ITプロジェクト

あなたはなぜ

ヒトの話を

聞けないのか？

伝わるコミュニケーションの方法●多聴多読のすすめ

PMファクトリー2017

目次

第2章　言わザルの巻　p14

1．　意見を述べる　p15

はじめに

わたしたちの日常のコミュニケーションの状況を振り返ってみると、ITエンジニアに限らず実はほとんどの人が、自分はコミュニケーション問題を抱えていると思っているのではないでしょうか。実際、コミュニケーションにおいては聞く・話す・読む・書くという四つの基本的なスキルに加えて考える力が必要とされ、このような難しい技をスイスイとこなす人などそうそういるわけがありません。交渉事やプレゼンが難しいのは当たり前のことで、自分だけがコミュニケーション下手で全くダメだと思うのは良い考えではないでしょう。

開発現場でよく聞かれるコミュニケーションの悩みとして次のようなものがあります。

- 人と話をするのが苦手。
- 大勢の人がいる場では意見を言い出せない。
- 自分の考えや意見をうまく伝えられない。
- 仕事の指示の仕方がうまくいかない。
- プレゼン能力を上げたい。

果たして、上手に話すことだけがコミュニケーションなのでしょうか？

ソフトウェア開発の現場におけるコミュニケーション

それではソフトウェア開発の現場におけるさまざまな問題の中でコミュニケーションの問題がどの程度の割合を占めているのでしょうか。

下記は、筆者において収集した500の事例におけるさまざまな問題の多さのワースト5を示したものです。

件数別のワーストリスク

失敗の原因	件数／%
1. コミュニケーション不良	66件　13%
2. リスク回避の失敗	65件　12%
3. 手抜き問題	46件　9%
4. 人材育成／ノウハウの継承問題	42件　8%
5. ドキュメント・ベース開発の欠如	38件　7%

ソフトウェア開発プロジェクトにおける問題の件数が最も多いのはコミュニケーションの不良でした。これから見ても、コミュニケーションの不良がプロジェクト全体に及ぼす影響の大きさが分かっていただけることでしょう。この事実は、コミュニケーションの不良が個々の開発者たちの業務品質、製品品質、生産性に膨大な悪影響を及ぼしており、さらにはプロジェクト組織をコントロールするリーダー、プロマネおよびSEなどにおけるコミュニケーション能力が、その責任や役割を果たすためにどれほど重要なのかを物語っています。

見たものがすべて

「聞いてびっくり見てびっくり」ということわざの代表的な一例として大昔（1969 年）に大橋巨泉氏が実演した万年筆のコマーシャルがありました。巨泉氏は譜面台を前にその新発売の万年筆を使いながらこう言ったのです。「みじかびの　きゃぷりきとれば　すぎちょびれ　すぎかきすらの　はっぱふみふみ」「わかるネ？ブハハハ・・」。これにはテレビの視聴者はみな仰天したものです。わかるネ？なんて言われても、エッ？？というのが大方の反応でしたが、CM放映後その

万年筆は爆発的なヒットとなりました。言葉の意味が全然分からないのにみんなに通じたのです。特に若い世代には。
このCMは「百聞は一見にしかず」と言われるように、視覚情報は聴覚情報を圧倒するということを見事に実証してみせました。このCMにおいて音声情報は意味不明でひとを驚かせるためだけに機能し、映像情報が本来の意味を伝えていたのです。極端を言えば音声情報は全くのデタラメでも視覚情報がしっかりしたものならば意味は伝わるということが分かりました。
人間の感覚は、聞いたものよりも見たものに支配されやすいのでしょう。

同様のことをノーベル賞受賞の行動経済学者ダニエル・カーネマンは次のように言っています。
「限られた手元情報に基づいて結論に飛びつく傾向は、直感思考を理解するうえで非常に重要である。この傾向は、自分の見たものがすべてだと決めてかかり、見えないものは存在しないとばかり、探そうともしないことに由来する。システム1（直感）は、印象や直感のもとになっている情報の質にも量にもひどく無頓着なのである。この『自分の見たものがすべて（What you see is all there is）』は、英語の頭文字をとって、WYSIATIという長たらしい略語が作られている。」（ダニエル・カーネマン、『ファスト＆スロー』（上）p129）

上記のエピソードは、人間のコミュニケーションにおいては音声情報よりも視覚情報が圧倒的な支配力を持っているということを示しています。それにもかかわらず、わたしたちの開発現場では視覚媒体の一つであるドキュメントがおろそかにされ、いまだに口頭ベースの貧弱なコミュニケーション中心・ソースコード中心の開発しか行われていないようです。

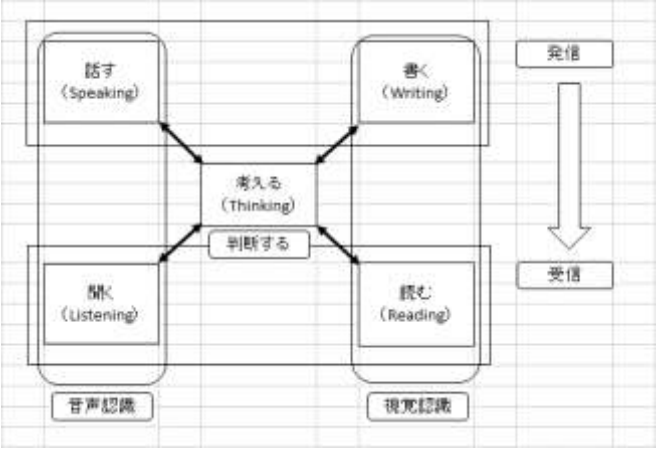
読み書きソロバン
生きていくために必要な三種の神器は「読み書きソロバン」だと昔から言われ続けてきました。この三つができれば無事に生活することができると代々の親たちは子供たちに言い続けてきました。そういうわけで、ソフトウェア開発を仕事とする私たちもそれにならって、コードを読むこと、コードを書くこと、ロジックを組むことに専念してきましたが、それだけではどうにもうまく行かないことが多すぎます。同じ問題が英語教育の分野でも言われており、中学・高校の通算 6 年にもわたる英語の教育にもかかわらず、読み書きは多少できたとしても、「話す」「聞く」についてはほとんどできないのが実態です。この例からも分かるように本来のコミュニケーションは、「読む」「書く」だけではなく「話す」「聞く」を加えた四つの能力によって成り立っています。「書く」と「読む」はワンセットで、「書き言葉」と言われ、「話す」と「聞く」はワンセットで、「話し言葉」と言われています。
この二つのセットをあんばい良く使いこなせて初めて相互のコミュニケーションが成り立ちます。

この「書き言葉」によるコミュニケーションは、人類が文字を獲得した紀元前四千年ころに始まったと思われます。人類の歴史600万年前に比べるとつい最近のことです。
最近になって獲得された書き言葉の文字認識という能力は、話し言葉に比べて脳に多くの負担をかけることが最近の脳科学でも明らかになっています。話し言葉の理解はすべて音を認識することで行われていますが、書き言葉の理解は少し複雑で、ひらがな・カタカナなどの表音文字は視覚で認識されたあと、脳内で音に変換され認識されているようです。さらに漢字などの表意文字は画像として視覚認識されたあと、脳内の音声処理とは別の場所で処理が行われているようです。つまり漢字・ひらがな・カタカナ混じりの日本語文字の認識には音声認識と画像認識が同時並行的に行われていることになります。
このようなことから、「話す」「聞く」こと」よりも「書く」「読む」ことの方が大きなエネルギーを必要とするため、私たちのコミュニケーションはどうしても「話す」「聞く」に偏りがちになります。

コミュニケーション能力の構成

コミュニケーション能力は、おおまかに言えば、先に述べた「話すこと（Speaking）」「聞くこと（Listening）」「書くこと（Writing）」「読むこと（Reading）」に加えて「考えること（Thinking）」が重要な働きをしています。 普通は前者の四つの能力がコミュニケーションの能力の範疇としてとらえられていますが、本書においては四つのコミュニケーション能力を統合的にコントロールしているという意味で「考えること（Thinking）」を第五番目のコミュニケーション能力として位置づけました。

これらの五つのコミュニケーション能力の関係を図示すると次のようになります。



【コミュニケーション能力の構成】

「話す」機能および「書く」機能は、情報の発信機能であり、「聞く」機能および「読む」機能は、情報の受信機能といえます。当然のことながら、発信がなければ受信は存在しません。

また、「話す」機能と「聞く」機能は、音声認識によって働き、「書く」機能および「読む」機能は、視覚認識によって働いています。両者は脳科学的には別々の場所で処理が行われおり、「話したこと」「聞いたこと」は忘れられやすく、「書いたもの」「読んだもの」は長く記憶されるという事実があるということを覚えておいたほうが後々役に立つことでしょう。

「話す」「聞く」「書く」「読む」というコミュニケーションの四つの機能はすべて「考える」機能という情報の意味付けあるいは意味の解釈を通してはじめて生きた情報として通用することになります。ただ話すだけ、ただ聞くだけ、ただ書いただけ、ただ読んだだけの行為は意味のない空虚なものとなります。

語学のコミュニケーション能力別のレベルを示す国際標準規格であるCEFR（Common European Framework of Reference）では、「聞く」能力および「読む」能力は「理解する」能力と定義し、「話す能力」および「書く能力」は、「やりとりする」能力および「表現する」能力と定義しています。

いずれにしても、この五つの能力のレベルは人それぞれに異なっており、それらの能力の不完全さはソフトウェア開発においても多くの問題を引き起こしています。

コミュニケーション能力進化の階層構造

コミュニケーション能力の進化の過程は子どもの成長の中に読み取ることができます。最初に聞くことに始まり、聞いたことをまねて口に出すことで話すことを体得し、言葉の意味を教わりながらだんだんと会話の力を成長させてきます。それから数年がたち次に物の形として文字を視ながらそれを読み、その意味を理解する訓練を受けます。文字が読めるようになったら、次に同じ形の文字を自分の手で再現することを教わります。

このようにして人は、その根源的な能力である「聞く力」から「話す力」を獲得、会話能力を得ることができます。しかし文字の読み書き能力は、口頭による会話能力よりも遥かに高度な訓練でしか得られない能力であるために、二十一世紀の現代においてさえ世界の平均識字率は75%程度だと言われています。

この成長・進化の過程は、逆にみると衰退・退化の過程とも言えます。人間の老化の過程で失われていくコミュニケーション能力は、「書く」→「読む」→「話す」→「聞く」の順になっているのでしょう。

組織における衰退・退化においても、コミュニケーション能力の喪失は同じような順番をたどります。そう言うわけで開発組織におけるコミュニケーション能力の劣化は、最初にドキュメントの劣化となって現れてくるのでしょう。 全体的なコミュニケーションの能力が劣化した組織においては、コミュニケーション能力の根源的な能力である「聞く」ことを活性化するところから再出発すべきでしょう。

【コミュニケーション能力進化の階層構造】

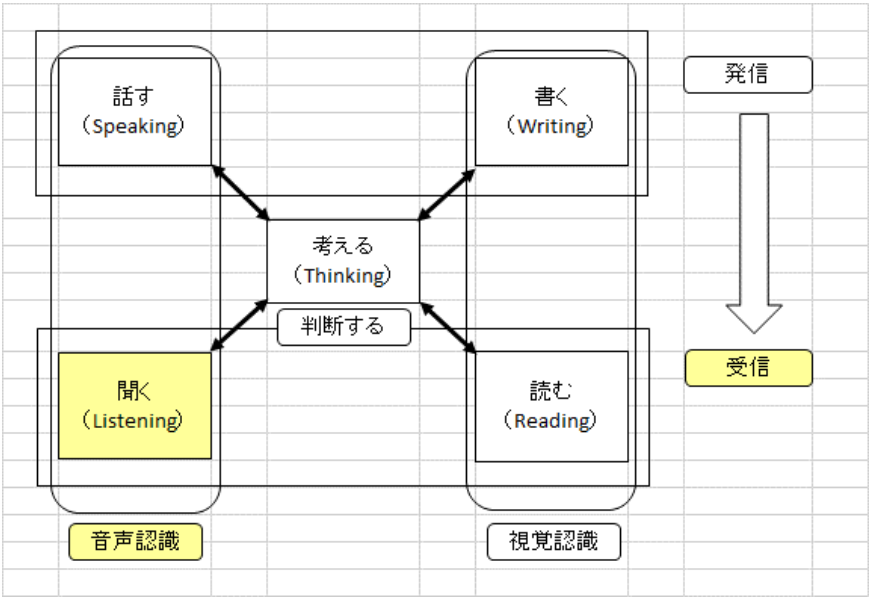
以下の本文においては、開発現場におけるコミュニケーション問題について、その能力の発生進化の順に従って、「聞く」コミュニケーション、「話す」コミュニケーション、「読む」コミュニケーション、「書く」コミュニケーション、「考える」コミュニケ―ションの順に、現場における具体的な事例に基づいて対応策を明らかにしたいと思います。

第1章　聞かザルの巻

　コミュニケーションの四つの能力の中で意外に難しいのが「聞く」コミュニケーションでしょう。日本語で会話をしている場合には気が付きませんが、英語を習っていた時にネイティブの先生から教えられたことは、まず「よく聞く」ことから始めましょうということでした。そうして正確な音が聞き取れれば、次は、その聞き取った音を正確に口に出すことができるということでした。聞き取れない音は口に出して再生させることはできないということは、耳の聞こえない人はしゃべることも困難だという事実と同じ原理です。

「話す」ことは比較的に容易でも、人の話を注意深く聞き、その意味や意図を正確に把握することは非常に難しいことです。よく「聞く」能力は、高いコミュニケーション能力の基礎になるものだといえます。

　本章では、「聞く」能力を、受動的な聞く（話を聞く）能力と積極的な聞く（質問する）能力の二つに分けて考えていくことにします。



１．　話を聞く（受動的な「聞く」）

一方的に自分の言いたいことだけを話す人は嫌われます。なぜかといえば人はみな自分の言いたい事があり、それを人に聞いて欲しいと思っているからです。コミュニケーションもギブ＆テイクの世界であり、それに違反した行為は人々に受け入れられません。どのような聞き方がコミュニケーションを促進するのかについて考えてみたいと思います。

Case＃１　人の話を最後まで聞くことが出来ない

相手の話に途中から割り込むことが多く、一方的に自分の意見ばかり述べるような結果になってしまいます。相手の話を聞かないという悪印象だけが残ってしまい意思疎通の障害になっています。

【アドバイス】 話の途中で割り込むことは礼儀違反

相手の話の途中で割り込むことは洋の東西を問わずマナー違反です。話を中断された相手は必ず感情を害しますので、続けて話そうと思っていた大事な話をする気持ちも失せ、割り込んだあなたの話についても聞き耳を持たなくなります。基本は相手の話がすんだ後に自分の話をすることです。相手の話が要領を得ずダラダラと続くような場合には、相手に断ったうえで話を中断し、内容を整理したうえで話をするようにお願いすることです。

他人との関係性をうまく保つためには、ギブ＆テイクであっても、テイク＆ギブではないでしょう。

One point　Lesson　：　自分の話を聞いて欲しければ、最初に相手の話を聞くこと

Case＃２　会話嫌いの後輩との会話が進まない

後輩との日常会話で、相手に話しかけても反応が鈍く、すぐ話しが終わってしまいます。会話嫌いな後輩とどのように接したらいいでしょうか。

【アドバイス】 相手との体験の共有がなければコミュニケーションはとりにくい

先輩後輩間に限らず会話がはずまない原因の一つとして、相手と自分の間で喜怒哀楽などの感情的な共有体験がない場合は一向に話ははずまないものです。つまり相手はあなたを気の置けない仲間だと思っていないということでしょう。もっと相手のことを知る必要があり、日常的に苦楽を問わず行動を共にし、思いやりをもって相手に接していれば日常会話は自ずとはずむでしょう。

One point　Lesson　：　相手の問題としてではなく、自分の問題として捉えること

Case＃3　部下の意見を十分に聞けない

信用・信頼できる部下の要件は、「指示した仕事のスケジュールを守ってくれること。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。

【アドバイス】 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

そんなできの良い部下などめったにいません。逆に、あなたは、「指示内容を的確に伝えていますか？妥当なスケジュールを確保していますか？部下の行き届かない部分にプラスアルファの支援をしていますか？部下の困りごとを聞き出す努力をしていますか？」。部下に対する要求は、そのまま形をかえてあなたへの要求となって返ってきます。これは指示者と指示される者の間に必ず発生する相互義務というものです。相手の信用・信頼を得るためには、この相互義務をお互いに果たす必要があるでしょう。

One point　Lesson　：　信用・信頼関係は相互義務を果たすところから

「聞く」コミュニケーションのチェックリスト

相手の話を聞くにあたって次のチェックリストを試してみよう

〇：できている　△：どちらともいえない　×：できていない

☑ Check List　＃1　：　**人の話を聞く場合のチェックリスト**

☐　人の話を最後まで聞いているか。

☐　相手の問題を自分の問題としてとらえているか。

☐　指示内容を相手が本当に理解したかを確認しているか。

【ふり返りメモ】

2.　質問する（積極的な「聞く」）

人の話を良く聞くことを傾聴と言いますが、ただ黙って聞いているだけでは傾聴にはならず相手が話している内容を十分に理解できてはじめて傾聴したことになります。理解できないことや不明に思ったことは、折り返し相手に質問してみなければ相互の理解は深まりません。質問をするということは積極的な「聞く」行為だと言えます。未確認のまま終わってしまったことで失敗を招いたいくつかの事例を考えてみたいと思います。

Case＃4　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

●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。
●質問を発声して、注目されることにストレスを感じる。
●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。
●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。

【アドバイス】 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、〇〇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聞くべきでしょう。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しないでしょう。
「無知とは、知識がないことではない。無知とは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。

One point　Lesson　：　聞くは一時の恥、聞かぬは一生の恥

Case＃5　不明点があるのに確認をちゅうちょしてしまう

仕様書に記載がない部分について、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。伝えたい事を簡潔にまとめて話すという事が苦手で、相手に伝える自信があれば話をしますが、そうでない場合はなかなか行動に移す事が出来ません。

【アドバイス】 口頭会話に自信がなければメモの力を借りること

上手に話したいという意識は大切ですが、それにばかりとらわれずに話のポイントを事前にメモなどに書き出しておくことが必要です。仕様の不明点や疑問点も事前に書き出しておけば、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くことで自然に相手に伝わりやすい会話ができるようになります。メモに基づいて相手に確認を取ることを習慣化すれば必ず良い結果を生むでしょう。

One point　Lesson　：　メモに語らせるということ

Case＃6　質問がしづらくて失敗を招いている

　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きます。

【アドバイス】 分からないことは、その場で質問すべし

　能力不足と質問ができないこととは、別の問題です。能力不足を補うために多くの質問をするという姿勢が真っ当な職業人のありかたでしょう。質問ができないということは、個人における、恥の感情を避けたいという過剰な自己防衛的な姿勢がもたらしたものです。このような感情的ないしは情緒的な姿勢では仕事をやり通すことはできません。プロは、自分の感情はいったん脇に置いておき、仕事として必要なことは必ず実行するという人たちのことです。このような姿勢のことを目的本位な仕事のやり方といいます。 自分の質問力や会話力を強化するためには、日次情報共有会議などで質問や疑問点を口に出すことに慣れるのが一番の近道でしょう。

One point　Lesson　：　理解不足だからこそ質問する必要がある

Case＃7　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

　途中からプロジェクト参加した場合、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。

【アドバイス】 分からないことは、その場で質問すること

　途中参加のプロジェクトでは不明点や疑問点が多くあるのは当然のことです。みんな忙しくしている状況では質問しづらいことだと思いますが、聞かなければ分からないことは聞くしかありません。短期間で自分を戦力化するためのアクションを次にチェックリスト形式で示します。

- ● プロジェクトに参加する前に行っておくこと
- □ メンバーの誰が何を担当しているのかを聞いておくこと。
- □ 自分に期待される役割を聞いておくこと。
- □ プロジェクトの進捗も含めて問題点を聞いておくこと。
- □ 自分の疑問点・不明点を列挙しておくこと。
- ● 最初の一週間で、下記を集中的に行うこと
- □ 仕様の骨子や全体像を大まかに把握すること。
- □ 自分の疑問点・不明点を集中的に個別に聞くこと。その時間を設けていただくこと。
- □ 予定の時間外でも、浮かんだ疑問点・不明点は遠慮せず直接質問をすること。
- ● その後に行うこと
- □ 毎日の短時間情報共有会議（実行されていなければ実行を促す）を利用して、その後に発生した疑問点・不明点を明らかにすること。

One point　Lesson　：　ものごとの基本的な理解は短期間に集中的に行うこと

Case＃8　似たような二つのクラスの存在による失敗

　ライブラリーの中に似たようなクラスAとクラスBがあり、なぜ同じようなクラスが二つもあるのかと疑問に思いましたが、それ以上詮索することなく一方を不要としてしまいました。しかし実際は両方が必要だとレビューで指摘されました。システム知識不足および経験不足でした。

【アドバイス】 ”なぜ？”の問いかけの実行を

　確かに知識や経験の不足は本質を見誤らせることになりますが、自分において何に対しても十分な知識や経験があるという場合は少ないものです。今回の問題における本質は、知識・経験の不足などではなく、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？」という質問を発せられなかったのかという一点にあります。常識的に考えて同一機能のクラスが二種類あるわけもないでしょう。その時に、相手に「なぜ同じようなものが二種類あるのですか？」と質問をしていたら、知識も増えた上に問題を引き起こすこともなかったでしょう。「まあいいだろう」というよう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、相手に「なぜ？」を投げかける必要があります。感情に流されずに、「なぜ？」という疑問を自分自身に問いかけ、それでもわからなければ他人に問いかける姿勢は失敗の回避のみならず自分の成長に非常に大きな効果をもたらします。聞かないことによる不利益と聞くことによる利益のどちらが得かは明らかです。 ”なぜなぜ手法”は、発生した問題の解決に有効に働きますが、問題が起こる前でも、自分の疑問点や不明点に対してはもっと有効に働くでしょう。

One point　Lesson　：　何か変だと思ったら、「なぜ？」と口に出して言ってみること

Case＃9　恥をさらすようで他人に聞けない

　分からない箇所があった場合に周りに対してコミュニケーションがとれず、アクションが遅れてしまいます。他人に仕事について聞くことに恥の様な考えが身についてしまっています。今後は、億劫な気持ちを抑えて、人に聞くことに慣れていくようにしたいと思います。

【アドバイス】 仲間との毎日のコミュニケーションを

　仕事について不明な点や分からないことを聞くのは仕事そのものであり恥でもなんでもありません。聞かずにあいまいなままにした結果、バグを作ることが恥です。都度いろんな質問をするのが気後れするのなら、毎日チーム内で15分から30分程度の短時間日次会議を定例化することを上長に提案してはどうでしょうか。そうすれば不明点を聞くチャンスが毎日自動的にあることになります。この短時間日次会議のやり方はチーム全員で、①昨日に実行した内容の報告、②今日の実行予定内容の報告、③今抱えている問題の提示、などについての全員での短時間連絡会をすることです。もちろんリーダーはこれらの報告内容に適切なアドバイスを行う必要があります。

One point　Lesson　：　必要な質問も仕事の内

Case＃10　仕様の想定違い

　要求仕様書の内容に疑問点がありましたが、他の人から間接的に聞いていた情報から想定して設計を進めたところ理解が違うことが判明してしまいました。

【アドバイス】 仕様は想定するものではなく確定するもの

　仕様に疑問があったのに、なぜ仕様担当者ではなく他の人に聞いたのでしょうか。仕様担当者に直接確認することにちゅうちょする理由は何でしょうか。それが問題の真因です。たとえ相手が苦手な人であろうがなかろうが、その人に聞かなければ分からないことはその人に聞くべきでしょう。仕事の遂行はあくまでも事実本位で、すなわち解決すべき問題を理解し、それらを着実に解消していく姿勢が必要です。相手に対する苦手意識などの感情本位で仕事を進めてしまうと、「〇〇だろう」とか「〇〇のはずだ」とかのような情緒的判断が多くなり、失敗の原因を自ら作り出してしまうことになります。仕様は、想定するものではなく確定するものです。

One point　Lesson　：　感情的な反応は選択を誤らせる

Case＃11　顧客の要求に対して、明確で有益な回答ができない

　システムの全体像の把握不足や仕様の理解不足のために顧客の要求に対して、あらゆる可能性や影響を考えた回答ができていません。

【アドバイス】 不明なことは直ちに分かっている人に聞くこと

　要求の意味や本質が分からないまま何かを無理に答えようとしてはいませんか。何でも自分ひとりだけで答えようとしてはいませんか。要求内容の意味が分からなければその場で要求者に質問してみてください。いくつかの質問をすることで、要求内容の意味も徐々に理解でき、何が重要で何が本質なのかも分かってくるでしょう。また社内の経験者に聞くことも有効です。一度で分からなければ何度でも切り口を変えて聞くくらいのしつこさが必要です。ただし同じことを何度も聞くと無能技術者だと見なされます。

　これらのコミュニケーション態度はすぐに実行可能であり、これを一年間続けることができれば、仕様知識は格段に増えることでしょう。

One point　Lesson　：　知っているふりをするよりも、他人から学ぶこと

Case＃12　客先要望の理解不足

客先から依頼された品質強化テストとして自社担当アプリケ―ション部分のテスト結果を報告しましたが、他社担当部分のミドルウェアも含めたテストを行っていない点を指摘され再テストとなってしまいました。

【アドバイス】ドキュメント・ベースの仕事を実行すること

この事例においては二つの問題点があります。一つ目はドキュメント・ベースの開発を行っていないことです。品質強化テストの内容について要求内容を明記した書面がおそらく存在していないと思われます。存在していれば、事例のような基本的なミスが起こるはずもありません。仕事の依頼内容についての書面を出さない方も悪いのですが、それを確認しようともしない担当者も愚かでしょう。仕事の依頼・請けは契約行為だという職業人としての基本的な認識が欠如しています。

二つ目の問題は、この開発担当者は毎日の会話の中で顧客の言っていることをほとんど聞いていないようです。品質強化テストが要求されるということは、品質が悪いということであり、要求される前までに何度も問題点を指摘されているはずです。当然のことながらミドルウェアとの関連性のないアプリケ―ションなど存在するわけもなく、自分の担当アプリケーションについてのみのテスト報告など受け入れられるはずもありません。

One point　Lesson　：　顧客の心配事を共有する努力を行うこと

Case＃13　顧客とのコミュニケーション不足で業務の優先順番が分からない

顧客価値の優先順位が明確に把握できていないため、自分のやりたい順に評価テストを行ってしまい効果的な評価業務ができていません。

【アドバイス】何が重要なのかは顧客に聞かなければ分からない

顧客要求の何が重要なのかを把握するためには、顧客との密接なコミュニケーションが必要です。とくにそれぞれの要求仕様の目的や意味および背景を顧客とのコミュニケーションの中で理解しておけば、どの仕様や機能が顧客にとって重要なのかは自ずと明確になってきます。

多重請負構造の中にあって実行は難しいと思われますが、仕様検討段階における開発者とエンドユーザーの直接会話は何らかの方法で必ず実行する必要があります。

One point　Lesson　：　伝言ゲームはやめること

「聞く」コミュニケーションのチェックリスト

質疑応答にあたって次のチェックリストを試してみよう

〇:できている　△:どちらともいえない　×:できていない

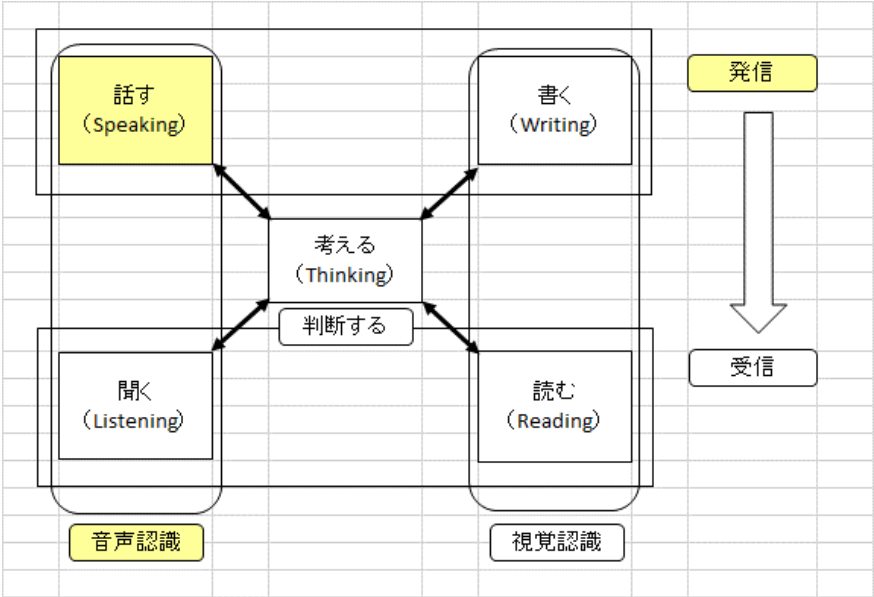
☑Check List　#2　:　**質疑応答に関するチェックリスト**

☐　要点をまとめたメモを用意しているか。
☐　発言すべきか否かを、恥ずかしいか否かではなく必要か否かで決めているか。
☐　相手が苦手な場合、必要な質問を避けてはいないか。
☐　疑問点・不明点をその場で質問できているか。
☐　分からないことを誰にも聞かずに放置してはいないか。
☐　変だと感じた時に“なぜ”の問いかけを行っているか。
☐　ものごとの理解は短期集中的に行っているか。
☐　日次情報共有会議などの実施によって質問力、発表力、理解力を向上させているか。
☐　顧客の発言内容の真意を把握できているか。
☐　顧客と密接なコミュニケーションを行っているか。

