HITACHI
Inspire the Next

SQiPシンポジウム 2018

アジャイルプラクティスを導入した開発における品質メトリクスの提案

2018/09/14

株式会社日立製作所 ITプロダクツ統括本部
プロダクツQA本部

松田 元輝　genki.matsuda.fm@hitachi.com

© Hitachi, Ltd. 2018. All rights reserved.

HITACHI
Inspire the Next

Contents

1. 自己紹介
2. アジャイルプラクティス導入
3. バグ摘出の傾向分析
4. 前倒し率の提案
5. 結論と今後の課題

© Hitachi, Ltd. 2018. All rights reserved.

HITACHI
Inspire the Next

1. 自己紹介

© Hitachi, Ltd. 2018. All rights reserved.

1 -1. 自己紹介

HITACHI
Inspire the Next

松田 元輝（まつだ げんき）

所属

日立製作所
サービス&プラットフォームビジネスユニット
プロダクツQA本部

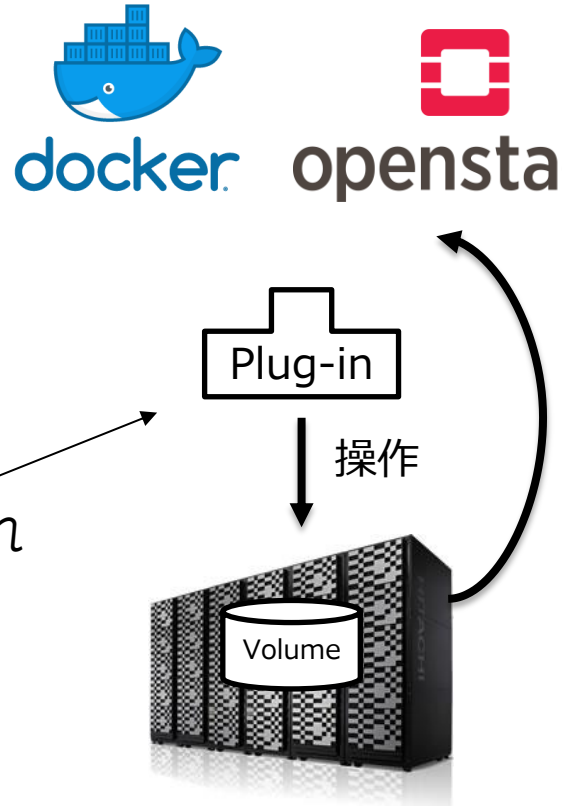
業務

レビュー、テスト、テスト自動化、品質や進捗の可視化

担当製品

日立ストレージとクラウドOSSを連携させるPlug-in

これ



SQiPとの関わり

2017年度SQiP研究会
演習コースⅡ「ソフトウェアメトリクス」に参加
ソフトウェアメトリクス分析を始める

*OpenStack®の文字表記とOpenStackのロゴは，米国とその他の国におけるOpenStack Foundationの登録商標/サービスマークまたは商標/サービスマークのいずれかであり，OpenStack Foundationの許諾を得て使用しています。日立製作所は，OpenStack FoundationやOpenStackコミュニティの関連企業ではなく、また支援や出資を受けていません。

*DockerおよびDockerロゴはDocker社の米国またはその他の国における商標または登録商標です.

© Hitachi, Ltd. 2018. All rights reserved.

HITACHI
Inspire the Next

2. アジャイルプラクティス導入

© Hitachi, Ltd. 2018. All rights reserved.

2-1. アジャイルプラクティスを導入したモチベーションと工夫

HITACHI
Inspire the Next

モチベーション

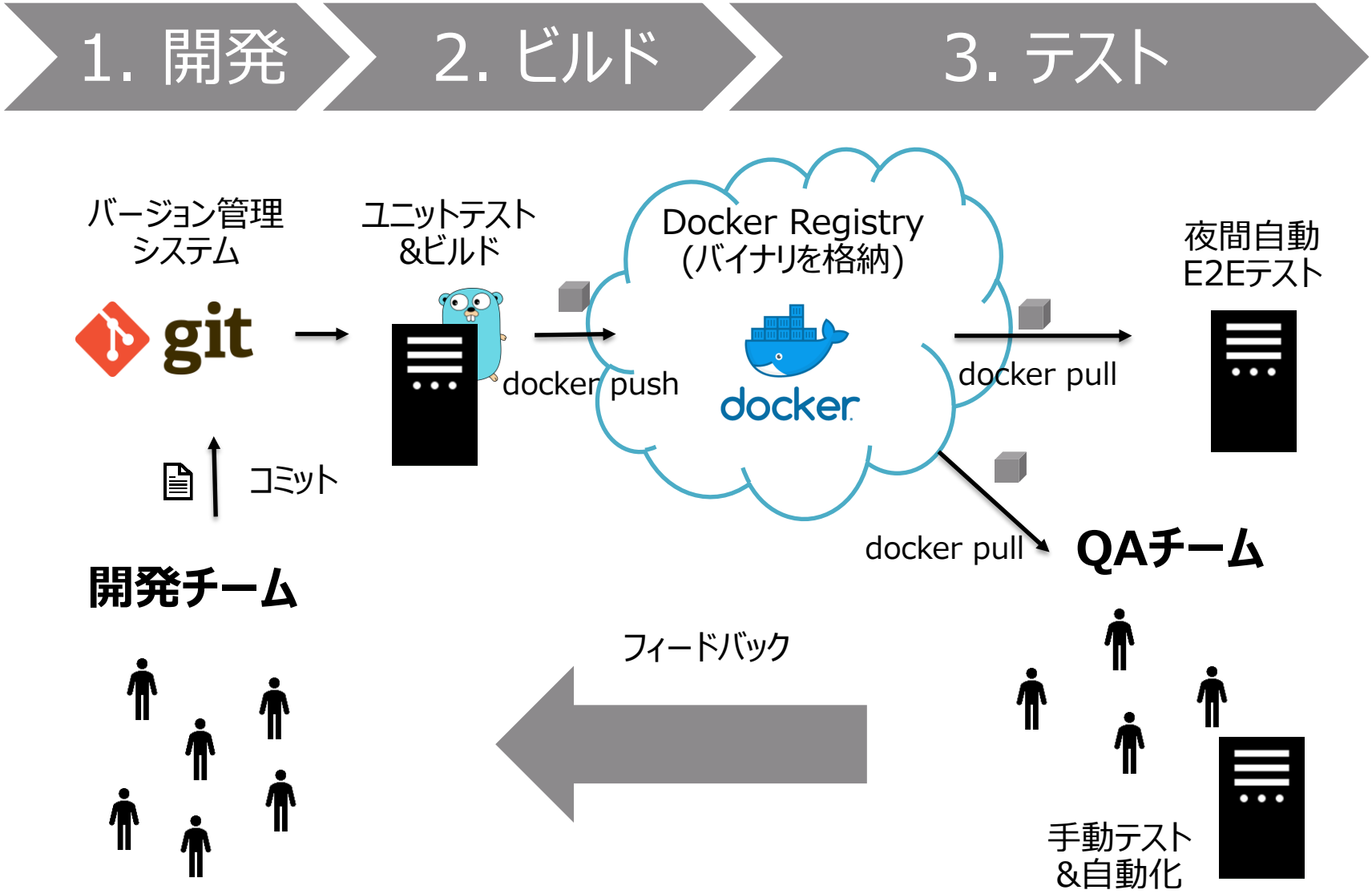
- 変化に強くしたい。
- 効率を上げたい。
- 早めにリスクを除去したい。（結合してみたらダメだった。。。を回避）
- 価値/品質を上げたい。

