

特別抜粋版



42の失敗事例で学ぶ
マネジメントのうまい進めかた

出石聡史
DEISHI SATOSHI

エンジニア 育成現場の 「失敗」集め てみた。

チームマネジメントの落とし穴が満載！

- ✓すべてが最優先「FIFO型業務依頼」
- ✓アレと同じに作って「現物合わせ仕様」
- ✓チームでひとりぼっち「疎結合配属」

「この職場、無理…」と思われる前に！

SE
SHOEISHA

ご案内

本資料は、2025年6月11日発売の書籍『エンジニア育成現場の「失敗」集めてみた。42の失敗事例で学ぶマネジメントのうまい進めかた』から一部を抜粋し、特別に編集したものです。

全書は『エンジニア育成現場の「失敗」集めてみた。42の失敗事例で学ぶマネジメントのうまい進めかた』でご覧いただけます。購入および、詳しい書籍情報は以下のリンクからどうぞ。

Amazon販売ページ



<http://www.amazon.co.jp/dp/4798189669>

書籍情報



<https://www.shoeisha.co.jp/book/detail/9784798189666>

本書を読み始める前に

●本書で扱う架空の組織とプロジェクトについて

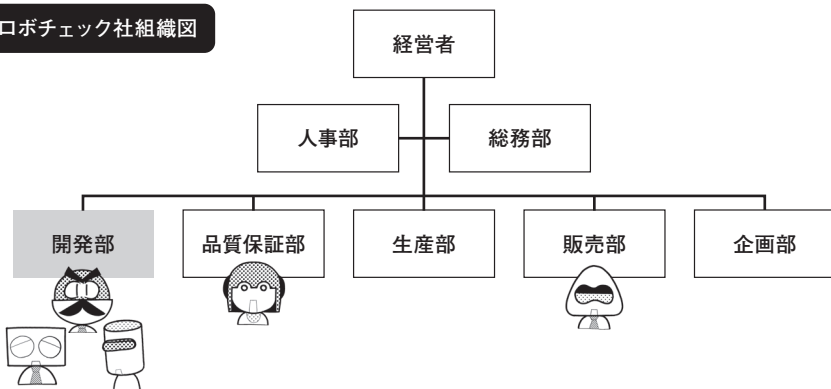
本書ではエンジニアの採用や育成の失敗を追体験できるように、架空の開発組織とプロジェクトを舞台にしたエピソードを紹介します。また登場人物も明らかにフィクションであるとわかるように、全員ロボットにしています。ただ、それ自体重要な要素ではありませんので、あまり気にせずに読み進めていただければと思います。

◆エピソードの舞台

本書の舞台となる株式会社ロボチェックは、ロボ部品の生産をサポートする検査機器のメーカーです。ロボ部品の生産において、計測、診断、調整をサポートし、品質をコントロールするためのハード、ソフトを含めたシステムを提供することがロボチェック社の主な業務です。

ロボチェック社は、開発部、生産部、品質保証部、販売部、企画部、総務部、人事部と、業務内容ごとに組織が分かれている機能別組織です。本書はこのうち「開発部」を中心に話が進んでいきます。

ロボチェック社組織図



◆エピソードの背景

本書の主人公であるチームリーダーのハルさんは開発部に所属しています。ハルさんはその高い技術力で、これまで様々なプロジェクトを担当し、失敗を重ねながらも最終的に成功に導いてきた実績があります。その実績を評価され、今年の課長昇格候補となりました。そこでブチョーさんとカチョーさんがハルさんの昇格について相談を始めたのですが……。



ハルさん今年は課長に推薦できるんじゃないか？「アームチェッカープロジェクト」も、なんやかんやあったが想定以上に売り上げておる。成果も出ておるしアピールもしやすいぞ。



そうなのですが、ちょっと気になることがありまして……。ハルさんは技術のエキスパートとして、とても優れた人材なのですが、課長としてはまだ課題があります。



うーむ。いわんとしていることはわかるぞ。チームマネジメントと人材育成の観点じゃな。



そうです。自分がなんでもできてしまうため、チーム力を引き出し、部下により経験を与えて成長させる視点が弱いのです。そこで1つお願いなのですが……。

カチヨーさんの提案は「今年いっぱいハルさんに係長として動いてもらうこと」でした。つまり、ハルさんが課長としての業務を一部受け持つことで、課長としての視座を得ると共に、適性を見ることにしたのです。



ということで、この1年は係長として、一部課長業務に挑戦してもらおうと思います。結果を出して来年度の課長昇進を目指しましょう！



えええ！？ 僕が係長ですか？ しかも来年度課長に？ いやいや、できる気がしないのですけど……。でも期待していただいていることは素直にうれしいです、ハイ。では頑張ってみます。



特にハルさんには、チームのマネジメントや人材育成の能力発揮を期待しています。チームのみんなが力を発揮し、成長できる職場となるよう、工夫をしてみてくださいね。



では早速じゃが、ソフトウェアチームに1名新入社員を配属することとした。紹介しよう。ガクさんじゃ。



はじめましてガクです。まだまだ未熟な点多いかと思いますが、皆様のご指導をいただきながら、1日でも早く戦力になれるようがんばります！ どうぞよろしくお願いいたします。

ハルさんは、まだ自分に係長や課長の役職は早すぎると感じていました。しかし、これを自分のキャリアを見直すチャンスと捉え、前向きに取り組むことにしました。初々しいガクさんを見て、さらにやる気も出てきました。果たしてハルさんは無事に課長になることができるのでしょうか。

●本書における用語の使い方

本書において、各種用語はそれぞれ下記の定義で使用しています。

要望 顧客の望むシステムへの期待。顧客の言葉そのもの。

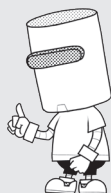
要求 システムに対する、インプットとアウトプットの期待。どんなインプットを行ったときにどういうアウトプットを期待するのか。要望を分析し、顧客の真の課題から効果的な要求を見出す。

要件 システムに対する、インプットとアウトプットの実態。どんなインプットを行ったときにどういうアウトプットが実際に返ってくるのか。システムの機能を外側から見たときの入出力を定義するもの。要件を実際にシステムとしてどう実現するか定義する。

課題 目標と現状との差異を埋めるための取り組み。作業を開始していないタスクも課題にあたる。

問題 目標と現状との差異そのもの。主に課題を実施したにもかかわらず、なお目標と現状との間に差異がある場合に用いる。

【主な登場人物】（というかロボ）



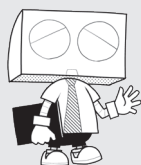
ハル

この本の主人公。高い技術力と知識を持ち、これまで技術者として事業に貢献してきたが、このたび課長候補として、ソフトウェア開発チームのマネジメントを任されることとなった。チームには新たに新入社員のカクさんも加わり、育成にも期待が寄せられるが、果たして無事課長昇進を果たすことができるのか。



ガク

ハルさんのチームに配属された新入社員。大学在学時からプログラミングに携わり、即戦力として期待されている。将来のキャリアビジョンはこれからだが、まずはエキスパートを目指している。社会人として不安いっぱい状態で、ハルさんのチームに配属となるのだが……。



カチョー

ハルさんの上司。課長。技術者としての実力を買ってハルさんを係長に任命し、課長の業務を一部任せることにした。ほんわかとした雰囲気とは裏腹に、多くの実績を上げている実力者。ハルさんによい経験をさせようと画策している。



ブチョー

開発部門の部長。ソフトウェアのことはあまり詳しくない。どちらかというと昔気質の管理職で、いつも無理難題を吹っかけてくる。部下を大事に考えている一方、組織の長としては厳しくありたいと思っている。



シンテク（コーハイ）

若手技術者。ソフトウェア工学の知識が豊富で、コーディングスピードも速い。若手随一の技術力を誇るが、調子のよいところがあり、深く考える前に手を動かしてミスすることも多い。ハルさんを師匠として尊敬している。ハルさんは以前、愛称としてコーハイと呼んでいたが、チームに人も増えてきたので、改めて名前で呼ぶようになった。



ノビシロ（シンジン）

昨年の新人。カクさんの1年先輩にあたる。もともとプログラミングの経験はなかったものの、持ち前の好奇心とコミュニケーション力を発揮して急成長中。理解力に優れ学ぶ力が強い。ドーナツが大好き。ハルさんは以前、愛称としてシンジンと呼んでいたが、新人のカクさんも入ってきたため、名前で呼ぶようになった。

