Improving Your Open Source Development Impact

企業においてオープンソース開発の効果を大きくする

Open source development requires a different approach than many organizations are accustomed to. But it becomes easier if you have a clear plan to follow. Fortunately, so many companies and individuals have already forged a path to success in contributing to significant open source projects. They have tried and true methods for establishing a leadership role in open source communities.

オープンソース開発は、多くの企業が慣れ親しんできた従来のやり方とは違ったアプローチを必要としますが、明確な計画を持って取り組みさえすれば、より容易になりえます。幸いなことに、多くの企業と個人が重要なオープンソースプロジェクトへの貢献において成功の道筋をすでに作り上げています。彼らは試行の末に、オープンソースコミュニティでリーダーシップを確立する方法を見出しているのです。

This practical guide will help you and your company improve your internal development process and prepare you to contribute to the open source projects that matter most to your company. Contributing to the Linux kernel is one of the hardest challenges for open source developers. So we’ll use this case as an example for this guide. Fortunately, the guidance will apply to nearly any open source project you’ll face.

この実用的ガイドは、企業内の開発プロセスを改善し、あなたの会社がもっとも重要視するオープンソースプロジェクトに貢献する準備を整えることを手助けします。Linuxカーネルに貢献することは、オープンソース開発者にとってもっとも厳しい挑戦の一つです。本ガイドでは、Linuxカーネルを事例として取り上げますが、本ガイドの指針はほとんどすべてのオープンソースプロジェクトに適用できます。

Contents

目次

What is impactful open source development?

Role of an open source program in improving development

Common areas for improvement

Top recommended practices

Metrics for tracking progress

Final words

[効果のあるオープンソース開発とは？](#影響力のあるオープンソース開発とはどんなもの？)

[開発の改善](#オープンソースプログラムが果たす開発方法の改善)に果たすオープンソースプログラムの役割

[共通的領域の改善](#共通領域の改善)

[推奨される取り組み、トップ10](#推奨される取り組み、トップ10)

[強化の進捗状況を追跡して調べる評価尺度](#進捗状況を調べる評価尺度)

[結論](#結論)

View All Guides »

Contribute on GitHub »

Contributors to this Guide

Ibrahim Haddad

VP of R&D and Head of the Open Source Group

Samsung Research America

R&D部門VP兼オープンソースグループ長、Samsung Research America

Gil Yehuda

Senior Director of Open Source

at Oath (Yahoo + AOL)

Open Sourceシニアディレクター、Oath (Yahoo+AOL)

Section 1

What is impactful open source development?

セクション　1

効果のあるオープンソース開発とは？

Developer contributions are one of the primary ways that your open source program will achieve the business goals you’re supporting with your organization’s open source strategy. There are many ways to influence the direction of an open source project, from testing code in different environments and adding to the code documentation, to funding the project or foundation and participating on the project board, or by using the code in other open source projects. But your company will exert the greatest influence in open source projects through the quality, quantity, and consistency of its code contributions. So it is in your organization’s best interest to provide the tools and processes that will help your team develop high-quality, effective open source code.

開発者の貢献は、企業が実行するオープンソースプログラムのビジネス目標－その目標は、企業のオープンソース戦略によって支えられる－を達成する主要な活動のひとつです。オープンソースプロジェクトの方向性に影響を与える方法には、多くの異なった環境でのテストやコードに関連したドキュメントの追加から始まって、プロジェクトやそれに関連したファウンデーションへの財政的援助や、プロジェクトの運営会議への理事としての参加に至るまで、あるいは、当該コードを他のオープンソースプロジェクトで利用するなど、たくさんのやり方があります。しかし、企業がオープンソースプロジェクトに一番大きな影響力を発揮できるのは、コード貢献の質、量、および、一貫性を通じたものです。したがって、企業の開発チームが高い品質の効果的なオープンソースコードを開発することを助けるような手段とプロセスを提供することは、企業の最善の利益となります。

Impactful open source code can influence the technical direction of the open source projects you contribute to, which in turn improves your products while minimizing the cost of maintaining that code.

影響力の大きなオープンソースコードは、企業が貢献しているオープンソースプロジェクトの技術的方向に影響を与えます。また、企業の保守費用を抑えながら、当該企業の製品を強化します。

The goal is to improve your development team’s efficacy through and with open source contributions. By helping your organization implement some of the recommended open source development best practices laid out below, you can:

目指すところは、オープンソース貢献を通じて、また、貢献によって、企業の開発チームの有効性を向上させることです。本ガイドの推奨するオープンソース開発ベストプラクティス（ある結果を得るのにもっとも効率のよい手法、プロセスなど）のいくつかを実装すれば、次のようなことが可能となります。

Reduce the amount of work needed from product teams

Minimize the cost to maintain source code and internal software branches

Improve code quality

Produce faster development cycles

Produce more stable code to serve as the base for products

Improve company reputation in key open source communities.

　　製品部門の作業量を削減

　　企業内のソースコードや独自ソースツリーを保守するための費用を最小限にする

　　コードの質を向上

　　開発サイクルの短縮

　　製品の基盤となるコードの安定化

　　重要なオープンソースコミュニティにおける企業の評価向上

Section 2

The role of an open source program in improving development

セクション　2

開発の改善に果たすオープンソースプログラムの役割

Your open source program can have both direct and indirect positive impacts on your products — whether they are consumed internally as infrastructure or services, or shipped as a commercial offering. These are impacts that you can measure and improve. They also provide tangible results to report to executive leadership – helping to show the value of your open source program and deliver ROI.

企業のオープンソースプログラムの成果物は、社内のインフラやサービスとして使われているか、あるいは、商用の提供物として外部に出荷されているかにかかわらず、企業の製品に対して明確な直接的効果と間接的効果の両方をもたらします。これらの効果は、その大きさを測定もできれば、さらに大きくすることもできます。また、目に見える結果として経営幹部に報告し、オープンソースプログラムの価値を示し、また、ROI(投資に対応した利益)を生み出すこともできます。

Direct enablement

直接的な効果

Your open source program can directly impact your organization’s open source code development by contributing the code itself. Every company structures its open source program and engineering contributions according to what best fits its own business goals and corporate structure (See the program structure section in our guide on Creating an Open Source Program). But contributing code is the best way to gain influence in an open source project and build your company’s reputation in the open source community.

企業のオープンソースプログラムは、コードを貢献することが、企業のオープンソースコード開発に直接的な効果を与えます。すべての企業は、ビジネス目標と企業構造に最も適した方法で、そのオープンソースプログラムを構成し、技術的貢献を行います。（本ガイド集のひとつ「オープンソースプログラムの創生」の中の「プログラム構造」のセクションをご参照ください。）しかし、コードを貢献することは、オープンソースプロジェクトに影響を与え、オープンソースコミュニティで当該企業の評価を上げる最善の方法なのです。

At Samsung, the open source program has its own dedicated engineering team which fulfills open source development requests from the company’s R&D and product teams. The program also helps bring internal Samsung code into various open source projects. And it has implemented numerous drivers related to Samsung products into upstream code. Product teams have their own developers who contribute to open source projects but they have less freedom as they are tied to product development. So the open source group receives requests such as “we need feature X implemented in the Kernel” and their engineering team delivers the code to product and to the Linux kernel.

