# 修 士 論 文

題　目　　オープンソースソフトウェアにおける
開発者スキルのばらつきとプロジェクトの
活性の関連分析

主任指導教員　　水野　修　准教授

京都工芸繊維大学大学院 工芸科学研究科

情報工学専攻

学生番号　　12622011

氏　　名　　大西　哲朗

平成26年2月10日提出

様式 2 号

学位論文の要旨（和文）

平成 26 年 2 月 10 日

京都工芸繊維大学大学院
工芸科学研究科長　殿

工芸科学研究科　　情報工学専攻
平成 24 年入学
学生番号　　12622011
氏　　名　　大西　哲朗　㊞

（主任指導教員　　水野　修　㊞　）

本学学位規則第 4 条に基づき、下記のとおり学位論文内容の要旨を提出いたします。

1.　論文題目

オープンソースソフトウェアにおける開発者スキルのばらつきとプロジェクトの活性の関連分析

2.　論文内容の要旨（400 字程度）

オープンソースソフトウェアのプロジェクトには，開発活動が活発なものとそうでないものがある．我々は，これを「プロジェクトの活性」と呼び，プロジェクトに携わる開発者達の技術的スキルのばらつきが，プロジェクトの活性と関連しているのかどうかを分析する．分析するデータは，GitHub リポジトリからダウンロードした約 8000 プロジェクトを使用する．まず，我々はプロジェクトの活性を，プロジェクトの最初のコミットから最後のコミットまでの期間でのコミット頻度を利用して定義した．次に，開発者のスキルをプロジェクト内で各開発者が更新したことのあるファイル数を利用して定義した．この開発者のスキルの分散を用いて，各プロジェクトにおける開発者のスキルのばらつきを定義した．最後に，定義したプロジェクトの活性と開発者スキルの相関を全プロジェクトに対して分析した．その結果，両対数をとることで，相関係数が 0.62 を示すことを示した．すなわち，開発者のスキルが均一化されていないプロジェクトが高い活性を示すことを確認した．

# Analyzing Relationship between Variance of Developers' Skill and Project's Activeness

2014 12622011 *Tetsuro ONISHI*

## Abstract

In the open source software projects, it is observed that some projects are highly motivated and active, but some are not active. We assumed that such difference of activeness is derived from the variance of developers' skills in each project. The research question is "Are the projects with high variance of developers' skill more active than that with low variance?" This thesis aims to analyse this research question. The data used in this study is obtained from the popular projects in GitHub. We collected values of metrics for the project activeness and the developers' skill from about 8000 projects. We defined the project activeness as the frequency of commits in a project, and the skill of a developer as the number of files that the developer has modified so far. The result of analysis showed that there is a positive correlation between the variance of developers' skill and the project activeness.

# 目　次

# 1. 緒言

ソフトウェアプロジェクトの特徴を大量のデータから抽出する手法として，近年リポジトリマイニングが注目されている．リポジトリとは版管理システムを用いてソフトウェアを開発する際に，過去から現在までの履歴を残しておく記録庫であり，ここにはソースコードだけでなく，開発者がコードを登録する時に残したコメントなども登録されており，時系列を追ってコードの変遷を追跡することができる．この記録を分析し，ソフトウェアの開発における有益な知見を発見することがリポジトリマイニングの目的である [1].

リポジトリマイニングで行われる研究の多くは，リポジトリに含まれるソフトウェアのメタデータから得られる情報に着目したものである [2]．それとは別に，ソースコードの開発者に着目した研究も存在する [3]．例として，そのソースコードに関わった開発者が多いほどそのコードは欠陥を含みやすくなるか否か [4]，といったものが挙げられる．また，プロジェクトに関わる開発者たちが構成する構造に着目するものが存在する．例えば，単一のオープンソースソフトウェアプロジェクトを分析したとき，その内部にはコミュニケーション構造などの形でソーシャルネットワーク構造が存在することが判明している [5].

リポジトリマイニングを行う対象としては，オープンソースソフトウェアを利用する．オープンソースソフトウェアはリポジトリのアクセスに制限がなく，また実験環境を再現しやすいため，研究対象として扱いやすい．オープンソースソフトウェアは企業による開発活動とは大きく異るが，オープンソースソフトウェアは多くの場面で利用され，高品質なものも少なくないため，これらの活動自体を調べる研究も多くある．なぜオープンソースソフトウェアがうまく運営されているのか，は多くのものが関心を寄せているが，未だ決定的な結論は出ていない．

数多く存在するオープンソースソフトウェアのプロジェクトには，開発活動が活発なものとそうでないものがある．我々は，この開発活動の活発さを「プロジェクトの活性」と呼ぶ．そして，このプロジェクトの活性に影響を与えている要因の分析を行う．その中で，仮説のひとつとして提案したのが，プロジェクト中の開発者の技術スキルとプロジェクトの活性の関連である．

俗説として，普通のプログラマとトップレベルのプログラマでは生産性が 10 倍近

く違う，と言われている．プログラマの能力をエビデンスを持って測る方法はまだ確立されていない．そもそもプログラマの能力を画一的な指標で測ることは難しい．「人月の神話」では，チームによる役割分担のプログラミングによって生産性を上げる方法が提案されている [6].

