プログラマの生き様

NP-complete

2015-12-31 コミックマーケット 89

第 1 章

プログラマとは

プログラマというのはもちろんプログラムを書く人のことです。ひとくちにプログラマと言っても、様々な種類があります。

1.1 ソフトウェア業界の種類

プログラマは、もう少し広い定義である**ソフトウェアエンジニア**の一部と言えます。ソフトウェアエンジニアが所属する**ソフトウェア業界**は、様々な属性により分類されているといえます。
例えば

- プラットフォーム: PC / Web /ゲームコンソール / スマートフォン / ハードウェア など
- 言語: 低級な言語 (C/C++) / 堅い言語 (Java/Scala) / 軽量言語 (Ruby/Python)
- 顧客: BtoB (企業向け) / BtoC (消費者向け)
- 作るもの: 業務アプリケーション / パッケージソフト / ゲーム / サービス
- 組織構造: 多重下請け構造 / 自社開発
- 業務範囲: 設計・実装・テスト・運用・全部

ほとんどの企業がどれかの属性を組み合わせて持っていると思います。例えば銀行システムを作る企業だと BtoB, Java, 業務アプリ, Web + サーバサイド + 専用ハードウェア (ATM) になってくると思います。(銀行システムを作る会社のことはよく知らないので推測ですが) また、ひとつの属性が決まると他の属性もおのずと決まってくることも多くあります。例えばハードウェア制御は現状ほとんど C/C++ を使っているのではないでしょうか。

1.2 世の中のプログラマ像

世の中でいわゆるプログラマとしてイメージされているのは、上記で言う **BtoB && Java && 業務アプリ && 多重下請け構造** の業界に属し、いわゆる **SI 業界**というやつです。中でも**多重下請け構造の最下層**な人たちは、まさに**世間のイメージ通り**のプログラマで、技術も伸びずに家にも帰れず安月給で使い捨てられる運命にある人です。**なったら死にます**。

それでは多重下請け構造の上層、いわゆる **SIer** はどうなのかというと、この人たちは基本的に**プログラミングをしません**。ソフトウェアエンジニアではあるかもしれませんが、プログラマと認識されることはほとんど無いようです。本質はコンサルタントであり、設計だけして (時々設計すらせ

ず) 下に丸投げし、**プログラミングは奴隷がすること**と考えています。新卒 1 年目に**奴隷の限界を知るために**プログラミングをさせられる程度で、あとの人生は出世レースに勤しむようです。終身雇用が主流で、歳とった時の給料は**非常に良い**ようです。(ただし今の若い人が引退する頃にこの業界が存続しているとは思いません)

この業界では上流の人のことを**システムエンジニア (SE)** と呼び、奴隷のことを**プログラマ (PG)** と呼びます。他業界ではプログラムを書くこと自体に誇りを持っている人が多く、その人たちを**間違っても SE と呼んではいけません**。

ゲーム業界もプログラマと深く関連付けてイメージされていると思います。筆者も子供の頃ゲームが好きで、ゲーム作る人になりたいという思いでプログラミングを始めました。しかしゲーム業界自体が**すでに死に体**なのでその考えは捨てましょう。伝え聞くに、この業界も**地獄のようなもの**らしいです。

1.3 この本で言うプログラマ

筆者は上記のような業界にはあまり詳しくなく、**Web && BtoC && 軽量言語 (LL) && サービス && 自社開発** な世界の住人です。この **Web サービス**業界は、技術を追求することと**そこそこの**給料を両立しやすく、自社開発なので**やらされてる感**のある仕事は多くありません。技術はオープンで、転職はしやすく、圧倒的に会社より労働者が強い世界です。勉強会や転職者のつながりで人材の交流は非常に活発で、プログラミングが好きでプログラマとして人生を全うしたいならよい環境だと思います。

筆者は Web 業界の人間であり、知り合いもほとんどが Web 業界、そもそも世間で名が知られているプログラマは多くが Web 業界の人なので、この本では、主に Web 業界のプログラマの生き様を紹介していきます。

また、Web 業界で働くソフトウェアエンジニアは**すべてプログラマ**なので、この本でプログラマという語は、多くの場合ソフトウェアエンジニア全体を指す意味で使われています。

第 2 章

プログラマのなりかた

プログラマになるにはどのような方法があるのでしょうか。世間では長く就職難が続いているようですが、ソフトウェア業界は常に人不足と言われ、もしかしたらかなり特殊な状況なのかもしれません。

2.1 大学に行って新卒採用

もちろん、普通に大学に行き普通に就職することでなれます。プログラマに直接関わる学部としては**情報工学科**などがあります。

BtoB の場合、SIer などの最上流は普通に**超大企業**なので、一流大学や情報工学科を卒業しないと、まず新卒で入ることはできません。一方、下流の企業は常に**奴隷不足に悩んでいる**ため、たぶん**人間でありさえすれば入れる**のではないでしょうか。大卒である必要もありません。

どちらも入社時にプログラミング能力はあまり問われません。前者は大企業なので手厚い研修と OJT が用意され、後者は **1 週間の研修終えると実務経験 3 年のプログラマとして売られていく**なんてうわさ話を聞きます。

Web 業界の場合、大企業とそれ以外とで大きく変わります。大企業の場合は、SIer と同じように将来性を買われて未経験でも採用されますが、実績があると圧倒的に優遇されます。研修はありますが、それほど手厚いというレベルではありません。研修は実績があるとスキップできたりします。

一方、中小企業の場合は、新卒段階で十分なプログラミング能力が要求されます。そもそも新卒採用をしていない場合もあり、その時は実績を持って突撃するか声がかかるのを待つしかないでしょう。もちろん即戦力が前提で、**研修など期待してはいけません**。

