THE LINUX FOUNDATION | Research

オープンソース メンテナーズ

世界で最も重要なオープンソース ソフトウェア プロジェクトが 直面している人々、実践、 制約を探る

2023 年 7 月

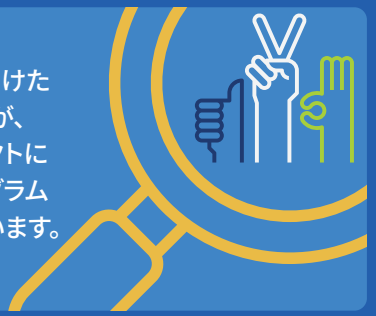
Alex Salkever, Linux Foundation Research

序文 Shuah Khan, Fellow, The Linux Foundation

オープンソース メンテナーズ

インタビューを受けた人の**75%**は、プロジェクトのメンテナーとコードのコントリビューターの両方を務めています。

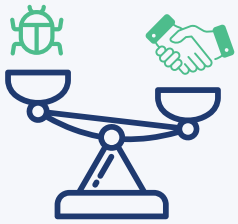
インタビューを受けた人の**1/3未満**が、彼らのプロジェクトに正式なDEIプログラムがあると答えています。



インタビューを受けた人の**38%**が、オープンソースの仕事に対して雇用主から高い支持を得ていると感じていると答えています。

インタビューを受けた人の**34%**が、自分のプロジェクトには正式なメンターシッププログラムがあると答えています。

コードの脆弱性に関する懸念は、新しいコントリビューターの採用など、運用上の重要な考慮事項とバランスをとる必要があります。



インタビューを受けた人の**53%**が、プロジェクトに正式な新規コントリビューターの採用プロセスがあると回答しています。

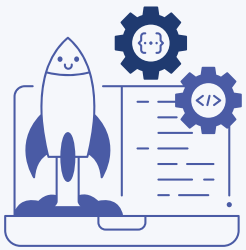
インタビューを受けた人のうち、自分のプロジェクトに強力な新しいコントリビューターパイプラインがあると答えたのは**わずか35%**でした。

インタビューを受けた人の**62%**は、プロジェクトにフルタイムで従事するために雇用されています。

インタビューを受けた人の**39%**が、オープンソースソフトウェアの仕事は自分たちの組織で高く評価されていると感じています。

インタビューを受けた人は、**プル リクエストにタイムリーに応答しない**など、特定の行動が新しいコントリビューターを落胆させることを認識しています。

自動化を採用しましょう。
人の手に取って代わるものではありませんが、拡張性や迅速な応答時間などの重要なメリットを提供します。



ソーシャル メディアでの言及や感謝の表明、コミュニティ コールは、新しいコントリビューターが困難な問題に取り組むことを奨励することができます。

目次

序文

オープンソース ソフトウェアは、私たちの世界のインフラストラクチャのバックボーンです。金融サービスや医療から、通信やインターネットに至るまで、オープンソースが世界を動かしていることは周知の事実です。

それはスマートフォンやノートパソコン、通信やデータのインフラ、物流システム、そして政府を動かしています。Linux や MySQL から、フロントエンド フレームワークからデータベース、コンテナ オーケストレーション、テレメトリーに至るまでの何千もの実行可能なプロジェクトまで、オープンソース ソフトウェアはどこにでもあります。コードがそれ自体を記述しないことは誰もが知っています。成功するオープンソース プロジェクトの中核には、メンテナーと主要なコントリビューターがいます。

コードの最初の行の設計と記述から、Linux や Kubernetes などの大規模なオープンソース コミュニティの監督と育成に至るまで、メンテナーとコントリビューターは、イノベーション、進歩、健全なエコシステムを推進します。彼らは、健全な自己統治とコミュニティの継続性を強化するためのルールとトーンを設定したり、プロセスを設定したりします。メンテナーとコントリビューターがいなければ、オープンソース ソフトウェアはすぐに時代遅れになり、不安定になります。重要なソフトウェア プロジェクトでは、負担が大きくなります。

比較的少数のオープンソース プロジェクトが、今日使用されているソフトウェアの驚くほど高い割合を占めています。これらの重要なプロジェクトについてもっと理解し、それらを実行する人たちの教訓を捉えることが不可欠です。彼らの教訓は、将来の創業者やメンテナーに重要なプロジェクトの情報を提供することができます。そして、オープンソースを広く持続可能なものにすることで、成功するオープンソース プロジェクトのプールを増やし、メンテナーシップをより充実させ、負担を軽減できると信じています。

オープンソースは常に、リスクを冒してプロジェクトを実現する人々のためのものであり、これからもそうあります。オープンソースの成功に捧げられた財団として、私たちはメンテナーに対して、彼らの生活をより良くし、より良い道を明らかにする義務があります。このレポートが、最も重要なプロジェクトから最も成功したメンテナーの知恵と教訓を記録し、すべての人が利用できるようにすることで、この取り組みに貢献できることを願っています。

敬具
Shuah Khan
LF Fellow / Maintainer
The Linux Project

要約

オペレーティング システムからデータベース、プログラミング言語に至るまで、世界はますますオープンソース ソフトウェア(OSS)に依存するようになっています。OSS は、グローバルなテクノロジー インフラストラクチャの多くの基盤となっています。エンジニアが、明示的にコードから利益を得ようとする企業のために、閉鎖的な環境で作業するときに構築するプロプライエタリ ソフトウェアとは異なり、OSS は、ソフトウェアを構築するために、プロジェクトの初期段階で、しばしば無給で働くメンテナーのウェブに依存しています。調査によると、私たちのテクノロジー インフラストラクチャは、数百のオープンソース プロジェクトに大きく依存しています。これらのプロジェクトは、ソフトウェア依存関係の不均衡な割合を占めています。これらのプロジェクトのメンテナーは、多大な負担を負っています。彼らが監督し、管理するプロジェクトは、世界経済の多くに責任を負っています。これらのプロジェクトの中断は、大規模な問題や停止を引き起こす可能性があります。

場合によっては、これらのメンテナーは、プロジェクトに対する組織的または財政的な支援をほとんど、あるいはまったく受けずに、単独で作業することもあります。メンテナーが、給与の一部としての明示的な義務を含む、重要な組織的および財政的支援を享受している場合でさえ、最も重要な OSS を維持する仕事、スーパー メンテナーの仕事を定量化し、明らかにすることは依然として困難です。このレポートでは、メンテナーがどのようにしてメンテナーになるのか、成功する OSS プロジェクトを成長させるための経験と観察、ソフトウェア コミュニティを成長させ、充実した十分な報酬のある生活を送るための要件のバランスをとるためのツールとベストプラクティスを文書化することを目指しています。

方法

このレポートのために、LF Research は、私たちの調査で重要な OSS プロジェクトのトップ 200 に含まれていると特定されたプロジェクトの 32 人のスーパー メンテナーに詳細な定性的インタビューを実施しました。メンテナーはさまざまな職歴の出身ですが、多くの感情と共通の経験を共有していました。

プロジェクトの規模と複雑さが増すにつれて、メンテナーは時間の要求の増加に直面します。これにより、新しいコントリビューターを歓迎する努力が少なくなりました。初期のコントリビューターは、競争が少なく、プロジェクトの創設者と関わる機会が増えたことから恩恵を受けました。より良いコントリビューターの経験を維持するための協調的な努力にもかかわらず、メンテナーは、時間の経過とともに悪化していることを恐れています。彼らは一貫して、新しいコントリビューターのための支援環境の育成に継続的に焦点を当てることの必要性を強調しました。これは、新しいメンテナーを引き付けることがより困難な成熟したプロジェクトや、より技術的な洞察力（一例としてメモリ管理）を必要とするプロジェクトの一部にとって特に重要です。

LF Research によるこれまでの複数の調査で観察されたように、メンテナーは、コミュニティーと仲間意識、最先端技術への取り組み、独自のコースを設定して活動に優先順位を付ける力、コミュニティが形成されるのを見る達成感など、オープンソースに取り組むことの本質的な報酬を享受していました。メンテナーは、キャリアの軌跡、地位、コミュニティでの尊敬、および別の仕事も可能であるというキャリアの自信において、外因的に利益を得ていることを認識していました。メンテナーの多くは、コミュニティの仕事のために現在の雇用を得ました。場合によっては、企業がメンテナーを雇ったのは、特に彼らの専門知識とプロジェクトのダイナミクスに影響を与える能力のためでした。しかし、ほとんどのメンテナーは、自分たちの組織が彼らの努力を十分に認識していないことに懸念を表明しました。

コントリビューションの拡大

インタビューを受けたメンテナーは、個人的なコントリビューターの関与の優先順位付けと時間の確保、包括的な言語の使用、関与のための複数のコミュニケーション チャネルの提供、明確なオンボーディング リソースの提供など、コミュニティを育成し成長させるための幅広いベストプラクティスと知恵を提供しました。成功したプロジェクトは、GitHub のフラグを通じて、またはコミュニティ チャネルに寄せられた質問への応答として、適切なバグやプル リクエスト（PR）を提案することで、初めてのコントリビューターをさらにサポートしました。メンテナーは常に、より高い能力や持続性のレベルを持つ人を特定し、次世代のスーパー メンテナーやプロジェクト リードを構築するために育成する必要があります。新しい PR のトリアージ プロセスを確立し、コミットと PR をタイムリーに処理するためのチームの取り組みを組織することで、コントリビューターの経験がさらに向上します。

要約

ガバナンスとコントロール

インタビューを受けたすべての人は、コミュニティのガバナンスと管理はプロジェクトの長期的な成功に不可欠であるが、初期段階では見過ごされることが多いと感じていました。ベストプラクティスには、行動規範を確立し、明示的にも暗黙的にも(行動とプロジェクト リーダーシップのトーンを通じて）礼儀正しさを促進することが含まれます。権力を分散させることで、コントリビューターを苛立たせ、コミュニティのイノベーションを遅らせる意思決定のボトルネックを防ぐことができます。コミュニティ管理における中立性は、すべてのコントリビューターの公正な扱いを保証し、複数の組織からの開発者の参加を奨励します。

ドキュメンテーション

当然のことながら、メンテナーは、彼らのプロジェクトがドキュメントを改善する必要があるという懸念を表明しました。メンテナーが提案したベストプラクティスには、プロジェクト リーダーが、ドキュメントがコード コントリビューションと同等の認識を持つプロジェクトの第一級市民であることを実証すること、ドキュメント コーディネーターを雇用すること、コード コントリビューションごとにドキュメントを提出することを要求すること、貢献を容易にするためにドキュメントに関する正式なイベントやプロセスを作成することなどが含まれていました。

資金調達

生活するのに十分なお金を稼ぐことが懸念されたのは、インタビューを受けた人のうち 1 人だけでした（彼は、JavaScript コミュニティ内の提携していない小規模なプロジェクトにも従事していました)。とは言っても、複数のメンテナーが、彼らが維持しているか知っている重要なオープンソース プロジェクトが停滞しているか、資金サポートがないために新しいバージョンを出荷していないことに不満を表明しました（インタビューを受けた多くのメンテナーは、雇用責任の一部ではないものも含めて、複数のプロジェクトを維持しています)。中小規模の重要なプロジェクトに対する資金メカニズムが不十分であることが、メンテナーが仕事を引き受ける意思のある組織でフルタイムの雇用を求める決定の重要な要因でした。インタビューを受けた一握りの独立したメンテナーのうち、全員が、財団や大規模な企業資金提供者を持たないオープンソース プロジェクトを維持する方法について懸念を表明しました。

多様性

ほとんどのメンテナーは、十分に多様なコントリビューターとプロジェクト リーダーを生み出すのに苦労しました。メンテナーの約半数は、明確な多様性の取り組みや目標を持っていませんでした。Outreachy などの多様性プログラムに参加したのはほんの一握りでした。多様性の育成に成功したメンテナーは、DEI の目標を、関連するガバナンスの取り組みとともにプロジェクトのトップレベルの目標にし、多様性プログラムに参加しました。最善の意図にもかかわらず、オープンソースは多様性に関して長い道のりを歩んでいます。

燃え尽き症候群の防止

メンテナーは、オープンソース メンテナーが燃え尽き症候群を防ぐために使用するさまざまな戦略とプラクティスについて議論しました。これらのプラクティスには、オープンソースの作業が決して終わらないことを認識すること、仕事と個人的な追求のバランスをとるライフスタイルを設計すること、過剰な管理作業を必要とする無給のプロジェクトを引き受けないこと、効率性を高めるためにワークフローを自動化すること、コミュニケーションと労働時間に関して境界を設定すること、燃え尽きたと感じたときに休憩を取ることなどが含まれます。これらの戦略の多くは、オープンソース プロジェクトに貢献しながら燃え尽き症候群を防ぐために、自己認識を育み、個人的な制限について現実的であることを含んでいます。

