

Predicting Failures with Developer Networks and Social Network Analysis



Andrew Meneely
Laurie Williams



Will Snipes
Nortel Networks, Research



Jason Osborne
Nortel Networks,

Department of Computer Science, Triangle Park, NC, USA. Department of Statistics, North Carolina State University, Raleigh, NC, USA. wbsnipes@nortel.com

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
- 多くの研究で高い制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
- 開発者の情報(ex.コード変更の数)をメトリクスに取り入れたモデル構築
- 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ことを明らかにした

開発者のネットワークに着目した
予測モデルの構築はされていない

目的とアプローチ

- 目的
 - コードチャームに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を追加した予測モデルの構築
 - 既存モデルと本研究で構築したモデルの比較

SNA

- 本論文におけるコネクト
 - 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
 -
- メトリクス
 - degree:繋がっているノードの数
 - betweenness:ネットワークの中心にどうか
 - closeness:全ノードへの距離の平均の重さ

ケーススタディ

- データセット
 - 対象プロジェクト
 - ノーテルネットワークスのプロダクト
 - 対象リビジョン
 - 3年間の年次リリースの3つのリビジョン
 - R_n, R_{n+1}, R_{n+2} で表す
 - 学習データ
 - 2,500ファイル/11,000ファイル

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

評価実験 (System Testing Modelの場合)

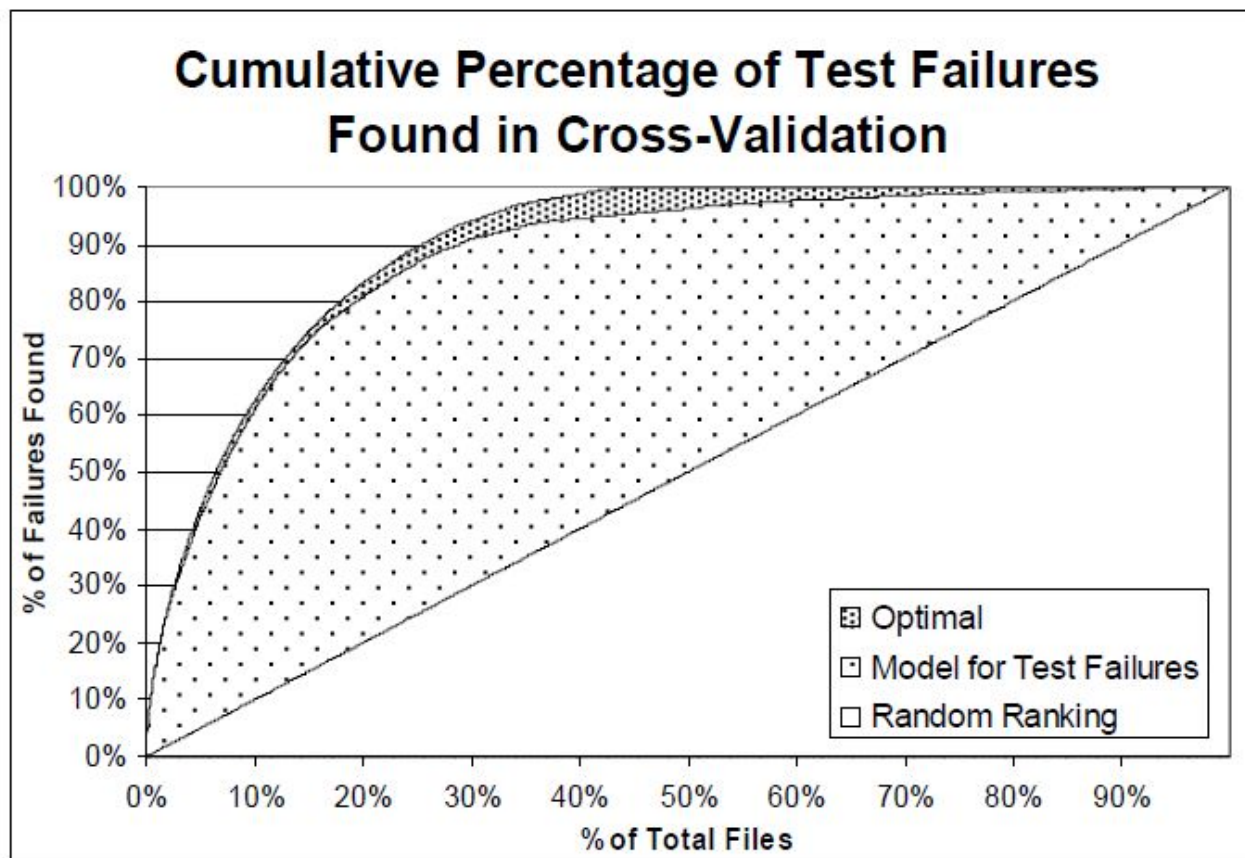
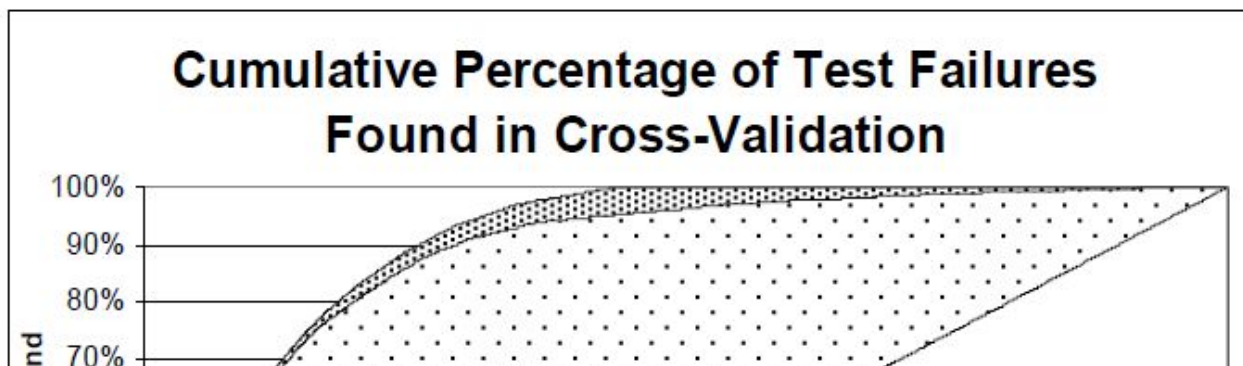


