OSS開発のプロジェクト事情による不具合修正時間の分析

若元 亮樹[1]　伊原 彰紀[1]　松本 健一[1]

概要：本論文は，オープンソースソフトウェア (OSS) プロジェクトにおける保守作業の理解に向けて，OSS プロジェクトへ不具合が報告されたときのプロジェクト事情（プロダクト進化，開発体制の変動）によってプロジェクトが不具合修正に向けた取り組みを開始するまでの時間（不具合対応時間）の違いを調査する．OpenStack プロジェクトを対象に開発履歴を分析した結果，不具合報告時に活動している開発者数（特に，コア開発者数）が，不具合対応時間に影響していることがわかった．

Toward Understanding Bug-fixing Time in Open Source Software Project Conditions

RYOKI WAKAMOTO[1]　AKINORI IHARA[1]　KEN-ICHI MATSUMOTO[1]

1. はじめに

昨今，オープンソースソフトウェア (OSS) がミッションクリティカルなシステムに導入される事例が増え，ソフトウェア開発企業（以下，企業と記述する）にとって OSS の利用価値が高くなっている．独立法人情報処理推進機構（IPA）の調査によると，企業が抱える事業の約 4 割では OSS を利活用している [1]．OSS が多くの企業で利活用されるようになった理由として，導入コスト削減はもちろん，品質の高さ，数多くリリースされている OSS から企業が選択可能であることなどが挙げられ，今後も企業による OSS の導入は加速しつづけると考えられる．

OSS の利活用が進む一方で，OSS プロジェクトによる保守作業に不安を抱える企業が多い [1]．なぜなら少人数で開発されている OSS も多く，長期間更新されていない OSS も存在していることが一つの原因である [2]．昨今，OSS の課題（例えば，OpenSSL の Heartbleed や，Apache Struts1 の脆弱性問題など）が，実社会の経済活動を揺るがすほどの影響を与える事例も報告されており，OSS を利活用する企業は，OSS に発見される欠陥への保守作業を評価することが喫緊の課題である．

しかし，企業の開発者が OSS 開発計画，保守計画を理解することは容易ではない．なぜなら，OSS プロジェクトでは，開発者がプロジェクトから自由に参加/離脱を行い，日々拡張を繰り返し，変更されるプログラムを長期間追跡している開発者は少ないため [3]，欠陥を埋め込んだ開発者が迅速に不具合を修正できないことも多い [4][5]．従って，OSS を利活用する企業は，対応に時間がかかる不具合を自身で修正せざるを得ないこともある [1]．

従来研究では，OSS プロジェクトによる保守作業を理解するために，OSS プロジェクトが不具合を修正するために要する時間，OSS プロジェクトが修正に取り組むか否か，を予測するためのモデル開発に関する研究が多数行なわれている [7][16][6][5]．これまでに開発されたモデルでは，修正担当者の専門性，欠陥の内容など，個別の不具合に関する情報に基づいてモデル開発が行なわれている．しかし，コミュニティで活動する開発メンバ，欠陥が混入しているモジュールはもちろん，プロジェクトが限られた開発工数を費やすモジュールも日々変化するため，リポジトリの開発履歴から得られる個々の不具合の特徴だけでなく，現在のプロジェクトの状況を考慮したモデルを開発することが必要と考える．

本論文では，不具合報告された時期におけるプロジェクト事情（プロダクト進化，開発体制の変動）の違いで，プロジェクトが報告された不具合に対応するまでの時間（不具合対応時間）の違いを調査する．企業は，OSS プロジェクトの判断によって，自ら修正するか否かを判断するため，

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

本論文では，OSS プロジェクトが修正する不具合，及び，修正しないと決定する不具合を対象とする．そして，OSS プロジェクトが修正完了するまでの時間，及び，修正しないと決定するまでの時間を不具合対応時間として計測する．

本論文は，OSS のプロジェクト事情による対応時間の違いの調査によって，OSS 利活用者が早く/遅く修正される不具合の把握し，OSS プロジェクトの不具合修正行動の理解を支援する．

本論文では，プロジェクトの状況に関してプロダクトの変化，開発者の行動について 4 つの仮説を立て，OpenStack プロジェクトの開発履歴を用いて検証する．

[プロダクト進化]

仮説 1: 不具合報告が多い時期に報告された不具合は，対応までにかかる時間が長い．

仮説 2: プロダクトの変更量が多い時期に報告された不具合は，対応までにかかる時間が長い．

[開発体制の変動]

