ソフトウェア開発チームの活性化に関する研究

筑波大学審査学位論文（博士）

２０１９

増　　田　　礼　　子

筑波大学大学院
ビジネス科学研究科　企業科学専攻

論文概要

我々の身の回りにある製品やサービスには，数多くのソフトウェアが利活用されている．パーソナルコンピュータ，スマートデバイス，家電製品，自動車など普段の生活の中で誰もが利用している製品から，銀行の預金管理システム，電車や航空券などの座席予約システム，Facebook や Twitter をはじめとした Social Networking Service (SNS) などのサービスもソフトウェアが支えている．このように，ソフトウェアは，現在の社会を支える重要な役割を担い発展している．これらのソフトウェアは，複数のソフトウェア技術者で構成されるソフトウェア開発チームによって開発，運用されることが一般的である．ソフトウェア開発は，決められた予算や期間で開発を進めねばならない．定められた制約を満たし，品質の良い製品を開発するには，ソフトウェア開発を効率的に進める必要がある．そのためには，チームの実力を最大限に引き出すことが重要である．

ソフトウェア開発チームの実力は，チーム間で大きなばらつきがあることが報告されている．一般的に，知識量や作業効率などの個人の実力は，個人が持っているスキルだけではなく，モチベーション (動機付け) の影響を受けると言われている．従業員の作業条件と作業効率の関係を分析したホーソン研究では，作業効率は，物理的な労働環境よりも，職場における人間関係や目標意識の影響を受けることを示している．また，生産性の向上には，目的を達成しようとする従業員らの集団的な意欲や態度の向上が必要であり，集団的な意欲や態度を向上させるためには職場の人間関係が重要であると述べている．効率的なソフトウェア開発には，開発チームの実力が発揮できるチーム状態であるかどうかが重要である．それゆえ，ソフトウェア開発を効率的に遂行するには，ソフトウェア開発チームがチームとしての実力を発揮しやすい状況，すなわちチームを活性化することが不可欠である．ソフトウェア開発チームを如何にして活性化するかは重要なテーマである．本研究では，チームの活性化に影響を及ぼす要因を明らかにすることを目的に，活性化しているソフトウェア開発チームの分析を行った．本研究は，汎用的なチーム間の実力差に関連する組織行動論，人間関係論，行動科学論などの社会科学と，ソフトウェア工学との境界領域を対象分野とした，ソフトウェア開発チームの活性化に関する研究である．

多様なソフトウェアが提供されるようになるにつれて，開発チーム数の増加と，チーム形態の多様化が進んでいる．2013 年に発表された International Data Corporation (IDC) の

調査によると，全世界には 1,850 万人のソフトウェア開発者が存在している．ソフトウェア開発は職業としてだけでなく趣味として実施している者も存在し，その割合は 3:2 と言われている．本研究では，この分類に従って，職業としての開発チームの活性化と，趣味の開発チームの活性化の 2 つを対象とする．代表的な職業としての開発チームは，企業における開発チームであり，趣味の開発チームとして代表的なものは，Open Source Software (OSS) 開発である．そこで，本研究では，企業における開発チームと，OSS 開発チームを研究対象とした．

本論文では，次に示す 3 つの事項を明らかにした．

研究テーマ 1 の「企業の開発チームにおけるチーム状態を計測する質問紙分析手法を探る」では，開発チームのリーダが自身のチームの状態を知ることができるツールの検証と最適な分析手法を探索した．企業の開発チームのリーダを対象としたプロジェクトの運営状況に関する質問紙調査を実施した松尾谷の研究では，プロジェクトの成否に大きな影響を与える因子は「仲間意識」と「役割認知」であることを示した．この研究では，質問紙調査によりプロジェクトの成否を判別することができる可能性を示しているが，一方で判別分析の検証は行っていない．そこで，プロジェクトの成否判別に関して高い判別分析結果を示した松尾谷の研究で実施した質問紙調査データを用い，この研究において不足している判別分析の検証を行った．また，企業の開発チームのリーダが使うツールとしては，調査および分析手法は簡便であることが望ましい．そこで，複数の分析手法を比較し，最適な分析方法を探った．分析方法は，統計モデルと機械学習モデルを用いた．統計モデルでは，説明変数を少数の因子や主成分に集約して用いた．機械学習モデルでは，説明変数を集約せずに用いた．その上で，両モデルの判別率を比較した．

質問紙調査の分析手法の簡便さにおいては，分析過程で因子分析や主成分分析などの統計処理を行う必要がない機械学習モデルは，質問紙調査結果の値そのものを用いて判別ができるため，統計モデルと比較すると簡便であり，開発チームにおいて利用することができると考える．教師ありデータを用いて実施した統計モデルの自己検証では，松尾谷の研究で示された統計モデルの結果と同様の高い判別結果が得られた．機械学習における自己検証の判別率は，統計モデルとほぼ同等かそれ以上の高い判別率であった．一方で，妥当性検証として行った Leave-One-Out-Cross-Validation (LOOCV) による交差検証においては，

統計モデル，機械学習モデル共に外挿となるデータに対しては著しく判別性能が低下することを示した．開発チームのリーダが自身のチームの状態を知るためのツールとして最適な分析方法は，機械学習モデルであると考えられる．しかし，実用化するためには，外挿における判別率の向上が求められる．研究テーマ 1 で試行した評価比較により，次の課題は，学習のための教師付きデータの蓄積，用いる変数の数や変数の加工なども含めた説明変数の改良であることが明らかとなった．

研究テーマ 2 の「OSS 開発チームにおけるメンバと開発量の関係を探る」では，OSS 開発チームがどのように開発を行っているのかについてメンバと開発量との関係の観点から調査し，分析を行った．GitHub における 50 件の OSS 開発プロジェクトを対象とし，OSS 開発に対して，企業のソフトウェア開発におけるコストモデルを応用して分析を行い，OSS 開発の特徴分析を試みた．分析には，言語や対象領域に制限を持たず，かつ実績のあるコストモデルである Constructive Cost Model (COCOMO) を用いた．COCOMO の適用に当たり，企業のソフトウェア開発の工数 (Person-Months) に相当する概念を OSS 開発に適用するため，開発期間を 1 ヶ月単位で区切り，その区間内で 1 回以上の Commit 記録があれば 1 単位とする Effort Person-Months を定義し，回帰的に得られたべき式の係数に着目した計測方法を探った．

分析の結果，OSS 開発においても，COCOMO における量的開発効率の関係式が成り立つことが確認された．また，OSS 開発特有の特性として，開発規模が増加すると開発効率が上昇することが確認された．対象ソフトウェアに関する成果物の著作者毎の開発量の詳細な分析を行った結果，著作者の開発量の順位データは全てべき分布になっていた．そこで，べき分布の Head 部分に影響を及ぼしていると考えられる大規模な流用と Tail 部分に相当する貢献度の小さい多数の開発者の存在を特異値として除去して分析を行った．特異値を除去した評価でも，特異値除去前に見られた OSS 開発特有の特性に変化はなく，特異値として除去した大規模な流用や貢献度の小さい多数の開発者の存在が開発効率に影響を及ぼしていないことを示した．OSS 開発チームでは，全開発量の 80 % が少数のメンバによって行われており，開発を推進するコアメンバの存在が確認された．

研究テーマ 3 の「OSS 開発チームにおけるチームの状態の指標化を試行する」では，研究テーマ 2 の結果を踏まえて，メンバ間の開発活動量のばらつきから開発チームの状態の

指標化を試みた．分析には，研究テーマ 2 で分析対象とした OSS 開発プロジェクトの開発記録 (Log データ) を用いた．活動量を計測するために，計測期間において Commit 記録がある月を 1 とする Effort Person-Months を計測の単位と定義し，総開発期間におけるメンバ毎の総 Effort Person-Months 数 (総活動月数) を基に稼働率を算出した．稼働率に相当する指標として経済学の分野で用いられるジニ係数とローレンツ曲線を用いた．ジニ係数とローレンツ曲線は，経済活動の成果である国全体の所得が各世帯にどのように分配されているのかを調べるときに，最もよく用いられる指標とグラフである．

研究テーマ 3 で試行した手法により，実際の OSS 開発プロジェクトにおいて Effort Person-Months の分布を基に，ジニ係数とローレンツ曲線を用いて，OSS 開発チームの活動量の分布を定量化した．指標として用いたジニ係数の値から，OSS 開発プロジェクトチームにおけるメンバ間の活動量には大きなばらつきがあることが明らかになった．OSS 開発プロジェクトは自由な参加が推奨されている．そのため，すぐに貢献することができない新規参画者が多い場合には，活動量のばらつきは大きくなり，これに伴い成果物に直結する Commit 数にもばらつきが生じると考えた．しかし，活動量のばらつきを示す Effort Person-Months のジニ係数と成果物に直結する Commit との相関は認められなかった．このことから，自由な参加による稼働率の低下の影響はほとんどないと推察される．続いて，OSS 開発プロジェクトの特性を探るため，ジニ係数の値を基にグルーピングを行い，プロジェクト間での各項目の分布を確認したが，活動量のジニ係数を用いて分類した OSS 開発プロジェクト間には，有意な差異は見られなかった．さらに，異なる計測期間で算出した Effort Person-Months のジニ係数を比較した結果，計測期間全てで算出したジニ係数より，最後の 1 年間のみで算出したジニ係数の値が小さいことが分かった．これは，活動量が少なく，活動期間の短い Contributor が多く存在していることを表している．OSS 開発プロジェクトへの参加や離脱はメンバの自由意志に基づくため，興味のある OSS 開発プロジェクトがあれば開発活動に参加することができる．これにより，OSS 開発プロジェクトに多くのメンバが関わることになる．OSS 開発チームは，開発を推進する少数のコアメンバがいる一方で，わずかな開発活動で OSS 開発チームを去って行く Contributor が多く存在している．OSS 開発プロジェクトのジニ係数を時系列で比較した場合の差は顕著に表れており，参加や離脱が自由な OSS 開発チームのメンバ構成は，時間と共に常に変化していると考えられる．この指標を用いて時系列分析を行うことで，OSS 開発チームの成長やチーム状態の変化などを分析することができるようになると考える．

本研究では，チームの活性化に影響を及ぼす要因を明らかにすることを目的に，リーダとメンバの関係といった観点から，企業の開発チームとOSS開発チームについて調査，分析を行った．企業の開発チームにおいては，複数の分析手法を用いて分析を行い，自己検証においては，松尾谷の研究で示されている統計モデル結果と同様の結果が得られた．このことから，開発チームにおいて，プロジェクトの成否に大きな影響を与える因子は「仲間意識」と「役割認知」であると推察される．ただし，本研究で示したモデルは改良の余地があることが明らかになったので，今後改良したモデルで分析を行った上で，プロジェクトの成否に影響を及ぼす要因を特定する必要がある．

OSS開発チームの実態調査では，OSS開発チームにおける明確なリーダの存在は確認できなかった．一方で，開発量や活動量の分布からは，開発を推進する少数のコアメンバとそれ以外の多数のメンバという2つのグループに分かれていると読み解ける．このことから，OSS開発チームにおけるリーダとは，コアメンバに相当すると考える．コアメンバは，自らの開発作業によって開発チームを牽引していく集団的なリーダであると言えよう．

本研究では，チーム形態が異なることから企業の開発チームとOSS開発チームを分けて研究を行った．研究結果から開発チームのリーダに関して考察すると，リーダの位置づけそのものが全く異なるものであると考えられる．企業の開発チームにおけるリーダは，役割としてのリーダであり，メンバに指示をして開発を推進していくリーダである．OSS開発チームにおけるリーダは，自ら開発をして進めていく集団的なリーダである．チーム形態が異なるだけではなく，リーダとしての位置づけも異なっている．そのため，企業の開発チームと，OSS開発チームを単純に比較して考えることはできない．

本研究により，企業の開発チームと，OSS開発チームのそれぞれにおいて，新たな知見を得ることができた．ソフトウェア開発チームの活性化に影響を及ぼす要因については追加分析が必要であるが，本研究は要因の特定に向けた一助になったと考える．今後も継続して両者の調査，分析を行い，開発チームの活性化について研究を行っていく所存である．

目 次

図 目 次

表 目 次

第1章　緒論

我々の身の回りにある製品やサービスには，数多くのソフトウェアが利活用されている．パーソナルコンピュータ，スマートデバイス，家電製品，自動車など普段の生活の中で誰もが利用している製品から，銀行の預金管理システム，電車や航空券などの座席予約システム，Facebook [1] や Twitter [2] をはじめとした Social Networking Service (SNS) などのサービスもソフトウェアが支えている．このように，ソフトウェアは，現在の社会を支える重要な役割を担い発展している．これらのソフトウェアは，複数のソフトウェア技術者で構成されるソフトウェア開発チームによって開発，運用されることが一般的である．ソフトウェア開発は，決められた予算や期間で開発を進めねばならない．定められた制約を満たし，品質の良い製品を開発するには，ソフトウェア開発を効率的に進める必要がある．そのためには，チームの実力を最大限に引き出すことが重要である．

ソフトウェア開発に限らず，さまざまな分野において，チームの実力は，構成メンバのスキルだけではなく，多様な要因によって大きなばらつきが生じている．一般的に，知識量や作業効率などの個人の実力は，個人が持っているスキルだけではなく，モチベーション (動機付け) の影響を受けると言われている [3–7].

従業員の作業条件と作業効率の関係を分析したホーソン研究では，作業効率は，物理的な労働環境よりも，職場における人間関係や目標意識の影響を受けることを示している [8]. また，生産性の向上には従業員らのモラール (morale) [1] の向上が必要であり，モラールを向上させるためには職場の人間関係が重要であると述べている [8].「モラール」とは，目標を達成しようとする集団的な意欲や態度を指す．一方,「モチベーション」は，主に個人の意欲や態度を対象とする場合に用いられる．従来，集団全体の意欲や態度を表す場合には「モラール」が用いられていたが，近年では，個人か集団かに関わらず，いずれの場合においても「モチベーション」という表現が多用されている．本研究では，引用箇所を除き，いずれも「モチベーション」という表現を用いる．ホーソン研究は，チームの実力も，

[1] モラールは，モラルと混同されやすいが，両者の意味は全く異なる．モラール (morale) は，目標を達成しようとする集団的な意欲や態度のことを指す．一方，モラル (moral) は，ルールや規則などのように明確に定められたものではない道徳や倫理を指す．

個人の実力と同様に，個々のメンバの実力 (個人のスキル + モチベーション) の総和だけではなく，チームのモチベーションに相当するチームへの期待や目標，チーム内の人間関係などの影響を受けていることを示している [8].

ソフトウェア開発においても，開発チーム間の実力には大きなばらつきがあることが報告されている [9,10]．効率的なソフトウェア開発には，開発チームの実力が発揮できるチーム状態であるかどうかが重要である．それゆえ，ソフトウェア開発を効率的に遂行するには，ソフトウェア開発チームがチームとしての実力を発揮しやすい状況，すなわちチームを活性化することが不可欠である．ソフトウェア開発チームを如何にして活性化するかは重要なテーマである．本研究では，チームの活性化に影響を及ぼす要因を明らかにすることを目的に，活性化しているソフトウェア開発チームの分析を行った．本研究は，汎用的なチーム間の実力差に関連する組織行動論，人間関係論，行動科学論などの社会科学と，ソフトウェア工学との境界領域を対象分野とした，ソフトウェア開発チーム (以下，開発チームまたはチームと記載する) の活性化に関する研究である．

多様なソフトウェアが提供されるようになるにつれて，開発チーム数の増加と，チーム形態の多様化が進んでいる．2013 年に発表された International Data Corporation (IDC) の調査によると，全世界には 1,850 万人のソフトウェア開発者が存在している [11]．その約 40 % に当たる開発者が，趣味でソフトウェア開発をしている開発者 (Hobbyist developers) であると報告されている [11]．つまり，ソフトウェア開発は職業としてだけでなく趣味として実施している者も存在し，その割合は 3:2 である．本研究では，この分類に従って，職業としての開発チームの活性化と，趣味の開発チームの活性化の 2 つを対象とする．

代表的な職業としての開発チームは，企業における開発チームである．本研究では，職業としての開発チームとして，企業における開発チームを対象とした．企業の開発チームには，業務を推進するリーダと実行するメンバが存在している．リーダには，仕事面だけではなく，職場環境や開発環境などの環境面にも配慮し，チーム内で良い人間関係を築くことが求められている [12–18]．企業の開発チームの活性化に関しては，さまざまな先行研究 [9,10,19–36] がある．これらの研究の多くは，開発チームのメンバ間のコミュニケーションやリーダとしての振る舞い (リーダシップ) といった観点に基づいている．一方で，開発チームのリーダとメンバの関係性といった観点からの研究は少ない [10]．松尾谷は，企業

における開発チームのリーダを対象に，開発チームの人間関係に重点を置いたプロジェクトの運営状況について質問紙調査を行った [10]．この研究では，質問紙調査結果からプロジェクトの成否を判別することができる可能性を示しているが，一方で判別分析の検証は行っていない．企業の開発チームでは，自身の開発チームの状態を調査することは一般的ではない．開発チームのリーダが自身のチームの活性状態を知ることができるツールがあれば，チームの問題をタイムリーに改善したり，次の開発チームの運営に活かしたりすることができるようになる．そのためには，質問紙調査結果を用いたプロジェクトの成否判別に関して高い判別結果を示した松尾谷の研究 [10] で行われていない判別分析の検証を行うことが重要である．未知のデータを用いて検証を行うことにより，汎用的に質問紙調査結果からプロジェクトの成否判別ができるのかどうかを確認することができる．また，実務の中での利用を考えると，分析手法は簡便であることが望ましい．そこで，リーダとメンバの関係性という観点から開発チームの活性状態を探る手法として，松尾谷の研究 [10] を基にして，不足している判別分析の検証と，最適な分析手法を探索することを研究テーマ 1 とした．

もうひとつの課題は，趣味の開発チームにおけるリーダとメンバの関係性を探ることである．近年，オープンソース・ソフトウェア (OSS: Open Source Software) は目覚ましい発展を遂げており，さまざまなアプリケーションが開発され，無償で使用されている [37]．OSS 開発チームは，今後のソフトウェア開発を牽引するチーム形態のひとつであると言えよう．そこで，多種多様である趣味の開発チームの代表として，OSS 開発を対象とした．OSS 開発に関しては，さまざまな研究が行われているが，その多くは個人のスキルや成長に焦点を当てた研究が占めている [38–42]．OSS 開発集団をチームと捉え，チームとしてどのように開発を行っているのかといった，開発チームの開発実態を明らかにした研究は見当たらない．そのため，OSS 開発チーム内における構成員の役割についても明らかになっていない．先行研究の調査により，OSS 開発チームにおけるリーダとメンバの関係性を議論するためには，その前段階として明らかにしなければならない課題が 2 つあることが分かった．ひとつは，OSS 開発チームの実態を明らかにすることである．そこで，OSS 開発チームがどのように開発を行っているのか，活性化している OSS 開発チームを対象とし，開発チームの実態を調査することを研究テーマ 2 とした．もうひとつの課題は，開発チームの状態を測定する手法を探ることである．OSS 開発チームは，企業における開発チームとは異なり，その規模はさまざまであり，大きいものは 1 万人を超えるメンバが参加している．

そのため，企業における開発チームのように構成員に対して質問紙調査を実施することは難しい．そこで，研究テーマ 2 の結果を踏まえ，OSS 開発チームの状態を計測する手法を探ることを研究テーマ 3 とした．

本論文の構成を次に示す．第 2 章にて，開発チームの活性化に関して，企業の開発チームと OSS 開発チームに分けて俯瞰し，リーダとメンバの関係性という視点で関連研究をサーベイする．サーベイにより明らかになった課題を基に，本研究のテーマを具体化し，設定する．第 3 章では，企業のソフトウェア開発プロジェクトの成否に影響を及ぼす要因に関する研究 [10] における質問紙調査結果を用いて，不足している判別分析の検証を行うと共に，企業の開発チームのリーダが自身のチームの状態を知ることができるツールとして最適な手法を探索し，その結果を示す．第 4 章では，OSS 開発チームの開発記録 (Log) を用いて対象ソフトウェアに関する成果物の著作者の開発状況を分析し，OSS 開発チームの実態を明らかにする．第 5 章では，OSS 開発チーム構成員の活動量の分布の偏りから，OSS 開発チームの状態を指標化する手法を検証し，その結果を示す．最後に，第 6 章では，開発チームの活性化に関する研究により得られた知見を総括し，開発チームのリーダとメンバの関係について考察する．

第2章　チームの活性化とソフトウェア開発形態に関する研究

2.1　緒言

本章では，まず，チームの活性化に関する歴史的な研究と，ソフトウェア開発形態の変化について概観する．続いて，企業の開発チームと OSS 開発チームのそれぞれについて，リーダとメンバの関係性という観点に焦点を当てて関連研究をサーベイする．最後に，サーベイにより明らかになった課題を基に，本研究のテーマを具体化し，設定する．

2.2 チーム活性化に関する実証研究

本節では，ソフトウェア開発と，他の分野におけるチーム活性化に関連する研究を概観する．

2.2.1 ソフトウェア開発におけるチーム活性化

本項では，ソフトウェア開発におけるチーム活性化に関する歴史的な研究をサーベイする．

Dijkstra が提唱したプログラムを記述する方法論である構造化プログラミング [43,44] は，開発チームの能力差と原因因子を調査し，構造化プログラミングを用いたチームが高い生産性を示すという調査結果に基づいている．この観点から捉えると，Dijkstra による研究は，プログラミング技法を用いてチームを活性化した研究と捉えることができる．

1981 年，Boehm が Constructive Cost Model (COCOMO) を提唱した [45]．COCOMO は，現在，ソフトウェア開発の工数や開発期間の見積もり手法のひとつとなっている．Boehm は，TRW 社の 63 の開発チームを対象に大規模な実証研究を行った．この調査では,「チーム力」を計量するため，複数の管理者がチーム力に関する項目に対してスコアリングを行い，チーム力を決定した．COCOMO では，ソフトウェアの品質・コスト・納期 (QCD: Quality, Cost, Delivery) に影響を与える最大の因子が「チーム力」であると示している [45].

寺本らは，国内通信メーカにおいて大規模な調査と分析を実施した [46, 47]．この分析では,「チーム力」を 3 つの因子に分解し，計量にバイアスがかからない工夫を行っている [46,47]．その結果,「チーム力」が最大の影響因子であるという Boehm の研究結果を追証する結果を得ている [46,47].

Boehm はその後も研究を続け，2000 年には，計算手法を改良し，拡張した COCOMO Ⅱ を提唱した [48–50]．COCOMO Ⅱ では,「チームの結束度」として「チーム力」を表している [48–50]

2.2.2 社会科学におけるチーム活性化

ソフトウェア産業が始まる以前より，他の産業においてチーム活性化に関する研究は行われている．代表的な分野は，経営学，組織行動論を始めとした社会科学である．本項では，社会科学におけるチーム活性化に関連する歴史的な発達を概観する．

1900 年代，急激な工業化により，市場は過当競争となっていった．経営者は，利益を求め，QCD の向上を推進する一方で，生産工場などの現場では労働強化となり，仕事意欲が

低下していった．1911 年に，Taylor が科学的管理法 (Scientific Management) [3] を提唱した．科学的管理法は,「課業管理」·「作業の標準化」·「作業管理のための最適な組織形態」の 3 つの要素から成り立っている [3]．科学的管理法とは，客観的な管理基準を作り，労使協調体制を構築することで，労働者のモチベーションが向上し，結果として，生産性が増強され，労働者の賃金の上昇につながり，労使が共存共栄できるという考えである [3]．Tylor の研究以降，モチベーションに関するさまざまな研究が行われ，個人の実力は，知識量や作業効率などの個人が持っているスキルだけではなく，モチベーションの影響を受けると報告されている [3–7].

1924 年から 1932 年まで，Mayo や Roethlisberger らにより，Western Electric 社の Hawthorne 工場において，科学的管理法の実証研究を目的として実験が行われた [8]．この一連の研究は，ホーソン研究 (Hawthorne Studies/ Hawthorne Works) と呼ばれている．この研究では，生産性向上のための条件を明らかにするため，まず，照明実験が行われ，その後，リレー組み立て実験，面接調査，バンク配線作業実験などの実験や調査が行われた．この研究により,「労働者の作業効率は，物理的な労働環境よりも，職場における人間関係や目標意識に左右される」ということが明らかになった．Mayo は，職場には公式な組織とは異なる非公式組織 (Informal Group) が存在し，この非公式組織における仲間意識や規範が作業効率に影響を与えると主張した [8]．それまで主流だった科学的管理法では，従業員は仕事の単純化と経済的報酬以外に対して無関心であることを不可避かつ望ましいことのように見せたのに対し，ホーソン研究の研究者らは，仕事において意味があると思われたものの多くが「他者との関係」であることを確認した [8]．Mayo と Roethlisberger は，生産性の向上には従業員のモラールの向上が必要であり，モラールを向上させるためには職場の人間関係の改善が必要であるとする理論，人間関係論を提唱した．

尾高は，モラールを「仕事への満足」,「仕事の意義の自覚」,「集団への帰属意識」,「集団の団結力」の 4 つの要因に分類している [51].「仕事への満足」と「仕事の意義の自覚」は，集団の中での個人の適応に関するもので，職務満足に焦点を当てている [52].「集団への帰属意識」と「集団の団結力」は，仲間との一体感や団結心といった集団維持意識に焦点を当てている [52]．角山は，モラールとは，仕事への満足感や集団との一体感を中心に据えた態度の概念であり，目標達成に向かう個人のエネルギーに焦点を置くモチベーションの概念とは異なる面を持っていると述べている [52]．このように,「モラール」とは，目標を達成しようとする集団的な意欲や態度を指す.「モチベーション」とは，主に個人の意欲や態

度を対象とする場合に用いられる．従来，集団全体の意欲や態度を表す場合には「モラール」が用いられていたが，近年では，いずれの場合においても「モチベーション」という表現が多用されている．本研究では，引用箇所を除き，いずれも「モチベーション」という表現を用いる．

2.2.3 チーム活性化に関する実証研究のまとめ

組織行動論におけるチーム活性化の研究では，個人の実力は，個人が持っているスキルだけではなく，モチベーションの影響を受けること [3–7]，QCD の向上には，集団としてのモチベーションが重要であることを示した [8]．その後に実施されたソフトウェア開発チームを対象とした Boehm の研究では，QCD に影響を及ぼす最大因子が「チーム力」であることを示した．このことは，開発チームのチーム力は，チーム間でばらつきがあることを意味している．つまり，チームの実力も，個人の実力と同様に，個々のメンバの実力 (個人のスキル + モチベーション) の総和だけではなく，チームのモチベーションに相当するチームへの期待や目標，チーム内の人間関係などの影響を受けていると考えられる．

これらの研究が示すように，ソフトウェア開発においても QCD を向上させるには，開発チームの実力が発揮できるチーム状態であるかどうかが重要である．それゆえ，ソフトウェア開発を効率的に遂行するには，ソフトウェア開発チームがチームとしての実力を発揮しやすい状況，すなわちチームを活性化することが不可欠である．ソフトウェア開発チームを如何にして活性化するかは重要なテーマである．本研究では，チームの活性化に影響を及ぼす要因を明らかにすることを目的に，活性化しているソフトウェア開発チームの分析を行った．

2.3 ソフトウェア開発チームの形態

本節では，ソフトウェア開発チーム形態を概観し，本研究における研究対象を定める．

近年，多様なソフトウェアが提供されるようになり，それに伴い，開発チーム数の増加と，チーム形態の多様化が進んでいる．2013 年に発表された International Data Corporation (IDC) の調査によると，全世界には 1,850 万人のソフトウェア開発者が存在し，開発チームを構成している [11]．その約 40 % に当たる開発者が，趣味でソフトウェア開発をしている開発者 (Hobbyist developers) [1]であると報告されている [11]．つまり，ソフトウェア開発は職業としてだけでなく趣味として実施している者も存在し，その割合は 3:2 である．

本研究では，この分類に従って，職業としての開発チームの活性化と，趣味の開発チームの活性化の 2 つを対象とした．代表的な職業としての開発チームは，企業における開発チームであり，趣味の開発チームとして代表的なものは，OSS 開発チームである．そこで，本研究では，企業における開発チームと，OSS 開発チームを研究対象とした．

[1]IDC の調査では，主な収入をソフトウェア開発から得ていないが，月に 10 時間以上コンピュータやモバイルデバイス用のプログラムを書く人を趣味の開発者 (Hobbyist developers) として区分している．

2.4 企業のソフトウェア開発チーム活性化に関する研究

本節では，企業におけるソフトウェア開発チームを対象とし，チーム活性化に関連する研究を概観する．

2.4.1 企業のソフトウェア開発チームを対象とした研究

企業のソフトウェア開発は，主にプロジェクト型の開発チームでソフトウェア開発を行っている．企業におけるプロジェクトとは，独自のプロダクト，サービス，所産を創造するために実施される有期的な業務 [53] であり，プロジェクト型の開発チームとは，企業などにおける固定的な組織とは異なり，プロジェクトにおける共通の目的の達成を遂行するために集められたメンバによって構成された集団である．近年では，開発チームのメンバは，企業内に限らず，異なる企業や組織に属するメンバも含む混成の集団が多くなってきている [54–58].

ソフトウェア開発において QCD の向上は，ソフトウェア工学共通の課題である [59]．ソフトウェア工学では，この課題に対し，技術的な側面とプロセス的な側面からのアプローチにより研究が行われている．たとえば，厳密な仕様記述方法，テスト技法などの技術的なアプローチからの研究 [60–62] や，プロジェクト遂行におけるリスク要因の分析やこれらの分析結果を基にしたプロセス改善の提案などのプロセス的なアプローチからの研究 [63–66] がある．ソフトウェア開発では，対象分野やプロジェクトの置かれている技術的，組織的，ビジネス的な状況などのさまざまな要因により，適した技法やプロセスを選ぶ必要がある [67].

