INFORMATION
SCIENCE
TECHNICAL
REPORT

NAIST-IS-TR2007006
ISSN 0919-9527

プロセス改善を目的とするODCを用いた欠陥修正工数分析

松村知子，森崎修司，玉田春昭，大杉直樹，
門田暁人，松本健一

March 2007

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

プロセス改善を目的とする
ODC を用いた欠陥修正工数分析

松村知子，森崎修司，玉田春昭，大杉直樹，門田暁人，松本健一
奈良先端科学技術大学院大学　情報科学研究科

１　はじめに

ソフトウェア開発における変更要求（バグ，仕様変更，設計変更）の発生は，コスト(工数)超過や納期遅延の主原因の一つである．そこで，工程や機能ごとに変更要求発生件数を調べ，その発生要因との関係を明らかにする研究や分析法が提案されている．変更要求の多くは，開発（設計・製造）で混入した欠陥（設計ミス,設計漏れ,コーディングミスなど)から発生するが,ODC(Orthogonal Defect Classification: 直交欠陥分類法)[2]は，欠陥管理票に基づいて欠陥をいくつかの種類に分類し，各種類の欠陥発生件数の偏りの有無を工程間や機能間で調べることで，改善すべき工程や重点的にチェックすべき機能を特定する．DCA(Defect Causal Analysis：欠陥原因分析)[1][3][4]では，各欠陥のより詳細な情報(自然言語による記述やインタビュー結果)を収集し，特性要因図(Fishbone diagram)などを用いて欠陥の原因を階層的に整理し，欠陥発生件数との関係を分析する．

ただし，開発におけるコスト超過や納期遅延を防ぐ観点からは，欠陥の件数だけでなく，欠陥の修正工数に着目した要因分析が必要である．一般に，欠陥の修正工数のばらつきは大きく，たとえ 1 件の発生であっても，深刻なコスト（工数）超過や納期遅延につながる場合がある．

本稿では，ある中規模システム情報システム開発を対象とし，欠陥を特徴付ける特性値を収集し，修正工数との関係を分析した結果について報告する．収集する特性値としては，欠陥の「原因」や「発生した機能」といった欠陥自体の情報に加え，欠陥の「混入工程」,「発見工程」,「修正工程」などの工程に関する情報が含まれる．先行研究として，同対象プロジェクトの試験工程（単体試験～総合試験）で発生した欠陥を用いて， 分散分析やｔ検定分析，箱ひげ図など統計的な手法を用いて分析を実施し,「発見工程」や「発見遅延」「問題を発見できなかった要因」が修正工数に影響を与えていることを確認した[5]．

本報告の中心となる 2006 年度は,分析対象を上流工程の欠陥（仕様書や設計書のレビューでの指摘事項）に広げ，上流工程で発見された欠陥の修正工数の特徴を分析した．さらに，下流工程では，プロジェクトの開発担当社毎に欠陥

の発見件数と修正工数を ODC を用いて分析した．プロジェクトに関っている開発担当者自身が問題点の把握を容易に行いプロセス改善のポイントを抽出できるように，それぞれ以下の欠陥の属性[1]を用いて分析を行った．

- ✧ 上流工程：エラーが混入した原因（以降，エラー原因）
- ✧ 下流工程：故障発生原因/仕様変更理由の種類（以降，問題原因）と問題を発見しなければならなかった工程で発見されなかった要因（以降，発見できなかった要因）

分析結果としては，発生した欠陥数と修正工数を示す．修正工数については，上流工程では箱ひげ図を用いて分布を示し，下流工程では各分類での延べ修正工数を示す．

本分析と従来の ODC との違いは，修正工数を用いる点である．また，DCA のような人手による欠陥の原因分析やデータ加工を行わない．本分析では，レビュー時に作成された議事録（以降，問題記述票と呼ぶ）やテスト工程で作成される欠陥票に入力された欠陥に関する属性値（修正工数を含む）をそのまま用いる．本分析を実施するため，開発開始前にあらかじめ分析に必要と思われる欠陥属性データを決定し，収集できるような環境や仕組みを整えた．

