システムエンジニアになるためには

内容

１．システム工学とは

システムエンジニアの前に，システムとシステム工学について明らかにする。

教科書や辞書にある“システム”の定義は，

　複数の構成要素が有機的に結合して目的を達成する

とある。これだけでは，意味が良く分からないので，従来の工学分野と比較検討してみよう。従来の工学では極論すれば，電気・機械など特定分野でのある種の評価による最適解答とその実現を求めることが目的であった。特に，工学として成立するためには，比較的きれいな形式化ができる部分から議論を進めた傾向が在った。

システム工学では，システムの構成要素が多数・多様であり，多様な要求･評価尺度のバランスを取りながらトータルとしてのベストを求める。このためには，他分野にわたる知識の利用と，現実の曖昧さ・不確定要素に対応する必要が在る。

1-1システムエンジニアはなぜ必要か

現在，コンピュータ技術を含む自動化機器の技術進歩は，目覚しいものがある。この結果自動化機器の設計方針にも以下のような，発想の転換が必要になっている。

従来(1990 年代以前)　機械で出来る部分の自動化＝機械中心の自動化

　↓

現代(1990 以降)　　人間が楽になる自動化=人間中心の自動化

このため，処理装置の設計においても，従来の“技術シーズで実現できる部分から実現する”方針ではなく，“利用者のニーズに合わせて必要技術を準備し，トータルとして満足の行くものを造る”必要がある。

この様に，設計者には複数分野の技術シーズを目的に合わせてバランスよく調和する能力必要である。従来の分野専門の工学技術者に加えて，システムエンジニアが必要になる理由はここに在る。一方，利用者のニーズも多様化・ネットワーク化して利用者自身でも完全につかみきれない場合が在る。

システムエンジニアは，システムの利用状況を将来まで見越して把握し，現状の技術シーズによる最適なシステムの提案と構築を行う必要がある。

また，物作りにおけるプロジェクト運営も，システムエンジニアの重要な業務である。

これは，複雑なシステムを理解する人間が，システムエンジニアしか居ない。しかも，システムエンジニアも全てを図面などに完全に記述できるのではなく，適宜作業者の質問に応じて明確化する必要が在るからである。

なお，従来型の機械の設計では，開発メンバーの大部分が機械のイメージや利用形態を共有していたが，現状のシステムでは，そのような共有感覚が無くなっている。

1-2システムエンジニアの業務

システムエンジニアの業務を大きく分けると以下の 3 面になる。

① 何を造るべきか明確にする

② どのように作るべきか指示する
③ 出来たものの良否を判定する

これを業務の流れで言うと以下のようになる。

① 基礎的情報収集
② 提案作成・見積もり
③ 打ち合わせ
④ 要求間の調停
⑤ 社内調整
⑥ 承認申請図作成・詳細見積もり
⑦ 製品に対する評価・保証の方針付け
⑧ 社内手配
⑨ 製作段階のフォロー
⑩ 試験段階のフォロー

以下にこれらの作業の詳細を述べる。

(1) 基礎的情報収集

この場合に収集すべき情報を分類すると以下のようになる。

(1-1)利用者に関する情報

- 利用者からの要求情報
- 利用者自体の情報

(1-2)社内情報

- システムの要求を明確にするための役に立つ情報
- 実現に対する社内事情の情報

利用者の要求情報は，発注仕様書の中に記述してある情報が基本であるが，これだけでは通常不十分であり，打ち合わせの中で付加されていく情報，今までの既存システムから知っておくべき情報など，形式化されていない情報が在る。

この情報のなかには，工期や予算のように数値化して明確化できる情報と，使い方のように曖昧さを含む情報がある。この様な曖昧さの中にこそメーカーとしての技術力を示すことが出来る。また，要求自体の中に矛盾がある場合も多くあり，これらの明確化と調停・バランスがシステムエンジニアの大きな仕事である。

また要求情報のなかには，こうしたいと言う積極的な情報のほかに，こうしては行けないと言う消極的情報(制約)も重要である。システム構築の実際では，ベストな解答を求めるのことはとても難しく，致命的な欠陥を持っていないので満足する場合も多くある。

一方，物を作るということのみ先行し，安易な妥協に走ることも良くない。最終的な物作りのためには，割り切りや妥協が必要になる場合もあるが，まずは理想的状況を明確にすることが，見通しを良くして正しい評価に導く。

この様な，システムに対する要求に合わせて，システムの使用環境を明確にすることが重要である。背景・環境に気を配ることにより『要求機能がなぜ必要か』が理解できることが多い。なぜを知ることで，評価を正確に出来る。この様な，利用者の要求の背後にある価値観の明確化が，システムエンジニアの重要業務であり，利用者の価値観と合わせれば，信頼を得るとともに応用の利く作業が出来る。

なお，利用者の情報は与えられるのを待つのではなく，積極的に引き出すことが必要である。この為には，設計側での思想を明示した提案も一つの方法である。この時，利用者の価値観に合わせることも重要であるが，新規提案で価値観を変換していくことも重要である。

利用者自体の情報は，組織に対する情報と，利用者からの情報の性質に関する情報である。利用者と一言で言っても色々な立場がある。設計者・実運用に当たるオペレータ，保守担当者(会社)，お金の支払部門などがある。さらに，階級の上下がからんでいる。これらの立場の違いを明確化すれば，要求の矛盾がなぜ生じたか納得が行く場合もあり，要求の優先度を着ける易くなる。

また，情報には現在の情報だけでなく，将来動向も重要である。現状の設計で反映すべき項目，予備項目などの形で生かす項目，将来用の隠し味にすべき項目・機能がある。この様な，＋αがシステムエンジニアの力の見せ所である。

社内情報の内要求を明確化に有効な情報は，一つは当該システムの常識と言われる情報である。今まで設計した同種システムの情報，利用者の作業状況に関する情報をできるだけ集めることが重要である。代表的な情報の処理に関してイメージを持ち，代表的な操作方式に関しても知っておく必要がある。さらに，システム独自の用語に関しても，辞書を作成するぐらいの気構えが必要である。また，現在のシステムだけでなく昔の製品を知っておくことで，色々な知見を得ることができる。

初めて，担当するシステムに関しては，上記の情報収集ができないかもしれないが，一般的なシステム設計の定石は役に立つし，類推が利きそうなシステムを調べることは有効である。

実現に関する社内情報は，当社製品の性能及びコストに関する情報と，生産体制の情報が重要である。本当に要求を実現できるのか，そのためのハードウエア構成はどうしたら良いか，ハードウエア構成の選択肢はどれだけあり，その利得はどうなっているか。最終決定は，先の話でも可能性に関してはあらかじめ検討する必要がある。

生産体制では，特にソフトウエアの人員確保が重要である。開発量の見積もりとキーマン確保が重要である。また，営利企業として常にコストに気を配り，設ける工夫をする必要がある。

情報収集の実行に当たっては，営業部門等から要求情報を入手し，当方からのシステム提案をうまく擦りあわせる必要がある。

作業内容を纏めると，以下のようになる。

① 資料の収集と読破

② 社内情報の調査

- 既存類似システムの情報調査
- 今回機能の実現可能性の検討
- 以上の検討から得たKnow Howの整理蓄積

(2) 利用者打ち合わせの準備

この段階での作業は重要であり以下の手順で行う。

① 提案書の作成

設計者としての考え方を明確にするためにも必要である。提案として全体の構想を記述していく間に，構想段階では見えない細部の不明確点や矛盾が見えることも多い。

② 議事録案の作成

会議に関しても，主導権を持って議事進行するためには，あらかじめ打ち合わせのストーリーを検討し，議事録の下書きを作り打ち合わせに臨むべきである。特に，打ち合わせにおいて，決定すべき事項を明記し，該当項目のみを記入できる用紙を持っていれば，スムーズに進行する。ただし，事前のストーリーに拘泥して，利用者の意見を押さえることは望ましくない。感情のエネルギーも保存されるため，後ほど不満となって爆発することがある。また，議論中に思い付いたことが，新発見につながることもある。この様な，脇道情報を収集するためにも，通常の打ち合わせを楽に進めることが重要であり，議事録の下書きが必要であう。

③ 見積もり

物造りは，営利活動であり奉仕活動ではない。適正な利益の追求は忘れては行けないことであり，作るべきもののイメージが見えてくれば，その場で実現手段の概略構成を検討し，概算での見積もりを行うべきである。この段階では，詳細な見積もりは不可能であるが，規模のめどを立てないと実現できないものを引き受けることになる。システムは，規模の大きさが変われば，定量的のみならず定性的にも変化する。

(3) 打ち合わせ

この段階では強引すぎて反発を買うことも行けないが，できるだけ主導権を持つことも重要である。また打ち合わせを通じて，下記の情報収集に勤めるべきである。

① 利用者の本当の要求

要求の中の優先度付け。潜在的な要求など，言葉にならない情報を明確にする。

② 体制(キーマン探し)

矛盾する要求は，キーマンを見極めて，優先度を着けてもらう必要がある。

③ 決められないことの切り分け

利用者が単に知らない情報は，次回までに調べてもらうことができるが，解らない場合は，いつまで経っても答えを得ることができない。この場合には，可変量を含んでシステム設計を行ったり，オプション処理を追加したり，EUC(エンドユーザコンピューティング)で，利用者自身に対応してもらう等の工夫が必要である。また，正確な値は分からないでも，他の値との関係や式の形は教えることもある。

とりあえず貰える情報を上手に使う必要がある。

(4) 要求間の調停

システムで実現するためには，種々の矛盾する要求の間を調停しなければならない。例えば以下のような矛盾がある。

① 要求の間での矛盾

② ニーズに技術シーズが追従しない

③　納期・コストに生産体制が追従しない

④　他のシステムとの競合(キーマンの取り合い)

この様な矛盾の解消には，利用者の立場を考慮した大局観が必要である。絶対に譲れない制約条件を満たした上で，他の条件をできるだけ最適化する部分がシステムエンジニアの腕の見せ所になる。特に，要求事項には本当に必要な MUST 項目と，要望的な WANT 事項があり，その見極めと WANT を満たすことでの魅力アップを上手に行う必要がある。

(5) 社内報告とレビュー

今までの，打ち合わせ内容を社内関係部門に報告し，チェックを受けると伴に手配予告を行う。レビューの参加者は，概略以下のとおりである。

①　前回の工事経験者，いなければ多くの経験のあるベテラン

②　製作担当者，試験担当者

なお，レビューを効率よく受けることも技術力の大きな要素である。上手く現状を報告する。そして，参加者のコメント意図を上手に理解する。これは，システムエンジニアの技術力そのものである。

(6) 承認図作成

今までの打ち合わせ結果を，図面化して利用者の承認を得る。このとき，打ち合わせ時点で発生した矛盾などを解決しておく。このようにして動かない“神様”の情報を作ることで一旦仕様を凍結する。この資料は，営業も含めてオーソライズする。また，この時点での詳細見積もりは重要である。

