オープンソース･ソフトウエア・プロジェクトのパフォーマンスと規定要因

2002年度　大川情報通信基金研究助成報告書

2003年9月

濱岡 豊
慶応大学商学部
hamaoka@fbc.keio.ac.jp

Comment welcome
ただし､未定稿の段階につき筆者に無断での引用を禁ず。

概要

Linuxをはじめとしたオープンソースソフトウエア が市場でも一定のシェアを得ているのに伴って、実務的な注目を受ける一方、オープンソースについての研究もいくつか行われている。しかし、これまでの研究は、開発プロセスもしくは、プロジェクトに参加する個人の動機にのみ注目してきたこと、「成功」したオープンソース・ソフトウエア・プロジェクトを取り上げており、プロジェクト間での定量的な比較が行われていないという限界がある。本研究では、オープンソース･ソフトウエアが、ユーザーによって開発され、サポートされ、プロモーションされるという特徴を踏まえて、オープンソース・ソフトウエアのパフォーマンス指標として、「開発プロセス」「開発されたソフトウエア」だけでなく「市場における成果」「コミュニティへのインパクト」「コミュニティの成員へのインパクト」という広い範囲について考慮する必要性を指摘した。さらに、これらを規定する要因についても考察し、包括的な分析の枠組みを提案した。

6万以上のオープンソース･ソフトウエア・プロジェクトを無料でホストしているSourceforge.netから2101のオープンソース・ソフトウエア・プロジェクトを選び、ダウンロード可能なデータを用いて実証分析を行った。25%のプロジェクトではソフトウエアが公開されていないこと、53.6%のプロジェクトでは1人で開発されていること、7割以上のプロジェクトでは、一人で開発の50%以上を負担していること（CVSへのコミットメント回数ベース)、公開してもソフトウエアのアップデートをしていないプロジェクトが21%、ダウンロード数のメディアンは844回など、オープンソース・ソフトウエア・プロジェクトの実態が明らかになった。クラスタ分析を行った結果、2101プロジェクトのうち1699プロジェクトは、ユーザーフィードバック、CVSへの貢献、ユーザーからの要望の解決率、ページビュー・ダウンロードともに平均以下であることが明らかとなった。

提示した包括的な分析枠組みを用いつつ、利用可能な変数に注目して仮説を設定し、Sourceforgeからのデータを用いて検定した。この結果、プロジェクトの認知度がダウンロード数を増加させ、ユーザーコミュニティの規模を拡大させる。そして、ユーザーコミュニティの規模が、プロジェクトの認知率を増加させるという正のフィードバックが存在していることが実証された。ユーザー規模の拡大は、ユーザーからのフィードバック（バグ報告、機能追加要求、ヘルプリクエスト）を増加させ、開発を活発化させること、開発者の数を増加させ、リリース回数を増加させる作用もあることも実証された。

Performance metrics of Open Source Software Projects and its Determiners

Theory and Empirical study of 2000 Open Source Software Projects

September, 2003

Yutaka Hamaoka

Faculty of Business and Commerce
Keio University

hamaoka@fbc.keio.ac.jp

Abstract

In this research, research framework for performance metrics of open source software project is proposed . That focuses *whole* process of OSSP, that is, (1)development process, (2)quality of developed software, (3)performance at market, (4)impact to user community, and (5)impact to individual community members. Determiners of those factors are discussed, then comprehensive framework is proposed. Publicly available data is compiled form Sourceforge.net, an internet site that host 60,000 OSSPs free of charge. Preliminary analysis shows, 1699 of 2101 OSSP are less than average in terms of the number of download, page view, development activity, user-to-user support activity. Based on proposed framework, hypotheses are developed . Statistical test confirms positive feedback loop: recognition to OSSP -> the number of download -> the number of community member ->recognition to OSSP.

1.はじめに

Linuxをはじめとしたオープンソースソフトウエア が市場でも一定のシェアを得ているのに伴って、実務的な注目を受ける一方、オープンソースについての研究もいくつか行われている。これまでの研究は、大まかに、2つに分類できる。一つ目は、ソフトウエア工学の観点から、「開発」に注目したものである。Raymond(1998)によるLinux、Mockus and Hersbleb(2000)によるApache、Tuomi(2000)によるLinux kernel、Aoki et al.(2001)によるJun、von Krogh et al.(2003)によるFreenetの分析などがそれらである。二つ目は、個人に注目し、オープンソース・ソフトウエアの開発やサポートに無償で参加/関与する「動機 motivation」に注目したものである。Lakhani and von Hippel (2003)のApacheのヘルプライン、Hertel et al.(2003)によるLinux開発者へのアンケート調査がそれである[注1] 。

このように、これまでの研究では、「成功」したオープンソース・ソフトウエア（プロジェクト）を取り上げており、プロジェクト間での比較が行われていない。また、オープンソース・ソフトウエアではユーザーが開発、サポート、プロモーションを行うにも関わらず、開発やヘルプなどそのプロセスの一部にしか注目していない。特に、Ghosh and Prakash (2001)のCVSの分析を除くと、定量的な比較はなされていない状況にある。

このような背景のもとで行われる、本研究の目的と構成は、以下の通りである。まず、開発スケジュール（期限）、費用や売り上げ目標などが明示されていないオープンソース・ソフトウエアの場合、「成功」の定義自体を考え直す必要がある。 これについては開発のみならず、その後のサポートなどのプロセスも踏まえてオープンソース・プロジェクトのパフォーマンス=成功を定義する必要がある。さらにパフォーマンスを規定する要因群を挙げ、包括的な分析枠組みを構築する（2節）。Sourceforge上の2000のオープンソース・ソフトウエアプロジェクトについて、公開されているデータを用いて、上記の枠組みを踏まえた定量的な分析を行うことによって、オープンソース・ソフトウエアのパフォーマンスの現状を把握する（3節）。さらに、包括的な枠組みのうち入手可能な変数に注目して仮説を設定し、実証研究を行う（4節）。

[注1] 金子ら(1998)は、オープンソース・ソフトウエアではないが、日本のシェアウエア作家へのインタビューをまとめている。

2.オープンソース･ソフトウエア・プロジェクトのパーフォーマンスと規定要因

この節ではまず、開発のみならず、その後のサポートなどのプロセスも踏まえてオープンソース・プロジェクトの特徴を挙げ、パフォーマンス指標を提示する。

2.1　オープンソース・ソフトウエア・プロジェクトの特徴

（1）開発段階

Opensource Initiative(2002)は、オープンソースの定義として以下の項目を挙げている。ソースコードを公開し、自由に利用、改変、配布できるというのが最大の特徴である。これによって、他者の作成したソフトウエアを再利用したり、他者と共同で開発を行うことが可能となっているのである。

図表　「オープンソース」の定義

1. Free Redistribution
2. Source Code
3. Derived Works
4. Integrity of The Author's Source Code
5. No Discrimination Against Persons or Groups
6. No Discrimination Against Fields of Endeavor
7. Distribution of License
8. License Must Not Be Specific to a Product
9. License Must Not Contaminate Other Software

出所)http://www.opensource.org/

このような特徴をもつオープンソース・ソフトウエアについて、Mockus and Hersbleb(2000)は、通常の産業界におけるソフトウエアの開発との違いを次のようにまとめている。

・オープンソース・ソフトウエアは、（数百名以上の）多数のボランティアによって開発される可能性がある[注2]。
・作業は割り当てられず、自分がしたいことを選んでする。
・システムレベルの明確な設計はない。詳細設計すらされないこともある。
・プロジェクトの計画、スケジュール、成果品のリストもない。

（2）ソフトウエアそのもの

これまでのソフトウエアおよびソフトウエアプロジェクトを評価するための指標（メトリクス metrics）として下記の項目が上げられている。これに対して、Raymond(1998)は、Linuxの開発方式の特徴として、「Linusの法則」、つまり「 はやめのリリース、ひんぱんなリリース。そして顧客の話をきくこと」をあげている。つまり、時間をかけてデバッグをして完成度を高めるよりも、多くのユーザーによって利用してもらい、バグを見つけてもらい修正する方が効率的だというのである。

ユーザーからのフィードバックの重要性、頻繁なリリースといった開発体制の特徴があることを指摘している。このように用いる開発プロセスによっては、バグの多さを品質の指標として扱うことは難しい。

注2　これが可能となるのも、ソースコードが公開され、修正を許しているからである。

図表　主要なソフトウエア・メトリクス

分類	項目	ソフトウエアのメトリクス
製品のオペレーション	正確性　correctness 信頼性　Reliability 効率性　Efficiency 統合性　Integrity 利用容易性　Usability	・量に注目した指標 1000行あたりのエラーの数　errors per KLOC(thousand lines of code) 1000行あたりの欠陥の数 defect per KLC 一行あたりの費用　$ per LOC ・機能に注目した指標 ファンクションポイントあたりのエラーの数　errors per FP(function point) ファンクションポイントあたりのの欠陥の数 defect per KLC ファンクションポイントあたりの費用　$ per LOC
製品の移行	保守性　Maintainability 柔軟性　Flexibility 試験可能性　Testability	
製品の改訂	移植性　Portability 再利用可能性　Reusability 接続性　Interoperability	

出所）Pressman(1997),p. 519および,ch. 3より作成

（3）サポート

企業によって販売されたものであれば、それをサポートすることは発売元の企業が行うべきことである。これに対して、企業によらずユーザーが開発したオープンソース・ソフトウエアの場合、ユーザー間でのサポートが重要な役割を果たす。Lakhani and von Hippel (2003)がApacheのhelp lineでの質問者、回答者へのアンケート調査によると、多くの質問は短時間で解決されており、ユーザー間でのサポートは有効に機能しているという。ユーザー間でのサポートについても重要な問題である。

（4）普及段階

Bonaccorsi and Rossi (2003)は、OSSの成功には、参加者の動機 motivation, プロジェクトの調整 co-ordination, 開発されたソフトウエアの普及 diffusionが重要であることを指摘している。[注3] 企業によるソフトウエアの場合、普及させるために広告などのマーケティングを行うことができるが、多くのオープンソース・ソフトウエアの場合、ユーザーがその役割を担うことになる。オープンソース・ソフトウエアではないが、古川(1993)、池田ら(1997)はパソコン通信のユーザーに対する調査によって、他のユーザーのメッセージが情報収集や商品についての評価に用いられていることを示している。

2.2　オープンソース･ソフトウエア・プロジェクトのパフォーマンス指標

オープンソースソフトウエアの場合、ユーザーがバグを発見することを前提として、テストが不完全な状態でソフトウエアを公開したり、仕様・スケジュールを明確に規定しないまま進められる例が多い。さらに、ユーザーが無償で開発に参加したり、その入れ替わりもあるという特徴がある。このようにオープンソース・ソフトウエアでは、ユーザーコミュニティが、バグ情報などのフィードバック、開発への参加、プロモーション、サポートといった広範囲な機能を果たす必要がある。よって、開発やソフトウエアの品質だけでなく、より広い範囲でパフォーマンス指標を考える必要がある。

これらを踏まえると、オープンソース・ソフトウエアについては、「開発プロセス」「開発されたソフトウエア」「市場における成果」「コミュニティへのインパクト」「コミュニティの成員へのインパクト」、それぞれについて考慮する必要がある。

[注3] 濱岡（2001,2002）は、消費者の消費における創造に注目し、創造しコミュニケーションすることの重要性を指摘し「アクティブ・コンシューマー」という概念を提示している。

（1）開発そのもの
　プログラマーは自発的に無報酬で参加する。このため、費用はかからず、期限が明示されることはない。オープンソース・ソフトウエアの場合、調整者がいるとは限らないので、自発的にできることを各自が行う。このため、適切な作業分担がなされているのかといった点を評価することが必要である。
（2）開発されたソフトウエア
　システム/詳細設計がされていないことが多いことや、バグがあることを前提に早くリリースすることが多いので品質については評価が困難である。

（3）市場における成果
　無料で配付されるものが多いため、売上といった金銭的な評価は不可能。ソフトウエアやプロジェクトの認知度、ダウンロード数、さらには他のコードへの再利用などの観点から評価する必要がある。

（4）コミュニティへのインパクト
　開発者、ユーザーなどがコミュニティを形成して開発やサポート、普及などを進めていくという特徴がある。よって、開発者の増加や、サポート活動の活発さ、ユーザーからのフィードバックの活発さなどを指標として考える必要がある。

（5）コミュニティの成員へのインパクト
　そもそも無償であるにも関わらず参加するのはなぜなのだろうか？そう考えると個人レベルでの指標を考慮することも重要である。オープンソース・ソフトウエアの場合、金銭的なインセンティブは使えないので、ユーザーのソフトウエアへの満足度の増大、ソフトウエア、コミュニティへのロイヤリテイの増大、成員の知識・スキルの増大、さらには、コミュニティをつなぐ成員間の信頼についても評価する必要がある。

図表　オープンソース・ソフトウエア・プロジェクトのパフォーマンス指標

分類	オープンソース･ソフトウエア・プロジェクトの特徴	測定項目の例
開発そのもの	・費用、期限 　プログラマーは自発的に無報酬で参加する。このため、費用はかからず、期限が明示されることはない。 ・適切な作業分担 　調整者がいるとは限らないので、自発的にできることを各自が行う。	開発のスピード 修正/アップデートの頻度 開発の活発さ デバッグのスピード 開発の分散の程度
開発されたソフトウエア	・品質 　システム/詳細設計がされていないことが多い。 　バグがあることを前提に早くリリースすることが多いので評価が困難。	・独創性 ・機能性 ・バグの少なさ
市場における成果	・売り上げ、シェアなど 　市場での成果が目標とされることはない無料で配付されるものが多いため、売上といった金銭的な評価は不可能。	・認知度 ・ダウンロード数 ・他のコードへの再利用
コミュニティへのインパクト	・開発者 　ユーザーが開発に参加する可能性がある。 ・サポート 　ユーザーがサポートを行う可能性がある。 ・ユーザーからのフィードバック 　バグ報告や機能要求など、ユーザーからのフィードバックによってソフトウエアの改善が進む。	・ユーザーの増加 ・開発者数の増加 ・機能追加要求 ・バグ報告とそれへの対応 ・ユーザー間でのサポート ・サポートの早さ ・デバッグの早さ
コミュニティの成員	・ユーザーが多様な活動に参加しうるので、個人レベルでも各種のノウハウなどの蓄積が進む。	・ユーザーのソフトウエアへの満足度の増大 ・ソフトウエア、コミュニティへのロイヤリテイの ・成員の知識、スキル ・成員間の信頼

2.3　オープンソース･ソフトウエアのパフォーマンスの規定要因

高いパフォーマンスを選るにはどうすればよいのだろうか？ここではその分析枠組みを提示する。Pressman(1997)は「ソフトウエア開発プロジェクト」のパフォーマンスに影響を与える要因を、「課題そのものの要因　problem factor」「プロセス要因　process factor」「製品要因　product factor」「資源要因　resource factor」「人的な要因　people factor」に分類している。

これらのうち、人的な要因、つまり開発組織の大きさはソフトウエアの成功を規定する要因であると同時に、オープンソース・ソフトウエアの場合には、ユーザーが開発に参加するという意味で成果指標のひとつでもある。

また、PressmanをはじめとしたSoftware engineeringでは、開発までしか扱わないが、前節で指摘したように、開発後の普及も重要なパフォーマンス指標である。つまり、市場の特性も考える必要がある。

さらに、人的要因については、コミュニティレベルと、コミュニティの成員レベルに分解して考える必要がある。　コミュニティの成員レベルでは、動機についての研究がいくつかみられる。無償であるにもかかわらず、オープンソース・ソフトウエアの開発に参加したり、質問に対して回答するのはなぜか、というのが研究者が注目している点である。これについて、Kollock(1999)は、サイバースペースに自己の創作したものなどを提供し貢献する動機として、「評判reputation」「互酬性reciprocityへの期待」「環境への影響感sense of efficacy」「コミットメント commitmentの４つを挙げている。また、Constant et al. (1996)は、Tandem Computersのhelp lineへの質問に回答したことがある者への調査によって、「自社にとって重要な問題だから（The problem is important to the company）」「よき社員となるため（Being a good company citizen）」「問題を解決することが楽しい（I enjoy solving problems）」などへの回答率が高いことを示した。同様に、Lakhani and Hippel (2003)は、Apache サーバーソフトウエアのhelp lineに投稿された質問への回答者への調査を行い、「オープンソース・ソフトウエアをプロモーションするため」「以前に助けられたことがあるから」「解答することが楽しいから」などへの解答率が高くなることを示した[注4]。

これらは、個人の動機に注目したものだが、プロジェクトもしくはコミュニティレベルで考えると、互酬性が特に重要であると考えられる。質問をしても返答がなければ、質問をしようと思わないだろうし、ソフトウエア自身への不満にもつながる可能性がある。一方、質問の当事者でない者にとっては、メッセージのやり取りが活発であることは、メッセージを投稿することへの意識的障害を低くするだろう。また、ヘルプに対しての対応、つまりサポートがしっかりしている方が利用意向も高くなるだろう。

