

モジュールごとの活動量を考慮した モジュールオーナー候補者予測

M3 山谷 陽亮
2016/04/12

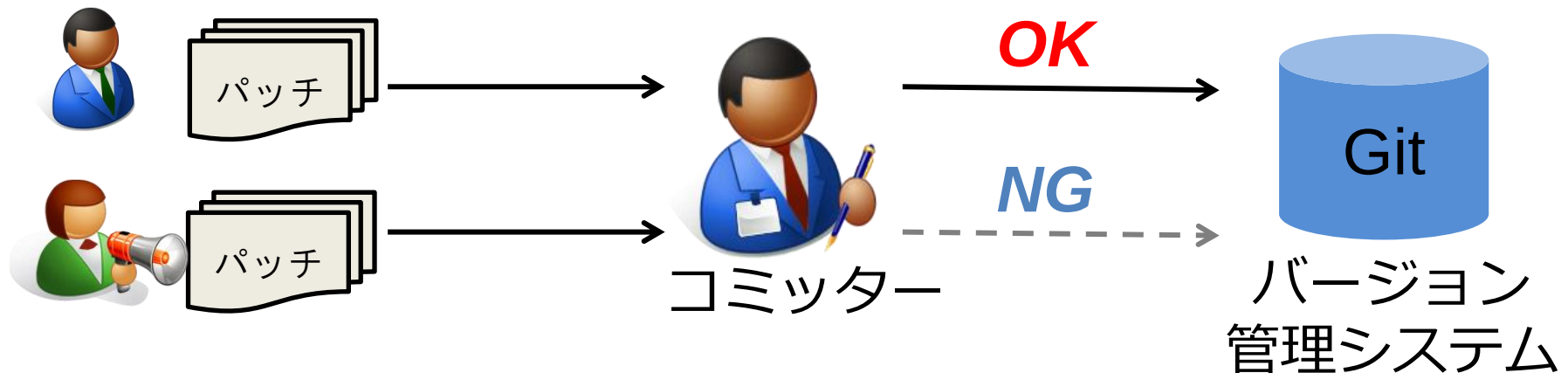
背景

- オープンソースソフトウェア(OSS)の普及によりOSSのユーザーが年々増加している
- 大規模OSSプロジェクトでは日々大量の不具合が報告されている[1]
- 報告された不具合はボランティアの開発者が中心となり修正が行われている

[1] John Anvik, Lyndon Hiew, and Gail C. Murphy. Who should x this bug?
In Proceedings of the 28th International Conference on Software Engineering (ICSE'06), pp. 361--370, 2006.

不具合修正プロセス

- 不具合を修正するためにパッチが投稿される
 - パッチ：コードの修正前後の差分を示すファイル
- コミッターによってパッチが検証される
 - コミッター：コードを変更する権限を持つ開発者



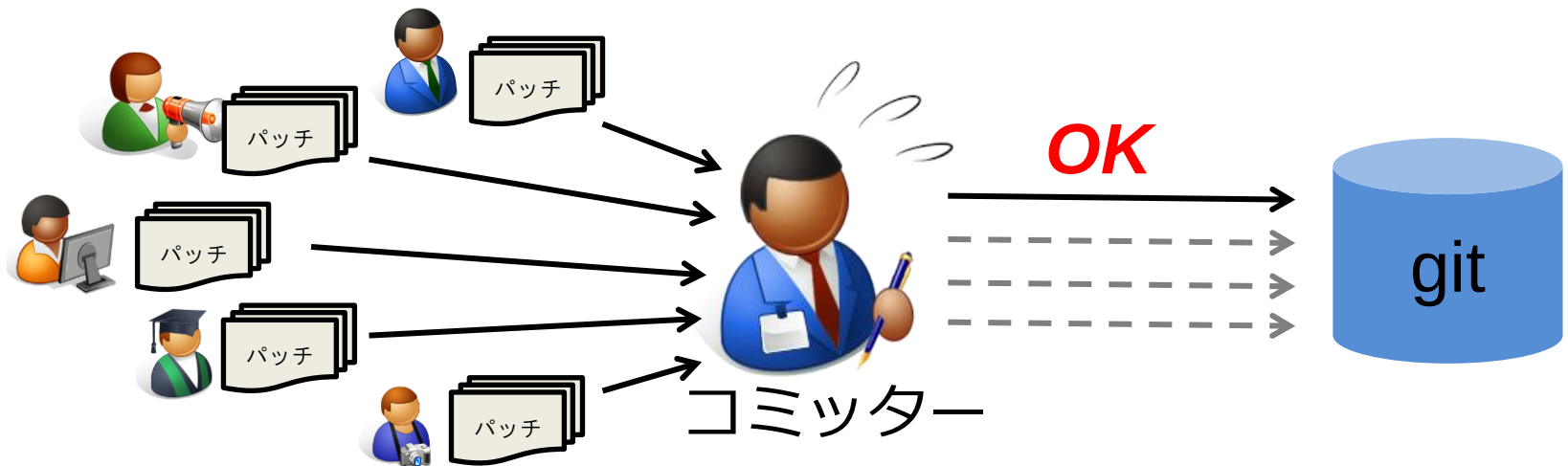
コミッター昇格

- コミッターはソフトウェアの品質管理の面で重要な役割を担っている
- 開発者にコミット権限を付与するかどうかの判断は慎重に行われ、既存のコミッターからの推薦や承認によってコミット権限が付与される [2]
- コミット権限を持たない開発者に比べコミッターは少人数で構成されている

