ソーシャルコーディングにおける有益な提案の抽出について

檀上未来　乃村能成　谷口秀夫
岡山大学大学院 自然科学研究科

DPS157
2013年10月17日

ソーシャルコーディングとは

ソーシャルネットワークの考え方をコーディングに適用する試み

ソーシャルネットワークサービス(SNS)(例：Facebook)

(1) 写真や文章を公開

(2) 他ユーザからのフィードバック

ソーシャルコーディング(例：GitHub)

(1) ソフトウェア開発プロジェクトを公開

(2) 他ユーザからのレビューや改善案

ユーザ同士の交流を促進

ソーシャルコーディングの普及

ユーザ同士の交流を促進

➡ ユーザは以下の2点を把握可能

(1) 他ユーザのコーディングスキル

(2) 他ユーザの興味のある分野

＜ソーシャルコーディングの利点＞

ユーザ同士の評判形成や相互補助の促進

GitHub

ソーシャルコーディングの仕組みを取り入れたサービス

ソフトウェア開発での利用が活発

会員数：350万人
ソフトウェア開発プロジェクト数：600万

＜コーディングに関わる作業＞

(1) ソースコードレビュー
(2) バージョン管理
(3) チケット管理
(4) ソースコードへのパッチの提示

上記の作業をSNSの操作になぞらえて提供

➡ 操作や開発の過程を可視化，記録

目的

SNSとの類似点：

（1）全ユーザが自由に提案できる

（2）他ユーザの前で非合理な対応はできない

類似点に起因する問題が発生：

（1）重複した提案が多く作成され，調整が必要

（2）調整による「ソーシャル疲れ」が発生

プロジェクトオーナーに対する負担が増大

有益な提案とそうでない提案を事前に判定
よく似た提案同士をまとめる
} 自動化したい

GitHubにおけるソーシャルコーディング

＜GitHubにおける開発の流れ＞

(1) ユーザ登録

　(A) プロフィールページを取得

(2) ソフトウェア開発プロジェクトを作成

　(A) ソースコードの共有スペース(repository)

　(B) 全ユーザが自由に書き込める掲示板(issues)

(3) 他ユーザのプロジェクトのソースコードに手を加える

　(A) 元プロジェクトのrepositoryをコピー(fork)

　(B) fork先プロジェクトを自由に改変

　(C) fork元プロジェクトに改良後のソースコードの取り込みを要求(pull request)

GitHubにおける提案

提案 = プロジェクトへのバグの発見報告や新機能の提案

issue：ソースコードの変更を伴わない提案

pull request：ソースコードの変更を伴う提案

＜GitHubにおける提案の処理の流れ＞

(1) 提案を作成

(2) 提案に用意される掲示板で提案の処遇について議論
（open状態）

(3) 議論が終了した場合，提案をclose

(4) さらに議論が必要な場合，提案をreopen

ソーシャルコーディングの問題点

問題を持った無益な提案が作成される

＜無益な提案＞

(1) 過去の提案と内容が重複した提案

(2) 提案者のミスが原因の不具合の報告

(3) プロジェクトで扱うべきでない問題や機能の報告

提案が有益かどうか判断する作業に時間がかかる

➡ 有益な提案の自動抽出，提示は有用

有益/無益な提案とは？

GitHubのプロジェクトの調査

＜調査の目的＞

有益な提案の特徴を明確化

＜調査の手順＞

(1) 調査対象プロジェクトの有益な提案と無益な提案を調査

(2) ソーシャルコーディングの問題点の存在を確認

(3) 調査した有益な提案をもとに共通する特徴を考察

調査方法

(1) 最新100のcloseされた提案を調査

(2) 各提案のWebページを見て有益かどうかを人手で判断

＜有益な提案の特徴＞

特徴1：提案が示すアイデアやソースコードがプロジェクトに取り込まれている

特徴2：提案に対して活発に議論が行われた

調査対象プロジェクト

fork数が多いプロジェクトのうち上位3つを選定
（fork数は2013年9月16日時点）

プロジェクト名	概要	提案データ取得日付	fork数
bootstrap	Twitter社が開発するフレームワーク	2013年7月30日	19,829
rails	Rubyで開発されたフレームワーク	2013年8月19日	6,498
html5-boilerplate	HTML5のためのフレームワーク	2013年9月5日	5,398