はじめに

OSS エコシステムの健全性は、最も積極的で責任あるメンテナーとコントリビューターからなるコア グループの質と成功に依存しています。オープンソース エコシステム、ひいては世界の技術インフラは、数百に及ぶ比較的少数のプロジェクトに大きく依存しています。LF Research は、Census II プログラム（最近では “’ Vulnerabilities in the Core,’ a Preliminary Report and Census II of Open Source Software” と “Census II of Free and Open Source Software—Application Libraries”）でハーバード大学と共同で進行中の作業において、最も重要なプロジェクトを依存関係の観点から分類することを研究し、試みてきました。何百万ものオープンソース プロジェクトの中で、プロジェクトとそのメンテナーとコントリビューターの重要なグループが大きな役割を占めています。LF Research は、これらのオープンソース領域のリーダーを”スーパー メンテナー”と”スーパー コントリビューター”と呼び、彼らは集合的に”スーパー コーダー”のグループを形成しています。

彼らの負担は重いです。世界中の何百万人ものユーザーとシステムが、彼らが監督するプロジェクトをダウンロードしています。彼らのプロジェクトのセキュリティの脆弱性は、大規模な世界的な混乱を引き起こす可能性があります。彼らのコードの脆弱性は”インターネットを破壊する”可能性があります。彼らはまた、これらの懸念と、より日常的ではあるが同様に重要な運用上の考慮事項とのバランスをとる必要があります。例えば、常に新しいコントリビューターを採用し、プロジェクト間の紛争に対する適切なガバナンスと裁定のシステムを設定し、より高いレベルの多様性を達成する必要性などです。同時に、彼らのプロジェクトが革新を続け、新しい技術サイクルを繰り返し続けることを保証します。

この調査は、世界で最も重要な OSS プロジェクトの 30 人以上のメンテナーとコア コントリビューターへのインタビューの結果であり、その多くは、Linux Foundation が Harvard University の Laboratory of Innovation Science と共同で発表した Census II レポートで特定された、最も広く使用されているアプリケーション ライブラリの 1 つです。

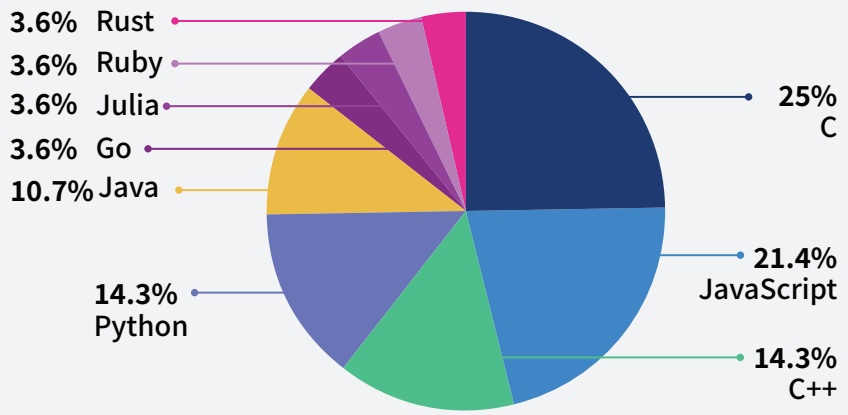
インタビューでは、最も重要なオープンソース プロジェクトを指揮する人々の洞察と知恵を収集します。インタビュー対象者は、オープンソース プロジェクト内で効果的に開始、拡張、管理、および革新する方法を説明し、オープンソース コミュニティと企業が同様に重要な作業をよりよくサポートする機会を特定します。

方法

LF Research は、重要なプロジェクトのスーパー コーダーに対して、約 1 時間にわたって 32 回のインタビューを実施しました。この調査のインタビュー対象者は、広大なオープンソース エコシステムのさまざまな役割とニッチを占める多様なソフトウェアのコントリビューターまたはメンテナーでした。これには、フロントエンド ライブラリ（Babel、Webpack、React、Storybook）、オペレーティング システム（Linux）、インフラストラクチャ（containerd、Kubernetes）、パッケージ リポジトリとマネージャ（Rust Cargo、npm、RubyGems、Gradle、Apache Maven/Maven Central）、データベースとストレージ（PostgreSQL、Ceph）、開発者ツール（LLVM、Git、cURL）、DevOps（Salt）、低レベル言語(Julia)、データ分析と機械学習のフレームワークとアプリケーション（PyTorch、NumPy、Jupyter）が含まれます。

図 1

調査対象プロジェクトの言語



LF Research には、数年から数十年までのさまざまな成熟度と年齢のプロジェクトが意図的に含まれており、1 人のメンテナーから数千人のコントリビューターと数十人のメンテナーを持つプロジェクトまでの規模が含まれていました。最後に、LF Research には、C/C++、Go、Rust、JavaScript、Java、Node.js、Python、Ruby、Julia など、複数の言語を使用するプロジェクトが含まれていました。そうすることで、LF Research は、これら 3 つのコア パラメータに基づいて、共通のパターン、課題、ベストプラクティスを特定しました。

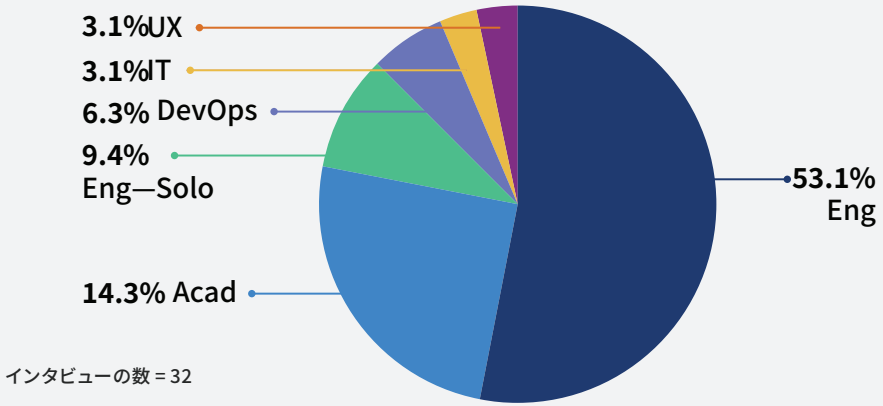
無料でダウンロードできる OSS プログラムは何百万もあり、主要なバージョン管理ツールやプログラミング コラボレーション ツール（GitHub、GitLab、Bitbucket）、Web サーバ、その他の場所など、複数の場所でホストされています。Census レポートのデータは、最も広く使用されている OSS について、限定的ではありますが重要な見解を提供しています。Census II によって評価されたソフトウェア プロジェクトには、さまざまな組織構造がありました。一部は、財団や資金援助を受けていない単一のメンテナーでした。一部のプロジェクトは、プロジェクト ガバナンスの一部として複数の委員会や組織を持つ、より大規模で複雑なソフトウェア プロジェクトでした。1 つの会社が一部のプロジェクトを完全に管理し、コントリビューターの大きなコミュニティが他のプロジェクトを作成しました。集計されたデータは、依存関係グラフを測定し、エンドユーザーの最大のグループが使用し、依存しているソフトウェア パッケージを評価しました。データは匿名化され、アプリケーションを実行している組織とのリンクを防ぐために変更されました。一部の例では、使用状況データ（主に月ごとのダウンロード）も調査され、Libraries.io などの OSS 使用状況に関するデータを収集および分析する他のデータソースを通じて利用できます。Libraries.io は、オープンソースの依存関係データとアプリケーション使用状況の重要なつながりである、パッケージ リポジトリとマネージャから使用状況データを収集します。

メンテナーとコントリビューターの人口統計に関する観察

32 件のインタビューには、さまざまな年齢、さまざまな場所のメンテナーが含まれていました。メンテナーは半ダースの国から来ていました。インタビューを受けた人のうち 6 人は女性で、残りは男性でした。全員がかなりの経験を持つ開発者とエンジニアでした。彼らは大規模な多国籍企業から小規模なコンサルタント会社まで、さまざまな企業で働いていました。1 人はオープンソース プロジェクトのみに従事していました。ほとんどが非常に大規模または小規模な企業で働いていました。大企業には Red Hat、Amazon、Microsoft、IBM、Meta、VMware などが含まれていました。”中規模企業”で働いている人はほとんどいませんでした。インタビューを受けた人のうち 2 人は Linux Foundation で働いていました。インタビューのうち 3 人は以前のプロジェクトの仕事に焦点を当てていました。インタビューを受けた人たちはメンテナーとコントリビューターの役割から離れていましたが、彼らの経験は依然として有用で価値があります。

図 2

メンテナーのこれまでのキャリアにおける役割



インタビューの数 = 32

メンテナーのキャリア パス

これらの重要なプロジェクトのメンテナーやコントリビューターは、さまざまな学問的背景を持っていました。かなりの割合の人が、大学でコンピューター サイエンスやソフトウェア エンジニアリングを専攻または副専攻として学んでいました。何人かは、後に学部生または大学院生として在学中にメンテナーになるソフトウェアに取り組んでいました。

これは、Linux カーネルやディストリビューション、Git、データベースなど、低レベルで複雑なプロジェクトに従事するメンテナーやコントリビューターに特に当てはまりました。また、これらのメンテナーは、大規模な確立されたソフトウェア会社で学校を卒業した後に仕事をする可能性が高く、そこでは、研究室の一部として、または成長するオープンソース プラクティスの一部として、オープンソースの仕事に参加していました。

これらのメンテナーの多くは、当初、Linux オペレーティング システムに触れて作業することを通じてオープンソースをスタートしました。また、プロジェクトの言語とコンピュータ サイエンスを学んだコントリビューターの割合との間にも相関関係がありました。C 言語で書かれたプロジェクトは、他の言語で書かれたプロジェクトよりも、大学で訓練されたコンピュータ科学者を惹きつける傾向がありました。メンテナーのもう 1 つの重要なグループは、自分たちのコンピューティング問題をよりよく解決するためにオープンソースに取り組み始めた学者たちでした。インタビューを受けた LF Research のメンテナーのうち、学術的な背景を持つ 1 人だけが、主に学者として雇用されています。ほとんどの人は、プロジェクトが十分なクリティカル マスを獲得した後、テクノロジー企業や財団の役割に移行します。

コントリビューター、メンテナー、あるいはその両方？

LF Research では、コントリビューターをコードを書いてプロジェクトに提出する人と定義し、メンテナーをコードレビュー、トリアージ、テスト、セキュリティ、ビルドとインフラストラクチャ、リリース管理など、プロジェクトの管理に従事する人と定義しています。インタビューを受けた人の

図 3

OSS プロジェクトで FTE に従事するメンテナー

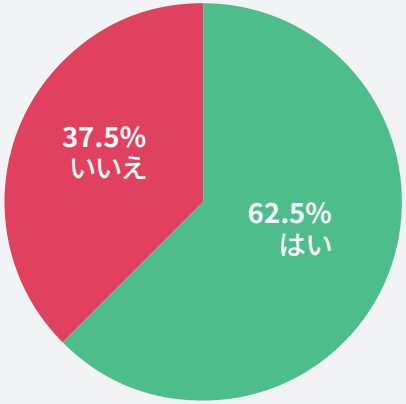
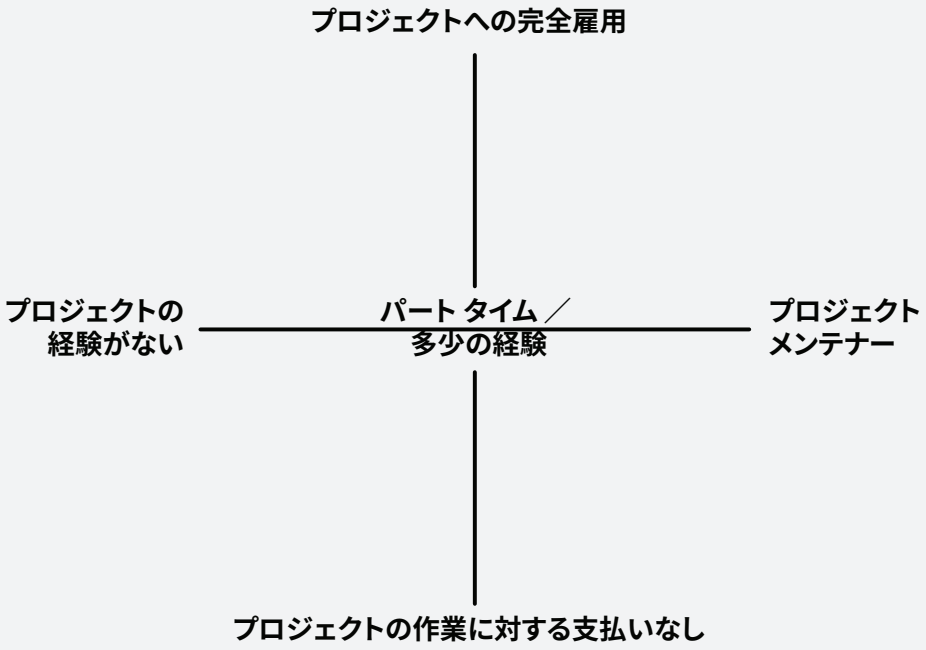