[2] C. Jensen and W. Scacchi. Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study. In Proceedings of the 29th International conference on software engineering (ICSE'07), pp.364-374, 2007

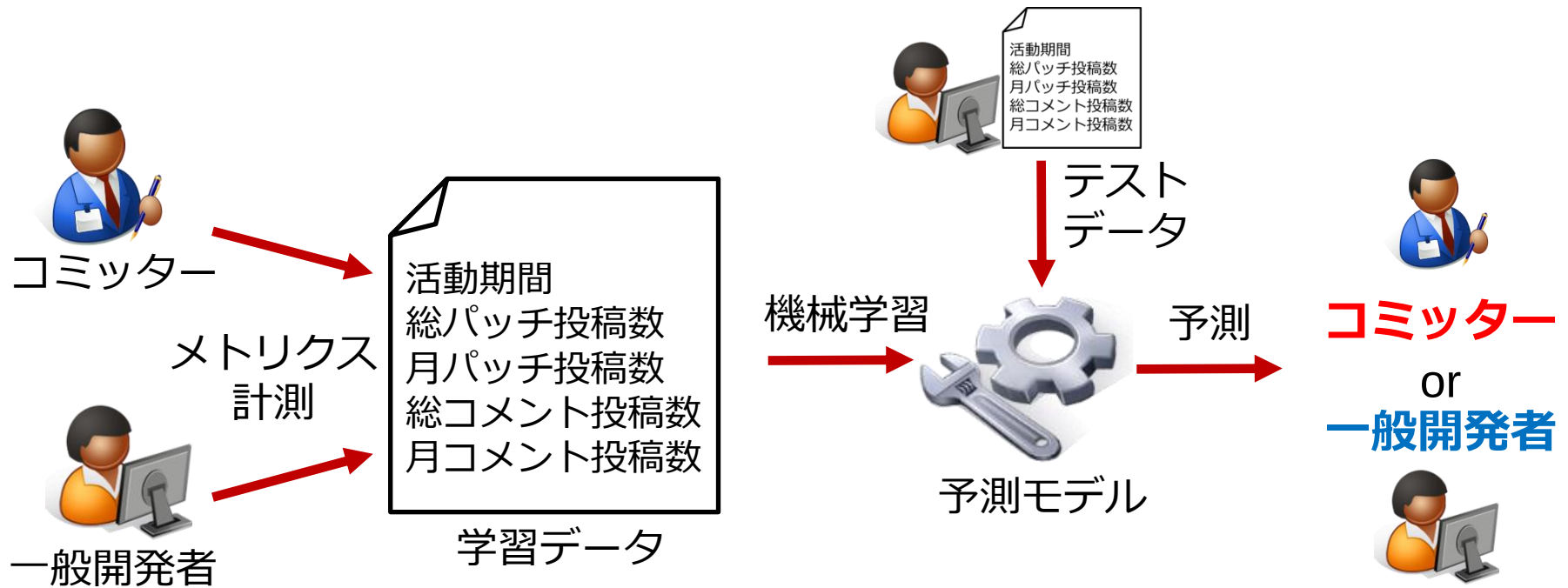
動機：コミッター不足

- 大規模OSSプロジェクトでは少人数で構成されるコミッターに過度な負担がかかっている
- コミッターに過度な負担がかかるとパッチの検証を効率的に行うことができなくなる



先行研究

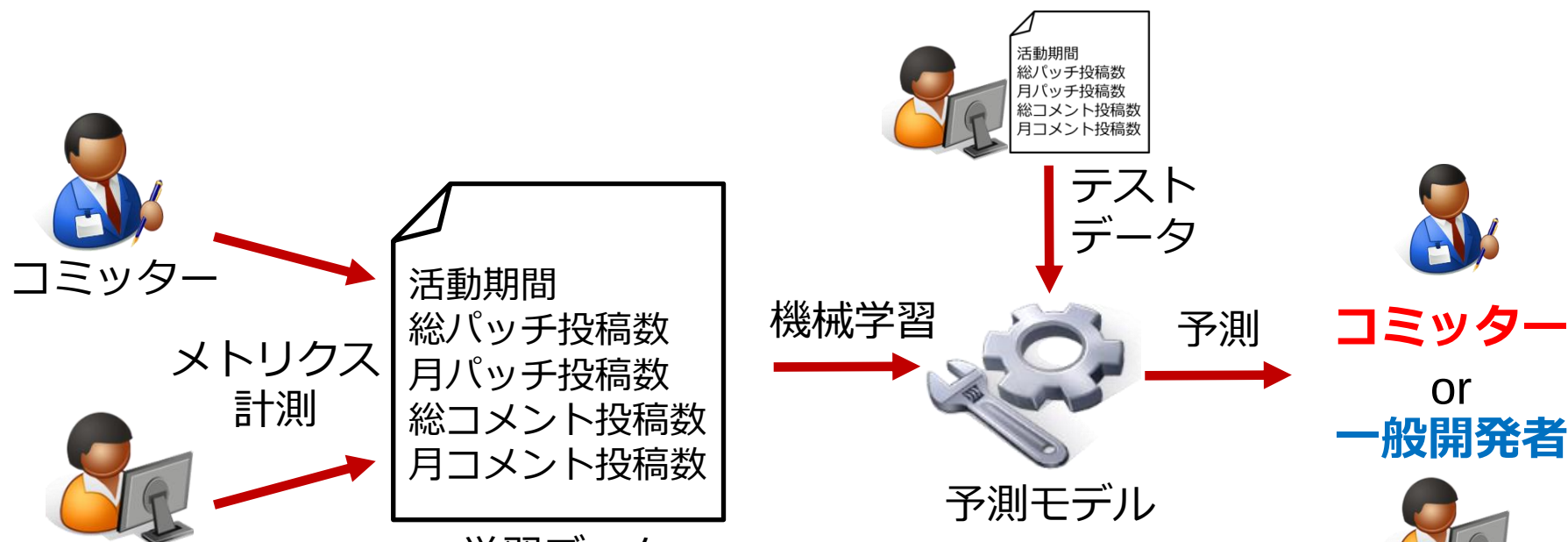
- コミッター不足を解消するためにコミッターに昇格可能な開発者を予測する研究が行われる[3]



