wakayama univ.

Mining Software Repositories to Study Co-Evolution of Production & Test Code

Andy Zaidman, Bart Van Rompaey, Serge Demeyer, Arie van Deursen

In Proceedings of the International Conference on Software Testing, Verification and Validation (ICST), pages 220–229, 2008.

発表者　吉富　楓雅

ソフトウェアの共進化

ソフトウェアシステムの進化

- ソフトウェアシステムを進化させるとき，ソースコードが中心的
- **ソフトウェアは多次元的である**
- 高品質なソースコードを開発するには，他の**アーティファクト(仕様，制約，文書，テストなど)**が含まれる必要がある[1]

  * **ソースコードは主要なアーティファクト**

[1]T. Mens, M. Wermelinger, S. Ducasse, S. Demeyer, R. Hirschfeld, and M. Jazayeri. Challenges in software evo- lution. In Proc. of the Int'l Workshop on Principles of Software Evolution (IWPSE), pages 13–22. IEEE, 2005.

テスト実施の際の課題

- テストの計画の未作成など，組織的なテスト活動が実施されていない
  - **リリースされたOSSには多くの欠陥が含まれている**
- プロダクトコードの変更と同時に，検証用のテストコードが変更(**共進化**)されていない

目的

- ソースコードに対してテストコードがどのように進化するのかを調査する

リサーチクエスチョン

- RQ1：プロダクトコードとテストコードの共進化は同期的に起こるか，段階的に起こるか？
- RQ2：プロジェクトの開発期間中にテスト作成の効率は向上するか？
- RQ3：テスト駆動型開発[3]などのテスト戦略を検出できるか？
- RQ4：テスト作成の効率とテストカバレッジの関係はあるか？

[2] E. Maximilien and L.Williams. Assessing test-driven development at IBM. In Proc. Int'l Conf. on Software Engineering (ICSE), pages 564-569. IEEE, 2003.

テスト駆動型開発

- Test-driven development: TDD
- **単体テストをコーディング**し，テストが動作するように**最低限の機能をソースコードに搭載した後**，プロダクトコードをコーディングする方法(**テストファースト**)
- アジャイル開発などにおいて推奨

３つの視点による
テスト共進化の調査

3つの視点

* 変更履歴(RQ1, RQ3)
  - 開発者のコミット動作を視覚化する
* 成長履歴(RQ1, RQ2. RQ3)
  - 時間の経過によるプロダクトコードとテストコードの成長を示す
* テスト品質進化(RQ4)
  - テストカバレッジを離散時間単位でプロットする

