

# **A Longitudinal Analysis of Bug Handling Across Eclipse Releases**

Zeinab Abou Khalil, Eleni Constantinou  
Tom Mens, Laurence Duchien, Clement Quinton

2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)

# 背景

- これまでの研究では、バグ修正の様々な側面に焦点が当てられてきた
  - 多くはリリース前後の区別なし
- バグ修正アクティビティを比較する調査を行っていない

# 背景

- Mozilla Firefoxのリリースサイクル切り替えの影響調査[24]
  - 適応には事前の準備に影響を受ける
- Eclipseは年間リリースから四半期リリースへ

[24] : F.Khomh,B.Adams,T.Dhaliwal,andY.Zou,“Understanding the impact of rapid releases on software quality,” Empirical Software Engineering, vol. 20, pp. 336–373, Apr 2015.

# 調査内容

- Eclipseのバグ履歴の調査
  - 予定されているリリースの前後でバグ修正時間がどのように異なるか
  - 四半期リリースがバグ修正プロセスにどのように影響するか

# RQ

1. 各リリースの前後でバグの数はどのように異なるか
2. バグの解決率と修正率はリリースごとにどのように変化し、各リリースの前後でどのように異なるか
3. バグのトリアージと修正の時間は、各リリースの前後でどのように異なるか
4. リリース期限に近づくと、バグのトリアージと修正の時間はどのように変化するか

# ケーススタディ

- Eclipse
  - 十分に長いリリース期間を持つ大規模プロジェクト
  - バグ修正に関する研究が広くされている
  - 安定した開発コミュニティと定期的なリリースプロセスを持つ

# データ

- Bugzilla APIを使用し、バグレポートとバグの履歴を抽出
- Platfom, JDT, Equinox, PEDのタグがついているバグレポート
- Eclipsの公式にリストされているバージョンのバグレポート
- 138,445のバグレポートを抽出

Eclipseのバージョン

|      |            |          |
|------|------------|----------|
| 3.0  | 2004/06    |          |
| 3.1  | 2005/06    |          |
| 3.2  | 2006/06/30 | Callisto |
| 3.3  | 2007/06/29 | Europa   |
| 3.4  | 2008/06/25 | Ganymede |
| 3.5  | 2009/06/24 | Galileo  |
| 3.6  | 2010/06/23 | Helios   |
| 3.7  | 2011/06/22 | Indigo   |
| 4.2  | 2012/06/27 | Juno     |
| 4.3  | 2013/06/26 | Kepler   |
| 4.4  | 2014/06/25 | Luna     |
| 4.5  | 2015/06/24 | Mars     |
| 4.6  | 2016/06/22 | Neon     |
| 4.7  | 2017/06/28 | Oxygen   |
| 4.8  | 2018/06/27 | Photon   |
| 4.9  | 2018/09/19 | 2018-09  |
| 4.10 | 2018/12/19 | 2018-12  |

[https://ja.wikipedia.org/wiki/Eclipse\\_\(統合開発環境\)](https://ja.wikipedia.org/wiki/Eclipse_(統合開発環境))

# メトリクス

$$ResRate(d) = \frac{|B_{resolve}(d)|}{|B_{report}(d)|}$$

$$FixRate_{report}(d) = \frac{|B_{fix}(d)|}{|B_{report}(d)|}$$

$$FixRate_{resolve}(d) = \frac{|B_{fix}(d)|}{|B_{resolve}(d)|}$$

- 報告, 解決されたバグの割合
- 報告, 修正されたバグの割合
- 修正, 解決されたバグの割合

# メトリクス(例)

$$ResRate(d) = \frac{|B_{resolve}(d)|}{|B_{report}(d)|}$$

$$FixRate_{report}(d) = \frac{|B_{fix}(d)|}{|B_{report}(d)|} \quad FixRate_{resolve}(d) = \frac{|B_{fix}(d)|}{|B_{resolve}(d)|}$$

