



# ソリューションビジネス のセオリー

- ◆ 初版 : 2003年4月
- ◆ 発行所 : 同友館
- ◆ 単行本 : 206ページ

14年ほど前、あるコンピュータメーカーから「ハードウェア、ソフトウェアというビジネスからそれらを統合し、顧客の課題解決をサポートするソリューションビジネスを当社の基本戦略としたい。ついてはこれを支えるSEの意識改革が必要だ。そのための教育をやって欲しい。」という引合が私の所へ来ました。約半年間かけてこのコースウェア(教育カリキュラム)を開発し、トライアルセミナーを実施しました。このセミナーは「提案型SE養成講座」と名づけられ、おかげ様で好評を博し、そのコンピュータメーカーのSEに本格的に実施していくこととなりました。セミナー開始当初は、SEに「ソリューションとは何か。提案とは何か」を教えていましたが、やっていくうちに受講生から「精神論ばかりではなく、具体的にどのように提案を進めていけば良いのか、そのやり方を教えてくれ」というリクエストが多くなってきました。そこで約1年かけてソリューションビジネスの「やり方」を試行錯誤で定義していき、ソリューションモデルの第1バージョンが完成しました。このソリューションモデルという「やり方」は「提案型SE養成講座」のコースウェアに組み込まれ、また私自身もコンサルティングビジネスの中でこの「やり方」を使っていきました。

その後この「提案型SE養成講座」は他のITベンダーでも行うようになり、実に約4000人、対象企業は公開セミナーでの少人数参加を含めると約200社に及んでいます。そして、セミナーの名称も「SEのためのインタビュー&プレゼンテーション」「提案型営業研修」「ソリューション研修」「問題解決研修」などとさまざまな名称で企業の目的に合ったスタイルで実施されています。

その対象はコンピュータメーカー、ソフトハウス、システムディーラー、ネットワーク企業などITベンダーだけでなく、私自身も思いもよらなかった企業・人たちが受講されることとなりました。計測制御機器・電子機器のエンジニア、教育ベンダー・人材情報誌・金融機関をはじめとする多くのセールスパーソン、インストラクター、フランチャイズのスーパーバイザー、さらには私と同業のコンサルタント…。そしてソリューションモデルもこの10年間で何回もバージョンアップがくり返されていきました。

「提案型SE養成講座」を同一企業に何回か実施し、終了後、そのソリューションベンダーの課題などを報告書に書いていると「そんなえらそうなことを書くなら、具体的にどうしたらよいかを教えてくれ」といわれ、ソリューションのための組織作り、マーケティング戦略・経営戦略の立案、さらには提案型SEの選抜などのコンサルティングをやらせていただくようになりました。

本書はこの「提案型SE養成講座」やコンサルティングで、私及び弊社が行ってきたことを50のセオリーという形で体系的に整理したものです。そしてその目的はソリューションビジネスを早期に立ち上げ、効率的にビジネスを進めていくうえでの理論的バックボーン・ビジネスモデルとして活用することにあります。したがって、本書はソリューションビジネスを実施していく企業のSE、システムセールス、マネジャー、経営者を第1の対象としています。

さらに「提案型SE養成講座」において「ソリューションの考え方」「やり方」をITベンダー以外の多くの企業の方たちが活用されたのと同様に、この50のセオリーも「自社のお客様に問題解決方法を提案する」必要がある多くの方たちに活用することができると思っています。

ITをはじめとして、さまざまなソリューションビジネスを展開する企業の多くの方々に本書が読まれ、このセオリーが実践され、そしてその実践の中から新しいセオリーが誕生することを期待しています。