工夫（導入したアジャイルプラクティス）

- 2週間などの単位(スプリント)でイテレーティブに開発
- **1スプリントでE2Eテストまで**やり切りフィードバックを早める
- ビルドを**自動化**して、毎コミットごとにバイナリを生成
- ユニットテストを**自動化**し、毎コミットごとにテスト
- E2Eテストも**自動化**して、毎日夜間にテスト
- **QAも設計初期段階から参画**し、仕様検討とテストを実施

© Hitachi, Ltd. 2018. All rights reserved.

2-2. 開発環境と体制



*GitおよびGitロゴはSoftware Freedom Conservancy, Inc.の米国またはその他の国における商標または登録商標です.

© Hitachi, Ltd. 2018. All rights reserved.

2-3. 製品事業の開発

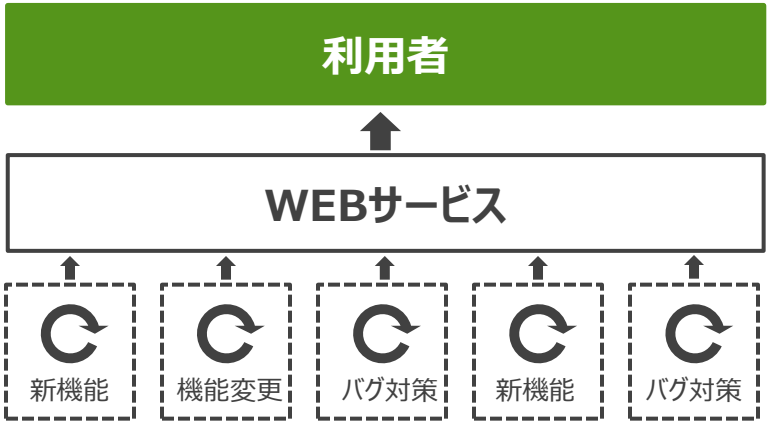
HITACHI
Inspire the Next

自組織の製品事業では、各製品のリリースは3ヵ月～6ヵ月に一度なので、スプリントごとにリリースはせず、**複数スプリント後にリリース**する。また、不特定多数のお客様へプログラムを提供するため、**一度リリースするとパッチ適用が難しい。**

そのため、製品開発に合った開発スタイルを追求した。

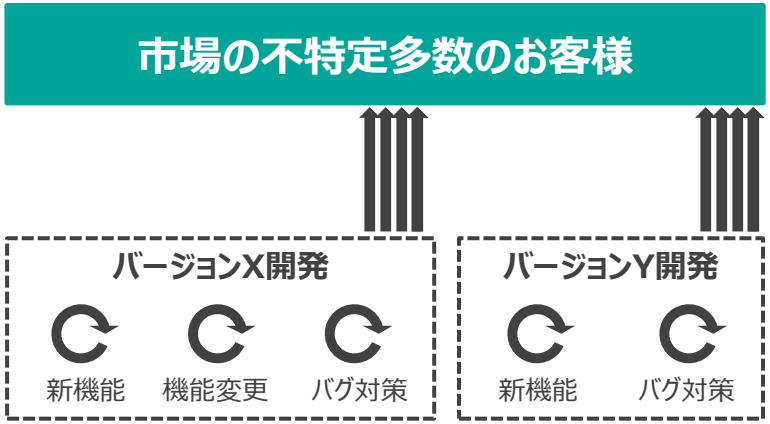
□ サービス系の開発

- スプリント、機能ごとにリリース
- 発生したバグにも俊敏に対応・リリース



□ 自組織における製品事業の開発

- リリースは3～6カ月に1回程度。
- バグがあったからといって、日立の都合のみでアップグレード/パッチ適用をすることができない。



© Hitachi, Ltd. 2018. All rights reserved.