調査結果

プロジェクト名	有益提案数/調査提案数	特徴1のみ満たす	特徴2のみ満たす	どちらも満たす
bootstrap	31/100	29	0	2
rails	47/100	41	1	5
html5-boilerplate	27/100	17	7	3

特徴1：提案が示すアイデアやソースコードがプロジェクトに取り込まれている

特徴2：提案に対して活発に議論が行われた

有益提案が占める割合

プロジェクト名	有益提案数/調査提案数	特徴1のみ満たす	特徴2のみ満たす	どちらも満たす
bootstrap	31/100	29	0	2
rails	47/100	41	1	5
html5-boilerplate	27/100	17	7	3

有益提案が占める割合は50%未満

(1) 無益な提案が多い傾向にある

特徴2のみ満たす提案

特徴2のみ満たす有益提案が少ない

プロジェクト名	有益	のみ	特徴2のみ満たす	どちらも満たす
bootstrap	31/100	29	0	2
rails	47/100	41	1	5
html5-boilerplate	27/100	17	7	3

特徴2のみ満たす有益提案が少なくない(7/27)

(1) 無益な提案が多い傾向にある

(2) どちらの特徴も有益な提案の抽出に有用

有益提案の自動抽出

プロジェクト内の有益提案数と無益提案数を把握しておくことは重要

有益提案数と無益提案数を継続的に把握

➡ プロジェクトの健全性の指標として活用

プロジェクト内の有益提案数の把握には時間と手間がかかる

例）調査では300提案に対して2人日要した

有益提案を自動的に抽出する方法を検討

GitHubで管理されているデータを活用できないか

有益提案の特徴とGitHubの関係

人間の感覚による判断を含む有益提案の特徴

特徴1：提案が示すアイデアやソースコードがプロジェクトに取り込まれている

特徴2：提案に対して活発に議論が行われた

GitHubから機械的に取得可能かつ判断可能な条件に置換え

プロジェクトがもつissue, pull request, forkなどの発生履歴

issue: タイトル，コメント数，参照されたコミット，作成日時…

有益提案の特徴と条件の対応

特徴1：提案が示すアイデアやソースコードがプロジェクトに取り込まれている

条件A pull requestがmergeされている

条件B 提案がcommitから参照されている

条件C 提案がcommit messageによりcloseされている

特徴2：提案に対して活発に議論が行われた

条件D 提案がclose後reopenされている

有益提案の条件の妥当性

判定有益提案 = 条件A～Dのうち少なくともひとつを満たす提案

有益な条件の特徴	判定有益提案数/有益提案数（正解データ）
条件A pull requestがmergeされている	54/53 (1件差)
条件B commitから参照されている	48/46 (2件差)
条件C commit messageによりcloseされている	21/21 (0件差)
条件D close後reopenされている	7/4 (3件差)
重複をのぞいた総有益提案数の割合	105/98

有益提案の条件を使ってほぼ全ての有益提案が抽出可能

➡ 条件A～Dは特徴1, 2を表現できている

調査結果と判定結果

		調査結果(人手)	
		有益	無益
判定結果(自動)	有益	93	5(偽陽性)
	無益	12(偽陰性)	190

＜正しく判定できた提案＞

判定結果で有益かつ調査結果で有益:93
判定結果で無益かつ調査結果で無益:190
} 283(94.3%)

＜偽陽性をもつ提案＞

判定結果で有益かつ調査結果で無益:5

＜偽陰性をもつ提案＞

判定結果で無益かつ調査結果で有益:12

不一致の原因

不一致の原因	詳細	提案数	総提案数
ユーザのミス	(1)必要のないreopen	2	5
	(2)提案の二重投稿	2	
	(3)プロジェクトオーナーの判断ミス	1	
条件の設定漏れ	(4)mergeされたpull requestからの参照	1	11
	(5)コメントからcommitへの参照	3	
	(6)多数のコメントと参加者	7	
その他	(7)GitHubのバグ	1	1

