

不具合管理システム利用時の 不具合修正プロセス改善の ための滞留時間分析手法

伊原 彰紀



大平 雅雄



松本 健一



滞留時間とは

- 滞留時間：
 - 不具合が報告されてから修正が完了するまでの時間
- 大量に報告された不具合に対応しなければならない現状では、滞留時間がきわめて長くなってしまう
- 例：
 - Linuxカーネルにおける不具合の残存時間は平均1.8年間

提案

- 不具合修正のどの部分がボトルネックとなっているのかを理解する必要がある
- 滞留時間を分析するための手法
 - 滞留時間を「未対応時間」と「修正時間」に分割し分析
 - 不具合修正開始から完了までの「修正時間」を詳細に分析
- Mozilla Firefox Projectの不具合修正プロセスを対象としたケーススタディを行う

関連研究(1)

- 不具合修正の問題点
 - Wangらはソフトウェア中に存在している不具合の数や修正された障害の数などを指標として用いている
 - Wangらの分析の結果
 - 不具合の修正開始までに相当な時間がかかっている
 - どれくらいの時間を要しているかについては明らかにしていない
 - 本研究では
 - 不具合の修正開始までにどれくらいの時間をようしているのか
 - 修正開始から完了までにどれくらいの時間を要しているのか

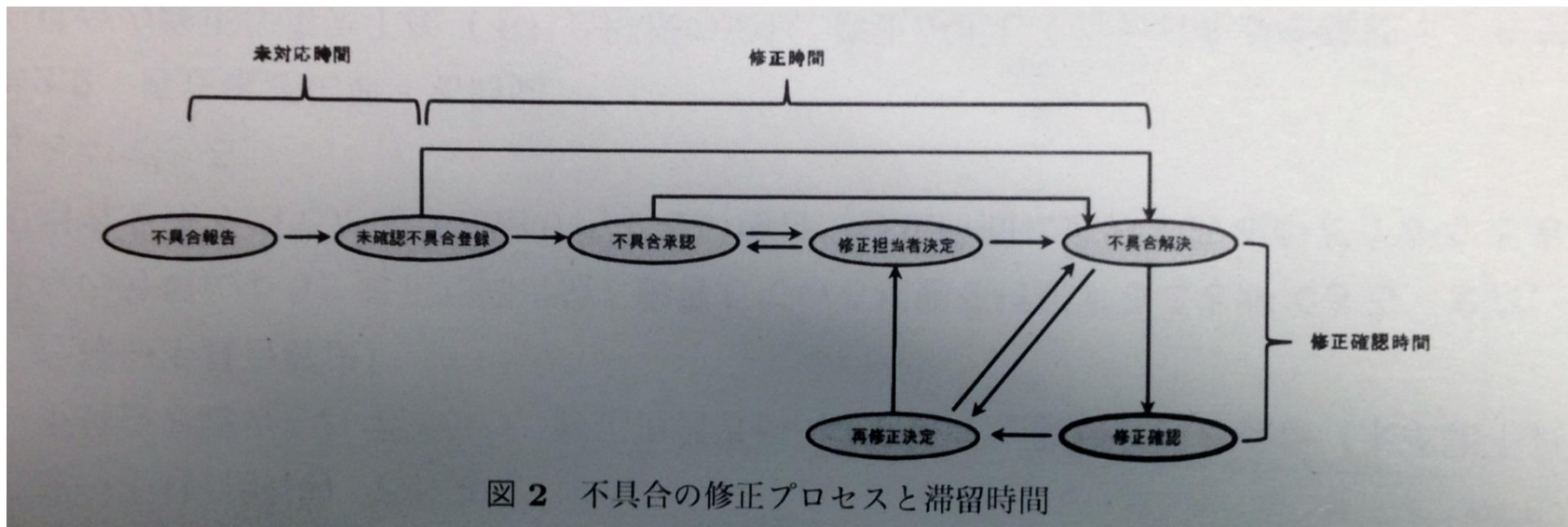
関連研究(2)

