INFORMATION
SCIENCE
TECHNICAL
REPORT

NAIST-IS-TR2008003
ISSN 0919-9527

ソフトウェア品質への影響計測のためのテスト工程での手戻り作業量の特徴分析

松村知子，松本健一

February 2008

NAIST

〒 630-0192
奈良県生駒市高山町 8916-5
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
8916-5 Takayama, Ikoma, Nara 630-0192, Japan

ソフトウェア品質への影響計測のための テスト工程での手戻り作業量の特徴分析

松村　知子†　　松本　健一†

†奈良先端科学技術大学院大学情報科学研究科

〒630-0192 奈良県生駒市高山町 8916-5

E-mail:　†{tomoko-m, matumoto}@is.naist.jp

1．はじめに

ソフトウェア開発における手戻り作業は，開発コストの増大・納期遅延の原因となるばかりでなく，プログラムコードの品質を低下させる危険がある．プログラムコードの不用意な変更は，デグレード等 2 次不具合混入の危険がある．一方，2 次不具合混入や回帰試験を減らすため，下流工程でのプログラム変更は，必ずしも理想的な形で完全に行わず，対症療法を行うことも多い．そのため，ソフトウェアが複雑化し，保守性や可読性などが悪化することで，さらに 2 次不具合混入の危険が増大し，手戻り作業を困難にしていく場合が多い．

一方，上流工程での手戻りは，コストが小さく，プログラムコードなどの最終成果物の品質への影響も小さいことが，これまでの研究で知られている[1]．そのため，上流工程でのレビューやインスペクションなどによる品質保証活動が推奨されるが，人的要素による影響が大きい作業であるため，その効率化やプロセスの改善に有効な汎用的な手法はなかなか提案されない．

本稿では，プログラムの品質低下の一因である下流工程での手戻り作業の量とその手戻り要求の特徴との関係を分析し，結果について報告する．下流工程での手戻り作業の量に影響する要因を抽出することによって，上流工程での品質確保のための対策のポイントを絞ることができる，と考えている．対策のポイントを絞ることによって，効率的・効果的な手戻り作業の低減を図り，品質劣化の防止が可能になる．

本稿で取り扱う手戻りの作業量は，プログラムコードの変更に関する量で，構成管理データから取り出す．手戻りの影響については，修正工数を用いた分析が一般的に良く行われる．これは，よりコストや納期に影響する要因の抽出に有効であるが，一方，品質への影響は変更量や変更範囲によって左右されると考えられる．Royce ら[3]は，下流工程での手戻り量(Rework)を，エラー要因と改善とに分類し，プログラムコードへのダメージ量と修復量を分けて計測することで，プログラムの Modularity，Changeability，Maintainability を評価する手法を提案している．

以降，2 章で分析手法について説明し，3 章で分析の適用事例について紹介する．4 章では，適用した結果得られた知見や問題点について説明し，5 章で今後の予定について記述する．

2．分析手法

2．1　分析対象データ収集

本分析に必要なデータは，構成管理システムおよび障害管理システムから取得する．構成管理システムからは，手戻り作業量が取得できる．構成管理システムは，プログラムコードのバージ

ョン管理のために，一般的にソフトウェア開発プロジェクトで用いられる CASE ツールである．代表的なツール CVS[1]は，ファイル単位でプログラムコードの更新履歴を管理すると共に，バージョン間の差分を提示したり，追加・削除行数を計測したりする機能を持つ．障害管理システムは，テスト工程で発見された不具合の管理に用いられる．障害管理ツール GNATS は,各障害の現象や原因，修正方法などの詳細な情報だけでなく，担当者や対応状況を追跡したり，メールで関係者に不具合の発生や対応状況を通知したりする機能を持つ．

近年の統合プロジェクト管理ツール(ClearCase[2]や Trac[3]など)では,構成管理と障害管理を関連付けて行う機能を持ち,容易にプログラムコードの変更とその要因を関連付けたデータを取り出すことができる．しかし，それぞれ独立したツールでも，障害管理ツール上の管理番号を，構成管理ツールへのファイル更新時にコメントに付記することによって,後から関連付けることができる．開発者は，通常の障害管理ツールと構成管理ツールの運用に関して，障害番号の入力と障害単位でのファイル更新という運用ルールを遵守することだけで,上記の分析データを入手することが可能になる．

2．2　手戻り作業量データ

今回用いた手戻り作業量データは，表 1 の示す 6 種類である．表 1 は，各障害対応データの名称，英語名称，略号，データの説明と計算式，各データが品質(劣化)に関して推定される影響を記述する．

2．3　手戻り要求データ

手戻り要求データは，実際にはテスト工程で発見される障害データを用いる．障害データの管理システムには，その目的にしたがって障害の現象や原因，修正方法などの情報が入力される．表 2 に本稿で分析対象としたプロジェクトで収集し，分析で用いる障害属性項目，各項目のとりうる値(選択肢)，項目説明を示す．実際には，障害管理データとして障害の詳細，再現方法，修正方法，確認などの自由記述データも含むが，特に分析に用いないため，ここでは割愛する．