ユーザのミスが原因：偽陽性をもつ提案

条件の設定漏れが原因：偽陰性をもつ提案

特徴をより正確に表現できる条件なら有益な提案を抽出可能

有益提案数の推移

提案数と判定有益提案の推移に対する推測

(1) プロジェクト開始時は有益な提案が多い

(2) 新バージョンリリース時は無益な提案が増加

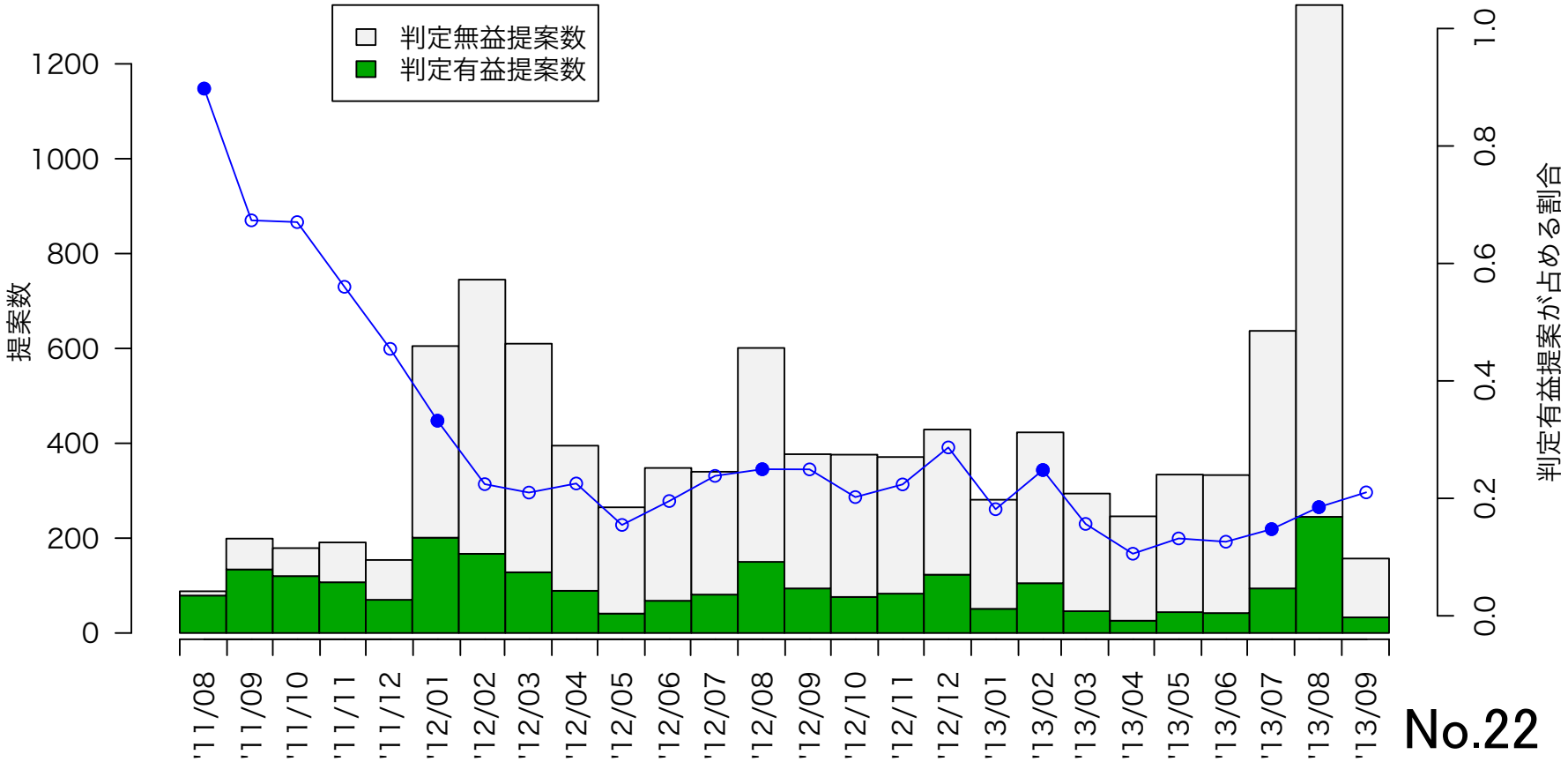
bootstrapの提案数と判定有益提案の推移で検証

bootstrapの有益提案数の推移

期間：GitHubでのプロジェクト開始時～調査時(約2年)