[3] Akinori Ihara, Yasutaka Kamei, Masao Ohira, Ahmed E. Hassan, Naoyasu Ubayashi, and Kenichi Matsumoto. Early identification of future committers in open sourcesoftware projects. In Proceedings of the 14th International Conference on Quality Software (QSIC'14), pp. 47--56, 2014.

先行研究

- コミッター不足を解消するためにコミッターに昇格可能な開発者を予測する研究が行われる[3]

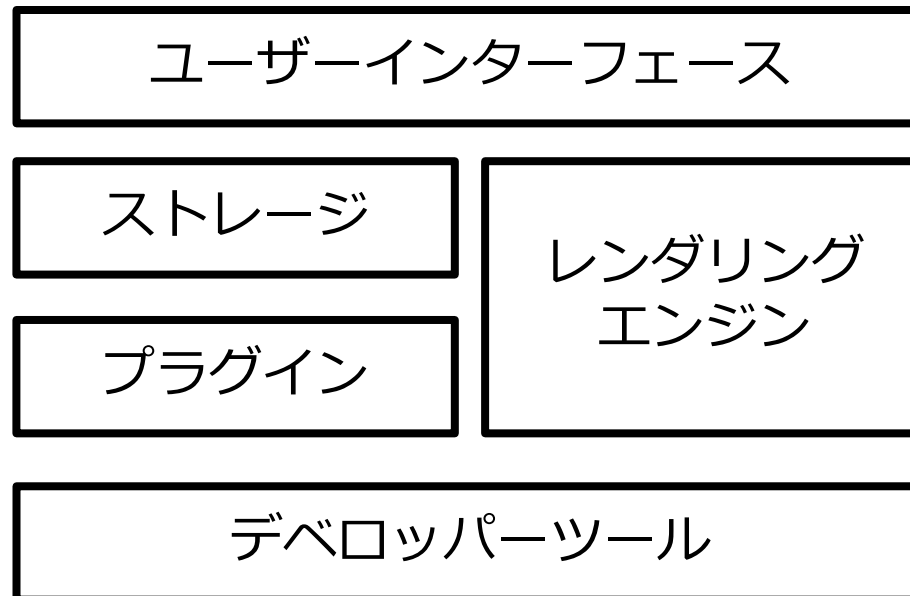


不具合修正の効率化のためには
モジュールごとにコミッターを予測する必要がある

ソフトウェアモジュール

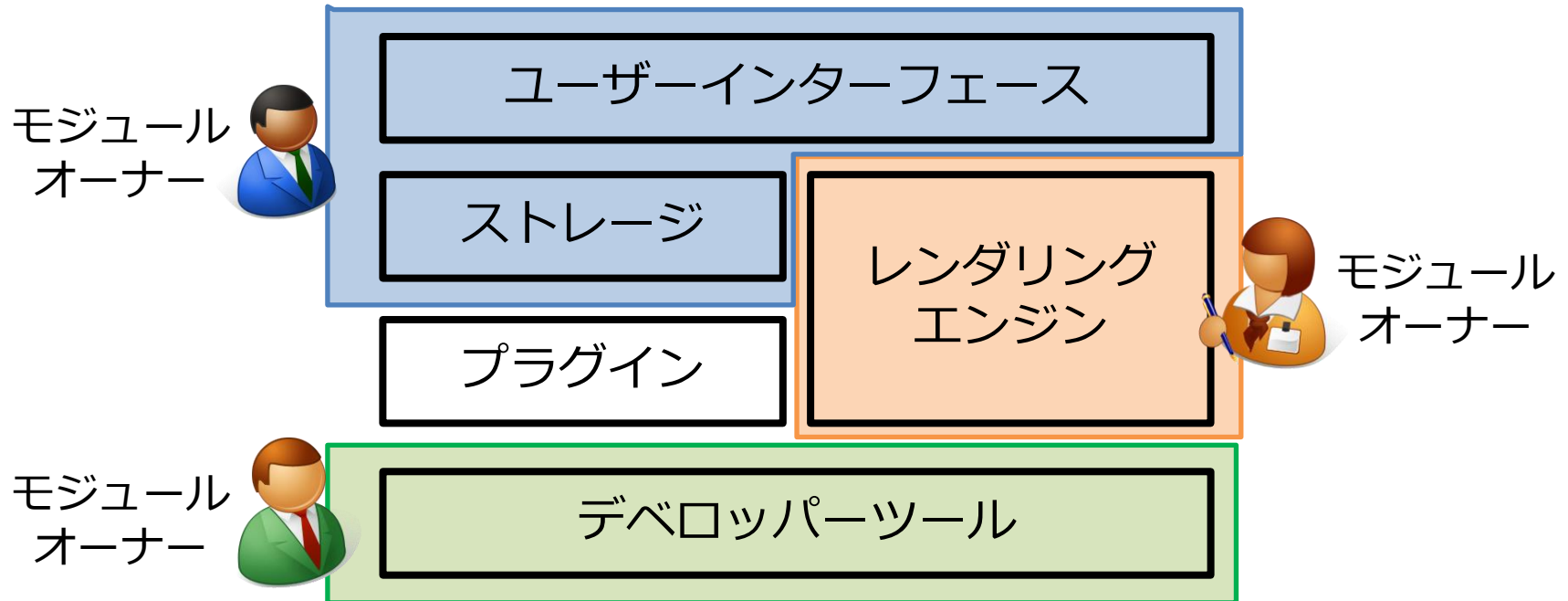
- ソフトウェアは保守性や再利用性を高めるために複数のモジュールが組合わさり構成される
 - モジュール：ひとまとまりの機能をもつ部品

