

6 プロダクト開発や技術的課題の解決を抽象化して社内外のプログラミングの活動を重ね合わせよう



松本亮介 | さくらインターネット（株）さくらインターネット研究所

これから社会人になる学生も、大学や大学院での研究の中で、さまざまなプログラミングを経験されてきたかと思う。個人的には、研究におけるプログラミングであるからといって、社会人、いわゆるエンジニアになってから取り組むプログラミングとは訳が違う、とは思っていない。しかし、時間の限られている大学生活の中では、社会人のエンジニアが取り組んでいるプログラミングの方法について知ることができないかもしれない。そこで本稿では、これから社会人になってプログラミングを新たに始めようとするエンジニアや、現時点で大学に所属しながらもプログラミングのスキル向上に興味のある学生や研究者に向けて、インターネット技術やWebサービスにかかわるエンジニアがどのようにプログラミングに取り組んでいるかについて紹介したい。

社内と社外のプログラミングを重ねる

会社に就職してエンジニアになると、社内のプロダクトに対してのみプログラミングを行っていくように思われるかもしれない。また、社外でのプログラミングは、社内の活動とは別で、趣味の一環として取り組まないといけないように感じているかもしれない。しかし、プログラミングのスキルを上達させるためには、いかに社内と社外でのプログラミングの活動を重ね合わせるかが重要である。さらに、そのプログラミング活動に対する楽しみや競争心、快楽といった純粋な感情を維持させていくことも大

事である。

そもそも、社内外のプログラミングが別のものとして分離されていると、社内での短期的な課題解決のために、会社依存の課題やプログラミングを行うことが増えてしまう。その場合、スキルとしてのプログラミングを考えたときに、所属する企業の中でのみ通用するスキルが身についてしまっている可能性がある。そうすると、たとえばキャリアとして転職を考えた場合に、自分のプログラミングのスキルがあまりに汎用的でないために、絶望を感じてしまうことにもなりかねない。実際に筆者も、新卒で入社した会社で、社内の中だけで素早く問題を解決しようというプログラミングに特化してしまっていた時期がある。その結果、自分のプログラミングスキルが外から見たときにまったく汎用的でないスキルであることに気付かされた。たとえば、プログラミングをするときに、モジュール化ができておらず再利用ができなかったり、コードにバグがあったときに、全体に渡ってコードの修正が必要になったりした。また、設計においても、システムのレイヤや依存関係を考慮せずに設計すると、とある個所にバグがあったときに、サービス全体に影響を与えてしまったり、作り直すシステムを局所化できずに、全体を置き換えるような作業が必要になったりする。そこで焦って、社内の業務時間外に趣味でプログラミングを学ぼうとするが、基本的に会社に8時間勤務し、時には残業をすることもあるため、その後家に帰って疲労を伴う中、別のプログラミングを

効果的に学び続けることは現実的ではなかった。また、社内ですらまったく新しいプロジェクトや課題が生じたときに、自身のプログラミングスキルが限定的であったため、都度学びなおす必要があつて効率が非常に悪かったことを覚えている。

ではどのように社内外でのプログラミングを重ねればよいか。そのためには、社内での技術的解決やプロダクト開発において、再利用と機能の置き換えが簡単にできるように、抽象化と疎結合な設計やコードの実装を意識して開発するようにすればよい。1つのプロダクトは、さまざまな機能を組み合わせで開発していく。また、多くのインターネット技術やコンピュータは各レイヤに分かれて構成されている。そのため、課題の解決やプロダクト開発において、技術の抽象化を意識したプログラミングは自然と目指すべき姿勢なのである。

プロダクトや技術的解決を抽象化し、疎結合な設計になるようにプログラミングすることで、各パーツは再利用性が高くなる。さらに、プロダクトを抽象化してレイヤに分けていくことによって、社内独自の仕様やビジネスロジックを含む個所を最小限にとどめられる。抽象化された技術やソースコードは、社内特有の仕様に依存しないため、外に公開しやすくなる。同時に、そのような抽象化された技術やソースコードは、他社においても同様に課題とすることが多いため、注目を浴びやすい。注目を浴びると、その公開されたソースコードに対してバグが報告されたり機能追加のソースコードが送付されたりして、まさに社外の業界とともにプログラミングができるようになる。

オープンソースソフトウェア（OSS）を使ってプロダクトを開発している企業では、OSSの変化にプロダクトを追従させていかなければならない。そのような状況で、時間的に効率が良いからといってプロダクトを直接的かつ密結合に実装を行っていると、やがてOSSのソフトウェアや業界の技術動向の変化についていけなくなり、プロダクトが硬直化し

てしまう。それによって、新しい修正に多大なコストがかかるようになり、最終的には会社としてもお客様に対しても大きな迷惑をかけながら大規模な修正を行う状況が生じる。そのためにも、日々頭を捻ってプロダクトの抽象化と疎結合なプログラミングや技術解決を心がけることで、そのサイクルがプロダクト自体の品質も向上させることになる。