仮説 3: コード変更を行う開発者が多い時期に報告された不具合は，対応までにかかる時間が短い．

仮説 4: 開発者間のコミュニケーションが多い時期に報告された不具合は，対応までにかかる時間が短い．

本論文では，それぞれの仮説に基づき，変化するプロジェクト事情が不具合対応時間に影響しているか否かを明らかにするが，複数の仮説間に関係が存在することも考えられる．例えば，開発要員が多い場合，プロダクトを検証する人数が多いため，欠陥が多く発見されることも考えられる．本論文では，それぞれの仮説間の関係についても調査した上で，修正時間に影響する要因を考察する．

続く 2 章では，一般的な OSS 開発における不具合修正プロセスについて述べ，3 章では，OSS 開発におけるプロジェクトの事情に基づく，不具合の対応について仮説を述べる．4 章では，3 章で述べた仮説の分析方法について述べ，5 章では，OpenStack を対象に分析を行った結果を述べる．6 章では，結果についての考察を行い，最後に 7 章で本論文のまとめと今後の課題を述べる．

2. OSS プロジェクトにおける不具合への対応

2.1 不具合管理

大規模な OSS 開発では，日々数十件から数百件の不具合が報告されている [8]．世界中に点在する開発者同士が膨大に報告される不具合の情報を共有するために，Bugzilla [*1]，Launchpad [*2] などの不具合管理システム (BTS) が利用されており，開発者は，報告された個々の不具合情報，また，その修正過程を BTS に記録する．具体的には不具合の基本情報（対象プロダクトやバージョン，修正の重要度や優先度），詳細情報（不具合の内容や不具合を再現するための手順），議論情報（不具合の内容や修正に関して行われる議論），状態遷移管理情報 (不具合修正処理の状態管理を行うために記録される情報) の，4 種類の情報が記録され，表 1 に各情報の内容を示す．この記録は，Web ユーザインタフェースを介して誰でも不具合の報告と修正作業進捗の閲覧が行えるため，他の開発者，および，利用者との情報共有，共同作業を実現する．

[*1] Bugzilla: https://www.bugzilla.org/

[*2] Launchpad: https://launchpad.net/

表 1 不具合管理システムに記録される情報

区分	内容例
(1) 不具合の基本情報	対象プロダクト 修正の優先度，重要度 修正状況
(2) 不具合の詳細情報	不具合の内容 不具合を再現する方法 欠陥のある箇所
(3) 議論情報	開発者による発言 発言日時
(4) 状態遷移管理情報	不具合修正記録 CC リスト管理記録

2.2 不具合のトリアージ

OSS プロジェクトに報告された不具合は全て修正されるとは限らない．OSS プロジェクトに報告される不具合の中には，現時点では修正されないが，将来のリリースで修正される場合もあり，OSS プロジェクト管理者は，報告された不具合に対して，修正を優先的に取り組む不具合を決定し，選別する作業を行う．OSS を商用ソフトウェアの開発に利活用する企業は，OSS に混入している不具合が OSS プロジェクトで修正されるか否かを把握することできれば，企業が自ら不具合の修正を行うか否かを決定することができるが，不具合管理システムに登録されている不具合の中には，OSS プロジェクトで修正されるか否か判断できない不具合も数多く存在する．

2.3 不具合の対応時間

従来研究では，不具合の特徴，担当者，そして，対応の種類によって，不具合の修正にかかる時間，対応されるまでにかかる時間について調査されている．

[不具合の特徴] 開発者は，不具合票の内容を確認し，過去に担当した経験のある機能に関する不具合，または，優先度，重要度の高い不具合から順に修正する傾向にある [9]．

[担当者] OSS 開発では自主的に修正に取り掛かる場合も多いが，担当者がプロジェクトから既に離脱している，もしくは，担当者が抱えているタスク量から修正に取り掛かることができないこともあり [12]，OSS プロジェクト管理者が，他の開発者へ修正を依頼することも多い [4]．

