ＩＴ　プロジェクト

ＳＥプロマネ愚行録

何と視野の狭いことか

何と愚かなことか

何と情けないことか

何と保身的なことか

ＰＭファクトリー　２０１７年

目次

本書の構成

ソフトウェア開発の中では、避けがたい問題が次から次へと発生してきますが、はたしてこれらの問題は本当に避けがたい問題ばかりなのでしょうか。最初からそれは仕方のないことだとあきらめているような問題はないのでしょうか。少しの常識と冷静さをもっていれば起こさなくても済んだ問題が非常に多いように思われます。

わたしたちの視野の狭さや視点のズレ、幼稚で浅はかな考え方、当たり前のことを当たり前に実行できない常識力の弱さ、少しの勇気のなさ、などによって引き起こされている問題に改めて焦点をあてることによって私たち開発者の心理・心情の弱点を知り、不要な問題の撲滅を目指したいと思います。

本書は、私たちがおかしがちな愚行について四つの視点、すなわち「視野の狭さ」、「考え方の浅さ」、「当たり前のことができない弱さ」および「一歩前に出られない保守性」について下記の４編にて構成されています。

第１編 【なんと視野の狭いことか】
第２編 【なんと愚かなことか】
第３編 【なんと情けないことか】
第４編 【なんと保身的なことか】

第１編　なんと視野の狭いことか

©いらすとや

はじめに

「目からウロコが落ちる」と言われるとおり、視野が開けることは新たな選択肢が見えてくることを意味し、さまざまな困難を乗り切る希望と突破力を与えてくれます。逆に言えば、視野の狭さは人の選択肢を狭め、孤立独善の状況に陥らせ、目前の問題への適切な対処を誤らせます。

本編においては、視野の狭さや視点の間違いに起因したさまざまな問題行動に対して１８の事例にそれぞれ病名をつけることで、それらに対する処方箋を探っていきたいと思います。

愚行＃１　：　視野狭窄（しやきょうさく）

【事例】『仕様を決める上で、表面的な解釈だけで仕様の範囲を決めてしまい、モレや考慮不足を起こすことがある』

☆これは焦りによる「視野狭窄」という病の一つです。視野狭窄は見える範囲を狭くし問題の解釈を誤らせ、人を愚行に走らせます。

日々ストレスの多い仕事を続けていると、多くの人はストレスを減らすために無意識に仕事を表面的に解釈しルーチンワーク的に流してしまうようになってしまいます。その人にとっては仕様の不明点や疑問点を持つこと自体がストレスになるのです。その結果、仕様の解釈違いやモレを発生させ、もっとストレスの多い失敗を自分で作ってしまいます。

最初にやるべきことは、自分の得意・不得意の分野に限らず、最初に仕様を見た場合にその仕様の意味を考え、提示されている仕様には必ず不明点や疑問点があるはずだという”問題意識”をもつことです。

ある事象を目の前にした場合、それを自分の目でよく観察し、それの意味するところを自分の頭で判断し、自分の行動を自分で決定するという一連の自律的な思考や行動が問題の最小化を図る最も良い方法です。

あせって表面的な解釈をしても失敗を起こし、もっとひどい目に会うだけでしょう。

愚行＃２　：　専門バカ

【事例】『顧客との雑談で、新聞や情報誌などで話題になっている時事ネタなどを話かけられてもうまく応対できず気まずくなることがある』

☆これは「専門バカ」という病気の一種で、専門領域についての知識は豊富だがその意味については全くの無知であり、ついにはひとの役に立つ仕事ができなくなるという重篤な病気です。この病気に効く薬は「教養」と呼ばれています。

教養とは、「人間生活をより高くより洗練されたものにするための、個人のたしなみ（新潮国語辞典）」であって、大学の教養課程を修了したことを意味しません。

読書の習慣や世情の色々なことについて見聞を広げることは自分の柔軟性を増強し教養を広げ顧客との対話を豊かなものにするでしょう。

愚行＃３　：　ヘボ将棋

【事例】『仕様決定にあたって、細部にこだわってしまい、全体の把握や、本筋から外れたところで過剰な仕様を決めてしまう傾向がある』

☆いわゆる「ヘボ将棋」と同じで、敵の飛車をやっと取ったと思ったら、次の一手で王手をかけられたことと同じです。大局観がなければ勝負には勝てないのと同じように、仕様の全体像やその骨子を最初に把

握しなければ、細部の仕様は決められないということです。本筋を把握すれば細部はそれなりに妥当性のある仕様に決まっていくものです。

愚行＃４　：　手抜き病①

【事例】『与えられた作業をこなしていくことで精一杯で余裕がないため、仕様の目的や意味を余り考えずに仕事に手を付けてしまいがちで、失敗ややり直しが出てしまう』

☆時間がないという切迫した状況でおこる「手抜き」という重篤な病です。仕事の目的や意味を知らないで行った仕事ではまともな結果を生むはずはありません。どこに何をしに行くのかも知らないで、とりあえずお使いに行くようなもので今どきの幼稚園児ですらやらない愚かな行為です。この愚行の大規模版が事前着手という、基本要件も決まっていないのにいいかげんな想定にもとづく開発着手なのです。

その手抜きで削減した時間は微々たるもので、そのせいで発生したやり直しの時間は長時間になったことでしょう。さて失った時間はどうしますか。

愚行＃５　：　手抜き病②

【事例】『他のファイルのデータ長をテスト中にチェックしたところ全て同一データ長だったので、Ａファイルのデータ長も固定長だと思い込んでしまったが、実際は可変長だった』

☆この問題は事実を自分の目で確認しようとしなかった「手抜き」病です。視野を狭くするどころか自分で視野を閉じてしまったということです。確認すべきはＡファイルのデータ長であって、他のファイルのデータ長は全く関係ありません。　Ａさんの出身地を知りたければＡさんに聞けばいいだけで、Ｂさん Ｃさんが東京出身だったとしてもＡさんはどこ出身だか分かるわけもありません。本件は思い込みとか勘違いというものではなく仕様書や設計書という原典を確認しなかっただけの手抜き行為です。

愚行＃６　：　片思い

【事例】『要件定義において合意したはずの仕様が、総合テストにおいて開発側の理解不足という理由などでバグ扱いされることがある』

☆これは「片思い」という恋わずらいにも似た病の一種です。

顧客担当者の能力レベルもさまざまですから、お互いに詰めた仕様でも実際に動作させて見ると予想しなかった不都合が見つかることもあります。それを一方的にベンダー側のバグ扱いにされることは問題がありますが、また一方的に顧客側の責任にすることにも問題があります。元々の仕様から変更されたのなら、その時点で新たな仕様変更要求として扱うべきです。このようなケースは普通に発生しますので、いちいち感情的にならず、顧客の現時点の最新の要求に従うべく追加仕様扱いとすればよいだけの話です。顧客との日常的なコミュニケーションを深めることで信頼関係を構築していけば、一方的に責められるようなことは減っていくでしょう。

愚行＃７　：　能天気

【事例】『終わりが見えない作業を実施しているとき、自分自身の予定が立たず、言われるままの作業となり、終わりが見えない為、モチベーションが下がります』

☆このような仕事のやり方を「能天気」（頭が空っぽ）というのでしょう。

仕事の目的、意味、範囲を着手する前に確認し、変更がある都度確認していればこのようなことは起きないでしょう。何も考えずに、とりあえず仕事を始めてしまうからこのようなぶざまなことになるのです。到着点は最初に検討をつけておく必要があります。

このような病気がメンバーにまん延しないように、リーダーは仕事の全体像を最初から把握しておき、到達点を明確に全員に伝えておく義務があります。能天気なリーダーをもったメンバーの方々にはお悔やみ申し上げます。リーダーの代わりをお務めください。

愚行＃８　：　フライング病

【事例】『仕様未確定で作業ボリュームが見えない状態での作業では、作業担当者の適切な割り当てもできず、日々のスケジュールも立てられず、目標も立てられない。いくらやっても終わらない気がする』

☆これは「フライング」病と呼ばれる最も悪しき病で、多くの下請け開発会社が罹患させられる病気です。「フライング」病の病原菌は「丸投げ」菌だと特定されています。

開発力が衰えたベンダーは開発会社というよりは商社化しており、仕様決定の能力も劣化したあげくほとんど丸投げ状態に陥っています。

この病を治すためには丸投げ菌を撲滅する必要があり、ベンダー側においては残された重要な責務である要件定義およびプロジェクトの統合管理を実行しない限りこの問題は解消できません。

下請け開発が生き残る方法は、要件定義力を身に着けるかこのような不良ベンダーから離れるかしかありませんが、当面は、ベンダーにおける要件定義に参加するような積極的な姿勢で取り組み、要求仕様の早期凍結に努力すると同時に要件定義能力を身に着けていく必要があります。

愚行＃９　：　結論を急ぐワナ

【事例】『顧客要件の実現方法を検討するにあたって、この方法でなんとなく出来そうだと思い、一度考え始めたことから抜け出せず、時間を浪費した結果、実現手段にたどり着けなかった。原点に返り別の切り口で検討したところ実現できたが、いったんこうだと思い込んでしまうと、なかなか自分自身の考えを変更することができない』

☆これは気持ちに余裕がない場合に陥りやすい「結論を急ぐワナ」という病気です。

思い込みは、決め付けの原因となり、自分の選択肢を狭めてしまいます。これはみんなが落込みやすいワナの一つで、仕事においてはいわゆるモグラたたきの状態に陥ってしまい、一向に出口の見えない状況にはまる原因の一つになっています。ある問題に対処する場合に最初に複数の解決策を同時に考えて

おくことが有効に働きます。複数の解決策候補の特質を相互比較して良いものから試していけば、そうそう失敗はしないでしょう。

愚行＃１０　：　リスク不感症

【事例】『最近は開発工程の分業化が進み、設計からテストまでを一貫して受注していないことによるリスクがある。たとえば設計工程だけを受注した場合、実際にプログラム製造を行う過程でしか見えてこない設計の問題を見落としやすい危険性が増えている。このようなリスクを減らすためには設計工程において製造以降を意識しなければならず、過去の設計ミスを設計に反映する必要がある。さらにそのノウハウを蓄積および共有する必要も出てくる』

☆開発の各工程はそれぞれに強い相互関係があり最初の工程から最後の工程に至るまであたかも一本の線でつながっている必要があります。

一つの開発組織ですべての開発工程を担当した場合でも、前工程におけるミスは工程を下るに従ってその悪影響は比例級数的に拡大していきます。 要件定義の誤りは複数個所の基本設計の誤りを生み、基本設計の誤りはさらに多くの詳細設計の誤りを生み、それらの多数の誤りはコーディングの誤りに直結し、テスト工程は間違いなく大混乱に陥ります。

そのように有機的に強力な相互関係をもっている各開発工程をコスト削減のために複数社に分割発注したらどのような結果に陥るのか冷静に考えれば誰にでも分かることです。利益増出に頭が支配されているマネジメントにはその恐るべきリスクが全く見えていないか、もしくはそのリスク排除は現場の仕事であって自分の責任ではないかのように思っているのでしょう。工程と工程の間を斧でバッサリと切れば血を見るのは当り前のことです。開発工程を分割した場合に起こる問題を理解し、それに対する有効な防止策を講じるのは現場の開発者ではなくマネジメントの仕事です。今求められているものは、強力な要件定義力と全ての工程および組織を束ねる統合マネジメント力の二つです。

愚行＃１１　：　思考錯誤症

【事例】『レビューは担当者における思い込みによるミスを排除するのに必要なことだが、すべての面を網羅したような詳細にわたるレビューができない。初期開発時は規模が大きいため、詳細まで完璧に行うと相当な時間がかかりレビューアの負担が増える』

☆これは「思考錯誤症」と呼ばれる開発者が罹りやすい思考障害の一種です。

レビューをする目的は成果物に含まれる誤りを正すことで、そのためにはその成果物はどうあるべきかという情報を前もって持っておく必要があり、その正しい姿とレビュー対象物を比較することでしか誤りを発見することも正すこともできません。

多くの設計製造レビューの実態は、そのような比較すべき正しい確固たる情報も持たずに、とにかく成果物の最初からすべての項目を逐次読み合わせて誤りらしきものを発見しようとするような愚かな行為が横行しています。そのようなレビュー方法では、膨大な成果物を全件チェックすることなど不可能なことで、

見つかった誤りらしきものもたまたま見つかったに過ぎない偶然の産物に過ぎないと言ってもよいでしょう。ポイントを絞ってレビューをすると言っても、何を基準にポイントを絞ろうとしているのかはなはだ疑問です。

レビュー漏れの対策として、次回からは有識者を参加させてレビューをしたいという言葉をよく聞きますが、そのように優秀な有識者が都合よくレビューに参加できるケースなど実際にはほとんどありません。

レビューの基準をもっていないチームにおいては有効なレビューは全く不可能です。レビューの基準として有効に働くものは、優れた要件定義書、過去の失敗情報および開発者における能力と経験の三つです。このような条件が必要であることもわきまえず、とりあえず読み合わせだけでなんとかレビューをするような行動は最初の思考自体が誤っており「思考錯誤症」と名づけました。

要件定義書については、もし開発リーダーが顧客の要求の骨子を把握・理解していたならば全件チェックできるでしょう。なぜそれを実行しないのでしょうか。

ボロな要件定義書は、ボロボロな初期設計書を生み出し、その初期設計書はもっと酷い詳細設計書とプログラムコードを生み出すことは誰にでも容易に理解できることです。

そんなボロボロな設計書をいくら全件チェックしたところでやりきれない以前に間違い・不明・疑問だらけになることでしょう。その結果まともなレビューも行われず、なかにはレビュー自体をスキップするなどの更に愚かなチームもあります。

有効なレビューの条件は次の通りです。

◎要件定義書は全件チェックを行い、優れたレベルに持ち上げること。

◎優れた要件定義書に従い、レベルの高い設計書・コーディングを作成すること。

◎開発の過去の失敗情報を収集しておくこと。

◎能力・経験のある開発者でチームを編成すること。

◎レビューは重要機能順および失敗の多い部分について優先的に行うこと。

愚行＃１２　：　前方不注意

【事例】『Ａ社開発において過去のＢ社の類似アプリケーションをそのまま流用したがバグとなってしまった。Ｂ社の画面遷移は二段階遷移で、上位層画面のメニューからさらに別のボタンで下位層画面へ遷移するものだったが、Ａ社においては下位層のメニュー自体が存在せず別ボタンは全く不要だった。改めてＡ社の仕様書を見たが一段階の画面のみの遷移と記述されていた』

☆この行為は交通違反でいうところの「前方不注意」で、ちゃんと見るべき仕様書を見ずに楽することばかりに気を奪われた結果の愚かな行為です。

仕事に着手する前には必ず何をやるべきかを示した根拠を確認しておく必要があります。あせって手をつけた仕事には失敗が多いものです。

愚行＃１３　：　想定開発症

【事例】『要求仕様の背景や意味を理解せず、自分の想定で設計を進めたが非常に複雑な構造になってしまった。その仕様の背景や意味を確認していれば簡単な構造で済んだはずで、途中で気が付いたが面子を考えると言い出すことができなかった』

☆これは「想定開発症」という病で、いまでも多くの開発者にまん延しており、開発者の重荷をさらに重くする結構しんどい病です。

科学的な根拠のある想定なら有効性はありますが、仕様の意味や背景という重要な根拠を確認しないような想定はどちらかと言えば妄想に近い思考であり、文学小説を書く場合にはすばらしい結果を生むこともありますが、応用科学技術の世界の仕事においては有害なものです。

また過ちに気づいた時点で正直に申告していれば本来の自分の面子や誇りも保てたことと思います。

愚行＃１４　：　感情過多による金縛り病

【事例】『ベンダーから評価業務を請けていますが、ベンダー側からの苦情が行き過ぎています。毎週、常駐先の全体ミーティングの時に文句を言われ、チーム全体の士気が落ちています。それでも、ベンダー側の苦情内容は、間違ったことは言っていないので反論できません』

☆これは「感情過多による金縛り病」とも言える病で、勝ち負けに異常にこだわる開発者が罹りやすい病です。世の中には「勝てば官軍負ければ賊軍」という言葉もあり、一概に勝ち負けにこだわる感情を否定するわけではありませんが、それも行き過ぎると自分の思考は停止し、負け状態からの復活の方策さえ考えられない状況に陥ってしまいます。

自分たちの業務品質の悪さが強烈に非難されており、さらに相手の非難の正しさを認めているのですから、何か次の改善策の一言が欲しいものですが、それが「チーム全体の士気が落ちています」だけでは、愚かであるし情けないことです。

愚行＃１５　：　分離分業病

【事例】『内部結合テスト以外のテストフェーズの品質向上に向けて指摘や指導ができていない。現状、各テストフェーズが単独で動くチーム編成になっており、全体のテストを管理し、役割分担をする工程がない』

☆これは「分離分業病」と言われ、孤独を嗜好する開発者が好んで罹る病です。いったん罹ると助かるものも助からない進行性の組織的なマヒを引き起こす重篤な病です。「統合管理」と「情報共有」という薬が効果的です。

仕事の分業化が進められた結果いつも見落とされるのは、これらの分割された仕事を統合管理するという仕事です。分担を決めたからそれで終わりというわけにはいきません。みんながチャントやってくれれば問題は起らないという考え方は現実を無視した考え方です。分割された仕事においてはみんなに見落と

される問題がほぼ１００％発生します。不完全な成果物が次の工程に何の注意喚起もされずに流されることで多くの後戻りが発生し、全ての工程において多くの時間と労力が浪費され製品の品質は低下する一方です。テスト工程のみならず開発全工程にわたる血の通った統合管理を実行する必要があります。