【ふり返りメモ】

第2章　言わザルの巻

コミュニケーションの最初のつまずきは、この「話すこと」にあります。他人と接することの苦手な人はふだんの雑談さえもスムーズにいかない場合があり、ましてや客先との会議やプレゼンテーションとなると非常な緊張感を伴い、話したかったことの十分の一も話せなかったという経験は誰にでもよくあることです。大部分の人にとっては、仕事において他人と話をするということは、失敗が許されないという心理的な圧力のために大きな緊張感を伴うものですが、この避けがたい圧力をいくらかでも緩和して「話すべきことを話せる」ようになるにはどうすればよいのかということについて、「意見を述べる」局面および「指示・依頼を伝える」局面の二つについて事例に沿って考えていきます。



1.　意見を述べる

「意見を述べる」とは、相手の話に対しての反応であったり、諸問題に対しての自分の考え方の表明であったりしますが、いずれにしても他人に何かを分かるように伝えることは非常に知的な思考力と緊張感を伴うものです。君の言っていることは良く分からないと言われてしまうと、ほとんどの人は大きなショックを感じてうろたえてしまいます。仕事においては始終「意見を述べる」局面に遭遇しますので、どうせ逃げられないのならその問題と正対し、少しでもましな意見の表明の仕方を現場の事例を通して学びたいと思います。

Case＃14　周りの反応が怖く有効なコミュニケーションがとれない

周りにどう思われているかが気になり、周りからの反応も怖いため有効なコミュニケーションが取れず、解決手段も浮かびません。結局、自分自身が臆病なためだと思います。またそのため、報告や相談を切り出すタイミングが遅くなります。心の中でこの様な点を問題だと感じても結局動けません。

【アドバイス】 他人の思惑だけに従って自分の行動を決めてはいけない

自分の所属する集団の反応によって自分の行動を選択するという姿勢は、日本人全般にみられる慣習の一つです。その良い面としては集団全員の思考・行動の方向性を一つにまとめることで困難な目標を達成する強力な武器になりますが、反面個々人の自主的な思考・行動を制限することにより自律性を損なうという負の局面をもっています。 いわゆる「空気を読む」という行動はこの延長線上にあり、過度に他人の思いを意識することで個人の自由闊達な思考・行動を押さえ込んでしまいます。他人の思いを過度に意識するくらいなら、他人に対する思いやりある思考や行動をとることの方が両者にとって有益なことでしょう。 ”良い仕事をするためには何をすべきか”ということに気持ちを集中させれば、自ずと報告・連絡・相談も自発的に頻繁に行われ、良いコミュニケーションができるはずです。

One point　Lesson　：　空気を読まずに仕事を読もう

Case＃15　臆病な性格がコミュニケーションを阻害している

自分自身の臆病さと、慎重な性格が情報の伝達もれや確認ミスなどのコミュニケーション問題を引き起こしています。根本的な性格を変えるとまでは行かなくても、仕事用の性格を手に入れられるよう、どうすればいいか悩んでいます。

【アドバイス】 コミュニケーションに都合の良い性格などはない

コミュニケーションに都合の良い性格などはありません。一見陽気でしゃべり上手な性格がコミュニケーションに有利なように思えますが、このような性格は半面では一方的あるいは雑なコミュニケーションに陥りがちです。どのような性格でも必ず正負の両面を持っています。自分の性格は変えられません。変えられないのであれば、それを活かす方法を考えた方が良いでしょう。仕事を進めるのに必要なことは何かを常に考え、伝えるべきことは伝え聞くべきことは聞くようにすることです。コミュニケーションに際して、自分の性格である臆病さを慎重さに活かすことで丁寧なコミュニケーションが図れるでしょう。

One point　Lesson　：　性格は変えられないが、行動は変えられる

Case＃16　コミュニケーションに対する不安感が強い

話が盛り上がっている中で自分の意見を言うのが下手です。自分の発言で空気が冷めてしまうのが怖く、相手の言葉を深読みし、理解を誤ってしまいがちです。相手の発言内容を確認したら「当たり前のことを聞くな」と思われそうで、コミュニケーションに対する不安が強く困っています。

【アドバイス】 不安感はいったん脇におき、必要な行動を始めること

コミュニケーションに対する不安の感情は誰でも普通に感じるものであるということを理解する必要があります。自分だけが特別に強く感じているという思いは人間性に対する正しい認識ではありません。自分にとって不都合な感情だと思えばコミュニケーションを避ける行動につながってしまいます。普通の感情の一つだと理解し、仕事としてやるべきコミュニケーションは何としてでも実行するという心構えで行動すれば不安感や脅えを克服できるでしょう。仕事においては自分の感情はいったん脇に置いておき、すべからくビジネスライクに仕事として割り切って伝えるべきことは伝え、聞くべきことは聞くようにしたいものです。 不安な感情はいったん脇におき、まず行動から始めてみることです。

One point　Lesson　：　不安にとらわれると思考・行動は停止する

Case＃17　苦手な相手とは話をしたくない

自分とうまが合わない人、あるいは強圧的な人とうまくコミュニケーションがとれません。

【アドバイス】 コミュニケーションは、人の好き嫌いではなく必要か否かで決める

職業人として仕事をしている中において、自分の性格に合わない人や強圧的な人などとも一緒に仕事をせざるを得ない状況に遭遇することも多々あります。

あるタイプの人に対する嫌悪感や苦手意識の元になっているのは、みんなに良く思われたいとか、みんなに好かれたいという気持ちの裏返しなのでしょう。この気持ちは実現不可能な感情です。すべての人に好かれることなど誰にもできることではありません。そのようなことよりも、好き嫌いの感情は一端脇に置いておき、「なすべきことをなす」「必要なことはする」という実務本意の姿勢で仕事に取り組むことが良い結果を生みます。あくまでも仕事に必要なことに焦点を絞ってコミュニケーションを進めるような姿勢のことを”ビジネスライク”な仕事のやり方と呼びます。

One point　Lesson　：　仕事はまずはビジネスライクに

Case＃１８　丁寧なコミュニケーションが苦手

自分の思い、感情を人に伝えるのが下手であることから伝えたいことが伝わらず、相手に誤解を招いてしまいます。面倒くさがりのため、かいつまんで物事を伝えてしまう癖があるので、じっくり腰をすえて話をするように心がけてはいますが、うまくいきません。

【アドバイス】 仕事においては感情を伝えることより、まず事実と背景を伝えること

他人とのコミュニケーションにおける丁寧さを確保するためには、結論を急がず、自分の感情に振り回されず、仕事において冷静な目を持つような努力が必要です。

仕事をうまく進めるには一貫した論理性と他人の納得が得られる情緒性の両方が必要です。理詰めばかりで性急に攻めても反感を買うばかりで、話しの裏に隠れている背景などの情緒的な話も取り混ぜて話をした方が相手の納得も得やすいものと思われます。

仕事におけるコミュニケーションは、感情を伝えることではなく、事実とその背景を伝えることに重点を置いた方が良いでしょう。

One point　Lesson　：　自分のことよりも仕事の内容について話そう

Case＃１９　仕事以外の話題についていけない

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感じることがあります。

【アドバイス】 教養の幅は人間性の幅を広げる

仕事に直結した話題について答えられない場合は気まずいでしょうが、時事問題について答えられなくても取り立てて気まずく感じる必要はないでしょう。無理に何か答えようとして話に詰まるよりも、その時事ネタについて知らないことを恥じつつ、その内容について「教えてください」と言う一言が大切なことだと思います。知らないことを尋ねられた時の最もよい対応方法は、相手に教えを請うことです。多分相手は、あなたが知らないことをバカにするのではなく、あなたに教えることに喜びを感じることでしょう。これは仕事に直結した話題でも同様です。

一方、新聞や情報誌などの時事の話題や特集などは自分の教養の幅を広げる役に立ちます。専門バカと言われないように世情の色々なことについても見聞を広げることは自分の柔軟性を増強してくれます。また何らかのテーマに興味をもった場合は、それについての書籍を読むことも自分の栄養になります。本を読む習慣は、まともな文章を書く能力や他人に対してものごとを説明する能力を育ててくれます。教養を広げることは楽しいことです。

One point　Lesson　：　専門バカはつまらない

Case＃20　レビュー、電話応対、知らない人との会話が苦手

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。例えば、レビュー中、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。

【アドバイス】 習うより慣れよ

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。予測できないものごとに対する正常な反応なのです。正常な感情を否定することは、その後の自分の行動を歪んだ不適切なものにしてしまいます。やるべきことは、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。このような行動を積み重ねていけば、自然に冷静な対応ができるようになるものです。

具体的に有効な姿勢や対応は次のとおりです。①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業務知識や一般常識を豊富にすること、などです。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ます。

One point　Lesson　：　仕事とは他人の困りごとを解決すること

Case＃21　想定外の話題に対応できない

全然想定外のところに突っ込みを入れられ、これじゃ打合せする意味が全然ないと全否定されると、言い返せなくなります。

【アドバイス】 コミュニケーションは勝ち負けではない

想定外の質問に対して反論することは難しいことです。あなたの頭の中は相手との勝ち負けだけに支配されているようです。この場合、あなたがすべきことは「何か言い返す」ことではなく、その想定外の自分が理解していない領域の知識について相手に教えを請うことでしょう。「打ち合わせをする意味がない！」というような侮辱的な言葉を受けても、感情的に反応することをやめ、「済みませんが教えて下さい」の一言が言えれば物事は好転するでしょう。

仕事相手、特に顧客とのコミュニケーションにおいて相手の話についていけないという場面が日常的に多く発生する場合は、あなた自身における勉強不足や情熱不足に原因があると思われ、なお一層の奮励努力が期待されます。

One point　Lesson　：　質問に答えられなければ、逆に質問をすること

Case＃22　正論・理想論に押されて自分の意見を主張できなくなる

正論というか理想論を強く語られ、うまく反論できず、言い負かされてしまいます。何を言っても、自分の知らない専門用語で言い返されたり、強く否定されたりで、手も足も出なくなります。

【アドバイス】 感情的な反発ではなく更なる学習を

他人の正論・理想論に反論できない理由は、自分にそのテーマについての持論や知識がないということでしょう。言いくるめられてくやしいのなら勉強し、専門知識を磨いて下さい。感情的に反発しているだけでは永久に言いくるめられる立場から脱出することはできません。もしその場で一矢報いたいのなら、自分が理解していないその「専門用語」とやらについて相手に頭を下げて教えを請うことです。もし相手がハッタリであなたを言いくるめようとしているなら化けの皮がはがれるでしょうし、本当にその知識に精通しているのなら、あなたは新しい知識について概略を知ることができます。いずれにしてもあなたは得をすることになります。感情的な反発は何の得にもなりません。

One point　Lesson　：　議論に勝ちたければ、もっと勉強すること

Case＃23　自分の主張の根拠について適切に説明できない

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。

【アドバイス】 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としては、仕様・機能の知識不足、運用知識の不足および技術知識の不足などがあります。

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、普段から継続的な学習を積み重ねておき、学習の結果を資料としてまとめておくくらいの努力は必要です。

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先に相手に、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方が有効です。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ような実践が必要です。

One point　Lesson　：　無い袖は振れない

Case＃24　上位の人に対して意見を言いにくい

自分より年輩の方に対して、意見があるのに言い出せません。「相手がどう感じるのか？嫌に思わないだろうか？」「今後やりにくくなったらどうしよう・・・」という思考が一歩を踏み出せない一因だと感じています。相手が間違っている・相手に問題があるとは思いますが、自分の意見に自信が持てません。

【アドバイス】 間違いを恐れるよりも一歩を踏み出すこと

経験が浅いうちは先輩たちからいろいろと間違いを指摘されますが、それは仕方のないことだと思います。誰しも間違いを指摘されることは気分の良いことではありませんが、同じ間違いをしない勉強をさせてもらったと考えれば、一時の恥は一生の得に変わります。反対に、間違いの指摘を自分に対する攻撃と思ってしまい、他人からの意見も受け付けない、自分の意見も言わないような姿勢になってしまえば自分の成長は止まってしまいます。

自分の意見に自信がない場合でも、何か変だと思ったら、その疑問を相手に伝えるべきでしょう。自分から先にボールを投げてみれば、相手からまたボールが投げ返され発展的なコミュニケーションになる可能性が高いと思います。なにごとも間違いを恐れずに一歩前に踏み出すことで新しい道が開けてくるものです。

One point　Lesson　：　物言わぬは腹ふくるるわざなり

Case＃25　相手が忙しそうにしているとコミュニケーションを遠慮してしまう

自分が忙しい時はお客さんも忙しいと思い、貴重な時間を頂くよりはまず自分で解決案を模索した方が良いと思ってしまい、コミュニケーションを遠慮してしまいます。

【アドバイス】 必要なコミュニケーションは双方の効率を向上させる

生産性の低い職場のリーダーたちは常に”忙しい”というオーラを出しまくっています。これはどの会社においてもよく見られる光景です。できの悪い人ほどそのようです。相手がそのような状態にあったとしても、どうしても確認しなければならないことには気後れすることなく時間を割いていただきましょう。それをしなければもっと忙しくなってしまうことを相手に伝えましょう。

One point　Lesson　：　考えても分からないことは分からない

Case＃26　コミュニュケーションやプレゼン能力を向上させる方法を教えて

開発のすべての工程におけるさまざまな問題の早期解決にはコミュニュケーション能力やプレゼン能力が必須だと思いますが、それらの能力の向上には何が必要でしょうか。

【アドバイス】 コミュニケーション・プレゼン能力の向上は実地体験の積み重ねで

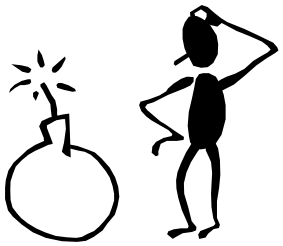
コミュニケーション能力やプレゼン能力を向上させるには、それらを実際に何度も実行し体験するのが最も良い方法です。日々の活動において、積極的にコミュニケーションを実行してください。たとえば日次会議などでは議長役を積極的に引き受け、また顧客プレゼンに先立って社内におけるプレゼンを数多くこなしてみてください。コミュニケーション能力やプレゼン能力は、ただ単に立て板に水でしゃべれるとか、かっこ良く話しができるとかと言うことではありません。相手に伝えたい内容を本当に自分がはっきりと理解し、それらの内容をきちんと整理し、誰が聞いても分かりやすく、何が重要なのか明白であることなどがポイントになります。本当のコミュニケーション能力やプレゼン能力の向上には幅広い教養と知識が必要となりますが、一歩ずつ実体験を積み重ねつつ多くの書物を読む努力を継続する必要があります。

One point　Lesson　：　流暢な弁舌、イクオール高いコミュニケーション能力というわけでもない

Case＃27　仕事の優先順位の変更にあたっては事後報告でも良いか

緊急性が高い、または確認や合意が必須の内容に対しては基本的には確認しているつもりですが、内容によっては自分や相手の業務多忙等を理由にして、優先順位を変更しても事後報告で済ませる場合があります。優先順位を変更しないと効率が悪く、問題が発生する場合、内容によりますが事後報告だとしても早めのタイミングで連絡を取ることを心掛けます。

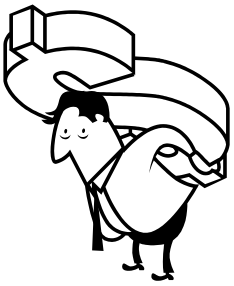
【アドバイス】 自分が負える責任の範囲を考えて行動すること



優先順位の変更にあたっては必ず事前報告および関係者との合意が必要です。このような行動がリスク回避と呼ばれるものです。緊急対応の割り込みが入った場合、もともとの最優先事項に必要な時間が不足することは明白です。優先的割り込み作業の要求を受けた場合は、必ずそのインパクトについて要求者との合意および上司の了解をとっておく必要があります。合意や了解なしで実行した結果に対して自分で責任が取れるでしょうか。事前報告にかかる10分間とリスクが問題化した場合のインパクトはどちらが重いかは明らかでしょう。

One point　Lesson　：　背負わないですむリスクは背負わないこと

Case＃28　伝えてはいけないこと、伝えるべきこと



顧客の管理職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。伝えることでお客さんが安心するのか、不安になるのかをまず考えるようにしたいと思い、改善しようとしています。考えていることが表情に出てしまう面についても、仕事用の顔として落ち着いていられるよう、変えていこうと考えています。

【アドバイス】 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く必要があります。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。

One point　Lesson　：　話して良いこと悪いこと

Case＃29　口約束だけでは責任の所在を問えない

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ります。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います。

【アドバイス】 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、必ずメール文書等での指示・依頼文書を取り交わしておくことです。文書ベースの仕事のやり方を習慣化することは重要なことです。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ことが肝心だと言えます。

One point　Lesson　：　言ったことは忘れられやすく、記録されたものは忘れられにくい

Case＃30　レビューで補足説明が多すぎ、本筋の説明から逸脱する

　設計書等のレビュー時、レビューを受ける立場として補足説明が多くなってしまい、本筋の説明から離れてしまいがちです。レビュアーがどこまで理解しているかが不明なため、後から説明を求められるのを防ぐためには、関連することは全て話をした方が良いと思っていますが、補足ばかりで本筋がわからなくなってしまう事は「過ぎたるは及ばざるがごとし」のように思えます。本筋を見失ってしまうのであれば、補足しないほうがまだ良いのかも知れません。

【アドバイス】 事前に要点を整理してからレビューを受けること

　説明したい内容が整理されてなく、すべて口頭で話をしようとするとこのような状態に陥りがちです。このような状態を避けるためには、説明資料において、これだけは必ず説明しなければならないと思うもの、すなわちその設計の骨子を事前に抽出しておく必要があります。次にレビュー開始時に、まずレビュアーに対して、例えば「今日説明する設計の骨子の項目は、〇〇と△△の二点です」という具合に、レビューの対象となるものの全体像を示しておくことです。最初に話題の焦点を明確にしておけば、その後に続く話は脱線しにくくなります。後は、設計資料に基づいて、設計骨子の重要な順に、それぞれ要求仕様は〇〇でした、それを実現する方法は△△にしましたという説明を行い、疑問点や不明点があればレビュアーの判断を仰ぐことが必要でしょう。相手の理解度を先回りして、あれこれと先に話をすることはかえって混乱を招きます。レビュアーの質問があれば、都度それに対して答えて行くだけで良いでしょう。

One point　Lesson　：　準備不足は混乱を招く

☑ **意見を述べることに関するチェックリスト**

　自分の意思の表明にあたって次のチェックリストを使ってみよう。
　〇：できている　△：どちらともいえない　×：できていない

Check List　＃3　：　他者に対する不安・恐怖度のチェックリスト

☐　他人の思惑だけに従って自分の行動を決めてはいないか。
☐　コミュニケーションに都合の良い性格を望んではいないか。
☐　不安感に縛られて、必要な行動を始められないことが多くないか。
☐　人の好き嫌いでコミュニケーションをしてはいないか。
☐　仕事においては感情を排して、事実と背景を伝えているか。

【ふり返りメモ】

Check List　#4　：　**話題の広さのチェックリスト**

☐　教養の幅を広げる努力をしているか。
☐　対話の実践に努めているか。
☐　コミュニケーションで勝ち負けにこだわってはいないか。

【ふり返りメモ】

Check List　#5　：　**意見表明のチェックリスト**

☐　否定的な意見に対して感情的な反発ではなく更なる学習を積んでいるか。
☐　うまく説明を行うために事前の準備をしているか。
☐　間違いを恐れるよりも一歩を踏み出すことに努めているか。
☐　良好なコミュニケーションで業務の効率を上げているか。
☐　実地体験によるコミュニケーション・プレゼン能力の向上に努めているか。
☐　自分が負える責任の範囲を考えて行動しているか。
☐　伝えて良いこと悪いことの判断は的確か。
☐　重要な取り決めは文書にて相互の合意を得ているか。
☐　事前に要点を整理してからレビューを受けているか。

【ふり返りメモ】

2.　指示・依頼を伝える

職場における「話す」コミュニケーションの重要な役割として部下や関係者に対する仕事の指示や依頼があります。口頭による指示や依頼は誤解や漏れが多く、言った言わないの論争の原因ともなっています。ここでは、口頭による指示・依頼の弱点およびその対処方法について考えてみたいと思います。

Case＃31　質問に答えすぎなのかも知れない

質問に対して、一聞かれたら十答える積りで対応していますが、もしかすると”そんな情報は要らない”と思われているかも知れません。やり過ぎでしょうか。

【アドバイス】 要・不要は相手に聞くこと

万全を期したいという気持は良く分かりますが、一つ聞かれたら普通それに関する注意事項なども含めて二つや三つの付加情報も伝えてあげるのが適当でしょう。自分で多過ぎると思うならば、相手に、その他の〇〇や△△についても知りたいかどうか聞いてみることです。まずは、自分の言いたいことではなく、相手が何を聞きたいのかを聞くことから始めたほうが良いコミュニケーションにつながるでしょう。

One point　Lesson　：　過ぎたるは及ばざるがごとし

Case＃32　一度に多くのことを伝えようとする結果、相手を混乱させてしまう

質問に対して、質問の本質以外の部分まで情報を伝えて相手を混乱させてしまいます。口頭で質問された場合、回答の準備が出来ている訳ではないので、後から後から伝えたい事が出てきてしまいますのでなかなか順序立てて説明することができません。伝わったかどうか、不安な場合は、簡潔な文章、または既存資料の情報をメールで連絡するようにしています。

【アドバイス】 相手が最も聞きたいと思われることを最初に伝えよう

口頭による会話では一度に多くのことを話したくなるものです。特に不意打ち的な質問を受けた場合、自分の中で正解を探す時間を稼ぐために色々と他の話題を出してしまうこともあるでしょう。このような場面を少なくするには、普段から仕事の中の懸案事項や問題点について良く考えておき、頭の中で整理をしておくことです。まずは、相手が聞きたいことだけについて答えるように努力したいものです。いろいろな条件や見解については相手の反応をみてから話をすればよいことです。これは報告書などのビジネス文書でも同様で、懸案事項の全体像を把握した上で、最初に相手が聞きたい結論を語ることが重要です。

One point　Lesson　：　一度に全てのものを望まないこと

Case＃33　忙しい時、一方的な自分の意見の押し付けになってしまう

忙しいときに、メンバーとのコミュニケーションが希薄になった状況の中で、打ち合わせ等で自分自身が相手の意見をよく聞くこともせずに、一方的に説得するばかりになってしまい、なかなか皆の納得を得ることができません。

【アドバイス】 忙しい時ほど、ていねいなコミュニケーションを

多忙でコミュニケーションが不足している時が、コミュニケーションが最も必要とされる時です。忙しければ忙しいほど、短時間で意思疎通ができるように、話を整理しポイントを押さえ、誰でも分かるような話し方で、書かれたものによって説明をし、相手からは、直ちに理解できなかった点について質問を出してもらうことが必要となります。いくらあせって自分の言いたいことだけ言っても、相手が理解してくれなければ仕事はまともには進まず、結局やり直しなどで余計な時間と労力を使うだけです。密接な双方向の意思疎通が最も必要な時は、超多忙な時であることを肝に銘じておく必要があります。

One point　Lesson　：　一人合点では仕事は進まない

Case＃34　伝わったと思った内容が伝わっていなかった

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。

【アドバイス】 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、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」のもめごとはよくあることです。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、相手に指示内容をその場で復唱させることで、指示内容が正しく伝わっているかどうかを確認する必要があります。さらに追って指示内容に関するドキュメントを渡しておく必要があります。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。

One point　Lesson　：　言いたい事は、まず伝わらないと思うこと

Case＃35　口頭確認に依存した結果、インターフェース接続に失敗した

自分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担当者間で違っていました。

【アドバイス】 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

他の機能とのインターフェース条件は必ず文書にて明確にし、お互いに合意しておく必要があります。機能のインターフェース問題は、往々にして担当者同士のコミュニケーションというインターフェースの不良が原因となる場合が多いものです。複数の担当者間におけるコミュニケーションのポイントは次の通りです。

・機能間で互いの責任範囲を明確にしておくこと。
・各レビューにて指摘された事項を漏らさずチェックしておくこと。
・両者間で、なぜなぜ分析を行い、問題点を共有すること。
・中途で発生した問題についても口頭のみで確認せず共有できる文書による確認を行うこと。

One point　Lesson　：　モジュールのインターフェースの前に人間のインターフェースを

Case＃36　あいまいな指示による勘違い

「A欄とB欄の表示文字数がことなるため、同じ文字数にすること」という口頭による指示を受けたが、どちらの文字数に揃えるのかを確認しなかったため再度の修正が必要になりました。

【アドバイス】 仕様の記述は論理記述にて

どちらの桁数に合わせるのか？という疑問が湧いたはずなのに、なぜ聞かなかったのか不思議に思われます。このように日本語の話し言葉は非常にあいまいな表現が多く、英語の口語ではあり得ないことです。さらに日本語文章においても、あいまいな表現が多く、論理性に欠けやすいため、仕様に関する指示やドキュメントにおいては数値や論理記号、表、図などを使用する必要があります。例えば、「A以上」という日本語は「≧A」で表し、「Aより大きい」は「＞A」で表せば誰も間違わないでしょう。

さらに要件定義書、仕様書、不具合票など開発におけるすべてのドキュメントは数値や数学的論理記号を多用し日本語記述のあいまい表現を一掃する必要があります。仕様の書き方や分析に関して、ロジックや数学の記号や書き方を採用することを強くお勧めします。

One point　Lesson　：　論より証拠を

Case＃37　新規作業者への説明不足による業務不履行

新規作業者に対するプロジェクトの運用ルールの説明不足が原因で、実行されるべき作業が行われていませんでした。

【アドバイス】 プロジェクトの運用ルールは文書化しメンバー全員に伝えること

プロジェクト業務運用（作業手順、報告手順等）に関する基本事項はすべてまとめて文書化しておき、その文書に基づいて都度新規作業者のみならずメンバー全員に周知徹底しておく必要があります。口頭だけでの指示説明では言い忘れや誤解が生じます。ドキュメント・ベースの仕事のやり方に切り替えましょう。

One point　Lesson　：　組織運用は統一ルールを示した文書から

Case＃38　個人に口頭で伝えたことは複数の人に伝わらない

個人に対して口頭で伝えたことを文書として残していないので、複数人に対して、同じ内容を十分に継承できていない場合があります。文書化するのに手間がかかりますが、口頭での説明ではイメージを伝えるのが難しいため、伝えたい本意をきちんと継承できていないと感じます。

【アドバイス】 仕事の情報伝達の基本は文書やメモ書きによること

開発に関する情報は、チーム全員による共有が基本です。特定の個人にだけ伝えておけば良いと思っていたことが、実は他のメンバーにも伝えておく必要があったということはよくあることです。情報共有のもっとも確実な方法は、文書やメモ書きによる伝達です。重要な情報なら全員を集めた場での口頭による説明も必要になります。

簡単と思われる事柄でも、複数の人から同じ質問を受けるようなら文書での伝達が必要だということになります。

One point　Lesson　：　情報共有は文書にて

Case＃39　メンバーからの質問に全部答えを与えてしまう

開発対象システムの仕様等に関して質問を受けた時に、相手に調べる時間を与えたり調べ方を教えたりしないで、自分が調べてあげてしまっている結果、メンバー自身で調査する方法を継承できていません。目先の時間効率を考えた結果、自分が調べて教えてあげたほうが早いので、そのようにしてしまいますが、それでは一向にメンバーが成長できない結果を招いています。

【アドバイス】 緊急度・優先度を判断して対応すること

部下の質問に全て答えるか否かは、まずプロジェクト全体としてのその緊急度・優先度を判断してからにすべきでしょう。単に自分が調べた方が早いから自分でやってしまうことばかりでは、部下は一向に成長しないばかりではなく自分自身もいつまでたっても忙しいばかりで自分自身の成長にも悪影響が出ます。十分な時間的余裕があればヒントだけを与えて部下に自分で調査させれば良いし、多少の余裕があるならば部下と一緒に途中まで調査してあげれば良いし、緊急ならば自分が代わりに調査しても良いでしょう。

One point　Lesson　：　答え過ぎは部下の成長を止める

Case＃40　言葉だけではなかなか伝わらない

自分の言っていることが相手にうまく伝わらないことがあります。何回も同じ事を言わないと伝わらないことがあり、例えば問題発見時、言葉で済むようなことでもうまく伝わらず、実際の現象を見せないといけないことがあります。何故、伝えたいことを、一方的に「伝える」ではなく、「伝わる」ように伝えられないのでしょうか。

【アドバイス】 話し言葉だけでは伝わりにくい

相手にうまく伝えるためのポイントは次の二つになるでしょう。

- 文書に語らせる、すなわち客観的な資料や数値データを示しながら説明をする。
- 相手の能力レベルに合わせた話し方をする。

口頭でのコミュニケーションは感情的ないしは情緒的な表現に傾きがちで、あいまいさが混入してしまうものです。これらのあいまいさを排除するためには、資料を使って数値やデータを示しながら話をすることが一番です。また話す相手の知性や技術的能力のレベルに合った説明を行わなければ話は伝わりません。話の要所ごとに相手の理解度を確認すれば相手がどの程度理解しているのかを判断することができます。理解度の確認方法は、相手に理解した内容を復唱させれば良いでしょう。また抽象度の高い内容は、具象的な例え話をすることで相手の理解を深めることができます。話の上手な人ほど例え話がうまいものです。