図 4

メンテナー典型グリッド



合計 75% が、プロジェクト メンテナーとコードのコントリビューターの両方を務めており、25% がメンテナーとしてのみ働いていました。メンテナーにインタビューするときに予想されるように、コードの貢献だけをしている人はいませんでした。オープンソース プロジェクトに費やされた時間の割合は大きく異なり、平均 70% でした。私たちのインタビューは、メンテナー 1 人で運営されているプロジェクトは 1 つのみでした。調査された他のすべてのプロジェクトでは、複数のメンテナーがワークロードを支援していました。

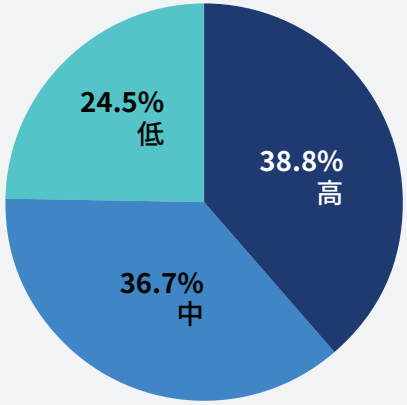
プロジェクトに従事している間、インタビューを受けた人のほとんどは、コントリビューターからコア コントリビューター、メンテナーへと進む道をたどっていました。大規模な組織内では、メンテナーは複数の役割とタスクを巧みに操る傾向がありました。例えば、Project Jupyter のメンテナーである Brian Granger 氏は、最初はプロジェクト コードの大部分を書いていました。現在は、資金調達、管理、技術アーキテクチャ設計、コードのレビュー、Jupyter の UX プラクティスの改善に重点を置いています。Amazon Web Services に勤務する Granger 氏は、Jupyter のコミュニティ ガバナンスにも Executive Council のメンバーとして引き続き関与しています。Linux Foundation の Shuah Khan 氏は、Linux カーネルのテストと QA プロセスを監督しています。また、プロジェクト ドキュメントの改善にも取り組んでおり、プロジェクトの多様性を高める取り組みの一環として、13 人のコントリビューターを積極的に指導しています。Khan 氏は当初、Linux カーネルのテストと QA インフラストラクチャに専門知識とアーキテクチャ設計を主に貢献していましたが、その後、より多くの管理活動を含む役割に進みました。

OSS プロジェクトでフルタイムまたはパートタイムで作業する

インタビューを受けた人のうち、2 人を除く全員が、プロジェクトへの時間の投資を支援する企業にフルタイムで雇用されていました。かなりの割合の人が、Vercel、Chromatic、Oxide Computer など、大規模なベンチャー支援企業で働いていました。1 人のメンテナーだけが、スポンサーシップ、ドナー、およびその他の形式のアド ホック プロジェクト資金からのみ支援を受けました。もう 1 人は仕事の合間にいましたが、過去 10 年間、主にオープンソース エバンジェリストとして、オープンソースで着実に働いていました。

図 5

組織的支援の認識度



インタビューを受けた人、彼らの雇用者、そして彼らのプロジェクトの間の関係は多様で、しばしば流動的でした。場合によっては、従業員はフルタイムで雇用され、特定のプロジェクトに以前に従事したことがなく、プロジェクトに従事するようになりました。場合によっては、従業員が雇用されたのは、コミュニティ内での地位と、企業がプロジェクトのロードマップを理解し、上流のコントリビューションを支援するのを助ける能力があるためでした。

従業員が、雇用主にとって戦略的なプロジェクトに時間の一部を費やすことを理解した上で雇用された例もあります。ここで取り上げたケースの中には、あるプロジェクトの主要なメンテナをすべて雇用している会社で働いていた人もいます。まれに、インタビュー対象者は完全に雇用されていたが、雇用主はプロジェクトでの仕事に無関心であった例もあります。2 つの例では、特定のプロジェクトのための機能を作ることで、顧客を支援することに特化した会社で働いていました。

優れたコントリビューターおよびメンテナーのプロジェクト オンボーディング経験

少数の例外を除いて、インタビューを受けた人たちは、プロジェクトに最初に参加したとき、歓迎すべきオンボーディング体験を楽しんでいました。これは、すべての場合において、取り組むべき一連の最初の問題を見つけることが必ずしも容易または明白であったと言っているわけではありません。しかし、一般的に、最初の創業者ではないインタビューを受けた人たちは、コミュニティに入った最初の経験が肯定的であったと報告しています。

それにもかかわらず、プロジェクトの開始以来、あるいは開始近くにコントリビューターやメンテナーを務めていた複数のインタビュー対象者は、成熟したコミュニティの規模、メンテナーに対する現在の要求、およびプロジェクト コードとアーキテクチャの複雑さの増大のために、今日のコミュニティがそれほど歓迎されないのではないかと疑っていると述べました。Andres Freund 氏は、最初に PostgreSQL にコードを提供したとき、次のように述べています。”... 応答が非常に迅速で詳細であったため、こう思いました、’クール！これはレビューや私の仕事生活で得たもの以上のものだ。’最近ではそうではないと思います。なぜなら、量があまりにも増えたので、そのような迅速な応答や詳細な応答はもはや

“今の仕事や他のすべてのことから一歩下がって、’よし、社会、技術、人類に最も大きな影響を与えることができることは何か？’と尋ねた場合、私はオープンソースに取り組むよりも優れたものを見つけるのに苦労するだろう”

—BRIAN GRANGER, CO-CREATOR AND LEAD MAINTAINER, JUPYTER

図 6

プロジェクトには正式なメンターシッププログラムがある

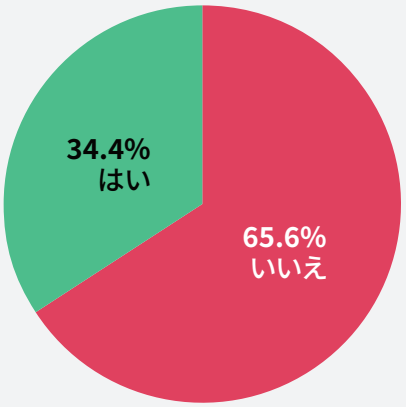


図 7

プロジェクトに正式な新規コントリビューター採用プロセス／プランがある

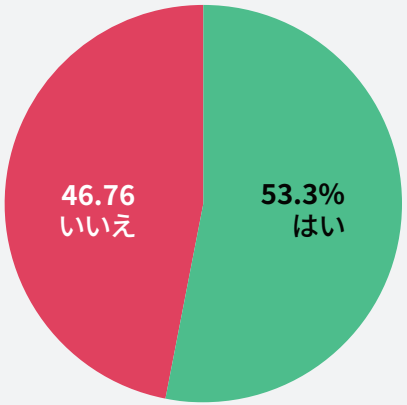
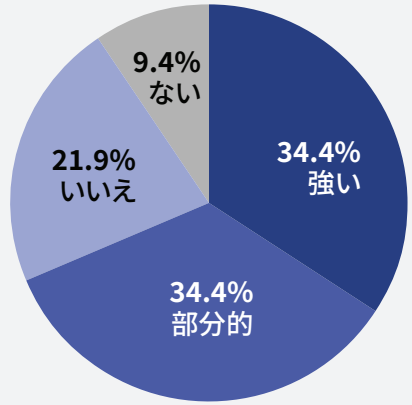


図 8

新たなコントリビューターパイプラインの強み



一般的ではないからです。”

何人かのインタビュー対象者は、彼らの最初のコントリビューションはしばしばエラーだらけで、今日のプロジェクトで温かい反応を受ける可能性は低かっただろうと述べました。彼らの同じコントリビューションが今日提出された場合、彼らの PR をラインを越えて統合するためにはかなりの努力が必要になるでしょう。さらに、初期のコントリビューターは、プロジェクトの初期段階であるため、より実質的な問題に取り組むことができると感じていました。メンテナーの時間をめぐる競争はそれほど厳しくなかったので、プロジェクトの創設者との豊富な関与（電子メールまたは GitHub 経由）はより可能性が高く、持続可能でした。また、プロジェクト コミュニティがあまり成熟しておらず、プロジェクト開発の主要分野の専門知識があまりカバーされていなかったために、貢献が歓迎された分野がさらに多くあった。少なくとも 1 つの例では、メンテナーはコミュニティの行動を注意深く観察し、非常に複雑で参加者がほとんどいない貢献分野を選択して、コミュニティに受け入れられる可能性を高めました。

インタビューを受けたメンテナーは、新しいコントリビューターを落胆させる特定の行動を認識しています：

- メンテナーは、タイムリーに PR に応答することができませんでした。
- メンテナーは、新しいコントリビューターの PR のデバッグを支援するために長い時間を費やすことをやめました。
- メンテナーは、潜在的に価値のある才能を特定するために、新しいコントリビューターを積極的にトリアージすることをやめました。
- コントリビューターたちは、会話や入力のための入り口を見つけるのに苦労しました。

プロジェクトのコア コミッター チームのメンバーである Freund 氏は、次のように説明しています。”（作業と PR の）量が非常に増加したので、（私が受け取った）そのような迅速な応答と詳細な応答はもはや標準ではありません。少なくとも PostgreSQL コミュニティでは、私が今日成功できるのか、または始めたときのように再び夢中になるのかさえわかりません”。言い換えれば、コントリビューターの経験は、時間の経過とともに低下する傾向があるというのが彼らの意見です。これは、コントリビューターの経験に対処し、維持するための協調的な努力にもかかわらず、しばしば行われました。

オープンソース メンテナーシップから受ける価値

メンテナーは、オープンソースと、オープンソースの価値を持つシステムで働く彼らの能力に対して、普遍的に感謝の意を表明しました。メンテナーが彼らの活動に対して得る具体的な利益には、内的価値と外的価値の両方が含まれます。LF Research が、FOSS Contributor Survey Report など、オープンソースのメンテナーとコントリビューターを対象としたこれまでの調査で一般的に示してきた内的価値には、次のようなものがあります：

- 共同体意識と仲間意識
- 最先端の技術に取り組む能力
- 自分のコースを設定し、活動に優先順位を付ける能力、仲間のグローバル コミュニティと協力する能力、ゼロから何かを作る喜び
- コミュニティが形成され、飛び立つのを見ることの達成感
- 本当に賢い人たちと一緒に仕事をするチャンス

”コミュニティの人たちに初めて直接会ったとき、私はみんなをハグしていました。私たちは長年にわたって一緒に仕事をしてきたので、とても感情的な瞬間でした”と Julia のリードメンテナーの一人は言います。

> ”オープンソースは、私にとって人生を変える経験です。クローズドソースの世界から来た私は、後部座席の乗客として乗るのではなく、運転席にいるような気がします。私は自分のキャリアをコントロールできるような気がします。私が何をし、何に貢献するかを直接コントロールできます。”
>
> —SHUAH KHAN, MAINTAINER, LINUX KERNEL

外部的な価値には、より良い雇用へのアクセス、さまざまな雇用者のために働き、彼らの働き方、場所、仕事のスタイルを選択できること、注目されたプロジェクトやトレンドのプロジェクトとの関連などの外部的なステータス シグナルなどが含まれていました。さらに、何人かのメンテナーは、彼らのオープンソースの仕事は、日々の仕事に直接貢献していないが、雇用者にとっての魅力に貢献していると述べました。インタビューを受けた人のうち、キャリア選択としてオープンソースで働くことに金銭的な懸念を表明したのは 1 人だけでした。偶然ではありませんが、そのメンテナーは、明確な商業的実体や金銭的エンジンを持たないプロジェクトを率いており、完全に寄付とスポンサーシップに依存していました。プロジェクトの資金調達に関する懸念を表明しているメンテナーの大多数は、JavaScript プロジェクトに取り組んでいました。この分野は、JavaScript エコシステムのダイナミクスと、独立して維持されている JavaScript プロジェクトに対する企業の資金調達レベルが低いために、伝統的に十分な資金を得るのに苦労してきました。

メンテナー ベストプラクティス

各インタビューの一環として、LF Research はメンテナーとコントリビューターに、プロジェクトの成功に貢献したベストプラクティスは何かを尋ねました。コミュニティの健康は不正確な科学ですが、この研究活動の対象となったコミュニティのほとんどは、新しいコードとバージョンの出荷という最も基本的な指標では健康でした。