2　分析事例

2.1　対象プロジェクト

本報告の対象としたプロジェクトは，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「マルチベンダ開発」である．具体的には，経済産業省の支援を受けて進められた中規模の情報システム開発プロジェクト「プローブ情報システム開発プロジェクト」を対象とした．このプロジェクトでは，著者らが所属する EASE（Empirical Approach to Software Engineering）プロジェクト[2]とソフトウェアエンジニアリング技術研究組合(以降，COSE と呼ぶ)[3]，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)[4]の 3 者が連携して，データ収集体制の構築とデータ分析を行った．COSE はユーザ的立場の会社とプロジェクト管理担当会社，開発担当５社から成り，ユーザ的立場の会社が要求を出し，開発担当５社が分担してサブシステムの開発を行い，プロジェクト管理担当会社が 5 社を統括した全体のプロジェク

---

[1] 欠陥の属性名称や属性の説明中の用語は，レビュー議事録やデータ収集時のツールの記述に準じる．本稿では，「エラー」は欠陥と同義であり，「問題原因」は「エラー原因」とほぼ同義であるが，下流工程の「問題」にはバグや仕様変更の他に実際には修正を必要としないもの（指摘ミスなど）を含む．「故障」は試験で動作異常と判断された現象である．

[2] http://www.empirical.jp/top.html

[3] ソフトウェアエンジニアリング技術研究組合（COSE）：COnsortium for Software Engineering

[4] http://sec.ipa.go.jp/index.php

ト管理を行う．開発はウォーターフォールプロセスにより進められ，具体的には，要件定義を受けて基本設計を全社の協働で行い，以降各社で詳細設計，プログラム設計，製造，単体試験，社内結合試験が行われた．その後，各社のプログラムを集め，組合全体として社間結合試験，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，2006 年度はその二年目にあたる．

2.2 収集データ

分析のためのデータは，対象プロジェクトで用いられる欠陥管理ツール(GNATS[5])と Excel ベースの問題記述票から収集する．

GNATS は Web ベースの欠陥管理ツールで，欠陥の対応状況を追跡（トラッキング）する機能や，メールで状況変化を通知する機能などを持つ．これは，主に試験工程で発生した欠陥管理に用いる．同ツールのリポジトリから，EASE プロジェクトで開発したツール EPM（Empirical Project Monitor）が必要なデータを収集し xml 形式で出力する．

問題記述票は，仕様書や設計書のレビューによって指摘される欠陥を記載する Excel フォーマットの文書で，レビュー単位で作成される．レビューは，必ず文書単位で行い，複数文書に対するレビューを一括で行う場合も，問題記述票を文書単位で個別に作成するよう各社に依頼した．EASE プロジェクトでは入力を簡易化したファイルを作成し，各社に配布するとともに，集計ツール(Excel マクロ）を作成し，分析に用いている．

本分析で用いる欠陥の属性と選択項目(カテゴリ)を以下の表 1 表 2 に示す．この他，修正工数は，上流工程の欠陥については重要度の小さいものを除いて 1 欠陥ごとに入力される．下流工程の欠陥に関しては，1 欠陥毎に修正工数が入力される．先行研究では，修正工数は欠陥発見から確認（回帰試験）までの総工数を記入したが，本報告の修正工数は分析・解析工数，修正工数，確認工数の 3 種類に分割し，それぞれにかかる作業の工数入力を開発関係者に依頼した．

対象データは，上流工程では基本設計工程と詳細設計工程，下流工程では単体試験と社内結合試験で発見された欠陥とし，その他の工程での欠陥データは用いなかった．

---

[5] GNATS(GNU Bug Tracking System)：http://www.gnu.org/software/gnats/

表 1　上流工程での欠陥属性データ項目一覧

項目名	選択項目(カテゴリ)	項目説明
エラー原因	A．要求の確認不足 B．設計条件の確認不足 C．実現方式の検討不足 D．設計技術の習熟不足 E．業務知識の習熟不足 F．周知連絡の不徹底 G．表現上の配慮不足 H．修正ミス	エラーが混入した理由

