住友電工情報システム

SPI Japan 2016 発表資料

# ビデオ撮影及び画面キャプチャによる測定・分析プロセスの改善

住友電工情報システム株式会社
第一システム部　第二開発グループ
原田　大輔

2016年10月12日

# 目次

1. 背景(行動分析の実施に至るまで)
2. 行動分析の実施方法
3. 行動分析の実施結果
4. まとめ

# 1.背景
# (行動分析の実施に至るまで)

# 背景

- 品質向上のために自動テストを導入
  - 開発手順(プロセス)の決定
  - 自動テストの教育実施

成果　品質は向上

問題　生産性は低下

課題　なぜ生産性が下がったか原因が分からず、対策を実施できない。

# 課題解決に向けて

**課題**

なぜ生産性が下がったか原因が分からず、
対策を実施できない。

**対策**

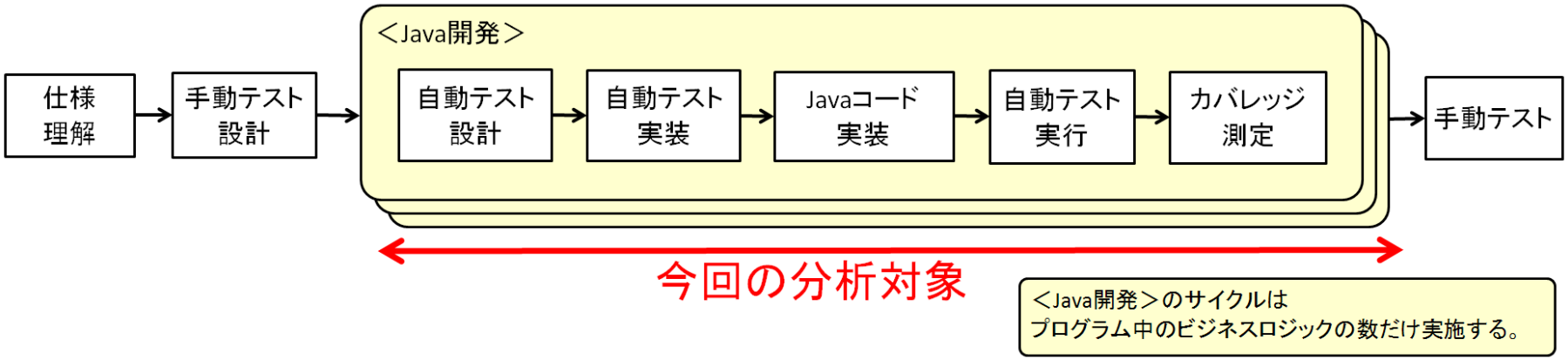
「行動分析」を実施する。

例)　主婦が台所で調理を行う様子をビデオ撮影(測定)。
　　前後30cmだけ移動している事が判明(分析)
　　⇒　より使いやすいキッチンの開発に成功[1]。

「どのような行動が」「どのぐらいの時間」
行われているかが明らかになり、対策を実施すべき
ポイントが明確になる。

# 2.行動分析の実施方法

# プログラム開発プロセスと分析対象




- 手動テスト
  ブラウザ上で実施するテスト。
  例)画面遷移の確認…など。
- 自動テスト
  JUnitによるビジネスロジックのテスト。
  例)拠点毎に入力内容の整合性チェック…など。

# 行動分析の実施手順

①ビデオ撮影の実施

②行動の特定

③開発プロセスの特定

④作業時間の分析

# ①ビデオ撮影の実施

より詳細な行動を知るために、次の二種類を撮影する。

| 行動の種類 | 測定方法 | 行動例 |
|---|---|---|
| PC上で行う行動 | 画面キャプチャ | ・自動テストの実装<br>・プログラムの問題点調査<br>(コードリーディング)　…等 |
| PC外での行動 | ビデオ撮影 | ・紙資料の閲覧<br>・紙にフローチャートを書いての<br>ロジック検討　…等 |

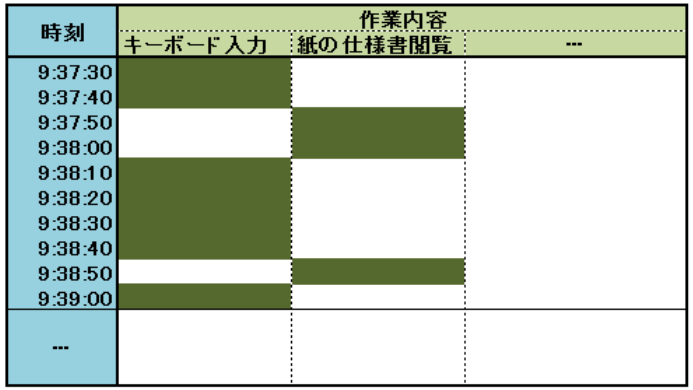
ビデオ撮影のみ　　…　閲覧している画面の詳細が分からず
画面キャプチャのみ　…　画面外の行動が分からず
　⇒　二種類を組み合わせる事で、詳細な行動が分かる。

