チュートリアル

オープンソースソフトウェア工学

伊原 彰紀，大平 雅雄

本チュートリアルでは，オープンソースソフトウェア (OSS) および OSS を活用したシステムを利用する組織・個人，および，OSS を利用してシステム開発をおこなうベンダ企業が，「安心して安全に」OSS を利用活用するための技術的方法論の確立を目指す学術分野「オープンソースソフトウェア工学」について，著者らがこれまでに取り組んできた研究事例を交えながら解説する．

Nowadays, Open Source Software (OSS) is often used by individuals/organizations, and many industrial systems with OSS are developed. In order to lead to use/develop safe and secure OSS, this paper overviews a research field called Open Source Software Engineering, through introducing existing academic studies.

1 はじめに

近年，オープンソースソフトウェア (OSS) の利活用を取り巻く環境が大きく変わりつつある．Android 端末の普及により，OSS の存在がエンドユーザにとっても身近な存在になっただけではなく，OSS 利用に対するベンダ企業の価値認識が，コスト削減から最新技術の導入（競争力の維持強化），あるいは，ロックインの回避といった観点に重点がシフトしているためである．例えば，最新のビッグデータ処理やクラウド構築のための基盤ソフトウェア（Hadoop や OpenStack など）は，多数のベンダ企業が OSS プロジェクトに参加し協力して開発がおこなわれている．

OSS がベンダ企業と連携する先進的な開発形態が登場し始めた一方で，従来の形態で開発を進めている OSS プロジェクトも数多く存在する．OSS プロジェクトの各種統計情報を提供している Open Hub[†1] には，2014 年 8 月時点で 66 万件以上の OSS プロジェクトが登録されているが，多くのプロジェクトは従来型の開発形態のものである．従来型の OSS 開発では，不特定多数のボランティア開発者が，ユーザからのフィードバックをインターネットメディアを通じて取り入れながら，柔軟に機能拡張をおこなったり不具合修正に対応している．

開発者にとっての従来型の開発形態のメリットの一つは，プロジェクトへの参加/離脱も含めて開発を自由意志でおこなえる点にある．自らが利用している OSS をより高品質・高機能にするためのプロジェクトに自主的に参加することは，他の共同開発者から高度な技術を学ぶ機会としても作用しており，多くの開発者が OSS プロジェクトに参加する動機付けにもなっている [9]．また，参加するプロジェクトの開発方針に不満があれば，既存 OSS をベースに新たにプロジェクトを立ち上げ，賛同する開発者らとともに好みの方向性で開発を継続することもできる．例えば Linux には，800 種類を超えるディストリビューションが存在すると言われている [†2]．

Open Source Software Engineering.

Akinori Ihara, 奈良先端科学技術大学院大学情報科学研究科, Graduate School of Information Science, Nara Institute of Science and Technology.

Masao Ohira, 和歌山大学システム工学部情報通信システム学科, Dept. of Computer and Communication Sciences, Faculty of Systems Engineering, Wakayama University.

コンピュータソフトウェア, Vol.33, No.1 (2016), pp.28–40.
[解説論文 (レター)] 2011 年 12 月 15 日受付.

†1 https://www.openhub.net/ (accessed 2014-08)

†2 http://distrowatch.com/search.php?status=All (accessed 2015-05-23)

多種多様な OSS が存在することは，多くの選択肢を与えるという点では，ユーザにとって大きなメリットである．ユーザはそれぞれのニーズに最適な OSS を選択し，原則的に無償で利用することができる．しかしながら，膨大な数の OSS の中から，安定した品質，かつ，サポートが継続して得られそうなものを選択することは容易ではない．また，ボランティア開発者主導の OSS プロジェクトは，商用ソフトウェアの開発に比べて品質管理や人的資源管理が安定しているとは言い難い．さらに，プロジェクトが分裂したり，消滅したりするリスクもある．独立行政法人 情報処理推進機構（IPA）の調査においても,「緊急時の技術的なサポートが得にくいこと」が，ソフトウェア開発企業が OSS を利用する上で最も大きなデメリットとして挙がっている [52].

近年では，ボランティア開発者によって支えられている OSS の課題について，実社会の経済活動を揺るがすほどの影響を与える事例が散見されている（例えば，OpenSSL の Heartbleed や，Apache Struts1 の脆弱性問題など）．大半の OSS は少人数で開発されているにもかかわらず [27]，社会に広く普及している場合も多く，重大な不具合が潜在していたり，不具合が顕在化してもすぐに対応できないことがある．このような事態を受け，巨大ベンダ（Google，Amazon，Cisco など）が OSS プロジェクトをサポートしたり†3，OSS の脆弱性パッチを修正した者に対して報奨金を支払うなど †4の試みもおこなわれるようになっている．これらは，OSS の価値が改めて評価されていることの一例であるが，世界の IT インフラを支える大企業が OSS に対して積極的に投資をおこなうという新たな流れは今後も継続するものと思われる．

†3 The Linux Foundation が主催するプロジェクト Core Infrastructure Initiative (CII) は IT 業界とその利害関係者が協力し合い，支援を必要としているオープンソース プロジェクトに対して資金援助する活動をしている．
Core Infrastructure Initiative: `http://www.linuxfoundation.org/programs/core-infrastructure-initiative` (accessed 2014-9-22)

†4 Vulnerability Reward Program: `https://www.google.com/about/appsecurity/patch-rewards/` (accessed 2014-9-22)

このように，新たな取り組みが見られる OSS 開発であるが，OSS および OSS 開発の成否は，プロジェクトに参加する開発者個々人の動機や社会性，プロジェクトの統治機構やプロジェクト内部での人間関係など，心理学・社会学的要素が強く関係する．そのため，定量的・体系的にソフトウェアの品質や生産性にアプローチする従来のソフトウェア工学では未だ解決が困難な課題が数多く存在する．本稿では，OSS および OSS を活用したシステムを利用する組織・個人，および，OSS を利用してシステム開発をおこなうベンダ企業が,「安心して安全に」OSS を利活用するための技術的方法論の確立を目指す学術分野を「オープンソースソフトウェア工学」と位置づける．学術会議としては，International Conference on Open Source Systems が 2005 年から開催されおり，今年 2015 年は，国際会議 ICSE (International Conference on Software Engineering) の併設会議として開催された†5．海外では以前から注目を集めている分野であるが，国内ではオープンソースソフトウェア工学に関する研究は未だ数少ない．本稿では，著者らがこれまでに取り組んできた研究事例を交えながら，オープンソースソフトウェア工学の現状と今後の発展について解説する．