何人かのメンテナは、コミュニティが新しいコントリビューターを見つけるのに苦労しており、この課題が今後のプロジェクトに悪影響を及ぼす可能性があるという懸念を表明しました。メンテナーは全員、コミュニティの健康を維持する方法について強い見解を表明しました。プロジェクトの規模、言語、ライフ サイクル ステージがこれらの見解に影響を与えました。

各プロジェクトのベストプラクティスがどのように生まれたかは、大きく異なります。ほとんどのメンテナーは、個人的な実践経験と試行錯誤を通じていくつかのベストプラクティスを作成しました。その他は、コミュニティ内でガバナンスとルールの特定の部分として成文化されました。多くのプロジェクトは、変化の時期を経験しており、コミュニティの相互作用、実行、成長を円滑にするために、一連の新しいベストプラクティスが採用されました。後から考えると、プロジェクトはコードを作成するときと同じように独自のプラクティスを扱い、改善の可能性のある方法について絶えず精査とレビューを行いました。

すべてのケースで機能する包括的なベストプラクティスのセットはありません。しかし、メンテナーとコア コントリビューターは、彼らのニーズを満たすパーソナライズされた組織的なベストプラクティスを構築する上で、優れた創造性を発揮しました。ここでは、インタビュー対象者によって有用で役に立つと特定されたベストプラクティスの内訳を示します。このリストは特定の実践分野にグループ化されていますが、多くの場合、ベストプラクティスは、コントリビューターの経験、コミュニティ ガバナンス、ドキュメンテーション、資金調達と現物出資の生成、多様性、燃え尽き症候群の防止など、複数の分野に影響を与えます。以下のセクションでは、LF Research がそれぞれについて詳しく説明します。

コントリビューターの経験

コントリビューターは、中立的なオープンソース プロジェクトの生命線です。成功したメンテナーの多くは、プロジェクトの非常に早い段階でコントリビューターの経験を優先し、コントリビューターにコメントやファイルのバグを奨励し、最終的にはコードの改善のための提案を（PR の形で）提出するように働きかけています。一部のメンテナーは、コントリビューターを奨励するために英雄的な手段を講じました。例えば、Storybook プロジェクトのリード メンテナーである Norbert de Langen 氏は、プロジェクト コードに関する提案や質問をメールで送ってきた人に直接話をしたいというリクエストとともに、ミーティング スケジュール リンクを送信しました。de Langen 氏は、プロジェクトの最初の 1 年間に、この方法を使用して 200 人以上の人と会いました。彼は、彼が後

に会った人の 20% 近くが Storybook のリピーターになったと推定しています。

成功するプロジェクトを構築するメンテナーは、多くの場合、初めてのコントリビューターがコントリビューションを改善し、自動チェックシステムを使用してコードの形式と構文をチェックするなど、コード提出の準備方法を学ぶのを支援するために、多大な労力を費やします。現在 Salt プロジェクトのメンテナーである Gareth Greenway 氏は、エラーだらけの PR を提出し、プロジェクト作成者のリーダーである Thomas Hatch 氏が読みました。最初の作成者は Greenway 氏に肯定的なフィードバックを与え、彼と協力してコードを修正しました。Greenaway 氏は他にも多くの PR を提出し、後に Salt メンテナーになりました。

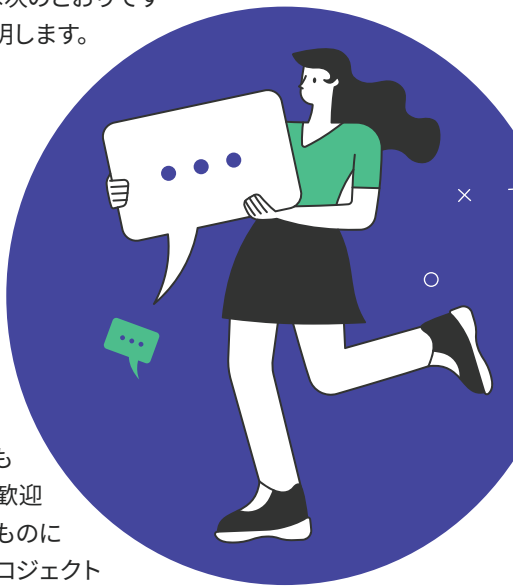
以下は、成功したメンテナが強力な貢献者体験を構築するための、その他の一般的な方法のリストです。

初めてのコントリビューターには個人的に回答してください。

初めてのコントリビューターがコミュニティと関わり始めたときには、個人的に対応します。これは、小規模なコミュニティの初期段階のプロジェクトではより実行可能ですが、多くのメンテナーは依然として個人的に一貫して対応しようとします。小規模で、より包括されたプロジェクトでは、個人的な対応時間はそれほど困難ではありません。たとえば、Linux ツールチェーンや containerd などのプロジェクトでは、多数のコントリビューターを必要とせず、C++ などのより困難なソフトウェア言語で作成されているため、メンテナは潜在的なコントリビューターやメンテナーから大量のインバウンド メッセージを見ることはありません。(注: これは、特にライフサイクルの後期段階にある古いプロジェクトでは、新しいコミュニティ メンバーやコントリビューターを採用する際の課題も示しています)。

自動グリーティング ボットとオンボーディング ガイドを設定します。

自動化はヒューマン タッチに取って代わるものではありませんが、簡単なスケーリングや迅速な応答時間などの重要なメリットを提供します。特に中規模および大規模のコミュニティでは、メンテナーは一般的に、Slack または Discord の新しいコミュニティ メンバーに挨拶し、彼らをオンボーディング リソースに誘導するための自動グリーティング ボットをセットアップします。Linkerd、Kubernetes、Ansible、GitHub、およびその他の多くの組織がこの機能を使用しています。一部の組織では、GitHub のボットを使用して新しいコントリビューターに挨拶しています。”Jupyter Lab（Project Jupyter の一部）は、GitHub ボットを使用して新しいコントリビューターを歓迎します。あなたがイシューまたはプル リクエストを開いていることを検出した場合、私たちは’こんにちは、これはあなたの初めてのコントリビューションです。プロジェクトへようこそ。コミュニティに参加する方法は次のとおりです’と返答します。”と Granger 氏は説明します。



包括的な言語を使用します。

これは、現在および将来のコントリビューターの多様なグループの参加を有意義に改善することができます。包括的な言語は、人種や性別に関して中立であるだけでなく、地域の言語の違いや、英語を話さない人がプロジェクトのコミュニケーションを容易に理解できることも考慮しています。包括的な言語は、歓迎的で、魅力的で、エネルギッシュなものになる可能性があります。これは、プロジェクトが痛みどめスタイルでのみコミュニケーションを行う必要があるということではなく、プロジェクト リーダーは、コントリビューターや他のプロジェクト参加者のコミュニケーション要件を認識する必要があるということです。例えば、プロジェクトは、コミュニケーションを簡素化するために、スポーツの紹介や地元のジョークなどの地域的な表現を最小限に抑えるよう努める必要があります。電子メール、Slack、オンライン ドキュメントやページに包括的な言語の提案を提供できるさまざまな自動分析ツールがあります。最も人気のあるオンライン文法チェック ツールである Grammarly は、包括的でジェンダーに中立な言語の提案を提供しています。
Textio は、明示的と暗黙的の両方のバイアスに対してジョブ リスト言語をスクリーニングし、明示的と暗黙的の両方でより包括的な言語を提案する人気のオンライン ツールです。常識はまた、言語をより包括的にするために大いに役立ちます。

プロジェクトのさまざまなエントリ ポイントと通信チャネルをすべて集約する簡単なページを作成します。

多くのプロジェクトには、メーリングリスト、チャット（Slack、Discord、IRC）、GitHub リポジトリ、YouTube のプロジェクト チャネル、Wiki など、複数のコミュニケーション手段があります。新しいコントリビューターがコミュニティとつながるのを助ける簡単な方法は、Ceph が“参加する”ページで行っているように、これらすべてのチャネルを簡単に見つけることができるようにすることです。

プロジェクト ドキュメントにオンボーディング セクションまたは”貢献方法”セクションを作成するか、プロジェクトのGitHubページにMarkdownファイルを作成します。

これには、コミュニティがどのように機能するかの説明、チュートリアルと主要なドキュメントへのリンクが含まれており、理想的には、主要なコミュニティ リーダーやメンバーと連絡を取る方法とともに、コントリビューションと参加の方法を特定します。ほとんどすべてのプロジェクトには、ドキュメントと GitHub のどこかにオンボーディング ページがあります。ベストプラクティスは、複数の場所からそれらにリンクして、新しいコントリビューターが簡単に見つけられるようにすることです。このページの情報が多ければ多いほど、より良いです。たとえば、GitHub の Jupyter の CONTRIBUTING.MD ページには、コントリビューションの回帰テストの方法へのリンクまで、詳細な情報が含まれています。

初めてのコントリビューターに対してバグやPRを提案したり（”good first issue”バグやPRと呼ばれることが多いです）、ヘルプを利用できるバグを特定したりします。

多くのプロジェクトでは、コントリビューターは、バグがプロジェクト内の誰かによって”所有されている”かどうかを知らないため、PR に飛び込んで提出することをためらっています（例えば、モジュールを管理する人は、バグ修正者の信頼できる幹部を持っているかもしれません）。ここで、新しいコントリビューターのためのバグのラベルが役に立ちます。これらの最初のバグは、ドキュメントやより基本的なコード修正の作成である可能性がありますが、実際にはプロジェクト リーダーの好みに依存します。現在、多くのプロジェクトが GitHub リポジトリにこれらのアイテムのラベルを持っています。例えば、VSCode には、要求されていないバグを識別するための“help wanted”ラベルがあります。Storybook.js には、あまり複雑でないイシューへとコントリビューターを誘導する“good first issue”ラベルがあります。これらのラベルを使用する上で重要な部分は、常にバックログを確保することです。例えば、この記事を書いている時点で、Storybook には 26 のイシューがラベルの下にあります。どこから始めればいいのかわからないコントリビューターにとって、これは理想的な方法です。

> “メンテナーの仕事は、コミュニティやコントリビューターに対応し、彼らがどこから来たのかを理解しようとし、フィードバックを与えることです。多くの場合、私たちの主要なバグのいくつかは、私たちが対応してきたために、時間内にキャッチされたり対処されたりしています。”
>
> —NEHA OJHA, Ceph

平均以上の能力を持つコントリビューターを特定するための基準またはメカニズムを確立します。

すべてのコントリビューターは価値がありますが、適切なスキルを持った人は、学習曲線が比較的短いため、より価値があるかもしれません。例えば、C 言語で書かれたプロジェクトでは、1 つの C 言語に堪能であることは大きなメリットです。高品質の PR を通じて技術的な洞察力を示したり、より困難な問題に取り組むことに固執したりするコントリビューターは、より複雑な初期プロジェクトに取り組むように奨励されたり、メモリ管理、コンパイラ、ネットワーキングなど、プロジェクトのコード ベースやサブシステムのより困難な側面に取り組むメンターと連絡を取ることができます。メンテナーは、主にコードの品質に注目したり、初期 PR を考えたり、プロジェクトに関する潜在的なコントリビューターとの電子メールの会話の内容を通じて、技術的な洞察力を特定しました。

どちらも、特に困難な問題やプロジェクト タスク（メモリ管理、コンパイラ、セキュリティなど）に取り組むコントリビューターを見つけるのが難しいプロジェクトでは、新しいコントリビューターとのアウトリーチと直接的なコミュニケーションが重要であることを強調しています。一部のプロジェクトでは、潜在的なコントリビューターのフィルターとして機能する知識の重複が必要です。たとえば、NumPy は主に科学者、エンジニア、およびデータ分析に Python を使用している人によって書かれていますが、プロジェクトのコードは C で書かれています。NumPy のメンテナーは、C コーディング スキルを持つ科学者がコントリビュートしてきた場合、強力なコントリビューターになる可能性が高いことを知っています。

欠陥のあるコントリビューションを拒否するのではなく、改善方法について提案し、対話を開始します。

一部のメンテナーにとって、PR や電子メールを通じて明確かつ論理的にコミュニケーションをとる潜在的なコントリビューターは、スニフ テストに合格します。”パッチの質よりも優れていたのは、何かを提出し、何をしようとしているのかを説明した人たちでした。私は彼らのところに戻って、’あなたがしようとしていることはわかりますが、X、Y、Z です’と言い、あなたは彼らと良い会話をしました”と、Git の元リード メンテナーである Jeff King は説明します。