Samsungにおいて、オープンソースプログラムは専任の技術チームを擁しており、R&D部門やプロダクト部門のオープンソース開発要請に応えています。同プログラムは、Samsungの社内で開発されたコードをいろいろなオープンソースプロジェクトに提供することにも手を貸しており、また、Samsung製品に関連した数多くのドライバーをアップストリーム向けに実装しています。プロダクト部門にもオープンソースプロジェクトに貢献する開発者がいますが、プロダクト開発に縛られているために、自由な行動が制約されています。そのため、オープンソースグループは、「カーネルに機能Xを実装する必要がある。」というような要請を受け、同グループの技術チームはプロダクト用のコードを提供し、さらに、Linuxカーネルにも提出します。

“Our expertise in committing code upstream has made us particularly valuable with these open source components, and our efforts have resulted in a reduction in the amount of effort required to maintain kernel code that’s used in Samsung products and services.”

「アップストリームにコードをコミットするノウハウのために、当該オープンソースコンポーネントにおいて、私たちは特に価値ある存在となっています。また、その活動によりSamsungの製品とサービスで使われているカーネルコードを保守するための総作業量が削減できています。」

Ibrahim Haddad – Vice President of R&D and Head of the Open Source Group at Samsung Research America

[Ibrahim Haddad](https://twitter.com/ibrahimatlinux) – R&D部門VP兼オープンソースグループ長、Samsung Research America

Open source program offices that are focused only on policy and process don’t always have a dedicated engineering team for contributing code. Instead, engineers across departments allocate a portion of their time to pushing code upstream for any changes they make to open source components.

ポリシーとプロセスにのみ特化したオープンソースプログラムオフィスは、コードの貢献のための専任の技術チームを通常持ちません。その代わり、いろんな部門の技術者が一定の時間を捻出して、オープンソースコンポーネントに対して行った変更をアップストリームに提出します。

Regardless of how the engineering team is structured, the goal is to minimize the technical debt with respect to those components. If upstream contributions do not happen, the product team will be stuck with large code bases that are out of sync with upstream. They’ll spend their time back-porting the updates to their out-of-sync fork instead of advancing the product.

技術チームがどのような組織に属していようと、目指すところはそれらのコンポーネントに関する技術的な借金を最小限にすることです。もしもアップストリームへの貢献がなければ、製品部門はアップストリームと同期していないコードベースを保守し続けることでしょう。製品の強化のためではなく、更新部分を同期されていない企業独自のソースツリーにバックポートするために時間を取られるのです。

Often an open source developer embedded in a product team will have a hard time striking any balance between upstream work and responsibilities (as a committer or maintainer) and their role in making the product happen.

多くの場合、製品部門で働くオープンソース開発者は、アップストリームの活動やそこでの責任（コミッターとして、あるいは、メンテナーとして）と、製品を実現するための役割との釣り合いをとるのに苦労します。

Each open source program office is uniquely structured depending on the needs of the business. Facebook, for example, has a dedicated Tools team within the open source program office that is responsible for building internal tools to help manage its open source portfolio. This includes the projects that Facebook shares, which are mostly hosted on GitHub, as well as the other external projects they contribute to such as the Linux kernel. (See our case study on Facebook’s open source program.)

各社のオープンソースプログラムオフィスは、ビジネスのニーズに合わせた組織構造を採用しています。たとえば、Facebookでは、オープンソースプログラムオフィスにツール専任のチームを持ち、同社のオープンソース資産の管理をやりやすくするための社内ツール群の構築に責任を持っています。そこには、Facebookが共有するプロジェクト（その多くがGitHub上で管理）もあれば、Linuxのように同社が貢献している外部プロジェクトもあります。（[Facebookのオープンソースプログラム](https://github.com/todogroup/guides/blob/master/casestudies/facebook.md)に関するケーススタディをご参照下さい。）

Indirect Enablement

間接的な効果

The open source program’s impact goes beyond the code the organization contributes to the various open source projects, however. From public relations and marketing, to legal support, developer training, and more, an open source program enables development in myriad ways. Here are four ways that the program can indirectly improve development through non-code contributions:

企業のオープンソースプログラムの効果は、さまざまなオープンソースプロジェクトに貢献するコードだけではありません。企業のパブリックリレーションやマーケッティングから始まって、法務サポート、開発者トレーニングなどに至るまで、オープンソースプログラムは、さまざまな切り口で開発に効果を及ぼしています。ここでは、オープンソースプログラムが間接的に開発の改善に寄与する、コード貢献以外の4つの切り口を示します。

External technical discussions. One major area to make indirect contributions is by influencing the various communities through participation in technical discussions. Your open source developers can do this by being active on mailing lists and IRC chats to participate in discussions and stay informed on the latest project updates. And on larger projects with formal governance structures you can sit on technical steering committees.

外部で行われる技術討議。間接的な貢献を行うことができる領域のひとつは、技術的な討議への参加を通じたものです。オープンソース開発者は、メーリングリストやIRCチャット上で積極的に活動することによってこれを行い、それにより、討議に参加し、プロジェクトの最新情報を常に受け取ることができます。しっかりとしたガバナンスの仕組みを持つ大きなプロジェクトでは、テクニカルステアリング委員会に委員として参加することも可能です。

Internal technical discussions. Internally, open source developers can participate in policy and architecture discussions to ensure the organization’s decisions match the direction of a specific project community. Open source developers should be present for any strategy discussions related to long-term planning for products that rely on open source code, for example.

社内で行われる技術討議。企業の決定が関連するプロジェクトコミュニティの方向性と適合することを確かなものとするために、オープンソース開発者は社内のポリシーやアーキテクチャーの討議に参加することができます。オープンソース開発者は、たとえば、オープンソースコードに依存する製品の長期計画に関連した戦略討議に立ち会うべきです。

Compliance support. Program managers can also provide assistance to resolve compliance issues and support the compliance team with the open source compliance inquiries they receive.

コンプライアンスの支援。プログラムマネージャーは、コンプライアンスの課題解決に援助を提供し、また、会社が受け取ったオープンソースコンプライアンスに関する問い合わせに対してコンプライアンスチームを支援することもできます。

Bug fixes. Finally, you can also help stabilize the code of the various open source projects you’re involved in. Find and fix bugs, test the fix and submit it upstream. This improve the code’s overall value for all the project’s users, including your company.

バグフィックス。最後に、当該企業が参加しているさまざまなオープンソースプロジェクトのコードを安定化させるために協力できます。バグを発見してフィックスする、バグフィックスをテストして、さらに、アップストリームに提出することもできます。これにより、当該企業も含めて、すべてのプロジェクトユーザーのためにコード全体の価値を向上させます。

Section 3

Common areas for improvement

セクション　3

共通的領域の改善

There are three general areas in which open source programs can improve developer productivity and efficacy in open source projects: culture, processes, and tools. Within each of these categories, there are several elements that you will need to address to fit the open source model.

