Predicting Failures with Developer Networks and Social Network Analysis

Andrew Meneely
Laurie Williams
Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Will Snipes
Nortel Networks, Research Triangle Park, NC, USA.
wbsnipes@nortel.com

Jason Osborne
Nortel Networks,Department of Statistics, North Carolina State University, Raleigh, NC, US

SIGSOFT '08/FSE-16 Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering Pages 13-23

背景

- ソフトウェアの不具合修正は非常にコストがかかる
- 多くの研究で高い精度の不具合予測モデルの構築がされてきている
- しかし，開発者の性質をメトリクスとした予測モデルはあまりない

先行研究

- 重要なコンポーネントほど不具合を含みやすい[1]
- 開発者の情報（ex.コード変更の数）をメトリクスにとりいれたモデルの構築[2]
- オブジェクト指向システムにおいて開発者の経験やスキルはフォールトプレーンに影響を与える[3]

開発者の能力を考慮した予測モデルは多いが，
開発者のネットワークをメトリクスにした予測モデルの構築はされていない

[1] Zimmermann, T. and Nagappan, N., "Predicting Defects using Network Analysis on Dependency Graphs," in 29th International Conference on Software Engineering, 2007.
[2] Mockus, A. and Weiss, D. M., "Predicting Risk of Software Changes," Bell Labs Technical Journal, vol. 5,pp. 169-180, 2002.
[3]Arisholm, E. and Briand, L. C., "Predicting Fault-prone Components in a Java Legacy System," in 2006 ACM/IEEE International Symposium on Empirical Software Engineering, 2006, pp. 8-17.

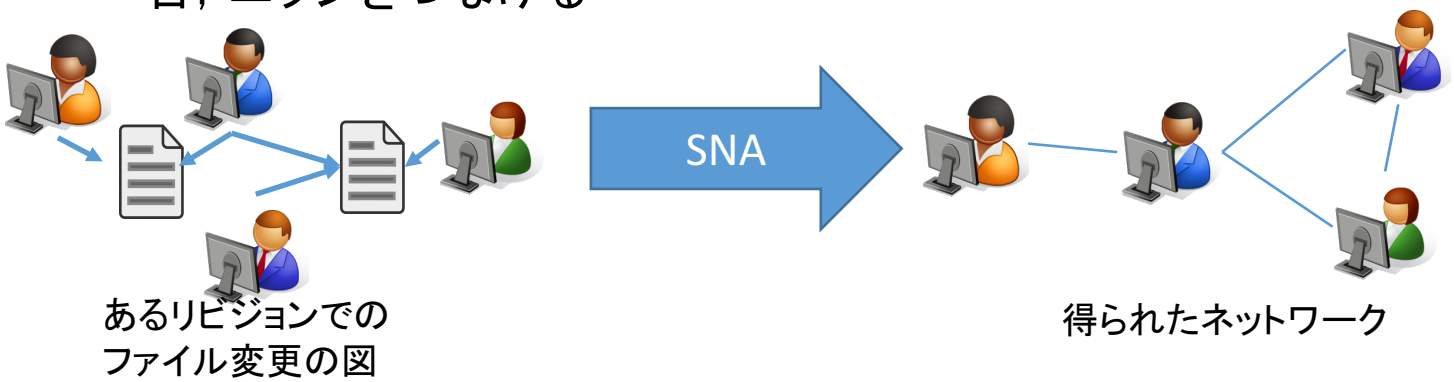
目的とアプローチ

- 目的
  - コードチャム（修正や追加などの変更）の情報にSNA（Social Network Analysis）を適用することで，不具合予測における人的要因を調査すること

- アプローチ
  - SNAを適応した予測モデルの構築（２つ）
  - 予測モデルの優先順位付け（predicted prioritization）と最適優先順位付け（optimal prioritization）での発見率の割合を比較

SNA

- 本研究におけるノードのつなげ方
  - 同じリビジョン間に少なくとも一つは共通のファイルの変更を行った場合，エッジをつなげる



- 本研究で用いたメトリクス
  - degree:あるノードが直接繋がっているノードの数
  - Betweenness：あるノードが全ての経路内でどれだけ含まれてるか
  - closeness:あるノードからその他全ノードへの距離の平均の長さ