続く 2 章では，OSS の歴史と開発形態について解説する．3 章では，OSS の開発と利活用におけるステークホルダについて解説する．4 章では，OSS 工学の研究意義について述べ，5 章では，これまでに OSS 工学としてまとめられた研究を紹介する．6 章では OSS 工学の発展について説明し，7 章で本稿のまとめをする．

2 OSS 開発の特徴

本章では，OSS の一般的な開発形態の特徴について解説する．Richard Stallman のフリーソフトウェア運動 [36] に端を発して今日に至る OSS 開発は，一般的な商用ソフトウェア開発企業の開発形態とは全く異なる特徴がある．OSS プロジェクトの開発形態は，Eric Raymond がバザール方式 [32] と名付けた

†5 International Conference on Open Source Systems 2015: `http://www.oss2015.org/`

ように，不特定多数の開発者がそれぞれ拡張したプロダクトを持ち寄り協力して作業をおこなう点にある．以下に，OSS 開発，すなわち，バザール方式のソフトウェア開発の具体的な特徴 [53] を挙げる．

(1) 開発者が自由に参加/離脱することが可能
OSS プロジェクトは不特定多数の開発者が自由に参加/脱退することが可能である．OSS 開発の初期段階ではプロジェクトの創立者を中心にソフトウェアが作り出されるが，開発が進むにつれ多数の開発者が参加し，プロジェクト規模が拡大する．プロジェクトに興味を失えば開発者は任意のタイミングでプロジェクトから離脱することができる．

(2) ボランティアでの参加が基本
OSS プロジェクトへ参加する目的は，金銭的な報酬を得ることではなく，高品質・高機能なソフトウェアの実現，世界中の参加者との協調作業の魅力や開発技術の学習や共有など，個人の知的好奇心の充足である場合が多い．したがって，OSS プロジェクトへの参加はボランティアであることが主流である．ただし，昨今では，企業の開発者，及び，OSS プロジェクトに雇用される開発者が参加しているため，金銭的な報酬を得ている開発者も増加傾向にある．

(3) 厳格な指揮系統が存在しない
開発者間での厳格な指揮系統は存在せず，バザールでの売買のような個人中心で自由な環境で開発がおこなわれる．このため開発者のコミュニケーションネットワークは，整理された階層構造は存在せず，動的に参加者が増減し相互に絡み合う複雑な構造を取る．

(4) ネットワークを介した分散開発環境
地理的に離れた世界中の参加者が，ネットワークを介して開発をおこなう．したがって，コミュニケーションを取るための手段としては，メーリングリスト（ML）や掲示板などの非対面でかつ非同期な媒体が基本となる．

バザール方式のソフトウェア開発には，「目玉の数さえ十分あれば、どんなバグも深刻ではない」 [32] という有名な譬え話がある．開発者らの共同作業によって意図しない不具合がプロダクトに混入したとしても，不特定多数の開発者の誰かが不具合を発見し対応するため，迅速な修正と短期リリースが可能であるという考え方である．しかし，近年の研究で，OSS 開発における作業量は開発者ごとに大きく異なることが示されている [12] [18] [22]．特に大規模 OSS 開発プロジェクトであっても，少数の開発者が大半の作業（機能拡張，不具合修正，等）をおこなっているという事実は，OSS 開発プロジェクトの継続性の観点から今後大きな問題となる可能性がある．

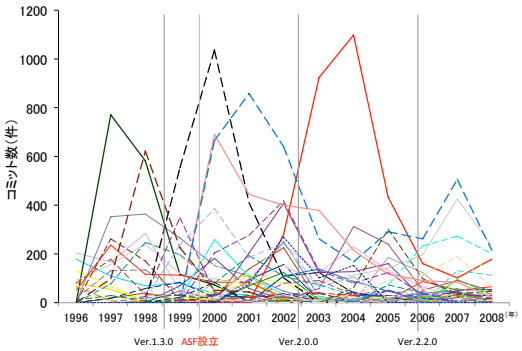


図 1 開発者のプロジェクト参加/離脱の流動性（Apache HTTP Server プロジェクト）

著者らがおこなった Apache HTTP Server プロジェクトの分析においても，開発者の作業量の不均衡と開発者の流動性について興味深い結果が得られている．図 1 は，Apache プロジェクトにおける各開発者のコミット数の推移を 1 年毎に集計した結果である．約 10 年の調査期間を通じてコミット数の多い状態を維持している開発者は存在せず，特定の期間には特定の少数の開発者が非常に高いアクティビティを示していることが見て取れる．多くの開発者は特定の期間に高いアクティビティを示し，それ以外の期間はアクティビティが低いか，あるいは，プロジェクトから離脱 †6 してしまうことを示している．Apache プロジェクトでは，開発者が入れ替わり循環しながら活動することにより高品質なソフトウェアが実現されていると言える．

†6 1 年以上コミットしていない開発者は離脱したとみなす．

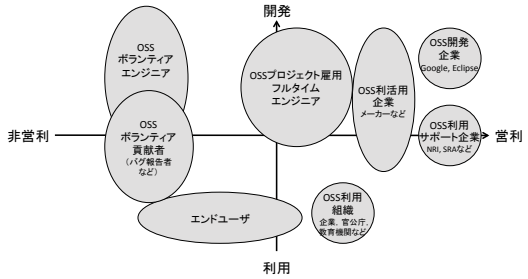


図 2 OSS の開発と利活用におけるステークホルダ

3 ステークホルダの広がり

前章では，ボランティアの開発者が中心となって開発を進める従来型の OSS 開発の特徴について解説した．従来型の OSS 開発では，そのエコシステムをボランティア開発者とエンドユーザという 2 種類のステークホルダから説明することが可能であったが，今日では，様々な目的と動機から OSS 開発のステークホルダは広がりを見せている．図 2 に示すように，OSS を利活用するために開発に関与する企業や，営利目的で商用ソフトウェアを OSS としてリリースする企業も増加しており，OSS 開発においても重要な役割を果たしている．本章では，OSS の開発と利活用において，近年特にその重要性が認識され始めたステークホルダの役割と課題について解説する．

