目次

1. 研究背景と目的

Free/Libre and Open Source Software (以降，FLOSS と称す) は，誰でも自由に利用することができるソフトウェアでありながら，非常に高い品質を実現している。高品質を実現するメカニズムとして，Eric Raymond は自身の著書である「The Cathedral and the Bazaar」の中で「十分な目があれば，全てのバグは洗い出される」と解説している [1] 。つまり，「十分なベータテスターと共同開発者がいれば，ほとんどのバグは直ぐに発見され，修正される」ということである。誰でも自由に利用することができ，高品質であるため，様々な分野で FLOSS は活用されている。

Synopsys, Inc.[1] が公開している FLOSS におけるセキュリティとリスク分析に関する調査報告書 (図 1) [2] によると，商用ソフトウェア (コードベース) の約 70% が FLOSS によって構成されている。また，Talk Openly, Develop Openly Group[2] が公開している FLOSS に関する調査報告書 (図 2) [3] によると，商用ソフトウェアの約 77% に FLOSS が用いられているなど，商用ソフトウェアの開発において FLOSS は重要な役割を担っている。

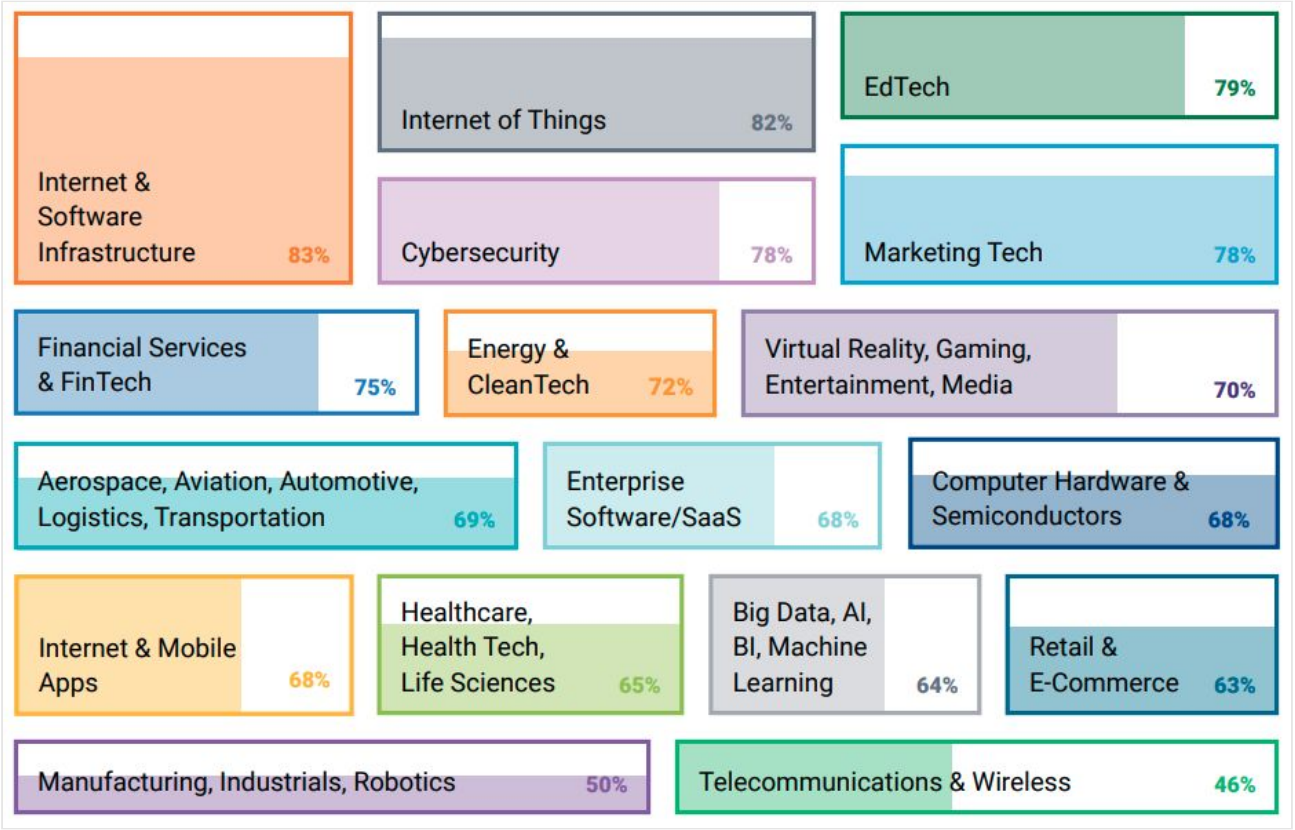


図 1 Synopsys, Inc. の調査報告結果 [2]

[1] 電子系設計ソフトウェアを開発・販売する企業
[2] FLOSS の活用や運用に関する情報を提供する

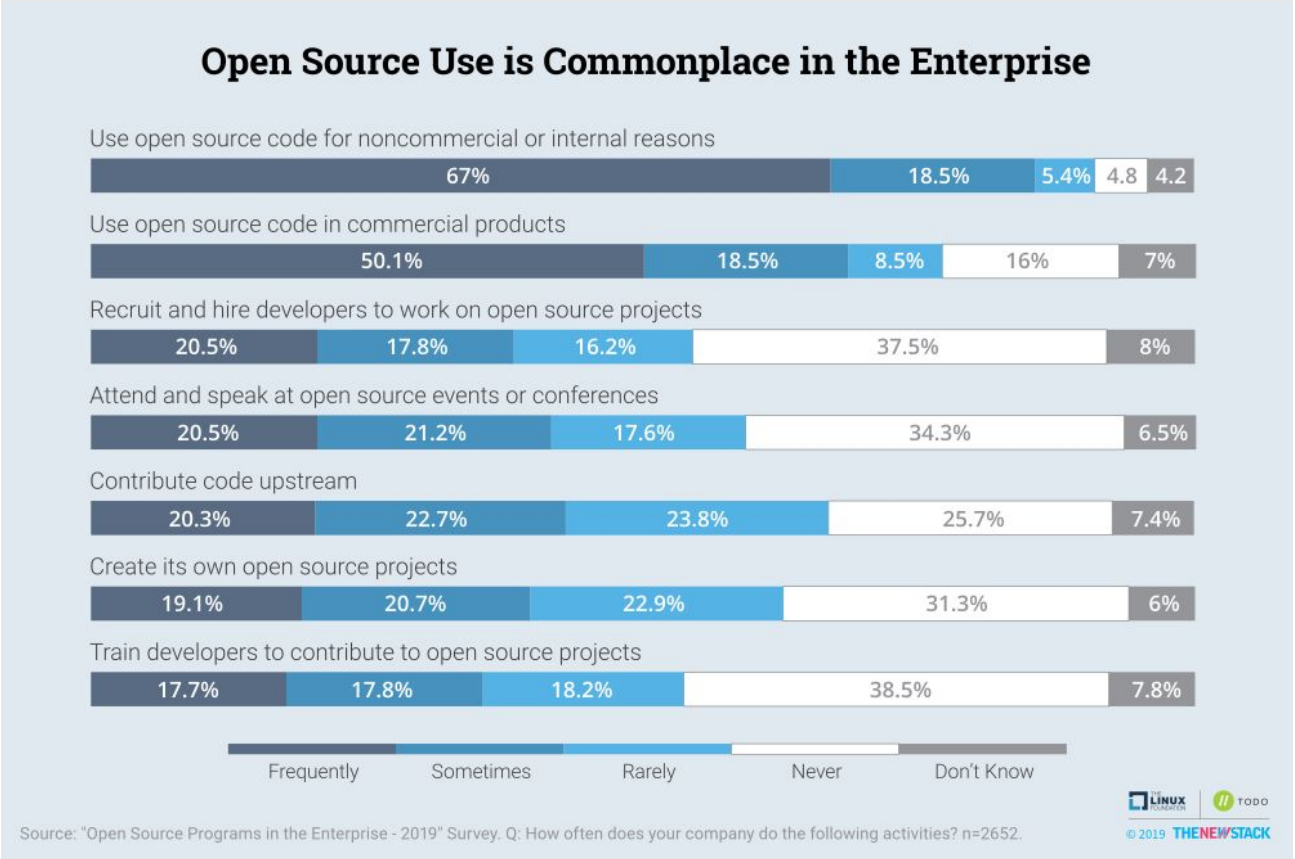


図 2 Talk Openly, Develop Openly Group の調査報告結果 [3]

Björn Lundell らによって行われたスウェーデンの企業における FLOSS の利用率調査 [4] や Øyvind Hauge らによって行われたノルウェーのソフトウェア業界における FLOSS の採用状況調査 [5]，Eugenio Capra らによって行われた FLOSS における企業の役割や影響の調査 [6]，Tetsuo Noda らによって行われた日本の地方自治体における FLOSS 活用政策と地域産業振興政策の課題調査 [7]，Katja Henttonen によって行われたインドにおける FLOSS ビジネスに関する課題調査 [8]，Tetsuo Noda らによって行われた東アジア諸国における FLOSS 政策現状と課題調査 [9]，Androklis Mavridis らによって行われたギリシャにおける FLOSS 採用の実現可用性に関する調査 [10]，Gustavo Pinto らによって行われたブラジルの FLOSS 開発者における年齢層や参加動機の調査 [11]，Davide Tosi らによって行われたイタリアのロンバルディア州における FLOSS の導入状況に関する調査 [12]，Gustavo Pinto らによって行われたギリシャの地方自治体における FLOSS の普及状況と使用状況に関する調査 [13] など，民間企業だけでなく行政府機関でも FLOSS が活用されている。

Tanya Percy らによって行われたアフリカの大学で公開教育資源を利用する際の課題調査 [14] や G. R. Gangadharan らによって行われた新興市場の教育環境で FLOSS の採用を促進する要因と採用を阻害する要因の調査 [15]，Björn Lundell らによって行われたスウェーデンの教育における FLOSS の期待度調査 [16]，Sergio Raúl Montes León らによって行われたエクアドルで FLOSS の大学院修士課程を創設する際の事前調査 [17]，Alexandru Coman らによって行われたルーマニアの高等教育で FLOSS を活用する際の課題調査 [18]，Debora Maria Coelho Nascimento らによって行われた FLOSS の開発を通じてソフトウェア工学を学ぶ取り組み [19]，Becka Morgan らによって行われた FLOSS の教育カリキュラム開発に向けた取り組み [20] など，教育現場でも FLOSS が活用されている。

上記で述べた通り，様々な分野で FLOSS が活用されている。その一方，Heartbleed などの深刻な脆弱性によって社会全体に影響を及ぼすようなセキュリティインシデントも発生している。Heartbleed は，2014 年 04 月 01 日 暗号通信ライブラリの OpenSSL Ver.1.0.1 で発見された脆弱性であり，共通脆弱性識別子識別番号 CVE-2014-0160 [21] が割り当てられている。この脆弱性は，以下のような甚大な被害をもたらした。

- インターネット上の Web サーバの約 17 %（約 50 万台）が影響を受ける [22]
- インターネットユーザの 6% が個人情報を不正取得される [23]
- アクセスランキング上位 100 の Web サーバ 44 台が影響を受ける [24]

後の調査によって，OpenSSL は，世界中の暗号通信を支える重要なソフトウェアにもかかわらず，わずか 3 人の開発者によって開発と保守が行われていたことが発覚 [25] 。その結果，Heartbleed は 2011年 12 月 31 日にバグが混入してから 2012 年 03 月 14 日に発覚するまで約 822 日間も放置され，社会全体に甚大な影響を及ぼすこととなった。このように，重要なソフトウェアであるにもかかわらず「FLOSS = 無料」という社会全体の意識によって，FLOSS 開発者への負担が増大している。重要な FLOSS には，十分な人的，経済的資源が行き渡るようなエコシステムの構築が必要である。本研究の目的は FLOSS が抱える問題を解決し，FLOSS を中心とした理想的なエコシステムの実現を目指すことである。そのための予備調査として，FLOSS が抱える問題の原因，課題を明らかにし，その課題に対するアプローチを提案する。続く 2 章では，FLOSS の概要として FLOSS の定義や代表的な FLOSS の紹介，FLOSS のステークホルダ，FLOSS ライセンスについて記述する。3 章では，FLOSS が抱える 3 つの問題を調査し，その原因や課題，対策について述べる。4 章では，調査した組織構造分析手法について記述する。5 章では，既存研究を参考に行った FLOSS プロジェクトの分析手順と，その結果について記述する。6 章では，既存研究調査や予備実験に対する考察と本報告書のまとめを記述する。

2. FLOSS の概要

本章では，FLOSS の概要について述べる。まず初めに，フリーソフトウェアとオープンソースソフトウェアの観点から FLOSS を定義する。次に，代表的な FLOSS を 8 つのカテゴリ別に紹介する。次に，FLOSS に関わるステークホルダを様々な観点から紹介する。最後に，FLOSS のライセンスについてコピーライトの観点から説明する。

2.1. FLOSS の定義

FLOSS は，フリーソフトウェアとオープンソースソフトウェアの双方を包括する用語である [26] 。フリーソフトウェアとオープンソースソフトウェアは同一視されやすいが，それぞれに異なった定義が存在している。本節では，まず初めにフリーソフトウェアとオープンソースソフトウェア，それぞれの定義について述べる。次に，FLOSS という用語が使用される切っ掛けとなったフリーソフトウェア運動について述べる。最後に，本報告書における FLOSS の定義について述べる。

フリーソフトウェアの定義は，Free Software Foundation[3] (以降，FSF と称す) の定義である The Free Software Definition (以降，FSD と称す) [27] と Debian Project[4] の定義である The Debian Free Software Guidelines (以降，DFSG と称す) [28]が存在する。FSD は，以下に示す 4 つの項目から構成されており，GNU の公式ウェブサイトで公開されている [27] 。1986 年 02 月に公開された初版 [29] から本報告書執筆時点まで，26 回の改定が行われている [27] 。

[3] Richard M. Stallman によって設立された非営利団体
[4] Linux ディストリビューションを開発するコミュニティ

1) 第零の自由: どんな目的に対しても，プログラムを望むままに実行する自由
2) 第一の自由: プログラムがどのように動作しているか研究し，必要に応じて改造する自由
3) 第二の自由: ほかの人を助けられるよう，コピーを再配布する自由
4) 第三の自由: 改変した版を他に配布する自由

DFSG は，以下に示す 10 つの項目から構成されており，Debian Project の公式ウェブサイトで公開されている [28] 。1997 年 07 月 05 日に初版である Version 1.0 [30] が公開され，2004 年 04 月 26 日に改定版である Version 1.1 [28] が公開されている。本報告書執筆時点では，Version 1.1 が最新版となっている。

1) 自由な再配布
2) ソースコード
3) 派生ソフトウェア
4) 原作者によるソースコードの整合性維持
5) すべての個人，団体の平等
6) 目標分野の平等
7) ライセンスの配布
8) ライセンスは Debian に限定されない
9) ライセンスは他のソフトウェアを侵害しない
10) フリーなライセンスの例

オープンソースソフトウェアの定義として Open Source Definition (以降，OSD と称す) [31] が挙げられる。OSD は，Open Source Initiative[5] (以降，OSI と称す) によって定義されており，以下に示す 10 項目から構成されている。OSD は，2007 年 03 月 22 日に最新版である Version 1.9 が公開されている。また，OSD は上記の DFSG を基に作成されているため，酷似している項目が多々存在する。

1) 自由な再頒布
2) ソースコード
3) 派生物
4) 原著作者のソースコードとの区別
5) 特定人物・集団に対する差別の禁止
6) 使用分野に対する差別の禁止
7) ライセンスの権利配分
8) ライセンスは特定製品に限定してはならない
9) ライセンスは他のソフトウェアを制限してはならない
10) ライセンスは技術中立

フリーソフトウェア運動は，ユーザが使うソフトウェアをユーザがコントロールすることを目的とした活動であり，FSF の創設者である Richard M. Stallman らによって行われている [32] 。Richard M. Stallman はフリーソフトウェア運動の一環として，フリーソフトウェアのフリーは「無償」ではなく「自由」であることを主張している。また，「無償」と「自由」を区別し「自由」であることを強調するために Free and Open Source Software の略称である FOSS ではなく，Free/Libre and Open Source Software の略称である FLOSS の使用を推奨している [26] 。

[5] Bruce Perens と Eric S. Raymond によって設立された権利擁護団体

Google トレンド[6] を用いて過去 5 年間の検索推移を比較調査した結果，フリーソフトウェアが最も検索されており，次いで Open Source Software の略称である OSS が検索されていることがわかった (図 3) 。認知度の側面から，用語としてフリーソフトウェアや OSS を用いることが適切であると考えられる。しかし，本報告書では Richard M. Stallman が主張する以下の 2 点を考慮し，最も包括的な用語である FLOSS を用いることとする。

- 「無償」と「自由」の区別
- フリーソフトウェアとオープンソースソフトウェアの区別

図 3 Google トレンドを用いた比較調査の結果

また，本報告書においては以下の条件を満たすソフトウェアを FLOSS とする。

- 改良，研究，実行，配布，複写，変更が可能なソフトウェア
- ソースコードがオープンアクセスになっているソフトウェア
- 適切なライセンスが割り当てられているソフトウェア

2.2. 代表的な FLOSS

現在，インターネット上にはスクリプトからオペレーティングシステムまで，様々な FLOSS が公開されている。本節では，その中から 8 つのカテゴリー (① ウェブサーバ，② ウェブブラウザ，③ オペレーティングシステム，④ セキュリティ，⑤ データベース，⑥ フレームワーク，⑦ プログラミング言語，⑧ 仮想化・クラウド) における代表的な FLOSS を紹介する。

① ウェブサーバ

Apache HTTP Server は，Apache Software Foundation[7] が中心となって開発が行われている FLOSS のウェブサーバであり，NCSA[8] に所属していた Robert McCool らによって開発されていた NCSA HTTPd が基になっている。W3Techs[9] で公開されているウェブサーバの統計情報 [33] によると，Apache HTTP Server の利用率は約 36% であり，インターネットのインフラ基盤として重要な役割を担っている。

[6] Google が運営しているキーワードの検索推移を比較可能なウェブサービス
[7] Apache HTTP Server を中心に FLOSS の開発や支援を行う非営利団体
[8] National Center for Supercomputing Applications の略称
[9] Q-Success によって収集されている利用統計情報が公開されているウェブサービス

Nginx は，NGINX, Inc.[10] が中心となって 2004 年 10 月 04 日から開発が行われている FLOSS のウェブサーバである。W3Techs で公開されているウェブサーバの統計情報 [33] によると Nginx の利用率は約 32% であり，上記の Apache HTTP Server と共にインターネットのインフラ基盤として重要な役割を担っている。

② ウェブブラウザ

Chromium は，Google が中心となって 2008 年 09 月 02 日から開発が行われている FLOSS のウェブブラウザであり，HTML レンダリングエンジンとして Blink を採用している [34] 。同じく Google が開発を行っているウェブブラウザとして Chrome が挙げられる。Chrome は，Chromium に自動更新機能やユーザ拡張機能などの独自機能が実装され，プロプライエタリソフトウェアとして配布されているウェブブラウザである [35] 。StatCounter Global Stats[11] で公開されているウェブブラウザの統計情報 [36] によると，Chromium の利用率は圏外である。しかし，Chrome の利用率は約 66% であり，ウェブの発展に大きく貢献している。

Firefox は，Mozilla Foundation[12] と Mozilla Corporation[13] が中心となって開発が行われている FLOSS のウェブブラウザであり，HTML レンダリングエンジンとして Gecko を採用している [37] 。また，Firefox は Netscape Communications Corporation によって開発が行われていたウェブブラウザである Netscape Navigator が基となっている。Netscape Navigator は， 1998 年 03 月 31 日にソースコードがオープンアクセスになっており，2002 年 09 月 23 日に Firefox としてリリースされている。StatCounter Global Stats で公開されているウェブブラウザの統計情報 [36] によると，Firefox の利用率は約 4% であり，上記の Chrome と比較するとユーザは少ない。しかし，技術の差別化など競争市場に大きく貢献している。

③ オペレーティングシステム

Linux Kernel は，Linux Foundation[14] が中心となって開発が行われている FLOSS のオペレーティングシステムのカーネルである。1991年 08 月 25 日，当時 21 歳でフィンランドのヘルシンキ大学に在籍していた Linus Torvalds によって Linux の初版である Ver.0.0.1 が公開され，今日まで開発が行われている [38] 。後述する Ubuntu や Android などのエンドユーザ向けオペレーティングシステムから交通機関や金融機関などのミッションクリティカルな情報システムまで幅広い分野で活用されている。

Debian は，Debian Project が中心となって開発が行われている FLOSS の Linux ディストリビューションであり，1993 年 08 月 16 日に当時，パデュー大学に在籍した Ian Ashley Murdock によって Debian の初版が公開された。Debian は，社会契約の中で DFSG を満たさないソフトウェアは公式として提供はしないと宣言しており，基本的にはフリーソフトウェアのみで構成されている Linux ディストリビューションである [28] 。W3Techs で公開されているオペレーティングシステムの統計情報 [39] によると，Debian の利用率は約 18% である。

[10] マイクロサービス向けの高性能アプリケーションを提供する企業
[11] StatCounter によって収集されている利用統計情報が公開されているウェブサービス
[12] Firefox を中心に FLOSS の開発や支援を行う非営利団体
[13] Firefox を中心に FLOSS の開発や支援を行う Mozilla Foundation の完全子会社
[14] Linux Kernel を中心に FLOSS の開発や支援を行う非営利団体

Ubuntu は，Canonical Ltd.[15] が中心となって 2004 年 10 月 20 日から開発が行われている FLOSS の Linux ディストリビューションである。上記の Debian から派生した Linux ディストリビューションであり，Debian 系の Linux ディストリビューションとして位置づけられていた。現在は，Ubuntu の社会的影響力が大きくなったことや Ubuntu から派生した Linux ディストリビューションが増加したことなどの理由から Debian 系の Linux ディストリビューションとしてではなく，Ubuntu 系の Linux ディストリビューションとして紹介されることも多くなった。また，上記でも述べた通り Debian は基本的にフリーソフトウェアのみで構成されている。しかし，Ubuntu はユーザフレンドリーであることを優先しているため，プロプライエタリなソフトウェアも公式の一部として提供する点が Debian と大きく異なる。W3Techs で公開されているオペレーティングシステムの統計情報 [39] によると，Ubuntu の利用率は約 46% であり，最もエンドユーザに使用されている Linux ディストリビューションの 1 つである。

CentOS は，CentOS Project が中心となって 2004 年 05 月 14 日から開発が行われている FLOSS の Linux ディストリビューションである。Red Hat Enterprise Linux から商標や商用パッケージなどを完全に除去した Linux ディストリビューションを目指しており，Red Hat Enterprise Linux と同様に 10 年間のセキュリティパッチサポート期間を設定するなど，長期的な情報システムの運用に適している。開発当初は，Red Hat とは完全に独立したコミュニティによって開発が行われていたが，2014 年 01 月 07 日から Red Hat のメンバーをコミュニティの中心メンバーとして受け入れるという形で Red Hat と提携している [40] 。W3Techs で公開されているオペレーティングシステムの統計情報 [39] によると，CentOS の利用率は約 19% であり，デスクトップ向けの Linux ディストリビューションとしては上記の Ubuntu に及ばない。しかし，サーバ向けの Linux ディストリビューションとしては高い利用率を誇る。

Android は，Google が中心となって 2008 年 09 月 23 日から開発が行われている FLOSS の モバイルデバイス向けオペレーティングシステムである。オペレーティングシステムのカーネルとして Linux Kernel を採用している。StatCounter Global Stats で公開されているモバイルデバイス向けオペレーティングシステムの統計情報 [41] によると，Android の利用率は約 74% であり，非常に高い利用率を誇っている。

④ セキュリティ

OpenSSH は，OpenBSD Foundation[16] が中心となって 1999 年 12 月 01 日から開発が行われている FLOSS の遠隔通信ソフトウェアである。SSH プロトコルは，暗号通信を用いてネットワークに接続された機器を遠隔操作するための通信手段プロトコルであり，TCP/IP プロトコルスイートのアプリケーション層に位置づけられる。OpenSSH は，OpenBSD だけでなく Ubuntu や CentOS など，様々な Linux ディストリビューションで標準インストールされており，SSH プロトコルの実装ソフトウェアとしてはデファクトスタンダードである。

OpenSSL は，OpenSSL Software Foundation[17] が中心となって開発が行われている FLOSS の暗号通信ライブラリであり，Eric A. Young と Tim J. Hudson によって開発されていた暗号通信ライブラリ SSLeay が基となっており，OpenSSL としては，1998 年 12 月 23 日から開発が行われている [42-43] 。また，Ubuntu や CentOS など，様々な Linux ディストリビューションで標準インストールされており，TLS/SSL プロトコル実装ソフトウェアとしてはデファクトスタンダードである。

[15] Ubuntu のサポートやトレーニングプログラムを提供する企業
[16] OpenBSD を中心に FLOSS の開発や支援を行う非営利団体
[17] OpenSSL を中心に FLOSS の開発や支援を行う非営利団体

OpenVPN は，OpenVPN Technologies, Inc.[18] が中心となって開発が行われている FLOSS の Virtual Private Network (以降，VPN と称す) ソフトウェアであり，2001 年 05 月 13 日に James Yonan によって初版が公開された。また，Ubuntu や CentOS など，様々な Linux ディストリビューションに対応しており，VPN ソフトウェアとしては世界中で使用されている。

⑤ データベース

MariaDB は，MariaDB Foundation[19] が中心となって 2009 年 10 月 29 日から開発が行われている FLOSS の RDBMS である。また，後述する MySQL の開発者であった Ulf M. Widenius によって MySQL の派生データベースシステムとして開発されたのが始まりである。DB-Engines[20] で公開されているデータベースシステムの統計情報 [44] によると，MariaDB の使用ランキングは第 12 位であり，Nokia や Red Hat，Samsung など様々な企業が採用している [45] 。

MongoDB は，MongoDB Inc.[21] が中心となって 2009 年 02 月 11 日 から開発が行われている FLOSS の NoSQL である。MongoDB Inc. は，インターネット広告会社である DoubleClick (現在は，Google の傘下) に所属していた Dwight Merriman と liot Horowitz によって設立された [46] 。DB-Engines で公開されているデータベースシステムの統計情報 [44] によると，MongoDB の使用ランキングは第 5 位であり，Adobe や Google，SEGA など様々な企業が採用している [47] 。

MySQL は，Oracle が中心となって開発が行われている FLOSS の RDBMS である。1995 年 05 月 23 日から開発されており，FLOSS のデータベースとしては非常に長い開発の歴史がある。DB-Engines で公開されているデータベースシステムの統計情報 [44] によると，MySQL の使用ランキングは第 2 位であり，GitHub や Booking.com，Pinterest など様々な企業が採用している [48] 。

