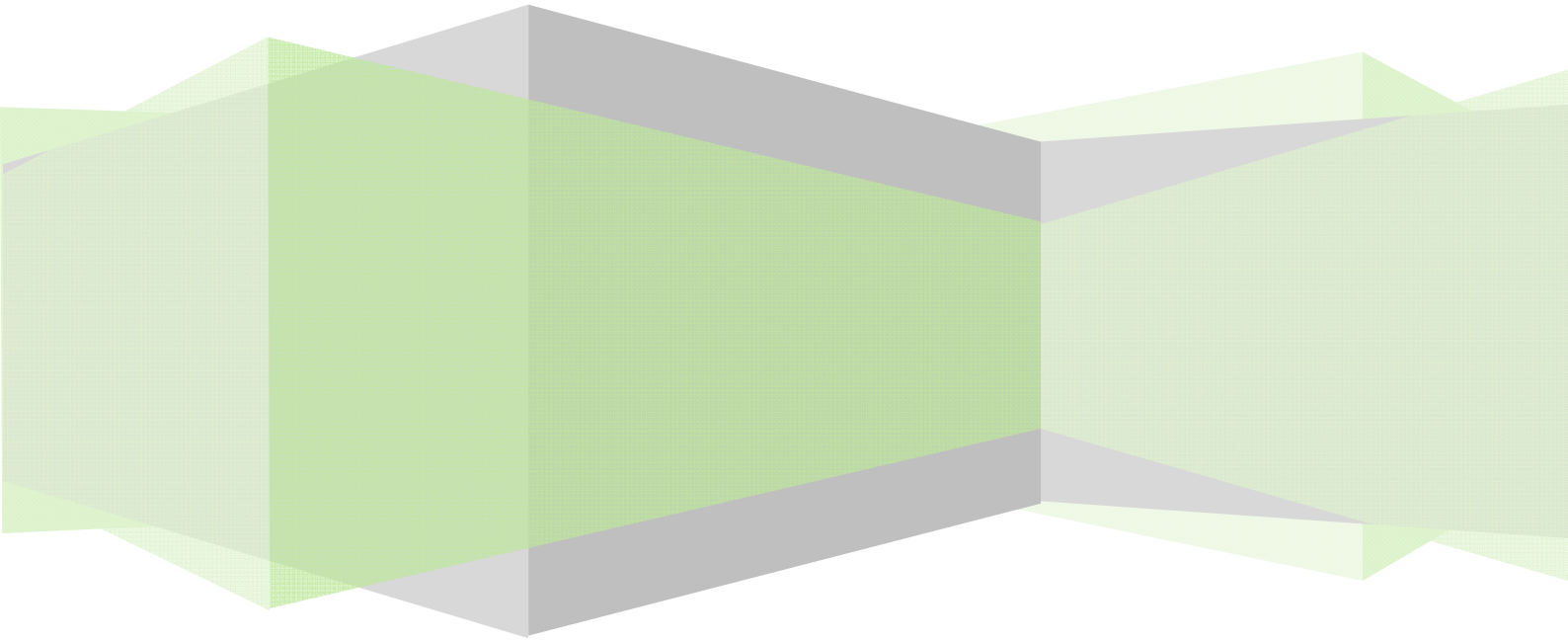
経営者のための IT 講座

今更聞けない IT 使いこなし術

エムズスタイル　前田稔



著作権について

『経営者のための IT 講座』（以下、本レポート）の著作権を含む知的財産権は全てエムズスタイルが保有します。
よって、本レポートの一部または全部を無断で複製、改変、配布、掲示、表示、アップロードすることはできません。
商用目的以外の私的使用に限って、閲覧、ダウンロード、印刷することができます。また、本マニュアルの全部または一部をコピーして販売するなどの営業行為も固く禁じます。

注意事項

以下の点にご注意ください。

（責任の所在）
『本レポート』は、様々なノウハウを公開していますが、これを利用して何らかの問題や損害が生じたとしても、当社は一切責任を負いません。すべては、あなたの自己責任においておこなうものとします。

（秘密保持）
『本レポート』の内容を一般公開することはできません。また、転売も禁止されています。

初めに

会社にも一人に１台のパソコンは当たり前の時代ですね。というかそれすらも言われなくなって久しいです。

家に帰っても、パソコンがあります。さらにインターネットのおかげ様々な情報も瞬時に手に入れることが出来ます。携帯のメール機能は当たり前ですし、スマホに至ってはもう従前のＰＣ同等の機能が使えてしまう時代です。

コンピュータとネットワーク化により、私たちの暮らしぶりは一変しました。もうコンピュータ無しの生活、仕事はあり得ないのではないでしょうか？

当然ですがこれがビジネスに関係ないわけありません。どんな会社でもITは不可欠な物となっています。ビジネスモデルを作るにしてもIT抜きでは語れません。しかし、本当にITに対して理解をしている経営者がどのくらいいるでしょうか？

「IT」といった瞬間に、「自分は専門家ではない」と逃げる方も大勢いらっしゃいます。確かに、「餅は餅屋」「専門家に聞け」とも言われています。しかし、それがどれだけ自分に影響を及ぼすのか？それを考えると、無頓着ではいられないはずです。

「専門家に任せておけばいい。トラブルが起きるのは全部そいつらの責任だ。」と考える方も多いと思います。果たしてそうなのでしょうか？では、その方に質問します。「いくらコストをITにかけるべきなのでしょうか？」これは経営的な問題です。わからないですまされる問題でしょうか？

例えば財務会計。これを知らずにビジネスをする経営者はいませんよね？
（たまにいるようですが・・・）

現在のIT技術、環境は非常に高スピードで発展してきています。しかしながら、経営にたずさわる方々がついて行っていないのが現状なのです。それがいろいろな問題を引き起こしています。現場の問題と思いがちですが、そのきっかけを作っているのが自分であることに気が付かないケースが多いのです。