One point　Lesson　：　資料を用い相手のレベルに合わせた説明を行う

☑ Check List　＃6　：　**指示・依頼に関するチェックリスト**

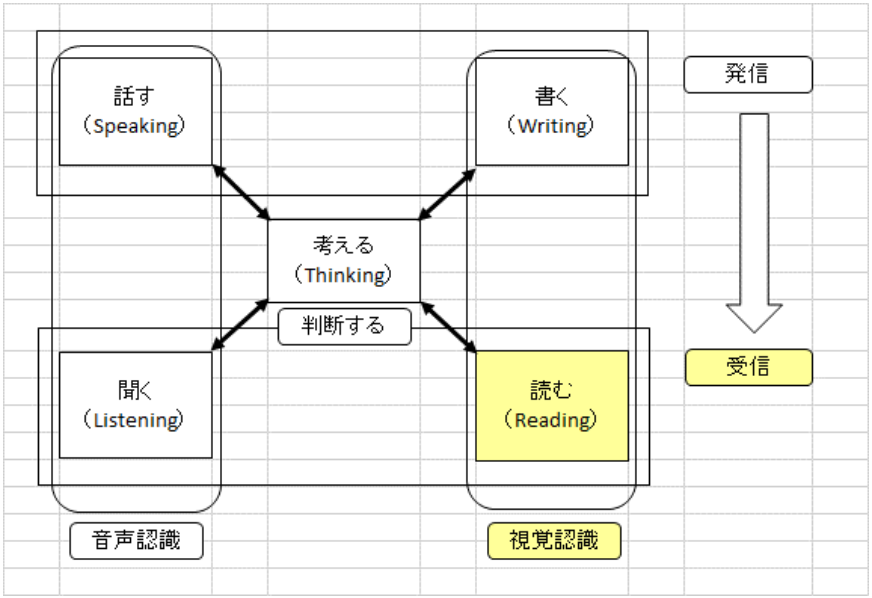
指示を行う、依頼事を頼む場合において次のチェックリストを試してみよう。
〇：できている　△：どちらともいえない　×：できていない

☐ 相手が何を聞きたいのか確認して説明を行っているか。
☐ 相手が最も聞きたいと思われることを最初に伝えているか。
☐ 忙しい時ほど、ていねいなコミュニケーションに努めているか。
☐ 資料に基づいたコミュニケーションを行っているか。
☐ 口頭依存のコミュニケーションになっていないか。
☐ 論理記述的な文書を作成しているか。
☐ 業務運用は手順書などの文書に基づいて行っているか。
☐ 文書による情報共有を行っているか。
☐ メンバー自身に考えさせることを実行しているか。

【ふり返りメモ】

第3章　読まザルの巻

「読む」能力は、「聞く」能力と並んでものごとを理解する能力です。技術や知識を習得する手始めは、ものごとを良く理解するところから始まります。理解する能力とは、ものごとの本質を把握する能力とも言えます。
人間の成長過程においては、「読む」訓練を積んだ後に「書く」ことを覚えます。良く「読む」能力がなければ、良く「書く」こともできないでしょう。
　本章では、これらの「読む能力」を使用する「読む」コミュニケーションについて、事例に沿って考えていきたいと思います。



Case＃41　仕様の誤解

A社システムの画面遷移の制御においてB社システムと類似の機能を流用したが、A社仕様は単一メニュー画面構成で、B社は複数メニュー画面構成であるという違いを見落としたため総合テストで不具合だと指摘された。 この問題の原因は、仕様や決定事項などが、どのような理由で、何の目的でそのように決まったのか理解しようとしていなかったことや、自分の思い込みに気づかなかったために仕様書から正しい仕様を読み取れなかったことにありました。

【アドバイス】 情緒的な判断によらず、まずは仕様書に基づくこと

仕様書にちゃんと記述されていたにも関わらず間違えてしまったと言うことのようですが、これは仕様の誤解というレベルではなく、仕様書を無視したということでしょう。自分の考えを中心におくのではなく、まずは仕様書に書いてあることを”正”とする姿勢で仕様書を見る必要があります。そしてもしその記述に矛盾があれば質問を出し納得できる答えを入手することが正しい仕事の手順です。

One point　Lesson　：　記憶に頼らず記録に頼ること

Case＃42　要求内容を勝手に深読みして仕様を膨らませてしまった

顧客に要求内容の確認をせずに、要求内容を深読みし過ぎた結果、余計な仕様を追加してしまいました。結果、開発ボリュームがかさみ余計なコストを発生させてしまいました。

【アドバイス】 仕様の表現はロジカルに

”深読み”とは、読みきれない不明点や疑問点を勝手に想定したということでしょう。もしそうなら、その時点でやるべきことは、その不明点や疑問点を即座に顧客に確認することです。要求仕様の内容に”深い”も”浅い”も存在させてはいけません。例えば、要求仕様に「A＋B＝C」と記述されていたならば、それは「A＋B＝C」以上でも以下でもありません。行間を読まなければいけないような要求仕様書や設計書は欠陥品なのです。要求仕様書や設計書などの開発にかかわる全てのドキュメントから推測や推定をなくし、可能な限り論理記号による表現を多用し、「非常に、少し」などのあいまいな日本語表現は使用厳禁です。ソフトウェア開発はロジックを作成する仕事であり、仕様を表現する方法も当然ロジカルな論理記号や図表による表現が効果的です。

One point　Lesson　：　全てのドキュメントから推測や推定をなくすこと

Case＃43　仕様書に記載されていた仕様が総合テストまで見落とされていた

既存案件の改造でしたが、既存案件と今回の機能仕様の比較をしていなかったために、見落としが発生してしまいました。既存案件との違いを全て洗い出す必要がありましたが、急遽担当者が変わり、新しい担当者が製造以降の作業を実施した為、仕様書を詳しく熟読する時間がなかったようです。

【アドバイス】ドキュメントによる仕事の引継ぎを

この問題は、担当者間のコミュニケーション・ミスによる問題です。最初の担当者が仕様の精査結果を設計書に残しておき、その資料に基づき交代要員に引継ぎを行っていれば、仕様が漏れるという大きな問題にはつながらないでしょう。”前と同じ”だとか”〇〇と言ったように”とかの口頭会話だけの開発スタイルからドキュメント・ベースの開発スタイルに切り替える必要があります。

One point　Lesson　：　口頭会話だけでは必ずモレが出る

Case＃44　仕様の矛盾に対する対応

仕様書通りの動作はするが、システムとしての統一性を考えると障害とせざるを得ない場面が発生することがあります。要件担当者は、顧客と同意が取れているので仕様通りと言うが、テスト担当者としては、それで良いのかどうか迷います。

【アドバイス】システム全体としての視点で判断すること

仕様の矛盾を発見した場合に何を”正”にするのかは難しい問題ですが、結局はどちらの仕様にした方が真に顧客要求に沿った結果が得られるのかが判断基準になります。また当該機能だけを見て判断せず他の機能への影響も考慮したシステム全体としての判断が必要となります。担当者同士で判断がつかない場合は顧客に直接判断を求める必要があります。

顧客の判断ではバグにしか見えないものを開発側が“仕様通り”と言い張って顧客の逆鱗に触れた場面を何度も目にしてきました。

One point　Lesson　：　仕様矛盾は顧客に判断を仰ぐこと

Case＃45　データ長が固定長ではなく可変長だった

あるデータファイルのデータ長を固定長でチェックしていましたが、実際は可変長で定義されており、チェック処理においてデータ長エラーとなってしまいました。何通りかのテストでは全て同一データ長だったため、固定長だと誤認してしまいました。他の人の意見や考え等を聞き、自分とは違う考え方を知れば、ものごとの本質をつかみやすくなると思いました。

【アドバイス】 自分の目で原典を確認すること

テストでは同一データ長だったから固定長だと判断することは、みんなが道路を80キロで走っていたからこの道路の速度制限は80キロだと思い込むようなものです。

この問題の本質は、原典による事実の確認を怠ったことにあります。自分の思い込みを避けるために、他の人の意見を聞くことや視点を変えてみることも有効ですが、まず第一番目になすべきことは自分自身で原典にあたって事実確認を行うことです。そのデータファイルが可変長であるか固定長であるかは、仕様書の定義を見れば一目瞭然です。またソースコードにおけるデータの定義を見れば分かることです。他人の意見を聞き、別の視点で考えてみるまでもなく、自分自身の目で原典を見れば分かることです。ものごとの本質を見抜くためには、ものごとを自分の目で観察し自分の頭で判断するという自律性のある姿勢や行動を行うことが必要です。

One point　Lesson　：　現物を自分の目で直接確認すること

Case＃46　テスト目的の不理解による無駄な作業

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ました。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であり、実績データを計算するというテストではありませんでした。テストデータ作成は無駄な作業となってしまいました。原因は、自分の経験不足や思い込みにあると思います。

【アドバイス】 思い込みの排除にはドキュメント・ベースの仕事を

この問題の真因は経験不足とか思い込みなどではなく、テストの目的が理解されていなかったことにあります。テスト仕様書には何と書いてあったのでしょうか。あるいはテストの指示者は何と指示したのでしょうか。それとも両方ともに存在しなかったのでしょうか。テストに限らず、あらゆる業務を開始する前には、その目的と方法を明確に理解しておく必要があります。何をすべきかあいまいなものを、自分の想像や勝手な思い込みでやったとしても、まともな成果が得られるはずもありません。テスト仕様書やチェックリストにテストの目的および具体的なテスト項目が記述されていれば、その通りのテストをやれば済むことです。今後は、想像や思い込みではなく、ちゃんとしたドキュメント・ベースの仕事を実行する必要があります。

One point　Lesson　：　この仕事の目的および達成方法は何か

☑ Check List　#7　：「読む」コミュニケーションのチェックリスト

読むことに関する次のチェックリストを試してみよう

〇：できている　△：どちらともいえない　×：できていない

☐ 仕様書を確認せず勝手な自己判断をする場合がないか。
☐ 仕様書の不明点や疑問点を要求者に確認せず勝手な想定をする場合がないか。
☐ 次工程の担当者が読んで分かるドキュメントを作成しているか。
☐ データ処理に関する制限値およびシステム要件目標値を仕様書で確認しているか。
☐ 仕様および機能の目的・意味・背景を仕様書で確認しているか。

【ふり返りメモ】

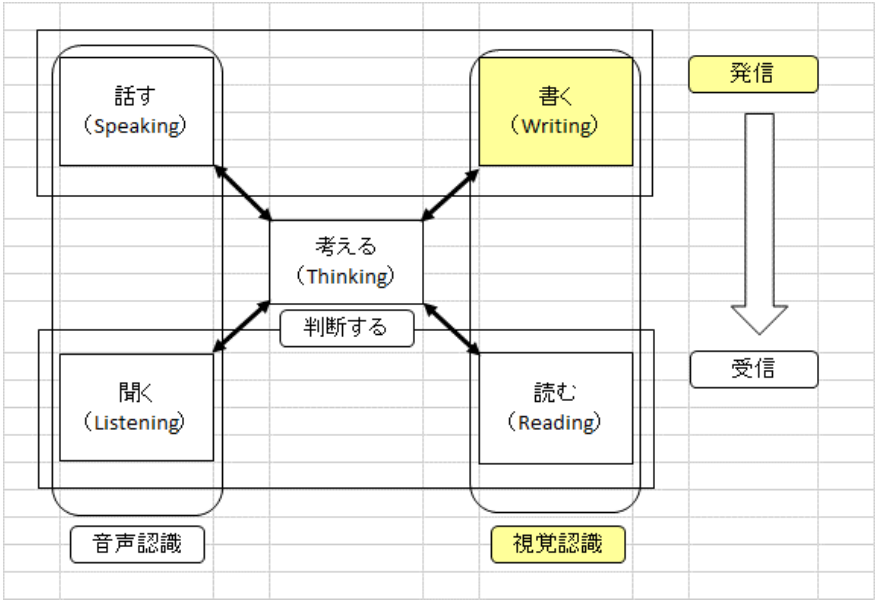
第4章　書かザルの巻

「表現する能力」の一つである「話す能力」については第2章で述べましたが、本章ではもう一つの表現する能力である「書く能力」について話をすすめたいと思います。文字・数字および図表などによって、ある事象を説明することは四つのコミュニケーション能力の中では最も難しいレベルにあります。「話す」および「聞く」は、生まれながらに備わった本能レベルの能力ですが、「書く」および「読む」は専門的な教育を受けなければ獲得できない能力です。

日本語における「書き言葉」は、漢字という表意文字やカタカナ・ひらがなという表音文字の二種類で構成されており、漢字は図表などと同じく視覚的な映像として脳内で処理されますが、カタカナ・ひらがなは視覚で読み込まれたあとに音声として脳内で変換処理が行われているようです。つまり「読む」「書く」においては、頭の中では音声処理と画像処理の二つのエンジンを同時に働かせることになり、脳の疲労度は非常に高いものとなります。そのため人間がこの能力を獲得するためには数十万年もの長い時間が必要でした。

ソフトウェア開発という仕事は、自然言語を書き、それをさらに機械言語に書き直す翻訳業務的な仕事だと言えます。開発は、最初に自然言語で書かれた要求仕様書によってキックオフされ、要求仕様書はさらに設計の言葉に変換され設計書が作られます、さらにその設計書は機械言語であるソースコードに書き直されコンピュータを動作させることになります。

コンピュータという仕掛けは、厳格な論理によって構成されなければ動作できないという性格を持っているために、これらの一連の自然言語文書および機械言語コードも同様に厳格な論理性が要求されます。しかしながら低コスト、短納期を迫られた場合に、四つのコミュニケーションの中で最初に犠牲にされるものは、もっとも難しくて手間がかかる「書く」コミュニケーションでしょう。「書くコミュニケーション」の劣化、すなわち「開発ドキュメント」の劣化が招く障害および対応策について事例にそって考えてみたいと思います。



1.　まともな業務文書が書けません

Case＃47　ちゃんとした業務文書が書けない

　文書やメールが長文になりやすく、相手に伝わらず情報伝達漏れが発生しやすくなります。よく理解してもらうために細かく丁寧に文章を記述している積りですが、どのようにしたら良い文書を書けるようになるでしょうか。

【アドバイス】 業務文書の三つのポイント

　業務文書作成の基本的なポイントは下記の通りです。

1. 受け取る側の利益・心情を優先すること

①まず結論の記述から始めること。
　忙しい相手はまず結論を知りたいはずですから、相手が持っていると思われる要望や疑問にまず答えることが大切です。
②自分の言いたいことを優先しないこと。
　特に不始末の詫び状などでは言い訳から記述を始めるのは禁物です。

2. 正確性を確保すること

①文章の主題・件名と本文の内容を合致させること。
　主題や件名が内容と合致していないと、受け取った相手は混乱します。メール返信の「Re：〇〇」は要注意です。内容や話題が変わったら件名も新たに書き直す必要があります。
②文章において、「誰が、何を、いつまでに、どのように、どれくらい」等を明確に記述すること。
③数値やデータによって質や量を表現することで、文書の客観性を保つこと。

3. メリハリの利いた文章にすること

①結論に始まり、続いてその経緯について簡略にまとめること。
　どんなに長い内容でも本文は1～2ページにまとめ、詳細内容は添付資料とする。
②同じ内容を、形を変えてダラダラと何度も繰り返さないこと。
③重要な事とそうでないことを区別すること。

One point　Lesson　：　結論を先に。言い訳は不要。

Case＃48　新人の文章作成能力が低くて困っている

新人の教育において、OJTの一環として日報をつけていますが、文章作成のレベルの低さに困っています。日報では、仕事のポイントがわかっていないような内容ですが、プログラムはしっかりと組めています。文章作成の能力を向上させる、または指導するにあたってのポイントはありますでしょうか？

【アドバイス】 文書作成のポイント

文書作成のポイントは以下の通りです。

☐ 書く前に、相手に伝えたいことを重要なものの順にメモに書き出しておく。
☐ 文書は重要なものの順に書く。
☐ 箇条書きは簡潔な表現に適している。
☐ 一つの文章はできるだけ短くする。複雑な内容は複数の文章に分けて書く。
☐ 報告書は、最初に結論を書き、次に説明を書く。詳細説明は添付資料とする。
☐ 報告書は、基本的に最初の１ページで全貌が分かるように書く。
☐ ”非常に”とか”おおよそ”などのあいまい表現はしない。
☐ 無用な誤解を避けるために数字・数値で論理的に記述する。
☐ 文章だけで説明が難しいものは、マトリクス・図・表などを使用する。
☐ 技術用語はプロジェクト内や社内で統一用語を使用する。

以上の事に気をつけるだけでも、かなり分かりやすい文章を書くことができます。

文書作成能力を向上させるためには、まずは日報の書き方から始めるのが良いでしょう。日報を誰が読んでも分かるレベルに持ち上げるためには、リーダーが文章を添削して返すくらいの努力が必要です。日報がまともに書けないならば、報告書・議事録・仕様書・設計書などもまともには書けないでしょう。いくらプログラムを組むことができたとしても、他人に仕事の情報を伝える技術文書がまともに書けないようではいつまでたってもドキュメント・ベースの開発は実現できないでしょう。

文書作成能力を向上させるためには基本的には優れた本をたくさん読むこと及び数多くの文章を書く経験を積む必要があります。

One point　Lesson　：　数値・図表による単純明快な記述をすること

☑ Check List　#８　：　**文書作成のチェックリスト**

文書作成の前に次のチェックリストを試してみよう

〇:できている　△:どちらともいえない　×:できていない

【準備】

☐　書く前に、相手に伝えたいことを重要なもの順にメモに書き出しておくこと。

【受け取る側の利益・心情を優先】

☐　最初に結論の記述から始めること。相手の要望や疑問にまず答えること。

☐　自分の言いたいことを優先しないこと。特に不始末の詫び状などでは言い訳から記述を始めるのは禁物。

【分かりやすさ】

☐　報告書は基本的に最初の１ページで全貌が分かるように書くこと。

☐　文章の主題・件名と本文の内容を合致させること。

☐　結論に始まり、続いてその経緯について簡略にまとめること。
　どんなに長い内容でも本文は１～２ページにまとめ、詳細内容は添付資料とすること。

☐　文章は重要なものの順に書くこと。

☐　一つの文章はできるだけ短くすること。複雑な内容は複数の文章に区切って書くこと。

☐　箇条書き等によって簡潔な表現に努めること。

☐　同じ内容を、形を変えて何度も繰り返さないこと。

☐　重要な事とそうでないことを峻別すること。

【正確性の確保】

☐　「誰が、何を、いつまでに、どのように、どれくらい」等が明確に記述されているか。

☐　数値やデータによって質や量を表現することで、文書の客観性が保たれているか。

☐　あいまい表現はしない。例えば、”非常に”とか”おおよそ”などの表現は厳禁。

☐　文章だけで説明が難しいものは、マトリクス・図・表などを使用すること。

☐　技術用語は、顧客・プロジェクト・社内で統一用語を使用すること。

【ふり返りメモ】

2.　開発手順書がありません

Case＃49　開発手順・方法について継承できていない

開発の手順・方法に関する知識を文書として残せていない為、現状、口頭やメールで通知しています。文書化については、日々の作業に追われて実施できていません。そのために新しいメンバーが入ってきた際に、再度説明する必要があり、また、自分が他のプロジェクト等に移った場合に引継ぎがスムーズに行われない結果になってしまっていると思います。

【アドバイス】ドキュメントによる開発および知識の継承を

最も効率的かつ正確な仕事のやり方はドキュメント・ベースの仕事をするということです。多数のメンバーの頭の中に分散されている知識を都度引っ張り出して仕事をする方法と、明確な定義が過不足なく記述されているドキュメントを根拠に仕事をするのとでは、どちらが正確で早くそして失敗が少ないかは明白です。不確かで感情に流されやすい人間の記憶に頼るような仕事のやり方から早く脱却する必要があります。口頭やメールでの情報伝達はリアルタイム性を確保する場合には必要ですが、これらの情報はすぐに雲散霧消してしまい長時間・長期間の業務活動の根拠には有効ではありません。根拠に乏しい仕事のやり方では、人によって出来栄えの幅が大きくぶれてしまい、また理解不足や誤解による失敗が多く発生するために後戻り対応で多くの時間とコストを失ってしまいます。 忙しいから文書化する時間がないのではなく、文書化してから仕事に着手しないから時間が足りなくなるのです。二、三人の小規模開発では文書なしの口頭ベースでのやり方で何とかなったのでしょうが、それ以上の規模の開発では全く不可能なことです。

One point　Lesson　：　忙しいから文書化できないのではなく、文書化しないから忙しくなる

Case＃50　手順書の作成や更新後れで知識の不一致や説明負荷が増える

装置の操作手順書等がタイムリーに提供できていないときがあります。作成タイミングが、リリースが一段落した後のタイミングになっており、必要なときに知識を共有できていないと感じています。また開発が暫定対応の段階にあったりすると出力が明確にできないために、資料の作成はどうしても後回しとなってしまいます。また、一度作成した手順書やウハウの更新タイミングを逃すことも多いです。 このような結果、メンバー内で装置等の環境への知識が合っていないときがあり、何回も同じ事を説明することがあります。

【アドバイス】ドキュメントは開発実務の最中に小まめに更新しておくこと

多くの人が使用する操作手順書は仕事の効率や品質向上に大きな影響を及ぼします。未定内容があったとしても、プログラム製造に着手する前に仮版でも良いので内容を更新しておくべきです。正式版のドキュメントは開発実務をやりながら補強・修正していくことが最も現実的です。これらのドキュメントに共通する一番大きな問題点は、その内容の更新にあります。多くの人は、仕事が一段落してから更新しようと考えるため、ほとんど更新されずに終わることが多いのです。多くの仕事はいつも忙しい状態にあり、一つの仕事が終わる前にもう次の仕事が入っており一段落する暇など実質的にはないために、後で作成しようと思っていても実行できないのが現実です。 ドキュメント更新のタイミングは、更新すべき内容が見つかったその時点が最適のタイミングです。開発作業は必ずドキュメントを見ながら行い、訂正箇所に気がつく度に、まずはドキュメントに手書きで修正を記入しておくことです。ドキュメントの電子ファイルの修正はその後になっても問題ありません。一日の最後の 30 分を、その日の振り返りや次の日の予定の設定およびドキュメントの更新に割り当てる習慣を身につけることが最も良い解決法です。

One point　Lesson　：　後でやろうと思ったことはほとんど実行されない

☑ Check List　＃９　：　**開発ドキュメントのチェックリスト**

開発ドキュメントに関して次のチェックリストを試してみよう
〇:できている　△:どちらともいえない　×:できていない

☐　開発手順書に基づいた開発を行っているか。
☐　開発実行中に開発ドキュメントをこまめに更新しているか。
☐　忙しいことを言い訳にしてドキュメントの更新を怠ってはいないか。
☐　開発ドキュメントはプログラムコードよりも重要度・緊急度が高いと思っているか。

【ふり返りメモ】

3.　何が書いてあるのか分からない仕様書

Case＃51　ベンダー側においてドキュメントの更新が行われていない

ベンダー側において、要求仕様書、ターミナル仕様書のメンテナンスが行われず、評価業務に支障をきたしています。以前指摘した誤記部分が次バージョンの開発時においても修正されていません。修正依頼は都度行っていますが一向に改善されません。 仕方なくこちらとしては、質問票にて確認が取れたものについては、内部資料としてハードコピーの要求仕様書・ターミナル仕様書へ直接書き込む形で随時更新しています。 ベンダー側との認識ずれを防ぐためには、更新した手書き修正版のコピーを定期的にベンダー側に送付する必要があります。

【アドバイス】ドキュメントの更新作業も仕事として請負うこと

ベンダー側におけるドキュメントの更新がどうしても進まないのなら、ドキュメントの更新作業も仕事として請負うことをベンダー側に提案する必要があります。本来、これらのドキュメントの更新は、ベンダー側における重要な職務の一環であり、ドキュメントの誤記や抜けを放置することは、自ら不具合の再生産を行っているのに等しい怠慢な姿勢です。改めるべきでしょう。

One point　Lesson　：　相手ができないなら、こちら側でビジネスとして請け負うこと

Case＃52　開発文書に独自性を発揮しようとするとできが悪くなる

開発文書の作成において、開発目的の記述・伝わりやすい表現などが必要ですが、表現や実現方法に自分独自の考えを盛り込みたいと思うことを優先する為、本来あるべき内容からずれてしまうことがあります。

【アドバイス】 開発文書は論理的かつシンプルに記述すること

独自性の発揮の具体的な意味合いがよく分かりませんが、独自の表現方法や実現方法を優先した場合ずれることがあるのなら、基本的なやり方をベースにして、その中で自分なりの表現力の豊かさを発揮すればズレやモレはなくなるのではないでしょうか。開発文書の役割は独自性を追求することではなく、厳格な論理性を基に誰が読んでも同じ意味を過不足なく容易に伝えられるようにすることです。仕事に限らずスポーツや趣味の世界においても、いきなり我流でやっても中々うまくやれないものです。やはり基本のフォームをしっかり身に着けた上で自分の特徴を出した方が良いでしょう。

One point　Lesson　：　我流によらず基本のフォームで

Case＃53　誤解を招く仕様書の日本語記述

AとB以外の場合は～を行う…という仕様で、A＆NOT（B）と、NOT（A＆B）を誤認してしまいました。過去の経験から、この仕様はこうすべきだ、という思い込みが原因でした。正しいと思っている為、この例でどちらが正しいかを顧客に確認しませんでした。今後は、この仕様で正しいのか？という疑問を常に持ち、視点を変えてみる等の訓練をしたいと思います。

【アドバイス】 論理的記述による仕様書を

相談者において既に解決法が示されています。要求者は、「AとB以外の場合は」という表記をやめ、「A＆NOT（B）」または「NOT（A＆B）」という表現方法を使うべきでしょう。このような論理記述法を使えば、間違える余地はゼロになるでしょう。日本語表記は、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、もっと徹底的に数学（論理式・ロジック）の言葉で記述した方が良いでしょう。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できるでしょう。仕様書の記述においては、論理式や図・表・フローチャートなどを多用し、日本語記述はそれらの補助的説明文ないしは仕様の意味・背景などの説明として使用するのが適切です。試しに現在開発中の仕様書の一部分を徹底的に数学（ロジック）の言葉や図・表・フローチャートで記述して見てはいかがでしょうか。あいまいな箇所が何%減るのか非常に興味深いものがあります。この活動はりっぱな改善活動の候補になり得るものと思いますので是非実行してください。

One point　Lesson　：　情緒的日本語文章は論理的な仕様記述には適していない

Case＃54　仕様書における不要と思われる記述

既定のドキュメントのフォーマットにおいて今回使用しない項目について記述が必要か否か迷います。関係ない項目への記入は無駄な手間になります。

【アドバイス】 要／不要は顧客に確認を

仕様書などの開発ドキュメントの多くは、その資料の一定の品質を保つためにフォーマット化されている場合が多いと思います。しかしながら、それぞれの開発の内容や特徴によっては、記述する必要のない項目があるのは事実です。自分の担当した開発において、そのような項目に気づいた場合は、何らかの存在理由が隠れているのかも知れませんので、上司や要求元のお客様に確認した上で記述するかどうかを決めれば良いと思います。現時点で必要性がなくても仕様条件や背景によっては必要な場合もあり、不用意に削除すると失敗につながる危険性があります。顧客に確認して、今回は不要だと分かったらフォーマットから削除するのではなく、フォーマットの欄に「N／A（not　affectの略）」、「今回は影響せず」などの文言を残しておくことをお勧めします。

One point　Lesson　：　既定のフォーマットを安易に崩さないこと

Case＃55　複数の仕様書間の不整合に起因した不具合

　ベンダー側提供の二つのシステム仕様書が存在し、仕様書間で記述内容の不一致が原因で不具合を発生させてしまいました。

【アドバイス】 上位システム側の仕様書記述を統一すること

　一つの仕様に関することは一箇所にまとめて書くのが原則です。複数の仕様書に同一機能に関して異なる記述を行っている箇所が多数あるとすれば下位システムの開発側にとって何が正なのか分からず、さらに同様の問題が他にどれくらい存在するのかも分からず、信頼性のある開発は不可能となってしまいます。二つの上位仕様書の矛盾点を発見しながら下位システムの開発を続行するには余りにもリスクが高いと思われます。この問題に対して下位システムの開発担当会社がとりうる対策は次のようでしょう。

●ベンダー側にて下位システム開発が準拠すべき仕様書を明示し、この仕様書における誤記部分を修正すること。

●複数の上位システム仕様書の下位システムに関する矛盾した記述の調査および修正業務そのものを発注していただくこと。

One point　Lesson　：　一つの仕様を複数の仕様書にまたがって書かないこと

Case＃56　システム仕様書の不備が多く指摘し切れない

　システム設計書の不備が多く確認し切れません。評価チームにおいては、変更部の誤記や不備は指摘できますが、その他の部分の誤記や不備までは指摘ができません。誤記や不備に関しては、テスト実施時に相違があることが確認できた場合は障害票として起草しています。障害票に起草していることで仕様書の修正が確実になされることが確認できるようにしています。