[対応の種類] 修正される不具合は，修正されない不具合よ

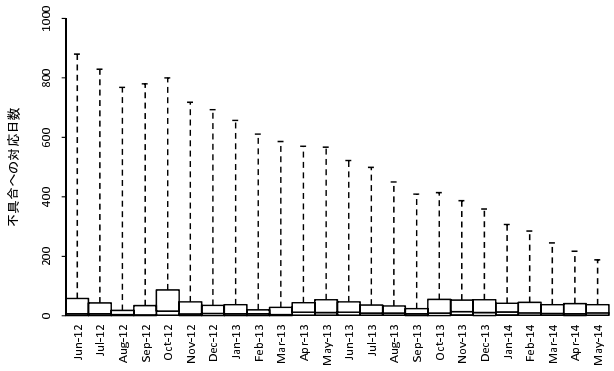

図 1 OpenStack プロジェクトの不具合対応時間

りも，修正までに時間がかかっている [13]．その理由は，修正される不具合の中には，簡単に修正されるものが数多く含まれる一方で，修正されない不具合は，修正しない/できない理由を調査するために時間がかかることもある．

従来研究では，不具合の特徴，不具合が混入する機能の担当者など，個別の不具合の特徴によって修正にかかる時間が変動することが示されている [7][10][11][13]．しかし，不具合が対応されるまでにかかる時間は，個別の不具合の特徴にのみ依存するわけではない．OSS 開発では，開発者の参加/離脱が激しく，また刻一刻と変化するプロダクトの進化によるタスク量の増減など，OSS，および，プロジェクト事情により不具合対応時間が変動すると考える．本論文では，OpenStack プロジェクトにおいて，開発者が修正完了，または，修正しないことを決定した不具合報告を対象に，報告時期による不具合対応時間の違いを調査した．対象とするデータは，2012 年 6 月から 2014 年 5 月までの間に報告された不具合 28,410 件である．図 1 に，各月に報告された不具合への対応時間を示す．実験対象期間内で，OpenStack プロジェクトで対応された不具合が，報告されてから対応されるまでの時間の中央値は 3 日-16 日であり，時期によって 2 週間以上の違いがある．従って，プロダクトの進化，プロジェクトの状態などによっても不具合の対応時間が変わってくることが示唆される．

本論文では，不具合報告時におけるプロジェクト事情による不具合対応時間の違いを明らかにすることにより，OSS を利活用している企業にとって，不具合の対応時間はもちろん，OSS プロジェクトにおける保守作業，保守計画の理解を支援できると期待する．

3. プロジェクト事情メトリクスと仮説

本章では，プロジェクト事情として計測するメトリクスと，プロジェクト事情によって不具合の対応時間に影響する仮説を説明する．

前章の図 1 にも示すように，不具合が報告される時期によって対応までにかかる時間が変動していることがわかった．本論文では，プロジェクト事情を把握するためのメトリクスとして，日々変更されるプロダクトの進化（タスク数，プロダクト変更量），および，OSS プロジェクトにおける開発体制の変動（開発者数，コミュニケーション数）を計測する．

[プロダクトの進化] OSS は，明確な納期がないため，リリースに関係なく継続的に変更が行われ，開発者，利用者からの要求が商用ソフトウェアに比べて早くプロダクトに反映される．Mozilla Firefox では，6 週間に一度のリリースを行い，高速な開発サイクルを実現しており，その他のプロジェクトでも数ヶ月から約 1 年の周期で新しいバージョンがリリースされ，プロダクトは刻一刻と更新されている．しかし，プロダクトの変更量や要求（不具合修正など）が多い場合，開発者は全ての要求へ対応することが難しく [5]，要求への対応に要する時間は長くなると考え，以下の 2 つの仮説を検証する．

仮説 1: 不具合報告が多い時期に報告された不具合は，対応までにかかる時間が長い．

仮説 2: プロダクトの変更量が多い時期に報告された不具合は，対応までにかかる時間が長い．

[開発体制の変動] OSS 開発では，開発者がプロジェクトへの参加/離脱が自由であるため，プロジェクトの開発体制は日々変化している．OSS 開発者の多くは，自身が担当する機能以外の修正を行わないため [14]，離脱した開発者が担当する機能の変更が遅れる可能性がある．特に，プロジェクトを管理する開発者がプロジェクトから離脱する場合，開発するソフトウェアが長期間更新されない場合もあり，開発者の流動性は開発プロセスの遅延に影響すると考え，以下の 2 つの仮説を検証する．