この文書は、今までシステムとは縁遠い方をイメージして書いています。システム書というといきなり難しい文言が多く記述されます。学ぶべき物はシステム技術ではなく、システムという概念です。ですので、できるだけわかりやすい例を示しながら説明していきます。

私のこと【自己紹介】

私とITの出会いは大学時代にまでさかのぼります。

大学入学のお祝いとしてNECのPC8001を買ってもらいましたのがきっかけです。

最初は雑誌に載っていたプログラムソースを手打ちしていたのですが､ひょんなところから独自にプログラムを書くことに｡そこからプログラミングを覚え､あっという間にゲームプログラマーになりました｡（日本テレネットという会社にいました｡当時まだスクウェア社も小さかった頃です｡）

そこで徹底したマシン語開発を行い､（言うと笑われるのですが､本当に）コンピュータの気持ちまでわかるレベルまで達してしまいました｡マシンサイクルを暗算し､レジスターの動き､割り込み処理をすべて頭でシミュレートまでしていたわけです｡

当然､ハードウェアに至るまでいじりました｡I/O設計なども当時の大学研究室では必要だったのでやっています｡

パソコンがどうして起動するのか？BIOSはどう動くのか？キーボードを押すとどうして画面に文字が表示されるのか？そのレベルまで理解しています｡

そのせいか今ではプログラムはしませんが､バグを見ただけでその SE のプログラミングが分かってしまいます｡

つまり､**IT に関しては普通の SE より深いレベルで理解しています**｡

その後､野村総合研究所へ入所｡金融数理学から始まり､トレーディングシステム（意思決定支援システム）の設計､大手金融機関へのシステム・コンサルまで実施しています｡
投資運用会社､大手都銀､信託銀行における次期システムのシステム化計画をリーダとして推進してきました｡米国大手金融機関の IT､CIO などの調査も行なっています｡

野村総合研究所退所後は SBI ホールディングスにてグループ全体的な企画提案などをして来ました｡

私の専門分野は業務面と IT 面の両側面＝ビジネスを実現するための IT 戦略です｡

様々な会社を見て感じたのが「あまりにも IT を理解していない経営者が多い」ということ。これがこのレポートを書くことを思い立った理由でもあります。

このレポートが何かしらの皆様のお役に立てたら幸いです。

エムズスタイル　前田　稔

目次

IT システムとは何か？

この章では、ITシステムとは何か？どうやって動くのか？
どういうところが得意で､どういうところが苦手なのか？をわかりやすくお話したいと思います。

IT システムとは何か？
「おれはシステムの専門家じゃないから別に知らなくてもいい。餅は餅屋。システムは専門家に任せておけばいい！」

まさか、こういって逃げている経営者はいらっしゃらないでしょうね？
今の時代、そういう方は残念ながら経営者失格です。

ITシステムは現在のビジネスにおいて読み書きそろばんと同じくらい重要です。

さすがに電話、ファックス無しで仕事をしている方はいませんよね？
それくらいITは仕事に不可欠なものになっていると思います。

会社のIT戦略はイコールビジネス戦略に等しいです。
今や、経営者にとってITシステムは「財務」と同等の地位にあるのではないでしょうか？
（Financial Technology、Information Technology、Marketting Technologyの３つが必須と思われます。）

ITシステム化の遅れ､セキュリティの対応の不備などから会社の屋台骨を揺るがしかねないのが現状だからです。

「でも、システムはどうもわからない・・・」

そういう気持ちはわかります。確かにわかりにくいです。
でも、ご安心ください。本来ITシステムはそんなに難しいものではないのです。

まず、英語でシステムという言葉を調べてみましょう。
【名-1】 系統｛けいとう｝、体系｛たいけい｝、システム、系
【名-2】 流儀｛りゅうぎ｝、整然｛せいぜん｝とした手順｛てじゅん｝、方式｛ほうしき｝
【名-3】 組織｛そしき｝、体制｛たいせい｝、制度社会｛せいど しゃかい｝の権力｛けんりょく｝
（Space ALCより）

整然とした手順、方式というのがもっとも近いと思います。

つまり、システムとは業務フローをあるルール、手順に従って処理させることを言うわけです。
「体系化」と言っても良いでしょう。
手段であるコンピュータのことは指していませんね。

そうなのです。
業務フローを理路整然とすべての例外系も含めて「体系」化すること。これがシステム化となります。ある意味、「業務手順書」もそのままシステムですね。

一方でITは・・・
IT＝Information Technology　情報技術、情報処理技術

つまりITシステムで「情報処理技術を使った体系」と言う意味になります。
コンピュータ・システムとも言いますね。

こう考えるとわかると思いますが、「システム」の方が広い考え方をしています。手段を限定したものが、「ITシステム」「コンピュータ・システム」となるわけですね。

コンピュータは優秀ですが、いかんせん融通はまったくききません。

つまり、**想定されていないことは何一つやってくれません**。気を利かせるなんてとんでもない。
こいつは手続きの化け物です。

「コンピュータ・システム」の対局にあるのが、「人間システム」です。
これは融通が利きます。人によりますけどね（笑）

機転も利きます。（これも人によりますが。）

少なくとも柔軟な対応ができるわけです。

ただし、コンピュータ・システムの利点は決められたことは、正しく、高速に処理できることです。コピーとかはお手の物。これはすごい。人間がやっていたのでは、間違いだらけです。

「システム」の根底にあるのは「業務処理フロー」です。
何をどう処理するのか？

これがわかればシステム化できるということです。
そこで、さらにコンピュータの得意な「定型処理」「大量処理」を切り出した部分をITを使って実現したものがITシステムとなるわけです。

ここまで書くとわかると思いますが、根底にあるのは**「業務処理をどうするかということ」**です。

まずはやりたいことを明確化し､業務フローを確定化してからシステム構築しなければならないことが重要だとわかると思います。システム・コンサル＝業務コンサルであるゆえんです。