2-4. 製品事業でアジャイルをする工夫

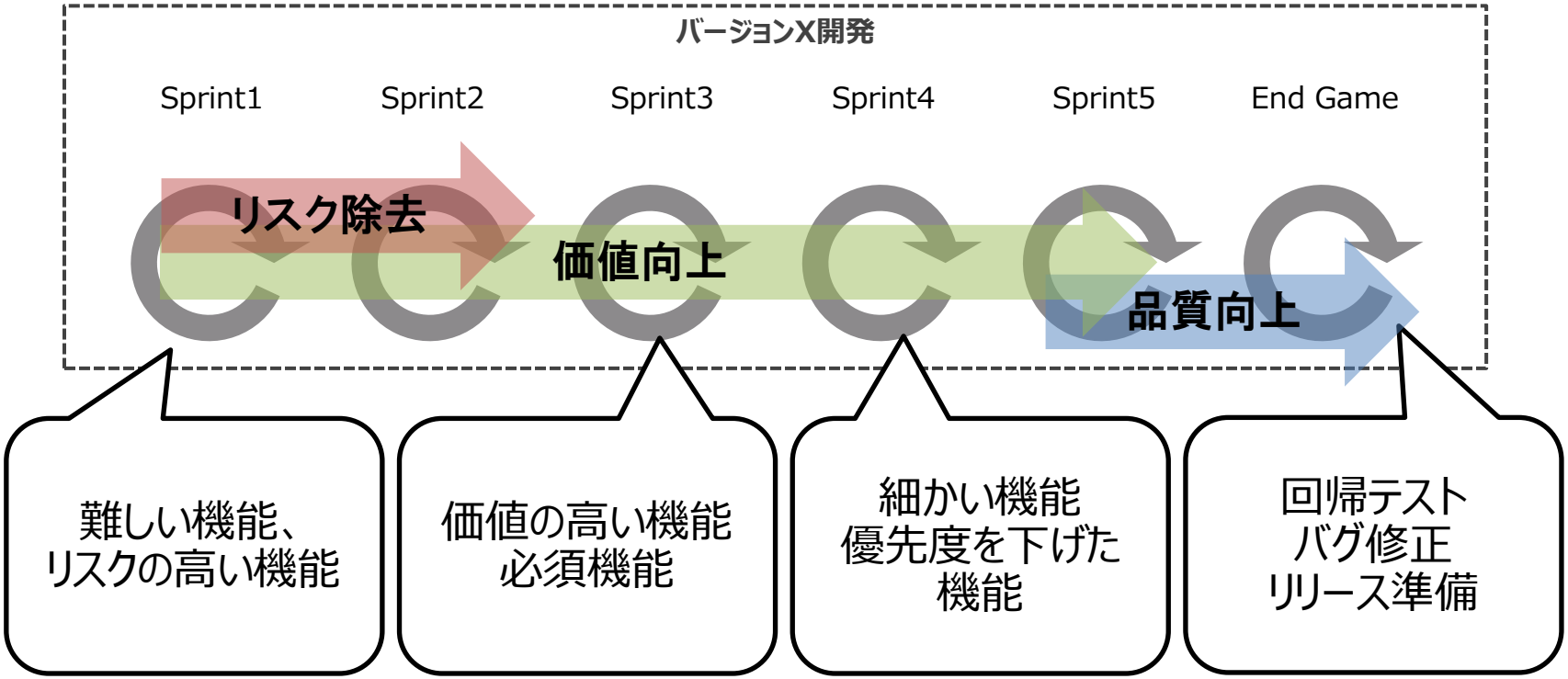
HITACHI
Inspire the Next

複数スプリントでリリースするので、最終スプリントまでに価値を最大化すればよい。

そのため、最初はリスク除去に力を入れる。

また、一度リリースすると回収が難しいので、リリース時には高品質が求められる。

最後のスプリントは品質向上に専念する。



© Hitachi, Ltd. 2018. All rights reserved.

2-5. 課題

HITACHI
Inspire the Next

しかし、いくつかのプロジェクトでは、

市場不具合を多く出してしまっていた。

そこで、市場不具合との関係が深そうな

E2Eテストのバグを分析してみた。

© Hitachi, Ltd. 2018. All rights reserved.