このように、これまでの要因群に加えて、人的な要因、市場の要因についても考慮する必要がある。

[注4] 中村、金子(1999)は「弱さ」に注目している。つまり、弱い者が「助けを求める」行動が他者による支援という行動を引き出すというのである。

図表2　オープンソース・ソフトウエアのパフォーマンスに影響を与える要因

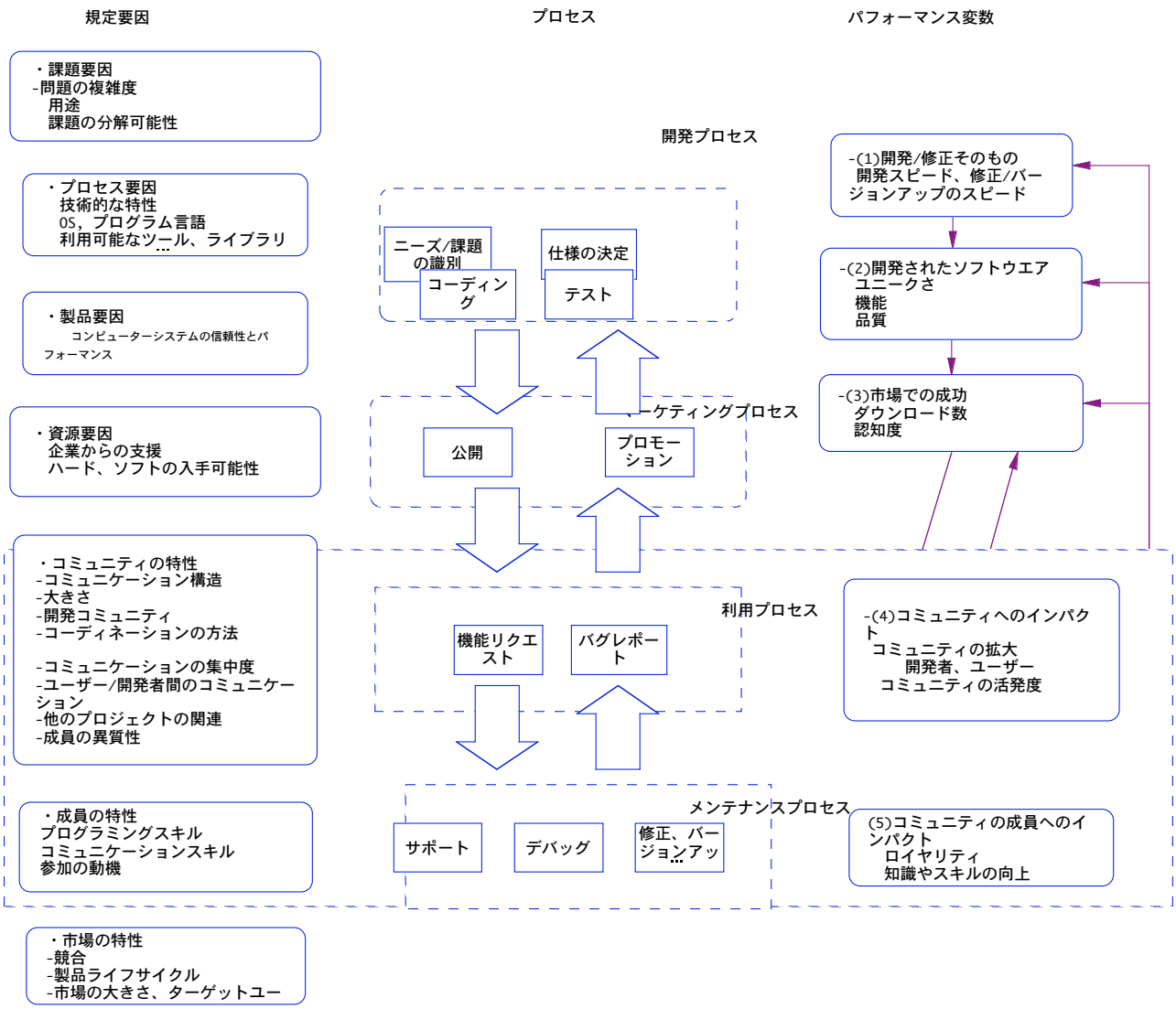
分類	通常のソフトウエア開発のパフォーマンスの規定要因	オープンソース・ソフトウエアに特有の事項
課題要因　problem factor	・解くべき課題の複雑さ ・課題の分解可能性 ・設計についての制約や要求の変化	
プロセス要因 process factor	・ソフトウエアの開発プロセス ・プロセスの分解 ・プロジェクト管理 ・用いる分析／設計の手法、利用言語やツール	
製品要因　product factor	コンピューターシステムの信頼性とパフォーマンス	
資源要因 resource factor	CASEツール、ハードソフトの入手可能性	
人的な要因 people factor	開発組織の大きさと専門性 ・チームの成員 ・リーダー ・チームのマネジメント方法 民主的/管理型 分散型/集中型 な分散方式、管理された分散方式、管理された集中方式 ・協調 ・コミュニケーション	ユーザーが開発したり、サポートする可能性があるため、チームの成員、サイズなどが変化する。 コミュニティが大きくなるとユーザーからのフィードバックが増加し、ソフトウエアの品質が改良される可能性もある。 つまり規定要因であると同時に、規定される要因でもある。
市場要因		ソフトウエアエンジニアリングでは開発、メンテナンスが考慮する範囲であり、普及まで考えていない。 市場での競争、市場規模などを考慮する必要がある。

出所）通常のソフトウエアについては、Pressman(1997),ch. 3より作成

前節で挙げたパフォーマンス指標、開発プロセス、そして本節で挙げた規定要因についてまとめた（図表　）。煩雑になるので示していないが、パフォーマンス指標間にも、市場での成功がコミュニティを拡大し（コミュニティへのインパクト）、それがユーザーからのフィードバックを増加させ（同）、開発を活性化させ（開発プロセスへの影響）るといった相互の関係もある。

また、コミュニテイ要因、コミュニティの成員要因については、パフォーマンス指標であると同時に、規定要因ともなりうる（コミュニケーションの円滑さが開発者やユーザーを増加させる）ことに注意したい。

図表　オープンソース・ソフトウエアのパフォーマンス、プロセスとその規定要因



注）要因間での因果関係については煩雑になるので表示していない。

3. オープンソース・ソフトウエア・プロジェクトの神話と真実

ここでは、前節で示したパフォーマンス指標に沿って実証分析を行い、オープンソース・ソフトウエア・プロジェクトの実態を把握する。分析対象は、sourceforge.net上でホストされているオープンソース・ソフトウエア・プロジェクトである。Sourceforge.netは、1999年11月以降[注5]、オープンソース・ソフトウエアプロジェクトに対して、開発に必要な資源を無料で提供しているサイトである[注6]。ホームページ、CVS、メーリングリスト、バグや機能追加についてのリクエスト用のTracker、自由にディスカッションできるforum、ファイルの格納とダウンロードを行う機能などが提供され、投稿されたメッセージや、ファイル別のダウンロード回数、CVSなどが公開されている。

データをダウンロードした2001年5月の時点で、20,052プロジェクト、165,245人がユーザーとして登録されていた[注7]。まず、それらのなかからランダムに2000プロジェクトを選択した[注8]。これらにファイルのダウンロード数トップ100、ホームページのページ閲覧回数トップ50、フォーラムへの投稿回数トップ50プロジェクトを加えた、2197プロジェクトを分析対象とした。ただし、これらのプロジェクトの概要ページをダウンロードしたところ、95プロジェクトについては無効であったので、有効サンプル数は2102となった[注9]。

プロジェクトによって、例えばファイルを公開している/いない、バグ・レポートシステムを使っている/いないなどの差異があるため、以下の記述的な分析では、変数によってサンプル数が異なることに注意されたい。また、前節で提示したパフォーマンス指標群のうち「ソフトウエアそのもの」「コミュニティの成員へのインパクト」についてはデータがないため評価できない。

3.1「開発プロセス」

・開発のスピード：初めてのファイル公開までの日数

2102プロジェクト中、528プロジェクト（25.1%）はファイルを全く公開していない。ファイルを公開している1574プロジェクトについて、プロジェクト登録して初めてファイルを公開するまでに要した日数をヒストグラムにまとめた。これをみると、登録後すぐにファイルを公開しているプロジェクトの割合が高くなっている。これは、ソフトウエアが開発された後に、Sourceforgeに登録したためと考えられる。

[注5] このサイトでは登録順にグループ番号が与えられている。グループ番号1はSourceForge.netのインフラとして使われているシステムを開発するプロジェクトalexandriaである。

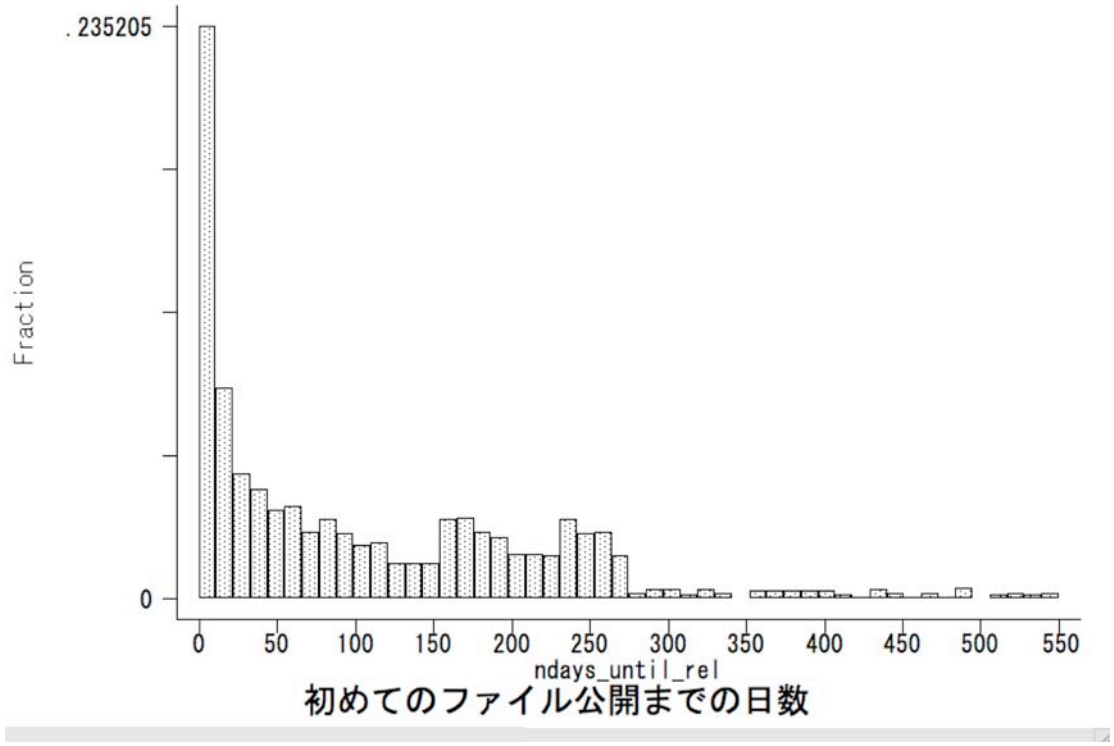
[注6] Home Page, Forums, Tracker, Support, Lists, Docs, Surveys, News, Files

[注7] プロジェクトについてはsourceforge.netに申請し、承認が必要。ユーザーについては自由に登録できる。2003年9月1日現在、68,800 プロジェクト、706,388 人がユーザー登録している。

[注8] まずは全期間でのアクティビティ・ランキングリストを用い、その上位200をすべて。それ以降については10間隔で選択した。ただし、アクティビティランキングについては上位13000位までしか公開されていなかったため、softwaremap(ソフトウエア一覧)からもプロジェクトを補足した。現在はすべてのプロジェクトの一覧をXMLで出力する機能があるため、より簡単にデータを得ることができる。

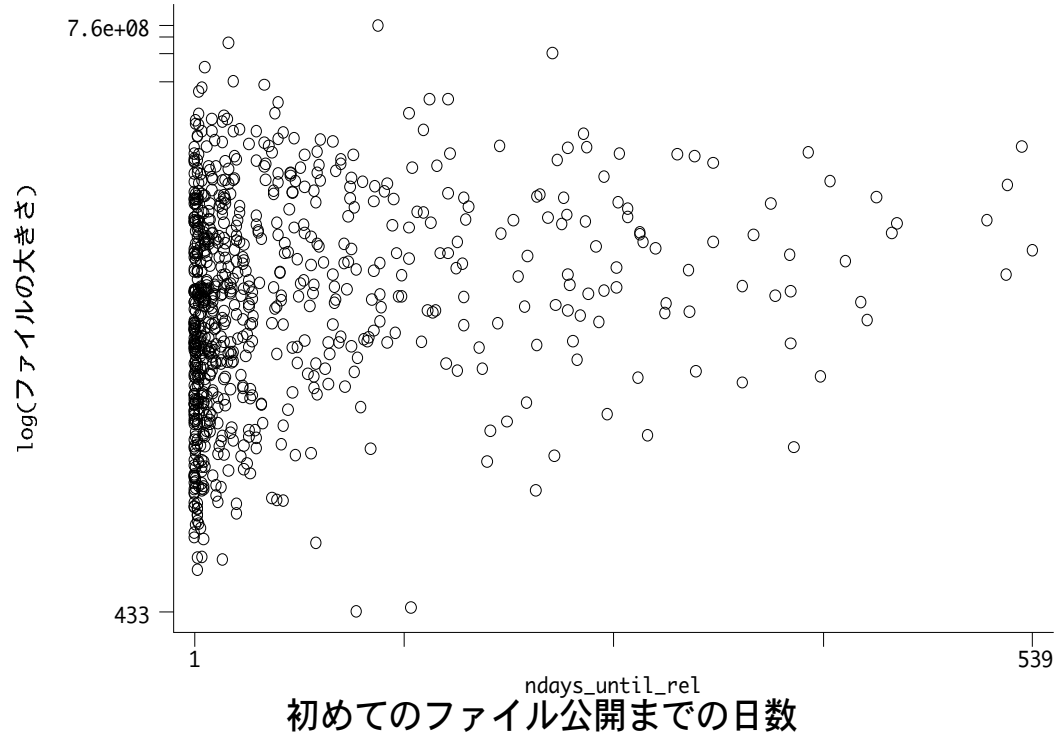
[注9] Invalid Projectとだけ表示される。

図表　Sourceforgeへの登録からファイル公開までの経過日数（N=1574）



公開までの日数と、ファイルのサイズをプロットした。大きなファイルほど開発に時間がかかり公開までの日数が長くなるといった明確な関係はない。

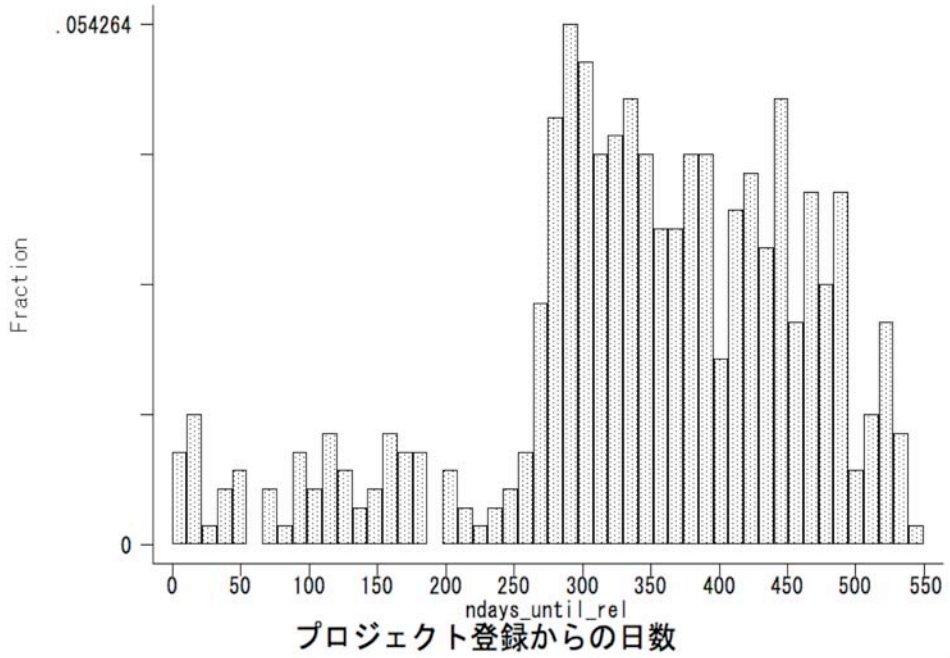
図表　公開までの日数とファイルサイズ



一方、ファイルを公開していないプロジェクトについて、Sourceforgeに登録してからの経過日数の分布をみた(図表)。200日以上経過してもファイルを公開できないプロジェクトの割

合が高いことがわかる。

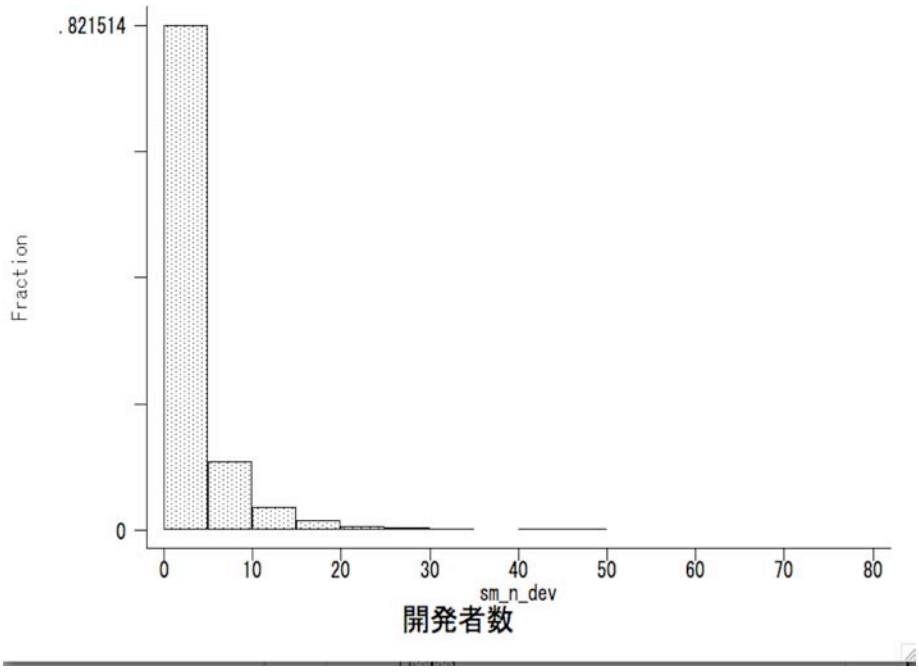
図表　ファイルを公開していないプロジェクトのSourceforgeへの登録からの日数（N=528）