手戻り要求データとしては，要件や設計変更もあるが，対象プロジェクトではこれらの変更要求は別管理で,収集できる属性項目も異なり,また数も少ないことから,分析対象としなかった．しかし，仕様・設計変更は，より品質への影響が大きい，と思われることから，本来対象とするべきと考えられる．また，プログラムコードの整理，可読性向上のための改善，リファクタリングなどの作業も下流工程で行われることがあるが,これらは品質の低下につながらない.従って,本稿では，テスト工程で発見される障害のみを手戻り要求として扱う．

以降,「発見工程」「重要度」などの障害属性を「障害項目」，各項目のとりうる値(選択肢)を「カテゴリ」と呼ぶ．例えば，障害項目「重要度」については「重大」「中度」「軽微」の 3 つがカテゴリとなる.表中の取り消し線つきのカテゴリは，類似のカテゴリを直下の下線付きのカテゴリに集約している．また，障害原因の取り消しカテゴリは,「その他」に集約した．

[1] Concurrent Versions System : http://ximbiot.com/cvs/cvshome/
[2] Rational Clear Case : http://www-06.ibm.com/jp/software/rational/products/scm/cc/
[3] Trac Project : http://trac.edgewall.org/

表 **1**　手戻り作業量データ一覧

作業量データ名	英語名	略号	説明	品質(劣化)に関して推定される影響
更新ファイル数	Number of updated files	NUF	障害を修正するために変更されたファイル数.	1 障害に対して変更されるファイルが多数にわたる場合，その障害が複雑，または仕様変更などを伴う大規模な変更と考えられる．このようなケースでは，プログラムの構造的な複雑度が増加し，保守性の低下が予測されると共に，修正ミスや漏れによる２次不具合も発生しやすい.
追加行数	Number of added lines	NAL	障害を修正するために追加された行数．実質は，新規で追加された行数＋変更行数．[CVS の log コマンドで表示される更新履歴に示される追加行数]	追加行数は，新たな機能や処理の追加(再コーディング)であり，当然追加箇所に不具合が混入する可能性が高い．また，コーディング中に作成されたプログラムコードとの形式や規約上の不整合が発生する可能性も高く，可読性の低下が予想される.
削除行数	Number of deleted lines	NDL	障害を修正するために削除された行数．実質は，削除された行数＋変更行数．[CVS の log コマンドで表示される更新履歴に示される削除行数]	削除行数は，品質への影響は少ない,と考えられる．ただし,誤削除の可能性はある．また,不要コードの削除に折る影響で 2 次不具合を発生させることも多い.
変更行数	Number of changed lines	NCL	障害を修正するために追加された行数＋削除された行数. NCL=NAL+NDL	1 つの手戻り要求に対する総作業量，と考えられる．品質への影響は，追加行数・削除行数に記述したとおりである．ただし，追加行と削除行が重複する，つまり行内の一部を変更する場合，その行を重複して計上してしまうため，実際の作業量より大きい値になる場合がある.
更新後行数	Number of total lines	NTL	更新したファイルの更新後の総行数	更新ファイル数と同じく，変更の複雑さや規模を示す．ファイルの分割方法（クラス単位や関数単位など）によってファイルサイズが変わるため，この値はその修正が影響する可能性があるシステムの範囲をより正確に示すと考えられる.
追加行率	Rate of added lines	RAL	変更が及ぶファイルの，更新前のサイズに対する追加行数の割合. 以下の式で表される. RAL= NAL÷（NTL-NAL+NDL）	影響範囲に対する追加規模を表す．大きい場合，大規模な機能追加などで，新たな不具合の混入やプログラム構造の複雑化が考えられるが，一方，一箇所に集中した追加の場合，影響範囲が小さくミスが少ない可能性もある．小さい場合，小規模で危険が少ないこともあるが，変更範囲が大きく，修正漏れや予期しない影響が発生することもありうる.
削除行率	Rate of deleted lines	RDL	追加行率と同様，変更が及ぶファイルの，更新前のサイズに対する削除行数の割合. 以下の式で表される. RDL= NDL÷（NTL-NAL+NDL）	影響範囲に対する削除規模を表す．追加行率と同じような影響が考えられるが，危険度は追加行率より低いと考えられる.

表 2　障害属性データ項目一覧

項目名	選択項目(カテゴリ)	項目説明
起票分類	- 仕様不具合 - コード不具合 - 仕様変更 - 不具合からの仕様変更	起票時の問題の種類
発見箇所	- 未入力 - 新規 - 改造(他システムからの流用) - 改造(システム内流用) - 再利用(未改造)	障害を発見した部分などの流用状況
発見工程	- 未入力 - 要件定義 - 基本設計 - 詳細設計 - プログラム設計 - コーディング(製造) - 単体試験 - 結合試験1(コンポーネント内) ~~- 結合試験2(コンポーネント間)~~ ~~- 総合試験~~ ー社間結合試験 - 実証実験(運用)	障害を発見した工程
発見作業	- 未入力 - 分析・解析中 - システム動作中(テスト･運用) - レビュー	障害を発見した作業
試験項目種別	- 未入力 - 正常系 - 異常系 - 非機能 - その他	障害を発生した試験項目の分類
障害処理機能	- 未入力 - 入力データチェック機能 - 演算機能 - データ編集機能 - ファイル更新機能 - データ出力機能 - 連動(組み合わせ)処理 - 限界処理 - 外囲条件異常検知機能 - その他	障害を発見したシステムの機能
ソフトウェアエラー分類	- 未入力 - インタフェースエラー - 論理エラー - データ定義エラー - テーブル定義エラー - 形式エラー	ソフトウェア上のエラー(誤り)の種別