なお，この時点でも仕様の不明確部分は生じるが，その部分は不明確として明記することが大切である。

(7) 製品に対する評価保証の方針付け

システムとして複雑なものを実現する場合には，その善し悪しをどのように判別したら良いか誰にも解らないと言う場合もある。このため，システムエンジニアはそのシステムの使用される環境にまで配慮しながら，システムの良否判定ができる尺度を構成する必要がある。特に，試験ができないようなものを，できるだけ作らない様にすることが重要である。

システムの良否判定を行うためには，理論面での見通しとモデルによるチェックが必要であり，大局観を持った判定が必要である。しかも，この判断基準を他のメンバーに説明する必要がある。

(8) 社内手配

以上の段階で要求事項が明確になれば，物造りのための手配に入る。この場合ハードウエアに関しては，通常の工場の物作り手配で流れる。しかしソフトウエアの場合には，設計間の打ち合わせ又は外部発注先への仕様連絡の形で手配になる。ソフトウエアの場合にはハードウエアのように管理体制の分業も進んでいないので，システムエンジニアが生産体制まで気を配る必要がある。どのような情報をどこに流すかと言うことも，システムエンジニアの腕の見せ所になる。あまり多くの詳細情報を流しても誰も見てくれない。相手に合わせた情報を，適切な表現形式で供給することが重要である。相手によっては詳細なフローが有効な場合もあるし，タイムチャートが有効な場合がある。表現しないで良いと

言う決断は，全体を見ているシステムエンジニアの判断事項である。

また，作業に関する標準などの指示も必要である。特に既存のしがらみで複数の暗黙的標準が対抗する場合がある。この様な状況を整理して，交通整理することもシステムエンジニアの重要業務である。複数の設計思想が入り乱れるとその間でトラブルが生じるものである。特に，最初は上手く動いていても改造時に大きなトラブルを引き起こすことがある。思想面の統一は，後で効くと言うことを念じておくべきである。

(9) 製作段階でのフォロー

特にソフトウエアの設計製作段階では，細かい疑問や改善提案が良く出てくる。この扱いに関しては，以下の 3 点を必ず守る必要がある。

① 口頭でのやり取りは禁止，必ずメールかFaxにする。(文書化と時刻管理)

② 回答はできるだけ速やかに行う。できない時にはいつまでの第1報を返す。

③ 変更時の波及検討を十分に行うこと。

(10)試験段階でのフォロー

試験段階では，製作側の理解の食い違いなどでトラブルが多発する。トラブル時の善悪判定は，システムエンジニアが決断すべき点が多い。また悪い部分が見えないが，おかしいと言う時にも関連部分を切り分けるのも，システムエンジニアの仕事になる。この様な場合は，勘に頼ることも多いが，常日頃から全体の動作を見たり，昔のトラブル事例を十分理解していれば，正しい判定につながることが多い。このような大局観を踏まえた，勘働きはシステムエンジニアの責任である。

1-3シーズ先行とニーズ先行

利用者に対して新しいシステムを提案する時に，以下の2方向がある。

① 現在の技術(新技術)でできることを提案する(シーズ先行)

② 利用上の問題点を解決する方向で提案する(ニーズ先行)

これだけ見ると，誰でもニーズ先行の仕事が重要と考えるであり，要らないものを売りつけられても仕方ないと考える人も多いであろう。しかし世の中での実例は，別のことを示している。

＜例1＞ あるゲーム機メーカーの社長の発言

次期CPUの発売に関する新聞記者の質問に対して。

『当社は世の中のハードウエアがどのように進歩しようが関係ない。お客様に提供するゲームがどう変わるかである。ゲームが実現できるならば今のハードウエアを変える必要はない。使い方の新しいものがでてそのための実現手段として，ハードウエアを求めるべきである。』

＜例2＞ スマホを与えられたベテラン

『こんなに細かい情報を載せても，老眼の俺には見えない。もっと絞ってくれ。』

2. システムエンジニアの扱う情報

システムエンジニアの扱う情報は，大きく分けると以下のようになる。

① システムの機能を示す情報
② システムの実現方法を示す情報

更に，入手先で分類すると

① 利用者側からの情報
② 社内などの内部情報

の2種類に大別できる。なお，個々の情報に対して信頼度の尺度を付加して考慮する必要がある。言い換えると，システムエンジニアは，曖昧な情報も上手に利用して物作りにつなげる必要がある。

(1) システムの機能に関する情報

システムの機能に関しては，“なぜ”そのシステムが必要かを明確にすることが第１である。“何を”の前に“なぜ”を明確にすることで，今後の見通しが良くなる。更に“なぜ”を明確にするためには，自然と使用環境などの周辺情報までの配慮が必要になる。この部分は，経験が物を言うことも多く，ベテランでは常識になっていることが若手に伝わらないこともある。このための仕組みを良く考えるべきである。

次に，機能的な面を明確にする必要がある。これは，対象システムの標準的な使用方法を明確にすることから始まる。この段階でもベテランでは常識となっていることを，明文化することが重要である。(場合によってはプロタイプのプログラムを含む)通常の動作を一通り明確にした後，異常時の処理まで検討する。異常処理に関しては多様な処理が絡むため，正常処理の全体イメージが確定した後でないと，議論が進まない。

なお，システムを良く理解しているベテラン達の間では，レビューは異常処理が中心になる。しかし機能理解が不十分な時には通常の動作例を確認する必要がある。この為には，既存のシステムを良く勉強すべきである。直接の類似点が見えなくても，抽象化してみれば参考になる場合が多い。

概要が明確になれば，代表的なデータで，入力から蓄積出力と一通りの流れを考えてみることも重要である。頭の中でも良いから動かしてみると，思わぬエラーが見つかることがある。個別動作と全体像のバランスを上手くとって，仕様明確化を図る必要がある。

機能が明確になれば性能面の検証に移る。まず，利用者側の要求と使用条件の検討から必要情報量と応答性を検討する。機能検討は定性的な議論が主体であるが，性能検討では概算値を含む定量的な検討が必要である。ただし，定量的に値を得ても，定性的な裏付け理解が重要である。言われた値を鵜呑みにするのではなく，突っ込んで理解することが良い技術者になる境目である。この紙一重を超えることが重要である。

概略の検討が終わると，個別詳細の検討に移る。この段階では，定型化を考えて進めることも可能になる。この時全データ量の見通しを立てることも重要である。データ量が一桁異なると，処理の性質が変化する可能性がある。

この様な検討を，文書化し早期に配布するのもシステムエンジニアの条件である。

(2)システムの製作に関する情報

システムを，実際に作るためにはソフトウエアとハードウエアの実現に関する詰めが重要である。そのためには，要求内容と使用可能な資源の比較検討が必要になる。

(2-1) ハードウエアの検討

要求機能を満たし，性能面でも満足できるものができるかの検討を行う。この為には，常日頃から製品の説明書になれておくことが重要である。また，当該機械の独特の表現に慣れることも重要である。更に注意することは，カタログの性能は最良条件の性能であり，実負荷条件ではその半分もでないことが多いと言うことである。このため，使用例などを良く知っておく必要がある。

一方，ハードウエアの手配に関しては，従来からの伝統ある工場方式で行われている。この流れは，ソフトウエアの立場で参考になること多い。このように，色々な面から改善のヒントをつかむのもシステムエンジニアの仕事である。

(2-2) ソフトウエアの検討

ソフトウエアに関しては，いわゆる make or buy(新作か購入か)の決断が必要である。この為，以下の 2 面に対する理解が必要である。

① 標準的なソフトウエア製品でできること。特にできないこととの境界点。

② 新規にソフトウエアを製作する時のコストと納期

現状の複雑なシステムでは，何らかの形でソフトウエアが関与しているので，ソフトウエアの製作に関する見通し無しではシステム設計を行うことは難しい。何ができるか，どの程度のコストでできるか，どれぐらいの工期か，そして誰ができるかこの様な情報を押さえる必要がある。特に，ソフトウエアの個人差を良く踏まえた把握が重要である。

また，不明情報の処理もソフトウエアにで処理することが多い。通常の手段は可変項目で対応することであるが，何でも変えれば良いと言うものでもない。特に，ソフトウエアエンジニアがシステムエンジニアに不安を持つと，可変部分を多く持つオーバースペックのソフトウエアになり，コストアップの要因になる。

(2-3)検証評価用の情報

検証評価用としての専用のシステムを作れればベストである。良いチェックリストがあれば，誰でも試験ができてトラブルの少ないシステムになる。しかし，現状では試験員の判断に頼る所が多い。このため，システムの目的とその理由を明確にし，広い視点で判断ができるようにすべきである。特に，当該システムの価値観を共有・伝承することが重要である。

また，試験がしやすいシステム設計が重要である。試験が容易になる切り口・機能の割り切りが重要である。

3. システムエンジニアの思考法

―いわゆる論理的思考法―

3-1論理的思考方法の分類

システムエンジニアに限らず技術者にとって論理的な考え方は重要である。ただし，一口に論理的と言っても大きく分けて2つの方法がある。一つは，数式や記号論理学などに代表される形であり，自然科学の思考法である。もう一つは，社会科学などで主に使用されている，仮定を主に用いる思考方法である。

自然科学の思考方法は，例えば数学などの様に厳密な構成が特徴であり，ある事象の後には必ず次のことが生じる，と言う形の必要十分条件による繋がりを良しとする。この思考法では，ある決められた約束事が成立する記号世界に推論すべき内容を写し取り，説明を作る能力が必要である。このような，推論方法に演繹法・帰納法がある。

社会科学では，明確な公理は存在せず前提条件とその結果の繋がりで，推論を進める。この様な曖昧な条件でのつながりは，自然科学者にとっては論理的ではないと言うかもしれない。しかし，多くの人間が納得すると言うことでは，この様な自然科学に乗れない論理も必要である。