Contents



はじめに～技術力を持たない技術者たち～	iii
本書を読み始める前に	viii

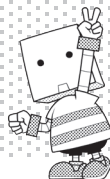
Chapter

1

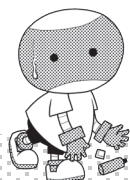
「新人研修」で失敗

Episode 01	やる気を削るタライ回し「新入社員レンタルサービス」.....	002
Episode 02	ダメになるまで報告しない「しなびたハウレンソウ」.....	008
Episode 03	無用の業務を作り出す「雑務ビルダー」.....	014
Episode 04	すべてが最優先「FIFO 型業務依頼」.....	020
Episode 05	完璧な報告を強制する「賢者のレポート」.....	026
Episode 06	謎のコメントで迷宮入り「ダイイングメッセージ型コメント」...	032
Episode 07	計画を見直さない「回すだけPDCA」.....	038
Episode 08	悪い情報は遠ざける「無菌培養育成」.....	044
Column	必殺技を持つ技術者を育てる	050

「OJT」で失敗



Episode 09	業務の目的を伝えない「指示だけ上司」.....	052
Episode 10	アレと同じに作って「現物合わせ仕様」.....	058
Episode 11	若手のアイデアに過度な期待を寄せる「フレッシュ厨」.....	064
Episode 12	最恐マルチ3兄弟の末弟「誰も使わないマルチユーザーソフト」....	070
Episode 13	最恐マルチ3兄弟の次兄「日本専用マルチランゲージソフト」.....	076
Episode 14	最恐マルチ3兄弟の長兄「増殖するマルチスレッド」.....	082
Episode 15	ルール無用の人任せ「仕様無法地帯」.....	088
Episode 16	テスト不要で機能を通す「裏口リリース」.....	094
Column	期待と現実のギャップを認識する	100



「チーム編成」で失敗

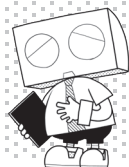
Episode 17	キャリアも希望も関係ない「とりあえず配属」.....	102
Episode 18	チームでひとりぼっち「疎結合配属」.....	108
Episode 19	俺は独りで学んできた「一匹狼エンジニア」.....	114
Episode 20	失敗の経験を与えない「一流技術者のメッキ」.....	120

Episode 21	コロコロチームが変わる「社内フリーター」	126
Episode 22	キャッチアップできない「伝統技術の継承者」	132
Episode 23	権限を持たせない「人間将棋の駒」	138
Episode 24	24時間開発できますか「技術ジャンキー化計画」	144
Column	よいチームのための6要素	150

Chapter

4

「評価」で失敗



Episode 25	成果がはっきりしない「ほんわか目標設定」	152
Episode 26	問題しか見えない「バグチケット起票型上司」	158
Episode 27	チームの成果を独り占め「エース技術者推し」	164
Episode 28	自己評価は絶対です「本人評価コピペ係」	170
Episode 29	結果だけをフィードバック「エラーコード式評価通知」	176
Column	成果主義の世界の片隅で	182

Chapter

5

「コミュニケーション」で失敗



Episode 30	すべてを知りたがる「メトリクスコレクター」	184
Episode 31	すべてのムダを排除する「やりすぎDX化」	190

Episode 32	悪いのは部下「つるし上げの連鎖」	196
Episode 33	権限は下におろさない「責任だけ移譲」	202
Episode 34	過保護がやりがいを奪う「上っ面ホワイト育成」	208
Episode 35	規律と効率でチームを縛る「束縛型上司」	214
Episode 36	自尊心を打ち砕く「偏見タグ付け上司」	220
Episode 37	人に興味のない「たたきたがり上司」	226
Column	管理職の資質チェックリスト	232

Chapter

6

「採用」で失敗



Episode 38	スキル重視の新卒採用「スペック偏愛採用」	234
Episode 39	いい人だと思ったのに「恋する面接官」	240
Episode 40	応募者をやりこめる「技術学会面接」	246
Episode 41	楽しいだけの「テーマパーク型インターンシップ」	252
Episode 42	誰もついてこれない「超難関インターンシップ」	258
Column	面接で探すべきは「普遍的な長所」	264

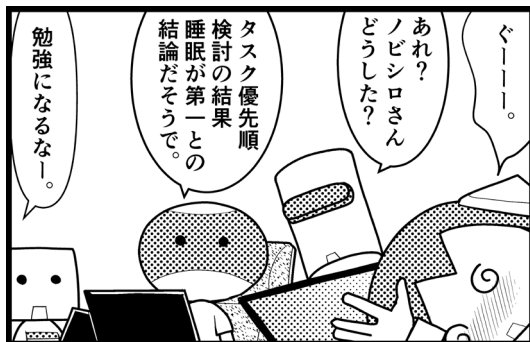
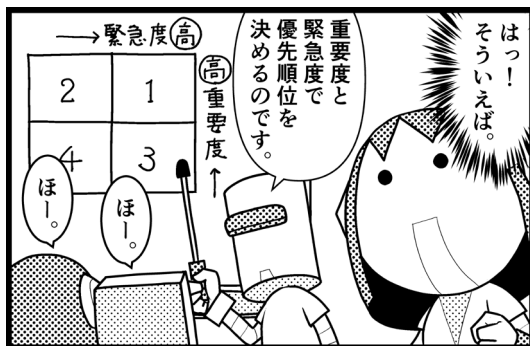
数多の失敗を乗り越えて～エピソード～ 266

おわりに～ソフトウェアは1人では作れない～ 268

「悩み」から探せる索引 270

すべてが最優先 「FIFO型業務依頼」 すでに新人のキャパは超えている

筆者の経験上、時間にゆとりがあるソフトウェア開発現場など、ついぞ見たことはありません。トップから若手に至るまで業務過多にもかかわらず、次々に新規業務が流れ込んできます。当然すべてをこなす時間はありませんから、優先順位を付け、断るべき業務は断らないと首が回リません。とはいえ、そんな取捨選択が新人にできるわけもなく、結局もらった業務をもらった順にこなそうとして破綻してしまいます。



そんなにたくさんの仕事はできない

現在ガクさんは、新入社員の教育カリキュラムとして開発部の様々なチームで業務経験中です。その傍ら、週に1回は配属先であるハルさんチームのミーティングにも参加をし、状況報告とチームの業務に対する理解を進めています。



「スキンチェッカー」の進捗報告、ありがとうございます。えーと今週の課題はシンテクさん担当のプロープ信号の取得が時々おかしくなる件か。ノビシロさん、原因を把握するために、データ取りを手伝ってもらえるかしら？



えええ。ちょっと今週はアプリの受入検査仕様書作りで手一杯ですー。しかもまたブチョーさんから謎の指令が来ていて……。



な、謎の指令……。あ、社内DX推進委員からのヒヤリングか！ あれは緊急でもないので来週以降に延ばしてもらおう。受入検査仕様書なら俺も書けるから一部引き受けるよ。



申し訳ないっす。データ整理をガクさんにもお願いできると助かるんすけど……。というのも今さっきメールでヒンシツさんから販売中の「アームチェッカー」にもヤバイ問題が出ていると連絡が……。

プロジェクトの課題も解決しないといけなのに、どんどん新しい問題が入ってきます。明らかに全員キャパオーバーです。ガクさんでもできることがあればと、自主的にシンテクさんのデータ整理を引き受けたのですが……。



ガクさん。お願いしていた棚の整理……じゃなかった、試作回路のデータ取りどうなった？ もし問題があるなら早く直したいんだけど。



あ、すみません。ハルさんをお願いされたデータ整理をやっていました。お急ぎなのですね？ では先にそちら取り組みます。



ガクさんデータ整理はできた？ ってあれ？ エレキのデータ取りしてんの？ あー実習より、こっちゃバいので優先してくれる？



え？ えーと、エレキチームからデータ取りは回路修正を急ぐから、優先度上げてほしいと。えええ？

あれもこれも最優先と言われて、ガクさんも困惑してしまいました。とはいえ体は1つ。いったいどうしたらいいのでしょうか。



これはもう残業をお願いするしかないのかなあ。でもハルさんとかシンテクさんとか、こんなものじゃない業務量なのだけど、いったいどうなっているのかしら？ いつどうやって仕事しているの？