	- その他	
重要度	- 未入力 - 重大 - 中度 - 軽微	障害の重要度
優先度	- 未入力 - 高 - 中 - 低	障害解決優先度
再現度	- 未入力 - 再現頻度大 - 再現度小 - 再現なし - 不明(未確認)	故障発生の再現頻度
障害原因	- 未入力 ~~- 仕様書記述漏れ~~ ~~- 仕様書記述誤り~~ ~~- 仕様書記述不明確(曖昧)~~ ~~- 仕様書記述標準違反~~ - 仕様書記述問題 - 仕様書修正漏れ ~~- 仕様書間不整合~~ ~~- 仕様からの展開時の見落とし~~ - 仕様書・コード間不整合 ~~- 仕様の理解不足~~ ~~- 仕様の確認不足~~ ~~- 仕様書の検討粗漏~~ - 仕様書理解問題 - コーディング時の言語用法の知識不足 - 再利用時のチェック漏れ - 修正時のチェック漏れ - 単なる凡ミス - 操作ミス ~~- 周知連絡の不徹底~~ ~~- 標準違反~~ ~~- 指摘ミス(仕様どおり)~~ ~~- 原因不明~~ - その他	故障発生原因の種類 仕様変更理由の種類
エラーを作りこんだ工程	発見工程と同じ	障害発生の原因となった工程
障害を本来発見すべき工程	発見工程と同じ	障害を発見しなければならなかった設計もしくはテスト工程
障害を発見できなかった要因	- 未入力 ~~- レビュー未実施(プロジェクト内レビュー/デザインレビュー/お客様レビュー)~~ ~~- レビュー指摘もれ~~ ~~- 再レビュー及び修正・確認もれ~~ - レビュー漏れ・見逃し - 工程間引継ぎコミュニケーション不足 ~~- 試験項目抽出もれ~~ ~~- テストそのもののもれ~~	障害を発見しなければならなかった設計もしくはテスト工程で，摘出されなかった要因

	- テスト(項目抽出・実施)漏れ - 環境上の障害で後工程にもっていった - 結果確認ミス - その他	

2．4　分析方法

手戻り作業量と手戻り要求データの関係を分析する手段としては,ノンパラメトリック検定を用いる．手戻り作業量として用いる構成管理上の変更データの多くは，非常に偏っていて一定の分布(正規分布など)に従っていないうえ，サンプル数についても偏りが大きいため，ノンパラメトリック検定（Kruskal-Wallis 検定・中央値検定）を用いる．漸近有意確率（p 値）で手戻り作業量に関する手戻り要求の属性項目の影響度を検証し,影響度が高い項目を抽出する.さらに,抽出された項目の各カテゴリについて，全データの平均値や中央値と比較を行い，特に差の大きいカテゴリについて，障害データの問題詳細や修正内容，開発者へのインタビューなどを元に考察する.

３．分析事例

3．1　分析対象プロジェクト

本論文で分析対象とするのは，今日の日本における情報システム開発の典型とも言える「マルチベンダー開発」である．具体的には，経済産業省の支援の受けて進められた約 330K ステップ規模の情報システム開発を対象とした．このプロジェクトでは，日本の複数の中核ソフトウェア企業が開発を担当し，奈良先端科学技術大学院大学 EASE プロジェクト(EASE：Empirical Approach to Software Enginnering)[4]と情報処理推進機構（IPA）下に設置されたソフトウェアエンジニアリングセンター（以降，SEC と呼ぶ）[5]が連携してデータ収集体制の構築とデータ分析を行った．開発はウォーターフォールプロセスにより進められ，ユーザ的立場の企業が要求を出し,プロジェクト管理担当企業の指揮のもとで開発担当 5 社がサブシステムの開発をそれぞれ行い，社内単体試験，社内結合試験を経て，社間結合試験，社間総合試験が行われた．本プロジェクトは，2 年にわたって新規開発(一年目)，処理の共通化と機能追加（二年目）が行われたが，その 2 年目のデータを用いる.

3．2　データ収集手順

本論文で分析するデータは，対象プロジェクトで用いられる障害管理ツール(GNATS[6])と構成管理ツール(CVS)から収集する．このプロジェクトは，2005 年 4 月開発開始で 2006 年 11 月頃まで開発が行われ，2005 年度の新規開発に対して 2006 年度は主にその改造と機能追加，および処理の共通化を行っている.障害管理ツールと構成管理ツールから上記のデータを獲得するためのツールとして，EASE プロジェクトによって開発された EPM ツール[4]を用いる．このツールは，障害管理ツールや構成管理ツールからデータを抽出し，それらの加工データを図表に可視化したり CSV 形式で出力したりする機能を持つ.