仮説 3: コード変更を行う開発者が多い時期に報告された不具合は，対応までにかかる時間が短い．

仮説 4: 開発者間のコミュニケーションが多い時期に報告された不具合は，対応までにかかる時間が短い．

4. 分析方法

本章では，分析を行うために用いたデータセットと，前章で述べた仮説を検証するための手法について述べる．

4.1 分析対象データセット

仮説を検証するために，本論文では，大規模 OSS プロジェクトである OpenStack プロジェクトを用いる．OpenStack は，OSS として提供されているクラウドサービスを実現するソフトウェアである．OpenStack プロジェクトは，多くの開発者が参加し，近年，活発的に開発が進められているプロジェクトの一つである．

OpenStack プロジェクトでは Launchpad を不具合管理システムとして利用している．本論文では，不具合管理システムを利用開始した 2012 年 6 月から 2014 年 5 月までの期間に報告された不具合を対象とする．2 章で説明したよ

表 2　データセットの概要

対応された不具合件数	19,852
報告された不具合件数	30,100
ソースコードの変更行数	12,924,505
参加している開発者の数	2,560
メーリングリストへの投稿件数	40,092

うに，不具合管理システムに記録されている不具合の中には，修正が完了した不具合，修正しないことが決定している不具合，修正途中の不具合を見つけることができる．本論文では，不具合票に *Fix Committed* と記録されている報告を修正が完了した不具合，*Won't Fix*（修正する必要が無い），または，*Invalid*（仕様である）と記録されている報告を修正しないことが決定している不具合とする．また，不具合の対応時間は，報告されてから修正が完了する，または，修正完了しないことが決定した日にちまでとする．

プロジェクト事情として，OSS のプロダクトの進化（不具合報告数，プロダクト変更量），および，OSS プロジェクトにおける開発体制の変動（開発者数，コミュニケーション数）を計測するために，本論文ではプロジェクトで利用されている不具合管理システム，メーリングリスト，バージョン管理システムの記録を用いる．具体的なメトリクスの計測方法は，次節の仮説の分析方法において述べる．

4.2　仮説の分析方法

分析対象期間中の各月のプロジェクト事情を示すメトリクス（各月の不具合報告数（仮説 1），プロダクトの変更量（仮説 2），コード変更を行う開発者数（仮説 3），または，コミュニケーション数（仮説 4））の時系列変化と，各月に報告された不具合が開発者に対応されるまでにかかる時間の中央値の変化を分析する．また，プロジェクト事情のメトリクス値による対応時間の分布を比較する．具体的には，仮説 1 の場合，不具合報告数が多い時期と少ない時期で 2 分割し，報告数が多い時期と少ない時期の対応時間の分布を比較し，プロジェクト事情が対応時間に与える影響を明らかにする．分布の比較にはウィルコクソンの順位和検定を用いる．

5.　分析結果

本章では，OpenStack プロジェクトにおける，プロジェクト事情による不具合対応時間を分析した結果を述べる．

[仮説 1] 不具合報告が多い時期に報告された不具合は，対応までにかかる時間が長い．

図 2 は，分析対象期間の各月で報告された不具合の件数を棒グラフ（左軸），不具合対応時間の中央値を折れ線グラフ（右軸）を示す．棒グラフは，対応された不具合（白色）と対応しないと決定された不具合（灰色）の積み重ねグラフで示している．分析対象期間の後半になるにつれて，1 か月あたりに報告される不具合数は増加している．図 6 で不具合報告件数の多い月，少ない月で，報告さた修正時間の分布を比較したが，統計的有意差（有意水準 5%） が見られなかった．従って，不具合報告件数と不具合対応時間との関係はなく，仮説 1 は支持されない．

[仮説 2] プロダクトの変更量が多い時期に報告された不具合は，対応までにかかる時間が長い．

図 3 は，分析対象期間の各月で変更されたソースコード追加行数（白色），削除行数（灰色）を棒グラフ（左軸），不具合対応時間の中央値を折れ線グラフ（右軸）を示す．2013 年 9 月頃に大規模な変更が行われているが，同時期の不具合対応時間は長期化してはいない．図 7 は，追加行数，削除行数の多い月，少ない月で，報告さた不具合対応時間の分布を比較したが，両方とも統計的有意差（有意水準 5%） が見られなかった．従って，変更量（追加行数，削除行数）と不具合対応時間との関係はなく，仮説 2 は支持されない．