- 期間dの間に30のバグ報告, 20のバグ解決,  
12のバグ修正

$$ResRate(d) = \frac{20}{30} = 0.66$$

$$FixRate_{report}(d) = \frac{12}{30} = 0.4$$

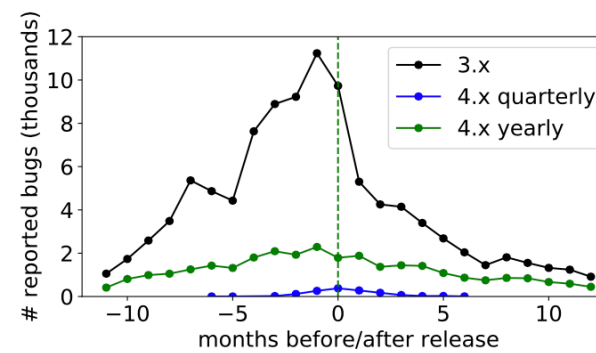
$$FixRate_{resolve}(d) = \frac{12}{20} = 0.6$$

# RQ(再掲)

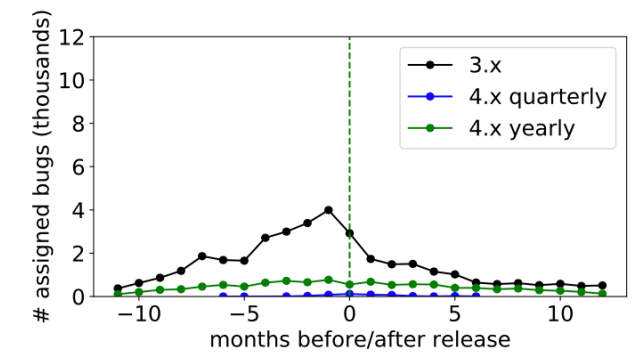
1. 各リリースの前後でバグの数はどのように異なるか
2. バグの解決率と修正率はリリースごとにどのように変化し、各リリースの前後でどのように異なるか
3. バグのトリアーჯと修正の時間は、各リリースの前後でどのように異なるか
4. リリース期限に近づくと、バグのトリアーჯと修正の時間はどのように変化するか

# RQ1 : 結果

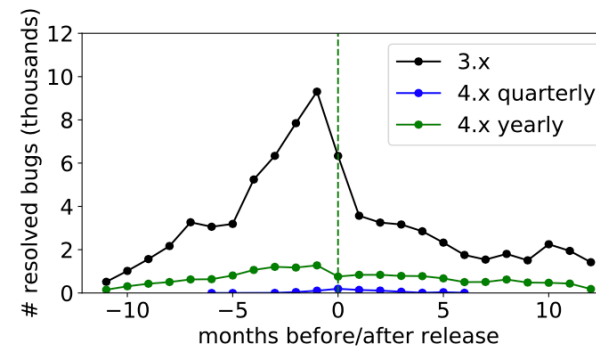
- 3.xシリーズでは、リリース前にバグ処理が活発に



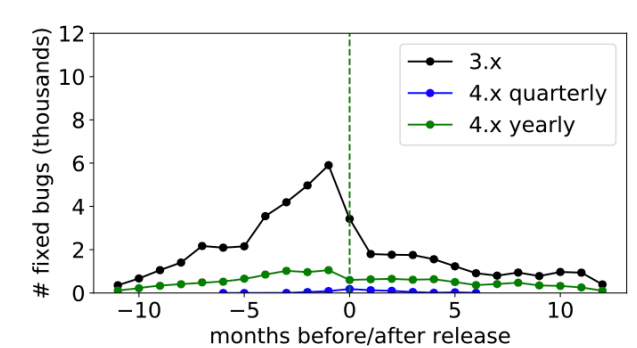
(a) number of reported bugs



(b) number of assigned bugs



(c) number of resolved bugs



(d) number of fixed bugs

Fig. 5: Monthly evolution of the number of bugs before/after a release, aggregated by release series (y-axis is measured in thousands of bugs).