最も困難な問題に取り組む努力に報います。

困難な問題は、プロジェクトにとって”ショー ストッパー”リスクや大きなボトルネックになる可能性があります。より困難なエンジニアリング タスクは、一般的に、より少ない志願者しか惹きつけません。この問題は、プロジェクトが成熟し、組織的な知識が少数の専門家集団に蓄積されるにつれて増大する傾向があります。このため、プロジェクトのメンテナーは、プロジェクトの初心者であっても、より困難な問題を解決したり、それに取り組んだりすることに関心を示すコントリビューターを育成するために、並外れた努力をしたと述べています。これらの努力を認め、報酬を与えるための、シンプルでありながら有意義な方法がいくつかあります。リリース ノートでの言及、ソーシャル メディアでの発信、コミュニティ コールでの言及は、困難な問題に取り組む努力を奨励するための、コストがかからず、労力の少ない方法のほんの一部です。メンテナーはさらに踏み込んだ方がいいかもしれません。これは、困難な問題に取り組んでいるコントリビューターに特別な贈り物を送ったり、公共の場やプロジェクト サイトでブログ記事を書くのを手伝ったり、いくつかの例を挙げると、彼らの仕事に関するカンファレンス プレゼンテーションを準備するのを手伝ったりすることを意味します。何よりも、メンテナーは、質問、コメント、電子メール、またはコード提出に対するタイムリーな応答を保証することによって、これらのコントリビューターを育成するために特に努力すべきです。これは、メンテナーの時間が最も不足していることへの関心を示しています。

コントリビューションのハードルを引き下げるよう努力します。

コントリビューターの中には、参加を希望しても、急な学習曲線、ドキュメントの不足、またはテクノロジーの使用における課題のために不満を感じる人もいます。これは、最終的に非常に価値があり生産的になる可能性のあるコントリビューターを怖がらせる可能性があります。賢いメンテナーはこの問題を認識し、コア作業の一部としてそれに対処しようとします。

Ceph では、Neha Ojha 氏が長い間、このプロジェクトは使いにくく、始めるのが難しいという認識に悩まされてきました。これに対処するために、彼女の最優先事項の 1 つは”ハードルを下げることです。新しいコントリビューターや新しいユーザーに対しては、常にハードルを下げることができます”と Ojha 氏は言います。彼女にとって、その重点の一部はドキュメントに置かれています。”私は Ceph ドキュメントの改善を強調しています。なぜなら、適切なドキュメントがあれば、最初の戦いはすでに勝利していると感じたからです。そうすれば、ユーザーは少なくとも Ceph を起動して実行することができ、私たちは彼らを関与させ続けるための小さな勝利を得ることができます。”

約束と献身を示している新しいコントリビューターを奨励するために、幸せなマイルストーンを作成します。

例えば、Storybook は、2 番目の PR がマージされた後、GitHub のプロジェクト チームにコントリビューターを静かに追加します。コントリビューターは通知を受け、一般的に喜んでいます。（これにより、無料の Copilot アカウントなど、他の GitHub のメリットも解放されます）。”人々

“私が提出した最初のプル リクエストには、500 近くのリント エラーがありました。私は今でも、Salt への貢献の中で最も多くのリント エラーの記録を持っていると思います。多くのオープンソース プロジェクトは、私に’これはゴミです。あちらに行ってください’と言ったでしょう。代わりに、Tom Hatch (Salt プロジェクトの作成者) は、’これは素晴らしい! 私たちは長い間これを必要としてきました。あなたはいくつかのリント エラーを持っているようです。それらを修正して、これを統合しましょう’と言いました。”

—GARETH GREENAWAY, SALT

は驚き、本当に幸せになります”と Storybook の de Langen 氏は言います。他のメンテナーは、各プロジェクト リリースで引用されたすべてのコントリビューターに特別な限定版リリース ステッカーを送るなど、簡単な認識手段を提案しています。NumPy は、リリース ノートの電子メールで特別な感謝を示すために、初めてのコントリビューターの名前の横に”+”記号を追加します。

新規コントリビューター専用の営業時間を設定します。

この時間は、コントリビューション プロセスを通じて彼らを指導し、彼らのコントリビューションが迅速にマージされる可能性を高めるために使用できます。新しいコントリビューターは、メンテナーからのメンターシップやメンテナーとの接触を楽しむこともよくあります。Linux カーネルの Shuah Khan 氏は、さまざまなバックグラウンドを持つ複数のコントリビューターに対して、定期的にミーティングを行い、電子メールで質問に答え、コントリビューションの準備を支援することでメンターしています。これはプロジェクトにおける彼女の仕事の一部です。”彼らは、私たちがアップストリームで考慮する必要があるパッチを提出するので、彼らの作業が、パッチを取得する方法をよりよく理解できるようにメンターされる方が良いです”と Khan 氏は説明します。”新しいコントリビューターからのパッチの品質を向上させることで、プロジェクト全体に利益をもたらします。”

新しいPRのトリアージ プロセスを確立し、それに従います。

これは当たり前のことのように思えます。しかし、重要なのは、プロセスを設定するだけでなく、主要なプロジェクト メンバーまたはメンバーのグループにトリアージのオーナーシップを与えることです (これは、トリアージ マスターとしての期間を通じてメンバーを交代させることを意味します)。ほとんどのプロジェクトはすでに、新しいパッチとリクエストの責任あるトリアージを目指していますが、多くの場合、仕事と生活が邪魔になることがあります。トリアージを処理し、優先順位を付けるためのプロジェクト リーダーシップからのサポートを処理するシステムがなければ、トリアージは優先リストの一番下に落ちる傾向があります。トリアージと初回対応が積極的に優先されない限り、アド ホック／ベスト エフォート アプローチになる可能性があります。これは、期待を管理せず、長い遅延を引き起こす可能性があるため、コントリビューターを消失する可能性があります。たとえば、Salt メンテナーは、トリアージ プロセスをオーバーホールし、特定の役割とカバレッジを作成して、トリアージが同じプロセスに沿って移動し、それに従うようにしました。”最近では、新しい問題が発生するたびに適切なトリアージが行われています”と、Salt メンテナーの Pedro Algarvio 氏は述べ、これは必ずしもそうではなかったと指摘しています。

遠慮せずに断ることです。

新しいコントリビューターは、プロジェクト コードやサブシステムの動作に重大な影響を与え、連鎖的で意図しない結果をもたらす可能性のあるアイデアを提案し、パッチを提出することがあります。強力なメンテナーは、迅速に対応し、自分のアイデアやパッチがなぜ問題があり、受け入れられない可能性があるのかを丁寧に、しかししっかりと説明しようと努力します。さらに、(公開されるべき) 回答は、後のコントリビューターを導いたり、将来の他の新しいコントリビューターにその点を説明する簡単な方法として役立つことができます。”No は一時的なものですが、Yes は永続的なものです。どのコードを追加することに同意するかには非常に注意してください。なぜなら、それは長期的な結果をもたらすからです”と、Red Hat の Fedora Project (下流の Linux ディストリビューション) の 3 人の主要なカーネル エンジニアの 1 人であり、Linux メンテナーでもあった Laura Abbott 氏は言います。カーネル エンジニアリング以外にも、Abbott 氏は Fedora やより広範な Linux コミュニティで、メンテナーとして、またコントリビューターのメンターやコミュニティの関心事のラングラーとして、非常に活発に活動していました。Abbott 氏はさらに次のように説明しています。”人々が尋ねていることに耳を傾けてください。しかし、コードを追加するよりもコードを削除する方がはるかに難しいことを認識してください。このことを念頭に置いて追加するようにしてください”。

コミットとPRを”バースト”処理するためのチーム作業を作成します。

PostgreSQL チームは定期的に”committfests”を組織しており、そこでは、意味のある保留中のコード提出とパッチがすべてプロジェクト チームから応答を受け取ります。このシステムは、リズムがあり、活動が年に数回繰り返される場合に最も効果的に機能します。

上部に正しいトーンが設定されていることを確認します。

オープンソース プロジェクトの創設とリーダーシップの DNA は、さらなる成功とコミュニティの成長に不可欠です。最高のプロジェクト創設者とリーダーは、ガバナンスを改善し、コントリビューターシップを成長させるためのプロセスと取り組みを促進し、支援します。一体性とコミュニティの健全性のトーンは、通常、プロジェクトの創設者とプロジェクトを開始する小さなチームによってトップに設定されなければなりません。

コミュニティのガバナンスと管理

プロジェクトの初期段階では、コミュニティのガバナンスと管理が考慮されないことがよくあります。これは驚くことではありません。初期段階では、プロジェクト作成者は、実行可能な最小限のプロジェクトに到達し、強力なソフトウェア基盤を作成することに重点を置いています。プロジェクトが組織内で生まれた場合、ガバナンスは多くの場合、内部機能であり、持続可能なコミュニティを構築するための長期的な要件ではなく、内部のニーズに焦点を当てています。複雑なプロジェクトでは、プロジェクトを監督する自然な”リード メンテナー”（作成者など）がいたとしても、最終的には成文化されたガバナンスと管理構造が必要になります。

小規模なプロジェクトでは、明示的なガバナンス ポリシーと構造は、日々の管理には必要ではないかもしれませんが、紛争を裁定し、コミュニティが関与するプロジェクトについて長期的な方向性を決定するためには不可欠になります。成文化されたガバナンスは、物事が順調に進んでいるときには見過ごされることがよくありますが、コミュニティが問題に直面したときには突然重要になります。ここでは、インタビュー対象者が提供したコミュニティのガバナンスと管理のためのベストプラクティスをいくつか紹介します。行動規範を確立するなど、明白なものもあれば、

> “私は変更を見て、実際に誰がその背後にいるのかを無視する傾向があります。このコミットを行っている（または変更を提案している）定期的なコントリビューターがいる場合でも、または完全な初心者がいる場合でも、私は両方を同じように見るようにしています。”
>
> —DANIEL STENBERG, LEAD MAINTAINER, CURL

あまり明白ではなく、より慣用的なものもあり、すべてのプロジェクトで広く機能するわけではありませんが、特定のプロジェクトで成功しているものもあります。

行動規範を制定します。

これは、歓迎すべき環境を維持したいと考える成功したオープンソース プロジェクトにとって、現在では一般的な慣行となっています。行動規範の例は簡単に見つかります。オープンソース プロジェクトのリーダーシップ チームやプロジェクト メンバーが行動規範に反対することはめったにありません。しかし、行動規範は、紛争を防ぎ、友好的な相互作用の最低基準を設定するために不可欠です。さらに、報告された行為違反に対処し、決定から偏見を取り除き、再犯者に対処するためのガイドラインとプロセスを文書化することがしばしば不可欠です。

一般的なコミュニティの礼儀作法を確立します。

行動規範は必要ですが、それだけでは適切なトーンを設定するのに十分ではありません。インタビューを受けた人の多くは、コミュニティに貢献し続け、後にメンテナーになることに同意した理由の一部は、コミュニティの礼儀正しさにあると一貫して述べています。経験豊富なメンテ

ナーはまた、議論が過熱しているように見える場合は、コミュニティ メンバーにトーンを和らげるように丁寧に要求したと述べています。これは、コミュニティの会話をより歓迎する方法であり、コミュニティのメンバーに対して、無愛想ではあるが善意のあるメンバーが、対話において無愛想でないトーンを使用することを目に見える形で奨励しました。

できるだけ早く自分の仕事から自分をデザインします。

メンテナーたちは、自分たちのプロジェクトに対する権限とコントロールを感情的に放棄することがいかに困難であるかについて議論しました。ほとんどの人が、自分たちのプロジェクトに強い愛着を感じていました。しかし、コントリビューターにインセンティブを与え、より自立した健全なコミュニティを構築できるようにするためには、コントロールを放棄することが重要であることを認識していました。一握りのメンテナーたちは、興味を示し、基本的な能力を証明できる開発者やボランティアに積極的に責任を与えようとしました。”最初から、私は’よし、この所有権を誰に譲渡できるのか？誰にこの責任を与えることができるのか？’と思っていました”と、Storybook のリード メンテナーである Norbert de Langren 氏は言います。彼は当初、この役割を 1 年間だけ果たすことを計画していました。

> “プロジェクトをサポートしている営利企業があるオープンソース プロジェクトは、1 つのコミュニティである必要があります。それはありえません。あなたには商業的な側面があり、彼らは独自のことをしています。そしてあなたにはオープンソース コミュニティがあり、彼らは独自のことをしています。それはうまくいきません。なぜなら、あなたは物事がオープンであるという目的を損なっているからです。”
>
> —GARETH GREENAWAY, SALT

徹底的な透明性を提供します。

メンテナーは、100% の透明性を達成するために、プロジェクトで発生するすべてのことをプライベート チャネルからパブリック チャネルにプッシュしようとしたと述べています。ほとんどの直感的な考えは、オープンソース コミュニティが透明性を期待していることを理解していました。ほとんどの場合、プライベートな接触は、民間企業がコードの変更を提出し、これを実現する最善の方法を理解したいと考えたときに行われました。メンテナーは、このようなサイドチャネルの議論を思いとどまらせました。根本的な透明性は、他の 2 つの理由から特に重要です。第 1 に、継承と継続性を簡素化し、第 2 に、コミュニティ全体が意思決定に参加できるようにします。