[仮説 3] コード変更を行う開発者が多い時期に報告された不具合は，対応までにかかる時間が短い．

図 4 は，分析対象期間の各月にプロダクトの変更を行った開発者数を棒グラフ（左軸），不具合対応時間の中央値を折れ線グラフ（右軸）で示す．分析対象期間の後半になるにつれプロダクトの変更を行う開発者が増加している．図 8 は，プロダクトの変更を行った開発者の多い月，少ない月で，それぞれ不具合対応時間の分布を比較を行い，開発者が多い時期ほど修正時間が長く統計的有意差（有意水準 5%） が見られた．従って，仮説 3 は支持されず，仮説とは反対の結果となった．

[仮説 4] 開発者間のコミュニケーションが多い時期に報告された不具合は，対応までにかかる時間が短い．

図 5 は，分析対象期間の各月に開発者がメーリングリストにメッセージを投稿した数を棒グラフ（左軸），不具合対応時間の中央値を折れ線グラフ（右軸）で示す．分析対象期間の後半になるにつれメッセージの投稿数は増加している．図 9 は，メッセージの投稿数が多い月，少ない月で，それぞれ不具合対応時間の分布を比較を行い，開発者間のコミュニケ＝ションが多い時期ほど修正時間が長く，統計的有意差（有意水準 10%） が見られた．従って，仮説 4 は支持されず，仮説とは反対の結果となった．

6.　考察

本章では，前章において OSS のプロジェクト事情が，不具合の対応時間の違いの結果について述べる．

6.1　コア開発者による不具合対応

仮説 3 より，プロジェクトで修正活動を行っているユニーク開発者数が多いほど，同時期に報告された不具合に対しての対応が遅れているという結果が得られた．OSS プロジェクトでは，開発者の参加/離脱が繰り返される一方，