例：ウェブブラウザのモジュール



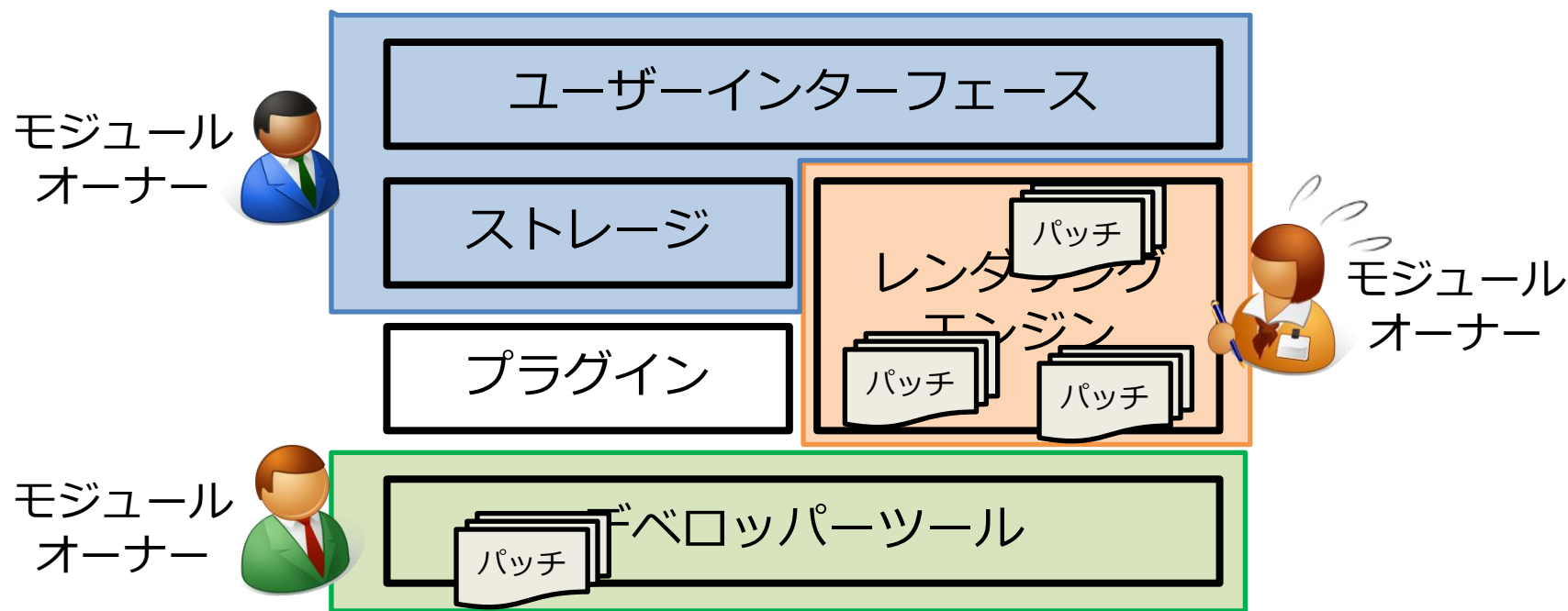
モジュールオーナー

- 大規模OSSプロジェクトでは各コミッターが担当すべきモジュールが決められている
 - モジュールオーナー：**
 - 特定のモジュールを担当するコミッター



モジュールオーナー候補者

- 不具合修正の効率化には人手が不足している**モジュールのオーナーに適した開発者（モジュールオーナー候補者）**を推薦することが必要となる



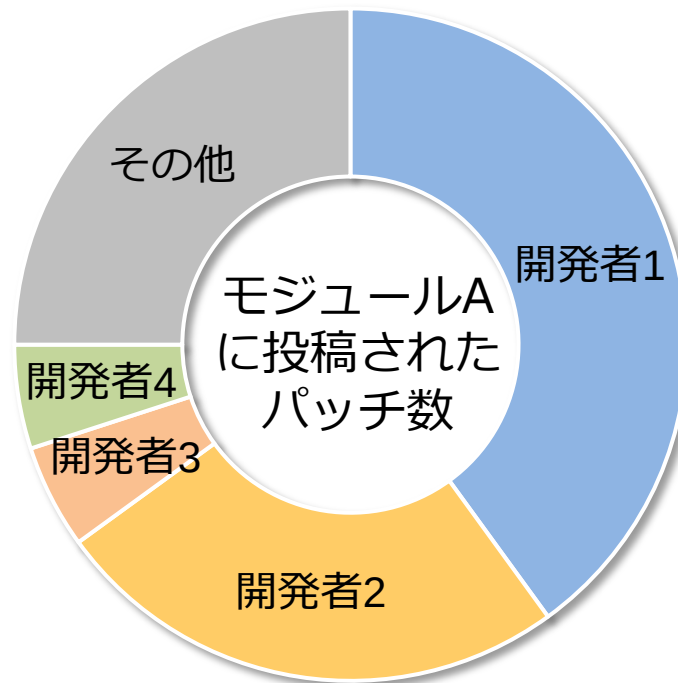
本研究の目的

- 目的
 - モジュールオーナーの不足を解消するために、モジュールオーナー候補者を推薦する
- アプローチ
 - モジュールオーナーの適性を示すメトリクスを定義
 - モジュールオーナー候補者を特定するための予測モデルを構築

モジュールオーナーの適性

- 仮説

- パッチ投稿の割合が高い開発者ほどモジュールの変更作業に関わった実績があるためモジュールオーナーの適性が高いのでは？



メトリクスの定義

- パッチ貢献度 (PCR : Patch Contribution Rate)
 - 特定のモジュールに占める投稿したパッチの割合
 - $\frac{\#patchDev}{\#patchModule}$ の値はべき分布となるため対数変換を行う

$$PCR = \log_e \left(1 + \frac{\#patchDev}{\#patchModule} \right)$$

$\#patchDev$: 開発者がモジュールに対して投稿したパッチの数

$\#patchModule$:

(コミッター) 昇格までにモジュールに投稿されたパッチの総数

(一般開発者) 最後に活動した日までに投稿されたパッチの総数

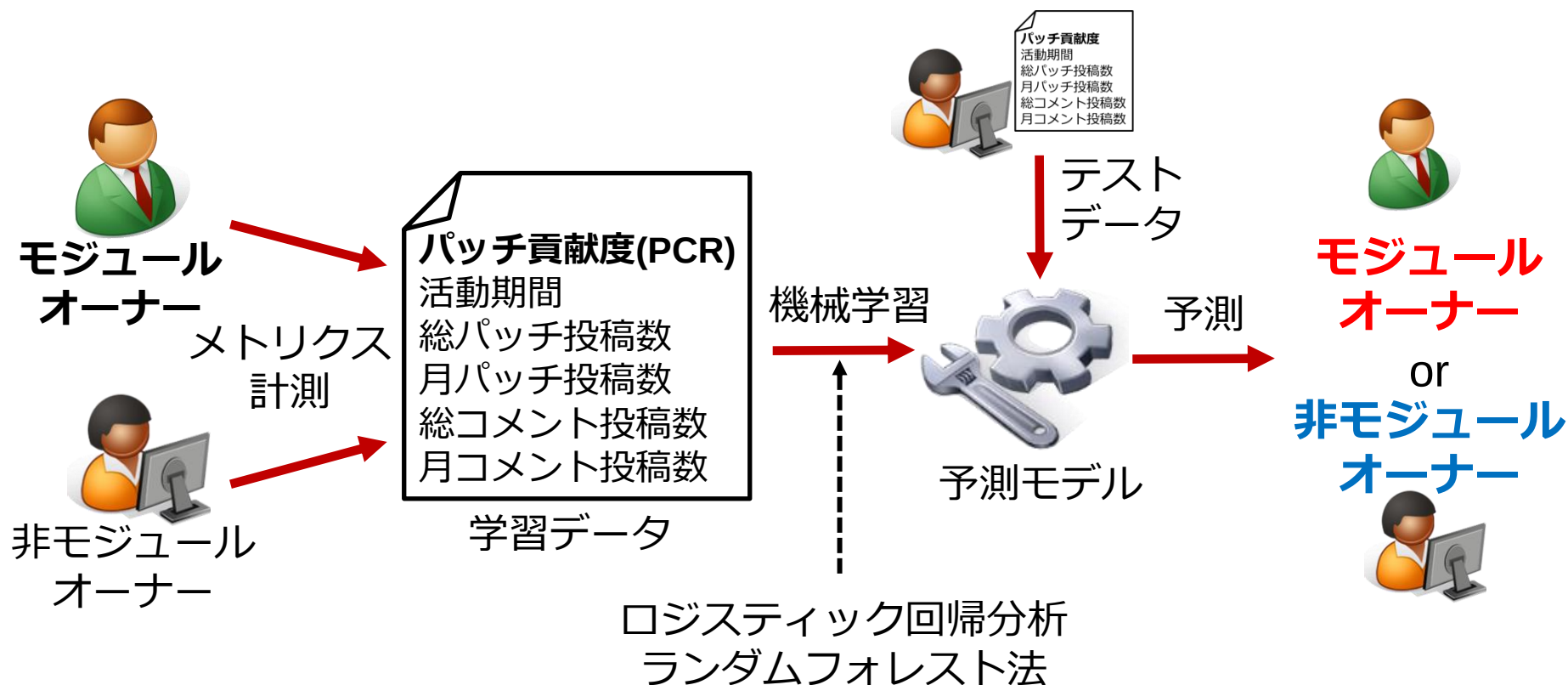
モジュールごとの活動量メトリクス

- 先行研究で用いられている活動量メトリクスをモジュールごとに計測する
- モジュールごとに計測される活動量メトリクス

メトリクス名	内容
総パッチ投稿数	各開発者が特定のモジュールに対して投稿したパッチの回数
月パッチ投稿数	各開発者が特定の1か月にパッチを投稿した回数の中央値
総コメント投稿数	各開発者が特定のモジュールに対して投稿したコメントの回数
月コメント投稿数	各開発者が1か月にコメントを投稿した回数の中央値
活動期間	各開発者の活動（パッチ・コメントの投稿）が実施された月数

モジュールオーナー候補者予測

- パッチ貢献度とモジュールごとの活動量メトリクスを用いてモジュールオーナー候補者を予測



評価実験

- 実験の目的
 - モジュールオーナー候補者を予測する際、パッチ貢献度が予測精度向上に寄与するか確かめる
- 対象プロジェクト

プロジェクト	期間	コミッター候補者	一般開発者
Eclipse Platform	2001/10 ~2012/10	195	13,297