このように，ソフトウェア工学では，技術的な側面とプロセス的な側面からのアプローチからの研究が主流であった．有期限で集められたメンバのスキルやバックグラウンドは多種多様であり，開発チーム間の QCD の差異は大きい [9,10,68]．次項では，2.2 節で述べたチームの活性化という人的側面からのアプローチを行った企業のソフトウェア開発における研究を概観する．

2.4.2 企業のソフトウェア開発チーム活性化に関する研究

ソフトウェア開発において，生産性が「技法やツール」,「プロセス」に依存することはよく知られており，前 2.4.1 項で述べた通り，さまざまな研究と実用化が行なわれている．1987 年に，Tom DeMarco らは,「プロジェクトでは，人に関する問題が技術的な問題より

もトラブルの原因になることは間違いない」と指摘した [25,69]．この指摘以来，アメリカでは，ハードウェア，ソフトウェアに次いで「ピープルウェア (Peopleware)」という言葉が認識され，プロジェクトの環境に着目した改善活動が行われている [25]．一方で，日本における「チーム力」に関する研究や実用化は，製造業などの他の分野と比較すると遅れている．たとえば製造業では「チーム力」を「現場力」と呼び，国際競争力の根源と位置付けられているが，ソフトウェア開発を始めとする IT 業界の現場力は低いという提言がされている [70–72].

ソフトウェア開発における人的側面からのアプローチを行っている研究としては,「従業員満足」 (Employee Satisfaction) の考え方を，日本のソフトウェア開発チームの特性に合わせて拡張し，共に働く同じチームの個々のメンバをパートナーと定義して大規模な計量を行った「パートナー満足」(PS: Partner Satisfaction) 研究がある [19,20]．PS 研究では，パートナー満足を主体とした人的な観点からさまざまな研究が行われている [9,10,21–25,27,28,30–35].

PS 研究は，Herzberg の二要因理論 (動機付け・衛生理論: Herzberg's Theory of Motivation) を出発点とし，仕事への満足要因群 (動機付け要因) を促進し，不満足要因群 (環境要因) を緩和することで，モチベーションを向上させ，開発チームの成功を目的としている [25]. 原田は，開発チームの成功というゴールを達成するためには，そのゴールの達成に必要な能力であるケーパビリティがチームに備わっていなければならないが，ケーパビリティを実現するためのエンジンとなる“ ヒト ”が動かなければチームの成功は難しいと述べている [25].

PS 研究では，モチベーションを測定する PS 尺度を開発し，2002 年から延べ 3,000 人以上に調査を実施している [9,25]．開発チームメンバを対象とした PS 調査では，チームメンバのモチベーションに影響を及ぼす 7 つの要因 (モチベーション・ドライバ) を特定している [23]．Fig. 2.1 に，この 7 つの要因 [23] を示すと共に，各要因について簡単に説明する．

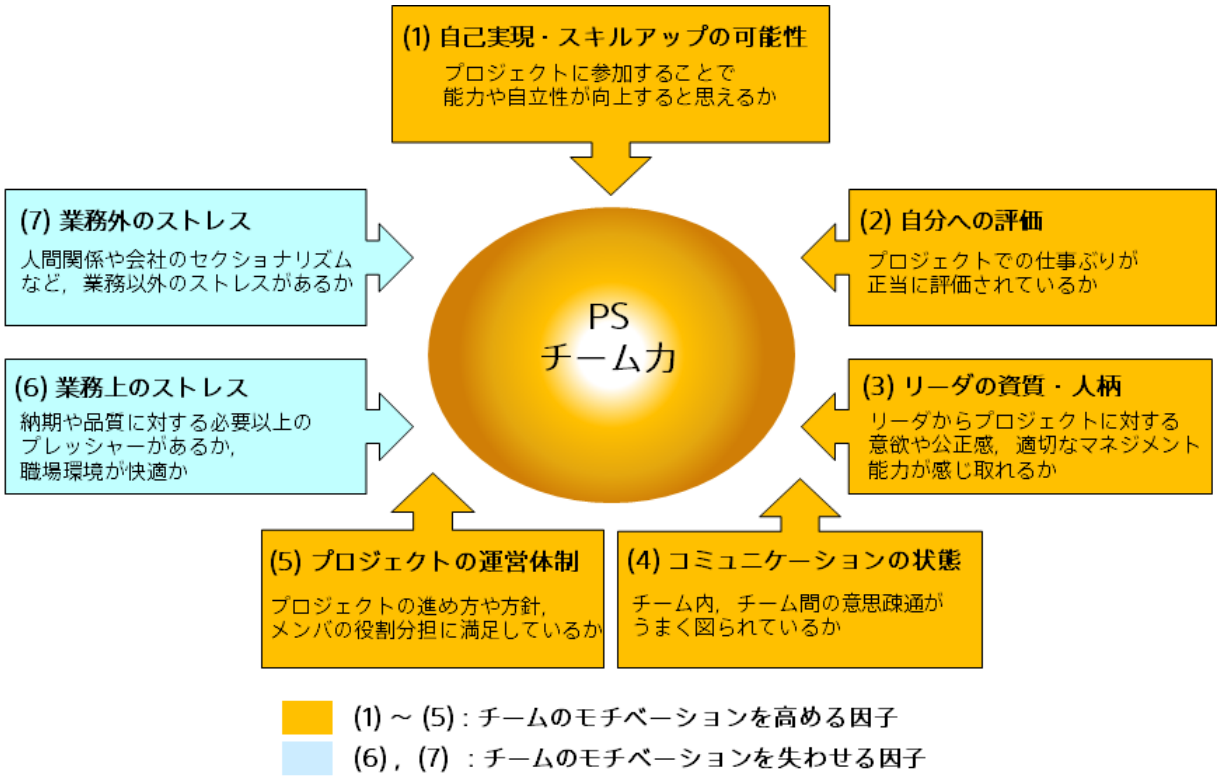

Fig. 2.1: Seven Drivers that Influence Team Members' Motivation [2]

(1) 自己実現・スキルアップの可能性 ：
プロジェクトに参加することで能力や自立性が向上すると思えるか

(2) 自分への評価 ：プロジェクトでの仕事ぶりが正当に評価されているか

(3) リーダの資質・人柄 ：リーダからプロジェクトに対する意欲や公正感，
適切なマネジメント能力が感じ取れるか

(4) コミュニケーションの状態 ：チーム内，チーム間の意思疎通がうまく図られているか

(5) プロジェクトの運営体制 ：プロジェクトの進め方や方針，役割分担に満足しているか

(6) 業務上のストレス ：納期や品質に対する必要以上のプレッシャーがあるか，
職場環境が快適か

(7) 業務外のストレス ：人間関係や会社のセクショナリズムなど，
業務以外のストレスがあるか

[2]出典：「チームメンバのモチベーションに影響を及ぼす 7 つの要因」[23] を基に筆者作成

Fig. 2.1 の (1) ～ (5) は満足要因であり，(6) ～ (7) が不満足要因である．PS 調査結果は，これら 7 つの要因の中で，モチベーションへの影響が大きい要因は，(4) コミュニケーションの状態と，(5) プロジェクトの運営体制であることを示した [23].

PS 研究では，これらの研究成果を基に，企業の開発チームにおいて，どのようにすればチームメンバのモチベーションを引き出すことができるか，チーム活性化の実践手法やノウハウに関する研究と実践を行った [25,27,73]．これらの実践手法やノウハウを用いてチーム活性化を目指したチームビルディングなどの取り組みに関する事例も報告されている [31,58,73–75]．榎田らは，プロジェクトに良いチームビルディングを導入するには，リーダが人間関係を構築するスキルを身につける必要があると述べている [73]．榎田らは，チームビルディングに必要なスキルを開発するため，集団維持に関するスキル，チームの規範と同調のスキル，コミュニケーションのスキルの向上に重点を置いた育成プログラムを策定した [73]．増田は，プロジェクトの成功に向けてチームとして課題と向き合っていくことを共通認識として，自主的に問題の解決を行うことができるチームを作ることを目的としてチームビルディングを導入し，10 年に渡って継続実施した事例を報告している [74]．この事例は，チームビルディング活動を行ってきたチームのメンバと，チームビルディング活動を行っていない他の約 70 チーム群のメンバに対して実施した質問紙調査の回答の比較では，両者の間に明確な差異があることを示した [31]．さらに，回答に有意な差異が生じている質問項目を抽出して主成分分析を行い,「自分たちのチームに対する自信や信頼」,「役割に対する納得感や態度」,「仕事に対する技術的な誇り」,「仕事に対する自由度」の 4 つの因子を抽出した．増田は，これらの因子が差異を発生させている因子であり，チームビルディングによって培われたチーム力の主要な構成要素であると述べている [31]．また，抽出した因子の組み合わせを変えて実施した判別分析の結果，特に大きな影響を与える因子は,「仲間意識」と「役割意識」に相当する「自分たちのチームに対する自信や信頼」,「役割に対する納得感や態度」の 2 因子であると述べている [31]．山川は，職種を超えた連携を目指し，デザイナーとソフトウェア技術者を同じチーム編成としてチームビルディングを実施した事例を報告している [75]．編成の組み替えとチームビルディングを実施してから 6 ヶ月経過後に質問紙調査を行い，同事業部内の他のチームとの比較を行った結果,「仲間意識」,「役割意識」,「規範意識」,「成果意識」,「仕事の難易度，環境」の全てにおいて明確な差異が認められたことが示されている [75].

2016 年の Google における研究 [36] では,「効果的なチームを可能とする条件は何か」を

研究目的とし，チーム内の状況にまで踏み込んだ調査を行い，チームの状態と生産性との関係を示している．Google の研究では,「誰がチームのメンバであるか」よりも「チームがどのように協力しているか」が重要であるとして，チーム効率性に影響を及ぼす順に次の 5 つの因子を挙げている [36].

(i) 「心理的安全性」:
対人関係においてリスクのある行動を取ったとしても，
このチームなら大丈夫だと信じられる

(ii) 「相互信頼」:
相互信頼の高いチームのメンバは，クオリティの高い仕事を時間内に仕上げる

(iii) 「構造と明確さ」:
職務上で要求されていること，その要求を満たすためのプロセス，
メンバの行動がもたらす成果について，個々のメンバが理解している

(iv) 「仕事の意味」:
仕事そのもの，またはその成果に対して目的意識を感じられる

(v) 「インパクト」:
自分の仕事には意義があるとメンバが主観的に思える

QCD に影響を及ぼしているのは，COCOMO の研究では「チーム力」であり [45]，PS 研究では「仲間意識」と「役割意識」である [10,31]．Google の研究結果は，COCOMO の研究や PS 研究と同様に，開発チームメンバの関係性が重要であることを示している．

2.4.3 リーダシップに関する研究

従業員を管理するマネジャーやリーダは，さまざまな背景や経験を持つ人を管理し，良い人間関係を醸成させる必要がある．本項では，リーダシップに関する研究を概観する．

1940 年代より，リーダシップ理論のアプローチ方法として，リーダの資質ではなく，良いリーダ行動とは何かを明らかにするリーダシップ行動論が発展した．

リーダシップ行動論に関する実証研究のひとつにオハイオ研究 (The Ohio State Leadership Studies) [12, 13] がある．オハイオ研究は，1950 年代に，心理学者の Shartle がリーダの行動を測定する尺度構築を目的に行った研究である．Shartle は，25,000 人以上の民間や軍の組織について調査，分析を行った [13, 76]．Shartle は，分析の結果，リーダの行動は「構造づくり」(Initiating Structure) と「配慮」 (Consideration) の 2 つに集約されることを明らかにした.「構造づくり」とは，組織の成果向上のためインフラの整備や部下の課題管理を徹底する行動である.「配慮」とは，部下と相互に信頼し合い，より良い人間関係を維持しようとする行動である．オハイオ研究では，構造づくり型のリーダシップと，配慮型のリーダシップは両立するとし，両方の行動が共に高いリーダが優れたリーダであるとしている．

1961 年，Likert は，マネジメント・システム論 (Likert's System of Management) を提唱した [14]．大手保険会社のリーダを対象に行った研究は，ミシガン研究 (The Michigan Leadership Studies) とも呼ばれている．Likert は，組織を管理者の下でメンバが相互作用する「システム」と捉え，システムを 4 つのパターンに分類した．4 つのパターンとは,「権威主義・専制型 (システム 1)」・「温情・専制型 (システム 2)」・「参画協調型 (システム 3)」・「民主主義型 (システム 4)」である．Likert は，これら 4 つの中で，民主主義型 (システム 4) の組織の業績が最も高いと主張した．この研究により，高い生産性をあげる組織は，人間的側面と目標指向的な作業チームの形成を図る「部下中心型」であり，関係志向のマネジメントを行っていることが多く，低い生産性の組織は，リーダが常に生産性をあげるよう圧力をかけようとする「仕事中心型」であり，課題志向のマネジメントを行っていることが多いということが明らかになった．

日本では，1966 年に，三隅が PM 理論を提唱した [15–17]．この研究では，450 項目の事前調査から 13 要因・232 項目の調査票を作成し，5,200 名に調査を実施した．研究の結果，三隅は，リーダシップは 2 つの行動機能の大小により構成されるとした．2 つの行動機能は，P: Performance (目的達成機能) と，M: Maintenance (集団維持機能) である．P 行動は，目標設定や計画立案，メンバへの指示などにより目標を達成する行動であり，M 行動はメンバ間の人間関係を良好に保ち，集団のまとまりを維持する行動である．三隅は，P 行動と M 行動の大小を大きい場合は大文字，小さい場合は小文字で表現し，4 つのリーダシップタイプ (PM 型，Pm 型，pM 型，pm 型) を示し，P と M の能力が共に高い PM 型のリーダシップが望ましいと示した [15–17]．また，P 行動には計画 P 行動と圧力 P 行動があり，計画 P 行動は職務を遂行するために部下を指導したり，自分の仕事上の有能さを示したり

するリーダの行動で，圧力 P 行動は職務を遂行するように部下に圧力をかけるリーダの行動であるとした．圧力 P 行動は，部下に心的抵抗，緊張，葛藤を与え，部下の動機付けを減少させるとしている [15–17, 77].

2016 年の Google における研究 [18] では,「優れたマネジャーの要件を特定する」ことを研究目的とし，社員アンケートのコメントと業績評価を分析し，評価の高いマネジャーに共通する 10 の行動様式を示している．次に，Google における優れたマネジャーの要件を示す [18].

(a) 良いコーチである

(b) チームに任せ，細かく管理しない

(c) チームの仕事面の成果だけでなく，健康を含めた充足に配慮し，
インクルーシブ (包括的) なチーム環境を作る

(d) 生産性が高く，結果を重視する

(e) 効果的なコミュニケーションをする (人の話をよく聞き，情報を共有する)

(f) キャリア開発をサポートし，パフォーマンスについて話し合う

(g) 明確なビジョンや戦略を持ち，チームと共有する

(h) チームにアドバイスできる専門知識がある

(i) 部門の枠を越えてコラボレーションを行う

(j) 決断力がある

Google の研究では，社員アンケートと業績評価から抽出されたものであるため，これらの要件が同じように他の組織で当てはまるとは限らないとし，Google がどのような点に着目したかを示している．Google では，社員アンケートと業績評価から，マネジャーがチームメンバのパフォーマンスや幸福度に大きな影響を与える存在であるかを判定したと示している．

これらの研究は，リーダには，メンバのパフォーマンス向上を目指し，仕事面だけではなく，職場環境や開発環境などの環境面にも配慮し，チーム内で良い人間関係を築くことが求められることを示している．

2.4.4 リーダとメンバの関係性に関する課題

2.4.2 項 で概観した企業におけるソフトウェア開発チームの活性化に関する研究や事例報告は，開発チームのメンバ間のコミュニケーションや，リーダとしての振る舞い (リーダシップ) といった観点に基づくものが多くを占めている．Table 2.1 に，開発チームの活性化に関連する先行研究を対象分野毎に分類して示す [3].

Table 2.1: Summary of Previous Research on Activation of Development Team

	チーム/組織	リーダ	メンバ	メンバ間	リーダとメンバ間
行動	[9] [24]				
計測/評価/要因分析	[29] [30] [31] [32]			[31] [35]	[10]
生産性/パフォーマンス	[19] [20]			[35]	
コミュニケーション	[21] [22] [26]		[21] [28]		
モチベーション	[23]		[23]		
チームビルディング	[27]			[27]	
プロセス	[22]				
人的マネジメント		[25]			
リーダシップ		[15] [16] [17] [18]			
組織文化				[36]	

企業のプロジェクト型の開発チームには，プロジェクトを推進するリーダと実行するメンバが存在している．2.4.3 項で示した通り，リーダには，仕事面だけではなく，職場環境や開発環境などの環境面にも配慮し，チーム内で良い人間関係を築くことが求められている．原田は，PS を高めてモチベーションを向上させ，プロジェクトを成功させるためのキーマ

[3]表中の番号は，参考文献の参照番号である．また，複数の分野を対象としている研究は，重複して掲載している．

ンはリーダであると述べている [25]．しかしながら，Table 2.1 に示した通り，開発チームのリーダとメンバの関係性といった観点からの研究は少ない [10]．

松尾谷の研究 [10] では，開発チームのリーダを対象に，開発チームの人間関係に重点を置いたプロジェクトの運営状況に関する質問紙調査を行った．調査の結果，プロジェクトの成否に大きな影響を与える因子は「仲間意識」と「役割認知」であると述べ，質問紙調査結果からプロジェクトの成否を判別することができる可能性を示している．この研究では，2 種類の質問紙を用い，2 つのプロジェクト群に対して「現場力」[4] を調査した．調査結果に対して主成分分析を行い，抽出された主成分を説明変数として用い，同質問紙内で回答されているプロジェクト成否を目的変数として判別分析を行った．分析の結果，それぞれのプロジェクト群において，95 % と 82 % という高い判別率であったことを示している [10]．一方で，この研究 [10] では，判別分析の検証を行っていない点が課題である．

企業の開発チームでは，自身の開発チームの状態を調査することは一般的ではない．開発チームにおいて，開発中に質問紙調査の分析ができれば，開発チームにおいてチームの活性状態を知ることができ，チームの問題点をタイムリーに改善することができるようになる．開発後に質問紙調査の分析ができれば，どこにチームの問題があったのかを知ることができ，次の開発チームの運営に活かすことができるようになる．そのためには，プロジェクトの成否判別に関して高い判別結果を示した松尾谷の研究 [10] で行われていない判別分析の検証を行うことが重要である．未知のデータを用いて検証を行うことにより，汎用的に質問紙調査結果からプロジェクトの成否判別ができるのかどうかを確認することができる．また，実務の中で利用するツールとしては，調査および分析手法は簡便であることが望ましい．質問紙調査の最適な分析手法を明らかにすることで，企業の開発チームのリーダが，自身のチームの活性状態を知ることができるツールとして利活用できるようになると考えた．

[4] 先行研究 [10] では，本研究における「チーム活性化」を「現場力」と表現している．

2.5 OSS 開発チーム活性化に関する研究

本節では，OSS 開発チームを対象とし，チーム活性化に関連する研究を概観する．

2.5.1 OSS の出現と発展

ソフトウェア工学では，ソフトウェア開発を取り巻く技術やビジネス環境の急速な変化，発展も共通課題のひとつである [67]．近年，OSS は目覚ましい発展を遂げている．Web サーバにおける LAMP (Linux, Apache, MySQL, PHP / Perl / Python) [78–83] や携帯端末における Android [84] などが核となり，さまざまなアプリケーションが開発され，無償で使用されている [37]．ソフトウェア工学においても OSS の特徴に関連して「オープンソース・ソフトウェア工学」のシリーズで多くの研究が報告されている [85–87].

OSS とは，ソースコードを公開し，改変，再頒布することができることができるソフトウェアの総称であり，オープンソース・ライセンスに従えば，誰でも自由に改変，再頒布を行うことができる．ソースコードとは，プログラミング言語で記述されたプログラムの設計図とも呼ばれるテキスト (文字列) データである．OSS 開発は，自発的に参加する開発者らによる活動が中心であり，限定メンバによるクローズドな環境で運営している開発を除き，誰でも，いつでも開発チームに参加，離脱することができる．

オープンソース・ソフトウェアの歴史は，著作権により保護され，開示されない商用ソースコードに対する反発から始まった [88]．Richard Stallman は，ソフトウェアを人類共有の財産として自由に利用することは，さらに優れたソフトウェアの開発に繋がると考え，ソフトウェアの実行，複製，頒布，学習，改善する自由を提供する「フリーソフトウェア」(Free Software) を提唱した [89]．その後，ソースコードの公開を保証しながらソフトウェアの開発を進めていくプロセスに着目した集団が現れ,「オープンソース・ソフトウェア」という名称を作りだした [88].

Linus Torvalds は，Linux の開発を始めたモチベーションは，ソフトウェアの自由を通じた人類の財産作りといった社会貢献的なものではなく,「それが僕には楽しかったから」(”Just for Fun”) であると述べている [90]．竹田は，OSS 開発チームへの参加動機に関する複数の調査結果をまとめ，次のように述べている.「Linus Torvalds のように知的好奇心を満足させるものもいれば，ソフトウェア開発能力の向上，自分の評判を高める，自分が作りたいソフトウェアを作る，そして，コミュニティでの評価を高めたい，などが参加動機であると答えるものもいる．評判を高めるためには，参加するプロジェクトが著名である必要が

あるが，実際のところ進行中のオープンソース・ソフトウェア開発プロジェクトは，数万ある．ほとんどは，無名のプロジェクトである．したがって，多くのプロジェクト参加者たちは，自らの使いたいソフトウェアが世の中にはないのだが，自分一人で作るのは大変だ，というような考えで参加していることになる．つまり，このような動機付けを中心に，知的好奇心の追求をまで含め，参加者は社会のためにソフトウェアを開発しているのではなく，参加者個人の自己利益を追求するためにプロジェクトに参加している.」[88]

このように，OSS 開発チームに参加するメンバの動機はさまざまであるが，OSS は，普及の過程において標準化が進んでおり，企業にとっても，投資コストの低価格化という点で注目を浴びている．近年，情報産業のビジネスの中心が，ハードウェアからソフトウェアに，そしてさらにソリューションへと移行していることから，OSS が企業戦略の中に深く取り込まれたり，企業間ネットワークを形成する手段となったりしており，OSS と企業の関係は進化を続けている [88]．新たに出現した OSS は，急速に発展しており，これに伴い，OSS 開発チームでは，新たな OSS や，OSS の改良が次々と行われている．今後のソフトウェア開発を牽引するチーム形態のひとつとして，OSS 開発の特性について明らかにすることは重要であろう．

2.5.2 OSS 開発の概要

本項では，OSS 開発の概要として，開発手法，主な開発環境，および OSS と OSS 開発に関する用語についてまとめる．

OSS の開発手法

1999 年，Raymond [91] は，OSS において成功を収めた開発方式を 2 つに定義した．ひとつは，Linux の開発方式である「バザール方式」であり，もうひとつは当時の一般的な開発方式であった「伽藍方式」である．

伽藍方式とは，中央集権的に伽藍を組み上げていくような方式であり，少数の高いスキルを持つ開発者が慎重に開発を行う．リリースに対しては，ソフトウェアが完成するまでは β 版すら出さないような厳しい考え方である．伽藍方式では，バグは「あってはならないもの」であるため，バグがない状態でリリースするために，少数の開発者が長期に渡ってバグ取りテストを行い，ソフトウェアを完全なものに仕上げていく [91].

一方，Linux で用いられた開発方式であるバザール方式とは，ソースコードを公開し，興

味を持った開発者が開発を行うため，いろいろな作業やアプローチが渦巻くバザールのようであるというところから，この名前が付けられた．任せられるものは他人に任せ,「早めに頻繁にリリースする」というように，リリースに対する考え方は厳しくない．バザール方式では，バグは「直すもの」である．リリースにより，利用者によってバグ探しが行われる．発見されたバグは，開発者によって修正され，次のリリースに反映される．バザール方式では，バグ探し，バグ修正が世界中の開発者と利用者によって行われるため，リリースのサイクルを非常に短くできる [91]．現在の OSS 開発は，バザール方式が中心となっている．

OSS の主な開発環境

OSS の開発は，主に GitHub [92] や Bitbucket [93] といったインターネット上のソフトウェア開発のプロジェクトホスティングプラットフォームを利用して行われている．プロジェクトホスティングプラットフォームは，作業中のソースコードの管理場所である．プラットフォーム上では，多種多様なソフトウェア開発が行われるため，開発対象ごとに Repository と呼ばれる単位で一元的にデータの構成管理を行っている．OSS 開発では，特定のソフトウェア (またはソフトウェア群) の開発活動をプロジェクトと呼ぶ．企業において一般的に用いられているプロジェクトは，Project Management Body of Knowledge (PMBOK) において，独自のプロダクト，サービス，所産を創造するために実施する有期性のある業務と定義されている [53]．しかしながら，OSS 開発におけるプロジェクトは，明確な活動期限を持たないものが多く，割り当てられた業務でもないため，PMBOK の定義とは全く異なる性質を持つものである．OSS 開発におけるプロジェクトの特色としては，予算を持たない，開発期間の設定がない，開発活動への参加義務がない，などがある．また，個々の OSS 開発のプロジェクトによって，その運営方法も異なる．よって，OSS 開発におけるプロジェクトの定義は困難である．そこで，本研究では，企業におけるプロジェクトの対比として，OSS を開発するプロジェクトのことを OSS 開発プロジェクトと総称する．また，本研究における OSS 開発プロジェクトは，一元的にデータの構成管理を行う Repository 単位での開発活動を指す．OSS 開発は，プラットフォームで OSS 開発プロジェクトを公開することにより，インターネット上で複数の開発者が並行して開発を進めていくことができる．プラットフォームの中でも，GitHub は最も勢いを増している [94].

OSS 開発に関する用語

GutHub を例に，OSS 開発における主な用語を簡潔にまとめる．

GitHub を利用するにはユーザ登録が必要である．対象の OSS 開発プロジェクトの Remote Repository を自身の Local Repository にコピーすることを Clone と言う．対象の OSS 開発プロジェクトの Remote Repository を自身の Remote Repository にコピーすることを Fork と言う．Fork は，何らかの追加機能の実装やバグ改修を実施する場合に使用するものである．そのため，オリジナルの Repository への貢献が前提であり，Fork した場合にはオリジナルの Repository の所有者に通知される．ユーザは，Clone や Fork した自分の Repository で開発を行う．GitHub では，バージョン管理システムである Git [95] を用いている．バージョン管理システムへ変更を登録することを Commit と言う．バージョン管理システムでは，全ての変更履歴を記録しており，Commit と共にメンバの活動が記録されている．1 つの Commit 記録には，追加や修正された対象のファイル，追加行数，削除行数の情報，メンバ名およびタイムスタンプが残されている．これらの記録を Log と言う．自身の開発環境でテストを行い，問題がなければ，オリジナルの Remote Repository へ反映させる申請である Pull Request を行う．Pull Request が受け入れられると，自身の開発内容がオリジナルの Remote Repository へ反映される．課題やエラーなどの課題管理は Issues で行われる．Pull Request が受け入れられたり，Issues において議論に参加したり，Code Review を行ったりするなど，プロジェクトに対して何らかの貢献を行ったユーザは，Contributor と呼ばれる．プロジェクトにより，Wiki [96] などのサービスを利用したり，メーリングリスト (Mailing List) を使ったりして，相互にコミュニケーションをとっているところもある．

また，GitHub ユーザは，Repository や他のユーザに Star を付けることができる．Star とは，Social Networking Service (SNS) の Facebook [1] や Twitter [2] の「いいね！」(Like!) のようなものであり，SNS 同様，Star により OSS そのものを始め，開発活動，開発者の人気度や期待度を見ることができる．

2.5.3 OSS 開発チームにおける役割に関する課題

OSS 開発に関しては，開発量から有能な開発者候補を予測する研究 [38]，OSS 開発プロジェクトにおける役割の変化などに関する研究 [39,40]，OSS 開発プロジェクトのプロセスモデルに関する研究 [41] など，さまざまな研究が行われているが，これらの研究は，個人のスキルや成長に焦点を当てた研究が多くを占めている．

Guimarães らによる OSS 開発コミュニティにおけるライフサイクルモデルの研究では，SourceForge [97] のダウンロード，バグ，フォーラムメッセージ，ソースコードアクティビティ情報などを用いて主成分分析を行い，スタートアップ，成長，成熟，衰退，死といった OSS 開発コミュニティのライフサイクルの段階 (Stage) によってコミュニティの有効性レベルと活動レベルに変化があることを示している [42].

竹田は，OSS 開発チームの役割について次のように述べているが，実際の調査は行っていない.「それぞれのモチベーションはさまざまであるが，基本的には自発的に開発プロジェクトに参加した個人で構成され，そこには明示的な指揮命令の系統はない．このような特徴から，OSS の開発組織はコミュニティとして扱われることが多い．コミュニティの統制は，主にプロジェクト開始者が発揮するリーダーシップと相互信頼とによる.」[88]

以上のように，OSS 開発集団をチームと捉え，チームとしてどのように開発を行っているのかといった，開発チームの開発実態を明らかにした研究は見当たらない．そのため，OSS 開発チーム内における役割についても明らかになっていない．OSS 開発では，有用な OSS では流用や派生開発が多数発生していること，メンバは自由にチームへの参加や離脱ができることから，チームの範囲を定めなければ，リーダやメンバの実態を調査することができない．OSS 開発チームにおける役割を調査するには，まず，OSS 開発チームの範囲を定める必要があることが明らかになった.

