IT人材の不足と育成における問題の分析とサイバータンクの解答2025

序論

1. 本稿の目的

本稿は、現代日本が直面するIT人材不足という構造的問題に対し、より高い解像度での分析を試みると同時に、教育現場および実務の双方から得られた知見をもとに、その本質に接近することを目的とする。
また、本稿は単なる問題提起に留まらず、サイバータンクが運営する教育事業の方針と実践を通じて、具体的な一つの回答を提示することを狙いとする。

2. 執筆者について

筆者は、自社開発、開発受注、プログラミングスクール事業を展開する「サイバータンク」代表の平野夏光である。(https://cybertank.jp)
サイバータンクはRuby on Railsをメインスタックとし、実践的教育も展開しており、その運営と日々のweb開発業務の中で得られた経験が、本論の出発点となる。
本稿における記述は、筆者個人の思索と現場感覚を色濃く反映したものであり、その意味で“学術論文”とは趣を異にする。
しかし、その不完全性を自覚した上で、むしろ「手触りのある構造的考察」を目指すために、本稿ではあえて主観的経験を起点とした論証方針を採用している。

3. 実験主義的立場と弁証法的接近

IT人材不足という問題は、現在のところ、あらゆる関係者が「確定的な解」を持ち合わせていない領域である。
そのため、本稿では一貫して実験主義的姿勢を貫く。すなわち、明確な普遍命題を打ち出すのではなく、局所的経験から導かれた仮説を提示し、反例や異論と向き合うことを前提とする。
こうした立場から、本稿は単なる知識の蓄積や外部データの羅列ではなく、今後も続くIT人材教育事業者の仮説と検証の反復を通じた“暫定的な解への接近モデル”の提示と補助を意図する。

4. 問題群の列挙

本論に先んじ序論の中でIT人材不足問題を7つの要因に分類し列挙する。

この着手点と挙げられる要因らそのものが、上に述べている「局所的経験から導かれた仮説を提示し、反例や異論と向き合うことを前提とする実験主義的姿勢」の現れであることに留意されたい。

本稿で取り扱う“IT人材不足”とは、単なる供給量の不足を指すのではない。
以下に示す7つの要因は、それぞれがこの問題を構成する独立の要因であり、同時に他の要因と密接に連動し合う複合的課題である：

1. 内的新規参入障壁
2. 外的新規参入障壁
3. 継続困難性
4. 離脱
5. 教育内容問題
6. 教育者スキル問題
7. 教育方法問題

本論では、これら7項目についてそれぞれ（ｉ）定義と構造分析、（ii）具体的な解決策の提示、という2層構造で展開する。
また、終章ではサイバータンクとしての具体的な取り組みを各項目に対応する形で記述し、本稿が単なる理論にとどまらず、現実的実践に根ざした応答であることを示す。

本論1：内的新規参入障壁

(ⅰ) 定義と分析

「内的新規参入障壁」とは、個人の内部に起因するIT分野への参入の困難性を指すものと定義する。
これは制度や社会的条件ではなく、本人の意志決定、認知的傾向、心理的抵抗、習慣や文化的背景に由来する「見えにくいハードル」である。

以下では、現場で観測される主な傾向を着手点とし、その具体化や分析を通して解決策への糸口とする。

現場で観測される主な傾向には、以下のようなものが挙げられる。

- 「自分にプログラミングは向いていない」という先入観による思考停止
- 「数学が苦手だったから無理」という教育履歴バイアス
- 「覚えることが多すぎて無理そう」という認知的過負荷感
- 「今から始めても遅い」という年齢的諦念
- 「今の仕事より安定しないかもしれない」という不安定性の過大評価

・構造的分析

この障壁の本質は、「認知のメタ構造」と「動機づけの不均衡」にある。
つまり、多くの人が自分の適性や努力可能性について体系的に再評価する機会を持たないまま、不十分な情報を元にした進路の決定ーITへの諦めーに至る。平たく言えば、自分のことも、ITのこともよく分からないまま断念しているケースが往々にして観測されるということだ。
この過程には、以下の3つの構造が見出せる：

1. 自己評価の非合理性
   → 主観的自己像が過去の失敗や社会的通念に固定されており、未来に対して開かれていない。

2. 情報の断片的理解
   → スキルの習得プロセスや業界実情を全体像として把握できず、誤解に基づく拒否感が形成される。

3. 内的葛藤と外的正当化
   →「変わりたい」という欲求と「リスクを取りたくない」という保守性が拮抗し、現状維持に自己正当化を与える。

特に、「情報の断片的理解」から来る誤解は多く見受けられ、以下のような例が挙げられる。

誤解例**1**:「プログラミングは数学的才能が必要」

確かにアルゴリズム分野や競技プログラミングでは数学的素養が求められるが、Web系開発においては高度な数式処理を日常的に要求される場面は極めて稀である。
むしろ、必要とされるのは以下のような文系的な力だ:

- 日本語と英語（およびコード）をまたぐ言語的変換能力
- 状況から問題点を切り分け、原因を絞り込む推理的態度
- 不明点をキーワードに変換し、検索によって発見する情報調査力

これらは、歴史学や哲学などの文系出身者が得意とする**「意味の理解と文脈の把握」**と極めて親和性が高い。
したがって、「数学が苦手だから無理」という認識は、プログラミングという行為の全体像を知らないことに起因する錯覚である。

つまり、教育履歴からくる主観的なバイアスと、プログラミングに対する難易度の過大評価や求められるスキルの質に対する無知が誤解を生み、結果的にプログラミングから遠ざかってしまうのだ。

誤解例**2**:「暗記量が膨大で覚えきれない」

プログラミングの学習ー特にweb系言語、フレームワークーは、語学のそれに近いことを非プログラマーは体感として持ち得ない。
文法（構文）や単語（関数名など）を丸暗記するのではなく、必要なときに必要な情報を検索し、使いこなす力が重視されることを彼らは知らないわけだ。。
むしろ、現代的な開発現場では「全部覚えている人間」はむしろ少数派であり、調べながら試行錯誤する姿勢こそが、能力として高く評価される。

- 実務で扱う技術は常に変化しており、“固定された正解”は存在しない
- チーム開発においては、むしろ「知らないときにどう振る舞うか」が能力評価の焦点になる
- 答えを覚えるのではなく、答えにたどり着くプロセスを身につけることが重視される

つまり、Web開発においては「覚える量が多そう」や「理系じゃないから無理だろう」というイメージは、“暗記型学習”という過去の教育体験の残像であり、現在の現場感覚とは大きく乖離した誤解なのである。

むしろ、文系出身者にこそ適性がある構造ということもできる。

事実、Web開発とは以下のような文系的タスクの連続である:

- 他者のコードの読解と解釈

- 設計思想の理解と表現の翻訳
- エラーメッセージの意味解釈と対応方針の調査

これらのプロセスにおいては、「国語的読解力」や「仮説思考」「比較・対照による判断力」が重視され、文系的な思考特性が大いに活かされる。
したがって、文系出身者がWeb系エンジニアとして戦えないという認識は、完全に誤っている。
むしろその逆であり、「問い方を学んできた人間」こそが、現代の開発現場において最も強い。

また、プログラミングの学習は、プログラミング言語の文法の勉強であり、それは語学を学ぶプロセスと似ている。

たとえば、Railsでデータベースに新しいデータを作成するには、次のような1行のコードを書く:

```
User.create(name: "Taro", age: 25)
```

この文を見たとき、「なるほど、`User`という種類のデータを、新しく作るんだな」という意味が読めれば、それでほぼ十分だ。
英語で言えば、

`User` = 主語
`.create` = 動詞（〜を作る）
`(name: ..., age: ...)` = 目的語（中身）

という風に、**語順と文法のルールに従って並べられた「言語」**である。

ここで重要なのは、この構文を丸暗記することではなく、構造と意味を理解することである。
一度「createは作る、括弧内は属性」と理解すれば、他のモデルでも同様に応用できる:

```
Article.create(title: "はじめてのRails", body: "楽しい！")
```

つまり、「新しいものを作る」という基本文法が頭に入っていれば、あとは語彙（モデル名やカラム名）を入れ替えるだけで自在に応用が可能となる。

この学習スタイルは、まさに語学学習のそれである。
英文法のルールを1つ覚えれば、後は単語を置き換えて話せるのと同様に、プログラミングも「型」と「文法」を覚えれば使える世界なのだ。

このように、教育履歴や学習経験によって形成された内的障壁は、**“ITエンジニア像”**に関する誤った想像の産物である場合が多い。
そのイメージを書き換えることができれば、多くの人が**“自分にも可能かもしれない”という初期