このまま業務が増えていくと、夜も休日も、会社でも家でも働き続ける生活になりそうです。そんな未来を想像して、ガクさんは暗鬱とした気分になるのです。



業務量は常に自分のキャパを上回る

新社会人になって驚くのは、想像していたよりいろいろな業務がものすごい物量で押し寄せてくることです。そう、会社は従業員それぞれに、一見こなせない量の業務や、解決困難な業務の遂行を要請してきます。常に自分がこなせる量より2〜3割増し、ひどいときには倍の業務が流れ込んできます。

「すべての業務はできない」とあきらめることを教える

とはいえ会社に入ったばかりの新人に対し、なんの予備知識もなくこの業務シャワーを浴びせかけてしまうと、途端に風邪を引いてしまいます。新人は要請された業務を、依頼された順（FIFO：ファーストインファーストアウト）にこなしますから、優先度の低い業務に時間を取られ、本当に大事な業務がおざなりになることもあります。



メイン業務以外にこんなに仕事が入ってくるなんて！ しかも全部急ぎだって！ そんなことある？ 業務をこなす自信がなくなるなあ。

しかも、いろいろな人から同時に業務要請が入りますから、新人のキャパはすぐにあふれてしまいます。まじめな新人ほど、言われたことをすべてこなさねばならぬと考え、すべてに対応しようとするのですが、そもそも業務量が自分のキャパを超えていますから、納期までに十分な結果が出せなくなっていきます。業務が滞っていくに

つれ、依頼者からは文句を言われ、ひどい場合には「仕事ができないやつ」と認定されてしまいかねません。

ここが失敗のポイントです。**業務のさばき方を教えず、どんどん業務指示だけを与え続けると**、いつしか業務が新人のキャパを超え、十分対処することができなくなってしまいます。その結果、新人は業務に対する自信を失い、「理想の自分」と「実際の成果」とのギャップに苦しむようになります。できるはずのことさえ、できなくなっていく。そんな状態で無理を続けていると体調にも不調をきたし、最悪メンタル不全に陥る可能性もあります。

優先度を付けることを指導する

まず大事なのは、新人に対する指示命令系統を明確に一本化することです。期待の新人ほど様々な人から業務依頼が舞い込むのですが、一旦新人の上司経由で依頼してもらうように調整します。新人は他から要望されたら断れませんか、上司も新人の業務を理解し、活動や成果をきちんと評価できるように、新人への依頼はすべて把握しておく必要があります。

そのうえで、業務に優先度（プライオリティ）を付けて、優先度の高い業務からこなしていくように指導します。優先度の低い業務は後回しにするか、いっそ断ってもいいんだよとアドバイスをします。

優先度の付け方として、一般的によく使われているのが緊急度と重要度のマトリク

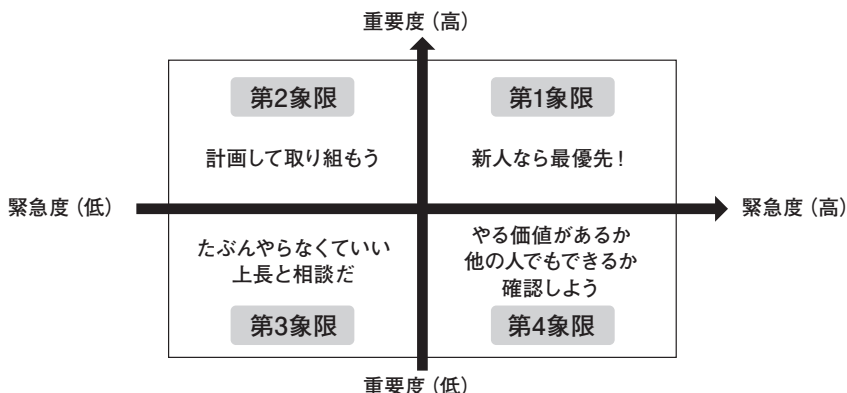


図 時間管理のマトリックス（新人向け）

スで判断するという手法です。これは、『7つの習慣』（スティーブン・R・コヴィー著）という書籍で紹介されている時間管理のコンセプトです。

ではこの中で最初に取り掛かるべきはどの象限でしょうか。普通に考えると第1象限、すなわち重要でありかつ緊急な業務ですが、この本では、実は最も大事なものは第1象限ではなく第2象限の「重要だが緊急ではない」業務であると述べられています。緊急性の高い目の前の業務をこなしていただだけでは、本質的で、より大きな高い目標を見誤るからです。

新人には「重要でありかつ緊急」な業務を最優先に

しかし、新入社員向けには、あえて第1象限の「重要でかつ緊急な業務」を優先度高く設定することをおすすめします。新人のころはまず業務を覚えること、上司の期待に応えること、そして**なぜこの業務が重要で緊急なのかを理解することが大事**です。なぜその業務に取り組む必要があるのか、目的は何か、これによって何が得られるのかなど、業務の内容や背景を理解し学ぶことが重要です。

やらないことを決める

次に肝心なのが第3象限です。新人に業務を依頼する際、第3象限に該当する作業は、自分自身のキャパシティや状況に応じて「やらない」という選択肢を選んでもよいと伝えてあげましょう（もちろん、事前に相談はしてもらうよう念押ししたうえで）。新人なのに依頼された業務をやらないと決めるなんておこがましいと思いかもしれません。しかし、最初に述べたように与えられた業務をすべてやりきることが難しい場合もあります。重要でもなければ緊急でもないものに貴重な時間を使う必要はないはずです。

断るかどうかを新人が判断するためには、まず上司が手本を見せることが効果的でしょう。こういう業務はやらなくていいという実例があれば、新人にも判断の基準が持てるようになります。

ただ本当に「やらない」ためには、利害関係者の合意が必要です。業務にはどんなステークホルダーがいて誰と折衝すべきか、新人にもわかるようにしておくことはチーム全体にとっても価値があります。ステークホルダーリストを作って随時更新しておくのがよいでしょう。

価値を明らかにして動機付けする

第4象限は最も扱いが難しく、新人にこの業務を依頼するなら、上司がしっかりと

業務の価値を明らかにすることが大事です。

この象限の業務は本質的に重要度が低いため、やらなくてもよい作業に見えます。しかし、緊急案件に対応することで、依頼者に感謝されて関係性がよくなり、その後の業務もやりやすくなる、という効果もあります。それゆえ、この業務を実施するにあたっては上司から業務の価値を伝え、本人にしっかり動機付けすべきでしょう。

新人のうちはまず「重要かつ緊急の業務」を解決していくことで、部署内の信頼感も生まれてきます。緊急度の高い業務はなんといっても緊急には間違いありませんから、急いで対応してくれるのは本当に助かります。

上司として業務を依頼する際は、まずはこの重要性和緊急性の2軸で業務を振り分けることを指導してみてください。また困ったときには気軽に相談を受け、相談しやすい環境を作ることも重要です。



業務があふれそうなら相談して。不要な作業なら俺から断るし。反対にガクさんにとってプラスになることもあるから、まずは相談してね。

とにかく言われた順で仕事をするのをやめていい、「やらないでいい」タスクがあるなら相談の上断っていい、ということを伝えるだけでも心が軽くなるものです。

まとめ



失敗

- ⋮ 新人にすべてを最優先としてキャパ以上の業務量を依頼し、体調不良やメンタル不全を招いた



回避策①

- ⋮ 新人に対する指揮命令系統を明確に一本化する



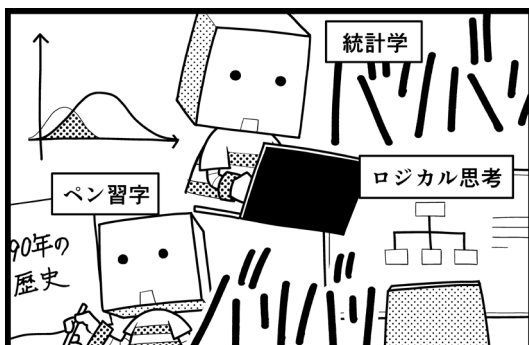
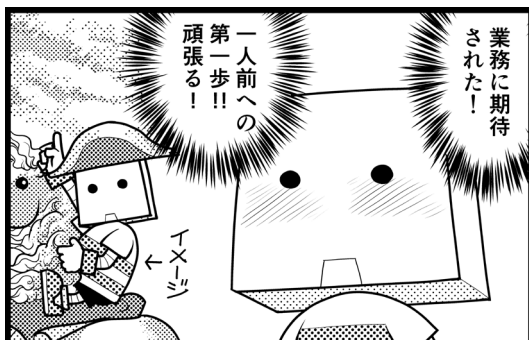
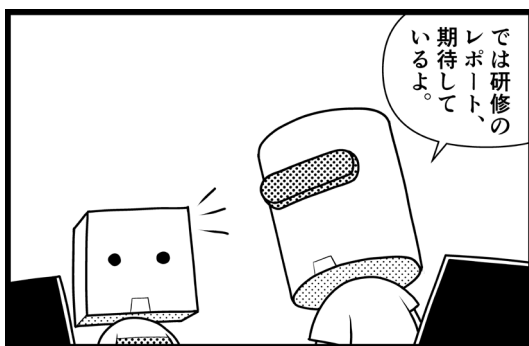
回避策②