我々は,「開発者のスキルが均一ではないプロジェクトのほうが，プロジェクトの活性が高いのではないか」という仮説を立てた．この仮説を分析するために，本研究では，オープンソースソフトウェアのリポジトリ情報をデータベース化し，そこから開発者スキルのばらつきとプロジェクトの活性を定義する．そしてそのふたつの指標の関連性を分析した．

本論文の以降の構成を述べる．2 章では，本研究にあたって設定した問題を挙げる．3 章では，本研究の背景について述べる．4 章では利用したリポジトリ群の概要について述べると共に，開発者スキルとプロジェクト活性の定義を行い，分析の手順を述べる．また，5 章では，分析の結果を述べる．最後に，6 章ではまとめと今後の課題を述べる．

## 2. 研究設問

本研究で示したい仮説は,「開発者のスキルにばらつきがあるほうが, プロジェクトが活性化する.」である. これは, 開発組織の人材に多様性を持たせることが，ソフトウェア開発のような創造的活動には良い影響を与えるのでは，との考えから導かれたものである. 現実に, オープンソースプロジェクトであっても開発組織の官僚化が進んだ結果として, 開発が停滞する事例が存在する. そのため, 開発が活性化し続ける条件を導出することは, ソフトウェア開発の健全な遂行に必要だと考える.

以上のことから，本研究では以下の研究設問を示すことを目的とする.

RQ 開発者のスキルのばらつき (分散) が高いプロジェクトほど，プロジェクトは活性化しているか？

この設問は，一見，一般的なソフトウェア開発プロジェクトに求められる要件と逆の主張をしている．すなわち，ソフトウェア開発では，開発者のスキルがばらついているのは，コスト見積もりの観点や品質保証の観点からも好ましくないと考えられることが多い．しかしながら，オープンソースソフトウェアにおける観察を行うことで，このような現象が発生しうるのではないかと考えるに至り，本研究の設問とした.

# 3. 研究背景

## 3.1 オープンソースソフトウェア

オープンソースソフトウェアとは，無料で利用でき，ソースコードの取得・改変・再配布が許可されているソフトウェアのことである．改変・再配布に関する制限はライセンスにより規定されている．改変後の再配布が義務であるライセンスもあれば，とくに制限しないものまで，さまざま．開発者は主にインターネットを通じて自由に開発に参加・離脱できる．Fogel が著書で述べたように，ほとんどのフリーソフトウェアプロジェクトは失敗する，と言われているが [7]，OS や Web サーバや開発ツールやブラウザなど，多くのソフトウェアが成功し，オープンソースソフトウェアとして提供されている．

基本的にはボランティアメンバーによって運営されているが，Canonical や Google など，企業が主導してオープンソースソフトウェアの運営を行うこともある．対義語として，企業によって売買されるソースコード非公開のソフトウェアをプロプライエタリソフトウェアと呼ぶ．本研究ではこれらを研究対象にしない．

本研究では，これらオープンソースソフトウェアのリポジトリをダウンロードし，それらの変更履歴から分析を行う．

## 3.2 ソフトウェアメトリクス

ソフトウェアメトリクスは，1970 年代後半頃からソフトウェアの特性を測るための指標として提唱され始めた概念である [8,9]．具体的には，ソースコードの規模，複雑さ，保守性などの「属性」の情報を定量的に示すことができる指標である．具体的な例としては，プログラムのステップ数やバグの数がこれに該当する．ソフトウェアメトリクスの多くはソースコードやプロジェクトの開発履歴から測定できる．そのため，バージョン管理システムを導入して開発されたオープンソースソフトウェアはメトリクスを用いた適用実験が行いやすいという利点がある．ソフトウェアメトリクスの用途としては，ソフトウェアの品質の測定や，誤り傾向の強いモジュールを判別するなど様々な方法が研究されている．

## 3.3 リポジトリ群

提案したソフトウェアメトリクスやモデル式を評価し，一般化するためにはできるだけ多くのプロジェクトに対して提案手法の適用実験を行うのが望ましい．実験対象のプロジェクトの入手先としては，SourceForge や GitHub など，オープンソースソフトウェアのリポジトリが集合的に公開されているポータル（Forge サイトと呼ぶ）が多数存在しており，それらの Forge サイトを利用するのが一般的である．

## 3.4 GitHub リポジトリ

Git は分散型のバージョン管理システムの一つである．コマンド一つでサーバ上から対象のプロジェクトのリポジトリをローカルにコピーすることができ，過去のソースコードを確認するのにサーバを参照する必要が無いという利点がある．また，ファイルの移動や名前変更に強く，それらの変更が行われてもある程度はファイルの更新履歴を遡って追跡することができる．これらの利便性から本研究では Git でバージョン管理されたオープンソースソフトウェアのプロジェクトを分析対象とする．

GitHub は Git のリポジトリを中心に扱っている有名な Forge サイトの一つであり，個人的なプロジェクトから有名なプロジェクトまで数多くのプロジェクトを保有している利用者の多い開発コミュニティである．

## 3.5 ソーシャルネットワーク

人が属している社会での，人同士のつながりをソーシャルネットワークと呼ぶ．家庭内では家族同士のつながりが見いだせ，クラス内では同級生同士のつながりを見出すことができる．従来それらは客観的な観測が難しい物だったが，近年のインターネットにおけるソーシャルネットワーキングサービスによって可視化されるようになった [16].