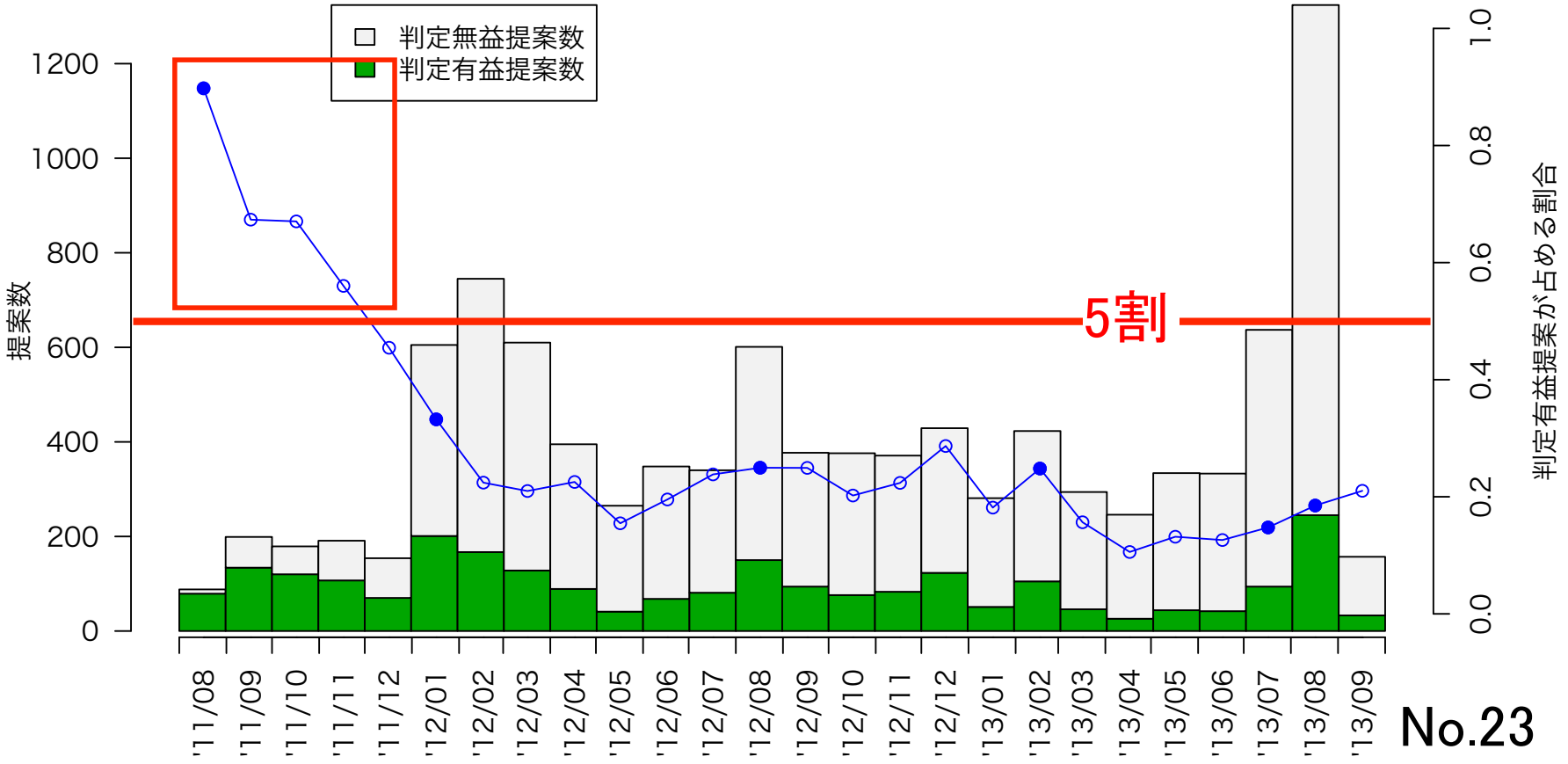
総提案数：10,303

判定有益提案数が占める割合：24%



プロジェクト開始時の有益提案数