Figure 2: Cumulative percentage of test failures found based on the prioritizations of the cross-validation predictions

評価実験 (System Testing Modelの場合)



既存手法とほぼ同じ制度で予測を行うことができた

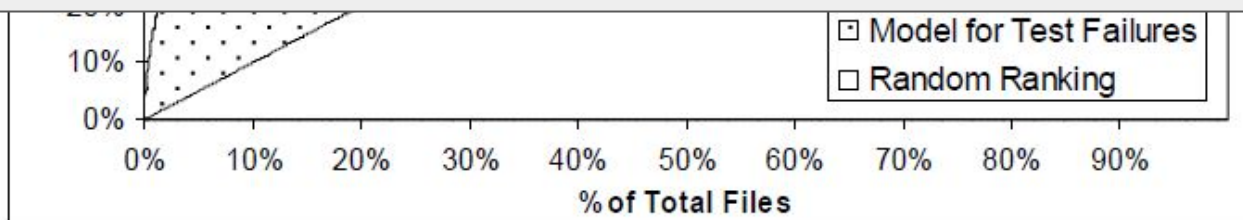


Figure 2: Cumulative percentage of test failures found based on the prioritizations of the cross-validation predictions

評価実験 (Post-Release Modelの場合)

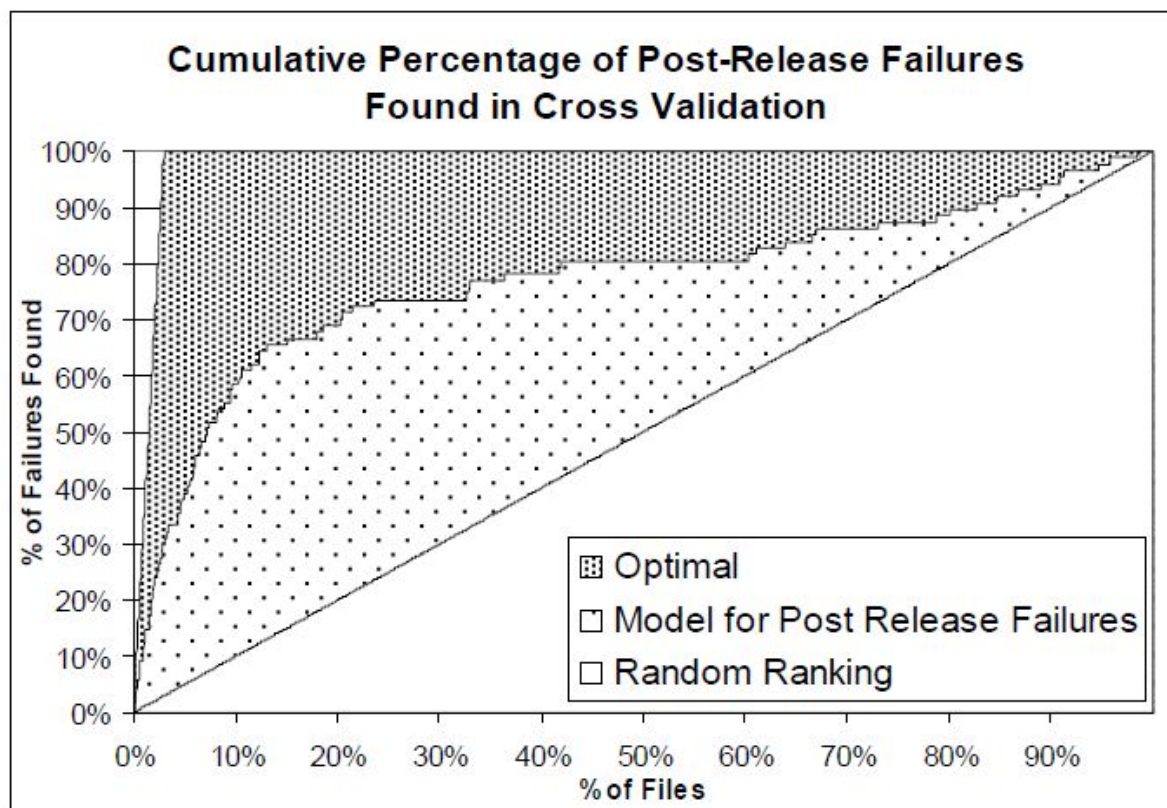
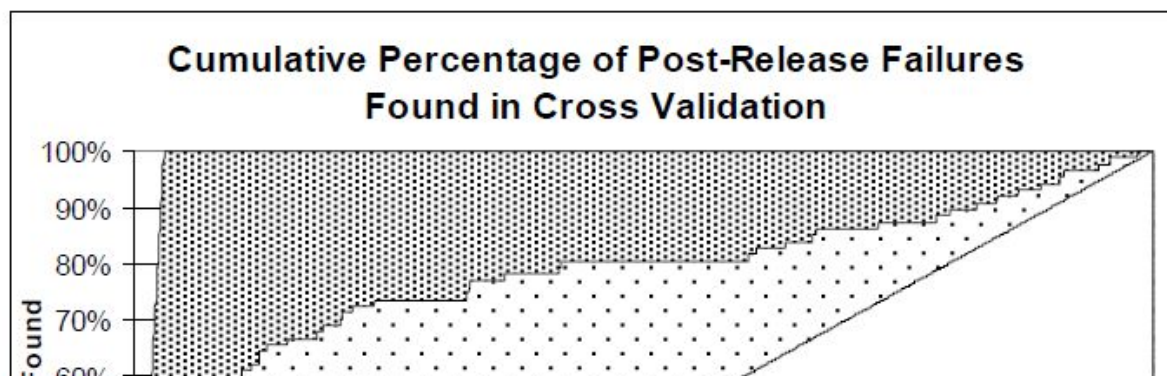