HITACHI
Inspire the Next

3. バグ摘出の傾向の分析

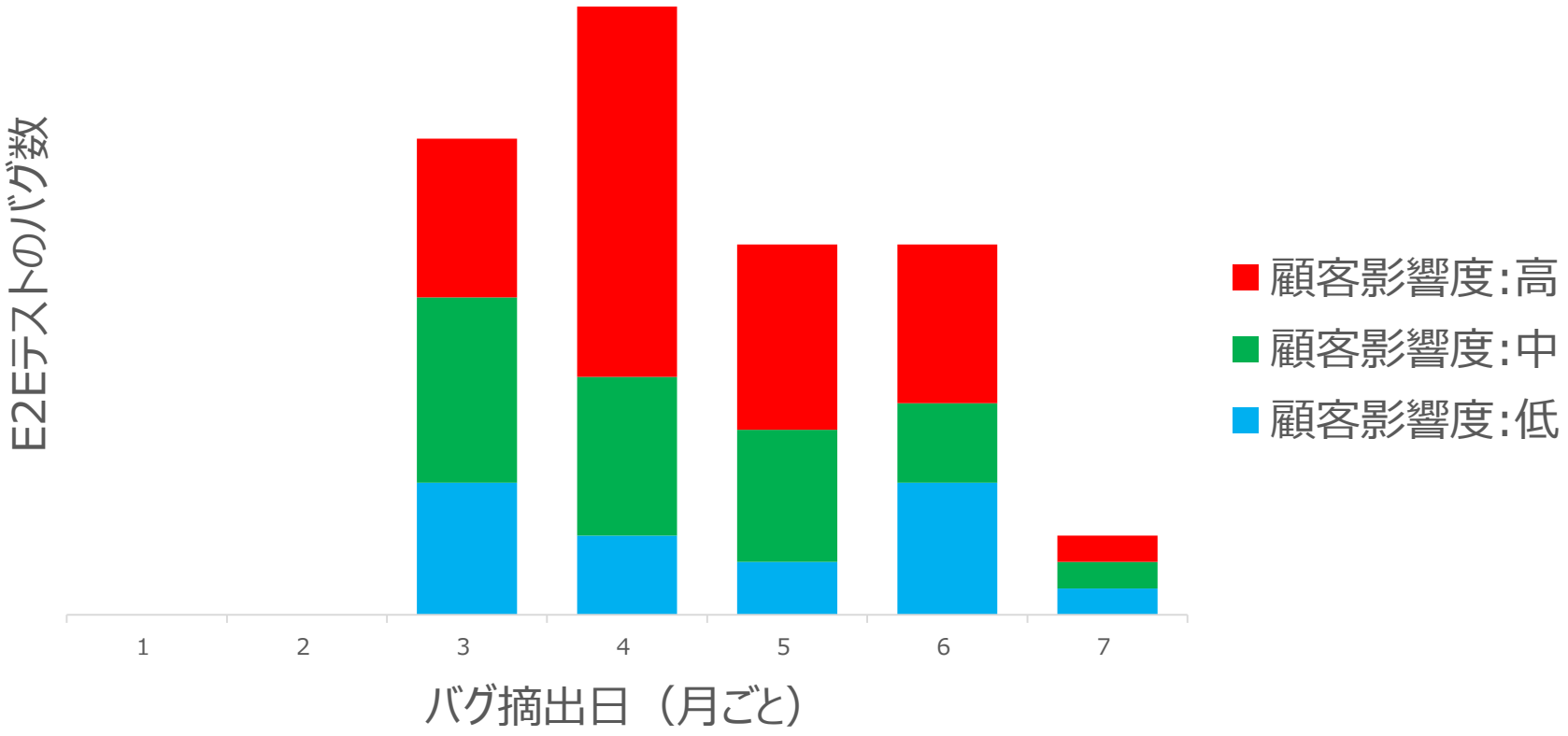
© Hitachi, Ltd. 2018. All rights reserved.

3-1. **市場不具合なし**プロジェクトのバグ摘出傾向

HITACHI
Inspire the Next

プロジェクトG

毎日自動E2Eテスト ＋ QAが開発の初期段階からテスト



© Hitachi, Ltd. 2018. All rights reserved.