本事例で用いたデータは，障害管理ツールや構成管理ツールから EPM で収集し xml 形式で出力する一次加工データ（以降，EPM データ）である．EPM データは，障害データに関して

[4] EASE(Empirical Approach to Software Engineering)プロジェクト:　http://www.empirical.jp/project/index.html
[5] Software Engineering Center : http://sec.ipa.go.jp/index.php
[6] GNATS(GNU Bug Tracking System) : http://www.gnu.org/software/gnats/

は障害管理ツールに入力されるすべての障害項目であるが，構成管理ツールからは主に次のデータを出力する．

- CVS 上の操作日時，操作種別（ファイル追加，ファイル変更，ファイル削除，ファイル/ディレクトリの取り出し，など），操作対象ファイル名，操作者，(ファイル更新の場合)追加行数，削除行数，変更後ファイル行数，更新時コメント etc.

今回の分析は，このデータを市販の表計算ツール(MS-Excel)や統計解析ツール(SPSS)を用いて行う．

データ収集にあたっては，各障害管理データと修正履歴データを関連付けて取り出せるように，開発開始前に障害管理ツールと構成管理ツールの運用ルールとして，以下の依頼を開発担当者に行った．

- 障害 1 件に対して，更新は 1 回（複数ファイルの場合も一括で更新）で行い，その際にコメントとして障害番号を入力する
- 原則として，同一ファイルで複数の障害の修正を一括して更新しない
- 複数の障害を一括で更新する場合は，対応した障害番号を列挙する
- 障害以外の更新に関しては，「未登録」というコメントと更新理由を入力する

3．2　データ加工処理

統計分析を行う前のデータ加工処理は，以下の手順で行う．

- ファイル更新データの取り出し : 構成管理システムから取り出される EPM データは，CVS 上の操作すべて（ファイル更新以外に，ファイルの取り出しなど）を含む．そのため，ファイルを更新した情報のみを取り出す．これは，EPM データの操作タイプによってフィルタリングできる．
- 障害番号の取り出し：ファイル更新時に入力されたコメントから障害番号を取り出す．これは，運用ルールに従って記述されている場合，容易に取り出すことができる．コメントに「未登録」または障害番号が複数記述されているものは，除外する．
- 手作業による障害番号入力：いくつかの障害については，コメントに障害番号の入力がなく，代わりに修正内容を詳細に記述している．それらについては，障害管理データの障害詳細情報や修正内容などの情報に基づいて，障害修正と対応付けられるものは手作業で障害番号を付加する．
- 障害データと手戻り作業量データの関連付け：障害番号をキーとして，手戻り作業量データを集計・計算する．例えば，同一障害番号に対する更新数から更新ファイル数，追加/削除行数の合計が，算出される．最後に障害管理データと組み合わせることによって，障害番号毎の分析用データを作成する．

また，いくつかの特殊なケースについて，以下の加工を行った．

- 追加行数の変更：構成管理データからの追加行数には，新規ファイル追加時の行数は含まれていない．そのため，新規ファイルの追加時には，その行数を追加行数に合算した．削除行数についても同じ操作が必要であるが，対象データ中で，障害修正のためにファイルを削除したケースは存在しなかった．
- 未入力データの扱い : 未入力というカテゴリがある障害項目について，「未入力」だった場合は欠損値として扱う

3．3　収集データの概要

1 障害に対して修正を行った手戻り作業量データが一意に対応付けられた割合は，2005 年度は全障害の約 16%だけだったが, 2006 年度は全障害の約 73%が対応付けられた. これによって，より詳細な分析が可能になった. 理由としては，一年目は障害管理，構成管理とも新しいツールを導入したこと，運用ルールが開発現場まで浸透していなかったこと，新規開発が主で単体試験の障害が多く，障害毎のファイル更新のオーバーヘッドが大きく，運用ルールどおりの CVS 運用ができなかったことがあげられる.

2006 年度の手戻り作業量データの概要を表 3 に示す.

表 3　手戻り作業量データの統計情報

		更新ファイル数	追加行数	削除行数	変更行数	更新後行数	追加行率	削除行率
度数	有効	358	358	358	358	358	358	358
	欠損値	0	0	0	0	0	0	0
平均値		1.66	27.75	16.29	44.04	1037.73	4.10	2.20
中央値		1	9	2	12	527.50	1.55	0.49
標準偏差		1.99	66.65	64.48	123.87	1567.69	7.27	5.29
分散		3.94	4442.55	4157.48	15344.67	2457648.80	52.85	28.03
最小値		1	0	0	0	0	0.00	0.00
最大値		25	917	809	1677	11785	65.17	54.02
パーセンタイル	25	1	3	1	5	333	0.50	0.17
	50	1	9	2	12	527	1.55	0.49
	75	1	22	9	29	1004	3.97	1.64

４．データ分析結果

表 4 と表 5 にノンパラメトリック検定の結果を示す.

これらの表から，両検定で有意差がある（p 値<0．01）障害項目と手戻り作業量データの組み合わせは，以下のとおりである.

- ソフトウェアエラー分類：更新後行数
- 障害原因：追加行数・変更行数・更新後行数
- 発見工程：更新ファイル数・追加行数・変更行数・更新後行数
- 修正工程：追加行数・更新後行数
- 重要度：更新後行数・削除行率
- 障害を本来発見すべき工程：追加行数・更新後行数
- 優先度：更新後行数・削除行率
- 発見作業：更新ファイル数・追加行数・削除行数・変更行数・更新後行数・追加行率
- 発見箇所：更新後行数

上記の障害項目について，詳細に見るために，各項目のカテゴリごとの統計値を比較する.