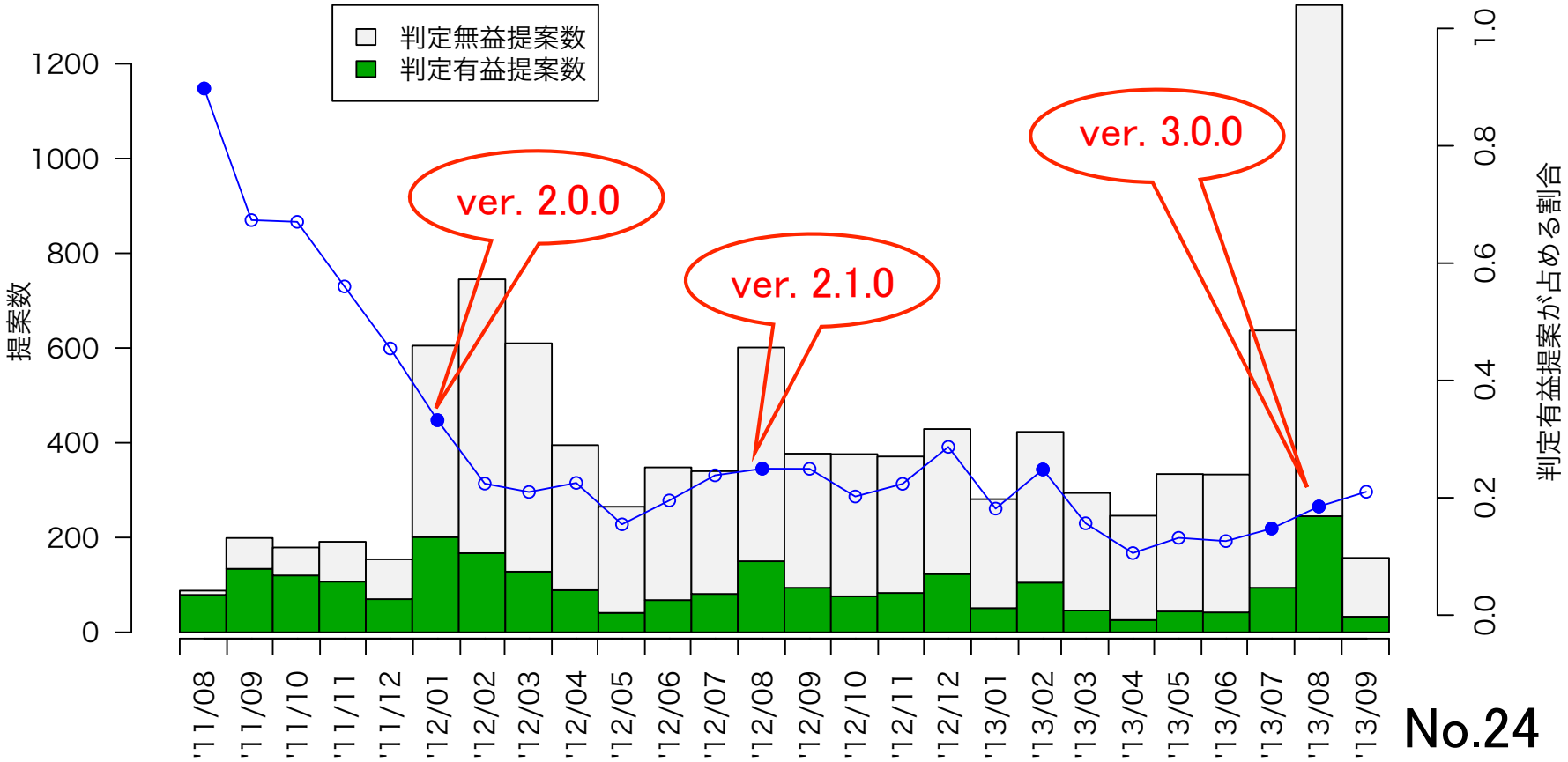
(1) プロジェクト開始後4ヶ月(2011年8月～2011年11月)
判定有益提案が占める割合が5割超
プロジェクト開始時は有益な提案が多い