表 2　下流工程での欠陥属性データ項目一覧

項目名	選択項目(カテゴリ)	項目説明
問題原因	- 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 - 仕様の理解不足 - 仕様の確認不足 - 仕様書の検討粗漏 - 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
(問題を)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 - 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 - 環境上の問題で後工程にもっていった - 結果確認ミス - その他	問題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，発見されなかった要因

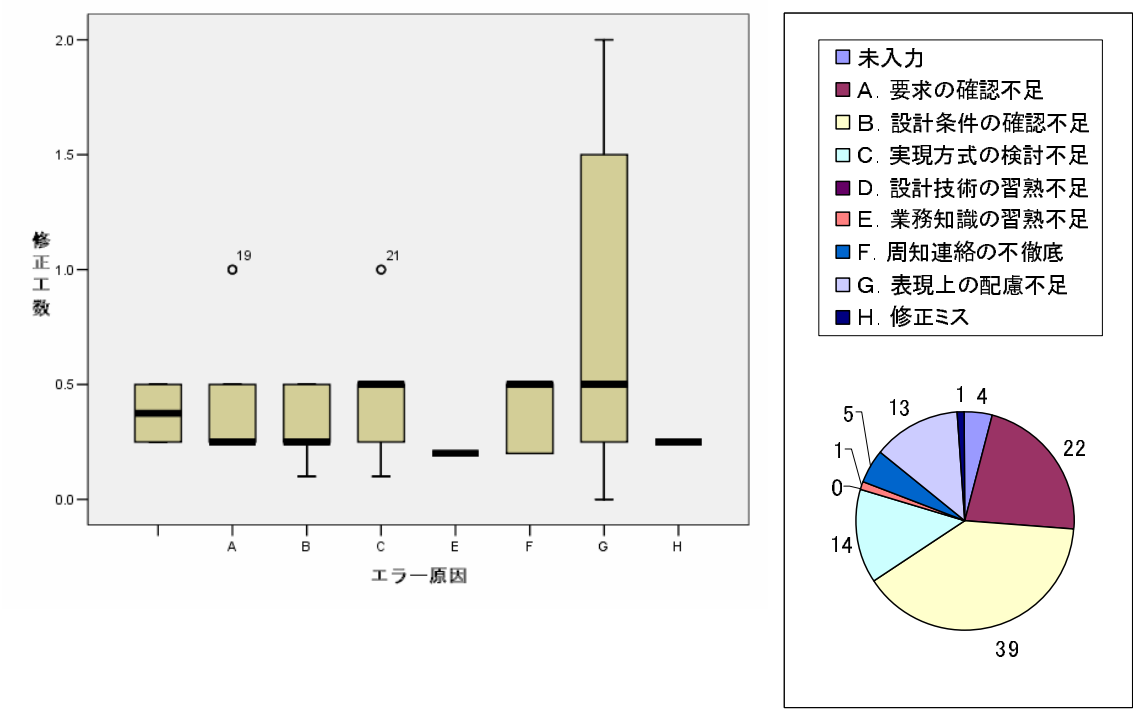

図 1　基本設計工程での欠陥数と欠陥修正工数

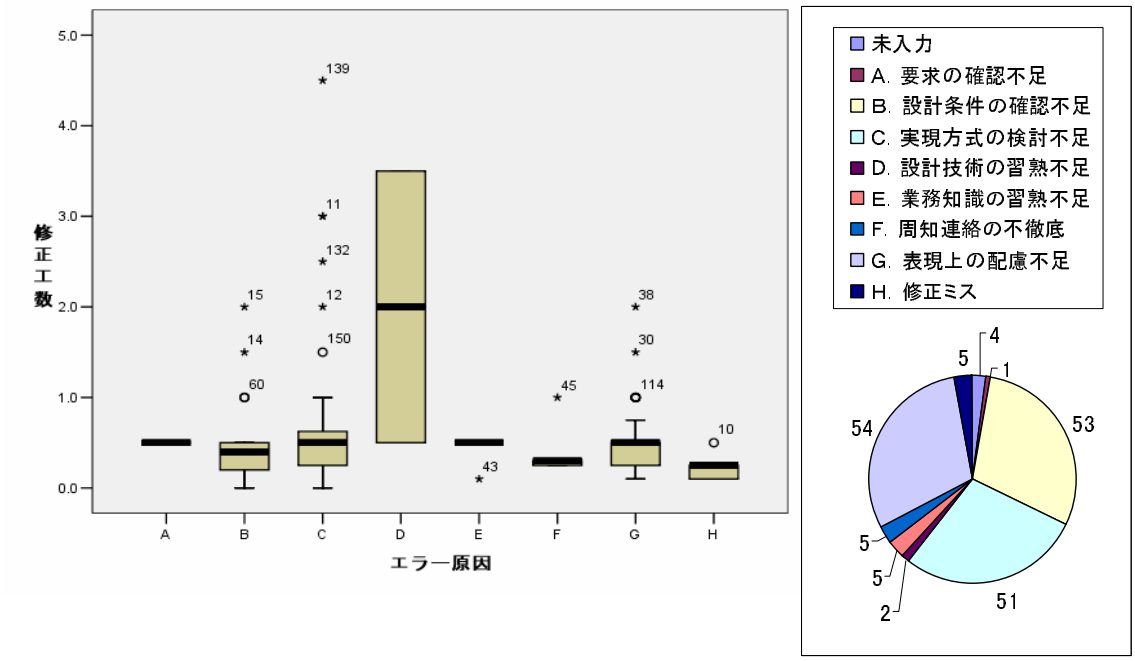

図 2　詳細設計工程での欠陥数と欠陥修正工数

2.3　分析結果

図 1は，基本設計工程で見つかった欠陥数と欠陥修正工数，図 2は詳細設計工程での欠陥数と欠陥修正工数を示している．基本設計工程では１０９件の重要欠陥中９９件の修正工数が入力され（10件は未入力)，詳細設計工程では１８７件の重要欠陥中１８０件の修正工数が入力されている（７件が未入力)．円グラフは，それぞれ修正工数が入力されたデータのエラー原因（表 1 参照）の割合（数値は件数）を示している．さらに，エラー原因の種類ごとに修正工数の中央値とバラツキを箱ひげ図で表現した．箱ひげ図は，縦軸を修正工数(人時)とし,中央値を横線で示し,データの25%から75%が存在する範囲を箱で示す．箱の 1.5 倍の範囲を垂直線（ひげ）で示し，ひげの範囲外のデータを極値として丸とアスタリスクで表している．基本設計でデータ数が多いのはA,B,C,G，詳細設計ではB,C,Gとなっている．