3.1 OSS ボランティアエンジニア，貢献者

ボランティアエンジニアはプロダクトの開発，プロジェクトの運営に対価を得ずに取り組む開発者である．貢献者は開発に関する技術的な貢献ではなく，不具合報告，ドキュメント作成，翻訳などに取り組む人々を指す．昨今，OSS プロジェクトへの企業の参入が増加しているが，現在でも OSS に参加する多くの開発者は開発技術の向上，不特定多数の開発者との共同作業に興味を持つボランティアとして参加している．一方で，前章でも述べたように，ボランティアエンジニア，貢献者は自由に参加/離脱する可能性が高く，日々の行動時間も把握することが難しい．我々は，開発者間の協調作業の効率化に向けて，開発者間の Awareness 理解 †7 を支援する研究に取り組んできた．日中は企業で勤め，夜間や週末に作業することが多く，開発者間で進捗や行動時間を把握することは容易ではないため，開発者の行動時間をメールの送信履歴などから把握するためのシステムを開発した [49]．OSS 開発における円滑な共同作業の実現に向けて，開発者の行動パターンの把握は重要な課題といえる．

3.2 OSS プロジェクト雇用フルタイムエンジニア

プロダクトの開発，プロジェクトの運営をおこなうことで財団やスポンサー企業から対価を得ているエンジニアである．多くの OSS プロジェクトでは十分な運営資金がないため，フルタイムで雇用されているエンジニアは少なく，大きな課題となっている．1 章でも紹介した OpenSSL の脆弱性 Heartbleed は，人員不足が原因とも言われている．OpenSSL の問題を受けて，富士通や，Google，IBM などの大企業は，広く利用されている OSS を開発するプロジェクトに対して人的・財政的な支援を開始しており，OSS は社会経済活動にとって重要な公共知財と認識されつつある．高品質な OSS の安定的な供給を維持するためには，フルタイムエンジニアの雇用が重要な課題といえる．

3.3 OSS 開発企業

OSS 製品，OSS を利用したサービスを開発・供給する企業である．代表的な企業として，Apache Software Foundation，Mozilla Foundation などの非営利組織が挙げられる．これらの開発企業は，高品質な製品やサービスを開発することはもちろん，多くのユーザに使用されることも大きな目的としているため，普及率拡大にはサポート業務も求められる．したがって，製品・サービスの利用者から報告された不具

†7 Awareness（アウェアネス）とは複数人による共同作業を行う際の状況情報への「気づき」を指す．複数人で行うソフトウェア開発プロジェクトの共同作業では，開発者は開発者に関する状況，成果物の進捗状況，などを把握することが必要不可欠であり，共同作業を一緒に行うメンバーがどこで活動しているのか，どのような状況にあるのか，など Awareness を確保するための研究である．

合については，不具合の再現と原因の究明，不具合の修正といった作業をできる限り迅速におこなう必要があり，影響波及分析 [19][23] やバグローカリゼーション [24][30][37][45] などの技術の高度化と普及が今後の課題になると思われる．

3.4 OSS 利活用企業

自らがリリースするソフトウェアの一部として OSS の一部もしくは全体を流用・移植し，活用する企業である．OSS 利活用企業の開発者は，自社のソフトウェアに合わせて OSS を改変したり拡張するが，リリースされる OSS が必ずしも高品質なソフトウェアとは限らない．新しくリリースされた OSS には不具合が多く含まれていることが多いため，バージョンダウンする利用者も多く存在する [20]．OSS の機能性，品質の両面を検証・評価する技術や枠組みの確立が求められる．

3.5 OSS 利用サポート企業

企業による OSS の利活用をサポートする企業である．代表的な企業として Red Hat などの営利組織が挙げられる．昨今 OSS を利活用する企業が，OSS の導入，保守にかかる膨大なコストが課題の一つとなっている．OSS の利活用をサポートする企業が抱える課題については，本チュートリアル第 2 回「企業における OSS 活用の実際（仮題）」に続く第 3 回「OSS ビジネスにおける課題と工夫（仮題）」で解説される予定である．

3.6 OSS 利用組織とエンドユーザ

企業ではシステムや業務の効率化やコスト削減などを目的に，エンドユーザは商用ソフトウェアの代替として OSS を利用する．

しかし，OSS 開発に関する知識を有する専門家が常勤していない組織や個人での OSS の利用は，表 1 [52] に挙げたように,「プロジェクトから緊急時の技術的なサポートが得にくい」,「セキュリティホールへの対応に不安がある」などの不安材料がある．これらの不安を払拭するために，OSS プロジェクトでは通常，ユーザ用 ML などが用意され，OSS 開発者にのみ頼るのではなくエンドユーザ同士で助け合う場が提供されている．昨今では，プログラミング技術に関する Q&A サービス Stack Overflow などが相互扶助サービスとして人気がある．Stack Overflow に含まれる膨大な質問の中からエンドユーザが求める情報を推薦する技術に関する研究も多数おこなわれている [29][34].

4 OSS 工学の意義

4.1 ソフトウェア工学との違い

ソフトウェア工学は，営利活動を目的とするソフトウェア開発企業を支援するための諸技術を提供し，ソフトウェア開発の生産性と品質を高めることが主たる貢献である．その一方，OSS 工学は，営利非営利に関わらず，OSS を開発する組織を支援するための諸技術を提供する．また，OSS 工学では，OSS の利用者も OSS 開発への貢献者と捉える．OSS 開発の支援のみならず，OSS の利用を通じて得られる知見を OSS 開発へフィードバックするサイクルまでを，OSS 開発の生態系（エコシステム）として支援する．OSS を安全に安心して利活用できるようにすることが主な貢献となる．

昨今，OSS プロジェクトに商用ソフトウェア開発企業の開発者が参加することがあり，ソフトウェア開発企業と OSS のそれぞれの開発者が連携できる環境が整い，徐々に OSS 開発の生態系が機能しつつある．また，商用ソフトウェアの開発データ（ソースコード等），および，開発記録を公開する取り組みもあり，これまでのソフトウェア開発企業のようなクローズドな開発形態を対象としたソフトウェア工学の知見では理解できないことも今後発生すると予想される．OSS 工学における，プロジェクトに自主的に貢献する不特定多数の開発要員の行動，及び，行動心理を理解し，ソフトウェア開発にフィードバックすることが今後求められる．