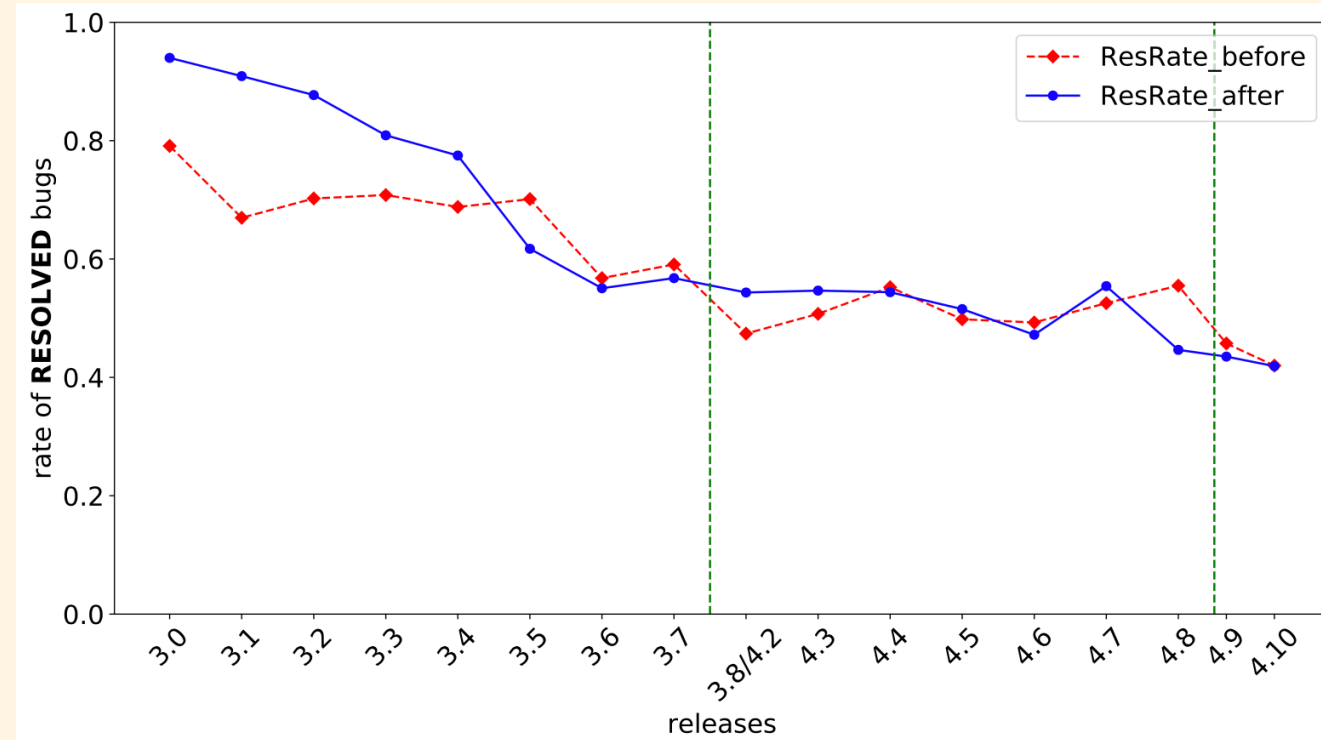
- 4.xシリーズでは、リリース前後でバランスの取れたバグ処理のワークロードが提案されている

# RQ(再掲)

1. 各リリースの前後でバグの数はどのように異なるか
2. バグの解決率と修正率はリリースごとにどのように変化し、各リリースの前後でどのように異なるか
3. バグのトリアージと修正の時間は、各リリースの前後でどのように異なるか
4. リリース期限に近づくと、バグのトリアージと修正の時間はどのように変化するか