・開発者数

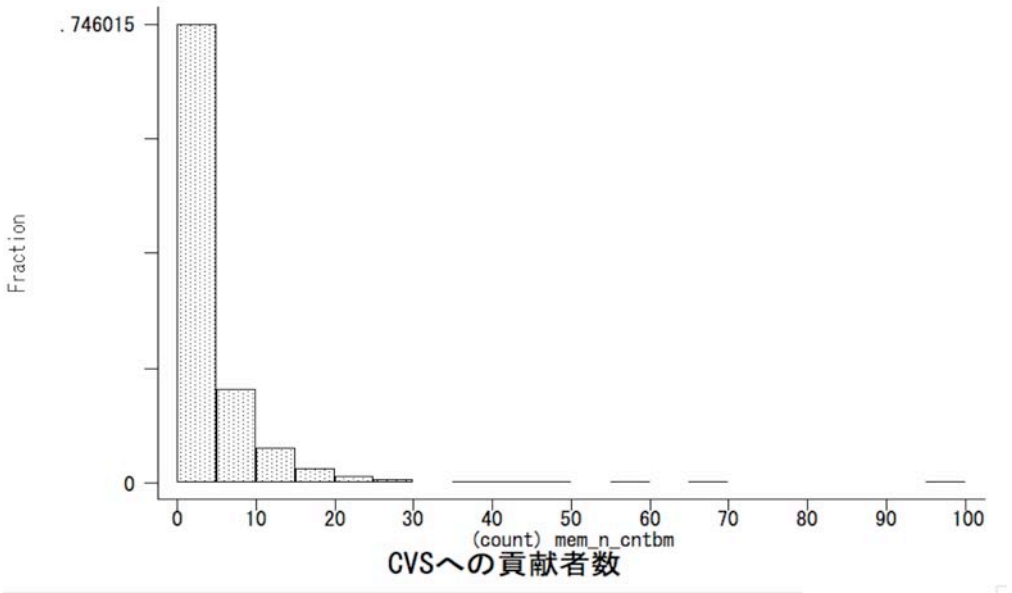
プロジェクトの概要ページには開発者がリストされている。その人数をヒストグラムにしたものが、次の図表である。グラフでは読み取りにくいが、開発者が１名のプロジェクトが1,127(53.6%)と最も高い割合となっている。開発者数が最大なのは、79名がリストされている、”The Great AIP(Artificial intelligence project)”である。Linuxは世界中の数百人の共同作業によって開発されているが、Sourceforgeに登録されているプロジェクトの多くは小さなプロジェクトである。

図表　開発者数の分布



941プロジェクトでは、CVSも公開している[注10] 。CVSにcommitした者の数をプロジェクトごとにカウントし、ヒストグラムを描いた。前にみた開発者の数と同様、コントリビューター1人のプロジェクトが最も多くなっている[注11] 。コントリビューターの数が最も多いのは100人が貢献した"jboss"プロジェクトである。ホームページに記載されている開発者とCVSへのコントリビューターの数が一致しないのは、例えばプロジェクトの管理は行うが実際のプログラミングなどは行わない、といったメンバーがいるためと考えられる。

図表 CVSへの貢献者数の分布（N=941）



注)CVSを利用しているN=941プロジェクト。

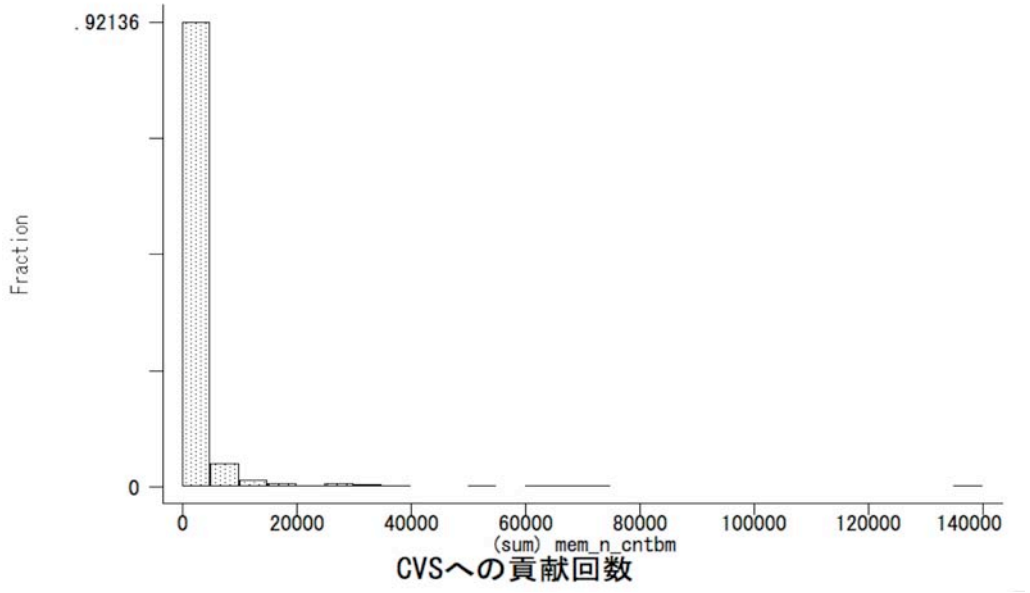
[注10] Concurrent Version Control Systemの略。共同でソフトウエアを開発するために必要な機能を提供するソフトウエア。修正したことによってバグが生じる可能性もあるので、修正前のものに復帰する、といった機能がある。だれが、いつ、どのファイルを修正したかがhistoryファイルに記録される。

[注11] 各プロジェクトは、ホームページに開発者のリストを掲載している。2101プロジェクト中、1127プロジェクトが1人で開発している。ここでは、CVSにcheck inしている者のみをカウントした。

・開発のアクティビティ[注12]

SourceforgeのCVSを利用している941プロジェクトについて、CVSへの総コミットメント回数の分布をまとめた。もっとも多いプロジェクトでは135,930回コミットされている。ただし、全体での平均は2095回であり、ヒストグラムに見られるようにほとんどのプロジェクトのコミットメント回数は2000回以下となっている。

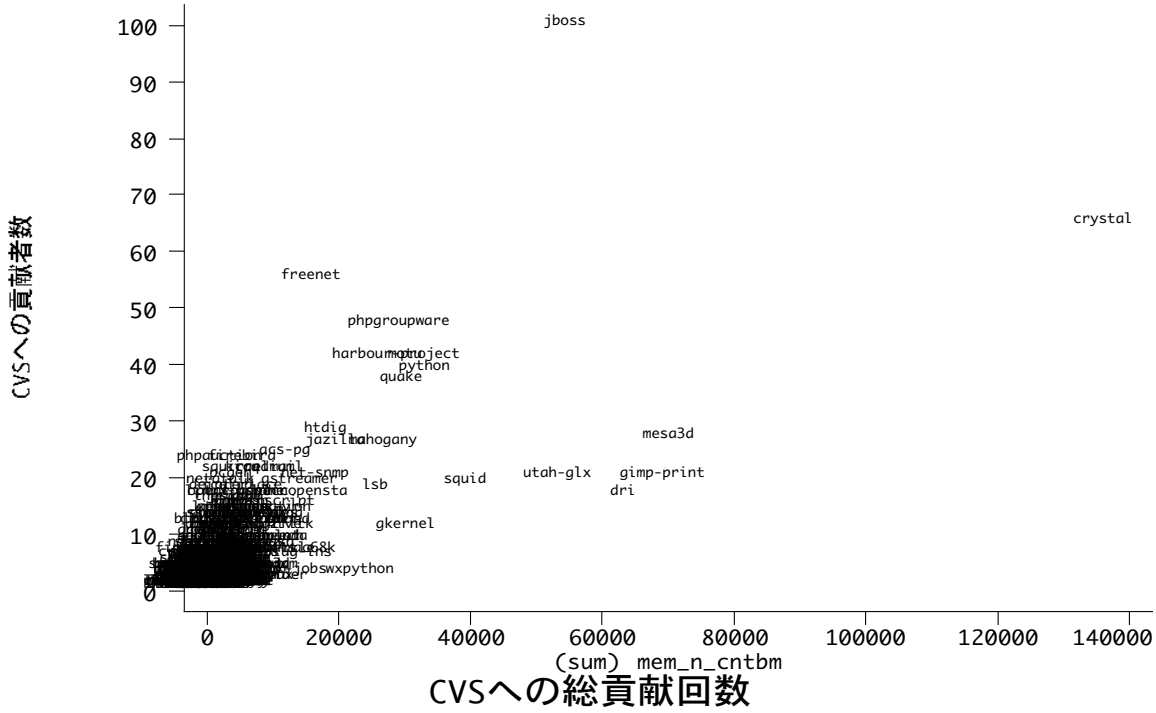
図表 CVSへのコミット回数の分布（N=941）



・開発の分担

各プロジェクトのCVSへのコミットメント回数とCVSへの貢献者数をプロットした。おおむね右上がりの関係がある。貢献者数が多くなるほど貢献回数が多くなる傾向が読み取れる[注13]。

図表 CVSへの総貢献回数と貢献者数の関係（N=941）



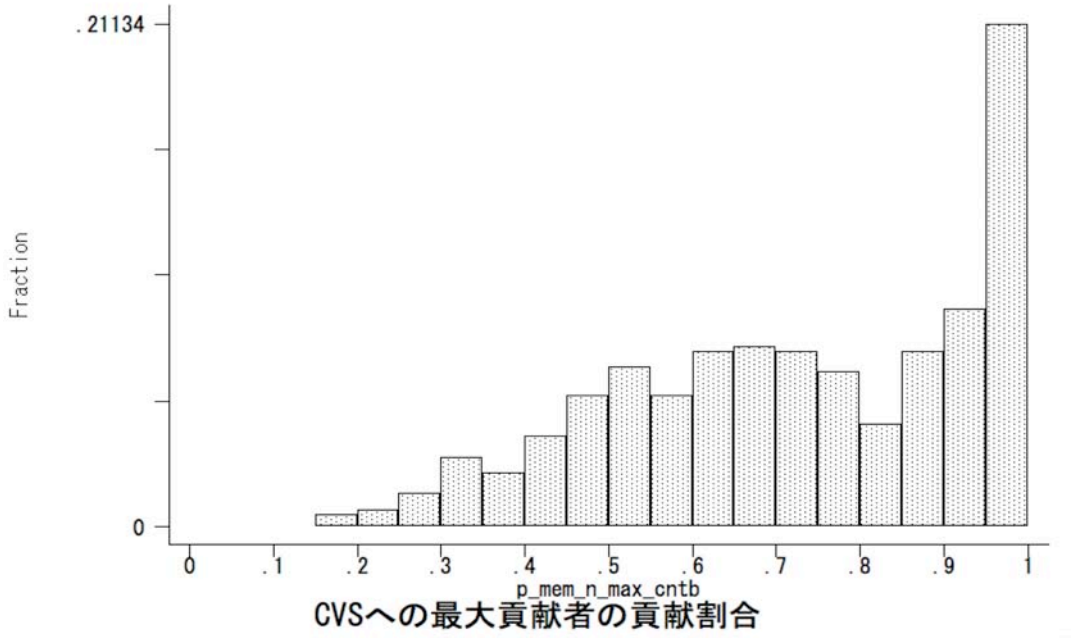
開発者の数だけでなく、開発が共同で行われることも重要な指標である。各プロジェクトについて、CVSにコミットした回数が最も多い者の、全CVSコミット回数に占める割合を算出し、

[注12] CVSでは、マニュアルなどのファイルも管理されている。ここでの開発とは、これらへの貢献回数もカウントされている。

[注13] 以下の散布図ではプロジェクト名をプロットする。

ヒストグラムにまとめた。この数値が大きいほど、１人で多くの開発を行っていることになる。ヒストグラムが示すように、21%以上のプロジェクトで、１人が全体の９０%の貢献をしている。この他、最大貢献者の貢献割合が50%を超えるプロジェクトが全体の7割以上を占めている。

図表　最大貢献者数の総貢献回数に占める割合（ヒストグラム）



注）N=コントリビューター（貢献者）数が２以上の548プロジェクト。

貢献回数の偏りを見るために、貢献回数についての正規化エントロピーと貢献者数をプロットした。正規化エントロピーは、貢献者がすべて同じ割合だけ貢献していれば１、一人だけが貢献していれば０になる[注14]。

人数が多くなれば分散度が高くなるのだとすれば、右上がりの傾向がみられるはずだが、そのようにはなっていない。

なお、右端にあるプロジェクトは貢献者数が100人ともっとも多い”jboss″プロジェクトである。

図表　CVSへの貢献者数と貢献回数の分散度

[注14] 分布の偏りを著す指標として、ここでは「正規化エントロピー normalized entropy」を用いる。
これは次式で与えられる。

$$e_{std} = -e/e_0$$

ここで、eは正規化前のエントロピーであり、コントリビューターの番号を i、プロジェクトへのコントリビューターの数をn、コントリビューターiのコントリビューション回数を c_i とすると次式で与えられる。

$$e = -\Sigma \ p_i \log p_i$$

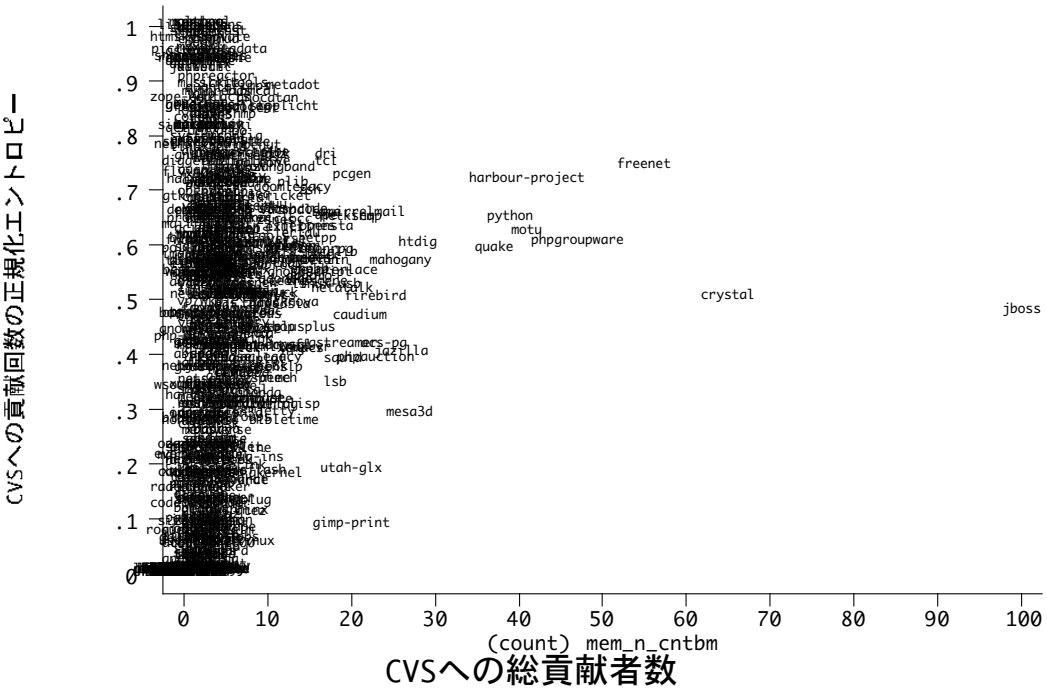
ここで、

$$p_i = c_i / \Sigma c_i$$

なお、正規化前のエントロピーeは、nとともに大きくなるため、コントリビューターの数が異なるプロジェクト間での比較ができない。このため、下記の標準化のための係数e_0で除して標準化する。

$$e_0 = -(1/n) \ \Sigma \ \log (1/n) = \log n$$

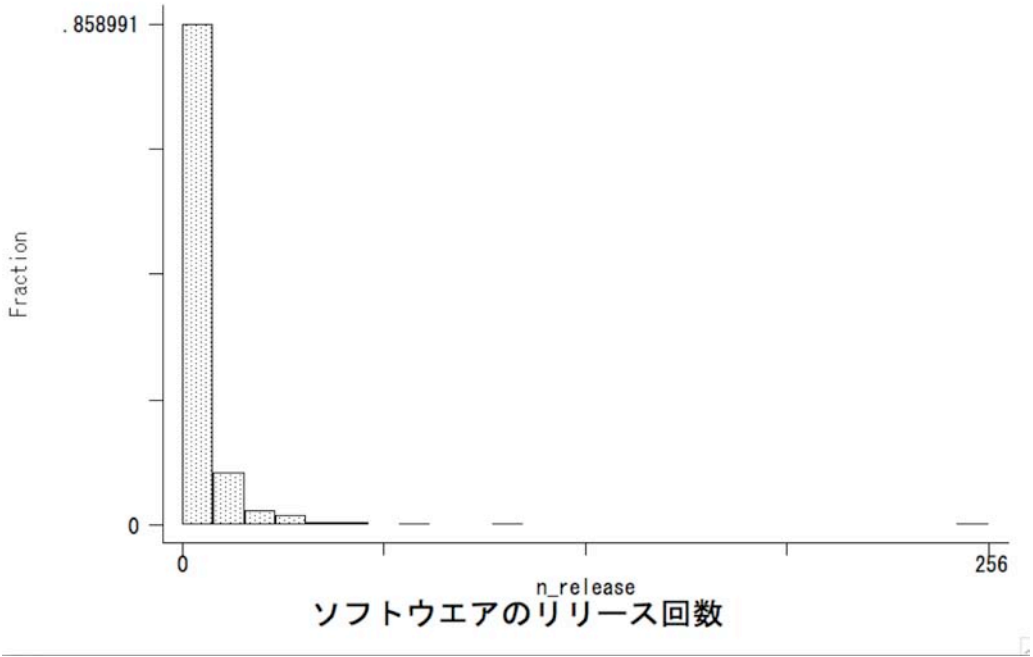
これによって、参加人数によらず、0-1の間の値をとる「正規化されたエントロピー」となる。これが1の場合、すべてのコントリビューターがそれぞれ同じ回数だけ貢献していることになり、0の場合、参加者のうち、一人だけが貢献していることになる。なお、参加者数が1人(n=1)の場合、e_0=0となり、正規化されたエントロピーが計算できないが、この場合は0と定義する。