PostgreSQL は，PostgreSQL Global Development Group が中心となって 1997 年 01 月 29 日 から開発が行われている FLOSS の RDBMS である。また，カリフォルニア大学バークレー校で開発されたデータベースシステムである Ingres から派生したデータベースシステムでもある [49] 。DB-Engines で公開されているデータベースシステムの統計情報 [44] によると，PostgreSQL の使用ランキングは第 4 位であり，上記の MySQL と同様に様々な企業で採用されている。

SQLite は，Hipp, Wyrick & Company, Inc が中心となって開発が行われている FLOSS の RDBMS である。SQLite の初版は D. Richard Hipp によって設計と開発が行われ，2000 年 08 月 17 日に公開された。その他のデータベースシステムと比較すると非常に軽量なため，デスクトップアプリケーションやモバイルアプリケーションに組み込まれ運用される場合が多い。DB-Engines で公開されているデータベースシステムの統計情報 [44] によると，SQLite の使用ランキングは第 9 位である。

⑥ フレームワーク

Angular は，Google が中心となって開発が行われている FLOSS のフロントエンドウェブアプリケーションフレームワークである。また，Angular は Version 1.X である AngularJS と Version 2.X である Angular が存在する。AngularJS では JavaScript を，Angular では TypeSctiprt を推奨プログラミング言語としている。AngularJS の初版は 2010 年 10 月 20 日に公開されており，フロントエンドウェブアプリケーションフレームワークとして，非常に長い開発の歴史が存在する。

[18] ネットワーク技術を提供する株式非公開企業
[19] MariaDB を中心に FLOSS の開発や支援を行う非営利団体
[20] データベースシステムの人気をランキング形式で表示するウェブサービス
[21] MongoDB のサポートやトレーニングプログラムを提供する企業

React は，Facebook が中心となって 2013 年 03 月 29 日から開発が行われている FLOSS のフロントエンドウェブアプリケーションフレームワークである。仮想 DOM や JSX などフロントエンドをサポートする様々な機能が搭載されている。また，ウェブ技術を用いてモバイルデバイス向けのネイティブアプリケーションを開発可能な React Native も提供されている。

Vue.js は，Evan You が中心となって 2014 年 02 月から開発が行われている FLOSS のフロントエンドウェブアプリケーションフレームワークである。Evan You は，上記の AngularJS の開発に携わっていた Google の開発者である。そのため，Vue.js の一部設計や機能は AngularJS から継承している。その一方で，双方向データバインディングやテンプレート機能など，フロントエンドの開発をサポートする様々な独自機能も搭載されている。また，上記の Angular と React は企業主導によって開発が行われているが，Vue.js はコミュニティ主導によって開発が行われている。

⑦ プログラミング言語

GNU Compiler Collection は，FSF が中心となって 1987年 05 月 23 日から開発が行われている FLOSS のコンパイラ群である。C/C++ や Objective-C，Fortran など多種多様なプログラミング言語に対応しており，開発や教育，研究などの幅広い分野で使用されている。特に，C/C++ のコンパイラとして用いられる場合が多く，C/C++ コンパイラのデファクトスタンダードとされている。

OpenJDK は，Oracle が中心となって 2007 年 05 月 08 日から開発が行われている FLOSS のJava 開発ツール群である。Java コンパイラである javac やドキュメンテーション生成ツールである javadoc，Java のデバッカーである jdb など，様々な開発ツールで構成されている。現在，Oracle がビルドした OpenJDK の他，Red Hat や Amazon など様々な企業がビルド，提供している OpenJDK が存在する。

Python は，Python Software Foundation[22] が中心となって 1990 年から開発が行われている FLOSS のインタープリタ型高水準汎用プログラミング言語である。オランダ出身のプログラマーである Guido van Rossum によって設計，開発が行われた。現在，Guido van Rossum はプログラマーから引退し，Python 開発の第一線から退いており，後任に関する情報は調査した範囲ではない [50] 。クラウドストレージサービスである Dropbox のフロントエンドとバックエンドに採用されるなど，多くの企業が採用している。また，記述方法が直感的であり学習曲線が緩やかであるなどの理由から，近年では文部科学省が高等学校の情報科教材に Python を採用するなど，教育現場で採用される場面も増加している [51] 。

Ruby は，鳥取県出身のプログラマーである，まつもとゆきひろが中心となって 1995 年から開発が行われている FLOSS のインタープリタ型高水準汎用プログラミング言語である。国際電気標準会議の国際規格に初めて認証された国産のプログラミング言語でもあり，GMOペパボ株式会社やクックパッド株式会社など国内企業の多くが採用している [52] 。

⑧ 仮想化・クラウド

Docker は，Solomon Hykes が中心となって 2013 年 03 月 13 日から開発が行われている FLOSS のコンテナ型仮想化ソフトウェアである。従来の仮想化技術であるホスト型仮想化やハイパー型仮想化と比較すると，ゲストオペレーティングシステムを展開することなくアプリケーションの仮想化が可能であり，オーバーヘッドの削減によるリソースの削減や動作の高速化などを実現している。また，Adobe や AT&T，Netflix など様々な企業が採用している [53] 。

[22] Python を中心に FLOSS の開発や支援を行う非営利団体

OpenStack は，OpenStack Foundation[23] が中心となって 2010 年 10 月 21 日から開発が行われている FLOSS のクラウド基盤ソフトウェアである。また，Rackspace Technology[24] が開発していたクラウドコンピューティング環境である Swift と NASA が開発していたクラウド基盤ソフトウェアである Nova が基になっている。AT&T や Adobe，CERN などの様々な企業が採用している [54] 。

VirtualBox は，Oracle が中心となって 2007 年 01 月 15 日から開発が行われている FLOSS の仮想化ソフトウェアであり，仮想化ソフトウェアとして非常に長い歴史が存在する。仮想化の種類としてホスト型仮想化に分類され，GUI による直感的な操作が可能であるなどの特徴を持つ。

2.3. FLOSS のステークホルダ

FLOSS の参加者は「エンジニアとエンドユーザ」や「コミッタとコントリビュータ」などと分類されることが多い。しかし，企業や政府機関のように役職が定まっているわけではないため，FLOSS の参加者を明確に分類することは困難である。また，近年になって営利目的で FLOSS を活用する企業や FLOSS の開発支援を目的とする非営利団体，これらの団体に所属しながら FLOSS を開発するフルタイムエンジニアなどが登場したことによって，FLOSS のステークホルダは複雑化している。本節では，複雑化する FLOSS のステークホルダを様々な視点から可能な限り分類する。

① エンジニアとエンドユーザ

FLOSS のステークホルダは，FLOSS を開発するエンジニアと FLOSS を利用するエンドユーザに大きく分類することができる。また，エンジニアとエンドユーザは FLOSS を中心とした以下のサイクルを繰り返すことによって品質の高い FLOSS を継続的に開発することができる (図 4) 。

(1) エンジニアが開発し，公開している FLOSS をエンドユーザが利用する
(2) FLOSS を利用したエンドユーザが不具合や改善案をエンジニアに伝える
(3) エンドユーザが報告してくれた不具合や改善案を基に FLOSS を改良し，再び公開する
(4) (2) 〜 (3) を繰り返す

[23] OpenStack を中心に FLOSS の開発や支援を行う非営利団体
[24] クラウドコンピューティングに関する技術を提供する企業

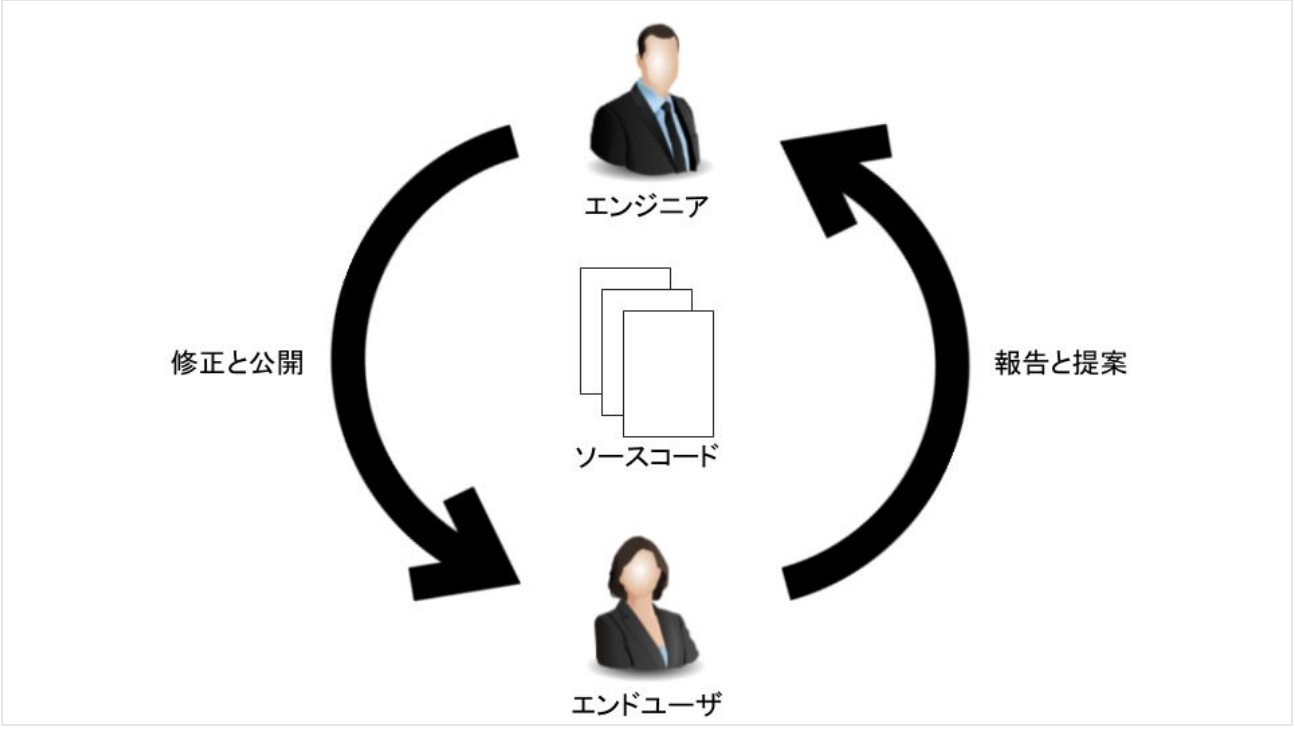


図 4 エンジニアとエンドユーザの関係性

② コミッタとコントリビュータ

既存研究 [55] より，FLOSS の参加者と貢献度の関係はパレートの法則に従っていることがわかっており，以下の (1) と (2) に分類することができる。(1) に属する参加者は定期的に FLOSS へ貢献する参加者であり，コミッタやコアメンバなどと呼ばれている。また，(2) に属する参加者は不定期的に FLOSS へ貢献する参加者であり，コントリビュータやペリフェリーメンバなどと呼ばれている。(1) と (2) の関係性をオニオンモデルを用いて表現 (図 5) することが多く，(1) と (2) の関係性を分析する研究も活発に行われている [56-61] 。

(1) 20% の参加者によって 80% の貢献が行われている
(2) 80% の参加者によって 20% の貢献が行われている

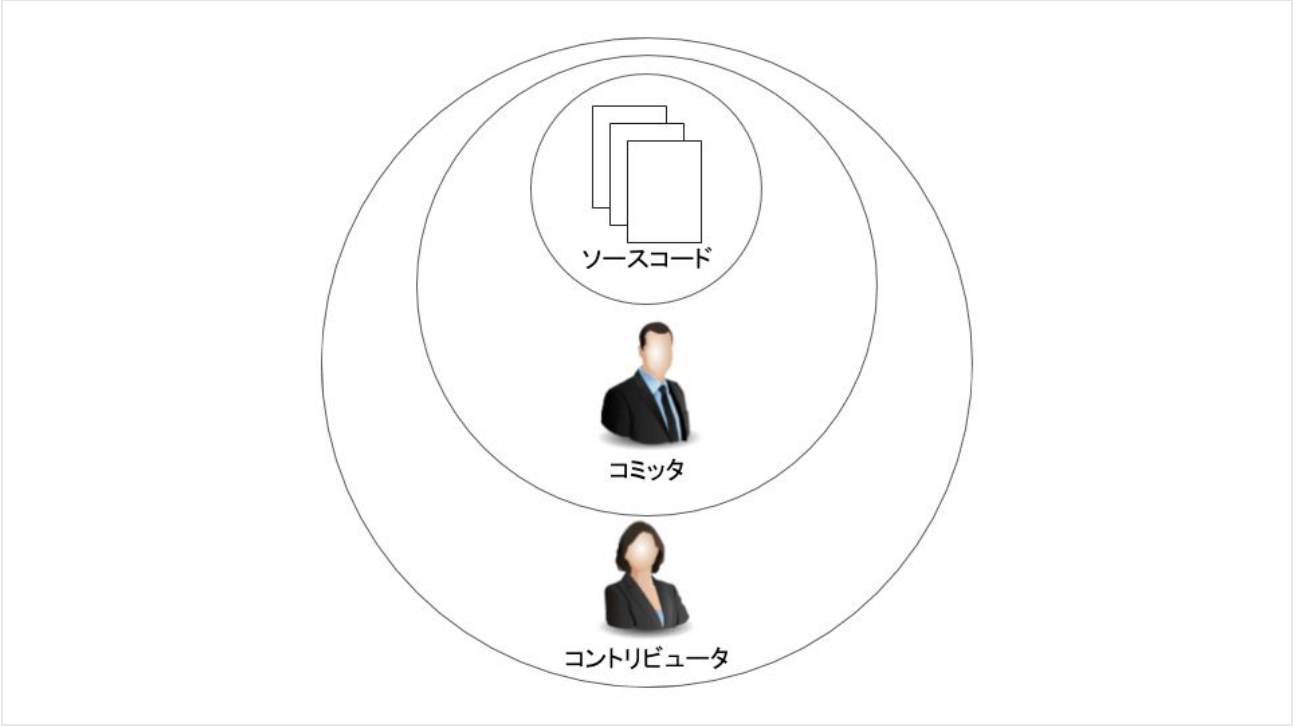


図 5 コミッタとコントリビュータの関係性

③ 企業と財団

近年，FLOSS を用いてビジネスを行う企業が FLOSS のステークホルダとして登場している。代表的な企業として Amazon や Google，Red Hat，Canonical Ltd. などが挙げられるが，各企業によってビジネスモデルは様々である。Amazon や Google などは，FLOSS を用いたサービスをエンドユーザに提供することで収益を生み出すビジネスモデルを採用しており，Red Hat や Canonical Ltd. などは，自身が開発する FLOSS のサポートやトレーニングプログラムを提供することで収益を生み出すビジネスモデルを採用している。上記のような企業やビジネスモデルが登場したことによって，FLOSS が企業や市場に及ぼす影響力や新しいビジネスモデルの提案など，FLOSS をビジネスの観点から分析する研究も活発に行われている [62-70] 。

また，FLOSS の開発を支援する財団も FLOSS のステークホルダとして登場している。代表的な財団として FSF や Linux Foundation，Mozilla Foundation などが挙げられる。財団の主な活動として，FLOSS に対して財団に所属する開発者を派遣することで人材的，技術的な支援や FLOSS に対して財団が保有する基金を寄付することで経済的な支援などが挙げられる。これらの活動を通して，財団は FLOSS の発展と維持に大きく貢献しており，FLOSS のステークホルダとして重要な存在である。また，FLOSS における財団の影響力や役割を分析した研究も活発に行われている [71-75] 。

④ ボランティアとフルタイム

従来，FLOSS の開発はボランティアによって行われていた。しかし，上記で述べた企業や財団によって雇用されたフルタイムエンジニアが FLOSS の開発に携わるケースが近年増加している (図 6) 。そのため，ボランティアエンジニアとフルタイムエンジニアの行動パターンの違いやコミュニティに及ぼす影響などを分析する研究が活発に行われている [76-79] 。

図 6 フルタイムエンジニア

2.4. FLOSS ライセンス

FLOSS ライセンスは，FLOSS の改良，研究，実行，配布，複写，変更ができる範囲を定めたものであり，OSI が認証した FLOSS ライセンス [80] や FSF が認証した FLOSS ライセンス [81] など様々な FLOSS ライセンスが存在する。また，FLOSS ライセンスに関する研究も活発に行われている [82-92] 。本節では，コピーレフトの適用状況に応じて，① コピーレフト型ライセンス，② 準コピーレフト型ライセンス，③ 非コピーレフト型ライセンスの 3 カテゴリーに分類 [93] し，各カテゴリーの代表的な FLOSS ライセンスについて記述する。

① コピーレフト型ライセンス

GNU General Public License Version 3 (以降，GPLv3 と称す) ライセンスは，FSF によって作成されたコピーレフト型ライセンスである [94] 。GPLv3 が適応されている FLOSS を改変して再配布する場合，または GPLv3 が適応されている FLOSS を組み込んだソフトウェアを再配布する場合は，再配布する二次的著作物に対して GPLv3 を適応する義務が生じる。しかし，ネットワーク経由で GPLv3 が適応されているプログラムと協調動作するソフトウェアに関しては GPLv3 を適応する義務は生じない。

GNU Affero General Public License Version 3 (以降，AGPLv3 と称す) ライセンスは，FSF と Affero, Inc. が共同で作成したコピーレフト型ライセンスである [95] 。上記の GPLv3 同様に，AGPLv3 が適応されている FLOSS を改変して再配布する場合，または AGPLv3 が適応されている FLOSS を組み込んだソフトウェアを再配布する場合は，再配布する二次的著作物に対して AGPLv3 を適応する義務が生じる。しかし，上記の GPLv3 とは異なり，ネットワーク経由で GPLv3 が適応されているプログラムと協調動作するソフトウェアに関しても AGPLv3 を適応する義務は生じる。

European Union Public License Version 1.2 (以降，EUPL-1.2 と称す) ライセンスは，European Union (以降，EU と称す) によって作成されたコピーレフト型ライセンスである [96] 。上記の GPLv3 同様に，EUPL-1.2 が適応されている FLOSS を改変して再配布する場合，または EUPL-1.2 が適応されている FLOSS を組み込んだソフトウェアを再配布する場合は，再配布する二次的著作物に対して EUPL-1.2 を適応する義務が生じる。また，上記の GPLv3 と異なる点として，米国の法律用語を欧州の法律用語に変換した点や EU の公用語 22 個に対応している点などが挙げられる。

② 準コピーレフト型ライセンス

GNU Lesser General Public License Version 3 (以降，LGPLv3 と称す) ライセンスは，FSF によって作成された準コピーレフト型ライセンスである [97] 。上記の GPLv3 同様に，LGPLv3 が適応されている FLOSS を改変して再配布する場合は，再配布する二次的著作物に対して LGPLv3 を適応する義務が生じる。しかし，LGPLv3 が適応されている FLOSS を組み込んだソフトウェアを再配布する場合は，LGPLv3 を適用する義務は生じない。

Mozilla Public License Version 2.0 (以降，MPL 2.0 と称す) ライセンスは，Mozilla Foundation によって作成された準コピーレフト型ライセンスである [98] 。上記の LGPLv3 同様に，MPL 2.0 が適応されている FLOSS を改変して再配布する場合は，再配布する二次的著作物に対して MPL 2.0 を適応する義務が生じる。また，MPL 2.0 が適応されている FLOSS を組み込んだソフトウェアを再配布する場合は，MPL 2.0 を適用する義務は生じない。

③ 非コピーレフト型ライセンス

Apache License 2.0 は，Apache Software Foundation によって作成された非コピーレフト型ライセンスである [99] 。コピーレフト型ライセンスや準コピーレフト型ライセンスと異なり，Apache License 2.0 が適応されている FLOSS を改変して再配布する場合に，再配布する二次的著作物に対して Apache License 2.0 を適応する義務が生じない。また，準コピーレフト型ライセンスと同様に Apache License 2.0 が適応されている FLOSS を組み込んだソフトウェアを再配布する場合は，再配布する二次的著作物に対して Apache License 2.0 を適応する義務が生じない。

BSD License は，カリフォルニア大学バークレー校によって作成された非コピーレフト型ライセンスである [100] 。コピーレフト型ライセンスや準コピーレフト型ライセンスと異なり，BSD License が適応されている FLOSS を改変して再配布する場合に，再配布する二次的著作物に対して BSD License を適応する義務が生じない。また，準コピーレフト型ライセンスと同様に BSD License が適応されている FLOSS を組み込んだソフトウェアを再配布する場合は，再配布する二次的著作物に対して BSD License を適応する義務が生じない。

MIT License は，マサチューセッツ工科大学で作成された非コピーレフト型ライセンスである [101] 。コピーレフト型ライセンスや準コピーレフト型ライセンスと異なり，MIT License が適応されている FLOSS を改変して再配布する場合，再配布する二次的著作物に対して MIT License を適応する義務が生じない。また，準コピーレフト型ライセンスと同様に MIT License が適応されている FLOSS を組み込んだソフトウェアを再配布する場合は，再配布する二次的著作物に対して MIT License を適応する義務が生じない。

3. FLOSS の既存研究調査

FLOSS の開発過程で発生する開発履歴をマイニングすることでソフトウェア工学的知見を得ようとする研究は活発に行われている。しかし，社会における FLOSS の役割や FLOSS を取り巻くステークホルダが変化したことによって，FLOSS が抱える問題も変化している。時代における問題を適切に把握することは重要である。そこで，本章では，まず初めに FLOSS に関する既存研究の収集手順について記述する。次に，収集した既存のサーベイ論文と FLOSS の活用事例報告書から問題を抽出する過程について記述する。次に，抽出した問題を小問題に細分化し，各小問題に該当する論文を分類する手順について記述する。最後に，論文を分類する過程で得られた軸を基にした FLOSS に関する既存研究の分析手順と結果について記述する。

3.1. 既存研究の収集

FLOSS に関する国際会議として，International Federation for Information Processing (以降，IFIP と称す) が開催している IFIP International Conference on Open Source Systems が挙げられる。IFIP International Conference on Open Source Systems は，FLOSS に関する ① ～ ② の範囲をカバーしており，16 回の開催 (2006 年 ～ 2020 年) でフルペーパーとショートペーパー合わせて 409 件の論文を採録している (図 7) 。

① ソフトウェアエンジニアリングの視点

- FLOSS のアーキテクチャ，構成とリリース管理，環境
- FLOSS の品質とセキュリティのテストと保証
- FLOSS プロジェクトリポジトリのマイニングと分析
- 従来の開発のための FLOSS からの教訓
- FLOSS と標準

② FLOSS 導入に関する研究

- FLOSS 導入，移行モデル，成功と失敗のケーススタディ
- 公共部門 (政府，教育，健康など) と「二次的な」ソフトウェア部門 (自動車，通信，医療機器など) におけるOSSの役割
- FLOSS 互換性のある IT ガバナンスアーキテクチャ
- FLOSS 開発のオフショアソーシング
- FLOSS のアプリケーションカタログ (機能性，プラットフォーム，サポートプロバイダ，トレーニングニーズ)

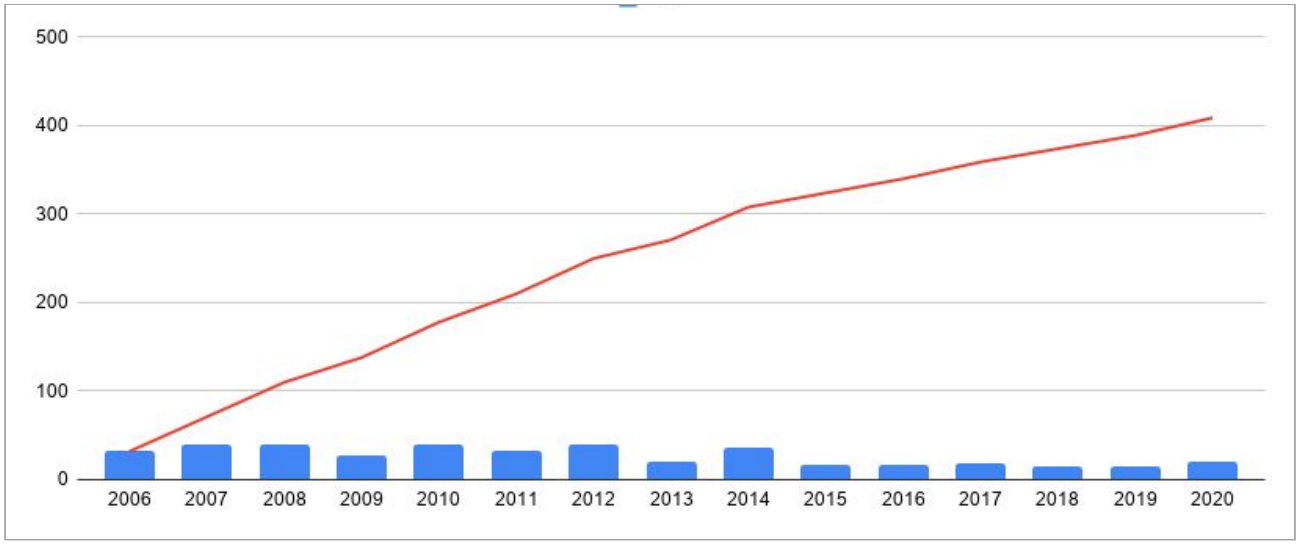
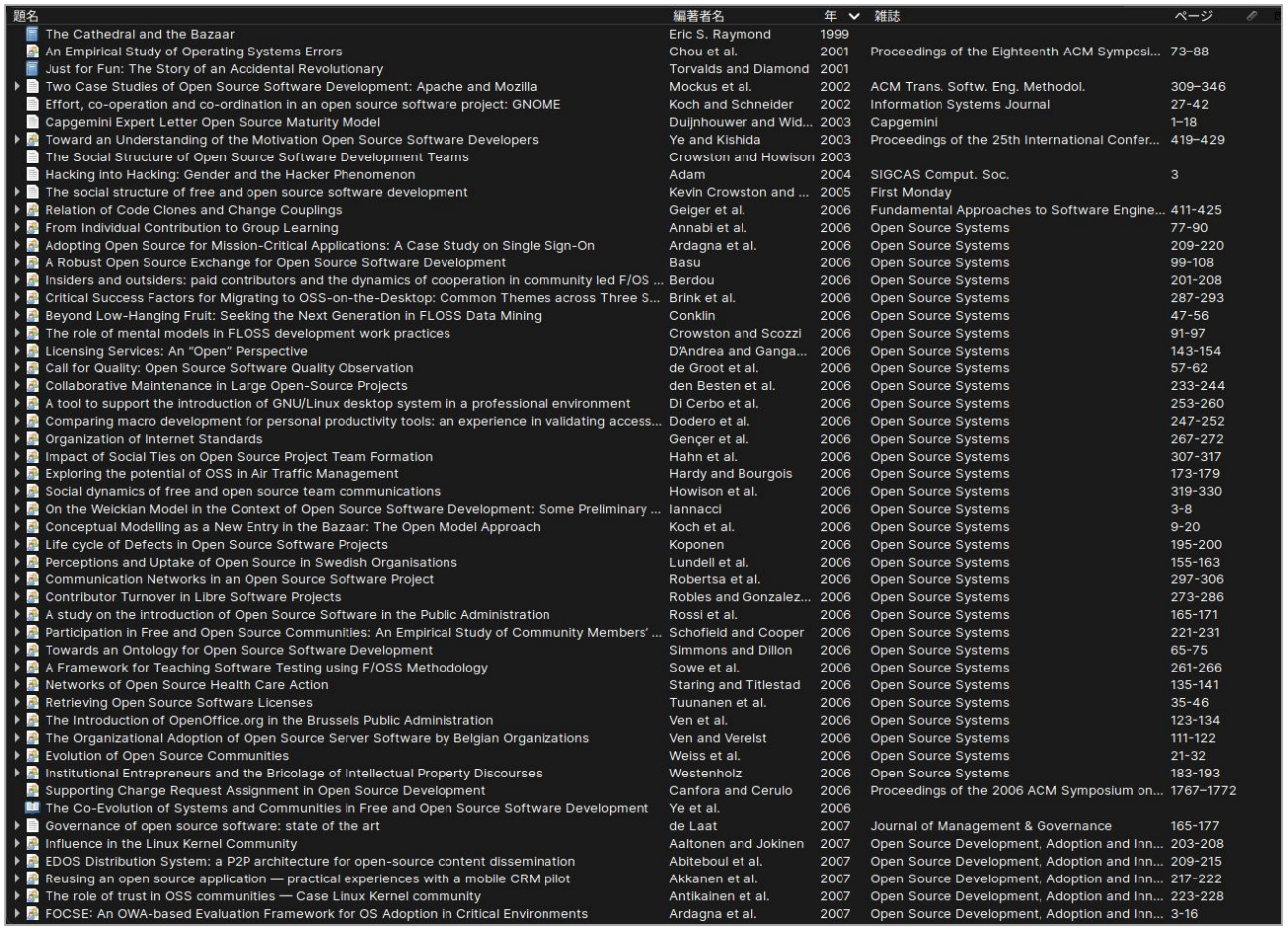