企業のオープンソースプログラムが、オープンソースプロジェクトにおける開発者の生産性と有効性を向上させる3つの領域があります：文化、プロセス、そして、ツールです。それぞれのカテゴリーに、オープンソースモデルを当てはめるには注意を要する要素がいくつかあります。

Culture

文化

Development model

Collaboration

Transparency

Meritocracy

Team formation

Hiring practices

Right success metrics

開発モデル

協業

透明性

能力主義

チーム構成

採用活動

正しい成功尺度

Processes

プロセス

Governance

Usage

Compliance

Contribution

Approvals

Operational model

ガバナンス

使用

コンプライアンス

貢献

承認

業務モデル

Tools

ツール

IT infrastructure

Development tools

Tracking metrics

Knowledge sharing

Code reuse

ITインフラ

開発ツール

追跡して調べるべき評価尺度

知識の共有

コード再使用

Challenges dedicated open source teams face in an enterprise setting.

専任オープンソースチームが企業の現場で取り組むべき課題

Culture

文化

Cultural challenges often stem from the fact that there is a gap between traditional software development practices and the requirements of open source development. You can bridge this gap by hiring open source experts and having them train other groups that aren’t familiar with the open source development model.

文化的な課題は、従来型のソフトウェア開発手法とオープンソース開発の要件との間にギャップがあるという事実から生じます。オープンソースの専門家を雇い入れ、オープンソース開発モデルに不慣れなメンバーを教育してもらうことによって、このギャップに橋を渡すことができます。

“We aim to also provide open source engineers with adequate time to fulfill upstream responsibilities so they can provide adequate open source leadership for Samsung.”

「オープンソース技術者にアップストリームの責任を果たすのに十分な時間を与えることで、彼らがSamsungにオープンソースのリーダーシップをもたらすことを狙っています。」

Ibrahim Haddad – Vice President of R&D and Head of the Open Source Group at Samsung Research America.

[Ibrahim Haddad](https://twitter.com/ibrahimatlinux) – R&D部門VP兼オープンソースグループ長、Samsung Research America

Processes

プロセス

Open source development is dynamic, moves very quickly, and has special requirements for compliance. Companies that don’t adapt their internal processes to meet this type of development can easily get left behind. Developers need to be enabled to contribute code upstream quickly and any internal code policies need to be modified to allow this.

オープンソース開発は、ダイナミックで、とても早く動きますが、コンプライアンスについて特別な要件があります。このような開発スタイルに合うように内部プロセスを順応させていない企業は、簡単に取り残されてしまいます。開発者は、素早くアップストリームにコード貢献することが可能となっている必要があり、社内のコードポリシーはこのような活動を容認するように変更される必要があります。

First, it’s vital to have a team in charge of maintaining proper open source compliance to avoid legal problems. (See our ebook, Open Source Compliance in the Enterprise.)

最初に、法務的な問題を避けるために、適切なオープンソースコンプライアンスに責任を持つチームを作ることは必須です。(Linux Foundationの電子書籍、[Open Source Compliance in the Enterprise](https://www.linuxfoundation.org/publications/open-source-compliance-enterprise/)をご参照ください。)

You’ll also need to have a simple internal approval model for open source use and contributions. Throughout the years, Samsung has moved from highly complex and cumbersome policies to a more simple approach for receiving, reviewing, and approving source code contributions. It’s a function of balance between all parties involved: legal, engineering, and open source. The compromise we have now supports the dedicated open source team which has a blanket approval to contribute to a number of open source projects. This is not the case for other teams who need to get different levels of approval depending on the nature of the code being contributed (e.g. simple bug fixes, code to improve existing functionality, code that offers new functionality, or starting a new project).

また、企業はオープンソースの活用と貢献のために、わかりやすい社内承認モデルを作る必要があります。ここ数年の間に、Samsungは、ソースコードの受け取り方法、レビュー方法、および、貢献の承認方法について、複雑でやっかいなポリシーからより分かりやすいやり方への移行を完了させました。それは関連するすべての組織、すなわち、法務部門、技術部門、および、オープンソースチームの間でバランスを取ったたものです。現時点の合意では、専任のオープンソースチームの活動を後押しており、同チームがたくさんのオープンソースプロジェクトに貢献することに対して一括承認を与えています。しかし、他のチームでは貢献するコードの性質に依存して異なったレベルの承認を必要としています。たとえば、バグフィックス、既存機能の強化のためのコード、新機能を提供するコード、あるいは、新規プロジェクトの立ち上げなどです。

Tools

ツール

You’ll also want to ensure your tools are compatible with the open source development model from the start. Create a setup that fulfills the needs of the open source program office and also meets corporate IT guidelines.

活動の当初から、使用するツールをオープンソース開発モデルと共存できるものにした方がよいでしょう。オープンソースプログラムオフィスのニーズをみたし、かつ、企業のITガイドラインに合った構成を作ってください。

Samsung, for instance, issues Linux devices that work with all of the tools its engineers need to participate in open source development. This environment allows developers to join the team without requiring any major changes to the way they work. They also support work from home and in fact have only very few open source developers in their Silicon Valley offices — all other staff are remote across the globe.

たとえば、Samsungでは、同社の技術者に、オープンソース開発に参加するのに必要なあらゆるツールが正しく動作するLinuxデバイスを支給しています。この環境により開発者は、通常の作業法に何の変更の必要もなくオープンソースプロジェクトのチームに参加できます。また、同社は在宅作業を後押ししており、実際にごく少数のオープンソース開発者だけがシリコンバレーのオフィスで働いています。他のスタッフは全世界の遠隔地に拠点を持っています。

Section 4

Top Recommended Practices

セクション　4

推奨される取り組み、トップ10

First, a bit of history. December of 2000 marked a major milestone in the history of open source when IBM pledged to spend $1 billion on Linux R&D. IBM was a true pioneer in the enterprise world putting its bet on Linux and open source when very few companies did — and definitely not at that scale. IBM had to learn a lot about working with open source software and the various project communities they were involved in.

少し歴史を振り返ってみましょう。2000年12月、オープンソースの歴史に画期的な出来事がありました。IBMがLinuxの研究・開発に10億ドルを投入することを公約したのです。IBMは、Linuxとオープンソースに賭け金を投じた企業世界の真のパイオニアでした。当時、そのような企業はほとんどありませんでしたし、まして、そのような規模の投資はありえない状況でした。オープンソースソフトウェアや、同社が参加したさまざまなプロジェクトコミュニティで活動するために、IBMは多くのことを学ばなければなりませんでした。

That was the starting point for enterprise open source and companies have since followed IBM by the dozens and then hundreds. Yet, thousands of companies are still entering open source as it is becoming the new normal of software development. The question is: How can we minimize the enterprise learning curve and speed up the process of “getting it right?”

それは、大企業のオープンソース参入のスタート点でした。それ以後、数十の、さらに後には数百の企業がIBMを追随しました。今でも、数千の企業がオープンソースに参入しています。オープンソースがソフトウェア開発の新たな標準になりつつあるからです。問題は、どうすれば企業の学習曲線を最小限とし、「正しく理解する」プロセスをスピードアップできるかというところなのです。

The answer is easy, but the application of that answer to specific company culture is the hard aspect. Let’s explore some of the “easy” aspects of learning from over 17+ years of enterprise experience with open source software. You’ll get to deal with the hard aspect of applying those lessons to your particular enterprise environment.