2.2 技術力で就職する

他の業種ではなかなかできない方法だと思いますが、Web 業界の場合、年齢・学歴・職歴にかかわらず**技術力さえあれば**就職できます。人格も問われません。極端な話、**中学卒業直後**からトップ待遇の給料で働くこともできます。

Web 業界では、**人月の神話**は完全に否定されており、生産性は能力により何十倍も開きがあることを実体験として理解しています。プログラマではない人事部では、その能力を測ることが難しく、伝統的に採用に強い影響力を持っていません。プログラマの採用面接はプログラマが行っている会社も多く、**現場のプログラマ**が**こいつと働きたい**と言えば、何より強力な採用動機となるのです。

エンジニアが選ぶので**エンジニアの世界でまともな人間**であればよく、つまり**世間一般では破綻した人格**でも問題ありません。それどころか、破綻していることを考慮して、いろいろ便宜を図ってくれる場合があります。

技術力で就職するには当然、技術力を何かしらの方法で可視化しなければなりません。現代ではたくさんの方法があります。

- オープンソース活動
- 技術コミュニティや勉強会の運営
- SECCON や ISUCON などの競技プログラミング
- セキュキャンなどの学生向けテクノロジーイベント
- ガチクラック 自主的なセキュリティ調査

技術コミュニティも本人が技術に明るくないと人は付いてこないため、結局はすべて**実際にプログラムを書いて動かす**ことが何よりも重要だと言えます。基本的にすべてコードは公開したほうが良いでしょう。コードを公開しない場合、競技プログラミングの実行時間など**数値的に比較可能**でないと判断できません。例えば**クローズドソースでゲームを作っても全く評価されない**でしょう。

この業界は技術さえあれば生きていけるので、声がかかった瞬間に学校を辞めて就職しても、将来困ることはないと思います。学歴を気にする人も少ないので学歴差別もされません。実際、現場のプログラマも「こいつ今すぐ学校やめてウチ来てくれないかな」なんて話をよくします。

2.3 まとめ

- ひとくちにプログラマと言っても硬い業界からゆるゆるな業界までいろいろある
- それぞれに合った入り方がある
- Web 業界ならクズでもニートでも留年してても中卒でも技術さえあれば入れる

第 3 章

プログラマの仕事

プログラマの仕事とはなんでしょうか? プログラムを作ることだと思っている人は多いと思いますが、違います。それは本質ではありません。本質を捉えるためには、まず最初に仕事とはなにかというところを明らかにしなければなりません。

仕事の本質なんて考えたことありますか? 難しい質問ですよね。世の中の人がこの質問になんと答えるかは全く想像できませんが、少なくともプログラマから見た仕事の本質とは、**問題を解決すること**です。

3.1 問題とはなにか

人や社会は様々な問題を抱えてきました。アレがしたい、ソレはしたくない、コレが面倒、など。**需要**と言い換えても良いかもしれません。

例えば、昔は各家庭でパンを焼くのが当たり前でした。しかし、各家庭に窯を持ったり、各家庭でパンをこねる作業をしたりするのは**非効率**です。パン作りが得意でない人もたくさんいたでしょう。社会が発展するにつれて、毎日のパン作りに使う労力が無駄だという問題が生まれました。

3.2 初期の仕事

いったい誰がこの問題を解決したのでしょうか。もう解っていると思いますが、パン屋という集団です。パン屋は大きな生産量を期待できる窯を備え、パン職人という労働力を集約し、パン作りのノウハウを蓄積し、効率的にパンを供給することができるようになりました。この仕事により市民はパンを作るという労務から開放され、問題は解決したのです。紀元 1 世紀のポンペイ遺跡にもパン屋跡があり、この頃すでにパンは各家庭で作るものではなくなっていたようです。

3.3 エンジニアの仕事

それではもう少し範囲を狭く、**エンジニア**の仕事の本質とはなんでしょうか。それは**テクノロジーで問題を解決する**ことです。**技術で問題を解決する**とも言いますが、テクニックと混同するのでテクノロジーとしておきました。工業的手法や工学的手法と言ってもいいと思います。

パンの例を続けると、社会が発展し人口が増えるにつれ、それに比例するようにパンも必要になりました。パンの生産量を増やすにはパン職人を増やすしかありません。しかし、職人を育成するの

には当然コストがかかり、また職人を増やしても増加する生産量は微々たるものです。ここで、パン職人の育成と生産量の増加が、パン需要の成長に対処できないという問題が生まれました。以前の仕事の方法を拡張し続けるだけではもう問題を解決できません。

こうなるといよいよエンジニアの出番になります。エンジニアは**パンを作る機械を開発**し、パン工場を建て、パンの大量生産を可能にし、問題を解決しました。現在我々が食べているパンのほとんどは工場で機械により作られたパンであり、街のパン屋が手作りで作っているパンはプレミアム価格が付きちょっとした贅沢品となっています。

パンに限らず様々な分野で、昔ながらの仕事の方法で手数を増やすだけでは需要の成長に間に合わず、世の中の仕事の多くが**エンジニアの仕事**に移り変わっていった、というのが 19 世紀から 20 世紀の流れだと言えるのではないでしょうか。

3.4 プログラマの仕事

プログラマの仕事の本質はもう見えましたか? それは**ソフトウェアで問題を解決すること**です。

大事なことなのでもう一度言います。**プログラマの仕事はソフトウェアで問題を解決すること**です。プログラムを作ることではありません。

3.5 プログラマの与えられる問題

さてそうなると、プログラマにどのような問題が与えられるか、ということが重要になってきます。これは業界によってかなり違い、むしろこの違いが業界や身分を分けていると言っても過言ではありません。