4.2 研究意義

ステークホルダが OSS を開発，利用する目的はそれぞれ異なり，各々の要求，課題を理解した上で行動することがステークホルダ間が連携/協調するために

表 1 企業が考える OSS のメリット・デメリット

メリット	
低価格で顧客に提供できること	61.9%
特定のベンダーに依存していないこと	46.8%
多くの種類の OSS から自社にあったものを利用できること	45.7%
関連情報がインターネット上に豊富に存在していること	45.0%
ソースコードを参照し，変更することができること	35.5%
デメリット	
緊急時の技術的なサポートが得にくいこと	67.3%
利用している OSS がいつまで存続するかがわからないこと	58.8%
バグの回収や顧客からの要請対応に手間がかかること	43.4%
将来のバージョンアップの計画が把握しにくいこと	37.6%
セキュリティホールに対するプロジェクトの対応に不安があること	31.1%

必要である．しかし，実際には他人の行動を把握して行動することは難しく，連携することが困難なこともある．例えば，OSS の開発サイクルは短く，開発者は新しいバージョンの開発に集中するため，旧バージョンが長期間保守されない場合もある．その一方で，利活用者は，旧バージョンの保守がおこなわれていないことを把握しないまま継続的に利用した結果，1 章で述べたような OSS の脆弱性により，社会を揺るがす問題に発展することがある．OSS 工学では，利活用者が開発者のシーズを，開発者が利用者のニーズを共有する仕組みを確立することにより，ステークホルダ間の円滑な連携/協調の実現を目指す．その一つのアプローチとして，これまでのソフトウェア工学の研究では，OSS の開発データを活用したアプローチ（リポジトリマイニングなど）が進められており，5 章で紹介する開発動向の理解，保守作業の支援技術などが数々提案されている．OSS を開発，利活用する各々のステークホルダが他人を意識し，それぞれの行動，思想を理解した上でソフトウェアを自由に活用し，発展させる生態系を確立する．

5 OSS 工学の研究事例

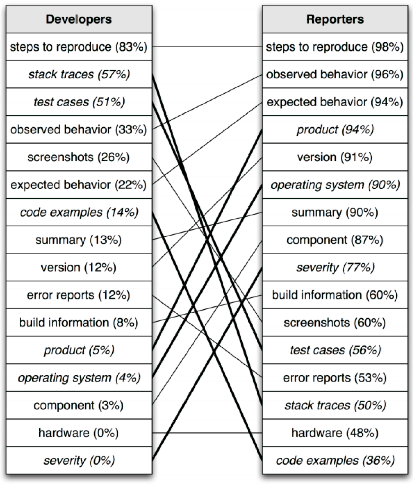
本章では，OSS 開発，利活用に関する研究事例を紹介する．

5.1 品質管理・理解

多くの OSS 開発では，ソースコード，不具合情報，レビュー情報などの共有・追跡のために様々なシステムが利用されている．具体的には，バージョン管理システム（Git[†8]，Subversion[†9]等），不具合追跡システム（Bugzilla[†10]，JIRA[†11]等），レビュー管理システム（gerrit[†12]，rietve[†13]等）などが挙げられる．基本的に誰でも，これらのシステムに記録される情報にアクセスすることができる．不特定多数の開発者による協調作業にとって必要不可欠なシステムであると言える．しかし一方で，不特定多数の開発者（エンドユーザも含む）が自由にプロジェクトに参加しこれらのシステムを利用するがゆえに，システムの利用者がプロジェクトの運用ルールに従わない，あるいは，従えない場合がある．その結果，過去に報告済み，修正済みの不具合が繰り返し報告される，修正のために必要な情報が記載されていないなどの問題が生じている．

例えば，多くの OSS プロジェクトでは，効率的に不具合を修正するために，不具合追跡システムの利用

†8 Git: `https://git-scm.com/`
†9 Subversion: `https://subversion.apache.org/`
†10 Bugzilla: `https://www.bugzilla.org/`
†11 JIRA: `https://www.atlassian.com/software/jira`
†12 gerrit: `https://code.google.com/p/gerrit/`
†13 rietve: `https://code.google.com/p/rietveld/`



(b) Most helpful for developers vs. provided by reporters.

図 3　開発者が不具合修正するために有用な情報と不具合報告者が実際に報告する内容の不一致 [2]

者（主に不具合報告のみをおこなうエンドユーザ）に向けて，公式 web サイトに不具合報告ルール（既に報告された不具合でないかを確認することや，エンドユーザ ML で事前に解決方法を相談すること等）を示している．しかし，大規模 OSS プロジェクトでは 1 日に数百件の不具合が報告されることもあるため，システムの利用者が不具合報告ルールを完全に守ることが難しい場合もある．不具合追跡システムに蓄積されている膨大な不具合報告の中から，類似する過去の不具合報告を各報告者が自力で見つけることが容易ではないためである．さらに，図 3 に示すように，報告者が提供する不具合情報と開発者が求める不具合情報にギャップが存在するため [2]，開発者が不具合を再現したり，不具合を特定することが困難な場合がある．

このように，不具合追跡システムに記録される大量の不具合報告は，不具合修正の効率を低下させる原因となるため，不具合情報を整理する研究が盛んにおこなわれてきた．例えば，報告される大量の不具合を一つずつ開発者が理解することは現実的ではないため，不具合が混入したモジュールの箇所や不具合報告に記載される文章（不具合の症状など）をマイニングし，重複する不具合を発見する技術 [25][38][40] や，不具合報告の要約文章を作成する技術 [31] などが提案されている．

開発者にとっては不具合報告の整理や理解が重要であるが，利活用者にとっては，報告された不具合が早急に修正されることが重要となる．そのため，不具合の修正を直接的に支援する研究も数多くおこなわれてきた．例えば，不具合混入ファイルの特定 [24][30][37][45]，不具合修正時間の予測 [3][14][47]，不具合修正の優先度推薦 [17] などの技術が提案されている．また，高品質な OSS を安全に利用するために，他者の OSS 利用数や不具合修正収束プロセスに基づく安定バージョンの推薦 [20][48] やライセンス管理支援 [10][11] なども提案されている．

なお，これまでの研究では基本的に，過去の開発プロダクトの軌跡（記録）を基にプロダクトの進化やステークホルダが形成するコミュニティの進化をモデル化している．しかしそれらは，日々進化する OSS プロジェクトがプロダクトを進化させている，あるいは，お互いに影響を及ぼす共進化の関係にあるということも考えられる [44][54]．昨今では，ソーシャルコーディングを代表例とした，ソフトウェア開発の人的・社会的側面が注目されているため，今後は開発プロダクトを中心に据えた分析や支援技術の研究よりも，コミュニティの進化がどのようにプロダクトに影響するのかを理解することに重点を置く研究が今後はより重要になる可能性がある．