変更履歴

- 目的：開発者の活動数の調査

1. プロダクトコードに関連する（単体）テストがあるかどうか

2. プロダクトコードとテストコードが同時に加えられ，改変されるかどうか

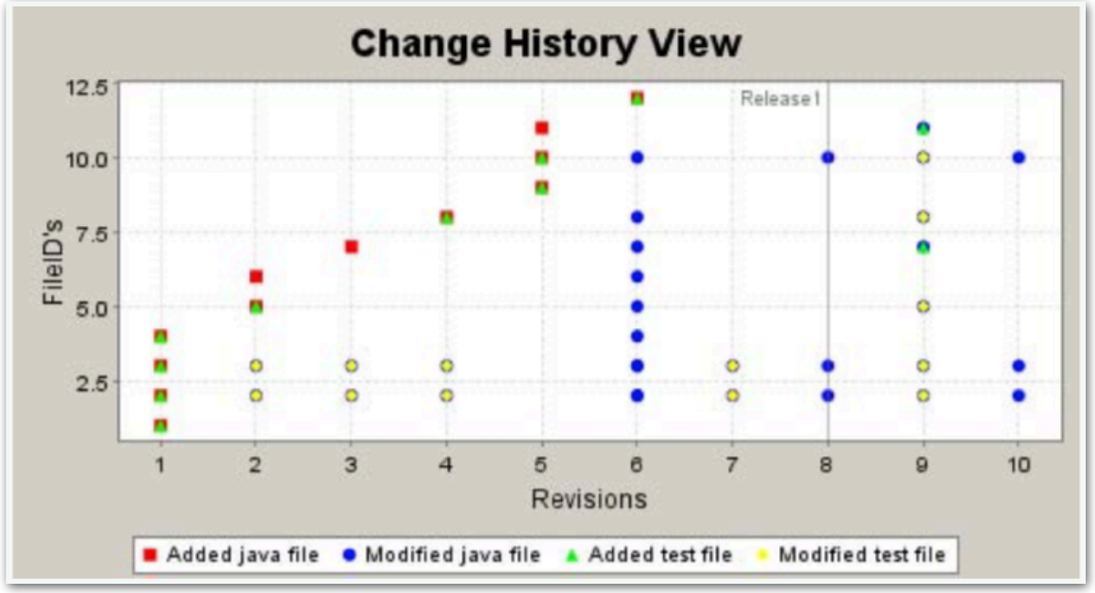


図１：変更履歴ビューの例

- Javaファイルを追加
- 変更されたjavaファイル
- テストファイルを追加する
- 変更されたテストファイル

成長履歴

- 目的：メジャーリリース直前のテスト作成効率が増加しているか・テスト駆動型開発を示すような成長パターンが存在するかの調査

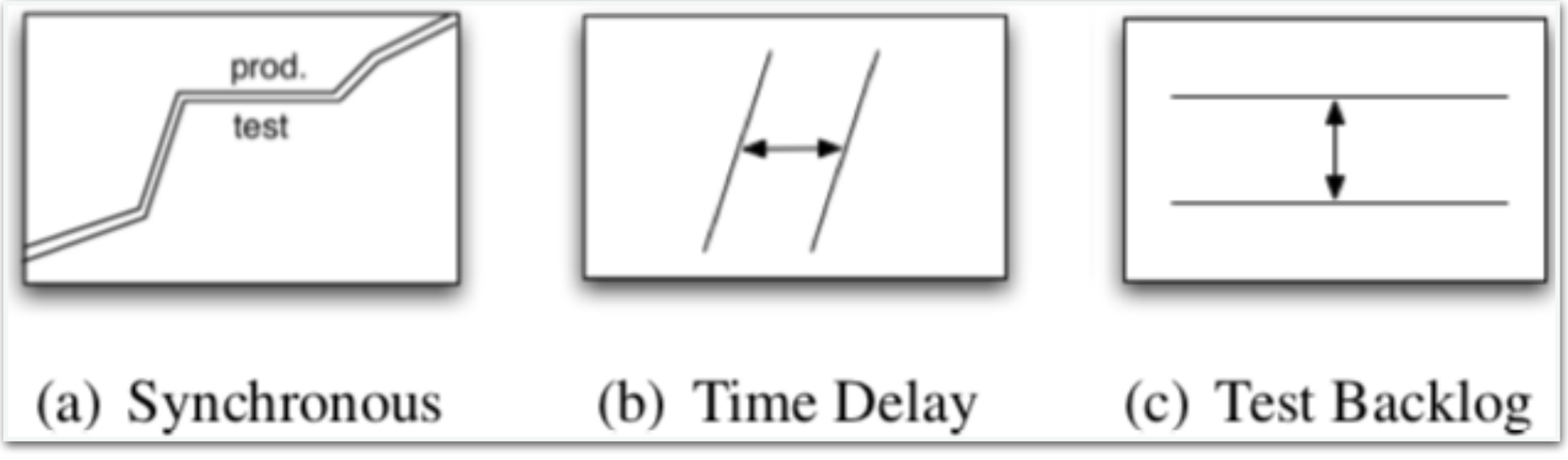
- ５つの指標：
  - プロダクトコードの行数(pLOC), テストコードの行数(tLOC), プロダクトクラス数(pClasses), テストクラス数(tClasses), テストコマンド数(tCommands)

成長履歴

a. プロダクトコードとテストコードを書く作業は同期する

b. １つの活動と関連する活動の時間が遅延している

c. 過去のテストや未着手のテストが時間の経過とともに蓄積されている



同期　　水平移動　　垂直移動

図２：プロダクトコードとテストコードの共進化

テスト品質進化

- 目的：ソフトウェアの長期的な”テストの健全性”を判断するための，テストカバレッジの描画
- テストカバレッジが増加すると，テストコードの割合は緩やかに増加するか一定となる
- テストコードの割合(tLOCRatio)と時間を用いてテストカバレッジを描画

$$\text{tLOCRatio} = \frac{tLOC}{pLOC + tLOC}$$

[3]H. Zhu, P. A. Hall, and J. May. Software unit test coverage and adequacy. ACM Comput. Surv., 29(4):366–427, 1997.