ケーススタディ

- データセット
  - 対象プロジェクト
    - Nortel Network（企業）のプロダクト
  - 対象ファイル
    - 2,500ファイル/11,000ファイル
    - 3つのリビジョンを対象（R_n, R_{n+1}, R_{n+2} ）

  - 学習データセット
    - R_n, R_{n+1}
  - *検証データセット*
    - R_{n+2}

提案モデルのメトリクス

Table 4: Candidate file-based metrics

Metric	Description
Code Churn	The number of lines of code that were either added or changed over the history of this file
Updates	The number of updates to the repository that included this file
Developers	The number of distinct developers who have updated this file over its history
(Sum/Average/Max) of Degree	The (sum/average/maximum) of each developer's degree over a file's history
(Sum/Average/Max) of Closeness	The (sum/average/maximum) of each developer's Closeness over a file's history
(Sum/Average/Max) of Betweenness	The (sum/average/maximum) of each developer's Betweenness over a file's history
Number of Hub Developers	The number of distinct hub developers who update this file

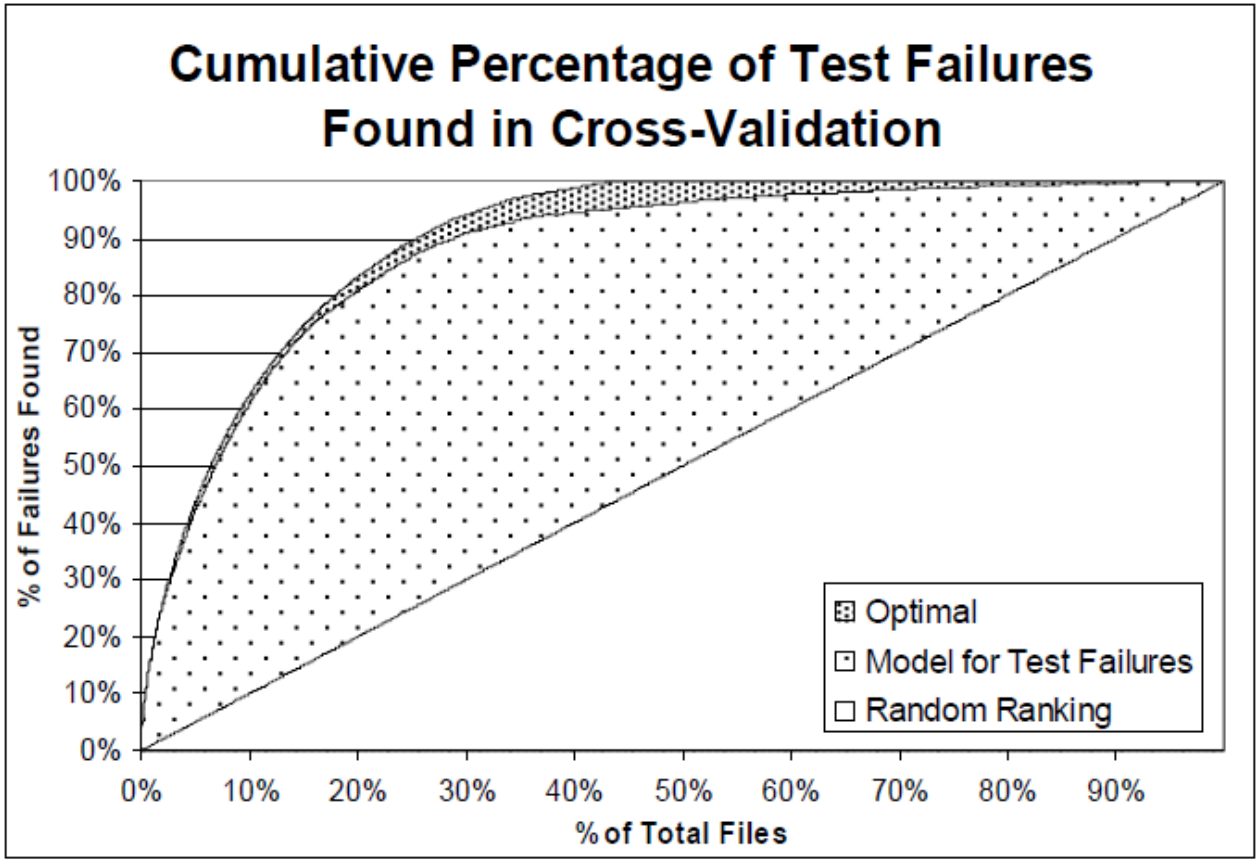
提案モデル①
System Testing Model

用いたメトリクス

Table 5: Variables of the test failure model

Metric	Transformation
Code Churn	Log
Updates	None
Developers	Square root
Sum of Degree	Square root
Sum of Closeness	None