ここで物理学などの自然科学の論理と，社会科学の論理を図示すると下記のイメージになる。

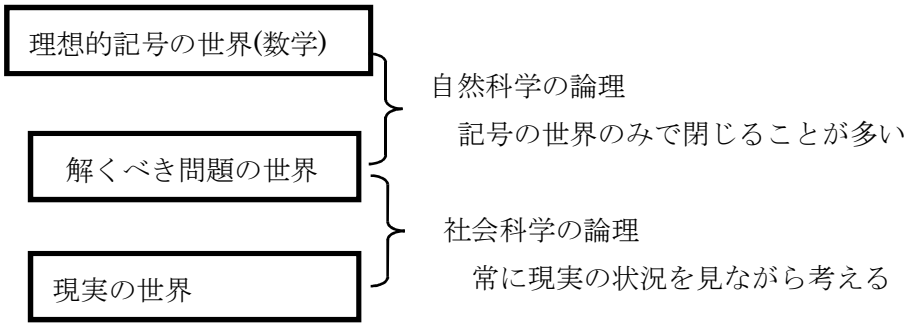


図3-１　自然科学と社会科学の論理

システムエンジニアの仕事では，利用者と言う人間臭い部分に関しては社会科学の論理，製品と言う工学的産物に関しては自然科学の論理と使い分ける必要がある。

以下では，この両者について少し詳しく述べる。先に自然科学の論理から入るが，これは諸学の基本であり，常に磨く必要がある。

3-2自然科学の論理

従来から，論理的な話の代表として出てくるのが，幾何学の証明である。現在我々が学ぶユークリッド幾何学では，点･線などの基本的事柄を定義し，これらの間で無条件で成立する少数の公理を設定している。そして，公理の組み合わせから多くの定理を推論し，幾何図形の性質を導いている。このように，有限の種類の対象物と，前提になる公理からスタートして多くの事柄を導くのが，幾何学の思考法であり，プラトンに始まる西洋哲学で厳密かつ正確な模範的思考法として位置づけられている。

これを，近代哲学の祖であるデカルト（1596～1650）は，“われ思う故にわれあり”から，スタートする演繹的な思考体系に纏め上げている。デカルトの考え方を簡単に纏めるのは難しいが，以下の４ヶ条は技術者の心得としても十分に活用できる。

① 明らかに真実と認めた上でなければ，いかなるものも真実とは受け入れてはならない。
② 考えている対象は，明らかな真実が見えるまで出来るだけ細かく分割せよ。
③ 何が明らかに真実であり，それから導かれるものが何か，順序立てて整理せよ。
④ 以上の手順で全てを尽くしたか確認せよ。

この 4 ヶ条は，ソフトウエアの試験などにもそのまま使用可能である。但し，明らかな真実が見つからない場合には，根本的にくずれる。デカルトの哲学の後ろにはキリスト教の神様を見ている。この様に根本的な真実を見渡せるのは神の技である。特にキリスト教は一神教であるので，固定した視点で表し易い。しかし“お客様は神様です”の世界では，多神教の世界であり，単純な演繹が使いにくくなる。この様な場合に対策は，別途説明する。

デカルト原理は換言すると，分解すれば解ると言う話である。これは，当時最高の機械である時計の理解から来ている。構成要素の振り子・歯車・ぜんまいこのようなものの個々の動きを理解して時計を理解する。一方，明らかな真実の発見には，観察の効果がある。当時，顕微鏡・望遠鏡などの観測装置の進歩があり新しい科学発見が，増加していた。

なお，論理的と言っても技術者の論理は，記号論理学や三段論法のように狭い話だけではない。ニュートンの運動方程式やマックスウエルの方程式が導く結論も，論理的に厳密に展開されたものである。この様な物理学や数学の体系に良く対応づけて話すことが，ハードなシステムに対する論理として重要である。

ただし，ここで狭い意味の論理に関して復習することも十分意味があると考える。帰納・演繹そしてアブダクションと言うことを十分知っている人は，次節まで飛ばしても結構である。

(1) 三段論法および演繹的推論

よく論理的推論の例として出てくるものに，以下のような例がある。

(大前提)	人間は死ぬものである	A→B
(小前提)	ソクラテスは人間である	C→A
(結論)	ゆえにソクラテスは死ぬ	C→B

このように三段階で推論するので，三段論法と言う。この推論を記号的に表現すると，右側の様になる。この場合に注意すべき点は，大前提に含まれている情報と，小前提に含まれている情報の間の関係が正しいことである。上記の例では，ソクラテスは人間の 1 例と言うことで正しい推論になったが，以下の例ではどのようになるであろうか。

(大前提)	独創的な発想をする人は天才である	A→B
(小前提)	私は独創的な発想をする	C→A
(結論)	ゆえに私は天才である	C→B

この場合，“独創的な発想をする人全てが天才とは言えない”ことに注意して欲しい。単なる，奇人の可能性がある。この様に，要素間の関係を正確に導かないと，形だけが正しくて中身の伴わない議論になる。上の例では，以下のように修正したらまともな推論に近づく。

(大前提) 独創的な発想をする人は天才である可能性がある

この点は，数学で使った必要条件と十分条件の使い分けで説明できる。復習すると，必要

条件は，ある事が行われる時に満たされるべき事であり，ただそれだけでは，そのことが行われると決定するには，情報が不足している場合である。“独創的な発想”は“天才”の必要条件であっても十分条件ではない。逆に，十分条件とはあることが行われる場合に，それだけで決定できる条件である。例えば“人間であること”は“死ぬ”ことの十分条件である。必要条件を広げて十分条件となった時，その条件は必要十分条件と呼ぶ。必要十分条件は，ユニオンショップの労働組合員と社員の関係である。社員は，労働組合員であり労働組合員は社員である。労働組合からの除名は，社員からの除名でもある。このように，必要十分条件は，あることが成立する最低限の条件である。
三段論法での誤りは，必要条件と十分条件の取り違えによることが多い。

⑵帰納的推論

さて，今のように前提から結論に向かう推論を演繹的推論と呼ぶ。演繹推論は，推論実行時に誤りの可能性が少なく，検証も容易である。このため，技術者の思考法としては基本的なものである。これは，図１で“理想的記号の世界”のみを考えているからである。

この様な立場とは別に，自然科学でも経験や実験を重視する考え方がある。これを例で示すと以下の様になる。

(事例 1) ソクラテスは死んだ
(事例 2) プラトンは死んだ
(事例 3) アリストテレスは死んだ
(結論)　人間は死ぬ

この様に，多くの事例から共通的な事項を見出して，一般的なことを述べるのを帰納法と呼ぶ。ここで注意しておくが数学的帰納法と言う証明の方法は，数学の公理を前提にして，有限の数字の性質を全ての数に展開する方法であり，演繹法の一種である。

帰納法の長所は現実と対応していることであるが，短所は以下の格言が示すとおりである。

“柳の下に二匹目の泥鰌はいない”

もう一つ実例を示す。

A 課の 2000 年の新人はハンサムであった
A 課の 2001 年の新人はハンサムであった

これから，A 課にはハンサムな新人が配属されると言う規則が導き出されるであろうか。

この短所に対応するために，実験においてはサンプルの偏りを避ける取り方を採用したり，統計的な検証をしたりする。また，別途理論からの演繹的結論に対する保証に，帰納的な推論を行う場合も多い。例えば，上記例では A 課が，お客様窓口部門であるとなると，少し規則化の可能性が出てくる。

コンピュータを利用すると，シミュレーションが比較的に簡単に出来る。シミュレーシ

ョン結果から規則性を読み取るのも，帰納的推論である。ただし，データを揃えれば規則性が単純に見えるのではなく，理論的基礎の見通しを持って，積極的に規則を読み出すことが重要である。ただし，先入観で見落としをすることも行けない。良く考えて，しかも現実の前では謙虚である。また，実験結果の理由“なぜ”を，常に考えることが重要である。

⑶仮説推論（アブダクション）

帰納法・演繹法と近代科学の推論法が確定した後に，1900 年代にもう一つの推論方法として仮説推論法（アブダクション）を，アメリカの記号論学者 C.S.パースが提案した。とりあえず，例で示す。

(事態)　　私が本屋に行くと，直ぐ側にくる美女の店員がいる
(適応規則) 好意を持つ人がいると側に生きたがるのが人間の習性
(仮説)　　彼女は私に好意を持っている

この方式の弱点は，適応規則の選択が色々あり，仮説が正しいと言う保証が無いことである。今回の適応規則は以下の 2 項目が正しそうである。

(適応規則 2)　万引きしそうな人間はマークすること
(適応規則 3)　しつこい立ち読み客が来たら近くで本の整理などして牽制する

このため一つの仮説にこだわらず，色々の可能性を検討する必要がある。また，仮説の有効性を検証するために，広い視野と明確な判断基準が必要であり，今までの経験と理論的な知識を総動員して，正しいものを選ぶ必要がある。

特に技術者にとって判断基準を持つことは重要である。意思決定を行った場合に，その決定が外れたらどのような被害を生じるか，利益と被害の評価を加えた上で，仮説を採用すべきである。

また，仮説の考えは，トラブル発生時に良く使う。機械が動かない，その原因として電源が入っていない。この様な仮説は誰でも使っている。経験ある技術者は，この様なトラブル原因仮説を上手に使うものである。

⑷演繹法・帰納法そして仮説推論法の比較

三段論法の推論法を纏めると，下図 3-2 のようになる。

A である	A であれば B の例がある	A ならば B である
A ならば B である		B である
B である	A ならば B であるらしい	A である可能性がある

図 3-2 三段論法による推論法

この 3 形式を，システムエンジニアの仕事に関連付けると以下のようになる。

① 初めてそのシステムと対面し，どのような原理で動いているか全く解らない。この状況では“AならばBである”形式のルールを発見するアブダクションを使うことが多い。

② 種々の状況が見えてくると，帰納法が使える状況で，“AならばBである”形式の規則を造るようになる。

③ システムに関する見通しができれば，基本的な前提や原理が見えてきて演繹規則を使って推論が進むようになる。いわゆるトップダウンの設計が出来る段階である。ベテランのシステムエンジニアはこの段階で仕事をすることが多い。

以上の内，①②の状況を改善することが重要である。以下では個々の改善策について述べる。

(4-1)仮説推論の効率化

仮説推論で考える時に，“複雑なシステムでも本当に難しい理論を使うのはほんの少しの部分である。” と言う経験則を念頭に置くべきである。通常，中学や高校で習った基本的なことを上手に応用することが大切である。漠然としたシステムでも，説明を受けると“そんなことは知っている”と言うことが多い。思い付きが出来るまでが遠いのである。

この様な状況を少しでも改善する方法を考えよう。以下のヒントがある。

① 目的を明確にし，関連するものを探すことで少しでも探す範囲を絞り込む。

② 探す内容の評価方針を設定し，探し易くする。

③ 類似例からガイドを得る

④ NM法のように抽象概念から方針を立てて発想を導く。一歩抽象化すると良く見えることがある。

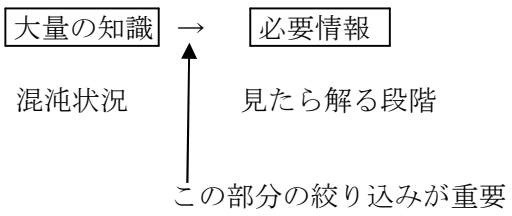


図 3-3 仮説の発見

この場合に，難しい項目を勉強する必要がある時には，別途その部分を切り出し検討する必要がある。難しい問題でも関連部分を限定し，目的を明確化して検討することにより，管理し易く対応できるようになる。