権限と意思決定を分散します。

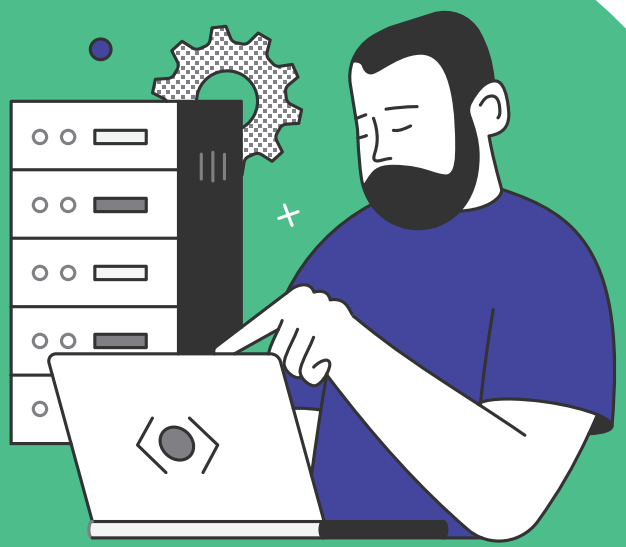
2022 年まで、Jupyter プロジェクトは重要な決定を下すために、コミュニティ全体の大まかなコンセンサスを使用していました。コンセンサスはプロジェクトの初期段階ではうまく機能していましたが、複雑さが増し、1,500 人以上のコントリビューターと 100 以上の GitHub リポジトリを持ち参加者の数が増加するにつれて、広範なコンセンサスの構築は管理不可能になりました。Jupyter の多くの部分は、理解して決定を下すために専門的な知識を必要としました。Jupyter の共同作成者でメンテナーの Granger 氏によると、50 人から 100 人のコントリビューターとメンテナーを持つサブプロジェクトでさえ、コンセンサスに苦労していました。これがイノベーションを遅らせていました。

その結果、プロジェクト リーダーは、コンセンサスが不十分なときに意思決定を行うのに苦労しました。これは、コミュニティ メンバーの不平や不満につながり、プロジェクト リーダーの燃え尽き症候群の一因となりました。公開された議論と審議の後、Jupyter は、意思決定を分割するために、Executive committee、Software Steering Council、およびワーキング グループを備えたモジュラー ガバナンス構造を採用しました。”私たちの新しいモデルは、適切なバランスのとれた権限チェック、明確な意思決定プロセス、説明責任のメカニズム、および責任ある透明性のある方法で利益相反を処理することを確実にするために、本当に

懸命に機能します。これらはすべて、企業、個人、および非営利団体が協力して、中立的で協力的な状況で Jupyter に取り組むことができるようにするためです”と Granger 氏は説明します。

コミュニティ管理における明確な中立性を維持します。

これは、単一の組織または企業によって管理されているプロジェクトでは特に重要です。経験豊富なメンテナーは、問題や PR の提出者の身元を無視するほど、すべての貢献者の中立的な見解を維持するよう努力していると述べています。メンテナーの中には、特に複雑な分野に取り組んでいることで知られている正規の人から貢献があった場合、それにフラグを付ける可能性があることを認めている人もいます。しかし、大まかに言えば、彼らはすべてのコントリビューターと提出されたすべての PR を新鮮な目で平等に扱うよう努力しました。中立性を維持する上で重要なことは、中立性がどのようなものであるべきか、中立的な当事者がどのように行動すべきかについての認識と注意を示すことです。これには、プロジェクトを非常に公的な方法で維持するという中核的な作業を行うことと、明らかにゼロサムではないプロジェクト開発を奨励することの両方が必要であり、支配会社を超えてプロジェクトをより商業的に有用なものにすることさえ必要です。



containerd ランタイム エンジンのコア メンテナーである Amazon の Phil Estes 氏は、次のように説明しています。”’木を切って水を運ぶ’と呼ばれる一般的に有用なコミュニティの雑用を行うことで、多くの場合、プロジェクトのリーダーシップを獲得する信頼性と信頼を得ることができます。例えば、’プロジェクト メンテナーの皆さん、これはプロジェクトをより観察可能または測定可能にするためのクールな方法だと思いますし、この機能はこれらの特定のユースケースにとって本当に価値があると思います’と言うときには、プロジェクトを健全に保つための重要な部分であれば、議論ははるかに容易になります。”

そうは言っても、多くの重要なオープンソースプロジェクトでは、主要なベンダーで働く人たちに与えられた時間で、独立した、あるいはホビイストのコミュニティ コントリビューターを見つけるのが難しいことがよくあります。これは、ツールチェーンやランタイム環境など、インフラストラクチャの最下層で動作するプロジェクトでは現実であり、よりニッチなスキルを必要とし、Kubernetes などの大規模なプロジェクトほど魅力的ではないと見なされる可能性があります。”’ああ、私はここにいて、containerd で働きたい。私はコンテナ製品を提供しているベンダーで働いているわけではありませんが、このテクノロジーが大好きです’と言ってくれる人が現れることはめったにありません”と Estes 氏は説明します。

テストセットを使用して、中立性を強制します。

メンテナーからのもう 1 つのアドバイスは、テスト セットを中立性のための強制関数として使用することです。”Meta はプロジェクトとしての PyTorch の重要性を理解していますが、内部顧客と外部顧客の間の利害の対立から私たちを守るもう 1 つのものは、優れたテスト セットです。私たちがオープンソースで実行し、内部でも実行するユニット テストは、コミュニティが必要とするものと会社が必要とするものを非常によく表しています”と、PyTorch のコア メンテナである Nikita Shulga 氏は説明します。

ドキュメンテーション

強力なドキュメントの文化を作ることは、オープンソース プロジェクト開発のすべての段階で不可欠です。ドキュメントは、新しいコントリビューターとプロジェクト ユーザーの両方にとって、よりスムーズなオンボーディングを可能にします。プロジェクトの規模が大きくなるにつれて、ドキュメントは、プロジェクト コードの”方法”だけでなく、エンジニアリング アプローチの背後にあるアーキテクチャ哲学を伝えるための重要なツールになります。ユーザーとコントリビューターは、Slack や GitHub でプロジェクト メンテナとコントリビューターに連絡する前にドキュメントを読むことはできませんが、ドキュメントは、一般的な質問

に対するカットアンドペーストの回答を可能にすることで、サポート タスクを簡素化することができます。不完全なドキュメント、特に一般的に要求される情報は、採用の障壁を高めるだけでなく、すでに過労のプロジェクト メンテナーとコントリビューターに不必要な作業を生み出します。

この調査のためにインタビューを受けたプロジェクト リーダーの大多数は、彼らのプロジェクトがドキュメントを改善する必要があることに懸念を表明しました。優れたドキュメント コントリビューターを採用し、維持することは困難です。頻繁に引用される 2 番目の問題は、新しいコードをマージすることと競合して、ドキュメントをオーバーホールしたいという願望です。さらに、彼らはドキュメント インフラストラクチャの問題や機能の欠如を挙げています、と Apache Maven プロジェクトのコア コントリビューターでメンテナーの 1 人は説明しています。

> ドキュメント全体の完全な再編成は、まさに誰も得ることができないタイプのコントリビューションです。なぜなら、おそらくすべてを変更する知識を持っている人が必要だからです。しかし、私たちはコンテンツを作り直す必要があります。実際、誰も完全にすべてを再編成する完全な能力を持っていません。それは現状のままです。私はそれを改善するために多くの努力をしましたが、今のところ、はい、それは現状のまま機能し続けています。

文書化に対してより高度で体系的なアプローチを構築したプロジェクトは、プロセスを改善し、より多くのコントリビューターを引きつけるために、さまざまなステップを実行します。ここでは、文書化の課題に対処するために成功したメンテナーが使用した戦術の概要を示します。

ドキュメントをプロジェクト階層の第一級市民にします。

これは通常、ドキュメントが重要であることをコミュニティに示すための、非常に目に見えるポリシーとプラクティスのセットを意味します。1 つのアプローチは、プロジェクト作成者がドキュメント委員会に参加し、ドキュメントの作業に関与することです。もう 1 つは、プロジェクトの全体的なレポート作業の一部として、ドキュメントの目標とメトリックを含めることです。Meta の React Project の元ドキュメント マネージャである Rachel Lee Nabors 氏によると、プロジェクトの創設者でメンテナーのドキュメント チームに対する態度は、プロジェクトの長期的な健全性を示す良い指標です。プロジェクトの創設者とコア メンテナがドキュメントに積極的な関心を持つと、プロジェクトが成長を続け、新しいユーザーを惹きつける可能性が高くなると感じています。

ドキュメンテーション コーディネーターを雇います。

エンジニアリングの世界ではあまりにも愛されていない活動であるため、ボランティアだけに依存するドキュメンテーションは苦労し、リソースの不足や離脱率の高さに悩まされる傾向があります。ほとんどの重要なプロジェクトには、フルタイムの有給コーディネーターまたはドキュメンテーション リーダー、あるいは有償プロジェクト メンテナーがいて、ドキュメンテーションの一部または単独を担当しています。

ドキュメント作成タスクのエンジニアを採用するか、潜在的なドキュメント作成者を育成するための教育プロセスを作成します。

複雑なプロジェクトの現実では、ドキュメント作成者はコードがどのように機能するかを技術的に深く理解している必要があります。Nabors 氏によると、最高のドキュメント作成者の多くはエンジニアリングのバックグラウンドを持っています。エンジニアの中には、書くという創造的な行為を楽しむために、実際にドキュメンテーションを楽しんでいる人もいます。

コントリビューターに対して、マージとビルドのプロセスの一部としてPRを適切に文書化するよう要求します。

Salt は開発プロセスでフラグを使用して、すべてのコード提出とパッチがドキュメント要件を満たしていることを確認します。”Salt の構造では、すべてのモジュール、モジュール内のすべての関数には docstring があります。docstring を持たない Salt を介して実行する関数を持つモジュールを誰かが提供している場合、そのプル リクエストはマージされません”と、Salt のメンテナである Gareth Greenaway 氏は言います。Linux カーネルも同様のポリシーに従っています。とは言っても、コードのドキュメント化は必要とされるドキュメントの一部にすぎませんが、必要な基盤を提供します。

ドキュメンテーションのスプリント、フォーラム、その他の作業を形式化します。

多くの組織では、エンジニアがコーディングを中断してドキュメントの作成を支援する”ドキュメント スプリント”を採用しています。しかし、ドキュ

メント スプリントをカレンダー上で実行される定期的なイベントにしたり、プロジェクト全体の計画に含めたりする組織はほとんどありません。

文書作成者が、彼らの努力に対して、リード メンテナーから公に賞賛されるようにしてください。

リリース ノートですべてのドキュメント コントリビューターに言及したり、プロジェクト ミーティングやオンライン イベントで彼らに大声を出したり、ドキュメント チームに独自の贈り物をしたりすることは、プロジェクトが”docstars”に報いる方法のほんの一部です。ソーシャル メディアで彼らを呼びかけることは、ドキュメント ライターへの感謝を示すための、もう 1 つの低コストで影響力の高い方法です。

資金やその他の支援

OSS の作成をどのようにサポートするかは、議論の余地のあるトピックです。ほとんどのオープンソース プロジェクトは、サービスや製品の収益を直接収集するわけではありません。収益を収集するプロジェクトの多くは、単一の企業によって管理されています。サポートを 2 つのバケットに分けることが重要です。給与、スポンサーシップ、またはビジネス収益の形でのコントリビューターとメンテナー自身の財政的サポートと、プロジェクトとその組織的要件（運用、マーケティング、インフラストラクチャ、財務など）のサポートです。

貢献者と保守者の経済的支援は、このプロジェクトのインタビュアーのうち、たった一人だけが懸念していた問題でした。クリティカルなオープンソース・プロジェクトでは、大半のメンテナと中心的貢献者がフルタイムの雇用を享受しています。ほとんどの場合、その雇用によって、彼らは自分の時間の一部または全部をオープンソース・プロジェクトに費やすことができます。このプロジェクトのインタビュー対象者のうち、2 人を除く全員が、主にフルタイムの雇用という形で、オープンソース活動に十分な経済的支援を享受していました。インタビュアーのうち 2 人は、メンテナへの支払いや運営コストをまかなうために、主にスポンサーからの資金に依存しているプロジェクトのメンテナを務めたことがあるか、現在務めています。メンテナーの 1 人は、コンサルタント会社の設立を支援し、そのコンサルタント会社は、そのメンテナーが携わっていたオープンソースプロジェクトに焦点を当てたコンサルティングやサービスを提供していました。ほとんどのメンテナーは大規模な組織で働いており、そこでは、プロジェクトへの貢献や作業を継続することが明確な職務の一部となっていました。