【アドバイス】 仕様書の誤記・不備は重大な不具合

　仕様書の不備や誤記のメンテナンスは基本的にベンダー側ないしは設計チームの義務です。ドキュメントに基づいた開発を実行するという文化のない開発組織では設計書のメンテナンスがおろそかにされ、その結果、誤記・不備・記述なしなどの仕様書が放置されています。開発の都度、設計工程において適切な設計書を作成しておき、それに従ったコード製造を行っていれば、常に最新版の仕様書は最新版のソースコードと一致した状態が保てます。一旦メンテナンスを放棄した設計書は開発の根拠を表す資料として使用できなくなり、その後は設計書なしの無定見な開発が行われるという道をたどることになります。

　評価チームとしては、仕様書の誤記・不備についても重大な不具合として扱い、開発チームに障害票をどんどん発行することで、開発チームにおける仕様書のメンテナンスを促進させていく必要がありますが、何はさておきベンダー側ないしは設計チームにおいてはドキュメント更新に対する緊急性・重要性を再認識すべきでしょう。

One point　Lesson　：　適切な仕様書・設計書に基づいた開発実行の重要性を認識すること

☑ Check List ＃10 ： **要求仕様書のチェックリスト**

仕様書に関して次のチェックリストを試してみよう
〇:できている △:どちらともいえない ×:できていない

＜要求仕様書のチェックリスト＞

1. 妥当であること

□ 顧客やユーザーのニーズと一致していること。
□ 上位のシステム要求仕様書などの関連する他のドキュメントとの矛盾がないこと。
□ 未確定項目がある場合は、どのように合意するか、依頼者と合意形成方法を決めておく。

---

2. あいまいでないこと

□ 要求仕様書に記述されている要求が、ただ一通りに解釈できること。
□ 要求仕様書の“良し悪し”を判断する手段や基準をもつこと。
□ 「範囲」を読み取れるように要求を表現すること。
□ 仕様は「仕様である」ことを明示し、説明は「説明である」ことを明示して記述すること。
□ 要求仕様書では、記述内容が“特定”できる表現になっているものを“仕様”とすること。
□ 要求仕様書の構成や内容は、後工程の読者に分かるように書くこと。
□ 「等」や「etc」の文言は使用しないこと。使用する場合は、〇月〇日までに決めるとコメントをつけること。

---

3. 完全であること

□ 顧客やユーザーの、情報システムに対するニーズが漏れなく要求仕様書に記述されており、かつ図表の参照や用語の定義などの、要求仕様書の形式が整っていること。
□ 「境界」は早い段階で決めること。
□ 「要求」のモレを防ぐために、カテゴリの分類や要求の分割・階層化に漏れがない、隙間がないことを確認できるようにすること。
□ 要求仕様書には、「操作性」「保守性」「交換性」などの「品質要求」を記載すること。
□ 階層化の基準として、以下を（状況によっては組み合わせて）使い、「隙間」なく分割すること。
時系列分割（時間軸分割）／構成分割／状態分割／共通分割
□ モレなく書くこと。
□ 要求仕様の番号をテストケースの番号とひもづけし、テストケースにモレがないことを確認すること。
□ 仕様をグループに分け、さらに集合を小さくし、混じり気のない仕様のグループを作る。
□ ＜グループ名＞に要求の性質を持たせるためには、範囲をあらわしていることを意識してグループ名を選ぶこと。
□ 「・・・・は、・・・・しない」という「否定表現」を避け、thenとelseの両方を明らかにすること。

---

4. 矛盾がないこと

□ 要求仕様書内部で矛盾や衝突がないこと。
□ ほかの機能の仕様と衝突していることに気づくためにも、仕様は早期に展開すること。
□ 早い段階で全体の仕様化を行うこと。

---

5. 重要度と安定度のランク付けがされていること

□ 各要求について、重要度と安定度を示す指標を明確につけておくこと。
□ 確認中の仕様をそのまま記述し、変わる可能性があることを明記すること。

---

6. 検証可能であること

□ 開発されたソフトウェアが、要求仕様書に記述された要求を満たしているかどうかを確認可能であること。
□ 検査部門の人に、「検査可能」という側面から要求仕様書のレビューを実施してもらう。
□ 品質要求（「操作性」「保守性」「交換性」など）はテストでも確認すること。

---

7. 変更が容易であること

□ 要求仕様書に対する変更が、容易に、完全に、一貫して行えるようになっていること。
 a）目次や索引、明確な相互参照が整備され、使いやすい構造になっていること。
 b）冗長でない、つまり、同じ要求が要求仕様書内で複数個所に記述されていないこと。
 c）他の要求と混ざらず、各要求を独立・分離して表現している。つまり、要求が互いに依存していないこと。
□ 重複なく書くこと。
□ 仕様書全体を「均一」に記述することにこだわらないこと。関係者間で共有できている認定仕様まで、詳細に記載しなくてもよい。
□ 仕様番号の確定作業は、仕様化の最初の段階では行わないこと。グループ分け確定後に行うこと。
□ 似た記述が続く場合に、何が違うかをすぐに読み取れるようにすること。

---

8. 追跡可能であること

□ 要求仕様書に記述された個々の要求に関し、その起源が明確であり、開発が進行するに伴って作成された文書等との対応付けがとれること。
 a）後方追跡可能性
 b）前方追跡可能性
□ 設計や実装の工程で明らかになった「仕様」は、要求仕様書に書き戻すこと。
□ 「要求」と「理由」をセットで表現すること。
□ 要求仕様には固有の記番号を付けること。

（参考資料： IEEE830品質特性、USDM）
これらの要件は要求仕様書のみならず設計書や他の開発ドキュメントにも共通するものです。

【ふり返りメモ】

4.　使えない設計書

Case#57　設計書の誤記が減らない

設計書の誤記が一向に減りません。自分の場合もそうですが、設計書を後で見ても誤記が多いと思います。各人のチェックが甘く、同じことを何度も記述することが多く、最初に書いた内容が間違えているとそれを引きずって誤記を増殖している結果になっています。

【アドバイス】 設計書を先に作成すること、誤記に気づいたらその場で修正すること

設計書の誤記をどのように発見しているのか実に興味深いものがあります。想像ですが、プログラムの作成後に設計書を書いているのではありませんか。そして改めて設計書を見直した場合に自分が作ったプログラムとは違った記述を発見しているのではありませんか。自分が作ったプログラムが”主”であり、設計書は”従”という現実を物語っています。もしそうではなく、先に設計書を作成して、後にプログラムを作成しているのならば、そのように誤記が多い設計書に基づいて作成したプログラムはバグだらけだと言うことになります。

下記の対策を実行する必要があります。

1. プログラム作成の前に設計書を完成させておくこと

なぜ設計書の誤記が減らないかの理由は、みなさんが、プログラム自体が最も重要で設計書はその次だという認識を持っているからでしょう。その認識は全く逆だと思いませんか。要求仕様書の間違い、設計書の誤記もバグであり、より上流の間違いがより大きな被害をもたらすものだと心底分かっていない証拠なのでしょう。とりあえずコードだけ作成しておき、設計書は後で暇な特に作成すれば良いという程度の認識では、暇な時はいつまでたっても来ないため、設計書の誤記はいつまでたっても修正されずに減ることもないでしょう。そのまま修正されない設計書は次の開発、その次の開発においても被害をまき散らし多くのコストと時間の浪費と品質低下の原因となり続けることでしょう。プログラム作成の前に設計書を作成してください。そしてプログラム作成時に設計書の誤記に気づいたら、その場で直ちに設計書を修正してください。たった数時間程度の誤記修正の手間と時間を惜しむことが、どれほど多くの無駄な手間と時間とコストの浪費になるかということを心底理解する必要があります。

2. 論理的かつ過不足のない設計書を書くこと

設計書の記述の要点は要求仕様書における要点と基本的に同じだと言えます。これらの技術文書の書き方の骨子は、日本語での記述は仕様や機能の概要・意味・背景などの説明などにとどめ、設計書の詳細な記述は、論理記号や図・表・フローチャートなどを使用し、あいまいな日本語記述をできるだけ避けることです。また同じ事を何度も何箇所にもわたって書くことは、一つの誤記が複数の誤記を生じることになりやめるべきです。設計書の品質の均一性を保つためにはフォーマット化も必要になります。

One point　Lesson　：　物を作る前に設計書を作っておくこと

Case＃58　詳細設計書がプアになる理由

詳細設計書の問題として次のようなものがあります。

- 詳細設計の時間が不足している。
- 詳細設計書に誤りや記述がないものが多く、新たなプログラム開発の前にソースコードを確認してドキュメントの修正が必要な場合が多い。
- 見積り条件に、既存設計書のメンテナンス費用が含まれていない。
- 客先において詳細設計書のメンテナンス費用が認められない。

上記の理由などによって詳細設計書はメンテナンスされなくなってしまいます。

【アドバイス】 こまめにメンテナンスを実行すること

事例内容から見えることは、ベンダー側および下請け会社双方におけるドキュメント軽視の姿勢です。下請け側から見れば、そもそもベンダー側できちんとメンテナンスしておくべき詳細設計書が不十分なのだから費用を出すべきだと考え、ベンダー側から見れば、それは下請け側の過去の開発費に含まれているはずだということになります。多くの開発組織においては動くプログラムが最も重要であり、設計書等は二の次であるというドキュメント軽視の姿勢が続いています。その結果、開発ドキュメントは使用に耐えない紙くず同然の状態になってしまっていると言っても過言ではありません。見積りに関しても、ベンダー側・下請け側双方においてはプログラム作成費用のみしか頭になく、設計書のメンテナンス費用など含まれない状況を生み出しているのです。過去の開発の都度、妥当なメンテ費用をこまめに見積りに含めて、こまめに設計書のメンテナンスを実行していれば、現時点のソースと設計書は必ず一対一で整合性を保っているはずです。それを怠った結果が、大幅にメンテナンスをしなければ使いものにならない設計書となってしまったわけです。そういう観点でみれば過去の開発において設計工程を受注していた下請け側にも責任の一端はあるのかも知れません。

はっきり認識する必要があることは、設計書はコードを作成する前に作成されなければならないということです。

現在の状況下において取りうる今後の対策は、仕様変更開発のつど下記をこまめに実行することです。

- 開発の都度、影響範囲内にある仕様書の記述をメンテナンスすること。
- メンテナンスに要する費用は設計費に含めておくこと。

One point　Lesson　：　ドキュメントのメンテナンス費用は必ず確保しておくこと

Case＃59　ドキュメントのメンテナンスがおろそかになっている

ドキュメントのメンテナンスがおろそかになってしまいます。自分で理解できたことで、一旦良しとしてしまい、ドキュメントにまとめる作業や、その後のメンバーへの共有等が疎かになっています。作業が立て込んでいると、優先できない作業でも時間に空きを見つけ進んで実行する必要があると思います。

【アドバイス】 ソース中心主義からドキュメント中心主義へ切り替えること

ドキュメントのメンテナンスは優先的な仕事ではないとお考えのようですが、ドキュメントのメンテナンスは最優先的な仕事だと認識する必要があります。また、皆さんは異口同音に時間がないからドキュメント更新ができないと言い続けています。その結果、信頼性が低いドキュメントで自分および皆さんはどのような被害を被っているかを良く考えてみてください。メンテされていないドキュメントは紙くず同様です。こまめに空いた 5 分、10 分を使って都度メンテすれば合計でも数～十数時間でメンテは済むでしょう。このこまめなメンテの時間を惜しんで、結局ソース解析に十数時間費やすのとどちらが得でしょうか。ドキュメントがメンテされていれば次の開発でも、他のメンバーが担当してもソース解析に費やす膨大な時間が不要になり、不具合の減少にも貢献します。どちらが得でしょうか。目先の小さな労を惜しんで本当の大きな得を捨てるような思考・行動は愚かなことです。ドキュメントの更新は時間が空いたらやるレベルの仕事ではなく、開発の必須業務の一つです。ソース中心主義からドキュメント中心主義へ切り替えなければ、いつまでたってもデスマーチ的な開発から脱出することはできません。

One point　Lesson　：　時間不足は言い訳にならない

Case＃60　適切なドキュメントがなく社員教育などがうまくいかない

社員教育・協力会社教育にあたって使用に耐えうるようなドキュメントがありません。

【アドバイス】 ドキュメント・ベースの仕事の実行を

後輩教育および協力会社の教育については、基本的にドキュメントに基づいて実施することが最も効果的です。教育とはとりもなおさず開発ノウハウの継承ということに他なりませんので、複雑な内容である開発のノウハウをその場しのぎに思いつくまま口頭で説明しても相手には伝えることはできません。開発ノウハウは全てドキュメントに書き表さなければ他人へ説明することはできません。すなわち記録された開発ノウハウとは、過不足のない要求仕様書、基本設計書、詳細仕様書、構成管理仕様書、評価設計書、開発管理表、プロジェクト完了報告書などに他なりません。これらのドキュメントの完成度が高ければ、適時これらの資料の一部を切り出して説明資料とするなど、後輩や協力会社への効果的説明も教育も可能となります。まずは自分の担当する領域におけるドキュメントの完成度を向上させ、それに基づいて後輩や協力会社の教育を行うことから始める必要があります。

One point　Lesson　：　ドキュメントのメンテナンスを怠らないこと

【設計書に関するチェックリスト】

設計書に関して次のチェックリストを試してみよう

〇:できている　△:どちらともいえない　×:できていない

☑ Check List ＃11 ドキュメント・ベース開発のチェックリスト

☐ 基本部の要求仕様書は、見積り時に提示されること。
☐ 詳細要求仕様書は妥当な時期までに凍結されること。
☐ 要求仕様書に基づいて基本設計書が作成されること。
☐ 基本設計書に基づいて詳細設計書が作成されること。
☐ 詳細設計書に基づいてプログラム作成が行われること。
☐ 詳細設計書に基づいて単体テストチェックリストが作成されること。
☐ 単体テストチェックリストに基づいて単体テストが行われること。
☐ 基本設計書に基づいて結合テストチェックリストが作成されること。
☐ 結合テストチェックリストに基づいて結合テストが行われること。
☐ 要求仕様書・実運用シナリオ資料に基づいて総合テストが行われること。

【ふり返りメモ】

☑ Check List ＃12 ドキュメント更新のチェックリスト

☐ 仕様調査時のQ＆A資料の内容は都度、設計書等に反映すること。
☐ 仕様書・設計書等は、テスト等での不具合修正の都度、更新すること。
☐ 仕様書・設計書等は、欠陥が発見された都度、更新すること。
☐ 仕様書・設計書等は、派生開発の都度、更新すること。
☐ ソフトウェア構造設計書は、派生開発の都度、明確になった部分を更新すること。
☐ 仕様変更影響度表は、仕様調査情報および評価テスト情報に基づき更新すること。

【ふり返りメモ】

5.　テストできない評価チェックリスト

Case#61　単体テスト漏れ

単体テスト実施漏れが発生している為に内部結合テスト以降の外部結合テスト、総合テストで障害が発生しています。評価チームとしては、開発チームとのレビューは実施していますが、レビューでは文章の羅列で行なう為、テスト範囲を見極めづらく漏れが出てしまいます。

【アドバイス】ドキュメントの見える化の実行を

単体テスト用チェックリストの精度を上げるためには詳細設計書の精度を上げる必要があります。ドキュメントの精度の要素としては網羅性および理解のしやすさの二点があります。

設計・製造内容を分かりやすく漏れなく表現した設計書および評価チェックリストの見える化は設計・製造の品質レベルを決定づける大切な要素です。開発ドキュメントの見える化に必要な具体的な例をチェックリストで示します。

【ドキュメントの見える化のチェックリスト】

（フォーマットに関して）

□　書類の品質の均一性保持のために、統一的にフォーマット化すること。

□　重複を避け、シンプルな記述にすること。

（記述内容に関して）

□　仕様や機能の目的・背景・意味を記述すること。

□　論理記号、数学的表記、図・表・フローチャート等を使用し、論理的な記述をすること。

□　複雑な事象表現にマトリクス図や共通パターン表などを使用すること。

□　パフォーマンス・レスポンス条件を記述すること。

□　異常系処理を記述すること。排他処理、非同期制御、タイミング、リトライ処理、タイマー値、レジストリ、設定値間矛盾、最小値・最大値制御（メモリー、データ長、伝文長、カウンタ、ポインタ）、メモリーリークなど。

□　誤記、抜けおよび変更は発見・発生と同時に更新し、ドキュメント内容とコードは、常時一致させること。

□　複数の仕様書間、設計書間の矛盾した記述をなくすこと。

□　仕様変更による修正影響度表を作成すること。

One point　Lesson　：　ドキュメントの品質がソフトウェアの品質に直結する

Case＃62　単体テストが進められない

新規開発システムの単体テスト工程以降のテストを請負っていますが、最初の単体テストは設計書ベースでテストを行う想定となっていますが、抜けや誤記の多い設計書ベースで単体テストを行わなければならないためテスト時にどうテストするべきか考える必要があります。また、テストするべき内容が設計書に書かれた内容から読み取れず、テスト抜けが発生しています。

結局、チェックリストもなく、チェックリストとなるような設計書を作ることになっていますが、納期、コストを理由に作れない状況に陥っています。

【アドバイス】 仕様書・設計書なしの状態から評価チェックリストを作成する方法

まともな詳細設計書がなければ、単体テストのチェックリスト作成も不可能で、単体テストも実施できないのは当たり前のことです。ベンダー側が提出すべき詳細設計書を完成できていないのに、チェックリストのような設計書の作成を要求するなどという滅茶苦茶な要求に対して貴社のマネジメントは一体何をしているのでしょうか。

また、そのような状況を仕方ないことだとあきらめて、やみくもなテストをしながらチェックリストを作っていくなどという方法しか頭に浮かばないのは情けない話です。このような方法でまともなテストができるわけもないということは当事者もご承知のはずです。

このような仕事は受注する前にお断りするか、詳細設計書が完成してから受注すべきでしょう。

このような状況に陥った場合に緊急避難的にとれる対策は、実運用に従って旧システムと新システムを平行して動作させることでチェックリストを作成し、単体テストを行うしかないでしょう。手順は次の通りです。

①旧システムと新システムを同時に動作させながら、関連仕様の機能についての入力条件・処理・出力の違いを理解し、ドキュメントに整理すること。これが新システムの基本的なチェックリストになります。

②上記の結果で判明した情報は、もし古い設計書がまだある程度利用可能ならば、その仕様書に新しい情報を書き込んで更新すること。

③単体テスト対象の仕様・機能の入出力条件に関する疑問点・不明点について、即刻、設計・製造者およびベンダーと直接話しをして明確にすること。

とにかく意味のあるチェックリストを事前に用意しなければ単体テストは不可能です。

One point Lesson ： テストの根拠を新システムと稼働実績のある旧システムの差異に求めること

Case＃63　単体テスト仕様書のテスト項目が不足している

単体テスト仕様書のフォーマットに問題があり、小分類を 1 件のテスト項目としているために必要なテスト件数が不足してしまいます。テスト仕様書のフォーマットの改定が必要です。

【アドバイス】 問題は詳細仕様書の記述が粗いことにある

本問題は単純に単体テスト仕様書のフォーマットや小分類の問題なのではないでしょう。フォーマットの改定をしても無意味です。

単体テスト仕様書のテスト項目が粗いのは、もともと詳細設計書自体の記述項目が粗いのではないでしょうか。詳細設計書が細目に渡って定義されていなければ、単体テスト仕様書も細目に渡った記述は不可能です。詳細設計書の見直しをまず実行し、テスト仕様書が問題なく書けるような詳細設計書の作成が必要です。

One point　Lesson　：　問題の本質はフォーマットではなく仕様書の精度にある

Case＃64　貧弱なテスト仕様書による評価作業

テスト仕様書の完成度が低い場合や極端な場合は存在しない場合、評価業務に後戻り作業が発生し、同じような評価を何度もやり直すことになり、やる気が大きくそがれてしまいます。なぜ誤字、脱字、コピペミス等が多い低品質のチェックリストになってしまうのでしょうか。評価設計の見積り検討時間が根本的に足りないために、雑な作業になってしまっているように感じます。またテスト内容理解の曖昧さが評価工程の見積りのズレの大きさの原因になっていると思います。 テスト仕様書の作成者に対して、早めの作成を催促し、明確な期限を決めるなど、自分から積極的にアクションを起こす必要があると思いますが、それだけで効果が出るのか疑問です。

- 時間不足でチェックリストの作成が雑になっている。
- 誤字、脱字、コピペミスが多い低品質のチェックリストしかない場合が多い。
- 最悪、テスト仕様書なしでテストの実行を要求される場合がある。

【アドバイス】 高い精度のテストは、高い精度の仕様書・設計書・チェックリストが必要

評価設計の見積り不足もテスト内容の曖昧さも、元をただせば、いつまでたっても決まらない仕様やあいまいな仕様に起因しているものと思われます。評価テストの元になるのは精度の高い詳細設計書・基本設計書・要求仕様書に他なりません。元々の仕様書・設計書の精度が高くなければ精度の高い評価設計書も評価チェックリストも作成できません。精度の高い評価テスト業務を実現するためには、要件定義者、設計者、評価者のそれぞれの業務能力を同時に高める以外に方法はありません。評価テストチームは、製造物の品質を確認することだけが自分たちの仕事ではなく、それ以前の設計・製造業務のレベルをチェックすることも自分たちの仕事であるという認識が必要です。

One point　Lesson　：　評価テスト者は設計者・要件定義者の業務品質もチェックすること

Case＃65　チェックリスト流用による評価作業は必ずしも効率的ではない

過去のチェックリストを流用する場合、そのチェックリストによるテスト結果の有効性や網羅性が不明なため、使ったとしても漏れや再確認が多く発生し、結果工数オーバーになってしまうこともあります。

【アドバイス】 新規に作成する気持ちで取り組むこと

過去のチェックリストがそのまま流用できる場合などそう多くはないでしょう。多くの派生開発は皆一品料理のオーダーメイドであり、全く同じ開発などほとんどありません。たまに似ている開発もありますが、その実態は似て非なるものだと思った方が無難です。チェックリストの流用は、開発におけるソースコードの無定見なコピペと同様に非常に危険な行為だという強いリスク認識を持つ必要があります。流用やコピペの行動の背景にある心理は、楽をして仕事をこなしたいという気持ちであり、その仕事の内容に対しての自分自身としての見解や考慮という自律的な思考や行動が大きく抜け落ちてしまうということです。そのような仕事は必ず失敗することでしょう。特に、ドキュメント・ベースの仕事を実行していない組織においては流用しようとしているチェックリストというドキュメント自体がメンテナンスされておらず、単なる紙くず同然のものになっている場合が多いのです。

もしどうしても流用したいのであれば、適正な詳細設計書と一々つき合わせてみて、全ての間違いを修正し、抜けを補うめんどうな作業を行う必要があります。そのようなことをする位なら、最初から自分たちでチェックリストを作成した方が結局効率的で正確なものが作成できるのではないでしょうか。

One point　Lesson　：　評価チェックリストの流用は危険なコピペのようなもの

Case＃66　影響度があいまいでは精度の高い評価業務は実施できない


Pieces & Parts

評価テスト設計前の調査工程では、新規仕様が既存仕様に影響を及ぼす部分の明確化が必要です。影響範囲が不明なままでは、「あるべき結果」が曖昧なままテスト設計をすることになり、テスト仕様書の精度、及び、後工程（テスト準備、実施）の精度が下がり、不具合検出を見逃し、NG なのに OK を付けてしまう等のリスクに繋がります。

開発仕様の大きさにもよりますが、特に既存部分の仕様は、範囲が広いため、テスト設計完了までに、影響度の洗い出しで上がった項目に対しての仕様を全て把握できない場合があります。影響範囲の調査や絞込みにあたっては影響度の大きい既存仕様の情報を開発チームから入手する必要があると思います。

- 仕様変更部の及ぼす影響範囲が不明なため有効な評価チェックリストが作成できない。
- 既存仕様の範囲が広すぎて影響度表の作成や影響度の大小の特定が難しい。
- 開発チームから評価チームへの影響度情報の提供がなされない。

【アドバイス】 設計書に拠らない正確な影響度表の作成方法

既存仕様への影響度を調査する仕事は、第一義的には開発チームの義務であると言えます。既存仕様に対する影響度を調査しなければ不具合のない設計も製造もできません。派生開発においては、開発チームは既存仕様に対する悪影響を避けながら設計および製造を行わなければ正しい開発を行ったとは言えません。評価チームは開発チームから提供された既存仕様に対する影響度表を基に評価設計を行うのが普通のやり方です。その上で評価チームは今までの評価の経験を活かして開発チームが見落しそうな影響部分のテストを追加するのが正常なあり方でしょう。影響度の調査を一から評価チームが行わなければならない状態は異常なことです。今後は影響度の調査および影響度表の提出を強く開発チームに要請してください。

影響度表の作成には正確な設計書が必要です。現在多くの設計書は適切な更新が行われていない場合が多く、影響度表の作成は困難を極めています。このような状況下において、ある程度使える影響度表を作成するためのポイントは下記の活動が必要です。

【影響度表作成のための準備活動】

単体テスト、結合テスト、総合テストにおいて発見された他のモジュールに影響した不具合ならびに影響が及んでいると気づいたことを、全のテスト実行者が協力して影響度表に継続的に書き込むこと。当初は大雑把な影響度表しかなかった場合でも、派生開発の都度このことを実行していれば、完全な設計書がない場合でも、回数を重ねるごとに非常に精度の高い影響度表になっていきます。開発チームにおける単体・結合テストおよび評価チームにおける総合テストにおいて両チームが連携してこの活動を地道に実行する必要があります。

One point　Lesson　：　内部テストにおける不具合情報をもとに影響度表を作成すること

Case＃67　品質の低いチェックリストになってしまう

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。

- 誤字・脱字・コピペミスが多い低品質のチェックリストが多いため評価作業の効率が悪い。
- 評価設計に必要な見積り工数の不足が雑なチェックリストの原因となっている。

【アドバイス】 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開発関係者全員が持つ必要があります。

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を行う必要があります。

One point　Lesson　：　評価工程を改善するには要件定義・見積り・設計工程の改善が必要

Case＃68　ドキュメントの品質が悪く評価テストの効率が下がる

障害票および質問票へ記載する時間に追われることにより、本来確認すべきテストが行えません。要求仕様書、ターミナル仕様書、初期設計書でそれぞれ内容が異なることが記述されている場合、どれが本当に正しいかの判断がつかないため、確認に手間取り、時間を浪費することになります。 これらのドキュメントの精度を上げる必要性を強く感じており、随時、ベンダー側へ更新のお願いを働きかけるようにすること、およびテスト設計書作成の段階で確認できることは事前に確認を済ませるようにしたいと思います。

【アドバイス】 精度の悪いドキュメントに対する対応策

精度の悪いドキュメントに対する対応策は次の通りです。

1. 貴社で設計・製造・評価を請負っている場合

①仕様検討工程および設計・製造工程において、仕様の疑問点・不明点をすべて解消しておくこと、および仕様書・設計書等における誤記や記述漏れを修正しておくことを開発リーダーに強く要求しておくこと。

②同様に、開発チームとベンダー間の仕様のQ＆Aを最初の段階から把握しておくこと。

③評価設計時にまだ疑問点や不明点が残っている場合は、まずは社内の設計・製造者などに何が”正”なのかを確認すること。

④評価チームにて見つけた仕様書などの誤記・不明点・矛盾点はドキュメントの不具合として不具合票を発行し、ベンダー側に通知すること。ドキュメントの不正点はバグ以上に悪質な欠陥であるという認識をベンダー側と共有すること。単なる書類の修正ではない。

2. 貴社で評価業務のみを請負っている場合

設計・製造工程が貴社担当でない場合の対応は困難ですが、下記の対応策を実行する必要があります。

①仕様検討・設計・製造の各工程における、担当他社とベンダー間における仕様の疑問点・不明点に関するQ＆A情報および仕様書・設計書等における誤記や記述漏れの修正情報のすべてを早い時点で入手しておくこと。

②評価設計時にまだ疑問点や不明点が残っている場合は、集中的にベンダー側にQ＆A表や直接的コミュニケーションなどの手段で、何が”正”なのかを確認すること。

③評価チームにて見つけた仕様書などの誤記・不明点・矛盾点はドキュメントの不具合として不具合票を発行し、ベンダー側に通知すること。

One point　Lesson　：　やるべき事をやるべき時に

Case＃69　評価設計ノウハウの継承ができていない

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。

【アドバイス】 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ないでしょう。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てください。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。

One point　Lesson　：　やる気のない者につける薬はない

Case＃70　統一的な要因表・テストパターン表のフォーマットを作成したい

チェックリスト作成前の要因表やテストパターン表の作成については、フォーマットや記載内容のイメージはありますが、まだ共有化するまでには至っていません。要因表やテストパターン表は、対象顧客システムごとに異なってくることが想定されますので統一的なフォーマットや記載内容のイメージについての検討が必要になります。要因表やパターン表を用意せずチェックリストを作ってもチェック項目の漏れが発生しやすく、またテストの目的や意味をメンバーに説明しにくくなりますので、早く統一的なフォーマットを作成したいと思います。