# RQ2 : 結果

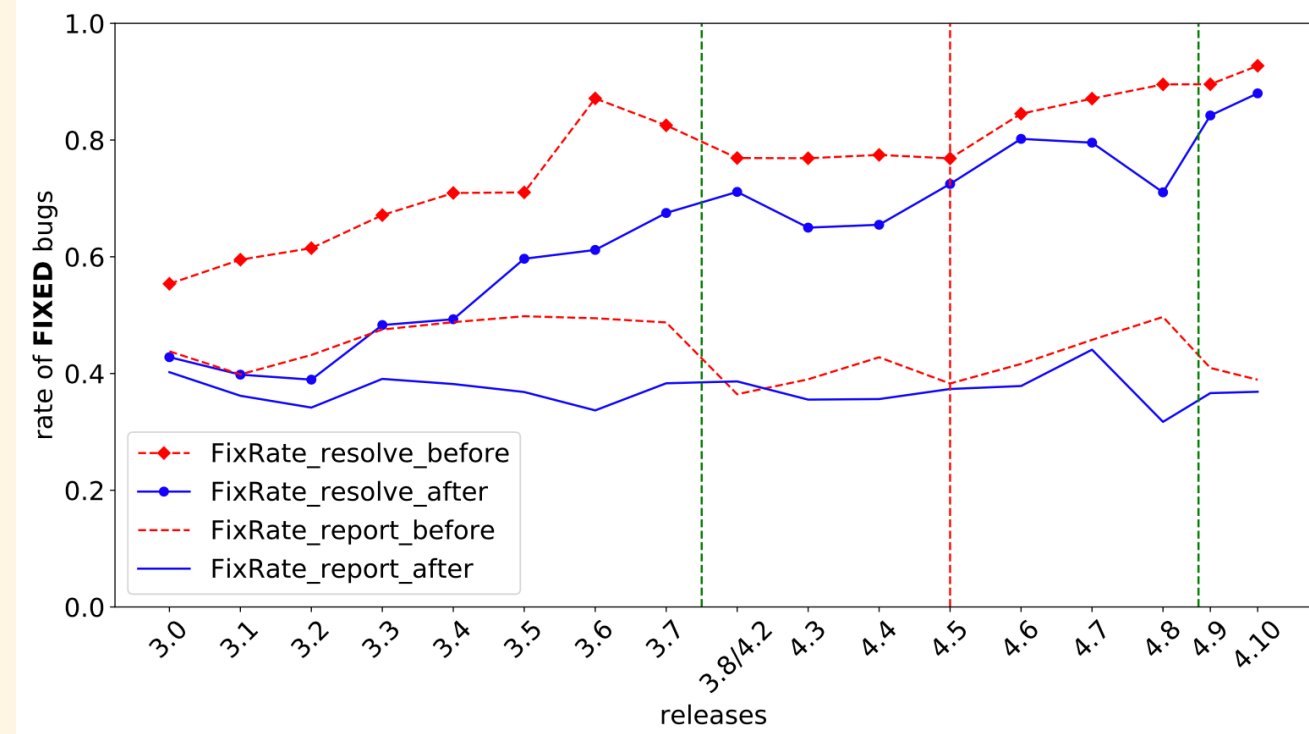
$$ResRate(d) = \frac{|B_{resolve}(d)|}{|B_{report}(d)|}$$



(a) Resolution rate evolution

- 解決率はリリースごとに低下する傾向
  - LATER, REMINDの使用停止に伴い解決率が低下
  - リリース前後で解決率に大きな違いはない

# RQ2 : 結果



(b) Fixing rate evolution

- 修正率はリリース前が高い
- 4.5以降にAERI\*の導入
  - AERIの導入は, バグ修正率のさらなる改善につながる

\* [https://www.eclipse.org/community/eclipse\\_newsletter/2016/july/article3.php](https://www.eclipse.org/community/eclipse_newsletter/2016/july/article3.php)

# RQ(再掲)

1. 各リリースの前後でバグの数はどのように異なるか
2. バグの解決率と修正率はリリースごとにどのように変化し、各リリースの前後でどのように異なるか
3. バグのトリアージと修正の時間は、各リリースの前後でどのように異なるか
4. リリース期限に近づくと、バグのトリアージと修正の時間はどのように変化するか

# RQ3：結果

- トリアーージ時間
  - リリース後よりリリース前の方がトリアーージ時間が短くなる
  - 年間リリースから四半期リリースによる有意差はない
- 修正時間
  - リリース後よりリリース前の方がバグ修正時間が短くなる

# RQ(再掲)

1. 各リリースの前後でバグの数はどのように異なるか
2. バグの解決率と修正率はリリースごとにどのように変化し、各リリースの前後でどのように異なるか
3. バグのトリアーჯと修正の時間は、各リリースの前後でどのように異なるか
4. リリース期限に近づくと、バグのトリアーჯと修正の時間はどのように変化するか