(4-2)帰納的推論法の効率化

システムに対して色々の情報があり，そこを整理して基本的な規則に纏める作業には，以下の要素がある。

① 動作の情報から決定的な属性を抽出して規則にまとめる。

② 可能性のある規則の評価を行い有効なものを抽出する。

③ 矛盾する情報の間で調停を行う。

この為には，KJ 法(QC 新 7 つ道具の親和図法)のように，バラバラの個別情報を纏め上げる方法が有効である。この段階でも，仮説を見つけることも重要であり，演繹的に理論の適用を考えることも重要である。ここで重要なことは，『人間の行動等はバラバラに見えるものでも，後ろに合理的な原理がある』と信じることである。また矛盾する情報の調停のためには，個々の情報に対して根拠と入手時期を明示して，優先度付けを行うことが重要である。この段階では，一つ一つの記述をまとめて一般化・抽象化して適用範囲の大きい規則を見出すことが重要である。

(4-3)演繹的推論法の効率化

この段階は，トップダウン的に考えがまとまるため，効率的な作業が可能になる。このような段階を，できるだけ多くして効果的なシステムエンジニアリング作業を図る必要がある。しかし，演繹的な作業は前提情報の見落としによるトラブル対策が必須であり，実施例での確認や，上級者のレビューが重要である。

また，自分の知識と現実のつなぎを良く行って，抽象的な理論の現実での実施例を見出すことが重要である。

3-3社会科学の論理

前節では，自然科学の論理について述べたが，システムエンジニアの仕事は，人間相手の仕事が多くなっている。このため自然科学のように，唯一絶対の真理という前提条件が成立しないことが多い。『お客様は神様です』これは，正しいが，この神様が複数の『八百万の神様の世界』である。例えば一つの注文に関しても，設計・購買・運用そして設備保守と言う分担で利害が異なってくる。更に，上役と部下の意見相違，設計者同士の方針不一致などを調停することが必要になる。このため，近代自然科学を生み出した西洋人が信仰している一神教でなく，日本の神道のような多神教の世界での思考方法が必要になる。

一方社会科学では，このような複雑な状況に関して，種々の理論を構築している。ここでは，社会科学での論理や理論構築方法が，システムエンジニアの仕事にどのようなヒントを与えるか，検討する。なお，自然科学的＝デカルト式論理の問題点を，18 世紀初頭にイタリアで活躍したヴィーコ(社会学の始祖)は，以下のように指摘している。

① 完全な真実を求めることで，真実らしい情報や豊かなイメージを捨ててしまう。

② 全ての事項を列挙することは不可能である。

③ このような検討を行っていると，時間を浪費し現実の論争に負けてしまう。

(1)論理の作り方

自然科学の理論は，アリストテレスの三段論法や，数学の証明のように厳密な形でできている。これは，デカルトの哲学にもつながっているが，一定の間違いの無い事柄から演繹の鎖を，伸ばしていく方法である。この鎖が一本で見やすいことが，特徴である。ただし，一個所の鎖が切れると，全てがバラバラになる危険性を含んでいる。更に考える前提は，自明の事柄と考えている。

一方社会科学の論理は，弁証法のように，日常的な言語の意味に大きく依存している。このため，曖昧なことも多く残った状況で推論する。下図は，『A と B が同時に起こる』と言う命題に対する，両者の対応である。

A と B が同時に起こる

自然科学者：A と B のどちらが原因か，それとも両者の原因の C があるか？追求

社会科学者：原因が見つかれば運が良い(割り切り)

このような状況で，他人を説得する場合には，できるだけ多くの資料・事例を集める。ヴィーコは，これを『一連の連鎖的推論より，百の三段論法で攻める。』と言っている。例えば，ある人を誉めるのに，『センター試験で XX 点とった人です』との言い方に対して，『1970 年代にオブジェクト指向方式の有効性を指摘し，1980 年代に分散処理方式の普及と，大型計算機の破綻を予言し，1990 年代にオープンソース活動の成功を予言し・・・』と言う風に，畳み掛けるように例示する表現を使用する。

こうしておくと，複数の裏付けで少しの例外では反論できない，丈夫な議論展開が可能に成る。図 3-1 で記述したように，社会科学は現実との確認を重要視するのに対して，自然科学の方は，単純な割り切りで表現できる美しい数式の世界に閉じこもっていることが多い。社会科学者は，自分の理論と知識に対して，以下のような表現で，自分の理論の正しさは，前提との相対的なものとしている。

『社会科学の理論は，社会的現実の一面を記述したモデルである。そして知識は，そのモデルの内の真実部分である。モデルは現実に近づくが，必ず誤りがある。』

『理論は実世界を記述しているジグゾーパズルの一部分である。我々は，その断片しか持っていない。しかも何が不足かと言うことすら知らない。このようにして集まっている図形を崩し，新しいまとめ方をする時にパラダイム変換が起こり，新理論が誕生する。現実に関する知識と言うのは，完全な絵になっている。いかにそれに近付けるかが理論の善し悪しになる。』

ところで，このような正しさの相対性は，工学の立場で物作りを行う場合にも，存在している。例えば，電気回路をオームの法則で設計しているが，精密に考えるためにはマックスウェルの方程式や相対性理論による修正が入ってくる。このような状況で，何を考えずに済ませるかは，暗黙的な約束事で処理している。

ただし，工学の中でも機械や建築などの伝統ある分野では，全体像を図面などを用いて伝えることで，周辺情報や価値観を伝えて，この問題の解決を図っている。

(2)社会科学の理論

以上の点を考えて，社会科学では理論の構築のための方法論を強化している。ここでは，その一部を紹介する。なお，社会科学の理論は工学の理論と以下の点で異なっている。

『工学は物を作る立場の理論であり，全体を見通しての議論が必要である。一方，社会科学の理論は説明のための理論であり，部分的なものを対象とすることが多い。』

このため，工学の立場で見れば，部分しか見ない身勝手な議論に見えるかもしれないが，手法には学ぶべき物がある。

(2-1)理論概念

社会科学と自然科学の相違点として，概念装置(理念型)がある。社会科学では，観察対象が社会現象であるため顕微鏡や望遠鏡のような観測装置が要らない。その代わりに，何に着目すべきかを自分で決める必要がある。例えば，色々な人の行動中特定のものに注目す

るなどである。このような議論の対象とする概念を自由に決めることが可能な点が社会科学の特徴であり，これらの概念を“概念装置”と呼んでいる。物理学に戻れば，ニュートンが“重さ”と言う曖昧な概念から“質量”を再定義して，力学を構成したように，良い概念が見つかれば，簡単に理論ができる。

理論を作るための概念には，変数と非変数がある。非変数は現象を分類するためのラベルであり，固有名詞・普通名詞または性質を表す。一方変数は，特定の次元に対応して変化する値であり，理論の構築には適当な変数を見つけて，その変化に対応する現象を見出すことが良い。なお，自然科学と異なり，完全に厳密な定義は不可能であり，概念の名称から連想的に引き出される情報にも気を配る必要がある。例えば，“ＸＸ事故”と言う概念には，事故一般の連想が重なっている。

(2-2)理論言明

言明とは，概念間の関連付けを行うもので，各種の現象を記述し予測を可能にするものである。言明には，確信度によって，仮説・定理・前提・公理など色々のレベルがある。ここで，注意すべきことは自然科学のように，絶対的な公理は考えず，その時の前提と言う風な相対的な表現である。こうすることで，下記の点を明確にする。

①自分の考えていることには前提条件がある。

②その前提は，絶対的に正しいとも，他人と共通であるとも限らない。

③以上を考慮して論理的に導いても正しいとは限らないと認識する。

さらに，言明にも以下の 2 種類があることに注意すること。

①理論の中での言明。言葉での説明など。

②現実世界との対応での言明。変数の測定方法など。

特に現実との対応には注意すること。人間的な要素を，直接的に計ることは難しい。例えば，ソフトウエアの生産性向上と言う場合に，実際に計れるものは，プログラムの行数や，完成までの時間などの間接的かつ部分的な情報が中心である。このような情報でも，入手できるだけましという発想が必要である。システムエンジニアたる者は，直接的なデータが無いからとあきらめず，間接的な情報でも何とかする執念が必要である。

(2-3)定義

上記のように，一つの概念に対する定義も以下の 2 種類がある。

①理論定義：言葉による説明

②操作定義：現実世界からの情報収集法(測定方法など)

定義は，両方が必要であることに注目のこと。物を作ることに注意したら現実に測定できる値のみに注意が向き，利用する時の意味が分からずに，情報を集めるように成る。例えば，“プログラムの行数ばかりで評価していると，無駄な命令が増えていた。”という状況がある。逆に，理論面が先行すると地に足のつかない話に成る。プログラムの本当の量などと議論している間に，プログラム自体を見て無駄な命令を数える。これも一つのデータである。このように，理想と現実の間を上手に走る必要がある。

3-4社会学的推論の訓練法

上記のような社会科学では，豊富な事例を引用しながら議論する。このために，ヴィーコは，下図の三段論法の A に相当する中名辞を多く見つける必要性を指摘している。

A ならばB である　　(例)***新しいことを予測する人は***頭が良い

C は ***A である***　　XX 氏は ***X を予測した***

C は B である　　XX 氏は頭が良い

図 3-4　中名辞の利用

これを実現するためには，下記の訓練が必要である。

①色々な事柄を記憶しておき，必要な時に思い出す能力を磨く

②記憶内容を現実のものに近付ける想像力の訓練

また，上記の“A ならば B である”部分を選別するためには，正しい価値観で望ましい B を選ぶことが必要である。この能力を磨くためには，若い時代に多くのものを見て“良いもの，良い仕事を見て本物の人財を知る”経験を積む必要がある。

このような経験を経て，共通感覚(いわゆる常識)を磨くことができる。また，議論の場においても，最初は口を慎み自己の感覚が場違いでないことを確認しながら，徐々に討議に参加する。講義を受ける場合も，最初は聴講生として聴くだけに徹する。その後，自分の考えと，常識ができれば議論に参加する。このような手順を意識することで，自分で自分の成長を計ることができる。

3-5まとめ

一口に論理的といっても，色々な形があることを分かったと思う。ここでは，自然科学の論理と社会科学の論理について概要を述べた。社会科学の論理について興味をもたれた方は，以下の参考書を見てほしい。

“理論構築の方法”J．ヘイグ　白桃書房

“学問の方法”　ヴィーコ　　岩波文庫

蛇足であるが，技術者にとって自然科学的な論理の展開は常識であり，その上での社会科学的な論理展開である。制御理論や電気回路として説明できるところに変な概念を持ち込む必要はない。また，快適さという風な人間的な側面も，温度・湿度のように測定し工学的にコントロールできる項目もあることに注意すべきである。自由に温度・湿度をコントロールできるようにしてから，好みの議論をすべきであって，はじめから測定し難いからと逃げていると，良い製品につながらない。一時流行したファジー制御に関しても，工学的に現象を押さえてしっかりとモデル化し，その上で近似手段としてファジーで丸めるなら良いが，いいかげんなモデルを作った後，現物合わせを使用とすると，いつまでも収束しなくなる。