【アドバイス】 統一フォーマットによる効率化を

評価テストの対象である顧客システムが異なったとしても、全業種・業態に共通する要因表・テストパターンは必要です。基本的にはいずれも情報処理システムであり、ある仕様に基づいて作成されたデータ処理について所定の入力条件に対して規定の出力が行われるか否かの正常系のチェックおよびエラー処理および異常処理が適切に実行されるかという異常系のチェックの二系統のテストで構成されています。これらのテストは、内容は異なっても、基本的に同じフォーマットの要因表およびテストパターンで実行可能でしょう。現在、それぞれのチーム毎に使用しているサンプルを集めて分析し、全チームでの使用が可能な統一フォーマットを作成することは可能でしょう。統一フォーマットを基にして、それぞれのチームにおける異なった業態別顧客システムについてこれまでに蓄積した評価のノウハウを盛り込んでいただきたいと思います。

要因表やパターン表の統一フォーマットの公開にあたっては、業態別のサンプル等も添付し、その意味を分かりやすく説明した手順書もしくはガイドラインを作成して下さい。今後はそれを使用すれば評価計画書の作成も容易になり、評価業務の精度や効率性も上がるだけではなく、後進の人たちの教育時間も大幅に短縮できるものと思います。

One point　Lesson　：　ベストでなくともベターなものを早く出すこと（巧遅は拙速にしかず）

☑ Check List　＃13　：　**評価テストドキュメントのチェックリスト**

評価テストドキュメントに関して次のチェックリストを試してみよう
〇：できている　△：どちらともいえない　×：できていない

☐　テストの前にテストドキュメントは用意されているか。
☐　テストドキュメントは精度の高い詳細設計書に基づいて作成されたか。
☐　詳細設計書は妥当な時期に提供されたか。
☐　仕様変更影響度に関する詳細情報は設計者から提供されたか。
☐　仕様変更影響部に関するチェック項目は網羅されているか。
☐　テスト対象の仕様・機能の入出力条件は全て網羅されているか。
☐　テスト項目の粒度は妥当か。
☐　コピペによる誤字・脱字・ミスはないか。
☐　テストドキュメントは統一的フォーマットが使用されているか。

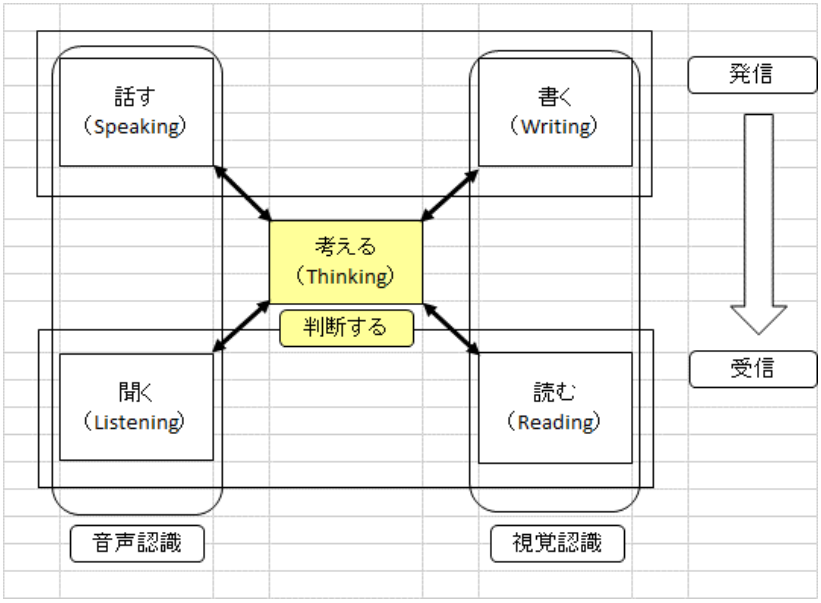
【ふり返りメモ】

第5章　考えザルの巻

私たちは、いつも二人の自分をもっているようです。ものごとに感情反応をする自分と、ものごとを冷静に判断する自分の二人の自分がいることを時々感じることがあります。感情反応を示す自分は、ものごとに対して深く考えることなく直感的な判断や行動をしている自分であり、普段はこの直感モードで自動運転をしているようです。一方、難しい判断が迫られる事態においては、直感的な自分が理性的な自分に、「これでいいのか？」というような問いかけをしていることに気がつくことがあるでしょう。このように、直感的な自分と理性的な自分の間の対話を、ここでは、「考える」コミュニケーションと呼びたいと思います。

「考える」コミュニケーション能力は、常に「話す」「聞く」「書く」「読む」の四つの基本的なコミュニケーション能力の間にあって、ものごとを統合的に判断する役割を果たしているようです。

本章では、この「考える」ことがコミュニケーションにおいてどのような働きをしているのかについて事例にそって明らかにしていきたいと思います。



Case＃71　古いスケジュール表で進捗報告をする担当者

ある担当者は、毎日のミーティングで古いスケジュール表に基づいて問題無しと報告し続けていましたが、実際には 6 日遅れであったことが分かりました。この担当者はプロマネの指示に従って工数を見直していましたが、スケジュール表を修正していませんでした。一方、プロマネにおいては、その担当が工数を見直すという報告を受けていましたが、見直し結果がいつ確定するのかを確認しておらず、担当者からの口頭での報告を鵜呑みにしていました。

結局、担当者・プロマネ間のコミュニケーションが十分でなく、スケジュール表が修正されていないことに気づきませんでした。今後の対策として、以下のことを実行する予定です。

①担当の計画が実現可能で、見積り工数と矛盾していないか、チェックします。

②毎日のミーティングにて、計画通りに作業が進んでいるのかを、進捗率と作業時間の両面でチェックします。

③変更発生の都度、工数やスケジュールに関する打合せを行います。

【アドバイス】 自分の頭でものごとを考えられるメンバーの育成を

日次会議で進捗に関してどのような会話をしていたのでしょうか。たとえば、単に”進捗遅れはありませんか？”などと聞くことはほとんど無意味です。〇〇業務は△△が期限ですが状況はどうですか、と言うような聞き方をしていれば、古いスケジュール表を見ていた担当者は、その場で間違いに気づいたはずです。また担当者においては、自分で工数を見直したにもかかわらず古いスケジュール表に基づいた報告を平気でしていたとは、全く信じられない行動です。自分で工数を見直したことも忘れていたとしたら、仕事に取り組む心構えができていないのではと思います。厳重注意を促す必要があります。リーダーによるチェック業務をこれ以上増やさないためにも、日々の進捗報告会の場を利用して、メンバー自身が自分の頭を働かせ行動するという自律性の訓練を実行してください。

One point　Lesson　：　何の考えもなく流れ作業的な開発をしてはいけない

Case＃72　問題の解決案を自分で考えられない

プロジェクトの遂行時に、発生した問題の内容によっては解決案を自分で検討できていない場合があり、解決案を自分で考えずに上司に聞いてしまう場面があります。自分で解決するという意識が足りないと思います。プロジェクトを管理している自分が問題を解決するという意識を持ち、上司には「どうしましょうか？」と聞くのではなく、「こうしようと思います」と報告するようにします。どうしても解決できない場合に上司に指示を仰ぐようにします。

【アドバイス】 問題の真因について考えること

この問題は意識の問題ではありません。なぜ自分で解決策を考えられないのかということが問題なのです。もっと自分で考えましょう。何かの問題が発生した場合、その問題の表面上の現象ばかりに目を奪われるのではなく、さらにその問題を発生させている根本的な原因について考えてください。本当の真因が分かれば、それを解決するための方法もいくつか考えられるでしょう。その方法の具体策について自分に知識がなければ、知識のありそうな人に聞くのもいいでしょう。自分で深く考えることもせずに、最初から「どうしましょう」などと人に聞くことはやめましょう。

One point　Lesson　：　考えることが嫌いな人はいつまでたっても進歩できない

Case＃73　不具合の真因発見は難しい

単純＋単純≠２単純
＝複雑

　不具合を分析する中で、何が原因かを考えた際、分析の着目点が悪く、別の方向になぜなぜの分析を行っていたため、真因に辿り着く事が出来ない事がありました。分析結果のレビューの中で指摘された事で、分析の方向が間違っている事に気付かされました。なぜなぜ分析を行う際、1つの視点では真因に辿り着く事は出来ない事がある為、複数の方向・視点で分析し、他者の意見も取り入れる必要があると思いました。

【アドバイス】 不明な問題の解き方

　ものごとの認識において、心理学では「人は、自分の都合の良いように判断しがち」といわれています。ものごとの事実や本質を見抜くためには、他者の視点で考えてみることで自分の思い込みを排除することが必要です。不具合問題に限らず、我々の仕事の大部分は「不明な問題を解く」ことにあります。不明な問題を解く方法として「数学の考え方17か条」を以下に紹介します。ちなみに、他者の視点は、次の資料の「⑩視点を変えよう」に相当します。

【数学の考え方17か条】（秋山仁、東京理科大学教授）

①「分類・整理しよう」 10進法、12進法、60進法、2進法、これらの数の表し方は、それぞれ、10、12、60、2というまとまりに数を分類・整理して、表されている。分類・整理することで対象物の特徴が明確になり、対象物間の相関関係も明確になる。

②「図や表にしよう」 図や表にすることで、それまで見えなかったことが見えてくるようになり、複雑な事象も単純な事象の組み合わせであることが判明する。

③「簡単な模型をつくろう」 図を書いたり、模型を作るなどして考えると、難しいことが考えやすくなったりする。ハードウェアにおけるモックアップやソフトウェアにおけるプロトタイプなどがそれである。

④「基準をそろえよう」 基準をそろえるとは、（1）ものさし・単位をそろえる、（2）基点を定める、ということであり、ばらばらの基準であったものが統一基準のもとに分かりやすくなる。

⑤「数学（ロジック）の言葉で表そう」 問題を方程式で表すということ。たとえば4y＝600。自然言語で長々と問題事象を記述しても見えなかったものが一目瞭然となる。

⑥「小さな例で試してみよう」 解けそうにもない複雑な問題は、条件を易しくした例で試して見れば意外と解ける場合も多い。

⑦「難しい問題は分割しよう」 難しい問題は、解けそうな大きさに分割して考えること。複数の要因が重なった問題は、要因ごとに分割して答えを積み上げることで全体としての解が得られる場合もある。

⑧「必要条件で絞り込もう」 必要条件とは、何かが成り立つために必要な条件のこと。複数の必要条件を出していくことで、問題を絞り込み、最終的に解を得る。

⑨「特定の要素に注目しよう」 データを整理し、それから何らかの傾向を読み取るためには、特定の要素に注目すること。特定の要素として、平均値・中央値・標準偏差などがある。

⑩「視点を変えよう」 だまし絵などに見られるように一見してある物に見えるものも、異なった角度から見た場合違うものに見える場合がある。しかし、その実体は同一である。一見して不明なものも違う視点で見れば、自明のものである場合がある。

⑪「逆に考えてみよう」 見えている姿が裏向きのため理解できなかったものが、それを反転しただけで理解できる姿が見えてくる場合もある。自分が持っている前提条件や思い込みを一たん否定したアプローチを取ってみることも有効だろう。

⑫「操作は1つずつ片付けよう」 難しい問題を考えるとき、いくつもの操作を同時に考えると混乱する。そのような場合は、1つずつ手順を踏んで考えよう。

⑬「知っている事実を活用しよう」 長方形の面積＝横x縦を知っていれば、その延長問題として三角形の面積＝底辺x高さ÷2が分かるように、すでに持っている答えを活用してより複雑な問題が解ける。

⑭「規則性を探そう」 例えば、バーコード。13桁の数の中にも規則性が隠れている。この数字は、商品に関する情報を表している。最初の2桁は国名、次の5桁がメーカー、さらに次の5桁が商品名を表している。そして、最後の1桁はチェックデジットと言い、バーコードの読み取りミスを防ぐ役割をしている。バーコードに書かれた数に注目してみよう。
（左から奇数桁目の数の和）＋3×（左から偶数桁目の数の和）＝10の倍数、となるように、チェックデジットは決められている。例えば「4983248006644」のバーコードを例に計算してみると、(4+8+2+8+0+6+4)+3×(9+3+4+0+6+4)=32+3×26=110 となり、10の倍数になっていることがわかる。つまり、バーコードを読み取ったときに10の倍数になっていなければ、どこかで読み取りミスをしている、ということがわかる。

⑮「対応をつけて考えよう」「対応をつける」とは、何かと何かをペアにして考えるということです。例えば、目の前に乱雑に置いてある数百個のボールを、「だぶらず、漏らさず」数える場合、番号を書いたシールを一つずつ貼っていけばうまくいくでしょう。物の数え方の基本原則は「だぶらず、漏らさず」であり、そのためにボールとシールをペアにすることにより容易にその目的は達成されるでしょう。

⑯「自然からヒントを得よう」 同じ質量のもので表面積が最も小さい形は何かを考えた場合、自然界における水滴や溶けた水銀などから類推すれば球ということが自然と判明してくるでしょう。

⑰「部分から全体を把握しよう」 さまざまなデータから全体の傾向や状態を推測できる。例えば、「標本調査」や標本調査をするときの標本の選び方の「無作為抽出」や，標本調査によって母集団の大きさを推定する「比率の推定」などの手法がある。

One point Lesson ： 問題の解き方は一通りではない

Case＃74　表面的な言葉の理解では本質が見抜けない

「情けは人の為ならず」の本当の意味について、テレビでの間違えやすい格言やことわざ特集で本当の意味が分かりました。【情けは人の為ならず】とは、「情けをかけることは他人の成長を止めてしまうので、行わないほうが良い」という意味ではなく、「他人に情けをかければ巡り巡って自分に返ってくるので、積極的に行うほうが良い」というのが本来の意味でした。 上記のように、自分自身においても、見えたもの、聞こえたものだけで判断してしまっていました。少し時間をおいて考えると本質が見えてくることもありますが、その時間を調整する事が出来ない場合もあります。ものごとの本質を見抜くために、自分一人の力には限界があるという認識のもと、多くの方と話を行い、見えたもの、聞こえたものから本質を想像するトレーニングを行うことや、意見交換を行うことで、他の人がつかんだ本質の共有を受け、自分の視野を広げたいと思います。

【アドバイス】 教養は本質を見抜く力になる

見聞きした情報の理解度の深さや広さのことを教養と言います。いままで見聞きしたこと、経験したこと、学習したことの量と質が自分の理解力や教養の深さや広さを決定します。他人が言っている一見よさそうなことにいきなり飛びつくことは危険です。他の複数の情報にもあたって、一旦自分の中で思考し、そしゃくする必要があります。それが自分の「自律性」を育て教養を深めます。外来のカタカナ言葉などは必ず英語の辞書でその原義を確かめて見る必要があります。例えばリストラ（restructuring）は、本来、改造する、改革するという意味であり、機能しなくなった組織の再構築を行うということですが、日本においては首切りという意味に改変されてしまったようです。せっかく欧米において発案されたすばらしい考え方の本質が歪められ、卑近な意味に歪曲されて流通していることは誠に残念なことです。

One point　Lesson　：　複数の視点が交差するポイントが本質に近い

Case＃75　顧客提示の方法で修正を行い失敗した

ある機能について顧客から提示された修正方法で対応を行いましたが、総合テストにて不都合なケースが発見されてしまいました。問題の原因としては、提示された修正方法に対して何も疑問を持たずに対応してしまったことにあったと思います。

今後は仕様変更の内容説明を受ける際には、仕様変更の目的の把握および提示内容の精査を必ず行うようにしたいと思います。

【アドバイス】 自律性のある行動を

原因として、顧客から提示された修正方法で対応したことが挙げられていますが、顧客のシステム部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ことを忘れないで下さい。依頼を受ける立場の我々にとって大事なことの一つに「自律性」をもつということがあります。つまり他人や他部署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有効な仕事の基となります。 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。この自分自身の目で確かめ、自分自身の頭で判断するという「自律性」については、チームの全てのメンバーに対しても促すようにしてください。

One point　Lesson　：　自分の目で観察し、自分の頭で考え行動すること

Case＃76　作業する前から細かく教えるのは教育にならないのかも知れない

コストや納期目標の達成を考えると、部下のスキルによっては細かい作業内容や方法まで指示を出してしまうことがありますが、スキルアップを考えると答えまでは教えずにもう少し本人に考えさせることも必要なのではという思いがあります。部下の自発的な質問に答えることは教育になりますが、作業する前から教えてしまうのは教え過ぎなのではないかとも思います。部下が目的を理解したうえで仕事内容に対し質問してもらうことが、考える力の教育に結びつくのではないかと思います。よって、仕事内容を教えていくのではなく、仕事の目的をしっかり伝えるということを大切にしていこうと思っています。

【アドバイス】 人のレベルに合わせて指導を行うこと

教育の基本は、それぞれの人のレベルに合わせて指導を行うことにあります。理解が足りない人には細かく、理解が進んでいる人には更に高度な指示が必要です。一律に、作業の前に細かな指示を出さないことが、全ての部下にものごとを考えさせる機会になるとは限りません。分かっていない部下への指示はつい細かくなり勝ちですが、いきなり自分で考えろと言ってもなかなか難しいでしょう。このような人に対しては少し面倒ではありますが、指示した内容のポイントを復唱させ、逆にその指示ポイントの意味を答えさせるようにして見てはいかがでしょうか。相手の持っている疑問点・不明点をどれだけ聞き出せるかが重要なポイントになるでしょう。

また、指示を与える人間においては最低限、その仕事の目的・意味・背景について明確に説明する義務があります。多くのリーダーにおいては自分自身がそれらのことについて理解していないため、部下に適切な指示ができていないように思われます。さらに部下に仕事を指示するにあたっては必ず資料に基づいて説明する必要があります。ドキュメント・ベースの仕事は、説明忘れを防止し、お互いの理解の一致を確認するのに有効です。

One point　Lesson　：　何事につけ、多すぎるより少し足りないくらい方が良い

☑ Check List　＃14　：　**「考える」コミュニケーションのチェックリスト**

考えることに関する次のチェックリストを試してみよう

〇：できている　△：どちらともいえない　×：できていない

☐　仕事には必ず問題が潜んでいるという意識で取り組んでいるか。
☐　問題の解決策を自分の頭で考えているか。
☐　複数の視点で問題を分析しているか。
☐　複数の解決策を考えているか。
☐　他人の言うことを鵜呑みにしていないか。
☐　問題を自分の目で観察し、自分の頭で判断し行動しているか。

【ふり返りメモ】

第6章　伝わるコミュニケーション

コミュニケーションの各能力に起因した問題を第1章から第5章で取り上げましたが、本章においては、各コミュニケーション能力が複合的に絡み合うことで発生している現場の問題を、特徴的なキーワードごとにグルーピング化し、それぞれについて事例に沿って検証を進めたいと思います。

1. コミュニケーション・ギャップ
2. 弱腰のネゴシエーション
3. やるべきことがやられない
4. 見えない目的・目標
5. 仕様誤解
6. そろわない足並み
7. 分かりあうことは難しい
8. 進んでいるはずが遅れていた
9. ずれているレビューの視点
10. ルール違反のコミュニケーション
11. 意味のない会議
12. 他人に教えたくない

1.　コミュニケーション・ギャップ

コミュニケーション・ギャップとは、情報伝達の断絶部分のことを意味しますが、情報の断絶は工程間、人間同士間、社内組織間、会社間などすべての情報伝達間において発生しています。それは、関わり合う人が多ければ多いほど、関わり合う組織が多ければ多いほど、情報伝達の精度および速度を劣化させ、開発を誤った方向に導き失敗を招いています。いくつかの事例に沿って検討を加えていきたいと思います。

Case＃77　工程別複数社分割発注の問題

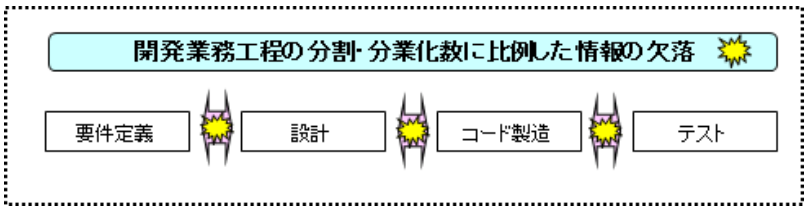
最近は開発工程の分業化が進み、設計からテストまでを一貫して受注していないことによるリスクがあるように思います。たとえば設計工程だけを受注した場合、実際にプログラム製造を行う過程でしか見えてこない設計の問題を見落としやすい危険性が増えているように感じています。このようなリスクを減らすためには設計工程において製造以降を意識しなければならず、過去の設計ミスを設計に反映する必要があります。さらにそのノウハウを蓄積および共有する必要も出てきます。できれば設計工程だけの受注はしたくありません。

【アドバイス】 分離分業への対抗方法

近年オフショア開発の進展に伴って、要件定義はベンダー、設計は国内下請け会社、製造は海外オフショア会社、評価は国内下請け会社もしくはオフショアなどと、一つのプロジェクトの各工程を複数の会社で分業するような開発形態が増えてきました。相談者が感じているように、この分業体制には多くのリスクや問題点があります。実際この「工程別分業方式」による開発で多くのプロジェクトが失敗しているのを見てきました。この方式の最大の問題点は、情報の共有が極めて困難であるということです。従来の開発体制は、ベンダー側にてプロジェクトの統合管理、要件定義およびシステムの基幹部の開発および評価を担い、不足要員を国内下請け会社で補う形が一般的でしたが、現在は外形的にはベンダー側における開発や評価業務は下請け会社にシフトされており、特に製造工程はコストメリットの追求のために海外オフショア会社に発注されている場合が多いでしょう。

この工程別分業方式の致命的な欠陥は次の通りです。

1. 各工程が異なった会社に分離・分散されたことによる共有すべき情報の分断化



一社で全ての開発工程を担う場合に比べて、複数社で分割した工程を担うことは極めて難しいことは誰にも容易に分かる道理のはずですが、多くのベンダーにおいてはその認識が極めて薄いようです。ソフトウェア開発は、一社で全てを実行しても失敗する危険性が高いのにもかかわらず、どのようなリスクがあるのかも深く考えず、単にリスク分散だとかコスト競争だとかの旗印のもとに複数社に分割発注することは正気の沙汰とは思えません。工程別分業方式における主なリスクは次の通りです。

①統合管理および統合コミュニケーションの欠落
　元請会社において下請け各社の進捗管理や品質管理をまとめる統合管理および統合コミュニケーションの実行がなされないこと。
②ドキュメント・ベースの開発の喪失
　ベンダーも含めた各社における完全な要求仕様書や設計書などのドキュメントに基づく開発および評価の実行がなされないこと。
③統合的プロジェクト体制の構築失敗
　ベンダーも含めた各社における開発力、すなわち必要な人材や体制の投入状況に関する検証の実行がなされないこと。
　ベンダー側において上記三点を実行していなければ、仕事も責任を放棄した丸投げ発注になり、プロジェクトは必ず失敗するでしょう。ベンダーの下に有機的に統合された分業体制を「情報共有分業」と呼ぶならば、そうでない分業体制は悪しき「情報分離分業」と言えるでしょう。
　上記三点に関しては、分業の有無にかかわらず本来一つの開発組織においても実行されなければならないプロジェクトを成功させる重要な要因です。下請け会社においては、ベンダー側において欠落している組織機能の補完を行う行動をとることが新たなビジネスチャンスともなるはすです。

2. 文化・言語が異なる海外オフショア会社との間のコミュニケーションの阻害
　十分な統合管理もできず、ドキュメント・ベースの開発もできず、十分な開発体制の構築もできないような組織において、文化も言語も異なる海外の会社に設計・製造・評価などの工程を発注したらどうなるかは誰にでも分かることでしょう。すでに分離・分断されている工程間にさらに深い溝を掘るような行為だとは思いませんか。要求仕様書の行間を読めというような非科学的な姿勢しか持ち合わせていない人たちが作成した貧弱な仕様書に基づいて設計や製造を発注されたオフショア会社においては、きっと書かれているものだけしか設計・製造できないでしょう。おまけにあいまいな日本語は非ロジカル言語であり、その日本語で書かれた貧弱な要求仕様書は中国やベトナムの技術者にとって意味不明なしろものなのかも知れません。まともなプログラムができなくて当たりまえのことでしょう。それを補うために新たにまたブリッジSEなる職種を作りだしたり、未完のプログラムの受け入れ検査組織を作ったり、バグだらけの未完のプログラムを国内で補修するために新たな技術者を用意したり、結局オフショアのコストメリットが無くなるばかりか、赤字に陥り、劣悪な品質や納期遅延の結果を生んでいるのでしょう。
　オフショアのみならず開発を成功させることのできる開発組織は、前記の統合管理、ドキュメント・ベース開発および優れた開発能力をもった組織だけでしょう。このことはベンダー側および下請け側の両方に言えることです。

One point Lesson ： プロジェクト成功の三要件は、統合プロジェクト管理、ドキュメント・ベース開発および優れた開発力にある。

Case＃78　他社とのコミュニケーション不足がQCDに悪影響を与えている

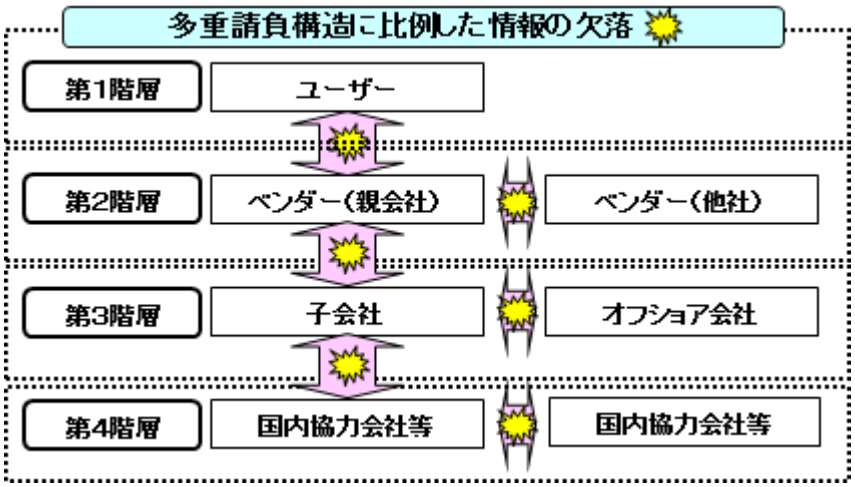
プロジェクト実施時の他社とのコミュニケーション不足のため、必要な情報が上がってこず状況が見えてきません。社内では対面での情報伝達が可能ですが、他社に対してはメールのみだと、返信が無いことが多々あり、こちらが想定していた情報と他社との認識のズレが発生しやすくQCDに問題が発生する可能性が高くなります。コミュニケーションが十分に取れないときは、対面で打合せを行う様にして、メール・電話での確認等、楽な手段に走らないようにしたいのですがなかなか実現できません。

【アドバイス】 感情的・情緒的な苦手意識を克服して積極的なコミュニケーションを

他社とのコミュニケーション不足、特に発注元会社とのコミュニケーション不足は広く見られる問題です。要求元の会社とのコミュニケーション不足は自社における業務品質や成果物のQCDに大きな影響を及ぼすことは誰でも容易に想像できることです。それにもかかわらず多くの場合は間接的なコミュニケーションばかりで皆さん何故か対面コミュニケーションをとろうとしません。原因は発注側・受注側の双方にあり、ともに仕事の遂行にあたっての使命感や情熱の薄さや当事者意識の不足もあいまって感情的ないしは情緒的思考が先行し、ビジネスライクな合理性に基づいた自律的な行動が行われないことにあるように思われます。そのような姿勢が”早くこの仕事から逃げたい”とか”緊張感を回避したい”とかの思考・行動を招き、必要な直接的コミュニケーションを回避する結果を招いているようです。そのような人たちの言い訳は常に”時間がないから”なのです。

要求仕様を確かなものにするためには要求元との密着した直接コミュニケーションが必須であり、また開発工程を確かなものにするためには開発・評価チームにおける情報共有の場である日次進捗会議の実行が必要です。

One point Lesson ： コミュニケーション・ギャップを埋めるには、コミュニケーションをするしかない



Case＃79　製造工程におけるデグレの懸念

設計担当を当社、製造担当を当社および中国オフショア社で機能ごとに開発していますが、複数社で同一ソースの修正が発生するため、デグレ的な影響が懸念されます。特に一部機能で同一モジュールの修正が発生するため、問題発生の確率が高くなります。

当社からは何度かソースレベルで重複しないように担当分けをしていただきたい旨の要望を具体的な機能分担も併せて提案させていただきました。しかしながらベンダー側においては、短納期のためマンパワー的に一社での開発が難しいことおよびオフショア発注目標のノルマ達成のために当社の提案は採用されませんでした。

【アドバイス】 複数社担当開発におけるリスクの解消を

複数社に製造担当が分かれた開発における主なリスクは次の通りです。

①仕様の理解が相互に異なったためにおこるデグレや実装漏れなど。

②同一モジュール製造に複数社がかかわった場合、1社の開発遅延がロードブロックとなり、他社が製造開始できないこと。

③1社の不具合によってテストが進捗できないこと。

④複数社におけるベースソース部の改修の同期がとれず正式版のビルドができないために結合・総合テストが開始できないこと。

⑤オフショア会社の場合の注意点は次の通りです。

・意思の疎通が取りにくい、時間がかかること。修正対応が遅いこと。仕様に記述されていないことは質問もしてこないし、製造することもないこと。日本語仕様の解釈能力が低いこと。ただし日本語仕様書自体の品質も悪いこと。