2.6 OSS 開発チームにおけるチーム状態の計測

本節では，まず，OSS 開発チームの状態を計測するに当たり，インターネットを通した集団の活動構造を表す研究について概観する．続いて，2.5.3 項で述べた課題を基に，本研究における OSS 開発チームを定義する．

2.6.1 インターネット上の集団における活動構造の指標化に関する研究

本項では，インターネットを通した集団の活動構造を表す研究について概観する．

北山は，29 のメーリングリストにおける投稿数を個人毎に計測し，投稿数の分布をジニ係数 (Gini Coefficient) とローレンツ曲線 (Lorenz Curve) で表した [98]．ジニ係数は，分布の偏りを表す指標であり，その分布をグラフで表したものがローレンツ曲線である．ジニ係数とローレンツ曲線は，経済活動の成果である国全体の所得が各世帯にどのように分配されているのかを調べるときに，最もよく用いられる指標とグラフである [99, 100]．北山がジニ係数を用いたのは，メーリングリストにおける個人毎に集計した投稿数の分布に偏りがあったためである．

Fig. 2.2 に，ローレンツ曲線の例を示し，国の世帯所得の分布を例にジニ係数とローレンツ曲線について説明する．ローレンツ曲線は，世帯を所得の低い順番に並べ，横軸に世帯の累積比をとり，縦軸に所得の累積比をとって，世帯間の所得分布をグラフ化したものである．世帯間の所得格差が存在せず，全ての世帯の所得が同額であるならば，ローレンツ曲線は 45 度線 (Line of perfect equality) と一致する．世帯所得の分布に偏りがある場合には，ローレンツ曲線は下方 (Line of Perfect inequality) に膨らんだ形になる．ジニ係数は，ローレンツ曲線の下方への膨らみ具合を，Fig. 2.2 の (A) の面積と (A) + (B) の面積の比で表したものである．ジニ係数の値は，0 と 1 の間をとるため，ジニ係数の値が 0 に近ければ分布の偏りは小さく，1 に近ければ分布の偏りが大きい．

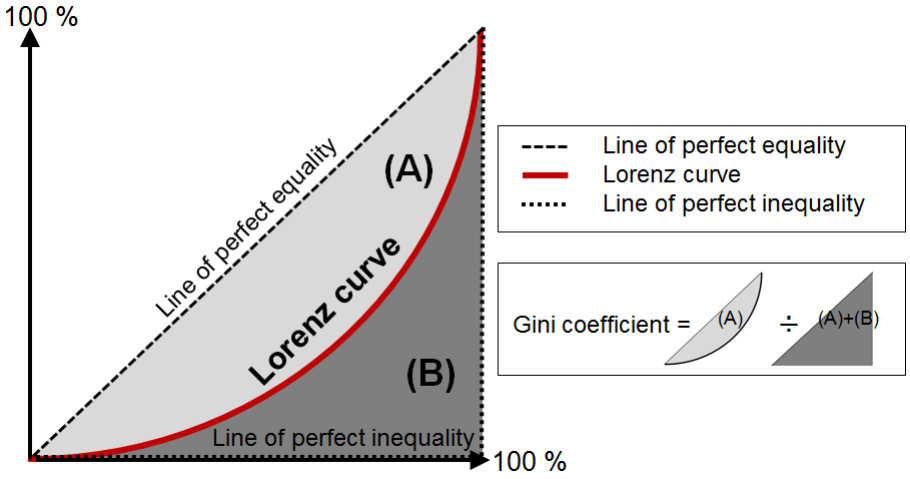

Fig. 2.2: The Gini coefficient and the Lorenz curve

北山は，所得の分配をメーリングリストの発言数の分配としてジニ係数を求め，世帯の累積比の代わりに横軸に投稿者の累積比をとり，所得の累積比の代わりに縦軸に投稿数の累積比をとってローレンツ曲線で示した．北山は，メーリングリストの投稿数の分布をジニ係数を用いて分析した結果，人の繋がりの維持を目的とする集団と情報を共有する集団とではジニ係数の値に差があると述べている [98]．人の繋がりの維持を目的とする集団ではジニ係数の値は小さく，情報共有を目的とした集団では，目的指向性が強くなるとジニ係数の値が大きくなっていることを示した．

北山の研究の他にも，ネットワーク上のデータの分析にジニ係数を用いた研究がある [101]．Dewan の研究では，Web の閲覧行動における特定のサイトに集中する程度をジニ係数で表現し，比較している．このように，ジニ係数とローレンツ曲線は，本来の目的である所得の均等度合いを表す以外にも活用されている．ジニ係数は所得均等性の評価指標として提案された係数であるが，分布の偏りを評価したり，表したりする指標として，さまざまな事項へ応用が可能である．

ジニ係数は，計算が容易であり，データの特性をひとつの値で表すことができる．一方で，分布の形が異なっていてもジニ係数の値は同値となる場合があるため，この点には注意しなければならない．分布の偏りは，ジニ係数によって，その程度を推測することができる [98]．

本研究では，OSS 開発チームのチーム状態を表す指標として，ジニ係数とローレンツ曲

線を応用することとした．

2.6.2 OSS 開発チームの定義

2.5.3 項において，OSS 開発チームにおけるリーダとメンバの関係を探る前段階として，開発チームの実態を探る必要があることが明らかとなった．さらに，参加も離脱も自由な OSS 開発チームの活動実態を探るためには，チームとしての対象期間と対象範囲を定める必要があることも明らかとなった．

まず，OSS 開発チームの何を分析対象とするのかについて検討した．OSS 開発チームに関わるメンバは，数人から数千人規模であり，大きいものでは 1 万人を超えている．そのため，企業の開発チームのように，メンバに対して質問紙調査を実施することは難しい．しかしながら，インターネット上で公開されている OSS 開発プロジェクトであれば，Repository 単位で開発開始時点からの全ての Log データの取得が可能である．そこで，OSS 開発チームの範囲を OSS 開発プロジェクトの Repository 単位とし，開発記録そのものである Log データを分析対象とすることとした．本研究において，OSS 開発活動を指す場合には OSS 開発プロジェクトと記載し，OSS 開発プロジェクトのメンバを指す場合には，OSS 開発チームと記載する．本研究では，Repository を開発チームとして捉え，OSS 開発チームの実態を探ることとした．一般的に，OSS 開発の利用者，開発者，愛好者らの集団を OSS 開発コミュニティと呼ぶが，本研究では，計量の単位を Repository とした．そのため，計量の単位として OSS 開発プロジェクト，OSS 開発チームと記載している場合には，Repository 単位での活動や集団を指す．

次に，チームの分析対象期間について検討した．Log データには，開発開始からデータ取得時点までの全ての情報が記録されている．OSS 開発チームへの自由な参加や離脱は，OSS 開発チーム特有の特徴であり，OSS 開発チームの活動実態を探るためには特定の期間を定めずに全ての期間を対象とするのが望ましいと考えた．そこで，OSS 開発チームの対象期間は，開発開始時点からデータ取得時点までと設定した．

本研究では，本項で定めた範囲と期間を OSS 開発チームとして，Log データを用いて，チームの実態に関する詳細な分析を行う．さらに，この分析結果を基に，開発チームの状態の指標化を試みる．これらの分析を通して，OSS 開発チームにおけるリーダとメンバの関係について探ることとした．

2.7 結言

本節では，これまでに挙げた関連研究の課題を踏まえて，本研究のテーマを示す．

ソフトウェア開発チームは，チーム形態の違いから，職業としての開発チームと，趣味の開発チームの 2 つに大別できる．本研究では，それぞれの代表的な開発チームとして，企業における開発チームと OSS 開発チームを研究対象とし，次の 3 つを研究テーマとして設定した．

- 研究テーマ 1: 企業の開発チームにおけるチーム状態を計測する質問紙分析手法を探る
- 研究テーマ 2: OSS 開発チームにおけるメンバと開発量の関係を探る
- 研究テーマ 3: OSS 開発チームにおけるチームの状態の指標化を試行する

研究テーマ 1 の「企業の開発チームにおけるチーム状態を計測する質問紙分析手法を探る」では，開発チームのリーダが自身のチームの状態を知ることができるツールとして，松尾谷の研究 [10] が示した手法を基に検証と探索を行う．

チームの状態を計測できるツールがあれば，リーダは，チームの問題をタイムリーに改善したり，次の開発チームの運営に活かしたりすることができるようになる．研究テーマ 1 では，高い判別分析結果を示した松尾谷の研究 [10] における企業の開発チームのリーダを対象に実施した質問紙調査データを用い，不足している判別分析の検証を行う．また，リーダが使うツールは簡便であることが望ましいため，複数の統計分析手法を用いて比較し，最適な分析方法を探索する．

研究テーマ 2 の「OSS 開発チームにおけるメンバと開発量の関係を探る」では，OSS 開発集団をチームと捉え，チームとしてどのように開発を行っているのか，明らかになっていない OSS 開発チームの開発実態をメンバと開発量の関係から分析する．研究テーマ 2 では，活性化している OSS 開発プロジェクトの条件を定め，膨大な OSS 開発プロジェクトの中から分析対象とする開発チームを抽出する．OSS 開発チームは Repository 単位と定め，開発開始時点からデータ取得日までを分析対象期間として Log データを取得し，分析対象

とする．分析対象の開発チーム単位で取得した Log データから，成果物の著作者の詳細分析を行い，OSS 開発の実態を明らかにする．

研究テーマ 3 の「OSS 開発チームにおけるチームの状態の指標化を試行する」では，研究テーマ 2 の結果を踏まえて，OSS 開発チームの状態を測定する手法として，活動状況の指標化を試みる．研究テーマ 3 では，研究テーマ 2 で分析対象とした開発チームの Log データを用いる．研究テーマ 2 で明らかにした OSS 開発チームの活動実態を基に，OSS 開発チーム構成員の活動量の分布の偏りから，OSS 開発チームの状態をジニ係数とローレンツ曲線を用いて指標化できるかを検証する．

これらの研究テーマの関連を Fig. 2.3 に示す．

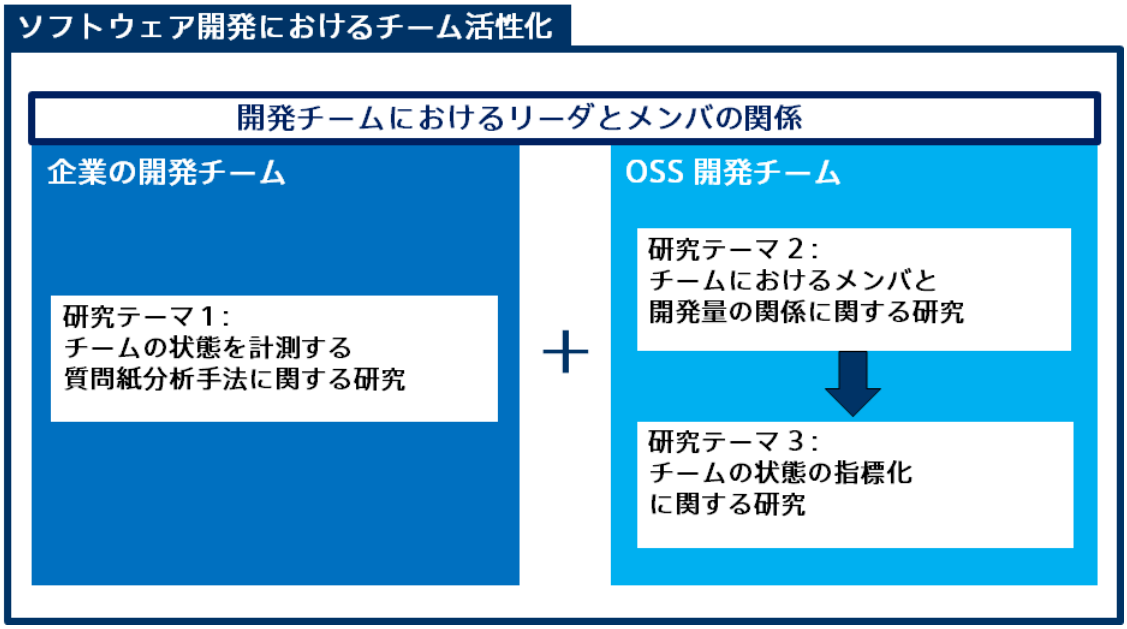

Fig. 2.3: Relevance of Each Study Subject

本研究は，ソフトウェア開発チームの活性化に影響を及ぼす要因を明らかにすることを目的に，活性化しているチームの分析を行う．先行研究の調査より，開発チームにおけるリーダとメンバの関係に焦点を当てることとした．Fig. 2.3 で示した通り，ソフトウェア開発チームは，職業としての開発チームと，趣味の開発チーム 2 つに分類し，それぞれを対

象として研究を行う．職業としての開発チームとしては，企業の開発チームを対象とする．趣味の開発チームとしては，OSS 開発チームを対象とする．企業の開発チームに対しては，開発チームの状態を計測する質問紙分析手法に関する研究を行う．これを研究テーマ 1 とする．OSS 開発チームに対しては，開発チームの実態が明らかになっていないため，まず，開発チームにおけるメンバと開発量の関係に関する研究を行う．これを研究テーマ 2 とする．研究テーマ 2 で明らかとなった開発チームの実態を基に，OSS 開発チームの状態の指標化に関する研究を行う．これを研究テーマ 3 とする．最後に，3 つの研究テーマの結果を基に，ソフトウェア開発チームにおけるリーダとメンバの関係について考察する．

第3章　企業の開発チームにおけるチームの状態を計測する質問紙分析手法に関する研究

3.1　緒言

本章では，企業の開発チームのリーダが，自身のチームの状態を知ることができるツールの検証と最適な分析手法を探索する．チームの状態を測定できるツールがあれば，リーダは，チームの問題をタイムリーに改善したり，次の開発チームの運営に活かしたりすることができるようになる．そこで，プロジェクトの成否判別に関して高い判別分析結果を示した松尾谷の研究 [10] における企業の開発チームのリーダを対象に実施した質問紙調査データを用い，この研究において不足している判別分析の検証を行う．また，企業の開発チームのリーダが使うツールとしては，調査および分析手法は簡便であることが望ましい．そこで，複数の統計分析手法を用いて比較し，最適な分析方法を探る．

3.2 開発チームの特性のモデル化

ソフトウェア開発の生産性や品質を目的変数とし，生産性や品質に対して影響を及ぼす要因を説明変数としてモデル化する研究がある．モデル化研究においては，ソフトウェアの QCD の関係をソフトウェア開発の見積もり式としてまとめた COCOMO が知られている [45,48,49,102]．ソフトウェアの生産性に影響を及ぼす要因群として，開発に用いるツールや技法といった技術的な特性と，これらの技術を使う人的資源である開発チームメンバの特性が考えられる．技術的な特性としては，セキュリティの要求レベル，メモリ制約，デバッグ環境の有無などがある．人的資源の特性としては，メンバのチームとしての経験，モチベーションなどが考えられる．人的資源の特性は，ソフトウェア開発に大きな影響力があるにも関わらず，技術的な特性と比べ，属人的であり，客観的な計量化が難しいという課題がある．

人的資源の特性のモデル化に関しては，人的資源の特性の計測をするために選定した質問項目と，モデルとして用いる分析手法との組み合わせにより，さまざまな研究が行われている [10,29,64]．松尾谷の研究 [10] では，質問紙調査結果に対して主成分分析を行い，抽出した主成分を用いてプロジェクトの成否を判別するモデルを示した．企業の開発チームのリーダが実務の中で利用できるツールとしては，統計処理で主成分を抽出することなく，調査結果そのものを用いて判別することができるモデルが良いと考えた．そこで，調査結果そのものに対して機械学習の枠組みを用いたモデルを試行することとした．本章では，松尾谷の研究 [10] で計測されたデータを用いて，機械学習の枠組みを用いたモデルを試行し，松尾谷の研究で用いられた主成分を用いた統計モデル [10] との比較および検証を行う．開発チームメンバのコミュニケーションの程度や仲間意識など人的資源の特性に焦点を当て，物理的な計量が困難である開発チームの特性について，質問紙を用いて計測したデータを説明変数とし，プロジェクトの結果を目的変数として，その関係をモデル化する最適な手法を探る．

3.3 質問紙と調査データの概要

本節では，本章で用いた質問紙と調査データの概要について述べる．松尾谷の研究 [10] では，まず，20 件のプロジェクト群に対して質問紙調査を実施した．その後，質問紙の評価を行い，質問紙を改良し，改良した質問紙で 55 件のプロジェクト群に対して質問紙調査を実施している．本章では，これら 2 つの調査データを用いて分析を行う．ここでは，プレ調査として実施した 20 件のプロジェクト群に対する調査データを調査データ A，本調査として実施した 55 件のプロジェクト群に対する調査データを調査データ B と表す．

本節では，まず，3.3.1 項で，松尾谷の研究 [10] を基に，質問紙の作成過程についてまとめる．次に，3.3.2 項にて，質問紙の構成を示す．最後に，3.3.3 項で，調査データの概要について述べる．

3.3.1 質問紙の作成過程

先行研究 [10] では，チーム活性化 [1] の計測方法として，心理尺度の手法を用いてチーム活性化の要素を尺度化した．心理尺度の作成手法として用いた質問紙法では，計測対象を構造的に捉え，下位尺度と呼ばれる要素に分解し，下位尺度毎に質問項目を作成して計測を行う [103,104]．プロジェクト・マネジメントにおける先行研究では，メンバのモチベーションを「職務満足感」,「仕事意欲」,「精神健康・ストレス反応」の 3 つの下位尺度に分けて質問紙を作成して調査を実施し，下位尺度毎に主成分を抽出して尺度化を行っている [19,20,27]．本章で用いた質問紙は，この質問紙法を用いて作成されている．

先行研究 [10] では，次の手順でチーム活性化の要素の尺度化と評価を実施している．

- ステップ 1 : チーム活性化の構成モデル [2] を作成し，下位尺度を想定する
- ステップ 2 : 下位尺度に対して質問項目を開発し，質問紙 A を完成させる
- ステップ 3 : 質問紙 A を用いて調査を実施する
- ステップ 4 : 質問紙 A の回答結果から，下位尺度毎に主成分分析を行い，チーム活性化要素をさらに複数の因子によって表す

[1] 先行研究 [10] では,「現場力」と表現している．

[2] 統計的に確認された下位尺度群は「構成概念」と呼ばれる．先行研究 [10] では，統計的に確認された構成概念と作業用の構成概念を区別するため，作業用の構成概念を構成モデルと呼んでいる．

- ステップ 5：正規化 (平均 0，分散 1) した複数の因子から，プロジェクト成否を導く判別分析を行う
- ステップ 6：分析を通して，下位尺度と質問紙 A の評価を行い，質問紙を改善する
- ステップ 7：改良した質問紙 B を用いて調査を実施する

上記の手順のステップ 3 で実施した調査が調査データ A であり，ステップ 7 で実施した調査が調査データ B である．

3.3.2 質問紙の構成

両調査で用いた質問紙について説明する．調査データ A で用いた質問紙 A は，顧客との関係性 (Relationship with Customers)，仲間意識 (Collegiality Consciousness)，役割意識 (Role Consciousness)，規範意識 (Norm Consciousness)，の 4 つの構成概念，合計 20 項目から構成される．4 つの構成概念の意味を次に示す [10].

仲間意識 ：チームメンバは目的を共有し，目的に向かって行動しているか

役割意識 ：チームの中で自分の役割を認識し行動したか，あるいは，他者は自分の役割を認識し行動しているか

規範意識 ：チームで共有しているチーム文化があるかどうか，また，その文化を受容しているか

顧客との関係性 ：顧客による指示の強さ，顧客との利害関係はどうか

質問紙 A の構成を Table 3.1 に示す．プロジェクトの時期に関する質問項目以外の選択肢は,「1：強い否定，2：否定，3：肯定，4：強い肯定」の 4 件法である．

Table 3.1: Questionnaire Composition: Case A (n=20)

Categories of Question Items		Number of Question Items
Constituent Concepts	Relationship with Customers	7
	Collegiality Consciousness	5
	Role Consciousness	5
	Norm Consciousness	3
Attribute Information	Timing of the Project	1
	Result of the Project	1
Total		22

調査データ B で用いた質問紙 B は，調査データ A の結果を基に改善された質問紙であり，仲間意識，役割意識，規範意識，成果意識 (Outcome Consciousness)，環境意識 (Environmental Consciousness) の 5 つの構成概念，合計 24 項目から構成される．5 つの構成概念の意味を次に示す [10]．仲間意識，役割意識，規範意識については，調査データ A で用いた質問紙 A と同じだが，顧客との関係性については，環境意識と成果意識へと構造が見直されている．

仲間意識 ：チームメンバは目的を共有し，目的に向かって行動しているか

役割意識 ：チームの中で自分の役割を認識し行動したか，あるいは，他者は自分の役割を認識し行動しているか

規範意識 ：チームで共有しているチーム文化があるかどうか，また，その文化を受容しているか

環境意識 ：チームが外部から受ける影響はどうか

成果意識 ：メンバのスキルやパフォーマンスはどうか

質問紙 B の構成を Table 3.2 に示す．プロジェクトの時期，役割，規模に関する質問項目以外の選択肢は，「1：否定，2：どちらかと言えば否定，3：判断できない，4：どちらかと言えば肯定，5：肯定」の 5 件法である．

Table 3.2: Questionnaire Composition: Case B (n=55)

Categories of Question Items		Number of Question Items
Constituent Concepts	Collegiality Consciousness	4
	Role Consciousness	7
	Norm Consciousness	5
	Outcome Consciousness	4
	Environmental Consciousness	4
Attribute Information	Timing of the Project	1
	Result of the Project	1
	Personal Role in the Project	1
	Project Size	1
Total		28

両質問紙ともに，回答者の負担軽減のため A4 用紙両面 1 枚にまとめ，回答は匿名で記入する方式となっている．いずれの質問紙にも,「そのプロジェクトは成功しましたか」というプロジェクトの成否 (Result of the Project) を問う質問が含まれている．

3.3.3 調査データの概要

まず，調査データ A のデータ群について説明する．調査データ A は，プロジェクトマネジメント系の研究会メンバで実務経験のある人を対象に実施された．各自が経験したプロジェクトに関する状況メモを作成し，そのチーム状況について詳しい発表を行った上で，質問紙調査が実施されている．調査対象は数社であり，収集されたプロジェクト数は，20 プロジェクトであった．各プロジェクトに関する正確な規模データは収集されていないが，中規模以上のプロジェクトであったと報告されている．

次に，調査データ B のデータ群について説明する．調査データ B は，特定非営利活動法人日本プロジェクトマネジメント協会が主催する PM シンポジウム 2013 に集まったプロジェクトマネジャーを対象に実施された．調査対象は，すべて異なるプロジェクトであった．この調査により収集されたプロジェクト数は，55 プロジェクトであった．

本章では，調査データ A と調査データ B を用い，目的変数をプロジェクトの結果に対する見解 (調査データ A では 4 段階，調査データ B では 5 段階) とし，説明変数の分類数を変えて比較する．

3.4 判別分析によるモデル評価

本章では，目的変数を説明するモデルとして判別式を用いた．本節では，判別式の作成方法とその評価方法についての概要と選択した方法について説明する．

3.4.1 判別分析

判別分析とは，対象とする個体の特性から，その個体がどの群に属するかを分類する手法である．一般的には，属する群が既に分かっているデータに基づいて判別式を作成し，まだ属する群が分からない個体の群を分類する．判別分析のモデルはいくつかあり，どの分析方法が優れているのかは対象によって変化する．そのため，対象とするデータを用いて実際にモデルを作り，評価を行って，どの判別分析モデルをどのように用いるかを選択することになる．代表的な判別分析の手法には，統計的な手法と機械学習の手法がある．次に，各々の代表的な手法について示す．

(A) 統計的な手法

代表的な統計的な手法としては,「線形判別分析」,「二次判別分析」,「正準判別分析」,「決定木」などがある．線形判別分析は，各群が正規分布していること，等分散であることが制約条件となっている．一般的には，属する群が分かっているデータに基づいてマハラノビス距離を計算し，対象データに対して計算したマハラノビス距離が最小となるように一次関数の直線を引き 2 群に分類する．各群が等分散でない場合には，二次判別分析を用いるのが良いとされている．二次判別分析では，各群で異なる共分散行列を用いてマハラノビス距離を計算し，マハラノビス距離が最小となるように二次関数の曲線を引き 2 群に分類する．線形判別分析と二次判別分析の目的変数は 2 群であるが，目的変数が 3 群以上となる多群に分類する場合には，正準判別分析を用いる．正準判別分析では，複数の判別式を作成して目的変数で指定した群に分類する．多数の変数の中から共通の因子を集約する主成分分析に近い考え方で，多変量の次元を集約し，判別を行う．決定木は，樹木状の階層構造を用いて対象データを分類していくモデルである．変数ごとに領域を分割し，2 分木に枝分かれさせて分類していく．

判別分析では，目的変数や説明変数として利用可能なデータの種類を量的変数に制約しているモデルがあるが，ダミー変数を用いたり，数量化したりすることで質的変数でも対応することができる．

本章では，目的変数が 3 群以上であることから，比較する先行研究 [10] でも用いられていた正準判別分析を用いることとした．

(B) 機械学習の手法

統計的な手法として代表的な手法として,「教師あり学習データを用いる手法」,「教師なし学習データを用いる手法」がある．機械学習では，既知の特徴を持つサンプルデータを教師データ (Training Data) として入力し，入力されたデータから有用な規則や判断基準などを抽出し，アルゴリズムを強化していく．そのアルゴリズムを用いてテストデータと呼ばれる未知のデータの分類を行う．テストデータの分類は，分類の分かっている教師データを基に，データの属性値に基づいて実施する．この手法を，教師あり学習 (Supervised learning) と呼ぶ．一方，教師なし学習 (Unsupervised learning) は，その名の通り分類が分かっていないデータを基に行う学習で，たとえば，クラスター分析がこれにあたる．

本章の研究では，分析データに含まれるプロジェクトの成果に対する見解を目的変数として判別を行うため，目的変数の分類は既に分かっているデータである．本章では，機械学習として，教師あり学習データを用いて質問項目の回答データを集約せずに用いて分析を行う．

3.4.2 モデルの評価方法

モデルの評価方法としては，教師データから作成したモデルを使って，元のデータに対する判別結果の判別率を用いる方法が考えられる．この方法だと，モデルの説明変数の数が多くなれば，当該データに対する情報が増えるため，判別率は上がる．モデルのロバスト性から評価すると，説明変数の数は少ない方が良い．つまり，同じ判別率だった場合には，利用した説明変数の数が少ない方がモデルとして良いと判断できる．

さらに，モデルとしての性能を評価するのであれば，学習に用いた教師データにおける判別率より，当該データの外側，外挿においても判別率を高めることが求められる．その方法として，交差検証がある．交差検証とは，検証データとしてモデル作成に用いなかったデータに対して検証を行うことである．

本章におけるモデルは，判別率，説明変数の数，交差検証における判別率から評価する．

3.5 比較モデル

本節では，先行研究 [10] で用いた質問紙調査の調査データを用いて，比較するモデルの検討を行う．3.3.2 項と 3.3.3 項 で示した調査データ A と調査データ B に対して比較を行った各モデルについて，3.5.1 項と 3.5.2 項に示す．

3.5.1 統計モデル

統計モデルの分析では，質的データを次の 2 種類の量的データに変換し，説明変数として正準判別分析を実施する．一つ目は，回答データの構成概念単位で因子分析を実施し，その結果として抽出された因子の因子得点を用いる．二つ目は，同様に構成概念単位で主成分分析を実施し，その結果として抽出された主成分の主成分得点を用いる．Table 3.3 に，各調査データにおいて変数として用いる因子数および主成分数を示す．

分析には，R 3.3.0 を用いた．因子分析および主成分分析には psych パッケージを用い，正準判別分析には MASS パッケージを用いた．

Table 3.3: Independent Variables Used in Each Model

Model	Case	Independent Variables	
		Variable	Number of Variables
Statistical Model	A (n=20)	Factors[a]	8
		Principal Components[a]	8
	B (n=55)	Factors[a]	8
		Principal Components[a]	8
Machine Learning Model	A (n=20)	Answer values of the question (Constituent Concept)	20
	B (n=55)	Answer values of the question (Constituent Concept)	24
		Answer values of the question (Constituent Concept + Attribute Information)	27

[a] Extraction by constituent concept unit.

3.5.2 機械学習モデル

機械学習モデルでは，説明変数として，質的変数である全質問項目の値を用いる．調査データ B については，Table 3.2 で示す属性も含めた質問項目でも確認を行う．Table 3.3 に，

各調査データにおいて変数として用いる質問項目数を示す.

本章の分析対象は，両調査データともデータ数が少ないため，教師データとテストデータに分割せず，調査データ全てを教師データとして用いて学習を行う.

分析には，R 3.3.0 を用い，正準判別分析には MASS パッケージを用いた.

3.6 モデルの評価

本節では，教師ありデータにおける検証結果 (ここでは自己検証と呼ぶ) と交差検証の結果を示す.

3.6.1 自己検証

Table 3.3 で示した各モデルに対して行った自己検証の結果を正誤表として示す.

まず，統計モデルにおける自己検証結果の正誤表を，Table 3.4 から Table 3.7 に示す．次に，機械学習モデルにおける自己検証結果の正誤表を Table 3.8 から Table 3.10 に示す．最後に，Table 3.11 において，各モデルにおける判別率を示す.

統計モデルにおける自己検証結果の正誤表[3]

Table 3.4: Discrimination Result Using Factors: A (n=20)

		Discrimination Number				Total
	Result	1	2	3	4	
Actual Survey Number	% 1	**100**	0	0	0	100
	% 2	0	**67**	33	0	100
	% 3	0	0	**100**	0	100
	% 4	0	14	14	**71**	100

Table 3.5: Discrimination Result Using Principal Component Factors: A (n=20)

		Discrimination Number				Total
	Result	1	2	3	4	
Actual Survey Number	% 1	**100**	0	0	0	100
	% 2	0	**100**	0	0	100
	% 3	0	0	**100**	0	100
	% 4	0	0	14	**86**	100

[3]正誤表で示す判別率は，項目ごとに小数点以下第 1 位で四捨五入しているため，合計値が 100 % にならない場合がある.

Table 3.6: Discrimination Result Using Factors: B (n=55)

		Discrimination Number					Total
	Result	1	2	3	4	5	
Actual Survey Number	% 1	**50**	0	0	33	17	100
	% 2	0	**0**	0	50	50	100
	% 3	0	0	**38**	50	13	100
	% 4	4	0	17	**67**	13	100
	% 5	0	0	0	27	**73**	100

Table 3.7: Discrimination Result Using Principal Component Factors: B (n=55)

		Discrimination Number					Total
	Result	1	2	3	4	5	
Actual Survey Number	% 1	**67**	0	0	33	0	100
	% 2	0	**0**	0	50	50	100
	% 3	0	0	**63**	38	0	100
	% 4	4	0	17	**75**	4	100
	% 5	0	0	0	18	**82**	100

機械学習モデルにおける自己検証結果の正誤表[4]

Table 3.8: Discrimination Result Using Question Items: A (n=20)

		Discrimination Number				Total
	Result	1	2	3	4	
Actual Survey Number	% 1	**100**	0	0	0	100
	% 2	0	**100**	0	0	100
	% 3	0	0	**100**	0	100
	% 4	0	0	0	**100**	100

[4]正誤表で示す判別率は，項目ごとに小数点以下第 1 位で四捨五入しているため，合計値が 100 % にならない場合がある．

Table 3.9: Discrimination Result Using Question Items: B (n=55)

		Discrimination Number					Total
	Result	1	2	3	4	5	
Actual Survey Number	% 1	**83**	0	0	17	0	100
	% 2	0	**50**	17	17	17	100
	% 3	0	0	**88**	13	0	100
	% 4	0	0	8	**88**	4	100
	% 5	0	0	0	0	**100**	100

Table 3.10: Discrimination Result Using All Question Items: B (n=55)

		Discrimination Number					Total
	Result	1	2	3	4	5	
Actual Survey Number	% 1	**100**	0	0	0	0	100
	% 2	0	**83**	0	17	0	100
	% 3	0	0	**100**	0	0	100
	% 4	4	0	8	**83**	4	100
	% 5	0	0	0	0	**100**	100

自己検証における判別率

Table 3.11: Discrimination Rate of Each Model

Model	Case	Independent Variables		Correct Rate (%)[a]	Error Rate (%)[a]
		Variable	Number of Variables		
Statistical Model	A (n=20)	Factors[b]	8	85.00	15.00
		Principal Components[b]	8	95.00	5.00
	B (n=55)	Factors[b]	8	54.55	45.45
		Principal Components[b]	8	65.45	34.55
Machine Learning Model	A (n=20)	Answer values of the question (Constituent Concept)	20	100.00	0.00
	B (n=55)	Answer values of the question (Constituent Concept)	24	85.45	14.55
		Answer values of the question (Constituent Concept + Attribute Information)	27	90.91	9.09

[a] The discrimination rates are rounded off to two decimal places.
[b] Extraction by constituent concept unit.