例えば SI 業界を見てみると、最上流にいる SIer が解決するのは、顧客の抱えている問題です。何かの機能を追加したい、何かを自動化したい、そういった問題を解決するシステムを考えます。SIer はそのシステムを実現するのが大変だという問題を新たに抱え、問題を分割再設定して下流の会社に流します。いくつかの層を経て問題は分割されていき、最下層のプログラマが解決するのは「このメソッドを実装して欲しい」といったような規模の問題になります。最下層のプログラマにとって、プログラマの仕事はプログラムを作ることで間違っていないわけです。

一方 Web サービス業界では全く違い、「世界はどんな問題を抱えているか」から考えます。与えられた問題を解決するのではなく、問題を自体を探すところから始まります。本当にそれが問題なのか、その問題を解決したら世界は喜んでくれるのか、それすら確信できません。もしかしたら Web 業界は「**世界は問題を問題として認識していないという問題**」を解決しているのかもしれません。誰も気づいていなかった世界の需要という大金脈を掘り当てた人たちは、皆大成功し大金持ちになっているのです。「もうそれが無かったころの生活なんて思い出せない!!」ってサービス 1 つ 2 つあるのではないでしょうか? 筆者にとっては twitter がそれです。需要を掘り当てるのも自分の能力と裁量次第。ここに Web サービスの楽しさが詰まっています。

Web 業界は下請け構造は無いので、自分でその問題を解決しなければなりません。Web プログラマの仕事の本質は自ら問題を見つけ自らその問題を解決することです。よいプログラムを書くのは結構なことですが、問題を解決しないのであれば無意味です。問題を解決できるならプログラムを書く必要すら無いのです。

3.6 まとめ

- プログラマの仕事は問題を解決すること
- 所属する組織や階層で問題の規模や性質が大きく違う
- 問題解決できないプログラマは無能

第 4 章

プログラマの性格

神話の時代より、

> プログラマの三大美徳は 「**怠惰**」 「**短気**」「**傲慢**」 である。

と言われています。7 つの大罪のうち実に 3 種が美徳とされるあたり、プログラマは実に罪深き生き物なのです。

現代でもこの美徳は十分に有効で、優秀なプログラマと言われる人はこの素質を高レベルに備えています。つまり**だいたいみんなクズ**です。

特に**プログラマがプログラマを採用する** Web 業界では、どんどんクズの純度が濃くなる傾向にあります。それが許容される世界なので、みんなクズであることを自覚しているし隠しもしません。優秀であればあるほど、より高レベルのクズを許容されるので、Web 系プログラマは常に優秀であろうとします。それと同時に、さらなる許容を引き出すために、よりクズの高みを行動で示します。Web 系プログラマが勉強熱心なのは、このクズ利権を守るためと言っていいでしょう。

一方、他の業界はどうかというと、例えば SIer などは完全に **BtoB の客商売**なので、**ものすごくまともな人**しかいないように見えます。Web 業界と違い、まともじゃなさを少しでも感じたら面接で落とされるでしょう。しかし、たまに勉強会などで見かける SI 業界のデキる人、懇親会等で話を聞いてみると**クズを隠せる程度の大人力のある**クズと判明することが多くあります。

結局今までに見たことのある優秀な人はほぼ全員この美徳を備えていました。演技が上手いなら SIer、下手でも Web 業界なら生きていけます。クズであることのデメリットはほとんどありません。逆にいかにも真面目一辺倒と言った感じの人はまずボンクラです。

4.1 どのような人が向くのか

具体的にどのような人がプログラマ向きなのか、身近な例としてゲームの好みを紹介します。

反復作業ができない

とにかく面倒くさがりです。単純作業や反復作業はできません。同じことを 2 度目にやるタイミングで自動化できないか考えます。自動化できないことはその作業そのものが苦痛になります。

例えば**音ゲー**はものすごく苦痛です。常に同じ譜面で完全に作業させられている感があります。面倒なので自動化したいけど、自動化したらゲームにならないので自動化できません。

パズドラの木曜ダンジョンも、面倒くさいのでスタミナを消費してドラプラを引くガチャにして欲しいです。

同じことの繰り返しでないものを好みます。アナログの**ボードゲーム**や、アーマードコアのような**対人ゲーム**は、常に新鮮で刺激的です。

自動化することが喜び

プログラマはゲーム自体の面白さにはさほど興味がない傾向があります。プログラマにとって一番楽しいことはコードを書くことなので、コードが書ける余地のあるゲームは、それ自体の面白さ以上のポテンシャルを秘めています。コードが書ける余地というのはつまり**自動化**のことです。

全然面白くないけど自動化できる**艦これ**はすごく人気がありました (過去形)。当時、艦これは**薄い本を楽しむために**絶対に押さえておかないといけないコンテンツでした。しかし、あまりにも面白くないので多くのプログラマはすぐに自動化を考えました。API を調べてみると**ものすごく簡単に自動化できる**ことが判明し、自動化が**すぐに成果に結びつく**予想が付いたので、実装に着手したのです。艦これ開始 2 週間後くらいにあったプログラマのイベントでは、そこかしこで艦これ API の話をしていた記憶があります。だいたいその頃にはみんな遠征の自動化は完了していました。

上の例で重要なポイントは、プログラマは**ゲーム自体のめんどくささ (怠惰)**、**自動化することのめんどくささ (怠惰)**、**自動化の成果がでるスピード (短気)** を考慮し、それでも自動化のメリットが遥かに強いと結論し、自動化に着手しています。自動化は規約違反ということは**一切考慮していません (傲慢)**。自動化はまさに三大美徳が存分に発揮された結果であると言えます。

なお世の中には、コードを書いて自機のアルゴリズムを設定し、戦闘はすべて自動で行う**プログラマ向けのゲーム**などもあります。

待てない

短気なので**ロードが長い**ゲームや**無駄にアニメーションがある**ゲームとかも嫌いです。パズドラの進化アニメーションとか最悪です。木曜ダンジョン後に控える大量のペンドラ進化は死にたくなるほどの苦痛です。ガチャのアニメーションも最悪です。プログラマの目にはとっくに返ってきている API **レスポンスしか見えていません**。

