SE 一年目のための仕様書の書き方　システム品質を高める文章術　宮古　環

「IT プロジェクトの成功率は 3 割以下」

なお、日本の開発現場では、**プロジェクトの失敗が明らかになると、個人攻撃をすることはあっても、根本的な問題を解決するようなアクションを起こす組織はあまり多くありません**。会社という組織にいる以上、目の前の成果よりも自己保身に走るのが人間の常です。個々の小さい問題だけなく、全体的な問題を解決する能力と、人員の裁量権がない限り、流れに飲まれてしまうのが安全でしょう。

結果として稼働が増え、成果物の品質も下がり、プロジェクトとしては赤字になるでしょうが、そういった現場の場合は往々にして赤字・黒字・成果物の品質よりも現場の人間関係を崩さないことを重視していることに失敗の原因があります。

そもそも前提条件が整っていない状態で詳細設計に入っていたのです。上流工程の設計書と異なり、様々な流れが合流する場所でもあるので、その上で流れが止まっていれば、謎のアウトプットが出てくるのも仕方がありません。なにしろ、インプットは出さないのに、進捗報告だけにこだわる人が多いのが現実だからです。

プロジェクトによってはデータベース管理者が軽視され、開発担当者と基盤担当者（OS 側）のみで、橋渡しをできるメンバーがいないケースもあるでしょう。この時、問題になるのは、お互いが使う技術的な専門用語がまったく異なる点と、基盤担当者側からは作業スコープやスキルマップ的に歩み寄ることが難しい点です。データベースの導入やインスタンスの作成、ユーザーの管理までは基盤担当者側で知識を持っているケースが多いのですが、あくまで OS 側の専門家としての予備知識であり、具体的な点に関しては基盤担当者側の用語を使って対話する必要があります。

データベースの種類こよりますが、基盤担当者側が担当できる範囲は以下の部分のみと考えて設計書作成にあたるのが賢明でしょう。

・パーティション／ディレクトリ設計
・OS から見た容量サイジング
・データベースバックアップ／リストア設計
・OS 観点でのセキュリティ設計
・拡張性設計
・データベース監視設計

また、開発担当者が担当できる範囲は、以下になります 0
・インスタンス（データベース）設計
・テーブル設計

・ユーザー設計
・データベース観点でのセキュリティ設計
・OS やネットワーク観点でのデータベースに対するセキュリティ設計
・データベース監査設計
・障害設計

単体試験一覧は、実装する機能のうち、最も小さい範囲で試験を行うものを指すケースが多いです。外資系の現場では「UNIT TEST」と呼ばれますが、意味は同じです。

性能試験の計画は、効率よく、かつ必要な情報を集められるようにしなければならないので、**一般的なエンジニアのスキルレベルでは作成できません**。なぜなら、OS から言語仕様、間に入るミドルウエアの挙動の詳細を含めた基礎をしっかりと身に付けている開発者以外は、効率的な試験計画を立てることが不可能だからです。また、クライアントへの説明が必要な場合もあるので、難しい用語や概念、挙動を分かりやすく説明できる能力も必要とされます。

実装工程は、特に説明は不要でしょう。**問題は手戻りが多発するケースが多い点**です。最悪、要件定義書まで遡って、実装までの工程で作成してきたドキュメントを修正していく作業が発生します。多くの場合、それに割く物理的なマンパワーがなく、設計書と異なる成果物ができ上がってしまいます。この点に関しては正しいアジャイル型の開発手法が変更に強いため、柔軟に吸収しやすいのですが、正しいアジャイル型で開発できる開発者が少なく、逆に悪い実績ばかり残っているので日本ではなかなか採用されません。
また、いざ実装作業をしていると設計書通りには実現できなかったり、要求を満たせなかったりといった解決しなければならない現実もあります。実装時のトラブル管理に関しては Trac や Readmine など様々なツール類が世の中に出回っているので、個人の能力に依存しないように各自で調整しましょう。

