［フレッシュマンに向けたプログラミングのススメ］ 基般

5 ゲームチェンジを楽しもう

渋川よしき | フューチャー（株）

車が走るゲームを想像してみてください．1つのゲームは，競争していてほかの車とぶつかると時間のペナルティが課せられるとします．もう1つのゲームは派手にぶつかるほど加速するとします．同じ操作のゲームでもスコアのルールが変わるとゲームプレイが全然変わってくるのは容易に想像できるでしょう．同じソフトウェアのスキルであっても，社会人と学生では評価が変わってくるため，ベストな動き方が変わってきます．どのようにハイスコアの稼ぎ方が変わるのか秘密を紹介します．

お仕事の基準はお金と時間

仕事でも研究でもなんでも，評価の基準となる物差しが必要となります．研究であれば可能性の検証，数パーセントの効率改善などが素晴らしい成果となります．ビジネスの世界の共通指標は，基本的にお金と時間です．会社のゴールがあって，それが部署のゴールに，最終的に個人のゴールへと，トップダウンに目標設定を行って仕事を行う会社がほとんどでしょう．最終的にはどこかのポイントにはお金が入ってきます．

時間はお金に換算できることが多いので最終的にはお金が基準となりますが，普段の活動では分かりやすい時間を基準に考えることも多いです．また次のような別の指標が出てくるかもしれません．これらの数値も最終的にはお金に換算されます．

- 月間アクティブユーザ数
- ユーザの継続率
- ユーザ1人あたりの売り上げ
- 不具合件数（少ないほどよい）
- ユーザの作業時間の節約

事業会社の内製開発でも，B2C（一般の人向けのビジネス）でも，B2B（対企業のビジネス）でも，開発によってもたらされる価値と，その開発にかかるコストが成立して，初めてシステムの開発が始まります．どんな形態で仕事をするにしても，数字で説明するスキルは大切になります．

たとえば，50人月かけてこのシステムを作って1万人のユーザが月に10分ずつ節約できるなら，月に10人月分のコスト削減になるので，5カ月で費用が回収できます，という感じです．ベンチャーの投資も，1年かけて開発し，3年間の間にこれだけ広告費を投入し，有料ユーザを何万人獲得できれば採算にのって5年で回収できる，みたいなビジネスプランを立ててベンチャーキャピタルに説明をして投資を受けるでしょう．ベンチャーでなくても，自社サービス開発の費用（ほとんどは人件費でしょう）を会社で用意する，社内システム刷新の費用対効果を計算して稟議を通してIT予算を獲得するなどもあります．逆に，たとえすばらしい機械学習アルゴリズムを考案し，効率が1パーセント改善されると仮定すると，1年で回収するなら，10億円のビジネスに適用しなければ1人の人件費も回収ができず，継続できない，というのが仕事でコードを書く世界ではあり得ます．

仕事でかかわる他者（上司だったりお客様だったり）が大事だと思っている数字でやりたいことがうまく説明できると，比較的簡単に信用が得られます．仕事の中でちょっとした研究をする時間を確保したり，勉強会参加の研修費用を出してもらったり，仲間に協力してもらったり，いろいろなことが可能になります．

時間を大切に

みなさんが働いて，その時間でソフトウェアを生み出していきます．そのソフトウェアがお金を生み出します．みなさんが 1 カ月働けばその分のお給料がソフトウェア開発にかかったコストということになります．実際には税金や社会保険の費用，開発機器，オフィスビル，バックオフィスの人件費などでお給料の 2 倍程度がコストになると言われています．

そうなると，短い時間で効率良くソフトウェアを生み出すことが大切になります．半分の時間で生み出せばそれだけ投資対効果は 2 倍です．また，多くの人と一緒に仕事をすることになりますので，チーム全体の時間を節約することが大切です．あとから大きな無駄が発生しないように，小さい時間の投資をすることが大切になったりします．

- 環境構築を短時間で終わらせられるようにドキュメント化したり，スクリプトで自動化をする
- バグの発見や引き継ぎなどの場面で，みんながコードを理解して読みやすくするためにコーディング規約や名前付けのルールをきちんと決めておく
- ちょっとの手間をかけて自動テストを導入する
- 生産性を高めるライブラリを使う
- いくつものソフトウェアで構成された複雑なマイクロサービスの状況を簡単に把握するために監視のミドルウェアを入れる