# 目次

## 序章

### 第1章 ソリューションビジネスに必要なもの

セオリー1:ソリューションビジネスはニーズイン・スペックアウト

セオリー2:ソリューションプロセスを6つに分ける

セオリー3:「プログラムとデータ」からオブジェクト指向へ

セオリー4:原価見積から投資金額提案へ

セオリー5:スケジュール表ではユーザー側作業を提案する

セオリー6:提案型SEに必要なものは知識・ノウハウ・経験

### 第2章 ニーズ予想モデル

セオリー7:「何か解決できるのか」からスタート

セオリー8:プロセス指向アプローチからデータ指向アプローチへ

セオリー9:タイプ別のニーズチェックリストを作る

セオリー10:データベースのニーズは「共有化」と「すて方」

セオリー11:常時接続はシステムを変える

セオリー12:インターネットはセキュリティ、Web技術、プロモーションの3つに分けて考える

セオリー13:万全なセキュリティはない

### 第3章 インタビュー・ニーズ発見モデル

セオリー14: ユーザーが「伝えたいニーズ」と「受け取ったニーズ」はちがっている

セオリー15: タイプ別にインタビュースタイルを考える

セオリー16: システムを考える前にインタビュー結果を整理する

セオリー17: ユーザーの言った通りに列挙する

セオリー18: グループリングは外部エンティティのクラスを作ること

セオリー19: 重みづけは「インタビューでのニュアンスの定量化」

### 第4章 ニーズ解決モデル

セオリー20: ソリューションスペックは稼働責任を負う

セオリー21: 案の選定では「なぜ選んだか」を用意する

セオリー22: 新システムのリスクは消えない

セオリー23: 基幹系・情報系・ネットワーク系に分けて提案する

セオリー24: データ分析はマクロからミクロへ犯人探し

セオリー25: 意思決定支援はシュミレーションモデルで

セオリー26: 非定型ニーズとリテラシーは逆行する

## 第5章 プレゼンテーションモデル

セオリー27: 提案書はSEが作り、説明する

セオリー28: ユーザニーズはまとめずに、そのまま書く

セオリー29: システムイメージとソリューションイメージを分ける

セオリー30: ドキュメントを提案書を中心にデータベース化する

セオリー31: 「読んでわかる提案書」を作る

セオリー32: 目的・ニーズ・スペックのベクトルが合っているか

セオリー33: プレゼンテーションでの質問を恐れるな

セオリー34: プレゼンテーションでの反対を恐れるな

## 第6章 プロジェクト管理モデル

セオリー35: 見積は過去の平均値

セオリー36: システム開発では部品流用する

セオリー37: システム開発は外注しない

セオリー38: 遅れないようにするのではなく、遅れたときのダメージを小さくする

セオリー39: エラーのないシステムはない

セオリー40: プロトタイプでスペックを凍結する

## 第7章 ソリューションを実施する仕組みづくり

セオリー41: SEのキャリアステップを明確にする

セオリー42: 提案型SEをプロジェクトリーダーにする

セオリー43: 資格試験を職種選定に使う

セオリー44: 提案型SEは研修で選ぶ

セオリー45: 提案型SEは企業をシステムとして理解する

セオリー46: 提案型SEはITをパターン化しておく

セオリー47: ケーススタディで戦略を立案する

セオリー48: ソフトハウスでは足りない職種を提案型SEでカバーする

セオリー49: ハード販売企業では提案型SEに提案させる

セオリー50: 異業種参入では提案型SEを育てる

## 序章 数字を使って感性力を高める

### トータルセオリー

ソリューションビジネスは「考え方」「やり方」を従業員に提示することからスタートする。

## コンピュータのビジネスへの活用

コンピュータは人よりも速くCompute(計算する)ために誕生した機械であり、数学者が円周率を最後まで計算したいという夢が生んだものです。コンピュータに計算させるには「計算の手順」を覚えさせることが必要であり、メモリーが生まれました。商用コンピュータの父、ジョン・フォン・ノイマンはこのメモリーに計算手順の代わりに仕事のやり方(=プログラム)を覚えさせれば、人間の代わりに色々な仕事をやってくれることに気づきました。そしてこれをIBM社が製品として完成させ、コンピュータは大学の研究室のおもちゃ(失礼！)から企業のビジネスツールへと変身を遂げました。

## プログラマー・SEの誕生

コンピュータは人間ほど頭が良くなく、英語や日本語のように柔軟性の高い言葉が通じません。COBOLなどのコンピュータ語という文法のしっかりとした言語を使う必要があります、コンピュータに仕事のやり方を教えるには人間がこのコンピュータ語を覚えなくてはなりませんでした。成人してから新しい言語を覚えるつらさは多くの方がご存知のとおりです。そこで人間の中の代表者がコンピュータ語を覚え、一般の人は彼に通訳してもらうという形で仕事のコンピュータ化は進められました。この代表者がプログラマー(プログラムを作る人という意味)であり、実際にコンピュータを使う人はユーザーと呼ばれるようになりました。コンピュータは人間の仕事をどんどん覚え、そのためコンピュータはどんどん売れて、プログラムはたくさん作られていきました。

プログラムは人間のようにコミュニケーションがとれないので、次第にプログラム同士がうまく連携をとれず、混乱するようになっていきました。プログラムとプログラムの関係、つまり仕事と仕事の間をきちんとしておくことが必要だと気づき、この関係をしっかりと考えることを「システム」と名づけました。こうして個々のプログラムを作る前に、システムをデザイン(設計)する人としてシステムエンジニア(本書ではSEと略す)が誕生しました。そしてこのSE、プログラマーを中心として企業のシステム・プログラムを作り、受託していくビジネスが生まれ、「システム開発」と呼ばれました。

一方、ユーザー側にもプログラムとプログラムの関係、つまり仕事と仕事の間を調整するセクションが必要となり、情報システム部が誕生しました。システム開発ビジネスを行なうSE、プログラマーから考えると彼らが顧客であり、彼ら情報システム部のことをユーザーと呼び、実際のシステム利用者はエンドユーザーと呼ばれるようになりました。