そしてもう一つ大事なのが**「何が何でもすべてIT化するのが正しいというわけではない」**ということです。
先ほど、書きましたように、システムは「体系」、手段は別です。
（融通の効く）人間処理もうまく組み合わせるのが良いということです。

なぜ IT システムはわかりにくい？

ここでわかりにくいのは**「IT」**だと思います。

そう、確かにIT関連用語はわかりにくい点は多いです。

無数のカタカナ（OS、TCP/IP、http、html、xml、JAVA・・・・）なんのことやらわかりません。
アルファベット3文字とかの省略語も多いですね。

そもそも現在のコンピュータ・システムの基本アーキテクチャーはノイマン型。メモリ領域に処理手続きを乗せ、それにしたがって順次処理させる方式です。

その手続きがプログラム。
その手続き記述する手段が言語になります。

そもそも、コンピュータの世界は２進数です。１と０だけで表現され、これだけで全てを処理します。
１が電圧Onの状態、０がOffの状態。だから電気的に処理しやすいわけです。
１と０の組み合わせで１ビット。これが６４セットあるのが６４ビットということです。２の６４乗までの数値表現が可能です。

データの転送と演算、条件判断。これだけで全て処理されているのです。
ここまでの話、**「計算機」**としては理解できると思います。

でも、目の前の「インターネット・サービス」と上記の「計算機」は結びつかないでしょうね・・・

イメージ的にはあるデータを処理した結果によって何か処理を変更する。。。これを繰り返しているだけです。

例えば、取得した情報（数値）が〇〇だったら、この場所にデータ〇〇を出力する。という具合です。

この１，０の数字の世界ではわからないので、より人にわかりやすい方法をとっていきます。ユーザ・インターフェイスの向上です。

入力としてキーボード、マウス、タッチパネル。出力はディスプレイ。
キーボードは「このキーが押されたら、こういうデータが入力されたことになります。」ということです。

で、ディスプレイへの表示は「ある場所（ビデオRAM）にデータを書き込むと、ハードウェアが処理して画面に文字がでます。」というようになるわけです。
（最近はWindowsなどがあるので、もっと複雑ですが・・・）

プログラム言語も１，０だけで表記されたマシン語、それをニモニック表記（数字の羅列を簡略記号化したもの）したアセンブリ言語と進化していきます。

ハードウェアの処理もパッケージ化します。

例えば、ハードディスクと情報のやり取りをしたいのであれば、命令セット＋パラメータで処理できるようにする、などです。
別のハードウェアを使っても、同じ命令セットで動くようにしておけば入れ替えが可能ですよね？汎用性が増します。
これがBIOS（Basic Input Output System）

言語もアセブリ言語ではわかりにくいので、もっと普通の言葉に近いものが開発されます。
C言語、Basic、JAVA・・・・。いろいろあるわけです。

ソフト的に基本操作が必要となってきます。
操作システム＝Operation SystemとしてOSが出てくるわけですね。

で、ユーザ・インターフェイスの向上のため、Windowsとなってきているわけです。

さらにその上で動くミドルウェア。いろんなものがどんどん乗ってきます。
より高度化し、抽象化されてきます。

このように１，０の世界から人間がわかりやすい、使いやすいように、そして汎用性を高めるにいろんな「決め事」をするわけです。

それが、様々な専門用語で表現されています。
その専門用語が余りにも多いため、逆にわかりにくくなっているわけです。

ちょっと数学とかに似ていませんか？数学のΣ記号とか∫記号とかです。

でも、ご安心ください。

これらはIT用語＝情報技術用語です。細かいことはこれこそ専門家に任せておけばいい話です。
全ては「より簡単に」「より汎用的に」処理するために出てきた技術です。

あなたがすべきことは**「何を目的として」「何をどうしたいのか？」「何をどう処理するのか？」**という業務目的と業務処理フローを明確にすることです。
どんな新しい技術が出たとしても、根本的なところは何一つ変わらないわけです。

コンピュータはなぜ動く？
この章は軽く読み飛ばしてください。
感覚的に理解してもらえば十分です。細かい部分まで知っている必要は無いです。

コンピュータは１，０の２進数で動いています・・・。
で、やっていることは演算と条件判断だけです。
なのに、どうしていろんなことができるのでしょう？

たぶん、このギャップがすごく大きく感じられると思います。
目の前のパソコンを見ても、１，０で動いているとは実感できませんよね？

ではゲームを例にとって考えてみたいと思います。

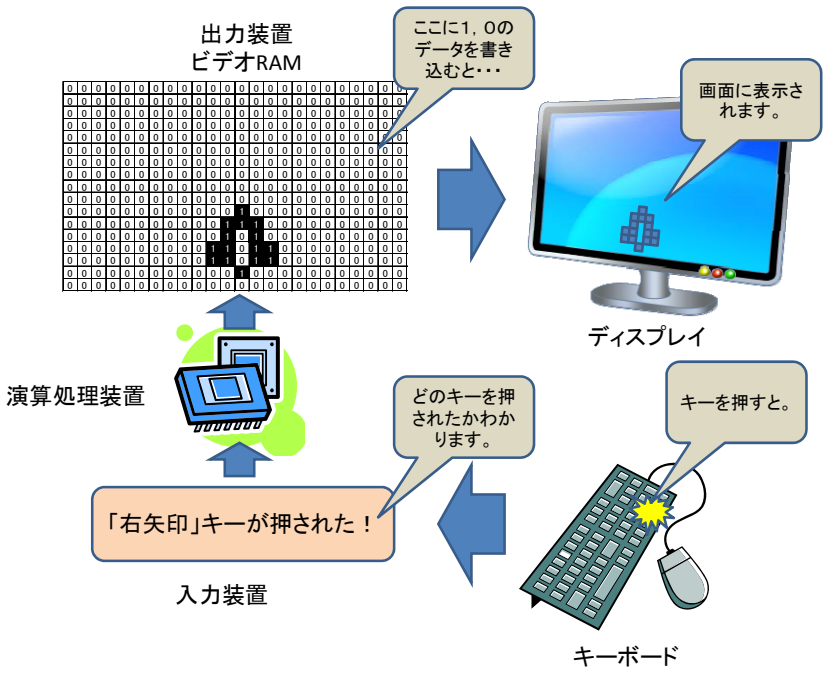
シューティングゲームを考えてみましょう。

まず、画面に絵を出す必要があります。

コンピュータのある特定のメモリ（ビデオRAM）はそこにデータを書き込むと画面の対応する座標に表示されるようになっています。
１が光っている状態。０が光ってない状態。
ハードウェアとしてそうなっているのですね。（最近はもっと複雑ですが、あくまでも簡単に書いています。）