統計値としては，平均値・中央値・標準偏差を扱い，データ全体の統計値（表 3）の値と比較し，特に差異が大きいものを取り出し，その理由を考察する.

表 **4** **Kruskal-Wallis** 検定結果（**p** 値）

障害項目	更新ファイル数	追加行数	削除行数	変更行数	更新後行数	追加行率	削除行率
試験項目種別	0.845	0.988	0.333	0.945	0.319	0.621	0.516
障害を発見できなかった要因	0.040	0.001	0.003	0.001	0.179	0.003	0.024
エラーを作りこんだ工程	0.069	0.027	0.050	0.026	0.077	0.622	0.130
ソフトウェアエラー分類	0.020	0.326	0.184	0.126	0.003	0.027	0.056
障害原因	0.006	0.000	0.205	0.000	0.017	0.008	0.393
再現度	0.027	0.063	0.170	0.044	0.122	0.072	0.380
発見工程	0.001	0.000	0.029	0.000	0.000	0.054	0.597
障害処理機能	0.539	0.019	0.317	0.136	0.868	0.322	0.380
起票分類	0.007	0.041	0.142	0.054	0.053	0.155	0.622
修正工程	0.007	0.003	0.171	0.011	0.000	0.669	0.898
重要度	0.160	0.725	0.045	0.864	0.000	0.063	0.000
障害を本来発見すべき工程	0.017	0.009	0.493	0.012	0.000	0.023	0.184
優先度	0.320	0.765	0.049	0.672	0.001	0.059	0.000
発見作業	0.000	0.000	0.000	0.000	0.000	0.000	0.007
発見箇所	0.110	0.055	0.692	0.158	0.000	0.848	0.086

表 **5** メディアン**(**中央値**)**検定結果（**p** 値）

障害項目	更新ファイル数	追加行数	削除行数	変更行数	更新後行数	追加行率	削除行率
試験項目種別	0.802	0.815	0.498	0.902	0.348	0.379	0.557
障害を発見できなかった要因	0.052	0.029	0.039	0.021	0.293	0.011	0.101
エラーを作りこんだ工程	0.060	0.008	0.002	0.001	0.049	0.352	0.214
ソフトウェアエラー分類	0.099	0.638	0.049	0.354	0.000	0.300	0.212
障害原因	0.012	0.001	0.175	0.009	0.007	0.025	0.828
再現度	0.081	0.175	0.320	0.074	0.147	0.343	0.277
発見工程	0.002	0.000	0.016	0.000	0.000	0.358	0.848
障害処理機能	0.707	0.045	0.557	0.086	0.779	0.175	0.513
起票分類	0.030	0.211	0.070	0.209	0.221	0.225	0.644
修正工程	0.011	0.006	0.142	0.001	0.001	0.617	0.899
重要度	0.196	0.426	0.209	0.893	0.000	0.621	0.001
障害を本来発見すべき工程	0.026	0.004	0.167	0.002	0.004	0.090	0.079
優先度	0.365	0.878	0.242	0.723	0.003	0.559	0.005
発見作業	0.000	0.000	0.000	0.000	0.000	0.004	0.067
発見箇所	0.121	0.026	0.616	0.098	0.000	0.932	0.219

更新ファイル数		度数	平均値	中央値	標準偏差		
全体		358	1.66	1	1.99		
障害項目	カテゴリ	度数	平均値	中央値	標準偏差	平均値差	中央値差
発見工程	詳細設計	1	3.00	3	.	1.34	2
	コーディング(製造)	85	1.13	1	0.37	-0.53	0
	単体試験	189	1.70	1	2.27	0.05	0
	社内結合試験	73	2.05	1	2.18	0.40	0
	社間結合試験	9	1.89	1	2.03	0.23	0
発見作業	分析・解析中	57	2.95	1	3.96	1.29	0
	システム動作中(テスト・運用)	230	1.49	1	1.30	-0.17	0
	レビュー	70	1.11	1	0.36	-0.54	0

追加行数		度数	平均値	中央値	標準偏差		
全体		358	27.7	9	66.65		
障害項目	カテゴリ	度数	平均値	中央値	標準偏差	平均値差	中央値差
障害原因	仕様書記述問題	21	35.9	16	48.71	8.2	7
	仕様書・コード間不整合	39	23.5	11	35.77	-4.3	2
	仕様書理解問題	73	35.9	13	46.89	8.1	4
	コーディング時の言語用法の知識不足	29	42.6	14	81.65	14.9	5
	再利用時のチェック漏れ	38	14.7	8.5	20.99	-13.0	-0.5
	修正時のチェック漏れ	35	10.6	5	13.59	-17.2	-4
	単なる凡ミス	83	26.7	5	103.58	-1.1	-4
	操作ミス	10	12.7	3.5	20.60	-15.0	-5.5
	その他	18	45.8	10	96.59	18.0	1
発見工程	詳細設計	1	84.0	84	.	56.3	75
	コーディング(製造)	85	12.6	5	20.19	-15.1	-4
	単体試験	189	28.2	10	52.60	0.4	1
	社内結合試験	73	42.1	11	116.04	14.4	2
	社間結合試験	9	41.1	19	53.02	13.4	10
修正工程	プログラム設計	3	68.3	23	87.32	40.6	14
	コーディング(製造)	81	14.2	7	20.61	-13.6	-2
	単体試験	168	31.3	10.5	54.58	3.6	1.5
	社内結合試験	66	41.7	11	121.56	14.0	2
	社間結合試験	6	57.3	29	59.53	29.6	20
障害を本来発見すべき工程	詳細設計	5	60.4	27	67.55	32.7	18
	プログラム設計	41	33.5	16	57.42	5.8	7
	コーディング(製造)	251	26.1	7	70.75	-1.6	-2
	単体試験	46	22.7	9	40.20	-5.1	0
	社内結合試験	12	51.8	16	93.21	24.1	7
発見作業	分析・解析中	57	75.8	28	140.09	48.1	19
	システム動作中(テスト・運用)	230	21.4	10	36.24	-6.3	1
	レビュー	70	9.7	3	17.97	-18.0	-6