図 7 論文の各年と累積採録数

IFIP International Conference on Open Source Systems で採録されている論文で引用されている論文を更に加える。引用されている論文の中には，FLOSS との関係が薄い論文も含まれる。そのため，引用されている論文のタイトルとアブストラクトに “Open Source” や “OSS”，“FLOSS” などの単語が含まれる論文のみを抽出した。その結果，IFIP International Conference on Open Source Systems で採録されている論文 409 件と合わせて 508 件の FLOSS に関する論文を抽出することができた。最後に，論文のタイトルやアブストラクト，採録年度などのメタデータをデータベースとして保存した (図 8) 。



題名	編著者名	年	雑誌	ページ
The Cathedral and the Bazaar	Eric S. Raymond	1999		
An Empirical Study of Operating Systems Errors	Chou et al.	2001	Proceedings of the Eighteenth ACM Symposi...	73–88
Just for Fun: The Story of an Accidental Revolutionary	Torvalds and Diamond	2001		
Two Case Studies of Open Source Software Development: Apache and Mozilla	Mockus et al.	2002	ACM Trans. Softw. Eng. Methodol.	309–346
Effort, co-operation and co-ordination in an open source software project: GNOME	Koch and Schneider	2002	Information Systems Journal	27-42
Capgemini Expert Letter Open Source Maturity Model	Duijnhouwer and Wid...	2003	Capgemini	1–18
Toward an Understanding of the Motivation Open Source Software Developers	Ye and Kishida	2003	Proceedings of the 25th International Confer...	419–429
The Social Structure of Open Source Software Development Teams	Crowston and Howison	2003		
Hacking into Hacking: Gender and the Hacker Phenomenon	Adam	2004	SIGCAS Comput. Soc.	3
The social structure of free and open source software development	Kevin Crowston and ...	2005	First Monday	
Relation of Code Clones and Change Couplings	Geiger et al.	2006	Fundamental Approaches to Software Engine...	411-425
From Individual Contribution to Group Learning	Annabi et al.	2006	Open Source Systems	77-90
Adopting Open Source for Mission-Critical Applications: A Case Study on Single Sign-On	Ardagna et al.	2006	Open Source Systems	209-220
A Robust Open Source Exchange for Open Source Software Development	Basu	2006	Open Source Systems	99-108
Insiders and outsiders: paid contributors and the dynamics of cooperation in community led F/OS ...	Berdou	2006	Open Source Systems	201-208
Critical Success Factors for Migrating to OSS-on-the-Desktop: Common Themes across Three S...	Brink et al.	2006	Open Source Systems	287-293
Beyond Low-Hanging Fruit: Seeking the Next Generation in FLOSS Data Mining	Conklin	2006	Open Source Systems	47-56
The role of mental models in FLOSS development work practices	Crowston and Scozzi	2006	Open Source Systems	91-97
Licensing Services: An "Open" Perspective	D'Andrea and Ganga...	2006	Open Source Systems	143-154
Call for Quality: Open Source Software Quality Observation	de Groot et al.	2006	Open Source Systems	57-62
Collaborative Maintenance in Large Open-Source Projects	den Besten et al.	2006	Open Source Systems	233-244
A tool to support the introduction of GNU/Linux desktop system in a professional environment	Di Cerbo et al.	2006	Open Source Systems	253-260
Comparing macro development for personal productivity tools: an experience in validating access...	Dodero et al.	2006	Open Source Systems	247-252
Organization of Internet Standards	Gençer et al.	2006	Open Source Systems	267-272
Impact of Social Ties on Open Source Project Team Formation	Hahn et al.	2006	Open Source Systems	307-317
Exploring the potential of OSS in Air Traffic Management	Hardy and Bourgois	2006	Open Source Systems	173-179
Social dynamics of free and open source team communications	Howison et al.	2006	Open Source Systems	319-330
On the Weickian Model in the Context of Open Source Software Development: Some Preliminary ...	Iannacci	2006	Open Source Systems	3-8
Conceptual Modelling as a New Entry in the Bazaar: The Open Model Approach	Koch et al.	2006	Open Source Systems	9-20
Life cycle of Defects in Open Source Software Projects	Koponen	2006	Open Source Systems	195-200
Perceptions and Uptake of Open Source in Swedish Organisations	Lundell et al.	2006	Open Source Systems	155-163
Communication Networks in an Open Source Software Project	Robertsa et al.	2006	Open Source Systems	297-306
Contributor Turnover in Libre Software Projects	Robles and Gonzalez...	2006	Open Source Systems	273-286
A study on the introduction of Open Source Software in the Public Administration	Rossi et al.	2006	Open Source Systems	165-171
Participation in Free and Open Source Communities: An Empirical Study of Community Members' ...	Schofield and Cooper	2006	Open Source Systems	221-231
Towards an Ontology for Open Source Software Development	Simmons and Dillon	2006	Open Source Systems	65-75
A Framework for Teaching Software Testing using F/OSS Methodology	Sowe et al.	2006	Open Source Systems	261-266
Networks of Open Source Health Care Action	Staring and Titlestad	2006	Open Source Systems	135-141
Retrieving Open Source Software Licenses	Tuunanen et al.	2006	Open Source Systems	35-46
The Introduction of OpenOffice.org in the Brussels Public Administration	Ven et al.	2006	Open Source Systems	123-134
The Organizational Adoption of Open Source Server Software by Belgian Organizations	Ven and Verelst	2006	Open Source Systems	111-122
Evolution of Open Source Communities	Weiss et al.	2006	Open Source Systems	21-32
Institutional Entrepreneurs and the Bricolage of Intellectual Property Discourses	Westenholz	2006	Open Source Systems	183-193
Supporting Change Request Assignment in Open Source Development	Canfora and Cerulo	2006	Proceedings of the 2006 ACM Symposium on...	1767–1772
The Co-Evolution of Systems and Communities in Free and Open Source Software Development	Ye et al.	2006		
Governance of open source software: state of the art	de Laat	2007	Journal of Management & Governance	165-177
Influence in the Linux Kernel Community	Aaltonen and Jokinen	2007	Open Source Development, Adoption and Inn...	203-208
EDOS Distribution System: a P2P architecture for open-source content dissemination	Abiteboul et al.	2007	Open Source Development, Adoption and Inn...	209-215
Reusing an open source application — practical experiences with a mobile CRM pilot	Akkanen et al.	2007	Open Source Development, Adoption and Inn...	217-222
The role of trust in OSS communities — Case Linux Kernel community	Antikainen et al.	2007	Open Source Development, Adoption and Inn...	223-228
FOCSE: An OWA-based Evaluation Framework for OS Adoption in Critical Environments	Ardagna et al.	2007	Open Source Development, Adoption and Inn...	3-16

図 8 FLOSS の論文データベース

3.2. 問題の抽出

フリーソフトウェア運動が始まってから約 30年，FLOSS に関する国際会議が始まってから約 15 年が経過し，FLOSS を活用，開発，運用，保守していく過程で様々な問題が発生することが分かっている。その問題を様々な観点から比較調査している研究や，その問題の原因となっている要因を特定する研究，その問題を解決するために取り組むべき課題を明確にする研究など，FLOSS に関する多くの研究が行われている。しかし，FLOSS の開発をサポートするツールやサービスの進化や FLOSS に関わるステークホルダーの変化によって，FLOSS が抱える問題も日々変化している。FLOSS が抱える現在の問題を正確に把握することは，FLOSS の活用効率向上や開発効率向上に必要不可欠である。そこで， ① 既存のサーベイ論文と ② FLOSS の活用事例報告書を基に，FLOSS が抱えている問題を抽出する。

① 既存のサーベイ論文

FLOSS のサーベイ論文として，Fabio Mulazzani らの 6 年間 (2005 年から 2011 年) で投稿された FLOSS に関するプロシーディングの関係性を分析する調査研究 [102] や Magnus Bergquist らの FLOSS が商用利用されるまでの歴史を調査した研究 [103]，伊原彰紀らの FLOSS に関するステークホルダと既存研究を調査した研究 [104]，Francis Bordeleau らの 15 年間 (2005 年から 2019 年) で進化した FLOSS を研究，技術，ビジネスの観点から調査した研究 [105] などが挙げられる。

Fabio Mulazzani らは，既存のシステマティックマッピングとシステマティックレビューを参考に 6 年間 (2005 年から 2011 年) で FLOSS に関する国際会議に投稿された論文を分析している [102] 。論文の分析手法として，既存研究の論文をクロスシテーションでクラスタリングし，各クラスタを検査することで主要な研究テーマや研究テーマの進化，FLOSS　　研究の主要な創始者や合成者を検索している。この調査によって，FLOSS プロジェクトに関するソーシャルネットワーク分析，データマイニングのためのツール開発、メンテナンスプロセスを理解するためのソースコードの変更履歴を分析する研究，特に不具合修正プロセスに関する研究が活発に行われていることがわかった。

Magnus Bergquist らは，1980 年代，1990 年代，2000 年代における FLOSS の社会的な立ち位置を分析している [103] 。1980 年代，FSF の創設者である Richard M. Stallman らによって行われていたフリーソフトウェア運動が FLOSS の始まりであり，商用ソフトウェアに対する反発が主な原動力となっている。1990 年代，OSI の設立と FLOSS の定義が策定されたことによって，FLOSS とプロプライエタリソフトウェアが共存する環境が整っていった。2000 年代，FLOSS はミッションクリティカルなシステムに採用されるなど，プロプライエタリソフトウェアに置き換わりつつある。

伊原彰紀らは，FLOSS に関するステークホルダと既存研究を分析している [104] 。FLOSS に関するステークホルダ調査では，「FLOSS ボランティアエンジニアと貢献者」，「FLOSS プロジェクト雇用フルタイムエンジニア」，「FLOSS 開発企業」，「FLOSS 利活用企業」，「FLOSS 利用サポート企業」，「FLOSS 利用組織とエンドユーザ」の観点から調査しており，各ステークホルダが抱える問題やモチベーションの違いなどを考察している。また，「品質管理・理解」，「人的管理」，「コミュニケーション管理・理解」，「OSS 利活用のサポート」の観点から FLOSS が抱える問題と既存研究を紹介している。これらの調査研究より，不具合修正プロセスやコミュニティの管理，開発者の管理，コミュニケーション理解などの研究が活発に行われていることがわかった。

Francis Bordeleau らは，研究，技術，ビジネスの分野において FLOSS がどのように進化してきたのかを分析している [105] 。研究分野では，プロプライエタリソフトウェアから FLOSS への移行，生成されたコードと開発プロセス，FLOSS ライセンス，FLOSS の品質に関する研究が活発に行われている。近年では，大規模コードリポジトリの分析と，その進化に関する調査，FLOSS コミュニティの分析，リソースの利用状況，セキュリティ問題などが活発に研究されている。技術分野では，ソフトウェアだけでなくハードウェアや Software-as-a-Service，モバイル，クラウドコンピューティング，ビッグデータ，機械学習などに大きく貢献している。ビジネス分野では，インフラ基盤や業務用アプリケーションがプロプライエタリソフトウェアから FLOSS に移行する傾向にあり，FLOSS ライセンスもビジネスに適用される傾向にある。

② FLOSS の活用事例報告書

上記で述べた通り，FLOSS は様々な分野で活用されている。FLOSS の活用方法は，以下の 2 つに大きく分けられる。本節では，(1) と (2)を実践報告書を基に紹介する。

(1)　作業効率の向上や経費の削減を目的に，業務に FLOSS を活用するパターン
(2)　開発効率の向上や開発コストの削減を目的に，システム開発に FLOSS を活用するパターン

(1) 業務における FLOSS の活用事例

行政府機関が経費の削減を目的に，FLOSS　を採用する事例が増加している。しかし，職員による新しいツールへの不満や教育コスト，既存ツールから FLOSS のツールへの移行コストなど，様々な問題も浮き彫りとなっている。

- ベルギーの組織で FLOSS のサーバを採用 [106]
- ブリュッセルの行政府機関で OpenOffice.org を導入 [107]
- ヨーロッパの行政府機関で OpenOffice.org を導入 [108]
- 南アフリカで FLOSS のデスクトップを採用 [109]
- キューバの教育現場で FLOSS ベースのスマートテレビを活用 [110]
- 大規模なソフトウェア開発で FLOSS のツールを活用 [111]

(2)システム開発における FLOSS を活用事例

企業のエンジニアだけでなく，研究者が開発効率の向上や開発コストの削減を目的に FLOSS を活用する事例も増えている。用途は，ネットワーク分析から大規模システムの構築まで様々である。

- FLOSS を用いた e-Learning システムの構築 [112]
- FLOSS を用いたソフトウェア性能測定システムの構築 [113]
- FLOSS を用いた無線メッシュネットワークの実装と性能分析 [114]
- FLOSS を用いたモバイルデータオフロード分析システムの構築 [115]
- FLOSS を用いた大規模計算機実験場監視システムの構築 [116]
- FLOSS を用いた農業灌漑システムの自動化 [117]
- FLOSS を用いた自動運転研究の DevOps 構築 [118]
- FLOSS を用いたソーシャルプラットフォームの構築 [119]
- FLOSS を用いたクラウドソーシング型プラットフォームの開発 [120]
- FLOSS を用いた企業向け顧客関係管理の DevOps 構築 [121]
- FLOSS を用いたエネルギー効率評価におけるデータコレクタの開発 [122]
- FLOSS を用いたエネルギーモニタリングのダッシュボード設計と実装 [123]
- FLOSS を用いたコーパスデータの保存・処理・リンク [124]
- FLOSS を用いた破砕鉱石粒子の画像における境界検出解析 [125]
- FLOSS を用いたクラウド型意思決定支援システムの開発 [126]
- FLOSS を用いた音声処理システムの設計と開発 [127]

上記の ① 既存のサーベイ論文と ② FLOSS の活用事例報告書を調査した結果，(1) 不具合修正時間の長期化や (2) 持続可能性の低迷，(3) 開発コストの増加などが問題として注目されていることがわかった。

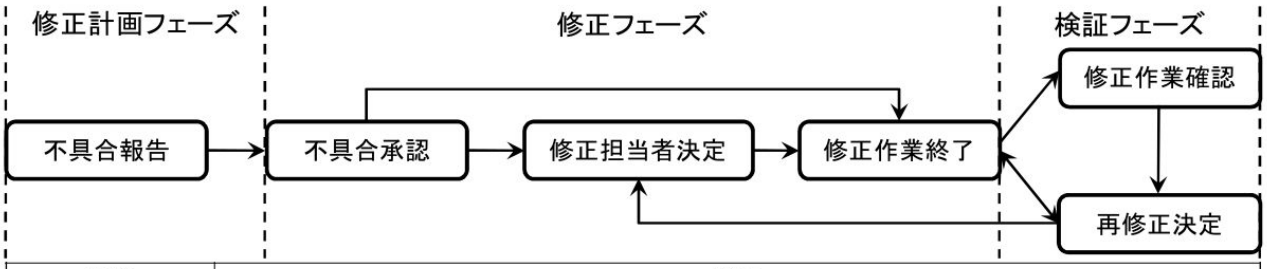
3.3. 既存研究の分類

3.2 節で，① 既存のサーベイ論文と ② FLOSS の活用事例報告書を調査した結果，(1) 不具合修正時間の長期化や (2) 持続可能性の低迷，(3) 開発コストの増加などが問題として注目されていることがわかった。そこで，(1) 〜 (3) の問題を細分化し，関連する論文の抽出しと分類を行った。本節では，(1) 〜 (3) の問題を細分化した結果と，そこに分類した論文について記述する。

3.3.1. 不具合修正時間

不具合修正時間は，不具合が報告されてから修正されるまでの不具合修正プロセス (図 9) が完了する時間を指す。不具合修正時間は，FLOSS の品質を測定する基準として用いられることもあり，不具合修正プロセスを分析する研究 [128-137] や不具合修正時間予測モデルを提案している研究 [138-140] などが活発に行われている。そのため，不具合修正時間は短くすることが推奨される。しかし，Andy Chou らの研究 [141] によると，Linux Kernel では不具合が修正されるまでに平均で 1.8 年，中央値で 1.25 年もかかることがわかっており，不具合修正時間の長期化が大きな問題となっている。この，FLOSS における不具合修正時間の長期化する原因として以下に示す 4 つが挙げられる。

① 重要な不具合報告の発見に時間がかかる
② 担当者の割り当てに時間がかかる
③ 該当ファイルの特定に時間がかかる
④ コードレビューに時間がかかる



状態	詳細
不具合報告	開発者又はユーザによって発見された不具合が報告（BTSに登録）された状態.
議論開始	BTSに登録された不具合情報に関して議論が開始された状態.
不具合承認	議論の結果，プロジェクト管理者によって修正すべき不具合として承認された状態.
修正担当者決定	プロジェクト管理者によって修正担当者が決定された状態.
修正作業終了	修正担当者によって修正作業が終了した状態.
再修正決定	不具合の修正が不十分であり，再度修正が要求された状態.
修正作業確認	プロジェクト管理者によって不具合が正しく修正されたことが確認された状態.

図 9 不具合修正プロセス [136]

① 重要な不具合報告の発見に時間がかかる

大規模な FLOSS には，1 日に数百もの不具合報告が届く。また，Thomas Zimmermann らの研究 [142] によって開発者が必要とする不具合情報とユーザが提供する不具合情報の間に差が生じていることがわかっている。そのため，FLOSS の開発者が本当に必要としている不具合報告を特定するのに時間がかかる。重複している不具合報告を検知する手法 [143-145] や不具合を分類することで重要度の高い不具合報告を効率的に探し出す研究 [146-148] が活発に行われている。

② 担当者の割り当てに時間がかかる

不特定多数の開発者が参加している FLOSS の開発では，開発者個人のプロフィールやスキルが不明瞭な場合が多いため，開発者に報告された不具合を割り当てのに時間がかかる。そこで，Gaeul Jeong らのマルコフ連鎖をベースとしたグラフモデルを用いて適切な修正担当者を自動的に決定する手法 [149] や Gerardo Canfora らのソフトウェアリポジトリに格納されている変更要求のテキスト記述を用いて適切な修正担当者を自動的に決定する手法 [150] などが提案されている。

③ 該当ファイルの特定に時間がかかる

報告された不具合に該当するファイルを特定するには時間がかかる。そこで，Armin Moin らが提案しているリビジョンログ解析とオープンバグリポジトリのテキスト分類を用いたバグの局所化手法 [151] や Anh Tuan Nguyen らが提案しているトピックベースのアプローチによる不具合報告からの該当ファイルの検索空間の絞り込む手法 [152]，Chakkrit Tantithamthavorn らが提案しているテキスト特徴量と変更履歴のマイニングを組み合わせて不具合報告に対して修正される可能性の高いソースコードファイルを推薦する手法 [153] など，報告された不具合に該当するファイルを自動的に特定する手法が提案されている。

④ コードレビューに時間がかかる

FLOSS では，修正パッチや追加機能などのソースコードをレビューするプロセスが存在する。しかし，コードレビューに時間がかかる場合が多い。そこで，Shoji Fujita らの，コードレビュープロセスを分析し，コミッタはコードレビューや編集を何度も行う傾向にあるが，コードレビューや編集を常に最速で行っているわけではないことを明らかにした研究 [154] や Jesus M. Gonzalez-Barahona らの，バグトラッキングツールの Bugzilla に保存されているデータを分析し，コードレビュープロセスを定量的に評価する手法を提案している研究 [155]，Toshiki Hirao らの，コードレビューにおいて合意形成が正常に行われなかった場合の影響を分析した研究 [156]，Adam Alami らの，コードレビューに関するインタビュー調査を基に分析を行い，コードレビューの成功に繋がる要因や改善案の提示を行っている研究 [157] など，コードレビュープロセスの分析と成功要因の特定を行う研究が活発に行われている。

3.3.2. 持続可能性

FLOSS における持続可能性とは，FLOSS が長期的に開発と保守が行われていることを指し，FLOSS における品質や成熟度 [158]，信用度 [159] と関係する重要な評価基準である。しかし，メイン開発者の脱退による開発の停滞やフォーク [160-165] によるコミュニティの分断など，FLOSS における持続可能性の低下が大きな問題となっている。この，FLOSS における持続可能性が低下する問題は以下に示す 4 つの小問題に分解することができる。

① 新規参入者が少ない問題
② 参加者のモチベーションが低下する問題
③ 離脱者が多い問題
④ 女性の参加者が少ない問題

① 新規参入者が少ない問題

Evangelia Berdou の GNOME と KDE プロジェクトに参加する開発者に対するインタビュー調査 [166] や Igor Steinmacher らの FLOSS への新規参入者が直面する課題と障壁を調査した研究 [167-168] など，FLOSS への新規参入者が減少する原因を特定する研究が活発に行われている。これらの研究より，FLOSS への新規参入者が向上しない主な原因として，FLOSS への参加手順がわからないという理由が挙げられている。そのため，FLOSS への参加手順を公式ウェブサイトや GitHub のREADME.md に明記するなどの対策が考えられる。

② 参加者のモチベーションが低下する問題

Yunwen Ye らの FLOSS における社会構造や共進化を分析することで開発者のモチベーションを調査した研究 [169] や Andrew Schofield らの FLOSS コミュニティに対する参加者の認識や所属理由，参加方法などを調査した研究 [170]，Paul Allan David らの階層的クラスター分析手法を用いて FLOSS 参加者のモチベーションを調査した研究 [171]，Felipe Fronchetti らの KSC クラスタリング分析手法やランダムフォレスト分類器を用いて FLOSS の参加者が魅力的に感じる要因を調査した研究 [172] など，FLOSS 参加者のモチベーションに影響を及ぼす要因を調査した研究が活発に行われている。これらの研究より，参加者の FLOSS における役割が FLOSS 参加者のモチベーションに影響を及ぼすことがわかっている。そのため，FLOSS の参加者に対して役割や権限を与えることがモチベーション向上に繋がると考えられる。

③ 離脱者が多い問題

Pratyush N Sharma らのロジスティック階層的線形モデリングを用いて参加者が FLOSS を脱退する要因を調査した研究 [173] や Ayushi Rastogi らの Google Chromium が採用しているバグトラッキングシステムをマイニングして参加者が FLOSS を脱退する要因を調査した研究 [174]，Courtney Miller らの調査と生存モデルを組み合わせた混合方法を用いて参加者が FLOSS を脱退する要因を調査した研究 [175] など，参加者が FLOSS を脱退する理由や要因を調査研究が活発に行われている。これらの研究より，参加者の FLOSS における役割や参加している FLOSS の規模や人気度が離脱の要因であることがわかっている。また，転職や進学などによって参加者の環境が大きく変化したことも離脱の大きな原因となっている。そのため，FLOSS の参加者に対して役割や権限を与えることがモチベーション向上に繋がると考えられる。

④ 女性の参加者が少ない

FLOSS の開発に参加する女性の割合は，他の分野と比較しても極端に少ない。FLOSS を継続的に開発するには，継続的な開発者の維持が必要不可欠である。そのため，Alison Adam が行ったハッカー倫理のジェンダー的側面に関する調査 [176] や Victor Kuechler らが行った女性の参加率に関する調査 [177]，Gregorio Robles らが行った FLOSS における女性の状況に関する調査 [178]，Vandana Singh らが行った FLOSS に参加している女性に対するアンケート調査 [179]，Huilian Sophie Qiu らが行った FLOSS への継続的な参加とソーシャルキャピタルの関係性を調査 [180]，Nasif Imtiaz らが行った GitHub におけるジェンダーバイアスの影響調査 [181] など，FLOSS と女性に関する様々な調査研究が行われた。その結果，女性同士でコミュニケーションが取れる環境づくりが重要であることがわかった。

KDE Community では，多様性対応の一環として女性専用スペースである「KDE Women」を提供している。Yixin Qiu らによって行われた「KDE Women」に関する調査 [182] より，女性専用スペースは有用であることがわかっている。しかし，Vandana Singh らが行った女性専用スペースの有無に関する調査 [183] より，ほとんどの FLOSS では女性専用スペースが設けられていない。そのため，女性専用スペースの設置促進が FLOSS の継続的な開発に繋がると考えられる。

3.3.3. 開発コスト

FLOSS を活用する大きな理由として Quality，Cost，Delivery (以降，QCD と称す) の向上が挙げられる。しかし, 以下の ① ～ ④ に該当する 4 種類の理由によって QCD が向上しない。または，FLOSS を採用したことによって開発コストが上がってしまうなどの事例が報告されている。