3-6工学の図面(参考)

厳密な情報の代表に見えるハードウエアの図面にも色々な曖昧さがある。例えば以下のような情報が読み取れる。

①機械図面

全体の組み立て・構成図から当該機器の動作・機能を読み取る。

②電気設備のスケルトン

トランスなどの主要設備の記号は，実際の配置を考慮して記述する。また，線の太さ記号の大きさは，設備の大きさを反映している。但し，完全な比例関係でなく，大きさの順

序関係を表現するのみである。

このように，図面でも含みの情報を利用していた。しかし，弱電からソフトウエアに至る道のりで，情報の厳密化と細分化が進み，図面の含み情報が消えていった。この一例が，UNIX の一つの原典と言われている“Software Tools：日本語訳 ソフトウエア作法”である。この原著には，プログラムリストは多くあるが，図が一つも無い。これは，初期のワープロの副作用であるが，ソフトウエアの厳密指向と全体図の軽視の結果である。

これと関連して，通常ソフトウエアの作業者は仕様書に無い物を勝手に作ってはいけないと，厳密な作業を行うように躾ている。この後で，システムエンジニアになる時には，曖昧な情報での意思決定処理が必要になる。この部分の倫理的ギャップもシステムエンジニアの育成障害になっている。

4. システムエンジニアの保有すべき技術と技能

4-1技術と知識

システムエンジニアは，幅広い分野の専門家と意見を交換し，システムに纏め上げる必要がある。このため，広い分野の専門家の意見を最低限理解し，彼らと意見を交換することができる技術レベルが重要である。

また，技術の深みを知るためには，最低限一つの専門分野で『これだけは自分に任せてくれ。』と言うレベルに到達する必要がある。また，このような『誰にも負けないレベル』に到達するためには，『自分の限界を超える経験』が必要である。このような経験を持つ人間は，難関に当たっても簡単に逃げない癖を持つ。

なお，逃げるというやり方にも大きく分けて二通りある。一つは仕事を拒否する逃げ方で，もう一つは技術的に曖昧なまま妥協の産物で物を作ることである。このような姿勢は，共同作業者が敏感に感じるものであり，信頼されないシステムエンジニアの成功可能性は低い。

以上を図示すると，下図のように幅広い一般知識の横軸と，深い専門知識を持った T 字型の技術力になる。なお，より望ましいのは，深い知識を 2 つ以上持っている，Π字型人間である。一つの分野にこだわると，見方が偏ることになるが，複数の分野を深堀していると多面的の見方が可能になる。特に，3 分野を経験した人間は，大抵の分野でも直ぐに専門家と話ができるようになる。

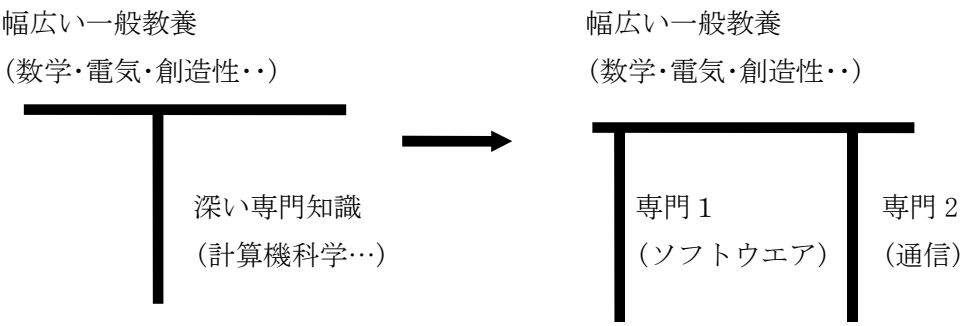


図 4-1 システムエンジニアの技術構造

このような幅の広い技術・知識を身につけても，使えなければ何の役にも立たない。このために，自分の持っている知識を他人に解りやすく説明することを常に心がけることが大切である。難しいことを，解りやすく説明することは，難解な数式を展開するよりも難しい。この難しさを解決するのが本当の技術力である。また，単純な規則で多様な場面で説明に使えるものを，多く持つことも大切である。例えば，フィードバックという言葉も，狭く取れば 1 ループ制御となるが，広い意味では制御の見返り監視信号もフィードバックである。このような，応用を上手に行えば説明が分かりやすくなる。

4-2技能

システムエンジニアの業務を遂行するためには，以下に述べる基本的な技能が必要である。とかく人間は楽にできることしか行わない習性を持っている。このためには，必要な技能を身につけて，知的な体力を充実させて楽に仕事ができるようにすべきである。

(1)読取り

システムエンジニアは，大量の情報を処理する必要がある。このため，ネットワーク情報を含む多様な文書から，必要情報を効率的に抽出する能力が必要である。

世間に出回っている速読方法は，効能書の 100%を期待しても失望するだけであるが，1 分間に 2000～2500 文字程度の読取りスピードは身につくものである。このため，普段から活字に親しむことが大切である。なお，一般的なヒントとして以下の 2 項目は有効である。

①文書を読まずに済ます訓練

書かれてある内容を予測し，予想と異なる部分のみを読む。IEEE(アメリカの電気関係学会)のスピードラーニングは，論文の速読方法として，参考文献等から内容を予測し，異なる部分のみを読むようにしている。

②悪文を読む訓練

我々が触れる文章は，名人が書くとは限らない。このため，学校教育で習うようなまともな文章だけではなく，支離滅裂な文章も解読する必要がある。このためには，日頃から新入社員の研修日記のような若い人の悪文に慣れておくことも重要である。特に，文章理解は相手の心を理解することであり，常に相手の心をおもいやる訓練をすれば，悪文でも述べたいことは理解できるものである。

この様な，相手の心を理解する訓練は，設計段階での検討内容を推測する能力を強化し，リピート工事や増設･改造時に既設設計の精神を理解することにつながる。

(2)聞取り

会話による情報収集は，文書による情報収集と同程度に重要である。この場合，単なる会話自体も重要であるが前後の処理も同程度に大切である。各段階の作業は以下のようになる。

step1：事前準備

何を質問すべきかの準備，相手に関する情報収集

setp2:会話中

明瞭な質問作成，正確な聞取り，収集情報の記憶・記録，次のテーマへの方向づけ

step3：事後処理
　収集情報の整理・記録

　上記の技能の中でも，基本になるのは聞取り能力である。一般に読取りに比較して聞取りの方がミスが多いと言われている。例えば，ある年の技術系新入社員 100 名程度に，2000 文字程度の文章に対して，四者択一方式で理解度テストを行うと，以下の結果が出た。
　読取りミス　　6%
　聞取りミス　　20%

　但し，この場合でも 2%程度は聞取りミスが無い者がいた。また，サイコセラピストなどの業務では，対話内容を全て記憶する訓練を行っている。この点を考えると，聞取り記憶という訓練も有効である。英語のヒアリングも，訓練により聞き取れる量が増えたが，日本語も同様に訓練の価値がある。例えば，ラジオの講座や講習会をテキスト無しで理解する様な訓練がある。
　また，積極的傾聴法の訓練を積むと，自然に相手の話を全部記憶できるそうである。なお，対話情報の聞き漏らしが出るのは，対話中に出た重要概念に執着したり，自分の思い付きが生じたことが原因となることが多い。この対策としては，メモを上手に取り執着心を開放することが有効である。メモは，忘れても安全にするための道具である。なお，積極的傾聴法では，一切メモなどを取らずに己をむなしくして，ひたすら相手に共感することで，全てを記憶する。この様なメモを取らない会話方法は，心理療法のカウンセリング手法の応用であることが多い。
　その他の項目では，情報収集・記憶及び記録(文章作成)が中心であるが，事前準備に関して，以下の点を注意してほしい。
　対話に当たって，相手に要求できることを明らかにすること。『肉屋に刺し身を売って欲しい』と言うことは，間違っている。さらに，『肉屋においしい刺し身について聞く』ことも，間違っている。このように相手に聞けること，話したがっていること(自慢したいこと)を聞くことが重要である。

(3)情報蓄積
技術者にとって，記憶能力は重要である。この表現は，誤解を招くかもしれないが，多様な事例を思い出し，その教訓を活かすことは技術力の向上と幅を増すために有効である。また，会社生活の時間価値向上のために『電話番号を記憶し電話帳を引く時間を節約する』などの意見もある。
このため，巷であふれている記憶術の訓練を行うことも，結構能力強化に役立つ。しかし，本当に重要なことは，単なるテクニックより『人間の記憶力は実際上無限の容量があり，脳細胞の内死ぬまでに使用するのは，10%程度である。だから，覚え過ぎることに不安を持つ必要はない。』という，信念を持つことである。実際に，記憶に関して問題になるのは，“記憶できない”ことではなく“必要な時に思い出せない”ことである。
以上の点を踏まえて，具体的な記憶力訓練として，以下の方法がある。
①記憶する時に思い出す手がかりをできるだけ多くする。

一つの方法は，感覚的イメージを豊にすることで，もう一つは，記憶する際にできるだけ説明を考えて，自分で納得しながら記憶することである。

②常日頃思い出す訓練をする。

夜眠る前に，一日にあったことを思い出す。散歩に出た時に帰ってから，道の周りにあったことを思い出してみる。この様な訓練を繰り返すと，次第に詳細に思い出すことができるようになる。また，1 日だけでなく前の方までさかのぼれるようになる。この訓練で，自分の記憶力に自信ができる。

(4)文章作成

入手した情報は，1 時的・恒久的に記録し，他の人間に報告する必要がある。このために，文書の形でアウトプットことは大切である。会社生活において，電子メールを含む文章作成の時間が多くあるので，この部分の合理化で大きな時間余裕が生まれる。貴重な頭脳は，創造的な仕事に振り向けるべきである。ただし，文書作成を止めて口頭の報告にしたり，記録を止め全て記憶するというものではない。

口頭報告を，受けた場合のやり取りは，文書化した場合に比べてはるかに時間を食うものである。更に，報告のタイミングを待つ間に自分の時間がつぶれる。また，記憶は保持するために頭のエネルギーを必要とするので，早く書き出して開放する必要がある。

以上で，文章を速やかに作成する能力の重要性が理解できたと思う。口頭で報告する場合でも，要点を文書やメールで送っておけば，報告時間は半減する。

文書力養成には，とにかく多く書くことが必要である。また，機会を作って良い文章を読む，さらに写してみる。この段階から次に書くべき事項の選択ができるようになる。

なお，書けない理由も，もう一歩踏み込んだ原因追求を行うと，対策が見える場合がある。

- 手の筋力が無いので書いていると疲れる
- 字が下手である。書くのがいやで人に見せたくない
- 良い文書とはどのようなものか解らない
- 書く事の重要性を理解していない