## ITの誕生

コンピュータが次々と仕事を覚えていくうちに、仕事(プログラム)をするために使ったデータが、別の仕事に活かせるのではというアイデアが生まれてきました。こうしてデータベース(他の仕事に使うためにデータを残しておく)、ネットワーク(仕事と仕事をつなぐ)という考え方が生まれ、これを実現する製品が数多く誕生してきました。従来のハードウェア(コンピュータ自身)、ソフトウェア(プログラム)の他に、これらデータベース、ネットワークのハードウェア、ソフトウェア、さらには新しいサービスなどが生まれ、これらは総称してIT(Information Technology、情報技術)と呼ぶようになりました。またITに関するビジネスを行なう企業はITベンダー(ITを届けるという意味)と呼ばれるようになりました。

## バブルの崩壊

日本では1980年代後半からのバブル期にシステム開発ビジネスがピークを迎え、大量のSE、プログラマーがITベンダーに採用、育成されていきました。この時代まではSEの下にプログラマーがつくという形でチームが組まれており、プログラマーには「格下」というイメージがありました。しかしプログラマーを大量に採用しなくてはならなくなり、イメージアップを図ることが必要となりました。そこでプログラマーもSEという名（おしゃれな？）で、採用されることになり、プログラマーという職種が消滅していきました。

そしてバブルは崩壊し、日本中の企業のシステム化は一段落を迎えることとなりました。

一方ITベンダーは大量のSEを抱える中で、システム開発ピークが終わり、さらにパソコンという低価格、低利益の商品の普及で、急速に業績を悪化させていきました。

## ソリューションビジネスの誕生

ITバブルの崩壊を迎え、ITベンダーは新しいビジネスの創造に迫られました。プログラム→システム→データベース&ネットワークと高付加価値化を続けてきたITベンダーの出した結論は各社共通のものでした。それは、SEという優秀な設計者をベースとして、さらなる高付加価値サービスを行なうソリューションビジネス(課題解決を仕事にする)でした。

ソリューションビジネスとは、従来のように顧客が今やっている仕事を「早く正確に合理的に」することではなく、これらを包含し、顧客の悩みであるさまざまな「課題」に対して、ITによる解決策を提案し、その解決を支援していこうというものです。ITという製品を売る製造業から、ITを使ったサービス業への変身を求めたわけです。このサービス業に変身するには従来のハードウェア、ソフトウェア、データベース、ネットワークなどITの各要素を統合する必要があり、この統合サービスのことをシステムインテグレーション(SIと略す)といいます。ソリューションビジネスはSIという従来のサービスの統合に、「解決策の提案」という新しいサービスを付加したものともいえます。

このソリューションビジネスを行なう企業を本書ではソリューションベンダーと呼びます。ソリューションベンダーはソリューションビジネスを実現するために、SEに「ユーザーの言われたとおりに仕事を正確に早くやる」というスタンスから、「ユーザーの課題の解決策を提案する」というスタンスへの変身を求めました。このSEを本書では提案型SEと呼びます。

## ソリューションビジネスはなぜうまくいかないか

ソリューションビジネスを標榜してからすでに10数年立ちますが、ITベンダーからソリューションベンダーへの華麗なる変身を遂げた企業は極めて少ないといえます。逆にそのソリューションマーケットの魅力とベンダーの脆弱性を見て、異業種からソリューションビジネスへの参入が繰り返されているのが現実です。

なぜソリューションビジネスはうまくいかないのでしょうか。答えははっきりしています。そのビジネスを担うべきSEがビジネス変革についてこれないからです。製造業であるITベンダーにとっては製品力がすべてといえますが、サービス業であるソリューションベンダーにとっては「人」がすべてです。

なぜSEはこのビジネス変革についてこれないのでしょうか。従来のプログラム→システム→データベース&ネットワークという変革にはほとんどのSEがついてこれました。それはシステム、データベース&ネットワークについてはきちんと理論化されており、その教育を受け、やり方を知り、その理解を実践していくことで変革が実行できたからです。ソリューションビジネスにはこの理論がありません。世にあるのは要求分析手法(ユーザーの要求を整理する)、データモデリング(データの現実、あるべき姿をモデル化する)などであり、そのほとんどが「図の書き方」です。これらの手法は何のために図を書くのか、書いてその後どうするのか、次の設計作業にどうつながっていくかということにはほとんど無頓着です。これでSEに手法を使えというのは乱暴であり、まじめなSEほど疑問を持ちながら不思議な図やフローを書き捨てています。

## ソリューションビジネスを理論化する

本書はソリューションビジネスの理論化、つまりセオリーを提示することを目的としています。このセオリーは、まえがきにも述べた「提案型SE養成講座」というセミナーと、筆者及び筆者の会社でコンサルティングしてきた実践がベースとなっています。「提案型SE養成講座」には約4000人の受講者があり、公開セミナー(企業単位ではなく個人単位で受講)に参加した企業を含めると約200社が受講しています。そしてこのセミナーをベースにコンサルティングした企業は約40社にのぼります。