地味な作業が多い試験工程（結合試験、システム試験、不具合管理、変更管理）

著者は、基盤系（ハードウエア、OS、ミドルウエア）の結合試験で、不具合や欠陥を何度も経験しました。プロジェクトのスケジュール上、余力がある時は徹底的に事前検証を行って、不具合や欠陥を回避する手段を事前に確立しておきますが、ハードウエアやミドルウエアを事前に触れない場合は、この段階で回避する手段を

確立しなければなりません。往々にして、大幅な仕様変更を伴う「手戻り」が発生し、製造元のベンダーは対応をしてくれないケースが多いので、スケジュール的に根本解決は難しいでしょう。そういったことが発生することを前提に、基盤チームが担当するツール類は変更に柔軟に対応できるようにしたり、何度も試験をする前提で試験を楽にできる補佐的ツールを作っておいたりするのがよいでしょう。

保守運用工程は「儲からない」ため、軽視される工程ですが、最もスキルや経験が必要とされる工程です。クライアントから見ても、システムが問題なく稼働し続けることは当たり前なのと、期間も開発工程と比べ何倍も長いため、本来は最も重視すべき工程です。言われるがまま、作業をするスタンスで保守運用に関わっていると、**同じことの繰り返しが多いので向上心を保つのが難しい工程**です。しかし、ある一定以上の向上心がある開発者から見れば最も実がある工程と言えるでしょう。

IT 業界では、「言い訳」や「他人任せ」「責任転嫁」が平然と横行して、ほとんど咎める人がいないのが現実です。一見、平和な現場であっても、もう習慣や慣習として流されてしまって、その行動自体に気が付いていないケースもあります。

著者の経験では、日本大手ベンダーの主任から課長までの役職についている人の「ほとんど」がそういった人で、良識がある人に出会う方が稀でした。また、傾向として本人は「自分が被害者である」と思っているのも特徴として挙げられます。自分が原因である可能性を疑うこと自体を完全に排除しており、周囲に悪い影響を及ぼしています。しかし、本人は被害者だと思っているので、どんなに説明や説得をしてもまったく伝わりません。泥棒に「泥棒は悪いことだ」と言っても通じないのと同じ話です。

文句ばっかり一人前で、当の本人は自発的に何もしません。IT 業界には特にそんな人が多いのではありませんか。

プログラマ向けのコンテンツにその傾向が特に強いですが、「言われるまで何もしない」「言われてやったら言った人間の責任」「自分は何も言わないのが最善」とまるで自分が世界の中心と言わんばかりの話が溢れています。

しかも、それが当然、当たり前、ごく普通のことだと考えている人が本当にたくさんいるのです。

・クライアントが悪い
・仕様書が悪い
・マネージャが悪い
・SE が悪い
・プログラマが悪い
・教えてくれない人が悪い　etc，etc……

こう言って、他人が何かやったりしてくれるのを待ち受けているのです。そして、少しでも自分に都合が悪いことがあれば「○○○してくれない人が悪い」と言います。

さらにちょっとでも他人がミスをすれば、それはもう鬼の首を取ったように大騒ぎをし始めます。とにもかくにも誰かを悪者にしたいだけのように見えますが、これで自分が悪くないと思われるとでも思っているのでしょうか？これがほかの業界から見た 1T エンジニアの特徴や傾向であったりします。

人間である限り、仕方がないと言えばそうかもしれませんが、向上心がある人か見れば本当に迷惑な話です。どうにかしようにも、

・結局、何をしたいのですか？
・文句や不満があるのでしたら、代替案を出してください。
・あなたの存在価値は、私たちから見て、どこにあるのでしょうか…（うわ、キツい！）

など、と質問すると途端に口を閉じます。

●今も忘れられない経験

著者の経験で特に凄いものがいくつかあります。慰勲無礼極まりない人には、星の数ほど出会いましたが、それを上回る経験を 2 つほど余談として挙げます。

1 つは著者が海外とのブリッジエンジニアとして仕事をしていた際、ある IT 企業のサーバー構築およびデータセンタの納品を担当していたのですが、「試験がすべて終わったから、業務アプリケーションを導入してほしい」と言われて片道 3 時間以上かけてデータセンタへ行きました。業務アプリケーション一式を導入するので、準備もそれなりにして向かいました。