・ソフトウエアの改訂 update

RaymondはLinuxの開発方式の特徴を、「早めのリリース、頻繁なリリース」という言葉で表現している。これは、多くのユーザーがバグを見つけてくれることを前提として、完成度を高める前から公開した方がよいというものである。リリース回数が１回のみ、つまり公開後、一度も更新していないプロジェクトが326、1回のみ更新しているものが147プロジェクト、2回のみが108プロジェクトである（ソフトを公開している1575プロジェクトの、それぞれ20.7%、9.3%、6.9%）。このように「オープンソースソフトウエアは頻繁にアップデートしている。」という指摘は必ずしも成立していないことがわかる[注15]

図表 ファイルのリリース回数の分布



注)N=1018。すべてのプラットフォーム、ファイルについてカウントした。ファイル更新回数が最大なのは229回のMySqlプロジェクト。

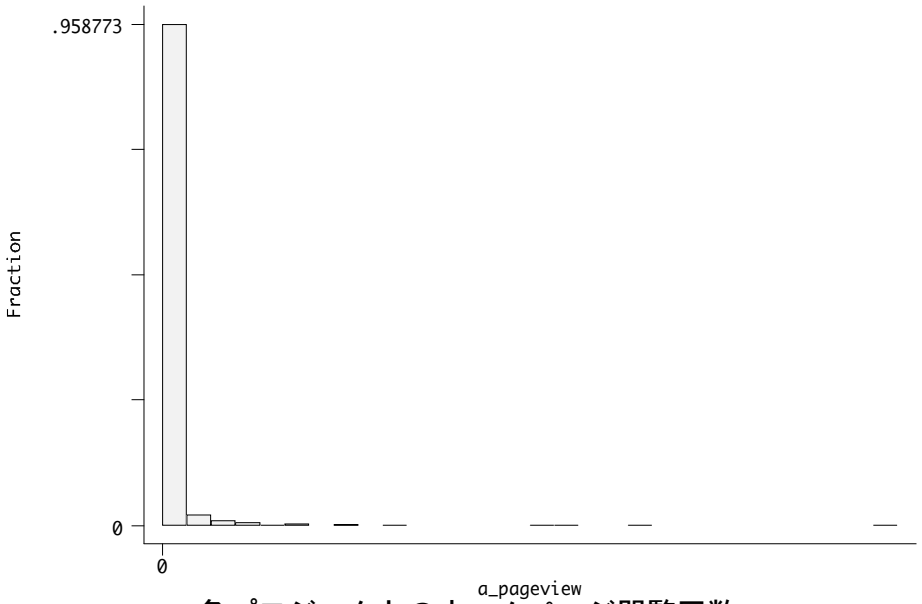
注15　なお、リリース回数が最も多いのはMysqlであり、Linux, BSD, Windowsなど複数のプラットフォームでのソフトウエアを開発している。

3.2　「市場における成果」

・ページビュー

　各プロジェクトのホームページ閲覧回数をヒストグラムにまとめた。　5,943,714回ページビューのプロジェクトもあれば、9回しか閲覧されないプロジェクトまでと、大きな格差がある。

図表　各プロジェクトのページビュー

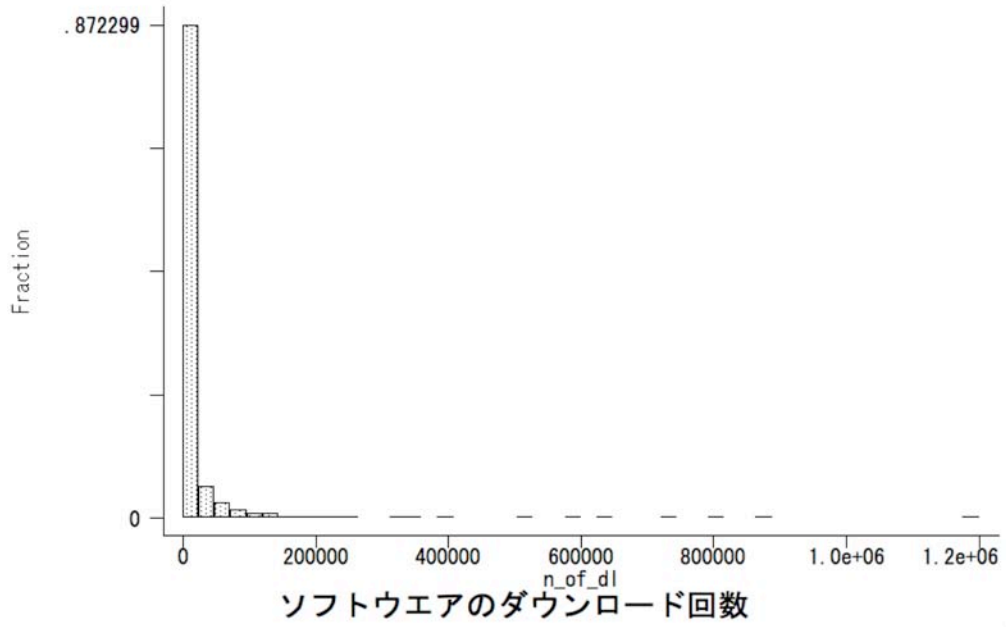


各プロジェクトのホームページ閲覧回数

・ダウンロード回数

　ファイルを公開しているプロジェクトの平均ダウンロード回数は、 16,663回。ただし、もっとも少ないものの0回から、もっとも多いプロジェクトの　1,199,080回まできわめて多きな格差があることがわかる。ヒストグラムをみても、ダウンロード数が少ない方に偏っていることがわかる。

図表　開発したソフトウエアのダウンロード回数の分布



ソフトウエアのダウンロード回数

3.3　コミュニティへのインパクト

・コミュニティの大きさとメッセージの投稿

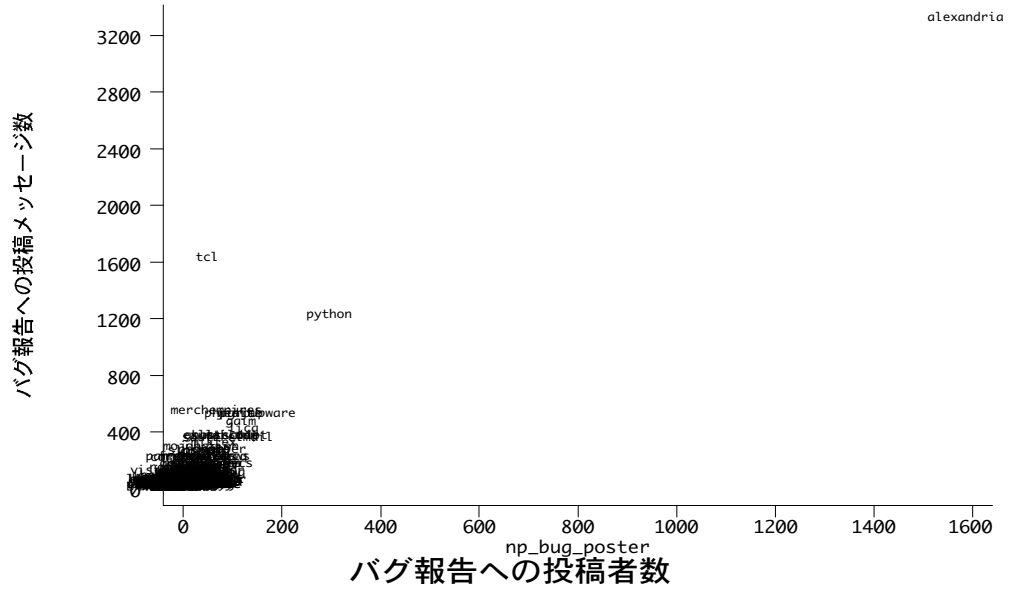
　Sourceforgeでは、バグ報告bug report、サポート要求support request、 機能追加要求

feature request、自由論題フォーラム、メーリングリスト[注16] でメッセージを交換することができる。それぞれについて、各プロジェクトのメッセージの投稿者数と投稿されたメッセージ数をプロットした。

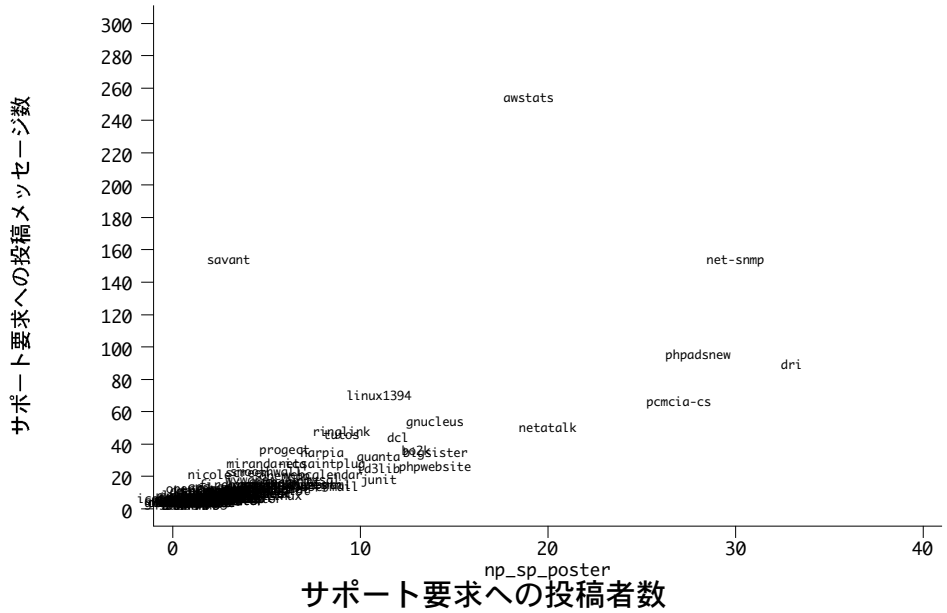
″alexandria″プロジェクトについては、バグ報告、サポート要求について他のプロジェクトよりも多くなっている。このプロジェクトは、Sourceforge.netのサイト運営に使われているソフトウエアそのものを開発するためのプロジェクトである。Sourceforge.netを訪れるユーザーならば必ず利用するサイトであり、ユーザーの数も多い。いずれも概ね右上がりの関係、つまり投稿者数が多くなるとメッセージ数も増加することが読み取れる。

図表　メッセージの投稿者数とメッセージ数

(1)バグ報告bug report



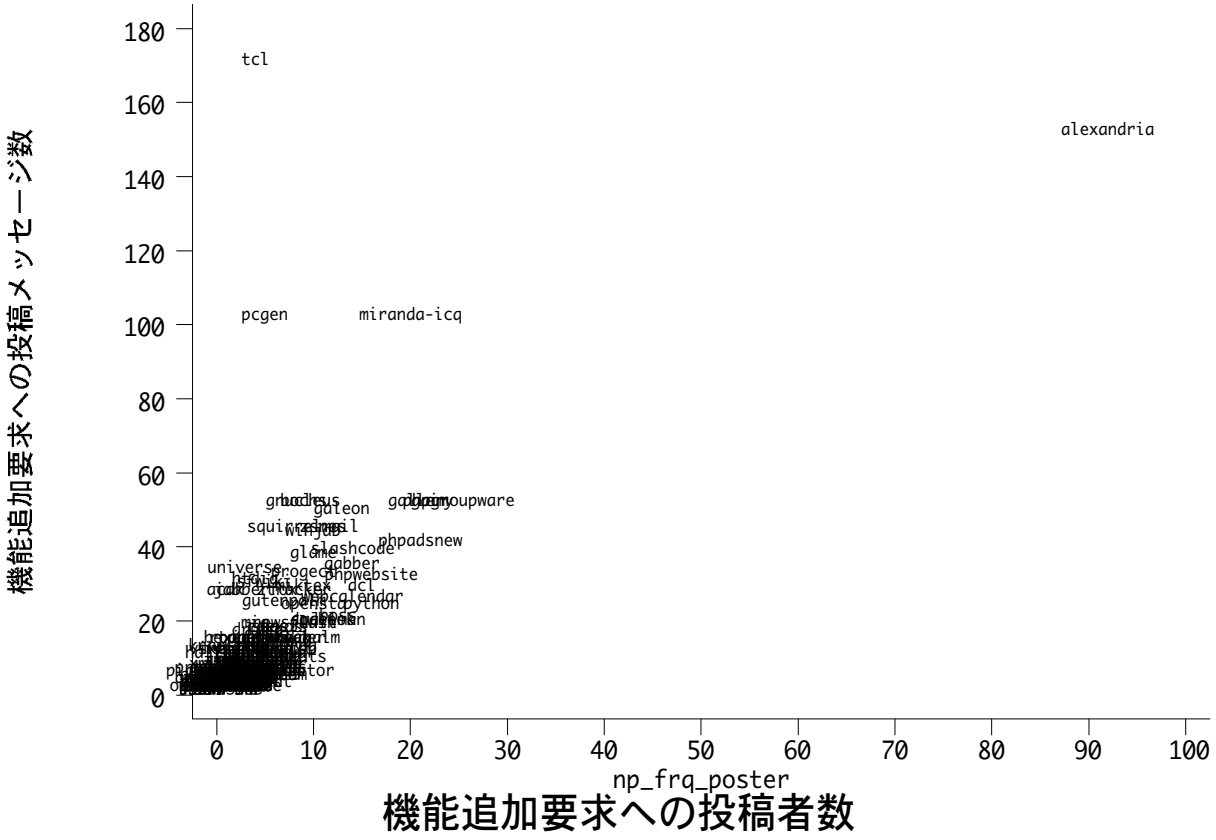
(2)サポート要求support request



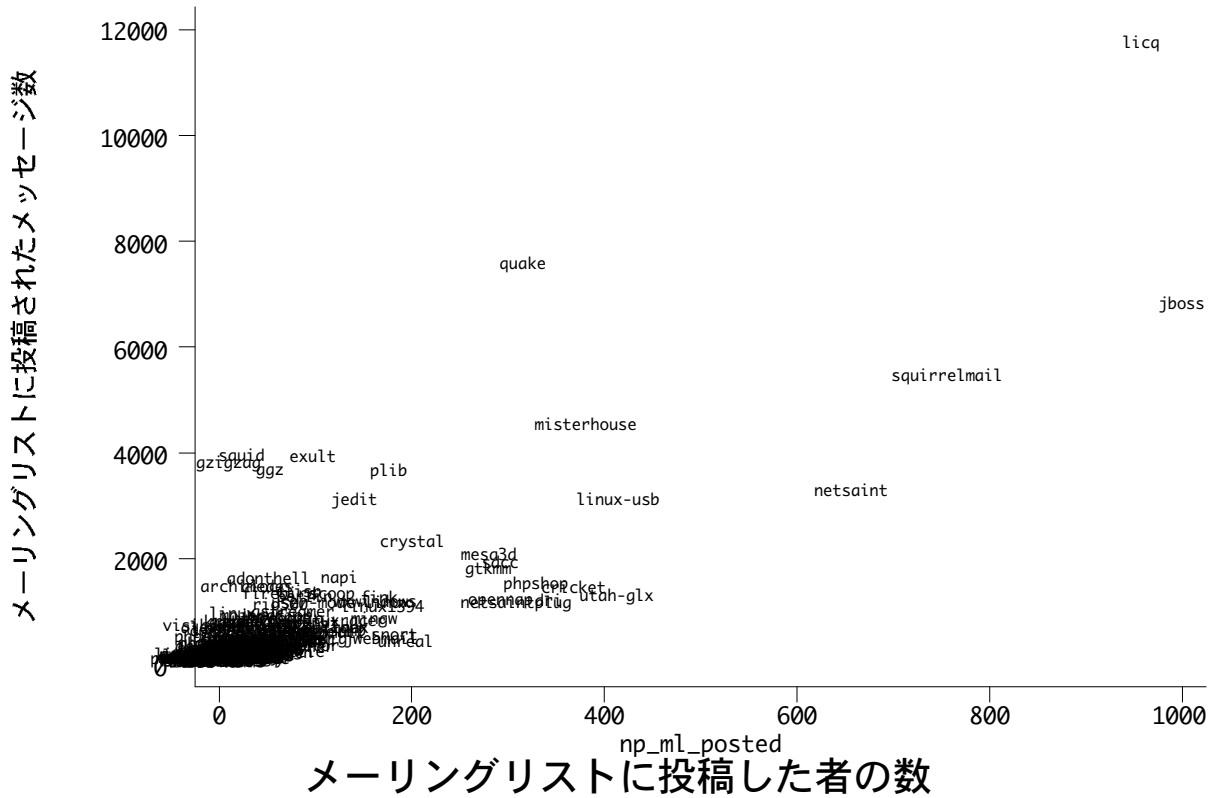
注）5990人が14152メッセージを投稿したalexandriaを除く

[注16] 自由論題フォーラム、メーリングリストについては、必要ならばいくつでも設置することが可能である。

(3)機能追加要求 feature request



(4)メーリングリスト

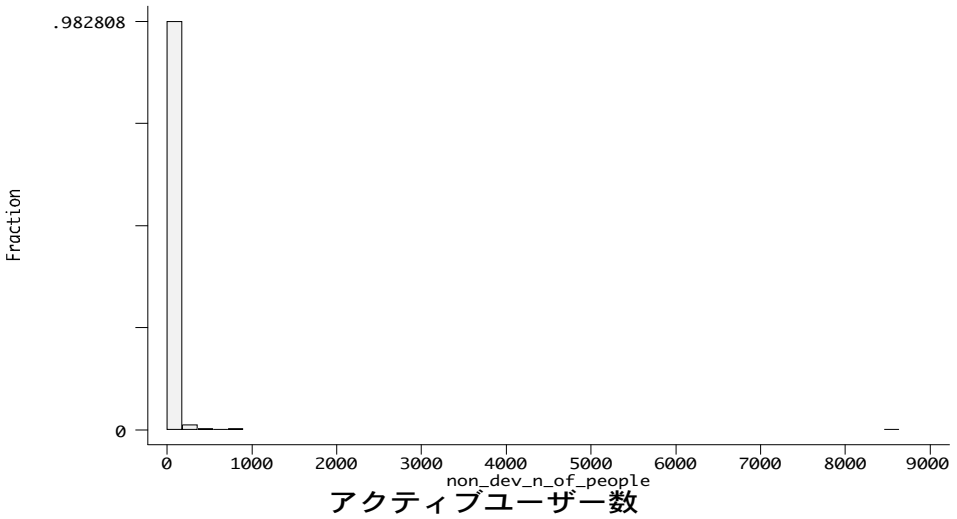


・アクティブ・ユーザー数

Sourceforgeでは、ダウンロード回数は公開されているが、ダウンロードした「人」の数は記録されていないため、ユーザーの数そのものを知ることはできない。そこでここでは、フォーラム(バグレポート、機能追加要求、パッチプログラム告知、サポート要求、論題自由）[注17] の投稿者をカウントした。つまり自分でメッセージを投稿したことが一度でもある「アクティブユーザー」である。