削除行数		度数	平均値	中央値	標準偏差		
全体		358	16.3	2	64.48		
障害項目	カテゴリ	度数	平均値	中央値	標準偏差	平均値差	中央値差
発見作業	分析・解析中	57	53.8	7	147.96	37.5	5
	システム動作中(テスト・運用)	230	10.2	2	25.87	-6.1	0
	レビュー	70	6.1	1	11.08	-10.2	-1

変更行数		度数	平均値	中央値	標準偏差		
全体		357	44.1	12	124.03		
障害項目	カテゴリ	度数	平均値	中央値	標準偏差	平均値差	中央値差
障害原因	仕様書記述問題	21	46.6	18	63.63	2.5	6
	仕様書・コード間不整合	39	34.3	13	61.54	-9.9	1
	仕様書理解問題	73	48.9	23	64.82	4.8	11
	コーディング時の言語用法の知識不足	29	78.1	24	135.53	34.0	12
	再利用時のチェック漏れ	38	22.1	11.5	30.66	-22.1	-0.5
	修正時のチェック漏れ	35	15.7	7	19.58	-28.5	-5
	単なる凡ミス	83	43.2	8	186.55	-1.0	-4
	操作ミス	10	19.5	5.5	32.40	-24.6	-6.5
	その他	18	104.1	15.5	265.67	60.0	3.5
発見工程	詳細設計	1	137.0	137	.	92.9	125
	コーディング(製造)	85	19.3	7	29.24	-24.8	-5
	単体試験	189	47.6	13	111.09	3.5	1
	社内結合試験	73	61.9	17	203.18	17.8	5
	社間結合試験	9	51.3	24	57.04	7.2	12
発見作業	分析・解析中	57	129.6	41	273.55	85.5	29
	システム動作中(テスト・運用)	230	31.6	13.5	56.38	-12.6	1.5
	レビュー	70	15.9	5	26.81	-28.3	-7

更新後行数		度数	平均値	中央値	標準偏差		
全体		358	1037.7	528	1567.69		
障害項目	カテゴリ	度数	平均値	中央値	標準偏差	平均値差	中央値差
ソフトウェアエラー分類	インタフェースエラー	38	883.4	505.5	1236.46	-154.3	-22
	論理エラー	132	1374.0	778	1758.52	336.3	250.5
	データ定義エラー	38	880.8	465	1327.78	-156.9	-62.5
	テーブル定義エラー	4	2499.8	1655.5	2228.68	1462.0	1128
	形式エラー	10	2115.3	772	2706.94	1077.6	244.5
	その他	10	2904.9	1211	3856.65	1867.2	683.5
障害原因	仕様書記述問題	21	2044.8	761	2544.12	1007.1	233.5
	仕様書・コード間不整合	39	1022.1	524	1207.71	-15.7	-3.5
	仕様書理解問題	73	1352.1	531	2296.61	314.4	3.5
	コーディング時の言語用法の知識不足	29	833.6	580	661.67	-204.1	52.5
	再利用時のチェック漏れ	38	1118.4	586.5	1418.06	80.6	59
	修正時のチェック漏れ	35	501.7	536	267.36	-536.0	8.5

	単なる凡ミス	83	712.8	412	968.86	-324.9	-115.5
	操作ミス	10	377.6	378	102.72	-660.1	-149.5
	その他	18	1703.1	932	2010.47	665.4	404.5
発見工程	詳細設計	1	1027.0	1027	.	-10.7	499.5
	コーディング(製造)	85	596.7	359	863.24	-441.0	-168.5
	単体試験	189	986.6	538	1447.24	-51.1	10.5
	社内結合試験	73	1539.3	755	1921.53	501.5	227.5
	社間結合試験	9	1856.7	532	3600.11	818.9	4.5
修正工程	プログラム設計	3	2817.3	2357	2706.03	1779.6	1829.5
	コーディング(製造)	81	625.8	411	878.45	-411.9	-116.5
	単体試験	168	1108.4	566.5	1595.24	70.7	39
	社内結合試験	66	1444.8	745.5	1778.53	407.1	218
	社間結合試験	6	2665.2	1020.5	4286.44	1627.4	493
重要度	重大	178	713.1	446	1086.33	-324.7	-81.5
	中度	110	1334.6	668	1726.27	296.9	140.5
	軽微	69	1356.1	588	2094.00	318.3	60.5
障害を本来発見すべき工程	詳細設計	5	798.8	640	512.47	-238.9	112.5
	プログラム設計	41	1391.4	624	1800.49	353.7	96.5
	コーディング(製造)	251	771.7	481	1127.61	-266.0	-46.5
	単体試験	46	1840.7	1018	2451.45	802.9	490.5
	社内結合試験	12	2246.6	1489	2649.63	1208.9	961.5
優先度	高	184	761.4	480	1170.42	-276.3	-47.5
	中	99	1327.6	654	1712.16	289.8	126.5
	低	74	1294.2	587.5	2034.33	256.4	60
発見作業	分析・解析中	57	1056.2	732	1165.91	18.4	204.5
	システム動作中(テスト・運用)	230	1217.8	585	1810.60	180.1	57.5
	レビュー	70	385.7	339.5	216.43	-652.0	-188
発見箇所	未入力	1	4210.0	4210	.	3172.3	3682.5
	新規	255	823.7	480	1377.58	-214.0	-47.5
	改造(他システムからの流用)	11	1091.5	861	1008.29	53.8	333.5
	改造(システム内流用)	79	1606.4	795	2013.44	568.6	267.5
	再利用(未改造)	12	1529.1	1132.5	1306.15	491.4	605