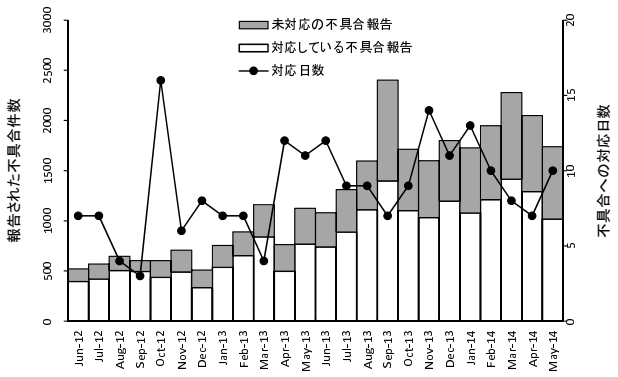

図 2　不具合報告件数と対応時間の関係

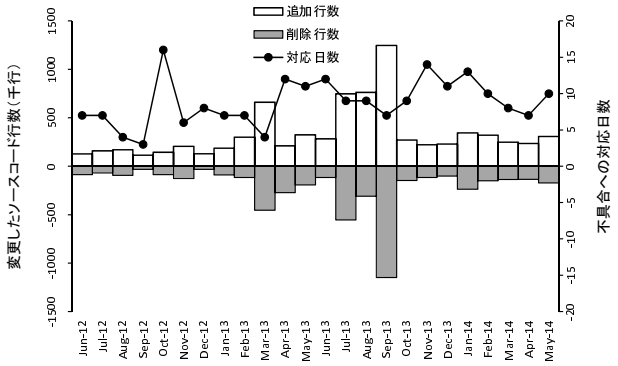

図 3　コード変更行数と対応時間の関係

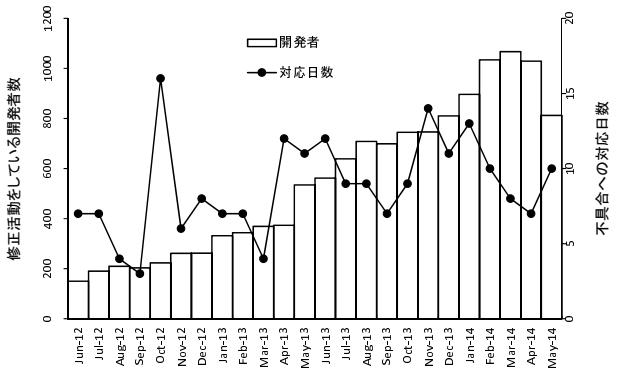

図 4　活動している開発者数と対応時間の関係

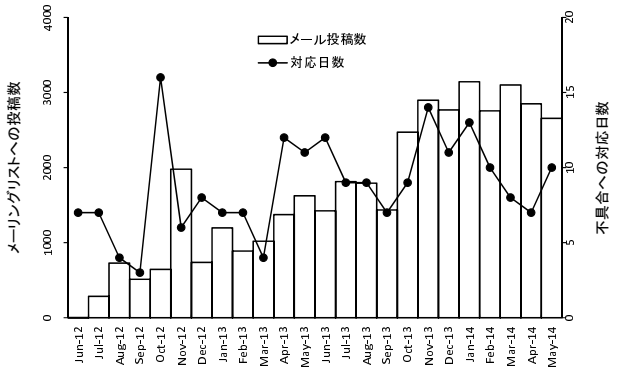

図 5　メーリングリストへの投稿数と対応時間の関係

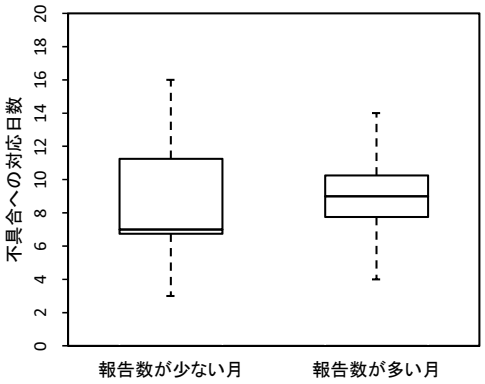

図 6　不具合報告件数と対応日数の比較

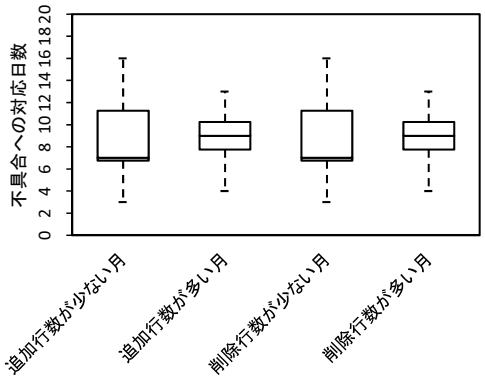

図 7　コード変更行数と対応日数の比較

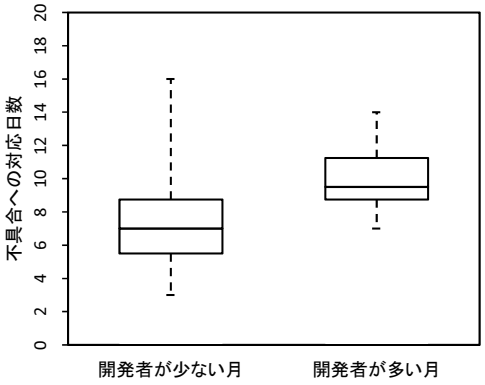

図 8　活動している開発者と対応日数の比較

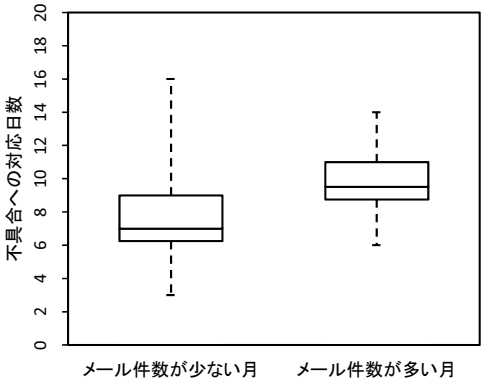

図 9　議論数と対応日数の比較

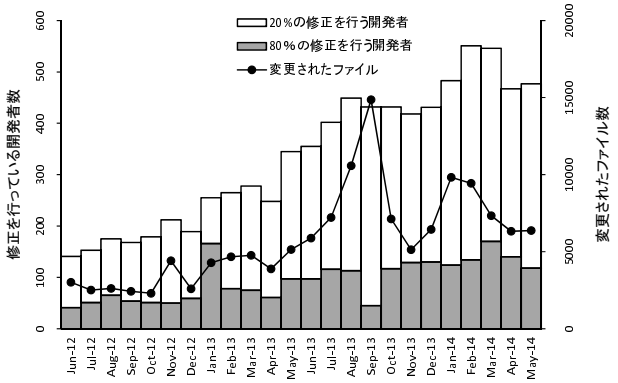

図 10 修正を行う開発者数と修正ファイル数の関係

長期間プロジェクトに貢献する開発者（コア開発者）も存在する．OSS 開発では，プロジェクトに貢献する開発者の中で積極的に貢献する 20%の開発者が，80%のプロダクトの更新を行っており [8]，このようなコア開発者が OSS 開発を支えている．不具合の対応時間に影響しているのは開発者の中の，特にコア開発者の貢献に依存していると考える．