3.6.2 交差検証

交差検証の手法として，本章では Leave-One-Out-Cross-Validation (LOOCV) を用いた．LOOCV は，対象となる調査データから 1 つのデータを抜き出してテストデータとし，残りのデータを教師データとする．そして，全ての調査データが 1 回ずつテストデータとなるように検証を繰り返す手法である．そのため，LOOCV は，一個抜き交差検証とも呼ばれる．LOOCV を選定した理由は，両調査データともデータの数が少ないため，ランダムに 2 分割して交差検証を行った場合には，個別のデータの影響が大きく，組み合わせによるばらつきにより，検証結果が安定しないと考えたためである．

Table 3.3 で示した各モデルに対して，自己検証の妥当性の検証として行った交差検証の結果を正誤表として示す．

まず，統計モデルにおける交差検証結果の正誤表を，Table 3.12 から Table 3.15 に示す．次に，機械学習モデルにおける交差検証結果の正誤表を Table 3.16 から Table 3.18 に示す．最後に，Table 3.19 において，各モデルにおける交差検証における判別率を示す．

統計モデルにおける交差検証 (LOOCV) 結果の正誤表[5]

Table 3.12: Leave-One-Out-Cross-Validation Result Using Factors: A (n=20)

		Discrimination Number				Total
	Result	1	2	3	4	
Actual Survey Number	% 1	**0**	50	50	0	100
	% 2	0	**0**	67	33	100
	% 3	13	13	**63**	13	100
	% 4	14	14	29	**43**	100

[5]正誤表で示す判別率は，項目ごとに小数点以下第 1 位で四捨五入しているため，合計値が 100 % にならない場合がある．

Table 3.13: Leave-One-Out-Cross-Validation Result Using Principal Component Factors: A (n=20)

		Discrimination Number				Total
	Result	1	2	3	4	
Actual Survey Number	% 1	**0**	50	0	50	100
	% 2	0	**33**	67	0	100
	% 3	13	13	**50**	25	100
	% 4	14	0	29	**57**	100

Table 3.14: Leave-One-Out-Cross-Validation Result Using Factors: B (n=55)

		Discrimination Number					Total
	Result	1	2	3	4	5	
Actual Survey Number	% 1	**33**	0	0	50	17	100
	% 2	17	**0**	0	33	50	100
	% 3	0	0	**13**	75	13	100
	% 4	8	4	21	**50**	17	100
	% 5	9	9	0	55	**27**	100

Table 3.15: Leave-One-Out-Cross-Validation Result Using Principal Component Factors: B (n=55)

		Discrimination Number					Total
	Result	1	2	3	4	5	
Actual Survey Number	% 1	**17**	0	0	83	0	100
	% 2	17	**0**	0	33	50	100
	% 3	0	0	**25**	63	13	100
	% 4	4	4	21	**54**	17	100
	% 5	0	27	0	18	**55**	100

機械学習モデルにおける交差検証 **(LOOCV)** 結果の正誤表[6]

Table 3.16: Leave-One-Out-Cross-Validation Result Using Question Items: A (n=20)

		Discrimination Number				Total
	Result	1	2	3	4	
Actual Survey Number	% 1	**0**	0	50	50	100
	% 2	33	**33**	0	33	100
	% 3	13	0	**25**	63	100
	% 4	14	29	29	**29**	100

Table 3.17: Leave-One-Out-Cross-Validation Result Using Question Items: B (n=55)

		Discrimination Number					Total
	Result	1	2	3	4	5	
Actual Survey Number	% 1	**17**	17	0	50	17	100
	% 2	17	**17**	17	33	17	100
	% 3	13	38	**0**	38	13	100
	% 4	13	8	25	**42**	13	100
	% 5	0	36	0	18	**45**	100

Table 3.18: Leave-One-Out-Cross-Validation Result Using All Question Items: B (n=55)

		Discrimination Number					Total
	Result	1	2	3	4	5	
Actual Survey Number	% 1	**17**	17	0	50	17	100
	% 2	17	**0**	17	33	33	100
	% 3	0	38	**0**	50	13	100
	% 4	21	17	25	**25**	13	100
	% 5	0	18	0	18	**64**	100

[6]正誤表で示す判別率は，項目ごとに小数点以下第 1 位で四捨五入しているため，合計値が 100 % にならない場合がある．

交差検証 **(LOOCV)** における判別率

Table 3.19: Discrimination Rate in Leave-One-Out-Cross-Validation of Each Model

Model	Case	Independent Variables		Correct Rate (%)[a]	Error Rate (%)[a]
		Variable	Number of Variables		
Statistical Model	A (n=20)	Factors[b]	8	40.00	60.00
		Principal Components[b]	8	45.00	55.00
	B (n=55)	Factors[b]	8	32.73	67.27
		Principal Components[b]	8	40.00	60.00
Machine Learning Model	A (n=20)	Answer values of the question (Constituent Concept)	20	25.00	75.00
	B (n=55)	Answer values of the question (Constituent Concept)	24	30.91	69.09
		Answer values of the question (Constituent Concept + Attribute Information)	27	25.45	74.55

[a] The discrimination rates are rounded off to two decimal places.
[b] Extraction by constituent concept unit.

3.7 結言

本章では，開発チームのリーダが，自身のチームの状態を知ることができるツールとして最適な手法を探索するため，松尾谷の研究 [10] で用いられた質問紙調査データを用い，複数の分析方法を試行した．分析方法は，統計モデルと機械学習モデルを用いた．統計モデルでは，説明変数を少数の因子や主成分に集約して用いた．機械学習モデルでは，説明変数を集約せずに用いた．その上で，両モデルの判別率を比較した．

質問紙調査の分析手法の簡便さにおいては，分析過程で因子分析や主成分分析などの統計処理を行う必要がない機械学習モデルは，質問紙調査結果の値そのものを用いて判別ができるため，統計モデルと比較すると簡便であり，開発チームにおいて利用することができると考える．

続いて，松尾谷の研究 [10] で行われていなかった判別分析の検証を行った．自己検証における機械学習の判別率は，Table 3.11 に示す通り，統計モデルとほぼ同等かそれ以上の高い判別率であった．一方で，妥当性検証として行った LOOCV による交差検証においては，Table 3.19 に示す通り，統計モデル，機械学習モデル共に判別率は著しく低下した．これは，用いたデータの範囲外 (外挿) となるデータに対する判別では，判別性能が低下することを示している．つまり，用いたデータにおける目的変数と説明変数の関係は，強い共通性を持っていないと推測できる．

この傾向は,「プロジェクト結果の見解は，人的属性だけでは決まらない」というモデル選択以前の問題であることを示している．しかし，影響があることも示しており，その影響の程度を明らかにすることが，今後の課題である．

本章では，物理的な計量が困難である開発チームの特性について，質問紙調査のコミュニケーションの程度や仲間意識などのデータを説明変数とし，プロジェクトの結果を目的変数として，その関係のモデル化を試みた．開発チームのリーダが，自身のチームの状態を知るためのツールとして最適な分析方法は，機械学習モデルであると考えられる．しかし，実用化するためには，外挿における判別率の向上が求められる．本章で試行した評価比較により，次の課題は，学習のための教師付きデータの蓄積，用いる変数の数や変数の加工なども含めた説明変数の改良であることが明らかとなった．

第4章　OSS 開発チームにおけるメンバと開発量の関係に関する研究

4.1　緒言

OSS 開発チームにおけるリーダとメンバの役割を探るためには，その前段階として，OSS 開発チームがどのように開発を行っているのか，その実態を明らかにする必要がある．そこで，本章では，メンバと開発量との関係の観点から， OSS 開発チームの実態について調査する．ここで言う OSS 開発チームとは，第 2 章で定義したものである．活性化している OSS 開発プロジェクトの条件は，長期に渡って開発が継続していること，成果物である OSS が評価されていること，参加メンバが多いこと，と定義する．分析対象は，この定義に従って抽出した OSS 開発プロジェクトの開発記録である Log データである．

4.2 OSS 開発チームの実態調査における分析ステップ

商用ソフトウェア (以下，ES: Enterprise Software) 開発では，ソフトウェア・プロダクトを商品と位置付け，その権利を著作権などによって保護している．ES 開発におけるコストモデルとはプロダクトの量的な開発効率を意味し，ソフトウェア工学では ES 開発における量的開発効率の改善が重要なテーマである．

一方，OSS 開発では，ソフトウェア・プロダクトの使用 (Use) は自由であり，改変や再頒布などの利用 (Exploit) もオープンソース・ライセンスに従えば自由に行うことができる．これにより，再利用が積極的に行われ，量的開発効率は OSS 開発において重要なテーマではないと扱われている．OSS 開発においては，コストモデルを OSS コミュニティからサービス提供者を経てサービス利用者に至るモデルと定義し，そのコストを最適化する研究が行われている [105].

このように，OSS 開発において量的開発効率は重要視されていないことから，OSS の量的開発効率に関する研究は見当たらない．しかしながら，ES 開発においても OSS の利用割合は増加傾向の一途にあり，OSS が社会にもたらす影響は無視できない状況にある．それゆえ，OSS 開発における状況を定量的に把握しておくべきである．そこで，本研究では，OSS 開発におけるメンバと開発量との関係から開発チームの実態を探ることを目的とした．具体的には，OSS 開発においてもメンバと開発量との間に関係が存在すると仮定し，ES 開発におけるべき式を用いたコストモデルを基に分析を行い，OSS 開発の特徴分析を試みた．分析には，言語や対象領域に制限を持たず，かつ実績のあるコストモデルである Constructive Cost Model (COCOMO) [48, 102] を用いた．COCOMO の適用に当たり，ES 開発の工数に相当する概念を OSS 開発に適用するため，回帰的に得られたべき式の係数に着目した計測方法を探った．

コストモデルは，開発結果のデータに対する統計手法である．本章で用いた OSS 開発のデータは，GitHub [92] から取得した．統計分析において，分析の精度を高めるためには，それ相応のデータ数を確保する必要がある．そこで，分析対象として，2012 年の 1 年間に GitHub に登録され，データ取得日である 2017 年 8 月 11 日まで継続的に開発が行われている大規模な 50 件の OSS 開発プロジェクトを抽出した．抽出した 50 件の OSS 開発プロジェクトにおける，成果物を格納している Git Repository の総量は 20, 037 MB，Git Repository に登録を行う手続きである Commit の総数は 226 万件，総開発者数は 36, 126 名であった．

本章では，次の 3 つのステップで分析を行った．

- ステップ 1 : OSS 開発においても，COCOMO における量的開発効率の関係式，すなわち工数と成果物の関係を表すモデル式 (工数=$a *$ 開発規模$^b +$ コスト因子) が成り立つことを確認する
- ステップ 2 : COCOMO のモデル式の係数から，OSS 開発特有の特性を探る
- ステップ 3 : OSS 開発特有の特性を生む原因を探る

本章にて得られた各ステップの分析結果の概観を以下に示す.

- ステップ 1 の結果 : OSS 開発においても工数 (Person-Months) に対応する計測方法を工夫することにより，COCOMO のモデル式が成り立つことが確認できた
- ステップ 2 の結果 : OSS 開発のべき乗係数は 1 より小さく，開発規模が増加すると量的開発効率が上昇することが確認された
- ステップ 3 の結果 : べき乗係数の差が生じる原因として，(a) OSS 開発の特徴である自由な使用による大規模な流用の影響，(b) 貢献度の小さい多数の開発者の影響，の 2 つを仮定して分析を行ったが，いずれも OSS 開発の量的開発効率への影響は認められず，OSS 開発特有の特性が大規模な流用や貢献度の小さい多数の開発者の存在によるものではないことが明らかになった

4.3 コストモデルと計測方法

本節では，先行研究として，量的開発効率 (以下，開発効率) をべき式で示す COCOMO のコストモデルについてまとめ，COCOMO のコストモデルを OSS 開発に適用するための方法について示す．

4.3.1 コストモデルの目的と式

ソフトウェア開発の主な資源は人的資源である．人的資源の量 (E:$Effort$) と成果物の量 ($KLOC$:$KiloLinesOfCode$) の関係を関数 f で表すと，式 (4.1) と考えられる．

$$E = f(KLOC, \alpha_1, \alpha_2, \ldots, \alpha_n) \tag{4.1}$$

式 (4.1) の α_1 から α_n は，開発効率に影響を与える因子であり，コスト・ドライバと呼ばれる．関数 f にべき式を用いて，実際のデータから具体的な因子とその値を推測し，まとめたものが COCOMO の研究である．

回帰分析によって得られた係数 $\hat{\alpha_i}$ はコスト・ドライバであり，開発規模以外の要因として $Effort$ に影響を及ぼしていると考えられる．コスト・ドライバの研究は，コスト・ドライバを明らかにすることにより，コスト・ドライバ $\hat{\alpha_i}$ の中で制御可能なものがあれば，その値を改善する活動により企業や組織レベルの開発効率を向上させることができるという考えから始まった．COCOMO におけるコストモデルの研究目的は，開発規模から工数を見積もるためではなく，包括的な分析により企業や組織レベルで開発効率を高めるコスト・ドライバを発見し，改善することにあった．Boehm が著書のタイトルを「Software Engineering Economics」と名付けたのも，この目的のためである [102].

回帰式として一般的なものは線形の重回帰分析である．しかし，線形の回帰式では多様なソフトウェア開発の統計データに対して良い結果が得られないことから，Boehm は統計値を対数に変換して回帰分析を行い，べき式で表現する COCOMO を完成させた．COCOMO のモデル式を式 (4.2) に示す．

$$E = a_b(KLOC)^{b_b} \prod_{k=1}^{15} c_i \tag{4.2}$$

式 (4.2) における a_b，b_b はアトリビュートと呼ばれる係数である．c_i もアトリビュートであるが，この部分がコスト・ドライバと呼ばれる係数である．COCOMO では，15 個のコスト・ドライバが抽出されており，次の 4 つのアトリビュート群に分類して定義されている．

Product attributes c_1〜c_3：プロダクトに対する特性であり，信頼性，データベース規模，複雑性の 3 つのコスト・ドライバで構成される

Hardware attributes c_4〜c_7：動作環境やハードウェアに対する特性であり，性能，資源制約に関する 4 つのコスト・ドライバで構成される

Personnel attributes c_8〜c_{12}：人的資源に対する特性であり，スキル，経験に関する 5 つのコスト・ドライバで構成される

Project attributes c_{13}〜c_{15}：プロジェクトに対する特性であり，ツール，方法論，スケジュールの 3 つのコスト・ドライバで構成される

これら 15 個のコスト・ドライバには，Very Low から Extra High までの 6 段階の評価レベルが定義されている [106]．それぞれの評価レベル毎に係数の具体的な値が定義されており，その最頻値は，0.9 〜 1.4 の幅の値に収まっている．Boehm の研究では，TRW 社における 63 プロジェクトを対象として回帰分析が行われている．それゆえ，式 (4.2) のアトリビュートやコスト・ドライバの値は TRW 社のデータに対する回帰分析の結果であり，絶対的なものではない．Boehm は，最大のコスト・ドライバとしてチームの能力を挙げている．Boehm は，その後もさまざまな対象における計測と分析を重ね，その研究成果は COCOMO Ⅱ として現在に至っている [48]．

日本においても COCOMO の実証研究が行われ，べき式による回帰分析が成立すること，および，コスト・ドライバは組織や対象分野によって異なることが報告されている [47]．この実証研究では，係数の大きさに差はあるが，人的資源やプロジェクトに対する特性には共通点が多く観られ，最大のコスト・ドライバはチームの経験や能力であった．

コストモデルの定義と検証は 1980 年代に完成したが，産業界においてコストモデルを単独で利用することは少ない．これは，産業界が求めているものは見積もりであり，コス

トモデルを見積もり式として用いるには機能や精度が不十分であるためである．プロジェクトの見積もりでは，開発規模と人的資源の関係だけで開発量や納期が決まることはなく，さまざまな状況を加味する必要がある．そもそも，プロジェクトの計画段階では，開発規模の見積もりすら困難な課題である．そのため，見積もりには，経験的なアプローチが主に使われている [107]．本章の目的は OSS 開発プロジェクトの見積もりではなく，OSS 開発における開発規模と人的資源の包括的な関係から，OSS 開発におけるコスト特性について調べることである．

4.3.2 COCOMO における分析方法

COCOMO のモデル式を使って分析するためには，開発規模 ($KLOC$)，工数 ($Effort$)，15 個のコスト・ドライバの評価レベルのデータが必要である．コスト・ドライバの評価レベルは，コスト・ドライバのアトリビュート毎に設定され，それぞれ 4 〜 6 段階から選択する [106]．たとえば，開発チームにおける未経験者の割合であれば，平均的な割合を基に増減を評定する．開発規模は，用いる言語によって定義が異なるが，COCOMO は包括的な分析であり，詳細な規定を守ることを求めていない．

分析では，COCOMO のモデル式 (4.2) を次のように変換した．まず，式 (4.2) の両辺を対数として変形させ，式 (4.3) とする．

$$\log(E) = b_b * \log(KLOC) + \log(a_b) + \sum_{k=1}^{15} log(c_i) \tag{4.3}$$

次に，式 (4.3) において，$\log(E)$ を Y，$\log(KLOC)$ を X，$\log(a_b)$ を A，$log(c_i)$ を C_i とすれば，式 (4.4) となる．式 (4.4) は，データを対数変換すれば，線形回帰式として分析に用いることができる．

$$Y = b_b X + A + \sum_{k=1}^{15} C_i \tag{4.4}$$

4.3.3 OSS における計測と分析方法

本章では，OSS 開発におけるコスト・ドライバを求めることを目的としていない．よって，C_i については計測を行わない．式 (4.4) では，$\sum C_i$ を誤差 e と考える．しかし，回帰

分析においては，誤差 e は結果的に A に含まれるため，$A_e = A + e$ として e を消去し，式 (4.5) とする．これにより，A と A_e は区別しない．

$$Y = b_b X + A_e \tag{4.5}$$

OSS 開発では，ES 開発における労務管理そのものが存在しないため，開発に費やした時間の記録は存在しない．それゆえ，OSS 開発においては，ES 開発で一般的に用いられている工数 (Person-Months) を単位として $Effort$ を計測することは難しい．分析を行うためには，$Effort$ を表す単位を定義する必要がある．そこで，OSS 開発に参加し，何らかの著作物を開発ライブラリ (本研究では Git) に著作者として Commit した人を計測対象とし，開発期間を 1 ヶ月単位で区切り，その区間内で 1 回以上の Commit 記録があれば，1 単位として定義した．Person-Months と区別するため，ここでは Effort Person-Months (以降，図表中では Effort-Months または EM と記載する) と呼ぶ．Commit 者ではなく著作者としたのは，OSS 開発にはレビューを行って Commit する活動が存在するため，この活動によるレビュワーとしての Commit と区別するためである．

$Effort$ の単位を Person-Months ではなく Effort Person-Months と定義したため，計測単位が異なることから，ES 開発と OSS 開発のコストモデルの係数を直接比較することはできない．また，Effort Person-Months は労働時間ではないため，Person-Months に換算することも難しい．計測単位は異なるが，OSS 開発を表すコストモデルの包括的な分析には影響が少ないと考えた．計測単位が異なることによるコストモデルへの影響の有無は，統計分析における検証により確認する．

4.4 OSS 開発プロジェクトの選定

本節では，OSS 開発の分析対象とした 50 件の OSS 開発プロジェクトの選定方法と選定した OSS 開発プロジェクトの概要を示す．

4.4.1 OSS 開発プラットフォームの選定

OSS 開発のコストモデルを分析するには，まとまった量の OSS 開発データを入手する必要がある．分析データの入手対象としては，バージョン管理システムを使用した Web 共有サービスである GitHub や Bitbucket [93] などが考えられる．さまざまなプラットフォームの中でも，GitHub は最も勢いを増している [94]．そこで，本章では GitHub 上で公開されている OSS 開発プロジェクトのデータを用いた．

4.4.2 GitHub における分析対象の選定

GitHub は，バージョン管理システムである Git を用いている．Git の履歴には，プロジェクトの成果物を集めたリリースリポジトリに登録を行う手続きである Commit と共にメンバの活動が記録されている．1 つの Commit 記録には，追加や修正された対象のファイル，追加行数，削除行数の情報，メンバ名およびタイムスタンプが残されている．GitHub には膨大な OSS 開発プロジェクトが存在するため，プロジェクトを選定する必要がある．本章では，次の手順でサンプルプロジェクトの選定を行った．

まず，GitHub の Advanced Search 機能を用いて，次の手順で 342 件のサンプルプロジェクト候補を抽出した．

(1) 長期的なデータを収集するため，GitHub が始まった初期の 2012 年に登録された OSS 開発プロジェクトで，2017 年まで継続しているものを抽出

(2) 大規模 OSS 開発プロジェクトとして，Repository サイズが 15 MB 以上のものを抽出

(3) 大規模 OSS 開発プロジェクトとして，開発者が自身の開発環境に複製を行い，開発作業を行う Fork 数が 200 以上のものを抽出

(4) OSS 開発ではプロダクト自身に価値があるものとし，OSS の利用者が評価する Star 数が 1,000 以上のものを抽出

次に，OSS 開発プロジェクトの著者数が少ないと偏りが生じることから，人的資源が投入されている OSS 開発プロジェクトとして Contributor 数，Commit 数に着目した．342 件のサンプルプロジェクト候補における Contributor 数と Commit 数を取得し，各値共に第 2 四分位 (中央値) 以上に属す 53 件の OSS 開発プロジェクトを抽出した．平均値ではなく，第 2 四分位を用いて分類したのは，いずれの値も偏りが大きかったためである．以上の条件を満たす 53 件の OSS 開発プロジェクトの中で，欠損値のある OSS 開発プロジェクト 3 件を除外し，50 件のサンプルプロジェクトを選定した．

選択した 50 件のサンプルプロジェクトを Table 4.1 に示す．Table 4.1 は，データ取得日である 2017 年 8 月 11 日時点の数値である．Table 4.1 の Contributors は OSS 開発プロジェクトに関わっている人数を表し，Commits はリリースリポジトリへの追加変更回数を表す．Stars は OSS 開発プロジェクトの特性ではなく，当該 OSS に対する利用者からの評価を表している．選定したデータに含まれる OSS 開発プロジェクト単位で集計した著作者数の合計は約 40 万人，リリース件数は約 226 万件であり，統計分析には十分な量であると考える．

Table 4.1: Selected 50 OSS Development Projects (as of 2017/08/11)

No.	Projects	Contributors (number)	Commits (number)	Stars (number)
1	ansible	3,120	25,886	587
2	atom	25,114	32,636	2,948
3	bokeh	39,606	32,962	369
4	bolt	6,368	16,327	248
5	cloudfoundry/bosh	3,400	22,325	233
6	canjs	1,294	15,287	236
7	CleverRaven/Cataclysm-DDA	1,521	9,032	154
8	llvm-mirror/clang	1,480	51,448	549
9	collectd	1,411	69,033	400
10	conda	1,678	9,438	299
11	contiki-os/contiki	1,465	10,540	136
12	owncloud/core	2,463	12,314	159
13	crystal-lang/crystal	5,471	35,529	433
14	darktable-org/darktable	8,899	9,856	234
15	DefinitelyTyped	1,353	19,829	209
16	django	12,006	34,561	4,487
17	druid-io/druid	27,878	24,790	1,461
18	PX4/Firmware	5,356	8,063	194
19	guardian/frontend	1,272	23,341	240
20	gratipay/gratipay.com	3,627	75,145	152
21	HabitRPG/habitica	1,090	11,472	147
22	hazelcast	5,177	15,810	351
23	caskroom/homebrew-cask	2,198	24,154	143
24	JetBrains/kotlin	11,479	65,105	3,751
25	libgdx	17,128	41,059	177
26	raspberrypi/linux	12,285	13,000	396
27	llvm-mirror/llvm	3,926	639,762	7,233
28	lodash	2,212	153,788	468
29	meteor	25,915	7,858	238
30	mpv-player/mpv	38,136	19,386	354
31	neo4j	6,167	45,344	219
32	getnikola/nikola	4,116	49,210	146
33	NixOS/nixpkgs	1,269	8,730	161
34	opencv	1,841	114,295	1,323
35	openlayers	17,864	21,998	766
36	phpmyadmin	2,849	21,524	188
37	hrydgard/ppsspp	2,478	110,816	776
38	PrestaShop	3,071	22,641	206
39	prestodb/presto	2,580	43,436	409
40	qemu	6,409	11,413	192
41	radare/radare2	1,321	55,398	747
42	ReactiveCocoa	5,259	15,829	394
43	rethinkdb	17,745	8,409	193
44	RIOT-OS/RIOT	19,681	33,307	135
45	servo	1,377	14,976	151
46	spring-projects/spring-boot	9,954	28,551	778
47	videolan/vlc	15,499	13,107	373
48	w3c/web-platform-tests	3,073	71,870	454
49	yiisoft/yii2	1,180	25,795	747
50	yii2	10,702	16,926	782
	Total	408,763	2,263,311	36,126

4.5 COCOMO のモデル式の成立確認および OSS 開発の特性の調査

本節では，ステップ 1 ： COCOMO のモデル式の成立確認，およびステップ 2 : OSS 開発の特性調査に対する分析方法とその結果について示す．

4.5.1 分析データの整形

OSS 開発プロジェクトの特徴は 2 つある．ひとつは開発手法として継続的インテグレーション (CI: Continuous Integration) が用いられており，頻繁にリリースが行われていることである．もうひとつは，ES 開発のような有期限のプロジェクトとは異なり，利用者の評価があれば何年も開発が継続されることである．選定した 50 件の OSS 開発プロジェクトはすべて CI であり，かつ，長期間に渡って開発が継続されていたため，OSS 開発プロジェクトの特性から期間を区切ることができなかった．そのため，分析データの期間を定義する必要が生じた．それぞれのプロジェクト開始時点から 12 ヶ月単位で期間を区切り，1 年を単位として分析データを整形した．この処理により 370 件の $KLOC$ と $Effort$ から成る分析データを得た．分析データの対数表示による度数分布を Fig. 4.1 に示す．分析用のデータは，Fig. 4.1 の度数分布図で示すように，対数正規分布とみなすことができる．データ数は 370 件であり，回帰分析を行う条件を満たしていると考える．

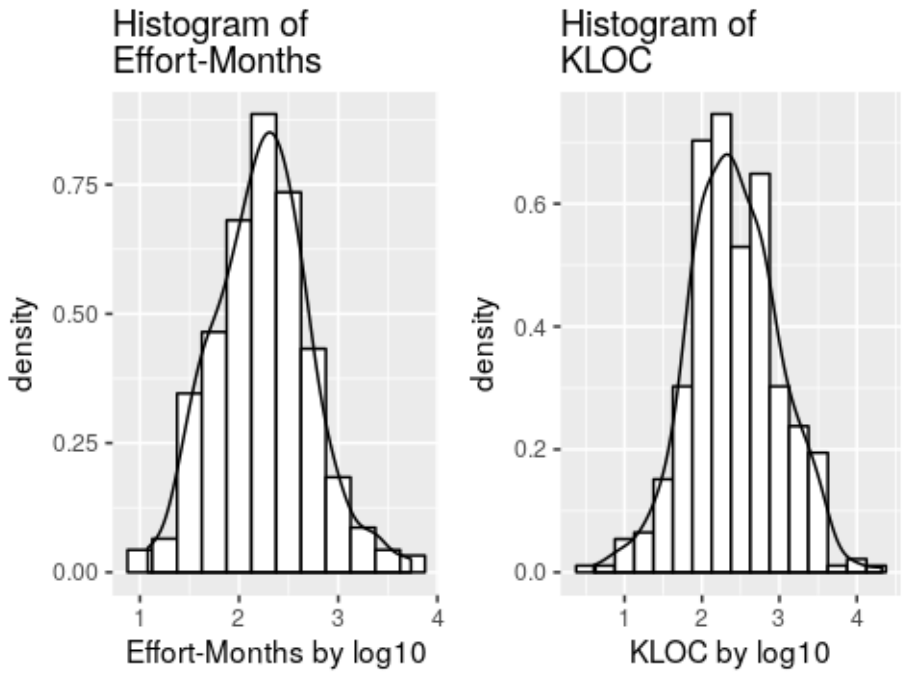

Fig. 4.1: Histogram of Sample Data

4.5.2 回帰分析と評価

年間データとして整形した 370 件のデータを log 変換し，単回帰分析を行った．単回帰分析の結果として得られた回帰線と 370 件のデータのグラフおよび Q-Q プロットを Fig. 4.2 に，回帰分析のサマリを Table 4.2 に示す．

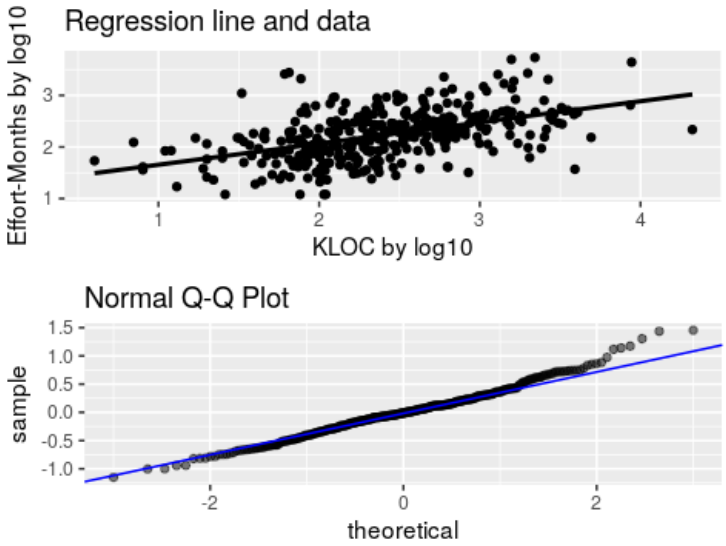

Fig. 4.2: Regression Line and Q-Q plot

Table 4.2: Correlation Analysis Result of 370 Data

	Estimate	Std.Error	t value	Pr($>\|t\|$)	Signif.codes [a]
(Intercept)	1.24	0.0937	13.3	2.0E-16	***
Lines	0.416	0.0378	10.9	2.0E-16	***