有効な分業のあり方は分断による分離分業ではなく統合管理を伴った共有分業にあります。

愚行＃１６　：　思考マヒ

【事例】『評価テスト工程でのテスト項目消化という単純作業は退屈でだんだんとやる気が低下してくる』

☆これは「思考マヒ」という病で、作業が単純なのではなく当人の頭の中が単純になってしまい、ついには深い眠気をともない、手抜き作業を誘発する病です。

そんなに単純な作業ならテストツールによる自動化をすればよいだけの話ですが、そのような知恵も浮かばないくらい頭がマヒしてしまっているのでしょう。

テスト項目の消化と言っても、各テスト項目の内容はそれぞれ異なっており単純な作業ではないはずです。それぞれのテスト内容の意味や背景について何も考えることなくチェックリストの字面だけを追ってテストしているだけではないのでしょうか。もしそうなら退屈な単純作業になってしまい、テスト業務という業務の品質自体が低下してきます。また一連のテスト内容には相互関係があり全体として、ある統一的な業務が正しく行われているかどうかという配慮も必要になってくるでしょう。

テスト業務に限らず、仕事内容の全体像を常に意識した上で、あるべき流れに沿った仕事の進め方をする必要があります。自分の仕事がマンネリ化してきたと感じたならば、何が原因でそう感じているのかをよく考え、改善の一手を打たなければいけません。

愚行＃１７　：　一人合点

【事例】『単体テストを担当した際に、意図した計算がされるようにテストデータを作成し計算値結果の計測の準備をした。しかしながら本来必要なテスト内容は、日付の変更によって元々ある前日および当日の実績データに対して正しくアクセスが切り替わるか否かということだった。作成したテストデータ作成は無駄な作業となってしまった』

☆これは「一人合点」という根拠のない先読みの癖をもつ開発者が陥りやすい病気です。自分が行うべき仕事の意味や目的を確認することなく自分勝手な思い込みに従って仕事を進めてしまい、その結果時間と費用を浪費することになってしまいます。

視野が狭いことは正しい目標を見誤らせますが、この「一人合点」は自分で勝手に目標を妄想する点において更にたちの悪い悪癖だと言えます。

愚行＃１８　：　集団パニック症候群

【事例】『緊急障害対応となると、チームのコントロールよりも障害対策が優先される結果、誰が何を行うのかわからず混乱してしまうことが多い』

☆これは「集団パニック症候群」といわれる病気で、突然発生した重大障害を目の前にしてチーム全員が冷静さを失い右往左往した結果、さらに大きな二次障害を起こすことも珍しくはありません。単一の障害でも混乱するようなチームなら、同時に発生した障害が四つも五つもあれば短時間に障害を終息させることは絶望的で長期間の対応が必要になることもあるでしょう。

集団パニックを起こさないためにはリーダーの冷静な問題分析力、的確な判断、および素早い指示・統率が必要となります。普段の開発行為においてこれらのことができていないリーダーにおいては、緊急時における対応は全く期待できません。平時における問題対応と緊急時における問題対応の内容は基本的に同じで、違いは対応行動の時間の長短にあるだけです。

問題対応の原則は以下の通りです。

①障害に関する情報・データを収集する。
②収集した情報・データを分析する。
③分析内容から障害の原因を特定する。
④対策を打つ。

リーダーは、全員に全ての情報を共有させつつ、上記の手順ごとに各担当者に対して適切に作業を割り振り、段取りよく対策を指示する必要があります。

第２編　なんと愚かなことか

©いらすとや

はじめに

本編はソフトウェア開発におけるわれわれの愚かな思考や行動に焦点を当ててみました。

人間の愚かさについて先人たちが述べたいくつかの言葉を紹介したいと思います。

○「愚かな人の常として、大きな事ばかりを望んで小さなことをなおざりにし、できもしない事に気をつかい、目の前にあるやるべきことをやろうとしない」（二宮尊徳、福住正兄著、二宮翁夜話）

○「不確かな状況に直面したとき、人間は情報を丹念に評価したり、関連のある統計データを調べたりしない。代わりに、「知的ショートカット」（mental short cuts）に判断をゆだねるのだが、そのせいで、しばしば馬鹿げた判断を下すことになる。このショートカットは、検討を速く行うというわけではなく、検討を全くやめてしまうというものだ。 カーネマン教授らが特定したさまざまなバイアスや盲点は、人間の愚かさの症状ではない。それらは人間性の本質的な要素であり、長い進化を遂げてきた脳が持つ、避けがたい副作用といえる。」（Jonah Lehrer、「人間の非合理性」を科学する、訳：ガリレオ -高橋朋子/合原弘子）

さらに個人の集団である組織において起こりやすい愚行、すなわちみんなで誤る集団愚行について海保博之・宮本聡介の両氏はその著作「安心・安全の心理学」の中でおおよそ次のように論述しています。

『人間が集団化した場合には集団極化現象、すなわち一人で何かを決めるよりも、みんなで決めるほうが極端な決定を導きやすいという現象が現れやすくなる。極端な決定には、極端にリスクが高い選択肢を選ぶこと（リスキーシフト）と極端にリスクが低い安全な選択肢を選ぶこと（コーシャスシフト）の二つがある。

このような集団極化現象が発生する理由は次の通りである。

①集団で決定することによる責任の分散化。

②集団討議のなかで知識の共有化が行われ、あたかも不確実さが減少したかのように感じられること。

③危険を恐れず前進することが社会的に高く評価されやすいこと。

④リーダーによる主張的意見の表明に引きずられること。

さらに集団全体の思考停止状態が招く集団愚行、すなわち集団で物事を決定するとき、決定内容が極端な方向にシフトするだけでなく、信じられないようなおろかな決定が下されてしまうことがある。 集団愚考は反対意見が言えないような凝集性が高い集団で発生しやすい。

組織集団のリーダーが注意すべき点は以下の通りである。

①中立を保ち、自分の主観的な意見を述べないこと。

②疑問や異論が積極的に言える雰囲気作りに配慮すること。

③集団内の大勢意見にあえて挑戦する役割のメンバーを入れておくこと。組織内に逸脱者の存在を認めることは、十分な社会的意味がある。こうした逸脱者を悪魔の代弁者と呼ぶ。』

（『安心・安全の心理学』リスク社会を生き抜く心の技法４８、海保博之・宮本聡介著 新曜社刊）

わたしたちの愚行につける薬があるのかどうか分かりませんが、開発現場にころがる愚行について検証してみたいと思います。

愚行＃１９　：インターフェースに失敗しました

【事例】『相互の担当部分を口頭確認で終わらせていたため、インターフェースの認識が両者間で違っていた』

☆文書によらない口頭だけの確認は誤解や漏れの原因となる

この失敗は機能どうしのインターフェースの失敗の前にすでに人間どうしのインターフェースに失敗している端的な例です。 文書によらない口頭だけの確認は必ず誤解や漏れの原因となることは経験上みんな良く知っているはずです。他の機能とのインターフェースに関しては一見簡単そうに見えても複雑な場合が多く、両者で読み合わせを行えば必ず疑問点や不明点が出てくるものです。それにもかかわらず過去に開発したものと同じようだからとか、自分は分かっているからとか、打ち合わせはめんどうだとか言うような雑な考え方は早々に捨てた方が身のためになるでしょう。

口頭確認だけで済ませて失敗しましたなどということは恥ずかしくて他人には言えない愚かな行為だと言えます。 インターフェースに限らずあらゆる開発行為は設計書や仕様書などのドキュメントに基づいて行うことを習慣化しなければ同じような問題をいつまでも起こすでしょう。

愚行＃２０　：部下の意見を十分に聞けません

【事例】『信用・信頼できる部下の要件は、指示した仕事に対してスケジュール通りに行ってくれること、指示した仕事に対しプラスアルファをしてくれること、不明点など自分から相談してくること、足りないことを助言してくれること、相談ができること、など指示した仕事をきちんとやってくれることにつきますが、現実的には指示する側の自分の問題として、相手の意見を十分に聞けず、自分の意見ばかりを言ってしまいます。フォローしたつもりでも実際にはうまくフォロー出来ていません。コミュニケーションが不足しており、なかなか相互の信用・信頼関係の構築ができません』

☆指示は相手の疑問点・不明点の意見を聞いて初めて完了する

そんなに何でもやってくれるデキタ部下がいたらお目にかかりたいものです。最初からこんなに自分の都合のよいことばかりを考えているから、部下の意見や疑問を聞こうというような姿勢も持つことができないのでしょう。このような姿勢のリーダーは少なからず存在しており、これはコミュニケーション不足というような問題ではなく、このリーダーにおける人間性の無知と自己中心的な我欲の問題でしょう。過ぎた自己中心性は人を愚行に走らせます。まずは自分の指示した内容を部下がきちんと理解したかどうかを相手に復唱させ、さらに疑問点や不明点について聞き出すくらいの丁寧な対応を実行する必要があります。

愚行＃２１　：仕事の丸投げに対抗できません

【事例】『顧客担当者における仕事に関する丸投げ的な発言や行動には、仕事に対する責任感が感じられず、丸投げされた側の自分のモチベーションは著しく低下してしまいます。一緒にリスクを負ってもらえるように、顧客を巻き込む知恵、話術を身につけることが必要だと考えるのですが、顧客の担当者の違いや、仕事の状況、背景、個人的な受け止め方の違いもあり、どうすれば良いのか分かりません』

☆どこまでも譲れるものではないと言うこと

丸投げをする顧客側の問題が一番の問題ですが、ここではそのような不条理さに対抗できない下請け側の問題についてだけ触れます。

これは知恵とか話術の問題ではなく、自分が感じている正直な意見を言う勇気がないだけの話でしょう。相手の責任感が薄いと感じたら、そう感じた根拠を言えば良いし、一緒にリスクを背負って分担して作業をすべきだと思ったら、そう伝えれば良いだけのことでしょう。

下請けの立場としては、今後の報復を恐れてなかなか発言の勇気を出すことが難しいとは思いますが、何も言わなければ一生ずっと丸投げされるがままの開発人生を送るしかないでしょう。もし一生奴隷根性で生きることが嫌なら、不条理な仕打ちに対しては、「どこまでも譲れるものではない」と言う気持ちを常にもち、少しの勇気を出して言うべき時に言うべきことを事実的な根拠をもとに発言する必要があります。

愚行＃２２　：簡単な印字変更にて単純な不具合を多発させました

【事例】『印字変更の仕事において、調査漏れにより変更漏れや影響する他の機能の印字不正などの単純な不具合を発生させてしまった』

☆自分の頭で考える訓練を

ある特定の個所の印字変更は、それは一か所だけを修正するだけですから実に簡単なことでしょう。しかし例えば、「○○票」という印字文言を「○○帳票」に変更する場合、特定の帳票だけではなく全ての帳票の変更が必要になるだろうし、さらにある帳票においては一文字増えただけで印字のレイアウト崩れが起きることもあるだろうし、さらには印字の変更により表示の変更の必要が出てくるのかも知れません。

変更の影響を深く考えもせずに、特定の箇所だけを修正し修正完了しましたなどと言うのは草野球レベルの話で、プロにおいては初心者だといっても許されるレベルの話ではありません。もっと自分の頭で考える訓練が必要でしょう。

愚行＃２３　：似たような二つのクラスの存在による失敗

【事例】『ライブラリでＡクラスとＢクラスという似たクラスの二つが存在したが、両者は同等なものだと考えＢクラスの方だけを採用したが、両者はアクセスできるメモリー領域が異なっておりＡクラスも必要だということがレビュー時になって判明した』

☆”なぜ？”の問いかけの実行を

この問題における本質は、知識や験不足などではなく、ＡとＢという非常に似た部材があるということに気づいたのに、「なぜＡとＢがあるのか？」という問いを最初に発せられなかったという点にあります。誰かに聞くか、それらの部材を自分で調べてみればその違いはすぐに分かったことです。常識的に考えてライブラリの中にまったく同じ機能のクラスが二種類あることなどあり得ません。

「まあいいだろう」というような情緒的な感情に流されずに、知らないものや分からないことに対して、「なぜ？」という問いかけを自分自身または他者に発する習慣を身に着ける必要があります。

愚行＃２４　：仕様の想定違い

【事例】『仕様が不明確で、仕様を決める担当者に確認せず人から聞いた内容から想定し開発を進めた結果、本来要求されている仕様と違っていたことが判明しやり直しとなってしまった』

☆仕様は想定するものではなく確定するもの

そもそも仕様を想定するという考え方が理解できません。仕様は想定するものではなく確定するものでしょう。多分こうだろうと自分勝手にある状況を想定したとしても神様でもない限り顧客が本当にやりたいことと同じ結果になることなど常識的にあり得ず、大外れの結果になるだけです。時間が切迫しているからこそ、やるべきことは仕様を想定することではなく顧客と仕様を早く詰め、それに従った開発を実行することが最も正確で速い方法です。

想定開発の大規模版が、いわゆる事前着手開発と呼ばれているもので、基幹仕様が決まってもいないのに開発メンバー全員がバラバラに穴だらけの仕様を各自勝手に解釈し開発を始めてしまうことがあちこちで当たり前のように行われています。これでまともな製品ができるわけもなく、実に愚かで残念なことです。

愚行＃２５　：仕事量を減らしたいという誘惑に負ける

【事例】『少し気になる処理があり担当者に確認した所、既存のままでも良いと言われたのでそのままにしていたが、後工程で別の担当者から障害として扱われ修正することになった。出来る事なら修正量を減らしたいという思惑が失敗のもとになってしまった』

☆気になる処理・動作の可否は文書による確認を

技術者としての自分はおかしいと判断したのに、楽をしたいというもう一人の自分が更なる確認を怠ったということでしょう。減らせる仕事とは、不要な作業だけであり、必要な作業は減らせません。楽に流れる

ような仕事の仕方は結局楽ではない結果を生むことになります。

愚行＃２６　：あいまいな指示による勘違い

【事例】『不具合修正の際に、Ａ欄とＢ欄の表示文字数がことなるため、同じ文字数にするように指摘され、Ｂ欄をＡ欄と同じになるように修正したが、実際はＡ欄の修正が必要だった』

☆仕様の記述は論理的に

「Ａ欄の表示桁とＢ欄の表示桁を揃えて」という指示もいい加減ですが、どちらの桁に揃えるのかを確認しない方もずいぶんといい加減なものです。とても技術者同士の会話とは思えません。

例えばＡ欄は５桁表示でＢ欄は７桁表示であった場合、Ａ欄を７桁表示にして、と指示するのが適切な指示の仕方であり、指示された方は何桁表示に合わせるのかという質問をすべきでした。技術者同士の会話において数値が出てこない会話は愚かな失敗を招く原因になっています。

愚行＃２７　：他人に無関心です

【事例】『チームにおける連携や連帯は、プロジェクトを失敗させない為、人材を育て守る為、よりよい成果を生み出す為に必要だと思いますが、多忙であるとか、自分の能力不足であるとか、基本的に他人事に無関心な自分の性格などの理由で、なかなかうまく行きません』

☆まずは自分の意思で一歩を踏み出すこと

チームにおける連携・連帯とはお互いに仕事で必要な情報を交換することであって、仲良しクラブをつくることではないでしょう。初心者や内向的な技術者はこのことを理解していないために、仕事に必要な情報の交換を避ける傾向が強く、その結果自分のみならず他の技術者にも迷惑をかけてしまうことが増え、自分の内向的な傾向をさらに強めるという悪循環に陥ることがあります。 相手に良く思われたいという過剰な気持ちは一旦脇に置いておき、仕事の進行上必要な情報の交換をたんたんと実行すれば良いだけのことです。 自分の感情や性格に悩む力を仕事の問題を解決する方の悩みに転換することが最も良い解決法でしょう。

愚行＃２８　：仕様凍結前に先行着手を行っている

【事例】『納期を確保するために仕様凍結前に先行着手を行うことで、中途での仕様変更の多発および他の仕様への影響などにより、大きな手戻りやミスが発生し開発業務が大混乱に陥ってしまうリスクがある』

☆重要な仕様から順に凍結・開発を行うこと

大きな手戻りが発生したりプロジェクトが大混乱に陥ることは結果であり、リスクは仕様凍結前に事前着手せざるを得ないような短納期を受け入れたり、もしくは強要されてしまったことにあるということに気づく必要があります。仕様が決まっていないということは開発着手ができないということを意味しており、それ以

上でもそれ以下でもありません。普通の頭で考えればそれは当り前のことですが、それを何とかしようと考えることは異常なことであり、必ず異常な結果を迎えることになってしまいます。仕様未決定で先行着手したプロジェクトにつける妙薬は、バカにつける薬がないのと同じようにありません。

リスクはそのような不可能な納期を強要する顧客および受け入れてしまう開発側の責任者にあります。事前着手という非合理的状況を回避する方法は、仕様決定をしかるべき時期までに完了させること以外にはありません。仕様決定能力が不足しているとマークされている顧客の仕事においては早い時点からの仕様検討開始および共同仕様検討などが必要であり、長めの仕様検討時間の確保が必要だということです。

愚行＃２９　：何度もミスをする後輩社員

【事例】『口頭ではなくメールや議事録に基づいて指示をしたのにもかかわらず、後輩社員がちゃんとやっていなかったり、間違っていたりが何度もあると、その都度、丁寧に指摘しなおしてあげることができず、ついつい頭に血が上り語気を荒げてしまうことがある』

☆原因はあなた自身なのかも。相手が理解できるレベルの説明および指示が必要

相手が普通レベルの理解力をもっており、過去の他の仕事では相応のレベルの結果を出していた場合、この問題は後輩社員の問題ではなく頭に血が上りやすいリーダー本人にあるでしょう。部下に指示を行う場合、その部下のレベルに合わせた説明の仕方が必要であるし、さらに当人が指示を正しく理解したかどうかをその場で確認する必要があります。マズイ指示の仕方を何回繰り返しても相手が理解できないのは当り前のことでしょう。