5.2　人的管理

多くの OSS プロジェクトでは，不特定多数の開発者が参加/離脱を繰り返しつつも，高品質なソフトウェアを開発し続けている．ただし，コア開発者の離脱により開発が停滞することもある [43]．プロジェクトの組織体系や開発者の流動性を理解することは，OSS 利活用者にとって OSS の継続的な開発，保守を見据えるために重要である．

OSS プロジェクトにおける開発者の貢献量はパレートの法則に従っている（2 割の開発者が 8 割のプロダクトに貢献し，残り 8 割の開発者が 2 割のプロダクトに貢献している）ことが多くの研究で指摘されてい

る [12] [18] [22]．従って，OSS を継続的に開発，保守するためには，プロジェクトを支えるコア開発者の育成が必要不可欠といえる．

Jensen らは，コア開発者，特に，リリースされるマスターリポジトリへのアクセス権限が与えられる開発者（コミッター）の昇格制度を理解するために，Mozilla.org, Apache Software Foundation, Netbeans.org プロジェクトの開発者にインタビュー調査をおこなっている [15]．調査の結果，コミッター昇格は主に，(1) 既存のコミッターからの推薦，あるいは，(2) 開発者が立候補した後におこなわれる既存コミッター間の議論，を通じておこなわれることが明らかとなった．しかし同時に，いずれのプロジェクトにおいてもコミッターの評価・選出基準は明確ではなく，コミッター候補者の過去の活動内容（具体的には機能拡張や不具合修正に関連する活動）やコミュニケーション能力などを，既存コミッターが彼らの経験や勘に基づいて総合的に評価していることが分かった．

著者らも，大規模 OSS プロジェクトに貢献するコミッターを対象として，一般開発者を新規コミッターに昇格させる際の評価方法についてインタビュー調査をおこなっている．調査結果から，コミュニケーション能力よりもプロダクトへの貢献が高く評価される傾向があることが明らかとなった [50]．これらの調査結果を踏まえ，我々は，昇格前の開発者の活動期間や活動量を機械学習することで，コミッター権限を付与するに足る開発者を推薦するためのモデルを提案している [51]．また，同様のモデルとして，プロジェクトに長期間貢献する可能性の高い開発者を早い段階で特定するためのモデルがある [46]．小規模プロジェクトではプロジェクト参加から数ヶ月以内に，大規模プロジェクトでも 1 年以内に長期間貢献する開発者を特定できることが示されている．これらの研究では，開発者の貢献を計測することで将来的にプロジェクトを支える開発者を特定しているが，日々変化するプロダクトに開発者が具体的にどのように貢献しているかの詳細については未だ不明な点も多い．開発者の貢献内容とプロジェクトの継続性の関係をより深く理解することは今後の課題である．

プロジェクトを支える開発者として，コーディネータと呼ばれるコア開発者の存在が重要であることも指摘されている．コーディネータは，開発者が形成するコミュニティとエンドユーザが形成するコミュニティの仲介や調整（例えば，ユーザからのフィードバックを開発者に伝達しプロダクトに反映させる，など）をおこなっており，プロジェクト全体の健全性とプロダクトの品質を維持する上で極めて重要な役割を担っている．著者らは，コーディネータがどのようにプロジェクトを支えているのかを理解するために，Apache，GIMP，Netscape を対象にケーススタディをおこなっている．開発者と利用者の ML を仲介するコーディネータを特定し，両 ML に積極的に参加するコーディネータが存在するコミュニティは，成功の可能性が高くなることを明らかにした [16] [53]．OSS プロジェクトにおいては，成果物に対する貢献だけでなく，プロジェクトに参加する開発者，利用者など，人と人との間を調整するコーディネータの役割も理解する必要がある．

また，昨今では，複数の OSS に貢献する開発者も数多く存在する．Yamashita らは，OSS プロジェクトにおける開発者の Magnet（引力性: 新規開発者の参加）と Sticky（定着性: 継続して同一プロジェクトに貢献）の性質を分析している．成功しているプロジェクト（Homebrew や Django など）ほど高い Magnet と Sticky を持つことを定量的に示している [4] [41] [42]．今後は，新たな開発者を誘い込むアプローチを確立することが，最新の知識や技術を取り込みながらプロジェクトを維持・発展させる上で重要になると思われる．

5.3 コミュニケーション管理・理解

OSS プロジェクトは，世界中に点在する開発者が共同開発をおこなう分散開発の形態をとることが多く，分散開発の成功事例としてどのようにコミュニケーションをとっているのかを調査した研究が数多く報告されている．

例えば，SourceForge.net を対象にした Robles らの調査 [33]，FreeBSD プロジェクトを対象にした Spinellis [35] の調査によると，OSS 開発は北米，ヨーロッパを中心に，アジア，オーストラリア，南アフリ

カ，南米など，地域や時差をまたがったグローバル環境下で非対面・非同期のコミュニケーションがおこなわれている．非対面・非同期のコミュニケーションによって，開発者は自身の都合の良いタイミングで，地域の異なる開発者に情報を発信できる．このような分散開発のメリットは，世界中に点在するいずれかの開発者による 24 時間体制での議論を通じた効率的な開発を実施できる点にある [8][21]．ただし，情報交換にタイムラグが生じる場合は，開発スピードや品質管理能力の低下など，ソフトウェア開発全体に悪影響をもたらすとも言われており [1][6][7][13][21][26]，OSS 開発における情報交換の実態をより正確に把握することは，OSS を利活用する上で重要になる．著者らは，12 の OSS プロジェクトを対象に，異なるタイムゾーンで活動する開発者間の電子メールのやりとりを調査している [55]．その結果，情報交換にタイムラグが発生しているプロジェクトが存在することや，プロジェクトによっては時差の異なる地域の開発者らが情報交換のタイミングを合わせてコミュニケーションを図っていることが分かった．