確信**を獲得できる。
それこそが、内的新規参入障壁を突破するための最初の突破口となる。

(ⅱ) 解決策案

サイバータンクでは、この内在的な障壁に対し、**「認知的再設計」と「小規模成功体験の早期獲得」**を主軸とするアプローチを取っている。

「内的新規参入障壁」の本質は、スキルや能力ではなく、本人の認知構造や誤ったイメージによって形成された心理的な拒否感にある。
したがって、これを解決するためには、技術的な指導や教材を与える以前に、
まず**「なぜ参入できないと思ってしまうのか」という構造を明示し、その誤解を解く情報を提供すること**が出発点となる。

多くの人にとって、「エンジニアの世界がどういうものか」「どんな思考で学んでいけばいいのか」がブラックボックス化しており、
その**“見えなさ”そのものが不安の原因**になっている。
よってサイバータンクでは、教育活動に先んじて、心的バリアを言語化・構造化して解きほぐす情報発信と初期設計を徹底している。

● 認知的再設計（＝ハードルの可視化と言語化）

- 学習者との初回面談において、「できないと思っている理由」自体の構造を対話的に明示化する
- 否定ではなく、“その思い込みがなぜ成立しているか”の探究から入り、参加者の“現在地”を可視化
- 自己効力感の再定義を通じ、「できるかもしれない」状態を意識的に設計する

● 心理的確信を高める情報提示

- 「数学ができなくてもWeb開発はできる」「文系的素養がむしろ武器になる」
- 「最初に学ぶのは“単語の意味”より“文の形”であり、暗記量は多くない」
- たとえば、以下のような文法的コードの具体例を提示する：

```
User.create(name: "Taro", age: 25)
```

このように、Web開発におけるコードは主語・動詞・目的語で構成された言語のようなものであり、
暗記ではなく構造の理解があれば応用が利く。
この語学的性質と、文系出身者の成功例などを用いて、“自分にもできそう”という感覚を早期に伝える。

● 小規模成功体験の早期獲得

- 初期段階で「簡単なWebアプリ」を完成させることを目標に設定し、“完成する感覚”を早期に植え付ける
- コードの意味理解よりも、「動いた」経験を通じた情動的な達成感を優先
- 初期フェーズにおける進捗の明確化と見える化を徹底し、迷子状態を最小限にする

● 習慣と文化の書き換え

- 「何かを学ぶこと」そのものに拒否反応を持つ学習者には、週1回の内省と対話セッションを設け、思考と習慣を丁寧に解凍
- できる・できないを結果ベースで捉えず、「どう考えたか」プロセスへの評価文化を導入する

以上のように、内的新規参入障壁は、教育コンテンツ以前の段階で**「心的環境の整備」を必要とする問題である。
したがって、カリキュラムや教材の良し悪しだけではこの障壁は突破できない。
必要なのは、「参加可能性そのものの再構築」であり、それをサイバータンクでは教育以前の“啓蒙設計”**として明確に位置づけている。

本論2：外的新規参入障壁

(ⅰ) 定義と分析

「外的新規参入障壁」とは、本人の内面や意思に依存せず、環境・制度・社会構造など外部的な要因によって生じる参入の困難性と定義した要因である。
本人が「学びたい」「やってみたい」と思っていたとしても、
資金・時間・地域・文化・制度・情報格差などの外的要因によって参入の機会が断たれているケースは少なくない。

以下では、現場で観測される主な傾向を着手点とし、その具体化や分析を通して解決策への糸口とする。

具体的に観測される外的障壁の類型には以下が挙げられる。

1. 経済的障壁
   – 学費や生活費の負担が大きく、スクール受講を選択できない
   – 学びたいが、無職期間を取る経済的余裕がない

2. 時間的障壁
   – フルタイム勤務や家庭の都合で学習時間を確保できない
   – 育児・介護など不可避の事情による時間制限

3. 地理的障壁
   – 地方や海外在住で、都市部の教育リソースにアクセスできない
   – 対面中心の教育機関に通うことが困難

4. 情報的障壁
   – どのスクールを選べばよいか分からず情報収集に時間がかかる
   – 自分に合う学習モデルを判断する材料が少ない

5. 文化的・言語的障壁
   – 教材や講義が日本語に偏りすぎていたり、外国人に不親切
   – エンジニア文化に触れてこなかった人が感じる無言の“排他性”

これらが例で挙げられる外的障壁の厄介さは、「本人の努力ではどうにもならないと錯覚されやすい」という点にある。
実際には、制度設計や提供側の工夫によって軽減可能な要素も多いが、
学習者の側からは**“あきらめるしかない理由”として受け入れられやすい**構造を持っている。

また、多くの外的障壁は複合的に絡み合っている。
たとえば、時間的制約を受ける人は、同時に経済的にも不利であることが多く、
情報格差は、そもそもその人がどこに住んでいるか（地理的制約）と連動している。

この意味で、外的新規参入障壁は、本人の資質に関係なく、格差を再生産しうる構造的問題として捉える必要がある。

(ⅱ) 解決策案

サイバータンクでは、こうした外的要因に対し、
「設計思想の透明化」「情報格差の軽減」「包摂的な文化設計」を軸に対応している。

● 情報格差と判断コストの軽減

- 学習モデルや成長ステップを具体的・視覚的に図解し、参入の判断材料を明確化
- 自分に合うかどうかを判断しやすくするための、**目的別ルート設計（例：副業向け／転職志望向け／趣味レベルなど）**を公開
- 成果物例・学習体験談などを積極公開し、透明性の高い情報提供を行う
- 本稿を始めとする、コピーライティングとは全く異質な高密度の情報発信を行う

● 包摂的な文化の構築

- 教育資料の表現を「前提知識なしの人」に向けて作成
- 技術・年齢・性別・職業背景にかかわらず、すべての学習者を対等に扱う文化的態度をチーム全体で共有
- エンジニアコミュニティにありがちな「分かってる前提の会話」や、「上から目線の回答」などを排し、学習者目線の説明文化を徹底

このように、外的新規参入障壁に対する戦略は、制度設計・教育設計・運用文化のすべてにまたがる総合的アプローチである。
重要なのは、「できない理由」を社会の中に配置して終わるのではなく、それを越える仕組みの設計と、越えられるという認識の共有である。
サイバータンクでは、どこに住んでいようと、どんな立場でも「参加可能性」が担保されていること自体が、参入の最初の勇気になると考えている。

本論３：継続困難性

(ｉ) 定義と分析

「継続困難性」とは、**IT**学習を開始したものの、途中で離脱・中断してしまう人が非常に多いという現象と定義する。
この問題は「やる気」や「根性」などで片づけられるべきものではなく、
構造的に発生しやすい条件が多数存在しており、教育設計上、最も慎重な介入が求められる領域である。

以下では、現場で観測される主な傾向を着手点とし、その具体化や分析を通して解決策への糸口とする。

観測される典型的な継続困難の原因は以下のように挙げられる。

1. 進捗の実感が得られない
   – コードを書いているが「何をしているのか分からない」
   – 理解できた感覚より、「なんとなく進んでいる」感覚の方が強い

2. 中だるみフェーズの出現
   – 初期のモチベーションが落ち、面白さより「作業感」が勝ってくる
   – 課題の“型”に慣れたことで、思考的刺激が減退する

3. エラーやトラブルによる挫折
   – 一つのエラーに数時間～数日詰まることで強烈な無力感に襲われる
   – 原因が分からないまま試行錯誤を続けることで、学習の意義が見えなくなる

4. 学習時間の優先度が下がる
   – 本業、副業、家庭の用事が優先され、学習が後回しになる
   –「後でやろう」が積み重なり、フェードアウトする

これらが例で挙げられる継続困難性の本質は、「やればできるが、やれない状態が持続する」という点にある。
これは意志の弱さの問題ではなく、

- 進捗の不可視性
- 正解の不在
- 学習の意味が実感できない構造

といった、学習デザイン側の不備によって“やりがい”が構築されないことが原因である。

IT学習は、基本的に一人称視点で孤独にコードと向き合う時間が長く、
それゆえ、「できるようになった」という感覚を適切に外から補強しない限り、
多くの人は**“実力がついていないように錯覚”して中断してしまう**。

(ⅱ) 解決策案

サイバータンクでは、継続困難性を人間工学的な視座から向き合うことで、
「進捗の可視化」「定期的な自己確認」「失敗の正常化」「社会的接続の確保」を軸にして学習の意味と成長実感を継続的に補強する設計を以下のように行っている。

● バディ制の導入による社会的補強

- サイバータンクでは、受講者同士を**2**人**1**組のバディとしてマッチングし、
  バグの調査や課題実装をペアで協力しながら進める学習スタイルを採用している

- この制度により、「一人で学び続ける孤独」が緩和され、社会的な接続が継続意欲に直結する

- バディとともに問題を乗り越える経験は、自己効力感（“自分にもできる”感覚）の増強に直結し、
  かつ「相手のためにやりきる」という他者志向の責任感も、自然な継続動機となる