# ①ビデオ撮影の実施

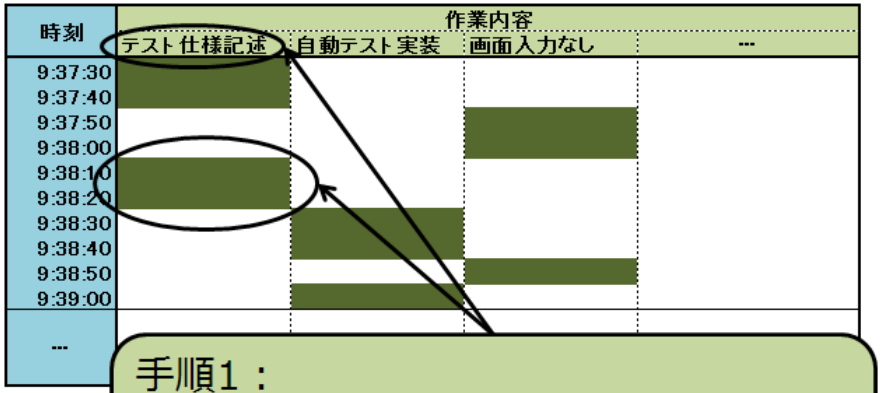
(当日のみ)

# ②行動の特定(行動分析表の作成)

＜ビデオ撮影結果＞

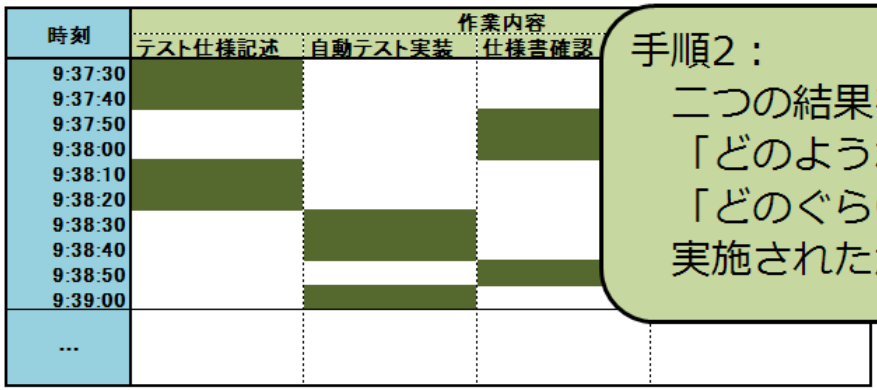



＜画面キャプチャ結果＞




手順1：
行動を特定し、その行動をしていた時間を塗りつぶす。

＜統合結果＞行動分析表




手順2：
二つの結果を統合し、
「どのような行動が」
「どのぐらいの時間」
実施されたかを明らかにする。

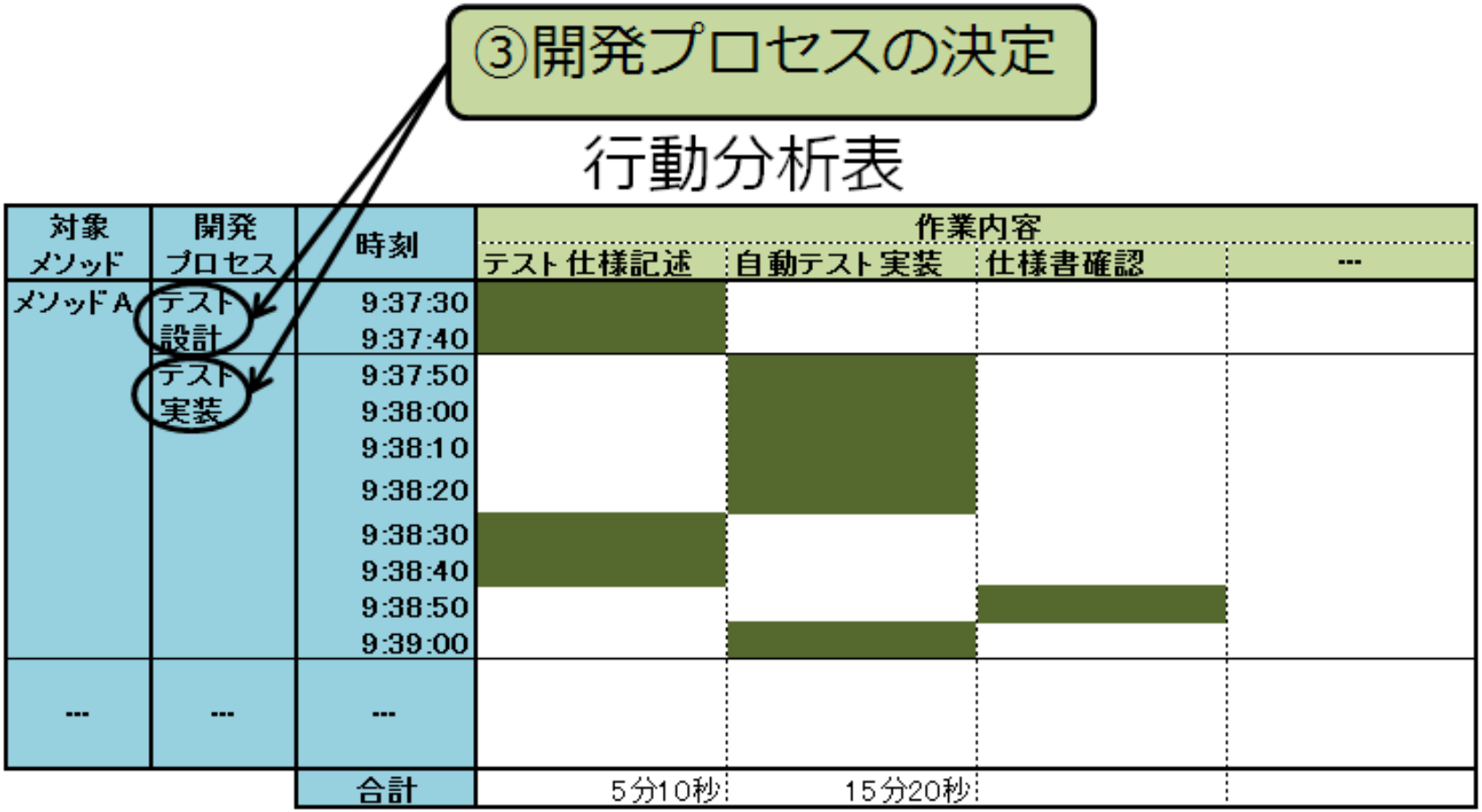
# ③開発プロセスの決定

各行動がどのプロセスで行われたかを判断するため、
各プロセスの開始基準・終了基準を定める。

| 開発プロセス | 判定基準 | |
|---|---|---|
| | 開始基準 | 完了基準 |
| 自動テスト設計 | 自動テストソースコードに<br>テスト仕様を記述し始めた時。 | 自動テストソースコードに<br>テストコードを記述し始めた時。 |
| 自動テスト実装 | 自動テストソースコードに<br>テストコードを記述し始めた時。 | プログラムロジックをJavaの<br>ソースコードに記述し始めた時。 |
| Javaコード実装 | プログラムロジックをJavaの<br>ソースコードに記述し始めた時。 | 1回目の自動テストを<br>実行した時。 |