① 品質調査にコストがかかる
② 脆弱性の管理にコストがかかる
③ リリース管理にコストがかかる
④ ライセンス管理にコストがかかる

① 品質調査にコストがかかる

企業が FLOSS を採用する場合，事前に採用する FLOSS が自社の品質基準を満たしているかどうか事前調査する必要がある。そこで，Cap Gemini によって提案された FLOSS の開発プロセスを評価する方法論である Open Source Maturity Model (OSMM) from Cap Gemini [184] や Navica によって提案された FLOSS の開発プロセスを評価する方法論である Open Source Maturity Model (OSMM) from Navica [185]，Raphaël Semeteys によって提案された Define・Assess・Qualify・Select の 4 ステップを反復して FLOSS を評価する方法論である Qualification and Selection of Open Source software (QSOS) [186]，カーネギーメロン大学が提唱している企業向けの FLOSS を評価する方法論である Open Business Readiness Rating (OpenBRR) [187]，Etiel Petrinja によって提案された Capability Maturity Model (CMM) を FLOSS 向けに拡張した評価モデルである QualiPSo OpenSource Maturity Model (OMM) [188] など，FLOSS 向けの品質評価フレームワークが数多く提案されている。

② 脆弱性の管理にコストがかかる

FLOSS を活用したシステムの開発では，複数の FLOSS を活用することも珍しくない。活用している FLOSS の 1 つに脆弱性が発生した場合，脆弱性の詳細調査やシステムに対する影響調査など，脆弱性を管理するためにコストがかかる。そこで，Amiangshu Bosu らの Android Open Source Project の開発記録を分析し，脆弱性が混入するきっかけとなったソースコードの変更間隔を特定した研究 [189] や Zakir Durumeric らの Heartbleed がインターネットに及ぼした影響や修正パッチの適応率，Heartbleed を悪用したサイバー攻撃などを調査，分析した研究 [24]，Samim Mirhosseini らの GitHub におけるプルリクエスト通知が依存関係にある FLOSS のアップデートに繋がるかを調査，分析した研究 [190]，Raula Gaikovina Kula らの GitHub における依存関係のバージョンについて調査し，多くの FLOSS が古いバージョンで放置されており，脆弱性の認識をしていないことを分析した研究 [191]，Alexandre Decan らの Node Package Manager (以 降，npm と称す) における脆弱性の修正プロセスや他の FLOSS への影響を調査，分析した研究 [192]，Brandon Carlson らの GitHub で公開している FLOSS が脆弱性のある依存関係を含むプロジェクトの数と脆弱性の報告プロセスの有無を調査，分析した研究 [193]，Andreas Bauer らの依存関係を文章化する際の課題を調査した研究 [194] など，脆弱性と依存関係に関する研究が活発に行われている。

③ リリース管理にコストがかかる

FLOSS のリリース管理として，機能ベースや時間ベースのリリース戦略などが挙げられる。これらのリリース管理は，それぞれ長所と短所が存在しており，適切なリリース管理を選択，実行することが求められる。そこで，Martin Michlmayr らの FLOSS におけるリリース管理プロセスを分析し，FLOSS 参加者の調整が課題であることを特定した研究 [195] や Petri Sirkkala らの FLOSS に関連するステークホルダを考慮したリリース管理プロセスのフレームワークを提案している研究 [196]，Bruno Rossi らのリリースとダウンロードの傾向を分析した研究 [197]，Masayuki Hatta の開発プロセスにおけるヘッド指向とリリース指向を比較分析した研究 [198]，Germán Poo-Caamaño らの GNOME プロジェクトを分析し，リリース管理とコミュニケーションの関係性について明らかにした研究 [199-200]，Akinori Ihara らの Apache Software Foundation で開発されている FLOSS を調査し，リリースサイクルと採用されているバージョンの関係性を分析した研究 [201]，Jose Teixeira の機能ベースのリリース戦略と時間ベースのリリース戦略を比較分析した研究 [202]，José Apolinário Teixeira らのOpenStack における機能ベースのリリース戦略から時間ベースのリリース戦略への移行した経緯や影響を分析した研究 [203] など，リリース管理に関する研究は活発に行われている。

④ ライセンス管理にコストがかかる

近年，FLOSS ライセンス違反が大きな問題となっている。原因として，FLOSS ライセンスの管理漏れが挙げられる。そこで，Timo Tuunanen らが提案しているソースファイルの依存関係とソースファイル内のライセンスを識別する手法 [204] やDaniel M. German らが提案している文章マッチングベースの FLOSS ライセンス自動識別手法 [205]，Yunosuke Higashi らが提案している FLOSS ライセンスルールの自動生成に向けた FLOSS ライセンス文の階層的クラスタリング手法 [206] など，FLOSS ライセンスの管理を支援する研究が活発に行われている。また，Black Duck Protex や FOSSA などのライセンス管理ツールが登場している。

3.4. 既存研究の分析

3.4 節の過程で，① ～ ③ が存在することが明らかとなってきた。

① FLOSS が抱える問題の多くに「品質」が関係している
② ソースコードベースの品質と組織構造ベースの品質が存在する
③ ミクロな視点とマクロな視点で異なる問題が存在する

① FLOSS が抱える問題の多くに「品質」が関係している

上記で記述した不具合修正時間や参加者の活動量などは，FLOSS の品質と関係する要因として数多くの研究が行われている。また，3.1 節で収集した既存研究の内，110 件の既存研究 (タイトルとアブストラクト) に「Quality」という単語が含まれるなど，FLOSS の研究において「品質」は重要なトピックであると推測できる。

② ソースコードベースの品質と組織構造ベースの品質が存在する

サイクロマティック複雑度やコードクローン率など，ソースコードベースのソフトウェアメトリクスは，プロプライエタリソフトウェアと変わらず使用できる。しかし，不具合修正時間や参加者の活動量などの組織構造ベースの品質は，FLOSS 特有の品質になる。そのため，プロプライエタリソフトウェアの開発を前提としている ISO/IEC 9126 [207] や ISO/IEC 25000 (SQuaRE) [208] などのソフトウェア品質における国際規格を，そのまま流用することが難しい。

③ ミクロな視点とマクロな視点で異なる問題が存在する

FLOSS におけるミクロな視点とは，ソースコード単体や FLOSS 参加者単体に焦点を合わせた場合を指す。この場合，コードクローン率や FLOSS 参加者の活動量などの問題が浮かび上がる。FLOSS におけるマクロな視点とは，FLOSS のコミュニティや FLOSS のステークホルダなどを 1 つのノードとし，FLOSS をエコシステムとして捉えた場合を指す。この場合，FLOSS コミュニティ単位の品質やステークホルダの影響などが問題として浮かび上がる。

① ～ ③ を視覚的に分析するため，② と ③ を，それぞれ縦軸と横軸にした 2 軸のマトリクスを作成する。作成した 2 軸のマトリクスに既存研究を描画した結果を以下に示す (図 10) 。描画したノードには，研究の概要と，その研究に該当すると判断した論文数を記述している。

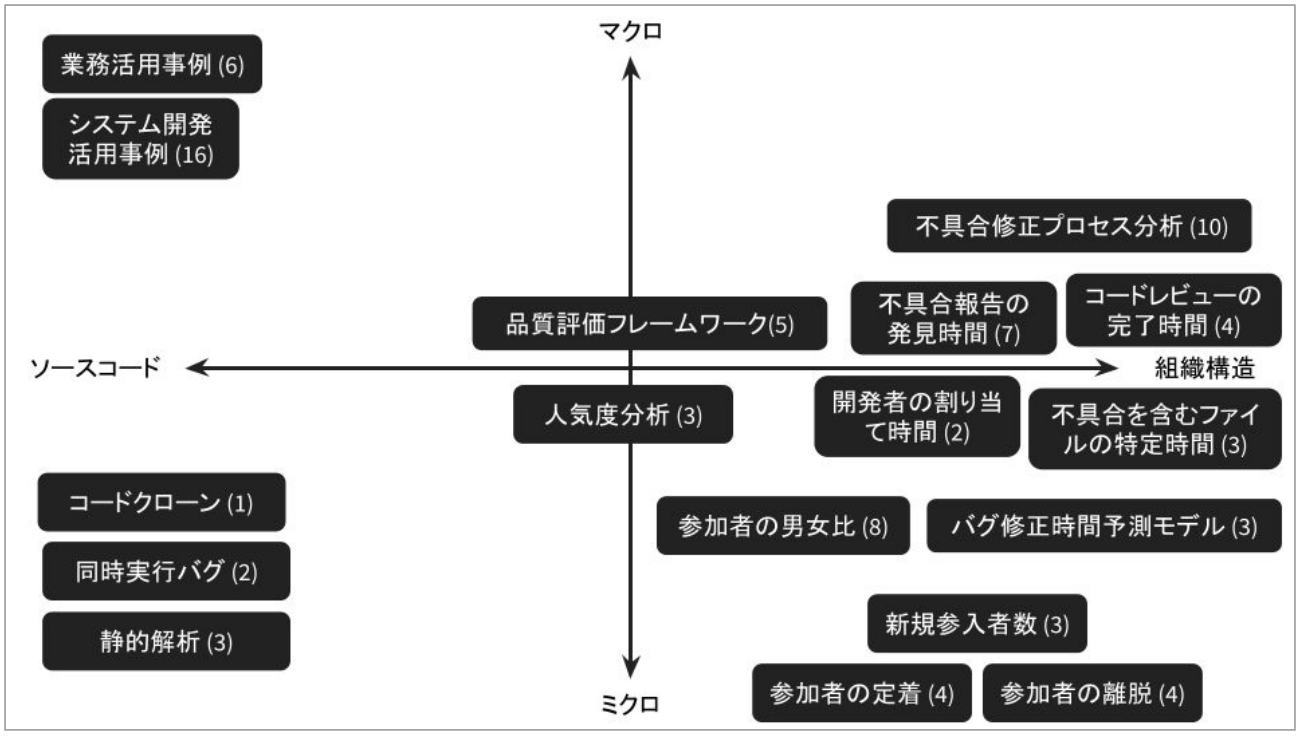


図 10 FLOSS の既存研究分析結果

4. 組織構造分析手法の既存研究調査

FLOSS は，プロプライエタリソフトウェアとは異なりボランティアが中心となった命令系統が存在しないコミュニティが開発している。それにもかかわらず，プロプライエタリソフトウェアと同等，もしくは同等以上の品質を維持している。品質を維持できる要因を FLOSS コミュニティの組織構造を分析することで特定しようとする研究は活発に行われている。そこで，FLOSS に関する既存研究で用いられている組織構造分析手法の他，社会学などの情報科学分野以外で用いられている組織構造分析手法を調査した。本章では，調査した 4 つの組織構造分析手法 (クラスター分析，ソーシャルネットワーク分析，Agent-Based Model，STAMP/STPA) の概要と既存研究について記述する。

4.1. クラスター分析

クラスター分析は，データをグループに分類するデータ分析手法の一種である。クラスター分析には，大きく分けて「階層的クラスタリング」と「非階層的クラスタリング」が存在する。それぞれについて詳しく記述する。

まず初めに，階層的クラスタリングについて述べる。階層的クラスタリングは，データの分析結果としてデンドログラム (図 11) を出力する。デンドログラムによって，データが分類されるまでの過程を視覚的に確認することができる。その一方，計算量が後述する非階層的クラスタリングに比べると大きくなるため，大規模なデータ分析では向かない。階層的クラスタリングの代表的な手法として Ward 法や群平均法，最短距離法などが挙げられる。

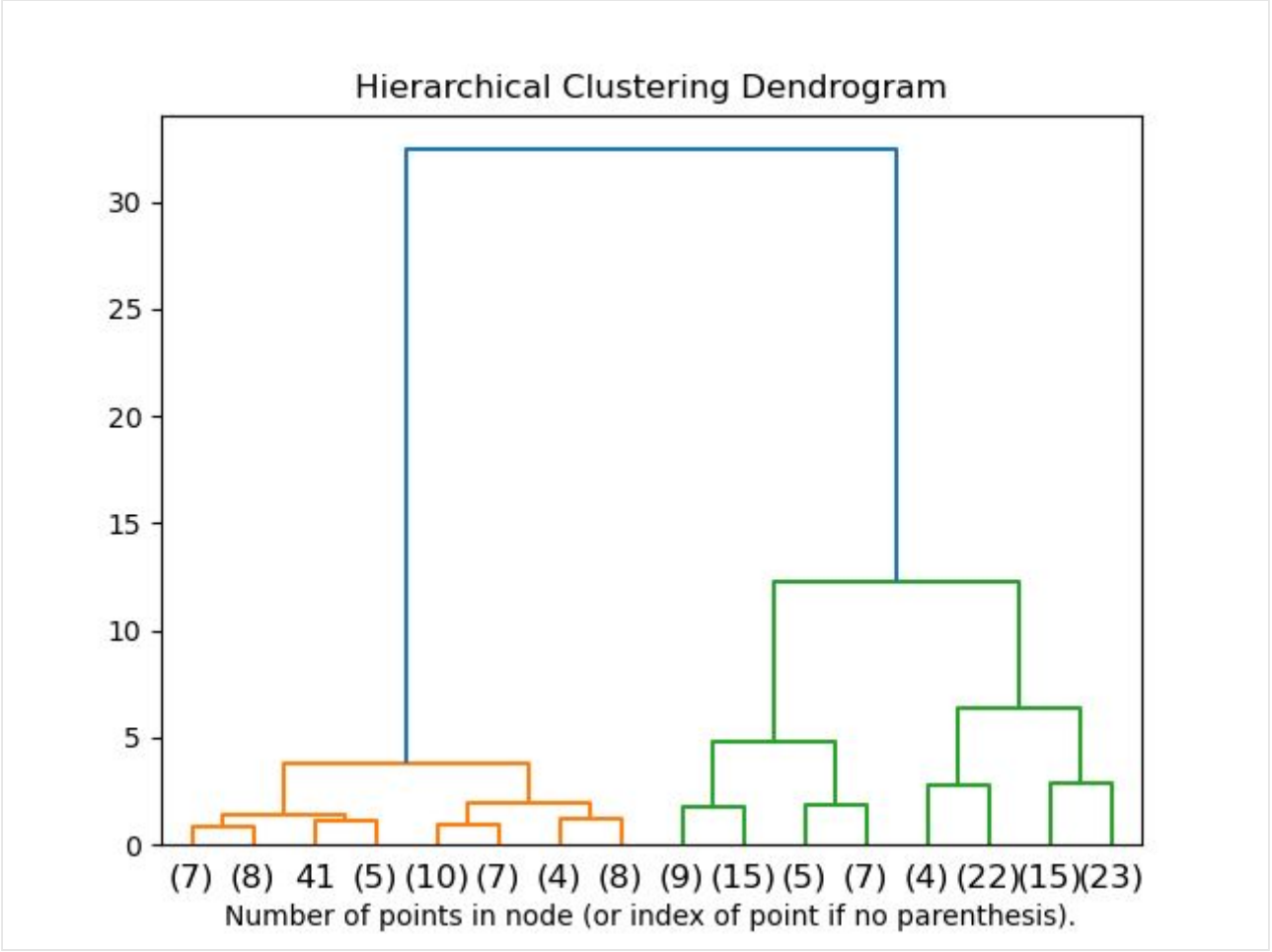


図 11 デンドログラム [209]

次に，非階層的クラスタリングについて述べる。非階層的クラスタリングは，似た性質を持つデータを自動的にグルーピング (図 12) する。階層的クラスタリングと異なり，分類されるまでの過程は確認することができない。しかし，階層的クラスタリングに比べると計算量が少ないため，大量のデータを分析する際に用いる。非階層的クラスタリングの代表的な手法として k-means が挙げられる。

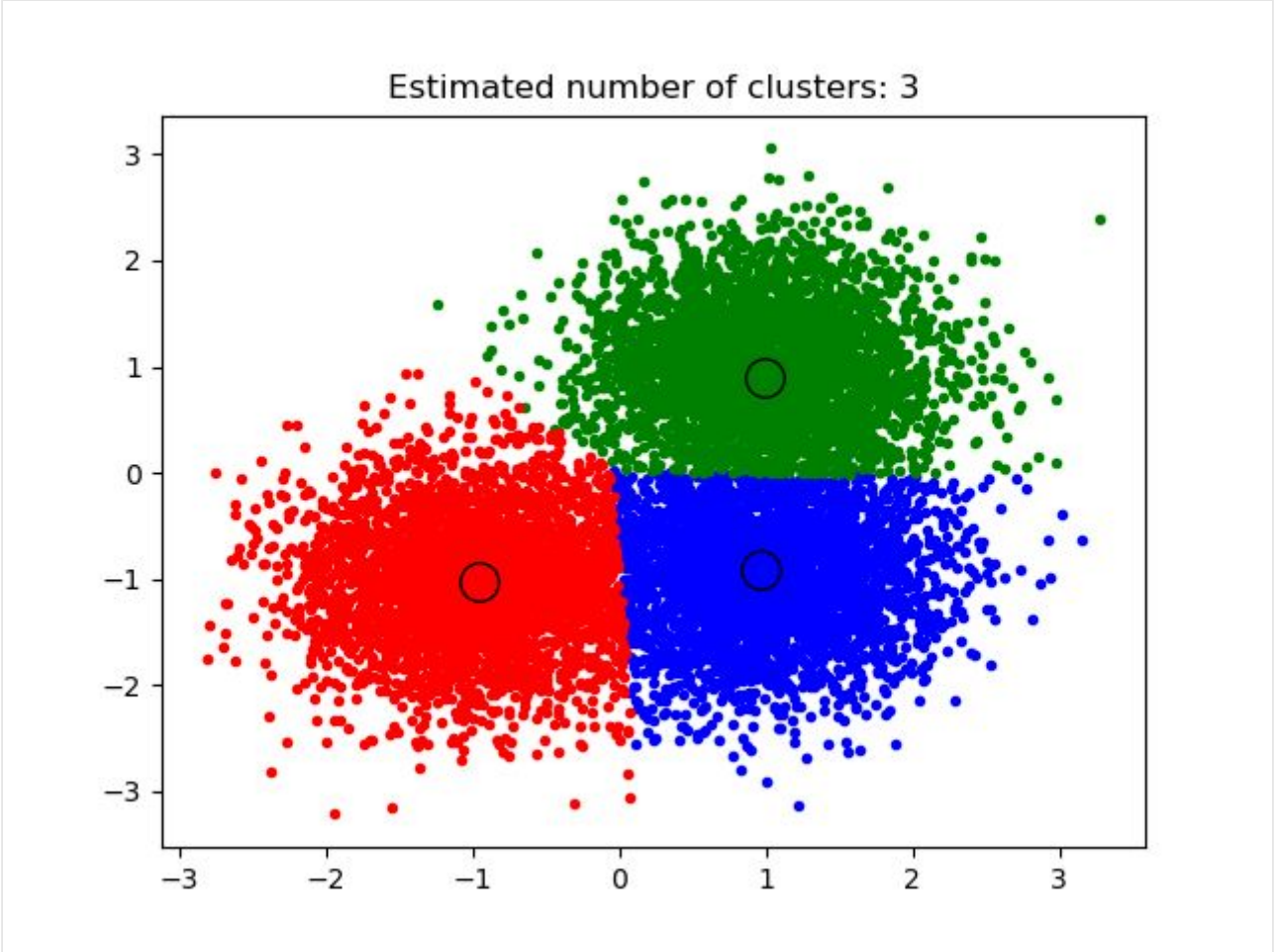


図 12 グルーピング [210]

尾上らは，階層的クラスタリングの 1 種である Ward 法を用いて FLOSS の開発者を分類する研究 [211] を行っている。開発記録としてソースコードホスティングサービスである GitHub に保存されている開発記録を用いている。GitHub 上に保存されている開発記録は GitHub API を用いることによって，PullRequest や CommitComment，IssueComment などのイベントタイプが付与されたデータとして取得することができる。イベントタイプは 15 種類に分類されているが，尾上らの研究では開発に関係があると考えられる 10 種類のイベントタイプに該当する開発記録のみを取得し，分析している。取得した開発記録を Ward 法を用いてクラスタリングし，期間や行動回数などから 5 つの開発者グループに分類している (図 13) 。

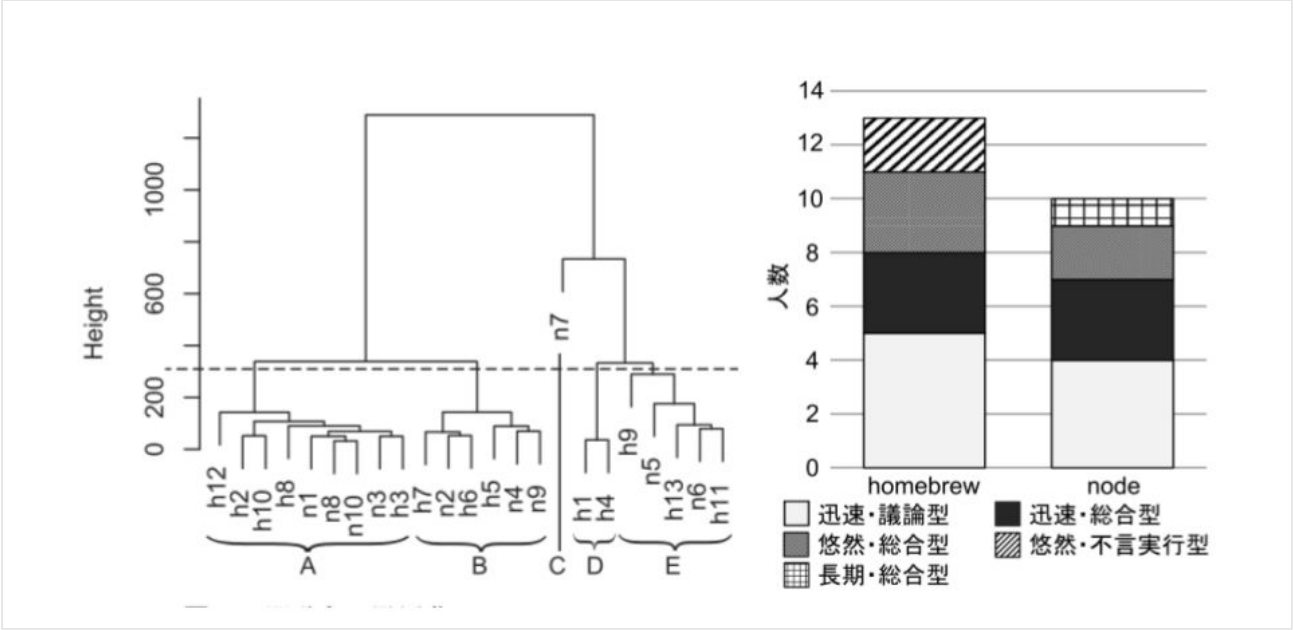


図 13 開発者の分類結果 [211]

4.2. ソーシャルネットワーク分析

ソーシャルネットワーク分析は，ネットワークとグラフ理論を使用してソーシャルネットワークを調査するプロセスである [212] 。ソーシャルネットワークをグラフ (図 14) として視覚的に確認することができ，ソーシャルネットワークの中心人物や，ソーシャルネットワークにおける影響範囲などを分析することができる。

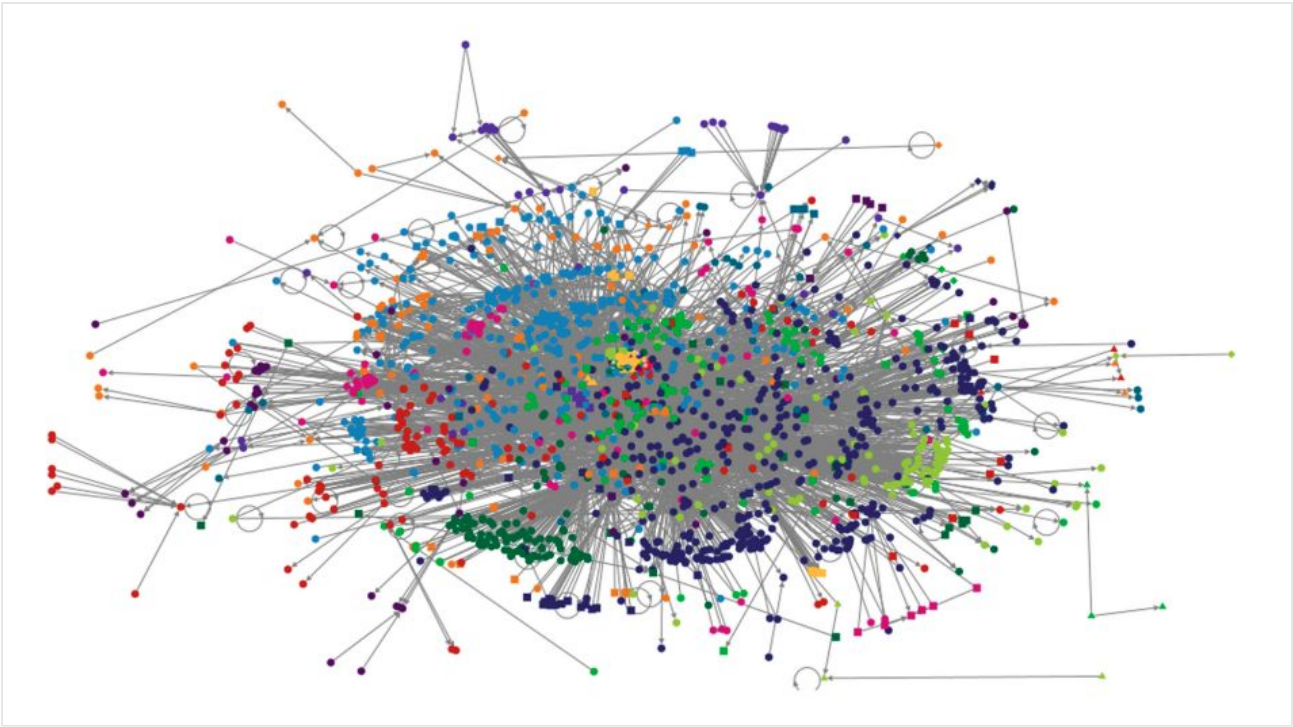


図 14 分類前のソーシャルネットワークグラフ [213]

構造的特徴量や指数としてノード数やエッジ数，中心性指数などが用いられる。特に，中心性指数は分析対象となるソーシャルネットワーク内におけるノードの重要性や影響力を測定する指標として様々な定義が存在する。代表的なものとして以下が挙げられる。

- 次数中心性: 各ノードに接続しているエッジの数
- 接近中心性: 特定のノードからネットワークへの距離
- 媒介中心性: 経路上に存在するノードの出現数
- PageRank: 重要なノードとノードへの経路数からノードの重要性を判断
- HITS: ハブ値と権威値からノードの重要性を判断