- OSS開発における不具合の滞留時間
 - MockusらのOSS開発が成功する理由の調査
 - 広い範囲で利用されている機能を持つモジュールに存在する不具合の滞留時間は短い
 - Herraizらは重要度の段階が複雑であることが問題
 - 重要度が高い不具合は低い不具合に比べて滞留時間が短い
 - 不具合の修正プロセスのどの部分で時間を要しているかを明らかにすることが必要
 - 本研究では
 - 修正プロセスを細かく分割することでボトルネックとなる部分を抽出

不具合管理システムの修正プロセス

状態	詳細
不具合報告	不具合管理システムに不具合の詳細を報告する
未承認不具合登録	不具合管理システムに不具合の詳細を登録する
不具合承認	不具合であることを承認する
修正担当者決定	修正の担当者が決定し、修正が開始される
不具合解決	不具合の修正が完了する
再修正決定	不具合が十分に修正されず、再修正を決定する
修正確認	不具合が修正されたかを検証する

未対応時間と修正時間



- 滞留時間を大きく二つに分類する
- 修正開始時刻
 - 不具合報告後, 最初にコメントが投稿された時刻

修正時間

- 修正時間の中でどの部分に時間を要しているか知るために
 - ある状態からある状態へ遷移するのにかかる時間を分析
- 不具合が登録されてから修正が完了するまでの作業時間
- 修正が完了してから修正が本当に完了しているかを確認する作業時間

不具合滞留時間分析手法

1. 滞留時間を「未対応時間」と「修正時間」に分割し分析するための手法
2. 複雑なプロセスが存在する不具合修正開始から修正完了までの「修正時間」を詳細に分析するための手法

未対応時間と修正時間の分割

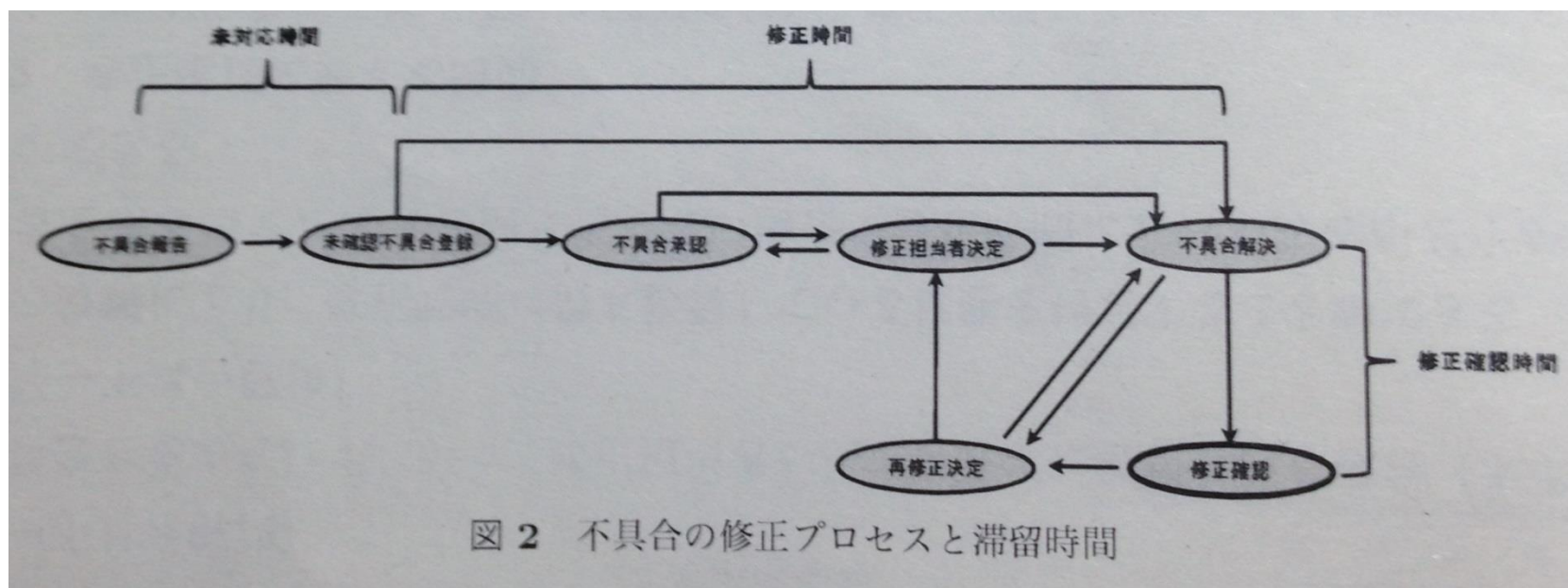
- 不具合が報告されてから修正に取り掛かるまでの時間(未対応時間)が長いことが、滞留時間を長期化させる要因の一つ
- 滞留時間を未対応時間と修正時間に分割することで、次の3種類の不具合を抽出する
 - a. 未対応時間のみが長い不具合
 - b. 修正時間のみが長い不具合
 - c. 未対応時間と修正時間が共に長い不具合