| 自動テスト実行 | 1回目の自動テストを<br>実行した時。 | 1回目のカバレッジ測定を<br>実行した時。 |
| カバレッジ測定 | 1回目のカバレッジ測定を<br>実行した時。 | メソッド開発完了基準を<br>満たした時。 |

# ③開発プロセスの決定

前ページの定義に基づき、開発プロセスを決定する。

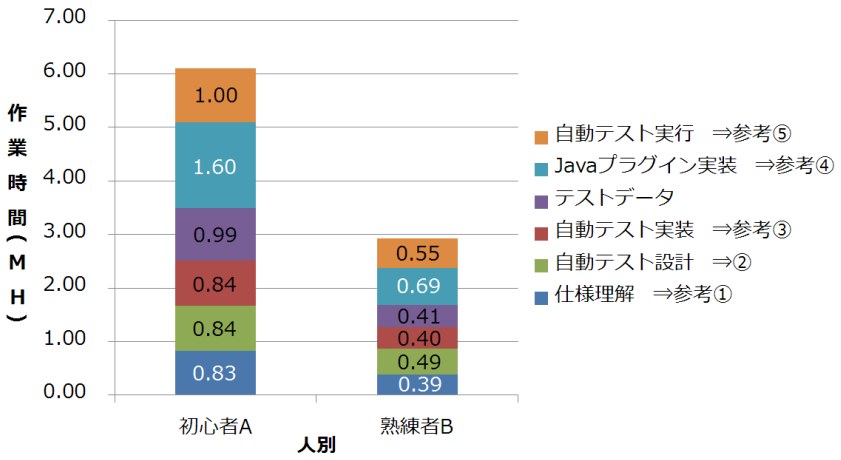


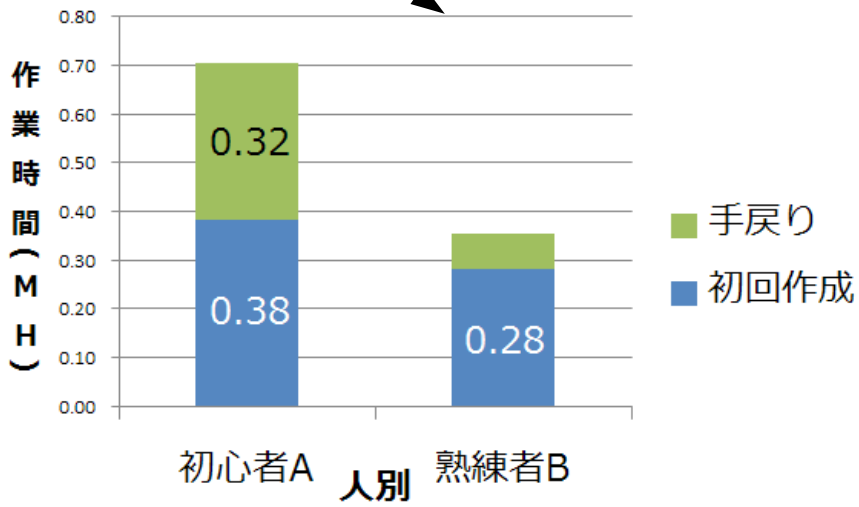
# ④作業時間の分析

行動分析表を基に、分析に必要なグラフなどを作成する。

行動分析表

| 対象メソッド | 開発プロセス | 時刻 | 作業内容 テスト仕様記述 | 自動テスト実装 | 仕様書確認 | … |
|---|---|---|---|---|---|---|
| メソッドA | テスト設計 | 9:37:30 | | | | |
| | | 9:37:40 | | | | |
| | テスト実装 | 9:37:50 | | | | |
| | | 9:38:00 | | | | |
| | | 9:38:10 | | | | |
| | | 9:38:20 | | | | |
| | | 9:38:30 | | | | |
| | | 9:38:40 | | | | |
| | | 9:38:50 | | | | |
| | | 9:39:00 | | | | |
| … | … | … | | | | |