答えは簡単です。けれども、その答を特定の企業文化に適用させる場合は難しい側面も存在します。オープンソースソフトウェアにおける17年以上にわたる企業の経験を学び、容易に適用できる側面から探っていきましょう。ここで学んだことを特定の企業環境に適用するという難しい側面にも対応できるようになるでしょう。

A word of caution: In addition to implementing these practices, you will need to lead a cultural shift from traditional software development practices to a more open and collaborative mindset. Internal company dynamics need to be favorable to open source efforts. As an open source leader inside your organization, you will face several challenges in terms of funding resources, justifying ROI, getting upstream focus, etc. These often require a major shift in mindset and a lot of education up the chain.

一言だけ注意：これらの取り組みを実装することに加えて、従来型のソフトウェア開発の手法から、もっとオープンで協業にふさわしい考え方へと組織文化の転換を推進する必要があります。企業内が、オープンソースに関連した活動に好意的であることが必要です。組織のオープンソースのリーダーとして、予算の獲得、ROIの説明、アップストリームの関心獲得など、いくつもの課題に直面します。これらには、組織の上層にまで及ぶ考え方の転換と膨大な教育を要します。

1. Hire key developers and maintainers from the project’s community
2. プロジェクトのコミュニティから鍵となる開発者やメンテナーを雇い入れる

It takes considerable time to grow internal open source expertise. Hiring key developers is a critical step that allows your organization to quickly gain skills and recognition.

社内でオープンソースの専門家を育成するのにはかなりの時間を要します。企業が手早くスキルと外部の認知を獲得するのに、鍵となる開発者を雇い入れるのは不可欠なステップです。

Two or three people are a great start towards making a noticeable impact in a large project such as the Linux kernel, attracting further hires, and allowing enough resources to mentor existing junior developers. (See our guide on Recruiting Open Source Developers)

2~3人の開発者で始めれば、Linuxカーネルのような大きなプロジェクトに注目されるような影響を及ぼし、さらなる開発者を惹きつけ、また、十分な資源を与えて若手開発者への教育を行うような方向へと進んで行くのに十分です。（本ガイド集の「オープンソース開発者を募集する」をご参照ください。）

The goal is to find people who have enough peer recognition to be influential in the community. There are typically three pillars to this: domain expertise, open source methodology, and working practices.

目標は、コミュニティで影響力があるという仲間うちの評価のある開発者を見つけることです。そのような影響力には一般に3つの柱があります；当該領域の専門性、オープンソースの方法論、そして、開発への取り組みです。

You also need to align corporate interests with individual interests: it’s very hard to motivate a senior open source developer to work when their personal interests don’t meet with corporate interests in a given project. For example, a Linux memory management expert may not be interested in working on file systems, a corporate priority. Therefore, finding a match in interests is critical for a long-lasting relationship.

また、企業としての関心と個人の関心を一致させることが必要です。上級のオープンソース開発者に、本人の関心が企業の関心と合致していないプロジェクトに参加してもらっても動機づけは困難です。たとえば、Linuxのメモリー管理の専門家は、企業として優先度が高かったとしても、ファイルシステムの開発に携わることには関心ないかもしれません。長期的な関係維持のためには、関心をマッチングさせることは不可欠です。

2) Allocate Time for Upstream Contributions

2) アップストリーム貢献に時間を割く

The core principle for hiring open source developers is to support your open source development and upstream activities. There is also the expectation that they should support product teams in their expertise areas. However, it’s not uncommon for product teams to exercise their influence in an attempt to hijack the time of the open source developers by having them work on product development as much as possible. If this happens, many open source developers will head to the door, seeking a new job that allows them to work on their upstream project before you realize what just happened.

オープンソース開発者を雇用する核心的原則は、企業のオープンソース開発とアップストリーム活動に対して支援することです。専門的な領域において製品部門を支援すべきであるという期待もあるでしょう。しかし、できるだけ多く製品開発のために働いてもらうことにより、オープンソース開発者の時間を乗っ取るべく開発部門が影響力を行使するというのはかなりよくある事例です。このようなことが起きると、多くのオープンソース開発者は、雇用した側に何が起きたのか理解もできないうちに、アップストリームプロジェクトで活動することを許す新しい仕事を探してドアーの外に向かうでしょう。

Therefore, it’s important to create and maintain a separation of upstream work and product work. In other words, it’s recommended to provide your open source developers with guaranteed time to meet their upstream aspirations and responsibilities, especially if they are a maintainer. For junior developers or other internal developers who are using open source in product components, such interactions with the upstream community will increase their language, communication, and technical skills. In the absence of such an upstream time guarantee, it’s easy for these team members to be sucked into becoming an extension of product teams, resulting in their upstream focus drying up in favor of product development.

したがって、アップストリームの活動と製品開発の仕事を分割する方策を案出し、維持することが重要です。言い換えれば、オープンソース開発者には、特にメンテナーとして活躍するような開発者には、アップストリーム活動に対する願望を満たし、そこで責任を果たすためのたす時間を保証しなければなりません。若手の開発者やオープンソースを製品のコンポーネントとして利用する社内の開発者の場合でも、アップストリームコミュニティとの交流は、言語・コミュニケーション・技術のスキルを向上させます。そのようなアップストリーム活動の時間を保証しないと、オープンソース開発チームのメンバーであっても、容易に製品部門に吸収されてしまい、結果として、製品開発を優先させてアップストリームに対する関心が干上がってしまいます。

You should also be the upstream partner for product teams; they often feel like they’re living inside a pressure cooker, especially in a consumer electronics environment. They often seem understaffed, lack critical resources to support parallel upstream development, and are under constant pressure for feature delivery within tight schedules. In such an environment, it’s very easy to overlook the benefit of upstreaming in favor of short-term time savings that can unfortunately lead to technical debt that has a higher cost in the long term. Open source teams can help by being a partner that focuses on delivering important code upstream, reducing this technical debt.

オープンソース開発チームは、また、製品部門のためのアップストリームパートナーになるべきです。製品部門は、特に家電業界の環境においては、圧力釜の中で生活しているように感じるようです。いつも要員不足であり、並行したアップストリーム開発に必要な資源が欠けており、また、厳しい開発スケジュールの下で機能提供を行うという定常的な圧力にさらされています。そのような環境で、目先の時間節約に走ると、アップストリーム活動のメリットが容易に見過ごされます。そのような時間節約は、不幸にして、長い目で見ればより大きなコストとなるような技術的負債をもたらします。オープンソースチームは、重要なコードをアップストリームに提供し、技術的負債を減らすことにフォーカスしたパートナーであることによって手助けとなります。

Encourage developers outside the open source team to learn from and contribute to the open source community. We provide as much help as we can with upstream code contributions, but our resources are limited, and we don’t always have the deep understanding of products that might be needed to adequately identify where code can be upstreamed. Better involvement in the open source community from teams outside our own allows us to get more important code upstream, and improves our ability to interact with the community.

オープンソースチームの外にいる開発者が、オープンソースコミュニティから学び、そこに貢献するよう奨励してください。オープンソースチームは、アップストリームのコード貢献に関してできる限りの支援を行います。しかし、チームの資源には限りがあり、コードをどこにアップストリームできるかを十分に見極めるのに必要とされる製品の深い理解があるとは言えません。オープンソースチーム以外の部門からのオープンソースコミュニティへの好適な関与は、より多くの重要なコードのアップストリーム化を可能とし、コミュニティと交流する力の向上に寄与します。

3) Create a Mentorship Program

3) 指導プログラムを作る

Grow open source talent in specific technology areas relevant to your products. It’s easy to hire a few resources from outside the company, but there are several limitations to this approach.