分析結果

※Checkstyle７とArgoUMLの２つのツールに実験が行われていましたが，時間の都合上Checkstyle７のみ結果を示します

Checkstyle 7

- Javaコードが特定のコーディング基準に準拠しているかどうかをチェックするツール

- ６人の開発者が2001年6月〜2007年3月で2260件のコミット

*738のクラスと47kSLOC(ソースコード47千行)を作成

Checkstyleの変更履歴

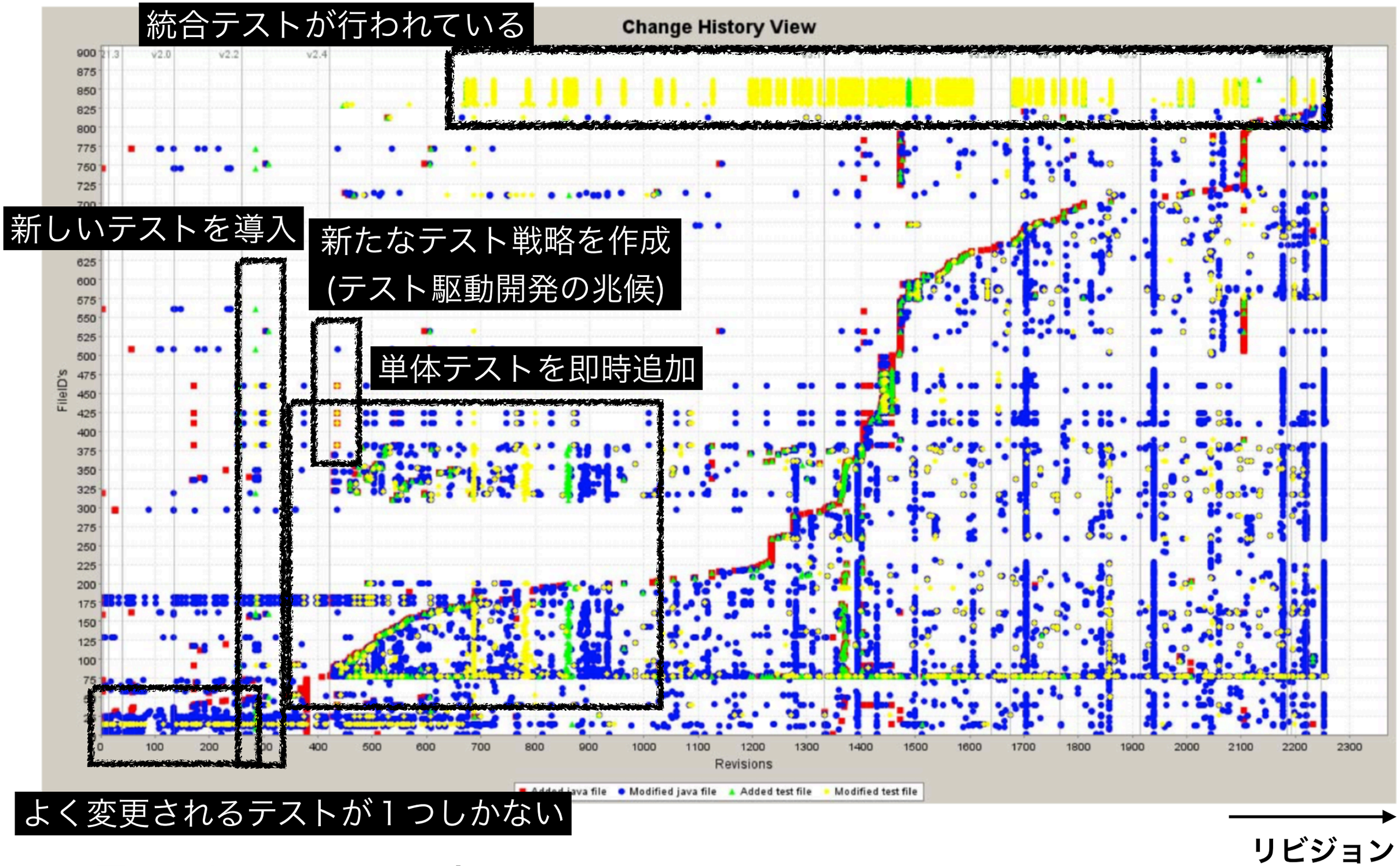
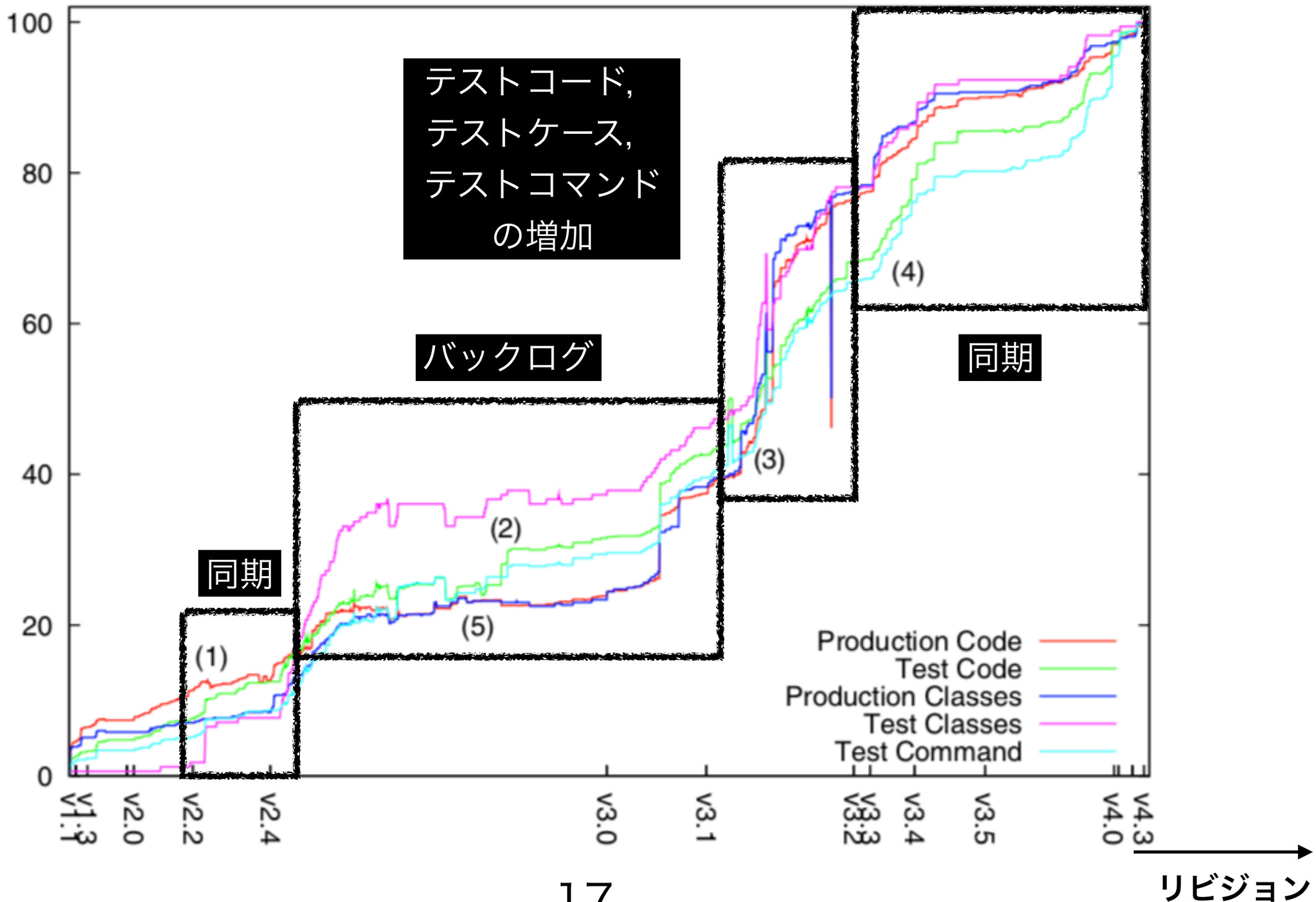
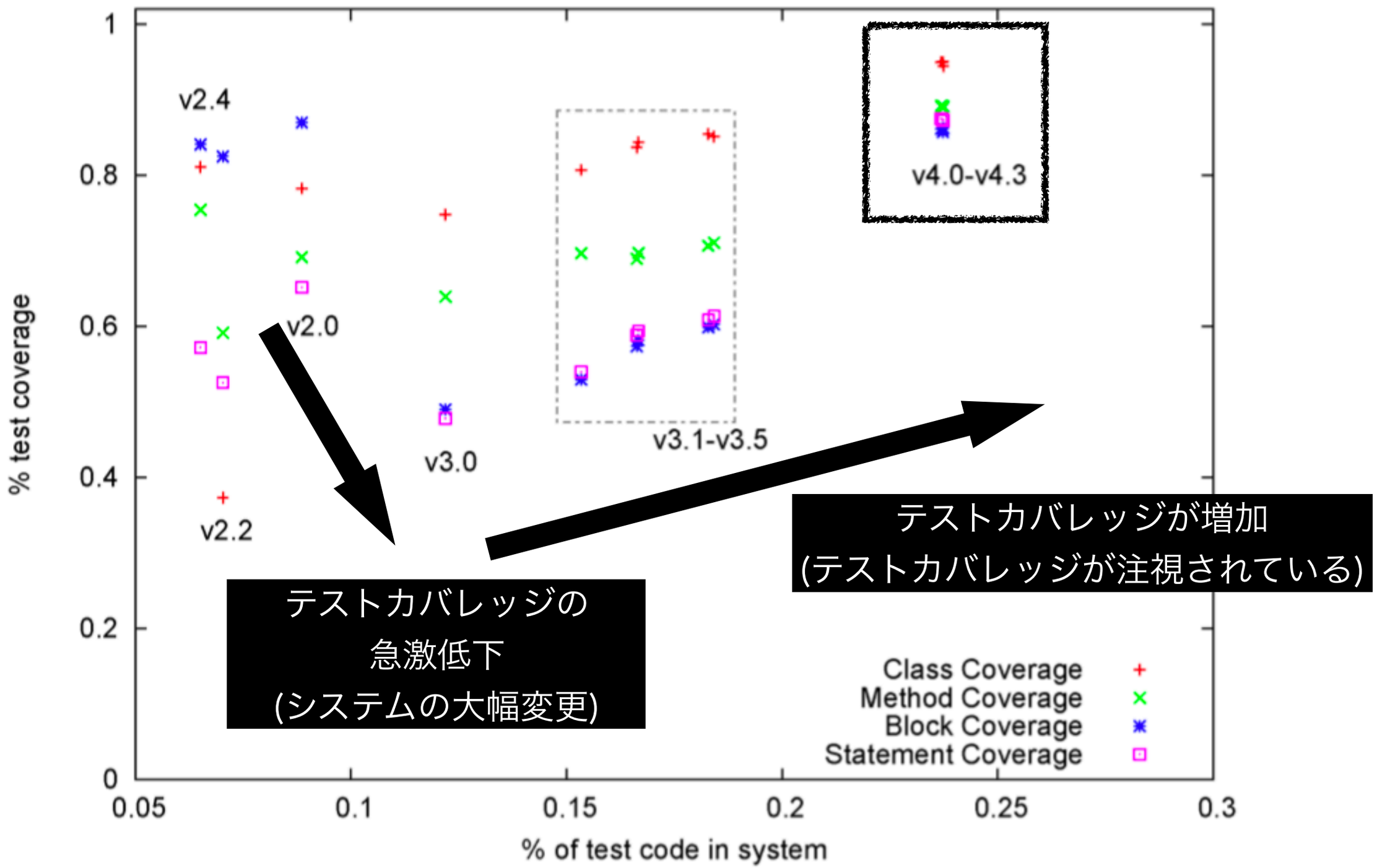