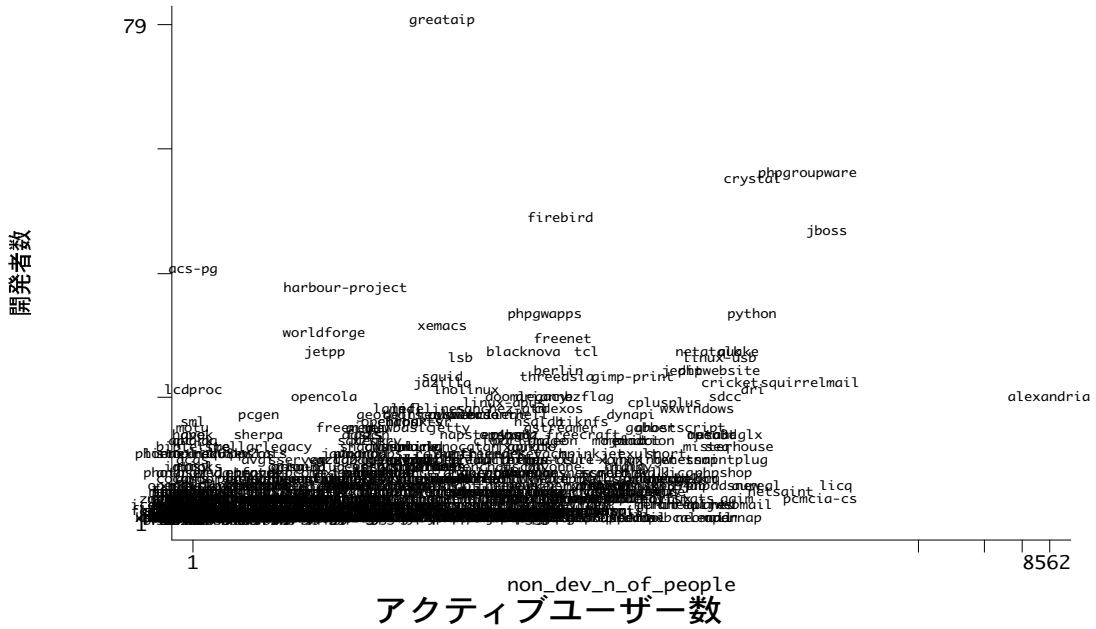
上のメッセージ投稿者数でみたように、8,562名の"alexandria"が最も多い[注18] 。800名を超えるものが２つあるが、平均は17.1人と多くはない。

図表　アクティブユーザー数の分布



・ユーザと開発者

アクティブユーザーの数と開発者の数をプロットした。コミュニティが増加すれば、開発コミュニティも拡大するのであれば、右上がりの関係が観察されるはずだが、このグラフでは、そのような傾向はみられない。



注）横軸は対数であることに注意

・問題への対応

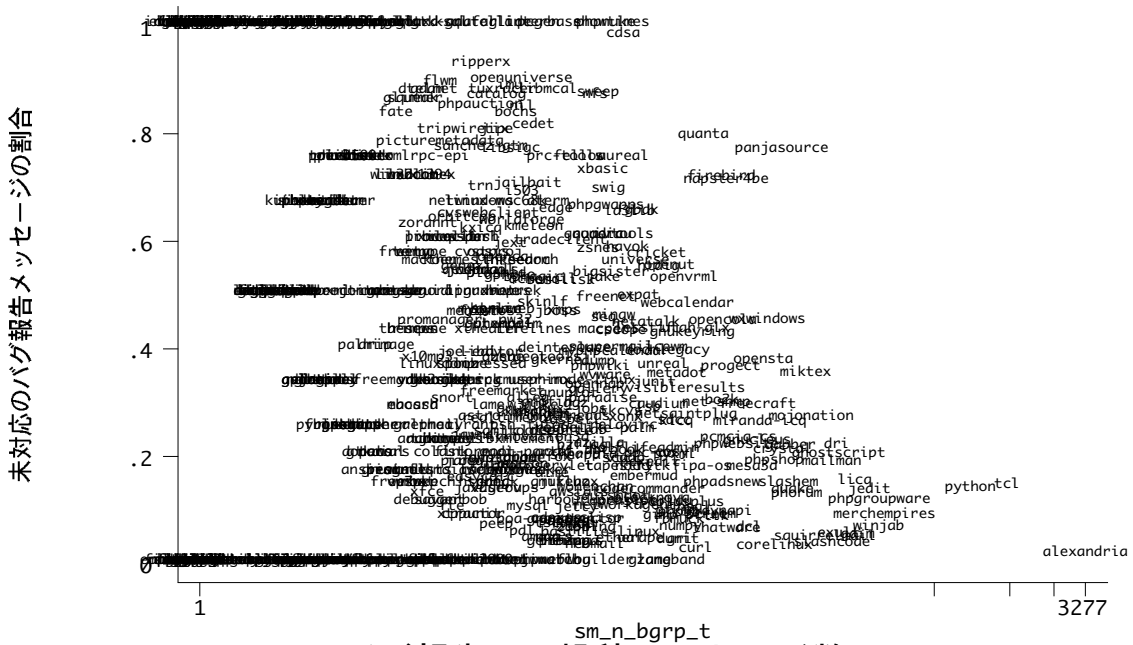
注17 メーリングリストについては、Sourceforgeにおけるユーザー名が利用されておらず、マッチングできないので分析からは除いた。

注18 今回のデータをダウンロードした2001年5月時点のSourceforgeへの登録ユーザー数は20万人程度であった。アクティブユーザーの割合はSourceforge登録者の4%程度ということになる。

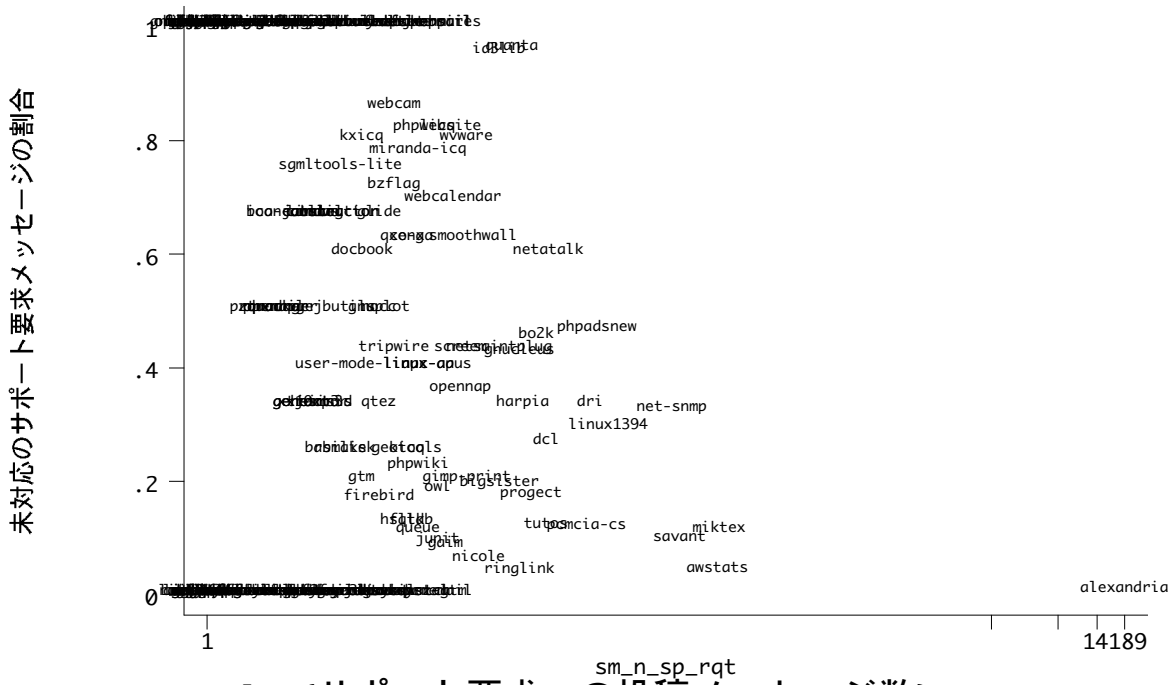
バグ報告、サポート要求、機能塚要求、パッチ確認では、メッセージが投稿されると、開発者らがそのメッセージを読み、対応する者を決めて解決に当たる。「投稿されたメッセージ数」と、「メッセージのうち問題が未解決なものの割合」をプロットした。横軸は対数であるが、右下がりの傾向にある。つまり、対応してくれるからメッセージがさらに増加するという関係にあるコトが推察される[注19]。

図表　ユーザー間での対応状況

(1)バグ報告bug report

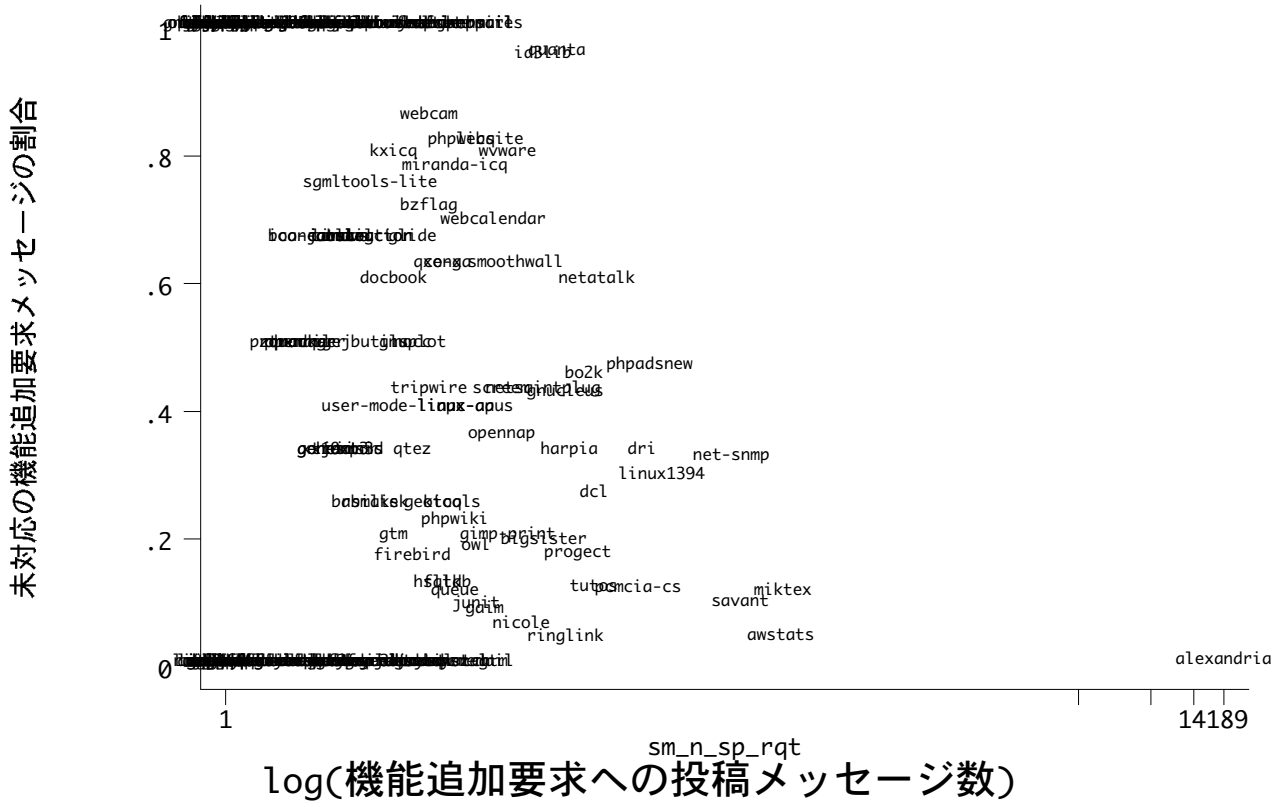


(2)サポート要求support request

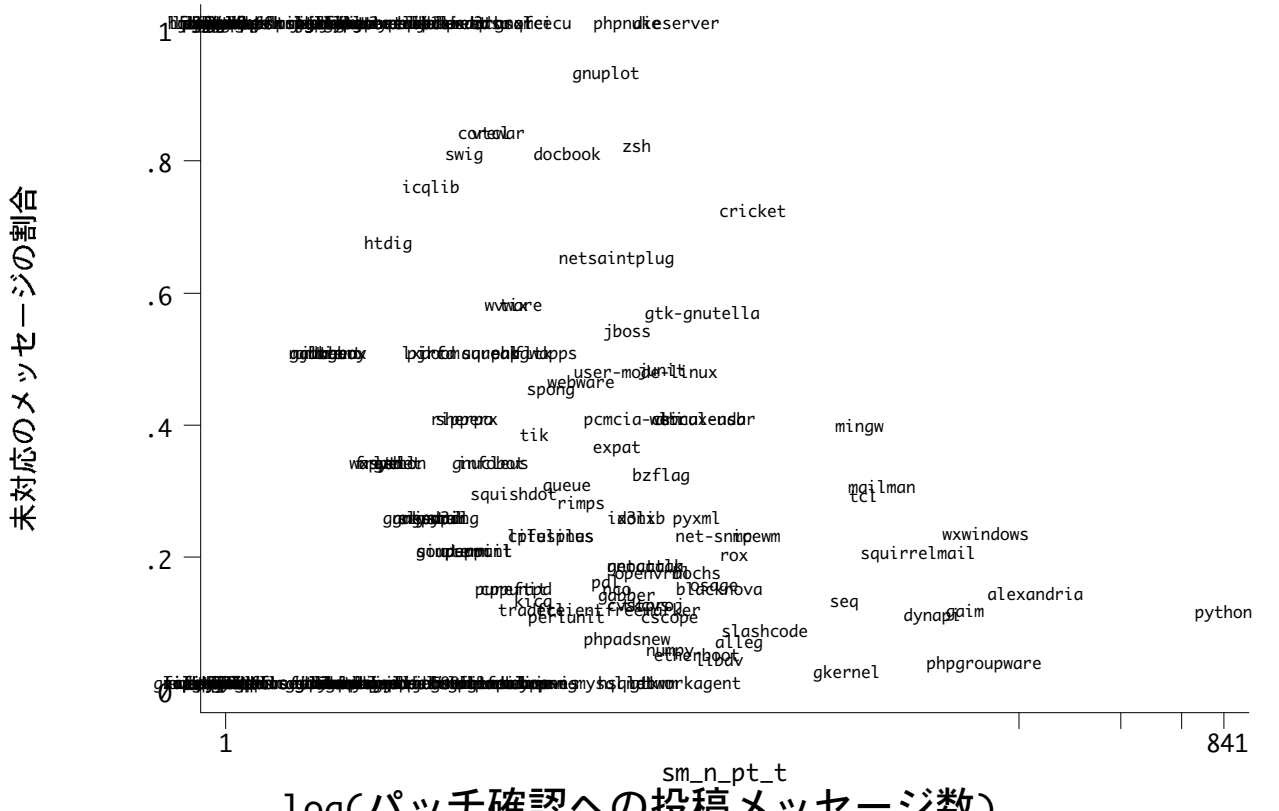


(3)機能追加要求 feature request

[注19] すべてのメッセージが公開されているので、メッセージが投稿されてから解決されるまでの期間を知ることもできる。本研究では、期間については無視したが、対応の早さに注目した分析も可能である。



(4)パッチプログラム確認要求 patche request



・ユーザー間でのディスカッション

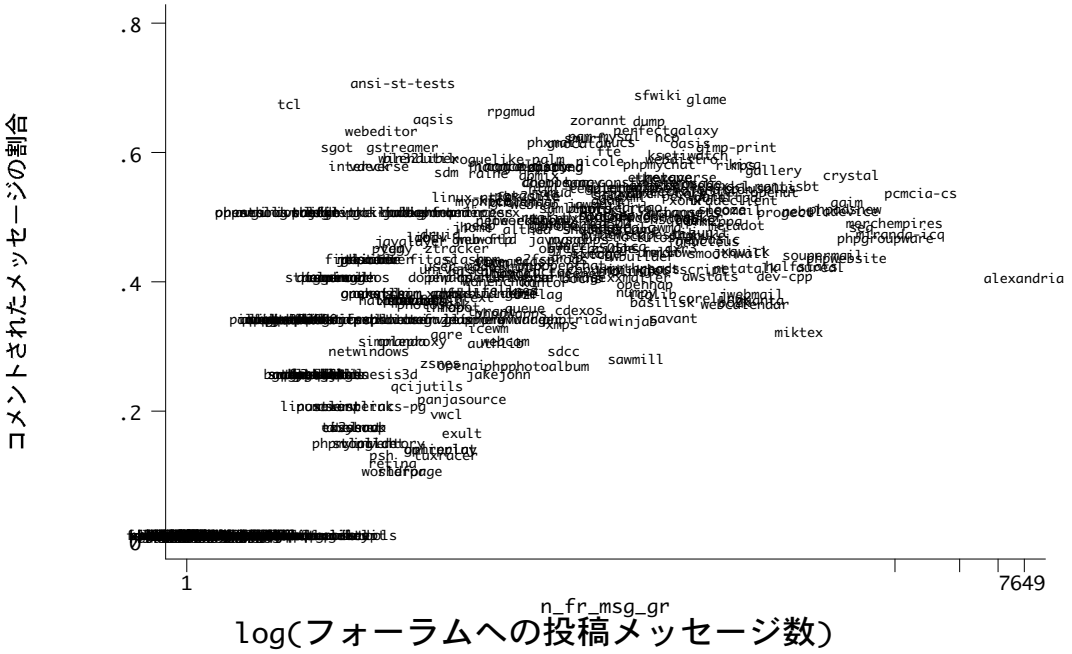
Sourceforgeでは、trackerとあわせて、フォーラムも開設できる。Trackerがバグ報告、