本研究ではオープンソースソフトウェアリポジトリにおけるソーシャルネットワークに注目する．同じソースコードファイルにコミットした開発者同士は，そのファイルを通じてつながっているとする．そしてそこから生まれるつながりを分析する．

## 3.6 関連研究

単一のリポジトリから何らかのネットワークを構築して調査を行う研究には次のようなものがある．Menely らは「充分な人数がいれば全てのバグはすぐさま発見され修正される」というリーナスの法則が正しいのかどうか調査する目的で実証研究を行った [10]．その後，開発者の活動性のメトリクスを提案し，ファイルに関わる人数による欠陥の出現率と除去される割合を調べ,「多すぎる人数が製作物を駄目にする」という説と「充分な人数がいれば全てのバグはすぐさま発見され修正される」という説に対して調査を行った [4]．他に，一ヶ月以内に同じファイルに更新した開発者を共同開発者と定義し，各開発者にアンケートを取って，実際に開発者が共同作業をしていると感じている開発者とリポジトリを解析して共同開発者と判断された開発者はどの程度共通しているかを調査も行っている．これによりソフトウェア工学にソーシャルネットワーク解析メトリクスを用いることの有効性を評価した [11]．Bird らはメーリングリストの内容やプロジェクトのリポジトリから分析を行い，オープンソースプロジェクトが無秩序なバザーのような構造をしているのかを調査した．社会参加者と開発振る舞いの関係を特徴づけて，大規模なオープンソースプロジェクトに存在する社会構造を検査した結果，存在する構造は典型的なバザー構造ではなく有機的にチームへ組織していく傾向にある事を確かめた [5]．その後，大規模で複雑且つ成功したオープンソースプロジェクトでは開発チーム内に小さな社会が自然発生していると考え，オープンソースプロジェクトが自己組織化していく過程を調査した [12]．また，その後，ソフトウェアのモジュール構造とそれによって生み出される開発者間のソーシャルネットワーク構造を，ソフトウェアの不具合予測に高い精度で用いることができる事を示している [13]．

ソーシャルネットワークに限った研究では，Mark S. Granovetter による「弱い紐帯の強み (The strength of weak ties)」が有名である．この研究では，労働者が転職のきっかけを得たのは親しい人からの情報ではなく，むしろよく知らない人からによるものであることを示す [14]．また，世界的なソーシャルネットワーキングサービスである Facebook が，自社のソーシャルネットワークと利用者について調査したところ，多様な情報は弱いつながりから流れてくることがわかっている [15]．また，Albert らは「新ネットワーク思考」にてネットワークの成長とノードの優先的選択による複

雑ネットワークのモデル化を行い，その性質をさまざまな社会現象に当てはめて，その頑強性と脆弱性について考察を行っている [16].

# 4. 開発者スキルとプロジェクトの活性分析

## 4.1 利用したリポジトリ

本研究では開発者のスキルとプロジェクトの活性分析という目的からも，収集するプロジェクトのリポジトリは多くの人が関わっているのが望ましく，有名なプロジェクトである方が良いと判断した．また，分析対象であるリポジトリはファイルの移動や名前変更に強い Git で管理されたものが良いと考えた．従って，本研究ではリポジトリの大部分を Git の有名な Forge サイトである GitHub から収集したものを用いている [17].

## 4.2 収集手順

Google BigQuery は Google の提供する企業向けのビッグデータ解析サービスである．サンプルのデータセットとして GitHub のアーカイブが含まれており，それに対するクエリを試すことができる．本研究では Google Big Query によって集めた情報を元に，GitHub から実際のリポジトリをダウンロードしたものを用いている．

本研究では，GitHub 上でウォッチャーが多いプロジェクトほど多くの人が注目している，つまり，有名で多くの人が関わっていると考え，次のような Query を実行した．

```
SELECT repository_name, watchers, repository_language, repository_description,
repository_url FROM (SELECT repository_name, count(repository_name) as watchers,
repository_language, repository_description, repository_url
FROM [githubarchive:github.timeline] WHERE type="WatchEvent"
GROUP BY repository_name, repository_language, repository_description, repository_url
ORDER BY watchers DESC) as watched WHERE watchers >=100
```

これにより，ウォッチャーが 100 人以上のプロジェクトの，リポジトリ名，ウォッチャーの数，主な開発言語，説明，リポジトリの URL が分かる．その上で，`git clone` コマンドにより，各リポジトリの実体を収集したものである．

なお，今回の分析のために収集したプロジェクトの数は 8140 であった．

## 4.3 メトリクスの定義

本節では，測定に用いるメトリクスを定義する．

### 4.3.1 開発者スキルの定義

一般的に開発者のスキルを測定するのは容易ではない．技術的なスキルを正確に把握するための指標は未だ研究途上である．

本研究ではその開発者がどれだけ多くの貢献をしているか，という点を重視する．オープンソース開発においては，スキルの高い開発者はそれだけ多くのコミットを実施することが多い．また，多くの機能に対するコミットを行うため，多くの場合では大量のファイルに対して何らかの手を入れる可能性が高い．また，高いスキルを持つ開発者ほど，多くの開発者と共同してファイルの修正を行っている可能性が高い．

そのため，本研究では「開発者のスキル」を，

- その開発者が修正するために触れたことのあるユニークなファイルの数．
- その開発者と同じファイルを変更したことがある開発者数 (共著者数)．

として定義する．以下，この定義を $S_1, S_2$ とそれぞれ呼ぶ．

$S_1$ ファイルベースのスキル定義

$S_2$ 共著関係ベースのスキル定義

### 4.3.2 活性の定義

オープンソースの成功を一概に定義づけるのは難しい．利用者の数やフォークしたプロジェクトなどを考えれば，多くの尺度が存在し，それらと単純に評価することはできない．

本研究は開発者の行動についての研究であるため，オープンソースの活性については，開発活動のみに限定して測定する．具体的には，リポジトリの最初のコミットが行われた日から最後のコミットが行われた日までの日数を計算し，その間に行われたコミット数を数える．そこからコミット数の日平均を算出することで，コミット頻度がわかる．