● 文系的視点の活用による意味補強

- 進捗が停滞した際には、コードの振る舞いを文章で説明する課題や、
  「なぜこの構文を選んだのか」を言語化させるタスクを挿入

- これにより、「何が分からなかったのか」が可視化され、次の一歩を明確にできる

- 文系的素養を持つ受講者にとっては、“書くこと”が前進の手がかりになる

● 進捗の可視化と小単位区切り

- 学習カリキュラムを小さなステップに分割し、常に「次にやること」と「今やったこと」が明確に見える状態を維持

- 各ステップで「自分が書いたコードが何をしているか」を文章化・口頭で説明するタスクを挟み、理解の言語化＝定着の加速を図る

- 「できたこと」の記録をスプレッドシートやGitHub履歴で見える化し、目に見える蓄積感を演出

● 自己理解と内省の仕組み化

- 毎週の内省・自己報告の場を設け、「どこでつまづいたか」「何ができるようになったか」を定期的に振り返る習慣を定着

- 講師やスタッフとの非評価的な対話セッションを通じ、成長過程への確信を形成

- 技術スキルだけでなく、「思考・推論・調査」の習得プロセスにも目を向けるよう誘導

● エラーと詰まりの“正常化”

- 受講者に対し「詰まることは正常」「分からない時間が一番学んでいる時間」という思想を初期段階から繰り返し共有

- 講師側も「即答せず、調べる姿勢を肯定」することで、“知らないことを恐れない”風土を構築

- Slackやチャットでの質問も、エラーとその経緯を共有すること自体が他者学習につながるように設計

このように、継続困難性は「意志の問題」ではなく、「学習構造にどれだけの継続可能性を組み込んでいるかの問題」である。
モチベーションを信じるのではなく、“モチベーションが薄れても継続できる設計”を組むことが教育機関の責任である。
サイバータンクでは、学習者が“やめたくならない構造”をつくることそのものを教育工学と捉え、実装し続けている。

本論4：離脱

(ⅰ) 定義と分析

「離脱」とは、一定期間**IT**学習に取り組んだにもかかわらず、業界参入前に学習を中止してしまう現象と定義する。
これは「継続困難性」と似て非なるものであり、**“ある程度の学習ステージを超えた後の離脱”**に特徴がある。

特に以下のようなフェーズで発生しやすい：

- 初級カリキュラムを終え、ポートフォリオ開発に進むタイミング
- 転職活動や案件応募を意識し始めた時点
- チュートリアルではない“ゼロからの設計”に挑戦したとき

このフェーズにおける離脱は、**「何をすればいいか分からない」「不安が大きい」**という理由が多く、
学習の“次の段階”に進むための心理的・戦略的な橋渡しがないことが原因である。

よくある離脱理由は以下のように挙げられる。

1. ポートフォリオ設計での挫折
   – “何を作るべきか”が分からない
   – 完璧主義に陥り、制作が止まる
2. 実務や転職への不安
   –「本当にこのスキルで通用するのか？」という根源的不信
   – 業界情報の欠如による判断不能
3. 「学ぶこと」に満足してしまう
   – 小さな成功体験を積みすぎて、「本番に踏み出す恐怖」が膨らむ
   – 完成された世界（教材）から、未定義の世界（現実）へのジャンプが怖い

この種の離脱は、「学んだが、動き出せない」ことによって生じる自己不信の蓄積が原因である。
本人は十分に努力してきたのに、
「なぜか最後の一歩が踏み出せない」→「やっぱり自分には向いてないのかもしれない」→「静

かに離脱」
というルートを取る。

特に独学系・非同期型スクールでは、この段階での伴走設計がない場合、離脱率が極端に上昇する。

(ⅱ) 解決策案

サイバータンクでは、「学び終えたあとにどう動くか」という問題に対し、
「構造化されたポートフォリオ設計支援」と「**Tech**ブログの宿題化」で営業フェーズのサポート、
「本番への接続設計」「戦略的な小勝ちの積み重ね」で実践力の養成 に取り組むことで離脱を防いでいる。

● ポートフォリオの構造設計支援

- 学習内容から無理なく発展可能な題材を元に、“選ばなくてもよいポートフォリオ案”を複数提示

- 「何を作ればいいか分からない」状態を避け、制作に集中できる状態を先に用意

- テンプレートからの発展型や、過去の卒業生事例を共有し、ゼロからの発想に対する恐怖を軽減

● **Tech**ブログによる“理解の言語化”と“動き出し訓練”

- サイバータンクでは、学習内容やバグ調査の過程・実装の工夫などを週**1**ペースで**Tech**ブログとして発信することを推奨し、
  実質的にカリキュラムの一部として取り入れている。

- この取り組みの第一の意義は、「人に説明する」ことを前提に学ぶ構造の獲得であり、
  文章化を通じて理解の深まり・定着・振り返りが促進される。

- 第二の意義は、学習成果を「外部に向けて出力する」経験を早期から積ませることにある。
  受講中から世の中に対して発信することが習慣になれば、
  ポートフォリオ完成後の「就職・案件応募フェーズ」においても心理的な障壁が小さくなる。

- そして第三の意義が、**Tech**ブログそのものがポートフォリオと並ぶ「キャリアの武器」として機能する点である。
  学習過程・思考の深さ・実装力を記録したブログ記事は、エンジニア採用を行う企業やクライアントにとって、
  履歴書では伝わらない“思考の跡”や“姿勢”を伝える最も有効な手段の一つである。

- 実際、Techブログは面接時の能力評価材料として非常に有用であり、**「この人は自走できる」「伝える力がある」**という好印象を残す効果が報告されている。

このように、Techブログは学習補助ツールであるだけでなく、

「思考の可視化 × 実装の記録 × 情報発信習慣」の結晶として、武器にもなるアウトプットである。

サイバータンクでは、これをポートフォリオと同列の成果物として位置づけ、

「技術」と「思考力」の両面から自己表現できる学習者の育成を図っている。

● 小さな“勝ち”を意図的に設計

- 自作機能を一部だけでもSNSやSlack上に公開するステップを仕組みに組み込み、「人に見せる経験」を初期から繰り返す
- 技術的にも心理的にも「完成させる」感覚を持つことが、“動き出す力”そのものになる
- 完成ではなく、「到達した実績」でマイルストーンを区切り、“やれた感”を累積させる評価設計

● 本番接続の思想設計

- 「学びは準備ではなく実戦の一部」という思想を、初期段階から一貫して伝える
- 開発演習においても、“クライアントがいる想定”“実運用前提”での設計条件を設定し、架空とはいえ現実と地続きの緊張感を共有
- ポートフォリオ制作は、**“業務シミュレーション”であり、“終点”ではなく“起点”**であると位置づける

このように、離脱は単に「やめた」という表面的現象ではなく、
“動き出せなかったこと”に起因する自己否定の蓄積として理解する必要がある。
したがって、最も重要なのは、学習から社会接続への“架け橋”の存在であり、それが設計されているかどうかである。

サイバータンクでは、受講完了者が迷わずに営業フェーズなどの「次の一歩」を踏み出せるような心理設計と情報設計を備えることで、
この“静かな脱落”を防ぐための構造を実装している。

本論5：教育内容問題

(ⅰ) 定義と分析

「教育内容問題」とは、プログラミング教育において提供されるカリキュラムや教材が、現代の実務や学習者の目的に対して最適でない、または不適切であるという構造的な問題と定義する。

この問題は、「内容が古い」「説明が雑」といった表面的なものではなく、
教育の構造と思想そのものの設計ミスに由来している。
とりわけ、現代のスクールや教材において以下のような傾向が見られる：

・代表的な教育内容上の問題点

1. 調べる力が養われない構造
   – 教材が“説明済み・整備済み”であることを前提とするため、受講者は「検索力」「切り分け力」「推測力」を使わない
   – 実務で最も重要な能力が教育設計から排除されている
   – 結果として、教材の外に出た瞬間に**“誰にも助けられないと何もできない人材”が生まれてしまう**。離脱にもつながる。
2. 目的と内容の不一致
   – 就職・副業・教養など、目的が異なるのに内容が画一的
   – 目的を明示しないまま「とりあえずHTMLから」などの型通りの進行

3. スタックの現実性と教育の乖離
   – 実務で使われない技術や構成を延々と学ばされる
   – 実務上の前提（Git、DB設計、APIなど）が軽視されがち

4. 断片的なスキル習得
   – フロントエンドだけ、APIだけなど、分断された断片知識の寄せ集め
   – “つながらない学習”が「自分は全体像が分からない」という不安に直結