- ⋮ 重要性和緊急性の2軸で業務を振り分け、優先度を付けさせる

完璧な報告を強制する 「賢者のレポート」

60%の完成度で、
おおむね用事は済んでいる

業務にはレポートがつきものです。出張に行ってもバグを修正しても、なんらかの報告を文書で残します。ただ新人の作るレポートは、上司からはどうにもお粗末に見えてしまいます。これも教育の一環と差し戻すのですが、十分な説明がなければ、新人には何が足りていないのか、理解できません。見えない完成に向けて、時間を無為に使いわせるだけです。



実験結果をレポートしたつもりなのに

ガクさんは今週、アルゴリズムチームで測定原理の研修です。刺激プローブでロボット皮膚に様々な刺激を与え、刺激に応じてどういう信号が返ってくるのか、実験を行っています。



実験お疲れ様です。ではこれまでの実験結果をまとめて報告書にしてくれるかな。レポートすることも業務だからね。



了解しました！ えーと、今日中ですね。Excelシートにデータは取ってあるので、すぐにお送りできます。

ガクさんはこんなこともあろうかと、測定したデータはすでにExcelにまとめていました。早速Excelに表紙を付けてアルゴリズムリーダーに送りました。



あーガクさん。このレポート、データしかないよ。このデータをどう分析したのかが大事なので、考察を追記してもらえるかな？

あれ？ 実験結果をまとめてと言っていたのに……とガクさんは少しもやっとなりましたが、早速レポートにグラフを追加し、考察を書き加えました。



考察を加えてくれてありがとう。だけど、これ事実と考察が混じっちゃっているよ。そこは分けてくれないと困るのだけど……。

またまた書きなおしです。改めてExcelの別タブに考察のためのシートを作成し、明確に実験結果と分けてみました。さすがにもうこれで完璧だろうと、リーダーに送ったのですが。



結局何がわかったのかよくわからないなあ。いっぱい考察は書いてくれているけど、最初のページに簡潔に目的と結果を書いてほしいんだ。そもそも目的ってなんだっけ？ その目的は達成できたの？

なんだかまたゴールが遠ざかった気がします。そういえばこのレポートの目的ってなんでしょう。目的は研修ということなのでしょう。研修は終わったので達成

できていますが、なんとなくそういうものではない気がします。

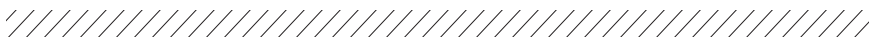


えーと、いったい何が記載できていたらいいのかなあ。すぐできると思っていたのに、結局今日1日このレポートにかかりっきりだよ……。

ガクさんは自分に求められていることがわからず、すっかり落ち込んでしまいました。



この調子だと、レポートだけであと何日も費やしてしまいそうだよ。実験結果があるからもういいんじゃないのかな。疲れてきちゃったなあ。



完璧なはずの俺のレポートがダメ出しされる件

企業でなんらかの活動を実施した際には、たいていの場合、レポートによる報告が求められます。上司からメールで「今日中に居室で作業上のリスクがある箇所を挙げてください」と指示があったなら、メールで「ブチョーさんの机のタコ足配線」と返すのも大事なレポートの1つです。

この頻繁に発生するレポートは、大学のときの論文作成とはまた異なるスキルが必要となります。新人の観点でじっくり細部までこだわり、美しい装丁までしつらえ、もうこれで100%完璧！と思えるレポートを作成したつもりでも、上司から見たら本当に書いてほしかったことが記載されていなかったり、いくつもダメなポイントが見えてしまったりするものです。「悪いけどこれ今日中に修正できる？」と、上司から差し戻される光景は、職場の日常風景ですね。

期待されているのにその期待がわからない

これは必ずしも新人の能力に問題があるわけではなく、次のような原因によるものです。

- レポートの目的を達成できていない（目的を理解していない）
- 提出形式が間違っている
- 内容が足りない

- 業務内容の理解が間違っている（正しく伝わっていない）
- そもそも自分の業務じゃなかった
- 実はもうレポートはいらない

目的不明なんてそんな馬鹿な、と思われるかもしれませんが。しかし、上司が想像している以上に、新人にとって会社からの期待というのはわかりにくいものです。例えば、金曜日にブチョーさんから「今週の成果を1行か2行にまとめてメールで送ってくれ」という業務要請があったとしましょう。1行程度ですからもんということはありません。早速新人のガクさんはメールで次のような1文を送りました。

To: ブチョー

件名: 今週の成果

今週はバグ対応しました。

すると、ブチョーさんから速攻で「全然わからん!」とメールが返ってきました。バグ対応はやったことであって成果ではありません。バグの対応をした結果、そのバグはFixできたのかどうかも不明です。

報告しろと言われると、ついつい自分のやったこと (Do) を報告しちゃうのですが、レポートを求めている側は基本的に結果 (Result) を欲しています。例えば「残バグ50件のうち今週は20件修正完了した。予定通りのVelocityが出ています」というような内容をブチョーさんは期待しているのです。

新人は会社勤めを始めたばかりですから、なかなか上司から求められていることがピンときません。きちんと期待に応えないと何度でもやり直しです。目的から外れたレポートを見ると、上司はつい、なんでこんな簡単なレポートも満足に書けないんだと叱責してしまいがちです。しかし、そもそも依頼したレポートにどんな内容を期待しているのか、実はリーダーから十分に伝えられていない場合もよくあることなのです。

ここが失敗のポイントです。レポートに関して**ゴールの条件や基準を十分に与えていないのに、繰り返しダメ出しを**することで、新人にとっては求められていることがわからず、何を改善すればいいのかもわからないまま、ひたすらトライアンドエラー

を繰り返し、時間とやる気を浪費してしまいます。やがて、自分の報告が理解されないのは上司のせいだと思ようになります。まるでいじめのように無駄な労力を強いられているだけではと思い、組織に対する不信感が芽生えます。そのうち、理解されない上司のもとではなく、もっと生産性の高い仕事ができる組織で働きたい、と思うようになるかもしれません。



ええ？ ガクさんまだレポート書いているのか。時間かけすぎだよ。レポートに対するリーダーの期待を理解できていないかもしれないなあ。少し話を聞いてみるか。

業務の目的と受け入れ基準を明確にする

そもそも業務の前には、業務の目的と受け入れ基準を伝えることが大切です。事例の原理実験研修であれば、

研修の目的：ロボット皮膚の刺激測定の原理を理解すること

レポートの受け入れ基準：「理解の程度とその根拠を示す」項目があること

ということを最初に伝えるべきだったのです。そもそもレポート以前に、目的もゴールも伝えずになんとなく業務を実施させていては、成果にたどりつけません。

また新人が周りに相談できる環境も必要です。もし上司から見て、新人が業務に時間をかけすぎていると感じたら、業務を正しく理解しているかどうか、自分が求められていることを正しく把握できているのかどうか、軽く聞いてあげるのがよいでしょう。

60%チャレンジ

業務を60%ぐらいできたら一旦提出させる、というのも1つの手です。前述の「今週の成果を1行か2行にまとめてメールで送ってくれ」という業務要請に関しても、とりあえず「残バグ50件のうち今週は20件修正完了した。」とだけ書いて出してくれれば、実はもうOKだったりします。もし上司が「予定通りなの？」と聞いてきたら「予定通り」と追記して出せばいいのです。じゃあ来週は、予定通りかどうか書くようにしよう、と書き方を修正すればよいのです。