Mozilla Firefox	1999/07 ~2013/01	530	76,931
GCC	1999/07 ~2014/03	141	10,282
GIMP	2001/02 ~2013/09	87	4,068
WebKit	2000/12 ~2014/04	363	8,017

モジュールの特定

- 以下の情報を基にモジュールを特定する
 - ソースコードの構造
 - 不具合管理システムのコンポーネント情報
 - プロジェクトのWikiページ

モジュール名	モジュールオーナー数
App	18人
Bar	66人
Bookmark	54人
Developer Tools	45人
General	73人
Preference	66人
Tab	64人

※Mozilla Firefox
プロジェクトでの例

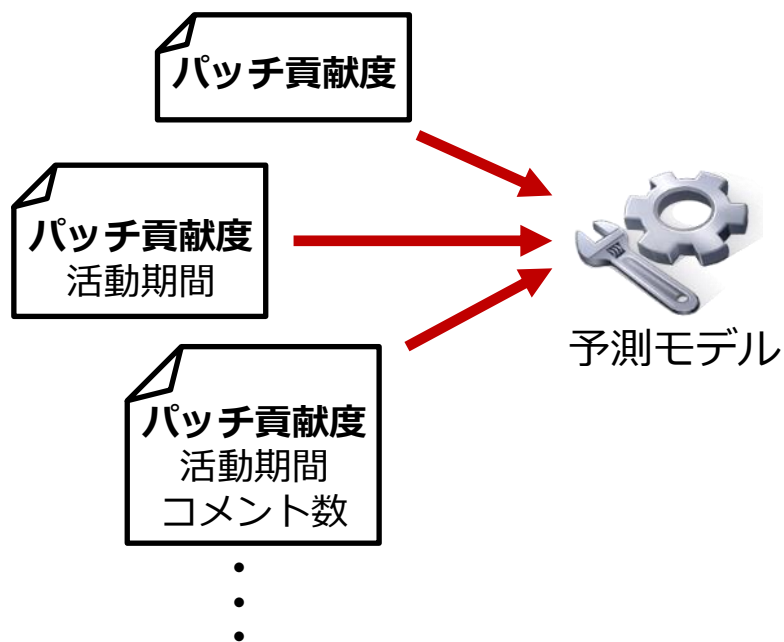
モジュールオーナーの判別

- 特定のモジュールに対して投稿されたパッチをバージョン管理システムに反映しているか否か
- 判別手順
 1. コミットログに含まれる不具合IDを取得する
 - Bug 2314 fixed.
 2. 不具合IDから不具合管理システムのコンポーネント情報を参照しモジュールを特定する
 3. 1, 2の処理を全開発者に対して行う