# RQ4 : 結果(blue boxes)

- 次のリリースが近づく  
と新バージョンのバグ  
の優先順位が上がる
  - より少ないバグで  
新バージョンを提供  
するため

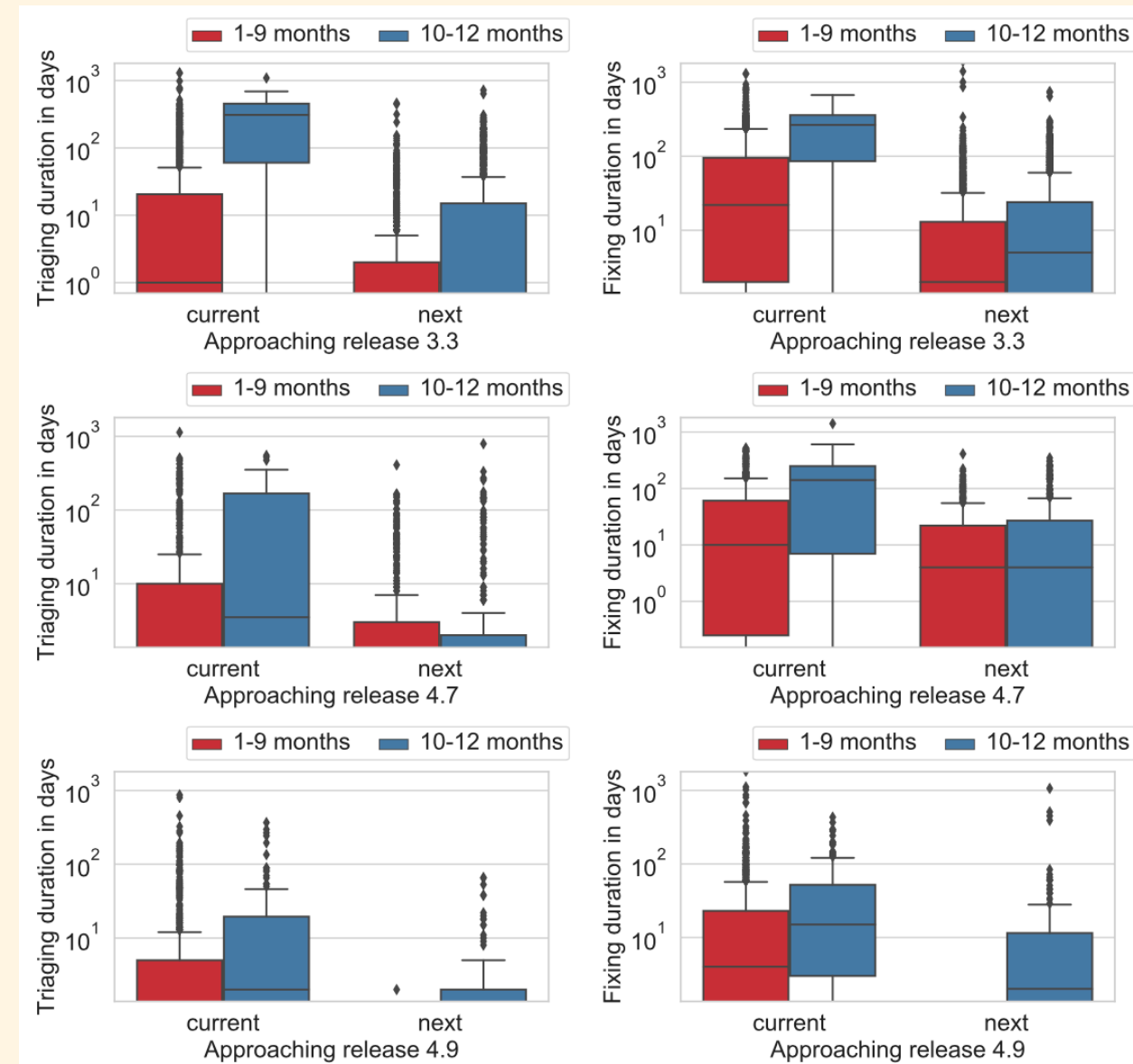


Fig. 8: Triaging time distributions (left figures) and fixing time distributions (right) when approaching next release.

