工数と顧客対応工数を除いた工数をいう。具体的には、図 6 で使用する用語により、以下の式で算出される。

開発工数＝設計製造工数＋上工程レビュー工数＋テスト工数＋テスト工程レビュー工数

併せて、テスト工程工数も以下のように再定義する。

- テスト工程工数：テスト工程工数（表 2、10 項目目と対応）から、管理工数と顧客対応工数を除いた工数をいう。具体的には、図 6 で使用する用語により、以下の式で算出される。

テスト工程工数＝テスト工数＋テスト工程レビュー工数

・開発生産性：以下の式により算出する。

開発生産性＝開発規模／開発工数（Line/人 H）

4.4　ＳＩ系プロジェクト精査済みデータによる分析

ＳＩ系特有の要因を考慮して分析をするため、タスクフォースをたちあげ、タスクフォース参画メンバーに所属組織のデータについてプロジェクトの遂行状況や工数データの分離状況を確認してもらった。さらにデータ精度を高めるためにデータ欠測や外れ値の除外等の精査を実施した。

また、ＳＩ系は請け負う範囲により工程に偏りが生じる場合もあるため、機能設計工程、詳細設計工程、コーディング工程、単体テスト工程、機能テスト工程の 5 工程の工数データに欠測がないプロジェクトデータのみを分析に使用することにした。

図 7 は、ＳＩ系の精査済みデータのうち、出荷後バグ基準値を達成しているデータ 28 件を、上工程バグ摘出率の 3 層別に、設計製造工数、上工程レビュー工数、テスト工程工数の中央値を積み上げ、上工程バグ摘出率 70%未満の層の値を 100 として比較したグラフである。

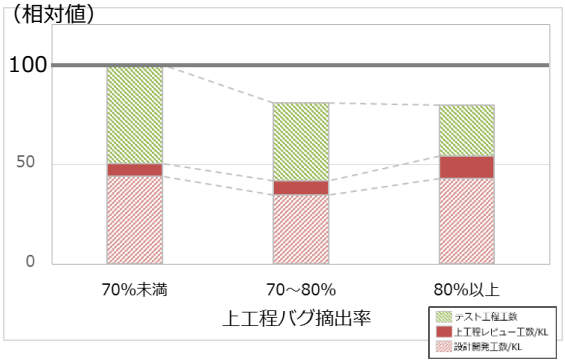

図 7　単位規模あたりの開発工数
〔ＳＩ系：精査済〕

図 7 では、開発工数は上工程バグ摘出率が高い層ほど減少している。すなわち、上工程バグ摘出率が上がるにつれて開発生産性が良くなっていることを示す。その理由を工数の種類別にみると、上工程バグ摘出率が高い層ほど上工程レビュー工数は増加しているが、設計製造工数は上工程バグ摘出率 80%以上の層で 70～80%の層より若干増加しているものの、70%未満の層よりは減少し、テスト工程工数は上工程バグ摘出率が高い層ほど減少しているためである。

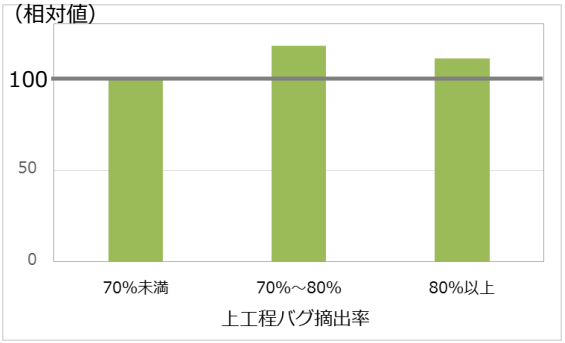

図 8　単位規模あたりのテスト項目数

図 8 は上工程バグ摘出率の 3 層別に、テスト項目数の中央値を、図 9 はテスト工程バグ数の中央値を、それぞれ上工程バグ摘出率 70%未満の層の値を 100 として比較したグラフである。

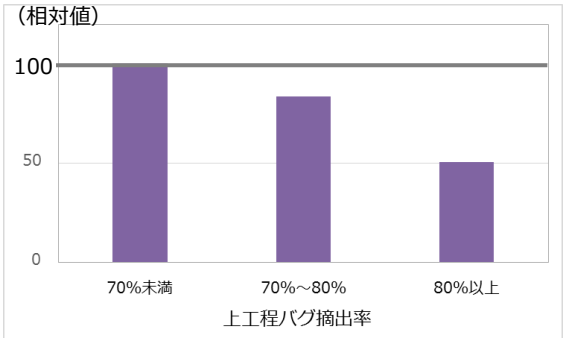

図 9　単位規模あたりのテスト工程バグ数

各層でほぼ同程度のテスト項目数でテストを実施(図 8)しているにもかかわらず、テスト工程で摘出されるバグ数は上工程バグ摘出率が高い層ほど減少(図 9)している。

これは上工程のレビューでバグを摘出することにより、テスト開始時点に潜在するバグ数が減少したためである。結果として同程度のテスト項目

数を実施しても摘出されるバグ数が減少するため、バグ修正にかかる後戻りが少なくなることによりテスト工程工数が減少し、全体の工数が減少した、すなわち開発生産性が高くなったと考えられる。

5. まとめ

品質および生産性向上の鍵を見つけるため、出荷後の品質状況、上工程のレビューにかけた工数、上工程の工数、テスト工程の工数、各工程の摘出バグ数等から分析を行った。上工程バグ摘出率を３つの層に分けて工数のかけ方を分析したところ、汎用ＳＷ系は、上工程でより多くのバグを摘出している層ほど、上工程のレビューに工数をかけているにもかかわらず、上工程工数、テスト工程工数とも減少し、結果として全体の工数が減少していることがわかった。ＳＩ系は、プロジェクトの特性と影響要因を考慮して工数の構造に着目し、開発作業の工数に限定して分析をしたところ、汎用ＳＷ系同様、上工程のレビューに工数をかけ、より多くのバグを摘出することにより、設計開発工数、テスト工程工数とも減少し、全体の工数が減少していることがわかった。すなわち、開発特性に関係なく、上工程のレビューにより多くの時間をかけ、バグを摘出することにより、品質が向上するとともに、生産性が良くなるといえる。

以上より、上工程のレビューにより多くの時間をかけてバグを摘出することが、品質と生産性向上の両立への鍵であるといえる。

6. おわりに

本論文では、上工程バグ摘出率と品質・生産性の関係を分析することにより、開発の特性に関わらず、上工程レビューでより多くの時間をかけてバグを摘出し上工程バグ摘出率を向上すると、品質が向上するとともに生産性が良くなることを実証した。

今回は精査済みデータの件数が少なかったので、今後はさらなる検証を加えたい。また、現場の個々の条件に合わせたレビュー強化方法等が必要であり、その検討や提案を行うことにより、品質・生産性向上へ貢献する所存である。

参考文献

[1] 誉田直美、「ソフトウェア品質会計」日科技連出版社、2010
[2] 柳田礼子、SQiP2016 品質リスクの早期検出に向けた分析、11,13 ページ、2016