サポート要求、機能追加要求といった用途に限定されるのに対して、フォーラムでは、自由なテーマで投稿が行われる。つまり、担当が割り当てられる訳ではなく、メッセージを読んだ者の自発的な意思によって、メッセージに対してコメントされるので、ユーザー間でのインフォーマルな議論が行われることが期待される。

メッセージが投稿されるだけでは、コミュニケーションは成立しない。投げかけられたメッセージに対して、誰かが返事をすることがコミュニケーションのはじまりである。そこで、「フォーラムに投稿されたメッセージ数」と「フォーラムに対して投稿されたメッセージのうち、コメントがついたメッセージの割合」をプロットした。

右上がりの傾向にあるということは、これは、メッセージが多い、つまり投稿者が多いほど、様々な興味を持った者が集まり、それによって多様なメッセージに対応できる、または、コメントがつくことによって投稿意欲が増すといったメカニズムが作用していると考えられる。

図表　フォーラムへの投稿メッセージ数とそのうちコメントされたメッセージの割合

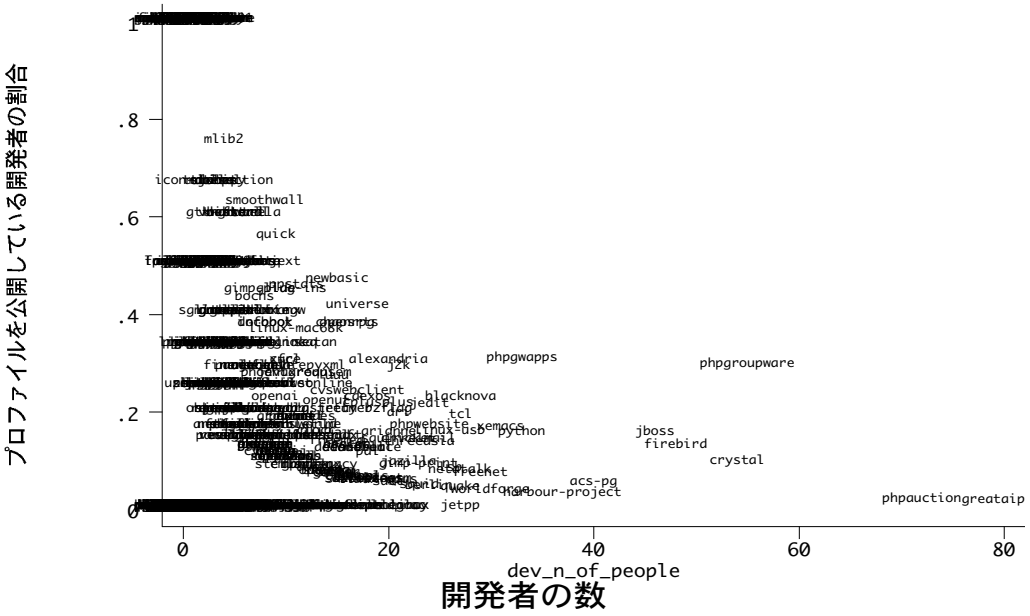


注）横軸は対数目盛。

・個人情報の開示

Sourceforgeに登録したユーザーは、自分のスキルなどのプロファイル情報を公開することもできる。開発者のうち、このプロファイルを公開している者の割合をプロットした。右下がりの傾向がみられる。

図表　開発者の数とプロファイルを公開している開発者の割合



3.4　成果指標によるプロジェクトの分類

ここでは前節で挙げたパフォーマンス指標のうち、Sourceforgeで入手可能なものを個別にみてきた。ここでは、これら指標すべてを考慮してオープンソース・ソフトウエア・プロジェクトを分類してみる。前述のようにプロジェクトによって、CVS、自由論題フォーラム、Trackerなどの利用状況は異なる。ここでは　参考までに、ここで分析に用いた主要な変数を用いてクラスター分析することによって、プロジェクトを分類してみる。

ここで用いた変数は下記の通りである。前述の通り、プロジェクトによってソフトウエア公開してないもの、CVSを使っていないもの、などがあり、欠損値が多い。ここでは欠損値はすべて０と置き換えることによって2101プロジェクトの分類を行う。

まず、因子分析によってこれらの変数から4つの因子を抽出した。

図表　因子分析の結果

	\	1	2	3	4
開発者の数	s	0.455	0.507	-0.083	-0.185
貢献者の数	r	0.422	0.641	-0.395	0.182
貢献回数	r	0.509	0.707	-0.160	-0.113
貢献についての正規化エントロピー	r	0.333	0.380	0.113	-0.283
プロジェクトホームページのページビュー	a	0.446	0.159	0.227	0.326
リリース回数	r	0.240	0.242	0.305	0.090
ダウンロード回数	r	0.264	0.201	0.321	0.410
バグ報告メッセージ数	r	0.835	-0.235	-0.005	-0.129
サポート要求メッセージ数	r	0.734	-0.554	-0.202	-0.048
機能追加要求メッセージ数	r	0.648	-0.096	0.209	-0.180
アクティブユーザー数	r	0.837	-0.443	-0.161	0.000
バグ報告の解決率	p	0.453	0.372	0.374	-0.160
サポート要求の解決率	p	0.376	0.176	0.252	0.062
機能追加要求の解決率	p	0.254	0.105	0.267	-0.161
フォーラムへのメッセージ数	r	0.856	-0.415	-0.039	0.101
フォーラムへのメッセージ数	r	0.842	-0.370	-0.027	0.111
公開されたソフトウエアに含まれるファイル	r	0.417	0.652	-0.324	0.128
因子の解釈		ユーザーフィードバック	ＣＶＳ貢献	解決率	ページビュー、ダウンロード

抽出された因子を用いて、まず階層型のクラスター分析を行いデンドログラムを描いた。その結果、６程度のクラスターがみられたため、クラスター数を6として階層型のクラスター分析を行った。各クラスタは次のような特徴を持っている。

・第1クラスタ
解決率が高いという特色をもった384プロジェクト。

・第２クラスタ
ページビュー・ダウンロードが高いという特徴をもった19プロジェクト。

・第3クラスタ
ユーザーフィードバックが非常に高いSourceforge(alexandria)のみ。CVS貢献、ページビューなどは低い。用途が限定されるからだと考えられる。

・第4クラスタ
CVS貢献つまり開発が活発な24プロジェクト。

・第5クラスタ
開発およびユーザーからのフィードバックが活発だが、解決率やページビュー・ダウンロード数は平均よりも低い4プロジェクト。ユーザー層が限定されている可能性が高い。

・第6クラスタ
すべてにおいて平均よりも若干低い、1669プロジェクト。

この結果に見られるように、開発、市場、コミュニティともに高いオープンソース･ソフトウエア・プロジェクトは存在しないことがわかる。

図表　非階層クラスター分析の結果（クラスター別の因子得点の平均値など）

		クラスタ番号					
		1	2	3	4	5	6
サンプル数		384	19	1	24	4	1669
因子の平均値	ユーザーフィードバック	0.06	0.73	43.35	0.15	1.04	-0.05
	ＣＶＳ貢献	0.24	0.29	-2.26	4.74	14.56	-0.16
	解決率	1.29	0.69	-1.71	-0.34	-0.60	-0.30
	ページビュー、ダウンロード	0.01	6.15	-2.16	0.12	-0.39	-0.07
含まれるプロジェクト		sgmltools-lite	The Freenet Prc	SourceForge	Harbour	Crystal Space 3	yapp
		CLOCC - Commc	CDex		Squid HTTP Pro	Direct Renderin	Fast Light Windo
		APC Action App	galeon		gkernel	jboss.org	dxflib
		Geheimnis	AFPL Ghostscript		GStreamer	Print plugin for	K-LUG Communit
		ztracker	Quanta+ Web Development Envir		wxPython		ORBit Resource
		myPHPCalendar	Back Orifice 2000		Utah-GLX		Production Basic
		Debugger Bob	Open Source Napster Server		Small Device C Compiler		REDLinux
		cscope	NetSaint Network Monitor		Firebird		PHPower
		Gnucleus	WebCalendar		Python		libpresence
		Gallery	MiKTeX		Linux for 68k Macintoshes		Perfect Galaxy
		CVSTProject	VirtualDub		net-snmp		pyGallery
		MinGW - Minima	Miranda ICQ Client		Linux Standard Base		LookOut
		qmailWebmin	jEdit		QuakeForge		glGUI
			Licq		Gimp Plug-ins CVS		Visual Perl
			sawfish		Mahogany mail and news client		phpWebRing
			Linux Aureal Driver		OpenSTA		Download It
			MySQL		ht://Dig		The MindX Proje
			WebMail/Java		OpenACS		GNU-CService
			Tux Racer		Qt Palmtop Environment		Target Campaign
					Mesa3D		Partition Image
					The Jazilla Project		GNU Work Group
					Linux VR		webmud
					MOTU		KSoundStudio
					phpGroupWare		phpMimeMail
							phpKISSdiary
							PHPIsis

注）第6クラスタについては、一部のプロジェクト名のみを示した。

3.5　本節での分析の総括

本節ではSourceforgeにおけるプロジェクトについて、前節で提示した成果指標のうち「開発」「市場での成果」「コミュニティへのインパクト」に沿ってプロジェクト間での比較を行った。すべての指標について、大きな成果を得ているのはほんの少数であることが示された。

・開発そのものについて

25%のプロジェクトではソフトウエアが公開されていない。また公開されるまでの期間が長いプロジェクトも多い。

早めのリリース、頻繁なリリースがLinuxの開発の特徴としてしてされているが、リリースが１回のみというプロジェクトの割合が高い。

開発者数が増えると分担が進むといった明確な関係もみられない。

開発者数は多いプロジェクトでも100人であり、多くのプロジェクトでは１人もしくは数人で開発している。

・市場での成果
認知度（ページビュー）、ダウンロード回数ともにプロジェクトによって大きなばらつきがある。

・コミュニティへのインパクト
ダウンロードした者の数は入手不可能であるため、メッセージを投稿したことがある者、つまりアクティブユーザーについてカウントしたが、これも大きなばらつきが見られた。

4. オープンソース･ソフトウエア・プロジェクトのパーフォーマンスと規定要因についての実証分析

前節ではオープンソース・ソフトウエアの成果指標について概観した。ここでは、2節で示したオープンソース・ソフトウエアの成果要因および、規定要因について、具体的な仮説を設定する。そして、これら仮説について、同じくSourceforgeのデータを用いて検定する。

4.1　仮説の設定

このデータセットでは、ソフトウエアそのもの、および成員の個人レベルについての変数は得られていない。このため、開発、市場、コミュニティへのインパクトの３つの領域に注目して仮説を設定する。

・開発の活発度について

開発者が多くなるほど開発は活発になるだろう。一方、開発が特定の人に集中すると開発は進まなくなるだろう。ユーザーからのバグ報告、機能追加要求によって開発（修正）が行われるだろう。さらにソフトウエアが複雑だと開発にも多くの努力が投入されるだろう。これらのことから以下の仮説を設定する。

H_{dev1}　：「開発の活発度」と「開発者の数」との間には正の相関がある。

H_{dev2}　：「開発の活発度」と「開発における分担の分散度」との間には正の相関がある。

H_{dev3}　：「開発の活発度」と「ユーザーからのバグ報告数」との間には正の相関がある。

H_{dev4}　：「開発の活発度」と「ユーザーからの機能追加要求数」との間には正の相関がある。

H_{dev5}　：「開発の活発度」と「ソフトの複雑度」との間には正の相関がある。

・開発者の数について

開発者が新しく加わるには、開発者コミュニティのオープンさが必要だろう。また、ユーザー数が多ければ、開発に加わる者も増加するだろう。これらより次の仮説を設定する。

H_{ndev1}　：「開発者の数」と「開発者のスキルの公開度」との間には正の相関がある。

H_{ndev2}　：「開発者の数」と「ユーザー・コミュニティの規模」との間には正の相関がある。

・開発の分散度について

ソフトが複雑であればｱﾙほど、分担して開発する必要が高くなる。また、開発者数が多いほど、開発の分担がより可能となるだろう。このことより次の仮説を設定する。

H_{dsip1}　：「開発の分散度」と「ソフトの複雑さ」との間には正の相関がある。

H_{dsip2}　：「開発の分散度」と「開発者の数」との間には正の相関がある。

・リリース回数について

開発が活発に行われれるほどソフトがアップデート、公開されるだろう。このことを次の仮説として設定する。

H_{up}　：「ソフトウエアのリリース回数」と「開発の活発度」との間には正の相関がある。

・ダウンロード回数について

プロジェクトが認知されているほど、サイトを訪れ、ダウンロードする者が増加するだろう。また、新しいソフトウエアが公開されれば、同じユーザーでもダウンロードするだろう。

これらから次の二つの仮説を設定する。

H_{down1}　：「ソフトウエアのダウンロード回数」と「プロジェクトの認知率」との間には正の相関がある。
H_{down2}　：「ソフトウエアのダウンロード回数」と「ソフトウエアのリリース回数」との間には正の相関がある。

・ユーザー・コミュニティの規模について
ダウンロード回数が増加すれば、ユーザーは増加するだろう。また、ユーザー間でのコミュニケーションが密になるほど、ユーザー間での信頼などが向上するだろう。このことを次の仮説に設定する。

H_{com1}　：「ユーザー・コミュニティの規模」と「ソフトウエアのダウンロード回数」との間には正の相関がある。
H_{com2}　：「ユーザー・コミュニティの規模」と「ユーザー間でのコミュニケーション密度」との間には正の相関がある。

・ユーザーからのフィードバックについて
Raymond(1998)は、”多くの目玉”があれば多くのフィードバックがあることを指摘している。つまり、時間をかけてデバッグをして完成度を高めるよりも、多くのユーザーによって利用してもらい、バグを見つけてもらい修正する方が効率的だというのである。このことから以下の仮説を設定する。

H_{bug1}　：「ユーザーからのバグ報告の数」と「ユーザー・コミュニティの規模」との間には正の相関がある。
H_{feat1}　：「ユーザーからの機能追加要求」と「ユーザー・コミュニティの規模」との間には正の相関がある。

H_{help1}　：「ユーザーからのサポート要求」と「ユーザー・コミュニティの規模」との間には正の相関がある。

報告をしても、それに対応、解決されなければ投稿する意欲をなくすだろう。逆に対応が迅速になされれば投稿する意欲も増すだろう。このことから以下の仮説を設定する。

H_{bug2}　：「ユーザーからのバグ報告の数」と「バグ報告の解決率」との間には正の相関がある。

H_{feat2}　：「ユーザーからの機能追加要求」と「ユーザーからの機能追加要求の解決率」との間には正の相関がある。
H_{help2}　：「ユーザーからのサポート要求」と「ユーザーからのサポート要求の解決率」との間には正の相関がある。

・解決率について
問題解決には多様な知識が必要である。ユーザーコミュニティが大きくなれば、多様な問題に対応することが可能となるだろう。このことから以下の仮説を設定する。
H_{res1}　：「バグ報告の解決率」と「ユーザー・コミュニティの規模」との間には正の相関がある。

H_{res2}　：「ユーザーからの機能追加要求の解決率」と「ユーザー・コミュニティの規模」との間には正の相関がある。

H_{res3}　：「ユーザーからのサポート要求の解決率」と「ユーザー・コミュニティの規模」との間には正の相関がある。

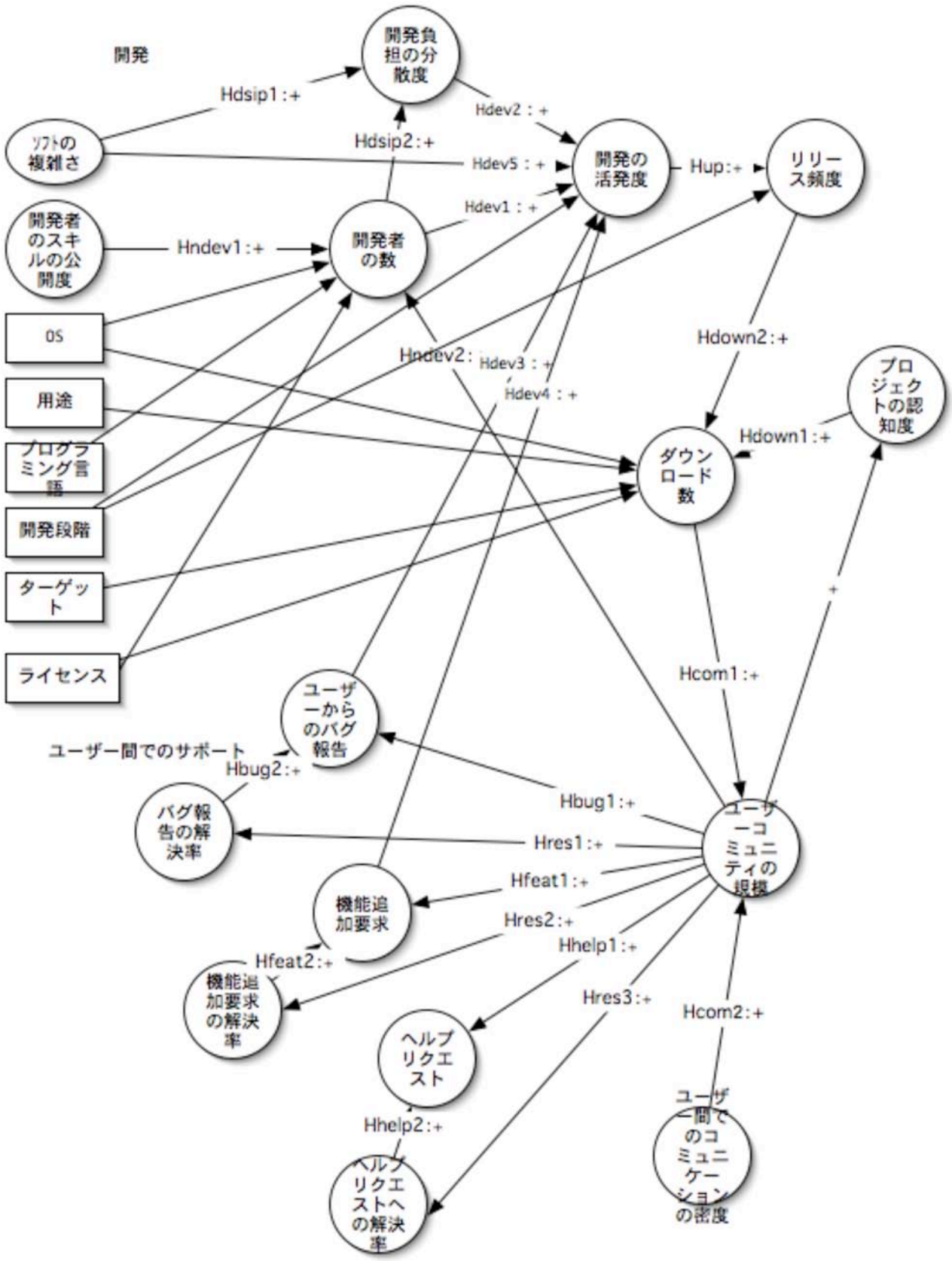
・プロジェクトの認知度について

ダウンロード回数が増加すれば、ユーザーも増加し、プロジェクトやソフトウエアについて、他者にクチコミするユーザーも増加するだろう。このことから以下の仮説を設定する。