追加行率		度数	平均値	中央値	標準偏差		
全体		358	4.10	1.5	7.270		
障害項目	カテゴリ	度数	平均値	中央値	標準偏差	平均値差	中央値差
発見作業	分析・解析中	57	8.79	4.4	12.341	4.69	2.8
	システム動作中(テスト・運用)	230	3.31	1.3	5.582	-0.79	-0.2
	レビュー	70	2.93	1.2	4.957	-1.17	-0.3

削除行率		度数	平均値	中央値	標準偏差		
全体		358	2.20	0.5	5.295		
障害項目	カテゴリ	度数	平均値	中央値	標準偏差	平均値差	中央値差
重要度	重大	178	3.02	0.7	5.955	0.82	0.2
	中度	110	1.31	0.3	5.337	-0.89	-0.2
	軽微	69	1.54	0.4	2.378	-0.66	-0.1
優先度	高	184	2.92	0.6	5.876	0.72	0.1
	中	99	1.35	0.4	5.574	-0.85	-0.1
	低	74	1.59	0.4	2.474	-0.61	-0.1

[更新ファイル数] 「発見工程」が進むほど，更新範囲が広くなる，というのは，常識的な結果といえる．9 件しかないが，「社間結合試験」で発見された障害で変更ファイル数が少ないのは，構成管理が社別に行われていて，社間のプログラムをまたいだ変更が取り出せなかったことも一因である．「発見作業」が「レビュー」というのは，ほとんどコーディングレビューで発見された障害である．また，「分析・解析中」に発見された障害というのは，類似障害を芋づる式に発見するケースが多かったことが，開発者へのインタビューでわかっている．

[追加行数] 「障害原因」が「再利用時や修正時のチェック漏れ」の障害は，比較的小規模な追加であり，「仕様書離間問題」や「仕様書記述問題」が原因の場合に追加行数が大きい．「コーディング時の言語用法の知識不足」による問題は，追加行数が大きいが，これは標準偏差も大きいので，ばらつきが大きいと思われる．基本的なクラスの使い方の間違いによるシステムを網羅する変更が必要な障害があり，影響範囲が小さいが追加行数が大きかったことがインタビューでわかっている．「発見工程」が進むほど，追加行数が大きくなる，「障害を本来発見すべき工程（混入工程）」が上流であるほど追加行数が大きくなる，というのは，常識的な結果といえる．一方，「社内結合試験」で見つかるべき障害の追加行数が，「詳細設計」で見つかるべき障害と同じく大きいのは，結合してみないと動作が確認できないような障害は，新たな比較的大規模のコーディング追加を伴うケースためと考えられる．

[削除行数] 「分析・解析中」の削除行数が大きいのは，変更ファイル数と同じく芋づる式の変更箇所があったためと思われる．標準偏差が大きいので，一般的な傾向とは思えない．「レビュー」による指摘は，主にファイル単位で行うため，見る範囲が小さく，削除も小規模である，と考えられる．

[変更行数] 追加行数と同じ傾向を示している．

[更新後行数] 「ソフトウェアエラー分類」による相違は，ファイルサイズ自体の大きさにもよる．テーブル定義ファイルは 1 ファイルサイズが大きい．「形式エラー」，「論理エラー」などは，ログ出力やデータ処理などを行う部分に多く，コードサイズが大きい．「インタフェースエラー」での更新後行数が小さいのは，プログラムがきちんと構造化され，インタフェース部分が小さい部分に集約されているからと思われる．「障害原因」や「発見工程」，「障害を本来発見すべき工程」関係は，ほぼ追加行数と同じ傾向を示している．一方，「重要度」や「優先度」が高いもので，更新後行数が大きいのは，重要な手戻りは，比較的小さな範囲の修正で収まっている，と考えられる．また，「発見箇所」「新規」作成部分での手戻りは，「再利用」や「改造」部分に比べ変更範囲が小さい．これは，再利用や改造部分が複雑で保守性が低い，もしくは再利用や改造部分で見つかる障害は，広範囲な対応が必要な場合が多い，と考えられる．

[追加行率] 追加行数と同じ傾向を示している．

[削除行率] 「重要度」や「優先度」が高い障害で大きくなっている． これは，更新後行数の逆で，範囲が小さいためと思われる．一応，差はあるが，あまり大きな差ではない．