# 3.行動分析の実施結果

# 今回の検証内容

**初心者A、熟練者B**の2名が同じプログラムを開発し、
自動テスト実装～カバレッジ測定までの行動差を分析する。

## 画面イメージ

| 契約先C | 12345 | 住友電工情報システム株式会社 |
|---|---|---|

| □契約 | ITサービスC | ITサービス表示名 | □契約年月 |
|---|---|---|---|
| □ | S001 | LAN使用料 | |
| □ | S002 | NW使用料 | |
| □ | S003 | ML使用料 | |
| □ | S004 | XXサービス | |
| □ | S005 | YYサービス | |

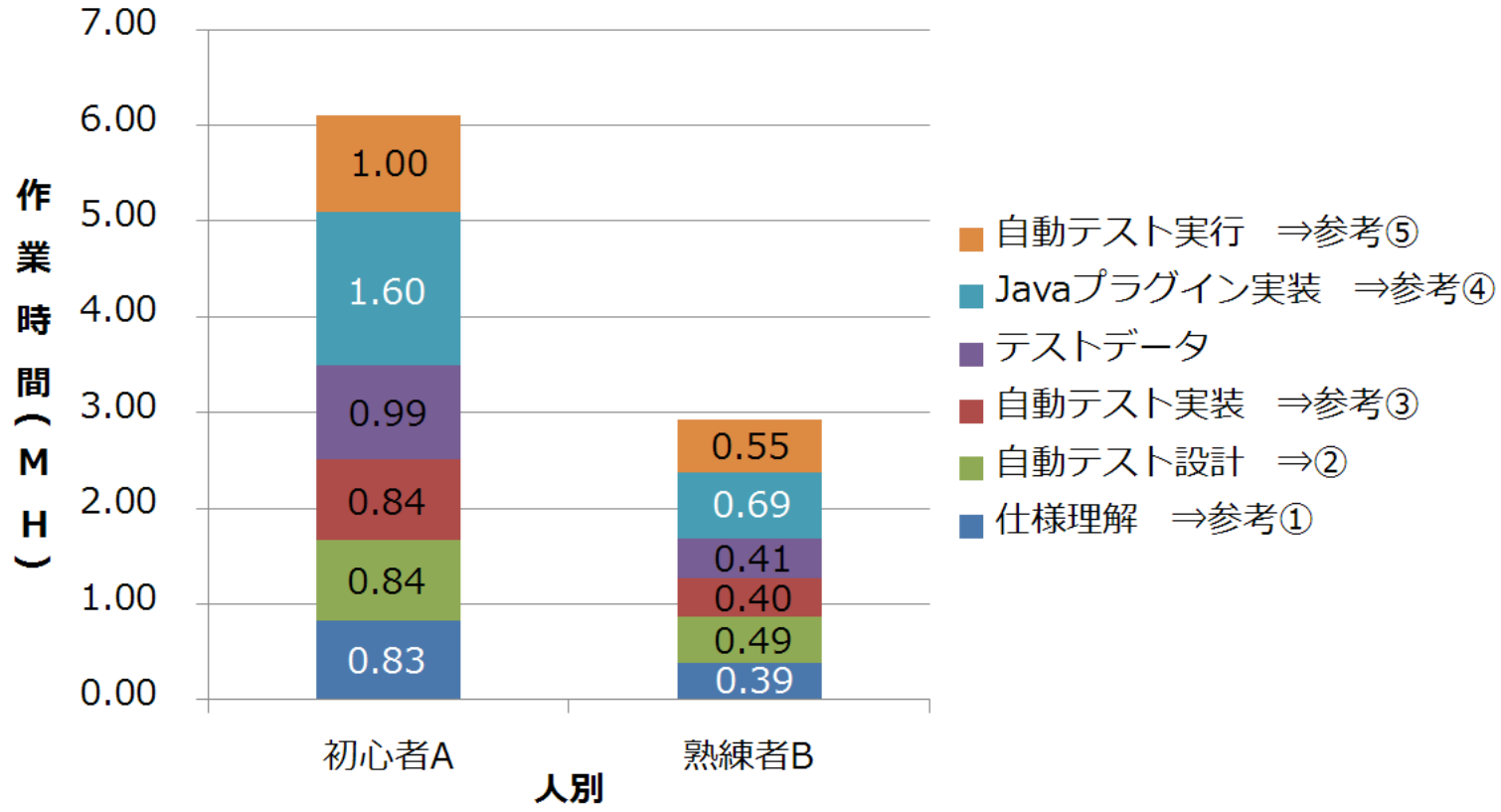
## 機能説明

契約先一件に対して、
複数のITサービスを
登録する。

## 実装内容

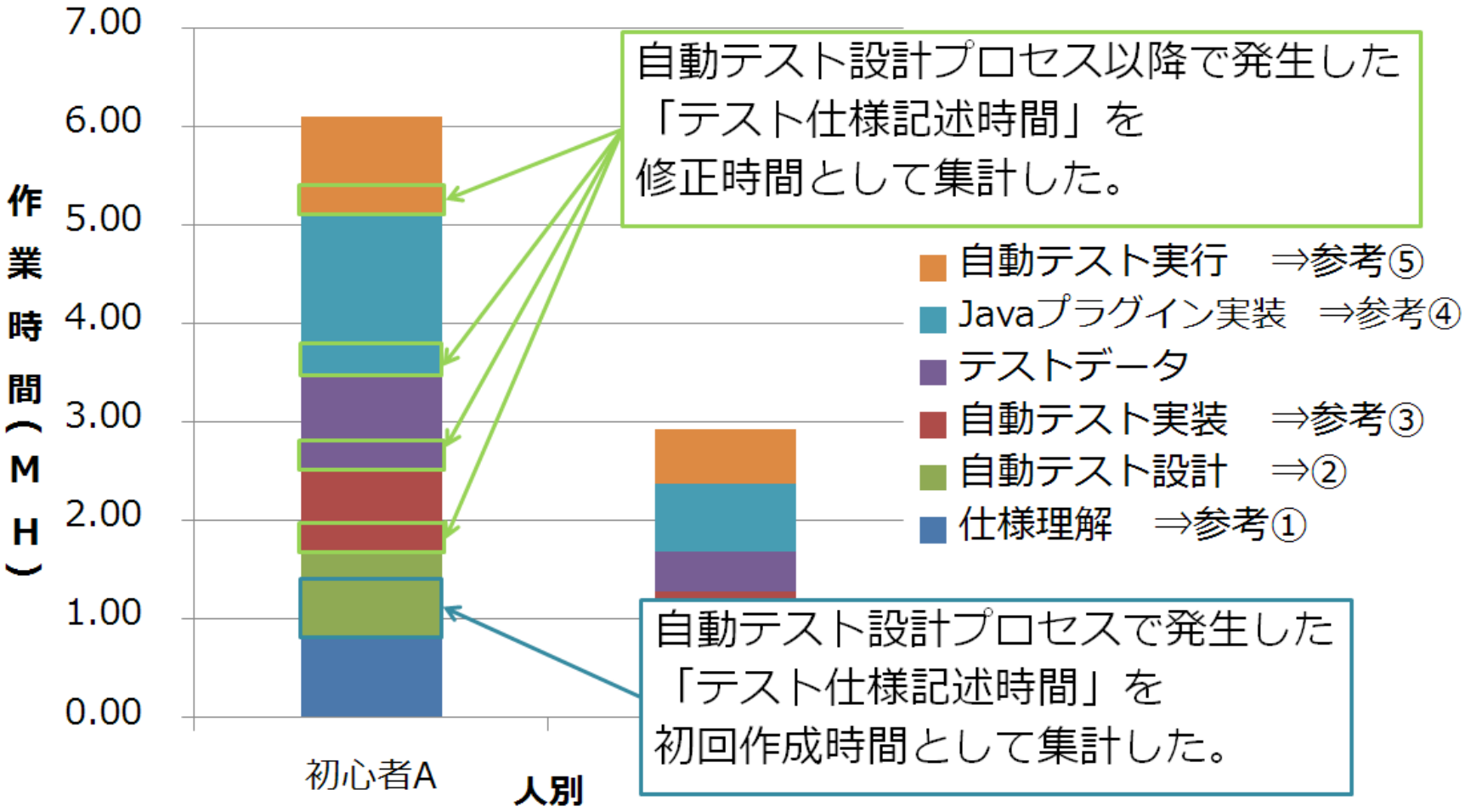
| メソッド | 内容 | ステートメント数 |
|---|---|---|
| A | 既にデータが登録済かの入力値チェック。 | 20 |
| B | 複数項目間のデータ整合性チェックと、エラーメッセージの加工処理。 | 70 |
| C | ヘッダ1行に対して明細N件が登録されているかのチェック。 | 30 |
| 合計 | | 120 |