集中力も長く持たないので、単純作業で長時間の集中力を要求する**音ゲー**は相性最悪です。プログラマの短気の前では 3 分ですら長時間なのです。

我慢が必要なゲームも嫌いです。マリオなどのアクションゲームは、敵の動きに**タイミングを合わせる**必要がありますが、それが待てません。やったことないですがメタルギアとかもたぶん苦手でしょう。

俺 TUEEEE が好き

傲慢なので俺 TUEEEE できるゲームは基本的に好まれます。他人との協力プレイもあまり興味がありません。勝敗にもこだわらずに自分のキルレートさえ高ければいいという考えもします。RPG は高レベルでボスを蹂躙するのが好きだし (ただしレベル上げは嫌い)、シミュレーションは大軍をもって無傷で勝利するのを好みます。無双シリーズなんかも向いているのではないかと思い

ます。

第 5 章

プログラマの勉強

プログラマを取り巻く環境は進化が**異常に**速く、常に新しい技術を学び続けなくてはなりません。あらゆる技術は公開され、すぐに多くの人が学ぶことになるので、学ばないことは相対的に技術力の低下になります。学ぶことを止めた瞬間から年老いていき、プログラマの定年に近づいていきます。必要な学習量は他のどんな職業よりも多いのではないでしょうか。

なお、技術が凄い人で学ぶことを苦痛に思っている人を見たことがありません。常に楽しんで学び続けられる人でないとプログラマとして大成するのは難しいのかもしれません。

5.1 何を学ぶか

とにかく広く浅く学ぶ

基本的にプログラミングに関する**あらゆること**を学ばなければなりません。古の時代から積み重ねてきた基礎的な技術から、バズワードと揶揄されるような中身のはっきりしないものまで、文字通り**目に付いたあらゆるすべて**です。

ただし完全にマスターする必要はありません。そんなのは不可能です。その技術でどんなことができるのかをざっくりに把握し、そして重要なのは**自分の知っている他の技術との関係性**を理解することです。新しい技術が自分の得意分野と関係するなら、将来性の判断もつきやすいでしょう。定着しそうな機運が見え始めたタイミングで深く学べばよいのです。現時点で関連性がなさそうなら、将来触れる機会もほぼ無いと思います。なんかこんな話題があったなぁ程度の記憶を脳の片隅に残しておけばいいと思います。すっかり忘れてしまっても構いません。

深く学ぶことより重要なのは、**目に付く数**を増やすことです。Twitter や技術系サイトを常にチェックし、アンテナを高く張りましょう。幅広い知識が得られるように情報源を選ぶのはなかなか簡単ではありません。Web 系の大企業なら社内チャットの**技術チャンネル**が必ずあるはずです。**興味の範囲外も含めて**様々な技術情報が勝手に集まって来るので非常に有用です。大企業の一番の利点はまさにこの点です。(技術チャンネルが無いならさっさと転職しましょう) 積極的に参加してたくさんの知らない単語を目にしてください。

あ、**誰が何の技術に興味持っているか**は非常に重要なので覚えておきましょう。

ひとつ深く学ぶ

興味がある技術が見つかれば深く学びましょう。一度に学ぶものはたったひとつで十分です。そのかわり深く深く徹底的に調べましょう。その技術について人に教えられるようになれば、コミュニティの中でもハブのような重要な役割になり、それ以外の技術の情報がどんどん入ってくるようになります。

5.2 いつ学ぶか

余暇の時間をメインに勉強するのはお勧めしません。だいたい仕事の後なんてのは疲れているし、休みの日には遊びたかったり用事があったり、勉強を習慣づけるには不適切だと言えます。勉強するのに最も適しているのは**業務時間中**です。

業務時間内で実際に仕事をする時間を **3 割程度**に抑えましょう。それ以外の時間は、チャットに勤しみ、Twitter を眺め、Web サイトを巡回し、インプットを増やします。仕事中になにか気づいたことがあれば、とりあえずその仕事はほっぽり出し、ブログを書くとか、ライブラリ化するとか、すぐにアウトプットするべきです。急がば廻れという言葉があるように、寄り道したほうが最終的に成果が出ます。もちろん、仕事の方は見積段階で 3 割程度の稼働を前提にスケジュールを組まないと、不幸な未来しか待っていないでしょう。

5.3 どうやって学ぶか

とにかくコードを書いて動かすことが大事です。しかしただサンプルを動かすだけでは面白くないしあまり意味がありません。**プロダクトコード**に導入してみるのが一番エキサイティングです。ブランチを切り、コードを書いてみて、プルリクエストまで出してみましょう。そのプルリクエストを見せ、チームメンバーに技術の紹介をしたり、より詳しい人がいるならレビューしてもらったりするとより効果的です。

5.4 まとめ

- 常に学び続けないと死ぬ
- 周りの人を上手く活用して学ぶ
- ひとつでいいのでものすごく掘り下げる
- アウトプットするとより良いインプットになって帰ってくる
- 業務は最高の学びの環境である

業務時間に学んだりプルリクしたりしたら怒られるような会社ならさっさと転職しましょう。会社は将来を保証してくれないけど技術は将来を保証してくれます。

第 6 章

プログラマの転職

Web プログラマ業界は非常に転職が盛んです。プログラマとして働くからには、いつか必ず転職を経験することになるでしょう。いつその時が来ても良いように、普段から転職を意識して仕事するべきです。