図 1 から，基本設計で見つかった欠陥の修正工数は，欠陥間でほとんど差がなく，0.25～0.5 人時に集中している．G（表現上の配慮不足）のみ，ばらつきが大きいが，これは基本設計での表現上の欠陥には修正工数が大きいものが存在する（設計書全体にわたって用語を変更する，など）ことを示している．一方，図 2からは,詳細設計工程になると，基本設計工程より中央値は特に高いわけではないが，データのばらつきが大きくなっていることがわかる．B（設計上の確認不足),C(実現方式の検討不足)など,設計内容にかかわる重要な欠陥で,修正工数の大きい欠陥が見られるようになる．

一方,図 3～6 は下流工程での問題原因と（問題を）発見できなかった要因の分類を用いて，発生件数と分類ごとの延べ修正工数を表にしたものである．図3図4がA社，図5図6がB社のデータである．各分類で発生件数と修正工数の比率に違いがないが，A社では，問題原因が『再利用時のチェック漏れ』で，発見できなかった要因が『試験項目抽出漏れ』の欠陥について，修正工数の比率が若干大きくなっている．B社では，問題原因が『仕様書の検討粗漏』で，やはり発見できなかった要因が『試験項目抽出漏れ』の欠陥について，修正工数の比率が若干大きくなっている．同じ『試験項目抽出漏れ』要因による欠陥でも，A社では『再利用時のチェック漏れ』B社では『仕様書の検討粗漏』と原因が異なる.A社開発担当者へのインタビューでは,試験項目抽出漏れが発生した場合,試験項目を見直し，試験項目を追加して再試験を行うため，対応する工数が増大する，ということが判明している．また，A社では，昨年度のコードの流用が多く，類似コンポーネント間でコードを流用している，という事情もあり，『再利用時のチェック漏れ』で発生したと思われる．コードの再利用に関する手順やプロセスの整備で，防止できる欠陥と考えられる．一方，B社では『仕様書の検討粗漏』が原因なので，仕様書の作成工程(設計工程)での作業プロセスの改

善で防止できると考えられる.