図２：Checkstyleの変更履歴

成長履歴


テストコード,
テストケース,
テストコマンド
の増加
バックログ
同期
同期
(1)
(2)
(3)
(4)
(5)
Production Code
Test Code
Production Classes
Test Classes
Test Command
100
80
60
40
20
0
v1.1
v1.3
v2.0
v2.2
v2.4
v3.0
v3.1
v3.2
v3.3
v3.4
v3.5
v4.0
v4.3
リビジョン

テスト品質進化


% test coverage
% of test code in system
0
0.2
0.4
0.6
0.8
1
0.05
0.1
0.15
0.2
0.25
0.3
v2.4
v2.0
v2.2
v3.0
v3.1-v3.5
v4.0-v4.3
テストカバレッジの
急激低下
(システムの大幅変更)
テストカバレッジが増加
(テストカバレッジが注視されている)
Class Coverage
Method Coverage
Block Coverage
Statement Coverage

結論と所感

結論

- ソースコードとテストコードの共進化について，３つの視点から検討

  - Checkstyleについて

    - RQ1：同期的に共進化
    - RQ2：テスト作成の効率が上昇
    - RQ3：テスト駆動開発の兆候が見られた
    - RQ4：テスト作成の向上に伴ってカバレッジも向上した

所感

- 単体テストと結合テストを区別せずにテストカバレッジを求めている
- 産業プロジェクトにも応用するようなので，そちらの論文も読みたい