Hrec*+　：「プロジェクトの認知度」と「ユーザー・コミュニティの規模」との間には正の相関がある。

設定した仮説は次のフロー図にまとめることができる。なお、ここで設定した仮説の記号、および仮説で想定される符号を矢印に示した。

図表　仮説と推定したモデル



注）矢印の向きは因果関係を、矢印の上のHという記号は対応する仮説および、想定される符号。

□は共変量として導入した変数。

4.2　仮説の検定

これらの仮説を線形の連立方程式で表現し、３段階線形回帰分析を行ってパラメーターを推定することによって仮説を検定した[注20]。前節でみたように、ダウンロード数、リリース回数など、パフォーマンス指標はばらつきが大きいため、対数をとった[注21]。仮説として想定した変数以外にも、「OS」「ソフトウエアの用途」「プログラミング言語」「開発段階」「ターゲットとするユーザー」「ライセンス」についてのダミー変数を共変量として加えた。　なお、次の条件を満たすプロジェクトのみを分析対象としたため、有効ｻﾝﾌﾟﾙ数は５００となった。

ソフトウエアを公開している
バグ報告Trackerシステムを利用
機能要求報告Trackerシステムを利用
サポート要求Trackerシステムを利用

推定の結果について、まず全体的なあてはまりの状況を示す。13本の方程式のうち、「機能要求メッセージの解決率」をのぞくとR2は0.15を越えている他、全ての方程式についてカイ２検定の結果が有意となった。

図表　推定結果（全体的なあてはまり）

従属変数	Equation	Obs	Parms	RMSE	"R-sq"	chi2	P
ダウンロード数	ln_of_dl	500	14	1.692815	0.5035	683.4212	0.0000
リリース回数	ln_release	500	4	.769831	0.1711	138.0827	0.0000
開発の活発度：コントリビューション回数	ln_cntb	500	8	.908709	0.7839	3580.667	0.0000
開発者の規模：CVSへのコントリビューター数	lnp_cntb	500	12	.590527	0.2778	430.3485	0.0000
コミュニティの規模：アクティブユーザー数	lall_n_of_~e	500	2	1.206958	0.1636	547.3151	0.0000
機能要求メッセージ数	lsm_n_freq_t	500	2	.7230784	0.1985	162.5745	0.0000
バグ報告メッセージ数	lsm_n_bgrp_t	500	2	.799832	0.7534	864.9261	0.0000
サポート要求メッセージ数	lsm_n_sp_rqt	500	2	.6189293	0.6295	301.8664	0.0000
機能要求メッセージの解決率	pcsm_n_fre~l	500	1	.1243794	0.0258	21.26611	0.0000
バグ報告メッセージの解決率	pcsm_n_bgr~l	500	1	.2853434	0.3014	240.7481	0.0000
サポート要求メッセージの解決率	pcsm_n_sp_~l	500	1	.2303447	0.2252	150.3965	0.0000
プロジェクトの認知度：ホームページビュー数	la_pageview	500	1	1.354301	0.5655	808.9213	0.0000
開発の分散度：正規化エントロピー	normed_enp~b	500	2	.2744602	0.1764	80.29228	0.0000

方程式ごとに回帰係数とそのz値、p値を示す。以下では上で設定した仮説を検定する。なお、仮説はすべて正の方向で示しているので、片側検定を行う。

・開発の活発度について

開発者の数についての偏回帰係数は0.727、z値は4.660でありz＞ $z_{0.01}$=2.32である。よって、この仮説は１％水準で支持される。

H_{dev1}　：「開発の活発度」と「開発者の数」との間には正の相関がある。

開発の分散度：正規化エントロピーについての偏回帰係数は0.216、z値は0.34でありz＜ $z_{0.1}$=1.28である。よって、この仮説は10％水準で棄却される。

H_{dev2}　：「開発の活発度」と「開発における分担の分散度」との間には正の相関がある。

[注20] 推定には、はSTATA/SE for Macのreg3プロシジャを用いた。

[注21] たとえばダウンロード0回の場合、対数が存在しない。このため実際には、各変数に１を加えたものの対数をとった。

バグ報告メッセージ数についての偏回帰係数は-0.385、z値は-4.20でありz< $z_{0.1}$=1.28である。よって、この仮説は10%水準で棄却される。

H_{dev3}　：「開発の活発度」と「ユーザーからのバグ報告数」との間には正の相関がある。

機能追加要求メッセージ数についての偏回帰係数は0.928、z値は5.07でありz> z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{dev4}　：「開発の活発度」と「ユーザーからの機能追加要求数」との間には正の相関がある。

ソフトウエアの複雑度（ソフトウエアのファイル数）についての偏回帰係数は1.04、z値は34.82でありz> z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{dev5}　：「開発の活発度」と「ソフトの複雑度」との間には正の相関がある。

・開発者の数について

開発者のスキルの公開度についての偏回帰係数は-0.069、z値は-0.84でありz< $z_{0.1}$=1.28である。よって、この仮説は10%水準で棄却される。

H_{ndev1}　：「開発者の数」と「開発者のスキルの公開度」との間には正の相関がある。

ユーザーコミュニティの規模についての偏回帰係数は0.514、z値は19.16でありz> z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{ndev2}　：「開発者の数」と「ユーザー・コミュニティの規模」との間には正の相関がある。

・開発の分散度について

ソフトウエアの複雑度：ソフトウエアのファイル数についての偏回帰係数は0.033、z値は3.64でありz> z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{dsip1}　：「開発の分散度」と「ソフトの複雑さ」との間には正の相関がある。

開発者数：CVSにコミットした者の数についての偏回帰係数は0.012、z値は5.17でありz> z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{dsip2}　：「開発の分散度」と「開発者の数」との間には正の相関がある。

・リリース回数について

開発の活発度：CVSへのコミット回数についての偏回帰係数は0.197、z値は11.18でありz> z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{up}　：「ソフトウエアのリリース回数」と「開発の活発度」との間には正の相関がある。

・ダウンロード回数について

プロジェクトの認知度：ホームページビュー数についての偏回帰係数は1.107、z値は17.61でありz＞ z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{down1}　：「ソフトウエアのダウンロード回数」と「プロジェクトの認知率」との間には正の相関がある。

リリース回数についての偏回帰係数は-0.161、z値は-0.92でありz＞ z0.01=2.32である。よって、この仮説は1%水準で棄却される。

H_{down2}　：「ソフトウエアのダウンロード回数」と「ソフトウエアのリリース回数」との間には正の相関がある。

・ユーザー・コミュニティの規模について
ダウンロード数についての偏回帰係数は0.620、z値は21.91でありz＞ z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{com1}　：「ユーザー・コミュニティの規模」と「ソフトウエアのダウンロード回数」との間には正の相関がある。

ユーザー間のコミュニケーション密度：フォーラムメッセージへのコメント率についての偏回帰係数は-0.074、z値は-0.72でありz＜ $z_{0.1}$=1.28である。よって、この仮説は10%水準で棄却される。

H_{com2}　：「ユーザー・コミュニティの規模」と「ユーザー間でのコミュニケーション密度」との間には正の相関がある。

・バグ報告について
ユーザーコミュニティの規模についての偏回帰係数は0.595、z値は5.66でありz＞ z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{bug1}　：「ユーザーからのバグ報告の数」と「ユーザー・コミュニティの規模」との間には正の相関がある。

バグ報告メッセージの解決率についての偏回帰係数は2.394、z値は4.54でありz＞ z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{bug2}　：「ユーザーからのバグ報告の数」と「バグ報告の解決率」との間には正の相関がある。

・機能追加要求
ユーザーコミュニティの規模についての偏回帰係数は0.193、z値は5.54でありz＞ z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{feat1}　：「ユーザーからの機能追加要求」と「ユーザー・コミュニティの規模」との間には正の相関がある。

機能追加要求メッセージの解決率についての偏回帰係数は5.218、z値は6.47でありz＞ z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{feat2}　：「ユーザーからの機能追加要求」と「ユーザーからの機能追加要求の解決率」と

の間には正の相関がある。

・サポート要求

ユーザーコミュニティの規模についての偏回帰係数は0.167、z値は3.17でありz>z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{help1}　：「ユーザーからのサポート要求」と「ユーザー・コミュニティの規模」との間には正の相関がある。

サポート要求メッセージの解決率についての偏回帰係数は2.447、z値は6.24でありz>z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{help2}　：「ユーザーからのサポート要求」と「ユーザーからのサポート要求の解決率」との間には正の相関がある。

・解決率について

ユーザーコミュニティの規模についての偏回帰係数は0.0257、z値は4.61でありz>z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{res1}　：「バグ報告の解決率」と「ユーザー・コミュニティの規模」との間には正の相関がある。

ユーザーコミュニティの規模についての偏回帰係数は0.195、z値は15.52でありz>z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{res2}　：「ユーザーからの機能追加要求の解決率」と「ユーザー・コミュニティの規模」との間には正の相関がある。

ユーザーコミュニティの規模についての偏回帰係数は0.125、z値は12.26でありz>z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{res3}　：「ユーザーからのサポート要求の解決率」と「ユーザー・コミュニティの規模」との間には正の相関がある。

・プロジェクトの認知度について

ユーザーコミュニティの規模についての偏回帰係数は1.511、z値は28.44でありz>z0.01=2.32である。よって、この仮説は1%水準で支持される。

H_{rec*+}　：「プロジェクトの認知度」と「ユーザー・コミュニティの規模」との間には正の相関がある。

図表　パラメーターの推定結果

従属変数	説明変数		偏回帰係数	z値	p値	対応する仮説番号	仮説の検定結果
開発の活発度：CVSへのコミット回数							
	開発者数：CVSにコミットした者の数		0.727	4.660	0.000	Hdev1	支持
	開発の分散度：正規化エントロピー		0.216	0.340	0.730	Hdev2	棄却
	バグ報告メッセージ数		-0.385	-4.200	0.000	Hdev3	棄却
	機能追加要求メッセージ数		0.928	5.070	0.000	Hdev4	支持
	ソフトウエアの複雑度：ソフトウエアの		1.044	34.820	0.000	Hdev5	支持
	開発段階	アルファ	0.106	1.330	0.180		
		ベータ	0.032	0.480	0.630		
		安定	0.090	1.170	0.240		
	定数項		0.355	3.170	0.000		
開発者数：CVSにコミットした者の数							
	開発者のスキルの公開度		-0.069	-0.840	0.390	Hndev1	棄却
	ユーザーコミュニティの規模		0.514	19.160	0.000	Hndev2	支持
	OSダミー	Linux	0.078	1.500	0.130		
		BSD	0.054	0.560	0.570		
		Microsoft	-0.075	-1.400	0.160		
		OS独立	0.201	3.210	0.000		
	プログラミング言語ダミー	C	0.056	0.990	0.320		
		C++	-0.024	-0.410	0.680		
		Perl	0.078	1.010	0.310		
		Php	-0.178	-1.990	0.040		
		Java	-0.157	-1.940	0.050		
	GPLライセンスダミー		-0.050	-0.960	0.330		
	定数項		0.072	0.800	0.420		
開発の分散度：正規化エントロピー							
	ソフトウエアの複雑度：ソフトウエアの		0.033	3.640	0.000	Hdsip1	支持
	開発者数：CVSにコミットした者の数		0.012	5.170	0.000	Hdsip2	支持
	定数項		0.000	0.000	0.000		
リリース回数							
	開発の活発度：CVSへのコミット回数		0.197	11.180	0.000	Hup	： 支持
	開発段階ダミー	アルファ	0.042	0.480	0.620		
		ベータ	0.054	0.760	0.450		
		安定	0.108	1.450	0.140		
	定数項		0.510	4.380	0.000		
ダウンロード数							
	プロジェクトの認知度：ホームページビ		1.107	17.610	0.000	Hdown1	支持
	リリース回数		-0.161	-0.920	0.350	Hdown2	棄却
	ターゲットユーザーダミー	エンドユーザー	-0.069	-1.020	0.300		
		開発者	0.026	0.440	0.660		
	OSダミー	Linux	-0.011	-0.180	0.850		
		BSD	-0.044	-0.400	0.690		
		Microsoft	0.017	0.250	0.790		
		OS独立	-0.037	-0.500	0.610		
	ソフトウエアの内容ダミー	開発	-0.009	-0.120	0.900		
		ゲーム	0.046	0.490	0.620		
		通信	0.032	0.410	0.680		
		インターネット	-0.005	-0.080	0.930		
		システム	-0.053	-0.770	0.440		
	ライセンス	GPLダミー	0.027	0.450	0.650		
	定数項		-2.655	-6.440	0.000		

注）表中のp値は両側検定の結果だが、仮説についてはz値を用いて片側検定を行った。
$z_{0.1}$=1.28、$z_{0.05}$=1.64、$z_{0.01}$=2.32

図表　推定結果（つづき）

従属変数	説明変数	偏回帰係数	z値	p値	対応する仮説番号	仮説の検定結果
ユーザーコミュニティの規模						
	ダウンロード数	0.620	21.910	0.000	Hcom1	支持
	ユーザー間のコミュニケーション密度：ユ	-0.074	-0.720	0.470	Hcom2	棄却
	定数項	-2.063	-9.950	0.000		
バグ報告メッセージ数						
	ユーザーコミュニティの規模	0.595	5.660	0.000	Hbug1	支持
	バグ報告メッセージの解決率	2.394	4.540	0.000	Hbug2	支持
	定数項	-0.569	-5.270	0.000		
機能追加要求メッセージ数						
	ユーザーコミュニティの規模	0.193	5.540	0.000	Hfeat1	支持
	機能追加要求メッセージの解決率	5.218	6.470	0.000	Hfeat2	支持
	定数項	-0.258	-3.270	0.000		
サポート要求メッセージ数						
	ユーザーコミュニティの規模	0.167	3.170	0.000	Hhelp1	支持
	サポート要求メッセージの解決率	2.448	6.240	0.000	Hhelp2	支持
	定数項	-0.191	-2.040	0.040		
バグ報告メッセージの解決率						
	ユーザーコミュニティの規模	0.195	15.520	0.000	Hres1	支持
	定数項	-0.115	-3.460	0.000		
機能追加要求メッセージの解決率						
	ユーザーコミュニティの規模	0.026	4.610	0.000	Hres2	支持
	定数項	-0.025	-1.680	0.090		
サポート要求メッセージの解決率						
	ユーザーコミュニティの規模	0.125	12.260	0.000	Hres3	支持
	定数項	-0.174	-6.410	0.000		
プロジェクトの認知度：ホームページビュー数						
	ユーザーコミュニティの規模	1.511	28.440	0.000	Hrec*+	支持
	定数項	5.571	38.870	0.000		

4.3　本節の総括

ここでは２節で提示したフレームをもちい、仮説を設定し、Sourceforgeにおけるオープンソース・ソフトウエアの公開データを用いてそれらを検証した。検定の結果を次のフロー図にまとめた。実線が支持された仮説、破線が棄却された仮説である。

これをみると、プロジェクトの認知度がダウンロード数を増加させ、ユーザーコミュニティの規模を拡大させる。そして、ユーザーコミュニティの規模が、プロジェクトの認知率を増加させるという正のフィードバックが存在していることが読みとれる。さらに、ユーザー規模の拡大は、ユーザーからのフィードバック（バグ報告、機能追加要求、ヘルプリクエスト）を増加させ開発を活発化させる。また、ユーザーコミュニティの規模は開発者の数を増加させ、開発を活発化させ、リリース回数を増加させる作用もある。ただし、

H_{ndev1}　：「開発者の数」と「開発者のスキルの公開度」との間には正の相関がある。

については棄却された。これはプロファイル（スキルを含む）を公開している者の割合が低

いことや、その人が作成/修正したコード（ソフトウエア）をみれば、そのスキルがわかるためではないかと考えられる。
　また、
H_{dev2}　：「開発の活発度」と「開発における分担の分散度」との間には正の相関がある。
　が棄却されたのは、分析対象に比較的小規模なプロジェクトが多く、一人でも活発に修正しているプロジェクトが多いことによると考えられる。

　次の仮説が棄却されたのは、ユーザーは、新しいリリースがあったからといって、必ずしもダウンロードするとは限らないことを意味していると考えられる。ユーザーがどのタイミングで新しいリリースに変更するか、個人レベルでの調査が必要である[注22]。
　H_{down2}　：「ソフトウエアのダウンロード回数」と「ソフトウエアのリリース回数」との間には正の相関がある。