本書のセオリーは、このセミナー、コンサルティングを実施し、それをSEが実践し、うまくいかないと修正し・・・と繰り返していった結果をまとめたものです。つまり実践の理論化であり机上の空論ではありません。

これを読まれるSEの方々にはソリューションビジネスがどのようなものかを理解していただき、自らがそのセオリーをどのように実践していけばいいのかを考えて欲しいと思います。もちろんセオリーどおりに実践しても、SE個人の力量によって結果は全く異なるものになります。しかしセオリーを知らずにデタラメにやる位なら、やらないほうがかえって良いといえます。

またこれを読まれるソリューションベンダーの経営者、管理者の方々には自企業を変身させるためのヒントにしていただき、本書をベースとして自企業のセオリーを確立して欲しいと思います。さらに本書を読まれるシステムユーザー、つまり一般企業の方々にはソリューションビジネスのねらいをよく理解し、上手にソリューションベンダーを使い、自社のシステムをより良いものにして欲しいと思います。

## セオリーの構成

本書はソリューションビジネスに必要な50のセオリーが7つの章にわたって書かれており、以下のような構成になっています。

## ソリューションビジネスの考え方

第2章 ソリューションビジネスに必要なもの



## ソリューションビジネスのやり方

第2章 ニーズ予想モデル



第3章 インタビューモデル・ニーズ発見モデル



第4章 ニーズ解決モデル



第5章 プレゼンテーションモデル



第6章 プロジェクト管理モデル



## ソリューションビジネスのシステム化

第7章 ソリューションを実施する仕組みづくり

第2章～第6章のやり方が中核であり、本書ではこれをソリューションモデルと呼んでいます。第1章はそのソリューションモデルの考え方であり、第7章は実際にソリューションベンダーがモデルを実施していく方法が書いてあります。ソリューションモデルについては先ほど述べたように約200社で適用されており、まずはそのまま適用していくことをお勧めします。そのうえで自社のユニークなやり方で他社と差別化していくべきだと思います。

## セオリーの使い方

各セオリーは「セオリー」「ソリューションイメージ」「ワンポイントアドバイス」「本文」から成り立っています。ソリューションイメージは各セオリーの考え方、ポイント、実例を図示しており、それを本文で説明するという形をとっています。「ワンポイントアドバイス」はソリューションビジネスをセオリーに沿って進めるうえで、陥りやすいミス、留意点が簡潔に書かれています。

一通りセオリー1から50までを順に読んでいただき、実際にソリューションビジネスを実施しながら使うときは、局面、局面でセオリーを目次、検索キーとして使って下さい。またソリューションイメージにはワークシートや提案書のサンプルなども入っていますので、参考にしてください。

ソリューションビジネスはビッグマーケットであるのに、各企業が思いつきで仕事をやっているユニークな世界です。早くビジネスの考え方をSE、セールス、管理者に徹底し、やり方を提示し、1つでも多くの実績を積んだ企業がリーディングカンパニーになると思います。本書がその考え方、やり方の標準化のために少しでもお役に立てれば幸いです。

## 第1章 ソリューションビジネスに必要なもの

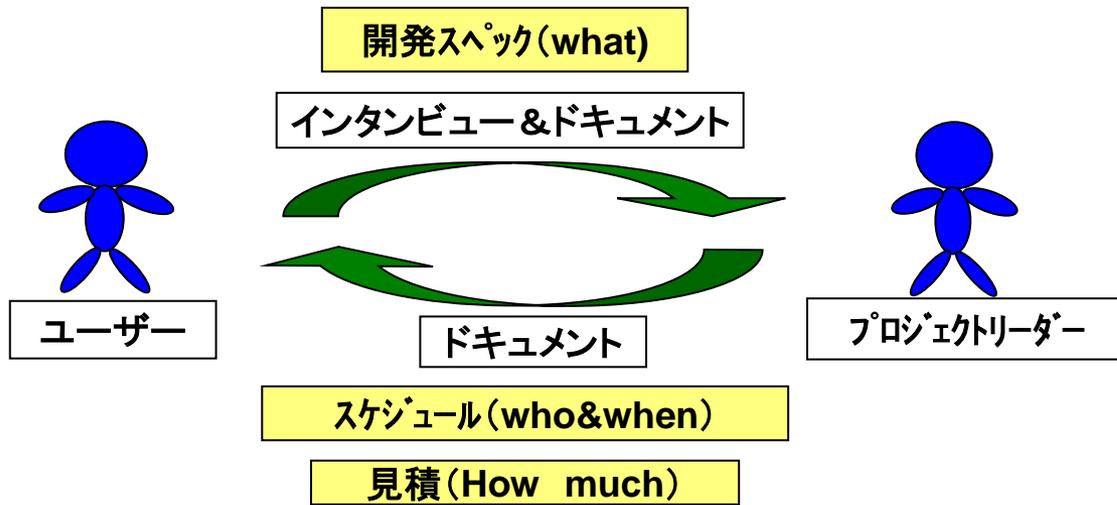
### トータルセオリー

ソリューションビジネスに必要なものは提案型SEの知識・ノウハウ・経験。知識と経験はナレッジマネジメントで身につける。ノウハウはソリューションモデルを提示し、体で覚える。