Figure 3: Cumulative percentage of post-release failures found based on the prioritizations of the cross-validation predictions and the optimal prioritization

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
既存手法よりも予測精度は低かった

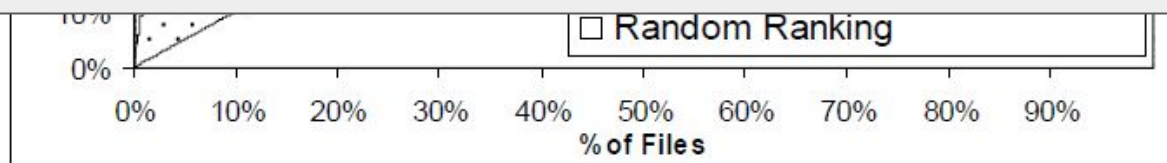


Figure 3: Cumulative percentage of post-release failures found based on the prioritizations of the cross-validation predictions and the optimal prioritization

評価実験(R_{N+2} の場合)

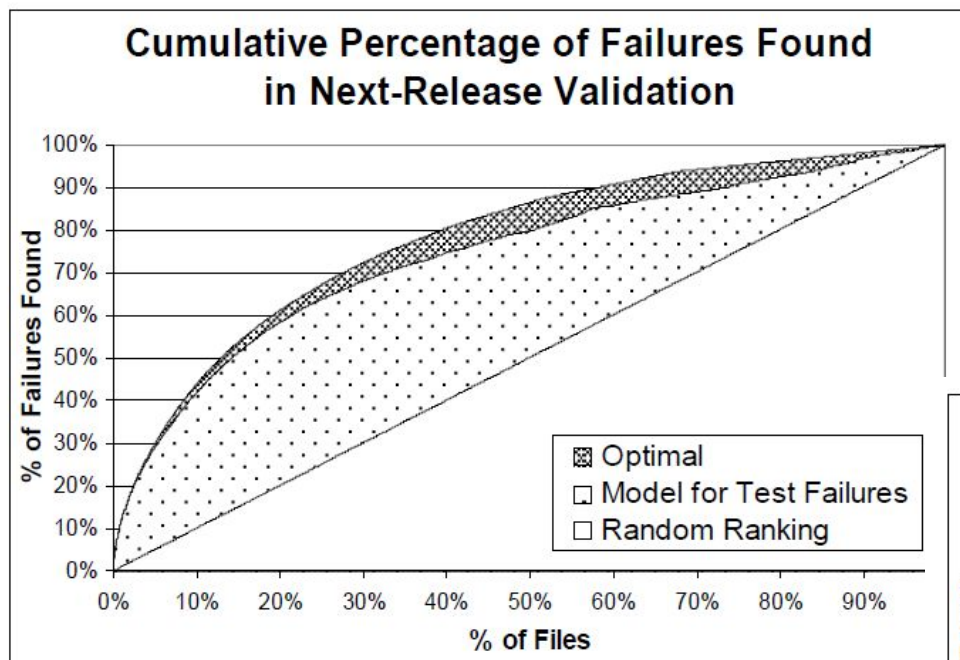


Figure 4: Cumulative percentage of test failures found on p existing files in R_{N+2} .