> “ドキュメントに貢献したいと思っている人たちのクラスがありますが、ドキュメントの変更は Ceph プロジェクトでの GitHub コミットも意味するので、彼らは方法を知りません。ドキュメントについて苦情を言いたいが、適切なオーディエンスに到達できないクラスがあります。私はそのギャップを埋め、これらのタイプの人々の声を聞き、支援するために DocuBetter を作成しました。”
>
> —NEHA OJHA, Ceph

ある例では、単独プロジェクトである重要なオープンソース プロジェクトのメンテナーが、プロジェクトの 1 つに大幅なアップグレードが必要だが、オープンソースに資金を提供するための現在のメカニズムが彼への直接資金提供を困難にしているという不満を表明しました。同様に重要なのは、メンテナーが、追加の調査なしには全範囲を理解できないため、アップグレード作業の適切な価格設定方法さえ知らないと指摘したことです。

概して、LF Research は、成功したプロジェクトがビジネスや非営利団体と同様の方法で行動することを発見しました。これは、ソフトウェア開発が最優先事項であることを意味しますが、プロジェクトは、資金調達、マーケティング、運用サポート、インフラストラクチャなど、他の運用要件を満たすためにもかなりの継続的な努力を払っています。多くの

組織やスタートアップ ビジネスが、これらの分野の 1 つ以上に対処しようとしています。これらの取り組みの大部分は、スポンサーシップの増加と資金源の構築に焦点を当てています。スポンサーや他の資金提供者にとって本質的に興味のない他の運用面に焦点を当てている組織は少なくなっています。いくつかの回答者は、世界的に、政府は、生物学、材料科学と化学、応用物理学、およびその他の関連分野における継続的な研究開発のための技術ツールとツールの両方の開発を促進するための助成金でオープンソース プロジェクトを支援することの重要性を認識し始めたばかりであると指摘しました。インタビュー対象者は、資金調達と支援のための以下のベストプラクティスを確立したことを確認しました。

プロジェクトの資金または支援の定期的な財源があることを確認します。

当然のことながら、プロジェクトの財務的な実行可能性は、プロジェクトを成功させるための鍵です。これには、プロジェクトが問題を解決しているが、企業製品の中核部分ではない企業での内部資金、学術助成金または学者による進行中の研究を通じた間接資金（Ceph、Apache Spark など、学術研究プロジェクトとして開始された複数の主要なオープンソース プロジェクト）、コンソーシアムまたは財団を通じた直接資金、およびプロジェクトをコア製品の一部として使用しているが、コミュニティとオープンソースを純粋に競争的ではなく、そのビジネスモデルに追加的かつ補完的であると見なしている企業での直接資金（Chromatic がその例）など、さまざまな形態があります。単独および提携していないプロジェクトの場合、潜在的な資金源には、スポンサーシップ（GitHub スポンサー、LFX、Open Collective）、有償サポートおよび依存モデル（Tidelift）、および 1 回限りの財団または業界助成金（OpenSSL およびその他の主要なオープンソース セキュリティ プラクティスに対する OpenSSF 助成金）が含まれます。

小規模で独立したプロジェクトの場合は、資金調達が必要かどうかを判断し、必要な場合は、資金調達リーダーを指名するか、 支援ビジネスを設定します。

インタビューを受けた 3 人の小規模な独立したプロジェクト メンテナーのうち、2 人には、資金調達を担当する従業員またはプロジェクト リーダーとしてプロジェクトに従事する人がいました。3 人目は、自分のオープンソース プロジェクトでの作業を、それらのプロジェクトでの作業を含まない本業よりも優先していませんでした。フルタイムの資金調達メンテナー（他の非コード タスクも処理していました）がいた 2 つの小規模なプロジェクトは、スポンサーシップやその他の形態のサポートに基づいて、プライマリ メンテナーに生活賃金を支払うことができました。リード cURL メンテナーは、勤務時間の一部を、有償クライアントが cURL の問題を解決するのを支援したり、クライアントが cURL コードベースに含めたい機能を設計してコーディングするのを支援するために費やしました。大規模にスケーラブルなビジネス モデルではありませんが、リード メンテナーの給与を維持し、メンテナーの雇用者の利益を得るのに十分な収益を生み出しました。

小規模な独立プロジェクトでは、できるだけ多くの管理作業をオフロードする必要があります。

オープンソース プロジェクトの現実は、非営利組織であり、財務、マーケティング、インフラストラクチャ、内部 IT などの処理を含め、これらの組織が必要とするすべてのチェック ボックスをオンにする必要があるということです。オープンソース プロジェクトに従事する開発者は COO ではなく、仕事のその部分を楽しんだり、オープンソース プロジェクトを責任を持って管理したりすることはめったにありません。これは、財団に参加し、オープンソース プロジェクトへの運用サポートの提供に重点を置く組織にサインアップし、GitHub などのプラットフォームで無料の自動化ツールを利用して、新しいコード ビルドのデプロイからライセンス選択に至るまで、主要な運用タスクを実行することで実現できます。

多様性

他の技術分野と同様に、インタビューを受けた人の大多数にとって、多様性は依然として課題です。最も重要なオープンソース プロジェクトの中には、多様なメンテナーとコントリビューターが重要な役割を果たしているものはほとんどありません。中には、実質的に何もないものもあります。インタビューを受けたすべてのメンテナーが、メンテナーとコントリビューター ベースの多様性を改善したいと述べていますが、多くのプロジェクトでは多様性戦略が定義されておらず、多様性を向上させるためのプログラムが限られているか、プログラムがないものもありました。とは言うものの、包括的な言語の使用など、現在では一般的な多様性の取り組みもあります（コントリビューターの経験を参照）。

小規模で、プログラミングのより複雑な領域（ランタイム環境、データベース）で動作するプロジェクトは、大規模なコントリビューター ベースを必要としません。これらの小規模なプロジェクトやチームでは、多様性を改善するための正式な取り組みを維持することが困難な場合が多く、メンテナーは複数の優先事項を調整する必要があります。最も差し迫ったニーズはコードを出荷することです。なぜなら、プロジェクトに参加してコントリビューションしようとする潜在的なコントリビューターのベンチがないからです。この問題は、C や Java などの古くて技術的な言語で書かれたプロジェクトで特に深刻です。”ほとんどの人にとって、これは古い言語です。非常にニッチな言語であり、ほとんどの場合、それに取り組んでいるのは年配の人たちだけです”と Stenberg 氏は説明し、cURL には定期的なアウトリーチ以外に明確な多様性戦略がないと述べています。”新しいコントリビューターを獲得し、より多くの多様性を生み出すことで、より良いことをしたいと思っています。私は Twitter に助けを求めて投稿しました。私たちはコミュニティに尋ねました。私はどんな提案にも非常にオープンです。”

Google Summer of Code や Outreachy などの多様性を改善するプログラムは、すでに時間に追われている小規模プロジェクトに対して、追加的な責任と時間的コミットメントを表している、とこれらのプロジェクトのメンテナーは述べています。大規模なプロジェクトや、財団や傘下グループまたは大規模な企業と提携しているプロジェクトは、これらのアウトリーチ プログラムを追求する可能性が高いです。メンテナーは、インターンの質と貢献に関して様々な結果を報告しています。

より高度で開発されたプロジェクトには、より多様なプロジェクト リーダーシップを構築するためのいくつかのベストプラクティスを含む、特定の多様性戦略がありました。これらのベストプラクティスには、次のようなものがあります：

多様性と一体性をプロジェクトの第一目標にします。

多くの重要なプロジェクトは、多様性と一体性をちょっとした後知恵として扱っています。彼らには、多様性を構築するための常任委員会や明確な取り組みがありません。彼らには目標や願望があるかもしれませんが、多様性を推進するための具体的なメカニズムはほとんどありません。前向きなプロジェクト メンテナーは、多様性と一体性をプロジェクトの主要な目標として含めることを目指しています。例えば、Jupyter は最近のガバナンス変更で、多様性と一体性の取り組みをプロジェクトの最高レベルにまで大幅に引き上げました。”私たちの新しいガバナンス モデルには、ガバナンス モデルの重要な部分である DEI standing committee があります”と、プロジェクトの作成者の 1 人でリード メンテナーである Granger 氏は言います。”その組織は、例えば、Jupyter のソフトウェアの全体的な方向性について決定を下す Software Steering Council の席を持っています。私たちは多様性と一体性を優先していますが、このレベルで常に優先してきたわけではありません。”

メンタリングと多様性への取り組みを組み合わせます。

後にコア メンテナーになれるコントリビューターを採用することは、戦いの半分にすぎません。ほとんどの人はオープンソースの経験が不足しており、メンテナーによる積極的な指導の恩恵を受けることができます。メンテナーは、パラダイムの規則を説明し、プロセスとグループ ダイナミックスをナビゲートするのを助けることができます。”この多数のメンタリングにはいくつかの課題があり、オーバーヘッドを最小限に抑えるために慎重な計画が必要です”と、13 人の個人を指導する Linux プロジェクトの Khan 氏は説明します。営業時間は、質問への回答、リソースの共有、必要に応じたツールとデバッグ技術のデモンストレーションのためです。Khan 氏にとって、これらの新しいコントリビューターの一部は、Linux Foundation の LFX メンターシップ プログラムの一環として、より定期的なコントリビューターになるためのメンティーになりました。

Outreachyなど、多様性を高めるサードパーティのプログラムに参加します。

テクノロジー企業の多様性と役割の育成に特化したプログラムはいくつかあります。最も有名なのは Outreachy です。The Linux Foundation は、さまざまな多様性奨学金にも資金を提供しています。Abbott 氏は次のように述べています。”私は Outreachy を信じています。これは、さまざまなコントリビューターやその他の人々を呼び込むことができる素晴らしいプログラムだと思います。Fedora が Outreachy でインターンをほぼすべてのラウンドで受け入れ続け、彼らがエンジニアリングだけでなく、設計やドキュメント作成など、さまざまな分野で働いていることに興奮しています。”

メンテナの燃え尽き症候群の防止

重要なオープンソース プロジェクトを維持する仕事は、消耗的で消費的なものになる可能性があります。インタビューを受けた人は皆、メンテナーの燃え尽き症候群という考え方に精通していました。何人かのメンテナーは、燃え尽き症候群の一部はオープンソース開発の異なる性質から生じると述べています。すべての活動は公開されており、コメントと精査の対象となります。参加者とリーダーシップの間の意思決定のコンセンサスのみに依存するプロジェクトは、コンセンサスを構築するために追加の作業と相互作用を必要とすることが多く、燃え尽き症候群を加速させる可能性があります。npm パッケージ リポジトリと Node.js のメンテナーである Myles Borins 氏は、次のように述べています。”例えば、Node.js のようなプロジェクトには、オープン モデルを実行し、真にオープンに管理された運営委員会を持ち、オープン ガバナンス システムを備えた、ある種の魔法があると思います。しかし、存在する曖昧さと、コンセンサス モデルと、多様な対立を抱えたり、常に親切にしたいだけの多くの人々とのバランスをとることの難しさは、ある程度、人々を燃え尽きさせる可能性のあるパターンをもたらすと思います。”

ほとんどのインタビュー対象者は、特定の”生存”の実践と戦略を作成していました。これらは複雑でも詳細でもなく、ほとんどが常識です。とは言っても、これらの実践を調査することは依然として有用です。燃え尽き症候群の予防方法には、次のようなものがあります。

> “そこでは Maslow のヒエラルキーのようなものが起こっています。メンテナーが十分なお金を持っていれば、彼らは別の仕事で過労する必要はありませんし、彼らは医療問題や住居や食料などのことを心配する必要もありません。そうすれば、燃え尽き症候群のリスクはかなり低下すると私は思います。”
>
> —JORDAN HARBAND, MAINTAINER
> QS と ES5-SHIM のメンテナー

決して”仕事を終わらせる”ことはできないことを認識します。

プロジェクトが終了するまで、オープンソースの作業は決して完了しません。PR やイシューは常に読まれ、回答されるのを待っています。成功したメンテナーは、自分たちの作業が決して終わらないことを認識し、受け入れています。”個人的に達成できないことについては、常に現実的であることだけです。それを他のこととバランスさせても大丈夫です。to-do リストのすべてを完了しなくても大丈夫です。なぜなら、オープンソースのメンテナーシップは、放っておけば無限の時間を埋めることができるからです”と Abbott 氏は説明します。