3-2. **市場不具合あり**プロジェクトのバグ摘出傾向

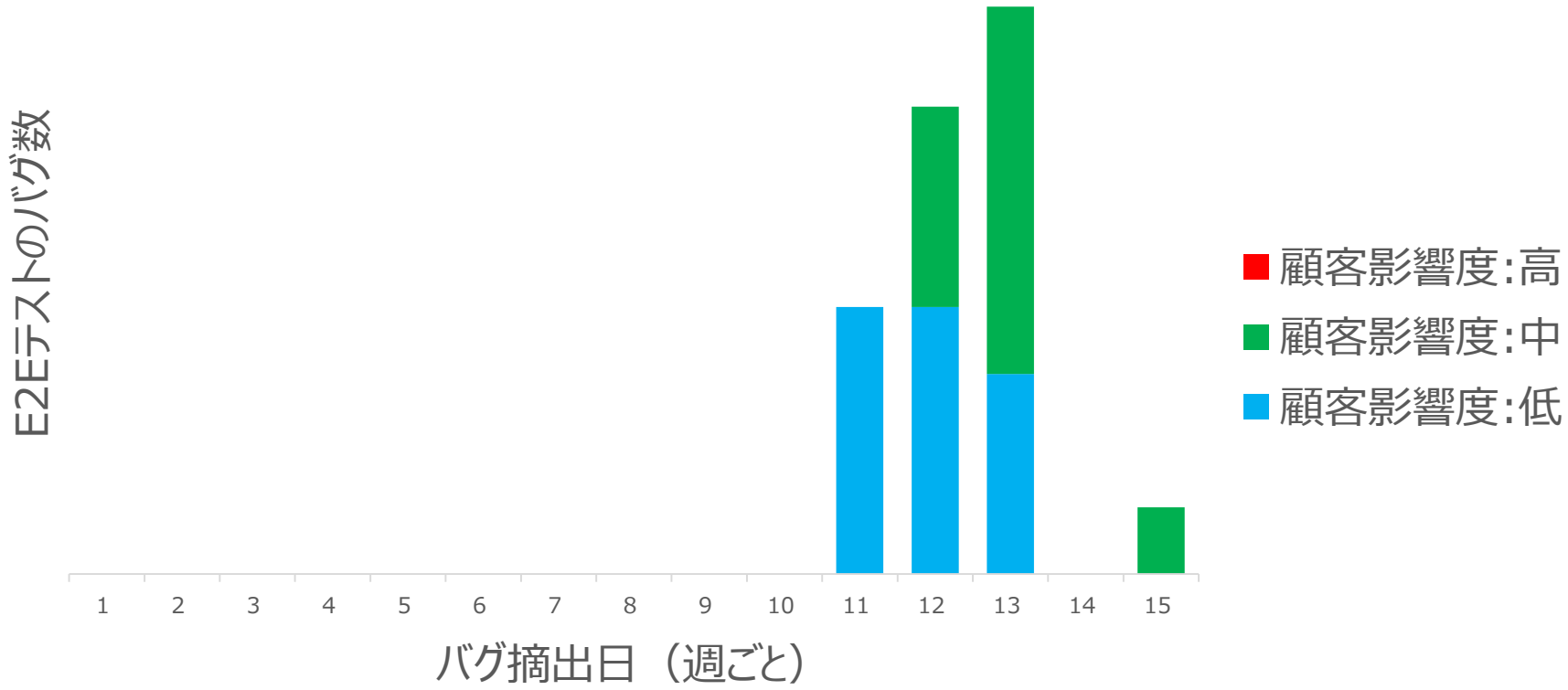
HITACHI
Inspire the Next

プロジェクトA

E2Eテストの開始が遅れた。

出荷先限定でリリースしたが、

すぐに品質改善版をリリース

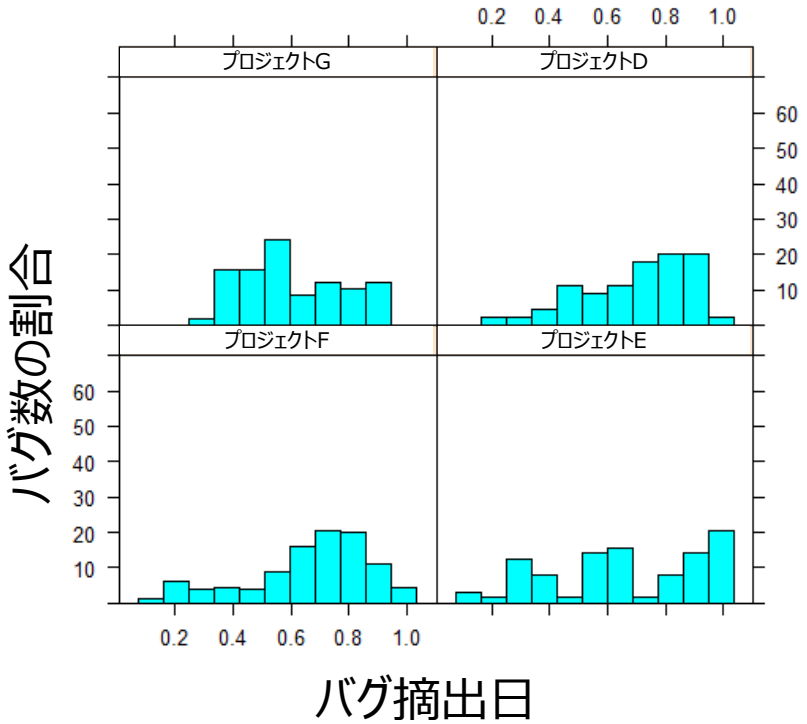


© Hitachi, Ltd. 2018. All rights reserved.

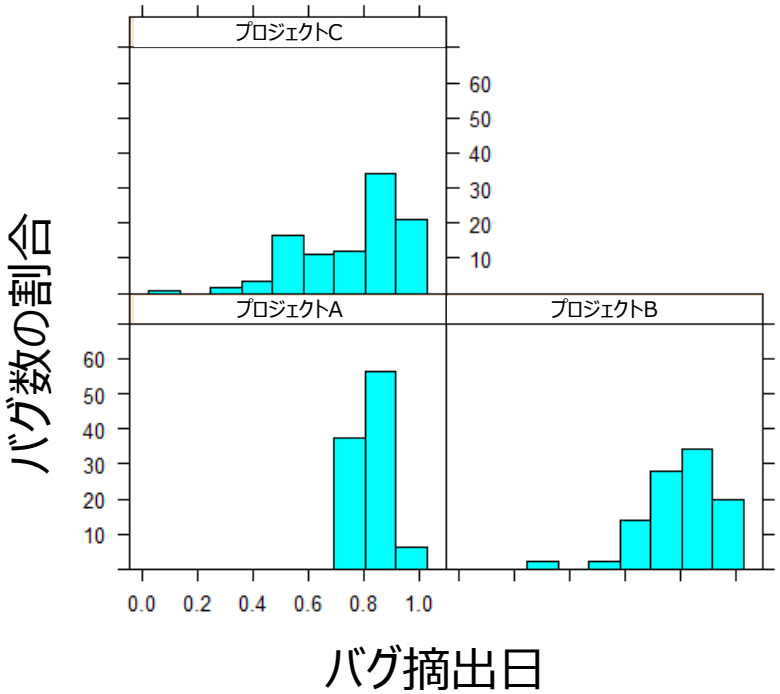
3-3. 各プロジェクトのバグ摘出傾向

HITACHI
Inspire the Next

市場不具合なし
バグ前倒せてる



市場不具合あり
バグ前倒せてない



そこで、<u>バグを前倒せていれば、市場不具合が少ない</u>という仮説を立てた。

現状だと、グラフでしか説明できていないため、前倒し具合を定量的に表し、

仮説を検証する。

© Hitachi, Ltd. 2018. All rights reserved.

4. 前倒し率の提案

© Hitachi, Ltd. 2018. All rights reserved.