このコミット頻度が高いほど，開発活動が活発であり，ソフトウェアの活性が高い，と定義する．

## 4.4 収集したメトリクス

収集したメトリクスを以下にまとめる．なお，以下のメトリクスは全て 1 プロジェクトごとに収集されている．

- num_authors: プロジェクトに関与した開発者の数
- auth_deg_min: 開発者間の共著関係の次数 $S_2$ の最小値
- auth_deg_max: 開発者間の共著関係の次数 $S_2$ の最大値
- auth_deg_mean: 開発者間の共著関係の次数 $S_2$ の平均値
- auth_deg_median: 開発者間の共著関係の次数 $S_2$ の中央値
- auth_deg_mode: 開発者間の共著関係の次数 $S_2$ の最頻値
- auth_deg_variance: 開発者間の共著関係 $S_2$ の次数の分散
- file_deg_min: 1 開発者が触れたファイル数 $S_1$ の最小値
- file_deg_max: 1 開発者が触れたファイル数 $S_1$ の最大値
- file_deg_mean: 1 開発者が触れたファイル数 $S_1$ の平均値
- file_deg_median: 1 開発者が触れたファイル数 $S_1$ の中央値
- file_deg_mode: 1 開発者が触れたファイル数 $S_1$ の最頻値
- file_deg_variance: 1 開発者が触れたファイル数 $S_1$ の分散
- commit: プロジェクトにおける総コミット数
- duration: プロジェクトが実施されている期間
- com_per_dur: 単位期間当たりのコミット数 (コミット頻度)

## 4.5 収集したデータの概要

収集したデータの概要は以下の通りである．

- プロジェクト数: 8,140
- プロジェクトの平均開発者数: 25.56
- プロジェクトの最小開発者数: 1

- プロジェクトの最大開発者数: 9,175
- プロジェクトの平均コミット数: 1,256
- プロジェクトの最小コミット数: 1
- プロジェクトの最大コミット数: 589,406
- プロジェクトの平均実施日数: 648.7 日
- プロジェクトの最小実施日数: 0.0 日
- プロジェクトの最大実施日数: 15858.8 日

# 5. 分析結果

## 5.1 開発者スキル(ファイル数)分散とプロジェクト活性の相関

ここでは4.3.1節で定義した開発者スキル$S_1$に基づいた分析を行う．

開発者スキルとプロジェクト活性の相関を調べるために，統計計算言語Rを用いた分析を行った．分析は統計分析言語Rを用いた．

まず，8140プロジェクト全てに対して，各プロジェクトのコミット頻度と各開発者が触ったファイル数の分散の相関を見るために，以下の分析を実施した．

```
> d <- read.csv("dataFrom8000.v3.csv")
> d$com_per_dur <- d$commit / d$duration
> d$duration_day <- d$duration / 86400
> d$com_per_dur <- d$commit / d$duration_day
> d$file_dig_stddev <- sqrt(d$file_dig_variance)
> g <- ggplot(d,aes(com_per_dur,file_dig_stddev))
> g+geom_point(aes(colour=num_authors,size=num_authors,
    alpha=num_authors))+scale_y_log10()+scale_x_log10()
    +xlab("Commit/Duration (count/day) (log)")
    +ylab("Stddev of file-based degrees of developers (log)")
```

図5.1に，この結果として得られたグラフを示す．横軸がコミットの頻度を表し，縦軸が開発者が触ったファイル数の分散を示す．また，グラフのプロットの直径が1プロジェクトに存在する開発者の数を示しており，直径が大きいほど多くの開発者を含んでいることを示している．なお，このグラフは両対数となっている．

図5.1のグラフからは，大まかにではあるが，相関関係が見てとれる．しかし，開発者のスキルの分散が0のプロジェクトが存在したため，相関係数は算出できない．そこで，ある程度の規模のプロジェクトに限定した分析を実施した．分析の手順は以下の通りである．ここでは開発者が100人より大きいプロジェクトに限定して上と同じ分析を実施している．また，この条件下では開発者のスキル分散が0のものが無くなるため，相関係数も算出できる．

```
> d100 <- subset(d,num_authors > 100)
> cor(log10(d100$com_per_dur),log10(d100$file_dig_stddev))

> g100 <- ggplot(d100,aes(com_per_dur,file_dig_stddev))
```