評価実験
（System Testing Modelの場合）



Optimalとほぼ同じ割合で不具合を発見できた

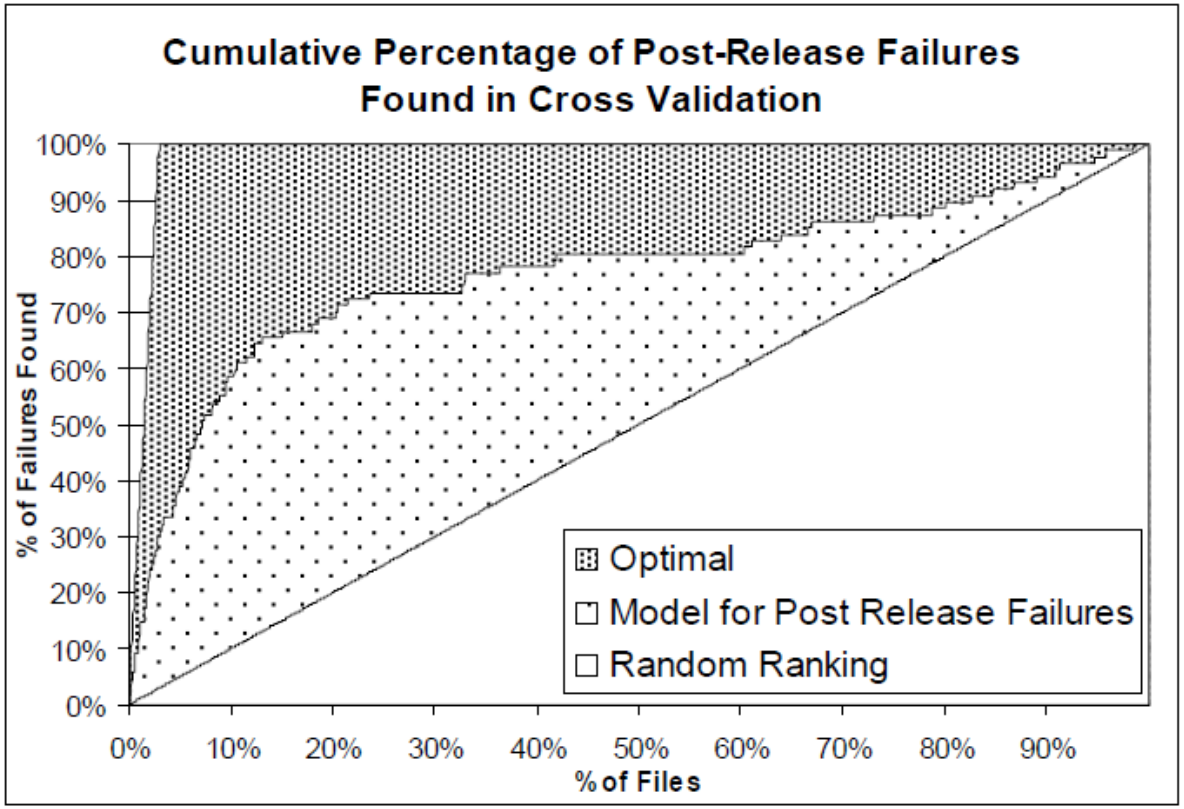
提案モデル②
Post-Release Model

用いたメトリクス

Table 6: Variables to metrics in the