# 分析結果　作業時間内訳

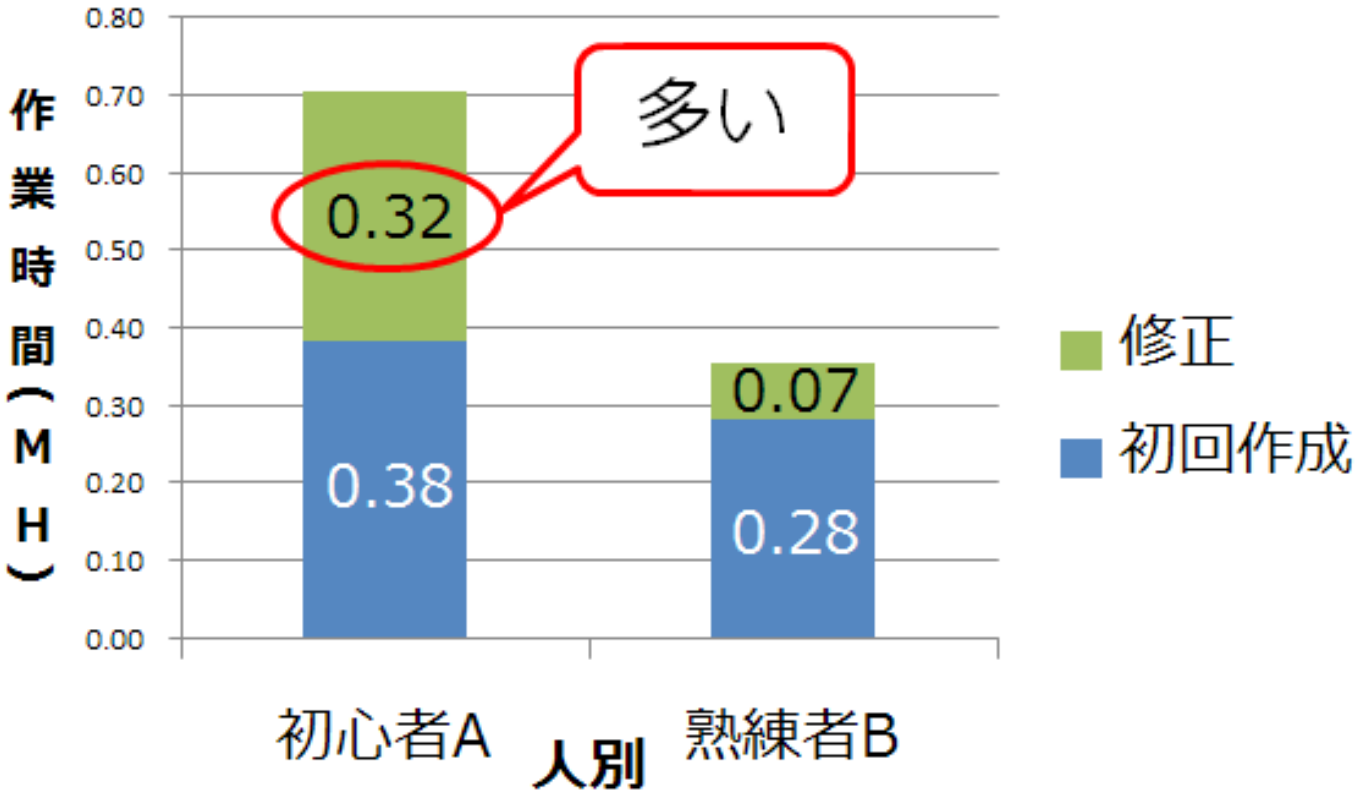



特定のプロセスで生産性が低下している訳ではなく、
全てのプロセスで約二倍の作業時間を要している。

# 自動テスト仕様記述時間の「初回作成」「修正」集計方法

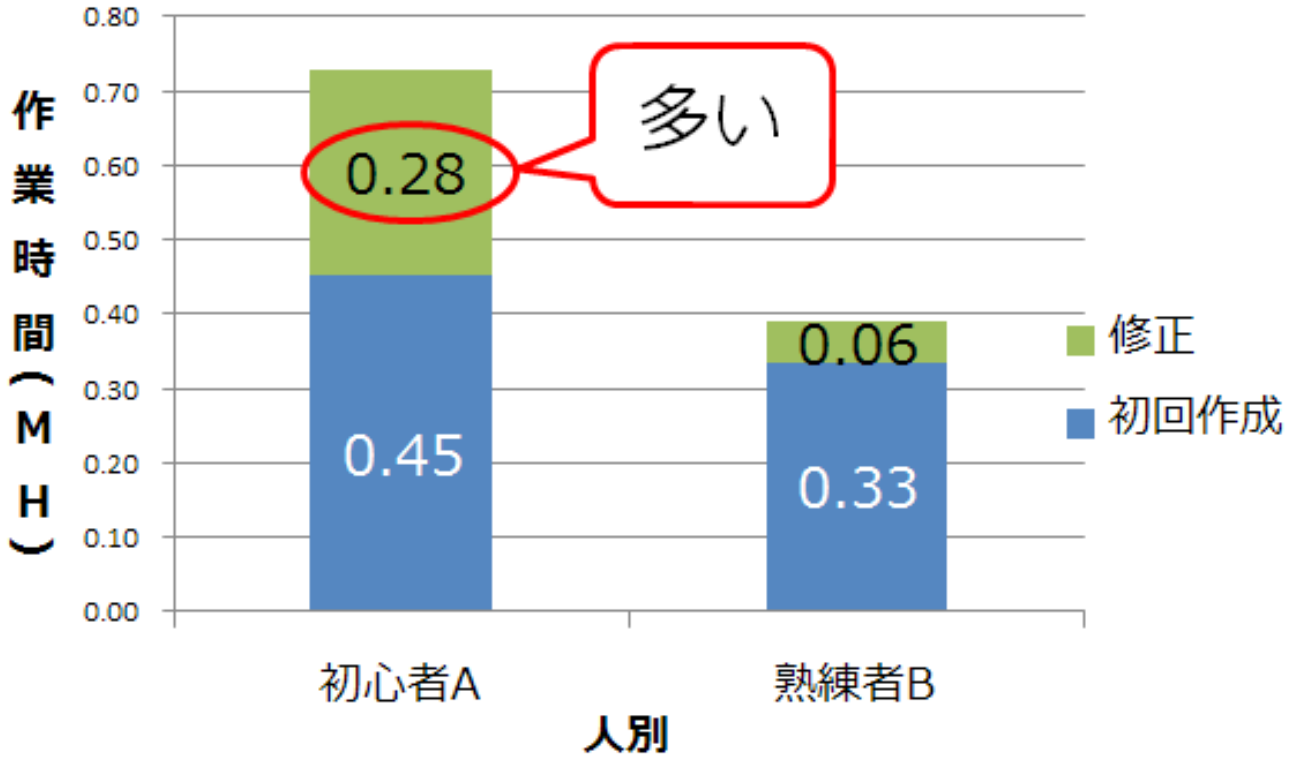


# 分析結果　自動テスト仕様記述時間の内訳




初回作成時間はほぼ同じだが、修正時間が長い。

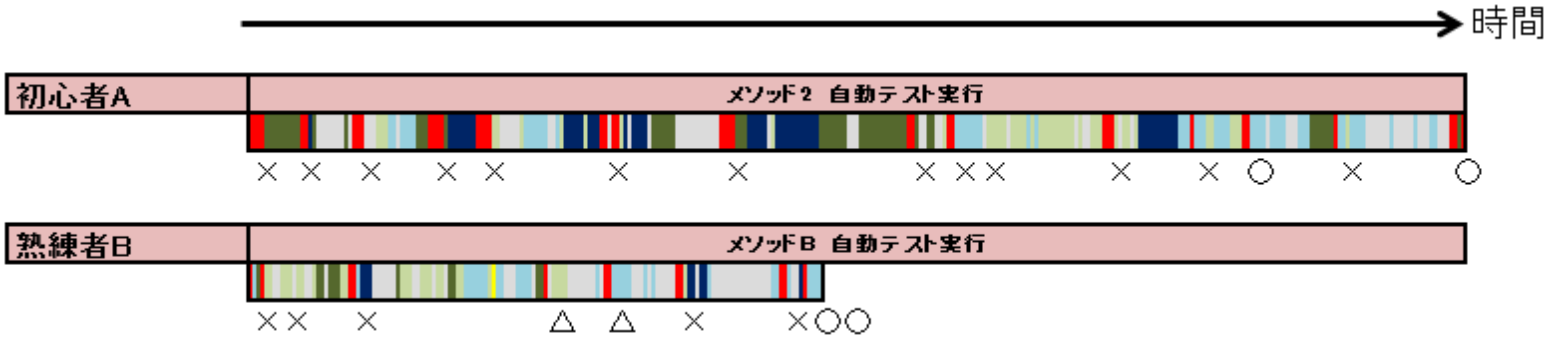
# 分析結果　自動テスト(JUnit)実装時間の内訳




自動テスト仕様記述と同じく、修正時間が長い。