5. 教材に哲学がない
   – 何のためにその技術を学ぶのか、背景や文脈が語られない
   – 単なる手順書の模倣で終わり、応用の想像力が育たない

6. 非同期教材の陳腐化
   – YouTubeや教材サイトにある動画コンテンツが数年前の技術スタックで止まっており、現代の構成と噛み合わない
   – 学習者が「動かないコード」「古い情報」に混乱し、本質以外の部分で躓く事態が頻出

このような問題群は、教育の「効率化・自動化・大量配信」志向が、学習の本質からずれてしまっていることに起因する。
つまり、「なるべく迷わせず、正解にたどり着かせよう」という親切設計が、
“調べることを学ばないまま完成した気になる”という危険な状態を生んでいる。

また、非同期型・動画教材においては「作りきったら終わり」であり、
環境変化や技術更新に対応する設計思想がないことも深刻である。
こうした教材は、時間が経つほど**“逆に学習コストを増やす”情報資産**となってしまう。

さらに我々が最も問題視するのは、「調べる力が養われない問題」についてである。

事実、非同期型、動画教材を終了した初学者が質問掲示板であまりに杜撰な質問文で疑問を投稿しているケースは珍しくない。

スクールでの学習期間でプログラムの全てを教えることは現実的に不可能であるのに加え、現場によっても採用している技術領域は千差万別である中で、調べる力を十分に養わないスクールが多いことはIT人材の供給不足のかなり大きな要因であると考察している。

(ⅱ) 解決策案

サイバータンクでは、教育内容問題に対し、
「実務構造に基づくカリキュラム設計」「全体性の明示」「技術哲学の共有」「目的別ルート」「調査力育成重視」を軸に設計している。

● 調べる力の訓練を中核に置く

- 講師が即答せず、「どう調べたか？」「どう推測したか？」を確認する設計
- エラーを見たとき、その意味を仮説化し、検索して検証する思考プロセスを反復訓練
- 「調べて正解に辿り着く力こそがITの実力である」という思想を初期から強調し、**“教えないことも教育”**と捉えている

● 言語化の訓練による“現場力”の育成

- サイバータンクでは、単にコードを理解・再現するだけでなく、それを他者に説明する力を育てることを重視している
- 特に、現場では以下のような高度な言語化スキルが求められる:
  - 「どの画面で、何をしたら、どうなったか？」という状況の分解と再構成
  - 「これをやるとこうなるのか？」「これが原因ではないか？」というクローズド質問・仮説提示
  - 「調べたが分からなかった」ことの調査ログと経過説明
- これらは単なる報告ではなく、“人に伝えることで自分も理解する”双方向の構造を持っており、
  コードの正確性と同等、あるいはそれ以上に現場で重要視される
- そのため、日常的なSlackのやりとりや講師への質問でも、
  「結論から述べる」「文脈を明示する」「質問を構造化する」などの言語運用の訓練がカリキュラムに内包されている
- この取り組みにより、実務現場で即座に信頼を得られる“説明力・報告力・整理力”が醸成される
- バディ制と組み合わせることで、言語化訓練を常態化する。

● 実務構造をベースにした教育設計

- RailsにおけるMVC・DB設計・API連携・Git・テスト・デプロイなど、現実の業務フローを模倣する形で一貫学習
- 「習ったが使えない」ではなく、「学びがそのまま現場対応になる」構成を意識

● 全体性の先出しと繰り返し設計

- 最初にWeb開発の全体構成（フロント～DB～本番環境）を抽象レベルで提示
- 各ステップで常に「これは全体の中で何か？」を意識づけ、“どこにいるか分からない”状態を防ぐ

● 技術に対する哲学の共有

- 技術選定や設計思想の背景（例：Railsはなぜ存在するか？MVCとは何か？）を説明
- 「コードとは何を表現するか」「なぜその書き方を選ぶのか」を問うことで、**“自分の頭で考える技術習慣”**を育てる

● 目的別の進路ルート設計

- 「転職したい人」「副業したい人」「教養で学びたい人」それぞれに学ぶべき順序と範囲を明示
- 汎用型のコースを押しつけず、目的と内容が接続している学びを提供

このように、教育内容の問題とは、「教えることの選定」ではなく、
「なぜそれを、どんな姿勢で、どう習得させるか」の構造設計である。

サイバータンクでは、時代遅れにならない設計思想・思考の汎用性・学習者の自走力を教育内容の中核に据え、
単なる知識の伝達を超えて、「戦える人材」の育成を目的としたカリキュラム構築を実践している。

本論6：教育者スキル問題

(ⅰ) 定義と分析

「教育者スキル問題」とは、プログラミングスクールや学習サービスにおいて、教える側のスキルが不十分であることに起因する学習の質的低下・成長阻害と定義する。

単なる「知識不足」の問題ではなく、教育のプロとしての“構造的スキル”が欠如していることに本質がある。

・よくある誤解とその本質

プログラミング教育の現場ではしばしば、
「知っていることをそのまま話せば、それで教えたことになる」という誤解が、質問掲示板、SNS、書籍、動画教材等さまざまな場所で見受けられる。
しかし、知識の羅列と、相手の理解を成立させることは全くの別物である。

教育とは、以下のような要素から成る複雑な技術体系である：

1. 説明：どこが分かりにくいかを把握し、言葉にする

2. 再構成：理解度に応じて順序や表現を変える

3. 例示：具体的なイメージを与える

4. 質問：相手の理解度を測り、対話で補う

つまり、教育とは**「理解の設計」**であり、単なる情報伝達ではない。

・代表的な教育者側の問題点

- 技術力偏重・教育力不足
  – 実務経験はあっても、教えるための分解力・構造化力が欠如している

- 理解していない人への共感欠如
  – 初学者の苦しみが理解されず、無意識に「なんで分からないの？」という圧力が発生

- 言語化力の欠如
  – 感覚的説明・決まり文句で済ませてしまい、本質を言葉で解き明かせない

- 思考プロセスの非開示
  – 問題解決の手順や検索の意図が共有されず、答えだけが提示される

- 人格依存型教育（属人化）
  – 一部の“熱血講師”に依存し、組織としての再現性がない

この問題の根底には、「教えられる人」と「教える技術を持った人」はまったく別の存在であるという理解の欠如がある。
教育者には次の二重スキルが必要とされる：

- 技術的理解

- 教育技術（説明、再構成、例示、質問への反応）

現場では、「元エンジニアだから教えられるだろう」という安直な登用が多く、
その結果、学習者が**「分からないことを分からないまま」**放置される状態が頻発している。

(ⅱ) 解決策案

サイバータンクではこの問題に対し、
「合目的的な講義設計」「思考過程の可視化訓練」「言語化技術の反復」「講師内フィードバック文化」「教育チームとしての運用」
という5軸で、教育者のスキルを属人的でない教育技術として体系化している。

● 「意味論的理解」に基づく合目的性の明示と教育設計

教育における「理解」には、**統語論的理解（syntactic understanding）と意味論的理解（semantic understanding）の二層が存在する。
統語論的理解とは、情報の表面的構造――すなわち記号や構文の正しさに関する理解であり、例えるならば「文法的に正しい英文を作る」ような能力である。
これは形式的な正しさに特化したスキルである一方で、“それが何を意味するのか”や“どのような状況で用いられるのか”**といった背景的・目的的文脈までは捉えていない。

一方で意味論的理解は、対象の機能・目的・利用場面に対する理解を伴う。
つまり、「それが何のためにあるのか」「どのような文脈で使うべきものか」を捉える力である。
プログラミング教育においては、この意味論的理解こそが重要であると考える。なぜなら、技術は合目的的（**teleological**）に用いられる実践的手段であり、
単に文法を覚えることは「使える」こととは一致しないからだ。

サイバータンクではこの認識に立ち、受講者の意味論的理解を成立させるために、全ての講義において**「合目的性」を起点とする5ステップ構成**を導入している。

- 概要：この技術が何を扱うか
- 使用場面：どのような課題や状況で必要とされるのか
- 関連技術：どの技術群と連関し、どう補完し合うか
- できること：この技術によって何が可能になるか（＝成果のビジョン）
- 文法・構文：それを使うための最低限の技術的手段

この構造は、「使い方を知る」ための骨格であり、受講者に**「ああ、これは○○のときに使えるな」という合目的的な気づき**を与える。
これにより、学習は単なる知識の記憶ではなく、実践への応用可能性を含んだ深い理解へと変化する。

さらに、講師に対してもこの構造に沿った講義設計・準備の義務化を行い、
教育マネジメントチームによるレビュー・フィードバック体制を整備している。これにより、属人的な教育から脱却し、理解可能性の再現性を持つ教育体制を確立している。