愚行＃３０　：時間に追われて目標が立てられない

【事例】『実作業では作業を細分化し明確な目標を立てていきたいが、時間に追われ明確な目標も立てずに作業を開始してしまうことがある。また作業の精度を妥協し、これぐらいならいいだろうという甘えが出てしまうことがある』

☆目標を立てないから時間に追われる

時間に追われて目標が立てられないような仕事とはどのような仕事なのでしょうか。一秒を争うような仕事はソフトウェアの通常の仕事には存在しません。通常の担当者レベルの仕事における作業目標の設定には数時間、長くても半日程度あれば十分なはずです。

仕事の量と質を把握し、最適な段取りを計画し、ＱＣＤの目標を設定し、完了目標日を決めておくことが、仕事を最も早く正確に進めるための最も合理的な方法です。

気ばかりあせっているからとりあえず仕事に着手してみようなどと愚かなことを考えるのでしょう。そのようなことをするから最後になって精度を落とすような手抜き仕事をせざるを得ないようなはめに陥るのです。

愚行＃３１　：先行着手という悪癖

【事例】『開発開始時に、関係者全員に開発の目的や全体像の説明が行われることが少なく、また開発者全員での要求仕様の読み合せも余り行われていない。このような場合、開発の目的や仕様の理解が曖昧なまま作業に入ってしまうので、各自がバラバラに割り当てられた作業にのみ注力してしまい、開発全体を視野に入れた設計・製造・テストができなくなるリスクが非常に大きくなってしまう。結果的に結合レベルの不具合が多発することになる』

☆基幹仕様未決定で開発に着手してはいけない

いつも時間に追われている開発者たちにおいては、とにかく仕様が分かっているところから開発を始めなければ間に合わないという強いあせりの気持ちに支配されています。それを表す代表的な言葉が”先行着手”という言葉です。一見もっともらしい言葉ですが、その実態は、とりあえず部分的にでも分かっているところから仕事を始めてしまうということで、仕様の基幹部分が決まってもいないのにもかかわらず各担当者がバラバラに開発作業に着手してしまうということです。このようなやり方をやった挙句の果てが開発のやり直しや手戻り作業となり、短い工期を自らの行為によって更に短くするだけでなく、取り返しのつかないつぎはぎだらけのソフトウェア構造を作ってしまいバグを多発させる結果を招いています。何を作るのかの骨子が決まっていないのに何を作ろうとしているのでしょうか。開発の最初の工程である要件定義工程は、どのような目的で何を作るのかを決める工程です。このことが全く不十分にしか実行されていないため、開発開始にあたって誰もその目的も要求仕様も説明することができないでいるのです。そのようなリーダーは”とりあえずやって見よう”というだけの、何も具体性のない全くの情緒的な思考しか持ち合わせていないようです。 プロジェクトを成功させるために最初に実行すべきことは、その開発の目的および、顧客が望む仕様を明確に定義すること以外にはありません。もういいかげんに先行着手という悪癖をやめるべきでしょう。

愚行＃３２　：自己チェックを省いてしまいます

【事例】『開発工程が押し詰まった時に、自己チェックを真っ先に削ってしまう。作成したボリュームに対してもともと見積りが甘い場合が多く、いつも時間不足になってしまう』

☆無駄な時間の排除を

これでは作りっぱなしということになります。その結果、バグの多いプログラムとなり修正に多くの時間が必要になっていることでしょう。自己チェックをきちんと実行した場合よりも多くの時間を費やすことになっているはずです。自分の時間だけではなく評価担当者の時間も奪う結果も招いています。見積りの精度を上げると同時に自分における開発業務における無駄な時間の把握とその改善に努める必要があります。業務における無駄な時間の把握に必要なポイントは次の通りです。

・仕様の不明点や疑問点を解消しないで想定による開発による後戻りがないか。
・仕様を完全に理解して設計に着手しているか。
・過去の設計ミスや頻発するミスを自分も繰り返してはいないか。
・重要な仕様の順に開発を進めているか。

愚行＃３３　：すぐにコードに手をつけ失敗する

【事例】『詳細仕様書をよく読まず仕様概要書だけを読んで実装を行ったが、不具合を発生させてしまった』

☆あせって手を付けたものは失敗が多い

多くのプログラマーにおいては、早くコードを作って早く仕事を終わらせたいという強い焦りが見られます。その焦りが仕様の理解不足と勘違いを生む原因にもなっています。自分における仕様の理解が済んでおらず、また仕様が未決定の段階にもかかわらず、みなさんすぐコードを書きたがります。また不具合修正においても同様に、あせって検討不十分なままソースに手を付けてしまう人が多いようです。この問題の本質は”焦り”にあると言えます。

焦ったあげく、間違った設計・製造・評価によって、ただでさえ少ない開発時間やコストの中から、更に多くの時間やコストを失っています。冷静に考えれば、何を開発すべきか明確でないのに、何を作るつもりなのでしょうか。開発は、「多分」とか「・・・のはず」だとかいう情緒的な判断に拠って着手すべきものではないでしょう。このあせりの気分は開発の事前着手という問題をも引き起こしており、失敗コストの大きな要因となっています。

不要な焦りで自分を追い詰め、間違った開発を行うよりも、多少の時間がかかったとしても、仕様の深い理解やその意味および背景を納得してから開発にとりかかった方がどれほど開発期間やコストの削減に役立つことでしょうか。

まず要求仕様や問題を目の前にしたら最初にやるべきことは、その仕様や問題についての背景や条件および意味について考えることです。課題や問題に正しく対処するためには、その本質を正しく把握し、押えるべきところはきちんと押さえることです、そうすれば開発行為自体は最大速度で仕上げることが可能となります。焦ってとりあえず手をつけることと事前に本質理解を済ませて開発に着手するのとでは、どちらが得なのかは明らかです。

愚行＃３４　：レビューをスキップすることがあります

【事例】『納期に間に合わないために、単体テストとテスト結果レビューの時間がいつも足らない。原因は、前工程の遅延にあり、そもそも結果レビューの時間を取っていない場合もある。遅延理由を検証し、納期の調整等を行うべきだと思うが、ほとんど無理な場合が多く、ガンバロー的な精神論だけで、なし崩しになる場合が多い』

☆レビューをスキップした場合の弊害

単体テストの結果をレビューしないということは、各自がそれぞれに単体テストで不具合と判断したものをメンバー間で情報共有することもなく、不具合の傾向を分析することもなく、各自でばらばらに修正するだけだと言うことです。それでは次の結合テスト工程においては多数の深刻な不具合が発生することになります。それによって、コード修正・単体テスト・結合テストのやり直しが必要になり貴重な時間をさらに大きく失うことになります。このプロジェクトはきっと大きな時間ロスをカバーするために深夜残業や休日出勤

を行い、それでも間に合わなければ総合テストが未完のままリリースを強行し市場で深刻なトラブルを多発する運命にあります。

単体テストレビューのわずかの時間や日数を惜しんだ結果です。なすべきレビューをスキップするのときちんと実行するのとでは、どちらが得なのか明白です。

愚行＃３５　：口約束

【事例】『よくあるケースとしては不具合発覚時における製造側責任／仕様側責任の押し付け合い問題がある。そういったケースでは”言った、言わない”でもめてしまい、自分においても自信が持てない状況になってしまう』

☆重要な取り決めは文書にて相互の合意を

文書、設計書、お互いのやり取りのメール文書は重要な証拠となります。特に仕様に関することや約束日時等の重要な約束事は必ず議事録や仕様書などの合意・承認を取っておく必要があります。特に緊急対応時には口頭の指示・依頼だけで済ませがちですが、必ずメールや文書等での指示・依頼を取り交わしておく必要があります。文書ベースの仕事のやり方を習慣化して下さい。仕事における約束事はすべて契約意識をもって文書によって進めるべき事柄です。

愚行＃３６　：目的や背景の説明がない依頼

【事例】『目的の説明がないまま作業を依頼されたり指示されたりすることが時折あるが、それでは自分が取り掛かった作業が本当に正しいのかどうか不安で、作業自体に対する創意工夫もできない』

☆意味の分からない行動は間違いや無駄の元凶となる

仕事を指示された時点で、指示者に必ず目的や背景および意味を確認することを習慣づける必要があります。目的が分からない仕事は極端に効率や品質が落ちてしまいます。また逆に自分が他の人に作業の依頼や指示を行う場合には、作業の目的・意味・背景についてきちんと説明をする必要があります。

愚行＃３７　：レビューや振り返りの習慣がない

【事例】『レビューや振り返りの習慣化がなかなかできません。原因としては、時間が無いことや、目的が明確になっていない為、多くの人は必要性を感じていないことがあります』

☆レビューや振り返りがなければ同じ失敗を繰り返す

時間がないとか目的が不明確とか言う理由は表面的な理由に過ぎないでしょう。本当は、やっても効果がないレビューや振り返り会議ばかりだったので、結局時間がもったいないし、やる意味も無いと思っているのではないでしょうか。レビューは、自分たちの行動に過ちがないかどうかのチェックであり、振り返りは、失敗した行動の真因を突き止め、再発防止策を考え、他のチームとその情報を共有するためのものです。前者は失敗の事前防止活動であり、後者は事後防止活動です。これらの意義は、改めて人から説明

されるまでもなく開発行為に限らず、安心安全を実現するためのあらゆる人間行動の基本だと言えます。
　時間不足だからこそ、余計な失敗によって時間と工数を失うことは許されません。時間不足だからこそレビューや振り返りが必要だとは思いませんか。その日暮らしの自転車操業から卒業したければ、効果的なレビューや振り返りを行う必要があるのではないでしょうか。今までやっていた効果が薄いレビューや振り返りの原因を突き止めて改善する必要があります。

愚行＃３８　：顧客提示の方法で修正を行い失敗しました

【事例】『顧客から提示された方法で仕様修正を行いましたが、総合テストにて不具合が出てしまいました』

☆自分の目で観察し、自分の頭で考え行動すること

　顧客の担当者がどれほどこのシステムに詳しいと言っても、所詮開発に関しては素人であると言う認識をもつ必要があります。依頼を受ける開発側にとって大事なことは、他人の見解を鵜呑みにするのではなく、要求課題に対して、常に自分の目で観察し自分の頭で判断し、行動するという基本を守ることが効果的な仕事のやり方だと言えます。 例え、一見簡単に思える仕様であっても、油断せずに必ず自分自身で内容を確認しておくことが必要です。あらゆる問題に対処するにあたってはこの「自律性」が有効に働きます。

愚行＃３９　：誤解を招く仕様書の日本語記述

【事例】『仕様書に「ＡとＢ以外の場合は～を行う…」という記述がありましたが、本来の要求は「Ａ＆ＮＯＴ（Ｂ）」であり「ＮＯＴ（Ａ＆Ｂ）」ではありませんでした。その結果修正となりました』

☆論理記述による仕様書を

　日本語文章はあいまいさを多く含んでいるため、論理（ロジック）を記述するには余り適した言語ではありません。仕様書の記述は事例文にあるように「Ａ＆ＮＯＴ（Ｂ）」と言うような論理式を多用すべきです。つまり数学やロジックの記号である（＋、－、÷、×、（　）、／、NOT、AND、OR、NOR）などを頻繁に使用した仕様記述にすることです。これであいまいな文章は大幅に削減され要求仕様書の精度は格段に向上します。

愚行＃４０　：単体テストの省略による品質低下

【事例】『開発費用の削減依頼を受け、ベンダー側担当者からの提案で、単体テストを結合テストと一緒に実施し単体テストの工数を削減する事で開発費用の削減を行いました。品質の低下が懸念されます』

☆絶対に単体テストをスキップしてはいけません

単体テスト工程を省略することについての問題点を示します。

①単体テストを省くことは技術者倫理上も品質確保の合理性上も許されることではありません。たとえば、建築予算が足りないので必要とされる柱を３０％抜いて建築してくれと要求されたら、あなたや会社はそれを仕方ない事として実行しますか？

②一度この様な悪行に手を染めると、次も同じことを実行するように必ず要求されてくるものです。今度はもっと費用を削減しろとの要求があったら結合テストもスキップして総合テストでやればいいと言うことになりませんか？それでも仕方ないと言い続けられますか？

③結合テスト工程で単体テストが実行できると本気で思っていますか？単体テストをスキップした場合、結合テスト・総合テストにおいて初歩的な不具合が頻発することになるでしょう。これでまともな開発ができるわけもありません。

やるべきことは、このような非常識な手抜き仕事ではなく、払える、払えないは別にして、まずこの開発にかかる金額の妥当性についてベンダー側が納得できるように説明を尽すことです。その上で、払えないと言われるのならば、今回の超過分は一旦ベンダー側に貸したことにしていただき、他の開発案件で上積みするような対応を相手に約束していただくことです。それもできなければ、他の案件に不足分を忍ばせておくくらいのことをすべきでしょう。絶対に製品自体が傷を負うようなことをしてはいけません。本件について金額的にすでに決定してしまったとしても、とりあえず会社の資金で単体テストは必ず正規に実行すべきでしょう。 ルール違反の要求に対しては、その危険行為の意味について要求者ときちんと話をする必要があります。

愚行＃４１　：コストの限界で顧客要望の達成を諦めてしまう

【事例】『顧客の信頼は仕事の成功、すなわち品質・コスト・納期の達成でしか得られません。しかし現実的には、コストを犠牲にしてまで顧客の要望を満足させることはできません。年々顧客の要求が厳しくなる中、自分およびメンバーの力量では両立することができない場合、顧客要望の一部を諦めざるを得ません』

☆カイゼン活動の実行を

顧客の信頼は品質・コスト・納期の達成であると言っているのにもかかわらず、実際はコストの限界で品質を犠牲にしていますと言っているのでしょうか。実際、コストの限界で多くの開発工程が時間切れとなり、それぞれの仕事が中途半端な未完成の状態で次の工程に回され、最後に市場において多くの障害を

生んでいます。これは仕方ないといって済む問題ではないでしょう。中途半端な仕事の結果が障害対応によってさらに多くのコストを生んでいるという事実に目をつぶっているだけのことです。

なぜコスト不足・時間不足になっているのか真剣に考える必要があります。多くの問題は開発の前工程に存在していることは誰でも知っていることですが、多くの人はその改善に着手しようとはしません。カイゼン活動は仕事には含まれていないという愚かな考え方のなせるわざです。カイゼン活動こそ仕事の中核業務であるという意識を持つ必要があります。一番目の問題は妥当なコストや納期条件での受注ができていないこと、二番目の問題は明確な要求仕様が提示されていないこと、三番目の問題は合理的なプロジェクトマネジメント能力および技術力の不足にあります。この問題を解決するためには現在までの自分たちの失敗を直視し、その真因を把握し、その改善目標を設定し、カイゼン活動を実行・継続する必要があります。

愚行＃４２　：順調だと思っていた進捗が実際は停滞していた

【事例】『離れた場所で作業していたメンバーから特に遅延の報告もなかったので順調だと思っていましたが、レビューを行った結果ほとんど作業が進捗していないことが発覚しました』

☆都合の良い思い込みより確認を

この問題はリーダーの責務の問題です。メンバーが離れた場所に居ようが近くに居ようがリーダーは毎日メンバーの状況を確認すべきです。「名ばかりリーダー」「名ばかり管理者」にならないように気をつけなければいけません。基本的に一つのチーム内におけるコミュニケーションはデイリーミーティングにおいて、①本日実行したこと、②明日実行予定のこと、および現在抱えている問題について報告・指示を実行する必要があります。

愚行＃４３　：他に影響はないという思い込みによる不具合発生

【事例】『この機能に影響は無いと思っていた修正が実は影響があり、設定ファイルの修正が必要だったということがありました』

☆ものごとの判断には証拠主義で

なぜ影響がないと思ったのでしょうか。さらに言うと、何らかの証拠に基づいて影響ないと判断したのなら根拠があると言えますが、単に感覚的に、そう思ってしまったというのなら、正しい仕事のやり方を逸脱したということになります。このような感覚的な根拠のない仕事の進め方をしている内は、また同様の問題を必ず引き起こすでしょう。ものごとにおいて何かを判断する場合は、必ず確かな根拠を求める必要があります。これが証拠主義（エビデンス主義）と言うものです。

愚行＃４４　：開発内部で仕様が正しく伝わりませんでした

【事例】『他のメンバーに要求仕様の説明を行い、実装作業をしてもらいましたが、仕様内容の伝達不足により誤った仕様で実装されてしまいました。これくらい分かるだろう、理解したはずだという思い込みが自分に有り、分かりやすく伝えることができず、難しい内容をはしょって伝えることになってしまいました。情報伝達にあたり、不安箇所があるのであれば、随時確認するといったフォローが必要だと反省しています。特に仕事においては、一方的に依頼したまま投げっぱなしとなる仕事のやり方はだめだと思いました』

☆正確なドキュメントによる仕様の伝達を

仕様の伝達は過不足のないドキュメントで伝達する必要があります。口頭だけの説明では正確に伝えることはできないものと強く認識する必要があります。さらに、ドキュメントが用意されていたとしても、自分でも不明・疑問点が沸くようなドキュメントでは役に立ちません。情報の伝達、特に仕様の伝達にあたっては、必ず相手に疑問点や不明点の質問を促し、理解した内容を復唱させることなどで、相手が本当に理解しているかどうかを確認する必要があります。

愚行＃４５　：納期を明確にしないベンダー側担当者

【事例】『ベンダー側の担当者から納期があいまいなまま仕事を振られることが多く、具体的なスケジューリングが無いまま振られた仕事に着手する外注設計者が見受けられます。このままではベンダー側が考えているスケジュールと、外注設計担当が考えているスケジュールに齟齬が発生する危険性があります。このような問題があるにもかかわらず、外注担当者が問い合わせてもあいまいな回答しかもらえない場合や、あきらめて問い合わせすらしない場合もあります。本来は、ベンダー側が最初にスケジュールを明確にすべきだと思いますが、具体的な回答がもらえないのであれば、外注設計者自らがスケジュールを提示して、ベンダー側から了承を得る方法しかないのですが』