Residual standard error: 0.418 on 368 degrees of freedom
Multiple R-squared: 0.243, Adjusted R-squared: 0.241
[a] ***:0.1% Significance

この結果から，式 (4.5) に係数の値を与え，式 (4.6) とした．

$$Y = 0.416X + 1.24 \tag{4.6}$$

式 (4.6) を，COCOMO のべき式に変換し，式 (4.7) とした．

$$E = 10^{1.24}(KLOC)^{0.416} = 17.3(KLOC)^{0.416} \tag{4.7}$$

COCOMO のべき式による回帰分析を OSS 開発に対して行うため，$Effort$ の計測単位として COCOMO の定義とは異なる Effort Person-Months を用いた．この定義が回帰分析に影響を与えていないことを確認する．

分析に用いた $Effort$ の単位が非線形であったとしても，分析対象の両対数データにおいて統計的な偏りがなければ，回帰分析への影響は少ないと考える．回帰分析が成り立っているかは，対数変換後のデータにおいて正規分布になっているかを確認すれば良い．Fig. 4.1 に示した通り，対数変換後の分布は対数正規分布と見なせる．さらに，この分布が正規分布に近いことを確認するため Q-Q プロットを用いて確認し，Fig. 4.2 で示した通り，ほぼ直線であることが確認できた．この結果から，計測単位として Effort Person-Months を用いることによる悪影響は生じていないと考える．

以上の結果から，ステップ 1 の結果として，選択した GitHub の 50 件の OSS 開発プロジェクトから得られた 370 件の年間データに対して，COCOMO のべき式が回帰式に従っていると考える．

4.5.3 COCOMO のモデル式における係数の比較

次に，ステップ 2 の COCOMO のモデル式における係数の比較について考える．式 (4.2) の a_b は，COCOMO では Person-Months (人月) であり，OSS 開発では独自に定義した Effort Person-Months である．$Effort$ の計測単位が異なるため，比較することはできない．一方，b_b は，$Effort$ の単位の影響を受けないので比較が可能である．COCOMO では b_b の値は 1.05 から 1.20 であり，開発規模が増加すると指数的に $Effort$ が増加し，開発効率が低下する．OSS 開発に対する回帰分析では，Table 4.2. より b_b の値は 0.416 であり，開発規模が増えると開発効率が向上することを示している．

4.5.4 COCOMO のモデル式の成立確認および OSS 開発特有の特性の調査結果のまとめ

選択した 50 件の OSS 開発プロジェクトのデータに対して回帰分析により COCOMO のべき式を求め，係数を評価したのでステップ 1 ： COCOMO のモデル式の成立確認，およびステップ 2 : OSS 開発特有の特性調査の結果をまとめる．

- ステップ 1: COCOMO のモデル式が成立することを統計的な手法を用いて確認することができた
- ステップ 2: COCOMO のモデル式のべき乗係数は，OSS 開発の分析によって得られた値と大差があった．COCOMO では開発規模の増加に反比例して開発効率は低下するが，OSS 開発では開発規模の増加に伴い開発効率が向上していることを示した

4.6 節では，OSS 開発のべき乗係数が COCOMO と比較して小さい値となる現象をステップ 3 : OSS 開発特有の特性を生む原因の調査として，その原因について分析する．

4.6 OSS 開発の特徴分析

本節では，OSS 開発の開発効率を特徴づけるべき乗係数の値について分析を行う．

4.6.1 べき乗係数の意味と OSS 開発の特性

量的生産を行う産業においては，一般的に規模を拡大すると量産効果と習熟効果により生産効率が向上する．ソフトウェア開発においては，一般的な産業と異なり，べき乗係数は 1 より大きく，開発規模の拡大が開発効率を低下させることが知られている．開発効率を低下させる原因として，ソフトウェア・プロジェクトの大型化によるリスクの拡大やコミュニケーション・ロスなどが挙げられている [108]．GitHub における OSS 開発の分析結果では，べき乗係数は 0.416 であり，開発規模の拡大により開発効率が向上していることが示された．この特徴が，計測上の違いから生じているのか，OSS 開発の特性から生じているのかについて調査する．

OSS 開発の特性に影響を与える要因として，第一にオープンソース・ライセンスの利用で許容されている流用が考えられる．流用による開発規模あたりのコストと，開発による開発規模あたりのコストは異なるため，流用の状況について調べる必要がある．膨大な数の Commit において，流用が想定される通常の開発や保守とは異なる Commit を識別する方法を探るため，次の手順で分析を行った．

- 手順 1：著作者別の開発量の偏りを調査
- 手順 2：貢献の大きな著作者の活動を時系列で調査
- 手順 3：流用が想定される特異な Commit を識別する閾値を仮定
- 手順 4：特異値を除いたコストモデルを作成し評価

4.6.2 著作者別の開発量の偏り

OSS 開発プロジェクトは，開放されたコミュニティであり，自由に参加することができる．OSS 開発では，コミュニティへの参加を推進するため，誤字修正など軽度の活動から貢献することを奨励しており，これにより多くの著作者が存在する．コミュニティの成果物への Commit は，主催者側のレビューなどの評価を経る必要があり，自由に行えるもの

ではない．OSS 開発では，誤記修正など開発量の少ない入門者的な著作者のものを含め，膨大な数の Commit が存在している．そこで，まず，開発量における著作者の分布特性について調査した．

特性を調査するため，1 ヶ月を期間として，月単位で著作者別に Commit 数を集計し，追加した開発規模をキーとして降順に並べた順位データを各プロジェクト単位で作成した．Table 4.1 の 2 番目に示した ”atom” プロジェクト [109] を例に挙げると，448 名の著作者が 2,029,562 行の追加を 29,500 回の Commit で登録している．atom プロジェクトを例として，順位データから求めた開発全期間を通した著作者の月別の追加行数の度数分布を Fig. 4.3 に示す．

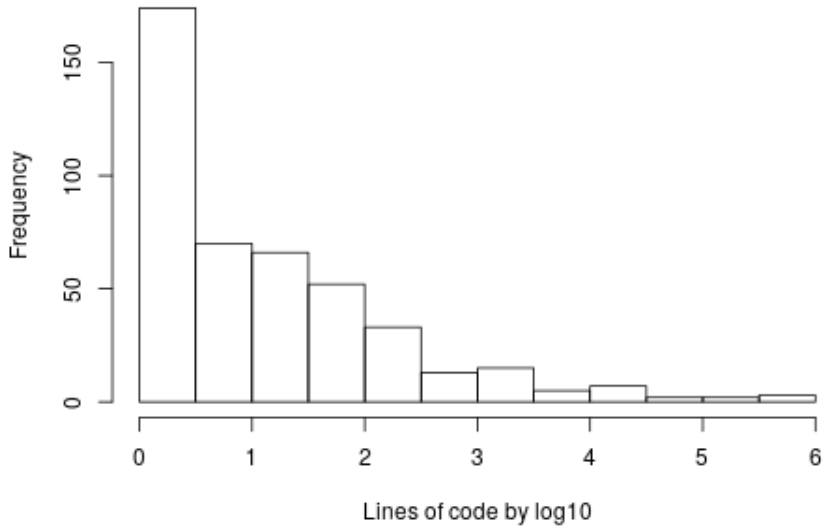

Fig. 4.3: Histogram of Lines in the atom Project by log10

Fig. 4.4 は，atom プロジェクトにおける著作者毎の開発規模の順位を 2 つの異なる表示で示している．上段の Ranking chart は，縦軸に著作者ごとの追加開発行数を月別開発量として示し，横軸は開発量の順位を降順に示している．下段の Ranking chart by log10 は，上段の Ranking chart を両軸とも常用対数 (log10) に変換して示したものである．順位データである Fig. 4.4 に示す通り，非常に偏った分布であること，および，両対数において直線になることから，べき分布であると推測できる．

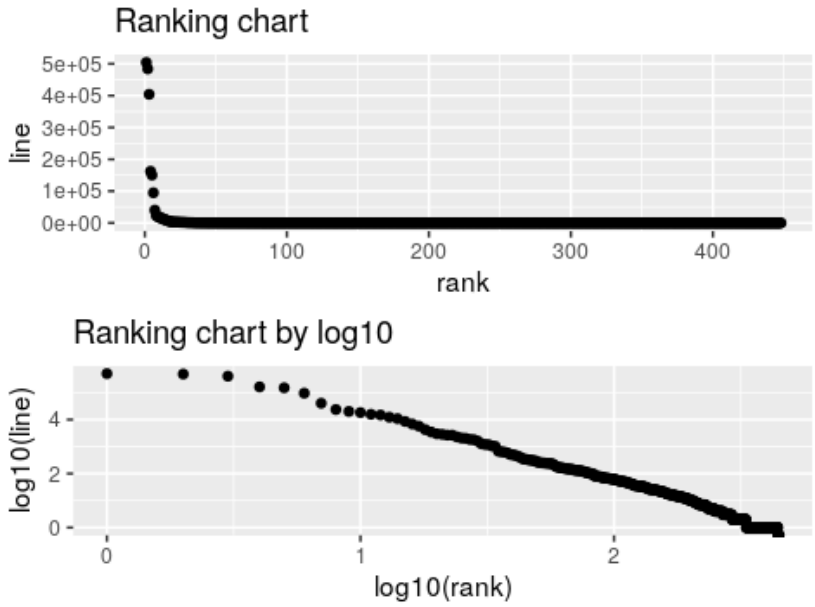

Fig. 4.4: Ranking of Additional Lines in the atom Project

対象とした 50 件の OSS 開発プロジェクトにおいて，atom プロジェクトと同様に順位データを作成した結果，同様のべき分布が観測された．べき分布は，平均値や分散などで特性を表現できないことが知られている．ここでは，各 OSS 開発プロジェクトのべき分布の特徴を表すため，各 OSS 開発プロジェクトにおける全開発規模の 80 %，90 %，95 % の規模を登録した著作者数の割合を求めた．その分布を箱ひげ図として Fig. 4.5 に示す．

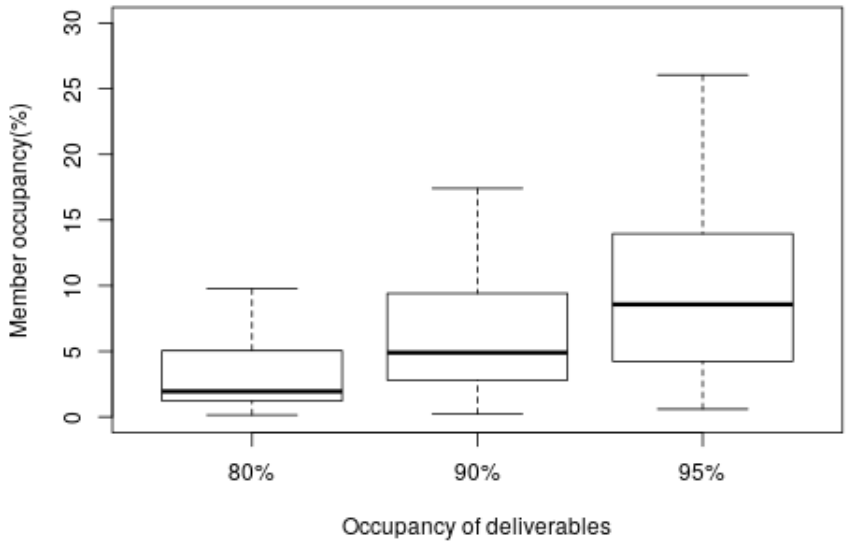

Fig. 4.5: Development Ratio for Deliverables in All Projects

Fig. 4.5 は，上位の四分位点を箱で表したもので，箱内の横線が第 2 四分位 (中央値) を示す．箱外に表示された上下の横線は外れ値を除外した最小値と最大値を表しており，外れ値がある場合には，白丸で示される．Fig. 4.5 に示した 80 % の場合，著作者数の全体比率は最小が 0.15 %，第 1 四分位が 1.2 %，第 2 四分位 (中央値) が 1.9 %，第 3 四分位が 5.1 %，最大が 9.8 %であった．Fig. 4.3 に示した atom プロジェクトでは，5 名の著作者が全開発規模の 80 % を Commit していた．一般的なパレート分布は 80% : 20% の比率であり，パレート則よりさらに偏った分布である．

4.6.3 著作者の時系列記録

Fig. 4.4 に例示した順位データから，べき分布の Head 部分と Tail 部分に特異なデータが存在していると考えられる．Head 部分は，上位の著作者による開発規模であり，流用による開発活動が含まれている可能性がある．Tail 部分は，少量の誤字修正など，単位 Effort Person-Months として扱うには小さすぎる開発活動である．Tail 部分の特異値は，追加と削除の規模が小さい Commit を検出することで，簡単に見つけることができる．

流用の特異値については，著作者の活動の中に混ざっていることから，分離する方法を

検討した．分離を行うために，著作者毎の Commit を時系列で分析した．開発効率に影響を及ぼす流用があれば，当該著作者の累積規模の時系列値は急激に大きくなる．そこで，各 OSS 開発プロジェクトにおいて累積登録規模が大きい著作者を中心に時系列データを作成した．例として，Table 4.1 の 44 番目に示す ”RIOT-OS/RIOT” プロジェクト [110] の時系列記録を Fig. 4.6 に示す．この時系列記録は，累積登録規模が多い 16 名の著作者を対象としている．グラフ上で線が途切れているのは，それ以前，あるいはそれ以降に Commit の記録が見当たらない場合である．

Fig. 4.6 において，開発規模が跳躍している部分が，流用が行われたと推察される Commit である．

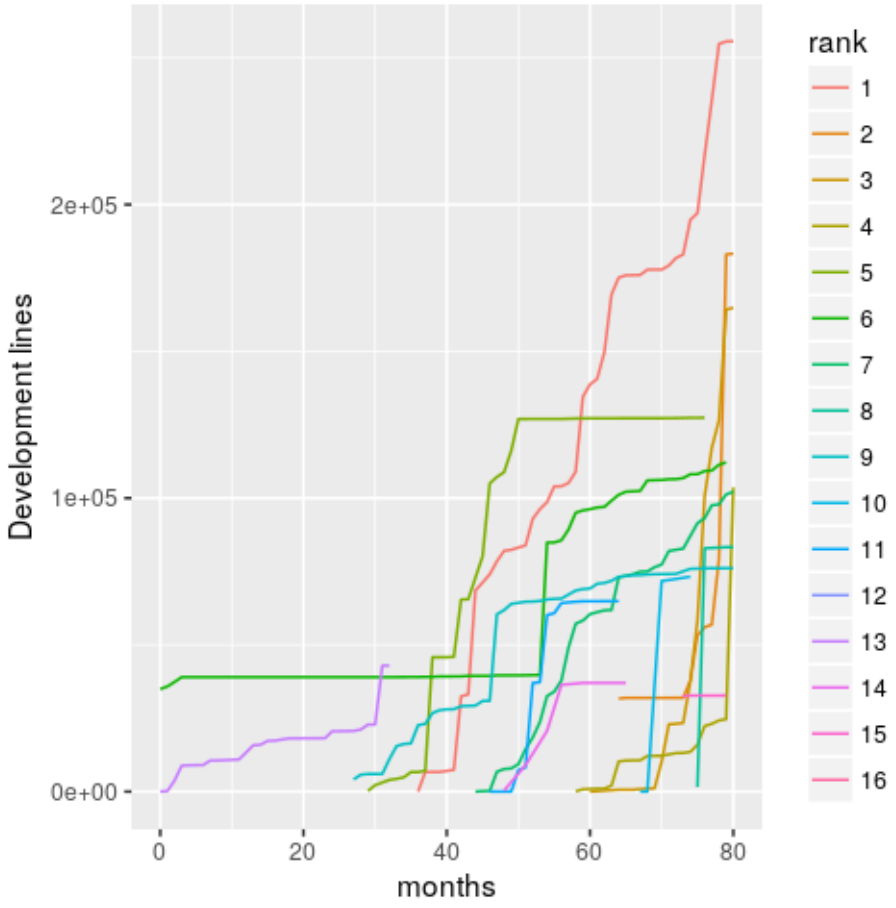

Fig. 4.6: Cumulative Lines of Monthly Commits in the RIOT project

4.6.4 特異値の分離とデータ整形

対象がべき分布であることから，Head 部分と Tail 部分の特異値を調べ，分離する値を定める必要がある．Fig. 4.6 で示した通り，Head 部分の特異値として流用と想定される規模の増加が観測された．OSS 開発において，ソースコードの流用は，オープンソース・ライセンスに従えば自由に行うことができる．また，OSS 間の依存関係により，流用している他の OSS にアップデートが発生した場合には，必然的に流用しなければならなくなる．OSS 開発において流用は頻繁に行われており，流用の規模はさまざまである．流用の閾値として用いるための参考値となる先行研究を調査したが，見当たらなかった．そこで，GitHub で OSS 開発を行っている 10 人のエンジニアに Fig. 4.6 を見せ，何 KLOC / Months から流用と考えるかアンケートを実施した．アンケートの結果，10 人のうち 7 人が 5KLOC，2 名が 3KLOC，1 名が 10KLOC と回答した．Fig. 4.6 の結果とアンケートの結果から，5 KLOC / Month 程度の飛躍が流用であると推察される．そこで，1 ヶ月間に 5 KLOC 以上の追加を分離することとした．ただし，OSS 開発プロジェクト開始時には大量の新規登録が行われるので，この分離則の対象外期間として 3 ヶ月間を設定した．Tail 部分は，追加と削除の合計が 10 LOC より小さい Commit を分離することとした．

月単位で Commit を著作者別に集計し，Head 部分と Tail 部分の特異値を除去することで分離を行った．除去したデータから 4.5.1 項と同様に回帰分析のための整形データを作成した．その結果を Table 4.3 に示す．$Effort$ の減少のうち 95 % は，Tail 部分の特異値であり，追加規模のほぼ 100 % は Head 部分の特異値であった．

この特異値除去の結果を 12 ヶ月単位でまとめ，分析用の 368 件の $KLOC$ と $Effort$ から成るデータを得た．分析データの対数表示による度数分布を Fig. 4.7 に示す．特異値除去前の Fig. 4.1 と比較して，顕著な差はない．特異値除去後のデータも Fig. 4.7 の度数分布図で示すように，対数正規分布とみなすことができる．

Table 4.3: Selection of Analysis Data

Operation	Effort	Commits	Add (KLOC)	Del (KLOC)
Before	123,514	1,381,655	237,698	166,191
After	91,651	1,283,941	132,299	126,330
Difference	31,863	97,714	105,399	39,861

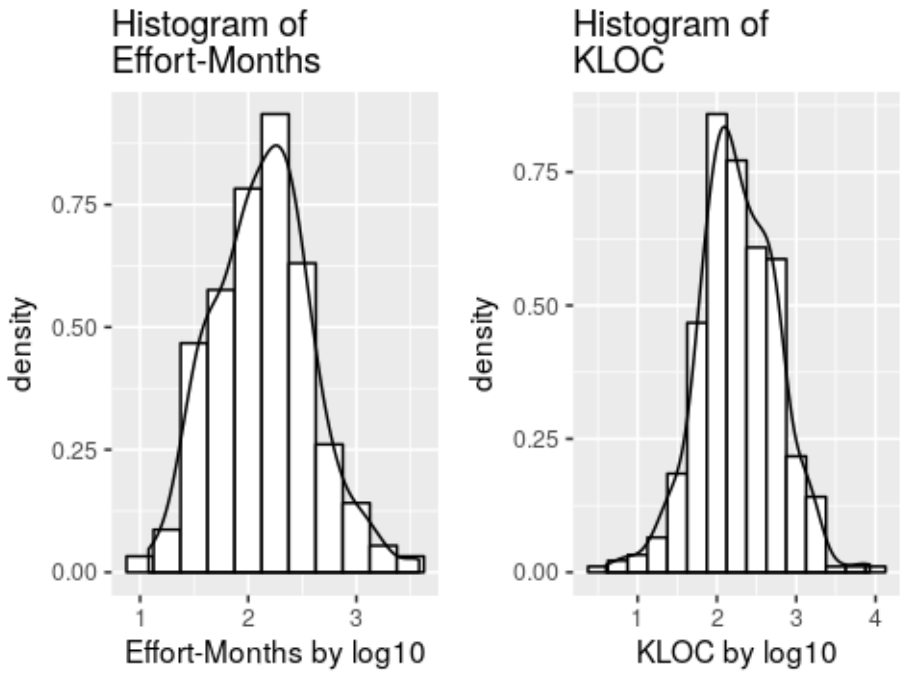

Fig. 4.7: Histogram of Sample Data

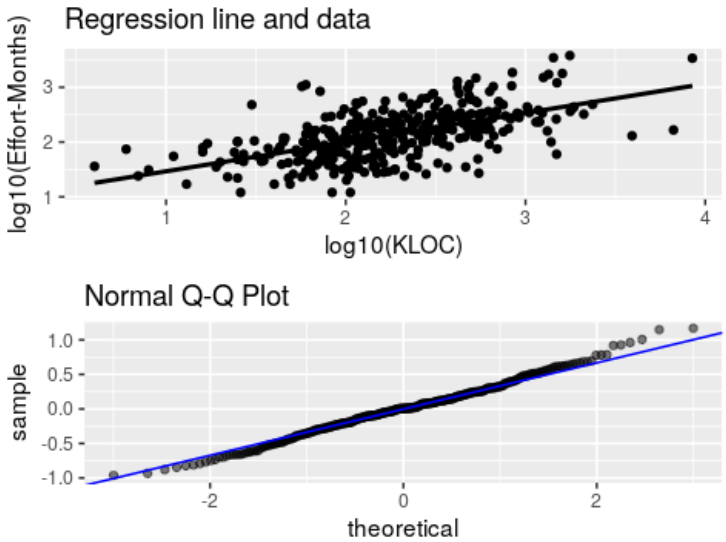

Fig. 4.8: Regression Line and Q-Q plot

Table 4.4: Correlation Analysis Result of Selected 368 Data

	Estimate	Std.Error	t value	Pr($>\|t\|$)	Signif.codes [a]
(Intercept)	0.932	0.0901	10.3	2.0E-16	***
Lines	0.533	0.0389	13.7	2.0E-16	***

Residual standard error: 0.368 on 366 degrees of freedom
Multiple R-squared: 0.339, Adjusted R-squared: 0.337
[a] ***:0.1% Significance

4.6.5 特異値を除去した OSS 開発のコストモデル

特異値を除去した 366 件のデータを log 変換し，単回帰分析を行った．分析により得られた回帰線とデータのグラフおよび Q-Q プロットを Fig.4.8 に，回帰分析のサマリを Table 4.4 に示す．Table 4.4 の結果から式 (4.5) の係数に値を与え，式 (4.8) とした．

$$Y = 0.533X + 0.932 \tag{4.8}$$

式 (4.8) を COCOMO のべき式に変換して式 (4.9) とした．

$$E = 10^{0.932}(KLOC)^{0.533} = 8.55(KLOC)^{0.533} \tag{4.9}$$

4.6.6 特異値除去モデルの評価

特異値を除去したデータに対する回帰分析の結果として，Table 4.2 と Table 4.4 を比較する．Adjusted R-squared が大きくなっていることから，特異値を除去した方が，$Effort$ と $KLOC$ の相関は高くなっている．Q-Q プロットについても Fig. 4.2 と Fig. 4.8 を比較すると，特異値を除去した方が開発規模が大きい部分での差が小さくなっているが，大きな違いは見られない．特異値除去前後のデータ比較の結果，両者の間に大きな違いは見られない．これにより，特異値を除去したモデルであっても，COCOMO のべき式が回帰式に従っていると考えられる．

べき式 (4.7) と (4.9) の差は，係数が 17.3 から 8.55 と半減している．この結果は，Tail 部分の微細な修正の除去の方が，流用による開発規模の増加より影響を与えたことを示している．べき乗係数は，0.416 から 0.553 と微増であり大きな変化は生じなかった．分析

対象の総開発規模は，月間で 5 KLOC を超える流用を取り除いたため 237 MLOC から 132 MLOC と半減しているが，開発効率に対する流用の影響が少ないことを示した．

特異値除去モデルの評価により，Head 部分は大規模な流用によって発現しているものではなく，OSS 開発チームには開発を推進する少数のメンバが存在していることが明らかとなった．ここでは，Head 部分に相当する開発を推進する少数のメンバを「コアメンバ」と呼ぶ．

4.6.7 OSS 開発特有の特性を生む原因の調査結果のまとめ

ステップ 2 : OSS 開発特有の特性調査の結果として，OSS 開発のコストモデルにおけるべき乗係数が 1 より小さい結果を得た．そこで，この結果の検証と原因を探るための分析を行った．この原因に対し,「大規模な流用による見かけの開発量の増加が影響している」と仮定し，OSS 開発プロジェクトにおける著作者の Commit を時系列で調べた．調査の課程において順位データを作成したところ，すべての OSS 開発プロジェクトにおいてべき分布であることが確認された．べき分布であることから，Head 部分と Tail 部分で異なった特性を持っていることが確認された．そこで，(a) 大規模な流用の影響，(b) 貢献度の小さい多数の開発者の影響，の 2 つがべき乗係数の差を生じさせる原因であると仮定し，Head 部分と Tail 部分に存在する特異値の除去を行った．特異値を除去したデータを用いて，回帰分析を行い，べき式 (4.9) を得た．べき乗係数は，特異値除去前と同様に 1 より小さい値であった．この結果から，(a) および (b) については，いずれも OSS 開発の開発効率への影響は認められず，OSS 開発特有の特性が大規模な流用や貢献度の小さい多数の開発者の存在よるものではないことが明らかになった．

4.7 結言

本章では，OSS 開発におけるメンバと開発量の関係，すなわち量的開発効率について，べき式を用いた COCOMO を適用し，次の 3 つのステップで，GitHub における 50 件の OSS 開発プロジェクトを対象として分析を行った．

- ステップ 1 : OSS 開発においても，COCOMO における量的開発効率の関係式，すなわち工数と成果物の関係を表すモデル式 (工数=$a *$ 開発規模$^b +$ コスト因子) が成り立つことを確認する
- ステップ 2 : COCOMO のモデル式の係数から，OSS 開発特有の特性を探る
- ステップ 3 : OSS 開発特有の特性を生む原因を探る

ステップ 1 の結果は，4.5.2 項に示した分析により，統計的に成立することが確かめられた．統計分析で得られた式 (4.7) のべき乗係数は 0.416 であった．COCOMO において標準値とされているべき乗係数は 1.05 〜 1.20 と 1 より大きい値である．OSS 開発のべき乗係数と COCOMO のべき乗係数には大きな差が観測された．

ステップ 2 の結果は，OSS 開発特有の特性は，ステップ 1 において観測された 1 より小さいべき乗係数である．この結果から，開発規模が増加すると開発効率が上昇することが確認された．

ステップ 3 として，OSS 開発における著作者の詳細な記録を調べた．調査の結果，本章で対象とした GitHub の 50 件の OSS 開発プロジェクトについて，著作者の開発量の順位データがすべてべき分布であることが分かった．べき分布の Head 部分では，大規模な流用が想定される Commit があり，これによる開発規模の増加が見られたため特異値として除去し，結果として分析データの開発規模が半減した．Tail 部分では，誤字修正のような微細な Commit があり，これを特異値として除去した．特異値を除去したデータに対し特異値除去前と同様の分析を行った結果，べき乗係数は 0.553 であり，特異値除去前と同様に 1 より小さい値であった．このことから，特異値として除去した大規模な流用や貢献度の小さい多数の開発者の存在は開発効率に影響を及ぼしていないことが明らかとなった．大規模な流用に関しては，開発効率に対し多少影響していたが，COCOMO におけるべき乗

係数との大きな差を生む原因は他にあると考える．大規模な流用以外に小規模な流用や CI による自動化などの影響も考えられるが，本章の分析では要因として確定するには至らなかった．大規模な流用以外の小規模な流用や別の高生産性因子の分析など，影響を及ぼしている原因の解明が今後の課題である．

1 MLOC の開発規模において，OSS 開発のべき乗係数 0.553 と COCOMO のべき乗係数の標準値 1.2 を比較すると，COCOMO の開発効率は OSS 開発の 87 倍となる．工数ベースで比較すると，OSS 開発では 5 人年，COCOMO では 435 人年の差となる．OSS 開発プロジェクトと COCOMO で示された ES における開発プロジェクトでは，さまざまな環境が異なるため，この値を単純に比較することはできない．しかしながら，近年目覚ましい発展を遂げている OSS 開発の分析は，ES 開発も含めたソフトウェア開発全体における開発効率向上に役立つ要因分析に繋がっていくと考えている．さらに詳細分析を行い，未確認である OSS 開発における高い開発効率の要因を明らかにすることは重要であろう．

第5章　OSS 開発チームにおけるチームの状態の指標化に関する研究

5.1　緒言

本章では，第 4 章の結果を踏まえて，活動量のばらつきから開発チームの状態の指標化を試みる．第 4 章で分析対象とした OSS 開発プロジェクトの Log データを用い，開発チームにおけるメンバの活動量の分布を基に，ジニ係数とローレンツ曲線を用いて，開発チームの状態を指標化する手法を検証する．

5.2 活動量の分布

OSS は，社会インフラとしてだけではなく，OSS 開発コミュニティの活動に参加するエンジニアの育成にも貢献している [40]．OSS 開発における育成の特徴は，エンジニアが On-the-Job Training (OJT) を得る機会の増大にあると考えられる．従来，エンジニアが OJT を得るには，ナレッジを持った企業やプロジェクトに雇用される必要があり，雇用されるには十分なスキルが求められるため，ビギナーにとって新たな OJT の機会を得ることが困難であった．特に，ソフトウェアのデバッグやテストなどの技能は OJT から学ぶ必要があり，閉鎖的な開発形態の企業においては，新入社員教育に多くのコストを費やしている．OSS コミュニティは，自由な参加を推奨しており，この OJT を得るための障壁が小さい．たとえば，GitHub [92] では，誰でもドキュメントの誤字修正から参加できることを示している [111].