教育とは、情報を正しく伝えるだけでなく、“それがなぜ重要か”を共に発見するプロセスである。
合目的性に基づいた意味論的理解の促進は、受講者の主体性と学習の持続性を支える根幹であり、
サイバータンクの教育設計思想の中核をなしている。

● 思考過程の可視化と開示(思考過程の可視化訓練）

- 講師は「仮説を立てる→キーワードを検索する→検証する」という思考の裏側を口に出して共有
- これにより、受講者は「解決策」ではなく**「問題解決の方法論」**を学ぶことができる
- エラー対応や設計判断などの判断根拠まで開示する姿勢が徹底されている

● 言語化能力の訓練と標準化(言語化技術の反復)

- 講師・メンター間で説明の語彙や構造を標準化・共有し、「分からないを解きほぐす技術」をチームで磨き合う
- 例え話、図解、構文の比較などを活用し、多面的に伝える訓練を反復
- 講師自身の言語化力を学習者にモデルとして提示することで、学習者の言語能力育成にもつなげている

● 講師間フィードバックと教育の継続改善(講師内フィードバック文化)

- 指導の場面を振り返るフィードバックの文化が定着
- 日々の指導記録やケース共有を通じ、教育技術自体が学習・改善される対象として扱われている
- 「指導スキルにもPDCAを回す」という方針が、教育者の成長を支えている

● チームとしての教育体制(教育チームとしての運用)

- カリスマ講師に依存せず、講師チーム全体で学習者を支える設計
- 進捗や理解度の共有により、一貫性のある対応・リレー形式の指導が可能
- 教える内容・順序・語彙のズレが少なく、教育品質の再現性と信頼性が確保されている

教育者スキルの問題とは、“伝えたか”ではなく、“伝わったか”を基準とする教育観の有無に起因する。
サイバータンクでは、「分かりやすさ」を偶然や個人技に委ねず、
再現可能な教育技術として体系化し続けることに重きを置き、仕組み化して取り組んでいる。

本論7：教育方法問題

(ⅰ) 定義と分析

「教育方法問題」とは、教育コンテンツではなく、その“届け方”＝教育手法に内在する構造的欠陥や非効率によって、学習成果が阻害されている状態と定義する。

教材が良くても、教え方・進め方・学びのフレームが適切でなければ、学習者のポテンシャルは引き出されず、むしろ逆効果になり得る。

現代的教育現場における典型的な方法上の問題は以下のように挙げられる。

1. 一方通行の講義・動画中心
   – 「見るだけ」「聞くだけ」で受け身になり、能動的な学びが生まれない
   – 疑問や理解度の確認が介在せず、学習効果が不透明なまま進行
2. “分かる”と“できる”の混同
   – 解説を見て「分かった気になる」が、実際に手を動かすと手が止まる
   – アウトプットせずに進行し、定着しない
3. 進捗・理解の“放置”
   – 初学者のつまずきや進捗の停滞を拾えず、黙って離脱する構造がある
   – 形式的なチェックインはあっても、実態把握にはつながっていない
4. 学習フレームの欠如
   – 「何をどこまで学べば何ができるようになるか」のルート設計がなく、受講者は“霧の中を歩いている”状態になる
   – ゴールも地図もなく、とりあえず次のレッスンへ進んでしまう

(ⅱ) 解決策案

サイバータンクでは、教育方法問題に対し、**「アウトプット先行型教育」「進捗の可視化と伴走」「学習導線の明示」「対話・共有ベースの学習」**を中核に据え、“教える”ではなく“引き出す”ための教育方法を設計している。

● アウトプット先行型のカリキュラム設計

- 解説を聞いてから実践するのではなく、まず手を動かし、自力で試す構造を先に設計
- 受講初期からWebアプリを「作る前提」で学習し、技術知識を“手段”として捉える学習姿勢を定着させる
- 解説は実践の補助として後置され、「分からなかったことの補完」として使われる構造

● 進捗と思考の“見える化”**(=**手順書の作成**)**とフィードバック**(=Tech**ブログの投稿**)**

- 各課題で「コードを書くだけ」ではなく、「どう考えたか」「なぜそう書いたか」をSlackやシートで言語化
- 講師はそれを確認し、単なる動作確認だけでなく、“思考の精度”に対するフィードバックを行う
- また、サイバータンクでは**「学習した内容を自分の言葉で手順書にまとめる」という課題を定期的に課しており、 – 問題設定 – 思考の経路 – 試行錯誤の記録 – 成功と失敗の分岐点 – 汎用的な解法パターン化 といった内容を文書化・構造化する訓練**が含まれている
- この「手順書作成」は、自己の思考プロセスを客観的に見つめ直す内省の場であると同時に、 – チーム開発でのナレッジ共有 – 引き継ぎ文書の作成 – 社外公開ドキュメントの執筆 といった実務に直結する言語化力のトレーニングにもなっている

- 「何ができたか」ではなく、「どう考えて進めたか」に焦点を当て、成長の本質を可視化する教育方法を実現している
- Techブログに学習記録や思考のプロセス等を記録しておくことで、アウトプットをより生産的にする。

● 学習導線と到達地点の明示

- 「今は全体の中のどこにいるのか？」「この単元が何に繋がるのか？」を随所で提示
- マイルストーン設計・ゴールの共有・振り返りタイミングを組み込むことで、“学びの地図”を持った状態で進行可能にする
- 教材単位ではなく、学習体験全体としての構造を設計

● 対話・共有ベースの学習進行

- Slackを活用したバディ制のペア学習により、常に「誰かに説明しながら理解する」状態を生む
- 質問・仮説・相談・発見を日常的に共有する環境が、“学びの定着と発展”のサイクルを自然発生的に促す
- 対話は理解確認の道具ではなく、理解の深化と拡張そのものであるという思想が前提

教育方法問題とは、「教える中身」ではなく、「どう学ばせるか」の設計思想の問題である。
サイバータンクでは、一方通行の授業を脱し、対話とアウトプットに基づいた能動的学習の場を構築することで、
学習者自身の自走力・論理構築力・実装耐性を引き出す教育方法を実践している。

結論

本論文では、日本におけるIT人材不足という社会的課題を、単なる統計的問題や技術的スキルの欠如としてではなく、
「構造と思想の問題」として再定義し、その複合的な要因を7つの視点から分析した。

1. 教育以前の壁:「始めること」すら困難な現実

人材不足が叫ばれる一方で、学び始めようとする人たちの多くが、心理的・経済的・制度的な壁に阻まれて、参入すら果たせない現実がある。
これは単に「やる気がない」わけでも、「才能がない」わけでもない。
多くは、間違った自己評価・過剰な情報格差・無策な制度設計によって、「無理だ」と思わされている状態である。

つまり、IT人材不足は、“教育が届いていない”のではなく、「教育にたどり着けない設計」になっているのだ。

2. 教育中の構造欠陥:「学んでいるのに育たない」現象

仮に何らかのきっかけで学び始めたとしても、今度は**「続かない」「動き出せない」「自信が持てない」**というフェーズにぶつかる。
これは、学習者の努力不足ではない。
教育内容が断片的すぎる／方法が古い／指導者が“理解”を設計できていないなど、
教育の内側に原因があるケースがほとんどである。

とりわけ深刻なのは、教育の場が「情報を与える場」になってしまい、
「行動を設計する場」「自走力を育てる場」として機能していないことである。

3. 教育とは“構造を設計する行為”である

この論文の全章を貫く根底的な思想は、
教育とは「教えること」ではなく、「理解が起こるように設計すること」であるという一点である。

そのために必要なのは、以下のような一貫した教育観である:

- 人はどう誤解し、どう理解し、どう行動するのかという人間の構造に対する洞察
- 技術ではなく、意味と背景と思想をセットで届ける教育設計
- “分かりやすさ”を偶然ではなく、技術として再現可能にする講師側の訓練
- 教えるより前に、学び始められるようにする戦略（心理・経済・制度面の設計）

本稿が提案するのは、単なる方法論や手法ではない。
“教育とは何か”の再定義である。

4. 最も重要なことは「調査力」である。

現役プログラマーとして筆者が常に思うことでもあり、過去の生徒にも常々伝え、サイバータンクのスクール事業でも軸にしているのは、生徒の調査力を高めることである。
「スクールを卒業すれば現場で通用する」というのは、はっきり言ってほとんど不可能である。スクールに在籍する期間が仮に1年だとして、どれだけ学んでもその言語やフレームワーク、関連する技術領域全てを体系的に網羅することは現実的ではない。
事実、実装現場においては使用技術や設計、仕様まで千差万別であり、どんなに優秀なプログラマーでも案件理解と現状のアセットの過不足の把握とキャッチアップには少なからず時間を要する。
そんな中、初学者にスクールが教えるべきなのは、知識はもちろん、「調査力」とその重要性であると考える。
基本構文や言語の思想、フレームワークの運用と応用など、現役エンジニアがいうところの「基礎的知識」を教え、それ以外は調査力の養成に集中した教育を施すべきだ。調査力がついてて、