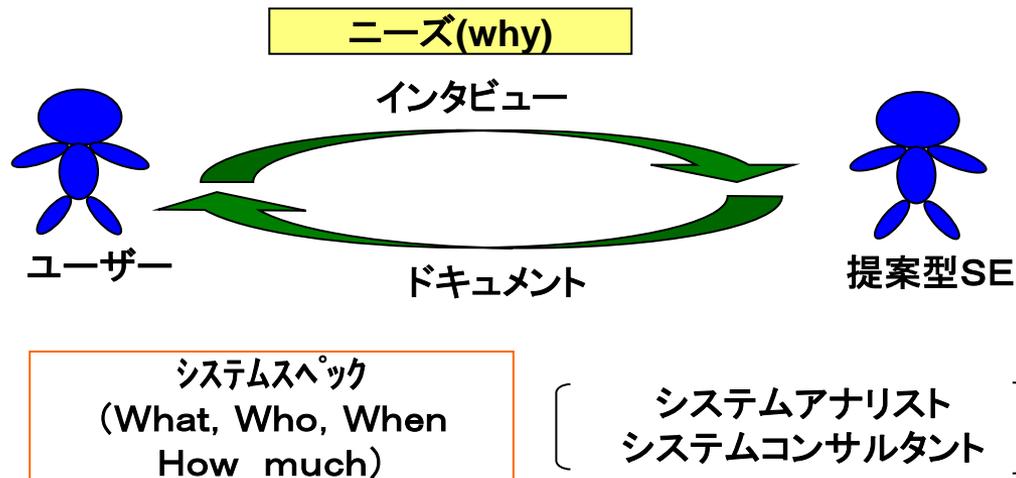
# THEORY1

ソリューションビジネスは  
ニーズイン・スペックアウト

# システム開発ビジネス



# ソリューションビジネス



同じ仕事をしているので本書では提案型SEと呼ぶ

システム開発ビジネスとは「そのシステムを使う人」つまりユーザーが、システムを開発することを業としているITベンダーに、開発を委託するビジネスです。そしてこのシステム開発に携わっている人たちをシステムエンジニア(SE)といいます。システム開発の出発点は「どんなシステムを作るのか、何を作るのか」(これを開発スペックといいます。Whatです。)であり、これをユーザーがまとめてITベンダーに伝えます。ITベンダーはまずそのシステム開発を行なうプロジェクトチームのリーダーSEを先に選任し(システムセールスが代行することもあります)、彼がこの開発スペックをヒアリングするか、開発スペックの書かれたドキュメントを受け取ります。プロジェクトリーダーはこの開発スペックをもとに必要なマンパワー(開発するために必要な人手)を考え、自社の人員・外注などの空き状況を加味して(Who)、工程表(When)を引きます。次にこれをベースとして開発するための原価見積(How much)を行ないます。さらに顧客の予算、希望納期と調整し、ITベンダーとしての最終的な見積・納期の提示をユーザーに行ないます。ここでユーザーと合意ができるとプロジェクトが結成され、システム開発に入っていきます。

①システムが開発スペックどおり作られているか⇒「品質管理」という。

②納期が間に合うか⇒「工程管理」という。

③原価見積どおりのコストで作られたか⇒「見積管理」(原価管理)という。

プロジェクトリーダーにとって最大の興味は②の納期です。①では仮にシステムトラブルが起きたとしても多くの場合、システム開発が悪いというよりも開発スペックに問題があるケースがほとんどであり、そうするとユーザー側の責任ともいえます。③の原価見積は最初から高くしておけば、それほど恐くありません。しかし②はどんなに余裕のある納期でも遅れるときは遅れます。開発スペックどおり作られているかどうかさえも見られない「動かない状態」が納期遅延です。プロジェクトリーダーのほとんどすべてが「納期恐怖症」です。

ソリューションビジネスでは同じく相手はユーザーですが、ベンダーは開発スペックではなく、「ニーズ」を聞くものを言います。ニーズとは「開発スペック＝What＝何を作るか」ではなく、「ニーズ＝Why＝なぜ作るのか」です。ニーズを聞いて、システムスペックを提案します。システムスペックとは先ほどのすべての要素、つまり開発スペック(What)、スケジュール(Who&When)、見積(How much)のことです。このシステムスペックの合意がユーザー側と終了すると、システム開発(これから作る)またはシステム組立(すでにあるシステムを組立てる。本書では以降これも含めてシステム開発といいます)を行ないません。本書ではニーズイン・スペックアウトとシステム開発をセットで行なう企業のことをソリューションベンダーと呼びます。