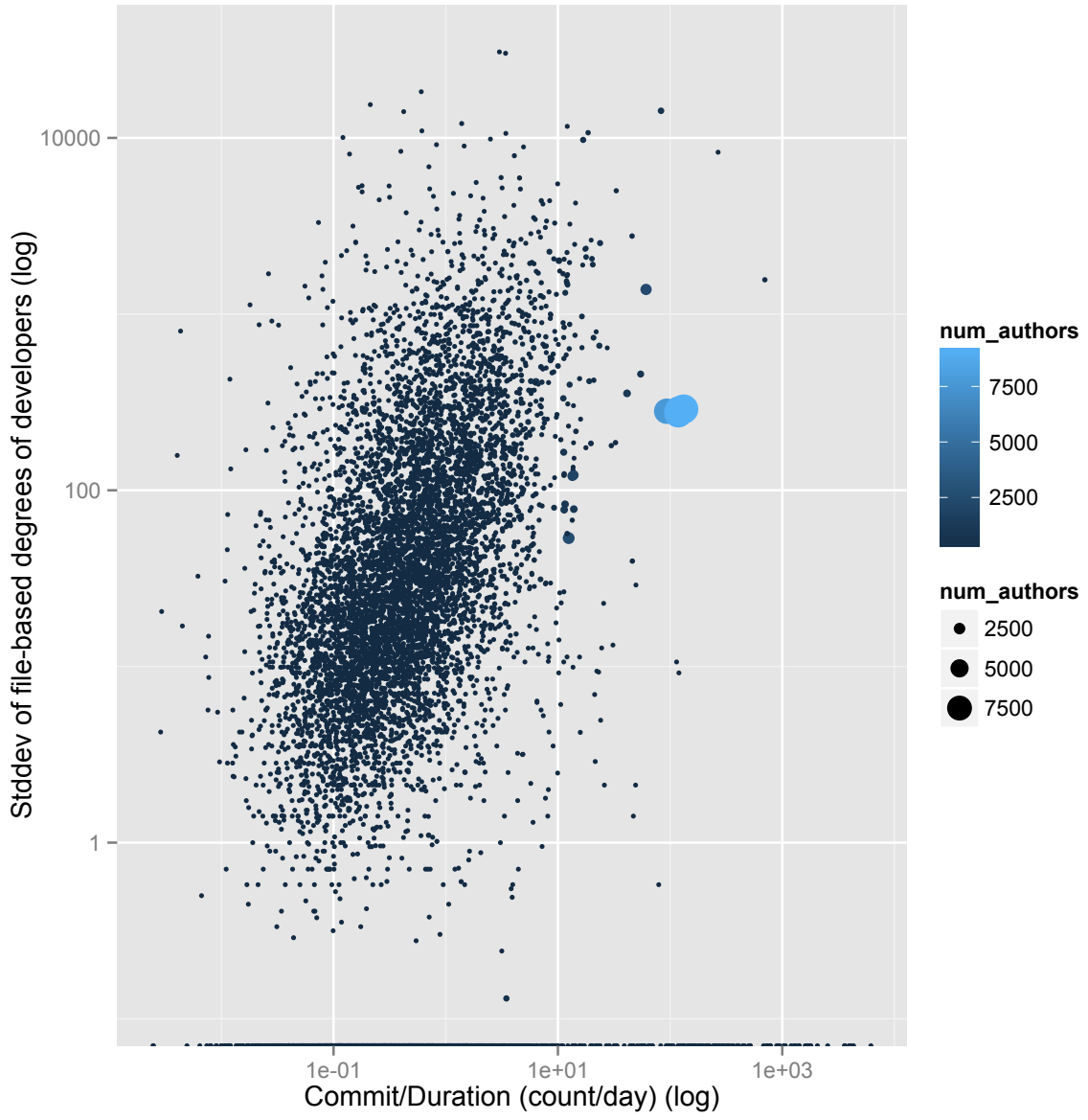


図 5.1: 全プロジェクトに対する開発者スキル ($S_1$) とプロジェクト活性の相関

```
> g100+geom_point(aes(colour=num_authors_log10,size=num_authors_log10))
    +scale_y_log10()+scale_x_log10()
    +xlab("Commit/Duration (count/day) (log)")
    +ylab("Stddev of file-based degrees of developers (log)")
```

図 5.2 にこの結果として得られたグラフを示す．この両対数グラフは正の相関を示しており，相関係数は 0.62 となった．

この結果から，2 章で設定した研究設問,「開発者のスキルのばらつき (分散) が高いプロジェクトほど，プロジェクトは活性化しているか?」に対する答えは「その傾向が見られる」とすることができる．

## 5.2 開発者スキル (共著者数) 分散とプロジェクト活性の相関

次に，4.3.1 節で定義した開発者スキル $S_2$ に基づく分析として，各プロジェクトのコミット頻度と各開発者の共著者数の分散の相関を見るために，R を用いて以下の分析を実施した．

```
> dd1 <- subset(d,auth_deg_variance > 0)
> gg1 <- ggplot(dd1,aes(com_per_day,auth_deg_variance))
> gg1+geom_point(aes(colour=num_authors,size=num_authors))
    +scale_y_log10()+scale_x_log10()+xlab("Commit/Duration (count/day) (log)")
    +ylab("Variance of co-authorship of developers (log)")
> cor(log10(dd1$com_per_day),log10(dd1$auth_deg_variance))
```

図 5.3 に，この結果として得られたグラフを示す．横軸がコミットの頻度を表し，縦軸が開発者の共著者数の分散を示す．また前節と同様，グラフのプロットの直径が 1 プロジェクトに存在する開発者の数を示しており，直径が大きいほど多くの開発者を含んでいることを示している．なお，このグラフは両対数となっている．

なお，ここでは分散の値が 0 となるものを除いて相関係数をとっており，相関係数は 0.49 となった．この結果から，共著者数とコミット頻度は正の相関を示していることが分かる．この結果からも，2 章で設定した研究設問に対し,「その傾向が見られる」とすることができる．