☆納期が明示されない仕事を請けてはいけない

納期が明示されない仕事の依頼は受けるべきではありません。派遣常駐の仕事であれ請負の仕事であれ、納期や仕様など開発業務の根幹である内容があいまいなまま仕事にとりかかるべきではないことは、誰にでも分かる常識であることを相手に伝え、なぜ納期が明示できないのか理由を聞く必要があります。このような場合は、依頼されたその場で直ちに、ベンダー側担当者に回答を求め、仮のスケジュールだけでも決めていただく必要があります。さらに正式納期を聞くまではその仕事は正式には受けていない状態にある旨を表明しておくことです。納期が決まるまで毎日問いかけて、数回言っても答えがない場合は、自社の管理職からその旨ベンダー側責任者に対して明確にするように依頼する必要があります。

愚行＃４６　：問題情報の共有がされていない

【事例】『社内で、業態毎、ユーザー毎で発生した不具合や問題、その際の改善策の情報共有を行なう場が少ない。プロジェクト進捗会議の場で、話が出ることがありますが、情報共有のためという目的では無いような気がします。あくまでも進捗状況や、人員調整の話が主となっている気がします。また、問題が起こった場合の情報を共有しても、責められるだけというマイナスイメージが強いのか、問題点の共有を行うメリットが見出せない理由もあると思います。　不具合情報や、問題発生時の情報は、業態、ユーザーが異なっていたとしても、各プロジェクトで共通する情報もあると思います。その際の不具合に対する改善策や、回避策までを共有することが出来れば、各プロジェクト共通の問題を防ぐ手段、改善案の提供や、情報入手を行っていけるのではないかと感じます』

☆問題・失敗・不具合情報を共有しない個人や組織は愚か者

問題や不具合情報の共有活動はなかなか実行されません。他人の失敗情報の共有どころか自分の失敗ですら振り返って歯止め策を検討することさえ実行されにくいのが現状です。多くの人は”いま忙しいからできない”と言い訳をしますが、これを実行しなければ何度も同じ失敗を繰り返すことになり、将来はもっと忙しくなることは明らかです。自分や組織にとって何が本当に得なのか良く考えれば答えは自ずと分かると思います。

失敗の経験およびその有効な対策の共有をしなければ組織も個人も同じ失敗を何度も繰り返すことになり進歩はできないでしょう。同じ失敗を何度も繰り返さないためにも、自分の恥だとかいう狭い情緒的な壁を乗り越えて、失敗経験の情報共有を進める必要があります。大きな失敗に関しては、その表面的な原因のみならず本当の原因である真因を明らかにし、有効な歯止め策の実行にまで至る必要があります。これらの記録は、チーム内は当然のこと他のチームに対してもリアルタイム的に情報共有すべきであり、組織として「失敗情報共有データベース」等の設置・運用が必須です。

愚行＃４７　：伝えてはいけないことを伝えてしまった

【事例】『顧客のマネージャーに正直に伝えるべき内容と、伝えるべきでない内容の判断が難しいと感じています。開発費の状況等、伝えない方がいい内容を伝えてしまったことがありました。自分の性格が、真面目というかバカ正直な面があるため、正直に伝えることが誠実であると考えがちなところがあります』

☆ばか正直は、”正直者”ではなく”ばか者”

相手によって会話の内容を変えることは必要です。特に仕事の会話で利害関係のある相手に対しては話すべきこと、話してはいけないことをちゃんと判断しておく必要があります。状況をわきまえず何でも話してしまうことは、真面目とか正直とかではなく単に愚かなことにしか過ぎません。自分・自社の仕事における立ち位置を常に把握しておき、仕事はある意味で一種の戦場だと心得ておく必要があります。

愚行＃４８　：店舗運用方法や他メーカ機器の仕様情報の共有を行っていません

【事例】『仕事の中で知りえた店舗運用方法や他メーカ機器の仕様情報の共有は行っていません。店舗運用方法や他機器の仕様については、知らないところで変更されることがある為、共有した情報を前提とされた設計が行われることが危険であると思っています。共有しなかった結果は、仕事の手離れが悪くなり、この部分は誰々等、特定の人に頼ることになってしまいます』

☆すぐに情報共有を行うこと

自分が知らないところで変更されるかも知れないので客先運用情報を他の開発メンバーに伝えないとは何と愚かな考え方でしょうか。そのようなことを言っていたのでは、いつまでたっても客先の運用情報は使えないことになってしまいます。客先の運用情報は、設計のみならず評価テストにおいても最重要な情報だという認識が全くないようです。実運用に則した設計やテストがどれ程重要かお分かりいただきたいと思います。客先の運用情報は普通なかなか入手することができません。その情報を入手しているのに他のメンバーには伝えないとは信じられない行動です。本音は、自分の出した情報で間違った設計をされ自分の責任にされたら困るということで、全くの自己防衛的な仕事のやり方だと言わざるを得ません。

そのような狭量な考え方は即刻やめにして、店舗運用方法や他社機器の仕様などの情報についても知りえる限り資料化し自社内にて共有する必要があります。そうすればプロジェクト全体の品質の向上に間違いなくつながります。これらの情報の信憑性や正確性については情報共有の場でチーム内にて検討を加えることである程度の信頼性は確保できるはずです。誰もあなたの責任など問うはずもありません。情報の共有は個人の役割の固定化を防ぐと同時にチームの能力の向上に非常に有効です。

愚行＃４９　：伝わったと思った内容が伝わっていなかった

【事例】『自分の中では「これは常識だ」との思いがあり、目的や方法の説明を省き作業指示を行いましたが、正しい作業が行われませんでした』

☆仕事におけるコミュニケーションは資料に基づいて

口頭による仕事の指示は必ずこのような行き違いを生みます。何年間も一緒に同じ仕事を行った仲間においてすらも、この口頭だけのやりとりに起因した失敗は数多く発生しており、「言った」「言わない」のもめごとは皆さんも時々体験されていることでしょう。ソフトウェア開発における指示の基本はあくまでもドキュメントベースに徹することです。時間に追われている場合は口頭での指示になる場合もあるでしょうが、その場合においても指示内容が正しく伝わっているかどうかを確認するために、相手に指示内容をその場で復唱させることが必要です。さらに追って指示内容に関する資料を渡しておくことです。説明に要する５分間を惜しむ結果、失敗による数時間・数日間を失うようなことは避けたいものです。

愚行＃５０　：メンバーへの教育がうまくいきません

【事例】『人にはそれぞれいろいろなタイプがあり、自分と気が合わないタイプだと自発的にコミュニケーションをとらないことがあり、メンバーへの教育がうまくいきません』

☆感情本位から目的本位への転換を

教育に限らずあらゆる仕事において、人の好き嫌いを基準にしてしまうものごとはうまく行きません。仕事は、人の好き嫌いなどという感情本位で行うものではなく、その仕事は何のために行うのかという目的本位で進めなければならないという認識を強く持つ必要があります。誰にとっても人の好き嫌いはありますが、仕事やメンバーの教育において、そのような感情的あるいは情緒的な態度では目的を達成することは全く不可能だと断言できます。仕事の仲間は、顧客の期待に応えるべく、好き嫌いという感情を乗り越えて、全員の思考と行動を集中して目的を達成する集団であり、決して仲良しクラブなどではありません。

人の好き嫌いで仕事を行ってきたことが、今までに自分にどのような不都合な結果をもたらしてきたのかを考えてみれば、このような姿勢は自分に益をもたらすよりも大きな害をもたらしているということが分かるはずです。人の好き嫌いで仕事をやろうとする姿勢の原因は、自分の中の過度の自己中心性や精神的な幼弱性にあり、まずこれらを克服するための行動をとる必要があります。

愚行＃５１　：慣れによる手順抜け

【事例】『評価作業にて、手順慣れで、やるべきチェックリストをとばしてしまいました。また予定ではエラーを発生させる仕込みを入れてからテストを実施するはずでしたが、仕込みをせずにテストを実施してしまいました。原因は、手順慣れによる一連操作で、パターンを無視して進めてしまったことにあったと思われます。失敗内容が軽度であったため、これらの問題行動に対する、真因の特定、再発防止策のまとめ、問題点の共有とも行いませんでした。今後は、操作内容を発声しながら行うなどで、ミスに気づけるようにしたいと思います』

☆自分の頭を働かせ、丁寧な仕事のやり方を

慣れと、やるべき手順のスキップとは別物でしょう。やるべきことをとばすことを手抜きと言います。慣れてきたから作業の効率やスピードが上がったということなら理解できますが、慣れてきたから手抜きをしてしまいましたという理屈はありません。例えば駅のホームで安全確認のために駅員がよく指差し呼称をやっています。右よし、左よし、前よし、後ろよし、とそれぞれを実際に確認した後に、発車オーライとなります。いくら慣れたからと言って、それをやったことにして発車オーライとは言えないでしょう。

このような手抜き行動の真因は何でしょうか？もっと自分の頭で考えてみてください。想像ですが、この手抜き問題を起こす以前から、自分の行っている業務の意味を自分の頭で考えることもしないで、ただ単に流れ作業的に仕事をこなしていただけなのではないでしょうか。分かれ目は、自分の仕事の意味を自分の頭で考えているのか否かということにあります。すなわち主体的・自律的に仕事に取り組んでいるのかどうかということです。

失敗内容が軽度だったとのことですが、それは結果論で、たまたま軽度だったのに過ぎないでしょう。このようなことを繰り返していると必ず重大なバグを見落とすようなミスをおかすことになるでしょう。自分の頭を働かせ丁寧な仕事に励む必要があります。

愚行＃５２　：資料のメンテナンスをしない

【事例】『評価手順書などの資料で修正が必要なものがあっても、やる気が起こらずメンテナンスするのを後回しにしてしまい、結果修正されないままになってしまうことが多いです。資料をメンテしないといけないのはわかっていますが、自分は内容を理解しているためメンテしなくてもいいと言う気持ちになってしまいます。又、たぶんこの資料は使われないだろうと思いもありメンテを実行しないこともあります』

☆ソフトウェア開発はドキュメントの作成に始まり、ドキュメントの更新で終わる

仕事で使用されるほとんどの資料は、自分たちの仕事の拠り所となる宝です。一度メンテナンスを怠った資料はずっとメンテナンスされない運命におかれるでしょう。そのようにしてせっかく最初に作成された有意義な資料が役立たずのただの紙クズとなってしまうのです。仕様書・設計書・手順書・その他のチェックリストも各自の仕事の中核となる資料は全てメンテナンスされなければいけません。一度中断したら紙クズになるものと心得ておいてください。メンテナンスが不要な資料などないでしょう。もしあるならその資料は本来不要な資料なのかも知れません。資料のメンテナンスは、修正が発生したその時にリアルタイムで修正を書き込むことを習慣化するしか方法はありません。あとで修正しようと思ってもやり切れるものではありません。”後でやる”ではなく”今やる”でしょ！

愚行＃５３　：システム性能比較の失敗

【事例】『システムの性能測定結果が過去バージョンと比較して劣化していたにも関わらず、お客様への報告が遅れ、評価終盤で問題になりました。又、無駄に何回も性能測定を実施してしまいました。問題の原因は、過去バージョンの測定条件、測定方法が曖昧で、同じ条件で測定が出来ていないことにありました』

☆アップル　トゥ　アップル比較を

性能比較は同一条件で行なわなければならないことは皆さん最初から分かっていたはずです。しかしそうしなかった理由が真因なのです。なぜそうしなかったのか、あるいはできなかったのか？それがはっきりしなければまた同じミスをすることになるでしょう。もし性能比較の大前提である”同一条件で行なうこと”という意識があいまいだったとしたら、自分たちの作業の意味・目的を理解していなかったと言うことになります。自分の目で観察し、自分の頭でよく状況を判断し、自分の意思で行動するということができていないこと、すなわち自律性の欠如が、この問題の真の原因でしょう。ものごとの判断基準があいまいなままで、とりあえず何かやって見ようというような気持ちでは常に失敗を招きます。本来、仕事は真剣勝負であり、テキトウな気持ちで成し遂げられるものではありません。　条件の異なったものを比較して優劣を決めようとすることは良くありがちなことです。そのような比較には全く意味がありません。欧米でも良くあるようで、

「Apple to Apple」という警句があります。つまりリンゴはリンゴ同士で優劣を決めるべきで、リンゴとミカンを比べても意味がないと言うことです。

愚行＃５４　：釣銭準備金に万券は必要か

【事例】『釣銭準備金入力で万券も含まれると思っていましたが、万券は含まれていませんでした。他業務で万券を含む機能があり、それと同様だと勝手な思い込みが原因でした』

☆常識力を磨くこと

釣銭準備金には常識的に万券は必要ないでしょう。万券より大きな単位の紙幣があれば別ですが。他の機能で万券を扱うものは、ドロア内の現金在高入力だと思いますが、そもそも釣銭準備金機能と現金在高入力ではお金を扱う業務といっても全く機能の意味合いが異なっています。このような常識的間違いをしないためには、機能の意味を考えて仕事をする必要があります。単に流れ作業的に仕事をこなしていると、このような間違いをおかすことになります。

愚行＃５５　：開発チームからの内部リリース情報がいいかげん過ぎる

【事例】『開発チームからの評価用プログラムの受入れ時に評価対象機能内容の把握に手間取ってしまいます。開発チームから提供される機能情報や結合テスト時の不具合管理表に記載されていない改修機能が多数あり、それらの把握がなかなかできません。

たとえば、機能Ａがあり、その機能Ａには更に機能 a、機能 b、機能 c などの小機能があった場合、評価チームへのリリース情報には、「機能Ａがリリースされます」との記述しかなく、評価テストを開始してから初めて、「機能 b」に関わるテスト実施中に、不具合だと思っていたものが実際は未実装だった、ということがあり、不具合なのか未実装なのかの判断をするまでに無駄な時間を消費してしまうことがあります』

☆開発チームは評価チームに対してまともなリリースノートを発行すること

開発チームから評価チームへの評価用ソフトのリリース時には、開発側から単体試験成績書および結合試験成績書の提出および未実装部情報、機能不全情報、改修中情報などを詳述した評価用リリースノートの提出を義務づける必要があります。評価用リリースノートの提出義務は明確にプロセス管理表に記載され、実行されなければいけません。単にプログラムを作りました、後の評価は評価チームでお願いしますなどと言う無責任なことを言う開発は開発としての自覚意識に全く欠けていると言わざるを得ません。これらの情報がなければ評価チームがいくら頑張っても有効な評価業務を行うことは不可能です。評価チームは黙っていてはいけません。仕方のないことだとあきらめていてはいけません。もしこのようなルールがないのなら開発リーダーおよび評価リーダーの双方において話し合いのもと新たな公式ルールを作成する必要があります。

第３編　なんと情けないことか

©いらすとや

はじめに

情けないという言葉にはさまざまな意味がありますが、ここでは主に「嘆かわしい、あさましい」と言う意味で私たちのさまざまなトホホな実態について見ていきたいと思います。

情けない状態とは、ひどくて恥ずかしくて残念で惨めな状態を指しています。そのような人を揶揄して「残念な人」と言い、そのような人が率いる組織は「残念な組織」と呼ぶことにします。

人が、自分を情けないと感じる局面は、何かの失敗をした時や、できるはずだと思った目標が未達成だった時や、他人が容易に実行できることができなかった場合などですが、これらの状況に遭遇した後の人々における対応には三つのタイプがあります。

一つ目は、「二度とこのような屈辱感を味わいたくない」と思い、その失敗の原因を克服するための行動を取る人たち、すなわち自尊心や誇りの修復を行うべくリベンジ（雪辱・再挑戦）を誓う人たちです。二つ目は、失敗の直後には「くやしい」と思い対策を講じ始めるが長続きできない人たちで、多くの人はこのタイプが多いと思われます。 最後のタイプは、「くやしい」という感情はわくものの、その失敗の原因を克服する知恵もなく努力を継続する気力もなく、最初からあきらめて何もしない人たちです。

改善努力が継続できない人たちにおいては、本書の事例を今一度読むことで、自分の中にある「くやしさ」にもう一度火をつけ、直面する問題を自分の頭で良く観察し、自分の思い込みや思い違いに気づくことで更なる前進を可能にしていただきたいと思います。

愚行＃５６　：分からない仕様の調査を放棄して外注に丸投げしてしまう

【事例】『仕様変更対象の業務に詳しくないため、最終的に協力会社に任せてしまい、高い確率で障害が発生することがあります。 精通していない機能の場合、現在の処理を理解するまでには多くの時間が必要になります。実際、その時間を工面することが出来なくなり協力会社に丸投げしてしまうことになり、この場合、レビューを実施しても効果は薄く、障害の検出は難しいと思います。このような場合、業務に精通している者に仕様内容を聞き自分の経験不足を補うべきだと思います』

☆仕様の意味や背景を知りたければ顧客に聞くこと

多くの開発者は仕様の意味や背景について余り知ろうとはしていません。仕様の意味や背景は、そのシステムの各機能の実運用の目的を理解するところから始めなければ理解することはできないでしょう。いつも誰かが決めてくれた要求仕様書の通りにコードを書くことばかりに注力してきた開発者において最も興味のあるものはソースコードだけになってしまったのではないかと思えるほどです。しかしいくらソースコードを読んでもその機能の実際の目的や意味を知ることは不可能でしょう。それにもかかわらず、ほとんどの要求仕様書にはその仕様の意味や背景についての記述がありません。仕様の理解を深めるためには、まず身近に有識者がいるならば、その人に教えを乞うべきでしょう、そのような人材がいなかったならば恥を忍んで、実際の運用に熟知している顧客に教えていただくしか方法はないでしょう。そして全員がそれぞれ担当する部分に関して、仕様書の先頭にその仕様の本当の意味および背景について記述を加える作業を地道に続ける必要があるでしょう。