これらに対する個別の処方箋が必要である。

(5)口頭発表

この手法は，いわゆるプレゼンテーションとして，色々の教科書がある。手法は色々あるが，基本は相手の立場を考慮した説明である。

なお，関連事項として根回しがある。日本の伝統的な手法として，会議にかける前に重要人物やキーマンに対する説明を事前に行う根回しがある。これを，密室での闇取引と考えて，悪く考える場合もあるが，人間の反応速度を考えると事前説明は，重要である。会議のその場で全てを理解し，適切な発言ができる人や場合もあるが，後で良いことを思い付く場合も多い。この様な，事後の発見で会議内容がひっくり返らないように，事前の資料送付や説明は重要なことである。なお，欧米で根回しが少ない理由の一つは，小学生時代から論理的言語技術を仕込まれディベート(論争)技術を仕込まれているので，その場での速やかな反応が可能なことがある。

(6)人間的魅力

これを技能に挙げるのは抵抗があるが，システムエンジニアには必須の条件である。ただし，アイドルタレントのような魅力は不要で，仕事の上での信頼関係が一番の魅力である。このための基本は“約束を守る”ことである。これは，簡単に見えて非常に難しいことである。期限・コスト及び他業務との関連と種々の事項が重くのしかかってくる。このためには，仕事の優先度を明確にし，対処する必要がある。“約束を守り，逃げない人”これが信頼される技術者の基本である。

4-3創造性開発

システムエンジニアに限らず技術者は，創造的に仕事を進める必要がある。これは，発明発見というように大上段に振りかぶったものに限らず，利用者の要求が当方の技術的・コスト的な制約にあわない時に，種々の工夫をする。同業他社に負けない提案を行う。このように，色々の局面で使用すべきものである。創造性開発に使用する発想法は，知識として知っていることも重要であるが，技能として訓練し身につけることも重要である。以下に主要手法の概要を述べる。

(1)KJ 法(親和図法)

文化人類学者の川喜多次郎氏が，データの整理手法として編み出したもので，個別断片情報をカードに記述し，空間的に配置することで，その背後に隠れている基本原則を見出す手法である。この手法は，利用者から漏れてくる断片情報の整理に役立つ。特に基本原則や価値観を見出すことは，システム設計中の意思決定を容易にする。

(2)等価変換法

一見無関係に見える事項の関連性を見出す手法で，類推能力の向上に役立つ。類推能力が強くなると，一見独立の作業が同じ作業の繰り返しに見える。この段階まで進めば，標準化が容易に行えるようになる。

(3)ゴードン法

これは，アイデアを他人から得る方法である。情報を欲しい人間は，抽象的な表現で，色々な人に質問し，ヒントをつかんでいく方式である。例えば『新規の芝刈り機』を，考案するのに『分離』等のキーワードで，インタビューして種々のヒントを探し，『不要物を焼く』等の発言から，バーナー式の雑草焼却装置を発送するようなものである。

これは，他分野情報からヒントをつかみ出す訓練として有効である。

(4)シネティックス

擬人化方法で，自分が検討課題の機構に成り切ったつもりで発想する。このために，想像力が必要であるが，慣れれば詳細なシミュレーションが実行できる。これを完全に行うことは難しいが，常に機械や利用者の立場で考える訓練として有効である。

4-4(参考)数学の解釈

数式や数字にも現実の使用状況での解釈が必要である。例えば，多変量解析という分野がある。これは，多数の行列データの中から数値処理で，必要情報を抽出する手法である。例えば因子分析の説明を見ると，以下のように書いてある。

“多数の変数の固有ベクトルを見出し，共通因子を発見する。”

これでは，理解できる人が少ないであろうが，以下のように表現すると理解しやすくなる。

“複数のデータが相互に関連している場合に，一つの式に纏め上げて各変数の重みをつける方法。例えば，身長，B,W,H などのデータからスタイルの良さを

$\alpha \times$身長$+\beta \times B+\gamma \times W\cdots$

という風な，一つの式にまとめる方法である。αやβの重みが大きいほど，スタイルの良さに関連し，小さければ無関係ということを示す。

このように,自分の言葉で説明すると，数式も血の通ったものに成る。

また，電気回路の基礎で使用する j ωに関しても，原点の微分方程式に戻って理解する必要がある。せめて,『L で発生する電圧は，流れている電流の微分に比例する。これは，変化を妨げる作用による。』程度の説明を行いたい。

5. 理解できないことを理解する方法

システムエンジニアの仕事において，良くある障害が

何をして良いか解らない。これは，一体何をするシステムかと自分で何も理解できない。あるいは，一応解っているのであるが，なんとなくしっくりこない。

である。この様な場合の対策を以下で述べる。

5-1理解するとは

まず第一に，理解するということが，明確になっているであろうか？反対条件の，理解できないということから，攻めてみよう。

①動作面

次の動作が予測できない。何をするかは解るが説明ができない。等

②認識面

見たことが無い，状況が識別できない，イメージが湧かない。等々

この原因は，大別すると以下の 2 面がある。

①部分的に難しいところがあって理解できない。例えば新しい制御方式を使うが，理論が難しく，式が理解できない。

②部分部分は何となく解るのであるが，全体としての動きが見えてこない。

この内①に関しては，<u>難しいことの勉強</u>という通常の学習法が，一つの答えとなる。この部分は，後で述べる。また，システム的に全貌をつかんだ後には，部分の詳細は解らなくても良い場合もある。全体での位置付けが必要度を判定し，最低限の必要情報のみを求める手法がある。さらに，当該部分を可変とし，後から付加する方法もある。

一方，②の全体として見えないと言うことは，システム特有の難関である。特に，何と

なく納得できないという状況は，後で大きなトラブルを生じることに成る。以下では，この様な状況に対する対策を述べる。

また，用語が解らない・動作が解らないと言う 2 面も分けて考える必要がある。用語に関しては，辞書を作ることが解決の一つであるが，動作に関しては個々の動作をシナリオで書いたり，説明書として記述を行うことが一つの答えである。但し，用語の後ろにある概念や，識別すべき状況が分からない場合がある。例えば，小学生に対して円周率の近似値が，分数表現でも複数あると言うことを，納得させるのは難しい。彼らに対して無理数を教えていないため，分数で全ての数が説明できると，考えているためである。

この様な，背景にある情報が不足している場合には，当該製品の使用状況を動作する舞台の上での振る舞いとしてイメージすることで，理解しやすくなる。さらに動作原理を理解すれば良く見える。こうして，最終的には，目的とするシステムの機能をモデル化してまとめると，解ったと言う感触がつかめるようになる。

以上の議論を踏まえて，理解を記述すると以下の側面がある。

①具体的な例を知っている

②該当の理論を知っている

③似たようなものから推測がつく

④より抽象的な上位概念を知っており，動作性質が分かっている

⑤当該物の構成が分かっており，部分に分けて理解できる

⑥当該製品のモデルが頭の中で構成できる

このうち，②の理論は大袈裟なもので無くても良いが，形式的に体系だったものであり，現状からの予測が利く必要がある。更に，具体例に関しても該当状況の説明ができること，その状況から次の予測ができること。そして予測が外れた場合にも，理由が説明できる程度には，理論的に知っておく必要がある。また，類似例からの類推は，良い見通しを与えて，全体として見えない状況に関して非常に効果があるが，細部においては誤差を生じる。この場合の検証にも定性的な理論による説明が必要である。

このように，形式的な理論は有効である。形式的な理論を作るためには，完全な体系まで行かないけれども種々の方法論がある。以下の節では，どのようにして形式的な理論を作成し，使用するかについて述べる。理論にも色々あるが，目的に合った理論の選択が重要であり，理論的用のためには，現実問題を上手に抽象化しその上で議論のできるモデル作成が重要である。

5-2抽象化の段階

概念の定義にも色々の抽象段階がある。例えば一人の三菱電機社員にも，××課員，××技術者，男，成人など色々の上位概念がある。一方，個人も種々の意味で変化する。例えば，1 日前には知らないことでも今日は知っているかもしれない。このように，一つのことの表現でも，色々な抽象化のレベルでの表現が可能である。このように，表現のレベルを工夫すれば，簡潔な表現が可能に成り理解を助けるが，抽象化しすぎて誤解を招くこともある。

この様な，段階的な抽象化を積極的に使用すると，どの様なシステムでも抽象化していくと，単純な目的を示すように成る。こうして，抽象のレベルを上げても『これは，何を

するものです』と言う表現を行えば，全体の見通しが良くなる。更に，一度簡単に表現したものがあると，その内容を具体化し詳細を逐次付加していくことが可能に成る。また，用語の定義に関しては，これは何の種類に属し，どのような部品でできている等の観点からの記述が必要であり，抽象度に関する意識が必要である。

通常，理論は高度に抽象化した理想的な世界で成立する。そのため，理論を現実の世界に適用する時には，一旦抽象化した表現を行った後，その上での適用可能性を検討するのも一つの方法である。例えば，CRT の画面に各種のシンボルを表示し，色を変えたりブリンクする場面を考える。これを，“人の注意を引く要素”と抽象化してみる。この場合は，“人間が一度に注意を払える項目は，７±2 である”と言う規則が出てくる。このように，一つの場面に関して適用できる理論を抽象度を上げながら探せことで深く理解できる。

なお，モデル作成に当たっては，特に登場する物の抽象度を同じにするように注意すべきである。例えば，人間一般と一人の人間の個性を，同列に論じるようなミスをしてはならない。

5-3類推による理解

類似のシステムと対応を考えると全体をつかむことが容易になる。相違点が少し位あっても全体の見通しや，動作の理解がある方が有効である。さらに，相違点を探すという風に，調査点を明確にし集中していくことは，最初の取りかかりから落ち着いた精神状態で作業ができる。

ただし，類推にはどこかで異なる危険性があると，認識の上作業する必要がある。類推の能力開発は，創造性開発の等価変換法の訓練が一番有効である。また，同じく創造性開発技法として有名な NM 法も，キーワードによる連想と言うことで，類推能力を磨くことができる。

さらに，システムエンジニアにとって，類推能力を必要とする場面は，例え話による説明である。提案内容の説明や，相手を説得する場合に，奇麗な理論式の組み合わせで説得するよりも，当人が具体的に理解している例え話引用すれば，効果が大きくなる。

例えば，あるコンピュータソフトウエアにおいて，デバッグ中にはエラーチェックの厳しいものを使用し，実運用時には性能向上のために，エラーチェックをはずしたものを使用したものを考えよう。ある人は言う。

『地上の訓練にはパラシュートを着用し，空に飛び上がったらパラシュートをはずす飛行士はどこにいる。』

反論はこのようになる。

『第二次大戦末期の日本の飛行機は，エンジンの能力が低く，B29 の飛んでいる高度まで上昇するのにも苦労し，少しでも軽くするために余分な物をはずした。だからパラシュートもつけられない。もっと安くて，性能の良い計算機を持ってきて，そんな贅沢を言ってくれ。』