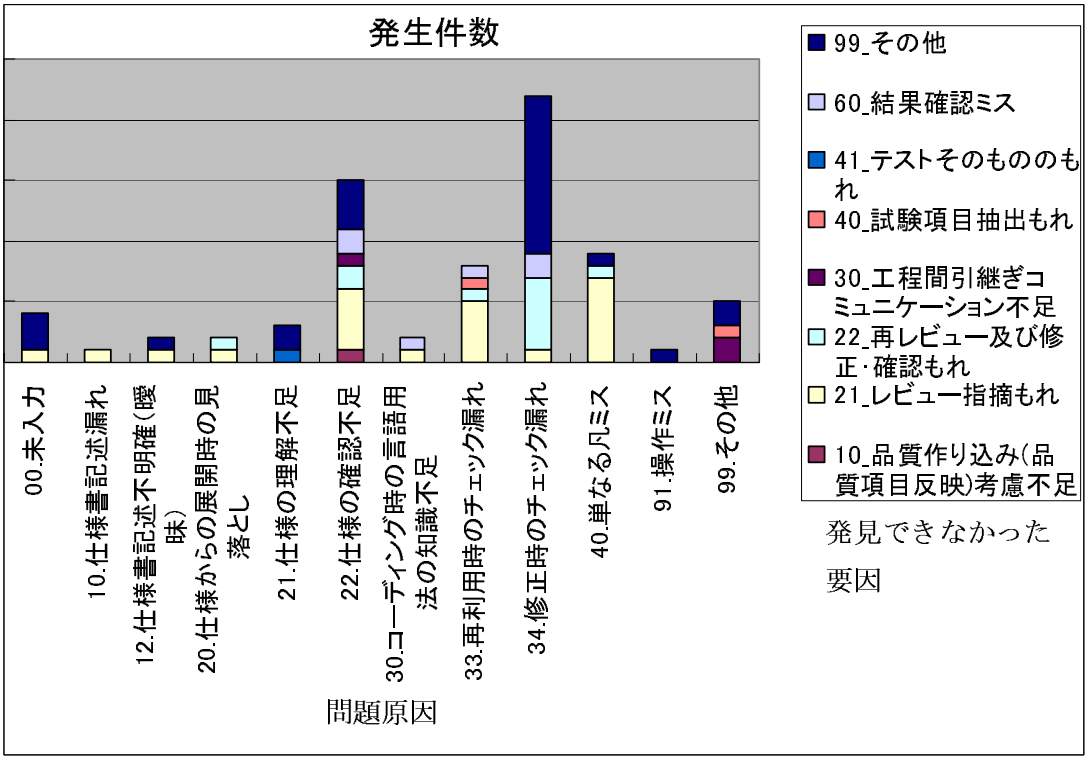

図 3　問題原因と発見できなかった要因による ODC（発生件数）・A 社

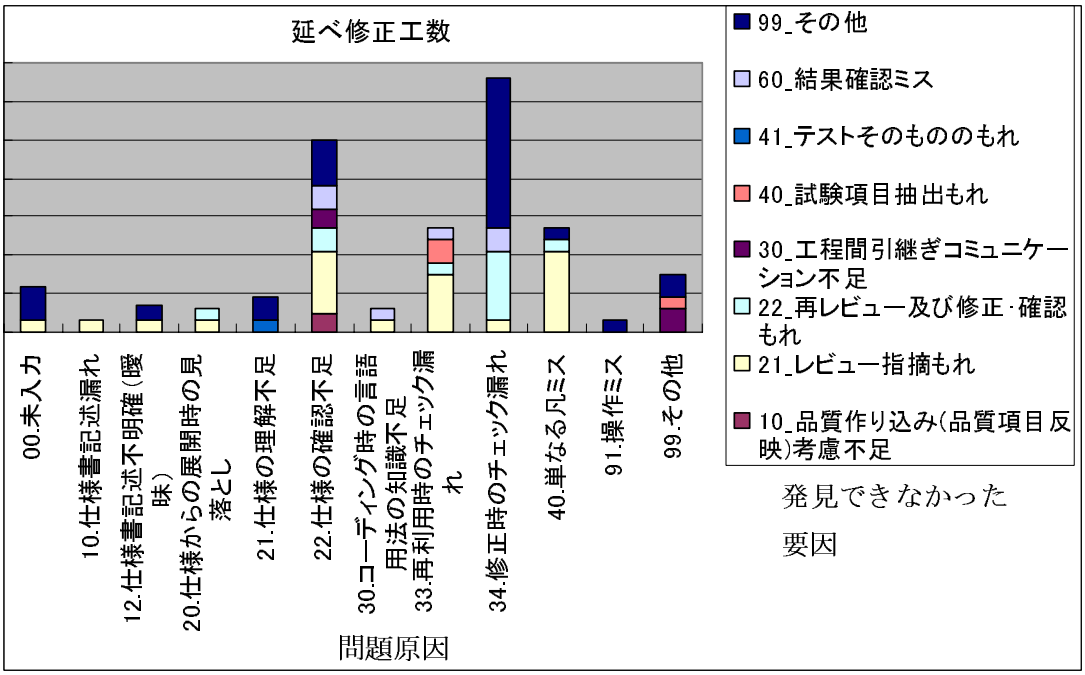

図 4　問題原因と発見できなかった要因による ODC（延べ修正工数）・A 社

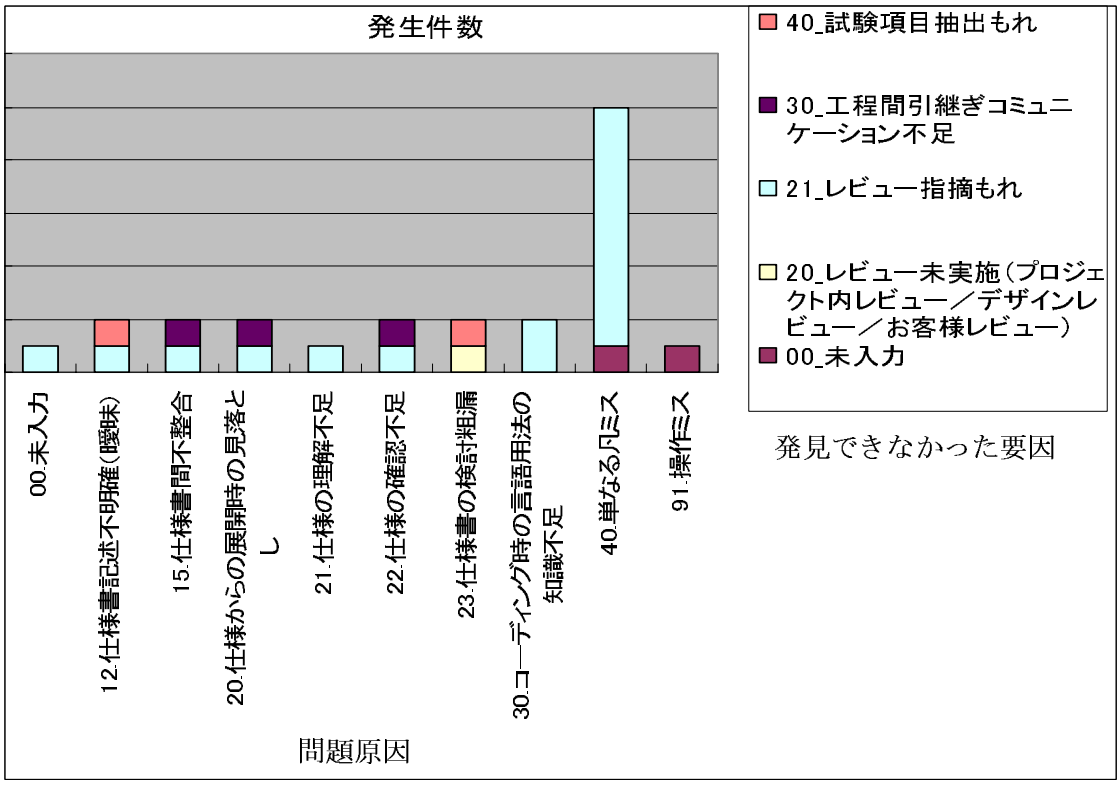

図 5　問題原因と発見できなかった要因による ODC（発生件数）・B 社

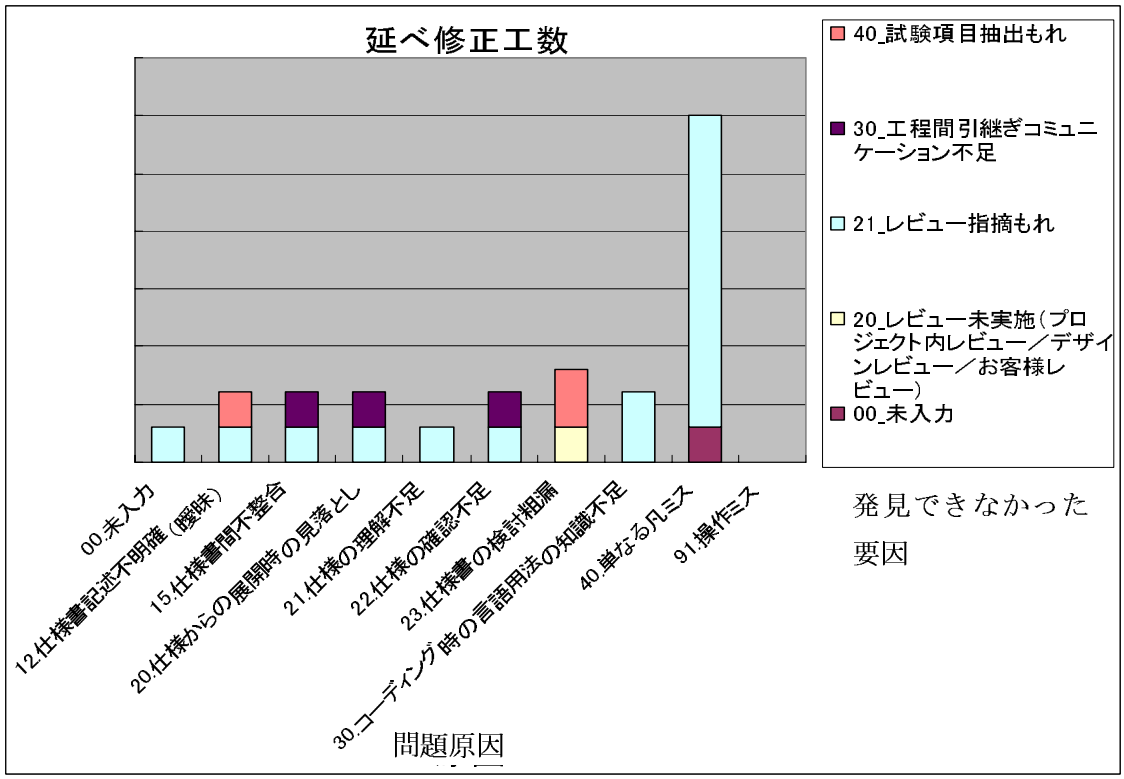

図 6　問題原因と発見できなかった要因による ODC（延べ修正工数）・B 社

3　考察

- ✧ **上流工程分析**：修正工数は，ほとんどの欠陥で 0.25～0.5 人時程度である．詳細設計で混入し，下流工程（テスト工程）で発見された欠陥の平均修正工数は 3.89 人時（同プロジェクト 2005 年度実績）[5]であり，これらの欠陥が詳細設計工程で発見された場合 0.25～0.5 人時程度で修正できるとすると，その差は非常に大きい．「欠陥の発見が遅れるほど修正工数が増大する」という法則[6]を裏付けている．一方，基本設計工程と詳細設計工程を比較すると，その差はあまりない．全体的な分布は，両工程とも 0.25～0.5 人時に集中している．しかし，詳細設計になると，「設計上の確認不足」や「実現方式の検討不足」など重要な要件に関して，修正工数が大きい欠陥が見られるようになる．