しかし、いざデー一タセンタ内に入ると仕様書上の機器がある場所には「空」のラックしかありませんでした。冗談のような話ですが、本当にあった話です。しかも、データセンタの方に協力していただき、センタ内を捜索したところ、それらしき段ボール箱の一群を発見できたのですが、受け取っている報告書では「ラッキンク」も「ケーブリンク」も「OS の導入」も「ミドルウエアの導入」も「単体試験」も「結合試験」も終わっているはずの機器の箱が、どこからどう見ても**未開封**でした。

あまりの出来事に 30 個ほどの対策を考えて、クライアントから順に状況を説明し、今後の対策を相談しようと試みたのですが、その IT 企業から出てきた回答は誰がどう見ても**「嘘」とわかる言い訳に次ぐ言い訳**で、クライアントを含む関係各社は今さら会社を変えるわけにもいかないので、しぶしぶ彼らの作業が終わるのを待った事件があります。この時の IT 企業の偉い方々の振る舞いは今も忘れられません。

ほかの例として、とある大規模基幹システムのフロント系の開発に関わった際の話です。著者は、PMO 兼サーバーなどの基盤関連のリーダーとしてその案件に関わ

りました。リプレース前のハード回りの設計・構築は著者がほぼ 1 人でやった上に、システム構成もほぼ変わらない案件でしたので、PMO と兼任で参加しました。業務アプリケーションの開発は、諸事情により落札案件として、別のスコープで切り出され管理されていたのですが、とある lT 企業が「できます！」「弊社の技術力は他社より上です！」「RFP の要求を何でも満たせます！」とそれはもう声高に主張し、見積り価格は最低ではなかったものの、RFP の要求を満たせるとの話でしたのでその会社が落札しました。

開発期間は 1 年弱、クライアントの厚意でかなりの人数が勤務できる作業場所（ビルの 2 フロア分）の確保など、当時としては破格の待遇で迎え入れられました。そして単体試験までの進捗報告会で上がってくる報告は立派なもので、クライアントも特に気にすることはありませんでした。

ところがプロジ工クト終盤の単体試験の計画になった瞬間に、目を疑うような事実に直面します。本来とうの昔にできていなければならない最重要の機能が何 1 つできていないように読み取れる資料が出てきました。巧妙に隠されていましたが、さすがに案件の肝となるところでしたので、関係者も一斉に「何かがおかしい」と気が付きます。それなりの役職の方々も現場のミーティングに参加して、状況確認をしましたが、その IT 企業の話では「遅延」が発生していて、重要なところがまだ手薄であるとの話でした。確認する側としてはそう言われてしまったら、一旦矛を収めるしかないので、全体のスケジュール調整をして様子見をしていたところ、単体試験は平穏無事に終わったとの報告と結果、そして結合試験の試験内容の話が上がつてきました。

すでにサーバー等の機器構築は終わっており、業務バッチも空ジョブを完成させて業務アプリケーションの導入待ちです。手が空いた著者は、「何か怪しい」と思って本腰を入れて、IT 企業の提出してきたものを確認してみたところ、IT に関わる知識がある程度ないとわからないように巧妙に内容が偽装されている疑いが出てきました。クライアントの合意を取ったところで、作業場所に現物を確認に行く調整をしたところ、なぜか IT 企業の重役もミーティングに参加すると回答が返ってきました。もうこの時点で疑いが確信に変わっていましたが、ともかく現場に向かうと、こちらはクライアントと PM を含めて 4 人しかいないのに対し、相手は 40 人近いメンバーをミーティングに参加させてくるという、正直何をしたいのかまったくわからない場が設けられました。

案の定、**言い訳に次ぐ言い訳**で、クライアントの権限で、強引に現物の業務アプリケーションを確認したところ「デモ用のモックアップ以外何 1 つできあがってない」ことが発覚しました。しかし、確たる証拠が目の前にあるにもかかわらず、その IT 企業の言い訳は止まりません。この後は、会社同士の調整になってしまったので正確なところは不明ですが、1 年半かけて作るはずだった業務アプリケーションは著者がリーダーのチームで 2 週間弱で作り上げられ、核となる部分は著者が間に合わせで作ったものが未だにブラックボックスとして動いていると聞いた時はあきれかえったものです。