ソーシャルネットワーク分析は，社会科学だけでなく，経済学や物理学，情報科学など，様々な分野で幅広く応用されている [214-220] 。Ilse Struweg は，南アフリカの議会で国民健康保険法案が発表された直後の社会状況をソーシャルネットワーク分析している [213] 。分析データとしてソーシャルネットワーキングサービスの 1 つである Twitter 上に保存されているデータを活用している。このデータをソーシャルネットワーク分析ツールである NodeXL Pro を用いて分析している。その結果，南アフリカ政府，インフルエンサーユーザ，ゲートキーパーの存在を明らかにしている (図 15)。

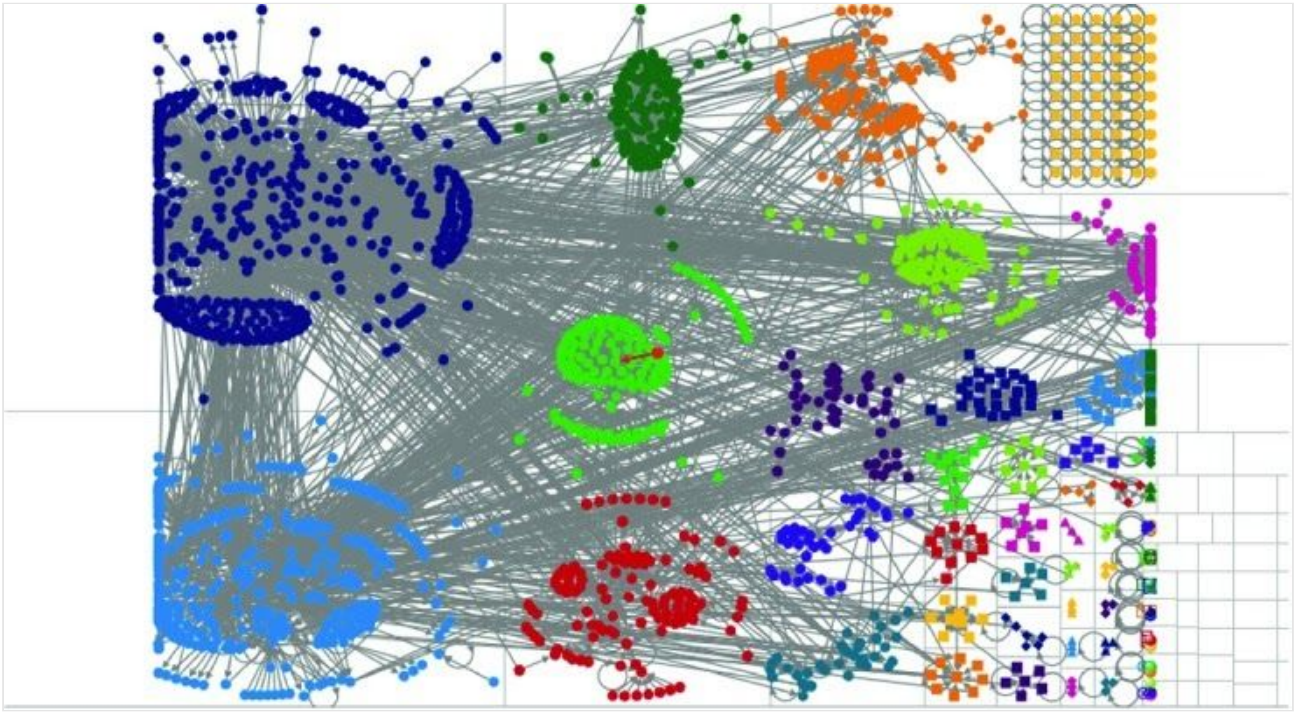


図 15 分類後のソーシャルネットワークグラフ [213]

また，ソーシャルネットワーク分析を用いて FLOSS を分析した研究 [SNA-5] も行われている 。Juan Martinez-Romo らは，企業が関与している FLOSS プロジェクトに対してソーシャルネットワーク分析することで，開発プロセスの効率性やリリース管理，リーダーシップの交代などの側面から FLOSS プロジェクトを分析している。分析対象のデータは，Concurrent Versions System や Subversion などのバージョン管理システムに保存されている開発記録をデータマイニングツールである CVSAnalY を用いて取得している。GNOME プロジェクトの公式個人情報管理ツールである Evolution と .NET Framework 互換プラットフォームである Mono を対象に (1) Distance centrality，(2) Betweenness centrality，(3) Coordination degree の指標を分析している。分析の結果，プロジェクト内で重要な役割を担っている開発者の特定やボランティア開発者とフルタイムエンジニアの比率，離職率の低さなどがわかっている (図 16) 。

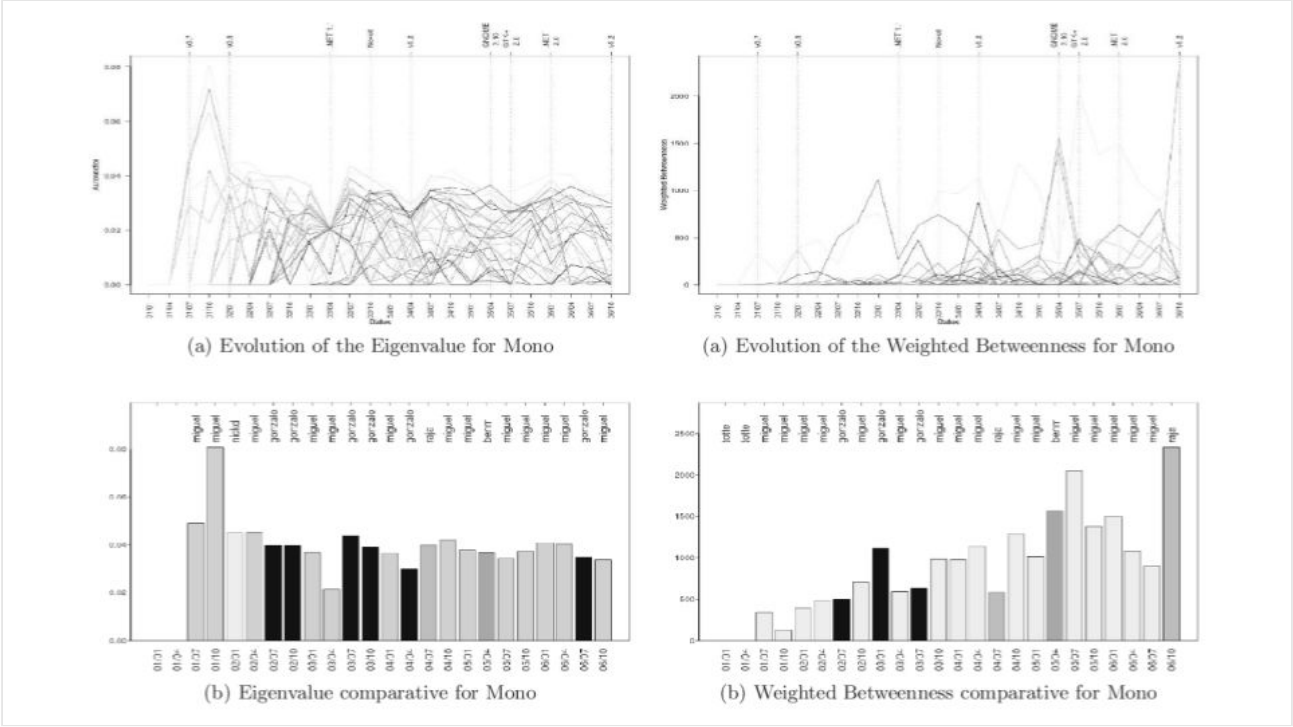


図 16 ソーシャルネットワーク分析を用いた FLOSS プロジェクトの分析結果 [SNA-5]

4.3. Agent-Based Model

Agent-based model (以降，ABM と称す) は，組織やコミュニティの個人または集団を模した自律的なエージェントの行動や相互作用をシミュレーションし，システム全体への影響を評価するための計算モデルの 1 種である [221-225] 。Nigel Gilbert は，ABM を以下の側面から説明している [226] 。

① 計算法
② 実験
③ モデル
④ エージェント
⑤ 環境

① 計算法

ABM は，計算社会科学の一形態である。つまり，コンピュータ・プログラムであるモデルを構築することを含む。モデリングの概念は，多くの社会科学でよく知られているように，社会的現実のある種の簡略化された表現を作成し，現実がどのように機能していると考えているかを可能な限り明確に表現することを目的とする。

② 実験

実験は，隔離されたシステムに何らかの作用を施し，何が起こるかをシミュレーションすることで構成されています。作用されたシステムは，作用を受けない別ののシステムと比較されることで評価される。

③ モデル

モデルは，ある実在する現象を表現したり，シミュレートしたりすることを目的としたものであり，これをモデルの対象と呼ぶ。スケールモデルや理想型モデル，類似モデルなど，様々なモデルが存在する。

④ エージェント

エージェントは，独立したコンピュータ・プログラムである。より一般的には，社会的アクター (個人，企業などの組織，国家などの組織) を表現するために使用されるプログラムの一部である。これらのプログラムは，それらが置かれている計算環境に反応するようにプログラムされている。この環境は，社会的行為者が活動している実際の環境のモデルとなっている。

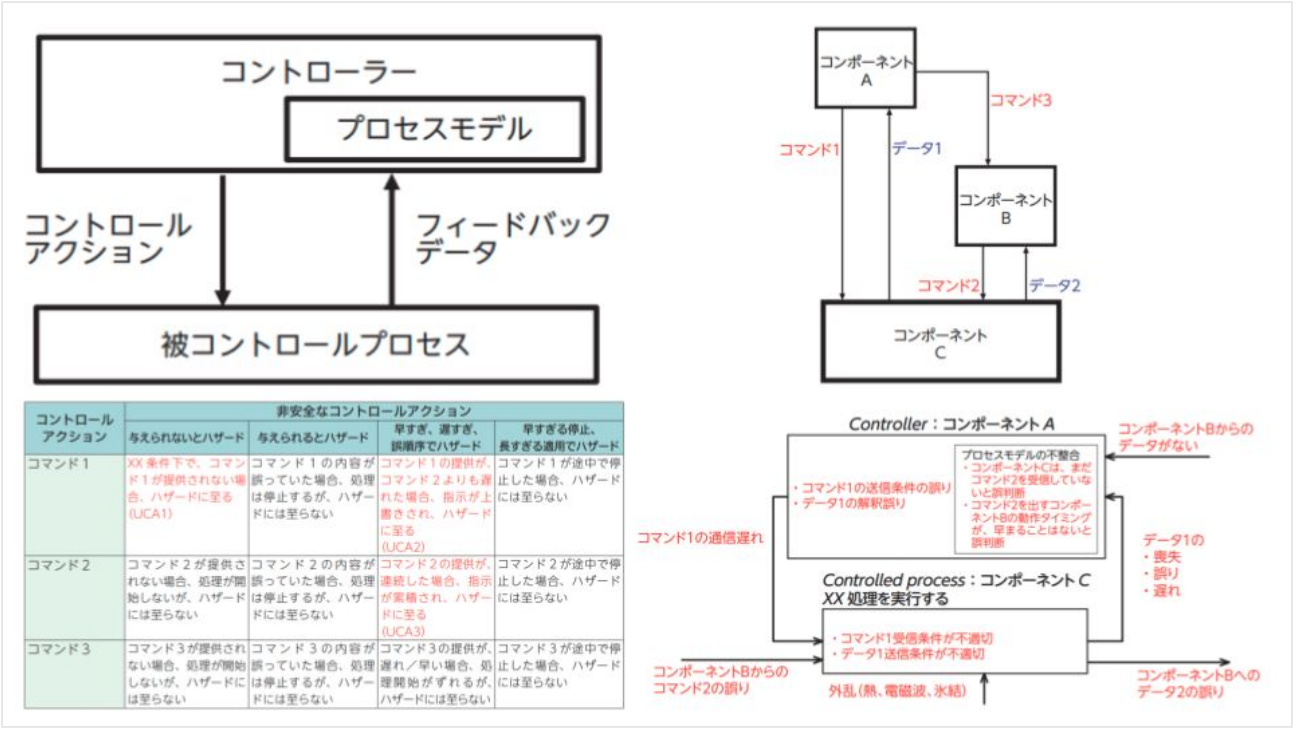
⑤ 環境

環境 は，エージェントが行動する仮想世界のことである。環境 は，エージェントに対して影響を及ぼさない。または，完全に中立的な媒体である。他のモデルでは，環境はエージェント自身と同じように入念に作られている場合も存在する。

4.4. STAMP/STPA

System Theoretic Accident Modeland Processes / System Theoretic Process Analysis (以降，STAMP/STPA と称す) は，マサチューセッツ工科大学の Nancy Leveson 教授が提唱する安全分析モデルと，そのモデルを用いた安全分析手法の総称である [227-228] 。Fault Tree Analysis や Failure Modes and Effects Analysis など，従来から用いられている安全性解析手法は，単体のコンポーネントから構成されるシステムの解析を前提としている。しかし，近年のシステムは複数のコンポーネントから構成されることが多いため，Fault Tree Analysis や Failure Modes and Effects Analysis などの従来手法では不十分である。STAMP/STPA では，4 つのプロセスに従ってハザードを抽出し，その要因までを特定する [SS-18-19] (図 17)。

1) アクシデント，ハザード，安全制約の識別
2) コントロールストラクチャの構築
3) 非安全なコントロールアクションの抽出
4) ハザード要因の特定



コントロールアクション	非安全なコントロールアクション			
	与えられないとハザード	与えられるとハザード	早すぎ、遅すぎ、誤順序でハザード	早すぎる停止、長すぎる適用でハザード
コマンド１	XX 条件下で、コマンド１が提供されない場合、ハザードに至る (UCA1)	コマンド１の内容が誤っていた場合、処理は停止するが、ハザードには至らない	コマンド１の提供が、コマンド２よりも遅れた場合、指示が上書きされ、ハザードに至る (UCA2)	コマンド１が途中で停止した場合、ハザードには至らない
コマンド２	コマンド２が提供されない場合、処理が開始しないが、ハザードには至らない	コマンド２の内容が誤っていた場合、処理は停止するが、ハザードには至らない	コマンド２の提供が、連続した場合、指示が累積され、ハザードに至る (UCA3)	コマンド２が途中で停止した場合、ハザードには至らない
コマンド３	コマンド３が提供されない場合、処理が開始しないが、ハザードには至らない	コマンド３の内容が誤っていた場合、処理は停止するが、ハザードには至らない	コマンド３の提供が、遅れ／早い場合、処理開始がずれるが、ハザードには至らない	コマンド３が途中で停止した場合、ハザードには至らない

図 17 STAMP/STPA の分析手順 [229]

Stuart Williams は，クルーズ・フェリー運航会社における組織的リスクを STAMP/STPA を用いて分析することによって，潜在的なハザードを特定する研究 [SS-11] を行っている。まず初めに，ガイドラインに従ってアクシデント，ハザード，安全制約の識別している。次に，コントロールストラクチャ (図 18) を構築している。最後に，非安全なコントロールアクションを抽出している。分析の結果，コミュニケーションパスのギャップ，リスクのトラッキングの欠如，リスクプロセスへの上級管理職の関与の限定などのハザードが抽出されている。

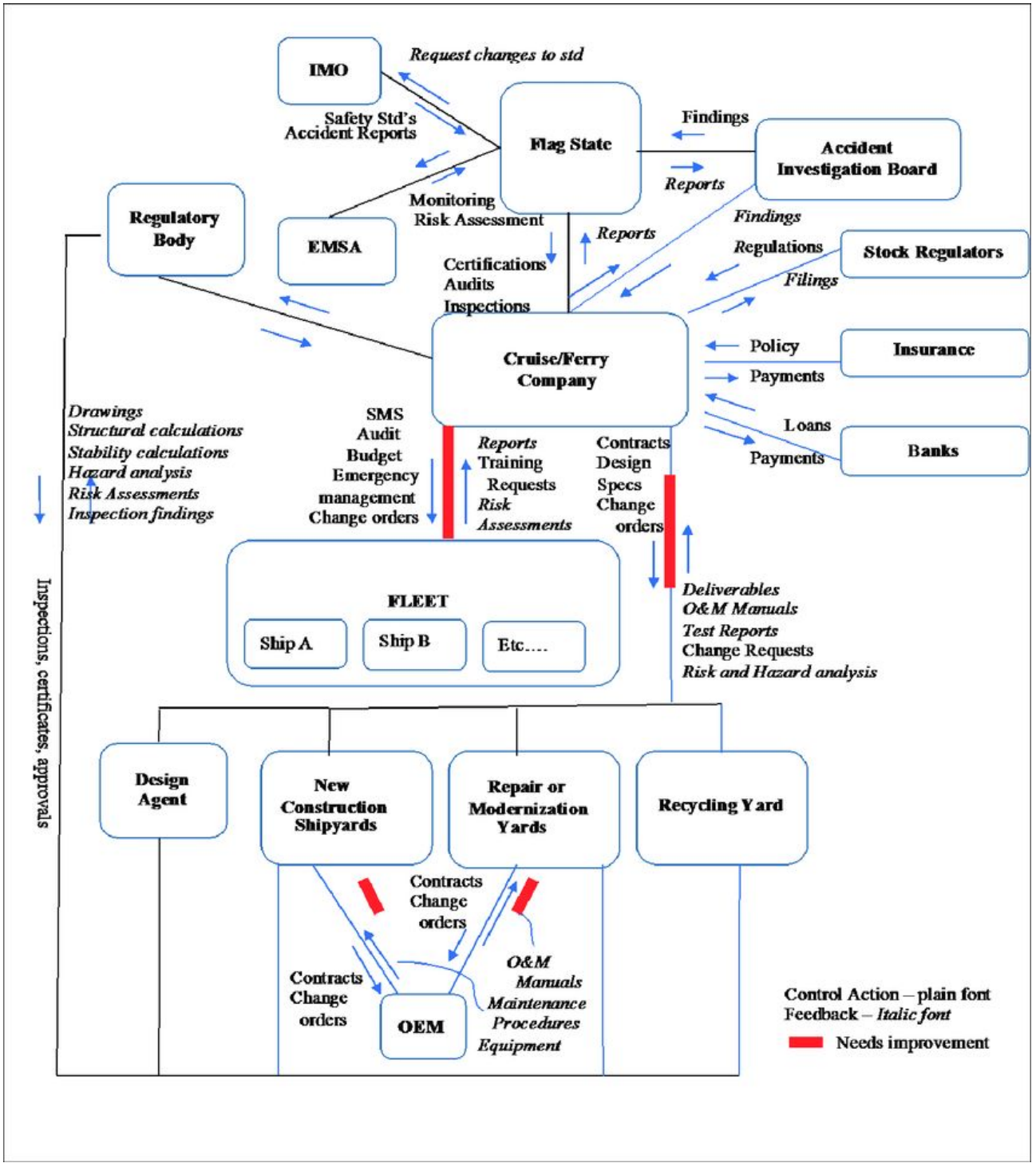


図 18 クルーズ・フェリー運航会社のコントロールストラクチャ [SS-11]

近年，システムや組織の潜在的リスクを STAMP/STPA を用いて特定する研究 [231-241] が活発に行われている。また，マサチューセッツ工科大学の Nancy Leveson 教授のもとで STAMP Workshop [242] が開催されている。日本では，独立行政法人情報処理推進機構が STAMP ワークショップ [243-245] を開催したり，STAMP/STPA に関する書籍 [229-230] を発行している。このように国内外で STAMP/STPA は注目されている。

5. FLOSS プロジェクトの分析

プロプライエタリソフトウェアプロジェクトでは，参加している開発者は企業に雇用されているフルタイムエンジニアであり，トップダウン型の厳格な命令系統が構築されている。その一方，FLOSS プロジェクトに参加している開発者は基本的にボランティアであり，厳格な組織開発は存在しない。このように，プロプライエタリソフトウェアプロジェクトと FLOSS プロジェクトでは，参加者や開発体系，コミュニケーション方法などに大きな違いが存在する。

プロプライエタリソフトウェアプロジェクト

- 参加者は雇用元の指示により参加する
- ソフトウェアを開発する企業内メンバやパートナー会社に参加者は限られる
- 仕事としてプロジェクトに参加する
- 活動時間や活動量はプロジェクトマネージャによって決定される
- 意思決定はトップダウンで行われる
- プロジェクトマネージャ，リーダ，メンバなど階層化された組織構造が存在する

FLOSS プロジェクト

- 誰でも自由に参加と脱退ができる
- 企業，個人など参加形態を問わない
- プロジェクトに参加することによる対価は基本的にない
- 活動時間や，その量は参加する人の意思による
- 意思決定は基本的な民主主義による合意の上行われる
- コミッタやコントリビュータなどの役割が定義されている

ソフトウェア開発における組織開発論や開発プロセス論などは，古くから活発に研究が行われている。しかし，上記で述べた通りプロプライエタリソフトウェアプロジェクトと FLOSS プロジェクトでは参加者や開発体系，コミュニケーション方法などに大きな違いが存在する。そのため，既存のソフトウェア開発における組織開発論や開発プロセス論が当てはまらない部分が多々存在する。しかし，FLOSS プロジェクトでは Linux Kernel を代表する高品質なソフトウェアを継続的に開発しており，FLOSS プロジェクトからソフトウェア工学的知見を得ようとする研究が増加している。

本章では，FLOSS プロジェクトで行われているソフトウェア開発プロセスやコミュニティ内で形成されている構造を理解することを目的とし，既存研究を参考に予備実験を行った過程と，その結果を示す。続く 5.1 節では，本予備実験で必要となる実験環境の構築手順について記述する。5.2 節では，FLOSS プロジェクトの参加者が行っている開発過程を分析することで，振る舞いの側面から FLOSS プロジェクトを理解しようとした手順と，その結果について記述する。5.3 節では，FLOSS プロジェクト内で行われているコミュニケーション履歴を分析することで，参加者同士における相互作用の観点から FLOSS プロジェクトを理解しようとした手順と，その結果について記述する。

5.1. 予備実験環境の構築手順

本予備実験は，Ubuntu 20.04.1 LTS (図 19) の環境下で行っており，Guido van Rossum によって設計されたインタープリタ型の高水準汎用プログラミング言語である Python 3 と Ryan Dahl によって設計された V8[25] を用いた JavaScript の実行プラットフォームである Node.js を活用している。

```
demo@VirtualBox:~$ cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04.1 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04.1 LTS"
VERSION_ID="20.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-poli
cy"
VERSION_CODENAME=focal
UBUNTU_CODENAME=focal
demo@VirtualBox:~$ uname -a
Linux VirtualBox 5.4.0-51-generic #56-Ubuntu SMP Mon Oct 5 14:28:49 UTC 2020 x86
_64 x86_64 x86_64 GNU/Linux
demo@VirtualBox:~$
```

図 19 Ubuntu の環境情報

Python 3 は，Ubuntu 20.04.1 LTS で標準的にインストールされているため環境構築を行う必要はない。しかし，Node.js は Ubuntu 20.04.1 LTS に標準的にインストールされていないため，まず初めに Node.js のインストール手順について記述する。Ubuntu に Node.js のインストール手順は複数存在する。本報告書では，Node.js のバージョン管理システムである n を用いたインストール手順を記述する (図 20) 。

① Ubuntu に標準インストールされているパッケージ管理 システムである Advanced Packaging Tool を用いて Node.js とパッケージ管理システムである npm をインストールする
② ① でインストールした npm を用いて n をインストールする
③ ② でインストールした n を用いて Node.js をインストールする
④ ① でインストールした Node.js，npm と ② ～ ③でインストールした Node.js，npm が競合するリスクを無くすため ① でインストールした Node.js，npm をアンインストールする
⑤ シェルシステムに再ログインし，① ～ ④ の設定を反映させる
⑥ Node.js のバージョン確認オプションを実行し，Node.js のバージョンが表示されれば成功である

[25] Google が開発するFLOSS の JIT Virtual Machine 型 JavaScript エンジン

```
$ sudo apt install -y nodejs npm ・・・①
$ sudo npm install n -g ・・・②
$ sudo n stable ・・・③
$ sudo apt purge -y nodejs npm ・・・④
$ exec $SHELL -l ・・・⑤
$ node -v ・・・⑥
v12.18.3
```

図 20 Node.js のインストール手順

次に，Python 3 で使用するライブラリのインストール手順について記述する。Node.js 同様に Python 3 のライブラリをインストール方法が複数存在する。本報告書では，Python 3 のパッケージ管理システムである pip3 を用いたインストール手順について記述する (図 21) 。

```
$ sudo apt install python3-pip
$ pip3 install PyGithub
$ pip3 install scikit-learn
$ pip3 install pandas
$ pip3 install networkx
```

図 21 Python 3 ライブラリ のインストール手順

また，以降の 5.2 節と 5.3 節で行った予備実験は，以下の環境下で行ったものである。

- Node.js Ver.12.18.3
- Python Ver.3.6.9
- Ubuntu 20.04.1 LTS

5.2. FLOSS 参加者の行動分析

FLOSS プロジェクトでは，プロジェクトマネージャ，リーダ，メンバなど階層化された組織構造が存在しない。その代わり，FLOSS 参加者の貢献方法やシステム権限の違いによってコミッタやコントリビュータなどに分類されることが多い。しかし，FLOSS 参加者は 1 人が 1 つの作業を行うのではなく，FLOSS 参加者自身が実行可能であると判断した行動のみを実行する。そのため，1 人がソースコードに適用するパッチ作成からドキュメントの整備まで様々な行動を実行する場合もあれば，不具合報告のみを行う参加者など多種多様であり，プロプライエタリソフトウェアプロジェクトにおける役割のように明確に分類することが困難である。そこで，FLOSS の開発記録に保存されている参加者の行動を分析し，FLOSS に参加している開発者の特徴を分類し，FLOSS プロジェクトにおける組織構造を理解する。

FLOSS の開発記録として，ソースコードホスティングサービスである GitHub に保存されている開発記録を利用する。GitHub に保存されている開発記録は，GitHub API [X] を用いることで誰でも取得することが可能である。プロトコル操作ツールである curl と GitHub API を用いて取得した開発記録を以下に示す (図 22) 。開発記録は JSON 形式で受信することができる。

図 22 GitHub の開発記録 (JSON)

GitHub API にも様々な種類が存在する。本予備実験では，GitHub API Events を用いて取得した開発記録を活用する。GitHub API Events では，15 種類のイベントタイプ (表 1) が記録されており，FLOSS 参加者が実行したアクションを詳細に分析することができる。例えば，開発者 A がプルリクエストを送信した場合，(10) PullRequestEvent が開発記録として保存される。また，開発者 A が送信したプルリクエストに対して，開発者 B がコメントを残した場合，(11) PullRequestReviewCommentEvent が開発記録として保存される。

表 1 GitHub API Events のイベントタイプと概要

	Event Types	Description
1	CommitCommentEvent	コミットコメントが作成
2	CreateEvent	Git ブランチまたはタグが作成
3	DeleteEvent	Git ブランチまたはタグが削除
4	ForkEvent	ユーザーがリポジトリをフォーク
5	GollumEvent	Wiki ページが作成または更新
6	IssueCommentEvent	Issue のコメントに関連するアクティビティ
7	IssuesEvent	Issue に関連するアクティビティ
8	MemberEvent	リポジトリの協力者に関連する活動