このニーズイン・スペックアウトという仕事は以下の理由で多くの場合無料で行われています。この時これを「提案」といいます。

- ・ソリューションビジネスがビジネスとして社会に認知されておらず、システム提案がシステム販売のためのプロモーションだと見られている。
- ・今までのシステム開発で似たような行為を無料で行ってきた。
- ・ソリューションベンダーがこの仕事にプロとしての自信がないので、顧客に有料だといえない。

この提案を誰がやるかと考えると答えは1つしかありません。それは提案の後にシステム開発作業(正確にいうと設計・開発)を担当するSEの責任者、つまり先ほどのプロジェクトリーダーであるSEです。もしシステムセールス(以降セールスといいます)がこれを行なえば世界一見ばえがよく、もしかしたら動かないかもしれないシステムを提案し、「後はSEによろしく」となってしまいます。SEにこの提案を行なわせると、これを提案型SEといいます。

ところが提案型SEはなかなかうまくいきません。その理由は次の2つです。

- ①提案型SEが「提案」を本職ではなくアルバイトだと思い、本職はシステム開発だと思っている。
- ②ユーザーが提案に金を払っていないので、提案に対して正当な評価(特にクレーム)をしてくれない。したがって提案型SEの提案技術が上がらない。

- ・ソリューションビジネスがビジネスとして社会に認知されておらず、システム提案がシステム販売のためのプロモーションだと見られている。

- ・これまでのシステム開発で似たような行為を無料で行ってきた。

- ・ソリューションベンダーがこの仕事にプロとしての自信がないので、顧客に有料だといえない。

この提案を誰がやるかと考えると答えは1つしかありません。それは提案の後にシステム開発作業(正確にいうと設計・開発)を担当するSEの責任者、つまり先ほどのプロジェクトリーダーであるSEです。もしシステムセールス(以降セールスといいます)がこれを行なえば世界一見ばえがよく、もしかしたら動かないかもしれないシステムを提案し、「後はSEによろしく」となってしまいます。SEにこの提案を行なわせると、これを提案型SEといいます。

ところが提案型SEはなかなかうまくいきません。その理由は次の2つです。

①提案型SEが「提案」を本職ではなくアルバイトだと思い、本職はシステム開発だと思っている。

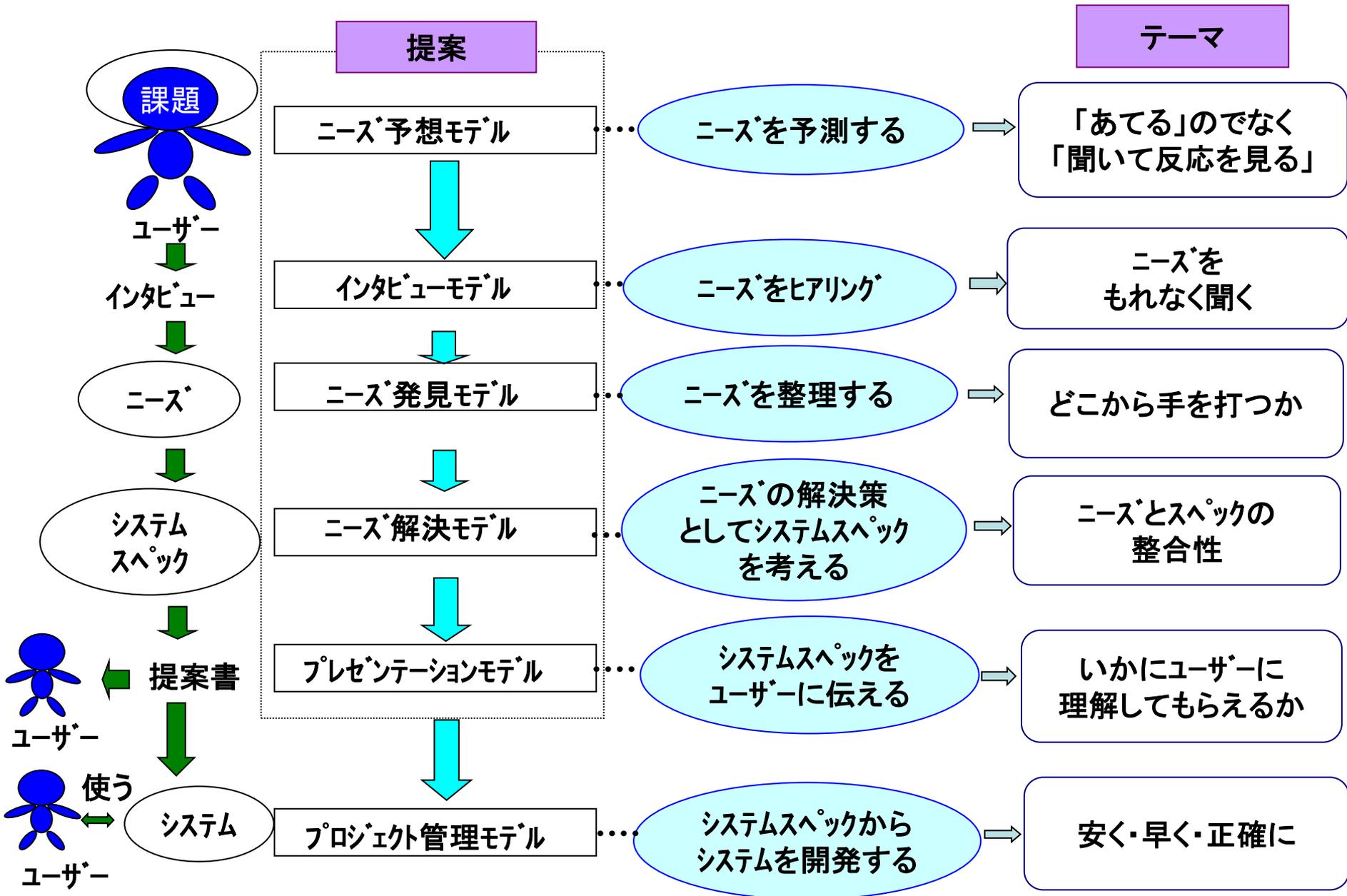
②ユーザーが提案に金を払っていないので、提案に対して正当な評価(特にクレーム)をしてくれない。したがって提案型SEの提案技術が上がらない。