愚行＃５７　：リリース用フラッシュＲＯＭの作成失敗

【事例】『ファームウェア開発業務で製造用フラッシュ ROM のデータファイルをリリースしました。しかしそのファイルを作成するためにバイナリーエディタでバイナリーデータのコピー＆ペースト作業がありますが、バイナリ編集中に DEL キーを押下してしまったらしく、１バイトデータが消えたファイルをリリースしてしまいました。 問題の原因は、チェックが不十分であったこと。書き込んだフラッシュ ROM をテストしてからリリースすべきでしたが、フラッシュ ROM を容易に脱着できる基板がその機種にはありませんでした。今後の再発防止策としては、リリース用のチェックリストにバイナリファイルのチェックリストを作成して確認できるようにし、更にツールを作成し手作業を減らしました。また、リリース用のファイル確認用にフラッシュ ROM を容易に脱着できる確認用基板を用意してもらうようにしたいと思います』

☆自分の仕事に問題意識を

いろいろ原因が書かれていますが、そもそもこのバイナリーエディターには書き込み元のデータと書き込み先のデータの同一性を比較する機能があったはずですが？比較機能がないエディターなど聞いたこともありません。もしなかったとしたら、このエディター自身が欠陥品であり、そのようなエディターを何の疑問も持たずに使用していた人間自身の欠陥です。失敗の真因は、物や仕掛けにあるのではなく、人にあるということを自覚する必要があります。

最終成果物の作成にデータのコピー＆ペースト作業という手作業があることについても何も疑問を持た

ずに、いつまでも危ない作業を続けていた人間自身の姿勢にも問題があります。相談者が提示されている解決策は、それぞれに効果がありますが、基本的な問題は、自分の作業に何も疑問を持たずにいた人間自身の自律能力の欠如にあります。全ての開発業務において、自分の目でしっかり観察し、よく考え、適切な判断を行い、実行するという自律性強化の努力が必要です。そのような実地訓練に最も適切なのがカイゼン活動です。自分たちの仕事に、第一に愛着を持ち、次に問題意識を持って、良い仕事をしたいと思うことから出発する必要があります。

愚行＃５８　：自分の主張の根拠について適切に説明できない

【事例】『打ち合わせなどで、自分の主張の根拠を求められた場合うまく説明ができないことがあります。あとで落ち着いて考えれば適切な回答が浮かんできますが、その場になったときに落ち着いた回答ができません。自分が正しいと思ったことに対する背景を整理するために準備をして、意見をはっきりさせておきたいと思います。自分の不足を判断したときは、一旦時間を置いて適切な回答をするようにしたいと思います』

☆うまく説明を行うためには事前の準備を

相手の質問にその場で適切な回答ができない原因には、その問題についての知識不足や事前検討不足などがあります。たとえば仕様や機能に関して適切な回答に窮する原因には下記のようなものがあります。

①仕様・機能に関する知識の不足

②仕様・機能の運用に関する知識不足

③技術に関する知識不足

これらは一朝一夕に解決できる問題ではありませんので常に経験者や有識者からの知識を収集して資料にまとめておくようにしておくことです。また今直ぐに出来る対策には事前準備として下記のようなものがあります。

①仕様追加・変更の影響度の事前調査

②仕様内容の骨子の事前の把握と整理

③問題点・疑問点の事前の掘り起こし

相手に質問される前に相手に対してこれらの事前検討内容を質問するような積極的な仕事のやりかたが有効だと思います。また、その場で適切な回答ができなかったことを過度に恥ずかしいとかみじめだとか思う必要はありません。同じ内容で二度と恥をかかないためには、その内容についてきちんと理解をしておき自分用の技術者ノートなどに内容を整理しておくことです。恥をかくことを恐れず、恥をかくことを自分の成長につながる行為に変えることを実践する必要があります。

愚行＃５９　：中途参加のプロジェクトにおいて不明点や疑問点を聞きづらい

【事例】『途中からプロジェクト参加したため仕様・設計・内部構造の知識不足や資料の存在場所などが分からず、その場にいる人に聞く事が多くなります。聞かなくても分かる事と聞かなければ分からない事が混在しており、その線引きがよく分からないために同じ事を複数回聞いてしまう事があります。他メンバーが多忙であり説明を聞きづらいため、忙しいメンバーの手をわずらわせないように質問事項をまとめて空き時間に回答してもらえるようメール等でやり取りをするとか、質問に対する回答についてはしっかりメモに残し同じ質問を繰り返さないようにするなどの努力が必要だと思います』

☆分からないことは、その場で質問すべし

途中から参加したプロジェクトでは不明点や疑問点が多く発生することはやむを得ないことです。相手を気づかってメールなどで質問したいとありますが、同じチームで同じ場所にいるのなら直接質問をして下さい。何度聞いても分からなければ質問の仕方を変えて分かるまで何度も聞いて下さい。各自の業務実行中に聞いたり聞かれたりするわずらわしさを避ける方法は、中途参加者に限らずプロジェクトの全てのメンバーが抱えている問題点や疑問・不明点をオープンにする場として日次進捗会議の実行が最も良いやり方でしょう。毎日、各自の進捗状況や問題点などについての情報共有を短時間でも良いので実行すべきでしょう。　２０分でも３０分でも良いので毎日の短時間の日次会議をチーム全員で実行することです。日次会議の内容は、各担当者からの報告とリーダーからの指示・コメントで下記３点の実施だけです。

１．昨日（ｏｒ今日）何を実行したかの報告。
２．今日（ｏｒ明日）何を実行する予定かの報告。
３．今抱えている問題（困っていることなど）は何かの報告。
＊長時間を要する問題は当事者とリーダーで別途打ち合わせをすること。

このことを毎日繰り返し実行すれば、分からないことがいけないとか質問がしにくいとか個人だけしか情報を持っていないとかの問題は無くなるはずです。

愚行＃６０　：新しい仕事にチャレンジしたくない

【事例】『新しい業務、例えば新ＯＳ・新ＨＷ対応や苦手な業務にチャレンジするように指示されたが、はめられた感を強く感じて前向きになれない場合は一歩前にでることができません。理由としては様々なリスクや不安を払しょくすることができないからです。たとえば、「品質が良いものをリリースできるか？」「納期が守れるか？」「何をどれだけ検討すれば良いか？」「役割が不明確」などがあります。これらを解消するには、「上司やメンバーとの信頼関係を作る」「自分自身に自信を持つ」「役割を確認する」「先送りにしたことは後片付けをしっかり行う」などを実行する必要があると思っています』

☆まずは不安やリスクを書き出すことから始めてみる

相談内容は新しいことはやりたくないとの表明のオンパレードです。未経験の仕事に直面した場合、誰でも自信がなく失敗の不安で頭が一杯になります。まずは直面する仕事の内容について不明点や疑問

点を書き出すところから初めて下さい。いろいろ不平不満はあるでしょうが、まずは自分が不安なことやリスクだと思っていることについて、更に具体的に詳細に書き出してみてはいかがでしょうか。新しい業務の内容、期待される自分の役割、納期、仕事の進め方などに関する疑問点・不明点・心配なことについて列挙してみてください。また一歩踏み出せない理由についても更に具体的に書き出してみてください。それぞれに対応方法が書けるものは書いてください。それが終わったら次のことについて指示者の上司と話をしてみてください。

①自分なりの対応方法が書けたものについて間違いや過不足がないか確認する。

②対応方法が不十分にしか書けなかったものについて、分かったところまでの説明をし、不明な部分について相談をして下さい。

③全く対応が書けなかったものについては、書けなかった理由を説明した上で再度指示・支援を仰いで下さい。

上記について上司と会話をすれば自分の努力では全くどうしようもないものと、どうにかなりそうなものとが明確になるでしょう。全くどうにもならないものについては上司に支援を要求する必要があります。足らないものは人・モノ・カネ・時間・情報・知識のどれかに当てはまるでしょう。これが正しいチャレンジの仕方でしょう。また未経験や困難な仕事に直面してからあわてないためにも、各分野のエキスパートを講師にしたチーム内勉強会や自主的な勉強を日常的に実行されることを期待します。

上司が「新しいことにチャレンジして」と指示したのはあなたにまだ期待していますとの意思表示です。仕事人は期待されているうちが華ですが、もしそれに応えたくないというなら、おそらくその職場でのあなたの将来は暗いものになってしまうでしょう。「無理を押し付けられた」とか、「ハメられた」というような感情を解消できないのなら、あなたには新しい仕事は無理だということでしょう。

愚行＃６１　：安易な仕様変更要求

【事例】『これで完結するとはとても思えないような度重なる仕様変更が繰り返されると、それまでに費やした検討・熟考の時間や労力が無駄になり、次の検討を行う気力も失せてしまいます』

☆自分も仕様検討に参加する姿勢で安易な仕様変更に歯止めをかけること

度重なる仕様変更を防ぐ工夫をしなければいけないでしょう。まず「相手は仕様を決める人、自分は設計・製造する人」と言う硬直した考え方を少し緩める必要があります。自分も仕様決定に参加するのだという姿勢を持ち、提示された仕様について「本当にそれで良いのか？」という疑問を常に持つことです。自分が提起した疑問に対して、仕様提示者から納得のいく回答が得られるまでは先に進んではいけないでしょう。特に相手の仕様提示者における能力にリスクがあると思われる場合は余計に注意が必要です。

そのような注意を払ったとしても仕様変更要求が出た場合は、再検討や再調査が必要になってめんどうでしょうが、一方これらの経験がその仕様に対する深い理解につながり開発の品質向上に貢献することになると考え、気持ちの切り替えを行って取り組んだ方が良い結果を生むでしょう。

愚行＃６２　：未経験の仕事に直面した時、一歩を踏み出せない

【事例】『これまでに経験の無い新技術を採用した開発プロジェクトに対して、知識や経験不足のため自信を持って前に出られず積極的に取り組むことができません。新しい情報や技術を取り入れるための業務外学習が必要だと思います』

☆事前・自主的な学習と経験者の下で一歩を踏み出すこと

そのようなプロジェクトに直面してからあわてても仕方ありません。自社の請けている仕事の傾向や業界の技術動向について日ごろから関心を持ち、主流となりそうな技術については事前に自主的な学習をしておく必要があります。自分が持っている現在の技術だけに依存していては、世の中の技術の変化に追随していくことはできません。

そうはいっても新技術の知識が必要とされるプロジェクトに参加させられた場合には、そのプロジェクト内で知識をもった技術者のサブにしていただき、その人の下で仕事を通して新技術を徐々にマスターすることが現実的です。業務外の自主学習も併せて是非実行して下さい。

愚行＃６３　：必要な行動ができない

【事例】『他の人からの依頼は、相手を待たせないためできる限り優先度を上げるよう心がけていますが、一方自分から確認や報告などをしなければいけない場面で行動できないことが多く、相手に迷惑をかけ、自分が困る結果を招いています。他の人からの要求に応える動機としては、「相手によく思われたい」「相手への思いやり」「相手に貸しを作りたい」「相手への恐怖心」などがあります。また自分から行動を起こさない原因としては、「相手に関心がない」「相手への恐怖心」「相手への怒り」などがあります。そこで行動しなければ後でどうなるかをすぐに考え、なるべく行動するよう心がけはしています』

☆仕事の意味についてよく考えること

他の人からの依頼には優先度を上げて対応するが、自分からしなければいけない行動はできないということは、他人の評価によって自分の行動を決めるという自律性のない人における典型的な行動パターンです。要するに自分自身のしっかりした考え方がないということです。他人からの依頼には優先度を上げるということは一見よさそうな行動に思えますが、決してそうではありません。たとえば顧客からの依頼は最優先事項の一つになりますが、常に手持ちの仕事と優先順位の比較の上、対応する必要があります。また社内の人間からの依頼についても同様です。優先度はその仕事の緊急度や重要性の比較で決まり、他人からの依頼がいつも最優先とは言えません。

また確認や報告が必要なのに行わないということは「義務」の不履行にあたる問題行為です。必ず実行すべきでしょう。仕事は、相手に対する好悪の感情や損得勘定を乗り越えて、なすべきことを着実に実行することです。もう一度、プロの仕事とは何か、仕事は誰のためにするのか、ということについて真剣に考える必要があります。

愚行＃６４　：レビュー、電話応対、知らない人との会話が苦手

【事例】『業務に限らず日常的なコミュニケーションにおいて突発的な応対全般が苦手です。特に、自分が考えていなかったような事象・用件に対して、いつもあせったような応対をしてしまいます。例；レビュー中などで予想していない質問を受けたとき。電話応対。知らない人との会話など』

☆習うより慣れよ

誰しも予測不可能な対応が迫られる局面は苦手なものです。レビュー、電話応対、知らない人との会話などは誰でも多かれ少なかれ苦痛を感じます。あせったり、あわてたりする感情を否定しないことから始めてください。未知なるものごとに対する正常な反応なのです。正常な反応を異物化し、敵視してしまうことは、正常な反応を否定することになり、その後の自分の行動は歪んだ不適切なものになってしまいます。特に「恥の文化」をもつ日本人全般が陥りやすい悪弊なのです。やるべきことは自分の感情を否定するというような感情的反応ではなく、相手が求めていることを冷静に見つめ、いま答えられることは答え、そうでないものについては後日答えるなどの事実本位・目的本位な行動をとることです。これらの積み重ねが自分の経験の積み重ねとなり、いつしか多くのものごとに冷静な対応ができるようになるものです。以下の実践を積み重ねてください。

①経験を積み場数を踏むことで慣れること

②自意識過剰である事実を認めること

③仕事や自主学習に励み業務知識や一般常識を豊富にすること

一時期の不愉快さに耐え努力を継続することで道は開けるでしょう。

愚行＃６５　：いわゆるサボリ問題

【事例】『業務に関係ない雑談や、ネットサーフィン、携帯操作等、多少なら構わないと考えていますが、頻繁に目つき、そのあげく残業しているというのはどういうこと？と感じています。注意をするタイミングが掴めないのと、注意することで自分との関係が悪化するのではという心配があります。また周囲の人のモチベーションは著しく低下してしまいます』

☆常習的なサボり行為には警告を

業務中に業務以外の私的なことに時間を費やすことをサボリ行為と言います。たとえばネットゲームをしたり、居眠りをしたり、長々と井戸端会議をしたり、果ては株の取引をしたりなどパソコンとネットワークを使った常習的なサボりを目にすることもあります。このような人たちは自分の仕事や人生をあまく見ているとしか思えません。常習者を見つけたら警告する必要があります。ただその場合の言葉の使い方には注意が必要です。その行為そのものを指して「いつも○○をしているようだけど止めたがいいよ」とか「こんなことやってまずいよ」などという直接的な言い方では逆うらみをかうだけでしょう。本人はまずいことを承知でやっている場合が多いので、婉曲法やユーモア的な表現で十分に相手の心にグサリと刺さるでしょう。たとえば、「暇みたいですね」とか「これを調べていただけませんか」とか「○○について教えていただけませんか」などの問いかけをタイミングよく行うことです。それでも何ヶ月も止めない場合は上長に知らせるべきでしょう。私

の経験では、そのような常習者は例外なく皆遠からず会社を辞める結果を招いています。

愚行＃６６　：臆病なため、気づいた問題点を言い出せません

【事例】『自分の意見を述べるべき時に言い出せません。他人が怖く、また自分に自信がないため、いつも後出しなど、積極的ではありません。このような性格がたたって、作業中に気づいた問題点についてもなかなか言い出せなくて、他の人が気づくまで放置していることがあります。作業の中で気づいた問題等が、自身のミスによって起きた現象だった場合など、居た堪れなくなります』

☆その繊細さを仕事に活かすこと

自分の不始末であろうが他人の不始末であろうが、不始末に気付いたら自分の感情はどうあれさっさと始末をつけましょう。それが仕事です。自分のミスを悔やむことは自然の感情ですが仕事というものは個人の感情で動かすものではなく目の前の現実がどうかによって動かすべきものです。ミスを悔やんで言い出せなくて他人任せで放置することの問題の方が元もとのミスの問題よりも大きな問題であるということに気付く必要があります。そして自分の持つその繊細さを仕事に活かす方がどれほど役に立つことでしょう。

愚行＃６７　：リーダーにもっと誠実さをもって部下に接していただきたい

【事例】『リーダーはいつも忙しそうにしており、相談したいことが有ってもなかなか声をかけられません。私はリーダーにもっと誠実さをもって部下に接していただきたいと期待しています。聞きやすい雰囲気をもったリーダーならば、もっと疑問点や不明点についてタイミングよく話を聞いてもらうことができ、リスクや問題点の早期発見につながり、プロジェクトのスムーズな運用ができ、チーム内の信頼関係も強化されるのではないかと思います』

☆リーダーが動いてくれるような頼み方をする工夫が必要

あなたが期待するようにリーダーが動いてくれるような頼み方をする工夫が必要でしょう。自分に割いてもらうのに必要な時間を伝え、都合の良い時間を聞いておき、事前に自分が聞きたい内容を整理しておくことが必要でしょう。また忙しいリーダーのことを考慮して、できるだけ短時間で済むように心掛ける必要があります。どうしても聞かなければ仕事が進められないような重大なことは、リーダーが忙しそうか否かにかかわらず、すぐにその旨をリーダーに伝えるべきです。部下が抱える問題に応えることは、リーダーの主要な仕事の一つなのです。そうは言っても、ダラダラと続く何が問題なのかも分からないような話をいつも持ち掛ける部下の話は、どんなに忍耐強い誠実性のあるリーダーでも積極的に話を聞く気にはなれないでしょう。

愚行＃６８　：信用・信頼のもとになる業務実績を積み上げられません

【事例】『信用のある人とは、「この人だったら問題なく任せられる」と思われるような人のことだと思います。信用・信頼は実績の積み重ねの結果として生まれてくるものだと思いますが、日々の仕事を振り返ってみると、時間が無いと言い訳してレビューを十分に行わないことや、面倒な作業を後回しにしてしまうことがあり、その結果、品質や納期に関して問題を起こしてしまい実績を積み上げられていません。また自社の利益を優先に考えてしまい、お客様が第一という意識が足りず、何らかの決定や作業を行うとき、お客様にとって問題ないかどうかをお客様の立場に立って第一に考えるようにしていない面も影響していると思います』