表 1 GitHub API Events のイベントタイプと概要 (続き)

9	PublicEvent	非公開リポジトリが公開
10	PullRequestEvent	プルリクエストに関連するアクティビティ
11	PullRequestReviewCommentEvent	プルリクエストレのビューコメントに関連するアクティビティ
12	PushEvent	コミットがリポジトリのブランチまたはタグにプッシュ
13	ReleaseEvent	リリースに関連するアクティビティ
14	SponsorshipEvent	スポンサーシップリストに関連するアクティビティ
15	WatchEvent	誰かがリポジトリにスターを付けたアクティビティ

GitHub API を使用する場合は，GitHub のアカウントを作成し，アクセストークンを発行する必要がある。本報告書では，GitHub　　アカウントの作成手順は割愛し，アクセストークンの発行手順のみを記述する。GitHub にログイン後，「Settings」→「Developer settings」→「Personal access tokens」に移動し，「Generate new token」をクリックする (図 23) 。

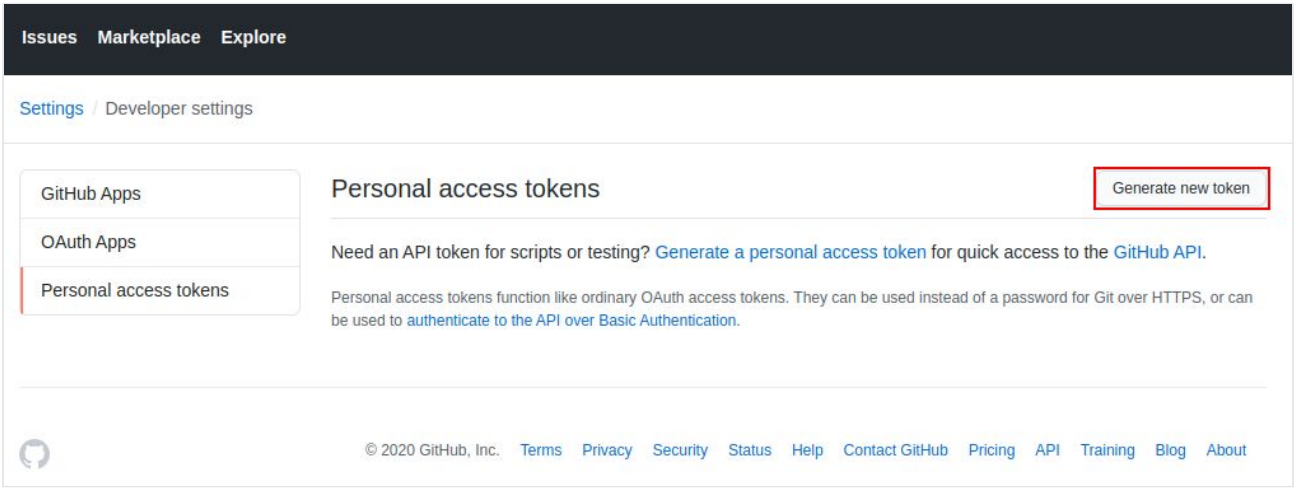


図 23 アクセストークンの生成画面 (1)

「Note」項目に任意の文字列を入力し，「Generate token」をクリックする (図 24) 。

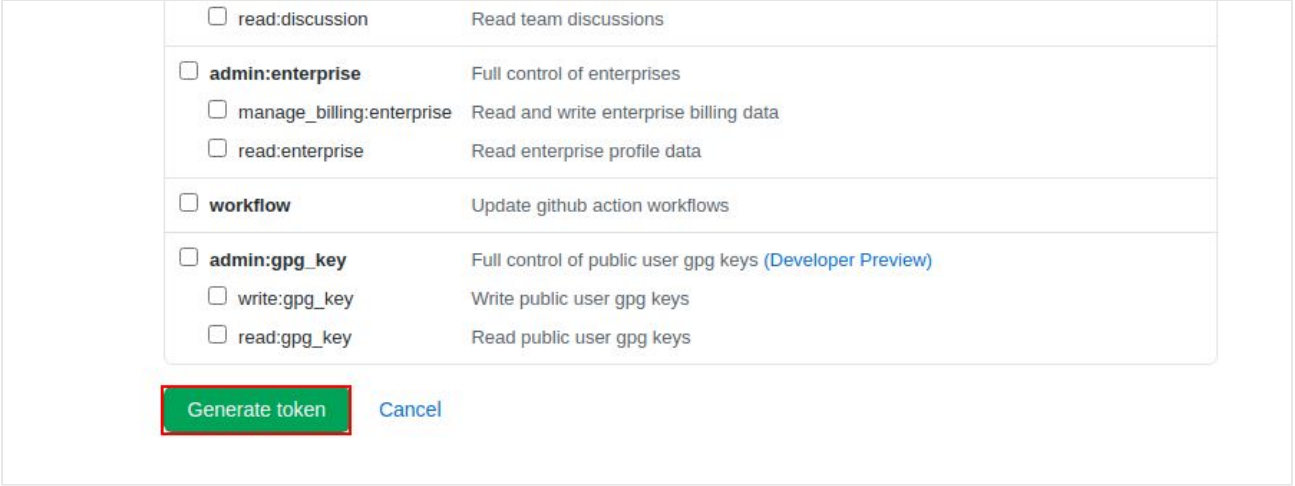


図 24 アクセストークンの生成画面 (2)

正常に実行された場合，画面上にアクセストークンが表示されるので記録しておく (図 25) 。

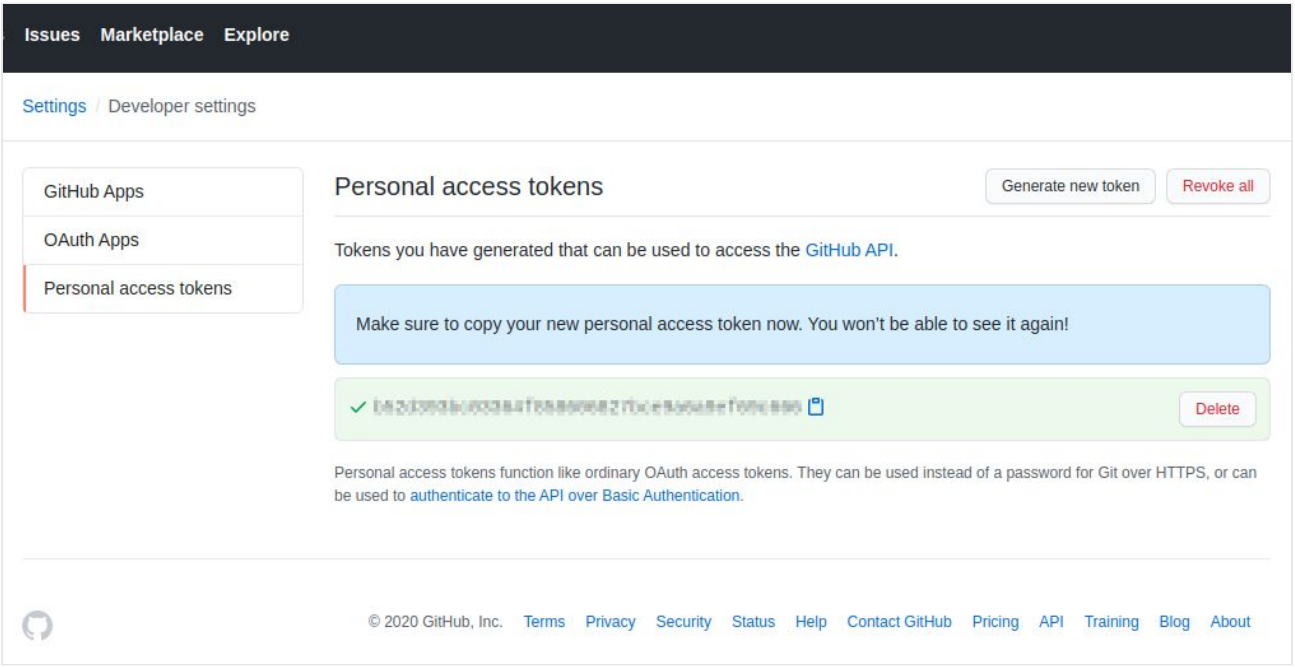


図 25 アクセストークンの生成画面 (3)

Python ファイル (付録 1) を任意の場所とファイル名 (本報告書では，get_dev_log.py とする) で保存する。[ACCESS_TOKEN] は，上記で発行したアクセストークンに置換する。[PROJECT_ID] は，分析対象のプロジェクト識別子 (本報告書では，python/cpython とする) に置換する。① 保存した Python ファイルを実行し，② 生成された JSON ファイルを確認する。JSON ファイルに GitHub のユーザ名とイベントタイプが出力されていれば正常に実行されている (図 26) 。

```
$ tree
.
└── get_dev_log.py
$ python3 get_dev_log.py ・・・①
$ tree
.
├── dev_log.json
└── get_dev_log.py
$ cat dev_log.json ・・・②
[{"user_name": "Dima Scherbakov", "event_type": "WatchEvent"}, {"user_name": "Pablo Galindo",
"event_type": "PullRequestReviewEvent"}, {"user_name": "Pablo Galindo", "event_type":
"PullRequestReviewCommentEvent"}, {"user_name": "Pablo Galindo", "event_type":
                                        (省略)
 "PullRequestEvent"}, {"user_name": "Raymond Hettinger", "event_type": "PushEvent"}, {"user_name": "Miss
Skeleton (bot)", "event_type": "IssueCommentEvent"}, {"user_name": "Raymond Hettinger", "event_type":
"PullRequestEvent"}, {"user_name": "Brett Cannon", "event_type": "IssueCommentEvent"}]
```

図 26 開発記録の取得手順

JavaScript ファイル (付録 2) とパッケージファイル (付録 3) を，上記の Python ファイルと同じディレクトリ内に保存する。JavaScript ファイルは，任意のファイル名 (本報告書では，app.js とする) で保存して問題ないが，パッケージファイルは package.json というファイル名で保存する。① 依存関係のあるパッケージをインストールした後，② JavaScript ファイルの実行結果を CSV ファイルとして出力する (図 27) 。

```
$ tree
.
├── app.js
├── dev_log.json
├── get_dev_log.py
└── package.json
$ npm i ・・・①
$ node app.js > dev_log.csv ・・・②
$ tree -L 1
.
├── app.js
├── dev_log.csv
├── dev_log.json
├── get_dev_log.py
├── node_modules
├── package-lock.json
└── package.json
```

図 27 開発記録の加工手順

Python ファイル (付録 4) を上記のファイルと同じディレクトリ内に保存する。① 保存した Python ファイルを実行する。Python ファイル (付録 4) では，dev_log.csv に保存されているデータを主成分分析と k-means を用いたクラスタリングの順で実行し，グラフとして出力する。主成分分析は，15 次元の開発記録を視覚可能な状態にするために，データの次元削減手法として用いた。また，k-means は非階層的クラスタリングの 1 種であり，特徴が似ている開発者をグループに分類するクラスタリング手法として用いた (図 28) 。

```
$ tree -L 1
.
├── app.js
├── clustering.py
├── dev_log.csv
├── dev_log.json
├── get_dev_log.py
├── node_modules
├── package-lock.json
└── package.json
$ python3 clustering.py ・・・ ①
```

図 28 開発記録の主成分分析と k-means 実行手順

上記の分析手順を，Python，OpenSSL，Flutter の FLOSS プロジェクトに対して実行し，グラフとして出力した結果 (図 29) を以下に示す。Python プロジェクトと Flutter プロジェクトでは，少数の開発者が同じ作業を行っている傾向がみられる。つまり，現在，活動量が多い開発者がプロジェクトを脱退することで，開発活動が止まるリスクを抱えている。その一方，OpenSSL プロジェクトでは，満遍なく活動量と作業の種類が分散しており，開発活動が止まるリスクも分散しているといえる。

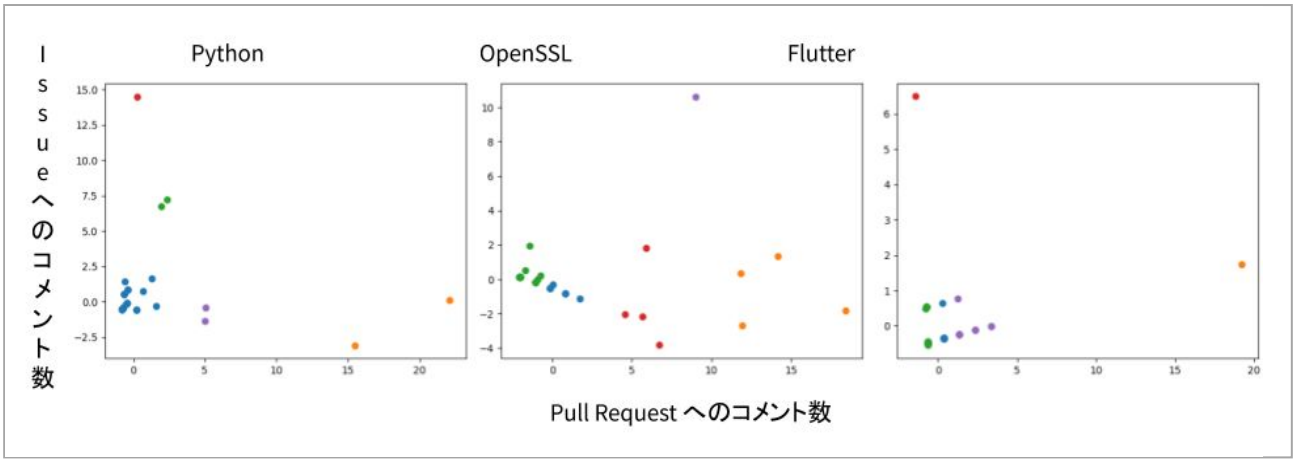


図 29 FLOSS 参加者の行動分析結果

5.3. FLOSS 参加者のコミュニケーション分析

プロプライエタリソフトウェアプロジェクト内では，基本的にコミュニケーションは対面で行われる場合が多い。しかし，FLOSS プロジェクトでは，世界各国に分散した開発者同士が参加しているため，対面でのコミュニケーションは基本的に行われない。そこで，従来より FLOSS プロジェクトにおけるコミュニケーションツールとしてメーリングリストが使用されてきた。しかし，近年では Slack などの新しいコミュニケーションツールが登場したことによって，FLOSS プロジェクト内でのコミュニケーション手段も変化しつつある。

1990 年代ごろの FLOSS プロジェクトではメーリングリストが代表的なコミュニケーションツールであった。これらのコミュニケーション履歴はアーカイブとしてインターネット上に保存されている場合があり，これらのコミュニケーション履歴からソフトウェア工学的知見や社会学的観点からコミュニケーションを理解する研究の発展に貢献している。
2000 年代になると Internet Relay Chat (以降，IRC と称す) プロトコルを利用したソフトウェアが登場したことによって，メールからチャットへコミュニケーション手段が徐々に変化していった。IRC は，リアルタイムなコミュニケーションが可能なため，FLOSS プロジェクト内のコミュニケーションツールも完全にメールからチャットに置き換わると思われていた。しかし，上記でも述べた通り FLOSS の参加者は世界各国に分散しており，時差の影響を大きく受ける。そのため，非同期システムであるメールが FLOSS プロジェクト内のコミュニケーションツールとして長らくデファクトスタンダードとなっていた。

近年では，GitHub や GitLab などのソースコードホスティングサービスとソーシャルネットワーキングサービスが融合した新しいサービスが提供されている。多くの FLOSS プロジェクトも，これらのサービスに移行しており，ソーシャルコーディングという新しい概念が生まれている。これまでメーリングリストなどで行われてきたコミュニケーションがソーシャルコーディングをサポートする新しいサービス上で行われるようになっている。

メーリングリストからソーシャルコーディングサービスにコミュニケーションツールが移行したことによって，FLOSS プロジェクト内のコミュニケーション手段や影響も大きく変化している。そこで，本節では代表的なソーシャルコーディングサービスである GitHub 上で行われているコミュニケーションを分析することによって，FLOSS プロジェクトの社会的側面を理解する。

GitHub では，FLOSS 参加者同士がコミュニケーションを行える機能を提供している (図 30) 。不具合の報告や追加機能の要望，修正パッチの送信など様々なアクションが行える。その過程で，その参加者が，どの参加者とコミュニケーションを行っているかを調べられる。

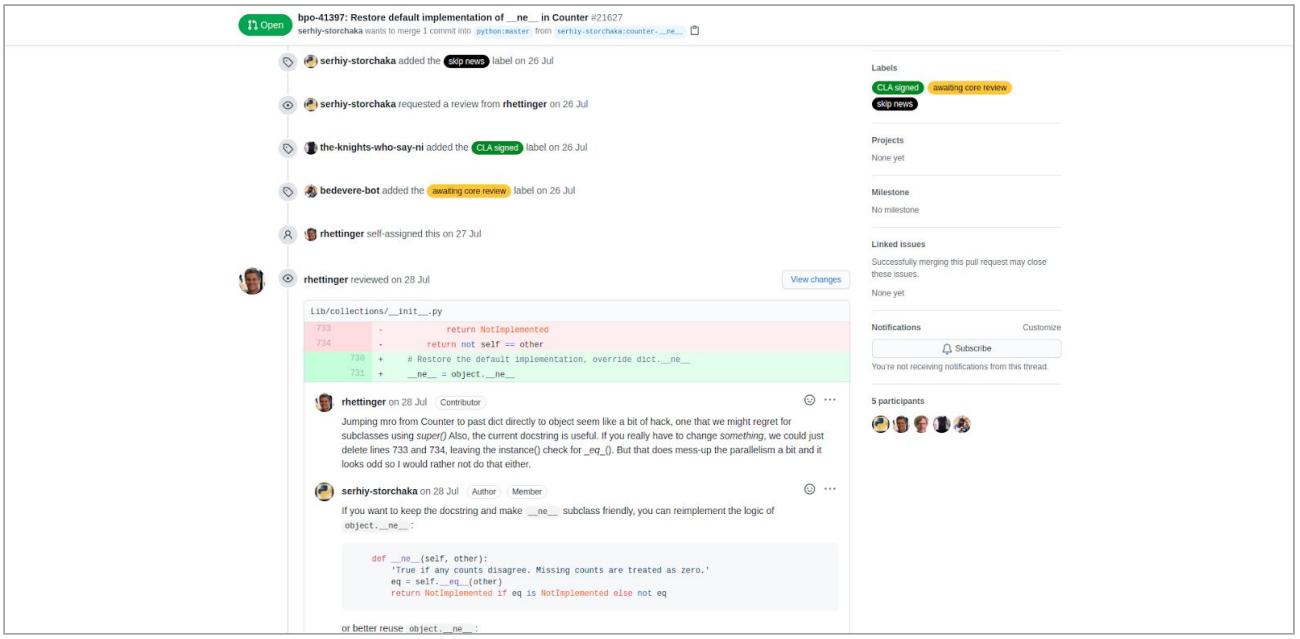


図 30 GitHub でのコミュニケーション

Python ファイル (付録 5) を任意のディレクトリに任意のファイル名 (本報告書では，cga.py とする) で保存する。Python ファイル (付録 5) の GitHub ID を分析したい FLOSS プロジェクトによって変更する。① 正常に実行されることによって FLOSS 参加者同士のコミュニケーションがネットワークグラフとして可視化されて出力される (図 31) 。

```
$ tree
.
├── communication_log.csv
└── sna.py
$ python3 sna.py ・・・ ①
```

図 31 ネットワークグラフ出力手順

上記の分析手順を，Python，OpenSSL，Flutter の FLOSS プロジェクトに対して実行し，グラフとして出力した結果 (図 32) を以下に示す。ノードは開発者 1 人を，エッジは 1 回以上のコミュニケーションを表している。各 FLOSS プロジェクトで，コミュニケーションの中心人物となる参加者が確認できる。また，相互的なコミュニケーションより一方的なコミュニケーションが多いことも確認できる。

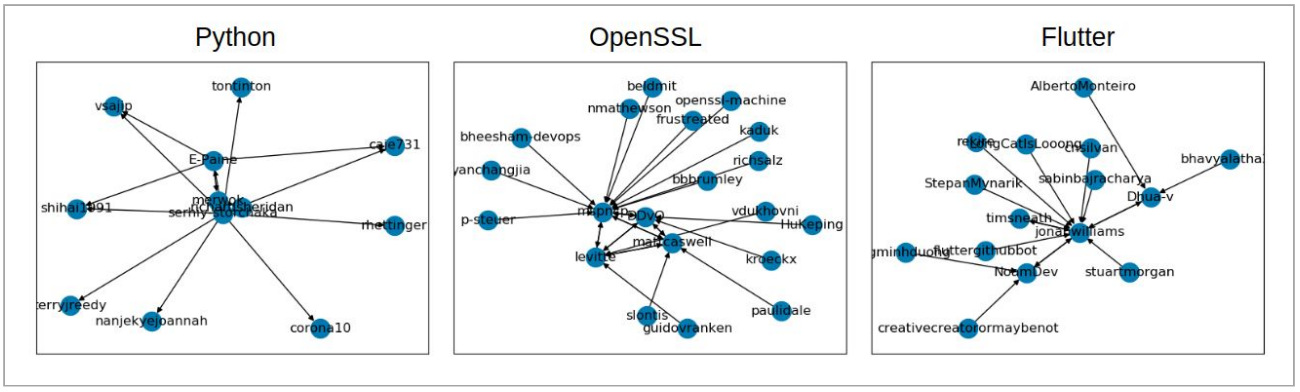


図 32 FLOSS 参加者のコミュニケーション分析結果

6. 考察

「3.1. 既存研究の収集」，「3.2. 問題の抽出」，「3.3. 既存研究の分類」より，FLOSS は多くの問題を抱えていることがわかった。特に，「3.3. 既存研究の分類」より，① FLOSS が抱える問題の多くは「品質」に関連していること，② FLOSS の「品質」にはソースコードベースの品質と組織構造ベースの品質が存在すること，③ ミクロな視点とマクロな視点で異なる問題が存在することがわかった。「3.3. 既存研究の分類」で得られた ① ～ ③ を基に 2 軸のマトリクスを作成し，FLOSS に関する既存研究を分析した。分析の結果，サイクロマティック複雑度やコードクローン率などのソースコードベースの品質ではなく，不具合修正時間や参加者の活動量などの組織構造ベースの品質に関する研究が活発に行われていることがわかった。また，ソースコードや参加者に焦点を合わせたミクロな研究は活発に行われているが，FLOSS のコミュニティや FLOSS のステークホルダなどを 1 つのノードとし，FLOSS をエコシステムとして捉えたマクロな研究は十分に行われていないこともわかった。

ソースコードベースの品質ではなく組織構造ベースの研究が活発に行われている理由として，ソースコードベースの品質は ISO/IEC 9126 や ISO/IEC 25000 (SQuaRE) などのソフトウェア品質における国際規格を流用可能であるため研究の新規性が低い。その一方，組織開発ベースの品質はプロプライエタリソフトウェアを前提としている既存の組織構造モデルや既存のソフトウェア開発プロセスモデルなどで説明できないため研究の新規性が高いなどの理由が考えられる。

プロプライエタリソフトウェアプロジェクト (再掲)

- 参加者は雇用元の指示により参加する
- ソフトウェアを開発する企業内メンバやパートナー会社に参加者は限られる
- 仕事としてプロジェクトに参加する
- 活動時間や活動量はプロジェクトマネージャによって決定される
- 意思決定はトップダウンで行われる
- プロジェクトマネージャ，リーダ，メンバなど階層化された組織構造が存在する

FLOSS プロジェクト (再掲)

- 誰でも自由に参加と脱退ができる
- 企業，個人など参加形態を問わない
- プロジェクトに参加することによる対価は基本的にない
- 活動時間や，その量は参加する人の意思による
- 意思決定は基本的な民主主義による合意の上行われる
- コミッタやコントリビュータなどの役割が定義されている

企業や非営利団体，フルタイムエンジニアなど，FLOSS を取り巻く新しいステークホルダの登場。また，GitHub や GitLab をベースとしたソーシャルコーディングの登場によって，FLOSS のエコシステムが形成されつつある。そのため，今後はソースコードや参加者に焦点を合わせたミクロな研究より組織に焦点を合わせたマクロな研究が活発に行われるようになると考えられる。

ここまでの分析より，マクロな視点かつ組織開発ベースの品質分析が FLOSS のエコシステム実現に大きく貢献ができる。なおかつ，研究としても新規性が高い分野であると推測できる。今後は，ソーシャルネットワーク分析などの社会学的アプローチで FLOSS のステークホルダを分析し，ステークホルダの特性や影響力などを特定していく。

7. まとめ

ここまで，本研究の背景と目的，FLOSS の概要，FLOSS に関する既存研究調査，分野を横断した組織構造分析手法の既存研究調査，FLOSS プロジェクトの理解を目的とした FLOSS 参加者の行動とコミュニケーション分析，FLOSS が抱える問題の考察について記述してきた。FLOSS の既存研究調査を通し，企業や非営利団体，フルタイムエンジニアの出現によるステークホルダの変化， GitHub や GitLab などをベースとしたソーシャルコーディングの登場による開発プロセスの変化やコミュニケーション手段の変化，これらの変化に伴い，FLOSS が抱える問題や開発プロセスも変化していることを把握できた。また，組織構造分析手法の既存研究調査を通し，組織構造分析手法 (クラスター分析，ソーシャルネットワーク分析，ABM，STAMP/STPA) の概要と分析によって得られる結果，様々な分野における適用事例を把握することができた。更に，FLOSS 参加者の行動履歴を対象したクラスター分析とコミュニケーション履歴を対象としたコミュニケーション分析を通し，FLOSS プロジェクト内で行っている開発プロセスやコミュニティ構造，コミュニケーション方法などに対する理解が深まった。最後に，FLOSS が抱える問題に対する理解を深めることで，本研究における方向性を明確にすることができた。今後は，本調査で得られた情報を基に，FLOSS を中心としたエコシステムの実現に向けて研究を進めていきたい。

参考文献

[1] Eric S. Raymond, The Cathedral and the Bazaar. O'Reilly Media, 1999.
[2] Synopsys, Inc., 2020 Open Source Security and Risk Analysis, https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2020-ossra-report.pdf, 2020, (Accessed: 2020-10-19).
[3] Talk Openly, Develop Openly Group, 2019 Open Source Program Survey Results, https://github.com/todogroup/survey/tree/master/2019, 2019, (Accessed: 2020-10-19).
[4] B. Lundell, B. Lings, E. Lindqvist, Perceptions and Uptake of Open Source in Swedish Organisations, Open Source Systems, Boston, MA, pp. 155–163, 2006.
[5] Ø. Hauge, C.-F. Sørensen, R. Conradi, Adoption of Open Source in the Software Industry, Open Source Development, Communities and Quality, Boston, MA, pp. 211–221, 2008.
[6] E. Capra, C. Francalanci, F. Merlo, C. Rossi Lamastra, A Survey on Firms' Participation in Open Source Community Projects, Open Source Ecosystems: Diverse Communities Interacting, Berlin, Heidelberg, pp. 225–236, 2009.
[7] T. Noda, T. Tansho, Open Source Introducing Policy and Promotion of Regional Industries in Japan, Open Source Software: New Horizons, Berlin, Heidelberg, pp. 214–223, 2010.
[8] K. Henttonen, Libre Software as an Innovation Enabler in India Experiences of a Bangalorian Software SME, Open Source Systems: Grounding Research, Berlin, Heidelberg, pp. 220–232, 2011.
[9] T. Noda, T. Tansho, S. Coughlan, Standing Situations and Issues of Open Source Policy in East Asian Nations: Outcomes of Open Source Research Workshop of East Asia, Open Source Systems: Grounding Research, Berlin, Heidelberg, pp. 379–384, 2011.
[10] A. Mavridis, D. Fotakidis, I. Stamelos, Open Source Migration in Greek Public Sector: A Feasibility Study, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 233–243, 2012.
[11] G. Pinto and F. Kamei, The Census of the Brazilian Open-Source Community, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 202–211, 2014.
[12] D. Tosi, L. Lavazza, S. Morasca, M. Chiappa, Surveying the Adoption of FLOSS by Public Administration Local Organizations, Open Source Systems: Adoption and Impact, Cham, pp. 114–123, 2015.
[13] S. Koloniaris, G. Kousiouris, M. Nikolaidou, Possibilities of Use of Free and Open Source Software in the Greek Local Authorities, Open Source Systems: Enterprise Software and Solutions, Cham, pp. 128–143, 2018.
[14] T. Percy, J.-P. Van Belle, Exploring the Barriers and Enablers to the Use of Open Educational Resources by University Academics in Africa, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 112–128, 2012.
[15] G. R. Gangadharan, M. Butler, Free and Open Source Software Adoption in Emerging Markets: An Empirical Study in the Education Sector, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 244–249, 2012.
[16] B. Lundell, J. Gamalielsson, Open Standards and Open Source in Swedish Schools: On Promotion of Openness and Transparency, Open Source Software: Quality Verification, Berlin, Heidelberg, pp. 207–221, 2013.
[17] S. R. Montes León, G. Robles, J. M. González-Barahona, L. E. Sánchez C., Considerations Regarding the Creation of a Post-graduate Master's Degree in Free Software, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 123–132, 2014.
[18] A. Coman, A. Cîtea, S. C. Buraga, Towards Open Source/Data in the Context of Higher Education: Pragmatic Case Studies Deployed in Romania, Open Source Systems: Integrating Communities, Cham, pp. 184–191, 2016.
[19] D. M. C. Nascimento, C. von Flach Garcia Chavez, R. A. Bittencourt, Does FLOSS in Software Engineering Education Narrow the Theory-Practice Gap? A Study Grounded on Students' Perception, Open Source Systems, Cham, pp. 153–164, 2019.
[20] B. Morgan, G. W. Hislop, H. J. C. Ellis, Faculty Development for FLOSS Education, Open Source Systems, Cham, pp. 165–171, 2019.