企業の製品に有効な特定の領域でオープンソース人材を育成してください。企業の外から何人かを雇い入れるのは容易かもしれませんが、そのアプローチには制約もあるでしょう。

The alternative approach is to convert your existing developers into open source contributors via training on the technical domain and open source methodology. These developers can then be paired with a mentor to further expand their skills.

代わりとなるアプローチは、既存の開発者を、特定技術領域とオープンソース方法論に関する教育を経て、オープンソース貢献者への転換させることです。そして、このような開発者は指導者とペアを組むことによって、さらにスキルを広げることができます。

Set up a mentorship program where senior, experienced open source developers provide mentorship to junior, less experienced developers. Typically, the mentorship program would run for 3 to 6 months; during this time, the mentor should supervise the work of the mentee, assign tasks, and ensure proper results. The mentor would also do code reviews for anything the mentee produces, and provide feedback before the mentee pushes the code to the upstream project.

指導プログラムを作り、それにより、経験を積んだ上級のオープンソース開発者が若手の、経験の足りない開発者を教育します。通常、指導プログラムは3~6か月をかけ、指導者は被指導者の作業を監督し、仕事を割り当て、確実に適切な成果を上げていきます。指導者は被指導者の作るすべてのコードについてレビューを行い、アップストリームプロジェクトにコードを提出する前にフィードバックを提示します。

The goal is to increase the number of developers the company has contributing code to the upstream project, and to improve individual effectiveness by increasing the quality of code and the percentage of code that is accepted into the upstream project. Generally speaking, no more than 4-5 mentees should be assigned to a given mentor, and ideally they should work in the same area as the mentor to make code reviews more efficient.

目指すものは、アップストリームプロジェクトにコードを貢献する開発者の数を増やし、また、コードの品質を上げ、アップストリームプロジェクトで受け容れられるコードの率を上げることによって個人としての実力を向上させることです。一般的に言って、ひとりの指導者には多くて4~5人の被指導者が割り当てられ、また、コードレビューが効率的となるよう、指導者と同じ専門領域で仕事をするべきでしょう。

4) Formalize an Open Source Career Path

4) オープンソースのキャリアパスを正式なものとする

Create an open source developer track in your human resources (HR) system so people hired as open source developers have a good sense of how their career will progress within the company versus other non-open source developers. Additionally, you should adjust performance-based bonuses to include goals related to open source development work. The metrics by which the performance of proprietary or closed source developers are measured are often different than those of open source developers.

オープンソース開発者として雇用した人々が、非オープンソース開発者との対比で、どのように昇進して行くかについて見通しを持てるように、人事システムにオープンソース開発者用のキャリアパスを作ってください。それに加えて、成果基準ボーナスには、オープンソース開発の仕事に関連した目標を設定できるよう調整すべきです。プロプラエタリーソフトウェアや非オープンソースソフトウェアの開発者の評価尺度は、オープンソース開発者のものとは異なっていることが多いでしょう。

At some companies, there is a clear distinction between open source and non-open source developers. But in many companies, the line is much more fluid, depending on its organizational structure and open source strategy. In reality, all modern developers have to work with open source. There are no closed source developers. Rather, there are times their code stays inside the company and times their code is published (contributed to a third party, or published as a new project). Your HR track and incentives should reflect your organization’s unique structure and approach to open source.

いくつかの企業では、オープンソース開発者と非オープンソース開発者の間には明確な区別があります。しかし、多くの企業では、組織としての構造やオープンソース戦略に依存して、境界は流動的です。実際には、最近の開発者はすべてオープンソースと関わって作業せざるを得ません。クローズドソースのみを扱う開発者はいないのです。どちらかといえば、コードは企業内に留まる時もあれば、公開される時 (外部の企業・組織に提供、あるいは、新規プロジェクトとして公開)もあるのです。企業の人事システムのキャリアパスやインセンティブは、オープンソースに対応した企業独特の組織構造やアプローチを反映させるべきです。

Finally, allow a work from home policy for open source developers, regardless of the general corporate policy related to this. Lately, we have witnessed a reverse in work from home policies across companies where many have either banned or created strict limitations to working from home. In the open source world, a work from home policy is almost mandatory because open source experts are located all over the planet, and this policy is often the only way to hire them.

最後に、企業としてのポリシーがどうであれ、オープンソース開発者には、在宅勤務を許可してください。近年、在宅勤務に関しては逆の傾向があり、多くの企業が禁止したり、厳しい制限をつけたりしています。オープンソースの世界では、在宅勤務のポリシーは、ほとんど必須の状況です。というのは、オープンソースの専門家は世界中のあらゆる場所を拠点としており、このポリシーを適用するのが雇用の条件となるからです。

There are operational benefits to a flexible work policy as well. Offering remote work options reduced employee turnover, and “job attrition rates fell by over 50 percent,” according to a study published by Stanford University. And a study by PGI, a leading provider of software services, found that 80 percent of workers reported higher morale when working from home, while 69 percent reported lower absenteeism.

フレキシブルな勤務ポリシーには、企業運用上の利点もあります。[スタンフォード大学が刊行した調査報告](https://people.stanford.edu/nbloom/sites/default/files/wfh.pdf)によると、リモートワークを選択できる従業員は転職率が下がり、また、「要員の減少率は50%も下がった」とのことです。また、ソフトウェアサービスのトッププロバイダーである[PGIの調査](https://remote.co/10-stats-about-remote-work/)によると、在宅勤務により作業員のモラルは80%向上し、欠勤率は68%下がったとあります。

5) Offer Training

5) トレーニングを提供する

It’s impossible for any company to hire all the senior and most expert developers in a given domain. This concept applies to the Linux Kernel and any other prominent open source project. Therefore, you must have a way for your company to increase the competence of its developers in a given technical domain. In addition to technical training, you’ll also need training to teach the open source development model and the basic concepts of open source legal compliance.

どんな企業であろうと、当該領域で専門性の高い上級開発者をすべて雇用するのは不可能です。この見方はLinuxカーネルにも、また、その他のいかなるオープンソースプロジェクトにもあてはまります。したがって、当該技術分野において企業の開発者の能力を伸ばす方法を持っていなければなりません。技術的なトレーニングに加えて、オープンソース開発モデルやオープンソースの法務コンプライアンスの基礎概念を教育するトレーニングも必要です。

Sample training courses include:

トレーニングコースの事例：

Technical training that covers the various areas in the Linux Kernel. Maintainers or senior developers usually present this to grow internal Kernel expertise; this expertise is vital to pass on given how challenging it is to hire expert Kernel developers.

Linuxカーネルのさまざまな領域をカバーする技術トレーニング。メンテナー、あるいは、上級開発者が社内でカーネル専門知識を育成するために説明。このような専門知識は、専門的なカーネル開発者の雇用が難しい状況下で、企業が前に進むために必須。

Open source development methodology course that teaches staff that’s new to open source how open source and Linux Kernel development works and how to best get engaged.