☆気分本位ではなく顧客価値の重要度順位の仕事を

信用・信頼は、良い仕事の積み重ねによって得られるものです。良い仕事をする条件として、優先順位に従った仕事の遂行ということがあります。仕事の実行の優先順位は、まずは顧客の価値の重要度の順であり、効果が大きく実行が容易なものの順になります。優先度が低いものは効果が低くて難易度が高いものですが、タイムリミットが迫っているものはその期限が第一優先となります。自分とって単にめんどうだからとか、やり易いとかいう気分本意の優先順位は有害です。また「時間がなかったから・・だった」という言い訳をよく聞きますが、なぜ時間に追われているのか一度良く考えてみて下さい。多くの場合の原因は、やらなければならないことや、その優先順位が不明確なために、最終的には納期ありきから逆算して時間を決めてしまうために時間に追われる結果に陥るのではないでしょうか。いつも時間に追われる状態ならば、仕事のやり方や技術レベルに基本的な問題があり、それぞれのカイゼン対策を実行する必要があります。

愚行＃６９　：ラップアップ（振り返り）が定着できない

【事例】『プロジェクト完了時、各メンバーが次のプロジェクト作業に追われてしまい、QCD 改善に対する情報共有とラップアップの定着ができていません。

同じメンバーにて次プロジェクトを実施することが少ないため、各メンバーの作業状況によりラップアップの優先順位が下げられてしまいます。また、改善資料作成等の案があっても次プロジェクトと並行して行えないことが多いため、実行に至りません。

今後は、プロジェクト実施時に並行しながら課題に対する改善資料・リスク回避資料をとりあえずメモ書き程度でも残すような習慣付けを行い、課題や改善策を風化させないためプロジェクト完了直後にラップアップの実施を必ず行いたいと思います。また、各プロジェクトの改善案を情報共有できる場を作る必要があると思います』

☆気持ちだけでは実現できません

対策として提案されている、とりあえずメモ書き程度でも残すというような程度の低い内容では有効な振り返りはできないでしょう。時間がないと言うならば、第一に時間が足りない原因を考えてみるべきであり、その対策を行うべきなのではないでしょうか。作業中にＱＣＤに関するデータの採取と分析および対

策の立案を行う必要があります。もし実行されていないのなら毎回のプロジェクトはいつも時間不足のうえに品質改善もできないでしょう。ラップアップに必要なデータは開発作業進行中に大方は完成しているのが普通の仕事の在り方です。これらのことを実行しないで、ラップアップのために新たに資料を準備・作成しようとするから振り返りやラップアップができなくなってしまうのです。バグなどの重要不具合や全体の不具合傾向分析も日々の業務の中で毎日更新すべきものです。それらの不具合票には当然、原因、対策、歯止め策などが記載されなければいけません。一つのプロジェクトが終了した時点でこれらの資料はラップアップの前にはすでに作成済みとなっているべきものです。リーダーはラップアップ時にこれらの資料および重要不具合や不具合の全体傾向をまとめてメンバーと情報共有するだけです。良い仕事はマメに地道に毎日コツコツと積み上げることでしか達成できません。

愚行＃７０　：類似不具合がなぜ発生してしまうのか

【事例】『過去に発生した不具合の類似不具合が、何度も再発することにいつも疑問を感じています。過去に不具合を出してしまい苦労したにも関らず、類似不具合が発生してしまっており、過去の反省がうまく生かされていないように感じています。類似不具合の再発防止のために、過去に発生している不具合一覧等を作成し、チーム内で情報共有を行うようにしたいと思います。情報共有を行うことで各自注意し、レビュー時にレビュアーが確認することで類似不具合を減らせると思います。また、各自が不具合を発生させないようにするという意識を持つことで、類似以外の不具合も減少すると思いました』

☆類似不具合発生の防止策

類似不具合の発生原因と防止策は以下の通りです。

【類似不具合の発生原因と防止策】

①不具合修正時に不具合発生の表面的な原因の修正のみしか実行しておらず、発生の真因の特定およびその恒久的な歯止めや解消を実行していないこと。

②過去の不具合を検索する仕組みや手段を持っていないこと。

③不具合情報をレビューに生かせないこと。

④過去の不具合に対する組織的な取り組みを行っていないこと。

例えば①に関しては、不具合部分のプログラム修正を行うだけではなく、問題の原因を単に「製造ミス」や「設計ミス」というレベルにとどめず、なぜ製造ミスや設計ミスをしたのかというレベルにまで踏み込む必要があります。真因はメンバー間の仕事の負荷の不均衡、知識不足、情報不足などでしょう。それらの真因に対する恒久的な対策を実行しなければ問題は解消されません。さらに関連ドキュメントの全ての修正も行わなければ、不良ドキュメントによる同じ不具合を何度も繰り返すことになります。

②に関しては、過去の不具合を検索するデータベースなどの仕組みを保有・運用していなければ膨大な不具合の中から自分が参考にしたい不具合を発見することは全く困難でしょう。

③に関しては、不具合の真因の特定と解消および不具合情報の容易な検索などの仕組みを構築し

なければ、各レビューにそれらの情報を供給することはできません。

④に関しては、以上の取り組みを個人的に行ったとしても大きな効果はまったく望めないでしょう。組織的な仕組みやルールのもとにこれらの活動を着実に実行していく必要があります。

まずは、重大不具合や頻発する不具合に絞って上記の対策が実行されることを期待します。

愚行＃７１　：困難な仕事に直面した時、一歩前に踏み出せない

【事例】『困難な仕事に直面した場合、その後の責任やリスクを考えると、自分が払う犠牲が大きいと判断してしまい、責任を負う覚悟もできず、一歩前に出るモチベーションも持てなくなります。もっと高い意識をもって行動をしたいと思いますが、思うようにできません』

☆困難に直面したときは条件付行動を選択すること

自分が一歩前へ出なかった結果、その仕事はどのようになったのでしょうか？誰か他の人がリスクと責任を負うことになってしまったのでしょうか。もしそうならば、他のメンバーがその役割を負い、自分は自ら一歩後退してしまったことになります。自分の成長の機会を自ら放棄してしまいました。リスクとか責任について自分が全てそのリスクや責任を負わなければならないという考え方は正しいのでしょうか。チームで仕事をしているのならあなたのリーダーや同僚たちと話し合い、共にリスクの解消を行い責任を分担するような方向性を見出すことが合理的かつ妥当性のある行動と言えます。自分で背負い切れないものを全て背負おうと思うこと自体に無理があります。無理な考え方の結果は、結局自分は何もしないと言う道理に合わない選択をすることになり後悔だけが残ります。物事の判断、特に困難な問題に直面した場合の行動の選択を”１か０か”の思考で決定すべきではないでしょう。”１か０”の思考、つまり”All or Nothing”思考では、多くの人は必ず楽な方つまり何もしないことを選択してしまいます。このような１／０思考は習慣化しやすく個人のモチベーションや成長の大きな阻害要因となります。困難な問題に直面したときには、「ここまでは自分が背負えますが、その他は誰かが背負っていただけますか」というような条件付の提案が有効でしょう。

愚行＃７２　：萎縮しがちな性格と勉強不足が信頼関係の障害になっている

【事例】『年上の顧客マネージャーや役職者と信頼関係を作ることは、次の仕事を取ってくる意味でも重要なことですが、なかなかできません。性格的に目上の人に対して萎縮・遠慮してしまうことが障害になっています。また、相手に「よく勉強しているな」と思われるような会話をするための知識も不足していると思います。普段から勉強するようにしたいと思っています』

☆自律性の発揮と明確な仕事観を

相手が顧客の目上の人ならば誰しも緊張したり臆病になったりすることは当たり前のことです。ものごとがうまくいかないことを自分の性格のせいにしてしまうと何も進展できないでしょう。人はさまざまな性格を持っていますが、その人が選択する行動もまたさまざまです。性格的に気弱な部分を持っている人がすべて萎縮した行動を取るわけでもありません。基本的に自分の性格を変えることはできませんが、行動の選

択は自由です。気弱な性格のままでも、自分の中の強い情熱や使命感に従って苦手な行動を行う人たちは大勢います。相手との力関係に振り回されずに、もう少し自分の主体性というものを発揮する必要がありそうです。自分の主体性や積極性は、ものごとを自分の目で確認・判断し行動する自律性というもので発揮されます。そのためには自分における”仕事は何のためにするのか”という仕事観をしっかりと持っておく必要があります。仕事とは第一義的には、「他人が困っている問題を解決する作業だ」ということをしっかりと自覚しておく必要があります。他者の困りごとを解決することによって給料をいただき、相手から感謝される結果が自己実現につながります。仕事に関する知識向上のために、自分に必要な具体的な専門領域を明らかにし、計画的な学習を着実に継続されることを期待します。

愚行＃７３　：残り作業があるのに帰社してしまいます

【事例】『残り作業があっても、優先度は高くないから今やらなくてもいいかなと考え、翌日まわしにしてしまいます』

☆仕事に対する取り組み姿勢の問題

これは意識の問題でしょう。残り作業だという認識があるということは、この仕事は本来今日中に仕上げるべきだと自分で分かっているからこそ”残り”という言葉になったのでしょう？明日に予定していた仕事ならば明日にやれば良いだけのことです。優先度とかいう言い訳ではなく、この残りの仕事を今日中に終了しなければ、明日以降自分や他の人に迷惑がかかるかどうかの視点で考える必要があります。迷惑や問題が出そうなら今日中に終了させるべきです。そうでなければ自分の裁量の中でこなせば良いだけのことです。これは誠意とかいう問題ではなく、仕事における自分自身の取組み姿勢の問題です。

愚行＃７４　：対人関係に弱く、チーム内の連携ができない

【事例】『自分の性格上、対人関係やコミュニケーション力が弱く、常にものごとをネガティブに考えてしまいチーム内における連携や連帯ができません。チームプレーは、仕事の成功や自分のスキル向上にとって必要なので、今後は、プラス思考を取り込み、前向きにコミュニケーションを取り、チーム内においては最終的なゴールの目標を明確し、目標意識を合わせることなどを行っていきたいと思います』

☆行動できないのは性格のせいではない

対人関係やコミュニケーション力が弱いことを自分の性格のせいにしても何も得るものはありません。ネガティブに考えるということは、ある一面ものごとのリスクや危険性を察知する力があるということであり、ものごとに対して批判的な見方ができるというプラスの面でもあります。分かれ目は、「だから行動するのは止めよう」なのか、「だから必要な準備をして行動しよう」なのかの一点にあります。「行動するのを止める」を選択する人の多くは、その原因や理由を自分の性格のせいにすることが多いものです。できなかった理由を性格のせいにしておくことで、自分を一応納得させようとしているだけなのでしょう。しかし現実に目をつぶることで一時の安心が得られたとしても、また目を開ければ、現実は前よりももっと荒涼としたひどい状況になっていることでしょう。性格を変えようとかプラス思考を取り込もうとかという実現性に乏しい考え方をや

め、目の前の現実の問題を小さなものから徐々に解決していくための行動を一つずつ積み重ねていくことから始める必要があります。今までできなかったことを一気にできるようにすることは不可能です。まずは取り組みやすいものから実行していくことをお勧めします。自分のネガティブなものの見方の対象を自分自身に向けるのではなく、現実の問題に対して向けるようにすれば、徐々に行動がとれるようになるでしょう。気分本位ではなく目的本位で行動するということです。

また、仕事は多数の仲間がお互いの役割を果たしていくことで達成されるという視点で考えてみてください。仲間のそれぞれの役割に対して自分の役割は何かということを具体的に深く考えて、それを言葉に書き表してみて下さい。具体的な日常の仕事について自分が貢献できることは何かについて考え、それを言葉にし、行動に結びつけるという努力はきっと役に立つでしょう。物事を深く考える力をつけるためにも読書されることをお勧めします。

愚行＃７５　：チームの士気の低下

【事例】『チーム内の士気が上がらない場合、自分ひとりが頑張っているという孤独感に陥り、モチベーションが下がってしまいます』

☆「積極性」「目標の設定」および「開発期間の確保」「チームプレーの発揮」がチームの士気を高める

チームの士気低下の原因はこの悩み相談にて提起されたすべての問題に起因しています。特にその中でも大きな比重を占めているのは次の４点です。

【チームの士気低下の原因】

①やらされ意識で行っている仕事

受身の姿勢で行っている仕事はいつしかルーチンワーク的な単純作業と誤認され、面白くない仕事と感じられ、言われたことだけを実施する仕事になってしまいます。

②目標が設定されていない仕事

目標がない仕事は、すなわち先が見えない仕事であり、どこまで頑張れば良いのか分からないため気力・体力の維持ができなくなる。

③過酷なスケジュール（長時間・長期間残業）

受身の姿勢および無計画と無目標が過酷な長時間労働を生む本当の原因です。

④チームプレーの欠如

チーム内で助け合いが行われない。リーダーの率先垂範が行われない。適材適所に欠ける。特定の人に負荷が偏っている。日次会議など頻繁な情報共有・目的共有・目標共有・問題共有が行われない。

これらの問題のインパクトの大きいと思われる順に対策を実行されるならば自ずと道は開けてくるでしょう。

愚行＃７６　：セキュリティが阻害する情報共有

【事例】『小日程表でチームメンバーの開発スケジュールを管理しており、各自がそのスケジュールにて自分の作業を把握した上でミーティング等に参加していると思い込んでいました。しかし実際は小日程のExcel ファイルに設定されていたパスワードが全員に周知されておらず、一部のメンバーは紙で配られた初期のスケジュールしか見ていなかったため、開発後期になるにつれ、何をいつまでに終わらせればいいのか正確に伝わっていないことがありました。 この問題の原因は、私自身が指示する／しないに関係なく、メンバーが過去何年も大きな問題なく開発していたため、サーバ上の小日程表を見ているものと思い込み、疑わなかったことにあります。また作業上の問題を確認しても小日程表が話題に上がったことがなく、本人以外からパスワードを知らないようだとの情報が上がるまで気づけませんでした。本人もそういうものだと思っており、敢えて上げなかった様です。今後の注意点として、自分にとっての当然と他者にとっての当然を混合しないようにしたいと思います』

☆重要事項はＩＴ頼りにしないこと

信じられないことです。本件は、事実認識とか本質の把握の話ではなく、もっとレベルの低い根本的な問題でしょう。小日程ファイルに何故パスワードのセキュリティが必要なのですか。セキュリティポリシーの規定でしょうか。必要以上のものにまで何でもセキュリティをかけるなど愚の骨頂です。小日程ファイルが流出して社会問題になるのなら必要ですが、見る必要性のある当事者たちが見られないような管理体系をだれが管理しているのでしょうか。必要以上の過剰な防衛的姿勢は確実に企業の生産性を落とします。また今まで長い間、小日程表が見られないことについて何もクレームをしなかった作業担当者の当事者意識のなさには寒気を覚えます。怒って下さい。ＱＣＤに関する重要事項の推移・変更などの発生の都度、メンバーにはコピーの現物を配布するようにして下さい。ＩＴに頼り切りでは本当に見ているのか見ていないのか分かりません。本当に重要な情報はコピーの現物を直接メンバーに配布すべきです。すべからく重要事項の押さえはアナログでフェイス・トゥ・フェイスの鉄則を守りましょう。

愚行＃７７　：リーダーによるパソコンをしながらの片手間のレビュー

【事例】『リーダーが多忙で時間を取るのが厳しい状況の中で、自分も含めてレビュー中にレビュアーがパソコンで他の作業をしており、レビュイーの発言を聞いているのか、いないのか分からないような場合があります。多忙さからの気遣いのなさだと思います』

☆人を軽視した愚かな行為

本人は気付いていないでしょうが、このリーダーの行為は不誠実以前の愚かで失礼な行為です。このようなレビューをしてみたところでレビュイーも気分が悪いだろうし、レビュアーのリーダーも何も頭に入ってこないでしょう。このような行為は仕事をなめ、人をばかにしているとしか思えません。このような人でもさすがにパソコンで作業しながら片目で上司と話をすることはないでしょう。人によって態度を変える人間は信用のできない人間です。このような行為を見かけたらちゃんと仕事と人に正面から向き合うように注意して下さい。

愚行＃７８　：無理な顧客要求を断りきれない

【事例】『顧客から依頼されたことに対して、スケジュールが厳しい等の理由で出来ない理由を説明し、断ったとしても、「無理と言われても困る」と切り返された場合、相手に対して、説得力のある理由で切返しができません。また、相手が伝える「困る」に対し、何が困るのかの具体的内容を聞けないままとなってしまいます』

☆困難な要求には困難な条件の提示を

相手が「それでは困る」と言うのは、すでにエンドユーザーに期限を切られている場合でしょう。逆にこちらが出来ない理由は何でしょうか？何か条件をつければ要求納期を達成できることはありませんか？下記に「見積納期３ヶ月、要求納期は１ヶ月」という事例を示します。こちらが提示した条件は大体次の通りでした。

１．暫定版リリースは２ヶ月後とする。正式リリースは３ヶ月後とする。

２．暫定版の品質保証はできないので、発生不具合に対する対外的な責任は全面的に要求側で持っていただくこと。但し不具合発生時には開発チームは実務面で全面的にバックアップする。店舗サポート業務にはサービス部署の支援をいただくこと。