post-release failure model

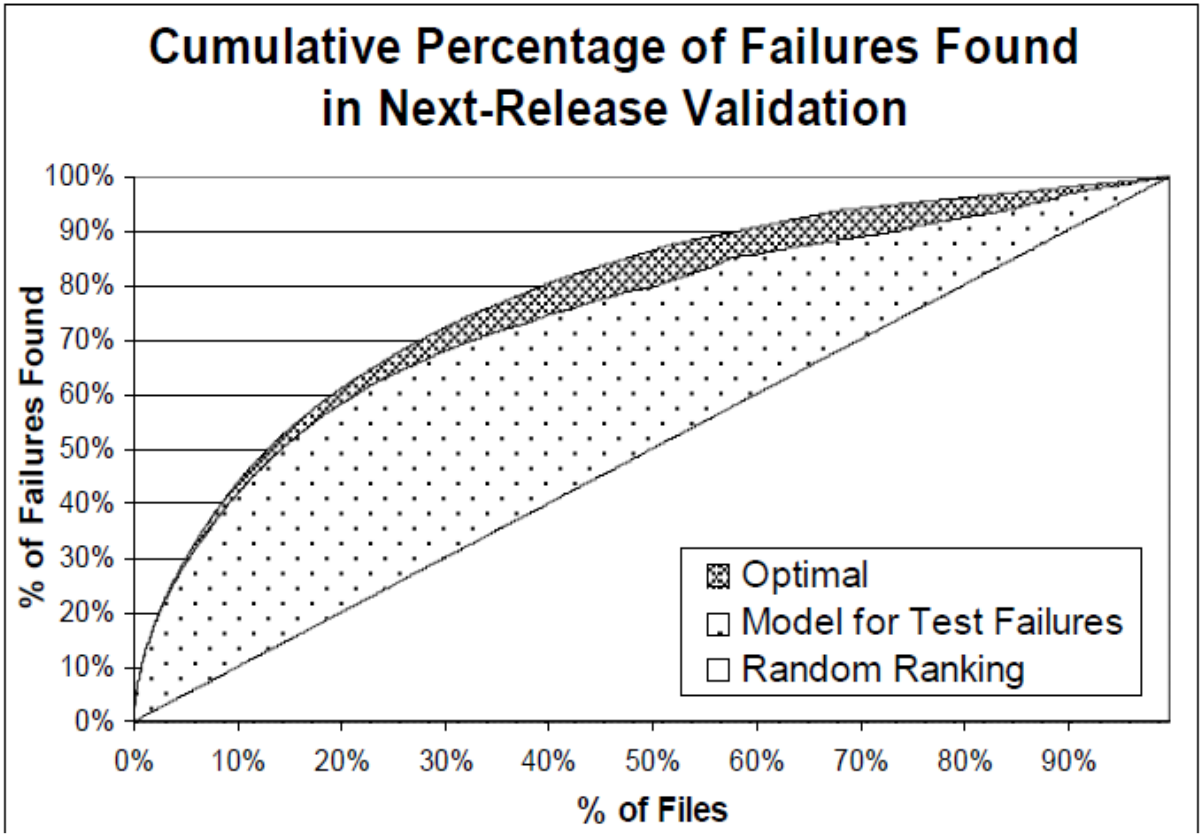
Metric	Transformation
Updates	None
Developers	Square root
Sum of Degree	Square root
Sum of Closeness	None