OSS 開発プロジェクト運営の側面から観ると，ビギナーの自由な参加により，稼働率は下がると考えられる．ビギナーの参加が増加すると，活動量のばらつきが大きくなる．活動量のばらつきは，ジニ係数によって，その程度を推測することができる [98]．北山の研究 [98] では，メーリングリストにおける個人毎に集計した投稿数の分布に偏りがあったため，一般的な分散ではなく，ジニ係数を用いて計測することを提案した．また，北山は，ジニ係数の差からメーリングリストの特徴を分析し，メーリングリストの種類によって差異があることを明らかにした [98]．第 4 章で示した通り，OSS 開発プロジェクトの開発量もべき分布であるため，ジニ係数で表すことができると考えた．そこで，本章では，開発活動の負荷分散に着目し，ジニ係数を用いて活動量の分布の計測と，ジニ係数の差を用いた OSS 開発プロジェクトの特徴の分析を試みた．

一般的に，OSS 開発の利用者，開発者，愛好者らの集団を OSS 開発コミュニティと呼ぶ．本章では，計測の単位として Repository 毎に集計している．そのため，計測の単位として OSS 開発プロジェクトと記載している場合には，Repository 単位での集団を指している．

5.3 活動量の分布に関するモデルと計測方法

企業におけるソフトウェア開発プロジェクトや企業経営では，要員の稼働率を高め，総人数を抑えることが，総人数を増やし，稼働率を下げるより効率的だと考えている．これは，要員の雇用により発生する間接コストが，雇用人数に比例する現実からも明らかである．

一方，OSS 開発においては，一部のコアメンバを除けば，従来型の雇用関係ではなく，自由な参加であることから Contribution と呼ばれる社会貢献としての関係である．OSS 開発は，30 年以上の歴史の中で発展し，現在における形態，つまり，GitHub に代表される OSS コミュニティ活動として拡大している．

OSS コミュニティの活動形態に関する研究は，ソフトウェア開発の側面だけでなく，社会科学の側面からも多様な研究が行われている [42, 112, 113]．ここでは，ソフトウェア開発活動の負荷分散に着目した．役割のアーキテクチャまで踏み込まず，総人数と稼働率によるマクロな分析を試みた．

OSS 開発プロジェクトは，要員を雇用していないため，稼働率をマネージできない．それゆえ，稼働率に代わる計測項目が必要となる．そこで，ジニ係数と呼ばれる指標に着目した．この指標は，母集団における配分が均等である状態を 0 とし，不均等の大きさを最大 1 で示す．経済学では，社会や国家における富の不均衡を表す指標として用いられている [99, 100]．また，社会科学では，インターネットにおける集団の特性を表す指標として応用されている [98, 101].

稼働率は，Effort Person-Months や Effort Person-Hours を Total Months か Total Hours で割った値である．Occ: Occupancy rate は，Ef: 全 Effort Person-Months の平均，To: Total Months とすれば，式 (5.1) で表される．

$$Occ = \frac{Ef}{To} \tag{5.1}$$

雇用関係が存在しない OSS 開発プロジェクトでは，Total Months を固定的に決めることが出来ないので，Effort Person-Months の分布の偏りを表すジニ係数を用いる．$Gini$: ジニ係数は，Effort の分布を $L(F)$ とすると式 (5.2) で表される．

$$Gini = \frac{\frac{1}{2} - \int_0^1 L(F)dF}{\frac{1}{2}} = 1 - 2\int_0^1 L(F)dF \tag{5.2}$$

ジニ係数を用いて稼働率に相当する値を出すとすると，式 (5.3) で表される．

$$Occ \fallingdotseq 1 - Gini \tag{5.3}$$

$L(F)$ の値は，OSS 開発プロジェクトのある測定期間における，Contoributor の Effort を用いる．i 番目の Contributor C_i の Effort Ef_{C_i} は，測定期間において活動が記録されている月数とする．たとえば，5 年間の測定期間において，3 ヶ月間に 5 回の Commit 行った場合，Commit 回数の ”5” ではなく，Commit 記録のある 3 ヶ月の ”3” を Effort Person-Months とする．活動量は，バージョン管理 (たとえば Git リポジトリの Log) の記録から求める．全 Contributor の Ef_{C_i} が $L(F)$ に対応する．ジニ係数は，Ef_{C_i} のベクトルから R の統計パッケージを用いて求める．

5.4 分析対象と分析方法

本節では，本章で用いた分析対象と分析方法について述べる．

5.4.1 分析対象

本章の研究では，新規参加者など，実際の OSS 開発活動にはすぐに貢献できないであろう Contributor が多く参入している OSS 開発チームのデータが求められる．この要件に該当する OSS 開発チームの条件は次の通りである．

- ある程度長い期間，継続して開発を行っている
- ある程度の Contributor が参加している
- 継続的に Contributor が参入している

第 4 章で用いた OSS 開発プロジェクトのデータは，これらの条件を満たしている．そこで，本章でも，第 4 章で調査対象とした OSS 開発プロジェクトを分析対象とすることとした．第 4 章で選定した 50 件の OSS 開発プロジェクトの開発記録 (Log) を 2018 年 11 月 30 日に再取得した．その際，Repository が移動しており，開発記録の取得ができなかった OSS 開発プロジェクトが 2 件存在した．そのため，本章では，48 件の OSS 開発プロジェクトのデータを用いることとした．

この 48 件の OSS 開発プロジェクトの開発記録から，OSS 開発プロジェクト毎の開発期間，総人数，総 Commit 数，総編集 File 数，総変更 Lines を集計した．総変更 Lines は，Commit 時に記録されるもので，Add lines と Delete lines を合算した値に等しい．

5.4.2 ジニ係数の算出

”openlayers” プロジェクト [114] のデータを例にジニ係数の算出方法を述べる．

openlayers プロジェクトは，ウェブページに動的な地図を表示させてくれる JavaScript のライブラリである．開発開始は 2006 年 5 月，著作者数は 267 人，Repository サイズは 77,614 KB，Fork 数は 1,778，Star 数は 4,453，Issue 数は 620 であった．

5.3 節にて述べた方法を用いて，次の手順でジニ係数を求めるための Effort Person-Months を算出した．

1. Git Log から openlayers プロジェクト の全期間の Contributor 毎の Commit 記録を収集 (19, 545 件)

2. 月単位で Contributor 毎の Effort Person-Months を算出 (1, 490 件)

3. Contributor 単位で Effort Person-Months を集計 (291 件)

測定期間において Contributor 毎に集計した Effort Person-Months が Contributor の活動量である．Effort Person-Months の分布を Fig. 5.1 に示す．

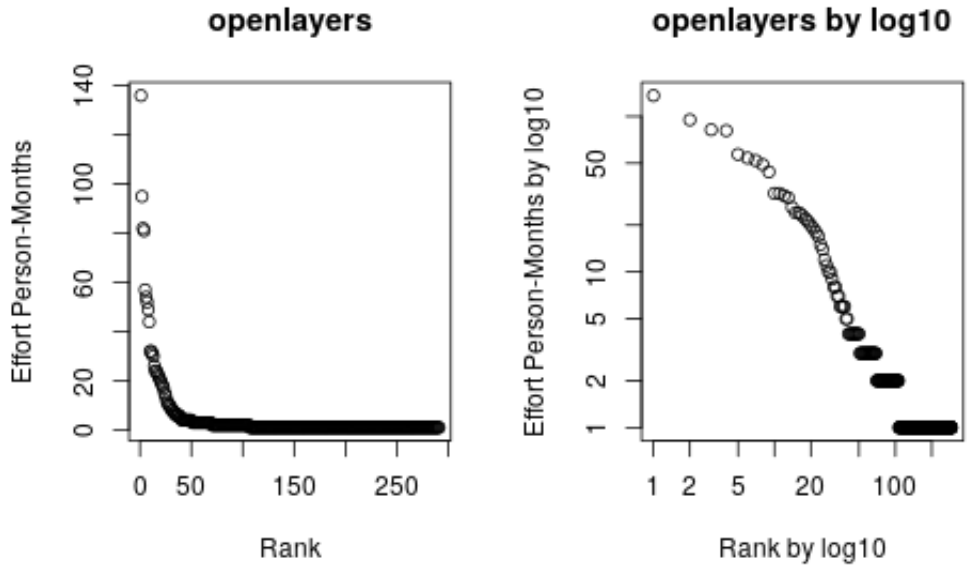

Fig. 5.1: Distribution of Effort Person-Months in the openlayers Project

Fig. 5.1 より，Contributor 毎の Effort Person-Months はべき分布であることが分かった．

Effort Person-Months の値を用いて，ジニ係数とローレンツ曲線を求めた．ジニ係数の値は，0.7217878 であった．Fig. 5.2 にローレンツ曲線を示す．

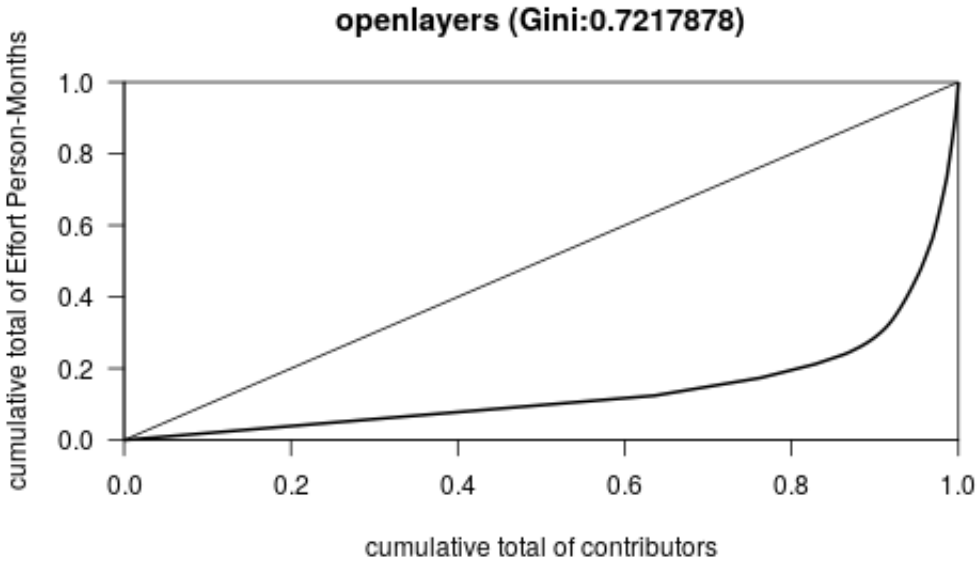

Fig. 5.2: Lorenz Curve of the openlayers Project

ローレンツ曲線の x 軸は，Contributor 単位での Effort Person-Months を昇順で累積したものである．Fig. 5.2 を見ると，全体の約 1 割の Contributor の活動量が多く，ばらつきが大きいことが分かった．

Fig. 5.1 から，活動量の分布はべき分布であり，Contributor には，べき分布の Head である継続して開発を行うコアメンバと，Tail に相当する活動量の少ないメンバが存在していることが分かる．

ジニ係数は，活動量のばらつきの程度を示しており，その値が大きいとばらつきが大きい．5.4.2 項で示した openlayers プロジェクトのジニ係数の値は 0.7223368 であり，活動量のばらつきが大きいことを示している．これにより，提案する方法で，活動量のばらつきの程度を計測できていると考える．

次節では，他の OSS 開発プロジェクトについて同様にジニ係数を求め，活動量のばらつきの程度が計測できるかを調べた．

5.5 OSS 開発プロジェクトにおけるジニ係数の計測

5.4 節では，1 つの OSS 開発プロジェクトを例にジニ係数の算出方法を中心に示した．本節では，他の OSS 開発プロジェクトにおいても同様に活動量の分布の程度が計測できるか確認する．48 件の OSS 開発プロジェクトの中から，参加人数を基に 3 件の OSS 開発プロジェクトを抽出した．分析対象は，48 件の OSS 開発プロジェクトの中から Contributor 数の 25 %，50 %，75 %に相当するプロジェクトを選定した．25 %は 5.4 節で示した openlayers プロジェクトであり，50 %は ”meteor” プロジェクト [115] ，75 %は ”llvm” プロジェクト [116] であった．本研究では，meteor プロジェクトと llvm プロジェクトにおける結果を示す．

5.5.1 meteor プロジェクトにおけるジニ係数の計測

meteor プロジェクトのジニ係数を算出する．

meteor プロジェクトは，リアルタイム Web アプリケーションフレームワークで，リアクティブな Web プログラミングを可能にしている．開発開始は 2006 年 5 月，Contributor 数は 476 人，Repository サイズは 77,585 KB，Fork 数は 5,028，Star 数は 40,706，Issue 数は 267 であった．

Effort Person-Months の分布を Fig. 5.3 に示す．

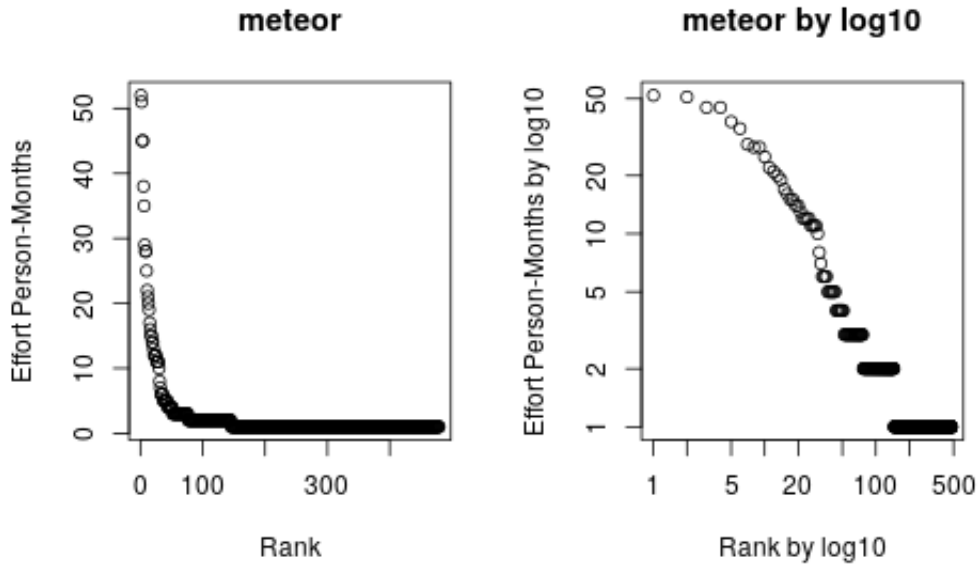

Fig. 5.3: Distribution of Effort Person-Months in the meteor Project

Fig. 5.3 より，openlayers プロジェクトと同様に，Contributor 毎の Effort Person-Months はべき分布であった．

続いて，ジニ係数とローレンツ曲線を求めた．ジニ係数の値は，0.5706287 であった．Fig. 5.4 にローレンツ曲線を示す．

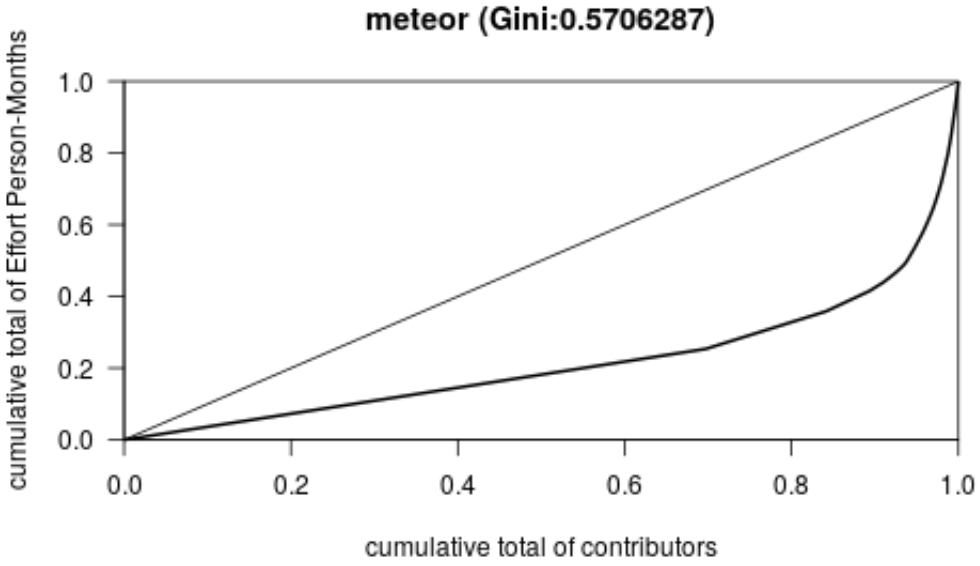

Fig. 5.4: Lorenz Curve of the meteor Project

Fig. 5.4 より，Contributor の活動量のばらつきは大きいことが分かった．

5.5.2 llvm プロジェクトにおけるジニ係数の計測

llvm プロジェクトのジニ係数を算出する．

llvm プロジェクトは，任意のプログラミング言語に対応可能なコンパイラ基盤である．開発開始は 2001 6 月，Contributor 数は 919 人，Repository サイズは 940,703 KB，Fork 数は 1,720，Star 数は 3,540，Issue 数 は 0 であった．

Effort Person-Months の分布を Fig. 5.5 に示す．

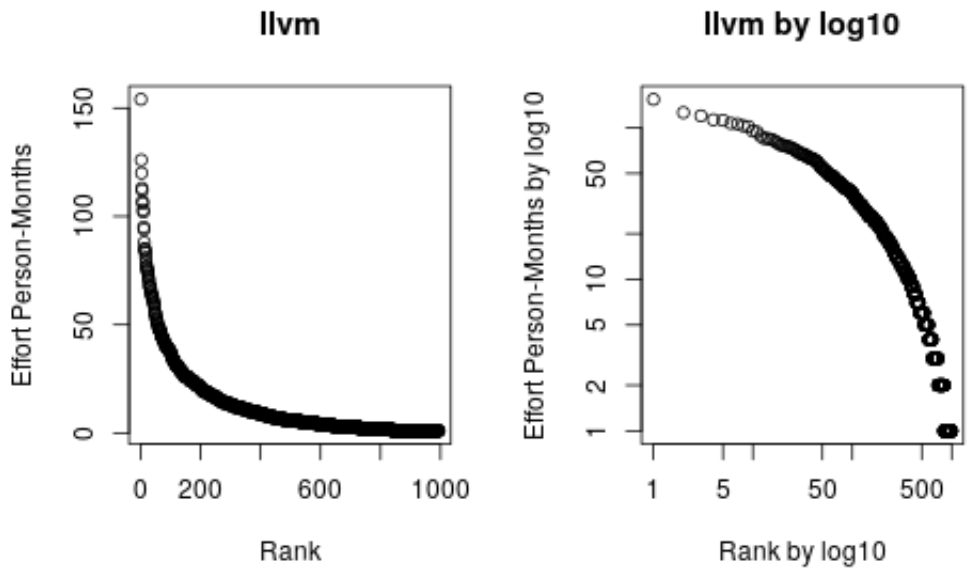

Fig. 5.5: Distribution of Effort Person-Months in the llvm Project

Fig. 5.5 より，前の 2 件と同様に Contributor 毎の Effort Person-Months はべき分布であった．

続いて，ジニ係数とローレンツ曲線を求めた．ジニ係数の値は，0.6091536 であった．Fig. 5.6 にローレンツ曲線を示す．

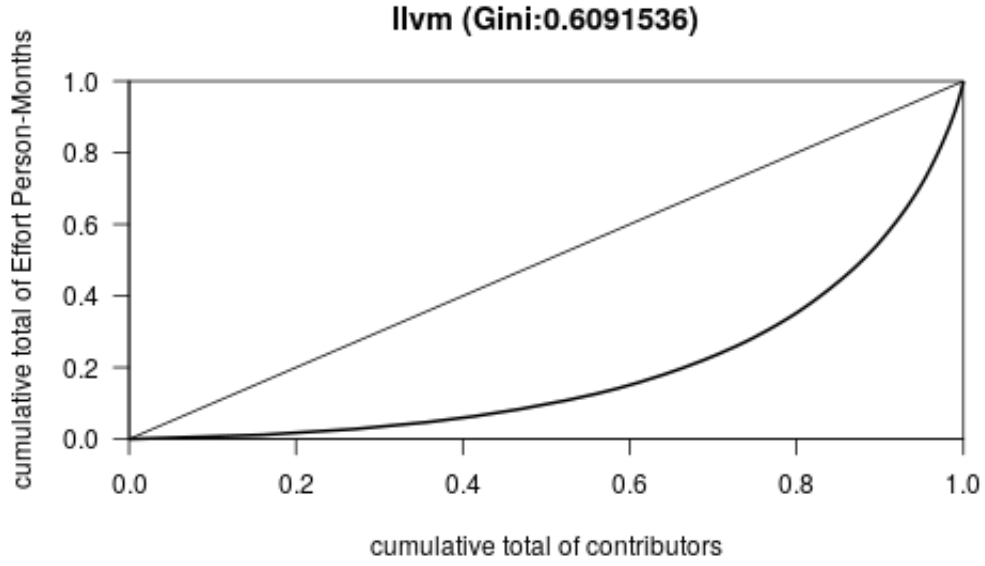

Fig. 5.6: Lorenz Curve of the llvm Project

Fig. 5.6 より，Contributor の活動量のばらつきは大きいことが分かった．

5.5.3 OSS 開発プロジェクトの計測値のまとめ

抽出した 3 件の OSS 開発プロジェクトにおいて，Effort Person-Months のジニ係数が算出できることが確認された．各 OSS 開発プロジェクトの計測値を Table 5.1 にまとめる．

Table 5.1: Data of 3 OSS Development Projects

Projects	Gini coefficient (value)	Period (year)	Contributors (number)	Effort Months (EM)	Commits (number)
openlayers	0.7223368	12	267	152	1,771
meteor	0.5708425	7	476	144	1,361
llvm	0.612601	17	919	2,181	15,171

Projects	Repository Size (KB)	Stars (number)	Forks (number)	Issues (number)
openlayers	77,614	4,453	1,778	620
meteor	77,585	40,706	5,028	267
llvm	940,703	3,540	1,720	0

次節では，48 件の OSS 開発プロジェクトにおけるジニ係数の値を比較する．

5.6 OSS 開発プロジェクト間の比較

5.5 節において，ジニ係数が算出できることが確認されたので，全 48 件の OSS 開発プロジェクトに対して展開した．本節では，48 件の OSS 開発プロジェクトのデータを用いて分析を行った．まず，ジニ係数が表す活動量の分布と Effort Person-Months のジニ係数と開発期間，Contributor 数，総 Effort Person-Months 数，総 Commit 数，Repository サイズ，Star 数，Fork 数，Issues 数との相関を確認した．次に，ジニ係数の値でグルーピングし，グループ間の差異から OSS 開発プロジェクトの特性を見いだせるかを試みた．

5.6.1 OSS 開発プロジェクトの開発活動状況

48 件の OSS 開発プロジェクトの各項目の箱ひげ図を Fig. 5.7 に示す．項目によっては大きな飛び値がある OSS 開発プロジェクトもあり，OSS 開発プロジェクト間のばらつきは大きい．

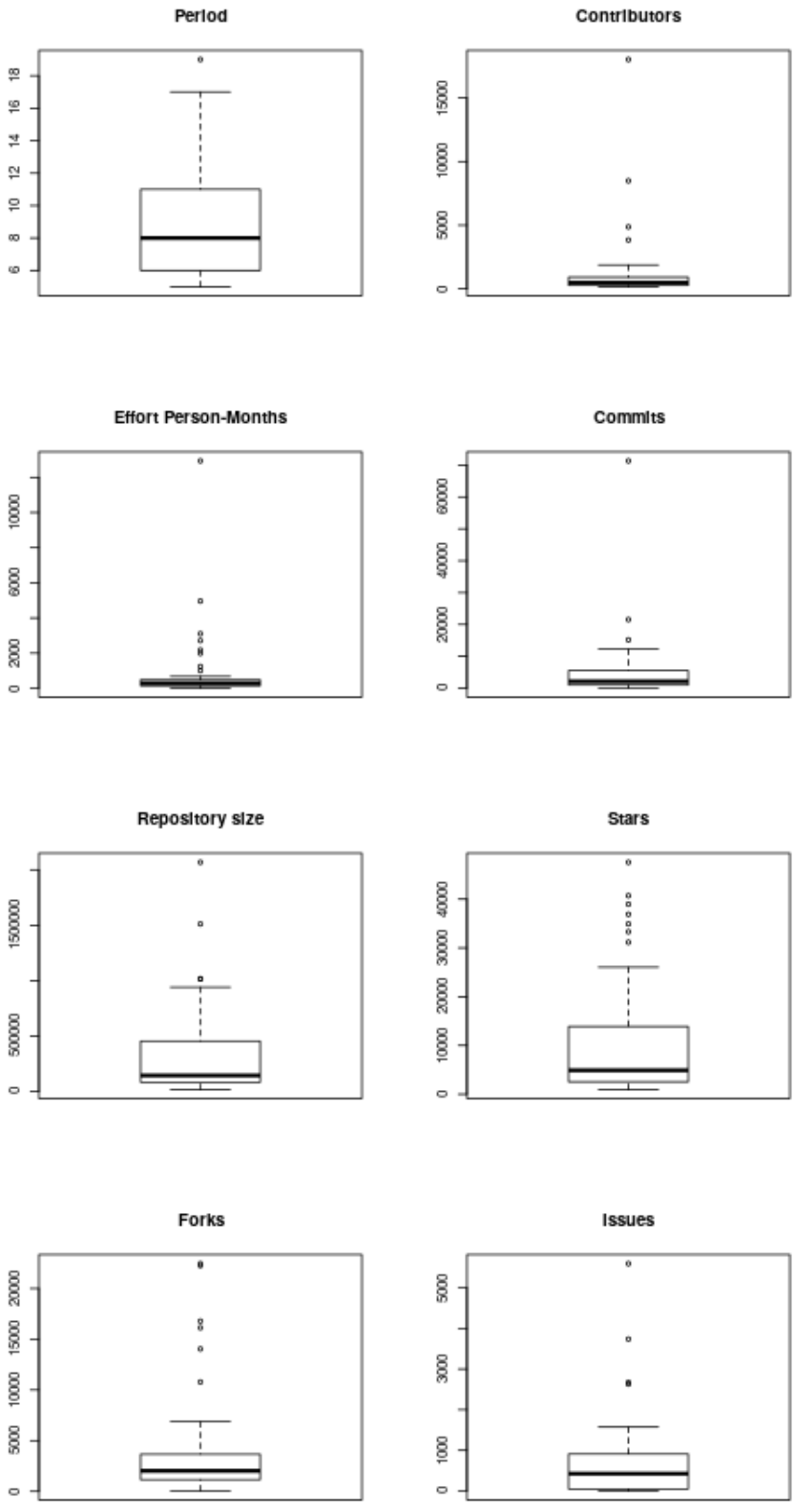

Fig. 5.7: Distributions of Each Item in All Projects

5.5 節において，ジニ係数が算出できることが確認されたので，全 48 件の OSS 開発プロジェクトに対して展開した．48 件の OSS 開発プロジェクトの計測値の分布を Table 5.2 に示す．

Table 5.2: The Gini coefficient of Effort Person-Months in All Projects

Projects	Gini coefficient
alluxio	0.456528295
ansible	0.397701759
atom	0.596716036
bokeh	0.551900801
bolt	0.52512242
bosh	0.568703593
canjs	0.615808125
Cataclysm-DDA	0.530086772
clang	0.646286998
collectd	0.520039031
conda	0.509424856
contiki	0.659731573
core	0.646599709
crystal	0.548875645
darktable	0.700263279
DefinitelyTyped	0.342616957
django	0.545401186
Firmware	0.608128626
frontend	0.574074397
gratipay.com	0.557206861
habitica	0.481409613
hazelcast	0.685803129
homebrew-cask	0.444498792
kotlin	0.703256817
libgdx	0.533597448
linux	0.698699508
llvm	0.609153563
lodash	0.392133205
meteor	0.57062871
mpv	0.697973527
neo4j	0.699829002
nikola	0.56474959
nixpkgs	0.6402569
opencv	0.529650395
openlayers	0.721787751
phpmyadmin	0.570580063
ppsspp	0.588737264
PrestaShop	0.573247902
presto	0.673951831
qemu	0.680393361
radare2	0.557233492
ReactiveCocoa	0.508363086
rethinkdb	0.682128814
RIOT	0.624580055
servo	0.559598784
spring-boot	0.545766758
vlc	0.726939227
yii2	0.478500387

求めた 48 件のジニ係数の分布を Fig. 5.8 左パネルに示す．図中の赤線は平均値 (Mean)，青線は第 2 四分位 (中央値：Median) を表している．Fig. 5.8 右パネルの Q-Q プロットは，データが正規分布かどうかの確認結果であり，散布図の点がほぼ直線上に並んでいれば，正規分布に従っていると考えられる．Fig. 5.8 右パネルの Q-Q プロットより，散布図の一部の点が直線から外れているが，ほぼ直線上に並んでいるため，正規分布に近い分布である．

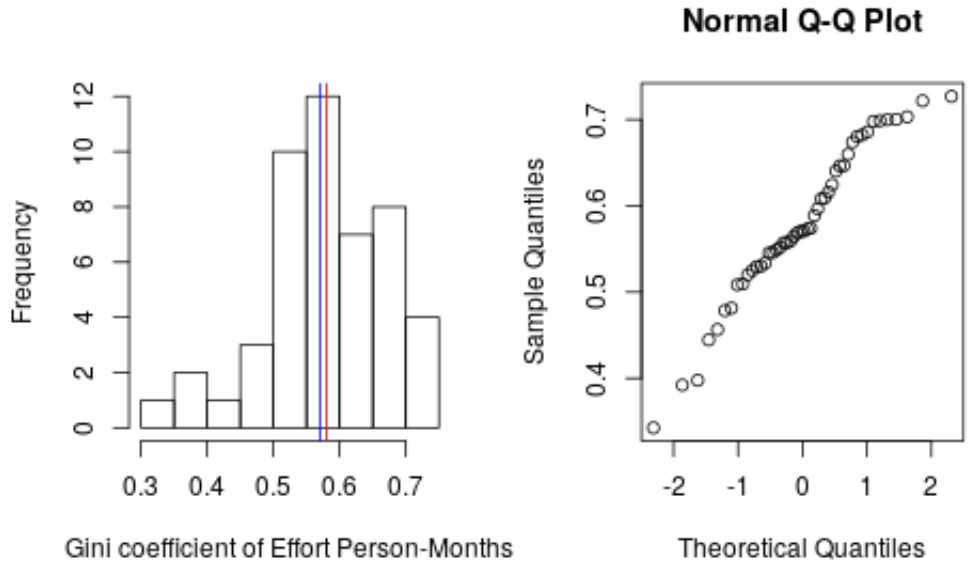

Fig. 5.8: Distribution of the Gini Coefficient of Effort Person-Months in All Projects