6.1 なぜ転職するのか

転職する目的は第一に**カネ**です。Web 業界は、SIer や他の製造業などと違い**作ったからカネになる**世界ではありません。サービスがヒットしないと、**給料が払えない**可能性すらあるのです。なので、年功序列のような自動で順調に給料がアップしていく仕組みは期待できません。社内での昇給はそれほど期待できないので、**給料を上げたければ転職する**のは一般的な考え方です。

インターネットはもはや全人類が頻繁に利用する重要なインフラなので、自社サービスがヒットしていないということは、ヒットしているサービスが他のどこかにあるということです。そのような会社は急成長のまっただ中にいて常に人材不足、そのため高給でプログラマを集めようとするのです。Web 業界そのものが、**Web プログラマ全体の労働力を、その時ヒットしているサービスに柔軟に分配する**生き物だと言っても間違ってないかもしれません。

第二の目的は**嫌になったから**です。Web 業界はいつでも簡単に転職できるので、嫌になったから、**他にやりたいことができたから**転職するということはよくあります。直接のきっかけとしてはこちらのほうが大きく、ストレスに対して給料が割に合わないと思ったら転職です。

6.2 なぜ転職しやすいか

基本的にプログラマは全体的に転職しやすいのですが、Web 業界は特に転職しやすい業界です。Web 業界は、スキルのコモディティ化が徹底的に行われており、自社の独自技術を嫌います。Web を構成する根本的な技術は **RFC** で公開され、OS やミドルウェアやプログラミング言語はほぼすべて**オープンソース**であり、あらゆる技術や知見はブログや勉強会を通じて会社の垣根なく共有されます。仕事の進め方も共通化されており、社内独自のワークフローは悪とみなされます。

このような文化により、Web プログラマはいつどんなタイミングでも、**他社に転職しても困らない**のです。流動性のない会社は常に世間から取り残され腐敗していくので、雇う側としても外からの新鮮な空気を入れたいという需要があります。このような理由から Web プログラマは転職しやすく、会社は社員に転職されないために、労働環境整備や福利厚生など様々な優遇措置を講じるので

す。そのため Web 業界では**労働者のほうが圧倒的に立場が強い**のです。

6.3 どうやって転職するのか

Web 業界で転職するためには、技術力を証明する必要があります。

大企業にいると、同僚が様々な会社に転職していくため、ほとんどの Web 企業とコネクションができます。転職したいと twitter でつぶやけば誰かに会社見学に誘われ、気に入ったら面接と続いていきます。採用する側からも友達案件は実力が測りやすく重宝されます。大企業でなくても、勉強会などでコネクションができていれば同じように転職できます。ハッカソンなどの実力が問われるイベントなら特に効果的でしょう。

人間関係が苦手な場合は圧倒的な技術力で殴りましょう。具体的にはライブラリを**オープンソース**で公開したり、**技術ブログ**を公開しましょう。喋れるなら勉強会に参加してセッションや LT で喋りましょう。人間関係が苦手でも、**相手が一方的にこちらを知っている**状況は作れます。それで十分です。

github でコードを公開していると、**転職エージェント**からメールが来ることもあります。今は転職を考えてなくても、将来のためにコネクションを持っておくと良いでしょう。ただの人売りじゃない**ちゃんと技術を考慮してくれる**エージェントを見つけておきましょう。

6.4 転職に対するイメージ

世の中の他の業界では、転職に対してかなりネガティブなイメージを持っているようですが、Web 業界では転職は当たり前であり、全くネガティブなイメージはありません。同僚の立場としても、残念ではあるけど喜ばしいことと捉えるでしょう。

Web 業界は非常に狭い業界で、交流も盛んなので勉強会などで頻繁に集まる機会があります。今生の別れのようなことは起こりません。**死にさえしなければ**。会社の繋がりより、一緒に働いた仲間という繋がりのほうが強固です。これを「狭い業界だしずっと友達だよ」略して「**狭業ズッ友**」と呼びます。そこから転じてズッ友[*1]といえば転職のことを言います。

Web 業界では労働者が非常に強いので、前職に遠慮せずに全く同じジャンルの**完全な競合他社**に転職することも少なくありません。昨日まで一緒に働いていた仲間が完全にライバルになるのです。他の業界だと、恩を仇で返す的な印象を持たれ、関係性は最悪に、下手したら裁判沙汰になるような案件ですが、こんな状況も Web 業界では悪いイメージは持たれません。むしろチームにとっては**どちらが沈んでも全員生き残れる**良い戦略と認識されます。

6.5 Web 業界以外の転職

いちおう Web プログラマ以外の業界も見てみましょう。

SIer は上位に行けば行くほど終身雇用です。給料も良いので出る理由が特に無いのだろうと思います。仕事内容ですら完全に独自文化に支配されていて、スキルのみならず使う語彙[*2]まで他社と

[*1] お菓子を持って挨拶回りしている人を見かけたら「あれ？ ズッ友ですか?」のような使い方をします。

[*2] なれる SE!も使われている単語で作者がどこ出身なのか判明するそうです。

は互換性が無かったりします。出世競争に敗れた人たちは、長く勤めたあと直系の子会社に出向したりするようです。

下位に関しては奴隷なので、転職活動をするための時間と体力を確保することがまず難しいでしょう。そもそもそういう会社に間違っても入らないことが大事です。退職してから転職活動しようにも、逃げることすらできないということもあるでしょう。しかし、**退職は労働者の権利**なので、必ず退職できます。余計な忠誠心や同情やサービス精神を発揮してはいけません。ドライに退職しましょう。退職するにも重要なのは先立つモノです。勤めている会社が**退職しづらそう**と感じたら、無駄遣いせず何ヶ月か生きられる貯蓄を作るべきです。

中間層はコンサルと技術をバランスよく両立でき、技術的にも優秀な人も多く、転職はしやすそうに見えます。というか Web 業界から見える SI 業界の人たちはこのあたりしかいません。ちなみに、なれる SE!に登場するスルガシステム、業界の中では中の上くらいだと思います。