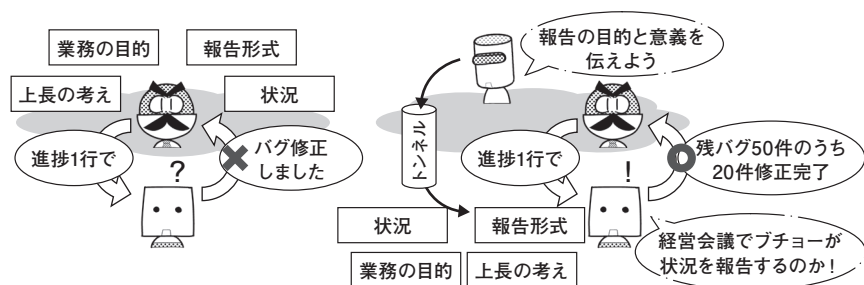


図 報告の目的と意義を伝え、相談に乗る

業務は誰かとの関係性の中で成立します。そのため何事も自分の中で勝手に作り上げた「完璧」な姿を目指すのではなく、60%ぐらいでキャッチボールを始めるというのが効率的です。意外と60%でも用事が済んでしまうケースが多いので、省力化にも繋がりますし、新人としても悩む時間が減ります。

上司としては、新人の業務に関する理解は不十分である、という前提を持って早めのコミュニケーションを心がけることが大事です。ふむ。これってちょっとAgileですよね。

まとめ



失敗

- 業務の目的、期待やゴールを伝えず、ダメ出しを繰り返すことで、組織に対する不信感を植え付けた



回避策①

- 業務の目的と受け入れ基準（ゴール）を示す

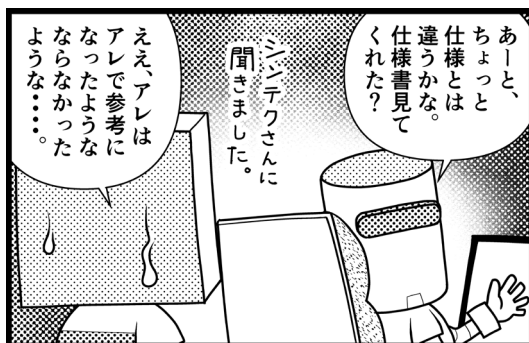
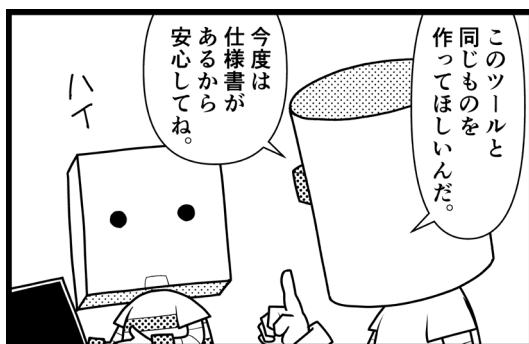


回避策②

- レポートを60%の完成度で提出させ、早めのコミュニケーションを取る

アレと同じに作って 「現物合わせ仕様」 何ができたならアレになるのかわからない

新人向けの課題を手抜きしてやっつけると、何かしらよろしくないことが発生します。仕様書もないまま「アレと同じものを作てね」とお願いしても、アレとはかけ離れた成果物が上がってきます。まだドメイン知識も暗黙知もない新人にとっては、指示された業務を理解できないまま取り組むことになり、結局自分の力でうまくこなせなかった、という自己否定感だけが残ります。



動くものがあるからできるよね

現在OJT中のガクさんですが、きちんとプロジェクトの業務をアサインする必要があるだろうと、スキンチェッカーの「校正アプリ」を担当してもらうこととなりました。本格的に製品開発の一端を任せられ、ガクさんもワクワクしています。



とはいえOJT期間中は開発に慣れることから始めよう。ここにシンテクさんが作ったアームチェッカー用の社内用データ通信ツールがあるので、これをスキンチェッカーに繋いで動かしてみようか。



早速ガクさんは開発環境を整え、ビルドを通そうとしたのですがドキュメントがありません。そのため、ツール製作者のシンテクさんに相談することになりました。



仕様書？ えーとねえ、通信プロトコルとファイルフォーマットの仕様書はあるかな。基本、アームチェッカーとスキンチェッカーの通信部は一緒だからそのまま動くはずだよ。たぶん。



えー、つまりビルド方法とか、ツールの機能仕様書とか要件定義書とかそういうドキュメント類はないんですね。



ないものは仕方ありません。ガクさんは戸惑いつつも、試行錯誤しながらコードをビルドし、実機に繋いで動かしてみました。



確かにビルドするだけで動きましたよ！ データを取得してファイルにも保存できているようです。



おお、やるじゃん。どれどれ……ん？ なんかデータがおかしい。あ、そうだった。1回で測定できるデータの数が違うんだった！

どうもビルドしただけでは想定通りに動かないようです。シンテクさんに修正箇所を指示してもらって作り直したのですが、それでもまだ問題があるようです。



アルゴリズムチームからどうもデータがおかしいって指摘がきているのよ。
いやでもこのツールはデータを引っ張っているだけだしなあ。

ガクさんは言われた通りに作業しているだけなので、いったい今何が起こっているのかわかりません。もしかしたら知らず知らずのうちに何か失敗したのかと不安になっています。なにしろ正解がわからないので、シンテクさんがコードをチェックしている間も、ガクさんはただ見ているだけです。

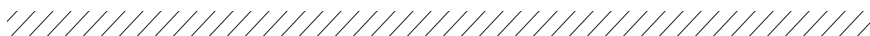


うわっ。前のアームチェッカー用のツールは、生データに少し加工してある。これが原因だけどなんで加工してあるんだ！？

ガクさんは自分の課題でシンテクさんの時間を奪っているため、いつも以上に恐縮しています。その様子を見ながらハルさんも恐縮してしまいました。



動くものがあるから楽勝と思ったのだけど、何も知らないガクさんに、現物合わせでアレと同じに作ってというのは無理があったなあ。



下手な手抜きで遅延を招く

仕様書の重要性は新人教育においても揺らぎません。むしろ業務についてこれから学ぶ技術者のためには、より重要性が高まるともいえます。

えーまた仕様書の話？ もういいよ、十分だよ。ちゃんと書いているよ、という声が聞こえてきそうですが、仕様書を新人にもわかるように書くというのはなかなか難しく、骨が折れます。

加えて、仕様書にソフトウェアのすべての動作を微に入り細に入りすべて記述することは、そもそも不可能です。どこかで記載するかどうかの線を引かないといけません。例えばWindowsやmacOSで定められている既定の動作（ドラッグアンドドロップなど）は、誰もその操作の結果が想定できるので、大雑把に書いても読み手は理解できます（ちなみにOSの想定とは動きをさせると、様々な問題を引き起こしますのでそれはそれで要注意です）。

そして、意外と陥りがちなミスとして、仕様書に「あの機能と同じ」と記載してしまうケースが挙げられます。

考えることを止めて「アレと同じ」と記載する誘惑

日々業務に追われているソフトウェア技術者にとって、やらなくて済むことは少しでも省きたいものです。無駄を省こうとするあまり「この機能はあのアプリと同じでいいんじゃない?」「もしかしたらコードもそのまま持ってこられるよね」などと考えた結果、「じゃあもう仕様書はいらないね」「とりあえず『あの機能と同じ』って書いておこう」という誘惑に負けてしまいます。

こうして仕様書を用意しなかったり、もしくは「あの機能と一緒に」としか書かなかったりすることで、次のような問題を引き起こしてしまいます。

- 違う機能と誤解する（あの機能ってアレじゃないの?）
- 機能が足りない（何ができれば全部なの?）
- 動作が悪い（ファイル開くの1分かかるけどいいですか?）
- 隠し機能がない（デバッグ情報が取れなくなっているけど?）
- そもそも同じじゃダメだった（ファイルフォーマットに情報を追加してほしかった）