5.6.2 活動量の分布と各項目との相関

活動量のばらつきによる影響を調べるため，Effort Person-Months のジニ係数と開発期間，Contributor 数，総 Effort Person-Months 数，総 Commit 数，Repository サイズ，Star 数，Fork 数，Issues 数との相関を調べた．各項目との相関を Table 5.3. に示す．

Table 5.3: Correlation with Each Item

	Period	Contributors	Effort Months	Commits
EM. Gini	0.551619	-0.27996	-0.05911	0.125054

	Size	Stars	Forks	Issues
EM. Gini	0.201585	-0.33521	-0.3401	-0.13725

Table 5.3 より，開発期間は相関があるが，その他の項目との相関はほとんどないか，弱い相関があるという結果であった．

5.6.3 ジニ係数の差異による分類

ジニ係数の値から OSS 開発プロジェクトの特性を探ることができるか試みた．グループ間の比較を目的として，ジニ係数の値で分類することとした．ジニ係数が正規分布に近い分布であることから，ジニ係数の値を用いて 48 件の OSS 開発プロジェクトから 3 つのグループを抽出した．Fig. 5.8 左パネルにおいて，ジニ係数の値が上位の 4 件の OSS 開発プロジェクトをグループ A，平均値と第 2 四分位 (中央値) を含む 4 件の OSS 開発プロジェクトをグループ B，ジニ係数の値が下位の OSS 開発プロジェクトは，グループ A と同様に 4 件とし，グループ C とした．

グループ毎に各項目の箱ひげ図を Fig. 5.9，Fig. 5.10，Fig. 5.11 に示す．ジニ係数の値が大きいグループ A は，他のグループと比較して，開発期間が長い傾向にある．その他の項目については，そもそもグループ内のばらつきが大きく，グループ間の比較による OSS 開発プロジェクトの特性として有意な差異を見いだすことはできなかった．

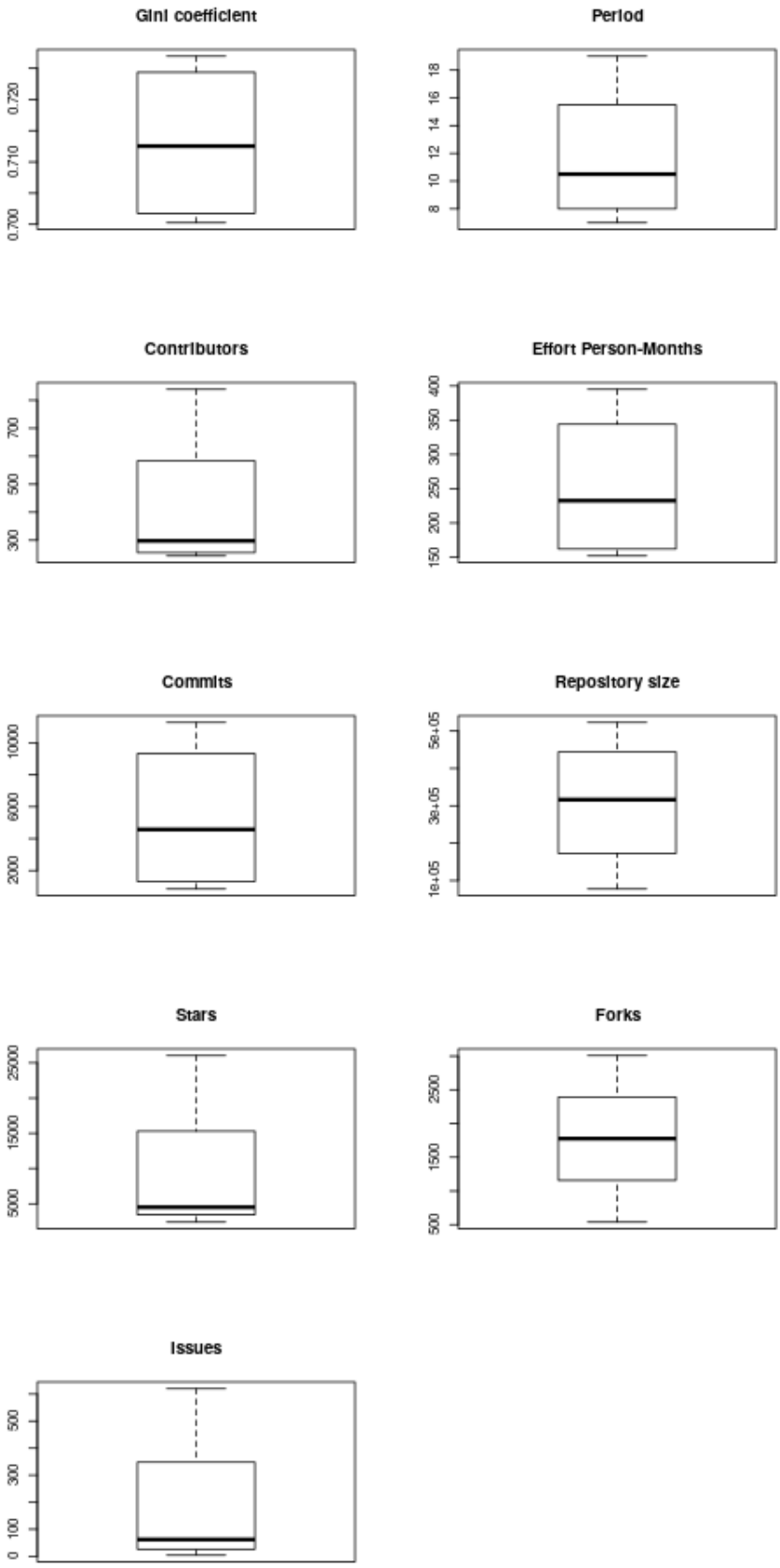

Fig. 5.9: Distribution of Each Item in Group A

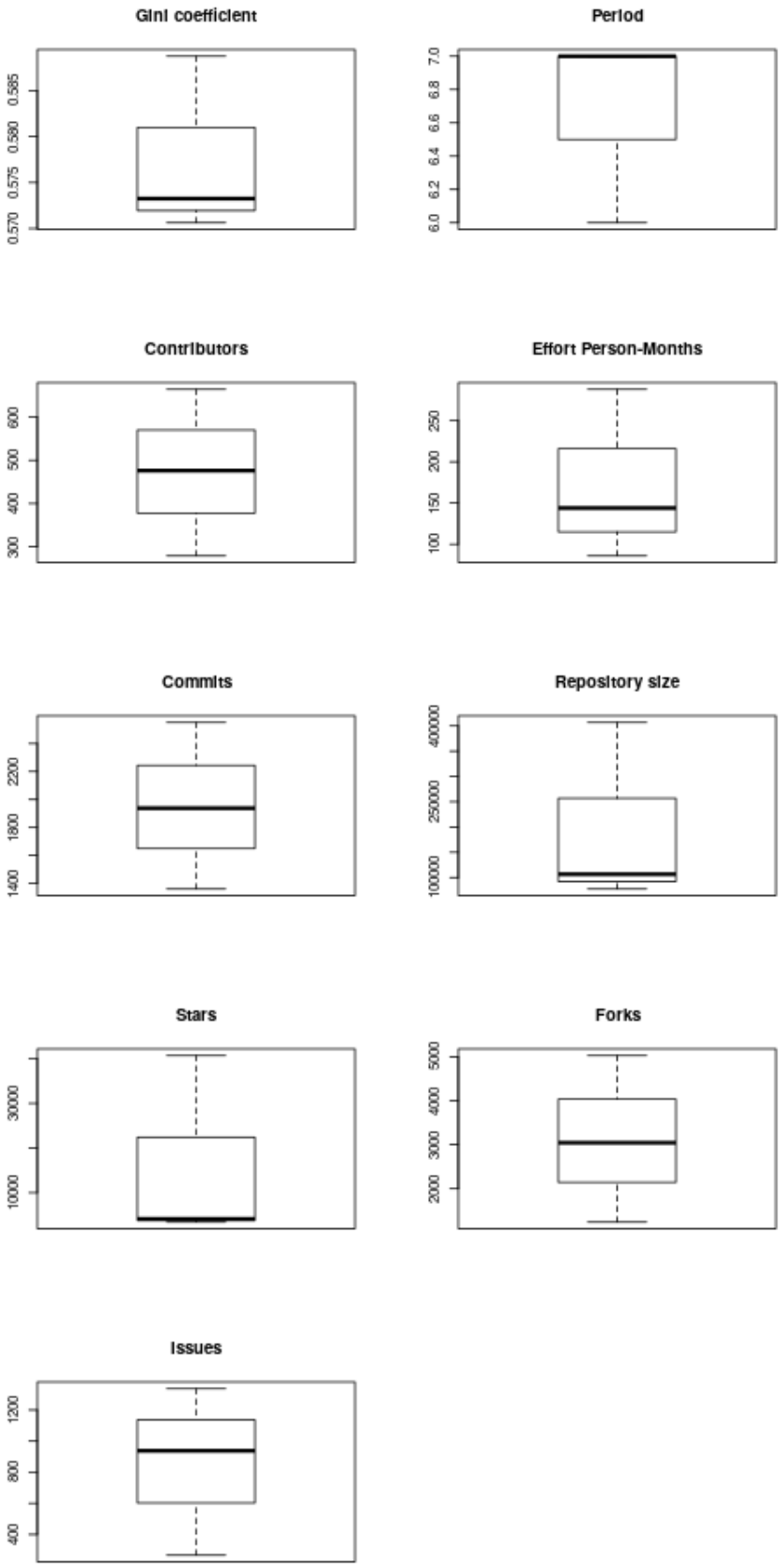

Fig. 5.10: Distribution of Each Item in Group B

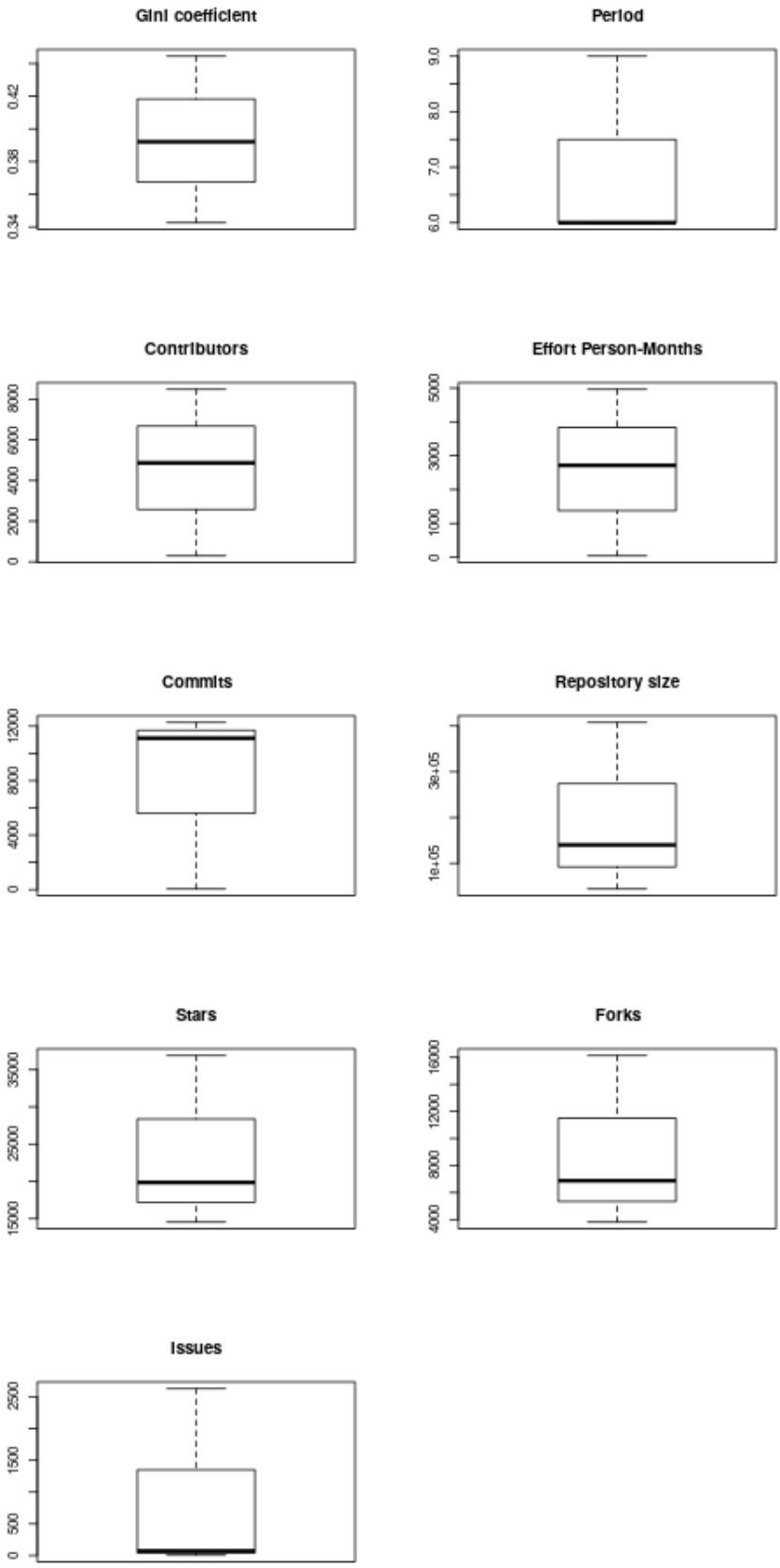

Fig. 5.11: Distribution of Each Item in Group C

5.7 計測期間の差異による比較

Fig. 5.1，Fig. 5.3，および Fig. 5.5 で示した通り，Contributor の活動量の分布はべき分布であった．べき分布の Head に相当する開発を推進する少数のコアメンバと，Tail に相当するビギナーなどの活動量が小規模な多数のメンバが混在していることが分かる．また，Table 5.3 より，活動量の大きなばらつきを示すジニ係数と，成果物に直結する Commit との相関は認められないことから，自由な参加による稼働率の低下の影響はほとんどないと推察される．

OSS 開発プロジェクトは，開発対象や，開発対象の難易度，OSS 開発プロジェクトの運営などの違いにより，多種多様である．多種多様な OSS 開発プロジェクトの中で，開発期間が長い OSS 開発プロジェクトはジニ係数が大きい傾向が観測された．本章の分析は，全開発期間を計測期間としている．Effort Person-Months は，計測期間において活動が記録されている月数をカウントしている．これは，全メンバが全期間いる前提として算出していることになる．そのため，計測期間内のある時点で限定的に活動を行い，その後，OSS 開発チームを離脱した Contributor であっても，全体の Effort Person-Months の人数としてカウントされていることになる．ここまでの分析では，計測期間全てを対象としていたため，離脱した Contributor の活動量も累積されている．これにより，開発期間が長い OSS 開発プロジェクトにおいてジニ係数が大きくなる傾向が観測されているのだと考えた．そこで，計測期間全てで算出した Effort Person-Months のジニ係数と，計測期間の最後の 1 年間だけで算出した Effort Person-Months のジニ係数を比較した．Fig. 5.12 に 48 件の OSS 開発プロジェクトにおける計測期間すべでで算出したジニ係数と計測期間の最後の 1 年間のみで計測したジニ係数の分布を示す．

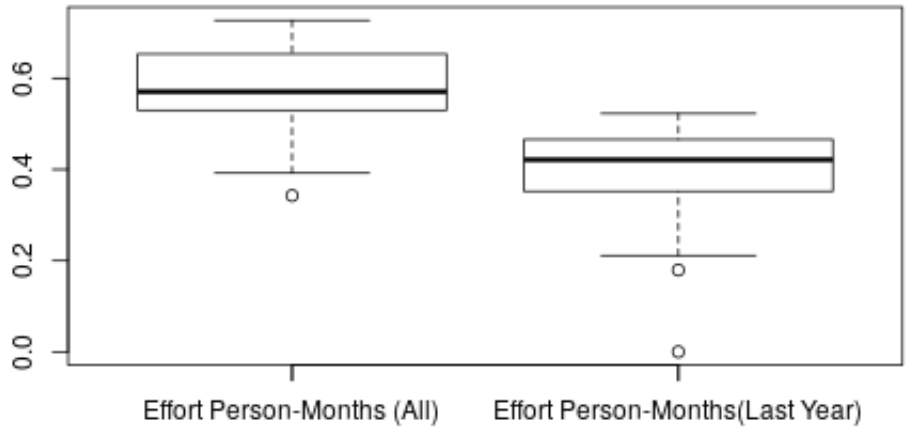

Fig. 5.12: Comparison of the Gini Coefficients for the Whole Period and the Last Year

計測期間の最後の 1 年間のジニ係数の値は全体のジニ係数の値よりも小さい値となっている．これは，全期間に比べて，1 回の Commit だけの Contributor など，活動量が少ない多くのメンバの累積が少ないためである．このジニ係数の差異は，OSS 開発プロジェクトに少しだけ参加して去って行った Contributor の稼働率を示していると考えることができる．OSS 開発プロジェクトの分析期間を区切ってジニ計数を算出し，時系列分析をすることにより，Contributor の活動状況を明らかにすることができると考える．

5.8 結言

本章では，OSS 開発プロジェクトにおける負荷分散に着目し，Contributor の活動量の計測を試みた．活動量を計測するために，計測期間において Commit 記録がある月を 1 とする Effort Person-Months を計測の単位とし，稼働率に相当する指標として経済学の分野で用いられるジニ係数を用いた．この手法により，実際の OSS 開発プロジェクトにおいて Effort Person-Months のジニ係数が算出できることを確認した．

活動量のばらつきによる影響を調べるため，Effort Person-Months のジニ係数と開発期間，Contributor 数，総 Effort Person-Months 数，総 Commit 数，Repository サイズ，Star 数，Fork 数，Issues 数との相関を調べたが，開発期間以外の項目との相関は認められなかった．活動量の大きなばらつきを示すジニ係数と成果物に直結する Commit との相関は認められないことから，自由な参加による稼働率の低下の影響はほとんどないと推察される．

次に，ジニ係数の値から OSS 開発プロジェクトの特性を探るため，グルーピングを行い，プロジェクト間での各項目の分布を確認した．開発期間が長い OSS 開発プロジェクトはジニ係数の値が大きい傾向にあったが，それ以外の項目において，活動量のジニ係数を用いて分類した OSS 開発プロジェクト間で有意な差異は見られなかった．

ジニ係数は，OSS 開発プロジェクトの開発期間との相関が認められたため，計測期間全てで算出した Effort Person-Months のジニ係数と，計測期間の最後の 1 年間だけで算出したジニ係数を比較した．比較の結果，最後の 1 年間のみで算出したジニ係数の値が小さいことが分かった．これは，活動量が少なく，活動期間の短い Contributor が多く存在していることを表している．OSS 開発プロジェクトへの参加や離脱はメンバの自由意志に基づくため，興味のある OSS 開発プロジェクトがあれば開発活動に参加することができる．これにより，OSS 開発プロジェクトに多くのメンバが関わることになる．OSS 開発チームは，開発を推進する少数のコアメンバがいる一方で，わずかな開発活動で OSS 開発チームを去って行く Contributor が多く存在している．参加や離脱が自由な OSS 開発チームの構成は，時間と共に常に変化していると考えられる．Fig. 5.12 で示した通り，時系列で比較した場合の差は顕著に表れている．この指標を用いて時系列分析を行うことで，OSS 開発チームの成長やチーム状態の変化などを分析することができるようになると考える．

第6章　結論

本研究は，ソフトウェア開発チームの活性化に影響を及ぼす要因を明らかにすることを目的に，活性化しているソフトウェア開発チームの分析を行った．関連研究の課題を踏まえて，開発チームのリーダとメンバの関係性に焦点を当て，企業における開発チームと OSS 開発チームを研究対象とし，次の 3 つを研究テーマとして設定した．

- 研究テーマ 1: 企業の開発チームにおけるチーム状態を計測する質問紙分析手法を探る
- 研究テーマ 2: OSS 開発チームにおけるメンバと開発量の関係を探る
- 研究テーマ 3: OSS 開発チームにおけるチームの状態の指標化を試行する

研究テーマ 1 の「企業の開発チームにおけるチーム状態を計測する質問紙分析手法を探る」では，開発チームのリーダが自身のチームの状態を知ることができるツールの検証と最適な分析手法を探索した．また，企業の開発チームのリーダが使うツールとしては，調査および分析手法は簡便であることが望ましい．そこで，複数の分析手法を比較し，最適な分析方法を探った．分析方法は，統計モデルと機械学習モデルを用いた．統計モデルでは，説明変数を少数の因子や主成分に集約して用いた．機械学習モデルでは，説明変数を集約せずに用いた．その上で，両モデルの判別率を比較した．

質問紙調査の分析手法の簡便さにおいては，分析過程で因子分析や主成分分析などの統計処理を行う必要がない機械学習モデルは，質問紙調査結果の値そのものを用いて判別ができるため，統計モデルと比較すると簡便であり，開発チームにおいて利用することができると考える．教師ありデータを用いて実施した統計モデルの自己検証では，松尾谷の研究 [10] で示された統計モデルの結果と同様の高い判別結果が得られた．機械学習における自己検証の判別率は，統計モデルとほぼ同等かそれ以上の高い判別率であった．一方で，妥当性検証として行った LOOCV による交差検証においては，統計モデル，機械学習モデル

共に外挿となるデータに対しては著しく判別性能が低下することを示した．開発チームのリーダが，自身のチームの状態を知るためのツールとして最適な分析方法は，機械学習モデルであると考えられる．しかし，実用化するためには，外挿における判別率の向上が求められる．研究テーマ 1 で試行した評価比較により，次の課題は，学習のための教師付きデータの蓄積，用いる変数の数や変数の加工なども含めた説明変数の改良であることが明らかとなった．

研究テーマ 2 の「OSS 開発チームにおけるメンバと開発量の関係を探る」では，OSS 開発チームがどのように開発を行っているのかについてメンバと開発量との関係の観点から調査し，分析を行った．GitHub における 50 件の OSS 開発プロジェクトを対象とし，OSS 開発に対して，企業のソフトウェア開発におけるコストモデルを応用して分析を行い，OSS 開発の特徴分析を試みた．分析には，言語や対象領域に制限を持たず，かつ実績のあるコストモデルである COCOMO を用いた．COCOMO の適用に当たり，企業のソフトウェア開発の工数 (Person-Months) に相当する概念を OSS 開発に適用するため，開発期間を 1 ヶ月単位で区切り，その区間内で 1 回以上の Commit 記録があれば 1 単位とする Effort Person-Months を定義し，回帰的に得られたべき式の係数に着目した計測方法を探った．

分析の結果，OSS 開発においても，COCOMO における量的開発効率の関係式が成り立つことが確認された．また，OSS 開発特有の特性として，開発規模が増加すると開発効率が上昇することが確認された．対象ソフトウェアに関する成果物の著作者毎の開発量の詳細な分析を行った結果，著作者の開発量の順位データは全てべき分布になっていた．そこで，べき分布の Head 部分に影響を及ぼしていると考えられる大規模な流用と Tail 部分に相当する貢献度の小さい多数の開発者の存在を特異値として除去して分析を行った．特異値を除去した評価でも，特異値除去前に見られた OSS 開発特有の特性に変化はなく，特異値として除去した大規模な流用や貢献度の小さい多数の開発者の存在が開発効率に影響を及ぼしていないことを示した．このことから，OSS 開発チームの実態としては，全体の開発量の 80 % が少数のコアメンバによって開発されていることが明らかとなった．OSS 開発チームでは，全開発量の 80 % が少数のメンバによって行われており，開発を推進するコアメンバの存在が確認された．

研究テーマ 3 の「OSS 開発チームにおけるチームの状態の指標化を試行する」では，研

究テーマ 2 の結果を踏まえて，メンバ間の開発活動量のばらつきから開発チームの状態の指標化を試みた．分析には，研究テーマ 2 で分析対象とした OSS 開発プロジェクトの Log データを用いた．活動量を計測するために，計測期間において Commit 記録がある月を 1 とする Effort Person-Months を計測の単位と定義し，総開発期間におけるメンバ毎の総 Effort Person-Months 数 (総活動月数) を基に稼働率を算出した．稼働率に相当する指標として経済学の分野で用いられるジニ係数とローレンツ曲線を用いた．

研究テーマ 3 で試行した手法により，実際の OSS 開発プロジェクトにおいて Effort Person-Months の分布を基に，ジニ係数とローレンツ曲線を用いて，OSS 開発チームの活動量の分布を定量化した．指標として用いたジニ係数の値から，OSS 開発プロジェクトチームにおけるメンバ間の活動量には大きなばらつきがあることが明らかになった．OSS 開発プロジェクトは自由な参加が推奨されている．そのため，すぐに貢献することができない新規参画者が多い場合には，活動量のばらつきは大きくなり，これに伴い成果物に直結する Commit 数にもばらつきが生じると考えた．しかし，活動量のばらつきを示す Effort Person-Months のジニ係数と成果物に直結する Commit との相関は認められなかった．このことから，自由な参加による稼働率の低下の影響はほとんどないと推察される．続いて，OSS 開発プロジェクトの特性を探るため，ジニ係数の値を基にグルーピングを行い，プロジェクト間での各項目の分布を確認したが，活動量のジニ係数を用いて分類した OSS 開発プロジェクト間には，有意な差異は見られなかった．さらに，異なる計測期間で算出した Effort Person-Months のジニ係数を比較した結果，計測期間全てで算出したジニ係数より，最後の 1 年間のみで算出したジニ係数の値が小さいことが分かった．これは，活動量が少なく，活動期間の短い Contributor が多く存在していることを表している．OSS 開発プロジェクトへの参加や離脱はメンバの自由意志に基づくため，興味のある OSS 開発プロジェクトがあれば開発活動に参加することができる．これにより，OSS 開発プロジェクトに多くのメンバが関わることになる．OSS 開発チームは，開発を推進する少数のコアメンバがいる一方で，わずかな開発活動で OSS 開発チームを去って行く Contributor が多く存在している．OSS 開発プロジェクトのジニ係数を時系列で比較した場合の差は顕著に表れており，参加や離脱が自由な OSS 開発チームのメンバ構成は，時間と共に常に変化していると考えられる．この指標を用いて時系列分析を行うことで，OSS 開発チームの成長やチーム状態の変化などを分析することができるようになると考える．

今後の検討課題として，次の 3 点が考えられる．

まず，企業の開発チームを対象とした研究の課題は，リーダが，自身のチームの活性状態を知ることができるツールとして利用することができるようにすることである．そのためには，学習のためのサンプルの蓄積が求められる．本研究で用いたデータは，20 件と 55 件であり，機械学習モデルで扱うにはサンプル数が非常に少なく，過学習の状態に陥っていると考える．サンプルを蓄積するためには，本質問紙を用いた調査を広く実施していくことが重要である．また，モデルのロバスト性を考えると，説明変数の数は少ない方が良いため，機械学習モデルで用いる説明変数についても変数の数や加工方法などの説明変数の改良が必要である．

次に，OSS 開発チームを対象とした研究の課題は，OSS 開発の高生産性因子の解明である．本研究では，大規模な流用が影響を及ぼしていると想定し，特異値を除去したモデルの評価を行い，影響していないことが確認された．しかし，小規模な流用や高生産性因子の分析など，影響を及ぼしている原因については，本研究で解明には至っていないため，継続した分析が必要である．

もうひとつの OSS 開発チームを対象とした研究の課題は，OSS 開発チームの成長やチーム状態の変化といった時系列分析である．本研究で用いた分析手法や指標を用いて OSS 開発チームの時系列分析を行うことで，OSS 開発チームがどのように成長していくのかを明らかにすることができると考えられる．

本研究では，チームの活性化に影響を及ぼす要因を明らかにすることを目的に，リーダとメンバの関係といった観点から，企業の開発チームと OSS 開発チームについて調査，分析を行った．

企業の開発チームにおいては，複数の分析手法を用いて分析を行い，自己検証においては，松尾谷の研究 [10] で示されている統計モデル結果と同様の結果が得られた．このことから，開発チームにおいて，プロジェクトの成否に大きな影響を与える因子は「仲間意識」と「役割認知」であると推察される．ただし，本研究で示したモデルは改良の余地があることが明らかになったため，今後改良したモデルで分析を行った上で，プロジェクトの成否に影響を及ぼす要因を特定する必要がある．

OSS 開発チームの実態調査では，OSS 開発チームにおける明確なリーダの存在は確認できなかった．一方で，開発量や活動量の分布からは，開発を推進する少数のコアメンバと

それ以外の多数のメンバという 2 つのグループに分かれていると読み解ける．このことから，OSS 開発チームにおけるリーダとは，コアメンバに相当すると考える．コアメンバは，自らの開発作業によって開発チームを牽引していく集団的なリーダであると言えよう．

本研究では，チーム形態が異なることから企業の開発チームと OSS 開発チームを分けて研究を行った．研究結果から開発チームのリーダに関して考察すると，リーダの位置づけそのものが全く異なるものであると考えられる．企業の開発チームにおけるリーダは，役割としてのリーダであり，メンバに指示をして開発を推進していくリーダである．OSS 開発チームにおけるリーダは，自ら開発をして進めていく集団的なリーダである．チーム形態が異なるだけではなく，リーダとしての位置づけも異なっている．そのため，企業の開発チームと，OSS 開発チームを単純に比較して考えることはできない．

本研究により，企業の開発チームと，OSS 開発チームのそれぞれにおいて，新たな知見を得ることができた．ソフトウェア開発チームの活性化に影響を及ぼす要因については追加分析が必要であるが，本研究は要因の特定に向けた一助になったと考える．今後も継続して両者の調査，分析を行い，開発チームの活性化について研究を行っていく所存である．

謝辞

本研究を遂行し，論文としてまとめるにあたり，多くのご支援とご指導を賜りました．

指導教官である筑波大学大学院 ビジネス科学研究科の津田和彦教授に心より深く感謝いたします．博士課程在学中，研究内容や進め方についてご指導いただくと共に，公私にわたって大変お世話になりました．先々まで見据えたご指導の中でいただいた多くのご助言をしっかりと肝に銘じ，今後も研究を邁進してまいります．副指導教官である筑波大学大学院 ビジネス科学研究科の吉田健一教授，木野泰伸准教授に深く感謝いたします．副指導教官をお引き受けいただき，研究内容や論文執筆にあたり的確なアドバイスをくださいました．筑波大学大学院 ビジネス科学研究科の教官の方々に深く感謝いたします．発表会の場などで，さまざまなアドバイスやコメントをくださいました．論文審査をお引き受けくださった覆面レフェリーの方々に，深く感謝いたします．重要かつ有用な多くのご指摘をいただきました．

博士課程進学の道を示し，背中を押してくださった有限会社デバッグ工学研究所の松尾谷徹氏に心より深く感謝いたします．研究の進め方や分析手法などについて，ご指導とご助言をいただくとともに，共同研究者としてさまざまな活動を行わせていただきました．オンラインやオフラインで重ねてきた多くの議論が，今の私の礎となっています．また，貴重な調査データをご提供くださいましたPS研究会のみなさまと，質問紙調査にご協力くださったみなさまに心より感謝いたします．東京工科大学の森本千佳子准教授に，心より御礼申し上げます．研究の心構えや研究内容について貴重なアドバイスをくださるとともに，「ゆるやかなつながりの勉強会 (ゆる勉) 」をご紹介くださいました．ゆる勉メンバのみなさまに，心より感謝いたします．みなさまの研究に対する真摯な姿勢から，多くの刺激をいただきました．帝京大学の藤田昌克教授に，厚く御礼申し上げます．津田研究室のゼミ等で，研究内容や英語での表現方法などについて貴重なご指摘を多々いただきました．

津田研究室の方々からは，さまざまな視点から有益な示唆やアドバイスをいただきました．研究室メンバからの示唆に富んだご意見やご指摘は，私にとって貴重な財産です．業務でお忙しい中，本論文のレビューをしてくださいました津田研究室の先輩博士である増田

聡氏，志田剛氏，植月啓次氏に，御礼申し上げます．有用なご指摘を多々いただきました．

システムズ・マネジメントコースの同期のメンバからは，研究に対する真摯な姿勢に刺激を受けるとともに，多くの励ましをいただきました．お互いの研究内容について議論をしたり，研究状況を共有したりする時間は，有意義でとても楽しく，研究を進めていく活力になりました．心より感謝いたします．

最後に，家族にとっては研究という”良く分からない世界” に熱中している私を温かく見守り，応援してくれた両親，弟家族，叔父，叔母に感謝します．大学院の合格を喜び，応援してくれた亡き父に報告できることを嬉しく思います．

みなさまからいただいた励ましの言葉は，研究を進めていく上で，大きな支えとなりました．この場を借りて，お世話になりましたみなさまに深く感謝いたします．ありがとうございました．

参考文献

[1] Facebook. https://www.facebook.com/ (accessed:Apr 03, 2019).