状態遷移に基づく分析

- どのように遷移して修正されたのか、またこの状態遷移にどれくらいの時間を要しているかを分析
- **修正作業**そのものに要する時間と修正作業完了後の**確認作業**に要する時間に着目する

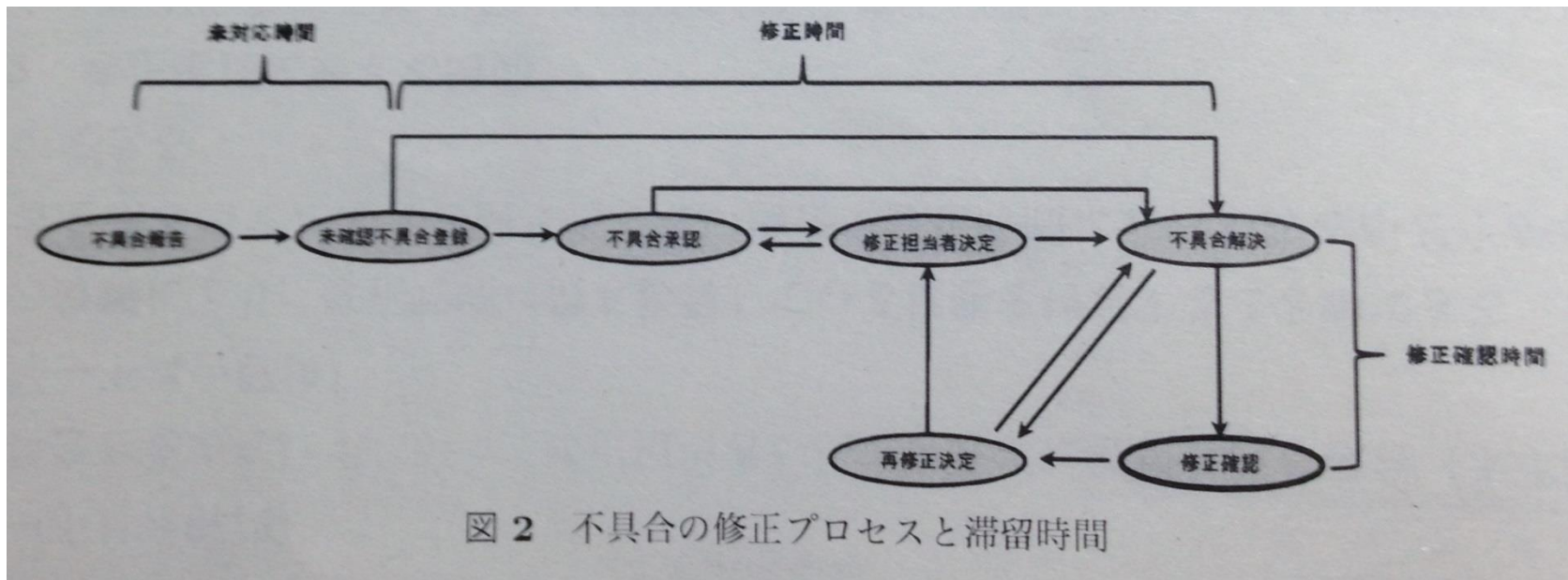
修正作業に要する時間

- 3つの不具合修正のパターン
 - 修正担当者が決定されてから不具合が解決
 - 不具合と承認された後、修正担当者が決定されずに解決
 - 不具合と認められず、かつ修正担当者が決定されずに解決