仕様書を書く人が「あの機能と同じ」と書いた時点で、考えることをやめてしまっていますから、必然的に正しく要求を表現できていません。

筆者も以前、業務委託をした際、つい仕様書に「あの機能と同じ」と書いて全く同様の問題を起こしてしまいました。業務を依頼された側としては「アレと同じ」ように作ったつもりでも、その機能がいったいなんのために、どういった作業を実行するために必要なかわからないまま作るしかありませんから、成果物が十分要求を満たすことができなかったのです。ちょっと手を抜いたばかりに、結局かかった納期は想定の数倍、使った費用も倍。すなわちほぼ4倍の損失になってしまいました。

決して合わない現物合わせ

この「あの機能と同じ」という指示は、当然新人にとっても致命的です。特に新人研修期間においては、プロジェクトに直接影響の少ない社内ツールなどから、実装をお願いすることが多いと思います。この社内ツールの類は残念ながら仕様書が不十分なケースも多く、致し方なく「このツールと同じに作って」という、現物をもとにした依頼になってしまいがちです。

仕様書が不十分では、付き合いの長い委託会社ですらトンデモ成果物を出してきます。会社に入って数か月の新人が「アレ作って」と言われても、正確にアレを作るはずがありません。

ここが失敗のポイントです。新人に対して**仕様書も用意せず、現物合わせで実装を依頼すると、十分に要求を満たす成果物を作ることができません**。さらに、元の要求や利用目的が伝えられていないため、上がってくる修正依頼にも十分に対応できない状況が生じます。ゴールが見えない中で失敗を繰り返すことで自己肯定感が下がり、その結果自力で開発できる自信を喪失してしまうのです。



仕様書がないなら少なくとも何ができればOKなのか、ゴールをすり合わせるべきだったし、このツールがそもそもなんのために必要なのか、目的を最初に説明するべきだったなあ……。

なぜその機能は必要なのか

新人に対してだけではなく、業務を依頼する際は常に目的を理解してもらうことが大事です。まず「なぜ」を伝えるのです。

ツールを作ってもらうのであれば、そのツールが必要なのはなぜか。そのツールはなんのために、どういうシーンで誰が使うのかをまず相手に伝えましょう。

仮に仕様書がない状況、あるいは明確に仕様書に機能のふるまいが記載されていない状況であれば、仕様書を作るところから依頼すべきです。仕様書といっても期待値（ゴール）を合わせるため、外部から見たふるまい（要件）を書けば十分です。仕様書は最もやり直しのコストが低い作業です。ソフトウェアを作りきってからダメ出しされるより、仕様書でダメ出しされたほうがダメージは少ないのです。また新人のころから仕様書が重要かつ必要なものであり、もの作りの前には必ず用意するということ

を体で理解してもらうことも重要です。

そして、これは新人教育に限った話ではなく、あらゆる育成に必要なことでもあります。リーダーは「あの機能と同じもの」は誰にも作れないと理解をしたうえで、そもそもの作る目的と要求に関わる人全員の共通認識となるように意識し、行動しましょう。

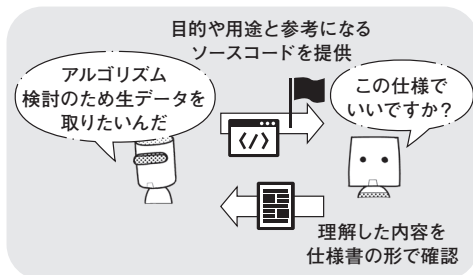


図 「なぜ」を理解してもらう

まとめ

失敗

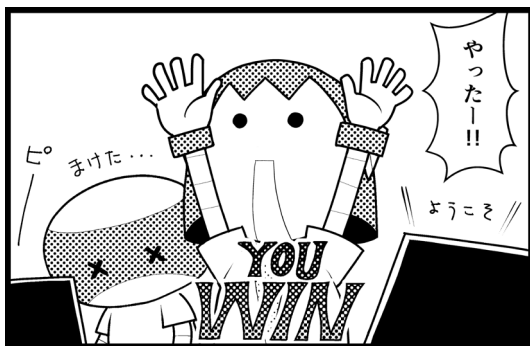
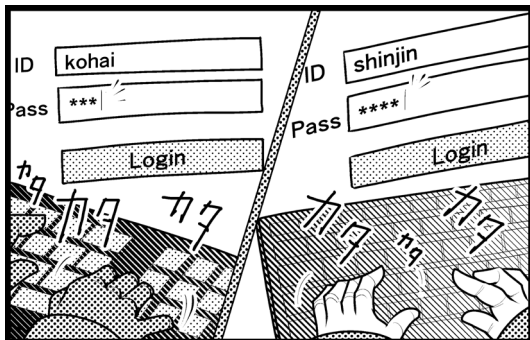
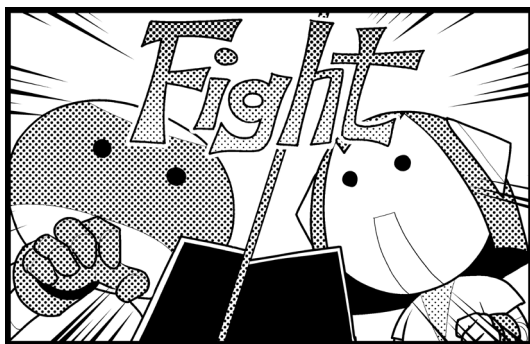
- ・ 目的も用途も告げず、現物合わせて作業を新人に依頼し、
- ・ 使えない成果物を作らせることで、新人の自己肯定感を
- ・ 低下させた

回避策

- ・ 目的を伝え、目的に合致した機能を仕様書に記載させ
- ・ 確認する

最恐マルチ3兄弟の末弟 「誰も使わない マルチユーザーソフト」 利用シーンがわかっていない

筆者が勝手に「ソフトウェア3大難問」と呼んでいるソフトウェア構造上の課題があります。マルチユーザー、マルチリンガル、マルチスレッドというマルチ3兄弟がそれです。今どきサービスを複数人で利用することは当たり前ですから、新人にマルチユーザーの仕組みを理解してもらうことは重要です。特に利用ケースの想定と要件をおざなりにしてしまうと、結局誰も使わないユーザー不在のソフトを作らせることになります。



その「管理者」って何する人？

ハルさんは、ガクさんが新人のうちに様々な経験を積むことが大切だと考え、現在取り組んでいる「スキンチェッカー用データ通信ツール」に少し難しい機能を搭載してもらうことにしました。



通信部分は安定して動作するようになったね。では今度は管理者モードと作業モードを切り替える機能を付けてみよう。作業者がつい誤って大切な測定データを消したりしないように、データを保護することが目的だよ。

生産などの業務で使うツールはポカヨケが大事です。作業者がやらなくていいことは最初からできないようにしておくことが求められます。早速ガクさんは作業モードと管理者モードを切り替えるボタンを取り付け、画面の表示を切り替えるようにしました。しかし、この後は何をどうしたらよいのかわかりません。そこで、たまたまそばにいたノビシロさんに相談を持ち掛けました。



お！ 管理者モードかー。意外に大変よね。ヒントは管理者ができて作業者ができない機能はなーんだ？

ノビシロさんのヒントでピンときたガクさん。通信ツールのデータ修正機能と削除機能を、管理者モードでしか使えないようにしました。これで作業者は不用意にデータを削除したり変更したりできないはずですよ。



惜しい！ これじゃ誰でも管理者モードに入れちゃうから、実は作業を制限できていないのと同じじゃないかしら。

確かにボタンをクリックするだけで誰でも管理者モードに入れます。管理者モードのまま運用されると意味はありません。



早速管理者パスワードを入力する画面を追加しました！ これで完成だと思うのですが、どうでしょう？

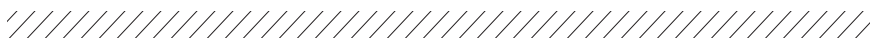


えーと、このパスワードは固定なの？ コードにハードコーディングしてあるのね。パスワードがばれたとき、管理者が変更できないのは困っちゃうかな。それとそもそもデータがCSVで保存されているから、通信ツールで機能制御してもCSVデータを直接変更できるのよねえ。うーん。ハルさんどこまでやれて言っていた？ まず要求が知りたいのだけど。

ガクさんとしてはこれでバッチリ完成と思ったのにデータ保護の観点からはぬけぬけ状態です。すっかり気落ちしたガクさんを見て、ハルさんは自分の指示に問題があったことを理解しました。