評価実験
（Post-Release Modelの場合）



ランダム手法と比較すると高い精度ではあったが
Optimalよりも発見率は低かった

System Testing Modelの評価実験



Optimalとわずかに同じ発見率だった

System Testing Modelの評価実験

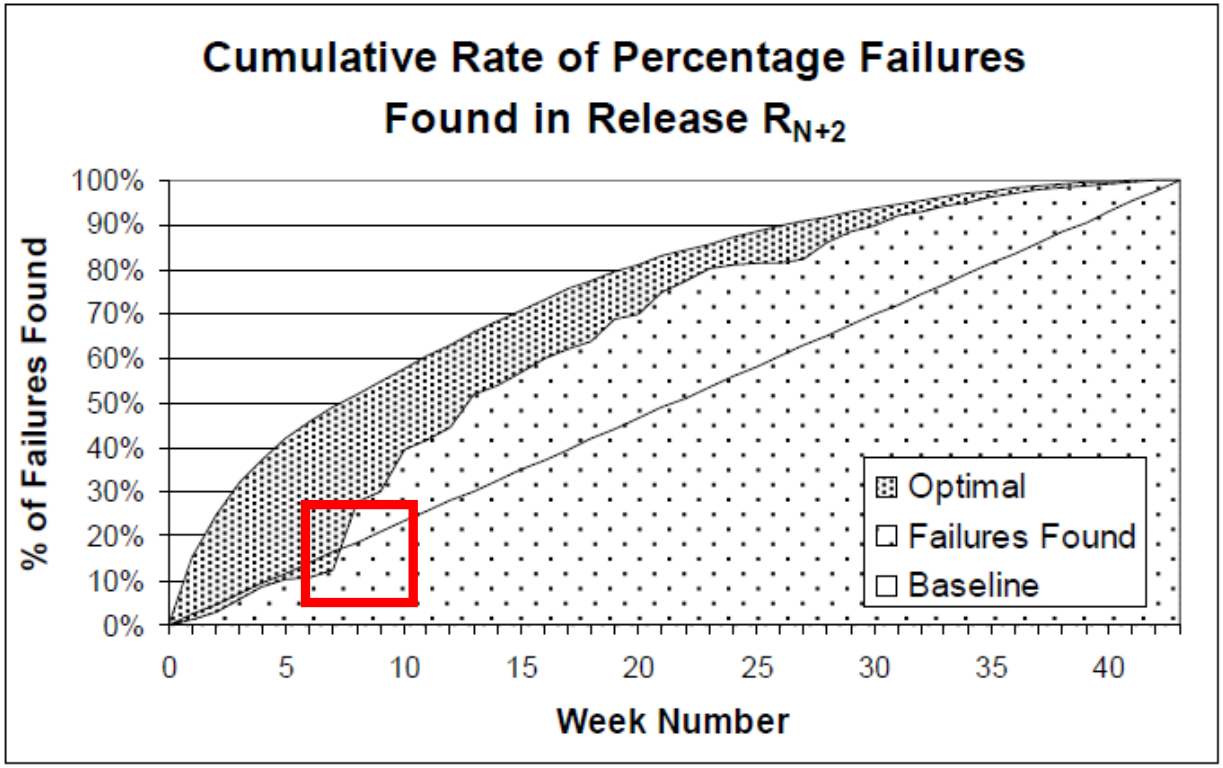
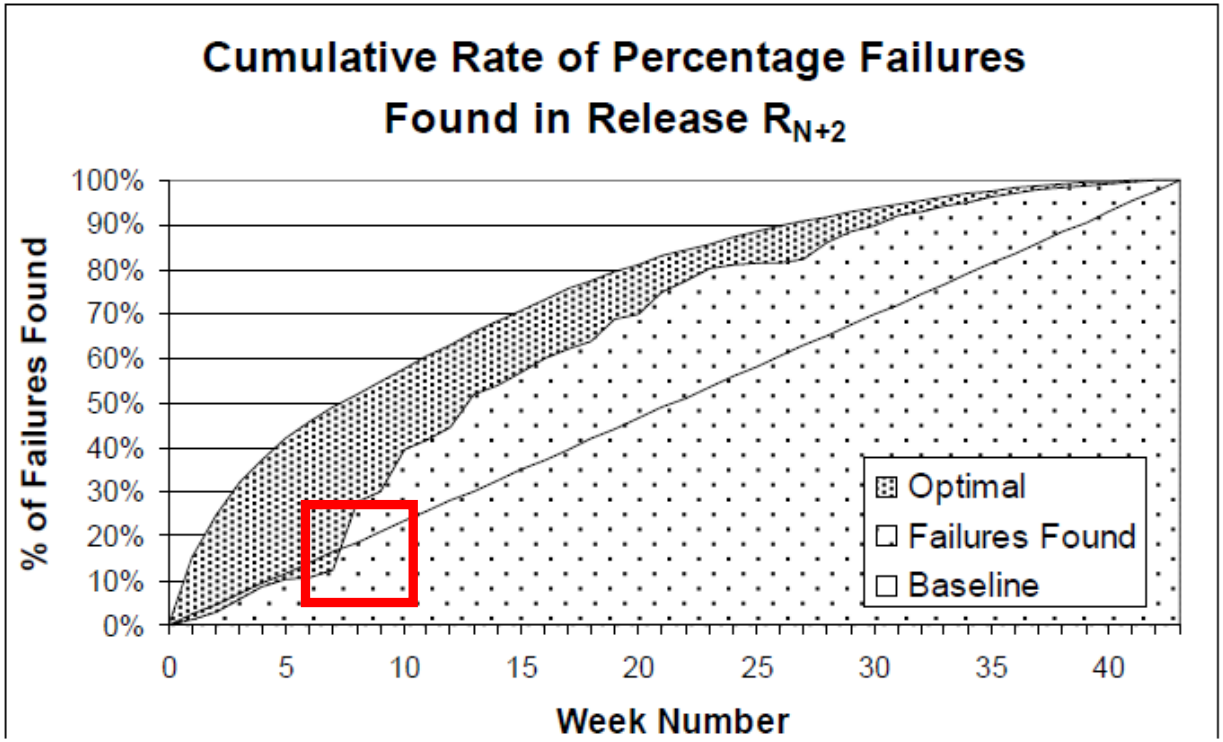