オープンソース開発の方法論のコース。オープンソースを初めて経験するスタッフに、オープンソースやLinuxカーネルの開発がどのように行われているか、および、どうすればうまく関わることができるかを教育。

Open source compliance course that teaches staff the basics of compliance principles and open source licensing. This should also be used to inform them of your company’s policy and process. [The Linux Foundation offers a free online open source compliance training for developers. The course is available here.]

オープンソースコンプライアンスのコース。スタッフに対してコンプライアンスとオープンソースのライセンスの基礎を教育。同時に、企業のポリシーとプロセスを伝達するためにも使われるべきです。Linux Foundationは開発者向けに自由に使用できるオンラインの[オープンソースコンプライアンスのトレーニングコースを公開](https://training.linuxfoundation.org/linux-courses/open-source-compliance-courses/compliance-basics-for-developers)しています。

6) Participate in Open Source Events

6) オープンソースイベントに参加する

Support your developers to attend and participate in open source conferences and events, including local community meetups, hackathons, and summits. Such participation helps them connect at a personal level with their peers, build relationships, have face to face social interactions, and participate in technical discussions that guide the project direction.

開発者が、ローカルなコミュニティの会合、ハッカソン、あるいは、サミットなども含めて、オープンソースのコンファレンスやイベントに出席し、参加するのを後押ししてください。そのような参加を通して、仲間との個人レベルの繋がりが生まれ、付き合い関係を築くことができ、顔を合わせた交流を行い、さらに、プロジェクトの方向を決める技術討議への参加ができるようになります。

If your developers have work that others might be interested in, help these developers prepare content to present. Finally, you can also sponsor events, both big and small, to increase external visibility within the project’s community. These events are also a great venue to look for talent!

開発者が、他の人々から関心を持たれるような成果を上げたなら、その内容を発表する準備に手を貸してください。最後に、プロジェクトコミュニティ内での企業の知名度を上げるために、大小のイベントのスポンサーとなることができます。これらのイベントは、人材を見つけるための機会ともなります。

7) Provide a Flexible IT Infrastructure

7) 柔軟なITインフラを用意する

Provide a flexible IT infrastructure that allows open source developers to communicate and work with the open source and Linux Kernel community without any challenges. Additionally, set up internal IT infrastructure that matches the tools used externally to help bridge the gap between internal teams and the Kernel community or any other open source project community for that purpose.

オープンソース開発者が、オープンソースやLinuxカーネルのコミュニティと問題なくコミュニケーションし、一緒に活動することができる柔軟なITインフラを用意してください。加えて、この目的に沿って社内のチームとカーネルコミュニティ、あるいは、他のオープンソースプロジェクトコミュニティとの間のギャップを橋渡しすることができるよう、外部で使用されるツールに適した社内ITインフラをセットアップしてください。

Much of your infrastructure will naturally evolve along with your organization’s open source culture, but it’s important to be aware of the necessity and plan for its implementation.

インフラのかなりの部分は企業のオープンソース文化と一緒に自然に進化していきます。けれども、オープンソースソフトウェアを実装したインフラを整備する必要性に気付いて、その計画を作ることが重要です。

There are three primary domains of IT services that are used in open source development: knowledge sharing (wikis, collaborative editing platforms, and public websites), communication and problem solving (mailing lists, forums, and real-time chat), and code development and distribution (code repositories and bug tracking). Some or all of these tools will need to be made available internally to properly support open source development. There is a chance this might conflict with existing company-wide IT policies. If so, it’s vital to resolve these conflicts and allow open source developers to use the tools they are familiar with.

オープンソース開発で利用されるITサービスには、主に3つの分野があります。すなわち、情報の共用(Wiki、共同編集プラットフォーム、パブリックWeb)、コミュニケーションと問題解決(メーリングリスト、フォーラム、リアルタイムチャット)、および、コード開発と頒布(コードリポジトリー、バグトラッキング)です。オープンソース開発を支援するのに、これらのツールのすべて、ないしは、多くが社内で利用できる必要がありますが、企業の既存のITポリシーとは矛盾が出て来る可能性もあります。そのようなことが起きると、矛盾を解消し、オープンソース開発者が馴れ親しんだツールを使えるようにすることが不可欠です。

These open source practices typically require an IT infrastructure that is free from many standard, limiting IT policies.

これらのオープンソース　プラクティスは、ITポリシーを制約するような多数の標準に束縛されないことを要求します。

“It took us years of constant discussion and negotiation to break from the traditional IT setup into a more flexible environment that supports our open source development. We made it work for us and with enough persistence you also can make it work for your open source team.”

「従来型のIT装備を打破し、私たちのオープンソース開発をサポートできる、より柔軟性のある環境へと移行するのに、何年にもわたる切れ間のない議論と交渉が必要でした。それが私たちのために働くようになりました。粘り強くやれば、あなたのオープンソースチームのために働くようにもできます。」

Ibrahim Haddad – Vice President of R&D and Head of the Open Source Group at Samsung Research America.

[Ibrahim Haddad](https://twitter.com/ibrahimatlinux) – R&D部門VP兼オープンソースグループ長、Samsung Research America

8) Track Developer Code Contributions

8) 開発者のコード貢献を追跡して調べる

Create an internal system to keep track of developer contributions and impact. Contributions can include upstream development, supporting product teams, knowledge transfer (mentoring, training), visibility (publications, talks), launching new open source projects, and establishing internal collaboration projects with other teams or groups.

開発者の貢献と効果を追跡して調べるために社内の仕組みを作ってください。貢献には、アップストリーム開発、製品部門への支援、知識の伝達(指導やトレーニング)、知名度向上(執筆物や講演)、新規プロジェクトの立ち上げ、さらには、社内の他の部門やグループとの協業プロジェクトの構築などがあります。

There are several toolkits that help track source code contributions; for instance, The Linux Foundation uses a tool called gitdm, which produces the data reported in the Linux Foundation yearly Linux Kernel report. This can be used to track both individual developers as well as the overall team performance. Individual developers can be tracked for the number of patches they submit, the patch acceptance rate (patches submitted divided by patches accepted), and the type of patch (e.g. if it is a new feature, enhancement of existing functionality, bug fix, documentation, etc.).

ソースコード貢献を追跡して調べるのに便利なツールキットがいくつかあります。たとえば、Linux Foundationはgitdmと呼ばれるツールを使い、Linux Foundationが毎年発行する「Linuxカーネルレポート」で紹介するデータを生成しています。本ツールは、開発者個人、および、チーム全体の両方の成果を追跡して調べるのに使えます。開発者個人としては、提出したパッチの数、パッチの受け容れ率(受け容れられたパッチ数 / 提出パッチ数)、さらに、パッチのタイプ(すなわち、新規機能、既存機能の拡張、バグフィックス、ドキュメントなど)を追跡して調べることができます。

Other tools like GrimoireLab can also be used to chart and visualize the metrics you want to track. See the next section on metrics for specific examples of what you should track.

GrimoireLabのようなツールでは、追跡して調べたい評価尺度をチャートにしたり、可視化するのに使えます。次のセクションでは、追跡して調べるべき具体的データの評価尺度について触れています。

9) Identify Focus Areas with Broad Impact

9) 大きな効果のあるフォーカス域を特定する

Contribute to and focus on areas that benefit more than one business unit or more than one product. This allows you to provide value and show ROI across multiple business units and increases your chances for more funding and support.