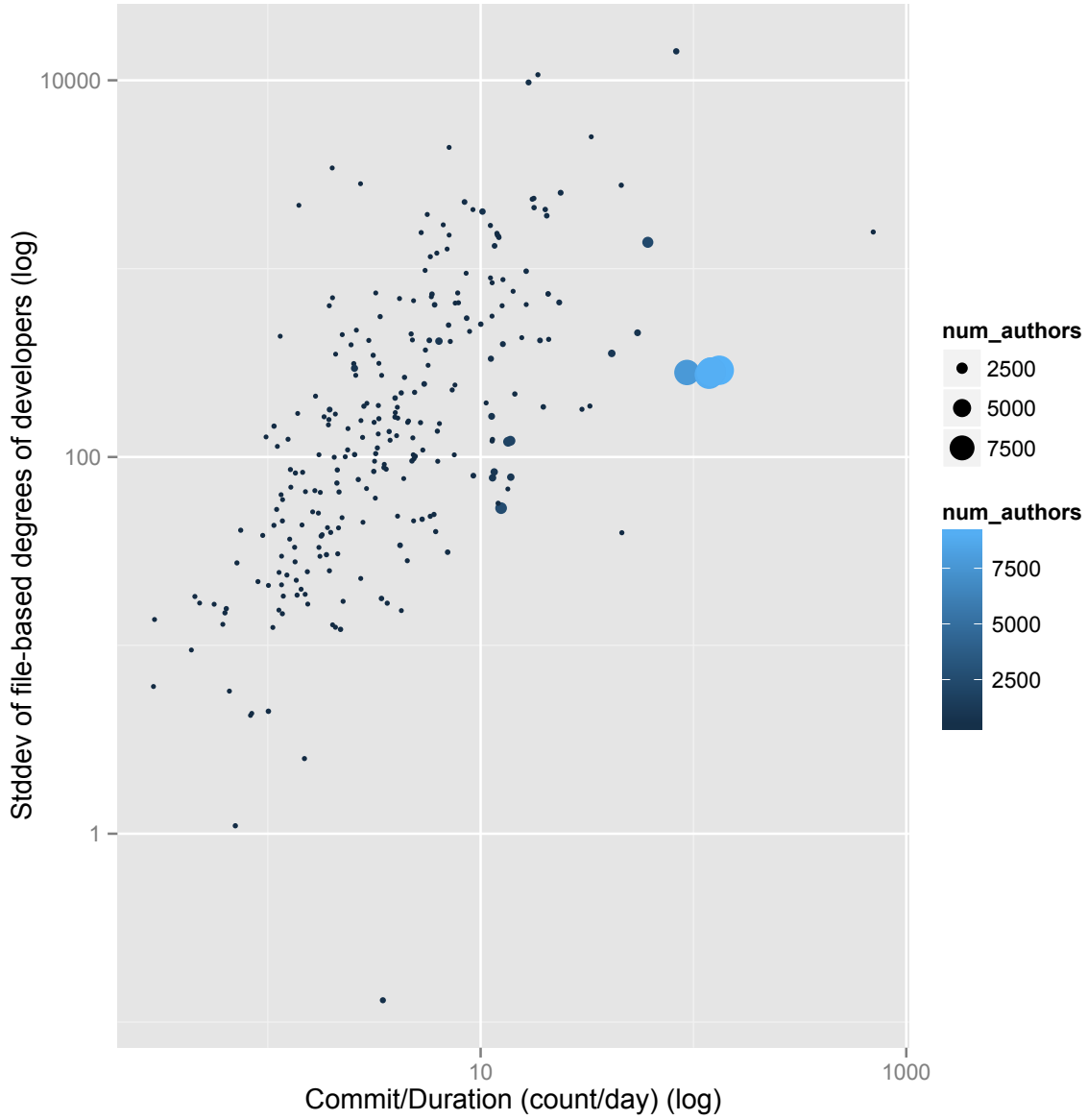


図 5.2: 開発者 100 人以上のプロジェクトに対する開発者スキル ($S_1$) とプロジェクト活性の相関

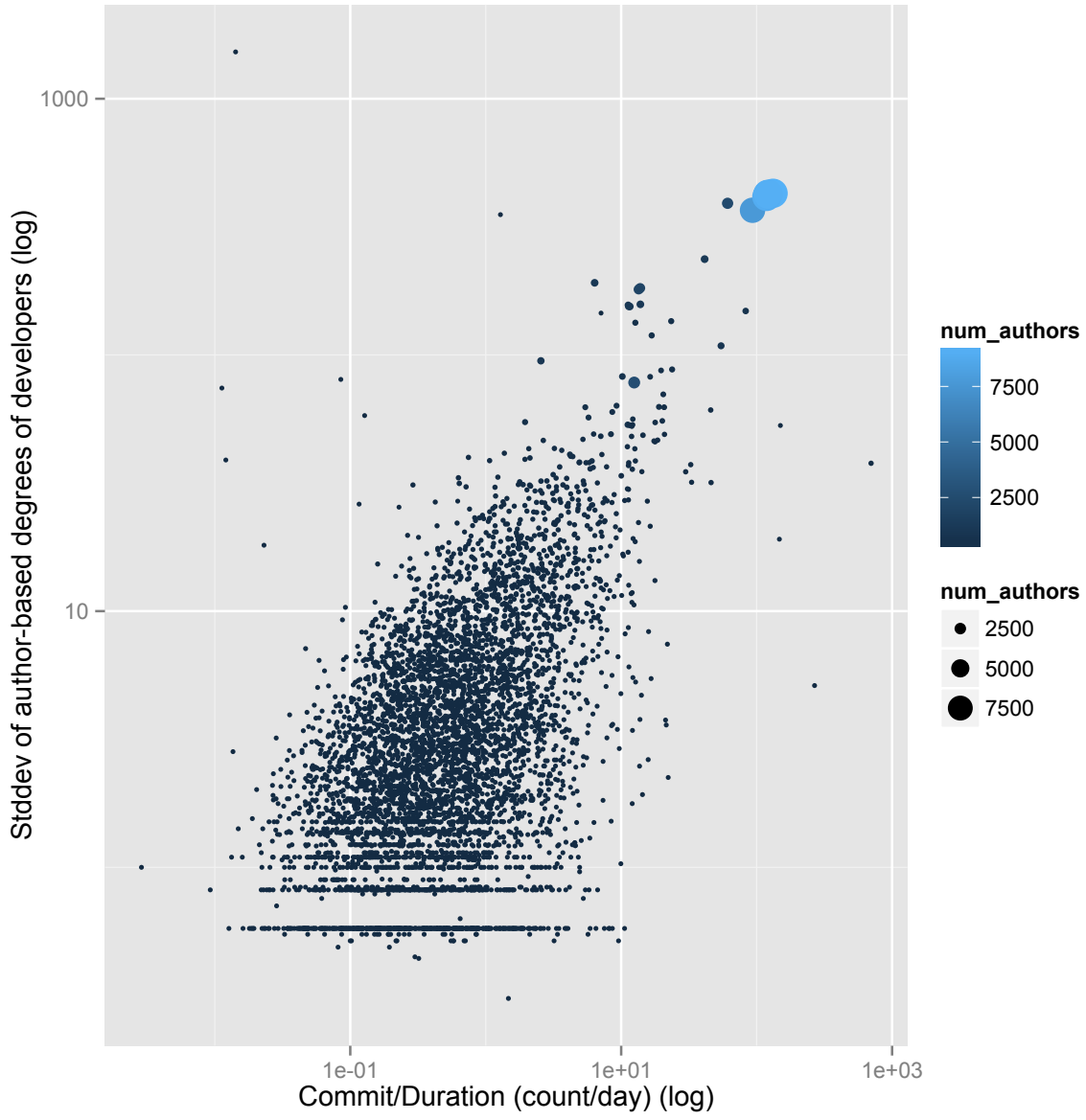


図 5.3: 全プロジェクトに対する開発者スキル ($S_2$) とプロジェクト活性の相関

## 5.3 結果に対する考察

半角カンマ,の後にはスペース

前節の分析の結果から,「スキルのばらつきがあった方がプロジェクトが活性化する」ことが示される. ここではこの結果が示すことについて考察する.

通常, チームでソフトウェア開発を行おうとするのであれば, 能力の高い人のみを集めることが妥当なように思われる, また, 別の考え方として, 開発の見積もりが行い易いように, より能力が平均的な人材で揃えることも考えられる. この場合, 管理されたソフトウェアプロセスであれば, 開発が進めやすく, 進行も安定しやすいように思える.

こういう疑問系はやめよう. エッセイではない.

しかし, オープンソースソフトウェアは, プロプライエタリソフトウェアとは違い, 開発者に参加を強制させることはできない. プロジェクトの活性は開発者の自主性とボランティア精神の結果として現れるものである. 管理をしようにもできるはずもない. にも関わらず, 成功したといえるほど多くの利用者を獲得したオープンソースソフトウェアは現に多くの分野でいくつもあり, プロプライエタリソフトウェアと遜色が無いほどに高品質であることもある. ~~これはどういうことだろうか.~~

その理由こそが, スキルのバラつきであり, 人材の多様性の大切さである, と~~著~~筆者者は考える. 人材に多様性を確保することで, 様々な状況の変化に対して強くなり, 弱点を補い合うことができるのである. 均質的な集団はひとつの要因で致命的なダメージを負ってしまう. 多様性によってそれを防ぐ. 結果として, ソフトウェアは活性化することになる.

もう少し踏み込んで具体的に考えると, スキルのばらつきによって, 開発者間のコミュニケーションが発生しやすいからだと筆者は予想する. 開発者のスキルにばらつきがあるということは, そのプロジェクト内でソフトウェアに対する疑問などが多く発生し, その結果として開発者間でのコミュニケーションが大量に発生することになる. コミュニケーションの増加は参加する開発者の活動を刺激し, その結果としてプロジェクト全体の開発活動が活発化すると考えられる.

ここで, ソーシャルネットワークが持つ複雑ネットワークの性質として, 頑強性をあげることができる. これは, ネットワーク上のどこかの経路が破損しても, 他の経路を経由することで, ネットワーク全体の効率には影響が出にくい, ということである. このような性質が開発者スキルのばらつきによって現れている, と筆者は考える. 将来

的には, これらプロジェクトのネットワークを分析することで, さらに詳細な分析が可能になると考えている.

# 6. 結言

本研究では，オープンソースソフトウェアのプロジェクトにおける開発活動が活発なものとそうでないものの違いを分析するために，これを「プロジェクトの活性」と呼び，プロジェクトに携わる開発者達の技術的スキルのばらつきが，プロジェクトの活性と関連しているのかどうかを分析した．この分析のために，GitHub リポジトリからダウンロードした 8140 プロジェクトから収集したプロジェクトネットワークデータを利用した．