この議論は，具体的なイメージが湧きやすいが，以下のような類推の危険性を含んでいる。

①第二次大戦中の日本の飛行機は，パラシュートを座席の一部に使っていたので，パラシュートを外すことはできない。

②軍国主義反対と言うことで反論が出て本論から脱線する。

①の様に例え側の間違いを指摘されて，論点が崩れてしまう。悪意を持った相手に，本論と例え話側の 2 つの攻撃点を見せてしまう。また，②の様に例え話に相手が興味を示さなかったり，悪意を持った場合には，本来の話にまで悪意が波及する可能性がある。特に，性的な話などは，受けることも多いが，反発も厳しい。

類推で，一番無難な対象は前からある同様の機械である。ハードウエアで行っていた機能をソフトウエアで実現しても，原理的に大きな変化があるとは限らない。また，技術の歴史を知っておくと，今回のシステムは従来のシステムの延長線上で見えてくる。このように，技術の歴史を見ていると，将来の予測もできるように成る。この様な技術予測は，一流の技術者の条件でもある。

5-4モデルの構築

一つのものを理解するために，その物の動きを頭の中でシミュレートできる，モデルがあると便利である。ここでは，技術的なモデル造りについて述べる。

技術的なモデルとは，どのようなものであろうか。一つの定義は，以下のように成る。

モデルとは，現実の実在物に対し，理論的や形状的などの何らかの類似性を有し，それを見たり動かしたりすることにより，技術者に知見を与えるもの。

これに加えて，システムエンジニアの使用するモデルには，以下の条件が加わってくる。

モデルには必要な情報が漏れなく入っていて，そのモデルだけで答えが出るようになっていること。言い換えると，電気的，機械的など色々上位にある理論をまとめて，必要な形で入れておく必要がある。さらに，外部の環境についても，必要に応じて取り込んである。

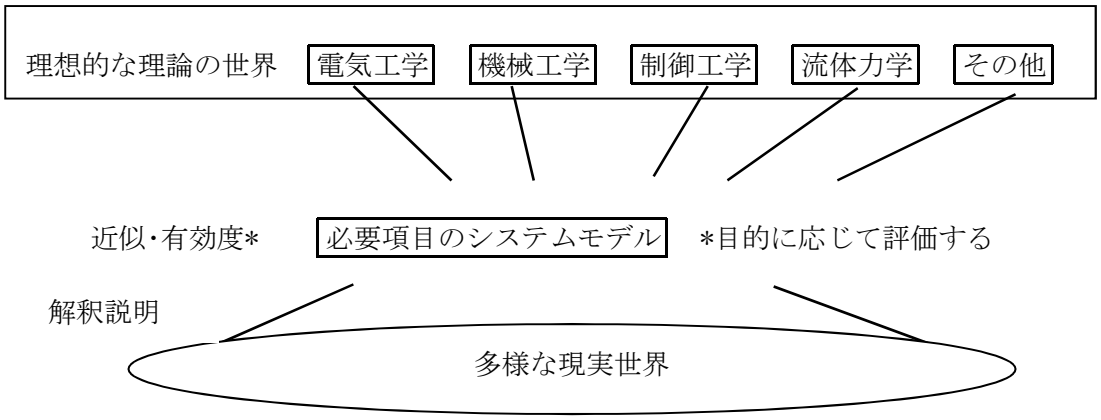


図 5.1　システムエンジニアの利用するモデル

ここで，外部の環境と言うのは，例えば発電機に関する制御のモデルを考えた場合に，送電の系統を含むようなものである。上位の理想的理論の役割は，モデルの動きを説明することであり，理論が現実の世界で役にたつようにするための一つの例として，モデルの形を取る。

ただし，現実の問題には，人間的要素が絡むが，人間に関する包括的な議論で，実際の

役に立つものは残念ながら現状は少ない。この様な場合には，できるだけ状況を区別し，理想的な記述により適応できるルールを，上位の理論の代用として使用する。

また，MN 法で使用する抽象的な概念も，理論の代用に使えることもある。この部分に関しては，まだまだ検討すべき事項が多い。別途述べた社会科学の理論構築法の利用も一つの可能性である。

以上を踏まえて，モデル構築の手順は以下のように成る。

① そのシステムの目的を明確にする。
② システムと環境の境界を明確にする
③ 記述する抽象のレベルを決める(個々の区別を含むか，一般的なもので代表するか等)
④ 構成要素を列挙する
⑤ 構成要素の機能を記述する　　上位理論を念頭において行う
⑥ 構成要素間の繋がりを確認する
⑦ システムの動作例を書いてみて，モデル上での動きに無駄･無理及び欠陥がないか確認する。問題点があれば，②から⑤に戻る。

ただし，上記の手順は一からモデルを作る場合であり，類推などで全貌のイメージがある場合には，いきなり⑦の動作例の記述から入って，修正する方が良い。この様なモデルは，計算機上のプログラムとして実現できれば，検証も容易で応用も広くなるが，単に頭の中で動かすだけでも，理解を深めるために有効である。

次に，モデルの特性と問題点について述べる。

(1)モデルの 2 面性

問題対応の情報を全て記述するモデルの実現方法にも，種々の方式がある。

図 5-1 の理論と現実の 2 方向で分けると，図 5-2 のように数式などで構成する理論的モデルと，模型などの現実的モデルになる。

両者の相違点は，現実的モデルにはモデル構成要素に，含み情報による広がりと曖昧さが残り，理論的モデルには現実に対する近似による情報の切り捨てが生じる点である。極端な言い方をすると，理論的モデルで使用する概念は，厳密な定義があるのでどのようなラベルをつけてもかまわない。さらに，関係者の個人的経験により左右される要素は，結果の解釈面以外では原則無い。

一方，現実的モデルの場合には，モデル構成要素の名前が重要な意味を含んでいる。このため，モデルの動作の理解・解釈にも個人の経験の深さに大きく左右される。

例えば，人間を含む監視制御システムのモデルでは，理論的モデルを考える場合には，操作する人間は，どのような条件でどの操作をするか，決められたルールを正確に記述すれば良い。(それしかできない)一方，現実的モデルの場合には，“原子力発電所の操作員”と言う具体的なラベルがつく。ここから，検討者は自分の知っている操作員のイメージから,,“厳しい訓練を受けた有能な操作員”や“操作した後の反応が遅いと苛々し何度も操作した操作員”などの今までの経験による知識を加味した検討を行う。

また，この様な含み情報を使用するためには，情報の取捨選択のための価値観が重要で

ある。この様な価値観は経験が育てる。また，常に誤りの可能性を含んで検討するので，マージンを十分に持った検討が必要である。

システム理解では理論と現実の間に相互の支持が入って，深い理解を得ることが理想である。しかしながら，両者の間には以下に示すような深いギャップがある。

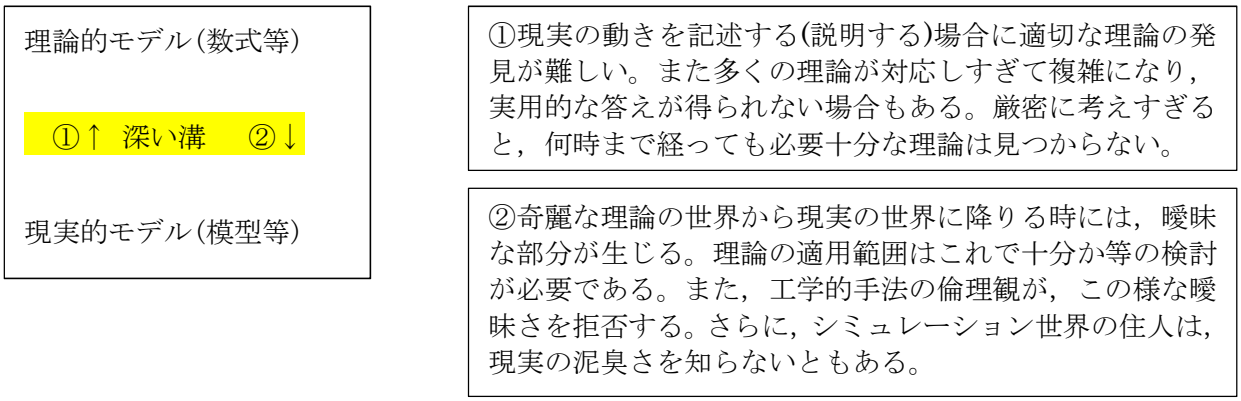
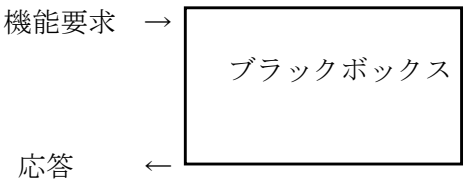


図 5-2 モデル世界の 2 方向

(2)モデルの不当な使用法

モデルの利用において，不十分な理解で誤る場合がある。これを大きく分けると，以下の 3 パターンに成る。

(2-1)ブラックボックス過剰

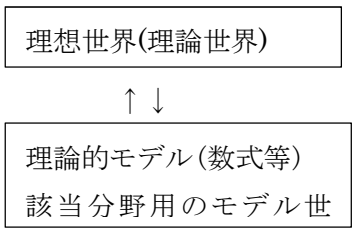


成熟した技術分野に多く見られる現象で，先人の成果を丸呑みにした結果等で発生する。通常の処理は支障が無いが，トラブル発生時の対応ができない場合が多い。
但し，全ての構成要素を理解することは無理であり，適度なブラックボックスは必要である。

図 5-3 ブラックボックス過剰

(2-2)地に足のつかない理解

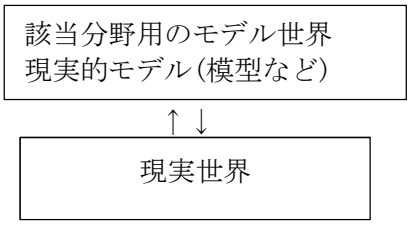
これは，現状の受験戦争に慣れた学生などに多い症状で，“試験に合格するだけしか理解していない”等の原因による。



理論世界だけしか見ず,式の意味が説明できないなどの症状である。また，一部のソフトウエア作業者等には，正解が決定できる領域しか考えない者もいる。

図 5-4 地に足のつかない世界

(2-3)経験則のみの場合



この場合は，経験則の適用限界が不明のため，適用過剰や不足になることが多い。また，他人に対する説明が上手にできないことが多い。
ハウツーもののマニュアルでもこの様な症状が生じる。この場合は，マニュアルから外れると応用が利かなくなる。