ひとつのビジネス部門とか、ひとつの製品を越えて利益が期待できる領域に貢献し、フォーカスを合わせてください。これによって、複数のビジネス部門にわたって価値を提供し、ROIを示すことができます。また、より多くの予算や支援を得るチャンスが増えます。

Focus your contributions on upstream projects that would directly benefit the company’s strategy and products. In open source development, it’s very easy to get carried away hopping between different interesting projects. In an enterprise setting where the open source group is considered a cost center, your driving force should be to focus on open source projects that support product development.

企業の戦略や製品に直接的に利益をもたらすアップストリームプロジェクトに貢献をフォーカスしてください。オープンソース開発においては、興味深いプロジェクトの間を飛び跳ねながら、いろいろなプロジェクトに夢中になることがあります。オープンソースグループがコストセンターとして認識される大企業の現場においては、製品開発を後押しするオープンソースプロジェクトに同グループの力をフォーカスすべきです。

Samsung does a yearly review of its product portfolio in an effort be involved in open source projects that are commonly used across as many products as possible. This list is then prioritized based on several factors, and they focus their efforts on the top projects. A methodology that drives your priorities is a great way to stick to what’s important, justifiable, and fundable.

Samsungにおいては、できるだけたくさんの製品で共通的に活用されたオープンソースプロジェクトの活動を分析し、その製品ポートフォリオを毎年レビューしています。この一覧表は、いくつかの要因をベースに優先度付けが行われ、優先度の高いプロジェクトに活動をフォーカスします。このように優先度の高い活動を推進する方法は、重要なもの、正当性のあるもの、そして、資金投入できるものからの逸脱を避ける良い方法です。

10) Foster Internal Collaboration

10) 企業内の協業を育てる

Collaborate with other business units that use the same open source projects in their products. These collaborations can take one or more of many forms:

製品の中で同一のオープンソースプロジェクトを活用するいろいろなビジネス部門と協業してください。このような協業には次のような形態がありますが、その中のひとつ、ないしは、複数の形態がとられる可能性があります。

Deliver training to their developers

Run a workshop on a specific topic or problem

Develop new functionality

Troubleshoot and resolve issues and bugs

Upstream existing code for which they have no resources to do

Help get them off an old fork and onto a mainline version

And more

　　開発者にトレーニングを提供

　　特定の話題、あるいは、問題についてワークショップを開催

　　新しい機能を開発

　　問題やバグの分析と解決を実施

　　資源のないグループに代わって、出来上がったコードをアップストリームに提出

　　以前にフォークしたツリーから、メインラインの版に移行するのを支援

　　その他

The goal of these collaborations is to help the product teams understand their needs and fulfill their product goals via open source enablement.

このような協業の目差すものは、オープンソースによって可能となるものを通じて、製品部門がニーズを理解し、製品目標を達成するのを後押しすることです。

Your company must share information and priorities across different divisions. To illustrate this, assume you are in an open source team and you are requested to support the implementation of a driver, but you are unable to get access to the hardware manual and instructions. This sounds a bit like playing darts with the lights off, and it is. Information sharing is a critical component to successful internal collaborations between the open source teams and everyone else.

このような協業の実施のためには、企業はさまざまな部門に渡る情報と優先度の共有がなされていなければなりません。例として、あなたがオープンソースチームに所属していて、ドライバーの開発を支援するように要請されたが、ハードウェアのマニュアルや命令セットへのアクセスが不可能だったとしましょう。これは暗闇の中でダーツをやるようなものです。オープンソースチームと他のあらゆる人々との間の社内協業の成功のためには、情報共有が必須な要素です。

Open Source at Samsung

Samsungのオープンソース

Samsung’s Open Source Group (OSG) was established in February 2013 to support two primary functions: the first is to provide open source leadership within Samsung by helping other divisions in the company understand how to participate in and benefit from open source development. The second is to serve as Samsung’s representatives in the wider open source community. The mandate of the team is to focus on enhancing key open source projects and technologies via active contributions to them, and to be actively involved and engaged with various open source organizations and foundations.

Samsungのオープンソースグループ(OSG)は、主に2つの機能を目指して2013年2月に設立されました。一番目は、他部門がオープンソース開発に参加する方法やそこから利益を受ける方法を理解するのを助けることにより、Samsung社内におけるオープンソースのリーダーシップを提供することです。二番目は、より広範なオープンソースコミュニティでSamsungの代表役を務めることです。本チームの任務は、活発な貢献を通じて鍵となるオープンソースプロジェクトやテクノロジーの強化にフォーカスすること、および、さまざまなオープンソース組織やファウンデーションに活発に参加することです。

Section 5

Metrics for tracking progress

セクション　5

強化の進捗状況を追跡して調べる評価尺度

Once you start implementing these open source best practices, you’ll need proper open source metrics to drive the desired development behavior. But the traditional metrics often used in product organizations don’t apply in the context of open source development.

これらのオープンソースベストプラクティスを実施し始めると、望ましい開発行動をさらに推進するために、適切なオープンソース評価尺度が必要になります。しかし、製品開発組織でよく使用されてきた従来型の評価尺度は、オープンソース開発の環境では応用できません。

For example, tracking the number of changesets or lines of code can be a good metric for open source development impact. But you may have multiple instances of desired functionality being implemented upstream because your open source developers lobby for support from the community. In this case, the number of changesets or lines of code doesn’t matter nearly as much as the technical leadership that team members provide to get code upstream and reduce the company’s downstream maintenance efforts. So the metrics you track should account for both activities.

たとえば、修正件数とコード行数を追跡して調べるのは、オープンソース開発の効果のよい評価尺度になりえます。しかし、アップストリームにおいて企業が必要とする機能を実装するために、複数の活動を行っていることがあります。企業の開発者がコミュニティの支持を取り付けるためのロビー活動を行うなどするからです。このケースでは、オープンソースチームのメンバーがコードをアップストリームに提供し、企業のダウンストリームにおける保守費用を削減するために果たした技術的リーダーシップと比較すると、修正件数やコード行数は到底重要とは言えません。追跡して調べる評価尺度は両方の活動を評価するものであるべきです。

Commits and lines of code over time

一定期間内のコミット数とコード行数

One of the most basic things to track is the number of commits and lines of code changed over a specific period of time, such as every week, month, or year.

追跡して調べるべきもっとも基本的な評価尺度は、特定期間内、たとえば、週ごと、月ごと、あるいは、1年ごとに提出されたコミット数とコード行数です。

bar graph of open source metrics

オープンソース評価尺度の棒グラフ

The total commits and lines changed per project per week is a good place to start tracking metrics.

プロジェクトごとに集計された週ごとの総コミット数と修正コード行数は追跡して調べる評価尺度として価値があります。

With this data, you can compare contributions from various internal development teams to identify where source code contributions are coming from and help ensure that resources are allocated appropriately.

このデータに基づき、どこからコード貢献が出てきているかを判断して、さまざまな社内開発チームの貢献を比較し、社内資源を適切に割り当てることが可能となります。

From here you can create charts that compare various internal teams for their cumulative contributions, percent of total contributions, and the amount of time it takes to get code committed upstream (see the following charts).