３．暫定版の市場導入は実験店舗用とし、限定数店舗にしか導入しないこと。全店導入は正式版をもって行うこと。

４．短納期実現のために設計者および評価者の増員に対するコスト増を認めていただくこと。

５．仕様は現時点で凍結したものだけとし、追加・変更は次のリリースとすること。

６．本件納期に関して、顧客責任者と開発責任者との直接の話し合いの場を設けていただくこと。

この条件を提示する前に開発リーダーたちと話し合い、２ヶ月後に正式リリース版並の品質確保を可能とする体制を敷き開発着手しました。結果的に要求側および顧客には上記全ての条件を受け入れていただき暫定版においても不具合なしで稼働できました。　上記内容の全てを誰もができる訳ではありませんが、相手が超非常識な要求を突きつけてきた場合のこちら側の対応も通常のルールでは認められない内容で攻めるしかないでしょう。これは一種の条件闘争と言えるものです。但し、こちらの条件が認められたら絶対にもう後には引けませんので、確実な実現性の裏づけを持っておく必要があります。

愚行＃７９　：顧客に対する過剰サービス

【事例】『競合他社より有利に円滑に業務を進める為、非常識な要求でも過剰なサービス精神で対応してきました。例えば、自分の役割ではない他部門との交渉事や、機能の制限事項への例外的な対応などがあります。その結果、過剰なサービスが標準となってしまい感謝もされなくなりました。業務的旨味が無くなった現在でも過剰なサービスを要求され困っています。企業として、どのような労役を何処までどのように提供するかの統一見解ないしはルールがあれば教えて下さい』

☆自分勝手な判断に拠らないこと

非常識な要求を受け過剰なサービスを行うことについて、上長も合意の上でやっていることなのでしょうか？それとも個人的にやっていることなのでしょうか？自分が思うところの非常識な要求とか過剰なサービスとかは、本当に非常識または過剰なのかについて、リーダーや上長と話をする必要があります。その上で、確かに非常識や過剰だと判断されることならば、それらを実行すべきではありません。他社より優位に立つため、ないしは業務を円滑に進めるために行うべきことは、第一に品質・コスト・納期の確保です。

仕事の基本はあくまでも契約内容に書かれたもの以上でも以下でもありません。その契約をベースに何をどこまでやるのかの判断は自社の責任者の判断によります。自分勝手な判断に拠ってはいけません。このような問題については必ず上長と相談する必要があります。

愚行＃８０　：開発者における仕様誤解が多すぎます

【事例】『間もなく客先の立会い試験が始まるというのに不具合が止まりません。結局、立会い試験において機能の不具合が３件、仕様変更が２件発生してしまいました。不具合 3 件はすべて仕様の誤解が原因でした。メンバーの多くは類似したシステムの開発経験が無かったために、機能仕様書には詳しい記述がなくても経験者は常識的に知っているような機能について、仕様誤解をしている箇所がありました。また他のメンバーが詳細設計以降を担当した機能が、全工程に渡って不具合が多い状況です』

☆仕様理解に関する注意点およびポイント

全工程において不具合が多い最大の原因としては仕様理解ができていないと言うことだと思われます。特定客先の仕様について未経験者が多い開発チームで最初に行われなければならないことは、開発着手前の準備期間中に経験者から基本的な仕様に関してのレクチャーを受けたり、そのシステム開発における過去の失敗事例について説明を受けたり、実機を操作することによってシステムの動作を体感しておくなどさまざまな対応が可能です。

仕様理解に関する注意およびポイントは以下の通りです。

【仕様理解に関する注意点およびポイント】

①まずは仕様の全体像の把握から始めること。

②要求仕様調査時に疑問点・不明点の発掘を行い、不明点解消に向けて要求元に対して積極的なアプローチを取ること。

③仕様検討の段階で要求者と徹底的な仕様検討を行うこと。
④仕様決定のＱ＆Ａは直接対話による確認を行うこと。
⑤不明なことは直ちに分かっている人に聞くこと。
⑥要求仕様の背景や意味を必ず明確にしておくこと。
⑦習得した内容を、ドキュメントによって他のメンバーに伝えること。
⑧早期の仕様凍結を行うこと。
⑨基幹仕様未決定で開発に着手してはいけない。

愚行＃８１　：顧客とのコミュニケーション不足のため業務遂行の優先順番が分からない

【事例】『目指すべきことは「顧客満足」であるから評価テストの順番を顧客価値本位で行うべきですが、なんとなくの感覚で、自分がやりたい機能からテストを行う気分本位のスケジュールを組んでしまいます。原因は、顧客の要望を汲み取れない為に顧客の価値認識が明確に把握出来ていないことにあると思います。今後は顧客との密な打合せを行い、リスクの潜んでいそうな機能や不安に感じている部分のテストを洗い出し、顧客の求めるテストの順序を明確にしていく必要があると思います』

☆何が重要なのかは顧客に聞かなければ分からない

顧客と密接なコミュニケーションを行うための具体的な方法を考えて実行してください。顧客との密接な打ち合わせを継続的に行なうためには顧客との定例的な打ち合わせをきちんと業務プロセスの中に組み込んでおく必要があります。単なる思い付きで実行したりしなかったりではいつか実行しなくなるものです。時期的には業務プロセスの初期の段階に集中的に実行し、評価業務開始前には、評価対象仕様の重要度の順位および不明点・疑問点を全て明確になるようにしておく必要があります。また開発との連携が必須であることを忘れないで下さい。

愚行＃８２　：ベンダー側からの情報提供が遅い

【事例】『テストも終盤にさしかかった時に、想定外の店舗システムの構成変更を伝えられることがあると評価条件が崩れて困ります。原因としては、ベンダー側ＳＥ、開発担当者間の情報共有の悪さおよび下請け会社への情報提供の遅さにあります。また、評価チームにおいては、要求仕様書の記述をそのまま鵜呑みにしてしまう傾向があり、実際の運用の意味を理解していない場合も多いと思います。

テストの終盤において、店舗のシステム構成が増えたとの情報を知らされても評価システム構成を増やしてテストする時間も機材の余裕もありません。また想定していた店舗のシステム構成が減った場合には、余計なテストをしたことによる時間の浪費が発生してしまいます。 評価チームとしては社内の開発チームとベンダー間においてやりとりしているＱ＆Ａ表を活用し、少しでも早い情報入手を行いたいと思います』

☆積極的に仕事に取り組むこと

ベンダー側の情報提供の遅さにも問題がありますが、下請け会社としての自分自身におけるコミュニケーションに関する姿勢にも大きな問題があるように思います。社内の開発チームがすでにベンダー側とＱ

＆Ａ表にて不明点・疑問点について情報交換をしているのに、なぜ評価チームはそれに参加していないのでしょうか。また店舗構成の情報など仕事を受注する段階で本気で入手する気持ちがあればいくらでも入手できるでしょう。何なら顧客の店舗を自分の目で見に行ったらどうでしょうか。要するに受身で仕事をしているのか積極的な姿勢で仕事をしているかの違いでしょう。

愚行＃８３　：品質の低いチェックリストになってしまいます

【事例】『評価業務実施時に手直しや見直し作業が多く発生し、作業が効率良く行なえません。主な原因は品質の低いチェックリストにあると思いますが、なぜ誤字、脱字、コピペミス等が多い品質の低いチェックリストになってしまうのでしょうか。評価設計の見積り工数が根本的に足りないために、チェックリストの作成が雑な作業になってしまっているように思います。また見積りの甘さは、テスト内容の把握の曖昧さに原因があるように思います』

☆単純コピペの絶対禁止

誤字、脱字、コピペミスが多いのはチェックリストだけではなく設計書やソースコードについても同様です。はっきり言えることはこれらの作成者は、作成した後に一度も見返していないということです。見返してないという行為は"手抜き仕事"をしているということです。時間や工数が足りないなどという言い訳は全く通りません。時間がなかったから信号を無視して交差点に突入して事故を起こしてしまったということと同じでしょう。これらの手抜き行為で最も危険な行為は"コピペ"です。自分の観察や判断を一切行わず、どこかの誰かが書いたものを単純にコピー＆ペーストすることなど仕事をしているどころか、自分で不具合のもとを量産しているようなものです。不具合を発見する職務の人間が自分で不具合のもとを量産しているのです。もし流用したいソースコードやチェックリストがあったなら、その全文を自分で検証し添削を入れた後に流用すべきでしょう。流用元における入出力やその他の条件が異なっている場合が多いのにもかかわらず、何らの検証もなく"コピペ"する行為は開発・評価業務中における最悪の行為の一つであるという認識を持つ必要があります。このことは組織の全員の方々にお伝えください。

またいいかげんな評価工数の見積りの原因は、いいかげんな設計見積りにあり、いいかげんな設計見積りの原因は、いいかげんな要求仕様の把握にまで遡ります。評価チームにおいては設計チームと連携し、最上流の要求仕様の明確化および早期凍結に最大の力を発揮することで、妥当な見積り、妥当な工数の獲得を実現する必要があります。まずは自分の工程を開始する前準備として、不明点・疑問点を完全に解消する努力が必要です。

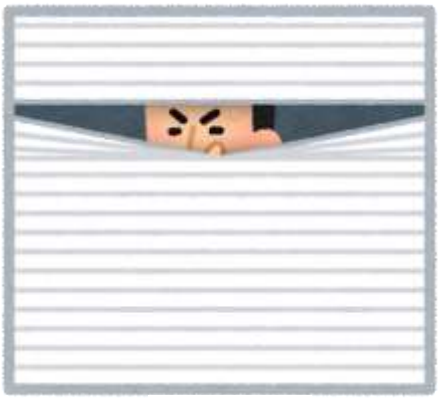
愚行＃８４　：不具合調査時に仕様の疑問点に気づく

【事例】『不具合調査時に仕様や機能についての疑問が発生することが多いのですが、なぜもっと前の設計工程などでそれらの疑問が出ないのかと思います。ベンダー側とのコミュニケーション密接にし、常に仕様に問題意識をもってレビューを行うことが必要だと思います』

☆要求仕様調査時に疑問点・不明点の発掘を

もっと早く問題に気づきたいと言うことは正しい考え方だと思います。それなら設計レビューや製造レビューよりもっと以前の要求仕様調査の段階のレビューが更に効果的でしょう。「何故このような機能が必要なのだろうか？」と思うことが最初にあり、その意味や背景を深く理解しようとすれば、いろいろな疑問点や不明点が浮かび、その後の設計工程・製造工程の精度は高まってくるに違いありません。何故この機能はこの仕様なのかという疑問は、その機能の運用がどのような意味であるかを知りたいと言うことに他なりません。その意味が本当に理解されていれば根本的な設計ミスや製造ミスも防げるし、適切なテストの実行も可能になります。

第４編　なんと保身的なことか



©いらすとや

はじめに

保身とは自分の身を守ることに他なりませんが、それでも行き過ぎた保身は自分の身を守るどころか逆に自分の身を亡ぼす元にもなりかねません。例えば、不慮の災難や事故に備えて保険に加入することは将来起こるかも知れない災厄に対するリスクヘッジとして有効に機能するでしょう。しかしそうであったとしても自分の収入に見合った保険料をはるかに超える金額の保険に加入してしまってはかえって自分の生活を成り立たないものにしてしまいます。保険の掛け過ぎは保険にはならず危険を招くことになります。

金額が目に見える場合にはその不都合さに誰でも気づきやすいものですが、日常の生活や仕事において遭遇する危険な状況の度合いの判断は非常に難しく、多くの場合人は過剰防衛的な反応や行動をしてしまいがちです。

過剰な自己防衛は自分を救うどころか、人を愚かな行動に走らせ自分に更なる危険を招き寄せる結果を招いているのです。

本章においては、保身すなわち過剰な自己防衛が招いている事例を検証し、妥当な自己防衛の道を探りたいと思います。

愚行＃８５　：カイゼン活動の優先順位が低い

【事例】『改善活動や、自己啓発となる作業を ToDo リストに入れ、やり忘れをなくすようにと管理し予定立てを行っていますが、基本的に業務優先の考えと、予定オーバーにより定時を越えた場合には残業をしての改善活動という点で気が引けてしまうことや定時後のモチベーション低下等により、後回しにしてしまいます。費用対効果で必要なものだと考え、優先度を意識的にあげ、エンドの日付を決めようと思います』

☆納期に追われたくなければカイゼン活動の実行を

現在の業務効率を何割も向上させる改善活動のどこに後ろめたさがあるのでしょうか。改善をやらずにいつもトラブルや時間に追われて、納期が強制的なタイムリミットとなり品質向上も図れずにいることの方がよっぽど後ろめたいことではないでしょうか。改善活動は業務における中核業務だという認識を持つ必要があるでしょう。

愚行＃８６　：カイゼン活動が進みません

【事例】『業務が多忙等の理由を付けて、プロジェクト業務以外である改善活動が積極的に出来ていません。改善活動は必要な事であると再認識し、「自分がやらないといけない」という気持ちでアイデアを積極的に出し、取り組んでいきたいと思います』

☆目の前の現実を自分の目で直視すること

あなたにとって改善活動はプロジェクト業務以外の仕事なのでしょうか？改善活動の第一義的な役割は、同じ失敗を二度と繰り返さないための対策を行うということにあります。このような仕事がどうしてプロジェクト業務以外のついでにやる仕事であるというような考えになるのでしょうか。多くの人たちは今でもこのような考え方から脱出できずに、あいも変わらず同じ失敗を繰り返し、旧態依然の生産性のあがらない仕事のやり方にしがみついています。実に愚かなことです。自分自身の目で現実を直視すれば、多くのひどい状況が見えてくるはずです。冷静に考えれば、改善活動はプロジェクト業務の中核的な業務であり、ついでの業務などではないでしょう。もっと自分自身の目で現実の問題を直視し、自分の頭でその問題を判断し、解決策を見つけ、自分自身で行動することが必要です。これが自律性の発揮ということです。

愚行＃８７　：カイゼン活動への取組み

【事例】『改善活動は、更なる負荷やリスクを背負うことになりモチベーションがわきません。全員で改善作業を行っていても、実務が忙しく、改善作業が中途半端な状態になり、改善活動を再開するきっかけを自分から切り出さない結果になってしまいます。改善作業が大事なことは十分にわかっていますが、実務でも忙しくなると、なかなか改善作業が出来ず、後回しにしようと思ってしまいます。改善による良い成果が体感できるかどうか疑問です』

☆カイゼン活動も実務のうちに入ります

改善活動は実務の内に入らないと思っている人がなんと多いことでしょう。建築に例えれば、傷や節くれだらけの板や柱を修復しないまま家を建てるようなものです。傷や汚れのついた柱の新築住宅に誰が住みたいと思いますか？ただプログラムを作りましただけが仕事とは言えないでしょう。失敗を繰り返さないための対策を考え実行することは改善活動の重要な活動の一つです。この活動を行わなければ、また柱に傷をつけ、板を汚すことになるのです。あなたたちが開発したプログラムに傷や汚れが見えないのでしょうか。改善活動は、余裕があればやってもよいというものではありません。

結局仕事に対する考え方に原因があるのでしょう。仕事に「やらされ感」を持っている内は最低限の仕事しかやる気は起きないでしょうし、仕事が「好き」なら何とか今より上手に仕事をやる工夫をしようとするものでしょう。さらに何かを改善するためには現状の負荷に加えて更なる負荷やリスクを背負う覚悟が必要となります。仕事が好きでなければ義務的に最低限の仕事だけをこなし、更なる負荷やリスクを負うことは全く嫌なことになるでしょう。

改善したいが時間が無くてできない、しかし改善はしなければならない、しかし時間がない、などとやりたくない思考がどうどう巡りしているだけです。簡単に言うと、その忙しい仕事のやり方を効率良く行える方法を考える必要があります。そうすれば時間の余裕が生れるだけではなく、効率的な仕事のやり方も同時に実現できるということです。現状を変えるためには最初に自分の時間や知恵を使うという投資が必要です。自分の負荷を増やさず成果を出したいという姿勢では現状は何も変えられません。仕事を好きになり自分の時間と労力を惜しまず仕事に励めば突破できます。

愚行＃８８　：パワーがかかるカイゼン活動になかなか取り組めない

【事例】『プロジェクトの改善活動について、「すぐに出来る、やる価値がある、時間やパワーを必要としない」ものは実行していますが、「なにをすべきか未定、やる前に準備が必要となる、パワーが必要なもの」はなかなか着手できません。必要だと思ったことは、時間がかかろうが、パワーが必要だろうが何が何でも成し遂げるという気持ちで取り組みたいと思います。また「チーム内みんなでやっていきましょう」感も大事ですが、時には引っ張っていく強引さも必要だと思います』

☆改善テーマ選択のコツ　―　費用対効果

パワーがかかると思える改善活動にはなかなか取り組めないでしょう。何もかも一挙に解決しようと思うと活動内容は膨らむばかりで取り組む気持ちも失せてしまいます。このような場合は、改善課題を細分

化し、それぞれにおける費用対効果を見積もり、その上で自分のチームの能力を勘案して費用対効果の大きな順に取り組むことをお勧めします。最初に取り組むものは、比較的に少ないパワーで、ある程度の成果が実感できるものがいいでしょう。１００の努力で１０の成果しか出そうにないものは、とりあえず除外してもよいでしょう。

どのような改善テーマを選ぶかということは、リーダーの能力に依存します。適切な改善テーマを選ぶためには、現状の品質や自分たちの生産性能力に関するデータが必要です、そのデータを基に、失われているコスト、改善に必要なコスト・パワー、見込み成果などの分析や計算をする必要があります。これによって初めて取り組むべき改善テーマの優先順位および最初に取り組むべき改善テーマを決定できます。気合だけでは改善活動はうまく進まないでしょう。現状をデータに基づいてよく観察してみれば、改善活動の対象となる課題は多く出てきますので、最初は成果が中程度で必要なパワーが小程度のものから始め、自分が決めた優先度の順に徐々にレベルを上げていくのが良いと思います。