5．考察

5．1 分析結果の考察

4 章の結果から，次のようなことがわかる．

- 発見工程が遅い，あるいは混入工程（「障害を本ら発見すべき工程」）が設計工程の場合，追加行数が多いことから，品質低下の危険が大きいと思われる．バグの発見が遅いほど，あるいは，バグがシステム中に滞留している時間が長いほど（混入工程から発見工程までの時間が長いほど）修正工数が増大する[1][2]，という多くの先行研究と類似した結果で，バグ発見の遅延がコストと品質の両面から影響が大きいことがわかった．

- 再利用時のチェック漏れや修正時のチェック漏れが原因で混入するバグによって，手戻りの作業量は少ないが，改造箇所や再利用箇所から発見される障害の影響範囲は大きい．このことから，既存のプログラムコードの再利用や改造は，修正量が少なく一見効率的だが，品質低下の危険があることを考慮する必要がある，ということがわかった．
- コーディングミスによる手戻り量はばらつきが多い．変更範囲は小さいが，追加量が異常に多くなる場合がある．

上記のことから，下流工程での品質低下を防止するために，以下のような留意点が考えられる．

- バグの検出は，品質の低下を防止するためにも，なるべく上流工程で行う．
- 再利用や改造部分については，手戻り範囲が大きくなる傾向にあるので，変更箇所の部分的なチェックよりも，変更していない部分についてシステム全体を俯瞰したチェックを重視するべきである．また，変更量などについては，通常の新規開発よりも大きく見積もるべきである．
- コーディングミスによる影響はばらつきが多いので，コードレビューはプログラマのスキルに応じて慎重に行う．

5．2　提案する分析手法の実用上の課題

データの収集，分析およびプロジェクト関係者への分析報告を行った過程で，把握できている問題と対応方法を以下に示す．

- 新規開発の場合，単体試験でのバグの数が初期に多く発生し作業が集中するため，1 障害毎のファイルの更新は手間がかかり，複数障害の一括更新が多い．
- 追加行数や削除行数は，コーディングルールやスタイルに依存する部分が多い．変更時に変更内容などをコメントとしてソースコードに追記する場合，追加行数が大きくなってしまう．また，不要なコードを削除する時も，/**/，#if0～#endif，完全消去の場合でそれぞれ追加行数・削除行数が異なる．分析の精度を確保するためには，コーディングルールの統一とそれに応じたデータ加工処理を行うべきである．

５．まとめと今後の課題

本稿では，プログラムの品質低下の一因である下流工程での手戻り作業の量とその手戻り要求の特徴との関係を分析した．手戻りの作業量とは，プログラムコードの変更に関する量で，構成管理データから取り出した．手戻り要求データは，実際にはテスト工程で発見される障害データを用いた．障害データの管理システムには，その目的にしたがって障害の現象や原因，修正方法などの情報が入力されており，本研究では，発見，修正，混入工程などの時間に関するデータと，エラー分類，障害原因，発見機能などエラーそのものの特徴に関するデータ，重要度や優先度など管理上のデータなどを用いた．

手戻り要求の特徴と手戻り作業量の関係を分析する手段としては，ノンパラメトリック検定（Kruskal-Wallis 検定・中央値検定）を用いた．手戻り作業量に影響する手戻り要求の特徴を抽出後，抽出された項目の分類ごとの平均値や中央値を比較し，特に差の大きい分類について，考察した．例えば，「発見工程」は変更ファイル数や追加行数など多くの手戻り作業量に影響する．これは，発見工程が遅くなるほど品質への影響が大きく，また影響範囲もが大きくなること

を示している．また，「改造」箇所や「再利用」箇所から発見される障害の影響範囲は大きい，「コーディングミス」による変更範囲は小さいが追加行数が異常に多くなる場合がある，などがわかった．

この結果は，実際に開発者へフィードバックされ，いくつかのサブシステムは開発者のスキルが低いため追加行数・削除行数が多い，あるいは，通常の開発では障害は「対応漏れ」が多く，コードの追加がメインとなるはずで，削除行数が多い場合は試行錯誤で開発している可能性が高い，といった現場での意見も聞くことができた．

今後は，異なるプロジェクトでのデータを分析し，品質への影響計測により有効な手戻り作業量を抽出したいと考えている．

謝　　辞

本研究の一部は文部科学省「e-Society 基盤ソフトウェアの総合開発」の委託に基づいて行われた．本研究にあたり多くのご協力を頂いた EASE プロジェクト関係諸氏に感謝します．

参　考　文　献

[1] B. Boehm, Software Engineering Economics, Englewood Cliffs, NJ: Prentice-Hall, 1981.

[2] 松村 知子，門田 暁人，森崎 修司，松本 健一，“マルチベンダ情報システム開発における障害修正工数の要因分析，” 情報処理学会論文誌，Vol.48，No.5，pp.1926-1935，May 2007.

[3] Royce, W. “Pragmatic quality metrics for evolutionary software development models. “ In Proceedings of the Conference on TRI-ADA '90 (Baltimore, Maryland, United States, December 03 - 06, 1990), pp.551-565,1990.

[4] 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，J88-D-I，no. 2，pp. 228-239，Feb. 2005.