基礎的な知識、コミュニケーション方法についてを知っていれば3ヶ月ほどでどの現場でもまともな戦力になることができるからだ。

5. サイバータンクの挑戦：教育の思想を実装にまで落とし込む

筆者自身の諸学者時代の経験や教育経験の中で見えてきたのは、「良い先生に出会う」でも「良い教材に出会う」でもなく、
「良い構造に巻き込まれること」こそが、人の成長を引き出す決定的要因だということである。

サイバータンクは、この構造を人力で支え、体系化し、反復可能にしようとする試みである。
我々が設計しているのは、Railsでもなく、教材でもなく、「人間が変わる一連の動線」そのものである。

6. 本論文の意義：再現される思想として

本稿の目的は、サイバータンクの自慢や実績を伝えることではない。
むしろその逆であり、“再現される思想”として、教育の構造を誰でも設計できるようにするための骨組みを公開することにある。
それが、私たちが持てる唯一の武器であり、日本全体の人材構造に波紋を起こす可能性のある知見だと信じている。

締めくくりに

IT人材不足は、ただの“人が足りない”問題ではない。
「人を人として育てる構造が足りない」という問題である。
そしてその答えは、資金や派手なシステムではなく、思想・構造・設計の積み重ねにしかない。

この論文が、「教育とは何か」「教えるとは何か」「どう人が育つのか」を問い直す一助となれば幸いである。
そして、どこかの誰かが、また別の形でこの思想を継ぎ、進化させてくれることを心から願う。

サイバータンクの事業取組①：内的新規参入障壁への対応

---

1. 対応する課題の再定義

内的新規参入障壁とは、「自分には無理だ」「向いていない」という主観的な思い込みや先入観が原因で、学びを始める前に挫折してしまう心理的障壁である。

この障壁は、実際のスキルや学習環境とは無関係に、
本人の中で完結してしまうため、**極めて発見されづらく、かつ非常に強力な“自己否定の壁”**となる。

2. サイバータンクの方針：教育以前に「認知設計」を行う

サイバータンクでは、内的新規参入障壁を教育の前段階における「構造的な啓蒙不足」と捉えている。
そのため、指導以前に行うべきは、“なぜ自分は無理だと思っているのか”を解体し直す認知設計である。

---

3. 実際の取り組み内容

● 初回面談で「不安」の構造を言語化する

- 学習者との面談では、いきなりカリキュラム説明に入らず、
  「なぜ今までプログラミングに手を出してこなかったのか」を深掘りすることから始める

- 「数学が苦手」「覚えるのが大変そう」「年齢的に遅いかも」などの主張に対し、
  それが本当に妥当な理由かどうかを、共に検証していく対話を行う

- このプロセス自体が、「無根拠な思い込み」を可視化し、
  “自分にもできるかもしれない”という初期確信を生む認知的再設計となる

● 初学者に特化した安心設計の導入

- コマンドラインや用語の初出には「必ず比喩や図解を用いる」「質問歓迎の空気を作る」など、心理的ハードルを下げる表現と空気設計を徹底

- チャットでのやりとりにおいても、“質問の仕方が分からない人”に向けたテンプレや例文を提供

- 「今さら聞けない」が発生しない文化を組織内に浸透させている

● 成功体験の初期配置

- 最初の1週間で「動くWebアプリ」を作るミニ演習を実施

- `User.create(name: \"Taro\")` のような簡潔な構文を通じて、“自分がシステムに命令できた”という感覚の獲得を重視

- この“最初の成功”こそが、内的障壁を押し返す最強の根拠となる

●「文系でもできる」の証明としてのコンテンツ設計

- 説明構成・教材・演習タスクのすべてにおいて、「理系的演繹」よりも「文脈理解・意味付け・言語化」に重きを置いている
- 実際の成功事例として、哲学科・文学部出身者がRailsエンジニアとして独立していったケースを教材・相談時に明示
- 「文系でもOK」という主張を、具体と実例で納得させる情報設計を行っている

4. 成果と展望

このようにサイバータンクでは、「学びの入口に立てない人」を見捨てない設計を徹底している。
実際、当スクールの受講者のうち約6割が「文系・未経験・ITに苦手意識あり」からスタートしており、
その多くが2〜3ヶ月以内に自信と意欲を取り戻し、自走モードに入っている。

今後は、この認知設計のパートをオンライン化・一般化し、教育以前の“思想普及”モデルとして提供する構想も検討している。

サイバータンクの事業取組②：外的新規参入障壁への対応

---

1. 対応する課題の再定義

外的新規参入障壁とは、本人の学習意欲があっても、社会的・制度的・物理的な条件により**IT**学習の機会が奪われている状態を指す。
たとえば以下のような構造が代表的である：

- 学費や生活費の負担により受講できない（経済的障壁）
- 時間や家庭事情でまとまった学習時間を確保できない（時間的障壁）
- 地方在住で学習資源や教育コミュニティに接続できない（地理的障壁）
- 「何から始めればいいか分からない」という状態のまま情報収集が停止する（情報的障壁）

これらは、本人の資質や意志の問題ではない。
構造の問題である以上、構造によって解決しなければならない。

2. サイバータンクの方針：情報格差と文化的障壁への集中攻撃

サイバータンクは、外的障壁の中でも特に**「情報格差」および「文化的排他性」**に対して戦略的に対処している。
理由は明快で、これらは費用や制度を変えずとも“設計”によって克服できる領域だからである。

3. 実際の取り組み内容

● 情報構造の分解と再編集

- 初学者が「どこから手をつければよいか分からない」という事態を避けるために、**複数のルートマップ（転職向け／副業向け／教養向け）**を整備
- カリキュラムや教材単位ではなく、「目標から逆算した技術地図」を共有することで、全体像を持った学びを可能にしている

● 判断を支える“ナビゲーション型”の導入相談

- 入学前のオンライン相談では、「どんな目的を持っているか」をヒアリングし、それに合わせたカリキュラムや到達目標の選定を行う
- 無理に申し込ませるのではなく、「今は始めない方がいい」という助言を行うことすらある
- この文化により、「自分に合うスクールを選べた」という主体的選択の感覚を持ったまま学習に入れる

● 言葉と文化のバリアフリー設計

- 技術文書にありがちな“暗黙知”や“慣れた人向け表現”を徹底的に排除
- Slack上のやりとりでも、「分かっていない前提」で語る文化が根付いており、“質問できる空気”そのものが設計対象とされている
- たとえば「ググれカス」のような文化は徹底排除し、「どうググればいいか」を教える文化を育てている

● スタッフの態度教育と評価基準の変革

- 「技術ができる」よりも「初学者に届く説明ができる」ことを重視した人材評価を実施
- 講師・チューターの採用時にも、「わかりやすく話す訓練をしてきたか？」を重視し、**“文化の入り口を守れる人材”**を選抜

● オンライン同期型授業の展開

- 早朝の朝活授業などで、日常の都合のいい時間でしっかり講師と時間を合わせて授業するスタイルを採用。

4. 成果と展望

今後は、「非同期型のエントリーコンテンツ」や「地方出身者同士のピアグループ」の実験を通じて、
さらに**“届く範囲の拡張”**に取り組んでいく予定である。

サイバータンクの事業取組③：継続困難性への対応

---

1. 対応する課題の再定義

継続困難性とは、学び始めた人が一定期間内に学習を中断・停止してしまう現象を指す。
これはモチベーションや努力の問題ではなく、教育構造の設計ミスが原因で“やめざるを得ない”状態に追い込まれることが多い。

主な発生要因は以下の通り：

- 「何をしているか分からない」まま進んでしまい、理解の実感が得られない
- モチベーションに依存した進行で、中だるみによる失速が発生する
- エラーや詰まりに対して、“自分だけできない”という無力感に陥る
- 学習優先度が生活の中で下がり、自然とフェードアウトしていく

これらはすべて、**「構造で支えれば防げる継続断念」**である。

2. サイバータンクの方針：感情と構造の両輪で継続を設計する

サイバータンクでは、「やめたくなる理由」を感情と構造の両面から設計し直し、
継続に“意志”を必要としない教育構造の確立を目指している。

3. 実際の取り組み内容

● バディ制による社会的責任感と孤独の排除

- 全受講者に**ペアを割り当てる“バディ制”**を採用し、バグ調査や課題実装を協力して進行
- これにより、自己効力感（“自分にもできる”という感覚）が高まり、
同時に「相手のために続ける」という社会的責任感による継続動機も発生

