あなたのスキルは社会に役立つ
エンジニアだからできる社会貢献

東日本大震災の発生直後に発足したHack For Japanや「市民が主体となって自分たちの街の課題を技術で解決するコミュニティ作り支援」を掲げるCode for Japanのメンバーを始めとして、日本各地で技術を活用した社会貢献活動が行われています。本連載では、防災や減災、地域の活性化や課題解決、そして人材育成など、「エンジニアだからできる社会貢献」の取り組みをお届けします。

第117回

接触確認アプリ「COCOA」OSSコミュニティの現状と課題

●COCOAコラボレーター　有山 圭二（ありやま けいじ）Twitter @keiji_ariyama

本稿の中で示される見解や方針は筆者である有山圭二個人のものであり、厚生労働省および内閣官房 情報通信技術（IT）総合戦略室を代表するものではありません。あらかじめご了承ください。

はじめに

```
while (Japan.recovering)
    we.hack();
}
```

接触確認アプリ「COCOA」が登場して1年が経ちました。

筆者は2021年3月からGitHubのCOCOAリポジトリ[注1]のコラボレーターをしています。筆者の役割は、GitHubのコントリビューターのみなさんとCOCOA開発チームの間の橋渡しをすることです。IssueやPull Requestを整理したり、活動しやすいように情報を公開したりするために、厚生労働省ならびに内閣官房 IT総合戦略室とともに調整しています。

本連載は「エンジニアだからできる社会貢献」をテーマにしています。実を言えば、寄稿のお話をいただいたとき、はたして自分は社会貢献ができているのだろうかと考え込んでしまいました。筆者がコラボレーターになったことでCOCOAが突然良くなった、などという都合の良いことは起きていません。COCOAの開発自体は開発チームが担当しています。開発チームは優秀で、COCOAの開発に用いられているXamarinはもちろんのこと、接触確認APIの取り扱いにおいても経験を積んだメンバーがそろっています。情報を公開するための調整にしても、内閣官房 IT総合戦略室や厚生労働省の方々がいなければ、筆者1人の力ではどうにもならないことばかりです。コラボレーター就任から4ヵ月、筆者自身はまだ何も成し遂げていない、というのが正直な感想です。

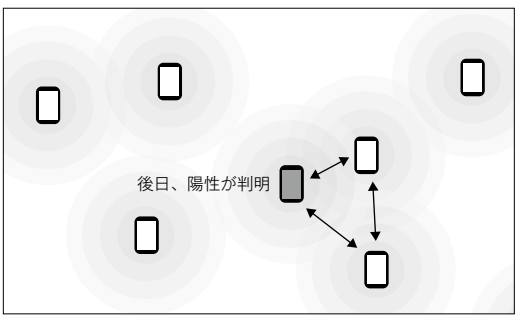
とはいえ、ここまで「前に進んできた」確信はあります。また、COCOAのOSSコミュニティが今どうなっているかを広く伝えることの必要性も強く感じています。これが今回、本稿の執筆を引き受けることになった理由です。

COCOAについて

```
while (Japan.recovering)
    we.hack();
}
```

まずはじめに接触確認アプリ「COCOA」について解説します。COCOAは、スマートフォンのBluetooth機能を利用して利用者同士の接触を検知・記録（**図1**）するアプリです。ある利用者が新型コロ

◆**図1　Bluetoothにより接触を記録**



注1　https://github.com/cocoa-mhlw/cocoa

ナウイルス感染症（COVID-19）の陽性判定を受けたときに、感染力があるとされる期間に接触をしたほかの利用者に匿名で通知できます（**図2**）。COCOAは、厚生労働省からリリースされており、本稿執筆時点で約2,800万回、ダウンロードされています。

COCOAは、オープンソースプロジェクト「COVID-19Radar」をベースに開発しています。また、COVID-19RadarのライセンスMozilla Public License 2.0に従って、GitHubでソースコードを公開しています。

課題

COCOAリポジトリを運営するうえでの大きな課題が、OSSコミュニティとCOCOAの開発チーム、両者間のスループットをどのようにして高めるかです。COCOAの開発チームは、GitHubとは別に非公開のインターナル・リポジトリでコードや課題を管理しています。だいたい2週間の単位で作業内容を決めている開発チームからすれば、個々のコントリビューターから散発的にIssueやPull Requestが提出されるGitHubへの対応は、都度割り込み作業となり、分量によっては計画した作業を圧迫します。

一方、コントリビューターの立場では、開発チームがインターナル・リポジトリで開発と課題管理をしているため、最新の開発状況がわからないことや、コードが公開されるのは新しいバージョンがリリースされたタイミングで、かつ、すべての変更が1つのコミットにまとめられているので、変更内容を追うのが難しいことなどの課題がありました。