修正確認に要する時間

- 不具合の修正作業完了後、修正が正しく行われたかを確認する作業
- 2つに分類することができる
 - 不具合の修正が確認された場合
 - 不具合が十分に修正されていなかった場合



分析対象外のデータ

- 機能追加に関するデータ
 - 不具合の滞留時間の分析を目的にしている
 - 不具合管理システムは、不具合報告だけでなく機能追加要求なども同様のフォームを用いて報告できる
- 修正が行われなかった不具合データ
 - 開発者の確認で不具合と見なされなかった
 - 今後修正される予定がない

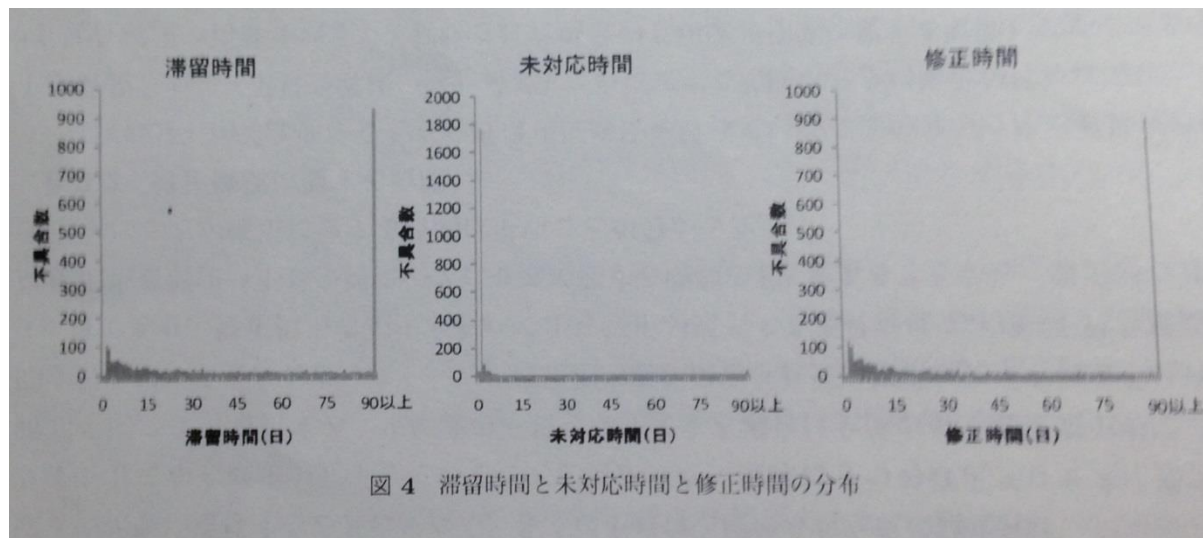
ケーススタディ

- Firefoxプロジェクトを対象とするケーススタディを行う

Firefoxの分析対象データの概要	
分析対象不具合数	2376件
データ取得開始日	2001年1月1日
データ取得終了日	2008年12月10日

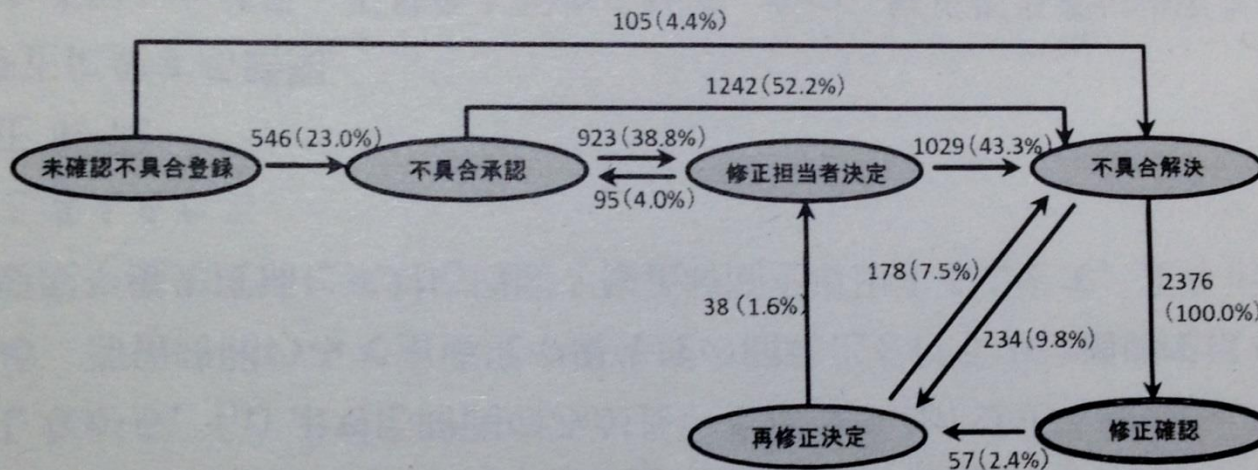