不特定多数の開発者が参加するプロジェクトにおける協調作業を理解することは容易ではないため，巨視的に協調作業の様子を分析するアプローチとしてソーシャルネットワーク分析が用いられてきた．ソーシャルネットワーク分析は，一個人，一組織をノード，個々のノード間の関係をエッジで表現し，個人間，組織間の関係性をネットワークとしてモデル化することで，複雑な社会現象の解釈を容易にするためのアプローチである．ネットワークモデルの構造的な特徴（ノード間の関係の強さ，ネットワークにおけるノードの中心度合い）を数値化する手法が数多く提案されており，組織・社会構造の性質を定量的に分析できるという特徴がある．従来研究では，開発者や利用者間の協調作業を理解するために，OSS プロジェクトにおける ML や掲示板におけるメッセージの送信者と返信者の関係からコミュニケーションネットワークを構築している．まつ本らは，ネットワーク分析を通して，開発者（またはユーザ）間の知識共有，議論の取りまとめの役割を担う開発者（コーディネータ）の存在がプロジェクトの維持，拡大に重要であることを結論付けている [53]．また，Bird らは大規模 OSS プロジェクトにおける開発者ネットワークの分析をおこない，いくつかのサブグループを抽出している．サブグループの開発者が密に連携している程，そのグループが開発するプロダクトの品質が高いことを示している [5]．ソーシャルネットワーク分析は，OSS 開発におけるコミュニケーションや協調作業を分析する上で強力なツールとして利用されてきたが，分析対象とするメディア（ML や不具合追跡システム）が異なれば同じプロジェクトであっても異なる結果や解釈が得られることがあり [28]，一貫性のある分析結果を得るためにはインタビュー等によって開発者に分析結果の妥当性を確認してもらうことが望ましい．

5.4 OSS 利活用のサポート

OSS に関する情報は，プロジェクトが公開するものだけとは限らない．昨今では，多くのソフトウェア開発者がブログ，ソーシャルサービスなどに OSS の利用方法などを投稿している．特に，Stack Overflow では，プログラミング技術に関する質問が数多く登録されている．2015 年 4 月現在で 900 万以上の質問があり，OSS の開発者・利用者にとっては技術ドキュメントの一種として利用されている [39]．国際会議 MSR (The Working Conference on Mining Software Repositories) 2013，2015 では，会議のオフィシャル web サイトで，Stack Overflow のデータセットを提供し †14†15，Stack Overflow をより活用するための技術についての論文募集がおこなわれた．具体的な研究としては，タグ付けの自動化，accepted answer の推薦などが挙げられている．今後は，Q&A サービスだけでなく，Web に投稿される記事から，信頼性の高い記事を発見するための技術などが期待される．

†14 MSRChallenge2013: `http://2013.msrconf.org/challenge.php`

†15 MSRChallenge2015: `http://2015.msrconf.org/challenge.php`

6 OSS 工学分野の展望

3 章で述べた OSS の利活用企業や利用組織は，利活用すべき OSS の選定の際に，OSS の品質や導入実績を調査している．また，OSS プロジェクトからのサポートが得られるかどうかについて，プロジェクトの運営状況から調査することもある．

しかし，様々な調査に基づいて利活用を決定した OSS が，恒久的に最適な選択であるかどうかは保障されない．その理由は，OSS 開発では日常的に新規あるいは派生のプロジェクトが誕生し，選択した OSS が機能面や品質面で常に優れているとは限らないからである．また，安全性や安定性の高さから導入を決定した OSS にセキュリティを脅かす不具合が検出されることもある．

OSS 工学では，公開されている膨大な OSS 開発記録を対象に詳細な分析をおこなうことができる．計算機の処理能力の向上により，今後は，OSS プロジェクトおよびプロダクトの進化の過程を数万件のプロジェクトを対象に解析することも不可能ではない．現時点での OSS の「健康状態」の把握のみならず，進化パターンの予測に基づいたプロジェクト継続性や品質の推定といった技術が求められる．

また，近年では，OSS 利活用企業や利用組織が積極的に OSS プロジェクトに参加・貢献する機会が増えている．今後もこの傾向は継続するものと考えられる．安心・安全な OSS を社会全体で恒久的に流通できるよう，ボランティアエンジニアと企業開発者らが共にメリットを得られる新たな開発形態を模索することも，OSS 工学に課せられた使命であると言える．

7 おわりに

本稿では，OSS および OSS を活用したシステムを利用する組織・個人，および，OSS を利用してシステム開発をおこなうベンダ企業が，「安心して安全に」OSS を利活用するための技術的方法論の確立を目指す学術分野「オープンソースソフトウェア工学」について，著者らがこれまでに取り組んできた研究事例を交えながら解説した．

かつてのソフトウェア開発企業は，OSS を利活用する主な目的がコスト削減であったが，昨今では，その理由に限らず，商用ソフトウェアに劣らない高機能，高品質な OSS を利活用することが目的となっている．さらには，ソフトウェア開発企業が，商用ソフトウェアのソースコードを一部公開し，外部の開発者による拡張，不具合修正を期待するなど，OSS の開発形態までもがソフトウェア開発企業に導入されつつある．

今後，OSS の導入，利活用を加速するために，利活用者がこれまで以上に OSS 開発に貢献し，OSS 開発者と利活用者の間で協力し合うエコシステムを形成することが必要不可欠である．そうすることで，OSS 開発と，商用ソフトウェア開発との区別が徐々になくなり，限られた開発者だけでなく，広い視点からソフトウェアの機能性，品質が拡張され，向上すると考える．

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，および，文部科学省科学研究補助金（基盤 (C): 15K00101）による助成を受けた．

参 考 文 献

[1] Al-Ani, B. and Edwards, H. K.: A Comparative Empirical Study of Communication in Distributed and Collocated Development Teams, *Proceedings of the 3rd International Conferenceon Global Software Engineering (ICGSE'08)*, 2008, pp. 35–44.

[2] Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., and Zimmermann, T.: What makes a good bug report?, *Proceedings of the 16th International Symposium on Foundations of Software Engineering (FSE'08)*, 2008, pp. 308–318.

[3] Bhattacharya, P. and Neamtiu, I.: Bug-fix Time Prediction Models: Can We Do Better?, *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR'11)*, 2011, pp. 207–210.

[4] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A., and Hsu, G.: Open Borders? Immigration in Open Source Projects, *Proceedings of the 4th International Workshop on Mining Software Repositories (MSR'07)*, 2007, pp. 6–13.