つまりビデオRAMにデータを書き込めばドットの絵が描けるのがわかると思います。

さて、このドット絵で宇宙船を描きます。



さて、次に宇宙船を動かしたいのですが、キーボードの左右矢印で動かすことにします。

コンピュータにはキーボードの入力装置があります。
その入力装置に問い合わせると、今押されているキーの番号を教えてくれます。

右矢印キーが押されたら右に動くことにしましょう。
入力装置に問い合わせ、「右矢印」キーを押していることがわかりました。

宇宙船を右に動かすには、まずこの宇宙船を消す必要がありますね。
全ての画像ビットを０にします。で、少し右にずらして描き直します。

いつキーを押すかわかりませんから、常に入力装置を監視している必要があります。（ループ監視）

実際にこのプログラムを書くと、一瞬で宇宙船は消えちゃいます。

なぜか？

コンピュータが速すぎるからです。
自分は一瞬押したつもりでも、コンピュータからみたらすごく長い時間押していることになります。

なので、途中にタイマーを入れて、コンピュータに少し待ってもらいます。

当然、宇宙船は画面右端まで行ったら止まらないと困りますよね？
これも宇宙船が消えてしまう原因の一つ。
画面右端まで来たら、止めなければなりません。

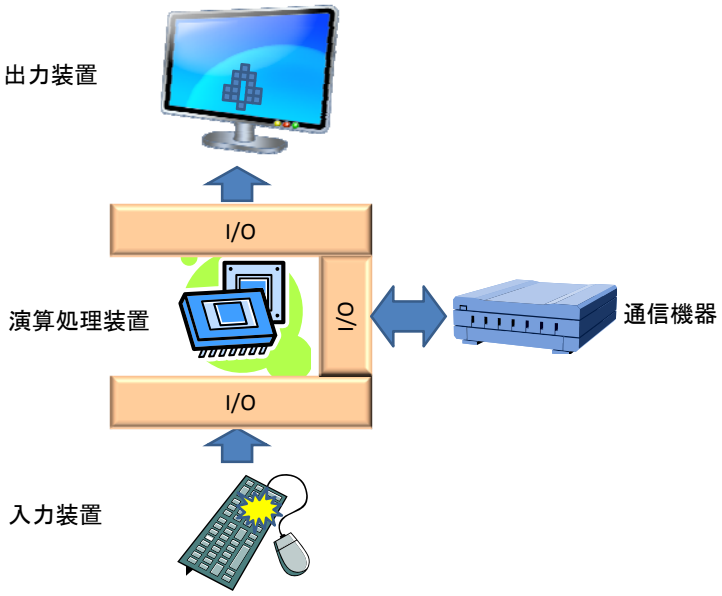
まぁ、左矢印キーだったら左に動くように追加すれば、とりあえず宇宙船は左右には動くようになるでしょう。

この先、スペースキーを押したらミサイルを撃つ、敵にあたったら判定して、爆発させる・・・・・
延々と処理が続きますがこの辺でやめておきます。

結構気の遠くなる作業を繰り返しているわけですね。

こう考えると、すべてデータ移動、演算、処理判定だけでできていることがわかると思います。

ここで画像装置、入力装置が出てきました。
これらハードウェアにデータを書き込んだり、データを取り込んだりもしているわけです。そうやってキーボードやディスプレイが連携しています。
通信機器もまったく同じです。データを書き込んだら相手に送信するように作られています。
これらすべて「決め事」ですね。