分析結果(1)

- 未対応時間と修正時間



- 全体の75%の不具合は1日以内に修正が開始されていた
- 全体の20%の不具合は1日以内に修正が開始され3か月以上要した
- 修正時間が短く、未対応時間が長い時間を要している不具合も存在

分析結果(2)



各修正プロセスにおける状態遷移件数 (数字は各状態間を遷移する不具合の件数, 括弧内の数字は分析データ数に対する割合)

- どのプロセスで長い時間を要しているか
 - 不具合承認から不具合解決に要する時間
 - 不具合解決から修正確認に要する時間

分析結果(3)

- 修正作業に要する時間
 - 不具合と認められず, 修正担当者が決定されず解決
 - 不具合と認められた後, 修正者担当者が決定されず解決
 - 不具合と認められた後, 修正担当者が決定されて解決

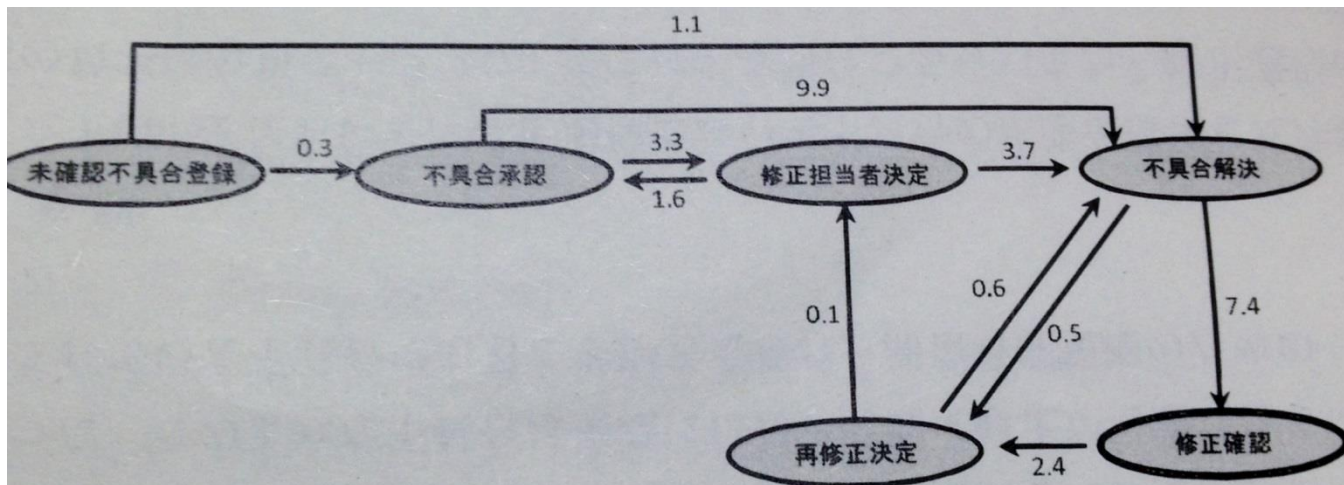


図 7 各修正プロセスに要する時間 (数字は各状態間に要する日数の中央値)