これらのリスクを解消する対策を実行する必要があります。週一回の定例会議だけではなく、自社のチーム内では日次の短時間情報共有会議を毎日実行し、日々の進捗の確認、全ての問題・課題の解消、情報共有を毎日実行してください。またベンダーおよび他社とのコミュニケーションも密接に行う必要があります。

次に肝心なことは、ベンダー側においては必ず複数社に対する統合管理を実施しなければならないということです。ベンダー側において、下記の役割の実行が不可欠になります。

【プロジェクト統合管理の役割】

①関係各社における進捗の相互調整

②関係各社における問題・課題の相互調整

③関係各社間の情報共有

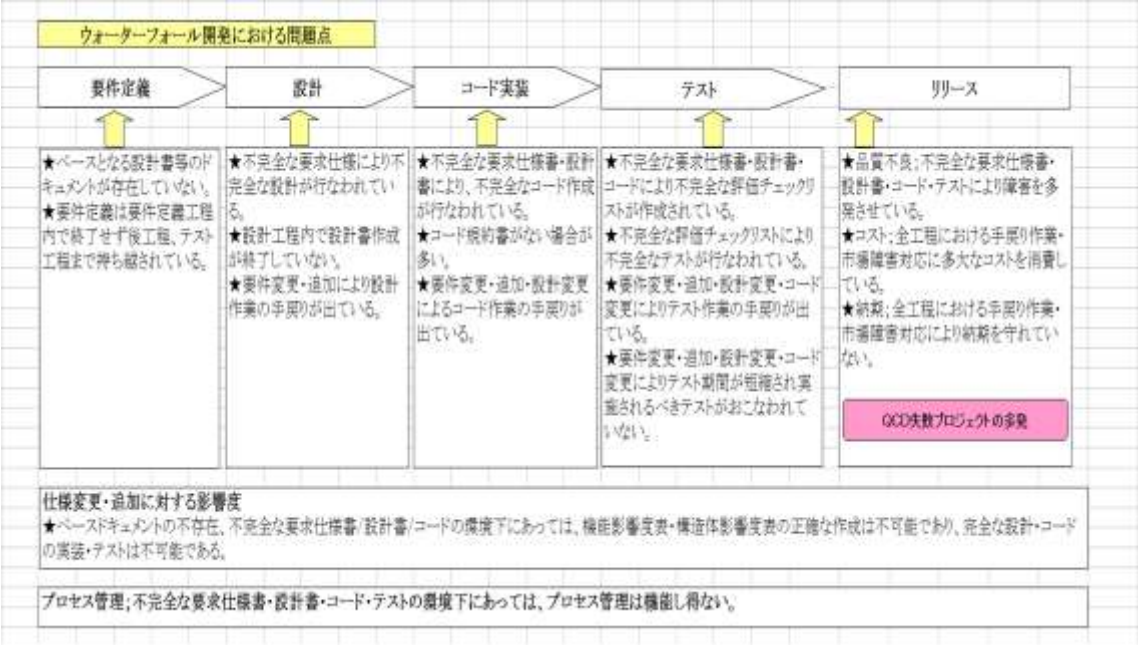
④関係各社間の協力体制および共通開発ルールの確立

⑤開発仕様の早期提示

今回は残念ながら複数の会社が同一モジュールのソース変更を行う場合があるようですが、変更理由記録・変更履歴やソースにおけるコメント記入など、後日のデバッグや不具合修正に必要な記録をとるような共通のルールを取り交わしておく必要があります。

短納期でしかもドキュメントが不足している場合の最後の手段はベンダー側を巻き込んで全ての開発メンバーが同一場所に集結して開発を実行することです。

One point Lesson ： 統合を容易にしたければ分割数を減らすこと。組織もしかり、ソフトウェアもしかり。



[ウォーターフォール開発における問題点]

☑ Check List ＃15 ： **コミュニケーション・ギャップのチェックリスト**

コミュニケーション・ギャップに関する次のチェックリストを試してみよう
〇：できている △：どちらともいえない ×：できていない

☐ プロジェクト参加各社の統合管理は実行されているか。
☐ プロジェクトの統合プロセス管理表は運用されているか。
☐ ドキュメント・ベース開発が行われているか。
☐ コミュニケーション・ギャップを埋める対策・減らす対策とっているか。

【ふり返りメモ】

2.　弱腰のネゴシエーション

Case＃80　無理な顧客要求を断りきれない

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、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。

【アドバイス】 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ヶ月という実例を示します。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先行着手しました。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。

なお見積りの精度向上のためには、過去のプロジェクトにおける①顧客要求、②見積提示した工数日程　③実績、の三つの比較を続け、それぞれの差異の原因理由を記録しておく必要があります。原因が明確になったらそれぞれの数値の乖離幅を小さくする改善活動が必要です。見積精度を上げ乖離幅を小さくするためには、下記の対策が必要です。

【見積精度向上の三つのポイント】

①仕様に対する知識に習熟すること。
②無理・無駄・リスクを排除する改善活動の実行。
③使用技術のレベルアップ。

これらの活動の積み重ねが精度の高い見積りとなり、顧客を納得させられるデータの裏づけとなります。

One point　Lesson　：　理のない要求には理をもって対抗すること

☑ Check List ＃16 ： **ネゴシエーションのチェックリスト**

ネゴシエ―ションに関する次のチェックリストを試してみよう
〇:できている △:どちらともいえない ×:できていない

☐ 相手の要求のQCDの何が無理なのかを数字を用いて説明できるか。
☐ 妥協点を見出すための条件の提示ができるか。
☐ 強力な交渉を行う前提としてQCDに関する自組織の実力値を把握しているか。

【ふり返りメモ】

3.　やるべきことがやられない

Case＃81　作業引継ぎ時のコミュニケーション不足

作業引継ぎにて時間不足のため、設計書をもらっただけで、それまでの開発の経緯・意図・背景や要件のポイントおよび注意点などの情報共有を行わなかったために製造の手戻りが多発しました。

【アドバイス】 作業の引継ぎには正確な文書と口頭による質疑応答が必須

設計書をもらっただけで説明を受けなかったのでは仕様の理解不足になるのは当たり前のことです。仕事の引継ぎにおいて不明点や疑問点を解消しなければ、困るのは自分自身です。

コミュニケーションの時間がとれなかったということですが、結局、後戻り作業多発となりもっと多くの時間と労力を失う結果となりました。時間がないと思った時に、最初にやるべきことはコミュニケーションをおろそかにすることではなく密接にすることです。結局やるべきことを手抜きせずに、仕様を完全に理解することから始めるというやり方は、その後の作業を最短時間で完了させるベストな方法です。

One point　Lesson　：　手抜きは時間の節約にはならない

Case＃82　依頼者が作業納期を明確にしない

別の担当者からのプログラムのチェックイン期限の確認があいまいなままズルズルと時間がたってしまい、結局チェックインが間に合わず、自分の担当プログラムとの結合テストが遅延してしまいました。自工程が遅れた場合に言い訳できるよう、期限を曖昧にしているとしか思えません。

【アドバイス】 期限を確認しないまま作業に着手してはいけない

本件の問題は、依頼者や前工程担当者における責任意識の欠如にあります。自分の遅れは棚上げしておいて、次の工程の担当者には期限厳守を要求してくるという悪行があちこちで蔓延しているようです。社内の各工程間のみならず、劣悪なベンダーにおいては要求仕様をいつまでも明確にできず、時間を浪費した挙句、下請けに対して残り少ない期間で開発完了を無理強いすることも少なくありません。

明確な期限の約束がない仕事は受けるべきではないし、また他人・他社に仕事を出す場合にも自分の責任下にある仕事の期限を明示した上で依頼をすべきです。

プロの仕事は、時間や期限を約束するところから始まります。

One point　Lesson　：　まず始まりの時間と終わりの時間を互いに約束し守ること

Case＃83　守られにくい約束

信用・信頼は相手との約束を守った結果で、その評価の積み重ねとして少しずつ得られていくものと思います。その為にはコミュニケーションをこまめに取り、約束の状況を確認することが必要ですが、実際は”面倒だ””やっても楽にならない”という考えが壁となり、”時間がない”と言い訳をしていることが多いと思います。

【アドバイス】 相互義務の履行を

約束を守ること、密接なコミュニケーションをとることが、信用・信頼の重要な要素となります。約束を守るということは、一方的なことではなく、相互の問題であるという視点が必要です。どのような約束ごとにおいても、相互に各々が果たさなければならない義務が存在します。相互の義務を果たすことで約束ごとが守られ、良い仕事を完遂することか可能となります。仕事を依頼・指示する側においては、実行者がその仕事を完遂できるように環境条件を整え、助言や指導をする義務があります。一方的な言いっぱなしや指示の出しっぱなしでは義務が果たされたとは言えません。また仕事を受託・指示される側においては、依頼者の要望の真意をよく把握し、依頼者の満足度の最大化を図るべく行動しなければならないでしょう。両者において、”面倒だ”とか”やっても楽にならない”というような自己本位的な考えの壁を乗り越える必要があります。

One point　Lesson　：　仕事を指示・依頼する者にもやるべき事がある

Case＃84　来るべき連絡が来ず問題が発生した

仕様書が更新された際の連絡が無かった為、変更を知らないまま結合試験を行ったところ、インターフェースに関する障害が発生しました。仕様変更の際には連絡が来る事が当たり前だと思っていたので変更の際には連絡を頂くよう要請していませんでした。

【アドバイス】 信用できない相手のフォローは念入りに

仕様変更がなされたなら当然相手から連絡が来るのが当たり前だと思いますが、相手の職務意識レベルが低いとこのような問題が発生します。このような問題を防ぐには、事前に相手がそのようなリスク含みの人間であると判断した場合は、仕様確定の工程後も相手に変更の有無について顔を合わせる度に確認しなければならないでしょう。仕事の相手となる人間のリスク度合いを初めから判断するようにして下さい。

One point　Lesson　：　リスクは人に在り

☑ Check List　＃17　：　**相互義務のチェックリスト**

相互義務に関する次のチェックリストを試してみよう
〇:できている　△:どちらともいえない　×:できていない

☐　期限・納期の約束を文書にて互いに合意しているか。
☐　仕事の依頼者側から、必要な情報が適切な時期までに提供されているか。
☐　仕事の受託者側から、必要な成果物が適切な時期までに提出されているか。
☐　作業引継ぎにおいて文書および口頭による質疑応答を行っているか。

【ふり返りメモ】

4.　見えない目的・目標

Case＃85　役割や目標が不明確で、指揮系統が乱れている

現在のプロジェクト体制は社内の複数の事業所所属メンバーの混成チームになっており、事業所単位にミーティングを行っていましたが、プロジェクトのコミュニケーションが悪くなりました。また、仕様変更の通知が担当者に直接くることや、不具合発生時の連絡経路が適切でない場合がありました。これらの原因として次の二点があったと思います。①開発体制は、自社の事業所単位にグループ分けをしていたこと。②メンバーの変動にともなって体制図や役割および目標の見直しを変更しておらず、自社内やベンダー側にも通知していなかったこと。

【アドバイス】 開発体制の機能最適化、コミュニケーション・指揮系統の正常化を

1. 開発体制の機能最適化

このプロジェクトのコミュニケーション乱れの原因は、開発体制を事業所単位のグループ分けをしていることにあります。プロジェクトの構成は開発機能に最適化した体制にしなければいけません。構成メンバーの所属事業所がどこであるのかということは開発機能とは無関係なことです。そのような不純な要素で体制を構築することは、開発組織の効率を阻害し、指揮系統の乱れの元となります。事業所単位のグループ分けを解消して開発機能に最適な体制に組み替えることによって、チーム内の各自の役割や目標を明確にする土台ができあがります。次に各自の役割や目標を具体的に明確にするためには以下の手順が必要となります。

①各自の役割や目標の何が不明確なのかを具体的に明らかにすること。

②その不明確な部分の責任をリーダー・サブリーダーの誰が担うのかを決めること。

リーダー一人で抱えきれない程の役割がある場合は、サブリーダーを増やしリーダーの役割の分散を図るべきです。根幹の職務についてはメインのリーダーが役割を持ち、プロジェクトの仕事をさらにいくつかに分割し、その責任を複数のサブリーダーが担えばチーム内の各自の役割や目標が不明確になることはないでしょう。

2. ベンダー側とのコミュニケーション・指揮系統の正常化

仕様変更やQCDにかかわる重要なコミュニケーションは、ベンダー側と貴社のリーダー間の一本のラインで実行する必要があります。重要情報に関して、両社に複数の接点がある状態を解消するようにベンダー側と取り決めておく必要があります。

ベンダー側担当者と貴社の担当者間での話し合いを否定するものではありませんが、それはあくまでも非公式参考情報としてしか認められません。メンバーに勝手な動きを許してはいけません。もしベンダー側から担当に直接仕様変更の依頼が来た場合は、リーダーを通して話をするようにベンダー側に回答させてください。このことにより、対外的なベンダーとのコミュニケーションの乱れおよび指揮系統の乱れを正常化することができるでしょう。

3. チーム全員のミーティングについて

リーダーは事前に、その会議で「何を討議するのか」および「何を決めるのか」についてしっかりと決めておき、出席者にもその内容を事前に伝えておき、ミーティングに臨んでください。その会議で不要な話題は、別途個別に話をするように議事運営をして下さい。

One point　Lesson　：　チームの指揮能力がない者がリーダーになってはいけない

Case＃86　目的・背景の説明がない依頼はゴールが見えず達成感も感じられない

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りますが、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ません。

【アドバイス】 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て下さい。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してください。

One point　Lesson　：　目的・背景が不明では、どこに向かって進めば良いのか分からない

Case＃87　何故、設計書に機能の「目的」が記述されていないのか

設計書に「処理概要」は記載されていますが、大前提となる機能の「目的」が記述されていません。また、設計レビュー時に、開発者に機能の「目的」を聞いても納得できる回答がもらえないことが時々あります。評価チームにおいても評価対象となる各機能の意味や目的が不明なままでは有効な評価テストが実施できません。

【アドバイス】 要求仕様書の筆頭に仕様の目的・意味・背景が記述されなければならない

仕様の目的・意味・背景などの記述は要求仕様書の筆頭に記載されるべきものです。しかしながら仕様の目的・意味・背景などが記述されていない要求仕様書が多いことも現実です。原因としては、要求仕様書を作成する側におけるシステム運用に関する実際的な知識不足および顧客とのコミュニケーション不足などが考えられます。仕様の目的・意味・背景が分からないままではまともな開発を行うことはできません。発生する不具合の何割かはこのことによって引き起こされています。要求仕様書にそれらの記述がなかった場合、開発チームは要求仕様書作成者に問い合わせなければいけません。設計チームは評価チームに対して設計書における機能の処理概要の説明と共にその目的・意味・背景の説明を行う義務があります。評価チームの最大の役割は開発したソフトウェアが仕様書通りに動作することを確認すること、並びに実際の市場における運用目的を達成しているか否かの検証を行うことです。仕様や機能の実際的な目的・意味・背景を知らなければ、設計書がいくら正確に記述されていても大きな誤解を生じる危険性と不安感が残ってしまいます。

One point　Lesson　：　目的・意味・背景が示されていない要求仕様書は、出発点と目的地が示されていない地図のようなもの

☑ Check List　＃１８　：　**目的・目標のチェックリスト**

目的・目標に関する次のチェックリストを試してみよう

〇：できている　△：どちらともいえない　×：できていない

☐　プロジェクトの目的・目標は数値とともに明確になっているか。

☐　自分の役割は数値とともに明確になっているか。

☐　自分のやるべき仕事の目的・目標は数値とともに明確になっているか。

【ふり返りメモ】

5.　仕様誤解

Case＃88　開発者における仕様誤解が多すぎる

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。

【アドバイス】 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、開発着手の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点およびポイントは以下の通りです。

- まずは仕様の全体像の把握から始めること。
- 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。不明なことは直ちに分かっている人に聞くこと。
- 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。
- 仕様決定のQ＆Aは直接対話による確認を行うこと。
- 要求仕様の背景や意味を必ず明確にしておくこと。
- 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。
- 早期の仕様凍結を行い、基幹仕様未決定で開発に着手してはいけない。

One point　Lesson　：　知らない事はできない

Case＃89　要求仕様理解不足による戻り作業

ベンダー側検収テストにて、仕様が異なると指摘を受け、設計からの後戻り作業となってしまいました。当初の仕様説明にて利用された資料が、ハード側の動作も含めた全体的な資料となっており、それで仕様を理解したつもりになっていましたが、開発機能としての要求仕様を完全に把握できていませんでした。反省点として下記の二点があります。

①仕様変更の目的を確認し、把握している仕様変更内容と目的が一致しているか確認すること。
②仕様説明の際、可能であれば実機の画面を見ながら理解した内容について確認を行うこと。

【アドバイス】 最初に仕様の目的・意味・背景について確認すること

全体的な資料で説明を受けた後、仕様に関して不明点や疑問点はなかったのでしょうか。普通いろいろな不明点・疑問点が出てくるため、Q＆Aシートでやり取りをするはずですが、今回はなぜ実行しなかったのでしょうか。一見簡単に思える仕様変更においても、必ず最初に仕様変更の目的や意味・背景について必ず確認することを習慣化してください。またご提案の対応策も是非実行してください。

One point　Lesson　：　あせりは失敗を生む

Case＃90不要なテスト要求？

機能範囲外のテストで不要だと分かっていてもベンダー側担当者から要求されると心配になり、安心する為にテストをしてしまい、工数が増えてしまいます。不要なテストを要求された時に相手を裏づけのある説明で納得させられるようにスキルアップをしたいと思います。その為にはベンダー側担当者より機能について熟知している必要があると思います。

【アドバイス】 情緒性を排して積極的なコミュニケーションを

今やるべきことは評価をしようとしている仕様とその機能について完全な理解と影響度の把握をすることでしょう。そのためには仕事に着手する前に自分の不明点や疑問点を全て解消するために、相手に対して直接的な会話およびQ＆A表などを通してコミュニケーションをすることが必要です。相手が分からなければ、その相手がさらにその事を知っている有識者に聞いていただくようにお願いすることが必要です。ベンダー側担当者との話し合いが足りていません。すぐに引き下がるような受身的な姿勢ではなく、分かるまでしつこく粘るような積極性が必要です。自分が納得できないまま仕事を進めないで下さい。まずは自分が考える不要の理由を相手に言ってみて下さい。相手の意見に更に疑問があれば「何故？」の質問を自分が納得できるまで繰り返して下さい。妥当な結論を見つけられるまで話をしてください。やはりテストが必要かも知れないし、テスト範囲が少なくなるかも知れないし、不要になるかも知れません。そのような積極的なコミュニケーションと合理的な仕事のやり方を通してしかスキルアップは実現できないと強くご認識ください。

One point Lesson ： 分からないまま仕事に着手してはいけない

☑ Check List ＃19 ： **仕様理解のチェックリスト**

仕様理解に関する次のチェックリストを試してみよう

〇:できている △:どちらともいえない ×:できていない

☐ 仕様を理解しないまま作業を進めていないか。
☐ 要求仕様書における疑問点・不明点をすべて洗い出したか。
☐ すべての仕様を完全に理解しているか。

【ふり返りメモ】

6.　そろわない足並み

Case#91　顧客とのコミュニケーションを通した開発準備活動

開発日程をスムーズに運ぶために、開発の準備段階から事前検討としてソフト資産の流用調査および開発規模見積りを行っていますが、顧客とのコミュニケーションが密接にできていない状況で、一週間に一回程度となってしまっており、予想よりスケジュールの前倒しができていない状況にあります。それでも、今回取り組み始めた事前調査という作業は、ある程度進行しており、仕様の理解度も上がってきており、質疑や確認などができる状況になってきています。今後は開発に入って行くため、具体的な問題に関して顧客と密接なコミュニケーションを取って行くようにしたいと思っています。

【アドバイス】 顧客とのコミュニケーションのポイント

納期を守ると同時に品質やコストの目標を達成するためには、開発直前の段階における準備が重要な役割を果たします。事前準備の主な役割はリスクの解消にあります。開発の初期工程における前準備は、顧客とのコミュニケーションが最も必要とされるもので、要求仕様の早期凍結や開発者における開発の全体像の把握および仕様の深い理解の達成には必須の活動となります。顧客との密接なコミュニケーションを確実に実行するためには、開発準備工程および要件定義工程において定例会を設定すると同時に所定の日を決めて集中的な質疑応答を行うなどの積極的な姿勢が必要となります。

One point　Lesson　：　顧客・ベンダーとのコミュニケーションの三つのポイント
①開発の全体像の把握　②顧客要件の優先（重要度）順位の把握　③不明点・疑問点の払拭。

Case#92　開発後半に集中するベンダーからの問題通知

ベンダー側における体制が不十分な場合、各工程の遅延情報や仕様追加情報・不具合情報などの大切な情報の通知が開発後半に集中し、しばしば対応が困難になります。プロジェクトが回っていない時こそ情報共有が必要ですが、ベンダー側も対応しなければならない課題に追われているため、情報共有が疎かにならざるを得ません。これらの問題への対応として、早い段階から、ベンダーからの情報をネットミーティングや対面での打合せにて収集できるように調整し、問題点の把握・共有を早急に行い、その結果を元にお互いの課題に対する改善案の提示・認識合わせができるようにしたいと思います。

【アドバイス】 開発初期段階における積極的な行動を

スケジュールが大幅に遅延してからのアクションは後手ごてになってしまいますので、とにかく開発の初期段階において顧客との密接なコミュニケーションを保つことが最も効果的です。しかしながら現実的には、ベンダー側とのコミュニケーションを開発の前半に集中的に行うことを実際に行っているチームはほとんど見かけません。相談者においても、ご提案の対応は願望レベルのままであり、実際は実行できていない状況にあると思われます。ベンダー側との密接なコミュニケーションを本当に実現したいのであれば、要求仕様の調査段階においてベンダーとの打ち合わせを計画的に実行するよう、自分の課題管理表およびチームのプロセス管理表に記載し、自分にその行動を強制するようにしてください。いつまでも下請けの受身姿勢のままではこの問題は解決できません。積極的な思考や行動が期待されます。

One point　Lesson　：　早い段階での計画的・積極的なコミュニケーションは値千金

Case＃93　ベンダー側からの情報提供が遅い

テスト実施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。

【アドバイス】 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。

One point　Lesson　：　待っていても必要な情報は手に入らない

Case＃94　コミュニケーション不足が仕事の失敗を招いている

コミュニケーション不足によりお互いの認識にずれが発生し、その結果、仕様勘違い等があり評価等で不具合が発生しています。

【アドバイス】 良好なコミュニケーションは毎日継続して行うことから

良好なコミュニケーションは、その回数（量）およびやり方（質）の二つによって決まります。コミュニケーションは、まず伝わらないと思うべきです。一回指示すればそれで終わりというようなものではないでしょう。毎日継続的に行うことが重要なポイントです。実装方法の指示をたとえ相手が勘違いしていたとしても、翌日どういう風にプログラムしているのか一言聞けば容易に間違いが発見できます。対面形式の日次会議での作業実施内容の確認や問題の掘り起こしはこのような勘違いを防ぐ最も有効な手段です。毎日、たとえ短時間であったとしても、全員での日次会議を実行し、「①昨日実施したこと、②本日の実施予定、および③現在抱えている問題」に関して報告および助言を行うことをお勧めします。

One point　Lesson　：　毎日情報を共有していれば、突然の遅延は起こらない

Case＃95　市場障害対応情報が横展開されていない

市場障害対応の際の障害内容、対応内容、歯止め対策内容などは、客先に報告されても、社内に横展開されていないケースが多いと思います。対応に躍起になって、終わったら全て終了した気分になり、ある意味負の遺産である為、触れたくないと言うような気持ちが働いているように思います。また、なんとなく市場障害レベルのものは管理者クラスのみで共用すれば良いというような風潮が社内の一部にはあるような気がし、継承させるタイミングも見つけづらいです。社内に横展開を行わなければ類似障害が起こると思います。

【アドバイス】 障害対策実行と同時に横展開すべき

非常にもったいないことです。市場障害対応が終わったら全てが終わりではないでしょう。これでは完全鎮火とは言えません。まだ火種が残っています。そのまま情報共有をしなければ皆さんの負の資産は年々累積していき、最後には組織全体が機能不全の悲惨な状況に陥ることになるでしょう。負の遺産を正の遺産に変えるためには、その失敗に学ぶ必要があり、だからこそ失敗の事実に触れなければいけません。誰が発生させたかではなく、その問題の真因は何で、どのような再発防止策を行ったかを学ぶことです。市場障害を管理者クラスだけで共有する風潮があるなら上長にそれではいけないと意見を言って下さい。継承するタイミングは、問題の市場対応が済んだその時です。対応内容や歯止め策などはその時点で社内公開の不具合対応データーベースなどで瞬時の公開および共有をすべきです。忙しいからとか、恥だとか言う問題ではないでしょう。

One point　Lesson　：　他人の失敗を喜ぶ人は多いが、他人の失敗に学ぶ人は少ない

Case＃96　メンバーが悪い報告をしない・期限厳守の気持ちが見られない

プロジェクトメンバーから、スケジュールが間に合わないという報告を受けることがありますが、大抵の場合、こちらから確認したときの回答として報告されます。なぜ自分から報告しないのでしょうか？スケジュールが間に合わないということは、評価メンバーや後続開発に影響が発生する可能性が高い為、いち早く報告する必要があります。なぜ黙っているのか分かりません。悪い報告を言い出しづらい雰囲気を作ってしまっているのかも知れません。

また、個別のフォローだけでは対応できない場合、やむなく再スケジューリングしますが、その後のスケジュールがオンスケジュールであれば安心してペースを落としてしまいます。なぜ元のスケジュールからの遅延を最小限に抑える努力をしないのでしょうか？期限を延ばせばオンスケジュールになるのは当たり前です。ペースを落として、何か別の問題が発生した場合、更に期限を延ばしてもよいと考えているのでしょうか？各メンバーに、遅延は絶対に許されないことだという共通認識を持たせることができなかったのかも知れません。

【アドバイス】 毎日の短時間連絡会を実施すること

リーダーとメンバー間のコミュニケーションが不足しているのではないでしょうか。チーム内での会話が少なく、進捗確認もたまにしか行わない場合、悪い報告は誰しもしにくいものです。

ところであなたのチームでは短時間日次連絡会議を毎日実行していますか？この日次連絡会議で毎日チームメンバーから状況報告を受けていれば、突然スケジュールが遅れるとか、悪い報告は言い出しにくいというようなこともなくなるでしょう。

One point　Lesson　：　毎日話をしていれば隠し事はなくなる

Case＃97　雑な仕事の劣等メンバーで困っている

一部のメンバーのスキルに下記の問題があります。

①内部レビューで指摘した内容の反映もれがあります。

担当者自身における見直し方法の悪さによる見直し不足が原因で、レビュー議事録に沿って指摘された事項を見直すように指導します。なお、レビュー議事録は見直しのチェックリストにも使用するため指摘事項は漏れなく記載するようにします。また、レビュー指摘項目の反映漏れがあった場合は、その反映方法を聞き、対応や見直しに漏れの無いような方法を担当者と検討します。

②仕様確認を怠ったまま設計・コーディングを行ってしまう。

「質問をするまでも無い」と勝手に判断して作成してしまうことが原因と思われます。仕様確認を行う意識を高めるために、設計およびコーディングのレビュー時に仕様確認したかどうかを担当者に確認し、仕様確認漏れを防ぎたいと思います。

③コーディングミスが多い。

担当者自身が、見直しをしていないことおよびデバッグをしていないことが原因と思われますので、デバッグを必須とさせます。

【アドバイス】 全員参加の短時間日次会議の実行を

これは雑な仕事の典型的なもので、担当者が仕事に集中していないか、そもそもやるべきことが何かを認識していないかのいずれかでしょう。

①および②の対策について、実行しようと思った日にリーダー自身が忙しくてついつい先延ばしになってできなかった場合にどうしますか。 また、③について、デバッグをしろと命令した後、1 回や 2 回はやったとしても、この担当者はまた実行しなくなる可能性が大きいと思います。ずっと継続して実行させるにはどうしたら良いと思いますか。

これらの問題に限らず多くの問題は、チーム内でのコミュニケーション不足によって引き起こされているとは思いませんか。例えば、毎日 20 分程度の短時間日次会議を実行して、その中でチームメンバー全員から、「昨日何を実行したか。明日何を実行する予定か。今抱えている問題は何か。」の３点について報告・応答を続ければ、「内部レビューで指摘した内容が全て反映されていない。仕様確認を怠ったまま設計／コーディングを行ってしまう。コーディングミスが多い＝見直しをしていない、デバッグをしていない。」の問題は毎日ベースで発見でき、毎日ベースで部下の教育・しつけもできるということになります。毎日ベースのこのショートミーティングを是非実行して下さい。

One point　Lesson　：　小さい事を積み重ねていけば大きな成果が得られる（積小為大）

Case＃98　誤った自己判断で不具合を出してしまう部下

新しい部下が以下の問題を抱えています。

・ベンダー側担当者に確認せず、自分の判断で作業を進め、結果不具合が発生する。

・不明点を質問せず、長時間自分で考え、無駄な時間が発生する。

対策としては、本人に自分の問題点を認識させ、改善を促すこと、毎日のメールによる作業報告の実行、ミーティング開催、ソースコードのレビュー、などにより問題点・不明点を把握し、解決案を指示したいと思います。