[5] Bird, C., Pattison, D., D'Souza, R., Filkov, V., and Devanbu, P.: Latent Social Structure in Open Source Projects, *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE'08)*, 2008, pp. 24–35.

[6] Cataldo, M. and Nambiar, S.: On the relationship between process maturity and geographic

distribution: an empirical analysis of their impact on software quality, *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)*, 2009, pp. 101–110.

[7] Colazo, J. A.: Following the Sun: Exploring Productivity in Temporally Dispersed, *Proceedings of the 14th Americas Conference on Information Systems (AMCIS'08)*, 2008.

[8] Crowston, K. and Scozzi, B.: Open source software projects as virtual organizations: Competency rallying for software development, *IEEE Proceedings Software*, Vol. 149, No. 1(2002), pp. 3–17.

[9] David, P. A., Waterman, A., and Arora, S.: FLOSS-US: The Free/Libre/Open Source Software Survey for 2003, http://www.stanford.edu/group/floss-us/, (accessed 2015-05-23).

[10] German, D. M., Manabe, Y., and Inoue, K.: A Sentence-matching Method for Automatic License Identification of Source Code Files, *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE'10)*, 2010, pp. 437–446.

[11] German, Daniel M.and Di Penta, M. and Davies, J.: Understanding and Auditing the Licensing of Open Source Software Distributions, *Proceedings of the 18th International Conference on Program Comprehension (ICPC'10)*, 2010, pp. 84–93.

[12] Goeminne, M. and Mens, T.: Evidence for the pareto principle in open source software activity, *Porceedings of the 1st International workshop on Model Driven Software Maintenance and 5th International Workshop on Software Quality and Maintainability*, 2011, pp. 74–82.

[13] Herbsleb, J. D., Mockus, A., Finholt, T. A., and Grinter, R. E.: An Empirical Study of Global Software Development: Distance and Speed, *Proceedings of the 23rd International Conference on Software Engineering (ICSE'01)*, 2001, pp. 81–90.

[14] Hewett, R. and Kijsanayothin, P.: On Modeling Software Defect Repair Time, *Empirical Software Engineering*, Vol. 14, No. 2(2009), pp. 165–186.

[15] Jensen, C. and Scacchi, W.: Role migration and advancement processes in OSSD projects: a comparative case study, *Proceedings of the International Conference on Software Engineering (ICSE'07)*, 2007, pp. 364–374.

[16] Kamei, Y., Matsumoto, S., Maeshima, H., Ohnishi, Y., Ohira, M., and Matsumoto, K.: Analysis of Coordination between Developers and Users in the Apache Community, *Proceedings of the 4th International Conference on Open Source Systems (OSS'08)*, 2008, pp. 81–92.

[17] Kim, D., Wang, X., Kim, S., Zeller, A., Cheung, S. C., and Park, S.: Which Crashes Should I Fix First?: Predicting Top Crashes at an Early Stage to Prioritize Debugging Efforts, *IEEE Transactions on Software Engineering*, Vol. 37, No. 3(2011), pp. 430–447.

[18] Koch, S.: Effort, cooperation and coordination in an open source software project: Gnome, *Information Systems Journal*, Vol. 12, No. 1(2002), pp. 27–42.

[19] Kouroshfar, E.: Studying the Effect of Co-change Dispersion on Software Quality, *Proceedings of the 2013 International Conference on Software Engineering (ICSE'13)*, 2013, pp. 1450–1452.

[20] Mileva, Y. M., Dallmeier, V., Burger, M., and Zeller, A.: Mining trends of library usage, *Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops (IWPSE-Evol'09)*, 2009, pp. 57–62.

[21] Milewski, A. E., Tremaine, M., Egan, R., Zhang, S., Kobler, F., and O'Sullivan, P.: Guidelines for Effective Bridging in Global Software Engineering, *Proceedings of the 3rd International Conferenceon Global Software Engineering (ICGSE'08)*, 2008, pp. 23–32.

[22] Mockus, A., Fielding, R. T., and Herbsleb, J. D.: Two case studies of open source software development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology*, Vol. 11, No. 3, 2002, pp. 309–346.

[23] Mondal, M., Roy, C. K., and Schneider, K. A.: Prediction and Ranking of Co-change Candidates for Clones, *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR'14)*, 2014, pp. 32–41.

[24] Nguyen, A. T., Nguyen, T. T., Al-Kofahi, J., and Nguyen, H. V.: A topic-based approach for narrowing the search space of buggy files from a bug report, *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE'11)*, 2011, pp. 263–272.

[25] Nguyen, A. T., Nguyen, T. T., Nguyen, T. N., Lo, D., and Sun, C.: Duplicate Bug Report Detection with a Combination of Information Retrieval and Topic Modeling, *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE'12)*, 2012, pp. 70–79.

[26] Nguyen, T., Wolf, T., and Damian, D.: Global Software Development and Delay: Does Distance Still Matter?, *Proceedings of the 3rd International Conferenceon Global Software Engineering (ICGSE'08)*, 2008, pp. 45–54.

[27] Ohira, M., Ohsugi, N., Ohoka, T., and Matsumoto, K.: Accelerating Cross-Project Knowledge Collaboration Using Collaborative Filtering and Social Networks, *Proceedings of the 2005 International Workshop on Mining Software Repositories (MSR 2005)*, July 2005, pp. 111–115.

[28] Panichella, S., Bavota, G., Penta, M. D., Canfora, G., and Antoniol, G.: How Developers ' Col-

laborations Identified from Different Sources Tell us About Code Changes, *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME'14)*, 2014, pp. 251–260.

[29] Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., and Lanza, M.: Mining StackOverflow to Turn the IDE into a Self-confident Programming Prompter, *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR'14)*, 2014, pp. 102–111.

[30] Rao, S. and Kak, A.: Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE '11), *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR'11)*, 2011, pp. 43–52.

[31] Rastkar, S., Murphy, G. C., and Murray, G.: Automatic Summarization of Bug Reports, *IEEE Transactions on Software Engineering*, Vol. 40, No. 4(2014), pp. 366–380.

[32] Raymond, E. S.: *The cathedral and the bazzar: musings on linux and open source by an accidental revolutionary*, O'Reilly and Associates, Sebastopol, CA, 1999.

[33] Robles, G. and Gonzalez-Barahona, J. M.: Geographic location of developers at SourceForge, *Proceedings of the 5th International Working Conference on Mining Software Repositories (MSR'06)*, 2006, pp. 144–150.