①の問題点をクリアするために、多くのソリューションベンダーは提案型SEを「システムアナリスト」(SAと略す)などと称して、SEにこの提案が本職だと意識づけようとします。SAに「君の仕事はシステム開発でなく、システム提案だ」とわかってもらいます。そのうえで、SAをこの提案という仕事だけで人事評価するように努力します。しかし、SAが次の工程であるシステム開発にノータッチというわけにはいかず、結局自分が提案したシステムは自分がプロジェクトリーダーとなって開発し、最後まで責任をとる以外に方法がありません。つまりSAもSEの仕事をやらざるを得ず、提案型SEとなっていきます。

②を解消するには提案を有料にするしかありません。この時提案はシステムコンサルティング、または基本設計と呼ばれ実施者は「システムコンサルタント」(ITコーディネーターもこれに近い線をねらっている)とよばれます。しかし、ユーザーは昨日まで無料の提案というサービスになかなか金を払ってくれません。そこで多くのベンダーは社名を変更します(××ソフトウェア、××システムなどを、意味が良くわからないアルファベット3文字に変更したり、××研究所、××ソリューションなどそれらしくする)。そして我社はシステム開発ではなく、システムコンサルティングを行い、これを含めたシステムを提供するソリューションベンダーだということを社外、社内に訴えるようにします。しかし不思議なことに社外はその会社をソリューションベンダーと認知しても、社内の人、特にSEは自社がソリューションベンダーに変わったことを認めず、相変わらずシステム開発会社と思い込んでいます。これを変えていくのが、残り49のセオリーといえます。

# THEORY2

ソリューションプロセスを6つに分ける



システム開発は素人にはできないことは誰でも理解できます。このシステム開発の「やり方」を持っているのがITベンダーといえます。しかしシステム提案は素人、つまり提案をやったことがなくても「見よう見まね」で、「自己流」で何となくできるような気がしてしまいます。冷静に考えれば、提案を行うソリューションビジネスのほうがシステム開発ビジネスよりも高付加価値であり、難しい仕事だということはわかります。ソリューションビジネスにおける提案工程はシステム開発工程よりも上流工程(先にやる仕事。システム開発は一時製造業を目指していたので1つ1つの段取りを工程といいます)であり、上流工程が常に下流工程(後でやる仕事)の責任をとるのが常識だということも理解できると思います。例えば、システム設計→システム開発→システム運用と進めていくのであればシステム設計工程はシステム開発工程、システム運用工程の上流工程であり、開発・運用工程の責任をとります。つまり開発工程で何かあっても設計をした人がその問題を解決すべきといえます。ソリューションビジネスは提案→設計→開発→運用と進めていくのですから、提案した人が設計・開発・運用というすべての工程の責任を取るべきです。