あーまたやってしまった。ノビシロさんの言う通りだ。最初に要求を伝えて要件から一緒に定義するべきだった。経験のないガクさんに要求や利用シーンの想定ができるわけがなかったのに……。



マルチ3兄弟はいつか必ず立ちはだかる

OJTを通じて新人には製品開発に必要な技術を習得してもらわなくてはなりません。特にマルチユーザー、マルチリンガル、マルチスレッドというマルチ3兄弟は、大学などでは扱う機会の少ない技術です。一方企業で開発するソフトウェアでは、このマルチ3兄弟に必ずどこかで出会うことになります。

それゆえ、新人のうちから正しく設計できるように経験を積んでもらいたいのですが、この3兄弟の扱いは非常に難易度が高いため、ベテランの技術者であっても問題を引き起こします。その中でも今回はまずマルチユーザーに関する失敗をご紹介します。

昨今、アプリケーションやサービスを自分用にカスタマイズしたり、データをユーザー各自で個別に保存したり、特定の人しか使えない機能があたりと、マルチユーザー機能は当たり前ものとなりました。

またアプリケーションもローカルで動くものだけではなくありません。メールや文章作成、表計算なども、インターネットの向こう側にあるサービスを利用する時代となりました。これらのサービスを受けるため、自分のアカウントを作ってログインする、というのが今では普通のこととなっています。

どこまでやったらマルチユーザー対応？

ごく当たり前で一般的なこのマルチユーザーの仕組みですが、実際きっちりと作りこむことは想像以上に難しいものです。

- 安全にログインできる？（多要素認証）
- 通信経路は守られている？（TLS）
- 鍵は安全に保持できている？（ハッシュ、耐タンパー性）
- 個人情報は保護している？（GDPR、個人情報保護法）
- バックアップは取れている？ 何かあったら復元できる？
- アクセス集中に耐えられる？（スケールアウト、サーバーレス化）
- 流出しない？（最新のセキュリティパッチ、SQL インジェクション等の攻撃に強いコーディング）
- ユーザーのパスワード忘れに対応できる？
- シングルサインオンできる？（SAML 認証、代理認証）
- ユーザーごとにリソースは独立している？（マルチテナント）

いずれのフィーチャーも最先端のセキュリティを理解する必要があります。技術的な問題もさることながら、セキュリティを強めれば強めるほど、利便性とは相反していきます。ちょっとデータを閲覧しようと思って自分には権限がないとか、データを取り直そうと思ったら管理者の許可がいるとか、現場で今すぐ使いたいのに使えないというケースが出てきます。場合によってはこのソフトが原因で生産性が下がるというケースも出てくるでしょう。なんでもセキュリティは強くすればいいというわけではありません。



正直社内ツールなのだから、あまりにも厳しいセキュリティ要件にしてみると、誰も使ってくれないソフトになるよなあ。

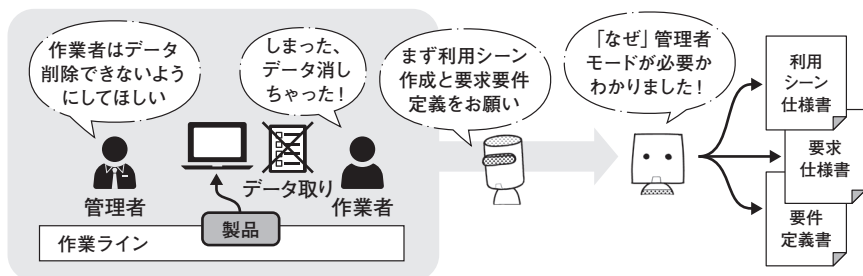
誰の役にも立たないマルチユーザー対応

マルチユーザー対応というのは、利用シーンによってはどこまでも厳しい仕様になります。それゆえ、新人に「ざっくり適当に作ってね」と丸投げするわけにはいきません。ここが失敗のポイントです。新人や若手技術者に対し、**どういうユーザーが、どのようなシーンや目的で利用するのかも明らかにせず、ふんわりとした指示でマルチユーザー対応を依頼すると、ユーザー不在の役に立たないソフトウェアを作らせてしまいます。**もしそれが製品なら、誰にも使われないため売上に繋がらず、下手をすると利用ケースの想定ミスから、情報漏洩などのセキュリティインシデントに繋がるかもしれません。どんなに高度な技術を使いこなせても、それでは意味がありません。

またこのとき、新人課題だからと利用シーンを考慮せず、安易にできたものでOKとしてしまうと、新人は自分自身で**なぜこの機能が必要なのかを考えなくなります。**新人が将来優れた技術者に成長し、活躍するためには、最初から「なぜ」に応える訓練をし、習慣化する必要があります。

利用シーンと要求を明確にする

まず新人向けの課題であっても利用シーンをしっかりと定義しましょう。「なぜ」その機能が必要なのか、その答えが利用シーンにあるからです。逆に利用シーンがないものを課題として設定してはいけません。誰も使わないものを作らせるなんて、地獄の責め苦でしかありません。



研修課題であったとしても、必ず利用シーンと要件定義から実施する。
具体的な機能が自分でイメージでき、かつその機能が「なぜ」必要なのかを考える癖が付く。

図 利用シーンを明確にする

事例のケースであれば、作っているのは社内ツールであり、ユーザーも社員です。生産時のデータ取りを社員が行い（利用シーン）、そのとき不用意にアプリからデータ削除や変更ができなければOK（要求）なので、データファイルもCSV形式で問題ないでしょう。しいていうなら不用意にデータファイルを削除してしまったときのため、バックアップを取っておくくらいの仕様でいいはずです。

ポイントは、教育目的であってもまず利用シーンと要求を明確にすることです。そのうえで要件をシンプルな機能で定義します。本質の「なぜ」を捉え、いかにシンプルに「なぜ」を解決できるか、頭をひねる訓練が大事です。

既存サービスを活用する

どうしてもフルセットのマルチユーザー対応が必要となれば、もはや仕組みを一から自分で作る時代ではありません。実現に必要なサービスやライブラリを外部から導入することを指導しましょう。特にセキュリティに関する機能は自分で作ってはいけません。世の中の熟練の技術者が長年有識者に叩かれながら鍛えあげた逸品がありますから、必ずそれを使うべきです。

その場合であっても、まずは利用シーンを描き「なぜ」を明確にしなければ、結局なんでもできる「全部入りソフトウェア」となり、失敗へまっしぐらです。

まとめ



失敗

- 新人課題として要件を明確に定義せず役に立たないソフトウェアを作らせ、「なぜ」作るのかを考える機会を奪った

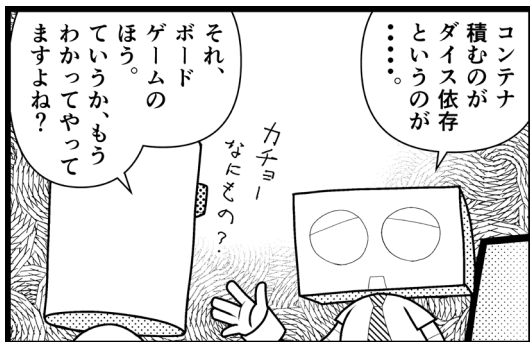


回避策

- 課題であっても利用シーン、要求要件をまず考えさせる

新しい技術が得られない

///



技術で冒険しなくていいから

ガクさんを他部署に移すというブチョーさんのたくらみを、ハルさんはカチョーさんとも結託してなんとか防ぐことができました。これでガクさんも腰を落ち着けてプロジェクトに取り組むことができます。

校正アプリも、要件定義書が完成し、設計のステップに入ることができました。ガクさんもここからが自分の力の見せどころだということで、できるだけ様々な書物を紐解き、ネットやAIの力も借りながらアーキテクチャを設計しました。ガクさん本人もなかなか面白いものができたと、勇んで設計書のドラフトをハルさんに見てもらったのですが。



設計書ありがとう。うーん力作、なんだけど……校正アプリにここまでのちゃんとしたアーキテクチャはいらないというか。もともになる通信ツールの構造をそのまま流用してくれたらよかったのに。

自分なりに頑張って考えたアーキテクチャですが、思いのほかハルさんの評価が低く、ガクさんはがっかりしてしまいました。確かにハルさんの言うように、アーキテクチャを刷新すると一から作り直しとなり、作業量もリスクも大きくなってしまいます。通信ツールの構造を踏襲すれば作業量も少なく、課題も少ないでしょう。