我々は，まず，プロジェクトの活性をプロジェクトの最初のコミットから最後のコミットまでの期間でのコミット頻度を利用して定義した．次に，開発者のスキルをプロジェクト内で各開発者が更新したことのあるファイル数と，各開発者の共著者数を利用して定義した．この開発者のスキルの分散 (標準偏差) を用いて，各プロジェクトにおける開発者のスキルのばらつきを定義した．最後に，定義したプロジェクトの活性と開発者スキルの相関をプロジェクトデータ群に対して分析した．その結果，開発者のコミット数に関しては，両対数をとることで，相関係数が 0.62 を示した．開発者の共著関係数に関しては，両対数をとることで，相関係数が 0.49 を示した．これらの結果より，開発者のスキルが均一化されていないプロジェクトが高い活性を示すことを確認した．

## 6.1 今後の研究課題

今後の研究課題として 3 点が挙げられる．

### 6.1.1 開発者スキルの定義の改善

本研究では主に管理や分析の難しさから，GitHub リポジトリに限定して分析を行った．これらの問題を解決できれば，GitHub 以外のオープンソースリポジトリも調査と分析を行うことができる．またリポジトリに限らず，プログラマー個人の活動履歴を追いかけることで，より詳しくプログラマの能力を測定するメトリクスを見つけることができると考える．

### 6.1.2 活性の定義の改善

リポジトリに限らず，バグトラッカーなどのメタデータ，利用状況の統計などを取得し分析することで，より詳しく正確な尺度で測定することができる．これらも膨大なデータを管理し分析するための仕組みをうまく構築する必要がある．より多様な尺度でソフトウェアの開発活動を測定することができるようになれば，より詳しく活性を測定することができる．

### 6.1.3 ネットワーク分析手法の改善

ソーシャルネットワークの分析手法として，今回は単純な尺度を用いた．ネットワークの分析を行うための，より適切な手法を開発できれば，より詳しい結果が得られると予想する．

# 謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本論文の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学部門水野修准教授に厚く御礼申し上げます．本論文執筆にあたり貴重な助言を多数頂きました，本学情報工学専攻椋代凜君，岡嶋秀記君，胡軼凡君，本学情報工学課程川島尚己君，河端駿也君，山田晃久君，学生生活を通じて筆者の支えとなった家族や友人に深く感謝致します．

# 参考文献


[1] H. Kagdi, M.L. Collard, and J.I. Maletic, “A survey and taxonomy of approaches for mining software repositories in the context of software evolution,” J. Softw. Maint. Evol.: Res. Pract., vol.19, pp.77–131, 2007.