4-1. 従来の前倒し率

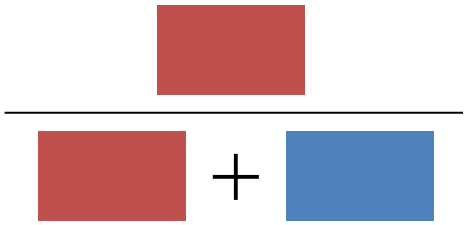
HITACHI
Inspire the Next

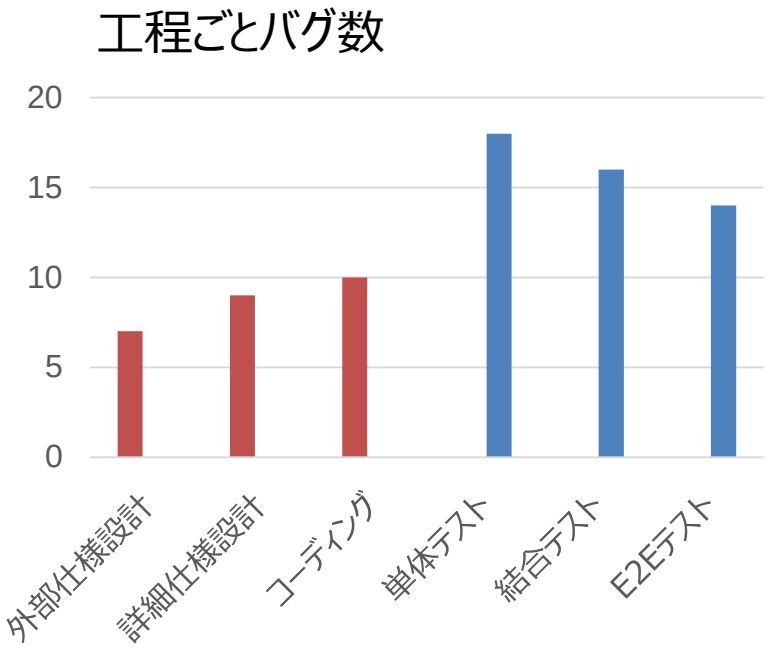
バグ前倒しの指標として、

「欠陥摘出前倒し率」がある。

どれだけ上流工程でバグを摘出できたか

を測定することが可能。





しかし、従来の「欠陥摘出前倒し率」は工程が明示的に分かれている

ウォーターフォールにしか適用できなかった。

© Hitachi, Ltd. 2018. All rights reserved.

4-2. 前倒し率改良のポイント

HITACHI
Inspire the Next

そこで、アジャイル開発のような、イテレーティブな開発にも適用可能なよう、

従来の「欠陥摘出前倒し率」を改良した。

改良のポイント

- バグトラッキングシステムのバグチケット作成日に着目することで、
  工程に関係なくバグの前倒しを計算可能にした。
- バグトラッキングシステムからのみ**自動集計**できるよう、収集対象を
  E2Eテストのバグのみに絞った。

※従来の「欠陥摘出前倒し率」はレビューの効果測定を目的としている。

今回の発表内容は、あくまで計算方法のみを参考にした。

© Hitachi, Ltd. 2018. All rights reserved.

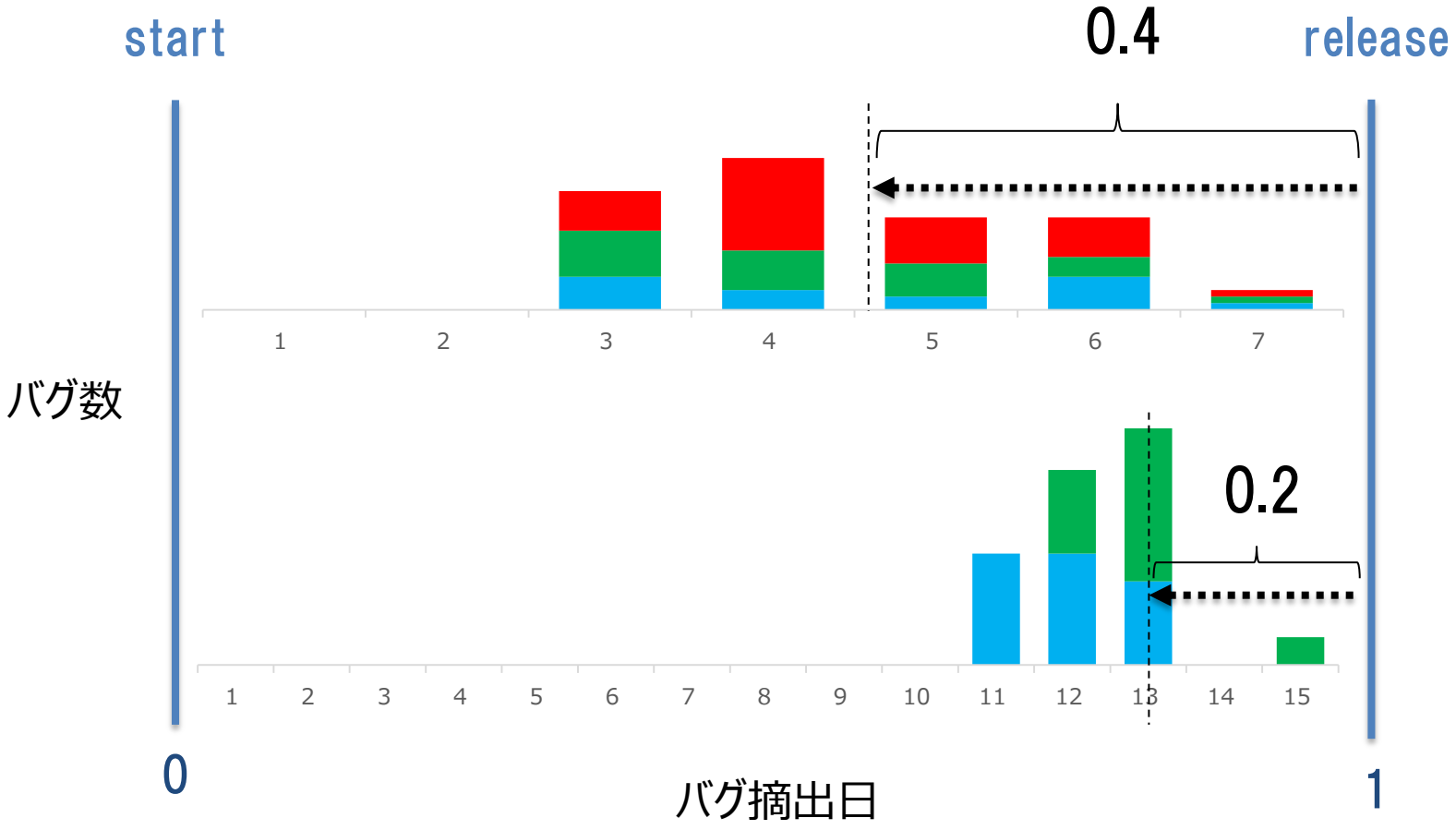
4-3. 前倒し率の考え方

HITACHI
Inspire the Next

バグ摘出日を0～1に標準化して、平均値をとって、1から引く。

0：プロジェクト開始日

1：プロジェクト終了日



© Hitachi, Ltd. 2018. All rights reserved.