これを用いて、さまざまな社内チームの累積貢献量、貢献全体に対する割合、アップストリームにコードをコミットするのに要した総時間数を比較するチャートを作ることができます。次のチャートをご参照ください。

chart of cumulative contributions over time

一定期間内の累積貢献のチャート

Cumulative contributions over time can be tracked to compare internal teams and identify teams that are increasing their involvement in a particular open source community (in this chart, it’s the Linux kernel).

一定期間内の累積貢献量を追跡して調べることにより、社内チームを比較し、特定のオープンソースコミュニティへの関わりが増加しつつあるチームを特定することができます。このチャートはLinuxカーネルの状況です。

chart of percent contribution

貢献量の割合を示すチャート

Displaying your company’s contributions as a percent of total over time allows you to identify the teams that contribute the most code.

一定期間内の企業の貢献量を全体に対する割合として示すことによって、もっともたくさんのコードを貢献したチームを特定することができます。

chart to commit code upstream

アップストリームにおいてコードをコミットした状況を示すチャート

The amount of time it takes to commit code upstream can be valuable for tracking your development efficiency. This table and chart shows how quickly various teams are getting their code contributed upstream and compares it to the community as a whole.

アップストリームにコードをコミットするのに必要とした総時間数は、開発の効率を追跡して調べるのに有効です。この表とチャートは、さまざまなチームがどれほど速くコードをアップストリームに貢献したかを示し、それをコミュニティ全体と比較します。

You can also use these metrics to compare your performance to other companies who are involved in the kernel ecosystem for instance (see the image, below). This competitive analysis helps you be better informed about the overall developer ecosystem for the project.

これらの評価尺度は、企業の実績を、たとえばカーネルコミュニティに関わっている他社と比較するのに使うことができます（下図をご参照ください）。このような競合分析は、当該プロジェクトに取り組む開発コミュニティ全体について、よりよい情報を得るのを助けます。

chart of cumulative open source contributions

累積オープンソース貢献量のチャート

Cumulative contributions can be sorted by company to see how your company stacks up against others.

累積貢献量は、企業ごとにソートして、あなたの会社が他社と比較して、継続的に蓄積してきた貢献度を示してくれます。

These metrics provide a much better idea of where your strengths and weaknesses are and can help inform your overall development strategy. Tracking your own contributions relative to competitors’, for example, provides valuable information that helps an organization position its products relative to competitors’ in the marketplace.

これらの評価尺度は、企業の強味と弱みがどこにあるのかについて優れた考察を提供し、また、企業の開発戦略全体について情報を得る助けとなります。たとえば、競合企業との貢献量を追跡して調べることにより、競合他社の製品と当該企業の製品の市場における位置づけ確認するのを助ける価値ある情報を提供します。

charts from the kernel development report

「Linuxカーネル開発レポート」に掲載されたチャート

Projects may publish contribution data independently as well. For example, you can also track contributors to the Linux kernel via The Linux Foundation’s Linux Kernel Development Report and LWN.

各プロジェクトは独立に貢献データを公開することが許されています。たとえば、Linuxカーナルへの貢献者をLinux Foundationの「Linuxカーネル開発レポート」とLWNで追跡して調べることができます。

The Linux kernel is of strategic importance to Samsung so the company chose to focus its development efforts there, among many other projects. They are now regularly among the top 5 contributors to the Linux Kernel by changesets. And the company is achieving the same progress with several other open source projects it deemed critical to the development of its products.

LinuxカーネルはSamsungにとって戦略的重要性があり、多くのプロジェクトを手掛ける中で、開発活動を同プロジェクトにフォーカスすることを選択しています。Samsungは、現在Linuxカーネルの修正件数に関して、定常的にトップ5貢献企業に入っています。また、同社の製品開発に不可欠だと考えられる他のオープンソースプロジェクトについても、同様にレベルが上がっています。

Being a top contributor isn’t a goal, in and of itself, but rather an indication that the organization’s development efforts are being accepted by the communities in which it participates. As the saying goes, you can either sit at the table as an open source influencer or be on the menu.

トップ貢献企業になること自体が目標なのではありません。しかし、企業の開発活動が参加しているコミュニティに受け容れられていることの指標であることは間違いありません。俗に言うように、オープンソースの影響力ある企業として食卓に座るか、さもなければ、メニューに載るかです。

“Being on the menu means sitting and crying in your office about how you need to maintain a ton of code out-of-tree because you can’t merge with upstream. We’d rather sit at the table.”

「『メニューに載る』とは、アップストリームにマージできないために、ツリーから外れた大量のコードをどのように保守するかを、オフィスに座って大声で議論することを意味しています。それよりは、食卓に座る（しっかりと発言する）方がいいでしょう。」

〔訳者注〕『メニューに載る』の意味は、意思決定の場に参加できなければ、他の人の意思であなたの運命が決まるという比喩です。

Ibrahim Haddad – Vice President of R&D and Head of the Open Source Group at Samsung Research America.

[Ibrahim Haddad](https://twitter.com/ibrahimatlinux) – R&D部門VP兼オープンソースグループ長、Samsung Research America

Section 6

Final words

セクション　6

結論

Effective open source development can’t be given, it must be earned. This leadership is earned through regular, ongoing participation and contribution.

効果的なオープンソース開発は与えられるものではありません。働いてかち取らなければなりません。このリーダーシップは、定常的で継続的な参加と貢献を通じてかち取られます。

By following some of the open source best practices set out by the pioneering companies in open source, you can make fast progress toward developing the internal open source expertise you require. You can then leverage that expertise to improve your products and services while reducing code maintenance costs.

オープンソースにおけるパイオニア企業が提示したオープンソースのベストプラクティスのいくつかを踏襲することによって、企業が必要とするオープンソースのノウハウ獲得に向けて敏速な進歩を遂げることができます。そうすれば、そのノウハウをテコにして、コード保守の費用を低減しつつ、製品やサービスを強化できます。

Happy hacking!

楽しくハッキングしましょう。

These resources were created in partnership with the TODO (Talk Openly, Develop Openly) Group – the professional open source networking group at The Linux Foundation. A special thanks goes out to the open source program managers who contributed their time and knowledge to making these comprehensive guides. Participating companies include Autodesk, Comcast, Dropbox, Facebook, Google, Intel, Microsoft, Netflix, Oath (Yahoo + AOL), Red Hat, Salesforce, Samsung and VMware. To learn more, visit: todogroup.org

これらのリソースは、TODO (Talk Openly, Develop Openly) Groupとの協力によって作成されました。TODOグループは、Linux Foundation傘下のプロフェッショナル オープンソースプログラム ネットワーキンググループです。本ガイド集のために時間と識見を貢献してくれたオープンソース プログラムマネージャーの皆様に特に謝意を表します。本ガイドの作成に参加してくれた企業は、Autodesk、Comcast、Dropbox、Facebook、Google、Intel、Microsoft、Netflix、Oath (Yahoo + AOL)、Red Hat、Salesforce、Samsung、VMwareの各社です。詳細は、[todogroup.org](http://todogroup.org/)をご参照ください。

This work is licensed under a Creative Commons Attribution ShareAlike 4.0 International License.

TODO Group

*この資料は、Creative Commons Attribution ShareAlike 4.0 International License (CC BY-SA 4.0 クリエイティブコモンズ標示⊸継承国際ライセンス) でライセンスされています。*