[2] Twitter. https://twitter.com/ (accessed:Apr 03, 2019).

[3] Frederick Winslow Taylor. *Scientific management*. Routledge, 2004.

[4] Abraham H Maslow. A theory of human motivation. *Psychological review*, Vol. 50, No. 4, p. 370, 1943.

[5] Douglas McGregor. Theory X and theory Y. *Organization theory*, Vol. 358, p. 374, 1960.

[6] Frederick I Herzberg. Work and the nature of man. 1966.

[7] Frederick I Herzberg. モティベーションとは何か. 4 月号, , 2003.

[8] Anteby Michel and Khurana Rakesh. The human Relations Movement: Harvard Business School and the Hawthorne Experiments (1924-1933). https://www.library.hbs.edu/hc/hawthorne/ (accessed:Mar 16, 2019).

[9] 木下瑞穂, 松尾谷徹. 情報プロジェクトにおける要員の組織行動とパートナー満足 (PS). 経営情報学会 全国研究発表大会要旨集 2002 年度秋季全国研究発表大会, pp. 32–32. 一般社団法人 経営情報学会, 2002.

[10] 松尾谷徹. IT に現場力は存在するのか: その計測と評価の試み. ソフトウェア・シンポジウム 2014 in 秋田 論文集, pp. 1–8. ソフトウェア技術者協会, 2014.

[11] Joab Jackson. IDC: Hobbyist programmers on the rise. https://www.itworld.com/article/2701410/idc--hobbyist-programmers-on-the-rise.html (accessed:May 01, 2019).

[12] Andrew W Halpin and B James Winer. A factorial study of the leader behavior descriptions. *Leader behavior: Its description and measurement*, pp. 39–51, 1957.

[13] C.L. Shartle. Early Years of the Ohio State University Leadership Studies. *Journal of Management*, Vol. 5, No. 2, pp. 127–131, 1979.

[14] Rensis Likert. New patterns of management. 1961.

[15] 三隅二不二. 新しいリーダーシップ. ダイヤモンド社, 1966.

[16] 三隅二不二. リーダーシップ行動の科学. 有斐閣, 1984.

[17] 三隅二不二. リーダーシップの科学-指導力の科学的診断法-講談社. 1986.

[18] Googlere:Work チーム. Google re:Work managers. https://rework.withgoogle.com/jp/subjects/managers/ (accessed:Mar 31, 2019), 2016.

[19] 松尾谷徹. パートナー満足によるソフトウェア生産性の向上. 第 20 回ソフトウェア生産における品質管理シンポジウム 2001. 一般財団法人 日本科学技術連盟, 2001.

[20] 松尾谷徹. パートナー満足 (PS) と人的リソースのパフォーマンス. プロジェクトマネジメント学会誌, Vol. 4, No. 1, pp. 3–8, 2002.

[21] 井沢澄雄, 松尾谷徹. PS 調査によるコミュニケーションとモチベーションの向上 (＜特集＞コミュニケーション・マネジメント). プロジェクトマネジメント学会誌, Vol. 4, No. 3, pp. 19–23, 2002.

[22] 込山俊博, 松尾谷徹. ソフトウェアプロセスにおけるコミュニケーションの側面とパートナー満足 (＜特集＞ コミュニケーション・マネジメント). プロジェクトマネジメント学会誌, Vol. 4, No. 3, pp. 24–27, 2002.

[23] 松尾谷徹. プロジェクトの成果を上げる原動力 7 つの要因を知りモチベーションを管理. 日経 IT プロフェッショナル 2003 年 8 月号, pp. 36–39, 2003.

[24] 松尾谷徹. IT-プロジェクトにおけるヒューマンファクタと組織行動の課題 (＜特集＞ヒューマンファクタのマネジメント). プロジェクトマネジメント学会誌, Vol. 6, No. 2, pp. 3–8, 2004.

[25] 原田奈美. チームメンバを動機付ける実践的アプローチ: ほめるとき・叱るときの MEH モデル (＜特集＞ ヒューマンファクタのマネジメント). プロジェクトマネジメント学会誌, Vol. 6, No. 2, pp. 12–18, 2004.

[26] 治田倫男. チーム運営におけるコミュニケーションの重要性: SI プロジェクトの事例研究 (＜特集＞ コミュニケーション・マネジメント). プロジェクトマネジメント学会誌, Vol. 7, No. 1, pp. 3–8, 2005.

[27] 榎田由紀子, 松尾谷徹. Happiness & active チームを構築する実践的アプローチ: チームビルディングスキルの開発 (＜特集＞ コミュニケーション・マネジメント). プロジェクトマネジメント学会誌, Vol. 7, No. 1, pp. 15–20, 2005.

[28] 森本千佳子. 若手メンバーのモチベーション向上に寄与するコミュニケーション (＜特集＞ コミュニケーション・マネジメント). プロジェクトマネジメント学会誌, Vol. 7, No. 1, pp. 21–25, 2005.

[29] 河村智行, 高野研一ほか. 情報システム開発の成否に影響を与える組織文化の要因の研究. 情報処理学会論文誌, Vol. 53, No. 12, pp. 2854–2864, 2012.

[30] 増田礼子, 森本千佳子, 松尾谷徹, 津田和彦. 質問紙回答からプロジェクトチーム指標構築アプローチの評価手法. プロジェクトマネジメント学会 2015 年度 春季研究発表大会予稿集, pp. 76–81. 一般社団法人 プロジェクトマネジメント学会, 2015.

[31] 増田礼子, 森本千佳子, 松尾谷徹, 津田和彦. チームビルディングのメトリクス ～ 10 年間の活動成果を測る. ソフトウェア・シンポジウム 2015 in 和歌山 論文集, pp. 134–140. ソフトウェア技術者協会, 2015.

[32] Ayako Masuda, Chikako Morimoto, Tohru Matsuodani, and Kazuhiko Tsuda. A Trial of Team Performance Evaluation and Team Learning Evaluation in Software Development. In *International Conference on Education, Psychology, and Social sciences 2015 Proceedings*, No. 0069, pp. 1–12, 2015.

[33] Ayako Masuda, Chikako Morimoto, Tohru Matsuodani, and Kazuhiko Tsuda. A Case Study of Team Learning Measurements from Groupware Utilization - A Proposal of Measurement Method for the Contribution Ration of Knowledge. In *8th International Conference on Computer Supported Education Proceedings*, Vol. 2, pp. 193–198, 2016.

[34] 増田礼子, 森本千佳子, 松尾谷徹, 津田和彦. チームの協働状態を測る:Team Contribution Ratio ～手間のかかる質問紙からの脱却～. ソフトウェア・シンポジウム 2016 in 米子論文集, pp. 121–127. ソフトウェア技術者協会, 2016.

[35] Ayako Masuda, Chikako Morimoto, Tohru Matsuodani, and Kazuhiko Tsuda. Construction of the Collaboration Skills Knowledge in Software Development. In *20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems Proceedings*, pp. 1129–1136, 2016.

[36] Googlere:Work チーム. Google re:Work team. https://rework.withgoogle.com/jp/subjects/teams/ (accessed:Mar 31, 2019), 2016.

[37] James Lee and Brent Ware. *Open Source Web Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP*. Addison-Wesley Professional, 2003.

[38] 伊原彰紀, 亀井靖高, 大平雅雄, 松本健一, 鵜林尚靖. OSS プロジェクトにおける開発者の活動量を用いたコミッター候補者予測. 電子情報通信学会論文誌 D, Vol. 95, No. 2, pp. 237–249, 2012.

[39] Chris Jensen and Walt Scacchi. Modeling recruitment and role migration processes in ossd projects. *ProSim05*, Vol. 39, , 2005.

[40] Chris Jensen and Walt Scacchi. Role migration and advancement processes in ossd projects: A comparative case study. In *Proceedings of the 29th international conference on Software Engineering*, pp. 364–374. IEEE Computer Society, 2007.

[41] Yuhong Tian. *Developing an open source software development process model using grounded theory*. University of Nebraska at Lincoln, 2006.

[42] André LS Guimarães, Helaine J Korn, Namchul Shin, and Alan B Eisner. The life cycle of open source software development communities. *Journal of Electronic Commerce Research*, Vol. 14, No. 2, p. 167, 2013.

[43] Edsger W Dijkstra. Structured programming, software engineering techniques. *NATO Science Committee*, p. 88, 1969.

[44] Ole-Johan Dahl, Edsger Wybe Dijkstra, and Charles Antony Richard Hoare. *Structured programming*. Academic Press Ltd., 1972.

[45] Barry W Boehm. Software engineering economics. *IEEE transactions on Software Engineering*, No. 1, pp. 4–21, 1984.

[46] 寺本雅則, 松尾谷徹, 上村松男. ソフトウェア開発過程を定量的に解析して生産性と品質を向上させるソフトウェアメトリクス. 日経エレクトロニクス 6 月 4 日号, 1984.

[47] 松尾谷徹. 開発コストのメトリクスの現状と課題. 昭和 63 年電気・情報関連学会連合大会予稿集, 1988.

[48] Barry Boehm, Chris Abts, A Winsor Brown, Sunita Chulani, Bradford K Clark, Ellis Horowitz, Ray Madachy, Donald J Reifer, and Bert Steece. Cost estimation with COCOMO II. *ed: Upper Saddle River, NJ: Prentice-Hall*, 2000.

[49] Barry Boehm, Chris Abts, and Sunita Chulani. Software development cost estimation approaches—A survey. *Annals of software engineering*, Vol. 10, No. 1-4, pp. 177–205, 2000.

[50] Barry W Boehm, Ray Madachy, Bert Steece, et al. *Software cost estimation with COCOMO II with CDROM*. Prentice Hall PTR, 2000.

[51] 尾高邦雄. 産業社会学講義: 日本的経営の革新. 岩波書店, 1981.

[52] 角山剛. 【第 16 回】職場モラール ～意欲・士気の研究～. https://www.hitachi-systems.com/report/specialist/psychology/16.html (accessed:August 25, 2019).

[53] Project Management Institute. プロジェクトマネジメント知識体系ガイド (PMBOK ガイド) 第 4 版. Project Management Institute, 2009.

[54] 角田雅照, 門田暁人, 宿久洋, 菊地奈穂美, 松本健一. 外部委託率に着目したソフトウェアプロジェクトの生産性分析. 信学技報, Vol. 106, No. 16, pp. 19–24, 2006.

[55] 山田茂, 福島利彦. 品質指向ソフトウェアマネジメント: 高品質ソフトウェア開発のためのプロジェクトマネジメント. 森北出版, 2007.

[56] 辻洋, 守安隆, 盛忠起ほか. オフショア・ソフトウェア開発の進化と技術者の経験知. 情報処理, Vol. 49, No. 5, pp. 551–557, 2008.

[57] 松村知子, 大平雅雄, 森崎修司, 松本健一. オフショア開発におけるユーザ・ベンダ間コミュニケーション情報の分析による仕様伝達の評価, 奈良先端科学技術大学院大学情報科学研究科テクニカルレポート. NAIST-ISTR2009005, 2009 年, Vol. 10, , 2009.

[58] 増田礼子, 松尾谷徹. 混成チームにおけるチーム力向上のための三者ヒアリング活用事例-チームビルディングはアイスブレイクだけではない-. ソフトウェア品質シンポジウム 2015. 一般財団法人 日本科学技術連盟, 2015.

[59] 秋口忠三. ソフトウェア工学分野の技術動向と展望. https://aiit.ac.jp/master_program/outlook/pdf/system01_akiguchi.pdf (accessed:Apr 01, 2019).

[60] 荒木啓二郎. ソフトウェア開発現場への形式手法導入. *SEC journal*, Vol. 6, No. 2, pp. 104–107, 2010.

[61] 栗田太郎ほか. フォーマルメソッドの新潮流: Part ii: 産業界への応用: 3. 携帯電話組込み用モバイル felica ic チップ開発における形式仕様記述手法の適用. 情報処理, Vol. 49, No. 5, pp. 506–513, 2008.

[62] 湯本剛, 植月啓次, 松尾谷徹, 津田和彦. データ共有タスク間の順序組合せテストケース抽出手法. 電気学会論文誌 C (電子・情報・システム部門誌), Vol. 137, No. 7, pp. 987–994, 2017.

[63] 藤垣裕子, 越河六郎. ソフトウェア開発作業における負荷要因. 産業医学, Vol. 34, No. 2, pp. 116–125, 1992.

[64] 古山恒夫, 菊地奈穂美, 安田守, 鶴保征城ほか. ソフトウェア開発プロジェクトの遂行に影響を与える要因の分析. 情報処理学会論文誌, Vol. 48, No. 8, pp. 2608–2619, 2007.

[65] 浜野康裕, 天嵜聡介, 水野修, 菊野亨. 相関ルールマイニングによるソフトウェア開発プロジェクト中のリスク要因の分析. コンピュータ ソフトウェア, Vol. 24, No. 2, pp. 2_79–2_87, 2007.

[66] 鎌田真由美, 細川宣啓, 渡辺千恵子ほか. 要求仕様書品質とプロジェクト成否の関連. 情報処理学会研究報告情報システムと社会環境 (IS), Vol. 2005, No. 115 (2005-IS-094), pp. 23–26, 2005.

[67] 岸知二, 細合晋太郎ほか. ソフトウェア工学の共通問題: 1. ソフトウェア工学の共通問題とは. 情報処理, Vol. 54, No. 9, pp. 878–881, 2013.

[68] 戸田航史, 角田雅照, 門田暁人, 松本健一. 工数見積もりモデルで予測できないソフトウェアプロジェクトの特徴分析. 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス, Vol. 105, No. 491, pp. 67–72, 2005.

[69] Tom DeMarco and Timothy Lister. Peopleware: Productive Projects and Teams. *Dorset House*, 1987.

[70] 遠藤功. 現場力を鍛える:「強い現場」 をつくる 7 つの条件. 東洋経済新報社, 2004.

[71] 遠藤功. 見える化: 強い企業をつくる 「見える」 仕組み. 東洋経済新報社, 2005.

[72] 谷古宇浩司. 業界の「現場力」はたった 30 点. http://www.atmarkit.co.jp/news/200607/20/mieru.html (accessed:Mar 13, 2019), 2006.

[73] 榎田由紀子, 松尾谷徹. チームビルディングスキルの開発事例 (＜特集＞ヒューマンファクタのマネジメント). プロジェクトマネジメント学会誌, Vol. 6, No. 2, pp. 9–11, 2004.

[74] 増田礼子. チームビルディングから組織文化へ -チームビルディング継続実施の効果-. ソフトウェア品質シンポジウム 2014. 一般財団法人 日本科学技術連盟, 2014.

[75] 山川紘明. 職種を超えた連携におけるチームビルディング適用とその効果評価. ソフトウェア・シンポジウム 2015 in 和歌山 論文集, pp. 11–19. ソフトウェア技術者協会, 2015.

[76] 竹林浩志. リーダーシップ研究におけるオハイオ研究の功罪. 観光学, Vol. 13, pp. 53–61, 2015.

[77] 小久保みどり. 2 つの主要なリーダーシップ理論の現代の企業への適用可能性. 立命館経営学, Vol. 41, No. 4, pp. 55–71, 2002.

[78] Linux. https://www.linux.com/ (accessed:Mar 01, 2019).

[79] Apache. https://www.apache.org/ (accessed:Mar 01, 2019).

[80] MySQL. https://www.mysql.com/ (accessed:Mar 01, 2019).

[81] PHP. http://php.net/ (accessed:Mar 01, 2019).

[82] Perl. https://www.perl.org/ (accessed:Mar 01, 2019).

[83] Python. https://www.python.org/ (accessed:Mar 01, 2019).

[84] Android. https://www.android.com/ (accessed:Mar 01, 2019).

[85] 野村佳秀, 木村功作, 福寄雅洋, 谷田英生. 『オープンソースソフトウェア工学』 シリーズ 企業における OSS 活用の実際. コンピュータ ソフトウェア, Vol. 33, No. 3, pp. 3_50–3_65, 2016.

[86] 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. 1_28–1_40, 2016.

[87] 眞鍋雄貴. チュートリアル 『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ライセンスへのソフトウェア工学からのアプローチ (サイバー増大号). コンピュータソフトウェア= Computer software, Vol. 34, No. 3, pp. 59–69, 2017.

[88] 竹田昌弘. オープンソース・ソフトウェアとビジネスとの関係に関する考察. 立命館経営学, Vol. 44, No. 3, pp. 49–66, 2005.

[89] Richard Stallman. *Free software, free society: Selected essays of Richard M. Stallman*. Lulu. com, 2002.

[90] Linus Torvalds and David Read By-Diamond. *Just for fun: The story of an accidental revolutionary*. Harper Audio, 2001.

[91] Eric Raymond. The cathedral and the bazaar. *Knowledge, Technology & Policy*, Vol. 12, No. 3, pp. 23–49, 1999.

[92] GitHub. https://github.com/ (accessed:Apr 03, 2019).

[93] Bitbucket. https://bitbucket.org/ (accessed:Apr 03, 2019).

[94] Ferdian Thung, Tegawende F Bissyande, David Lo, and Lingxiao Jiang. Network structure of social coding in github. In *2013 17th European Conference on Software Maintenance and Reengineering*, pp. 323–326. IEEE, 2013.

[95] Git. https://git-scm.com/ (accessed:Apr 03, 2019).

[96] Wiki. https://en.wikipedia.org/wiki/Wiki (accessed:Apr 03, 2019).

[97] Sourceforge. https://sourceforge.net/ (accessed:May 08, 2019).

[98] 北山聡. 組織内コミュニティの計量: ジニ係数とべき分布の視点から. 2009.

[99] Joseph L Gastwirth. The estimation of the Lorenz curve and Gini index. *The review of economics and statistics*, pp. 306–316, 1972.

[100] 中村和之. 経済指標の見方・使い方：所得格差を測る指標－ジニ係数とローレンツ曲線－. http://www.pref.toyama.jp/sections/1015/ecm/back/2005apr/shihyo/ (accessed:Jan 7, 2019).

[101] Rajiv M Dewan, Marshall L Freimer, Abraham Seidmann, and Jie Zhang. Web portals: Evidence and analysis of media concentration. *Journal of Management Information Systems*, Vol. 21, No. 2, pp. 181–199, 2004.

[102] Barry W Boehm, et al. Software engineering economics. *New York*, Vol. 197, , 1981.

[103] 西口利文小塩真司. 質問紙調査の手順. ナカニシヤ出版, 京都, pp. 1–131, 2007.

[104] 掘洋道. 吉田富士雄 (編): 心理測定尺度集 (2) 人間と社会のつながりをとらえる“対人関係・価値観”, 2001.

[105] 桑田喜隆, 石坂徹, 横山重俊, 合田憲人ほか. オープンソース・ソフトウエア・コミュニティのコストモデルと運用最適化に関する考察. *SIG-KSN*, Vol. 20, , 2017.

[106] Basic COCOMO.
https://en.wikipedia.org/wiki/COCOMO/ (accessed:Nov 27, 2017).

[107] Roger S Pressman, 西康晴, 榊原彰, 内藤裕史. 実践ソフトウェアエンジニアリング—ソフトウェアプロフェッショナルのための基本知識—, 2005.

[108] 佐藤達男ほか. 価値創造型の新しいプログラムマネジメントモデルの構築. 2014.

[109] GitHub: atom. https://github.com/atom/atom/ (accessed:Nov 27, 2017).

[110] GitHub: RIOT. https://github.com/RIOT-OS/RIOT/ (accessed:Nov 27, 2017).

[111] GitHub: The largest open source community in the world. https://github.com/open-source (accessed:Jan 20, 2019).

[112] Jin Xu, Yongqin Gao, Scott Christley, and Gregory Madey. A topological analysis of the open souce software development community. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pp. 198a–198a. IEEE, 2005.

[113] Minghui Zhou and Audris Mockus. What make long term contributors: Willingness and opportunity in OSS community. In *Proceedings of the 34th International Conference on Software Engineering*, pp. 518–528. IEEE Press, 2012.

[114] GitHub: openlayers. https://github.com/openlayers/openlayers (accessed:Jan 23, 2019).

[115] GitHub: meteor. https://github.com/meteor/meteor (accessed:Jan 23, 2019).

[116] GitHub: llvm. https://github.com/llvm-mirror/llvm (accessed:Jan 23, 2019).

関連業績リスト

学術論文

[1] 増田礼子，森本千佳子，松尾谷徹，津田和彦.「大規模オープンソース・ソフトウェアプロジェクトにおける開発効率の計測」，電気学会論文誌 C (電子・情報・システム部門誌)，Vol.138，No.8，pp.1011-1019，2018.

国際会議

[1] Ayako Masuda, Tohru Matsuodani, and Kazuhiko Tsuda. ”A Comparative Study Using Discriminant Analysis on a Questionnaire Survey Regarding Project Managers’ Cognition and Team Characteristics”. 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Vol.2, pp.643–648，2017.

[2] Ayako Masuda, Tohru Matsuodani, and Kazuhiko Tsuda. ”Team Activities Measurement Method for Open Source Software Development Using the Gini Coefficient”. 2019 IEEE International Conference on Software Testing, Verification and Validation (ICST), pp.140-147, 2019.

付録

質問紙 A

【プロジェクト名】

質問内容 以下の質問に対して1から4のうちどれかを選択して回答欄に記入下さい	回答欄
1. プロジェクトに参画していた時期はいつごろですか 1：10年以上前、2：10～5年前、3：5～2年前、4：2年前～現在進行	
2. プロジェクトのお客様との関係性を確認します	
(1)お客様側の統率力が強く、指揮命令系統が強く浸透している 1：強い否定、2：否定、3：肯定、4：強い肯定	
(2)お客様とSE側は比較的良好な関係にあると思う 1：強い否定、2：否定、3：肯定、4：強い肯定	
(3)お客様とSE側の利害関係の調整を行う余地がある 1：強い否定、2：否定、3：肯定、4：強い肯定	
(4)プロジェクト全体の方針が明確だと思う 1：強い否定、2：否定、3：肯定、4：強い肯定	
(5)プロジェクトの活動は統制のとれた計画や指示がなされている 1：強い否定、2：否定、3：肯定、4：強い肯定	
(6)メンバーの作業時間管理を厳格に行い、その成果は従事する時間によって評価される 1：強い否定、2：否定、3：肯定、4：強い肯定	
(7)メンバーの評価はその成果物に対する評価が優先され、作業時間では評価しない 1：強い否定、2：否定、3：肯定、4：強い肯定	
3. 仲間意識を確認します	
(1)作業について苦戦しているメンバーがいると、他のメンバーが支援している 1：強い否定、2：否定、3：肯定、4：強い肯定	
(2)仕事以外の雑談なども頻繁でメンバー間の非公式なコミュニケーションも多い 1：強い否定、2：否定、3：肯定、4：強い肯定	
(3)自分自身は今回一緒になったメンバーとまた作業をしたいと思っている 1：強い否定、2：否定、3：肯定、4：強い肯定	
(4)作業グループの中でメンバーどうしは頻繁なコミュニケーションがなされている 1：強い否定、2：否定、3：肯定、4：強い肯定	
(5)作業グループ間で頻繁なコミュニケーションがなされている 1：強い否定、2：否定、3：肯定、4：強い肯定	

質問紙 A

質問内容　　回答欄
以下の質問に対して1から4のうちどれかを選択して回答欄に記入下さい

4. 役割について確認します

（1）プロジェクトメンバー毎の作業分担が明確で、はっきり決まっている
　　1：強い否定、2：否定、3：肯定、4：強い肯定

（2）メンバーそれぞれの役割分担に対して公平感がある
　　1：強い否定、2：否定、3：肯定、4：強い肯定

（3）作業を計画した時に役割分担が漏れていることがある
　　1：強い否定、2：否定、3：肯定、4：強い肯定

（4）作業計画で規定した分担作業がなされず、漏れることがある
　　1：強い否定、2：否定、3：肯定、4：強い肯定

（4）メンバーは与えられた役割分担に熱意をもっている
　　1：強い否定、2：否定、3：肯定、4：強い肯定

5. プロジェクトの規範について確認します

（1）メンバーどうしで挨拶をすることがあたり前になっている
　　1：強い否定、2：否定、3：肯定、4：強い肯定

（2）メンバーは、緊急のトラブルがあったとき、役割を越えてプロジェクトのために動いている
　　1：強い否定、2：否定、3：肯定、4：強い肯定

（3）仕事をうまく進めようとして、暗黙のルールがありますか
　　（明文化してないが、積極的な情報開示方法がある、作業のきまりなど）
　　1：強い否定、2：否定、3：肯定、4：強い肯定

（4）質問の（3）で「ある」と答えた人は、その内容を記載してください。

・・・該当プロジェクトで存在した暗黙のルールを下に書いてください

6. 最後に

（1）そのプロジェクトは成功しましたか？
　　1：強い否定、2：否定、3：肯定、4：強い肯定

ご協力ありがとうございました

質問紙 B

TP(Team Performance)計測
簡易計測版

R1.1 2013-11-22
PMシンポジウム

この調査票は，現場力に関連する様々な特性に対し，弱みや強みを明らかにして，現場力を強化するための対策立案に使用します．匿名なので，忌憚の無い回答をお願いします．
簡易計測は，主に終了したプロジェクトを対象に，振り返って調査を行うものです．目的は，成功失敗とチーム力の構成要素との関係を明らかにすることです．調査結果は，回答者にWebで公開します．

質問内容	回答欄	Q

対象とするプロジェクトを思い浮かべ，そのプロジェクトの特性に関する質問

あなたがプロジェクトに参画していた時期はいついごろですか．

1：10年以上前， 2：9年～5年前， 3：4年～最近， 4：現在進行中 [ ] A

そのプロジェクトは成功しましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] B

そのプロジェクトにおけるあなたの役割．

1：メンバー， 2：チームリーダー， 3：PMとチームリーダ， 4：PM， 5：品質管理などスタッフ [ ] C

そのプロジェクトの規模について

1：100名以上， 2：～50名ほど， 3：～25名ほど， 4：～10名ほど， 5：9名以下 [ ] D

仲間意識に関する質問

あなたは，当時のメンバーと，また仕事をしたいと思ってますか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 1

仕事以外の雑談なども頻繁で，メンバー間の非公式なコミュニケーションもありましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 2

新規メンバーへの援助は良い方でしたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 3

メンバーは皆，仕事を達成しようとする意欲を感じましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 4

役割意識に関する質問

他者の役割分担について理解し，無理なくうまく機能していましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 5

役割分担で抜けが生じ，たらい回しになることはありましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 6

メンバーは自分の役割をよく理解していましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 7

役割で，スキル不足のメンバーのことが気になりましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 8

発生したトラブルなどに対し担当が決まらず先延ばしされることが良くありましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 9

リーダー役は，メンバーの状況を良く把握し仕事を調整していましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 10

あなたは自分の役割に熱意をもって対応していましたか．

1：否定， 2：どちらかと言えば否定， 3：判断できない， 4：どちらかと言えば肯定， 5：肯定 [ ] 11

裏面に続く

質問内容	回答欄	
規範やチーム文化に関する質問		
仕様書やマニュアルなど共有物は整備され活用されていましたか(オンラインを含む).		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		12
守らなければいけない無駄なレポートや約束事は多いと感じましたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		13
不席者の所で電話がなっていたら，だれかが素早く対応していましたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		14
メンバーどうしで挨拶をすることがあたり前になっていましたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		15
プロジェクトの進め方について良く説明をしたり，聞いたりしましたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		16
成果に関して		
メンバーのスキル以上に成果を出したと思いますか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		17
問題が発生した時，その解決はかなり早い方でしたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		18
技術力はとても高いと感じましたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		19
成果に貢献していないメンバーが多いと思いましたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		20
環境や仕事の難易度について		
仕事の技術的な要求レベルは高い方でしたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		21
仕事の納期はかなり厳しい方でしたか		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		22
仕事に要求される品質は高い方でしたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		23
不可能と思われる理不尽な要求がありましたか.		
1：否定，　2：どちらかと言えば否定，　3：判断できない，　4：どちらかと言えば肯定，　5：肯定		24

その他，コメント，ご意見がありましたら.

以上です. ありがとうございました.
分析結果は1週間後にWebに掲載しますので，この質問紙とIDを交換して下さい.