# RQ4 : 結果(current)

- currentのバグは, early periodより pressure periodのトリアージ, 修正に時間がかかる

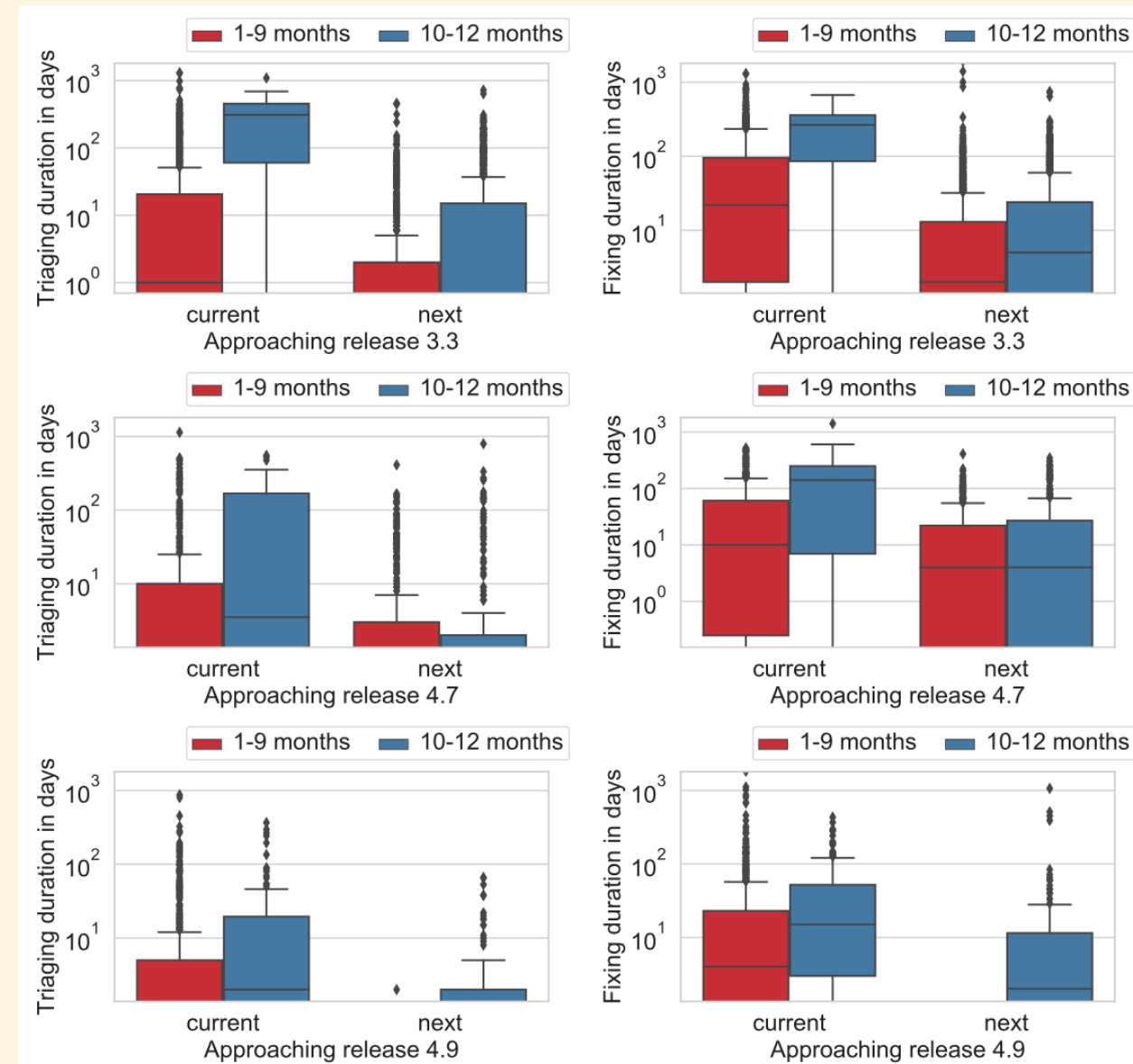


Fig. 8: Triaging time distributions (left figures) and fixing time distributions (right) when approaching next release.