新バージョンリリース時の無益提案数

(2) 2012年1月，2012年8月，2013年8月などに無益提案数が急激に増加

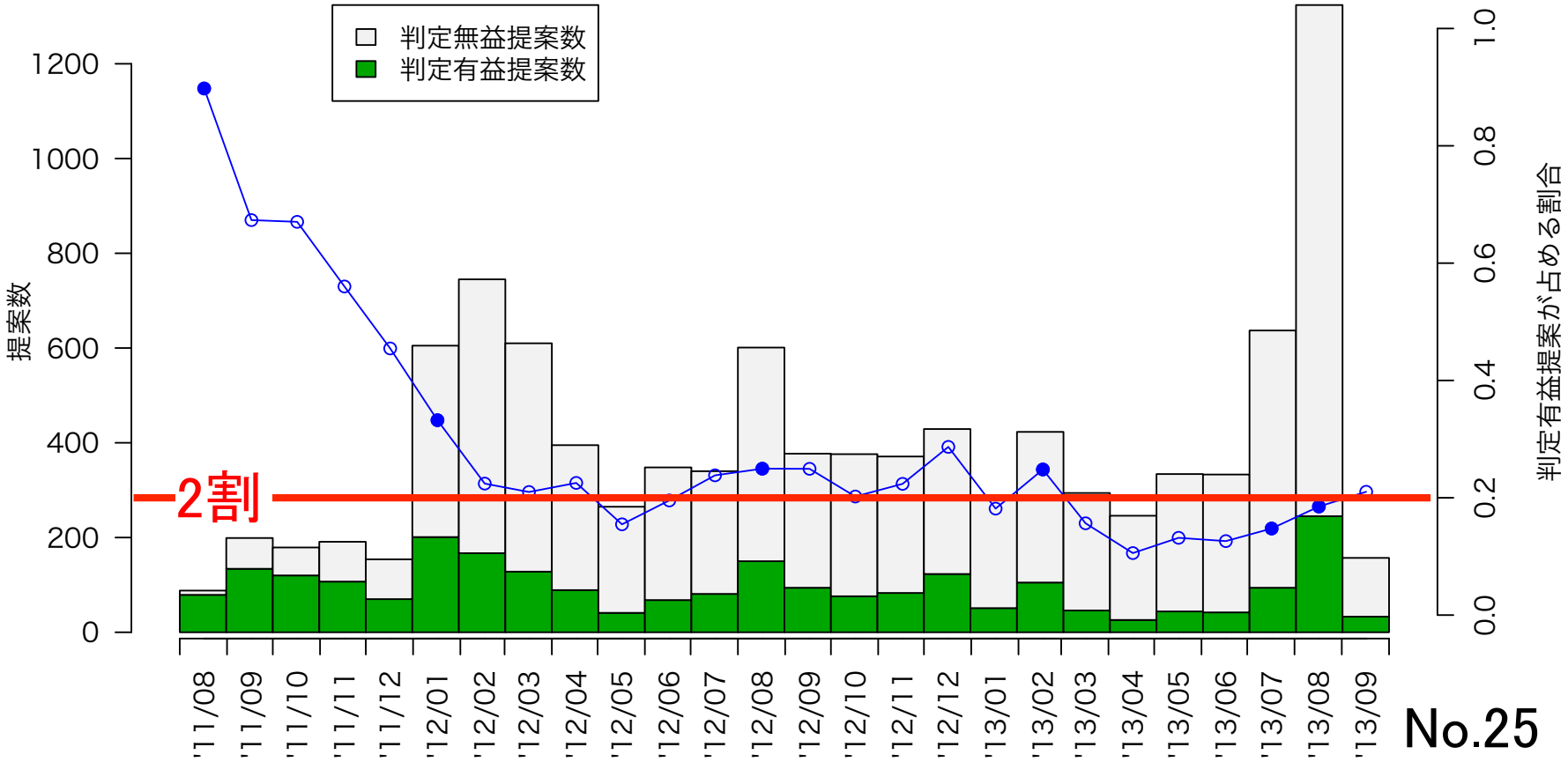
新バージョンリリース時は無益な提案が増加



有益提案の占める割合の推移

(3) 2012年1月以降，有益提案は2割前後で安定

提案数の変動によらず，有益提案の割合はほぼ一定



おわりに

(1) GitHubのプロジェクトに対して有益な提案数を調査

(2) 有益提案は50%未満

(3) 調査に使用した有益提案の特徴をGitHubのデータで表現

(4) bootstrapに対して提案数と有益提案数の推移を検証

(A) 有益提案は平均24%

(B) 開発初期をのぞいて，有益提案はプロジェクト全体を通してほぼ一定

＜今後の課題＞

(1) 有益な提案の更なる条件の発見

(2) 提案作成時に有益かどうか予測可能なシステムの作成

予備等

リリース日	バージョン名
2011年8月18日	version 1.0.0
2012年1月31日	version 2.0.0
2012年8月20日	version 2.1.0
2013年2月7日	version 2.3.0
2013年7月27日	version 3.0.0 リリース候補版
2013年8月19日	version 3.0.0