4-4. 前倒し率の求め方

HITACHI
Inspire the Next

各バグ$bug[i]$に関して、$bug[i].date$（摘出日を0～1に標準化した値）を計算する。

$$bug[i].date = \frac{\textbf{バグ摘出日} - \textbf{プロジェクト開始日}}{\textbf{プロジェクト終了日} - \textbf{プロジェクト開始日}}$$

すると前倒し率は、

$$\text{前倒し率} = 1 - \frac{\sum_{i=1}^{N} bug[i].date}{\mathrm{N}} \qquad (1 - \textbf{平均})$$

となる。

例えば、90日間のプロジェクトで、$bug[1]$の摘出日がプロジェクト開始から20日目だとすると、$bug[1].date$は

$$bug[1].date = 20/90 = 0.222$$

のように計算する。
また、$bug[1].date$ ～ $bug[8].date$の平均が0.700なので、前倒し率は

$$\text{前倒し率} = 1 - 0.700 = 0.300$$

となる。

前倒し率の計算例

バグ	摘出日	bug[1].date
bug[1]	20日目	0.222
bug[2]	40日目	0.444
bug[3]	50日目	0.556
bug[4]	66日目	0.733
bug[5]	79日目	0.878
bug[6]	79日目	0.878
bug[7]	81日目	0.900
bug[8]	89日目	0.989
平均		0.700
前倒し率		0.300

© Hitachi, Ltd. 2018. All rights reserved.

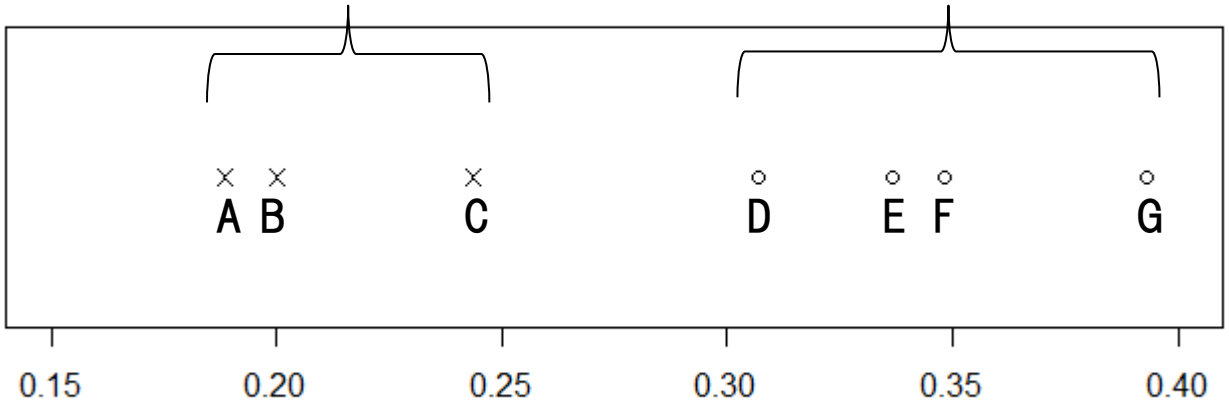
4-5. 前倒し率の有用性検証

HITACHI
Inspire the Next

各プロジェクトの前倒し率と市場不具合を調べた。

前倒し率<0.25
→市場不具合あり

前倒し率>0.30
→市場不具合なし



○:市場不具合なし
×:市場不具合あり

市場不具合の有無で、t検定を行ったところ、p値が0.028となったので、市場不具合があるグループとないグループの間には、前倒し率に有意な差があると言える。

© Hitachi, Ltd. 2018. All rights reserved.

4-6. 重み付前倒し率

HITACHI
Inspire the Next

仮説

リリース直前に深刻なバグが出ると品質リスクが高くなるのでは？

バグの**重要度で重み付け**した前倒し率と

市場不具合との関連を分析する。

バグの重要度、付け方の例

重要度	定義
Major	システム停止
Minor	出力メッセージ不正
Trivial	ちょっとした誤植

※重要度は顧客影響度や修正難易度などからプロジェクトごとに定義する値。バグ修正の優先度付けや、リリース判定に使うので、大体のプロジェクトで入力するようにしている。

© Hitachi, Ltd. 2018. All rights reserved.

4-7. 重み付前倒し率の求め方

HITACHI
Inspire the Next

今回は、各重要度の重みを単純に（線形）付けた。
※重要度と重みの付け方は各プロジェクトで定義してよい。

重要度	重み	バグ数
Major	3	N_3
Minor	2	N_2
Trivial	1	N_1

重み付前倒し率

$$= 1 - \frac{\sum_{i=1}^{N_3} 3 * bug[i].date + \sum_{i=1}^{N_2} 2 * bug[i].date + \sum_{i=1}^{N_1} 1 * bug[i].date}{3 * \mathrm{N}_3 + 2 * N_2 + 1 * N_1}$$

© Hitachi, Ltd. 2018. All rights reserved.