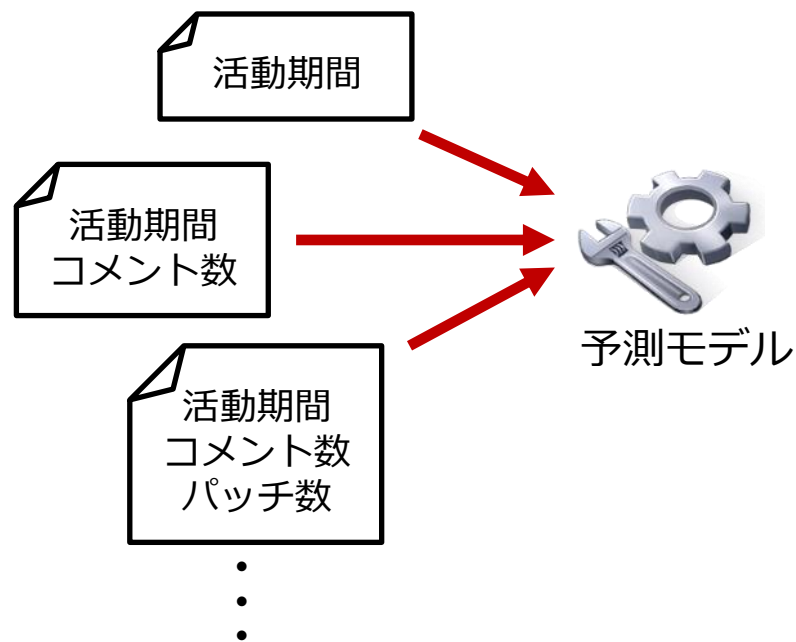
構築する予測モデルと比較評価

- メトリクスのすべての組み合わせでモデル構築
- パッチ貢献度を含むか否かで精度の向上を比較

パッチ貢献度を含むモデル

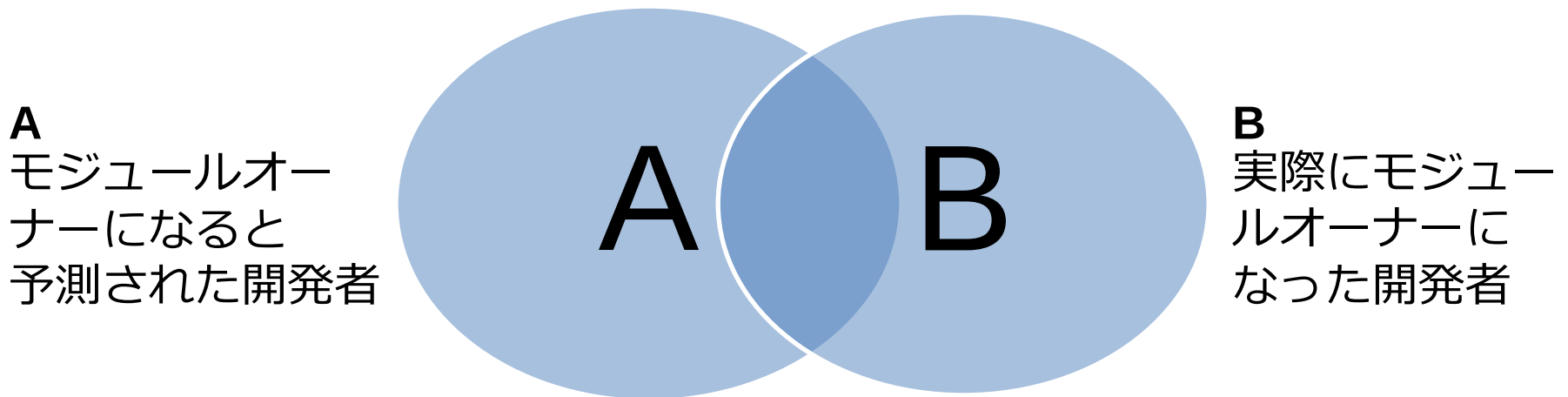


パッチ貢献度を含まないモデル



評価指標

- 予測モデルを適合率, 再現率, F1値で評価する



$$\text{適合率} = \frac{A \cap B}{A}$$

$$\text{再現率} = \frac{A \cap B}{B}$$

$$\text{F1値} = \frac{2 \times \text{適合率} \times \text{再現率}}{\text{適合率} + \text{再現率}}$$

実験手順

1. モジュールオーナーと非モジュールオーナーのそれぞれのデータを2分割し、学習データとテストデータを作成する
2. モジュールオーナーの数だけ非モジュールオーナーの学習データからランダムに抽出し学習
3. テストデータの全てで予測精度を算出する
4. 1から3の手順を100回繰り返す

結果：ロジスティック回帰分析

プロジェクト	PCR有無	予測モデル	適合率	再現率	F1値
Eclipse Platform	PCR有	PCR	0.146	0.967	0.248
	PCR無	パッチ数	0.107	0.856	0.183
Mozilla Firefox	PCR有	PCR	0.278	0.992	0.412
	PCR無	パッチ数	0.276	0.912	0.409
GCC	PCR有	PCR	0.074	1.000	0.129
	PCR無	パッチ数	0.031	0.917	0.044
GIMP	PCR有	PCR	0.124	1.000	0.219
	PCR無	パッチ数	0.115	0.813	0.201
WebKit	PCR有	PCR	0.213	0.880	0.332
	PCR無	パッチ数	0.189	0.905	0.308