仕方ありません。せっかく作ったアーキテクチャ案ですが、改めて通信ツールをベースに考え直しました。再度ハルさんに設計書を見てもらったのですが、どうにもまだ問題があるようです。



設計書見直しお疲れ様。確かに通信ツールベースになっているのだけど、なんで通信にTLS使うの？ 校正アプリは社内ツールだからいらないよ。あってもいいけどパフォーマンスも悪くなるし。

ガクさんは、ハルさんからTLS通信について指摘を受ける可能性があることを想定していました。そのため、事前にシンテクさんと相談をしていました。

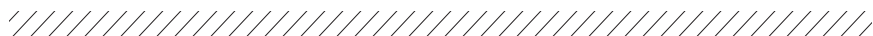


それは理解したうえで、勉強のためにトライさせてほしいのです。別の暗号化機能のためにライブラリを導入しますので、それを利用してTLS通信も実現できます。なんとかなるんじゃないかと思います。

ガクさんからのお願いに、ハルさんも内心ハッとしました。新人のうちから新しい技術にチャレンジすることを抑制し、リスクの少ない業務だけをさせていては、全く学びに繋がりません。ハルさん自身も若いころには様々な新規技術を試していたことを思い出しました。



そうだね。何か1つチャレンジすることを入れよう。ただしTLSの実装は優先順位を下げて、通常の通信ができてからやってみよう。



ソフトウェア開発老人会の憂鬱

ソフトウェアの進化ってものすごいんですね。例えばC++にしても、筆者が最初に覚えてからC++98、C++03、C++11、C++14、C++17、C++20ときて、今ではC++23が登場しています。もはや当初のC++とは全く別の言語になりつつあり、初版で覚えた自分は「C++を知っている」とはとても言えません。

開発環境も同様です。かつては「テキストエディタは何を使っているの？」などとのんきに言っていたところから、Turbo-Cのような統合環境が出てきて、これは便利じゃ爆速じゃと感動していたものです。しかし今では、AIを搭載した開発環境がとんでもなく進化し、まるでSFのように、AIバディとコーディング&デバッグすることが当たり前の世界が到来しています。

また最近ではDockerなどコンテナ型の仮想環境の登場により、開発環境の一発復元、リリース版の一発デプロイも可能になってきています。JenkinsなどのCI環境とも相性がよく、複雑な状況や問題解決においても、安心かつクイックな対応ができる環境が整いつつあります。

ソフトウェア開発においては日々新しい技術を研究し、学び、試し、実践していけば、自分の持っている知識はどんどん時代遅れになります。懐かしそうに「俺の時代はFORTRANをカードでコーディングしたものだ」みたいな昔話をしても、そりゃあ大変でしたね、で終わりです。

進化に取り残されたガラパゴス技術者

とはいえ、製品開発にとって必要な技術は限られています。むしろ製品にとってあまりに新しい技術はリスクが多く採用できません。ちゃんと味見をしてリスクを確認してからでないと、製品にいらぬ課題を持ち込んでしまいます。そのため製品開発の現場においては、枯れた技術が好まれます。製品の性格にもよりますが、スキンチェッカーのような工場で使う計測機器であれば、別にC++20なんて不要です。USBだってType-Cにする必要はありません。Wi-Fiですら「USBがあればいいよね？むしろセキュリティ的にはなしだよ！」で終わります。できるだけ実績のある技術だけで作り、冒険はしません。

しかし、こと技術者としては、このような状況は面白くありません。もちろん製品開発を通じて学ぶ点はたくさんありますが、新しい技術を全く取り入れないでいると、どんどん世間から置いてきぼりとなってしまいます。今は社内で成果を出せていても、そのうち新規技術が必要となったときに手も足も出なくなります。技術がなければ製品としての魅力を上げることができず、企業としての価値も下がり始めます。いつしか自分たちの力では製品を作り上げることもままらなくなるでしょう。

ここが今回の失敗ポイントです。今製品作りに不要だからといって、**新規技術を取り入れる努力を怠ると、組織はそのうち魅力ある製品を作る力を失います**。また新人にとっても自らのスキルを磨く機会が失われ、世の中から取り残されたガラパゴス技術者になってしまいます。5年10年がたった後、自分の力が同世代に比べて成長していないことに気が付いても、もう遅いのです。



別に転職を勧めるわけではないのだけど、技術者として市場価値が下がるような育成はどう考えてもダメだよなあ。

新しい技術に触れる機会を作る

今の製品には不要でも、技術者である限り、なんらかの新しい技術に触れる機会を、継続的に作っていくことが重要です。将来にわたって価値ある製品を作り上げるコアとなるのは技術力です。技術力のなくなった企業は他社に頼るしか生きる道がなくなります。そして技術とは人です。人が技術を持つための仕組みがなければ技術力は失われます。企業を成長させるためには、継続して技術力を取り込む施策が必要です。

15%カルチャーは使われない

技術力向上を、メンバー任せにしても効果はあまりありません。業務に追い立てられ技術獲得は後回しにされてしまいます。

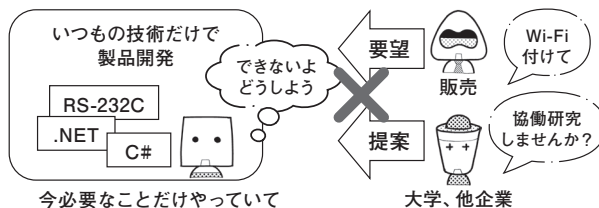
スリーエム（3M）社の「15%カルチャー」をご存じでしょうか。これは会社の成長に貢献できるテーマであるなら、業務時間のうちの15%をどのような研究や行動に使ってもよいという不文律です。同様のルールを導入している企業も多いかもしれませんが、実際のところ、いくら週の15%の時間を使っていい、なんなら必要なお金を出すと言っても、このルールを浸透させるのはなかなか難しいものです。なぜなら、他にやるべき業務で時間がいっぱいだからです。ただでさえ忙しいのに、わざわざ製品や業績評価と関係のない業務を好き好んでやることはありませんし、15%カルチャーを使うことで、プロジェクトに15%遅れが出ようものなら、逆に業績評価が下がるかもしれません。おそらく、15%カルチャーがうまく機能するには3M社独自の「カルチャー」が鍵なのでしょう。

技術獲得を業務とし評価する

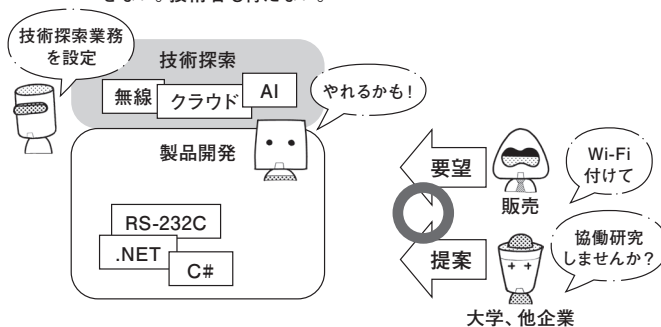
そのカルチャーがない企業にとっては、技術獲得のための時間を業務として設けることが効果的です。技術獲得自体を成果と捉え、業務として評価します。

具体的な施策としては、定期的な新規技術勉強会や、若手技術者を集めた新規技術で遊ぶ会のような機会を設けるのがよいでしょう。今ならAIを使ったり、エレキやメカメンバーと一緒にロボットを作ったりするのも楽しいと思います。

こういった施策は、部門やチームの全員が同じタイミングで時間を確保し、実施することが大事です。チーム全員が参加する形にすれば、否応ありません。一方で、自由参加にすると、多くのメンバーがプロジェクト業務を優先してしまうでしょう。誰か1人でもプロジェクト業務を行っていれば、他のメンバーも「自分もプロジェクトを進めなくてはならない」と考えてしまい、取り組みに集中できない状況が生まれてしまいます。計画的に将来の技術力を蓄える時間を作るのはリーダーだけです。



今必要なことだけやっつけても、将来必要な技術は獲得できない。技術者も育たない。



☒ 新規技術を業務として取り込む

まとめ

失敗

… 新規技術を獲得する努力を怠り、技術者の育成と企業の成長を阻んだ

回避策

… 業務として技術獲得の時間を確保し、成果に加える