[2] A. Mockus, “Amassing and indexing a large sample of version control systems: Towards the census of public source code history,” Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories, pp.11–20, MSR ’09, IEEE Computer Society, Washington, DC, USA, 2009.

[3] J.C. Munson and S.G. Elbaum, “Code churn: A measure for estimating the impact of code change,” Proceedings of the International Conference on Software Maintenance, pp.24–, ICSM ’98, IEEE Computer Society, Washington, DC, USA, 1998.

[4] A. Meneely and L. Williams, “Strengthening the empirical analysis of the relationship between linus’ law and software security,” Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp.9:1–9:10, ESEM ’10, ACM, New York, NY, USA, 2010.

[5] C. Bird, “Sociotechnical coordination and collaboration in open source software,” Proceedings of the 2011 27th IEEE International Conference on Software Maintenance, pp.568–573, ICSM ’11, IEEE Computer Society, Washington, DC, USA, 2011.

[6] F.P. Brooks Jr., The Mythical Man-Month, Addison-Wesley, MA, 1975.

[7] KarlFogel(著)，高木正弘 (訳)，高岡芳成 (訳)，オープンソースソフトウェアの育て方，オライリージャパン，2009.

[8] N.E. Fenton and S.L. Pfleeger, Software Metrics : A Rigorous & Practical Approach, PWS Publishing, 1997.

[9] K.H. Möller and D.J. Paulish, Software Metrics : A Practitioner’s Guide to Improved Product Development, IEEE Press, Chapman & Hall Computing, 1993.

[10] A. Meneely and L. Williams, “Secure open source collaboration: an empirical study of linus’ law,” Proceedings of the 16th ACM conference on Computer and communications

security, pp.453–462, CCS '09, ACM, New York, NY, USA, 2009.

[11] A. Meneely and L. Williams, "Socio-technical developer networks: should we trust our measurements?," Proceedings of the 33rd International Conference on Software Engineering, pp.281–290, ICSE '11, ACM, New York, NY, USA, 2011.

[12] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, pp.24–35, SIGSOFT '08/FSE-16, ACM, New York, NY, USA, 2008.

[13] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, "Putting it all together: Using socio-technical networks to predict failures," Software Reliability Engineering, 2009. ISSRE '09. 20th International Symposium on, pp.109–119, Nov. 2009.

[14] M.S. Granovetter, "The strength of weak ties," American Journal of Sociology, vol.78, no.6, pp.1360–1380, 1973.

[15] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic, "The role of social networks in information diffusion," Proceedings of the 21st International Conference on World Wide Web, pp.519–528, WWW '12, ACM, New York, NY, USA, 2012.

[16] アルバート・ラズロ・バラバシ (著)，青木薫 (翻訳)，新ネットワーク思考—世界のしくみを読み解く，NHK 出版，2002.

[17] 出原真人，"巨大ソフトウェアリポジトリ群へのマイニングに基づくソフトウェア開発者間のネットワーク分析," Master's thesis，京都工芸繊維大学 大学院工芸科学研究科，2013.

# 付録 A.　利用した R コマンド

## A.1　全プロジェクトにおける開発者スキル ($S_1$) とプロジェクト活性の相関

```
> d <- read.csv("dataFrom8000.v3.csv")
> d$com_per_dur <- d$commit / d$duration
> d$duration_day <- d$duration / 86400
> d$com_per_dur <- d$commit / d$duration_day
> d$file_dig_stddev <- sqrt(d$file_dig_variance)
> g <- ggplot(d,aes(com_per_dur,file_dig_stddev))
> g+geom_point(aes(colour=num_authors,size=num_authors,
    alpha=num_authors))+scale_y_log10()+scale_x_log10()
    +xlab("Commit/Duration (count/day) (log)")
    +ylab("Stddev of file-based degrees of developers (log)")
```

## A.2　100 人以上が関わるプロジェクトにおける開発者スキル ($S_1$) とプロジェクト活性の相関

```
> d100 <- subset(d,num_authors > 100)
> cor(log10(d100$com_per_dur),log10(d100$file_dig_stddev))

> g100 <- ggplot(d100,aes(com_per_dur,file_dig_stddev))
> g100+geom_point(aes(colour=num_authors_log10,size=num_authors_log10))
    +scale_y_log10()+scale_x_log10()
    +xlab("Commit/Duration (count/day) (log)")
    +ylab("Stddev of file-based degrees of developers (log)")
```

## A.3　全プロジェクトにおける開発者スキル ($S_2$) とプロジェクト活性の相関

```
> dd1 <- subset(d,auth_deg_variance > 0)
> gg1 <- ggplot(dd1,aes(com_per_day,auth_deg_variance))
> gg1+geom_point(aes(colour=num_authors,size=num_authors))
    +scale_y_log10()+scale_x_log10()+xlab("Commit/Duration (count/day) (log)")
```

```
    +ylab("Variance of co-authorship of developers (log)")
> cor(log10(dd1$com_per_day),log10(dd1$auth_deg_variance))
```