- PCRのみを用いたモデルが最も精度が高かった
- モジュールオーナーを漏れなく予測できた

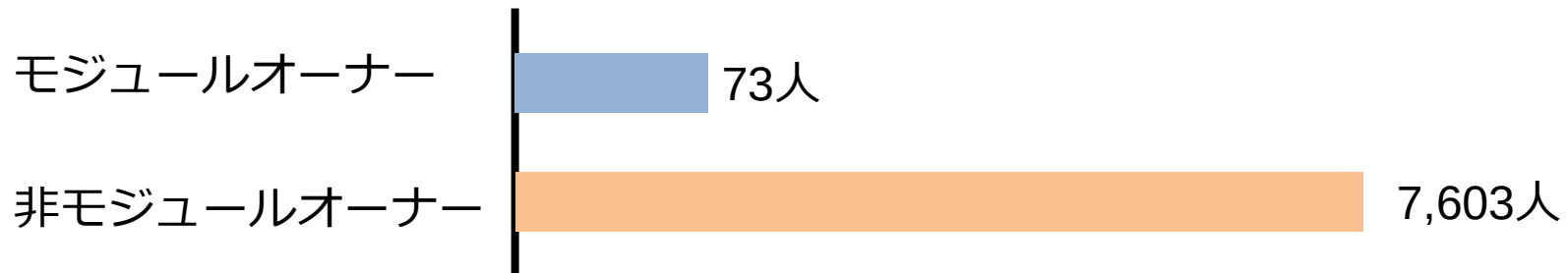
結果：ランダムフォレスト法

プロジェクト	PCR有無	予測モデル	適合率	再現率	F1値
Eclipse Platform	PCR有	PCR	0.124	0.967	0.218
	PCR無	活動期間+パッチ数	0.116	0.909	0.196
Mozilla Firefox	PCR有	PCR	0.267	1.000	0.404
	PCR無	活動期間+パッチ数	0.267	1.000	0.404
GCC	PCR有	PCR	0.074	1.000	0.129
	PCR無	先行研究のモデル	0.017	0.938	0.033
GIMP	PCR有	PCR+活動期間	0.129	0.938	0.225
	PCR無	パッチ数	0.123	0.875	0.213
WebKit	PCR有	PCR+活動期間	0.168	0.987	0.284
	PCR無	活動期間+パッチ数	0.165	0.989	0.280

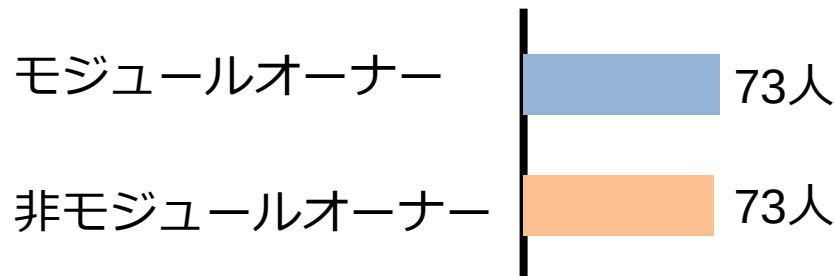
- PCRはモジュールオーナー候補者を予測する際
予測精度向上に寄与することがわかった

考察

- 適合率が低い原因はモジュールオーナーに比べて非モジュールオーナーの数が膨大であるため



- テストデータに含まれるモジュールオーナーと非モジュールオーナーのデータ数をそろえる



結果：ロジスティック回帰分析

プロジェクト	PCR有無	予測モデル	適合率	再現率	F1値
Eclipse Platform	PCR有	PCR	0.789	0.518	0.625
	PCR無	パッチ数	0.762	0.470	0.581
Mozilla Firefox	PCR有	パッチ数+PCR	0.664	0.438	0.528
	PCR無	パッチ数	0.699	0.435	0.536
GCC	PCR有	パッチ+コメント+PCR	0.969	0.927	0.948
	PCR無	コメント数	0.556	0.625	0.588
GIMP	PCR有	PCR	0.528	0.575	0.550
	PCR無	先行研究のモデル	0.501	0.504	0.502
WebKit	PCR有	PCR	0.750	0.506	0.604
	PCR無	パッチ数	0.709	0.481	0.573

- PCRを用いたモデルの適合率は0.528～0.969
- PCRを含まないモデルに比べ3.5%～74.3%向上

結果：ランダムフォレスト法

プロジェクト	PCR有無	予測モデル	適合率	再現率	F1値
Eclipse Platform	PCR有	PCR	0.722	0.662	0.691
	PCR無	活動期間+パッチ数	0.690	0.509	0.586
Mozilla Firefox	PCR有	パッチ+コメント+PCR	0.640	0.564	0.600
	PCR無	コメント数+パッチ数	0.632	0.544	0.585
GCC	PCR有	PCR	0.854	0.854	0.854
	PCR無	活動期間+コメント数	0.559	0.375	0.449
GIMP	PCR有	PCRを含む全メトリクス	0.574	0.721	0.639
	PCR無	コメント数	0.571	0.575	0.573
WebKit	PCR有	活動期間+PCR	0.667	0.690	0.678
	PCR無	活動期間	0.651	0.688	0.669

- PCRを用いたモデルの適合率は0.574～0.854
- PCRを含まないモデルに比べ0.5%～52.8%向上

考察：まとめ

- サンプルング方法を工夫することで適合率が向上することがわかった
- 追加実験で行った方法は特定の開発者がモジュールオーナーになるか否かを事前に知っておく必要がある
- 適切なサンプルング方法を検討する必要がある

まとめ

- モジュールオーナーの適性を示すメトリクスであるパッチ貢献度を定義し5つのOSSプロジェクトでモジュールオーナー候補者予測を行った
- 実験の結果、パッチ貢献度がモジュールオーナーに適する開発者を特定するのに有用であることが明らかになった

近日中にすべきこと

- 開発者にインタビュー
 - モジュールオーナーが決められているのか？
 - 決められているとすればどういう人になるのか？
- モジュールオーナーの定義を検討する
 - コミット数が1回の開発者がモジュールオーナー？
- モジュールオーナー昇格前の活動分析
 - 実際のモジュールオーナーは昇格前にどんな活動をしているのか？この分析を基にメトリクスを決定