6.6 まとめ

転職しやすいプログラマになるには

- 特に普遍的なスキルを身に付けろ
- コードを公開しろ
- 人脈を作るか一方的でいいから顔と名前を売れ

第 7 章

プログラマの生存戦略

プログラマとしてこの先生きのこるためには、きちんと戦略を立てて行動したほうがよいでしょう。この業界も他と変わらず生きのこるには**コネ**が重要です。きちんと周りと信頼関係が築けていれば、**なにかやばいこと**[*1]が起こっても救ってくれる人が一人はでてくるんじゃないでしょうか。生存戦略は当然、**給料を上げる** (=社内の評価を高める) ことにも繋がります。

7.1 仕事は評価されない

仕事をどれだけ完璧にこなしても、チームの数名にしか評価されません。直属の上司ですらたぶん評価してくれないでしょう[*2]。まぁ仕事を完遂することは最低条件なので。当然チーム外や他社の人間には全く評価されません。

評価されるためには、他者によい影響を与える存在にならなければなりません。例えば、ライブラリを作って共有をしたり、仕事の方法を改善したり、勉強会を開催したり、そのような**組織全体を改善する**働きは高く評価されます。もっとも、ただ行動するだけではダメで、自分がよい影響を与えていることを広く周知しないと、誰からも気づかれないので評価されません。きちんと評価されるためには**セルフプロデュース**が必要なのです。

7.2 尊敬するプログラマを見つけよう

ブログや勉強会で会った人の中に、この人凄いと思えるよう人がいるのではないでしょうか。そのような尊敬できるプログラマを見つけて師と仰ぎましょう。記事に書かれた技術の質問をし、抱えている技術的な問題を相談しましょう。できるなら転職して一緒に働くことを考えてもいいと思います。どれだけ凄い人でもすべてのジャンルの凄い知識を持っているわけではありません。まだ少しの知識しかなくても、その人の役に立ち信頼を勝ち取る可能性は十分あります。

世の中の有名なプログラマは技術力が高いだけでなく、だいたいセルフプロデュースに成功している人たちです。たった一人でも、その人に信頼され対等に技術の話ができるようになれば極太のパイプになります。

[*1] 会社が無くなるとか事故で記憶を失うとか逮捕されるとか

[*2] 給料が上がらないと嘆いている若者は、だいたいここを勘違いしているように見えます。

7.3 全部さらけ出そう

できること、できないこと、興味のある技術、書いたコード、技術的に悩んでいること、趣味嗜好、性癖、買ったエロ本、全部さらけ出しましょう。上に書かれたような打算計算も全部さらけ出しましょう。プログラマは基本的に子供なので社交辞令などは通用しません。きちんと信頼関係を結ぶには、素直になってすべてさらけ出すのが一番です。

7.4 クズになろう

人間の本質はクズですが、ほとんどの人はそれを上手く取り繕って生きていると思います。プログラマの世界ではクズのほうが優秀とされるので、構わず本心を出していきましょう。特に「**めんどくさい**」はプログラマが一番言い続けなければならない言葉です。めんどくさいことにはめんどくさいと言いましょう。クソコードにはクソコードと言いましょう。プログラマとして正常な判断能力を持っているアピールになります。もちろんそれらを改善するのを忘れずに。

7.5 正しく dis ろう

何かを dis ることは、技術的スタンスの強烈なアピールに繋がるので積極的に dis りましょう。ただし相手を間違えてはいけません。世界的に合意が取れるものを dis らないといけません。例えば **php** とか **php** とか **php** とかです。Lisp を dis ったらその瞬間にプログラマとしてのキャリアは終わります。

7.6 雑談をしよう

とにかく技術ネタの雑談をしましょう。チャットでも良いんですが、やはり対面で立ち話をするのは非常に重要です。社内をブラブラ歩いて色々な人と話をしましょう。変なガジェットで遊んでる人やアホな技術ブログを書いてバズってる人とかいますよね? そんな人は格好の餌食です。たくさん雑談し、色々なものに興味があることを周りにアピールしましょう。

まぁ本質的には雑談することが重要なのではなく、何にでも興味を持つことが重要なんですが。

7.7 お祭り事に参加しよう

メインの業務に関わる仕事は考えなしに引き受けてはいけませんが、メイン業務以外の頼まれごとには積極的に参加して顔や名前や恩を売りましょう。勉強会のスタッフ手伝ってくれとか、**生放送に出演してくれ**[*3]とか、そういうお願いは積極的に受けていきましょう。スタッフとして参加すると、参加者の人たちには会社の代表的なプログラマとして認識されます。社内に対しては、お祭り事に楽しんで参加できるアピールになり、巡り巡って**超会議で会社を代表してハッカソン**[*4]に出場することになったりします。何より気軽に仕事を頼めるイメージを作れるのがいいですね。気を

[*3] まぁ普通の会社では無いよね
[*4] まぁ普通の会社では絶対に無いよね

つけないといけないのは、業務外のお願いをされた時に使える時間を確保しておくことです。そのためには本業のほうを素早く完璧にこなさなければなりません。

7.8 自由人を気取ろう

自由人になりましょう。誰にも縛られず、何にも縛られず、「**あいつは自由にやらせると一番成果が出る**」と言われるようになりましょう。完全な自由 (=**信頼**) を手に入れるには、高い能力と成果が必要です。

最初は少しずつ自由人の片鱗を見せていくのです。誰かに指示されること無く、非効率があったら改善し、複雑なコードがあったらライブラリ化し、そういった実績を作っていきましょう。ただし、良かれと思ってやったことが間違った行動だと一気に信頼はなくなります。コードのボトルネックを調査するように、業務も正しい改善点と改善方法を見極めることが重要です。