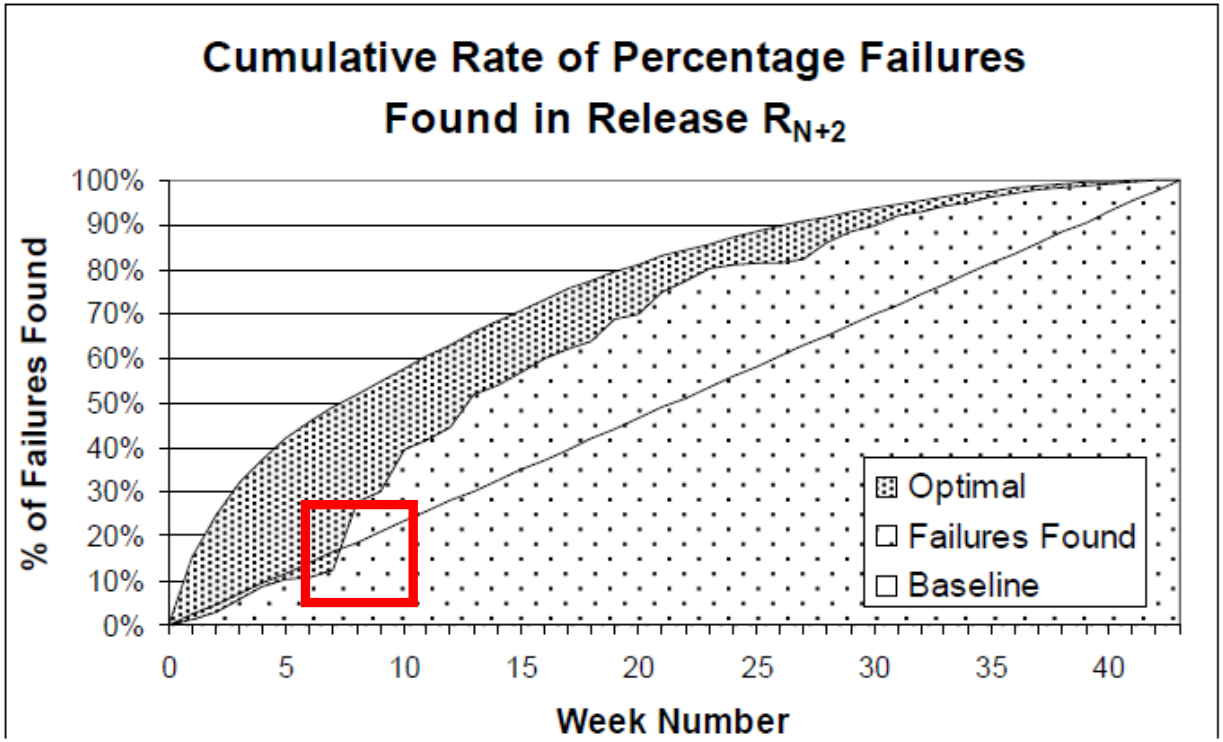
Figure 5: Cumulative percentage of observed failures found over time during R_{N+2}

System Testing Modelの評価実験



8週目で提案モデルで見つけた不具合の割合が
baselineの割合を超えている

System Testing Modelの評価実験



提案モデルは多くの不具合を
早い段階で検出するのには適している

追加分析

Table 8: Spearman correlation coefficients from cross-validation of the training set

	Average	Standard Deviation
SLOC Model	0.400	0.03
Code Churn Only	0.706	0.10
Network Metrics Only	0.514	0.05
System Test Model	0.778	0.03

コードチャムのみ，ネットワークメトリクスのみよりも
どちらも合わせたモデルの方が高い精度を出している

まとめと所感

- まとめ
  - 開発者のネットワークを考慮した不具合予測モデルを提案した
  - 不具合発見率は，System Test Modelではoptimalと同じくらい，Post Release Modelは低い
  - 提案モデルでは多くのバグを早い期間で見つけることができる
- 所感
  - コミュニティとバグを関連付けたのおもしろい
  - バグ関連の論文を読んでこなかったので読みにくかった