みなさんは本誌を読まれているぐらいなので，大抵の IT 会社ではエリート扱いでしょう．IT 業界は人気な業界ということもあり，会社によっては大学までソフトウェアを学んでいなかった人たちも入ってきます．業界全体が人手不足なので，教育に投資して絶対数を増やす会社はそれだけで社会貢献をしていると言えますが（弊社も行っています），みなさんがそのような周りの人に積極的に教えて，スキルをアップしてもらう，というのも全体の効率改善には大切です．

時間の短さとカレンダーの日付の早さが異なる点は注意が必要です．今日中に終わるべきタスクがなかなか終わらない！ということは大学のレポート作成などではよくあるでしょう．筆者も学生時代は，睡眠時間を削ってなんとか実験レポートをやりきっていました．日付はもちろん大事ですが，仕事では長時間労働は制限されています．会社は労働時間をきちんと管理する義務があります．働き方改革関連法案もあり，以前からあった残業時間への規制はさらに厳しくなりました．遅れているからといって，こっそりと徹夜して終わらせるというサービス残業は法律違反です．また，若いうちはよいのですが，だんだん体力は落ちてきます．長時間労働でカバーするのを習慣化すると，子育てや介護等で仕事に使える時間が減少した場合，仕事のパフォーマンスが落ちるということになりかねませんし，効率改善を常に考えて残業をしすぎない習慣を若いうちからつけていくのが大切です．

仕事のアウトプットにこだわる

何社かで採用活動を見てきましたが，いままでどんな仕事をこなしてきたか，どんな立場でそれにかかわってきたかの職務経歴が一番重要視されます．「この言語でこういう仕事をしたい」という希望や趣味で行ってきたことは参考にしかなりません．目の前にある仕事をきちんとこなすことが自分の市場価値を決めます．

自分の価値を高めるような仕事を選びたいですよね？　みなさんが何か「○○したい」という気持ちを持っていても，思うだけではなかなかそういう仕事が回ってくるチャンスはきません．口頭で説明しても，気持ちの強さまでは伝わりにくいです．その場合は，趣味の時間も活用して，動くソフトウェアを作ったり，ブログやら薄い本やらを書いて「形」にすると「自分がこれだけやりたい」と思っている気持ちが上司や周りの人に伝わります．周りの人も，仕事を任せるならモチベーション高くやってくれる人にお願いしたいと思っています．その結果として自分のやりたい仕事を引き寄せて，業

務時間の中でさらに知識を深めて，成果も出し，実践的な技能も身につきます．

仕事をしていて身につけた知識や，調べたことはなるべく外に発表した方がよいです．社内のイントラやWikiに書いても，案外後から見つけにくいものです．外部に公開しておく方が検索エンジンで見つけやすくなります．もちろん業務内容をそのまま書いてはいけないこともあるでしょうし，上司，あるいはお客様にきちんとOKをとることが大切です．

技術は新しすぎても古すぎても良いわけではない

趣味や研究でコーディングする場合はなるべく新しくて楽しい技術を選定する人が多いでしょう．しかし，仕事でコーディングをする場合は，必ずしも新しければ新しいほど良いとは言えません．新しいけど安定していなければその対応に時間が取られることもあり得ます．バージョンアップ作業も時間を消費しますし，その際の検証，不具合修正そのどれも時間がかかります．

かといって古いものを選ぶのも正解ではありません．近年ではセキュリティに関する意識がかなり高まっています．大事なお客様情報を保持しているサーバのセキュリティホールを放置してしまい情報漏洩につながったとすると，賠償だけではなく，失った信用を取り戻すのはとても大変です．商用のサービスやミドルウェアであれば必ずサポート期限が決まっていてそれを超えると修正が得られなくなります．システムにインストールされているソフトウェアに，脆弱性データベースに登録されている危険なバージョンに該当するものがないかを自動検知するサービスもあります．Node.jsであれば，標準のパッケージマネージャがインストール時に危険性が指摘されたバージョンがあるかどうかを自動チェックします．脆弱性をなるべく早く潰していくことが大切です．