7.9 まとめ

- 仕事をするだけじゃ評価されない
- セルフプロデュース大事
- 主張するの大事 沈黙は死
- 技術が本気で好きなら自然と理想的な振る舞いに近づく

第 8 章

プログラマの一日

プログラマは普段どんな生活を送っているのでしょうか。**ごく普通の会社**[*1]に勤める**ごく普通のプログラマ**[*2]のある一日を見てみましょう。

8.1　11:00 起床

朝 11 時に起床。裁量労働制なので時間に囚われた生活はしない。**目が覚めたときが朝**だ。

この日は予定があるのでいつもより早起き。普段の朝はさらに 2 時間くらい遅い。シャワーを浴びて 30 分ほどで出発。

8.2　11:50 映画を見る

通勤途中の日本橋で映画を見る。この日は**ガルパン劇場版**だ。

控えめに言って最高だった。ガルパンは人生。戦車道は乙女の嗜み。まだ見てない人は Blu-ray 買って見よう。

8.3　14:20 出社

映画を見終わって出社する。ぎりぎり社食に間に合う時間なので勤怠を押してすぐにリフレッシュルームに向かう。

それほど安くもない社食を使うメリットは、ランチを囲みながら部署の垣根なくコミュニケーションを取れる・・・なんてわけがなく、みんな**スマートフォンを弄りながらボッチメシ**だ。社食を使うメリットは気兼ねなくスマートフォンを弄り続けられるからだ。

食後は漫画週刊誌を読む。電子書籍の仕事してるしこれは仕事の一環だ。

8.4　16:00 朝会

朝 16 時に朝会が開かれる。裁量労働ではあるがこの時間には出社している必要がある。月に 1 回くらいの頻度で遅刻する。

[*1] 当社比

[*2] 当人比

朝会では昨日やったこと、今日やること、抱えている問題などをチーム全員で共有する。チームの中には 7:00 くらいに出社してる人もいて、その人は朝会が終わってしばらくすると帰る。

8.5 16:30〜 仕事

適度に休憩を挟みながら仕事をする。ディスプレイの一面にはニコ動を開き、**MMD 艦これ**タグを連続再生する。最近は仕事が忙しくてなかなか消化が進まない。全員常にヘッドフォン装備で、何かあったら基本的にすべて **Slack 経由**で話をする。Slack には仕事チャンネルの他に技術や趣味や雑談のチャンネルがあり、様々な会話が流れている。およそ 8 割が雑談チャンネルだ。適度に仕事を挟みながら Slack での雑談を楽しむ。

なお、回復したスタミナを無駄にしないように、きちんとソーシャルゲームもチェックしなければならないのは言うまでもない。

8.6 20:00 マッサージ

こんな仕事をしていると、いつも肩こり腰痛に悩まされる。疲れを取るために社内に設置されたマッサージ室に行く。市価よりだいぶ安いので重宝している。**ありとあらゆる福利厚生のうち最も価値のあるものはマッサージ**で間違いない。肩甲骨がガチガチに固まっているので今日は 50 分のコースを選ぶ。

8.7 21:30 夕食

Slack ではらへと発言して仲間を募りメシを食いに行く。深夜営業しない店はこの時間帯がラストオーダーなのでその前に。ぶらぶら東銀座を歩き適当な店に入る。店を選ぶポイントは、何が食いたいかではなく、**給料日までどれだけカネが残っているか**である。

8.8 22:00〜 本気の仕事

この時間帯になるとだんだん社内から人が減っていくが、**会社生活部**の面々はまだ仕事を続ける。なぜならこの時間帯は、静かで割り込みもなく、CI も空いているので仕事が一番捗るゴールデンタイムだからだ。リファクタリングやライブラリづくりなど、一人でできる仕事に没入し時間が経つのも忘れコードを書き続ける。MMD には付き物の**紳士向け動画**も、この時間帯なら気兼ねなく見ることができる。

終電は 24 時頃だが全力でスルーする。なぜならこの時間帯の電車には**ゾンビしか乗ってない**からだ。襲われることはないが、同じ空間にいるだけで感染するというもっと危険なやつだ。

さらに 1〜2 時間ほど働いた頃、脳の限界を感じるのでやっと仕事を終え帰る。当然終電はないので、家までの約 5km を一時間くらい**歩いて**帰る。普段なかなか運動などできないのでいい運動になる。歩いている最中は電子書籍を読んだりデレステをやったりする。短い人生、一瞬でも無駄にしてはいけないのだ。

8.9　27:00〜 自由時間

27 時前には家に到着する。ここから**完全な**自由時間なので趣味に費やす。録画したアニメやニコ動を見ながらゲームやプログラミング。会社ではできない**歌いながらプログラミング**も家でならできてしまう。

8.10　31:00 就寝

twitter を見ると起きたというつぶやきがあり、世の中が動き出すのを感じる。そろそろ明日があるので就寝。

第 9 章

プログラマの結婚

できません。[*1]
ご清聴ありがとうございました。

[*1] 周りで結婚したプログラマを見てみると、だいたい学生時代からの彼女と結婚している例が多いようです。学生時代に彼女がいなかった人、社会人になってから別れた人はその後ずっと独り身である傾向が非常に高くなっています。このようなキチガイな生き様なので基本的に新規に彼女を作ることや維持することは非常に難しいと言えます。最近の新卒を見てみると学生の段階で結婚している例もチラホラあり、これは非常によい戦略であると言えます。ただ、周りの既婚者の話を聞く限り、勉強会参加やプログラミング漬けの休日を過ごす自由はかなり制限されるらしいです。そもそも一般的に嫁という生き物は家族を最優先すべきだというとんでもない思想を持っているらしく、技術を突き詰めたいプログラマとは決定的に相容れない存在なのではないでしょうか。やはり技術を極める道を歩む上で嫁だの彼女だのは邪魔でしかなく、そのようなものを必要とするのは軟弱だと言わざるをえません。常に機械と対峙し機械と対話するプログラマは、やはり普通の人間の幸せなど望んではいけないのです。

