［フレッシュマンに向けたプログラミングのススメ］ 基般

# 7 プログラミングと「あ」の書き方を文章で説明することは同じ

増井雄一郎

フレッシャーズの皆様，ようこそ，企業社会へ．

私は，常々，学校という社会と企業を中心とした社会には大きなギャップがあり，「学校で学んできたことやルールと，企業社会のルールがあまりにも違う」ということが問題だと思っています．

少しでもギャップを解消するため，高校生に「世の中では変な大人もたくさん活躍している」ということを伝える，「近未来ハイスクール」というキャリア教育のお手伝いもしています．

さて，このエッセイは，特にITを学んできた人たちに，今まで学んだことと，これから社会で自分の技能を役に立てることのギャップを少しでも埋める手助けになれないかと思って書いています．

皆さんは今までプログラミングを学んできたことと思います．それは授業の中で出された課題だったり，研究の中で見つけた現実の問題を解決するために，事象を観察し，解決方法を考え，手を動かしコーディングをして，実行して，デバッグして，検証したりすることだったと思います．研究によっては解決までたどり着かなかったこともあるでしょう．

最初に「学校と企業ではルールが全然違う」と書きましたが，この「手を動かしコーディングをして」の部分を他の手法に変えることで，仕事のほとんどは成り立ってしまいます．

実は学校と企業がやっていることはほとんど同じなのです．授業との違いは「答えがあるのかないのか」「答えが出ない場合は単位は落ちないが，ヤバいことになる」ぐらいです．でも実際世の中のITプロジェクトの成功率は3割ぐらいと言われているので，大抵の場合は解けなくてもヤバいことにはなりません．

色々なルールは異なりますが，「課題解決」という手法については大差ないのです．

私はプログラミングは課題解決のフレームワークの1つだと思っています．実際にコードを書くことは問題解決手法の1つにすぎません．そしてこのフレームワークはさまざまなケースに応用できます．

これを読んでいる多くの方は，情報系の大学や大学院を出てITに関する仕事をしていくのだと思います．今まで学んだ言語やテクニックがそのまま活かされないことも多いと思います．

しかし私たちがプログラミングを通じて学んだことは「コードを書く」ことだけではありません．

プログラミングというのは「課題を観察して，抽象度の高い形で抽出する」という課題認識と，「抽象度の高い問題の解決方法を考える」ことと，「抽象度の高い解決方法を，現実に落とし込む」実装部分に大きく分かれます．今までこの流れのすべてを学んできたと思います．このことはすごく大きな財産になります．

逆に言えば，コードを書く以外の手法を手に入れれば，プログラミングを通じて学んだこのフレームワークはどんな仕事にでも転用できます．

世の中では「エンジニアが○○万人不足」とか「ディジタル人材不足」などITに関する人材不足の話がこの20年以上語られ続けています．なぜこのようなことになっているのでしょうか．

私は，多くの企業人が「課題を観察して，抽象度の高い形で抽出する」という部分を忘れ，「手元の問題の解決方法を実装する」ことばかりに注力して

いることに原因があると考えています．職業年数が長い人ほどこの罠に陥りがちです．

特に就職してから勉強をサボっている人ほどこの傾向が強くなります．新しいことを学ぶことは，抽象度の高い仕組みを覚え，理解し，現実に落とし込んで活用することです．勉強することは，そのものを覚える価値もありますが「抽象度が高い仕組み」に触れることにも大きな意義があります．

多くの企業人は「同じような仕事」だけをずっと続けていく中で，学生の頃に学んだ「課題認識」や「抽象度の高い問題の解決方法を考える」ことを忘れ，最後の工程にある「解決方法を実装する」ことばかりに目がいくようになっていきます．その方が短期的にはお客さんや上司には喜ばれるでしょう．しかしそれは視野を狭くし，長期的な成長を大きく阻害します．

「エンジニア 35 歳定年説」という言葉がありますが，それは約 10 年勉強しなかった場合，20 代前半に学んだ技術や手法が陳腐化し，新しいことを学ぶ術も忘れた人が陥る状態だと思います．そして多くの企業人がここへ向かっています．

みなさんには企業に入っても学び続け「抽象度の高い仕組み」に触れ続けてほしいと思っています．そうしないと「エンジニアはコードを書いていればいい」という狭い役割だけをこなしていくだけになっていきます．

また，実際にコードを書くということについても，「製品を構成するコード」以外にも多くの場面で活躍できることがあります．

私が所属していた（株）トレタ[☆1]では，図 -1 を使ってエンジニアがどのようなフェーズで，どのような能力を発揮して開発やビジネスにかかわっていくのかを表しています．この図は業務のごく一部，新規事業開発だけを表しています．課題発見から企画，開発まで各フェーズにエンジニアがかかわっていきます．