インターナル・リポジトリについては、存在自体が秘密というわけではありませんでした。COCOAのベースになったプロジェクト「COVID-19Radar」がGitHubとAzure DevOpsの2つのリポジトリを同期しながら開発を進めていて、その後、開発チームが自分たちの使い慣れているDevOps環境をメインに選んだのが経緯のようです。また、COVID-19Radarが採用するライセンス（MPL 2.0）の関係で、リリースバージョンについてはソースコードが入手可能になっている必要があることから、GitHubリポジトリをコード公開の場所としていました。

これまでのおもな取り組み

これまで、課題解決のために取り組んできたことを紹介します。個別の項目に分けてはいますが、実際には複雑に絡み合っています。また筆者だけで取り組んでいるわけでもありません。銀の弾丸はどこにもなく、関係者全員でYak Shaving[注2]しながら取り組みを続けているというのが現状です。

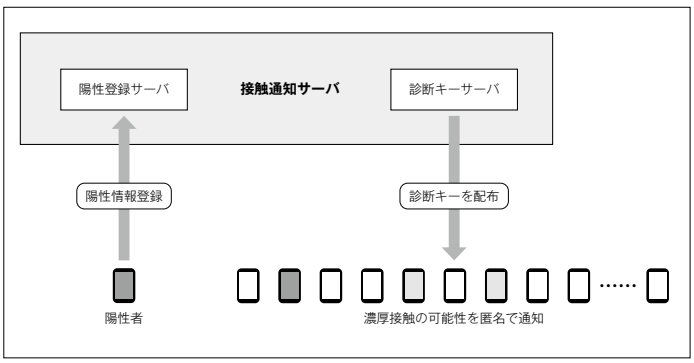
Issueの取り扱い

コントリビューターがGitHubに投稿したIssueは、まず内部の課題管理システムへ転記されます。当初は、転記済みであることは内部の課題管理システムを確認しないとわからなかったため、Issueの件数が増えるにつれて徐々に取りこぼしが発生しました。

そこで、内部の課題管理システム上のIDをIssueのDescriptionに記載することで、Issueが転記済みか、内部の課題管理システム上のどのアイテムにひも付いているかを確認できるようにしました（現在では補助的にラベルも付けて管理しています）。開

注2　ある問題を解決しようとすると、別の問題が出てくるような状態が延々と続くことを指します。

◆図2　接触通知サーバのしくみ



発チーム向けの情報をGitHubに掲載することについては不安の声がありましたが、筆者の知る限り、オープンソースプロジェクトで非公開のリポジトリや、課題管理システムが別に存在するケースはとくに珍しくはありません。何より、Issueの取りこぼしを防ぐことが重要という点で全員の意見が一致しました。

また、コントリビューターからの提案で「プロジェクトボード」も整備しています（**図3**）。作業の進捗を見渡せるいわゆるカンバンとは違いますが、過去のバージョンでどのようなIssueが解決されたか確認したり、次のバージョンでの作業候補を整理したりするのが目的です。Issueを通じて積極的に情報収集する取り組みも始めています。新しい画面デザインを公開して意見を募集したり、アプリのアクセシビリティについて聞いたりしています。アプリのレビューで寄せられた不具合報告について、筆者の手元で再現できなかったので情報提供を求めるIssueを立てたところ、数時間の内に不具合が発生する機種と原因、解決策の特定にまで至ったこともあります。

Pull Requestの取り込み

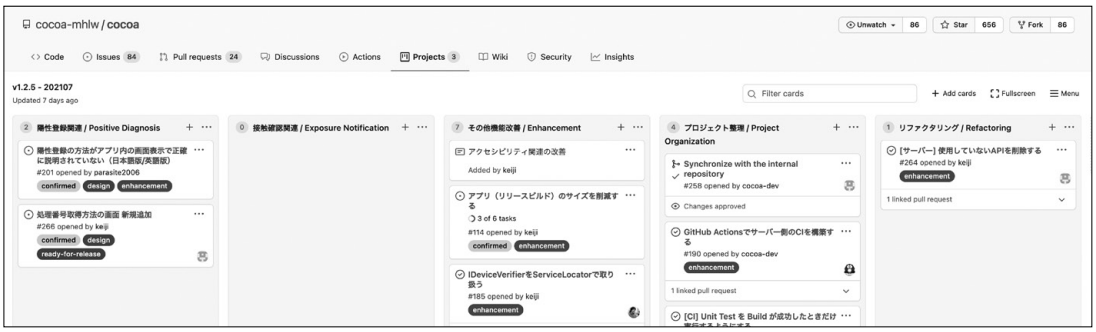
オープンソースコミュニティからのPull Requestを取り込むうえで行ったことは、GitHubからインターナル・リポジトリにコードを取り込むしくみ作りです。筆者がコラボレーターになってから数回、GitHub主導でPull Requestを取り込んだところ、開発チーム側の負担が予想以上に大きいことがわかりました。開発チームがGitHubとは別にあるインターナル・リポジトリで作業をしていることはすでに述べました。これまでGitHub側には、インターナル・リポジトリ側の変更をリリースごとに1つのコミットとして追加してきました。同じブランチであってもGitHubとインターナル・リポジトリでまったく異なる歴史をたどっています。