愚行＃８９　：低レベルの発言が恥ずかしいので発言をちゅうちょする

【事例】『①低レベルの発言をして恥をかくのが怖いため、自分よりも技術力が優れている人がいる場合や、自分よりもまとめる力を持つ人がいる場合に発言を控えてしまいます。
②大人数での打ち合わせ時に、疑問点や不明点等があった場合に、その場で挙手して発言することができない。打ち合わせ後に個々に聞いて回ってしまう。
③質問内容で自分のレベルが知られることを恥ずかしいと感じる気持ちがある。
④質問を発声して、注目されることにストレスを感じる。
⑤打ち合わせ、会議などで、自分から意見、発表を言わない、又は、意見を求められるまで言わないことがある。
⑥自分の意見に自信がない時や、そんなことを今さら聞くのか、などと変に思われたくない。
⑦自分が言わなくても、誰か言ってくれるだろうなど考えてしまう』

☆発言すべきか否かは、恥ずかしいか否かではなく必要か否かで決める

誰しも初歩的な質問をすることは恥ずかしいものです。公式の顧客との打ち合わせなどにおいては特にそうでしょう。しかしある疑問が仕様の中核的な問題だと感じたら、「初歩的な質問かも知れませんが、○○はこの場合何を意味しているのでしょうか？」という聞き方をすることは何も恥ずべきことではありません。また社内の日常的な打ち合わせにおいては、自分の知らない言葉や技術に関してはどんどん質問すべきでしょう。たまにしか発言しない人にとっては自分の発言は非常なプッレッシャーとなりますが、いつも発言していれば他人にとっては初歩的な質問だったとしても誰も気にはしないでしょう。

「無知とは知識がないことではなく、問いを発することができない状態を指す」という言葉もあります。

また感情は消そうと思っても消せるものではありませんので恥ずかしいという感情は一端脇に置いておき、実務本意でどんどん発言してみたらいかがでしょうか。分からないことや疑問点はドシドシ発言して聞くことです。後で聞いて回るなど非効率の極みで、聞かれた方も何故さっき聞かなかったのかと不信感を抱くことにもなりかねません。　自分のレベルを知られることが恥ずかしいなどと思う必要はありません。日々の仕

事を通してあなたのレベルはすでに皆さんご存知のはずです。自分を良く見せたいとかいう気持ちも分からないではないですが、これは自分にとって何の役にも立たないどころか害になるものだと思った方が良いでしょう。自分を良く見せることではなく、自分が良くなることに注力した方がよほど自分のためになるでしょう。

愚行＃９０　：不明点があるのに確認をちゅうちょしてしまう

【事例】『仕様書に記載がされていない部分について、仕様確認を取らずに思い込みで製造してしまった事がありました。納品前や納品レビューの場にて確認をとりましたが、実は仕様面で問題があり、戻り作業となったことがあります。私は伝えたい事を簡潔にまとめて話すという事が苦手です。内容によりますが、相手に正しく伝える事が出来る自信があれば話をしますが、少しでも上手く伝える事が出来ないと思ってしまうと、なかなか行動に移す事が出来ません』

☆恥を恐れず不明点はその場で聞くこと

コミュニケーションのポイントは、どうしたら「伝わるか」に意識を置くこと及びそのためには事前に自分の中で内容を整理しておく必要があります。上手に話したいという意識は大切ですが、そればかりにとらわれないようにした方が良いでしょう。具体的に言うと、うまく話をしたいのなら、その話のポイントを事前にメモなどに書き出しておくことです。書き出せば、自然に内容は整理され、どの順に話をすべきか見えて来ます。整理されたメモを見ながら話をすることを繰り返していけばスムーズなコミュニケーションができるようになるでしょう。また、仕様確認の件も同様に仕様内容についての不明点や疑問点を必ずメモに書き出しておき、このメモに基づいて必ず相手に確認を取ることです。

愚行＃９１　：質問がしづらくて失敗を招いている

【事例】『指示内容の理解が不十分なのは自分の能力不足のせいだと思うと、疑問点や不明点があっても質問を出せません。その結果、理解不足のまま作業が進んでしまい失敗を招くことになります』

☆分からないことは、その場で質問すべし

”質問がしにくい”ということは個人的な感情の問題です。仕事の基本は、個人的な感情に流されず、仕事本位・目的本位に従ってやるべき仕事を淡々とビジネスライクに実行していくことにあります。そのためには毎日２０分でも３０分でも短時間の日次進捗会をチーム全員で実行してください。そのことによって話をすることに慣れ、自分の疑問点・不明点・主張などを気軽に話すことです。このことを毎日繰り返し実行すれば、分からないことがいけないとか質問がしにくいとかの感情もいつしか薄れていくでしょう。

愚行＃９２　：やる気がない会社へ常駐した時

【事例】『やる気がない会社へ常駐した時は自分もやる気が失せてしまいます。たとえば、スケジュールはあるが何が何でもスケジュールを守るという気がなく、スケジュール遅延に対する対策が単なる再スケジュールで、根本的に遅れに対する対策がなされていません。またこのようなプロジェクトでは上長が率先して指揮を執っていません。派遣者としての立場でどのように対処すべきでしょうか』

☆やる気がない派遣常駐先での自分の行動の仕方

自分の仕事に弊害や障害が出る場合は放置してはいけません。自分の仕事内容に関係する全てのことについての不明点・疑問点は全て明確にさせておく必要があります。やる気がない担当者であっても必ず期限を切って疑問点をすべて投げかける必要があります。主要な問題に関しては妥当な回答がない場合にどのような影響が発生するのかも警告しておく必要があります。これらのことは必ずＱ＆Ａシートなどの文書と口頭の両方で行う必要があります。それでも有効な回答が得られず問題化しそうならば、その担当者の上長に対して、貴社の上司から話をしていただく必要があります。自分の仕事については何としても完遂させる必要があります。

愚行＃９３　：中途半端に助けた時の責任問題

【事例】『チーム内における助け合いは、信頼関係を築き、協力し合うために必要だと思います。一人でやることには限界があり、それを超えるためにも必要です。それは最終的には組織力を高めると思います。しかし、中途半端に助けた時の責任が問題です。それがやり遂げられるかどうかが微妙な場合は助けることをためらってしまいます。メンバー同士の連帯も同様で、沈みそうな船に乗る場合、救うことができるかどうかが微妙な場合は壁になります。両方とも参画するときには躊躇してしまいます』

☆責任など問われません、できる範囲での支援を

確実に助けられると確信できなければ支援を躊躇するとありますが、責任問題はどうするのかなどと力んで考える必要はないでしょう。あなたが保有する技術と時間の一部を提供することで困っている仲間の負担の一部が軽減できればそれで良いのです。自分の力の一部だけで足らないと判断される場合は、他のメンバーからの支援も必要になるでしょう。そのような判断になったならリーダーや上司に相談の上、更なる支援者を募れば良いだけの話です。支援者が、相手の仕事の全てに対して責任を問われることはないでしょう。

あなたの１％の助力で息を吹き返す仲間やプロジェクトがあるのかも知れません。困っている仲間を助ける場合、その人が自分にとってどのような人なのかによって、あるいは自分の余力・能力によって、助けられる範囲はさまざまでしょう。それはその時々の状況によって判断するしかありません。少ししか助けられないなら少しでも、中ぐらいなら中ぐらいで、絶対助けたいなら全力で助けることです。一番悪いのは身の回りの人々に対する”無関心”さ、や特定の人に対する”無視”の態度でしょう。仕事人、プロフェッショナルは人の好き嫌いでは仕事はしないものです。

愚行＃９４　：惰性に流された仕事

【事例】『特定の目標もなく慣れによる惰性での仕事を行っている場合はやる気が低下します』

☆惰性的な仕事をやめるには目標の設定を

惰性で仕事をしている状態とは、自分の仕事内容に対して良し悪しの自己判断をまったく放棄している状態です。どのような仕事においても、その仕事のあるべき到達結果、すなわち目標を想定した上で、自分の仕事内容がそれに合致しているかどうかを常にチェックする必要があります。また惰性的な仕事から脱するためには、なぜ惰性になっているのかの原因を特定し、何らかの改善活動を行う必要があります。まずは自分ないしはチームのＱＣＤに関する具体的な短期的目標を定めるところから始めて下さい。目標を設定するためには、現在の自分ないしはチームにおけるＱＣＤの状況を具体的な数値で把握する必要があります。

愚行＃９５　：一人で担当する仕事のやり方には問題がある？

【事例】『規模が小さいプロジェクトでは要件定義、システム設計を一人で担当する事があります。開発においては、要件定義からの視点およびシステム設計からの視点がありますが、一人で行うと同じ視点で見てしまい、対応範囲の抜けが発生することもあります。顧客担当者においても、兼務している事はありますが、二人以上で対応する事が殆どです。 どんなに規模が小さいプロジェクトでも、担当は二人以上必要だと思います』

☆一人でできることは一人でやろう

一人でできる小さな規模の仕事をわざわざ二人でやるメリットはあまりないと思います。システム開発の仕事は歴史的に見ると、最初は少ない人数で全工程を担っていましたが、規模が大きくなるにつれて、どんどんと分業化・専門化が進んできました。分業化・専門化の大きな欠点はいわゆる専門バカと呼ばれるもので、自分の領域については詳しいが、他の領域については全く興味を持たない状況を生み、複数の領域間の矛盾を生み出してしまうことにあります。その典型的な例が、個々のモジュールは完璧に動作したと思ったが、結合したら全く使用に耐えない結果になったというものです。相談者における考え方は、その延長線上にあるもので、一面無理もないとは思いますが、やはり一人で開発できるものは可能な限り一人で実行することが、技術者として多面的および統合的視野を獲得する良いチャンスだと思います。

リスク分散の視点では複数の担当者で実行した方が安心でしょうが、技術者育成の視点で見ると、一人で全工程を担当することはまたとない成長の機会のはずです。要件定義・設計・製造・評価・リリース・稼動までの一連の全ての工程を一貫して経験することは、重い責任と緊張感にさらされますが、半面やりがいがあり非常に楽しい業務です。これによって全ての工程における実際的な問題をその当事者として意識することができ、分業体制下においても他の工程の問題点や苦しみを理解できるようになります。これができてはじめて一人前の技術者だと言えるのだと思います。これは大きなプロジェクトにおいては全く不可能なことで、小さなプロジェクトにおいてしか実行できないことです。なかなか得がたい経験だと思いますがどうでしょうか。

愚行＃９６　：相手との信頼関係の構築が難しい

【事例】『最近は意識してコミュニケーションをとるようにしていますが、以前は同じチームの人ともあまりコミュニケーションをとっていなかったため、信頼関係を築けていなかったと思います。仕事における信用・信頼の基準は、その人に安心して仕事を任せられるかどうかにあると思います。また、他の人と考え方などにずれがある場合に、自分の考えが正しいと思い込み、相手の考えや意見を受け入れることや尊重することができませんでした』

☆相手の思いを感じ取るところからコミュニケーションを始めること

仕事における信用・信頼の基準として「仕事を任せられること」だと言われることが多いものですが、何の条件もなしに仕事をまかせられる人などそうそう居るものではないと思います。リーダーにおいて重要なことは「仕事の任せ方」にあるでしょう。相手のレベルによっては細かい指示が必要にもなるでしょう。一言いえば何でもやってくれる人などほとんど居ません。

「仕事の任せ方」とは、言い換えれば「指示の仕方」「コミュニケーションのとり方」に他なりません。相手のレベルや性格に応じた指示の仕方、確認の取り方、話の仕方をする必要があります。このことは対顧客、対上司、対同僚との間のコミュニケーションについても同様です。最初に相手が求めることを感じ取り理解することが重要です。

愚行＃９７　：顧客の導入目的・要望の吸い上げが出来ていません

【事例】『顧客の導入の真意を理解するためには、フェイス・トゥ・フェイスの話し合いが必要ですが、顧客との間に、ベンダー側開発担当およびベンダー側ＳＥが入っており、なかなか会える機会がないのであきらめています。ベンダー側開発担当者へ希望は伝えてあり、機会があれば同行させて頂けることになっていますが、なかなか実現しません。次の案件の受注が決まっていない段階で顧客に会うことはタイミング的に難しいものがあるようです。しかしながらベンダー側開発担当者と親密な関係を築き、次案件の話も普通の会話の中から聞きだせるようなことから始めたいと思います』

☆仕様知識の拡大および品質確保の実績を積み上げること

自社との間に複数の組織が介在している場合、顧客の要望の把握は非常に困難で、直接会話の機会もほとんどないのが実状だと思います。ベンダー側においては、要件をまとめる力がないという印象を顧客に与えてはいけないとか、多重請負を行っていることを知られたくないとか、ベンダー側組織の面子などの理由により、下請け開発会社の担当者を顧客に同行したくないというのが本音だと思います。下請け会社の担当者の同行が認められるのは、顧客がこれらの理由を気にしない場合および下請け会社の技術者抜きでは要求仕様の凍結が不可能な場合などだと思われます。今後、仕様知識の獲得を十分に広げ、問題のないリリース実績を積上げ、ベンダー側の担当者に教えるくらいのレベルに到達できれば顧客との会議に出席できる機会は必ず来ると思います。

愚行＃９８　：開発機材の貸し出し期日を守らないベンダー側担当者

【事例】『リスクを抽出し、それに対する回避策として期限を定めても、その後のアクションが足りません。対ベンダーに期限を設け、機材の提供依頼を行い、間に合わなかったとしても、結局リリース日が延びるわけもなく、開発・評価のスケジュールの無理な短縮を行うだけとなってしまいます。ベンダー側に要求する前から結果が想定でき、何も変わらないことが多いため、「どうせだめだろう」と思ってしまい、あきらめてしてしまいます。 ベンダー側担当者が動かなければ、その上司の耳に入るように行動する、という方法も考えられますが、こういう手段ではなく、ベンダー側担当者を動かす方法を考えていきたいと思います。機材を用意する関連部署に連絡したら後は知らないといった姿勢が強いベンダー側担当者が多いように思います。こういった姿勢をやめさせるためにどのようなことをすればいいのか答えがなかなか見つかりません』

☆組織的な対応を

機材の提供が遅れて一番困るのは相談者自身なのではありませんか。遅れた日数を、リリース日を変えずに自分たちのスケジュール調整でどのように調整するのでしょうか。開発や評価に必要な日数を削るということでしょうか。それによって起ることは開発品質や評価品質の低下しかないでしょう。それでも「どうせだめだろう」と言ってあきらめて良いのでしょうか。まずは、そのようないい加減なベンダー側担当者に直接何度でもクレームを言って下さい。機材を用意する義務と責任があるのはベンダー側のプロジェクトリーダーです。その様な責任者不在の丸投げ仕事にいつまでも忍従していてはいけません。直接ベンダー側担当者に言っても聞き入れられないのなら、自社のマネージャからベンダー側の責任者にクレームを言うことをちゅうちょしてはいけません。これらのことは道理にかなった妥当な行動です。ベンダー側にも自社にも顧客にとっても利益につながる行動です。丸投げを正当化できる理屈などどこにもありません。対ベンダー側に関する、同様の問題を他のリーダーやメンバーと共にリストアップし改善策を検討し、組織的な対応を行う必要があります。

愚行＃９９　：不具合修正残が評価テスト進捗に影響する

【事例】『開発における不具合修正が残っていることなどで、しばしば評価業務が滞ってしまいます。各担当の方には改善をお願いしていますが一向に改善されません』

☆前工程に対する積極的な関与を

前工程側担当者の責任意識の欠如に起因した問題です。 発見済みの不具合、特にその後の評価が進められないようなロードブロック的な不具合をいつまでも修正できないでいるような状態を開発チームに許しておいてはいけません。開発チームも忙しいのだからという言い訳は全く無意味なことです。すでに工程は評価に移っているにもかかわらず、不具合修正のスピードを上げることのできない開発チームがスケジュール全体の足を引っ張っています。この問題は評価担当の問題ではなく開発担当の問題です。不具合修正のスピードに追いついていない原因としては、不具合が多すぎること、開発人員不足、開発能力不足、情報共有の悪さ、などが考えられますが、いずれにしても真因を特定し、改善を急ぐ必要があります。評価チームといえども、よく見受けられる待ちの姿勢ではなく、評価業務が始まる以前から前工

程の品質状況や進捗状況に積極的に関与する姿勢が必要です。

愚行＃１００　：評価設計ノウハウの継承ができていません

【事例】『評価機器の環境構築のノウハウは情報共有出来ていますが、評価設計のノウハウの情報共有がきていません。「やりたいな～、作りたいな～」で止まっています。日々の業務をこなすことで一日が終わってしまい、これらの改善活動へ着手できません。その結果評価メンバーの成長を図れていません』

☆できない訳とやる方法

常識的に考えてできそうもない事を除いて、あることを実行したいと思っているにもかかわらず実行できない原因には二つが考えられます。一つは、本気でやる気がない場合です。ボーナスを支給するから受け取りに来なさいと言われたら、みなさんは万難を排して受け取りに行くことでしょう。

もう一つは、具体的にどういう風に実行して良いのかが分からない場合でしょう。評価設計のノウハウを継承したいと思っていても、評価設計自体の理解が浅ければ評価設計マニュアルを作成することもできません。これでは評価メンバーの成長を図る前に自分の成長を図る必要があります。

何かをやろうと思った場合、一挙にりっぱな完成形を目指そうとしないことです。評価設計マニュアルを作りたいと思った場合、まずどのようなマニュアル構成にするのかの目次作りから始めてみたらどうでしょうか。書きたい内容を読み手の気持になってどのような項目の順に並べたら良いのか考える必要があります。その後、各項目について自分が最も得意とするものから毎日数行でも書き進めることです。自分の不得意領域については得意な人に依頼すれば良いでしょう。この作業に毎日の３０分を割くことは可能でしょう。一週間で約２時間、一ヶ月で約８時間、半年で４８時間となります。本当にやる気ならばできることでしょう。 もしこのマニュアルが完成し、他のメンバーに有用なものであったならば自分たちが投資した時間の何倍もの効果を組織全体に及ぼすことができるでしょう。