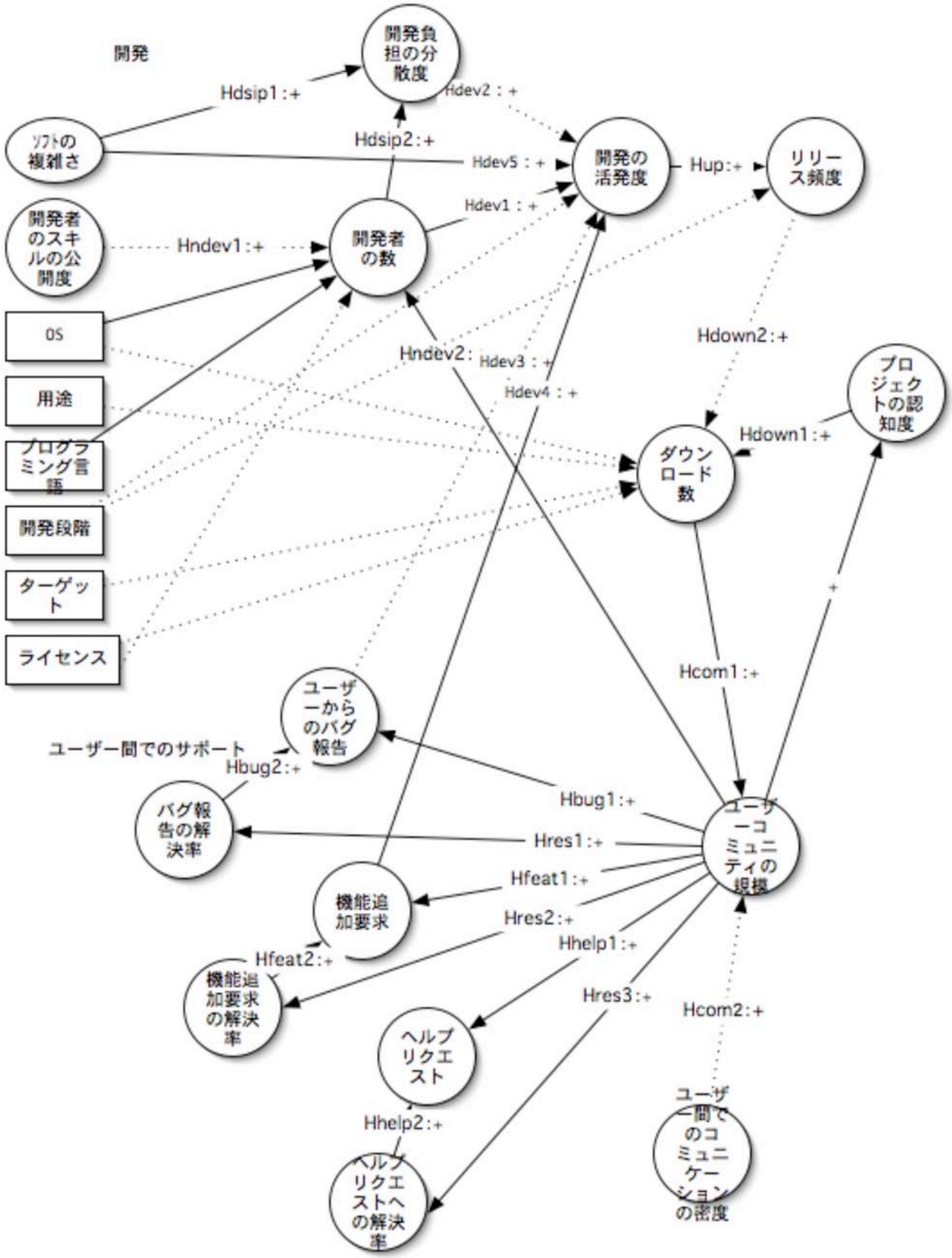
　次の仮説が棄却されたことについて、ユーザーは、「ユーザー間でのコミュニケーション密度」ではなく、メッセージ数そのものを観察しているのかもしれない。
　H_{com2}　：「ユーザー・コミュニティの規模」と「ユーザー間でのコミュニケーション密度」との間には正の相関がある。

　下記の仮説が棄却されたのは、たとえば投稿者の誤り、対応の必要がないようなバグなど、様々なレベルのバグ報告があるためではないかと考えられる。

　H_{dev3}　：「開発の活発度」と「ユーザーからのバグ報告数」との間には正の相関がある。

[注22] Sourceforgeでは誰がダウンロードしたかはデータとして入手できない。

図表　仮説の検定結果


開発
開発負担の分散度
Hdsip1:+
Hdev2:+
ソフトの複雑さ
Hdsip2:+
Hdev5:+
開発の活発度
Hup:+
リリース頻度
開発者のスキルの公開度
Hndev1:+
開発者の数
Hdev1:+
OS
用途
プログラミング言語
開発段階
ターゲット
ライセンス
Hndev2:
Hdev3:+
Hdev4:+
Hdown2:+
プロジェクトの認知度
Hdown1:+
ダウンロード数
Hcom1:+
ユーザーからのバグ報告
ユーザー間でのサポート
Hbug2:+
Hbug1:+
バグ報告の解決率
Hres1:+
ユーザーコミュニティの規模
Hfeat1:+
機能追加要求
Hres2:+
Hhelp1:+
Hfeat2:+
機能追加要求の解決率
Hres3:+
Hcom2:+
ヘルプリクエスト
Hhelp2:+
ヘルプリクエストへの解決率
ユーザー間でのコミュニケーションの密度

5. 本研究の総括

5.1　本研究からの知見

本研究では、注目を集めているオープンソース・ソフトウエア・プロジェクトについて考察および分析を進めた。これまでの研究の限界として、成功したと考えられる特定の事例にのみ注目していること、さらに開発についてのみ注目していることを指摘した。

オープンソース･ソフトウエアが、ユーザーによって開発されることから、開発コミュニティ自体の構成が変化すること、ユーザーがサポートなどをする可能性があるといった特徴を踏まえて、オープンソース・ソフトウエアのパフォーマンス指標として、「開発プロセス」「開発されたソフトウエア」「市場における成果」「コミュニティへのインパクト」「コミュニティの成員へのインパクト」という広い範囲について考慮することが必要であることを指摘した。さらに、これらを規定する要因についても考察し、包括的な分析の枠組みを提案した。

Sourceforgeにおける2101のオープンソース・ソフトウエア・プロジェクトについて公開されているデータを用いることによって、提示したパフォーマンス指標についての分析を行った。オープンソース･ソフトウエアは、多数の開発者によって共同で開発されているというイメージがあるが、今回の分析では開発は少人数で行われているものが多いこと、さらには開発作業自体も特定の者に集中する傾向があることがわかった。

この他、ほとんどの指標について分布には大きな偏りがみられた。つまり、非常にダウンロード数が多い、ページビューが多い、開発者数が多いといったプロジェクトは、きわめて少ないことが示された。クラスタ分析の結果でも2101プロジェクトのうち1699プロジェクトは、ユーザーフィードバック、CVSへの貢献、解決率、ページビュー・ダウンロードともに平均以下であることが明らかとなった。

提示した分析枠組みのうち、利用可能な変数についての仮説を設定し、Sourceforgeからのデータを用いて検定した。この結果、プロジェクトの認知度がダウンロード数を増加させ、ユーザーコミュニティの規模を拡大させる。そして、ユーザーコミュニティの規模が、プロジェクトの認知率を増加させるという正のフィードバックが存在していることが実証された。さらに、ユーザー規模の拡大は、ユーザーからのフィードバック（バグ報告、機能追加要求、ヘルプリクエスト）を増加させ開発を活発化させ、開発者の数を増加させ、開発を活発化させ、リリース回数を増加させる作用もあることが実証された。

5.2　今後の課題

本研究は、これまでの研究と異なり、開発やサポートだけでなく、普及段階も含めた総括的な分析枠組みを提示し、2000プロジェクトについて公開されているデータを用いて定量的に分析したという大きな特徴をもっている。ただし、定量的な分析については、一定期間のデータをプールしてクロスセクションの分析を行った。公開されているデータは時系列での集計も可能である。パネルデータ分析によって、たとえば期間Ｔにおけるバグ方向がＴ+1の開発活動を活発にするといった因果的な分析をより明らかにすることができる。

さらに、コミュニティにおけるメッセージのやりとりについては、メッセージの解決率、密度しか考慮しなかったが、社会ネットワークとして扱うことも可能である。CVSでのファイルへのコミットメント状況を人×ファイル×時間で集計・分析すれば開発の分担の状況をより詳細に把握することができる。これとメッセージのやりとりを考えれば、情報の交換、共有と開発の生産性といった興味深い分析も可能となる。

オープンソース・ソフトウエア・ムーブメントはIT業界だけでなく、企業がつくり消費者が買うという、これまでの産業システム自体を逆転させるポテンシャルをもっている。筆者は企業と消費者とが相互に影響を与えながら長期的に進化していくという「共進化マーケティング」を提示したが［濱岡(1995、2001)］、その一例としてオープンソース･ソフトウエア・プロジェクトは非常に興味深い研究テーマを与えてくれる。ここで示した課題にとりくみながら研究をすすめていきたい。

参考文献

Aoki ,Atsushi, Kaoru Hayashi , Kouichi Kishida , Kumiyo Nakakoji , Yoshiyuki Nishinaka , Brent Reeves , Akio Takashima , and Yasuhiro Yamamoto (2001)"A case study of the evolution of Jun :an object-oriented open-source 3D multimedia library ", Proceedings of the 23rd international conference on Software engineering , pp. 524 - 533

Baym, Nancy K.(1995)'The Emergence of Community in Computer-Mediated-Community', in Jones, Steven G. ed.(1995), p.138-163

Bonaccorsi, Andrea and Cristina Rossi (2003)"Why Open Source software can succeed ", Research Policy , Vol. 32, No. 7, pp.1243-1258

Brooks Jr., Federick P.(2000)The Mythical Man-month: Essays on Software Engineering: Anniversary Edition, Addison-Wesley Publishing Co.(滝沢ら訳『人月の神話　オオカミ人間を撃つ銀の弾はない』アジソン・ウエズレイ・ジャパン、1996年)

Clark, Kim B. and Takahiro Fujimoto(1991)Product Development Performance, HBS Press(『製品開発力』ダイヤモンド社)

DiBona, Chiris, Sam Ockman, and Mark Stone ed.(1999)Open Sources: Voices of the Open Source Revolution, O'Reilly: Cambridge(倉骨彰訳『オープンソースソフトウエア』オライリー、1999年)

Friedkin(1998),A Structual Theory of Social Influence, Cambridge Univ. Press:

古川良治(1993)「電子コミュニティの虚と実」, (川上善郎、川浦康至、池田謙一、古川良治『電子ネットワーキングの社会心理　コンピュータ・コミュニケーションへのパスポート』誠信書房]), p.106-137

Ghosh,Rishab and Vipul Ved Prakash (2001)"The Orbiten Free Software Survey", *First Monday*, volume 5, number 7(http://firstmonday.org/issues/issue5_7/ghosh/index.html)

Granovetter,Mark S. (1973)'The Strength of Weak Ties', Ameriacn Journal of Sociology, Vol.78, pp.1360-1380

Hagel III, John and Arthur Armstrong(1997)NET GAIN, HBS Press(マッキンゼー・ジャパン・バーチャルコミュニティ・チーム訳、南場智子監修『ネットで儲けろ』日経BP, 1997年)

濱岡豊(1994a)「レビュー　消費者間相互依存性／相互作用」『マーケティング・サイエンス』,Vol.2,No.1-2,pp.60-85

濱岡豊(1994b)「消費者の意思決定とクチコミの影響のメカニズム」『東京大学大学院　博士学位(学術)論文』

濱岡豊(1995)「共進化の観点からのマーケティング戦略論の再構築」『第1回　日本マーケティング協会助成研究報告書』

濱岡豊(1996)「コミュニケーションは社会システムを救えるか　　社会システムのシミュレーション分析を通じて」［竹内啓監修『50年後の社会システムを考える』統計研究会］

濱岡豊(1999)「アクティブ・コンシューマ・モデルについての試論」マーケティング・サイエンスワークショップレジメ

濱岡豊(2001)「アクティブ・コンシューマ　創造しコミュニケートする能動的な消費者モデルの開発に向けて」未来市場開拓プロジェクト・ワーキングペーパー（東京大学経済学部)

濱岡豊(2001)「共進化マーケティング 消費者が開発する時代におけるマーケティング」『学術振興財団　未来市場開拓プロジェクト・ディスカッションペーパー』東京大学経済学部

濱岡豊(2002)「アクティブ・コンシューマーを理解する」『一橋ビジネスレビュー』冬号

Hauben, Michael and Ronda Hauben(1998)Netizene: On the History and Impact of Usenet and the Internet, (井上、小林訳『ネティズン』、中央公論社、1997年)

Hertel, Guido , Sven Niedner, and Stefanie Herrmann(2003)"Motivation of software

developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel", Research Policy , Vol. 32, No. 7, pp.1159-1177
池田謙一、柴内康文、古森鎮哉(1997)「グループメディアとしてのパソコン通信　電縁形成のダイナミクス」, (Niftyネットワークコミュニティ研究会『電縁交響主義』NTT出版), , p.100-123,
池田謙一編(1997)『ネットワーキング・コミュニティ』, 東京大学出版会
Jones, Steven G. ed.(1995)CyberSociety: Computer-mediated communication and community, Sage
金子郁容監修、宮垣元、佐々木裕一(1998)『シェアウエア　もうひとつの経済システム』, NTT出版
北山聡(1997)「フォーラムの生態」, (Niftyネットワークコミュニティ研究会『電縁交響主義』NTT出版), , p.34-67,
国領二郎、田村隆史、森田正隆(1997)「共感が生み出す価値」, (Niftyネットワークコミュニティ研究会『電縁交響主義』NTT出版), , p.244-269,
Kollock, P. (1999). The Economies of Online Cooperation: Gifts and Public Goods in Cyberspace. Chapter 7, Communities in Cyberspace, M. A. Smith and P. Kollock eds. London,Routledge.
川上善郎、川浦康至、池田謙一、古川良治(1993)『電子ネットワーキングの社会心理　コンピュータ・コミュニケーションへのパスポート』, 誠信書房
Lakhani, Karim R. and Eric von Hippel (2003)"How open source software works: "free" user-to-user assistance", Research Policy , Vol. 32, No. 7, pp. 923-943
Lerner, Josh and Jean Tirole(2000)'The Simple Economics of Open Source', Discussion Paper HBS,
Levy, Steven(1998)Hackers: Heroes of the Computer Revolution, (古橋、松田訳『ハッカーズ』工学社、1987年)
Markus, Lynne M., Brook Manville, and Carole E. Agres(2000)'What Makes a Virtual Organization Work?', Sloan Management Review, , Fall, pp.13-26
Mockus, A., FIELDING, R.T., and Hersbleb, J.(2000)"A Case Study of Open Software Development: The Apache Server ',", Proc. 22nd. International Conference on Software Engineering, Limerick, IR, pp. 263-272
McLaughlin, Margaret L., Kerry K. Osborne, and Christine B. Smith(1995)'Standards of Conduct on Usenet', in Jones, Steven G. ed.(1995), p.90-111
中村雄二郎、金子郁容(1999)『弱さ』岩波書店
Pressman, Roger S.(1997)Software Engineering: A Practitioner's Approach 4th ed., McGraw-Hill: NY
Raymond, Eric S. (1998)The Cathedral and the Bazaar(http://www.tuxedo.org/esr/writings/cathedral-bazaar/) 山形浩生訳「伽藍とバザール」http://www.post1.com/home/hiyori13/freeware/cathedral.html)
Robbins, Stephen P.(2001)Organizational Behavior 9th ed., Prentice Hall: NJ
Shah, Sonali(2000)Sources and Patterns of Innovaiton in a Consumer Prducts Field: Innovations in Sporting Equipment, Sloan School of Management Working Paper #4105
佐々木裕一、北山聡(2000)『Linuxはいかにしてビジネスになったか』, NTT出版,
Scott, John(1991)*Social Network Analysis: A Handbook*, Sage
Shah, Sonali(2000)Sources and Patterns of Innovaiton in a Consumer Prducts Field: Innovations in Sporting Equipment, *Sloan School of Management Working Paper* #4105
嶋口、竹内、片平、石井編(1999)『製品開発革新　マーケティング革新の時代2』, 有斐閣
Sproull, Lee and Sara Kiesler(1998)Connections: New Ways of Working in the Network Organization, MIT Press(加藤丈夫訳『コネクションズ　情報ネットワークで変わる社会』アスキー出版、1993年)
Torvalds, Linus and David Diamond(2001)Just for Fun: The Story of an Accidental

Revoutionary, HarperCollins(風見潤訳『それがぼくには楽しかったから』小学館、2001年), , ,
Tuomi, Ilkka (2000)"A Case Study of Open Software Development: The Apache Server ',", Working paper(http://opensource.mit.edu)
von Hippel,Eric A.(1988)The Source of Innovaton, Oxford Univ. Press (榊原清則訳『イノベーションの源泉』ダイヤモンド社、1991年)
von Hippel, Eric (1994)'Sticky Information" and the Locus of Problem Solving: Implications for Innovation', *Management Science*, Vol.40, No.4(April), pp.429-439
von Hippel, Eric (1998)'Economics of Product Development by Users: The Impact of "Sticky" Local Information', *Research Policy*, Vol.44, No.5(May), pp.629-644
von Hippel,Eric A.(1999)'Toolkits for User Innovation', Working Paper, MIT
von Krogh, Georg , Sebastian Spaeth and Karim R. Lakhani (2003)"Community, joining, and specialization in open source software innovation: a case study ", Research Policy , Vol. 32, No. 7, pp.1217-1241
Wasserman, Stanley and Katherine Faust(1994)Social Network Analysis: Methods and Applications, Cambridge Univ. Press
安田雪(1997)『ネットワーク分析　何が行為を決定するか?』, 新曜社
安田雪、木村泰之(1997)「デジタル・ソリディラリティ」, (Niftyネットワークコミュニティ研究会『電縁交響主義』NTT出版), p.70-97