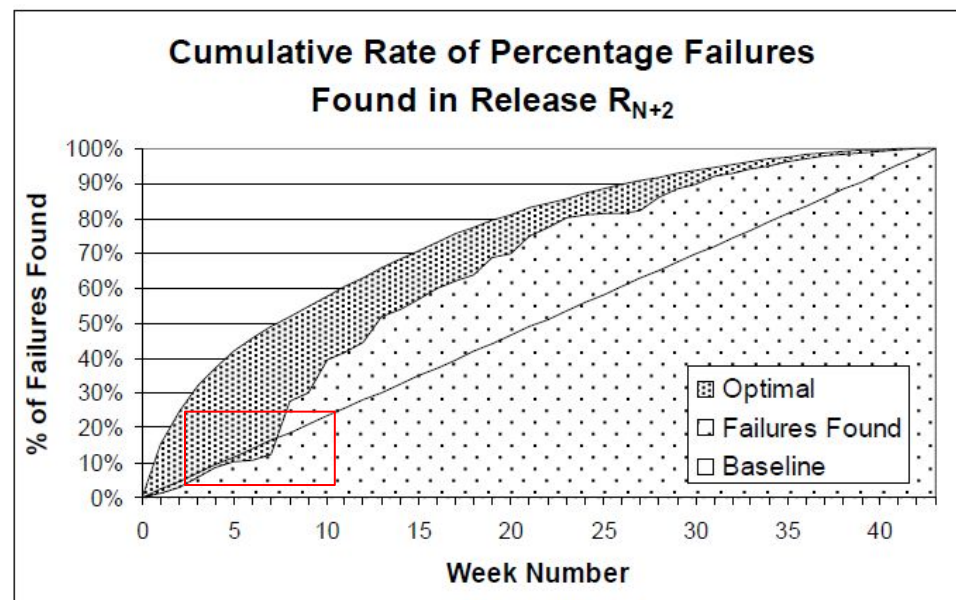
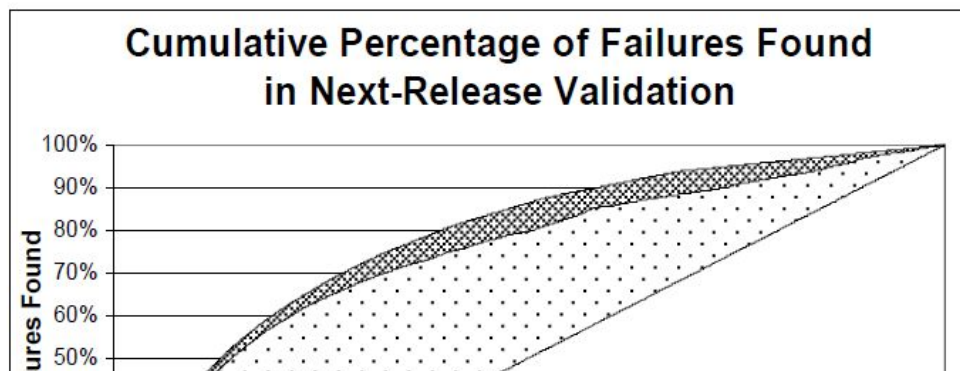


Figure 5: Cumulative percentage of observed failures found over time during R_{N+2}

評価実験(R_{n+2} の場合)



提案手法は多くの不具合を素早く見つけるのには最も適している

Figure 4: Cumulative percentage of test failures found on p existing files in R_{N+2} .

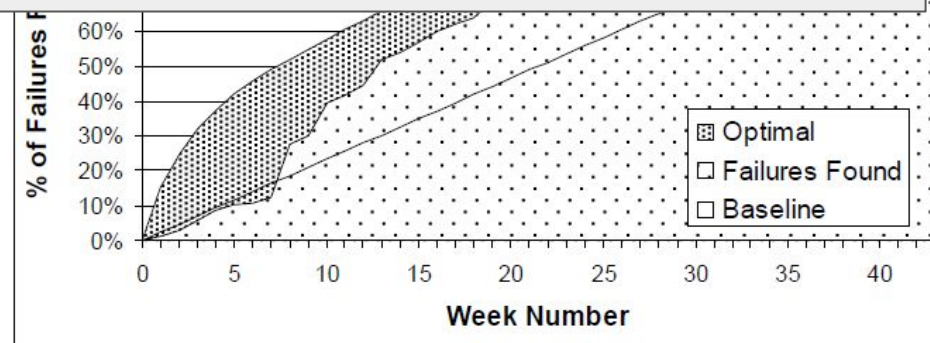


Figure 5: Cumulative percentage of observed failures found over time during R_{N+2}

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
 - next-releaseの場合は既存手法よりも早い段階で不具合予測をすることができた
- 所感
 - コミュニティとバグの研究おもしろそう