⇒　修正時間が長い原因を調べるためには
さらなる追加データと分析が必要。

# 分析結果　自動テスト実行～カバレッジ測定における行動の差




合計時間のみならず、
行動順も見える図を作成。

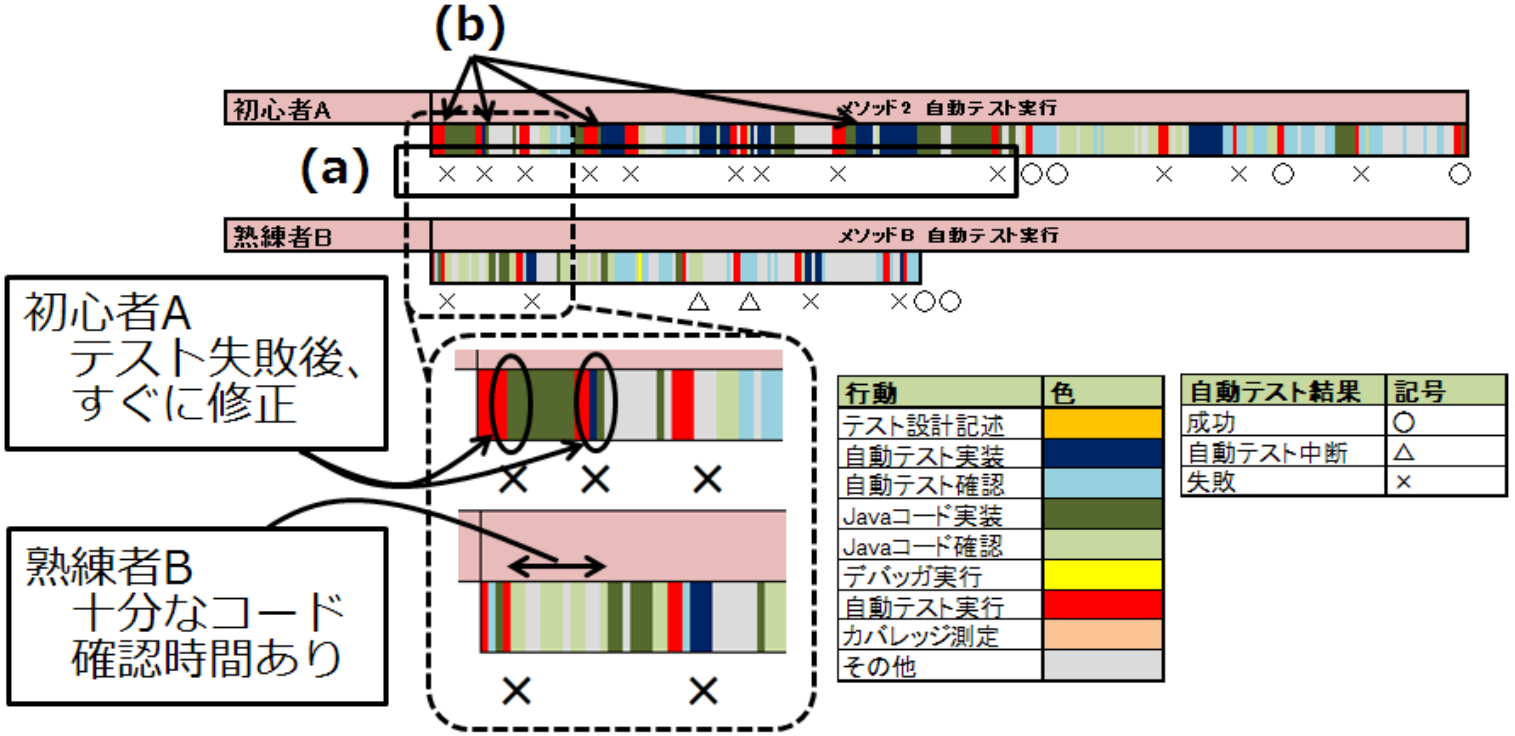
| 行動 | 色 |
|---|---|
| テスト設計記述 | |
| 自動テスト実装 | |
| 自動テスト確認 | |
| Javaコード実装 | |
| Javaコード確認 | |
| デバッガ実行 | |
| 自動テスト実行 | |
| カバレッジ測定 | |
| その他 | |