- Slack上での「進捗の共有・相互助言」も活発に起こり、**学習の場が“孤独にならない設計”**となっている

● 進捗の可視化とフィードバックの仕組み化

- 全課題には「何ができたか」「どう考えたか」を言語化するフォームやSlack報告を導入
- 受講者の進捗とつまずきが可視化され、講師が行動の変化ではなく“思考の痕跡”を見て伴走できる設計になっている

● 思考の可視化ツール：手順書の作成

- サイバータンクでは、学習した内容やバグ解決の流れを**「自分の言葉で手順書にする」**という課題を必須化
- これにより「再現性のある思考法」が定着し、“できたこと”が“他人に教えられる力”に進化する
- 実務における報告・ドキュメント作成・ナレッジ共有の基礎力も同時に養われる

● 小さな“勝ち”を設計に組み込む

- 学習初期から、自作コードをSlack・SNS・バディに見せる構造を設計
- 「1人で完成させる」が目的ではなく、「誰かに届いた」「反応が返ってきた」という社会的報酬を学習内に織り込む
- 成果物だけでなく、仮説・試行錯誤・やり直しプロセスも共有可能とし、“完成主義”ではなく“発信主義”の文化を浸透させている

4. 成果と展望

サイバータンクでは、2024年以降の受講者継続率が**90%**以上で安定しており、
離脱者の多くは、やむを得ないライフイベントや進路変更に起因する。

今後はさらに、学習データの自動可視化ツールの開発や、バディ同士の関係構築支援機能の導入を通じて、
「継続できる設計」から「勝手に続けたくなる設計」へと進化させていく構想である。

サイバータンクの事業取組④：離脱への対応

---

1. 対応する課題の再定義

「離脱」とは、一定期間学習を続けていたにもかかわらず、就職・案件受注・自己実現などの出口に辿り着く前に学習活動が停止してしまう現象である。
これは「継続困難性」とは異なり、すでに一定の学習経験を積んだ人に発生する**“出口付近での迷走”**とも言える。

代表的な離脱の原因は以下のとおり：

- ポートフォリオの制作で詰まり、先に進めなくなる
- 転職活動・営業活動への心理的なハードルが高すぎて踏み出せない
- 「学ぶこと」に満足してしまい、動き出す意義を見失う
- 完璧主義的な性格が足かせになり、公開・提出ができなくなる

これらはいずれも、“実力が足りないから”ではなく、“出口が設計されていなかったから”起こる構造的失敗である。

2. サイバータンクの方針：学びの「着地設計」を初期から埋め込む

サイバータンクでは、離脱を教育の“出口”設計の問題と位置づけ、
学び始めた瞬間から**「どう終わるか」「どう動き出すか」を同時に設計する**方針を採っている。

3. 実際の取り組み内容

● ポートフォリオ設計の“誘導モデル”提供

- 「ゼロから自由に作ってください」ではなく、複数の題材テンプレートや構成例を提示
- 過去の実績や“実務に近い構成”を紹介することで、「何を作ればいいか分からない」状態を回避
- 技術より「意図と完成」の構造を重視し、自己満足ではなく“伝える作品”の作成を支援

● 学習中からの**Tech**ブログ執筆習慣化

- 学んだ内容、バグ調査、実装方法をブログとして週**1**でアウトプットするよう設計
- この習慣がポートフォリオ完成後の発信への心理的ハードルを下げると同時に、
  – 成長記録
  – 他者評価
  – 就活時の“裏資料”
  としても機能する

- 実際、Techブログ経由で面接が決まった例や、記事の構成力を評価された例も複数報告されている

● “本番接続”を見越した教育設計

- 教材や開発課題においても、**「誰かに使われる前提」**で仕様・デザインを考える構造を導入
- 「動けばOK」ではなく、「使う人に届くか？」を意識した設計・文書化・UI思考を日常的に訓練
- 結果的に、学習と実務の接続がスムーズになり、“いざという時に動ける人”を育てる

● 「一発勝負」ではない到達戦略の共有

- サイバータンクでは、「転職・副業に成功しなければ意味がない」という一発勝負的思考を否定
- 学習・作品・記事・Slackでの発言など、あらゆる出力を“複数の武器”として蓄積する戦略を伝える
- これにより、「今はまだ動けない」人も、“動けるタイミングでいつでも使える素材”を持った状態になれる

4. 成果と展望

離脱率の高さは、実力や才能の問題ではない。
「動き出し方が分からない」「出口の構造がなかった」という教育側の設計ミスである。

サイバータンクでは、ブログ・**Slack**・**GitHub**・ポートフォリオ・**Tech**ブログ・面談内容などすべてを出口で“動くための材料”として育てる文化と構造を作っており、
実際に正社員としてのキャリア以外に未経験からフリーランスになった例も存在する。

今後はさらに、**“動き出すためのミニゴール設計”や“発信の自動補助ツール”**を整備することで、
「静かな離脱」が起こらない学習構造の標準化を目指していく。

サイバータンクの事業取組⑤：教育内容問題への対応（完全版）

---

1. 対応する課題の再定義

教育内容問題とは、教育カリキュラムや教材が、学習者の目的や現代の実務に対して最適化されていないことによって起こる、成長の阻害・納得感の欠如・不信の蓄積を指す。

この問題は単なる「教材の質」の問題ではなく、次のような構造的・思想的なミスマッチが本質である:

- スキルを教える順序や範囲が目的とズレている
- 学習者の“全体像”の把握が想定されていない
- 文脈や哲学を欠いた“作業の手順”だけが提供されている
- 教材が更新されず、古いスタックや環境前提のまま放置されている
- 「調べる力」や「言語化する力」といった実務力が育たない
- 「文系でもできる」ことを証明する設計思想がない

2. サイバータンクの方針:実務接続 × 思考習慣育成 × 継続進化

サイバータンクでは、教育内容を**“知識の配布”ではなく、“構造化された思考訓練”**として設計している。
その根幹にあるのは、「意味がある構文」「納得できる設計」「再利用可能な思考」の3本柱である。

加えて、**「教材自体が進化し続けること」**を前提とした実践体制を構築している。

3. 実際の取り組み内容

● 実務構造に準拠した学習設計

- RailsのMVC設計、DB設計、RESTfulルーティング、Git管理、Herokuデプロイ、テスト、API連携など、現場で実際に使われる技術構成を体系的に設計
- 「どの技術が、なぜそのタイミングで必要なのか？」という文脈ごと学べる構成
- 技術単体ではなく、現場で使う理由・背景・目的に基づいた技術選定力も育成

● 全体構造の提示と反復

- 学習開始時点で「Webアプリとは何か」「ブラウザ〜DBまでの流れはどうなっているか」を視覚的な技術マップで提示

- 各単元ごとに「この機能は全体のどの構造の一部か？」を再確認させ、自己の現在地を見失わせない設計を導入

● 教育内容への技術哲学の埋め込み

- Railsの思想（Convention over Configuration）、MVCの責務分離、設計パターンの選定理由などを毎回の演習や**Slack**回答の中で伝えている
- 「この構文は何を前提にして設計されているか？」という問いを共有し、“意味で学ぶ”学習構造を確立

● 調べる力と考え抜く力の訓練

- 教材・質問対応ともに「正解を渡す」ことを避け、調べ方・試し方・分解の仕方を見せることを重視
- 「分からないことを、どうやって理解に変えるか」という思考プロセス自体を反復学習の対象としている
- 詰まった時のSlack質問には、回答よりも質問者自身の切り分けを誘導するアプローチを採用

● 言語化力と説明力の定着

- 手順書の作成、Slackでの説明、Techブログ、ペアでの口頭報告などを通じて、**「考えたことを他人に伝える技術」**を体系的に育てている
- この言語化能力は、報告・相談・レビュー・設計共有などすべての現場実務で不可欠な力であり、卒業後にその効果が可視化される

● スタッフ全員の“現場感覚”保持と教材進化

- 講師・メンターを含むスタッフ陣は、**Rails**を用いたフリーランス受託開発や自社サービス運営に日常的に関与しており、実務の感覚を維持
- 最新技術動向にも追従しており、**Rails 8**、**Kamal**、**SolidQueue/SolidCable**といった新機能も既に運用・教材化済である**(2025**年**4**月段階**)**
- 教材は「教える側の思いつき」で作られず、現場のトライ＆エラーを咀嚼した上で体系的に整備されている

4. 成果と展望

このようにサイバータンクでは、教材そのものが「動的」であり、「人に依存しない構造」「思考習慣の訓練装置」として設計されている。

受講者からは「技術も教わったが、思考法と仕事の姿勢が変わった」というフィードバックが多数寄せられている。

今後はさらに、以下の2点を強化していく予定である：

- **目的別技術マップ（転職／フリーランス）**の開発と公開
- 教材更新の自動可視化＋バージョン管理＋受講者フィードバック反映型運用への発展

サイバータンクは、「教育とは内容ではなく構造である」という立場から、**教材と思想の“進化型教育”**を推進していく。

サイバータンクの事業取組⑥：教育者スキル問題への対応

---

1. 対応する課題の再定義

教育者スキル問題とは、技術的な知識を持っている人が必ずしも「教える」ことができるとは限らないという構造的な課題である。
教えるという行為は、以下のようなスキルを要する高度な“構成的コミュニケーション”技術である：