第 10 章

プログラマの未来

個人単位では昔から言われる**プログラマ 35 歳定年説**、最近では業種全体が**人工知能に置き換えられる説**なんかも言われています。プログラマという職に未来はあるのでしょうか。

10.1　35 歳定年説

35 歳定年説は、いわゆる SI 業界において、35 歳前後に現場を離れマネジメント業に移行するという現象です。これは年功序列の世界で、35 歳の給料とプログラマという職の価値が吊り合わないから起こります。ここで注目するべきは、SI 業界においてプログラマはただの作業者であり、その価値はマネジメントより遥かに低く見られているということです。

これは完全に Web 業界には当てはまりません。Web 業界では技術が尊重され、プログラマは創造的な専門職とみなされています。最近では、マネジメントをしなくても高い給与を用意する会社も増えてきました。Web 系なら学び続けられる限りプログラマを続けられる環境が整ってきました。

昨今の急激な技術の発展により、SI 業界でも技術重視が進むかもしれません。そうなった場合 SI 業界でもプログラマ 35 歳定年説は無くなる可能性はあります。

10.2　人工知能に置き換えられる説

これも作業者としてのプログラマが置き換えられるのは正しい予想だと思います。しかし Web 業界のような創造的なプログラマが置き換えられるのはもっともっと先でしょう。人工知能自体を創りだすのも創造的なプログラマです。人工知能に対し仕事を指示することが、今後の一般レベルプログラマの仕事になるのではないでしょうか。

そもそも、作業者としてのプログラマが置き換えられるのは今に始まったことではありません。今でもプログラマは機械に対して仕事を指示し、機械が大量のコードを作っています。**フレームワーク**がそうだし、古くは**コンパイラ**もそう言えるのではないでしょうか。

今後、世の中のあらゆる仕事が自動化されるでしょう。その自動化を作るのはプログラマの仕事です。最終的にプログラマの仕事が人工知能に奪われるのはたぶん正しい予測です。が、その時にはすでに他のあらゆる仕事が人工知能やロボットに奪われていて、最終的に人類を滅亡させるロボットの起動コードをプログラマが実行するんだと思います。人類最後の職業はプログラマだと予想します。まぁ当分先の話でしょう。

10.3 まとめ

- 近年の技術重視の影響でプログラマ生涯現役の道が見えてきた
- 創造的でありつづけることができれば今後も仕事はある
- すべての仕事を機化し、最後に人類を滅亡させるロボットの起動ボタンを押したい

あとがき

プログラマという職は、最も自由で、最もエキサイティングで、最もクレイジーな仕事だと思います。世間と折り合いがつかない人にも広く開かれた業界です。中学行ってない奴、フル留年から中退コンボ決めた奴、心を病んでる奴、LGBT な奴、おおよそ他の業界では爪弾きにされるような奴らがたくさん、この業界では楽しそうに働いているのを見てきました。その事実はもっともっと誇っていいと思います。

今の仕事が辛い人、世間と上手くやっていけない人、Web 業界に来いよ!!!

さて今回の本、正直言ってあまり面白くないと思います。書いてて全然気分がノってきませんでした。なんだか雑多な文章を書き連ねただけのような感じがします。自分が満足できてない本でお金を取るのもなんだかなぁと思ったので、もともと gitbook で無料配信してるし、いっそ**本も無料にしちゃえ**ってことで無料にします[*1]。その代わり賽銭箱を用意して、入れたければ賽銭入れていいよという形にします。

というのも、実はちょっと前にドワンゴでエンジニア発の企画コンペがありまして、**パトロンプラットフォーム**の企画を出したところ採用され、半年くらいそちらの仕事をしていました[*2]。周りのエンジニアからはかなり好評で、手応えを感じていたんですが、法務周りの整理が付きこれからというところで、**kawango に完全否定されて**企画が終了しました。

所詮 kawango はコンテンツビジネスの人間でしかないと失望しました。新しい世界を目指す人間にとったら老害でしかなかったというのがよくわかりました。

その企画の骨子は、

> **コンテンツにカネを払う時代はすでに終わっていて、ヒトにカネを払う時代に移行しなければならない**

という思想です。この思想を実験するため、今回の本は**本自体ではカネを取らず、寄付を受け付ける**という形にしました。

今後のこと

このボツになった企画は、自分で引き取り**同人活動として実現**します。ヒトにカネを払うように移行した世界というのは、つまり行き着く先は**著作権の否定**です。パトロンプラットフォームで実現したいのは最終的に**著作権のない世界**[*3]です。ありとあらゆる著作物は自由にコピーされるべき

[*1] 給料増えて印刷代とか気にしなくてよくなったし

[*2] 法務周りの調整がほとんどでした

[*3] プログラマはそういうのほぼ気にしなくていい世界に生きてますよね

です。オープンソースはソフトウェアの進化スピードを驚異的に速めました。漫画や音楽も同様にコピーされ模倣され超スピードで効率的に進化するべきです。パクリだトレースだので無駄に時間を消費する暇は人類にはありません。

このサークルでやるか別のサークルでやるか決まっていませんが、次回以降は**反著作権サークル**として (も) 活動していきます。

プログラマの生き様

発行日	C89 2015.12.31
著　者	まさらっき
発行者	NP-complete
リポジトリ	https://github.com/np-complete/c89
Gitbook	http://masarakki.gitbooks.io/c89-the-way-of-programmer-life/
印　刷	Kinko's