4-8. 重み付前倒し率の簡単な計算方法

HITACHI
Inspire the Next

Excel®でも簡単に計算可能

bug[i].date*重要度

バグ	摘出日	bug[i].date	重要度	重み付date
bug[1]	20日目	0.222	1	0.222
bug[2]	40日目	0.444	1	0.444
bug[3]	50日目	0.556	1	0.556
bug[4]	66日目	0.733	1	0.733
bug[5]	79日目	0.878	2	1.756
bug[6]	79日目	0.878	3	2.633
bug[7]	81日目	0.900	2	1.800
bug[8]	89日目	0.989	3	2.967

重要度合計14

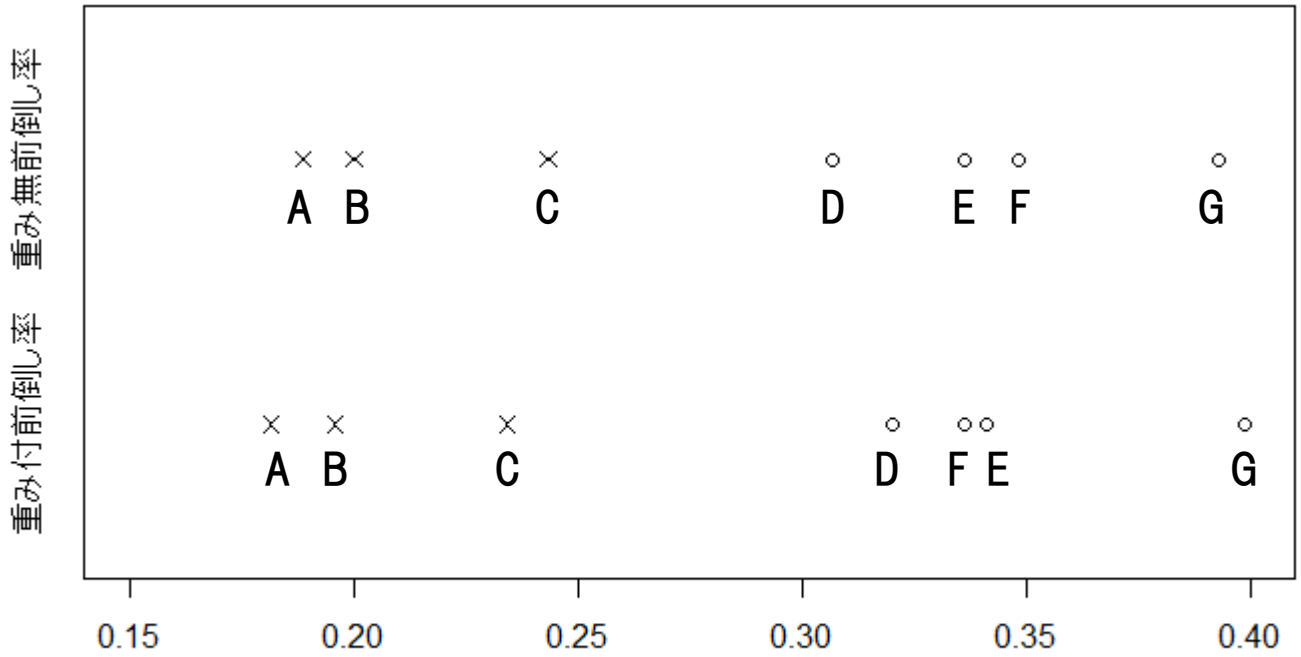
重み付date合計11.111

$$\text{重み付前倒し率} = 1 - \frac{\text{重み付date合計11.111}}{\text{重要度合計14}}$$

*Excelは，米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

© Hitachi, Ltd. 2018. All rights reserved.

4-9. 重み付前倒し率の有用性検証



○:市場不具合なし
×:市場不具合あり

t検定結果: p値が0.016 (重み無だとp値0.028)

つまり、重み付前倒し率の方が市場不具合の予測精度が高い

ご参考：Wilcoxonの順位和検定ではp値が0.057

© Hitachi, Ltd. 2018. All rights reserved.

5. 結論と今後の課題

© Hitachi, Ltd. 2018. All rights reserved.

HITACHI
Inspire the Next

前倒し率適用可能プロジェクト

- バグ摘出日を管理している（バグトラッキングシステムを使っている）
- スプリントごとのリリースせず、複数スプリントのちリリース

© Hitachi, Ltd. 2018. All rights reserved.

5-2. 前倒し率のメリット/デメリット

HITACHI
Inspire the Next

メリット

- イテレーティブな開発でも市場不具合の予測が可能。
- バグトラッキングシステムでバグを管理していれば計算を簡単に自動化できる。
- 前倒しを意識することで、すぐにE2Eテストをするようになり、
  開発チームに対する早いフィードバックを促すことが可能。

デメリット

- プロジェクト終了時でないと測定できない。開発中は前倒し率の推移や、
  バグ摘出傾向を見せることで、なるべく前倒してテストしようという意識を付けて
  もらう。

© Hitachi, Ltd. 2018. All rights reserved.

HITACHI
Inspire the Next

今後の課題

- 今回の発表ではサンプル数が少なかったため、前倒し率と市場不具合の関係は継続検証する。
- 他メトリクスとの組み合わせた時の市場不具合への寄与を測定したい。

© Hitachi, Ltd. 2018. All rights reserved.

参考文献

野中 誠, 小池 利和, 小室 睦 (2012)『データ指向のソフトウェア品質マネジメント―メトリクス分析による「事実にもとづく管理」の実践』日科技連出版社

© Hitachi, Ltd. 2018. All rights reserved.

使用ロゴ

HITACHI
Inspire the Next

- Git logs
  - Title: Full color Git logo for light background
  - Author: Jason Long
  - Source: https://git-scm.com/images/logos/downloads/Git-Logo-2Color.png
  - License: CC BY 3.0
- gopher
  - Title: gopher
  - Author: Takuya Ueda
  - Source: https://github.com/golang-samples/gopher-vector/blob/master/gopher.png
  - License: CC BY 3.0
- Git Logo
  - Title: Full color Git logo for light backgrounds
  - Autor: Jason Long
  - Source: https://git-scm.com/images/logos/downloads/Git-Logo-2Color.png
  - License: CC BY 3.0

© Hitachi, Ltd. 2018. All rights reserved.

HITACHI

Inspire the Next