これらの積み重ねでコンピュータは動いているわけです。

なんとなくわかりましたか？

キーから指を離したら止まる。
画面の右はじまで行ったら止まる。
速すぎてもいけないので、待ちを入れる。

こんなの当たり前のように見えますが、コンピュータには当たり前という言葉はないのです。
つまり、すべての動作を全部定義してあげないと行けないわけです。

想定外のことが結果「バグ」となるわけです。

ここが最も重要なところです。
コンピュータはとてつもなく膨大なデータを瞬時に処理できるけど､手取り足取り教えないといけない大馬鹿者、という感じですね。

実際の業務系システムを構築する場合でも同じです。
例えば、処理するときに必要なデータが本当にあるのだろうか？

正常系ではあるのが当たり前ですが､何かの処理の遅延によりデータがそこに無い場合だってありえます。
もし、考慮していなかったら、エラーとなってしまうわけですね。

IT システムと建築

ITシステム構築をわかりやすく説明するために私がよく使う比喩表現は**「建築」**に置き換えることです。

実際ITシステム構築は、建築とすごく似ていると思います。

例えば、家の建築を考えてみてください。

家を建てるとき、役割分担されているのにすぐ気が付きませんか？
大工さんが全部やっているわけではないですよね？

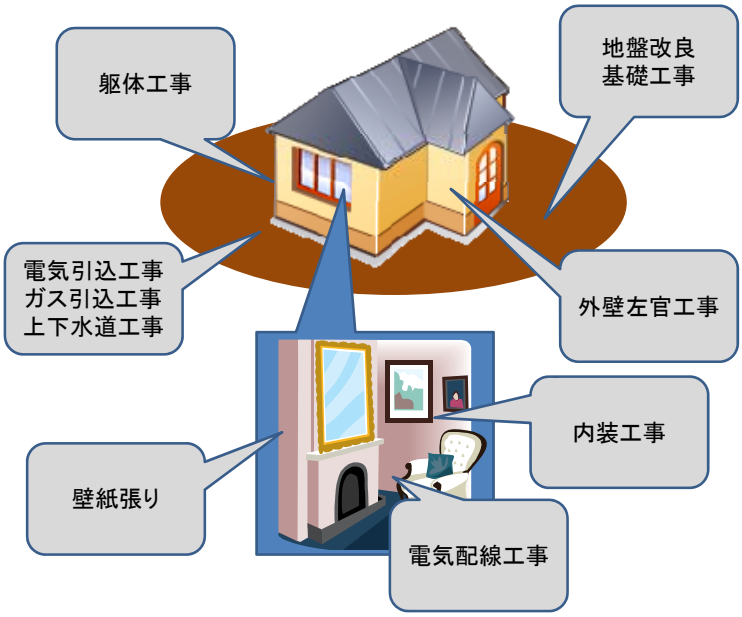
まず、重機をつかって地面をならします。
そこで基礎を作っていく。
土木的作業が必要となります。

そこには当然ライフライン（水道、ガス、電気を引き込みますよね。）

で、その上に柱が立つ。躯体部分の工事です。

壁を作ると、内装と外装工事。内装は壁紙を貼ったり、キッチンを据え付けたり。内装業者の仕事です。

外装は壁を左官仕上げなら左官屋さん。ペンキ仕上げならペンキ屋さん。
サイディングなら大工さんでできますね。



いろんな技術を持っている人が集まって作業をしているわけです。

基礎工事をITでのインフラ作業、
柱を立てるのをアプリケーション開発、
内装工事をWEBの画面構築
と言うように考えるとなんとなくわかりませんか？

当然、スキルが異なります。

「システム屋」とひとくくりには当然できませんよね？

大工さんに左官仕事は業務外なのと同じです。

構造面でもいろんなことがあります。
木造軸組、２ｘ４、ユニット方式。

それぞれ顧客の要望によって異なってきます。

それぞれの工法の利点、欠点もありますね。
木造軸組の良さは増改築が楽であること。柱構造だから開口を広く取れること。

２ｘ４は規格であるので、部材が安く調達できること。また大工さんの腕にあまり左右されない工法であること。反面、耐力壁を使うので、壁を勝手にとりはずせない＝増改築の自由度は狭まります。

これって、そのまま「ITシステムの開発方式、開発言語」に置き換えるとわかります。
なんでも手作りのCなのかクラスを応用するJavaなのか？それともパッケージを買ってくるのか？
構築手法ですね。

全てに共通しているのが、「発注者はどうしたいのか？」ということです。
この「要望」を正確に伝えないことには希望の家は建たないわけですね。

そして一番大事なのが「設計士＝アーキテクト」。

お客様の要望を聞いてどういう構造、工法をとるべきか考える人です。

これはITシステムでも全く同じです。

大工さんだけいても、設計士がいなければ・・・
開発者だけいても、このアーキテクトがいなければ・・・

ここまで書けばどうなるか、容易に想像つきますよね？
下請けしかしたことない開発業者を安いからといって使ったり、買収したりしてもダメということですね。
「要件」をきちんと理解し、「ITシステム設計」が出来る人が必要ということです。

IT コストを削減するには？

システムの見積はわかりにくい・・・

システムの見積を依頼された方は結構いると思います。

で、この見積り、各社各様。金額も差があります。

「なんで、こうもバラつくのか？」不思議に思ったことがありませんか？

こういうとき「こんな感じのシステム」程度の情報で、見積もりを依頼していませんか？

実は、この程度の情報では正確に見積もりは出来ません。

作成する画面の数などから、今までの経験を基にざっくりと見積もりをしているのが現状です。

先ほど、システム構築を「住宅建設」に例えましたが、ひとつ大きな相違があります。
「家」の場合は、必要要件は想像できますよね？
家の場合は、玄関があり、居間があり、キッチンがあり・・・・。これが基本要件。

ですが、業務システムの場合は、業務要件がわからないケースが多いのです。**開発側はその業務のプロではない**からです。

要件がわからないので、「エイや！」で見積もっているのが現状です。で、何かあると怖いので、バッファを乗せます。

業者によっては、今後のメンテナンスでカバーしようと開発費を格安で出してくるケースもあります。

結果、各社各様の見積りが出てくるわけです。

この状態で、相見積もりをとって安いところに頼んだ・・・。

もし、そこが仕事ほしさに安く出しており、さらにリスクを考えずに安い金額を提示していたとしたら？

怖いですね。

コストを安くするには？

では、安く構築するにはどうしたら良いのか？

簡単です。**「要件定義」**をきちんとすれば良いのです。

つまり、様々な事項をすべて洗い出し、何をしたいのか明確にすれば良いのです。

でも、通常これをきちんときる人があまりに少ないのです。

「要件定義」とは何をしたいのかを明確にすることではありますが、どのくらいの人がアクセスするのか？サービスは２４時間３６５日止めては行けないのか？というような周辺の「要件」もきちんと記述する必要があります。

でも、これが結構難しいのですね。

なぜか？

この「要件定義」はIT化を目的としての話なので、IT目線で語ってあげなければならないからです。

実は、この業務側とIT側のギャップ、意識のズレがそのまま見積りの高さに現れているわけです。
社内にこれができる方がいれば良いのですが、いない場合はどうすれば良いのか？