| 自動テスト結果 | 記号 |
|---|---|
| 成功 | ○ |
| 自動テスト中断 | △ |
| 失敗 | × |

自動テスト実行結果を
データとして追加。

作業時間の差のみならず、
両者の行動の違いも見えてきた。

# 分析結果　自動テスト実行～カバレッジ測定における行動の差




| 行動 | 色 |
|---|---|
| テスト設計記述 | |
| 自動テスト実装 | |
| 自動テスト確認 | |
| Javaコード実装 | |
| Javaコード確認 | |
| デバッガ実行 | |
| 自動テスト実行 | |
| カバレッジ測定 | |
| その他 | |

| 自動テスト結果 | 記号 |
|---|---|
| 成功 | ○ |
| 自動テスト中断 | △ |
| 失敗 | × |

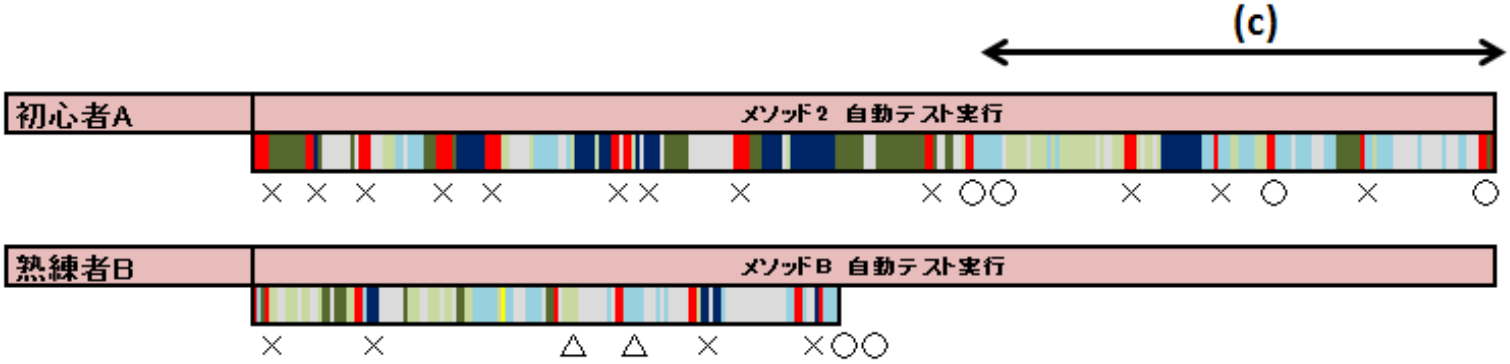
【初心者の特徴】

(a)自動テスト実行回数、失敗回数が多い。

(b)コード確認時間が短く、すぐに修正に着手している。

⇒　修正が場当たり的。

# 分析結果　自動テスト実行～カバレッジ測定における行動の差




後にヒアリングを実施した。

① カバレッジ測定において
　自動テストが通らない
　個所があった。
② 自動テストを追加した。
③ 自動テストがNGになった。
④ プログラム本体を修正した。

| 行動 | 色 |
|---|---|
| テスト設計記述 | |
| 自動テスト実装 | |
| 自動テスト確認 | |
| Javaコード実装 | |
| Javaコード確認 | |
| デバッガ実行 | |
| 自動テスト実行 | |
| カバレッジ測定 | |
| その他 | |

| 自動テスト結果 | 記号 |
|---|---|
| 成功 | ○ |
| 自動テスト中断 | △ |
| 失敗 | × |

【初心者の特徴】
(c)自動テスト成功後もソースコードを修正している。

# 分析結果まとめ

## 修正時間が多い

- テスト仕様記述の手戻り時間
- テスト実装(JUnit)の手戻り時間

さらなる原因調査

## 場当たり的修正

## 自動テスト成功後の修正

# 分析結果に基づいた対策案

- デバッグ時の修正は、十分にコードを確認して行う方が良い事を周知する。

**【生産性が高い人の特徴】**

原因を見つけ、それによって誤った結果が起こっている事を頭の中で動かして確認している。

VS

**【生産性が低い人の特徴】**

原因かもしれない個所をすぐに修正し、実際に動かして確認している。

- 生産性が低い人の特徴的な行動を監視・指導する。
  - 場当たり的修正
  - 自動テスト成功後も修正

# 4.まとめ

# まとめ

## ビデオ撮影による行動分析手法の確立

今回の方法でPG開発以外の行動も分析可能である。

(※)ただし、原因の特定～対策案の作成には多少の試行錯誤が必要。

## ビデオ撮影時間の２～３倍で分析が可能

| 被験者 | PG時間(MH) | 行動分析時間(MH) |
|---|---|---|
| 初心者A | 7 | 15 |
| 熟練者B | 3.5 | 8 |

(※)参考文献[2]では3倍。

## 今後の課題

対策案の実施。

# 参考文献

[1] 書籍『データサイエンティスト最前線』(2015),
日経BP社.
[2] クーマー，ヴィジェイ(2015)『101デザインメソッド』
渡部典子訳，英治出版.

ご清聴いただき
誠にありがとうございました。