図 5-5 経験のみの場合

5-5複雑なシステムへの対処法

大規模複雑システムに対処するためには，大きく分けて以下の対処法がある。

①荒いモデルに単純化してまとめる。
　手法１　変数を一つにまとめる
　手法２　関連性の薄いものを無視する
②分割してその関係として把握する。
　手法１水平分割　同一レベルの部品の繋がりとして把握する。
　手法２垂直分割　上下に広がる階層関係による記述

これらの手法も，着手できるまでの切り口発見が難しい。経験的には，とりあえず一旦モデル化し，後から修正する方が早いことが多い。とにかく，簡単化し全体イメージを把握できるように持ち込むまでが重要である。
また，以下に示す切り口もある。
①物の流れ・エネルギーの流れ・情報の流れ に分割して表示する。
②正常生産物(製品)と付随生産物(廃棄物，熱，振動など)の分割
この場合，物とエネルギーには保存法則が成立する。一方情報に関しては，組み合わせ条件や外部からの影響を加味する必要がある。この部分が，情報処理の入ったシステムの難しい点である。

5-6難しい項目の勉強

これは，いわゆる勉強である。ただし，学校と異なることは，目的が明確と言うことである。例えば，数学でも“××を理解するための数学”となる。また，要求習熟段階も専門家の説明について行ける段階から，自分で新しいことを考え出すまで多様なレベルがある。ここで重要なことは，試験のための勉強ではなく自分の納得のための勉強である。数式に関しても，物理現象に関連して理解することが重要である。たとえば，div～は，ある一点から出る線と関連する等の理解が必要である。

この様な学習では，急がば回れで周辺知識を勉強してから，今の問題を解く方が速いことがある。時間は有限であるが，人間の頭の容量は実質上無限である。このように，色々

と使える知識を知っていると，多くの場合に応用が利き，さらに知識が増えるものである。知識は経験に完全に代えることはできない。しかし 1 の経験を 10 倍に膨らますことは知識の活用による応用力強化で可能である。この事を考えて，一歩踏み込んだ理解，一歩広げた知識吸収を行って欲しい。

6. プロジェクト管理

6-1システムエンジニア・プラントエンジニアとプロジェクト管理

システムエンジニアやプラントエンジニアは，大規模なシステムの計画･設計業務を行う。このため，多くの人と協力することが必要であり，彼らの作業をまとめるプロジェクト管理者の立場に立つことも多い。

設計的立場と管理者の立場には大きな違いがあり，この切り替えができないと『名選手必ずしも名監督ならず』となる。

管理者の立場は，当然ながら自分で作業ができないと言うことであり，以下の点に注意が必要である。

①自分の効率を犠牲にしても，全体として効率を上げる必要がある。

②作業者の能力を過信しては行けない。自分で行う場合には，限界に挑戦し能力を広げることは重要である。しかし，これを他人にまで期待しては行けない。

③上記と関連するが，平均的人材で運営できる負荷配分が重要である。

6-2管理の必要性

ここで，なぜ管理が必要か明確にしておく。管理は，電子回路に例をとればフィードバックループのようなもので，安定性を引き出すが最大出力より低い出力で，満足する必要がある。このように，何かを行うためには，資源ゼロではできず，コストアップの要因に成る。管理のための管理，惰性の管理では，損益の足を引っ張るばかりである。なぜこれを行うのか，このメリットは何かを常に考える必要がある。

ここで，管理の必要がない組織を考えてみよう。例えば，非常に優秀なエンジニアの集団で，仕事は周囲から無理なく与えられる。更に，開発環境・必要設備は各人に行き渡っている。彼らの作業は十分に速く，要求納期には確実に納入できる。また，作業に対する評価は十分高く，十分な支払いを受けとることができる。この様な状況では，各人が手が空き次第次の仕事を引き受ける，と言う形式で最低限の管理で業務遂行が可能に成る。

しかし，この場合でも各構成メンバーに能力差や得意分野がある場合には，最適業務割り当てのための管理が必要になる。また，見方を変えるとこの様な仕事の受け方は，各人の能力を最大限に引き出すのではなく，マージンを持った対応である。各人の能力の限界を見ながら活用することで，より大きな成果が期待できる。このためには，クリティカルな領域を，安定に運用するための管理が必要である。さらに，将来の業務展開・各人の能力開発と活用には，対極的な判断が必要であり，管理者としての腕の見せ所である。

ここで管理の定義を，明確にする。

プロジェクト管理とは，与えられた資源(金，人，物)を有効に活用して，行うべきことを決め(計画)，計画どおりの進捗を確認し差異発生時にはしかるべき対策を行い，併せて今後の改善に活用できる情報蓄積を行う作業である。

この作業中一番大切なことは，正しい計画である。計画段階での無理は，後工程で補うにも限界がある。

6-2計画

計画段階では，大きく分けて以下の情報を決定する。

①誰が何を行うかを決める体制

②いつまでに，何を行うかを決める工程

この中でも，“誰に何を行わせる”を決定することが一番重要である。『体制表ができれば50%は成功した』と言う極端な意見もある。逆に，『体制ミスは後で補いがつかない。体制ミスでプロジェクト崩れは決定した。』これも良く聞く話である。

体制表作成時の注意点は，平凡な言葉であるが“適材適所”である。特に，自分で担当する場合と異なり，他人や他社に依頼する場合に，少しの無理は頼めても“火事場の馬鹿力”を期待しては行けない。“少しの無理はしても，無駄な抵抗はしない”と言うレベルで，平均的人材が通常の努力で実現できる，プロジェクト運営が重要である。無理がとおっても道理も引っ込まない。

なお，平均的人材のできる仕事と言う発想は，物作り現場の発想である。これが，研究所の場合には，“先端技術での成果を得るために，優秀な人間が特別な条件で達成する，ピーク値を追求する”ことになる。たとえば，ソフトウエアで良く話題になる，オブジェクト指向の開発であるが，継承関係が入り組んだ場合に，上手に扱える能力を持った人間の数は限られている。個人の職人芸に頼っていては，製品としての物作りに支障をきたすことが多い。本来研究所の任務は可能性を示すことにあり，安定した物作りのための検討は別途行うことが必要である。

また，適材適所の中には個人の個性もある。ソフトウエアの場合で目に付くのは，“机上中心型”と“デバッグ中心型”である。じっくり考えて一発で決める性格と，小回りが利いて，小刻み修正が上手な性格の違いである。この様な差を微妙に工程表や体制表に反映させることが，より安定したプロジェクトの運営鍵である。ただし，デバッグに関しては，行わないことがベストと言う説もある。ＩＢＭのクリーンルーム方式は，製作完了時点から，完全と言う前提で，従来のハードウエアと同じ考えであり，ソフトウエアのみ特別扱いすることを，否定している。あまり，従来方式の押し付けは，混乱を招くが，ソフトウエアの極端な特別視も良くないと考える。

体制表や工程表に個人差まで加味するためには，常日頃の観察が重要である。特に教育訓練の反応を見ておくことは重要である。日本海海戦で，連合艦隊の東郷平八郎司令長官は，今までの訓練を通じて部下の技量を信じ，相手側の訓練量を見切って，同じ事はできないと確信を持ったから，敵前で隙きだらけになる，艦隊の転回ができたと言う説がある。これは，一般には“東郷司令長官の度胸の良さ”と伝わっているが，毎日弁当を持って砲術訓練を見た結果での，確信を持った行動と考えたい。決断力は，プロジェクト管理者の重要要素であるが，できるだけ裏付けを求めることが大切である。

また，第 2 次大戦中にマレー沖に，イギリスの新鋭戦艦が来たと慌てる幕僚に対して，山本司令長官が『××部隊には，最新鋭の一式陸上攻撃機を補充している。』と言った発言も，良い管理の一面を示している。トラブルが起こってから対応する人間よりも，事前に予知し，手を打っている人間を評価すべきである。

6-3実行段階

プロジェクト進行中は，動くべき時に動き，動くべきでない時には動かないことが重要である。このためには，進捗を正しく見る必要が有る。ただし，報告を作成することも作業者の負荷に成ることを忘れてはならない。進捗把握の手段は大きく分けて，以下の 3 方法になる。

①決められた成果物の完了(図面など)

②各種レビュー

③作業者の自己申告

この内，①，②荷関しては客観性が有るが，③の自己申告には客観性がない。このため，作業者の信頼度を計るためにもレビューが必要である。

また，複雑なシステムに対処するためには，設計段階での各種シミュレーションが重要である。これは，計算機上で実現するのみならず，設計者の頭の中やデザインレビューの席上で行われることもある。

また，デザインレビューによる設計ミスの発見及び設計者の納得のためには，デザインレビューを受ける側にも，それなりの理解力が必要である。言い換えると，少しぐらいの誤解や経験不足はレビューで補えるが，根本的な力不足は補えないと言うことである。

なお，工程が遅れた時の対応は，単純に人を増やすことは一番下手な対策である。事情の解らない人間の参入は，説明の手間を増し混乱に拍車をかけるだけである。この様な事態での戦力の逐次投入は結局，一番高コストとなる。能力のある人間を，短期に集中投入することが望ましい。最後に，最悪の事態で守るべき事項の優先順序は，以下のように成る。

①工事の利益，これは後で取り返しが利く

②顧客先の信用，これを取り返すには非常に長い期間が必要

③社内の組織間の信用，これがなくなれば組織でなくなる

6-4試験段階

この段階では，成果の読取りと不具合情報への対応が主要業務に成る。特にどの部分が原因か，不明な場合の担当部分の割り当てが重要業務となる。間違いでも良いから決断しなければ，前に進まない。但し，間違った時の早期修正も重要である。

もう一つ重要なことは，小手先の手直しによるもぐらたたきにならないように，大局的判断が必要と言うことである。とかく，試験中は視野が狭くなる。注意すべきである。

6-5完了後の後始末

失敗した工事はもちろん，成功した工事の理由も明確にすべきである。なぜ成功したか，この条件を明らかにしないと“柳の下の二匹目の泥鰌”を，いつまでも追い求めることに成る。逆に，偶然の助けによる不安定な成功でも，要因が明らかになれば次回は安定した成功に持ち込める可能性がある。また，協力会社が絡む場合は，双方のミスを明確にしておくことも重要である。このように当たり前のことを，積み重ねることが成功する管理の一要因である。

また，利益評価に関しては，単一工事での戦術的評価と，長期戦略的評価を明確に区別すべきである。よく，コスト面で工事の失敗が決まった後に，その工事の戦略的重要性が浮上する場合がある。これは，計画段階で明確にすべきであって，後づけで浮かぶ問題ではない。また，計画ミスを実行段階の戦術的努力でカバーした場合も，計画上のミスはミスとして明確にしておく必要がある。

なお，完了後の反省が不十分であった例は，日露戦争後の日本軍である。少数精鋭の艦隊決戦に凝り固まった海軍は，長期消耗戦に対応できなかった。また，陸軍は昭和でも馬で大砲を移動する計画であった。優秀機種の後継機種の開発・投入に失敗した管理者は，これを笑う資格はない。