最も良い方法は、**「システム化計画」**を依頼することです。

一緒に考えて「要件定義」をやってもらうわけです。

「え？システム開発前に、お金出してそんなことするの？」
「そんなの営業行為の一貫じゃないの？」
そうおっしゃる方がいそうですね・・・・。

確かに、システム作る前にコストが発生するのはちょっと抵抗感があるかもしれません。

ですが、前述した見積り方式を再度読んでみてください。

バッファをのせているという話をしたと思います。

皆さんが逆の立場だったとしたら、どれくらいバッファを乗せますか？
倍近く出しても怖いかもしれませんね。

「そんなのプロだからわかるんじゃないの？」
はい、想定はできますが、客側がどこまで考えているのか正直わからないのが現実です。人の気持まで読み取れません。

それこそ、先に書いたゲームの話ですが、「画面の右端で宇宙船は止まる」というのは、考えた人間からは当たり前の「要件」ですが、知らない人にとっては「想定外」なわけです。

どれだけ「想定外」な話があるのか？

これを明確にするのが「システム化計画」です。
・何をゴールにするのか？
・どういうことを求めるのか？
やりたいことを列挙して整理していくわけです。

このシステム化計画の結果得られるのが「要件定義書」です。

家の例で言いますと、設計図、部材（品質要求度合）、耐震要求度・・のようなものが決まった状況です。
ここまでわかれば、リスクは相当減ります。
自ずとバッファは必要なくなりますので、コストは３～４割は削減できるはずです。

別にシステム化計画をしたところに発注しなくても良いのです。
相見積もりを取りたければ、この要件定義書をベースに各開発ベンダーに「RFP＝Request for Proposal（提案要求書）として提出すれば良いわけです。

これを出せば、見積の前提条件が定まるので、粒のそろった提案見積書が出てくるはずです。

システム化計画とは何か？

早い話、「どういうものを作りたいのか？」を明確にして、「何を作るのか？」「その時、重要な事項は何か？」というのをはっきりさせることです。

要件定義を導き出すための作業ですね。

ただ要件定義をしてください、とよく言われることがあると思いますが、ほとんどの人は何を書いて良いのかわからないと思います。それをはっきりするのがシステム化計画です。

当然、そこには将来の構想も含まれます。
「今後、こういう機能を追加したい。」とか。

またまた住宅建築の例を出しますが、
木造軸組と２ｘ４の大きな相違は、増改築の自由度です。
木造軸組は柱構造ですので、開口を広く取れるし、増改築も容易。
一方２ｘ４は壁構造ですから、壁＝耐力壁。なので勝手に壁をぶちぬくことはできません。

と、言うように違ってきます。

システムも全く同じ。将来の拡張性を考えるかどうかでも変わってきます。

さらに要件定義より先の、概念設計までシステム化計画でやってしまいます。

例えば、業務フローを一般的にするのであればパッケージソフトの利用でも良いわけです。
「パッケージソフトに業務を合わせる」ということですね。

ビジネス戦略的に差別化ポイントでなければパッケージ。
差別化したいところはスクラッチ開発（１から手作り。）というようなことも一緒に考えていきます。

つまり、システム開発の背景―ビジネス環境、戦略をきちんと押さえ、何が求められているのかを明らかにし、どういう作りにすれば良いかまで計画する作業となります。

保守費について

開発費にあわせて考えなければならいのが「保守費」です。

この保守費ってなんでしょうか？

これは「ITシステムを維持するためのサポート費用」のことです。

開発したらしっぱなしというわけには行きません。サポートが必要です。
例えば、OSのセキュリティ・パッチを当てなければならない場合。これで果たして問題なく動くのかどうか？こういう時の問い合わせ先が必要だからです。
開発側もこのために人を配備しておく必要があります。そのための費用です。

やはり開発した人間が一番よくわかっているのですよね・・・。

このコストを下げる方法は、開発側から考えてみればわかると思います。

つまり、この人員確保の費用がきちんとまかなえれば良いわけです。
パッケージであれば、汎用性が高いのでたくさんの顧客がいます。そうすれば、少ない専門人員でカバーできますよね？
受託開発の場合は、その後定期的に改善案件などを貰えれば、人員を確保することも可能となります。

また、サービスレベルも重要ですね。
何か問題が起きた時に即時に処理するとなると、その人員は常に手を空けて待っている必要があります。
でも、時間的余裕があれば、その限りではない。

早い話、人的リソースをどう管理するか？でコストが変わってくるということです。

なぜ障害は発生するのか？

なぜシステム障害は発生するのでしょうか？

これに結構頭を痛めている経営者の方も多いことでしょう。
「システムは電源さえ入れていれば､動くのでは？」と言われる経営者の方も結構いますね･･･。

なぜ発生するのか考えてみましょう。

システムではいろんなハードウェアの上で動いています。
で、リソースという言葉があります。
ハードウェアのキャパシティとか性能ですね。
つまり、全てのものには「限界」があるわけです。「リソース無限大」ではないわけです。
例えば、サーバにデータをためるのであれば、ハードディスク容量があります。
これを超えたら、システムは正常に動きませんよね？

サーバの処理限界を超えたら、当然正常に処理されないわけです。

なので､常にハードウェアリソース稼働状況などを監視していなければならないのです｡スイッチ入れていればOKというわけではない理由はここにあります。

ハードウェア障害だってあります。
よくあるのが、ファンが壊れること。これは電源を落として、上げた時によく起きます。

なぜか？動いているときは慣性が働いていますが､一度止めてから再稼働するときは負荷がものすごくかかるわけです。こういうときに壊れますね。ハードディスクも同様。
特にメンテナンスや移動のためにハードウェアの電源を落とし、上げた時によく発生します。

ということですので、日頃の運用監視、メンテナンスが大事となります。

ですが、人的事故も当然起きます。

メンテナンスの手順を間違ってしまった。
パラメータの設定を間違えてしまった。

そのため正常に動かなくなったというケースです。

これはきちんと手順書を作ってそのとおり運用していれば防げる問題ですね。

あと結構あるのが、要件定義漏れ、考慮漏れというものです。

先にも書いたように、システムは想定されていないことへの対処はできません。

ある日の処理は、通常の時の処理と異なるとか、イレギュラーケースを想定していないことによる障害は結構あります。

この「要件」を出せるのはユーザ側です。
「要件」に無い処理はされない。ここは重々認識してください。

IT 人材について

ここでは人材について考えてみたいと思います。
IT戦略をたてるためには必要不可欠な人材です。

CIO について

CIO＝Chief Information OfficerとはIT戦略の中で最も重要な人材は業務とITの両方を理解している人となります。つまり業務、ITの双方のギャップを埋められる人物のことです。