[34] Saha, A. K., Saha, R. K., and Schneider, K. A.: A Discriminative Model Approach for Suggesting Tags Automatically for Stack Overflow Questions, *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR'13)*, 2013, pp. 73–76.

[35] Spinellis, D.: Global software development in the FreeBSD project, *Proceedings of the 2006 international workshop on Global software development for the practitioner (GSD'06)*, 2006, pp. 73–79.

[36] Stallman, R. M.: *Open Sources : Voices from the Open Source Revolution*, O'Reilly Media; 1st edition, 1999.

[37] Tantithamthavorn, C., Teekavanich, R., Ihara, A., and Matsumoto, K.: Mining a Change History to Quickly Identify Bug Locations : a Case Study of the Eclipse Project, *Proceedings of the IEEE 24th International Symposium on Software Reliability Engineering (ISSRE' 13)*, 2013, pp. 108–113.

[38] Thung, F., Kochhar, P. S., and Lo, D.: DupFinder: Integrated Tool Support for Duplicate Bug Report Detection, *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE'14)*, 2014, pp. 871–874.

[39] Vasilescu, B., Serebrenik, A., Devanbu, P., and Filkov, V.: How Social Q&A Sites Are Changing Knowledge Sharing in Open Source Software Communities, *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW'14)*, 2014, pp. 342–354.

[40] Wang, X., Zhang, L., Xie, T., Anvik, J., and Sun, J.: An Approach to Detecting Duplicate Bug Reports Using Natural Language and Execution Information, *Proceedings of the 30th International Conference on Software Engineering (ICSE'08)*, 2008, pp. 461–470.

[41] Yamamoto, M., Ohira, M., Kamei, Y., Matsumoto, S., and Matsumoto, K.: Temporal Changes of the OpenOpen of an OSS Community: a Case Study of the Apache HTTP Server Community, *Proceedings of the 5th International Conference on Collaboration Technologies (CollaboTech'09)*, 2009, pp. 64–65.

[42] Yamashita, K., McIntosh, S., Kamei, Y., and Ubayashi, N.: Magnet or Sticky? An OSS Project-by-project Typology, *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR'14)*, 2014, pp. 344–347.

[43] Ye, Y. and Kishida, K.: Toward an Understanding of the Motivation Open Source Software Developers, *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, 2003, pp. 419–429.

[44] Ye, Y., Nakakoji, K., Yamamoto, Y., and Kishida, K.: *The Co-Evolution of Systems and Communities in Free and Open Source Software Development*, Chapter 3, Idea Group Publishing, Hershey, PA, 2004.

[45] Zhou, J., Zhang, H., and Lo, D.: Where should the bugs be fixed? - more accurate information retrieval-based bug localization based on bug reports, *Proceedings of the 34th International Conference on Software Engineering (ICSE'12)*, 2012, pp. 14–24.

[46] Zhou, M. and Mockus, A.: What Make Long Term Contributors: willingness and Opportunity in OSS Community, *Proceedings of the 34th International Conference on Software Engineering (ICSE'12)*, 2012, pp. 518–528.

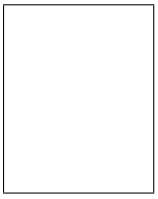
[47] 正木仁, 大平雅雄, 伊原彰紀, 松本健一: OSS 開発における不具合割当パターンに着目した不具合修 ☒時間の予測, 情報処理学会論文誌, Vol. 54, No. 2(2013), pp. 933–944.

[48] 藤野啓輔, 伊原彰紀, 本田澄, 鷲崎弘宜, 松本健一: OSS の不具合修正曲線に基づく未修正不具合数の予測の試 , 第 *21* 回ソフトウェア工学の基礎ワークショップ *(FOSE'14)*, 2014, pp. 57–62.

[49] 伊原彰紀, 山本瑞起, 大平雅雄, 松本健一: OSS 開発における保守対応の効率化のためのアウェアネス ☒腑轡好膳, 情報処理学会 マルチメディア, 分散, 協調とモバイル *(DICOMO'10)* シンポジウム論文集, 2010, pp. 1620–1629.

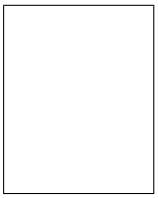
[50] 伊原彰紀, 亀井靖高, 大平雅雄, ブラムアダムス, 松本健一: OSS プロジェクトにおけるコミッターの承認に対する同 の理解, 第 *21* 回ソフトウェア工学の基礎ワークショップ（*FOSE'14*）, 2014, pp. 45–50.

[51] 伊原彰紀, 亀井靖高, 大平雅雄, 松本健一, 鵜林尚靖: OSS プロジェクトにおけるコミッターの承認に対する同　の理解発車の活動量を用いたコミッター候補者予測, 電子情報通信学会, 2012, pp. 237–249.
[52] 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ェア活用ビジネス実態調査（2009 年調査）.
[53] まつ本真佑, 亀井靖高, 大平雅雄, 松本健一: OSS コミュニティにおけるコラボレーションの理解, 情報社会学会, Vol. 3, No. 2(2010), pp. 29–42.
[54] 山谷陽亮, 大平雅雄, Passakorn, P., 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データマイニング手法, 情報社会学会理学会論文誌, Vol. 56, No. 1(2015), pp. 59–71.
[55] 亀井靖高, 大平雅雄, 伊原彰紀, 小山貴和子, まつ本真佑, 松本健一, 鵜林尚靖: グローバル環境下における OSS 開発者の情報交換に対する　差の影響, 情報社会学会, Vol. 6(2011), pp. 13–30.



伊 原 彰 紀

2007 年龍谷大学理工学部卒業．2009 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．2012 年同大学院同研究科博士後期課程修了．同年同大学院同研究科助教．博士 (工学)．ソフトウェア工学，特にリポジトリマイニング，オープンソースソフトウェア開発・利用支援の研究に従事．電子情報通信学会，情報処理学会，IEEE 各会員．



大 平 雅 雄

1998 年京都工芸繊維大学工芸学部電子情報工学科卒業，2003 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同大学情報科学研究科助教を経て，現在和歌山大学システム工学部准教授．博士（工学）．ソフトウェア工学，特にリポジトリマイニング，ソフトウェア開発における知的協調作業支援の研究に従事．電子情報通信学会，情報処理学会，ヒューマンインタフェース学会，ACM 各会員．