社内外でのプログラミングと相互レビュー

社内外のプログラミングに対する活動を重ね合わせることで、具体的にどういうメリットを得られるかを紹介する。これまで社内での技術的解決やプロダクト開発を抽象化かつ疎結合に開発することで、その成果を社外で公開でき、注目されやすくなることを述べた。その結果、自身が社外に公開したソースコードに対して、社外のエンジニアからパッチや報告が送られてくる。まさにそれが、エンジニアとして最も成長できる機会へと繋がっていく。

社内のエンジニアだけでなく、社外のスキルの高いエンジニアと協力することにより、市場を巻き込んだプログラミングと相互レビューを行えるようになる。これによって、たとえば社内には在籍していないような、特定領域の技術について詳しいエンジニアとパッチの議論が行われたりする。また、抽象化を心がけることで、社内プロダクトに対してOSSを適用しやすくなる。社内で適用する際に足りない機能や利用する中で見つけたバグを報告する機会も増えてくる。そうすると、社外のエンジニアからコードレビューを受けたり、設計について議論できるようになったりする。

時には、世界的に有名なエンジニアにコードレビューを受ける機会も得られて、そのレビューが自分のプログラミングスキルに与える影響はとてつもなく大きい。プログラミングのスキル向上のための取り組みは、簡単に国境を越えるのである。また、

自分自身もそういうエンジニアと共にプログラミングをしていることに情熱を感じ、誇りを持ってプログラミングに取り組めるようになる。そのように生じる内発的動機づけは、プログラミングのスキルを大きく成長させる機会になる。

具体的には、GitHub に公開されているソフトウェアを通じて、社内での取り組みを OSS として公開し、同様の課題を持っている社外、それだけでなく世界中のエンジニアと協調して、機能の提案やコードをレビューする。その中で、自分がコードを書くだけでなく、社内以外のさらに汎用性の高い議論を行いながら、コミュニティやチームの一員としてレビューを行う。社内と社外両方を巻き込んでソフトウェア開発を行うことにより、より効果的にプログラミングのスキルを学ぶことができる。同時に、それらのコミュニケーションを通じて、英語を自然と学ぶ機会も得られる。

抽象化を心がけた活動を行ってれば、次第に社内のプロダクトも技術動向や OSS の変化に強い構成になってきているため、社外のソフトウェアを使いやすくなることも述べた。社外のソフトウェアを積極的に活用しながら、実運用上生じたバグを報告したり、足りない機能を自ら実装したりして、元のソフトウェアへ貢献することもできる。その場合には、GitHub の Pull-Request や Issue といった機能が活用できる。Pull-Request は、追加実装したコードを送るための機能で、作者やコミュニティからレビューを受けられる。方針やバグ報告について実装前に議論したい場合には、簡易に報告ができる Issue を活用する。これらを活用することで、質の高いコードレビューやチーム開発を社内外で行うことができる。チームやコミュニティでの開発では、社会に貢献できているという感情が芽生え、漠然と

1人でプログラミングをするよりも情熱を持って継続的に取り組みやすくなる。

このように、社内外のプログラミング活動を重ねることによって、プログラミングのスキル成長に大きな影響を与え、それと同時に社内のプロダクトの品質向上にも繋がっていく。市場を巻き込んだプログラミングの成長機会に連鎖的に繋がっていくのだ。

市場を巻き込んで楽しむ

本稿では、社会人になってプログラミングを改めて学ぶ中で、効率良く、かつ、内発的動機付けによって活動できる取り組み方について述べた。社内外のプログラミング活動を重ねることは、一見難しいことのように思えるが、その活動が結果的に自身の成長にもプロダクトの成長にも繋がっていく。ぜひ、これから社会人になってプログラミングを行う際には、プロダクト開発や技術的課題の解決において、抽象化と疎結合な設計を意識して取り組んでいただくとよいのではないだろうか。その取り組みを活かして、社内外でのプログラミング活動を重ね合わせ、会社や社会に貢献しながらも、無理なく継続的に楽しくプログラミングのスキルを高めていけるようになると、筆者としても幸いである。

(2019年2月10日受付)

■松本亮介（正会員） matsu1229@gmail.com

京都大学博士（情報学）、さくらインターネット研究所上級研究員、ベバボ研究所客員研究員、Forkwell 技術顧問、セキュリティ・キャンプ講師、本会各種委員、松本亮介事務所所長。2018 年より現職のさくらインターネット研究所で上級研究員を務める。第 9 回日本 OSS 奨励賞や 2014 年度本会山下記念研究賞など、その他受賞多数。2016 年に本会 IPSJ-ONE において時流に乗る日本の若手トップ研究者 19 名に選出される。