- 相手の理解度を把握し、情報を適切な粒度に再構成する力
- わかりにくさを見抜く感性と、その説明を最適化する表現力
- 誤解や曖昧さを「質問」「例示」「図解」などで明確に取り除く力

しかし多くの教育現場では、「エンジニア経験がある＝教えられる」という誤認に基づいて講師が登用され、
結果として学習者が“分かったふりのまま置き去りにされる”構造が起こっている。

2. サイバータンクの方針：「伝える技術」をチーム単位で体系化する

サイバータンクでは、「教える」とは**“理解を再現させること”であり、技術と同様に鍛える対象である**と定義している。
そのため、「説明・例示・再構成・質問の活用」といった教育スキルを講師チーム内で形式知化し、教育自体を言語と構造で再現可能な技術として蓄積している。

3. 実際の取り組み内容

● 思考プロセスの“実況”と再構成の訓練

- 講師陣は、受講者の質問に即答するのではなく、**「どう考えて、どう仮説を立て、どう検索したか」**を口頭で逐次開示

- これにより、受講者は「答え」ではなく「再現可能な思考の流れ」を学ぶことができる

- 「調べる力」「分解する力」「検証する流れ」を実務の再現として体感できる授業設計

● 言語化の型をチームで共有・訓練

- 講師全員が「説明の粒度」「順序」「使う比喩」「Slackでの返信文法」まで共有し、**“属人性の排除”と“教育品質の均質化”**を実現

- 「文系でも分かるように伝えるには、どう説明するか？」という問いを共有し、説明の**PDCA**をチームで回す体制を敷いている

● フィードバック文化と教育スキルの自己評価

- 教える側同士が互いの**Slack**返信や**1on1**対応にコメントし合う文化が浸透

- 「その教え方は良かった」「この言い回しだと誤解が生まれやすい」といった観察と言語化の積み重ねにより、教育自体を改善対象として扱う運用が可能になっている

● チーム制によるリレー型指導と“文化の継承”

- 教育が「講師の個性に依存しない」よう、**リレー型の対応（Slack／面談／レビューなど）**を前提にチームでの共有が徹底されている

- 初学者の心理的安全を守るため、「どの講師に聞いても一貫した回答が返ってくる」状態を実現し、信頼感と安心感のある学習空間を構築

- この文化設計自体が、学習者に対する「チームで支える」という信頼のメッセージとなっている

4. 成果と展望

サイバータンクでは、「教える力」を学習と同じく訓練すべきスキルとして正式に定義し、評価の対象にしている。
実際に、「教える側が育ち続けている」「どの講師の説明でも腹落ちする」といった声が多く、
教育品質の安定と属人性の排除において高い効果を上げている。

今後はさらに、教育メソッドのマニュアル化と社外公開などの取り組みを検討している。

教育者スキルは「資質」ではない。
思考と説明の技術を言語と構造により再現可能にし、チームとして育てていくことが、
サイバータンクの「教えること」に対するスタンスである。

サイバータンクの事業取組⑦：教育方法問題への対応

---

1. 対応する課題の再定義

教育方法問題とは、「何を教えるか」ではなく「どう教えるか」の設計に失敗していることによって、学習効果が著しく低下する現象を指す。
同じ教材・同じ情報を使っていても、「伝え方」「進め方」「確認の仕方」によって理解度は大きく変わる。
にもかかわらず、多くの教育現場では以下のような問題が散見される：

- 一方向の講義型に依存し、学習者の能動性や状況理解を無視した進行
- 「質問が出なければ理解している」とみなす設計不全
- “動けばOK”という思想による、思考の置き去り
- 成果物の見た目重視で、中身の設計思想や過程の振り返りがなされない
- フィードバックが断片的・気まぐれで、成長の循環が起こらない

これは単なる指導技術の未熟さではなく、「学習とは何か」という思想設計の欠如に起因する構造的な問題である。

2. サイバータンクの方針：進捗・思考・習慣の“見える化”による設計主義

サイバータンクでは、教育方法を**“本人の努力に依存しないように設計された環境そのもの”と定義している。
つまり、「続けた人が育つ」のではなく、「育つように仕組まれた環境を用意する」**という逆転の思想である。

3. 実際の取り組み内容

● 合目的性を起点とした授業設計と「意味論的理解」の促進

プログラミング教育においては、文法や構文を正しく覚える**統語論的理解（syntactic understanding）だけでは、実務的な応用力や実装能力にはつながらない。
重要なのは、「この技術は何のために存在し、どんな問題を解決するために使うのか？」という意味論的理解（semantic understanding）**であり、
その理解がなければ、学んだ知識が断片化され、応用不能な“暗記学習”に終始してしまう。

サイバータンクでは、全ての講義設計において「合目的性」を重視し、以下の5ステップ構造を教育フレームとして導入している：

1. 概要：その技術が何を扱うものか（定義）

2. 使用場面：どんな文脈でそれが必要とされるか（ユースケース）

3. 関連技術：どの技術と連携・依存関係があるか（体系理解）

4. できること：それによって可能になること（ゴール）

5. 文法・構文：実装のために必要な最低限の文法（手段）

この5つをセットで提示することで、「学ぶ意味」が明確になり、単なる記憶から脱却し**“使い方を知っている状態”としての理解**が定着する。
さらに、講師にはこの構造に基づいて事前準備・教材作成を義務付けており、意味論的理解を引き出すための再現可能な授業設計を講師全体で共有している。

また、このフォーマットを講師の授業準備に義務づけることで、授業内容の品質を一定水準に保ちつつ、誰が担当しても同じ構造で伝えられる再現性を確保している。

これにより、教育が特定のカリスマ講師に依存しない設計となり、属人化を防ぎながらも学習者にとっての「理解しやすさ」が制度的に担保される。

この構造化されたアプローチは、初学者にとって**「これは何のためにあるのか？」という問いに答えながら学ぶ安心感をもたらし、
学習者自身が知識を“目的に応じて選択的に使いこなす”**姿勢へと変容する礎となっている。

● 進捗の可視化と構造的フィードバック

- 学習の「どこまでやったか」ではなく、「どう考えたか」「なぜそうしたか」まで含めて報告させるSlackフォームを活用

- 講師陣はそれに対して、「技術的アドバイス」と同時に「思考構造へのコメント」を返すことが標準化されている

- これにより、“動いたコード”より“考えた過程”に対して成長フィードバックが返る設計が成立している

● 思考の見える化：手順書作成タスク

- 受講中、学習者には定期的に**「手順書」または「調査記録」の形式で自らの思考過程を文書化する課題**を課している

- これは以下の力を同時に鍛える仕組み：

    - 理解を他人に伝える言語化力

    - 設計・修正・再試行の思考再構成力
    - チームで共有可能なナレッジ変換力
- また、手順書作成自体が「内省の場」となり、“なんとなく理解”のまま次に進むことを防ぐ

● 評価文化の転換：「結果」より「過程」に評価軸を置く

- テストや成果物の完成度だけを評価せず、“どう考えてそこに至ったか”という過程を最重要視
- これにより、「自分は人より進みが遅いから劣っている」と感じる人でも、自分の思考が評価される経験を積むことができる
- この「思考評価文化」こそが、学習者の安心と持続性を支えている

● 習慣としての学び：定期内省と習慣化支援

- 毎週、「何を学んだか」だけでなく「何に詰まったか」「どんな感情があったか」を共有する時間を設け、Techブログの投稿やバディとの共有、共同作業により、学びを“出来事”で終わらせない設計を導入
- 教育内容を消費ではなく蓄積として扱い、学ぶこと自体が“文化”として身体化される状態を目指している

4. 成果と展望

サイバータンクの教育方法は、「学ぶ力そのものを育てる」ことに重心を置いている。
ただ正解に導くのではなく、「自分で解けるようになる構造」を与え、
さらには「学んだことを他人に伝えられるようにする」ことをゴールとして設計している。

その結果、多くの受講者が**卒業後も自律的に学習を継続し、他者の学習支援にも関わる“再帰的学習者”**となっている。

今後は、以下の展開を構想中：

- 手順書・調査記録などの学習資産を、他者が検索・再利用できる公開型知識ベースへと展開
- 「教育方法を教える」ための教員養成トラックの整備
- 教育方法の設計ノウハウ自体を書籍化・教材化し、教育者コミュニティへの提供