【アドバイス】 リーダーとの毎日ベースのコミュニケーションを

自己判断で誤った作業を進める部下は、ベンダー側担当者に確認する前にリーダーであるあなたに相談するべきでしょう。相談者においては普段からもっと部下とのコミュニケーションを深め、部下が勝手な自己判断で作業をしていないかチェックする必要があります。

また、長時間自分で考えても答えを見つけられないでいる部下は何故リーダーのあなたに相談に来ないのでしょうか。リーダーは毎日、部下の状況を把握しておく必要があります。

本件の問題の真因は、部下自身の問題ではなく、相談者におけるリーダーとしての振舞い方の問題だと言えます。

短時間日次進捗会議で直接顔を合わせて会話をし、彼らの問題に対して適切なアドバイスを毎日行えばご相談の問題は全く発生しなくなるでしょう。

One point　Lesson　：　リーダーの行動を変えれば部下も変わる

Case＃99　情報の個人的な抱え込みが情報共有を妨げている

電話確認した内容などは担当者同士しか理解できてなく、議事録にも残りません。そのために情報が個人にしか残らずに問題発生時などに、他の担当者が状況を把握できないことや、やりとりの確認が残っておらず記憶しかないために議論ができないなどの弊害をもたらしています。電話内容の議事録のルール化（メールによる伝達や仕様書への記載など）が必要だと思いますが、ルールを決めても守られない懸念があります。

【アドバイス】 重要事項は必ず文書に残し、直ちに全員に伝えること

個人が持っている情報、特に仕様や機能に関する情報はメールで他のメンバーに通知するだけではなく、日次会議で全メンバーに報告する必要があります。さらにこれらの情報は、プロジェクト毎に統一的な仕様確認Q＆Aフォームでの管理をする必要があります。仕様や機能に関する情報は単なるメールや電話・口頭だけでの仕様確認は厳禁です。やむをえない場合でも改めてQ＆Aフォームにて合意確認をして下さい。

One point　Lesson　：　文書に残さない情報はすぐに消えてしまう

Case＃100　店舗運用方法や他メーカ機器の仕様情報の継承を行っていない

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継承を行っていません。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。継承しなかった結果は、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。

【アドバイス】 すぐに情報共有を行ってください

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してください。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。

One point　Lesson　：　過剰な自己防衛は百害有って一利なし

Case＃101　対人関係に弱く、チーム内の連携ができない

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。

【アドバイス】 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。

One point　Lesson　：　ネガティブ思考を生かし目的本位の行動を

☑ Check List　＃20　：　**情報共有のチェックリスト**

情報共有に関する次のチェックリストを試してみよう
〇:できている　△:どちらともいえない　×:できていない

☐　要求者との直接かつ密接なコミュニケーションを行っているか。
☐　要件定義工程において仕様の疑問点・不明点をすべて解消しているか。
☐　ドキュメントに基づいた情報共有を行っているか。
☐　相手からの情報提供に対して待ちの姿勢になってはいないか。
☐　チーム内での日次ベースの情報共有会議を行っているか。
☐　重要障害情報は社内で共有されているか。
☐　仕様情報は社内で共有されているか。
☐　技術ノウハウ情報は社内で共有されているか。
☐　客先システム運用情報は社内で共有されているか。

【ふり返りメモ】

7.　分かりあうことは難しい

Case＃102　結合テストがうまくいかない

機能を結合したときに、データフローやインターフェース的に不整合が発生して、そもそも動作しないレベルの不具合が結合試験で多数発生しました。原因は、各担当者が、互いにインターフェースの確認をせずに設計・実装していることにありました。担当者間・機能間での連携を意識して設計するよう指示し、開発開始時に全員に対して全体像と互いの連携度合いなどをリーダー格の人が説明するようにしました。その結果、同じレベルの不具合は出なくなっており、一定の効果は出ていると思います。

【アドバイス】 結合テスト不良は、自分と相手のコミュニケーション不良に起因する

ソフトウェアのインターフェース不良の問題は、担当者同士の人間的なインターフェースの不良、すなわちコミュニケーションの不良に起因しています。この問題は仕事における仲間同士のコミュニケーションや情報共有が徹底していないチームにおいて必ず発生するものです。ハードに例えるなら、口径5mmのナットとビスを作らなければならないのに、ナットの作成者は6mmのナットを作り、ビスの担当者は4mm のビスをそれぞれ勝手に作っているようなものです。それぞれの担当の頭の中には、ただナットとビスを作ることしかなく、何 mm でなければ用を足さないのかという基本的な要件についての話合いや了解が抜け落ちています。とりあえず作ってみようという様な安易な考え方では、まともな結合は無理です。

複数の担当者で組織的な開発を行う場合には、常にリーダーが指導力を発揮し全員の情報共有を継続的に行わなければならないという強い意志が必要です。誰がリーダーだか分からないような集団は必ず失敗するでしょう。2003 年にこれと同様な問題をN社が大規模システムのオフショア開発で発生させ、結局国内で全て作りなおして20億円もの欠損を出したプロジェクトは記憶に新しいところです。

One point Lesson ： インターフェースとは相互理解ということ

Case＃103　他人に興味がない

相手を信用・信頼するためには、相手がどのような人が知り、自分がどのような人かを知ってもらうことが必要だと思います。相手のことを知らなければ、相手の能力も分からず、相手を信用・信頼することもできません。しかしながら自分は性格的に他人とのつきあいが苦手で、他人に対しての興味もわかないため、職場での信用・信頼関係も希薄です。

【アドバイス】 過剰な自己防衛の姿勢を改め、先に他人のことを理解すること

自分のことを理解して欲しければ、まず先に相手のことを知る必要があります。人はみな自分のことを理解して欲しいと思っています。そのような状況の中で、先に自分のことをいくら述べてみたところで、ほとんどの人は聞く耳を持たないのは当たり前のことです。まず先に相手のことを聞くことから始めなければ、だれもあなたの言うことに耳を傾けてはくれません。

他人の目や評判が怖いというのは、集団主義的な日本の組織においてよくみられる心理的な傾向です。その結果がもたらす過剰な自己防衛の姿勢が、他人に興味をもたないという心理的な態度を生んでいるのかも知れません。このような人が多くいると、人々はお互いに孤立化し、意思疎通も行われなくなり、連携・連帯もチームプレーも行われなくなり、仕事はことごとく失敗するという悲惨な結果となってしまいます。 過剰な自己防衛的姿勢から自分を解放するためには、ものごとに感情的・情緒的にいちいち反応することをやめ、仕事本位・目的本位に業務に集中することが必要でしょう。気分や感情はいったん脇に置いておき、なすべき仕事の目標に向かって平常心をもって進むことがよい結果を生むでしょう。

One point　Lesson　：　まずは他人の言うことに耳を傾けてみる

Case＃104　期待はずれの結果

プロジェクトメンバーまたは顧客に対して、こちらが期待する結果の報告をしてもらうよう依頼をしたはずなのに、「できていない」や「期待はずれの結果」を途中報告も無しに結果として報告されると、出来ていない部分の穴埋め作業が追加で発生してしまい、信頼や期待への裏切りと感じ、モチベーションが下がってしまいます。

また逆に、自分の仕事の結果が、自分における評価より他人から見た評価が低かった時は、その差が大きいほどモチベーションが低くなります。意欲的に取り組んだつもりの時ほど、相手の評価が低いと自信がなくなります。

【アドバイス】 相手の力量を見極め正確なコミュニケーションを実行すること

期待外れや期待に応えられなかったという問題の多くは、依頼者・被依頼者両方に問題があります。まず依頼した方の問題としては、相手の力量や負荷状況の判断の誤りや依頼や指示方法のまずさなどがあるでしょう。頼んだのにちゃんとやってくれない結果になってしまう原因の大部分は頼む側の問題と言っても過言ではないでしょう。次に依頼を受ける側の問題としては、依頼者が期待する内容をきちんと確認しておかなかったということです。実行すべき内容は何か、その数や量や品質レベルの数値はどうか、それらをいつまで完成させるのか、複数の課題ならその重要性の順番は何か、依頼された仕事の意味や背景は何か、等についてきっちりと確認をとっておく必要があります。

両者共通の問題は、口頭のみあるいは不十分な資料のみで仕事の依頼や受託が行われた場合にはこのような問題が発生しやすいということです。そんなはずではなかったという結果にならないように、仕事を依頼する方は相手の力量や負荷状況を判断すると同時に、十分な説明や背景・条件などを資料に基づいて説明し、受ける側においては必ず不明点・理解不十分な点・条件等について遠慮なく質問をすることが必要です。硬直的で一方的な上下関係だけの相互関係ではこれらのコミュニケーションは行われにくく、日常的にコミュニケーションがスムーズに行われる雰囲気や習慣を作り出しておく努力が必要です。

One point　Lesson　：　一喜一憂している内はまだ半人前

Case＃105　視野の狭さや経験不足が信頼関係の構築の障害になっている

自分の視野が狭いことや経験不足のために相手が何を求めているのか察知できていないことが、相手との信頼関係を築く障害になっています。信用を得るために、まず相手が何を求めているのかを明確に知り、約束事を守ることから始めたいと思います。具体的には、①相手の要望を察知した上で約束事を明確にする　②業務プロセスに従って都度報告連絡相談を行い相手に安心感を与える　③約束した品質、納期、リスクを守る　④相手が困っていることを察知し約束以上のものをする、などを実行したいと思います。

【アドバイス】 相手の要望の明確化には察知力よりドキュメント力を

今後実行したいと提示されている四項目の実行は簡単なことではありません。これらの四項目の実行に成功している人は非常に少ないのが現実です。これらのことを実行するためにはもっと具体的な行動目標の設定が必要でしょう。

①については、相手の要望をどのように察知するのでしょうか。その要望に合理性や妥当性があるのか否か、その要望を実現するために自分に必要なことは何か、などを検討しなければいけません。その上で自分もその内容に納得できなければ、そもそもお互いの約束は成立しないでしょう。いい例として、いつまでも決まらない要件定義・要求仕様があります。相手の要望を全て書面に書き出すことから始めなければ、要求内容における合理性・妥当性の有無さえ明確に検討することはできません。ソフトウェア開発において、何事かを”明確”にすることは、その事を文書化・ドキュメント化することでしか実現できないということを肝に銘じておく必要があります。口頭だけのコミュニケーションで仕事の内容が明確化できるほど我々の仕事は簡単な仕事ではないでしょう。社内・社外を問わず、仕事の約束事に関する事項は、白板メモでも議事録でもその場で両者が確認できる物で確認しなければいけません。仕事は全てドキュメント・ベースで実行する以外に”確かな”方法はありません。

②についても同様です。業務プロセスを”明確”にした”プロセス管理表”というドキュメントに従って仕事を遂行していますか。そうでなければ報告・連絡・相談も単なる思い付きや気分によって適時性も守られず適当な思いつきでしか行われないでしょう。

③についても同様です。仕事を開始する前に品質・納期・コストを書面にて表し、両者・両社にて確認・合意しておく必要があります。これが”確かな”約束というものです。

④約束した以上のものを実行するのは、約束したものを完全に達成できるようになってから考えればよいことです。

”明確”とか”正確”とか”確実”とか呼ぶにふさわしいものは、現在の我々においては、過不足ない事実を表す”ドキュメント”以外にないことを肝に銘じて行動すべきでしょう。

One point　Lesson　：　有用なドキュメントは皆に富をもたらす

Case＃106　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

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。

【アドバイス】 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てください。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。小職も新人時代に、先輩に不明点を聞きにいった時、今忙しいので話をしている暇なんかないと怒られたことがありましたが、この件については先輩がエキスパートだとみんなが言っており、先輩に話を聞きに来たのに何故怒られなければいけないのですかと聞き返しました。それに対して先輩は”エキスパート”という言葉に喜んだかどうかは分かりませんが、ニッコリ笑って残業時間に教えていただくことになりました。

One point　Lesson　：　人が動いてくれるような頼み方をすること

Case＃107新メンバーの育成に悩んでいる

新メンバーを指導していた人からの報告では、約1ヶ月前からプロジェクトに入った協力会社の人が、問題が発生しても報告してこないというより、聞かないと報告しないか、ぎりぎりに報告して来るようです。また、一つずつ細かく指示を出し、説明をしないと作業ができないとのことです。 また、その新メンバーに質問しても返答が返ってこないことがあり、そのときは同じ会社の人が代わりに返答してしまっており、周囲もそれに慣れてしまっていたようです。

問題の発覚が遅れた原因としては、新メンバーと指導役の人に、問題点などを個別にヒアリングしていなかったことや、新メンバーに発言の機会を十分与えなかったこともあるのではと思います。 今後の対策としては、毎日のミーティングにおいて、新メンバーの作業状況と問題点は、新メンバー自身に報告させるようにします。またその他の問題がないか見極めるために当分の間、その新人を注視したいと思います。

【アドバイス】 日次短時間会議に参加させ訓練を

上記のご相談内容から最初に感じられることは、協力会社から派遣された要員たちとの直接的なコミュニケーションが不足しているということです。協力会社に限らず自社のメンバー間における直接的なコミュニケーションも不足しているのではないかと懸念されます。

新メンバーの方は社会経験に乏しい新人であり、さらに協力会社からの派遣要員であることを考慮すれば上記の問題はやむをえない一面もあるかと思います。その人においても経験もないのに即戦力を要求する周囲の目には耐え難いものがあるかと思われます。貴社においても面接の上、採用したのですから、リーダーにおいては、この人との直接的な会話を増やし、開発者としての基本的な行動姿勢を教育されるようにお願いしたいと思います。

日次短時間の進捗会議は新人教育の場としても有効だと思いますので、この新メンバーも是非参加させてください。毎日、①昨日やったこと、②今日やること、③今抱えている問題、の三つを報告させることによって基本的な開発者としての行動スタイルが身に付いてくるでしょう。そのためには退社前、ないしは出社直後に、これらの報告内容を自分の手帳などに簡潔にまとめておくような指導もされてはいかがでしょうか。

One point Lesson ： 新人の育成はチーム全員で

Case＃108　対人関係力・コミュニケーション力の継承は難しい

継承が難しいものとして、客先・社内の対人関係を察知する能力、時間・予算の管理、コミュニケーション能力などがあります。これらのことについては、話をしたりアドバイスをしたりしますが、自分も上手く伝えられていなかったり、本人が納得しているのか分からなかったりしています。

ここ最近は、プロジェクトリーダーの立場で動ける人材は増えてきていると思いますが、マネジメントの立場で動ける人材が育っておりません。マネジメントできるようになれば、強み／弱み、得手／不得手の不均衡を解消し、今より高い能力を発揮できると思います。

【アドバイス】 妥当性の力と合理性の力を成長させること

対人関係能力、コミュニケーション能力などは形に表しにくく自己評価も難しいものですが、これらの能力を伸ばす基本的な力は、妥当性の力と合理性の力の二つをいかにバランス良く成長させることができるかにかかっています。妥当性とは、簡単に言えば道理にかなうと同時に実用的である考え方や行動のことです。一方、合理性とは、科学的な思考に基づく分析力、判断力、実行力のことです。これらの力を成長させるためには自分を孤立させず、道徳的な常識をわきまえ、科学的な学習に励み、集団との連携の中で切磋琢磨を行い、仲間との信頼関係を築き、自分に固執せず柔軟性を発揮し、弱き者を助け、もてるノウハウを広く他に継承していくような行動が必要だと思われます。

マネジメントができるようになるから、問題点を解消し高い能力を発揮できるのではなく、困難な問題点を解消するなどの能力を発揮できる人だけがプロジェクトマネージャになれるのです。

One point　Lesson　：　経験の積み重ねと学習が対人関係力を強化する

Case＃109　阿吽の呼吸でやったはずが、抜けが出てしまう

基本的な開発作業として、目的の把握、不明確部分の明確化および、文書・コード作成・調査などの実作業がありますが、チームメンバー間は阿吽の呼吸で進むことが多い為、依頼する側もされる側も、そこそこ同じ仕事を続けていると、それで何とかなるケースが多いと思います。しかしその結果、手順を踏めば抜けないような内容が抜けてしまうことがあり、失敗や不具合の原因となってしまいます。

【アドバイス】 ソフトウェア開発は厳格な論理性の追及に拠ること

阿吽の呼吸が通用する世界は単純作業の世界や情緒性や芸術性が重要視される世界だけでしょう。阿吽とは、言わなくても分かるでしょうと言う意味ですが、我々のソフトウェア開発の仕事は、言わなければ分からない世界です。そこまで言わなくても分かって欲しいとか行間を読んで欲しいと言うような日本人特有の情緒的なやり方で、我々はどれ程多くの失敗をしてきたことでしょうか。

ロジックの仕事を行う限りにおいては、仕事における阿吽や情緒性は障害につながります。人間的な交流と仕事における厳格性をはっきり区別する必要があります。開発業務においては、過不足のない論理性に貫かれた仕様書や管理表などのドキュメント・ベースに基づいた仕事のやり方を基本にしてください。阿吽のやり方はいつしか情緒的な仕事になり”テキトー”な仕事になってしまいます。

One point　Lesson　：　阿吽で通じるほどソフト開発は甘くない

Case＃110　評価メンバーへの教育がうまくいかない

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、評価メンバーへの教育がうまくいきません。まず、今の現場でメンバー間のコミュニケーションを取り、いろいろ評価手法などの話をすることで、自分も含めスキルアップしていこうと思います。

【アドバイス】 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っていただきたいと思います。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。

【感情本位から目的本位への転換のポイント】

- 自分の関心を自分自身から仕事そのものに移すこと。顧客の要求するものは何かということを良く理解し、その実現に向けて情熱を注ぐこと。
- たとえ嫌いな相手であっても仕事に必要なことは必ず実行すること。
- 人の好き嫌いで自分の行動を変えないこと。誰に対しても一貫した姿勢と行動を保つこと。
- 自分ためにという過度な自己中心性を捨て、顧客要求の実現に向けてチームやメンバーのために貢献すること。

自分を害するものを減らしたければ、これらのことを実行する必要があるでしょう。

One point　Lesson　：　人の好き嫌いは仲間を減らし世間を狭くする

☑ Check List　＃21　：　**意思疎通のチェックリスト**

☐　相互理解に努めているか。
☐　他人の言うことに耳を傾けているか。
☐　相手の力量を見極めたコミュニケーションを行っているか。
☐　ドキュメントを利用したコミュニケーションを行っているか。
☐　人が動いてくれるようなコミュニケーションを行っているか。
☐　全員参加のコミュニケーションを行っているか。
☐　毎日ベースのコミュニケーションを行っているか。

【ふり返りメモ】

☑ Check List　＃22　：　**感情本位のチェックリスト**

☐　人の好き嫌いで仕事をしてはいないか。
☐　好き嫌いで仕事を選んではいないか。
☐　目的に沿った仕事の遂行ができているか。

【ふり返りメモ】

8.　進んでいるはずが遅れていた

Case＃111　順調だと思っていた進捗が実際は停滞していた

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。

【アドバイス】 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日次の進捗会において、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・指示を実行しなければいけないでしょう。

One point　Lesson　：　毎日の5分の会話が致命傷を防ぐ

Case＃112　メンバーの増加に伴い、全体進捗の把握が難しくなっている

現在は一つの物件が結合テスト段階にあり、次の物件が仕様検討の段階にあり、メンバーの増強を行いながら両物件に対応していますが、全体の進捗を把握するのが難しくなってきました。 体制の問題としては、チームメンバーは社内の各事業所からの応援者も多く統制が困難であること、およびPMは他の案件を抱えているために専任できない状況にあります。

結合テストについては各開発作業およびテスト環境構築が遅延しており、先が見通せない状況に陥っています。各メンバーは毎日、進捗報告書を更新する運用となっていますが、機能および工程単位となっており全体進捗が把握できていません。

【アドバイス】 仕様の再整理、優先順位ごとの開発、および視覚的・量的把握が可能な進捗管理を

結局、プロジェクトの進捗遅れ自体が問題で、さらに進捗管理表が正しい進捗状況を把握するのに役立っていないと言うことでしょうか。短納期にあせって、とりあえず作れる所から作っていこうとするサミダレ的進め方が災いとなっています。ともかく作るということを１～２日ストップしてでも全体の仕様を再整理し、全ての不明点・疑問点を解消した上で、最重要な仕様から優先的に開発を進めていくようにすべきだと思います。また進捗管理は、まずは量的な把握ができる管理表でないと意味がありません。現在使用しているのがジグザク線のガントチャートだけなら正確な量的把握はできません。各機能、各モジュール、部品単位ごとに予想ステップ数を設定しておき、それに対して何%進捗したかとか、デバッグ終了とか、単体テスト確認終了とかの工程毎の進捗が分かる表で管理しながら、さらに数日おきに担当者にヒアリングを繰り返し、実態を把握することが一番です。 作業場所が離れている人には電話ヒアリングも止むを得ませんが、週に一度は直接コミュニケーションをとってください。 PMは何故このプロジェクトを見られないくらいに多忙なのでしょうか。もしめんどうがみられないのなら他のPMなりのアサインを上司に依頼する必要があるでしょう。

One point　Lesson　：　進捗管理は単品管理で

Case＃113　顧客用スケジュール表と内部スケジュール表

開発スケジュール表には、顧客提出用と内部スケジュールの二種類が存在することがあります。今回は内部スケジュール表の説明をしてもらっていなかったため顧客提出用のスケジュール表だけしかないものと思い、それに基づいてテスト計画のスケジューリングを行いましたが、内部スケジュール表が実際はあったことが後になって分かり、テスト計画のやり直しが必要になってしまいました。このような大事な情報についてはキックオフの時点からの共有が必須だと思います。

【アドバイス】 開発と評価チームの連携を

開発と評価チームの連携やコミュニケーションがうまく行ってないようです。スケジュールは仕様と並んで最重要なプロジェクト要件です。基本的には顧客提出用と内部用のスケジュールは基本的な工程については一致させておく必要がありますが、細部については顧客提出用と内部用と異なったスケジュール表が存在することはよくあることです。そのような場合、開発チームは評価チームに対して、その違いについて説明する義務があり、評価チームは開発チームに説明を求める必要があります。評価グループは開発グループと密着して妥当なスケジュールは何かを共有しておく必要があります。開発リーダーと評価リーダーは頻繁に会話・文書両方でのコミュニケーションを毎日実行する必要があります。

One point　Lesson　：　二重帳簿は間違いの元

☑ Check List　＃23　：　**進捗管理のチェックリスト**

☐　毎日ベースでメンバーごとの進捗管理を行っているか。
☐　単品ベースでソフトウェア成果物の進捗管理を行っているか。
☐　基本的なマイルストンが異なった複数のスケジュール表が存在してはいないか。

【ふり返りメモ】

9.　ずれているレビューの視点

Case#114　詳細にわたるレビューができない

担当者の思い込みミスを排除するために、詳細かつ完璧な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。

【アドバイス】 レビューは重要機能順および失敗しやすいポイントに従って行う

限られた時間内で全てのミスを発見することは不可能であり非合理的です。レビューの本来の目的は、“基本的なミスを発見すること”にあり、細かな問題の発見は、デバッグ・単体テスト・結合テストおよび総合テストなどの一連のテストの役割です。

またレビューの精度は、ドキュメントの精度に依存します。過不足のない要求仕様書・設計書等のドキュメントに基づいたレビューは基本的なミスの発見に必須のものとなります。

レビューのポイントは下記の通りです。

【レビューのポイント】

- レビューは重要機能順（顧客価値の順）に行うこと。
- レビューを有効に実行できるレベルの機能仕様書や設計書を用意しておくこと。例えば処理パターンが複雑な機能においては、機能を箇条書きにしただけのものでは不足でマトリクス表で表現したものにするとか、長々とした日本語文章表現ではなく数字やロジック表現を多用し誰が読んでも誤解や思い込みの余地のない資料でレビューを実行することなどです。
- 過去の類似の機能開発での失敗例を参考にして類似のミスをしていないかのレビューを行う。

One point　Lesson　：　できない完璧さを目指すよりも有効な妥当さを目指すこと

Case#115　顧客レビューにて否定的な意見しかもらえない

レビューをしてもこちら側の説明に対して否定的な意見ばかりで、前向きな説明をしたくなくなる雰囲気になってしまいます。レビュー内容について相手に反論させないくらいまで調べられたら良いのですが、調査する期間も、製品に対する知識もまだまだ少ないのでなかなか実行できません。

【アドバイス】 相手を説得するには相手以上の知識・情報が必要

これについては相手の反論に対して説得できない自分の能力の問題なのではないでしょうか。相手の反論が正当か誤っているのか分からない程度なら相手を説得することはできません。今後、あきらめることなくしっかりと継続して勉強し知識を増やすしか解決策はないでしょう。

One point　Lesson　：　論争に勝つには相手を上回る力量が必要

Case＃116　レビュー対象が膨大でレビューし切れない

レビュー対象物が膨大なためレビューに必要な時間が足りません。膨大なレビュー対象物が一目で分かるような資料を作成することや、中間レビューを実施することで、効率的なレビューができるようにしたいと思います。

【アドバイス】 設計レビューのポイントは基本的な理解のチェックにある

設計レビューといっても全ての設計内容についてレビューすることは限られた時間の中では不可能なことです。設計レビューの本来的な意義は、要求仕様の全体像との整合性があるか、基本仕様の理解に間違いがないか、などを確認していく作業です。設計レビューにおけるポイントは次のようになります。

【設計レビューのポイント】

前提として、設計は重要な仕様順に実行されること。

● 事前に自己レビューを行うこと

・詳細にわたるレビューは自己レビューの段階で実施しておき、単純なミスはこの段階で全て排除しておくこと。

・関連する過去の設計ミス、頻度の高い設計ミスなどを網羅した自己レビュー用の基本的なチェックシートを用意しておくこと。

● 中間レビューと本レビューの二段階レビューを行うこと

中間レビューにおいては重要仕様の理解が正しいか、およびその内容が過不足なく設計に反映されているかどうかをチェックすること。本レビューにおいては重要仕様についての最終チェックを行うと同時に、時間の許す限りその他の仕様のチェックを行うこと。

● レビュー内容のポイント

・不明仕様を想定して設計している部分はないか。想定部は速やかに確定させること。

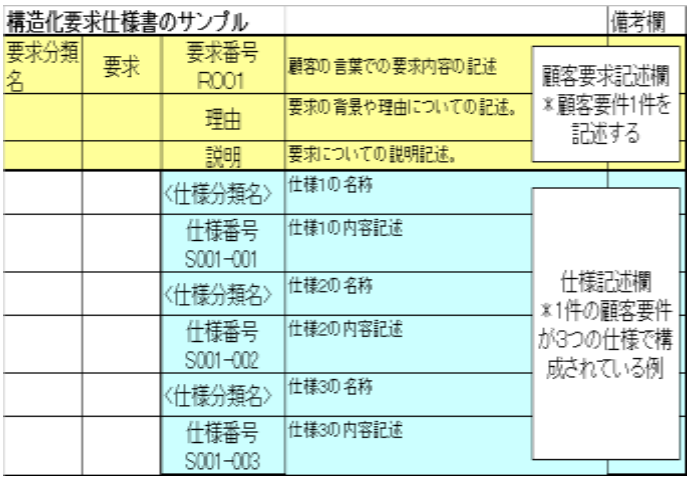
・要求仕様の意味を正しく理解しているか。

・入力条件は正しいか、処理のロジックは正しいか、出力は正しいか。

・異常処理は適切か。

・レスポンス・パフォーマンスの考慮は適切か。

One point　Lesson　：　中間レビューは効果的に働く



[構造化要求仕様書]

Case＃117　レビュー等で初歩的な不具合を発見できない

単体テストレベルで発見すべき不具合が結合テスト以降で検出される場合が非常に多いと思います。このことに関連して設計・製造に関するベンダー側レビューでの指摘事項も多い状況です。各工程におけるレビューを実施しているのにもかかわらず、なぜ開発の早い段階のレビューや単体テストで多くの不具合の発見ができないのか分かりません。

設計レビュー、コードレビュー、評価設計レビューをどのようなやり方にすれば不具合を減らせるのでしょうか。また担当者の成果物に対する精査も行っていますが、どうしても指摘漏れが発生してしまいます。複数人でレビューをする、静的解析を行う等、いくつかの方法が考えられますが、今後、検討していきたいと思います。

【アドバイス】 過不足のない要求仕様書・設計書に基づいたドキュメント・ベースの開発を

まずレビューについてですが、全項目のレビューを実行することは実際問題として不可能です。複雑なロジックの関連性や影響度まで考慮してレビューを行なうための膨大な資料の用意や全てをチェックする時間的な余裕はないでしょう。レビューの目的は仕様の必須要件および基本的な考え方に誤りがないかをチェックするだけに絞り込むべきでしょう。レビューに個々の不具合の発見を期待すべきではないし、無理なことだと思います。レビューで不具合が発見されることもありますが、それはラッキーだった位の認識で良いと思います。

設計・製造のベンダー側レビューでの問題の指摘が多い原因は、要求仕様の内容を十分に把握しないまま設計・製造を行なっているからでしょう。最大の原因はベンダー側におけるあいまいな要求仕様および仕様凍結の遅さ、および下請け側においては仕様の不明点・疑問点についての明確化努力の不足にあります。そのような状況下で作成された設計書は欠陥だらけで指摘件数も多くなります。この問題を解決するためにはベンダー側と下請け側の双方が協力して仕様の早期凍結に努力をする必要があります。