ハイブリッド ライフスタイルを受け入れます。

燃え尽き症候群を経験していないメンテナーの中には、オープンソースの流れを回避するように人生を設計した人もいます。彼らは実際にプロジェクトから離れることはありませんでしたが、オープンソースの柔軟性を利用して、健全な外部活動を維持し、家族や友人と時間を過ごしました。例えば、’qs’ のメンテナーである Jordan Harband 氏は、オープンソースのメンテナーシップとガバナンスの仕事から休暇を取らないことを選択しましたが、パートナーや子供たちと時間を過ごし、運動し、友人

と過ごすことを可能にするライフスタイルを構築しました。これは万人向けではありませんが、彼にとってはうまくいったと彼は述べています。

無償の趣味として、急速に拡大し、管理オーバーヘッドを必要とするオープンソース プロジェクトに取り組むことは賢明ではありません。

これはインタビューを受けた人たちからの一般的なアドバイスであり、彼らの大多数は、急速に成長しているオープンソース プロジェクトに従事するために十分な報酬を得ていました。多くは楽しみのためにオープンソース プロジェクトに副業として従事していましたが、これらのプロジェクトのどれも、有料のオープンソース作業と同じレベルの管理と調整の作業を必要としませんでした。”これが必ずしも報酬を得ているわけではなく、勤務時間外に楽しくないことに余分な時間を費やさなければならないのであれば、メンテナーの燃え尽き症候群を引き起こすという点では別の話かもしれません”と、ビルドとリリース ツールに焦点を当てた PyTorch メンテナーである Eli Uriegas 氏は指摘します。

ワークフローを最適化し、自動化するための効率性のハックを常に探します。

スーパー メンテナーは、プロジェクト アクティビティをフィルタリングし、最も重要なアクティビティと問題に集中できるようにするために、オーダーメイドのセットアップを行う傾向がありました。これには、電子メール ラベルの積極的な使用、GitHub でのフィルター、ワークフローを自動化するためのボットの使用、またはプロセスを自動化したり、より大きな焦点を可能にするその他のメカニズムが含まれます。Webpack の Tobias Koppers 氏のような一部のメンテナーにとって、これを行う最善の方法は、1 つのチャネル（彼の場合は GitHub 内）でのみ会話に応答することです。

境界を設定し、それに従います。

経験豊富なメンテナーは、さまざまなパラメータの境界に厳密に従うと述べています。例えば、ほとんどの人はプロジェクトに関する Twitter の会話に参加することを拒否しています。ほとんどの人は、プロジェクトに関するプライベートな電子メールの会話を避け、すべての議論を公開された GitHub リポジトリや電子メール リストサービスにルーティングしたいと考えています。メンテナーの中には、プロジェクトの GitHub リポジトリ外のコメントやメッセージを単に無視する人もいます。さらに、多くのメンテナーは、友人や家族と時間を過ごすことを保証するために、就業日の境界を設定しています。コミュニティが境界を理解すると、それを尊重し、それに従う傾向があることがわかりました。

燃え尽きたと感じたら離れてください。

複数のメンテナーが、燃え尽き症候群を感じ始めたら、休暇を取るか、プロジェクトの作業を中断すると言っています。これには、燃え尽き症候群がどのようなものであるかについての自己認識が培われたことも一因です。ほとんどのメンテナーは、大規模なバージョン プッシュ、年次プロジェクト計画セッションまたはオンライン イベント、脆弱性修復作業など、重要な期間の例外を除外しています。

結論

30 人以上のメンテナーへのインタビューを通じて、LF Research は多くの驚くべき話を聞き、最も成功したプロジェクトのいくつかが、オープンソース プロジェクトの設立、育成、成長という無数の課題を効果的にナビゲートした方法を学びました。LF Research は、これらの洞察が、成功したメンテナシップとプロジェクトの形成と管理のパターンとアンチパターンを文書化することによって、将来の世代のメンテナーを可能にすることを期待しています。これらのパターンは、オープンソース プロジェクトとそのメンテナーが直面する多くの一般的な問題に対する詳細な状況ガイダンスを提供するとともに、これらの問題を克服する方法、さらにはこれらの問題が発生して重大な関心事になるのを防ぐ方法についてのアイデアを提供することができます。詳細な逸話と提案の中で、LF Research は、メンテナーにプロジェクトを管理するための基本的なツールキットと、ポジティブなワーク／ライフ バランスを作り出すためのアイデアを提供することを目的としています。このレポートは、膨大なトピックの表面をかすめているだけです。簡潔にするために、私たちの研究からの多くの有益なメンテナーの話は、この論文には含まれていません。とは言うものの、これらのインタビューから、LF Research は、プロジェクト経験とプロジェクト コード品質の両方を改善できるメンテナーのためのいくつかの具体的なアクション アイテムを獲得しました。

プロジェクトの属性を決定します。

次に示す 4 つのカテゴリからプロジェクト属性を特定することは、メンテナー戦略を作成する上で重要です（表 1 を参照）。たとえば、重要度の高い小規模で複雑なプロジェクト（OpenSSL がその好例です）では、高度なスキルを持ったメンテナーを採用することと、そのプロジェクトに依存する企業やその他の企業から資金源を確保することの両方を検討する必要があります。また、重要度の高いプロジェクトは、プロジェクト ライフサイクルのできるだけ早い段階でコミュニティの意見を取り入れて設計および実装された、透明性のあるコミュニティベースのガバナンス プロセスから利益を得ます。Kubernetes のように、資金が豊富な非常に大規模で複雑なプロジェクトは、ガバナンスと中立性を促進するという問題に直面します。重要度が中程度で複雑度が中程度のプロジェクトは、メンテナーを引き付けるのに苦労する可能性があるため、企業スポンサーを探す方がよい場合があります。属性の各組み合わせには独自の推奨事項が付属しており、汎用的なプレイブックはありません。とは言っても、特定の属性タイプの共通パターンは、プロジェクトの戦略と方向性を示すのに役立ちます。プロジェクトは時間の経過とともに変化し、進化します。しかし、最初から、4 つの属性に関してプロジェクトがどこに位置しているかをある程度把握することは可能です。

表 1

適切なメンテナー戦略を作成するためのプロジェクト属性

サイズ	小	中	大
サポート	資金なし	多少の資金	充分な資金
複雑さ	単純	中程度	複雑
重要度	低	中	高
ライフサイクルの段階	スタートアップ	急成長	成熟

属性に基づいて戦略とロード マップを作成します。

オープンソース プロジェクト管理は場当たり的であることが多く、構造が多すぎると生産性が阻害される可能性があることを認識しているため、メンテナーは、プロジェクトの大まかな戦略を作成して、アクティビティに焦点を当て、コントリビューターや共同メンテナーを導くことが有用であると考えるでしょう。この戦略ドキュメントは、アクティビティが属性に一致することを確認するために、年 1 回再確認する価値があります。たとえば、プロジェクトがテクノロジー エコシステムの重要な依存関係になる方向に進んでいる場合、プロジェクトは追加のセキュリティ アクティビティを優先する必要があります。プロジェクトが明らかに初

期段階から急速な成長に移行している場合、メンテナーはオンボーディング インフラストラクチャの構築とドキュメントのランピングにより多くの時間を費やす必要があります。ここでも完璧なレシピはありませんが、属性とステージに関するメンテナーシップへの思慮深いアプローチは、常にノイズの多いプロセスで重要なことに焦点を当てることを可能にします。

プロジェクトの主要な指標を特定し、定期的に追跡します。

プロジェクトとコミュニティの健全性は非常に主観的であり、メンテナーとコミュニティの成功の定義に依存しています。たとえば、主に大企業によって支援されているフロントエンド フレームワークは、新しいコントリビューターのパイプラインや、コントリビューターが絶えずバグ レポートを提出し、修正を提案しているかどうかにはあまり関心がありません。また、スタックの奥深くにあり、コーディングが複雑なプロジェクトは、コミュニティの成長よりも、既存のコミュニティ パフォーマンスの出荷日やコードの質などを測定することを好む場合があります。とは言うものの、プロジェクトの健全性指標を決定し、それらを測定することは、より良いプロジェクト管理とより生産的なメンテナンスのための重要なステップです。

CHAOSS プロジェクトは、コミュニティの健全性指標の膨大なリストを提供しており、メンテナが選択できるパレットと優れた出発点を提供しています。

プロジェクトのベストプラクティスを特定します。

オープンソース プロジェクトの形成とメンテナンスは圧倒的なものになる可能性があります。焦点と明確さが重要です。上記の 3 つのガイダンスは、フレームワークと戦略を示しています。プロジェクトの健全性に影響を与え、望ましい分野でプロジェクトの改善を推進するための戦術的なステップとプロセスを導入することは、プロジェクトのリーダーシップとコミュニティが協力してプレイブック、ベストプラクティス、および特定の作業方法をレイアウトする場合に最も効果的に機能します。プロジェクトがさまざまな分野にわたって詳細なガイダンスをレイアウトすることは可能ですが、最も大規模でリソースの豊富なプロジェクトを除くすべてのプロジェクトで、メンテナーは、最も影響が大きいと特定された分野とメトリクスにベストプラクティスを集中させたいと考えるかもしれません。たとえば、Salt は現在、コア プロジェクト全体でより高品質のコードとより少ないコードを必要とする規模になり、より多くの機能が Salt-community-managed extensions に移行しています。この目標を達成するために、Salt はテスト コンプライアンスを提出プロセスの必須部分にしました。これまでは、プロジェクトがコア プロジェクトのコードベースに新しいコントリビューターを集めることに重点を置いていました。これは、プロジェクトの焦点を絞った分野に影響を与えるためにベストプラクティスを活用する方法を強調する小さな例です。

LF Research は、これが会話の出発点となり、成長するメンテナー コミュニティの助けを借りて、LF Research がこの知識、ヒューリスティックス、観察の本体を成長させて、この最初の取り組みよりも広い範囲の状況とタスクをカバーできることを願っています。オープンソース メンテナーは、グローバルなイノベーションを加速し、グローバルな課題を解決し、人々の生活を向上させる企業、政府、非営利の技術を構築する上で重要な役割を果たしています。参照知識の本体とこの分野のコンピテンシーのより大きなプールを作成することは、メンテナーとコントリビューターが自分自身を助けるのを助け、私たちが以前に来た人たちから学んだことを基礎にすることで、最終的にオープンソースをより有用で成功させることになります。

謝辞

このレポートは、多くのオープンソース プロジェクト メンテナーの助けがなければ不可能でした。彼らは時間をかけて
彼らの仕事について私たちと話し、彼らの知恵と洞察を共有してくれました。私たちは感謝したいと思います。

- Rachel Lee Nabors, Brian Granger, Phil Estes (Amazon Web Services)
- Myles Borins, Andres Freund (Github, Microsoft)
- Eli Uriegas, Nikita Shulga (Meta)
- Shuah Khan (Linux Foundation)
- David Edelsohn, Neha Ojha (IBM)
- Siddesh Poyarekar, Carlos O’Donnell (Red Hat)
- Anil Sharma, Gareth Greenaway, Pedro Algarvio, Megan Wilhite (VMware)
- Norbert de Langen (Chromatic)
- Daniel Stenberg (wolfSSL)
- Laura Abbott (Oxide Computer)
- Tobias Koppers (Vercel)
- Joel Orlina, Jason Swank, Hervé Boutemy (Sonatype)
- Sterling Greene (Gradle)
- Liz Rice (Isovalent)
- Ralf Gommers (Quansight)
- Henry Zhu
- Jordan Harbend

著者について

Alex Salkever 氏は、”The Driver in the Driverless Car”や”Your Happiness Was Hacked”など、テクノロジーがビジネスと社会に与える影響に関する 4 冊の本の共著者です。彼は多くのテクノロジー企業で製品とマーケティングのシニア リーダーシップの役割を果たし、BusinessWeek のテクノロジー エディターも務めました。長年のオープンソース支持者であり、2016 年から Linux Foundation の特別プロジェクトに参加しています。

この日本語レポートは、以下の文書の参考訳です。
Open Source Maintainers

翻訳協力：橋本修太

THE LINUX FOUNDATION | Research

2021 年に設立された Linux Foundation Research は、オープンソース コラボレーションの規模の拡大を調査し、新たな技術動向、ベストプラクティス、オープンソース プロジェクトの世界的な影響に関する洞察を提供しています。プロジェクトのデータベースやネットワークを活用し、定量的・定性的手法のベストプラクティスに取り組むことで、Linux Foundation Research は、世界中の組織のためにオープンソースの知見を提供するライブラリを構築しています。

この著作物を参照する場合は、以下のように引用してください:
Alex Salkever,“ Open Source Maintainers: Exploring the people, practices, and constraints facing the world’s most critical open source software projects,” foreword by Shuah Khan, The Linux Foundation, July 2023.