もし，これらの欠陥が基本設計工程で見つけることが可能だとすると，基本設計工程で「設計上の確認」や「実現方式の検討」を十分に行うことで，次工程の欠陥修正工数を効率的に低減することが可能である，と言える．
- ✧ **下流工程分析**：先行研究では，ODC で分類された欠陥の平均修正工数を計測し，「発見すべき工程」と「発見工程」の差が大きい欠陥について，平均修正工数が大きくなる，という事実を確認した[7]．本報告では，ODC による件数と延べ修正工数の分析を同時に見ることによって，各組織の ROI（Return on Investment:投資対効果）を考慮したプロセス改善のポイントを抽出することを試みた結果を示している．例として，「問題原因」と「問題を発見できなかった要因」という 2 つの欠陥属性を用いて，同じ問題を発見できなかった要因（『試験項目抽出漏れ』）で修正工数が比較的大きいことがわかった．しかし，1 社では問題原因は再利用時の作業であり，他社では仕様書作成時の作業である．このように問題の所在を詳細に把握し，かつ問題解決時の低減コストが比較的大きいものを抽出することができるので，プロセス改善の具体的な対象を絞り込み適切な対処をすることが可能になる，と考えられる．

4　まとめ

本報告では，ODC を用いて，上流工程での欠陥修正工数の特徴分析と，下流工程での ROI に配慮したプロセス改善のポイントを抽出の試みについて，実際のプロジェクトのデータを用いた計測・分析と結果について述べている．具体的には，以下の 2 つの分析パターンについて，報告した．