調査方法

＜調査方法＞

(1) 最新100のcloseされた提案を調査

(2) ブラウザ上のUIを見て各提案について有益かどうかを判断

＜有益な提案の特徴＞

特徴1 提案が示すアイデアやソースコードがプロジェクトに取り込まれている

(1) ユーザが提供したソースコードがプロジェクトに取り込まれる

(2) プロジェクトオーナーが作成したソースコードを直接プロジェクトに追加する

特徴2 提案に対して活発に議論が行われた

(1) コメント数と参加者数が多く，かつコメントが特定のユーザのものに集中していない

(2) closeした後にコメント数が増加

調査結果

プロジェクト名	有益提案数/調査提案数	特徴1のみ満たす	特徴2のみ満たす	どちらも満たす
bootstrap	31/100	29	0	2
rails	47/100	41	1	5
html5-boilerplate	27/100	17	7	3

(1) 有益提案が占める割合は50%未満

⇒ 無益な提案が多い傾向がある

(2) bootstrap, railsにおいて特徴1のみ満たす有益提案が多い

(2) html5-boilerplateにおいて特徴2のみ満たす有益提案が少なくない(7/27)

⇒ どちらの特徴も有益な提案の抽出に有用

調査結果

プロジェクト名	有益提案数/調査提案数	特徴1のみ満たす	特徴2のみ満たす	どちらも満たす
bootstrap	31/100	29	0	2
rails	47/100	41	1	5
html5-boilerplate	27/100	17	7	3

(1) 有益提案が占める割合は50%未満

⇒ 無益な提案が多い傾向がある

(2) bootstrap, railsにおいて特徴1のみ満たす有益提案が多い

(2) html5-boilerplateにおいて特徴2のみ満たす有益提案が少なくない(7/27)

⇒ どちらの特徴も有益な提案の抽出に有用

条件の設定漏れ

不一致の原因	詳細	提案数	総提案数
ユーザのミス	(1)必要のないreopen	2	5
	(2)提案の二重投稿	2	
	(3)プロジェクトオーナーの判断ミス	1	
条件の設定漏れ	(4)mergeされたpull requestからの参照	1	11
	(5)コメントからcommitへの参照	3	
	(6)多数のコメントと参加者	7	
その他	(7)GitHubのバグ	1	1

(4) pull requestから参照されたデータを保持していない

(5) コメントからの参照は文字列として扱われる

(6) 提案が持つデータの有益かどうかの判定基準が不明
2.3%を占める：解決すれば正しい判定結果が96.7%(+2.4%)

有益提案の条件

＜自動抽出＞

有益提案の特徴をGitHubから取得可能なデータに置き換える

(条件A) pull requestがmergeされている

merge：pull requestによる提案が取り込まれる

(条件B) 提案がcommitから参照されている

プロジェクトオーナーが開発を進めた

(条件C) 提案がcommit messageによりcloseされている

issueに対して有益なソースコードが提供された

(条件D) 提案がclose後reopenされている

close後にさらに議論が必要だと判断された

これらの条件を満たす提案を有益提案と判定

ソーシャルコーディングの問題点

問題を持った無益な提案が作成される

＜無益な提案＞

(1) 過去の提案と内容が重複した提案

(2) 提案者のミスが原因の不具合の報告

(3) プロジェクトで扱うべきでない問題や機能の報告

＜無益な提案が作成される原因＞

(原因1) プロジェクトへの理解度の低い開発者が存在

(原因2) プロジェクトの参加者が時間を経て変化

提案が有益かどうか判断する作業に時間がかかる

➡ 有益な提案の自動抽出，提示は有用

有益な提案とは？