[21] MITRE, CVE-2014-0160, https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0160, 2014, (Accessed: 2020-10-19).
[22] Netcraft, Half a million widely trusted websites vulnerable to Heartbleed bug, https://news.netcraft.com/archives/2014/04/08/half-a-million-widely-trusted-websites-vulnerable-to-heartbleed-bug.html, 2014, (Accessed: 2020-10-19).
[23] Pew Research Center, Heartbleed's Impact, https://www.pewresearch.org/internet/2014/04/30/heartbleeds-impact/2/#main-findings, 2014, (Accessed: 2020-10-19).
[24] Z. Durumeric, The Matter of Heartbleed, Proceedings of the 2014 Conference on Internet Measurement Conference, New York, NY, USA, pp. 475–488, 2014.
[25] Ben Grubb, Man who introduced serious 'Heartbleed' security flaw denies he inserted it deliberately, https://www.smh.com.au/technology/man-who-introduced-serious-heartbleed-security-flaw-denies-he-inserted-it-deliberately-20140410-zqta1.html, The Sydney Morning Herald, 2014, (Accessed: 2020-10-19).
[26] Richard M. Stallman, FLOSS and FOSS, https://www.gnu.org/philosophy/floss-and-foss.html, 2016, (Accessed: 2020-10-19).
[27] Free Software Foundation, What is free software?, https://www.gnu.org/philosophy/free-sw.en.html, 2019, (Accessed: 2020-10-19).
[28] Debian, Debian Social Contract, https://www.debian.org/social_contract.en.html, 2004, (Accessed: 2020-10-19).
[29] Richard M. Stallman, https://www.gnu.org/bulletins/bull1.txt, GNU's Bulletin, Volume 1 Number 1, pp. 8, 1986, (Accessed: 2020-10-19).
[30] Debian, Debian Social Contract, Version 1.0, https://www.debian.org/social_contract.1.0.en, 1997, (Accessed: 2020-10-19).
[31] Open Source Initiative, The Open Source Definition, https://opensource.org/osd, 2007, (Accessed: 2020-10-19).
[32] Richard M. Stallman, Free Software Is Even More Important Now, https://www.gnu.org/philosophy/free-software-even-more-important.en.html, 2020, (Accessed: 2020-10-19).
[33] W3Techs, Usage Statistics and Market Share of Web Servers, https://w3techs.com/technologies/overview/web_server, 2020, (Accessed: 2020-10-19).
[34] Google, Blink: A rendering engine for the Chromium project, https://blog.chromium.org/2013/04/blink-rendering-engine-for-chromium.html, Chromium Blog, 2013, (Accessed: 2020-10-19).
[35] Fossbytes, Difference Between Google Chrome And Chromium Browser, https://fossbytes.com/difference-google-chrome-vs-chromium-browser/, 2018, (Accessed: 2020-10-19).
[36] StatCounter, Browser Market Share Worldwide, https://gs.statcounter.com/browser-market-share, 2020, (Accessed: 2020-10-19).
[37] Mozilla, Gecko, https://developer.mozilla.org/en-US/docs/Mozilla/Gecko, MDN, 2020, (Accessed: 2020-10-19).
[38] L. Torvalds, D. Diamond, Just for Fun: The Story of an Accidental Revolutionary. HarperInformation, 2001.
[39] W3Techs, Usage statistics of Linux for websites, https://w3techs.com/technologies/details/os-linux, 2020, (Accessed: 2020-10-19).
[40] Karanbir Singh, CentOS Project joins forces with Red Hat, https://lists.centos.org/pipermail/centos-announce/2014-January/020100.html, 2014, (Accessed: 2020-10-19).
[41] StatCounter Global Stats, Mobile Operating System Market Share Worldwide, https://gs.statcounter.com/os-market-share/mobile/, 2020, (Accessed: 2020-10-19).
[42] OpenSSL Software Foundation, Changelog, https://www.openssl.org/news/changelog.html, 2020, (Accessed: 2020-10-19).
[43] OpenSSL Software Foundation, OpenSSL Repository, https://github.com/openssl/openssl/commits/master?after=012903063900340b972a6a8d20c0a18c37a89428+27235&branch=master, Github, 2020, (Accessed: 2020-10-19).

[44] DB-Engines, DB-Engines Ranking, https://db-engines.com/en/ranking, 2020, (Accessed: 2020-10-19).
[45] MariaDB, MariaDB Customer Stories, https://mariadb.com/ja/resources/customer-stories/, 2020, (Accessed: 2020-10-19).
[46] MongoDB, About Us, https://www.mongodb.com/company, 2020, (Accessed: 2020-10-19).
[47] MongoDB, Our Customers, https://www.mongodb.com/who-uses-mongodb, 2020, (Accessed: 2020-10-19).
[48] MySQL, MySQL Customers, https://www.mysql.com/customers/, 2020, (Accessed: 2020-10-19).
[49] PostgreSQL, A Brief History of PostgreSQL, https://www.postgresql.org/docs/current/history.html, 2020, (Accessed: 2020-10-19).
[50] Dropbox, Thank you, Guido, https://blog.dropbox.com/topics/company/thank-you--guido, Dropbox Blog, 2019, (Accessed: 2020-10-19).
[51] 文部科学省, 高等学校情報科「情報Ⅱ」教員研修用教材(本編), https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1416746.htm, 2020, (Accessed: 2020-10-19).
[52] Ruby Association, Ruby活用事例, https://www.ruby.or.jp/ja/showcase/, 2020, (Accessed: 2020-10-19).
[53] Docker, Customers, https://www.docker.com/customers, 2020, (Accessed: 2020-10-19).
[54] OpenStack, Use Cases - Applications of the Technology, https://www.openstack.org/use-cases/, 2020, (Accessed: 2020-10-19).
[55] M. Goeminne, T. Mens, Evidence for the pareto principle in open source software activity, In the Joint Proceedings of the 1st International workshop on Model Driven Software Maintenance and 5th International Workshop on Software Quality and Maintainability, pp. 74–82, 2011.
[56] A. Barcomb, A. Kaufmann, D. Riehle, K.-J. Stol, B. Fitzgerald, Uncovering the Periphery: A Qualitative Survey of Episodic Volunteering in Free/Libre and Open Source Software Communities, IEEE Transactions on Software Engineering, vol. 46, no. 9, pp. 962–980, 2020.
[57] A. Barcomb, K. Stol, D. Riehle, and B. Fitzgerald, Why Do Episodic Volunteers Stay in FLOSS Communities?, 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), pp. 948–959, 2019.
[58] C. A. Amrit and J. van Hillegersberg, Exploring the impact of socio-technical core-periphery structures in open source software development, Journal of information technology, vol. 25, no. 2, pp. 216–229, 2010.
[59] H. Masmoudi, M. den Besten, C. de Loupy, J.-M. Dalle, “Peeling the Onion”, Open Source Ecosystems: Diverse Communities Interacting, Berlin, Heidelberg, 2009, pp. 284–297.
[60] K. Crowston and I. Shamshurin, Core-Periphery Communication and the Success of Free/Libre Open Source Software Projects, Open Source Systems: Integrating Communities, Cham, pp. 45–56, 2016.
[61] T. Kilamo, T. Aaltonen, and T. J. Heinimäki, BULB: Onion-Based Measuring of OSS Communities, Open Source Software: New Horizons, Berlin, Heidelberg, pp. 342–347, 2010.
[62] B. Lundell et al., Addressing Lock-in, Interoperability, and Long-Term Maintenance Challenges Through Open Source: How Can Companies Strategically Use Open Source?, Open Source Systems: Towards Robust Practices, Cham, pp. 80–88, 2017.
[63] J. Lindman, Y. Tammisto, Open Source and Open Data: Business Perspectives from the Frontline, Open Source Systems: Grounding Research, Berlin, Heidelberg, pp. 330–333, 2011.
[64] J. Lindman, J.-P. Juutilainen, M. Rossi, Beyond the Business Model: Incentives for Organizations to Publish Software Source Code, Open Source Ecosystems: Diverse Communities Interacting, Berlin, Heidelberg, pp. 47–56, 2009.
[65] J. Teixeira, J. Salminen, Open-Source Software Entrepreneurial Business Modelling, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 80–82, 2014.
[66] M. Müller, C. Schindler, W. Slany, Introducing Agile Product Owners in a FLOSS Project, Open Source Systems, Cham, pp. 38–43, 2019.
[67] Ø. Hauge, S. Ziemer, Providing Commercial Open Source Software: Lessons Learned, Open Source Ecosystems: Diverse Communities Interacting, Berlin, Heidelberg, pp. 70–82, 2009.
[68] S. J. Deodhar, K. C. Saxena, M. Ruohonen, Hybrid Business Models in Software Product Industry: Patterns and Challenges, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 362–367, 2012.

[69] T. Noda, T. Tansho, A Study of the Effect on Business Growth by Utilization and Contribution of Open Source Software in Japanese IT Companies, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 216–217, 2014.
[70] T. Noda, T. Tansho, S. Coughlan, Effect on Business Growth by Utilization and Contribution of Open Source Software in Japanese IT Companies, Open Source Software: Quality Verification, Berlin, Heidelberg, pp. 222–231, 2013.
[71] B. Schwab, D. Riehle, A. Barcomb, N. Harutyunyan, The Ecosystem of openKONSEQUENZ, A User-Led Open Source Foundation, Open Source Systems, Cham, pp. 1–13, 2020.
[72] D. Riehle, S. Berschneider, A Model of Open Source Developer Foundations, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 15–28, 2012.
[73] J. L. C. Izquierdo, J. Cabot, The Role of Foundations in Open Source Projects, Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Society, New York, NY, USA, pp. 3–12, 2018.
[74] J. Lindman, I. Hammouda, Investigating Relationships Between FLOSS Foundations and FLOSS Projects, Open Source Systems: Towards Robust Practices, Cham, pp. 14–22, 2017.
[75] J. Lindman, I. Hammouda, Support mechanisms provided by FLOSS foundations and other entities, Journal of Internet Services and Applications, vol. 9, no. 1, p. 8, Feb. 2018.
[76] E. Berdou, Insiders and outsiders: paid contributors and the dynamics of cooperation in community led F/OS projects, Open Source Systems, Boston, MA, pp. 201–208, 2006.
[77] A. Nguyen Duc, D. S. Cruzes, C. Ayala, R. Conradi, Impact of Stakeholder Type and Collaboration on Issue Resolution Time in OSS Projects, Open Source Systems: Grounding Research, Berlin, Heidelberg, pp. 1-16, 2011.
[78] D. Riehle, P. Riemer, C. Kolassa, M. Schmidt, Paid vs. Volunteer Work in Open Source, 2014 47th Hawaii International Conference on System Sciences, pp. 3286–3295, 2014.
[79] M. Claes, M. Mäntylä, M. Kuutila, U. Farooq, Towards Automatically Identifying Paid Open Source Developers, Proceedings of the 15th International Conference on Mining Software Repositories, New York, NY, USA, pp. 437–441, 2018.
[80] Open Source Initiative, Licenses by Name, https://opensource.org/licenses/alphabetical, 2020, (Accessed: 2020-10-19).
[81] Free Software Foundation, Various Licenses and Comments about Them, https://www.gnu.org/licenses/license-list.html.en, 2020, (Accessed: 2020-10-19).
[82] D. Skidmore, Stakeholder value, usage, needs and obligations from different types of F/LOSS licenses, Open Source Development, Adoption and Innovation, Boston, MA, pp. 343–348, 2007.
[83] D. M. German, J. M. González-Barahona, An Empirical Study of the Reuse of Software Licensed under the GNU General Public License, Open Source Ecosystems: Diverse Communities Interacting, Berlin, Heidelberg, pp. 185–198, 2009.
[84] D. M. German, M. D. Penta, J. Davies, Understanding and Auditing the Licensing of Open Source Software Distributions, 2010 IEEE 18th International Conference on Program Comprehension, pp. 84–93, 2010.
[85] C. Jensen, W. Scacchi, License Update and Migration Processes in Open Source Software Projects, Open Source Systems: Grounding Research, Berlin, Heidelberg, pp. 177–195, 2011.
[86] G. Hofmann, D. Riehle, C. Kolassa, W. Mauerer, A Dual Model of Open Source License Growth, Open Source Software: Quality Verification, Berlin, Heidelberg, pp. 245–256, 2013.
[87] Y. Manabe, D. M. German, K. Inoue, Analyzing the Relationship between the License of Packages and Their Files in Free and Open Source Software, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 51–60, 2014.
[88] T. Maryka, D. M. German, G. Poo-Caamaño, On the Variability of the BSD and MIT Licenses, Open Source Systems: Adoption and Impact, Cham, pp. 146–156, 2015.
[89] R. Viseur, G. Robles, First Results About Motivation and Impact of License Changes in Open Source Projects, Open Source Systems: Adoption and Impact, Cham, pp. 137–145, 2015.

[90] C. D. dos Santos, Changes in free and open source software licenses: managerial interventions and variations on project attractiveness, Journal of Internet Services and Applications, vol. 8, no. 1, p. 11, Aug. 2017.
[91] Y. Wu, Y. Manabe, D. M. German, K. Inoue, How are Developers Treating License Inconsistency Issues? A Case Study on License Inconsistency Evolution in FOSS Projects, Open Source Systems: Towards Robust Practices, Cham, pp. 69–79, 2017.
[92] O. Fendt, M. C. Jaeger, Open Source for Open Source License Compliance, Open Source Systems, Cham, pp. 133–138, 2019.
[93] 独立行政法人情報処理推進機構, OSS ライセンスの比較および利用動向ならびに係争に関する調査, 2010.
[94] Brett Smith, A Quick Guide to GPLv3, https://www.gnu.org/licenses/quick-guide-gplv3.en.html, 2014, (Accessed: 2020-10-19).
[95] Free Software Foundation, GNU Affero General Public License, https://www.gnu.org/licenses/agpl-3.0.en.html, 2016, (Accessed: 2020-10-19).
[96] European Union, European Union Public License 1.2, https://spdx.org/licenses/EUPL-1.2.html, 2016, (Accessed: 2020-10-19).
[97] Free Software Foundation, GNU Lesser General Public License, https://www.gnu.org/licenses/lgpl-3.0.en.html, 2016, (Accessed: 2020-10-19).
[98] Mozilla Foundation, Mozilla Public License, https://www.mozilla.org/en-US/MPL/, 2012, (Accessed: 2020-10-19).
[99] Apache Software Foundation, Apache License 2.0, https://www.apache.org/licenses/LICENSE-2.0, 2004 , (Accessed: 2020-10-19).
[100] Open Source Initiative, The 3-Clause BSD License, https://opensource.org/licenses/BSD-3-Clause, 2020, (Accessed: 2020-10-19).
[101] Open Source Initiative, The MIT License, https://opensource.org/licenses/MIT, 2020, (Accessed: 2020-10-19).
[102] F. Mulazzani, B. Rossi, B. Russo, M. Steff, Building Knowledge in Open Source Software Research in Six Years of Conferences, Open Source Systems: Grounding Research, Berlin, Heidelberg, pp. 123–141, 2011.
[103] M. Bergquist, J. Ljungberg, B. Rolandsson, A Historical Account of the Value of Free and Open Source Software: From Software Commune to Commercial Commons, Open Source Systems: Grounding Research, Berlin, Heidelberg, pp. 196–207, 2011.
[104] 伊原彰紀, 大平雅雄, 「オープンソースソフトウェア工学」シリーズ オープンソースソフトウェア工学, コンピュータ ソフトウェア, vol. 33, no. 1, pp. 28–40, 2016.
[105] F. Bordeleau, P. Meirelles, A. Sillitti, Fifteen Years of Open Source Software Evolution, Open Source Systems, Cham, pp. 61–67, 2019.
[106] K. Ven, J. Verelst, The Organizational Adoption of Open Source Server Software by Belgian Organizations, Open Source Systems, Boston, MA, pp. 111–122, 2006.
[107] K. Ven, D. Van Nuffel, J. Verelst, The Introduction of OpenOffice.org in the Brussels Public Administration, Open Source Systems, Boston, MA, pp. 123–134, 2006.
[108] B. Rossi, B. Russo, G. Succi, A study on the introduction of Open Source Software in the Public Administration, Open Source Systems, Boston, MA, pp. 165–171, 2006.
[109] D. Brink, L. Roos, J. Weller, J.-P. Van Belle, Critical Success Factors for Migrating to OSS-on-the-Desktop: Common Themes across Three South African Case Studies, Open Source Systems, Boston, MA, pp. 287–293, 2006.
[110] E. Rosales Rosa, A. A. Fírvida Donéstevez, M. González Muño, A. P. Fuentes, Smart TV with Free Technologies in Support of Teaching-Learning Process, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 147–152, 2014.
[111] J. G. Cruz, A. Sadovykh, D. Truscan, H. Bruneliere, P. Pierini, L. L. Muñiz, MegaM@Rt2 EU Project: Open Source Tools for Mega-Modelling at Runtime of CPSs, Open Source Systems, Cham, pp. 183–189, 2020.
[112] F. Di Cerbo, G. Dodero, G. Succi, Social Networking Technologies for Free-Open Source E-Learning Systems, Open Source Development, Communities and Quality, Boston, MA, pp. 289–297, 2008.

[113] C. A. Ardagna, E. Damiani, F. Frati, S. Oltolina, M. Regoli, G. Ruffatti, Spago4Q and the QEST nD Model: An Open Source Solution for Software Performance Measurement, Open Source Software: New Horizons, Berlin, Heidelberg, pp. 1–14, 2010.
[114] I. Armuelles Voinov, A. C. Cedeño, J. Chung, G. González, A Performance Analysis of Wireless Mesh Networks Implementations Based on Open Source Software, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 107–110, 2014.
[115] J. M. Koo, J. P. Espino, I. Armuelles, R. Villarreal, Use of Open Software Tools for Data Offloading Techniques Analysis on Mobile Networks, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 111–112, 2014.
[116] C. L. B. Possamai et al., PROINFODATA: Monitoring a Large Park of Computational Laboratories, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 226–229, 2014.
[117] B. J. P. Quezada, J. Fernández, Automation of Agricultural Irrigation System with Open Source, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 232–233, 2014.
[118] C. Berger, An Open Continuous Deployment Infrastructure for a Self-driving Vehicle Ecosystem, Open Source Systems: Integrating Communities, Cham, pp. 177–183, 2016.
[119] I. Routis, A. Tsadimas, M. Nikolaidou, Building a Social Platform Using FLOSS to Support Collaborative Communities: The ReWeee Case Study, Open Source Systems: Enterprise Software and Solutions, Cham, pp. 171–180, 2018.
[120] F. M. Mendes, R. Poppi, H. Parra, B. Moreira, EJ: A Free Software Platform for Social Participation, Open Source Systems, Cham, pp. 27–37, 2019.
[121] S. Schork, F. Zahid, D. Pradhan, S. Kicin, A. Schwichtenberg, Building an Open-Source Cross-Cloud DevOps Stack for a CRM Enterprise Application: A Case Study, Open Source Systems, Cham, pp. 3–11, 2019.
[122] D. Atonge et al., The Development of Data Collectors in Open-Source System for Energy Efficiency Assessment, Open Source Systems, Cham, pp. 14–24, 2020.
[123] S. Ergasheva, V. Ivanov, I. Khomyakov, A. Kruglov, D. Strugar, G. Succi, InnoMetrics Dashboard: The Design, Implementation of the Adaptable Dashboard for Energy-Efficient Applications Using Open Source Tools, Open Source Systems, Cham, pp. 163–176, 2020.
[124] D. Mukhamedshin, O. Nevzorova, A. Kirillovich, Using FLOSS for Storing, Processing and Linking Corpus Data, Open Source Systems, Cham, pp. 177–182, 2020.
[125] V. N. Kruglov, Using Open Source Libraries in the Development of Control Systems Based on Machine Vision, Open Source Systems, Cham, pp. 70–77, 2020.
[126] K. Aksyonov, O. Aksyonova, A. Antonova, E. Aksyonova, P. Ziomkovskaya, Development of Cloud-Based Microservices to Decision Support System, Open Source Systems, Cham, pp. 87–97, 2020.
[127] A. Tarasiev, M. Filippova, K. Aksyonov, O. Aksyonova, A. Antonova, Using of Open-Source Technologies for the Design and Development of a Speech Processing System Based on Stemming Methods, Open Source Systems, Cham, pp. 98–105, 2020.
[128] T. Koponen, Life cycle of Defects in Open Source Software Projects, Open Source Systems, Boston, MA, pp. 195–200, 2006.
[129] A. Capiluppi, M. Michlmayr, From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects, Open Source Development, Adoption and Innovation, Boston, MA, pp. 31–44, 2007.
[130] J.-M. Dalle, M. den Besten, Different Bug Fixing Regimes? A Preliminary Case for Superbugs, Open Source Development, Adoption and Innovation, Boston, MA, pp. 247–252, 2007.
[131] C. Francalanci, F. Merlo, Empirical Analysis of the Bug Fixing Process in Open Source Projects, Open Source Development, Communities and Quality, Boston, MA, pp. 187–196, 2008.
[132] J.-M. Dalle, M. den Besten, H. Masmoudi, Channeling Firefox Developers: Mom and Dad Aren’t Happy Yet, Open Source Development, Communities and Quality, Boston, MA, pp. 265–271, 2008.

[133] B. Rossi, B. Russo, G. Succi, Analysis of Open Source Software Development Iterations by Means of Burst Detection Techniques, Open Source Ecosystems: Diverse Communities Interacting, Berlin, Heidelberg, pp. 83–93, 2009.
[134] D. Izquierdo-Cortázar, G. Robles, J. M. González-Barahona, Do More Experienced Developers Introduce Fewer Bugs?, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 268–273, 2012.
[135] A. JONGYINDEE, M. OHIRA, A. IHARA, K. MATSUMOTO, Good or Bad Committers? —— A Case Study of Committer's Activities on the Eclipse's Bug Fixing Process, IEICE Transactions on Information and Systems, vol. E95.D, no. 9, pp. 2202–2210, 2012.
[136] 伊原彰紀, 大平雅雄, 松本健一, OSS 開発における不具合修正プロセスの現状と課題：不具合修正時間の短縮化へ向けた分析, 情報社会学会誌, vol. 6, no. 2, pp. 5–16, 2012.
[137] E. Kouzari, L. Sotiriadis, I. Stamelos, Process Mining for Process Conformance Checking in an OSS Project: An Empirical Research, Open Source Systems: Enterprise Software and Solutions, Cham, pp. 79–89, 2018.
[138] R. Hewett, P. Kijana Thin, On modeling software defect repair time, Empirical Software Engineering, vol. 14, no. 2, p. 165, May 2008.
[139] P. Bhattacharya and I. Neamtiu, Bug-Fix Time Prediction Models: Can We Do Better?, Proceedings of the 8th Working Conference on Mining Software Repositories, New York, NY, USA, pp. 207–210, 2011.
[140] 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て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, vol. 54, no. 2, pp. 933–944, Feb. 2013.
[141] A. Chou, J. Yang, B. Chelf, S. Hallem, D. Engler, An Empirical Study of Operating Systems Errors, Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles, New York, NY, USA, pp. 73–88, 2001.
[142] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, C. Weiss, What Makes a Good Bug Report?, IEEE Transactions on Software Engineering, vol. 36, no. 5, pp. 618–643, 2010.
[143] X. Wang, L. Zhang, T. Xie, J. Anvik, J. Sun, An approach to detecting duplicate bug reports using natural language and execution information ACM/IEEE 30th International Conference on Software Engineering, 2008, pp. 461–470, 2008.
[144] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, C. Sun, 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, New York, NY, USA, pp. 70–79, 2012.
[145] F. Thung, P. S. Kochhar, D. Lo, DupFinder: Integrated Tool Support for Duplicate Bug Report Detection, Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, New York, NY, USA, pp. 871–874, 2014.
[146] P. Hooimeijer, W. Weimer, Modeling Bug Report Quality, Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering, New York, NY, USA, pp. 34–43, 2007.
[147] G. Rodríguez-Pérez, J. M. Gonzalez-Barahona, G. Robles, D. Dalipaj, N. Sekitoleko, BugTracking: A Tool to Assist in the Identification of Bug Reports, Open Source Systems: Integrating Communities, Cham, pp. 192–198, 2016.
[148] Y. Kashiwa, A. Ihara, M. Ohira, What Are the Perception Gaps Between FLOSS Developers and SE Researchers?, Open Source Systems, Cham, pp. 44–57, 2019.
[149] G. Jeong, S. Kim, T. Zimmermann, Improving Bug Triage with Bug Tossing Graphs, Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, New York, NY, USA, pp. 111–120, 2009.
[150] G. Canfora, L. Cerulo, Supporting Change Request Assignment in Open Source Development, Proceedings of the 2006 ACM Symposium on Applied Computing, New York, NY, USA, pp. 1767–1772, 2006.
[151] A. H. Moin, M. Khansari, Bug Localization Using Revision Log Analysis and Open Bug Repository Text Categorization, Open Source Software: New Horizons, Berlin, Heidelberg, pp. 188–199, 2010.
[152] A. T. Nguyen, T. T. Nguyen, J. Al-Kofahi, H. V. Nguyen, T. N. Nguyen, A topic-based approach for narrowing the search space of buggy files from a bug report, 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp. 263–272, 2011.