- ✧ **観点 1**：上流工程（基本設計工程と詳細設計工程）におけるエラー原因別の欠陥数の割合と欠陥修正工数の分布
- ✧ **観点 2**：下流工程（単体試験工程～社内結合試験）における問題原因と問

題を発見できなかった要因別の欠陥数と欠陥修正工数比率

いずれのケースでも，修正工数を考慮した問題の対処やプロセス改善のポイントを抽出する手がかりを得ることができた．

今後は，プロセス改善ポイントを抽出するために，より有用な欠陥属性の組み合わせによる分析パターンを確立し，プロセス改善の実施による品質向上およびコスト低減の効果を評価したいと考えている．

参考文献

[1] D. Card, “Learning from our mistakes with defect causal analysis,” *IEEE Software*, Vol.15, No.1, pp.56-63, Jan./Feb. 1998.

[2] R. Chillarege, I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray and M. Wong, “Orthogonal defect classification-a concept for in-process,” *IEEE Trans. on Software Engineering*, Vol.18, No.11, pp.943-956, Nov. 1992.

[3] R. Mays, C. Jones, G. Holloway and D. Studinski, “Experiences with defect prevention,” *IBM Systems Journal*, Vol.29, No.1, pp.4-32, 1990.

[4] T. Nakajo and H. Kume, “A case history analysis of software error cause-effect relationships,” *IEEE Trans. on Software Engineering*, Vol.17, No.8, pp.830-838, Aug. 1991.

[5] 松村 知子，門田 暁人，森崎 修司，松本 健一，“マルチベンダー情報システム開発における欠陥修正工数の要因分析”，情報処理学会論文誌，Vol.48，No.5 (掲載決定).

[6] A. Endres and D. Rombach, *A Handbook of Software and Systems Engineering; Empirical Observations, Laws and Theories*, Pearson Education Limited, UK, 2003.

[7] 大杉 直樹，松村 知子，森崎 修司，“ソフトウェア開発の「見える化」を支援するデータ分析力：エンピリカルアプローチによる既存データの有効活用”，JISA 会報，No.80，pp.13-29，Jan. 2006.