IT戦略をたてるためにはこのCIOが必須です。

ですが、残念なことに日本ではこのCIOにはあまりお目にかかったことはないです。CTO＝Chief Technologyl Officerはいるようですが。

技術は手段であり、目的ではないです。大事なのは業務側です。
業務遂行のためにどの技術が使えるのか？あるいはこういう技術を使えばこういうビジネスが実現できるということがわかるかどうか？ここが重要ですね。

日本でCIOがあまりいない理由。
私個人としては優秀なSI会社（System Integrator）がいるせいではないかと見ています。
彼らが全部やってくれます。責任も負ってくれます。（高いですけど）

そもそもSIという言葉は日本だけのようです。
海外、おとなりの韓国にもSIはいません。
なので、彼らは自分で考えなければならないわけです。自分で責任をもって進める必要もあります。

ITシステムを建築に例えていましたが、ここでも似た構図がありますね。
ハウスメーカ、ゼネコンです。

これも日本特有だと思っています。
すなわち**「おまかせ主義」**。

発注する側から見れば、全部お任せしてしまえば良いわけですから楽です。
となると、スキルとしては「ベンダー使い」「交渉術」「寝業師」となるわけで・・・

実際、システム責任者という肩書きがついていながら、上記のことしかしていない方を多く見てきました。
何かあるとすぐにベンダーに「提案して！」という人たちです。ありえないですね・・・。

正直、今後のビジネス差別化を考えるとここは由々しき問題です。
経営の立場では無視できない問題だと思います。

ではどうすれば良いのか？

両方を備えた人物は少ないのは事実。
であれば、業務とITをそれぞれ熟知した人たちをチームとして動かせば良いわけです。
実際、大手金融機関相手に過去にそのようなコンサルを行ったことがあります。

もし育てるのであれば、IT素養のある人間を育てるのがベストです。

IT 人材の採用方法

ではそのIT素養のある人材はどう確保するのか？
ここも経営者の頭を悩ます問題だと思います。

先ほど書いたように、ITは多岐に渡っています。どんな人材が必要なのでしょう？

ベンチャー企業等でよくありましたがITインフラ系の人をそのまま採用してしまうケースです。
事業を立ち上げる時に真っ先に行われるのがインフラの整備。ネットワークやメールなど。そのためその担当者がそのままIT専任になってしまうケースです。
ですが、アプリケーション開発・設計にはまったく違う知識・技術が必要なのは今まで読んできてくだされればわかると思います。

では、開発経験者を応募して書類選考するとします。

職務経歴書を見ると
・大手金融機関において業務システムの開発を担当
とか書いてあるケースがあります。ここで「おぉ、〇〇業務系をやったことあるのか？」と思ってはいけません。

たいていは下請けとしてある一部のモジュール開発をしていたのが大半です。つまり言われたこ

とはできるけれど・・・というケースです。

こういう場合「もっと現場に近いところで、上流工程をやりたい」ので応募しました。というのがよくあるケースですね。

つまり、職務経歴書だけではどこまでの素養があるのかわからないのです。

では、どうすれば素養を見ぬくことができるのか？

実はITシステム化のセンスは「抽象化能力」なんです。

プログラム言語などは単に知識に過ぎません。この抽象化ができないとIT設計なぞまず無理です。

この抽象化、日本人は結構苦手ですね。そういう訓練していないせいだと思います。

この抽象化能力があるかどうかは絵を描かせてみればわかります。
物事を絵を使って示すことができるか？概念図を作ることができるか？

例えば、ある業務プロセスフローを絵で示すことができるか？ということです。
その時、処理を分類・整理して絵にする＝抽象化できるか？ということです。

このセンスがあれば、あとは教育してシステム知識などつければ大丈夫です。
逆は無理です。センスとはその人に備わっている能力だからです。

用語観点から

IT用語を書き連ねるとキリがないので、気になる言葉だけをピックアップしました。

フルスクラッチ開発

これは、０から開発する方法です。
オーダーメードですね。

オーダーメードなので、自分の思うように作れます。
ですが、当然高くつきます。

最近はさすがに全部０から作ることはないですね。
WEBサーバやDBサーバなど定型的な動作をするものは、パッケージソフトを使います。

フルスクラッチで開発しなければならない部分は、個人的には「差別化」すべき部分だと思います。
つまり、パッケージではどうしようもない部分のこととなります。

最近では様々なモジュール、クラスなども出てきているので、完全フルスクラッチ開発は無いと思います。

できるだけ有りモノをうまくつかい、どこを差別化するか？というのが開発のポイントとなります。

パッケージソフト

一定の処理をパッケージ化したのでパッケージソフトと言われます。

フルスクラッチ開発が洋服で言うと「オーダーメード」だとすれば、このパッケージソフトは「吊るし」ですね。

普段、PCなどにインストールしているソフトはほとんどがパッケージソフトだと思います。

会計ソフトなどもパッケージですね。

パッケージソフトの良い点は「安い」ということです。

最大公約数の機能を盛り込んでいるわけなので、多くの人が使えます。
つまり、たくさん売りやすいということ。自ずと安くなります。

じゃ、欠点はということですが・・・

フルスクラッチ開発の反対ですね。**差別化ができない**ということです。

定型処理であればパッケージソフトで良いということになります。
ですが独自の処理や、差別化したい部分には当然パッケージソフトは適しません。

よく「パッケージをカスタマイズする」という話がありますが、これは決してやってはいけないことです。

なぜかと言いますと、保守の範疇から外れてしまう可能性が高いからです。

パッケージはどんどんバージョンアップしていきます。
ですが、カスタマイズされたものはバージョンアップされないケースがあります。
もし、元になったパッケージの保守がなくなったとしたら・・・・。そのパッケージをわかる開発人員をいつまでも確保しておくわけに行かないので、いつかその日が来るはずです。