本章では，追加分析として，分析対象期間の各月で積極的に更新を行う開発者のうち，その月に変更されたファイルの 80%の変更に貢献した人数を調査した．図 10 は，分析対象期間の各月でプロダクトの変更に貢献した開発者を棒グラフ（左軸），変更されたファイル数を折れ線グラフ（右軸）を示す．棒グラフは，80%のファイル変更に積極的に貢献した開発者数（灰色）と 20%のファイル変更に貢献した開発者数（白色）の積み重ねグラフで示している．分析対象期間の前半から後半にかけて，プロダクト更新に貢献した開発者のユニーク数が増える一方，80%のプロダクトを更新するコア開発者の人数の変動は少ない．2013 年 9 月におけるコア開発者は，同じ月に貢献した開発者全体の約 10%（43 人）であった一方で，ユニーク開発者数が最も多かった 2014 年 2 月では，コア開発者数は，同じ月に貢献した開発者全体の約 24%（134 人）である．しかしながら，変更されるファイル数は分析対象期間の後半に向けて増加し，また，仮説 1 で示したように利用者や開発者からの要求も分析対象期間の後半に向けて増加している．従って，プロジェクトへの要求，更新ファイルが増加し，ユニーク開発者が増加する一方で，コア開発者の人数の変動が少ないことから，コア開発者一人あたりの負担が大きくなり，分析対象期間の後半にかけて不具合対応時間が長期化したと示唆される．

6.2 制約

本論文ではプロジェクト事情という観点から不具合の対応時間を分析したが，不具合の対応時間は従来研究で実施されてきた，個々の不具合の特徴にも依存することが示唆される．ただし，昨今の研究 [4] において，不具合の対応時間が，担当者の変更によって長期化すると述べられている．本論文の仮説 3 においても，開発者の人数（特にコア開発者の人数）が影響していることが示されたことから，今後は，モジュール担当者だけでなく，プロジェクトの開発体制，コア開発者の貢献についても調査をしていくことが必要と考える．

また，本論文では，不具合が報告された時期のプロジェクト事情を調査し，プロジェクト事情と当該不具合の対応時間を調査した．しかし，プロジェクト事情の変動が，同じ月に影響するとは限らず，次の月や近い将来に報告された不具合の対応時間に影響することも考えられる．本課題は，山谷ら [17] が提案する遅延相関分析を用いることで分析可能であると示唆される．

本論文では，大規模な OSS プロジェクトである OpenStack プロジェクトの開発データを用いて分析を行った．今回得た知見の有意性を確認するために，今後複数の OSS プロジェクトのデータを用いて分析する必要がある．

本論文では，分析対象として 2014 年 5 月までに報告された不具合を対象とした．2014 年 5 月に報告された不具合には未だ対応されていない不具合が存在している．本論文では，多くの不具合は 1 ヶ月未満で対応されるため，本研究では，2014 年 6 月末までに対応されたか否かを判定していることから，結果の妥当性は高いと考える．

7. おわりに

本論文では，日々変化するプロジェクトの状況を理解しつつ，OSS プロジェクトにおける不具合の修正計画を見積もりを実現するために，プロダクトの進化（不具合報告件数，プロダクト変更量），開発体制（開発者数，コミュニケーション数）が不具合対応時間及ぼす影響について分析を行った．大規模 OSS である OpenStack プロジェクトを対象に分析を行った結果，特に，開発者の行動の変化は，不具合の対応速度に影響を与えていることがわかった．今後は，従来研究で取り組まれてきた個々の不具合の特徴に加え，本論文が対象とする日々変動するプロジェクトの開発体制と，個々の開発者の行動変化から OSS プロジェクトにおける不具合修正の可否，及び，不具合修正時間の予測モデルの開発に取り組む．

謝辞

本研究の一部は，頭脳循環を加速する戦略的国際研究ネットワーク推進プログラムによる助成を受けた．

参考文献

[1] 独立行政法人 情報処理推進機構 “第 3 回オープンソースソフトウエア活用ビジネス実態調査 (2009 年度調査)”, 2010 年.