ソフトウェアのバージョン番号の付け方や管理方法にはいくつかありますが，近年，重要とされるシステムでよく見られるのが，LTS（ロング・ターム・サポート）です．積極的な機能追加をするバージョンと並行して，長期間サポートをし続けるバージョンを提供してくれています．後方互換性を維持しつつ，セキュリティの問題や不具合の修正は手に入ります．サポート期間が長いものを使う方が，メンテナンスを減らせますし，いつ行えばよいか，見通しが立てやすくなります．バージョンアップ作業であっても作業時間がかかる以上，誰かがそのお金を負担します．B2Bの開発ではお客さん側で稟議を回すなどの必要があります．内製開発やB2C開発などでも，その作業で時間を取られる分，新機能開発の時間が減ったりするでしょう．

もちろん，全方面に渡って守りの姿勢を保たないといけないわけではありません．特定の戦略のコアになるコンポーネントは最新のものを使って競争力に繋げることもありますし，それがOSSであれば，そちらの開発にもコミットしつつ最新版を良くしていく，ということもあります．使える時間という資源が有限なので，不要な競争力を生まないところでは節約しつつ，強みになるところに使うことが大切です．コンピュータサイエンスやネットワーク，データベースといった枯れにくい知識に投資することで投資効率をあげる方法もあります．

システムの時間軸の長さ

2000年問題など，古いソフトウェアのメンテナンスは時折ニュースになります．あれは一部の怠慢な会社の所業なんでしょうか？　必ずしもそうではありません．また，長期間に渡る銀行のシステム刷新もニュースで見かけて，なぜそんなに長くかかるのか疑問に思った方もいるでしょう．ソフトウェアというのは思った以上に長期間に渡って使われたり影響を残すことがあります．

たとえば，みなさんが新車でホンダのフィットを買ったとしましょう．2019年4代目のフルモデルチェンジが行われると噂されていますが，安いので現行モデルを買ったとします．3代目の発売は6年前でした．フィットはシャシー開発も一緒に行うようなホンダの基幹車種です．きっと開発期間も長くコストもかけて開発されて

いるでしょう．とりあえず 5 年とします．その後 11 年間乗り続けて，そのときの車検でタイミングベルトを交換します．その交換した部品は，17 年前に発売されて 11 年前に販売されましたが，実は 22 年前に設計された部品，ということがあり得ます．フィットをベースにした派生車種だと，発売時点でさらに 2 ～ 3 年追加されることがあり得ます．データの寿命が CAD のソフトウェアやそれを動かしている OS やハードウェアの寿命よりもはるかに長くなります．それでも，きわめて重大な不具合であればリコールで改善対策品を作る必要があるため，そのタイミングでデータアクセスが必要かもしれません．民生品ですらこうですので，いまだに世界を飛んでいるボーイング 777 とか，とんでもないですよね？

もちろん，みなさんがかかわるシステムのすべてがこういうわけではなく，数年でリプレースされるものも多いと思いますが，もしかしたら，そのような長い目線が必要となるタスクにかかわることもあるでしょう．以前から動いていたシステムとの相互接続など，古いシステムに配慮する必要も出てくるでしょう．オープンで，多くの人がかかわって改良された規格に寄せることで，仕事を楽にする，といったことも必要になるでしょう．

社会人としてソフトウェアを書く楽しさ

いろいろ書いてきましたが，変わることの好き嫌いはあると思いますが楽しさはあります．純粋な研究開発をするチャンスを得ることは難しくなるかもしれませんが，仕事の中でうまく折り合いをつけて新しいことにチャレンジできる職場は多いでしょう．また，何よりも自分が作ったソフトウェアそのものをほかの人が使う，それにより，ユーザの生活や仕事が改善されるのを見たり実感するのはうれしいことです．多くの人に使ってほしいと思って，認知度の高い B2C の企業に目を向ける方は多いと思います．筆者は電気電子工学科でしたが，やはり就職ではコンシューマーエレクトロニクス企業が人気でした．ですが，お客さんと直接対話しやすい社内システム開発も楽しいですし，有名企業の大きなビジネスのコアを支える B2B の開発も，B2B2C の開発もどれも楽しみを見つけることはできると思います．ぜひ，新しい環境を楽しんでください．

（2019 年 1 月 18 日受付）

■渋川よしき　y.shibukawa.7i@future.co.jp

東京工業大学電気電子工学科卒業．（株）本田技術研究所の自動車設計支援システム開発，（株）ディー・エヌ・エーでゲームミドルウェアや開発支援ツールの開発を経て，現在フューチャー（株）でコンサルタントをしている．