パッケージを使う時のコツは「業務をパッケージにあわせる」ことです。
それ以外のことは期待しないことですね。

最近のクラウドでのASPサービスはこのパッケージと同様です。
当然、自由度に限界があります。

ただ、よくできたパッケージはそのカスタマイズ要望も想定されていることもあります。
パラメータ設定のみで変更できることもあります。

あと重要なのがAPI（Application Interface：外部接続のためのインターフェイス）ですね。
これがあれば、外部のシステムと接続し、差別化や特殊処理をそこで実行できるからです。

いずれにしろ、カタログだけではなく、実際のものを見て念入りに調査すべきです。
営業マンの「できますよ！」は信じないことです。
たいてい「（こうこうこういう前提を置けば・・・・）できますよ！」だからです。

ITシステム開発はなんでもそうですが、「やればできます」です。

徹底的に調べましょう。
これもシステム化計画の最も大きな取組課題（Make or Buyの議論）です。

ソリューション

これは韓国系サービスでよく言われる言葉です。
日本ではあまり馴染みがない発想かもしれません。

これはフルスクラッチ開発とパッケージの間に位置します。
要は基本的動作をするモジュール群が揃っていて、それを組み合わせて作る手法です。
「セミオーダー」ですね。

実は、業務処理を分析すると、ほとんど同じようなパターンのものが多いんです。

例えば、与信業務。
個人の情報を取得し、信用機関に問い合わせて、その結果を取得し・・・
という感じです。
当然個々のデータ構造は変わってきますが、大枠の作業は皆同じ。
つまり、この基本的動作が揃っていれば、後はプログラムを付け足すだけでできるということです。
このインフラ部分がソリューション。

有名なところではオンライン証券会社のトレーディングシステムがあります。
日本でも結構な会社で韓国系ソリューションを使っています。

この特徴は画面処理系、データ処理系、データ送受信系がきれいに分かれていることです。
この処理記述によっていろんな分析データや画面が作ります。
そのため非常に迅速に新しい機能を追加したりすることができます。
（しかし、提供側が使いこなしているかどうかは別問題ですが。）

クラウド

これも、最近よく聞く言葉ですね。

クラウド＝Cloud＝雲のこと。
ホント、雲をつかむような話です（笑）

特にこちらにアプリケーションをインストールすることなく使えるサービスのことを指します。自分の方ではなく、すべての処理を「あちら側」で処理されるサービスを言うわけです。あちら側は「雲の向こう側」ということでクラウドと言われているようです。

実はこれ、別に新しい話でもなんでもないです。
まぁ、クラウドは一般的に**アンブレラワード**とも言われています。
つまり、「傘」みたいに全体を包含した言い方ですね。

早い話、昔で言うと「ホスト」「共同利用型」サービス。
概念的には新しくもなんともありません。

ただ、技術的に**「仮想化」**は大事ですね。
これによって実現できた部分は大きいです。

仮想化は簡単に言うと、ハードウェアの物理的制限を取り払うことができる技術です。

昔は、ハードウェアにソフトウェアを直接インストールしていました。
これだと、ハードウェアの物理的制約を直接受けてしまいます。
ハードウェアを個々の会社で買ったりしなければなりませんし､キャパシティが足りなければ追加購入、老朽化したら入れ替えをしなければ行けません。先で書いたように、ハードウェア障害は結構起きます。

で､このハードウェアの上に全体を管理するソフトを載せるわけです。ある意味これで隠蔽します。

前にも書きましたが、ハードウェア自身にはI/O（入出力）システムがあり、普通はそれを通じてやり取りします。

それをもう管理ソフトという１枚皮をかぶせ､そこで仮想的にハードウェアを作り出すわけです。
ソフトウェアから見ればやりとりするためのI/Oがあれば良いわけです。
なのでインストールされたソフトから見たら違いはわかりません。

こうしておけばハードウェアの増強や、入替えとかが簡単になるということです。また、ハード

ウェアの稼働率を上げることもできます。
ミニマム構成から大規模システムまで柔軟に対応できるようになります。
最終的にはコスト削減につながります。
このメリットが最も大きいですね。

Amazon の EC2 がパブリック・クラウドでは一番知られているでしょう。
なぜ Amazon がこういうことを始めたのか？
Amazon のサーバ群に最も負荷がかかるのはクリスマス商戦時期です。
でも、その時期のためだけにサーバ増強するのはもったいないですよね？
で、空いている時は他の人に使わせてしまおう！ということなのです。

同じ事を企業（グループ）内で実施すればプライベート・クラウドとなります。こちらの方が、利用するところが限定されますし、物理的なサーバの場所なども管理できますので、最近増えてきていますね。

ただ、自由度が上がる分、それをハンドリングするための運用技術が必要になります。

最後に

この無料レポートを最後まで読んでくださりありがとうございます。

まだまだ書き足りない部分はたくさんあります。お伝えしなければならないところも・・・。

ご意見、ご感想、ご質問がありましたら是非とも下記までお願いいたします。
Info@emz-style.com

また、システム化計画、IT 推進化計画、IT 人材採用の支援、その他アドバイスなど必要な場合もお気軽にご相談ください。

エムズスタイル
前田　稔

追記
http://www.emz-style.com/
が私のサイトになります。ここからいろんなコンテンツにつながります。
IT だけではなくいろんな分野でも精通しています。
基本姿勢は「なんにでも興味を持ち、チャレンジすること」。いろんな経験を持つことでいろんな視点から物事を考えられるようになります。
良かったら覗いてみてください。