[153] C. Tantithamthavorn, R. Teekavanich, A. Ihara, K. Matsumoto, Mining A change history to quickly identify bug locations : A case study of the Eclipse project, 2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 108–113, 2013.
[154] S. FUJITA, An analysis of committers toward improving the patch review process in OSS development, Proc. 21st IEEE Int. Symp, Software Reliability Engineering (ISSRE' 10), pp. 369–374, 2010.
[155] J. M. González-Barahona, D. Izquierdo-Cortázar, G. Robles, M. Gallegos, Code Review Analytics: WebKit as Case Study, in Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 1–10, 2014.
[156] T. Hirao, A. Ihara, Y. Ueda, P. Phannachitta, K. Matsumoto, The Impact of a Low Level of Agreement Among Reviewers in a Code Review Process, in Open Source Systems: Integrating Communities, Cham, pp. 97–110, 2016.
[157] A. Alami, M. L. Cohn, A. Wąsowski, Why Does Code Review Work for Open Source Software Communities?, in 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), pp. 1073–1083, 2019.
[158] S. Andrade, F. Saraiva, Principled Evaluation of Strengths and Weaknesses in FLOSS Communities: A Systematic Mixed Methods Maturity Model Approach, Open Source Systems: Towards Robust Practices, Cham, pp. 34–46, 2017.
[159] M. Syeed, J. Lindman, I. Hammouda, Measuring Perceived Trust in Open Source Software Communities, Open Source Systems: Towards Robust Practices, Cham, pp. 49–54, 2017.
[160] L. Nyman, T. Mikkonen, To Fork or Not to Fork: Fork Motivations in SourceForge Projects, Open Source Systems: Grounding Research, Berlin, Heidelberg, pp. 259–268, 2011.
[161] G. Robles, J. M. González-Barahona, A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 1–14, 2012.
[162] J. Gamalielsson, B. Lundell, Long-Term Sustainability of Open Source Software Communities beyond a Fork: A Case Study of LibreOffice, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 29–47, 2012.
[163] L. Nyman, T. Mikkonen, J. Lindman, and M. Fougère, Perspectives on Code Forking and Sustainability in Open Source Software, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 274–279, 2012.
[164] M. Gençer, B. Özel, Forking the Commons: Developmental Tensions and Evolutionary Patterns in Open Source Software, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 310–315, 2012.
[165] A. "Emerson" Azarbakht, C. Jensen, Longitudinal Analysis of the Run-up to a Decision to Break-up (Fork) in a Community, Open Source Systems: Towards Robust Practices, Cham, pp. 204–217, 2017.
[166] E. Berdou, Learning and the imperative of production in Free/Open Source development, Open Source Development, Adoption and Innovation, Boston, MA, pp. 235–240, 2007.
[167] I. Steinmacher, M. A. G. Silva, M. A. Gerosa, Barriers Faced by Newcomers to Open Source Projects: A Systematic Review, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 153–163, 2014.
[168] I. Steinmacher, M. A. Gerosa, How to Support Newcomers Onboarding to Open Source Software Projects, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 199–201, 2014.
[169] Y. Ye, K. Kishida, Toward an Understanding of the Motivation Open Source Software Developers, Proceedings of the 25th International Conference on Software Engineering, USA, pp. 419–429, 2003.
[170] A. Schofield, G. S. Cooper, Participation in Free and Open Source Communities: An Empirical Study of Community Members' Perceptions, Open Source Systems, Boston, MA, pp. 221–231, 2006.
[171] P. A. David, J. S. Shapiro, Community-based production of open-source software: What do we know about the developers who participate?, Information Economics and Policy, vol. 20, no. 4, pp. 364–398, 2008.
[172] F. Fronchetti, I. Wiese, G. Pinto, I. Steinmacher, What Attracts Newcomers to Onboard on OSS Projects? TL;DR: Popularity, Open Source Systems, Cham, pp. 91–103, 2019.
[173] P. N. Sharma, J. Hulland, and S. Daniel, Examining Turnover in Open Source Software Projects Using Logistic Hierarchical Linear Modeling Approach, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 331–337, 2012.

[174] A. Rastogi and A. Sureka, Does Contributor Characteristics Influence Future Participation? A Case Study on Google Chromium Issue Tracking System, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 164–167, 2014.
[175] C. Miller, D. G. Widder, C. Kästner, and B. Vasilescu, Why Do People Give Up FLOSSing? A Study of Contributor Disengagement in Open Source, Open Source Systems, Cham, pp. 116–129, 2019.
[176] A. E. Adam, Hacking into Hacking: Gender and the Hacker Phenomenon, SIGCAS Comput. Soc., vol. 33, no. 4, p. 3, 2004.
[177] V. Kuechler, C. Gilbertson, C. Jensen, Gender Differences in Early Free and Open Source Software Joining Process, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 78–93, 2012.
[178] G. Robles, L. A. Reina, J. M. González-Barahona, S. D. Domínguez, Women in Free/Libre/Open Source Software: The Situation in the 2010s, Open Source Systems: Integrating Communities, Cham, pp. 163–173, 2016.
[179] V. Singh, Women Participation in Open Source Software Communities, Proceedings of the 13th European Conference on Software Architecture - Volume 2, New York, NY, USA, pp. 94–99, 2019.
[180] H. S. Qiu, A. Nolte, A. Brown, A. Serebrenik, B. Vasilescu, Going Farther Together: The Impact of Social Capital on Sustained Participation in Open Source, 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), pp. 688–699, 2019.
[181] N. Imtiaz, J. Middleton, J. Chakraborty, N. Robson, G. Bai, E. Murphy-Hill, Investigating the Effects of Gender Bias on GitHub, 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), pp. 700–711, 2019.
[182] Y. Qiu, K. J. Stewart, K. M. Bartol, Joining and Socialization in Open Source Women's Groups: An Exploratory Study of KDE-Women, Open Source Software: New Horizons, Berlin, Heidelberg, pp. 239–251, 2010.
[183] V. Singh, W. Brandon, Open Source Software Community Inclusion Initiatives to Support Women Participation, Open Source Systems, Cham, pp. 68–79, 2019.
[184] F.-W. Duijnhouwer, C. Widdows, Capgemini Expert Letter Open Source Maturity Model, Capgemini, pp. 1–18, 2003.
[185] Navica Inc., The Open Source Maturity Model is a vital tool for planning open source success, http://www.navicasoft.com/pages/osmm.htm, 2009, (Accessed: 2020-10-19).
[186] Origin, A, Method for Qualification and Selection of Open Source Software (QSOS), http://www.qsos.org, 2004, (Accessed: 2020-10-19).
[187] Wasserman, M.P., Chan, C., Business Readiness Rating Project, BRR Whitepaper 2005 RFC 1, http://www.openbrr.org/wiki/images/d/da/BRR_whitepaper_2005RFC1.pdf, 2005, (Accessed: 2020-10-19).
[188] E. Petrinja, R. Nambakam, A. Sillitti, Introducing the OpenSource Maturity Model, 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, pp. 37–41, 2009.
[189] A. Bosu, J. C. Carver, M. Hafiz, P. Hilley, D. Janni, When Are OSS Developers More Likely to Introduce Vulnerable Code Changes? A Case Study, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 234–236, 2014.
[190] S. Mirhosseini, C. Parnin, Can Automated Pull Requests Encourage Software Developers to Upgrade Out-of-Date Dependencies?, Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, Urbana-Champaign, IL, USA, pp. 84–94, 2017.
[191] R. G. Kula, D. M. German, A. Ouni, T. Ishio, K. Inoue, Do Developers Update Their Library Dependencies?, Empirical Softw. Engg., vol. 23, no. 1, pp. 384–417, Feb. 2018.
[192] A. Decan, T. Mens, E. Constantinou, On the Impact of Security Vulnerabilities in the Npm Package Dependency Network, Proceedings of the 15th International Conference on Mining Software Repositories, New York, NY, USA, pp. 181–191, 2018.
[193] B. Carlson, K. Leach, D. Marinov, M. Nagappan, A. Prakash, Open Source Vulnerability Notification, Open Source Systems, Cham, pp. 12–23, 2019.
[194] A. Bauer, N. Harutyunyan, D. Riehle, G.-D. Schwarz, Challenges of Tracking and Documenting Open Source Dependencies in Products: A Case Study, Open Source Systems, Cham, pp. 25–35, 2020.

[195] M. Michlmayr, F. Hunt, and D. Probert, Release Management in Free Software Projects: Practices and Problems, Open Source Development, Adoption and Innovation, Boston, MA, pp. 295–300, 2007.
[196] P. Sirkkala, T. Aaltonen, I. Hammouda, Opening Industrial Software: Planting an Onion, Open Source Ecosystems: Diverse Communities Interacting, Berlin, Heidelberg, pp. 57–69, 2009.
[197] B. Rossi, B. Russo, G. Succi, Download Patterns and Releases in Open Source Software Projects: A Perfect Symbiosis?, Open Source Software: New Horizons, Berlin, Heidelberg, pp. 252–267, 2010.
[198] M. Hatta, Two Modes of Product Development: Head-Oriented vs. Release-Oriented, Open Source Systems: Long-Term Sustainability, Berlin, Heidelberg, pp. 368–370, 2012.
[199] G. Poo-Caamaño, L. Singer, E. Knauss, D. M. German, Herding Cats: A Case Study of Release Management in an Open Collaboration Ecosystem, Open Source Systems: Integrating Communities, Cham, pp. 147–162, 2016.
[200] G. Poo-Caamaño, E. Knauss, L. Singer, D. M. German, Herding cats in a FOSS ecosystem: a tale of communication and coordination for release management, Journal of Internet Services and Applications, vol. 8, no. 1, p. 12, Aug. 2017.
[201] A. Ihara, D. Fujibayashi, H. Suwa, R. G. Kula, K. Matsumoto, Understanding When to Adopt a Library: A Case Study on ASF Projects, Open Source Systems: Towards Robust Practices, Cham, pp. 128–138, 2017.
[202] J. Teixeira, Release Early, Release Often and Release on Time. An Empirical Case Study of Release Management, Open Source Systems: Towards Robust Practices, Cham, pp. 167–181, 2017.
[203] J. A. Teixeira, H. Karsten, Managing to release early, often and on time in the OpenStack software ecosystem, Journal of Internet Services and Applications, vol. 10, no. 1, p. 7, Apr. 2019.
[204] T. Tuunanen, J. Koskinen, T. Kärkkäinen, Retrieving Open Source Software Licenses, Open Source Systems, Boston, MA, pp. 35–46, 2006.
[205] D. M. German, Y. Manabe, K. Inoue, A Sentence-Matching Method for Automatic License Identification of Source Code Files, Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, New York, NY, USA, pp. 437–446, 2010.
[206] Y. Higashi, M. Ohira, Y. Kashiwa, Y. Manabe, Hierarchical Clustering of OSS License Statements toward Automatic Generation of License Rules, Journal of Information Processing, vol. 27, pp. 42–50, 2019.
[207] ISO/IEC, ISO/IEC 9126-1:2001, Software engineering -- Product quality- Part 1: Quality model, 2001.
[208] ISO/IEC, ISO/IEC 25000:2005 Software Engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE, 2005.
[209] scikit-learn, Plot Hierarchical Clustering Dendrogram — scikit-learn 0.23.2 documentation, https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html#sphx-glr-auto-examples-cluster-plot-agglomerative-dendrogram-py, 2020, (Accessed: 2020-10-19).
[210] scikit-learn, A demo of the mean-shift clustering algorithm — scikit-learn 0.23.2 documentation, https://scikit-learn.org/stable/auto_examples/cluster/plot_mean_shift.html#sphx-glr-auto-examples-cluster-plot-mean-shift-py, 2020, (Accessed: 2020-10-19).
[211] 尾上紗野, 畑秀明, 松本健一, GitHub上の活動履歴分析による開発者分類, 情報処理学会論文誌, vol. 56, no. 2, pp. 715–719, Feb. 2015.
[212] E. Otte and R. Rousseau, Social network analysis: a powerful strategy, also for the information sciences, Journal of Information Science, vol. 28, no. 6, pp. 441–453, Dec. 2002.
[213] I. Struweg, A Twitter Social Network Analysis: The South African Health Insurance Bill Case, Responsible Design, Implementation and Use of Information and Communication Technology, Cham, pp. 120–132, 2020.
[214] M. Studer, Community Structure, Individual Participation and the Social Construction of Merit, Open Source Development, Adoption and Innovation, Boston, MA, pp. 161–172, 2007.
[215] M. A. Balieiro, S. F. S. de Júnior, C. R. B. de Souza, Facilitating Social Network Studies of FLOSS using the OSSNetwork Environment, Open Source Development, Communities and Quality, Boston, MA, pp. 343–350, 2008.
[216] J. Martinez-Romo, G. Robles, J. M. Gonzalez-Barahona, M. Ortuño-Perez, Using Social Network Analysis Techniques to Study Collaboration between a FLOSS Community and a Company, Open Source Development, Communities and Quality, Boston, MA, pp. 171–186, 2008.

[217] A. Wiggins, J. Howison, K. Crowston, Social Dynamics of FLOSS Team Communication Across Channels, Open Source Development, Communities and Quality, Boston, MA, pp. 131–142, 2008.
[218] A. Azarbakht, C. Jensen, Drawing the Big Picture: Temporal Visualization of Dynamic Collaboration Graphs of OSS Software Forks, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 41–50, 2014.
[219] A. Bosu, J. C. Carver, How Do Social Interaction Networks Influence Peer Impressions Formation? A Case Study, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 31–40, 2014.
[220] M. M. M. Syeed, I. Hammouda, Who Contributes to What? Exploring Hidden Relationships between FLOSS Projects, Open Source Software: Mobile Open Source Technologies, Berlin, Heidelberg, pp. 21–30, 2014.
[221] 寺野隆雄, 社会システムの研究動向1—世界と日本の事情— 計算機科学と社会科学のはざまで生きる社会シミュレーション, 計測と制御, vol. 52, no. 7, pp. 568–573, 2013.
[222] 出口弘, 社会システムの研究動向2—研究のための方法論— 社会システムの制度デザインの方法論：政策科学の方法としてのエージェントベースモデリング&シミュレーション, 計測と制御, vol. 52, no. 7, pp. 574–581, 2013.
[223] 高橋真吾, 社会システムの研究動向3—評価・分析手法(1)— モデルの解像度と妥当性評価, 計測と制御, vol. 52, no. 7, pp. 582–587, 2013.
[224] 倉橋節也, 社会システムの研究動向4—評価・分析手法(2)— モデル推定と逆シミュレーション手法, 計測と制御, vol. 52, no. 7, pp. 588–594, 2013.
[225] 市川学, 後藤裕介, 社会システムの研究動向5—研究のためのツール— シミュレーション言語の特徴と比較, 計測と制御, vol. 52, no. 7, pp. 595–600, 2013.
[226] N. Gilbert, Agent-based models, vol. 153. Sage Publications, Incorporated, 2019.
[227] N. Leveson, A new accident model for engineering safer systems, Safety Science, vol. 42, no. 4, pp. 237–270, Apr. 2004.
[228] 福澤寧子, STAMP/STPAによるシステム安全・セキュリティ解析, システム／制御／情報, vol. 62, no. 4, pp. 130–133, 2018.
[229] 独立行政法人情報処理推進機構, はじめてのSTAMP/STPA ～システム思考に基づく新しい安全性解析手法～, 2016.
[230] 独立行政法人情報処理推進機構, はじめてのSTAMP/STPA（実践編）～システム思考に基づく新しい安全性解析手法～, 2019.
[231] T. Bjerga, T. Aven, E. Zio, Uncertainty treatment in risk analysis of complex systems: The cases of STAMP and FRAM, Reliability Engineering & System Safety, vol. 156, pp. 203–209, Dec. 2016.
[232] D. Uesako, STAMP applied to Fukushima Daiichi nuclear disaster and the safety of nuclear power plants in Japan, S.M. in Engineering and Management, 2016.
[233] C. K. Allison, K. M. Revell, R. Sears, N. A. Stanton, Systems Theoretic Accident Model and Process (STAMP) safety modelling applied to an aircraft rapid decompression event, Safety Science, vol. 98, pp. 159–166, Oct. 2017.
[234] X. Meng, G. Chen, J. Shi, G. Zhu, Y. Zhu, STAMP-based analysis of deepwater well control safety, Journal of Loss Prevention in the Process Industries, vol. 55, pp. 41–52, Sep. 2018.
[235] H. Jianbo, Z. Lei, X. Shukui, Safety analysis of wheel brake system based on STAMP/STPA and Monte Carlo simulation, Journal of Systems Engineering and Electronics, vol. 29, no. 6, pp. 1327–1339, Dec. 2018.
[236] K. M. A. Revell, C. Allison, R. Sears, N. A. Stanton, Modelling distributed crewing in commercial aircraft with STAMP for a rapid decompression hazard, Ergonomics, vol. 62, no. 2, pp. 156–170, Feb. 2019.
[237] P. Yang, R. Karashima, K. Okano, S. Ogata, Automated inspection method for an STAMP/STPA - Fallen Barrier Trap at Railroad Crossing -, Procedia Computer Science, vol. 159, pp. 1165–1174, Jan. 2019.
[238] F. G. R. Souza, D. P. Pereira, R. M. Pagliares, S. Nadjm-Tehrani, C. M. Hirata, WebSTAMP: a Web Application for STPA & STPA-Sec, MATEC Web Conf., vol. 273, 2019.
[239] S. Williams, Use of STAMP/STPA to Model Organizational Risk and Safety Management at Cruise and Ferry Companies, MATEC Web Conf., vol. 273, 2019.
[240] A. P. Goncalves Filho, G. T. Jun, P. Waterson, Four studies, two methods, one accident – An examination of the reliability and validity of Accimap and STAMP for accident analysis, Safety Science, vol. 113, pp. 310–317, Mar. 2019.

[241] N. A. Stanton, C. Harvey, C. K. Allison, Systems Theoretic Accident Model and Process (STAMP) applied to a Royal Navy Hawk jet missile simulation exercise, Safety Science, vol. 113, pp. 461–471, Mar. 2019.
[242] Massachusetts Institute of Technology, STAMP Workshop | Partnership for Systems Approaches to Safety and Security (PSASS), http://psas.scripts.mit.edu/home/stamp-workshops/, 2020, (Accessed: 2020-10-19).
[243] 独立行政法人情報処理推進機構, 第1回　STAMPワークショップ in Japan：IPA 独立行政法人 情報処理推進機構, https://www.ipa.go.jp/sec/events/20161205.html, 2016, (Accessed: 2020-10-19).
[244] 独立行政法人情報処理推進機構, 第2回　STAMPワークショップ：IPA 独立行政法人 情報処理推進機構, https://www.ipa.go.jp/sec/events/20171127.html, 2017, (Accessed: 2020-10-19).
[245] 独立行政法人情報処理推進機構, 第3回 STAMPワークショップ：IPA 独立行政法人 情報処理推進機構, https://www.ipa.go.jp/ikc/events/20181203.html, 2018, (Accessed: 2020-10-19).

付録

```
from github import Github
import json

dev_log = Github(
    '[ACCESS_TOKEN]'
  ).get_repo(
    '[PROJECT_ID]'
  ).get_events()

dev_log_json = [
  {'user_name': i.actor.name,
  'event_type': i.type
  } for i in dev_log
]

with open('dev_log.json', 'w') as f:
  json.dump(dev_log_json, f)
```

付録 1 開発記録取得スクリプト

```
const _ = require('lodash');
const fs = require('fs');

const dev_log = _(
  JSON.parse(
    fs.readFileSync(
      'dev_log.json',
      'utf8')
    )
  )
  .groupBy('user_name')
  .map((value, key) => {
    return {
      'user_name': key,
      'event_type': _.countBy(value, 'event_type'),
    };
  })
  .value()

for (i in dev_log) {
  console.log(
    dev_log[i].user_name + "," +
    (dev_log[i].event_type.CommitCommentEvent ? dev_log[i].event_type.CommitCommentEvent : 0) + "," +
    (dev_log[i].event_type.CreateEvent ? dev_log[i].event_type.CreateEvent : 0) + "," +
    (dev_log[i].event_type.DeleteEvent ? dev_log[i].event_type.DeleteEvent : 0) + "," +
    (dev_log[i].event_type.ForkEvent ? dev_log[i].event_type.ForkEvent : 0) + "," +
    (dev_log[i].event_type.GollumEvent ? dev_log[i].event_type.GollumEvent : 0) + "," +
    (dev_log[i].event_type.IssueCommentEvent ? dev_log[i].event_type.IssueCommentEvent : 0) + "," +
```

付録 2 開発記録の加工スクリプト

```
      (dev_log[i].event_type.IssuesEvent ? dev_log[i].event_type.IssuesEvent : 0) + "," +
      (dev_log[i].event_type.MemberEvent ? dev_log[i].event_type.MemberEvent : 0) + "," +
      (dev_log[i].event_type.PublicEvent ? dev_log[i].event_type.PublicEvent : 0) + "," +
      (dev_log[i].event_type.PullRequestEvent ? dev_log[i].event_type.PullRequestEvent : 0) + "," +
      (dev_log[i].event_type.PullRequestReviewCommentEvent ? dev_log[i].event_type.PullRequestReviewCommentEvent : 0) + "," +
      (dev_log[i].event_type.PushEvent ? dev_log[i].event_type.PushEvent : 0) + "," +
      (dev_log[i].event_type.ReleaseEvent ? dev_log[i].event_type.ReleaseEvent : 0) + "," +
      (dev_log[i].event_type.SponsorshipEvent ? dev_log[i].event_type.SponsorshipEvent : 0) + "," +
      (dev_log[i].event_type.WatchEvent ? dev_log[i].event_type.WatchEvent : 0)
    )
}
```

付録 2 開発記録の加工スクリプト (続き)

```
{
  "main": "app.js",
  "dependencies": {
    "lodash": "^4.17.20"
  }
}
```

付録 3 package.json

```
from matplotlib import pyplot as plt
from sklearn import datasets, preprocessing
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import numpy as np
import pandas as pd

existing_df = pd.read_csv(
    "dev_log.csv",
    index_col=0,
    thousands=','
)

existing_2d = PCA(n_components=2).fit(existing_df).transform(existing_df)
existing_df_2d = pd.DataFrame(existing_2d)
existing_df_2d.index = existing_df.index
existing_df_2d.columns = ['PC1', 'PC2']

cls = KMeans(n_clusters=3)
result = cls.fit(existing_df_2d)
```

付録 4 主成分分析と k-means 処理スクリプト

```
plt.scatter(
  existing_df_2d['PC1'],
  existing_df_2d['PC2']
)

plt.scatter(
  result.cluster_centers_[:, 0],
  result.cluster_centers_[:, 1],
  s=250,
  marker='*',
  c='red'
)

plt.show()
```

付録 4 主成分分析と k-means 処理スクリプト (続き)

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.DiGraph()

G.add_nodes_from([
  "mspncp",
  "mattcaswell",
  "DDvO",
  "levitte",
  "openssl-machine",
  "p-steuer",
  "slontis",
  "kaduk",
  "kroeckx",
  "bbbrumley",
  "vdukhovni",
  "yanchangjia",
  "paulidale",
  "frustreated",
  "beldmit",
  "HuKeping",
  "bheesham-devops",
  "nmathewson",
  "guidovranken",
  "richsalz",
])
```

付録 5 ネットワークグラフ出力スクリプト

```
28 G.add_edge('mattcaswell', 'mspncp')
29 G.add_edge('DDvO', 'mspncp')
30 G.add_edge('levitte', 'mspncp')
31
32 G.add_edge('mspncp', 'mattcaswell')
33 G.add_edge('DDvO', 'mattcaswell')
34 G.add_edge('levitte', 'mattcaswell')
35
36 G.add_edge('mspncp', 'DDvO')
37 G.add_edge('mattcaswell', 'DDvO')
38 G.add_edge('levitte', 'DDvO')
39
40 G.add_edge('mspncp', 'levitte')
41 G.add_edge('mattcaswell', 'levitte')
42 G.add_edge('DDvO', 'levitte')
43
44 G.add_edge('openssl-machine', 'mspncp')
45 G.add_edge('p-steuer', 'mspncp')
46 G.add_edge('slontis', 'mattcaswell')
47 G.add_edge('kaduk', 'mspncp')
48 G.add_edge('kroeckx', 'DDvO')
49 G.add_edge('bbbrumley', 'mspncp')
50 G.add_edge('vdukhovni', 'levitte')
51 G.add_edge('yanchangjia', 'mspncp')
52 G.add_edge('paulidale', 'mattcaswell')
53 G.add_edge('frustreated', 'mspncp')
54 G.add_edge('beldmit', 'mspncp')
55 G.add_edge('HuKeping', 'DDvO')
56 G.add_edge('bheesham-devops', 'mspncp')
57 G.add_edge('nmathewson', 'mspncp')
58 G.add_edge('guidovranken', 'levitte')
59 G.add_edge('richsalz', 'mspncp')
60
61 nx.draw_networkx(G)
62 plt.show()
```

付録 5 ネットワークグラフ出力スクリプト (続き)