次に、単体テストレベルで検出されるべき不具合が結合テスト以降で発見される理由は、単体テストが十分でなかったということでしょう。これは単体テストのチェックリストの不完全さや、単体テストのチェックリストの元である詳細設計書の不備不足誤記、さらにはその元である基本仕様書および要求仕様書の不備不足誤記などの複合的な要因があるからでしょう。

まずは発生した不具合や指摘された内容の要因分析を実行して下さい。不具合発生や指摘事項の原因をグルーピングしてフィッシュボーンチャートにして、どの要因が多いのかの見極めができれば有効な対策が可能です。

要件開発リスク管理MAP
作成更新日 XXXX年XX月XX日

要求番号	要求内容	顧客重要度	技術難易度	仕様凍結度	仕様凍結目標期日	リリース	未凍結仕様問題記述
1	要件概要記述 顧客要求内容の概略ポイントを記述する。	MUST	中	80%	2010/10/1	1	・未仕様凍結の理由および内容の記述 どのような要求内容が決まっていないのか理由も含めて記述する。 ・仕様凍結目標期日までに間に合いそうか、そうでないか等対策も含めて記述する。
2	要件概要記述	MUST	高	50%	2010/10/1	1	・顧客重要度が高い項目で仕様凍結度の低いものは要リスクである。これ以上仕様凍結度が進捗しないのなら、顧客との話し合いにて重要度を下げ2次リリース等に移動するアクションが必要。
3	要件概要記述	MUST	中	60%	2010/10/1	1	・顧客重要度が高い項目で仕様凍結度の低いものはリスクである。リスクの度合いによる色分けを行う。黄色信号は危険・至急アクション要、黄色は要注意・アクション要の項目である。
4	要件概要記述	NEED	低	90%	2010/11/1	1	
5	要件概要記述	NEED	中	30%	2010/12/1	2	・顧客重要度および仕様凍結度の状況につき顧客との話し合いにて、2次のリリース項目に決定した。
6	要件概要記述	WANT	低	20%		ドロップ	・顧客との打合せにて要件項目からドロップされた。

[要件リスク管理表]

One point　Lesson　：　不具合の多くの原因は低品質なドキュメントに在る

☑ Check List ＃24 ： レビューの基本

☐ 重要機能順（顧客価値の順）にレビューを行っているか。
☐ 仕様および機能の目的・意味・背景を正しく理解しているか。
☐ 必要な仕様書や設計書および資料を用意してレビューを行っているか。
☐ 入力条件は正しいか、処理ロジックは正しいか、出力は正しいか。
☐ パフォーマンス・レスポンスの考慮は適切か。
☐ メモリーやCPU速度等のハードウェア条件の考慮は適切か。
☐ 想定仕様によって開発をしている部分はないか。
☐ 過去の類似の失敗例を参考にしてレビューを行っているか。

【ふり返りメモ】

☑ Check List ＃25 効果的なレビューのポイント

☐ レビューの時間を計画的に確保しておくこと。
☐ レビューは対象物の全件チェックではなく基本部のチェックに絞り込むこと。
☐ レビュー対象物の一覧表を作成すること。
☐ 過去のミス・不具合の傾向に基づいてレビューの重点を絞っておくこと。
☐ 基幹機能に関する基本的な理解が正しいか。
☐ 基幹機能の実現方法は適切か。
☐ 中間レビューを行うこと。
☐ 共通的なレビュー項目のひな型（チェックリスト）を作成しておくこと。
☐ 仕様変更の影響部分のレビューも行うこと。
☐ 担当者は、事前に自己チェックを済ませておくこと。
☐ レビュワーは、担当者が見落としがちな異常系、過去の評価モレ等の広い視点を持つこと。

【ふり返りメモ】

10.　ルール違反のコミュニケーション

Case＃118　ダラダラと続く雑談を止められない

チーム内での、良い意味でのコミュニケーションではなく、メンバー間のなんとなくダラダラと続いていく雑談はまったくの時間の無駄で仕事の生産性を落とす原因となっています。その場の雰囲気を壊さず、他の人たちに嫌味を感じさせない方法でダラダラ雑談にストップをかける方法が分かりません。

【アドバイス】 ダラダラ雑談には終了宣言を

目くじらを立てて「いつまで雑談ばかりしているのですか」などと言っても逆恨みを買うだけでしょう。そこはポーカーフェイスで「そろそろ本題の話に戻していただけませんか」で話を切り出し、本題についての自分の意見を述べてみるのも良いかも知れません。

雑談の効用は、その場の雰囲気をなごませたり、話し手の本音を知ることができたり、また有用な情報を得ることができたりなどにありますが、ダラダラと続く雑談は単なる井戸端会議に過ぎない場合がほとんどでしょう。誰も主導権を発揮しない会議や、論点のはっきりしない会議などでもよく見受けられる日本人特有の光景です。誰も何も決めない会議のことを小田原評定といいます。会議が雑談に流れ、何も決められない会議になってしまう訳は、出席者が個々に明確な主張を持って参加していないことにあります。根本的な原因は、自律性や当事者意識の欠落にあるでしょう。すなわち問題を解決しようという意思も弱く、その問題の当事者であるという意識も希薄な人間たちが何人集まっても実のある結論は何も導き出せないということです。

One point　Lesson　：　井戸端会議は時間ドロボーと思うべし

Case＃119　コミュニケーションのルールが守られていない

テスト障害記録票は評価チームが起草し開発チーム宛に発行されますが、その後の修正処理は開発チームと製造チームだけにて行われ評価チームへのフィードバックが行われないことがあります。修正内容は、開発・製造・評価の三者間で共有されなければ、その後の評価業務において修正の影響評価をすることができません。

【アドバイス】 チーム間のコミュニケーションはルールに従うこと

評価チームに対する障害票への回答は開発グループの義務です。プロジェクトのルールで決められているなら例外なく守られなければいけません。開発チームは期日までに回答できないのならば、回答できる日を評価グループに知らせる義務があります。また評価グループは、期日少し前に開発グループに予定日に回答をもらえるか確認することが必要です。これは仕事における”相互義務”の一つです。一旦ルールが破られればプロジェクトはどんどん破綻する方向に向かって進みます。そのほかのプロジェクトのルールで守られていないものがないのか再点検を行ってください。

One point　Lesson　：　常識もルール化しなければ守られない時代になってしまった

Case＃120　会社間のコミュニケーションは伝わりにくい

急遽対応しなければならない案件について、関連会社とスケジュールのやり取りを行った際、納品日を初めは木曜日で合意していた内容が状況の変化により水曜日、最終的には火曜日になりました。しかし担当者の方に確認したところ、伝わっておらず再度確認し、最終的には火曜日で納品していただきました。納品日の変更連絡を複数の人間で行ったことで、社内の連携も取れておらず、会社間の確認も怠っていた為発生した問題です。

【アドバイス】 会社間の重要事項に関するコミュニケーションはリーダー対リーダーで

会社間の約束ごとの変更に関するコミュニケーション、特に納期や仕様に関する変更については、基本的にそれぞれの会社のそのプロジェクトの責任者間で実行しなければいけないでしょう。自社のリーダーと他社のリーダー間でやるべきであり、複数の人間間でやるべきではありません。重要事項変更に関する組織間の接点は責任リーダー同士の一点でのコミュニケーションを文書ベース（メールも可）にて行うことが大原則です。コミュニケーションの相手を間違えると本件のように納品日が二転三転することや、内容の理解に差が出たりします。コミュニケーションの相手を間違えないようにしましょう。やむを得ず担当者間で行った重要な合意事項については速やかにリーダーに対面および文書にて報告しなければいけません。

One point　Lesson　：　組織間のコミュニケーションは責任者同士で

Case＃121　ビジネスマナーの継承ができていない

部下に対して社会人としての行動、礼儀などのビジネスマナーの指導ができていないなと思います。他人事ではなく自分自身もできているかどうかと思う部分が多々あります。なかなか指導できない理由としては、そのようなことは言うまでもなく「当たり前だろう」と思うことや、私自身に自信がないことがあります。 このようなことで部下にビジネスマナーが身についていないので、客先レビューの時などお客様と対面する場で、問われた時に、「無言になる」、「わからないのに、適当なことを言おうとする」、「声が小さく、何を言っているのかわからない」、などこんな当たり前のことができなくて上司として恥をかくことがあります。

【アドバイス】 コミュニケーションに関する基礎的な振舞いを身につけること

相談内容はビジネスマナーというよりか、コミュニケーションに関する基礎的な振舞いができていないということです。相手の質問に対して無言であったり、適当なことを言ったりすることなど社会人以前の問題があるようです。相手の問いかけに正しく応答できるようにするためには普段から次のような行動に心がける必要があります。

【顧客とのコミュニケーションのポイント】

①事前準備を行っておくこと。話題や議題が決まっているような会議においては、事前に聞かれそうな内容について事前検討を行い、答えを用意しておくこと。客先レビューならば、当然要求仕様の理解についてチェックされますので、仕様の理解内容およびその実現方法について説明をすれば良いだけの話しです。部下がちゃんと応答できないのは、上長が部下に開発仕様の目的・意図・背景などの骨子を説明していないことや、社内での事前レビューを行っていないからでしょう。出たとこ勝負で、部下に丸投げでは自分が恥をかいても当然でしょう。原因は部下にあるのではなく、上司のあなたにあるのですから。

②発表力やプレゼンテーション力を強化するために、普段の開発業務の中で全員に発表やプレゼンの機会を与えること。 公式の場、特に客先との会議の場に慣れていない人、特に若手の人にとっては上手に話をすることは非常に難しいことです。緊張する場面での質問には上手に答えられないでしょう。普段の業務活動において、社内レビュー時、社内会議、社内報告会、社内発表会、日次会議などを利用し、チームメンバーの全員が議長や発表者になる機会を設ければ、客先とのレビュー会議などにおいて物怖じすることもなくなるでしょう。社内でのコミュニケーションが不活発であるのに客先とのコミュニケーションがうまくいく道理はありません。

One point　Lesson　：　ビジネスマナーの習得には、OJTによる訓練と事前の準備で

☑ Check List　＃26　：　**コミュニケーション・ルール違反のチェックリスト**

☐　ダラダラ続く井戸端会議に忠告を与えているか。
☐　伝えるべき部署に情報を伝えているか。
☐　組織間・会社間のコミュニケーションは責任者同士で行っているか。
☐　ビジネスマナー等の躾は日常業務の中で訓練されているか。

【ふり返りメモ】

11. 意味のない会議

Case＃122 何も発言しない人の会議への参加は無駄

何の発言もしない人が打ち合わせに参加しており無駄を感じます。直接的に関わらない人を会議に参加させても有益ではなく、自分含めその人の生産性低下につながります。会議は本当に必要な者のみ人選して行うべきだと思います。

【アドバイス】 必要のない人まで会議に召集してはいけない

まったくご指摘の通りです。出席の必要がない人たちまでも召集する会議は、皆さんの貴重な時間を大規模に失わせてしまう時間どろぼうの元凶です。全員招集の会議においては、全員と共有する議題を先に行い、個別の問題はそのあとで議論すべきでしょう。 招集者は、用の済んだ出席者には退席指示を都度出すようにすべきです。全く用事のない人を召集することや、いつまでも拘束しておく意味はないでしょう。このような会議に遭遇したら、その場で会議召集者か議長に要望事項として伝えられるようにお願いします。

One point Lesson ： 保険のかけすぎは組織の体力を消耗させる

Case＃123 結論を出さない、出せない会議が多い

会議の進行・内容に疑問があります。だらだらと想いを話すだけで結論が出ないことが多いと感じます。またやたらと会議時間が長く、議長も誰だか分からず、事前にアジェンダの通知もありません。何を決めるべきか、何について議論するのか事前に決まっていれば議事もスムーズに進行し無駄な時間は省略できるはずだと思います。

【アドバイス】 何も決めない・決められない会議には、その場でクレームを

井戸端会議的な会議はやめるべきです。何も決めない、決められない会議のことを”小田原評定”といいます。会議の議長はその会議を招集した人のはずです。招集者は事前に出席者に会議の目的、背景、討議問題内容について知らせておくべきでしょう。また会議においてはその目的に従って、誰が・何を・いつまでに実行するのかを決めるべきでしょう。いっぽう出席者は事前に自分なりの方針を決めた上で出席すべきです。また討議内容や結論については当日中に議事録を関係者全員に配布すべきでしょう。そのようにやっていない会議に遭遇したら、その場で招集者に対して上記クレームを発言してください。

短時間で結論を出すためには、野球で言えば4回裏、延長戦もありというようなだらけた気持ちではなく、9回裏2死満塁の打席に立った気持ちで緊張感をもった会議の運用が必要であり、従来のやり方を改め会議の時間と回数、参加人数もそれぞれ半分にするくらいの取り組みも必要でしょう。

One point Lesson ： 事前の準備なしには何事もうまくは行かない

☑ Check List　＃27　：　**欠陥会議のチェックリスト**

☐　会議の目的・背景・討議内容を事前に出席者に伝えているか。
☐　何を討議するのか事前に決めているか。
☐　何を決めるのか事前に明確にしているか。
☐　誰が・何を・いつまでに実行するのかを決めるようにしているか。
☐　不必要な人を招集してはいないか。
☐　ダラダラ雑談に終了宣言を行うことができるか。

【ふり返りメモ】

☑ Check List　＃28　：　**顧客との会議のポイント**

☐　開発仕様の全体像の把握、仕様の目的・意味・背景を理解しておくこと。
☐　事前準備・事前検討を行い、顧客からの想定質問への答えを用意しておくこと。
☐　顧客からの宿題事項は、状況を聞かれる前に答えること。
☐　自分の言いたいことよりも顧客の話を良く聞くこと。
☐　疑問点・不明点については臆することなく、その場で確認を取ること。
☐　リーダークラスにおいては、発表力やプレゼンテーション力を強化するために、普段の開発業務の中で発表やプレゼンの機会を自ら求めること。
☐　リーダークラスにおいては、議事進行や調整能力を磨くために、社内会議等にて議長役や書記役を自ら買って出ること。

【ふり返りメモ】

☑ Check List　＃29　：　**ドキュメント・ベース業務の遂行**

☐　顧客会議／連絡における依頼・指示・決定内容は議事録にて管理されているか。
☐　関係各社との会議／連絡における依頼・決定内容は議事録等にて管理されているか。
☐　社内会議／連絡における依頼・決定内容は議事録等にて管理されているか。
☐　顧客・ベンダーからの仕様変更は、仕様変更管理表にて管理されているか。
☐　依頼・指示・決定内容等は、「誰が、何を、いつまでに、どのように」が明記されているか。

【ふり返りメモ】

12.　他人に教えたくない

Case＃124　自分の知識を伝達せず、他のメンバーの作業効率を落とした

今までに仕事をする上で覚えた知識について文書化していないために他のメンバーに継承できていません。自分が知っている事を伝達しなかった為に、他のメンバーが作業に時間がかかったこともありました。

【アドバイス】 有用情報はすぐに伝えること

日々の開発作業上、自分が気づいたことで仲間のメンバーにも必須の情報や知識はたくさんあると思われます。そのような情報については、文書化などと大上段に構えずに、自分の手帳に書きとめておき、日々実行されている短時間日次進捗会などで他のメンバーにもすぐに伝えるようにしてください。このようなことをプロジェクトの全員が小まめに実行すればプロジェクトの業務品質や効率性の向上に必ず貢献することになります。このような有用な開発情報や知識を、その都度、プロジェクトの「技術情報掲示板」などにみなさん全員でこまめに登録するような活動を実行されることを期待します。

One point　Lesson　：　こまめな知識の集積が全体の力となる

Case＃125　他人へのシステム知識の継承が十分ではない

担当しているシステムの知識を十分に継承していません。聞かれたら教えますが、自分から自発的に改まって教えるような場は設けていません。心の中では、譲ることで他の人に仕事を奪われてしまうことを怖れており、また自分を頼りにされていたいという気持ちがあります。その結果、自分の作業比重が重くなったり、自分が困ったときに内容の深い相談にのってもらえる相手ができなかったり、レビューで指摘事項が出て来ず隠れた問題が発見されないまま市場に出てしまう危険がある、などの弊害があると思います。

【アドバイス】 自己防衛的姿勢からの脱却を

自分の優位性を保ちたいという守りの姿勢になると自分のノウハウを人に教えたくなくなるものです。継続的なスキルアップを怠ったために自分が持っているノウハウが少ない人においては特に自己防衛的になるものですが、常に新しいノウハウの習得に努力している人においては既に他人に継承したノウハウの次のステップに進んでいるため、他人へのノウハウの継承に何らの抵抗感も持たないものです。そのような人が本当の優秀な技術者でありリーダーとしての資格を持っている人と言えます。自己研鑽に励まれて、他人へのノウハウの継承を惜しみなく実行して下さい。

One point　Lesson　：　他人への譲りは、いつか自分への譲りとなり戻ってくる

Case＃126　細部のノウハウ情報の継承ができていない

製品に実装される前に自分で行った調査で得たファームウェア開発の細部の情報については、他のメンバーへほとんど継承できていません。継承しても、その後使われる事がないと判断した場合や、コンプライアンス的に継承出来ない特殊仕様の場合もあります。 しかしながら、継承していない情報があると、問い合わせが自分に集中するので困る時もあり、極力、メモや説明ドキュメントを残すようにはしていますが、完全ではないので、より多くを残すように努力したいと思います。

【アドバイス】 設計・製造のツボ的なノウハウはドキュメントによる継承が必要

細かい部分に関しても設計・製造のいわゆる”ツボ”的なノウハウはドキュメント化による継承を行う必要があります。またコンプライアンス的なノウハウに関しても自社組織内においてはノウハウの継承は必要であり、極秘的なノウハウに関しては信用できるメンバー間のみで情報共有するような注意が必要です。

One point　Lesson　：　ノウハウの継承は負荷の分散と組織力の強化の結果を生む

Case＃127　自作公開した用語集やマニュアルが余り利用されていない

自分が調査した用語、マニュアル等を記載した EXCEL ファイル（ノウハウ集）について仲間のメンバーにも公開しています。これらの情報が、渡した相手（後輩）の作業効率アップのために有効に活用されることを期待すると同時に、他のメンバーによって追加された情報がまた自分のところに戻り、全体としての有用な情報が蓄積されると思います。後輩が、こういうのが必要だと感じていましたと言ってくれたので、継承することを考える点でのきっかけにはなったと感じました。しかし現実的には、この資料を複数人に渡しましたが、今現在活用されていないだろうなと感じています。

【アドバイス】 ドキュメントの鮮度を保つこと、用語の意味の統一を図ること

用語集やマニュアル類などのドキュメントは生鮮品と同じで、その鮮度、すなわち最新情報が網羅されているかどうかが命です。常に更新されていなければいつの間にか使用されなくなります。社内の共有ドキュメントについても同様のことが言え、常に内容が更新され進化するように、多くの社員の参加によって育てていくようにすれば、多くの人にとって貴重な資料となります。

また用語集に関して言えば、業界標準である専門用語についてはすでに多くの書籍やネット検索を利用できますが、狭い業界での独自の専門用語については、自分たちの手で作成する必要があります。このような独自の専門用語については定義があいまいになりがちで、一つの会社内においてすら異なった意味に使用される場合も多いと思われます。我々の仕事は言葉によって遂行されていますので、同じ言葉が異なった意味で使用される場合があれば、それは仕様の誤解やコミュニケーションの誤解につながってしまいます。実際、このような言葉の定義のあいまいさが原因で引き起こされる失敗や不具合は結構多いと思われます。そのような不要な失敗を防ぐためには、自社および顧客内で使用されている専門用語については用語集としてまとめておき、その意味を統一する必要があります。またこれらの用語集は新入社員や新たに参加するメンバーたちにとっても組織における共通のコミュニケーションの手段を提供する重要な資料となります。

One point　Lesson　：　統一用語集は失敗や不具合を減らす

Case＃128　開発スキルの継承ができていない

開発スキルについては自分で考えたり調べたりする中でしか身に付かないと言う思いがあったため、自分自身の開発スキルすなわち、調査の方法や自己レビューの方法等については、メンバーへの継承が余りできていないと思います。開発スキルの継承が余りできていませんので、チーム全体としては調査不足や成果物の不備が無くならず、内部／外部のレビューでの指摘や不具合が減らない状態です。最近は、開発スキルについてもある程度は教える事で継承していくべきなのかなと考えています。

【アドバイス】 あらゆる知的資産はどんどんと後輩に継承、公開すること

開発スキルでも仕様ノウハウでも何でもあらゆる知的資産はどんどんと後輩に継承して下さい。後輩に自分で考えさせることは必要ですが、考える材料がなければ毎回先輩に聞くしか方法がありません。そのような徒弟制度的なやり方では現在の仕事のスピードにはとうてい追いつけません。

先輩としてまずやるべきことは開発スキルに関するノウハウを皆から集め蓄積しデータベースやドキュメントとして社内公開し常時メンテしておくことです。後輩たちはこのような各種のノウハウデータベースをまず見て自分で調査し、それでも分からなければ先輩に質問をするというような方法が継続的な効率性およびノウハウの継承を可能にするでしょう。

One point　Lesson　：　ノウハウ検索データーベースの価値は高い

Case＃129　実務スキルの上手な教え方が分からない

実務でのスキルについて、継承したいと考えていますが、なかなか継承することが出来ていません。教えなければいけないことを、相手が理解出来るまで上手く伝えられていません。

自分自身は、伝えることが苦手で、教え方が下手なので、一方的なやり方の説明になってしまうことが多々あります。実際、評価設計の要因表作成では、内部レビューにて毎回同じ指摘をしてしまうことが多いので困っています。この困っていることを解決していくにあたり、上手く伝わるように相手が理解できるように教えられるようになるには、教える側は、どの様に、伝えていくことが良いのでしょうか？

【アドバイス】 ドキュメントに語らせるようにすること

資料が貧弱で口頭に頼る説明ではうまくいかないと思ってください。相手が分かるように説明するには、まず書いたものすなわち要領よくまとめられたドキュメントを用意する必要があります。実務スキルの説明に必要なドキュメントはすでに実務の中で作成したものを使用すべきです。新たに教育用として作成する時間はないでしょう。それらのドキュメントは誰が読んでも誤解を生まないような分かりやすいものである必要があります。説明文は、最初にその目的・意味・背景の説明に始まり、可能な限り論理記号や数値を多用し、図・表・フローチャートなど視覚的な表現を使用することが必要です。他人になった積りで、自分で読んでみて疑問や不明点が出ず分かり易ければ合格でしょう。自分の口ではなく、いわばドキュメントに語らせるようにすれば楽に相手に説明できるようになります。

One point　Lesson　：　視覚的資料に基づく説明は口頭説明に優る

Case＃130　ラップアップの継続的な実行は難しい

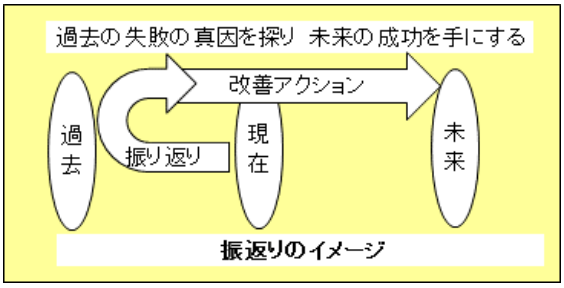
過去のプロジェクトでは、リーダーが声をかけないとラップアップが開催されない場合がほとんどでした。現在でも実行されないことがあり、継続的なラップアップの実施は難しいと思います。リーダーに言われなくても、メンバーが自発的に取り組んで欲しいと思います。

【アドバイス】 ラップアップはQCD数値に意味を見出し、リーダー主導にて実行すること

ラップアップはプロジェクトを振り返る重要な最後のけじめであると同時に、次なる仕事へのノウハウの継承という重要な役割を担う行為です。この仕事はリーダーの重要な仕事の一つであり、メンバーの自主性に任せておくような性質のものではありません。リーダー主導で必ず実行してください。またラップアップは業務遂行中に採取し分析を済ませておいたQCDの指標数値に基づいた振り返りの実行および次なるプロジェクトへの申し渡し事項をまとめるようにする必要があります。単に、〇〇の失敗をしないように“がんばろー”と言うような情緒的な会議にすることだけは避けるようにしてください。

One point　Lesson　：　“がんばろう”だけでは頑張れない

Case＃131　振り返り会議を行う習慣がない



新規プラットフォームの標準製品を開発しましたが、仕様書不備により問い合わせが頻繁に発生し、問い合わせの対応のために時間がかかり本来の作業に支障が生じました。開発中における個人レベルの振り返りや設計・製造・評価工程におけるレビューも不十分でした。更に問題なのはプロジェクトとしての最後の振り返り会議も行なわれず、再発防止策や問題点の共有はされていません。

振り返り会議は規模の大きいプロジェクトでしか行われていません。部署の体質として規模の小さいプロジェクトで反省会を行う習慣がありませんので、今後は規模の小さいプロジェクトであっても振り返り会議を行うべきだと思います。

【アドバイス】 レビュー、振り返り会議の絶対励行を

レビューも振り返り会議も実行されなければ、失敗の経験に学ぶ機会が全くないということになります。そのようなプロジェクトは規模の大小を問わず、手抜きプロジェクトと呼んでも良いでしょう。

日常的にこのようなやり方を続けていれば品質も利益も悪化し、人材も全く成長しません。ご自分が一担当者の立場であったとしてもプロジェクトのリーダーおよびマネージャに問題提起をする必要があります。次回からは必ず、自己レビューや社内レビューおよび振り返り会議を開発の規模にかかわらず実行してください。

「振り返り」を行っていない組織は「改善活動」もほとんど実行していないと思われ、それでは技術者個人のスキル向上も、組織力の向上も図れず、毎年バグに追い回されるだけの利益も上がらない集団のままでしょう。

One point　Lesson　：　賢者は失敗に学ぶ

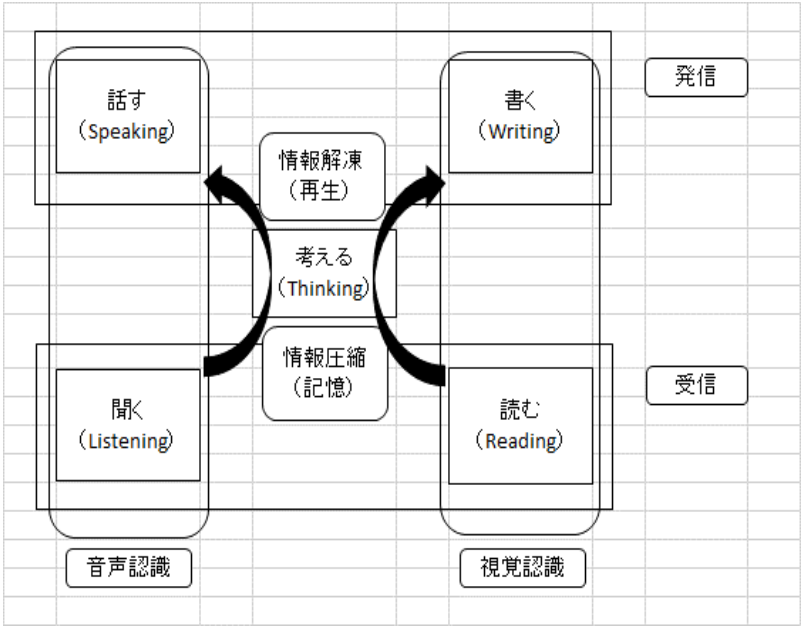
☑ Check List　＃30　：　**ノウハウ継承のチェックリスト**

☐　自分が獲得したノウハウをドキュメントに残しているか。
☐　自分のノウハウを仲間に継承しているか。
☐　チームが獲得したノウハウをドキュメントに残しているか。
☐　チームのノウハウを組織的に継承しているか。
☐　統一専門用語集は作成されているか。
☐　ノウハウ集などの知的資産がデーターベースとして集積され、常時検索可能か。
☐　プロジェクトにおけるラップアップは実行されているか。

【ふり返りメモ】

おわりに

コミュニケーション能力の構造を情報の圧縮・解凍の視点から見ると次の図のようになります。



「聞く」情報は圧縮され脳内に記憶され、また脳によって解凍され「話す」情報として再生されます。同様に、文字や図は「読む（見る）」ことによって脳内に記憶され、また脳によって解凍され「書く（描く）」情報として再生されます。

このことでも分かるように「話す」前の段階に「聞く」ことがあり、「書く」前の段階に「読む」ことがあるのです。 多くの人たちは、早く上手に話せるようになりたい、上手に文章が書けるようになりたいと願って話し方や書き方にばかり気を奪われていますが、本当に上手になりたいのなら、人の話を良く聞き、多くの書物を読むことに注力した方が良いと思われます。

コミュニケーション能力のレベル差は、量的には、どれだけ人の話をたくさん聞いたか、どれだけたくさんの書物を読んだかによって生じ、また質的には、聞いた情報・読んだ情報の圧縮率（記憶率）および解凍率（再生率）に依存しているのでしょう。

圧縮率や解凍率は頭脳の良し悪しにも依存しているでしょうが、人の興味の度合いや感動の度合いが最も強く影響していると思われます。われわれ凡人においても興味や感動は天才・秀才たちに劣らないものを持っているはずで、良く聞くこと、良く読むことから始めたいと思います。

引用

イラスト： いらすとや http://www.irasutoya.com/