分析結果(4)

- 修正確認に要する時間
 - 不具合の約10%が再度修正を行っていた
 - 完全に修正されていないと判断された方が確認作業に要する時間が短くなる

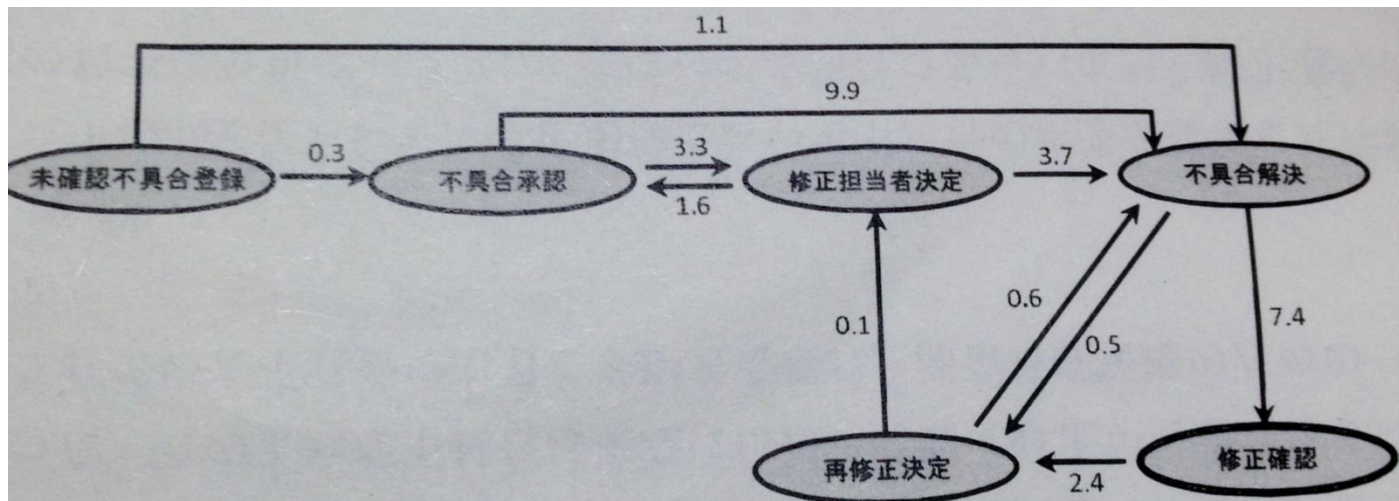


図 7 各修正プロセスに要する時間 (数字は各状態間に要する日数の中央値)

考察(1)

- 未対応時間と修正時間
 - 未対応時間の長い不具合は修正が開始されるまでに1週間以上の時間を要していることが分かった.
 - 本分析手法は修正状態を把握する際に有用である
 - 不具合を重要度や優先度別に未対応時間と修正時間を導出することで, より正確な重要度と優先度の簡略化が実現できる.

考察(2)

- 修正に要する時間
 - 不具合と認められず, かつ, 修正担当者が決定されずに解決される場合の修正に要する時間が最も短い
 - 不具合は修正担当者を決定した場合の方が修正が早く解決されることが分かった.
- 修正確認に要する時間
 - 全体の10%程度再修正が行われたことが分かった.
 - 修正プロセスに要する時間のうち修正が確認される作業時間に長い時間を占めていることが分かった.

まとめ

- 不具合の滞留時間の短縮を目的として不具合修正のどの部分がボトルネックとなっているかを理解するために不具合管理システム利用時の不具合の滞留時間を分析するための手法
 - 未対応時間と修正時間を分割して分析し修正状態の把握
 - 修正プロセスを分類したことで滞留時間を短縮するための修正手法を明らかにすることができた
 - 修正確認作業を分類したことで修正確認作業の状態を把握することができた