また、GitHubリポジトリがリリースコードの公開場所である以上、GitHubリポジトリのmasterにリリースされない変更を取り込んでしまうと、リリースに含まれない変更を取り除いたブランチを作成する必要があります。とはいえ、大きな変更をオープンソースコミュニティで手がけるときに、1つのPull Requestでリリースコードの準備を整えたうえで、その機能をリリースすることが決定するまでPull Requestが取り込まれないというのも、動きが非常に遅くなります。

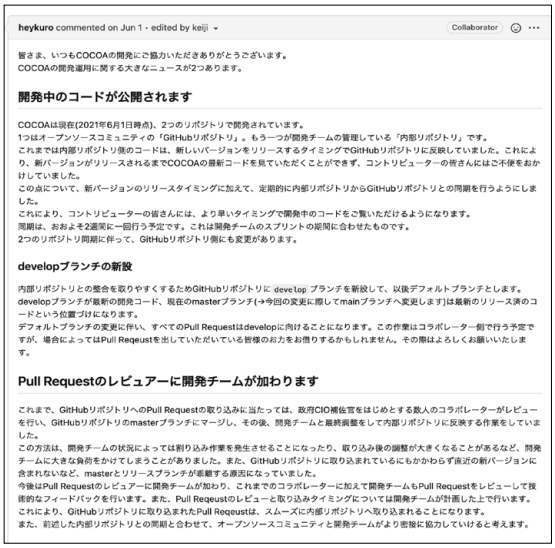
そこでまず、インターナル・リポジトリに合わせてGitHub側のブランチ構成を変更することにしました（**図4**）。それまでmasterだけだったブランチを、リリースコードを公開するmainと、開発用のコードを公開するdevelopに分けました。コントリビューターはdevelopに向けてPull Request（**図5**）を送ることになります。次に、開発チームが変更したコードについて、これまでのリリースタイミングに加えて、だいたい2週間に1回の頻度でdevelopブランチへ反映するようにしました。

大きな変更についてはfeatureブランチを用意して、そこで作業をすることにしました。コントリビューターには、まずfeatureブランチにPull Requestを送ってもらいます。そして、ある程度ま

◆**図3　COCOAリポジトリのプロジェクトボード**



◆図4　ブランチ構成変更のIssue



とまった段階でdevelopブランチにPull Requestをするという流れを想定しています。さらに、これまでGitHub側では手つかずだったCI環境もGitHub Actionsを用いて整備しました。これはPull Requestを最初にチェックする筆者の負担を減らすためですが、1ヵ月ほどかけて試行錯誤しながらビルドとテストを整備したところ、通りすがりのコントリビューターが一晩で高速化してくれました(**図6**)。これは実にオープンソースコミュニティ的な出来事だったと思います。

今後の取り組み

ここまで、COCOAリポジトリの取り組みを紹介しました。まとめてみるとあっさりしたものだなと思いますが、これらを実現するために調整を続けた日々を思い出すと、やはり感慨深いものがあります。

今後は、引き続きIssueの活用やPull Requestの取り込み促進、そしてコントリビューターを増やす取り組みを続けていきます。たとえば今後のリリース予定や機能のロードマップなど、提供する情報を増やしていきたいと個人的には考えています。プロジェクトとして、どの機能を重視しているかが明確に伝わることで、コントリビューターのみなさんの力をCOCOAに効率的に反映できます。

◆図5　Pull Requestのコードレビュー

開発チームから公開するコードについては、頻度こそ増えたものの、変更が1つにまとめられている点で改善の余地があると筆者は考えています。今後はもっとわかりやすく、細かい粒度で公開していくように働きかけをしていきます。また、IssueやPull Requestなどを通じてコントリビュートしてくださっているみなさんの名前を記録することも必要だと考えています。コミットログだけでなくドキュメントとして記録して、可能であればCOCOAのアプリやWebサイトから見られるような形になるように提案したいと考えています。

以上、接触確認アプリ「COCOA」のオープンソースコミュニティの現在についてをお届けしました。読者のみなさんも、これを機会にCOCOAのリポジトリを覗(のぞ)いてみてください。それではまたの機会に、ご報告できる日を楽しみにしています。SD

◆図6　GitHub Actionsの高速化前後