[2] Masao Ohira, Naoki Ohsugi, Tetsuya Ohoka, and Kenichi Matsumoto, “Accelerating cross-project knowledge

collaboration using collaborative filtering and social networks," Proceedings of the International Workshop on Mining Software Repositories (MSR'05), pages.1-5, 2005.

[3] Christian Bird, Alex Gourley, Prem Devanbu, Anand Swaminathan, and Greta Hsu, "Open Borders? Immigration in Open Source Projects," Proceedings of the 4th International Workshop on Mining Software Repositories (MSR'07), pages.6-13, 2007.

[4] Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann, "Improving bug triage with bug tossing graphs," Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (ESEC/FSE'09), pages.111-120, 2009.

[5] Akinori Ihara, Yasutaka Kamei, Akito Monden, Masao Ohira, Jacky Wai Keung, Naoyasu Ubayashi, and Kenichi Matsumoto, "An Investigation on Software Bug Fix Prediction for Open Source Software Projects -A Case Study on the Eclipse Project -," Proceedings of the International Workshop on Software Analysis, Testing and Applications (SATA 2012), pages.112-119, 2012.

[6] Daniel Alencar da Costa, Surafel Lemma Abebe, Shane McIntosh, Uirá Kulesza, Ahmed E. Hassan. "An Empirical Study of Delays in the Integration of Addressed Issues" Proceedings of the 30th IEEE International Conference on Software Maintenance and Evolution (ICSME'14), pages.281-290, 2014.

[7] 正木仁, 大平雅雄, 伊原彰紀, 松本健一, "OSS 開発における不具合割当パターンに着目した不具合修正時間の予測", 情報処理学会論文誌, volume 54, number 2, pages 933-944, 2013 年.

[8] Marcelo Serrano Zanetti, Ingo Scholtes, Claudio Juan Tessone, and Frank Schweitzer, "Categorizing Bugs with Social Networks: A Case Study on Four Open Source Software Communities," Proceedings of the 2013 International Conference on Software Engineering (ICSE'13), pages.1032-1041, 2013.

[9] Israel Herraiz, Daniel M. German, Jesus M. Gonzalez-Barahona, and Gregorio Robles, "Towards a Simplification of the Bug Report form in Eclipse," Proceedings of the 5th International Workshop on Mining Software Repositories (MSR'08), pages.145-148, 2008.

[10] Rattikorn Hewett and Phongphun Kijsanayothin, "On Modeling Software Defect Repair Time," pages165-186, volume 14, number 2, 2009.

[11] Pieter Hooimeijer and Westley Weimer, "Modeling Bug Report Quality," Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'07), pages.34-43, 2007.

[12] Masao Ohira, Ahmed E. Hassan, Naoya Osawa, Kenichi Matsumoto, "The Impact of Bug Management Patterns on Bug Fixing: A Case Study of Eclipse Projects," Proceedings of the International Conference on Software Maintenance (ICSM'12), pages.264-273, 2012.

[13] Cathrin Weiss, Rahul Premraj, Thomas Zimmermann, and Andreas Zeller, "How Long Will It Take to Fix This Bug?," Proceedings of the 4th International Workshop on Mining Software Repositories (MSR'07), pages.1-8, 2007.

[14] Hironori Hayashi, Akinori Ihara, Akito Monden, and Ken-ichi Matsumoto, "Why Is Collaboration Needed in Oss Projects?: a Case Study of Eclipse Project," Proceedings of the 5th International Workshop on Social Software Engineering (SSE'13), pages.17-20, 2013.

[15] Amiangshu Bosu, and Jeffrey C. Carver, "Impact of Developer Reputation on Code Review Outcomes in OSS Projects: An Empirical Investigation," Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'14), pages.33-42, 2014.

[16] Hongyu Zhang, Liang Gong, Steve Versteeg, "Predicting bug-fixing time: an empirical study of commercial software projects," Proceedings of the International Conference on Software Engineering (ICSE'13), pages.1042-1051, 2013.

[17] 山谷陽亮, 大平雅雄, Passakorn Phannachitta, 伊原彰紀, "OSS システムとコミュニティの共進化の理解を目的としたデータマイニング手法," 情報処理学会論文誌, volume 56, number 1, pages 59-71, 2015 年.