私たちが提供できるものは「コードを書いて製品を作る」ことだけでなく，問題解決の全体に対して，エンジニアリングを使って効率的な手法や新たな視点を届けることです．それは「製品を作る」ことだけではありません．

狭義にエンジニアやプログラマというと，「コードを書いて製品を作る人」と捉えられ，期待される役割がこの範囲に限定されてしまいます．日本の IT やエンジニアが海外に出て行けない理由はここにあると思っています．

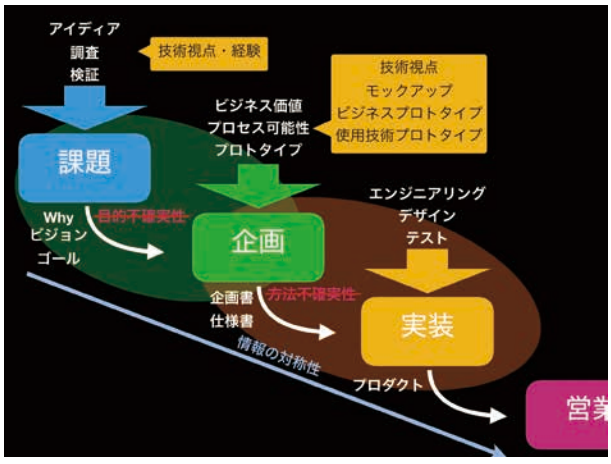
しかし，実際には「技術を軸にして課題発見から解決をする」「抽象度の高い解決方法を考え，広い範囲に適用する」ということがエンジニアの仕事だと私は思っています．

そしてエンジニアがこのような課題解決が可能なことを自分で主張し，証明していく必要があります．

（株）トレタでは，全社員に向けて入社時研修をやっており，その中で「アプリはどのようにして作られるのか」という話を 1 時間ほどしています．

その中で私は「プログラミングとは国語に近く，説明書と同じ物だ」という話をしています．

世の中の多くの人は，プログラミングとは英語や数学のようなもので，いわゆる「理系」の仕事だと思っています．しかし私にとってはプログラミングをすることと，文章を書くことはまったく同じよう




■図 -1　新規プロダクト開発環境フロー

[☆1] https://corp.toreta.in/

に捉えています．

コンピュータに分かるように指示を書くのにはプログラミング言語を使い，人間に伝えるのは日本語や英語を使います．ただ使う言語が違うだけで「お願いしたいことを的確にまとめて説明する」ということは同じです．しかし，これだけのことが難しいのです．

研修の中で，例として「“あ”の書き方を，文章でだけで説明してください」という話をしています．「あ」という文字は誰でも書けますが，それを文章だけで説明しろと言われると途端に難しくなります．私は，まずは適当に正方形を書いてもらい，それに縦横を16分割する線を引き，分割の横軸に1～16，縦軸にA～Pと割り振ってもらいます．次に「1のC」から「16のC」に一本線を引いて，続いて「8のA」から「7のE」「10のM」に線を引きます．このような感じで「あ」の書き方を文章だけで説明することができます．これ以外にも，多次元式で一筆書きで「あ」を書く数式を作る方法もあるでしょう．

このように，普段，何気なくしていることでも，文章にして誰でも分かるようにすることはすごく難しいのです．

行動を文章にするには，自分のしていることを観察し，分解し，いろんな状況でも再現できるように抽象化し，他の人が理解できるように具体化して説明する必要があります．

エンジニアはコードを書くとき，「あ」の書き方を説明するように，一見簡単な事象を複雑なコードに落としていきます．

今までプログラミングを通じて学んだ「事象を分解して，抽象化し，コードとして再構築する」ということを自覚的に行い，さまざまなシーンでそれを発揮することで，みなさんは「コードを書く」だけでなく，エンジニアとしてさまざまな課題解決や価値を発揮することができるでしょう．

それはコードの場合もあれば，仕事の仕方の場合もあれば，上司や顧客との関係の場合もあるかもしれない．でも今まで学んだことは必ず活きてきます．むしろ「いいからやれ」というようなショートカットをしてくるダメな先輩方を叱っていってください．

さまざまな技術やプログラミングを通じて学んだことを活かして，新しい価値を作って届けていくことを，このエッセイを読んだフレッシャーズに心がけてもらえれば，私は非常に嬉しいです．みなさん，がんばってください．

（2019年2月25日受付）

■増井雄一郎　masui@masuidrive.jp

「風呂グラマー」の愛称で呼ばれ，トレタやミイルを始めとしたB2C，B2Bプロダクトの開発を行うかたわら，オープンソースへのかかわりなど，外部へ向けた発信を積極的に行っている．