# RQ4 : 結果(next)

- トリアーージの時間にほとんど違いはない
- 一部のバージョンを除きバグ修正の時間にほとんど違いはない

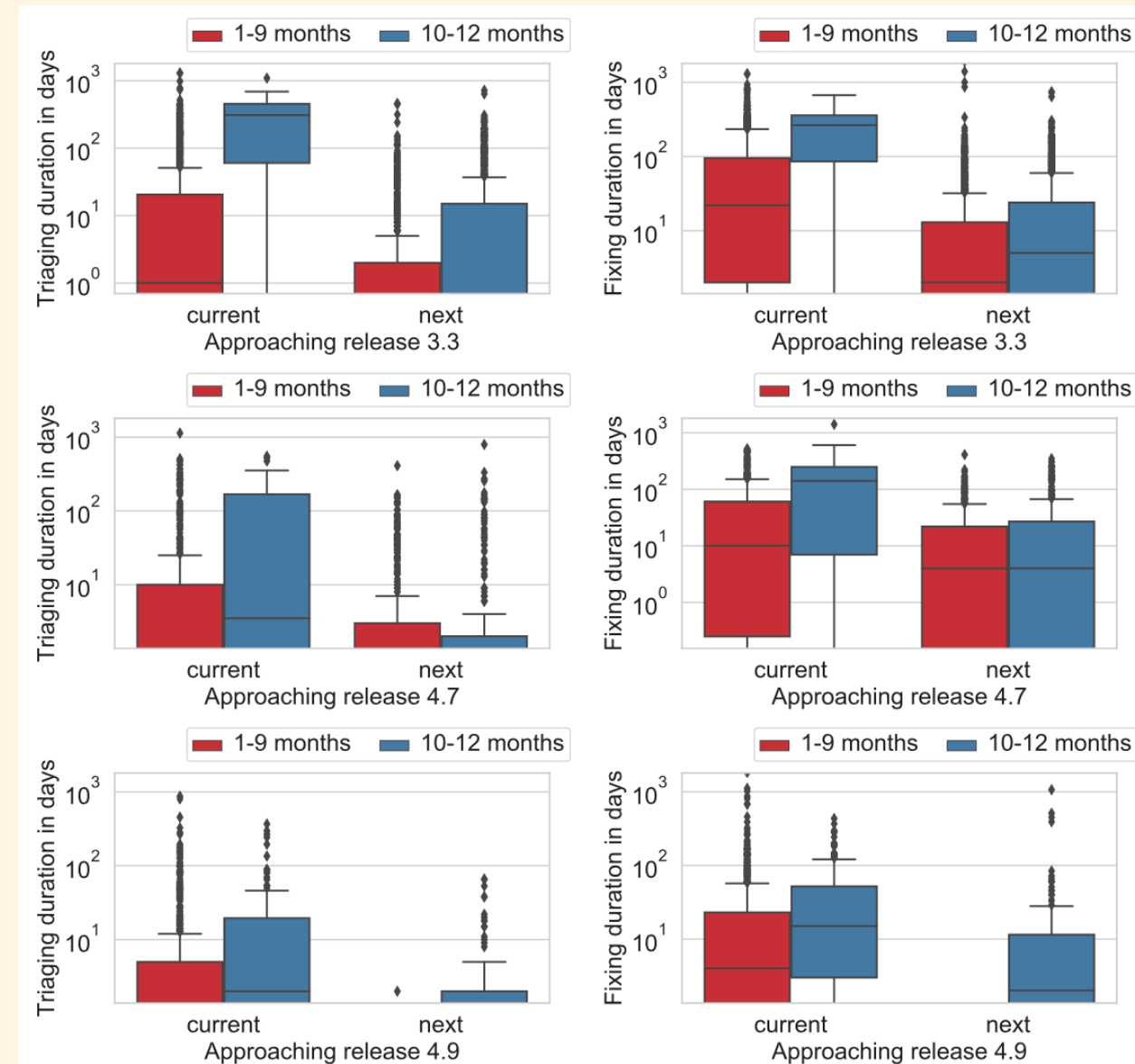


Fig. 8: Triaging time distributions (left figures) and fixing time distributions (right) when approaching next release.

# まとめ

- 年間リリース16回と四半期リリース2回を分析
- LATER, REMINDの使用停止及びAERIの導入
  - バグの解決率は低下, 修正率は増加
- 年間リリースから四半期リリースへの移行の悪影響はみられない