これほど大切な仕事であるにも関わらず、多くのソリューションベンダーは「やり方」を決めずに、そして従業員に教えずにやろうとします。「SEという優秀な技術者にやらせればできるだろう」という安易な気持ちです。どんなに優秀な人でも「やり方」を知らずに新しい仕事はできません。さらにやり方を各自が勝手に考えてやっていると、やり方を考える時間、そしてその仕事に熟練するまでの時間が膨大なものとなってしまいます。仮にそれがうまくできるようになっても人に教えることはできませんし、同一企業でありながら、あまりにも仕事の品質が人によって違いすぎます。

ソリューションビジネスを行うすべての企業は、世に「ソリューションビジネスを始めます」という訴えをする前に、ビジネスのやり方を定義し、社内で習知し、訓練をして、そのビジネスのプロを育成すべきです。そうでなければソリューションビジネスを受けるユーザー側があまりにもかわいそうです。このやり方が本書でいうソリューションモデルです。

ソリューションビジネスは新しく付加された「提案」と従来からあるシステム開発（設計・開発）という2つのフェーズに分けることができます。後者のシステム開発のやり方は1970年代から研究され、ソフトウェア工学という形でアメリカでまとめられ、日本では「プロジェクト管理手法」という形で広く知られています。

一方提案の方はセオリー1で述べたとおり、無料でやってきたこともあり、ビジネスつまり商品なのか、それともシステム開発という商品を売るためのプロモーションなのかがはっきりしないままです。

提案はソリューションイメージにあるように、5局面(システム開発のプロジェクト管理を含めれば6つ)に分けることができ、その局面ごとにやり方を定義することが必要です。医者、弁護士などクライアントにインタビューし、ソリューションしていく他のビジネスではこのやり方が確立され、モデル化されています。やり方を工夫している医者、弁護士などいません。やり方が同じでも結果は異なり、名医とヤブ医者が存在します。提案の第一歩はユーザーのニーズを予想すること(ニーズ予想モデル)です。このモデルの目的は競馬の予想のようにニーズを「あてる」ことではありません。予想したニーズをユーザーにぶつけてみて反応を見るというものです。「こんなニーズを抱えていませんか」というスタンスです。医者でいえば病気を予測し、病気別に「ノドは痛くありませんか」という質問を数多く用意することです。

次がインタビューです。どうもSEはこれが嫌いで(人と話すことが好きな人がSEになることはまずないといえます)、提案をやりたくない人が多いようです。このモデルの目的はニーズを「もれ」なく聞くことです。ここでの「もれ」がシステムを作ってから判明すると致命傷となります。これがインタビューモデルです。医者でいえば病状をすべてもれなく聞くということです。手術をしてから新しい病状を聞いても遅すぎます。

インタビューが終わったら、いきなりシステムスペックを考えずに一旦これを整理すべきです。ユーザーの真のニーズを発見するという意味でニーズ発見モデルと呼びます。実際にやってみるとこの整理・発見工程が提案の中でいかに大切かがわかります。ユーザーがシステム提案を受け入れないケースは、ニーズ(why: インタビューで話したこと)の整理ができていないため「何のためのシステムか」がわからなくなっていることがほとんどです。自分の気持ち・ねらいがうまく伝わらないSE、企業にシステムを頼むのが怖いのです。本書では、インタビューモデルと合わせて解説します。

整理が終わると「ニーズを解決するシステムスペック」を考えます。このモデルで大切なことは世界一良いシステムを作るのではなく「ニーズを解決するためにはどういうシステムスペックがベストか」という視点です。これがニーズ解決モデルです。この段階にくるとほとんどのSEはニーズよりもシステムの作りやすさ、開発後の操作性ばかり気にします。医者でいえばどのような治療法を取るかという局面であり、治療しやすい方法でなく病状にあった治療法が大切なことはわかると思います。

システムスペックが固まったらドキュメントにまとめます。このドキュメントは提案書、企画書、基本設計書(本書ではすべて提案書と呼びます)などと呼ばれます。この提案書の書き方、説明の仕方の2つをプレゼンテーションモデルと呼びます。医者でいえば患者にいかに病状・治療法を告知するかです。

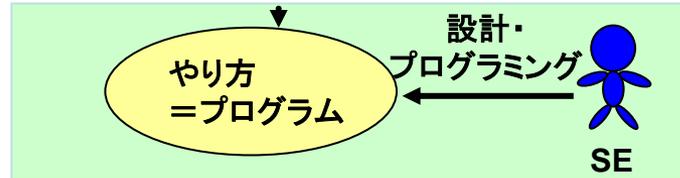
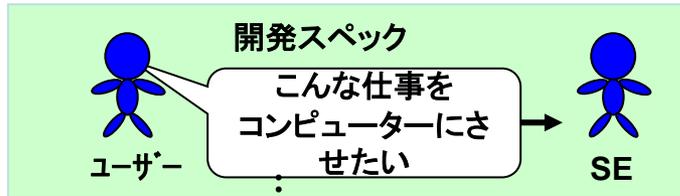
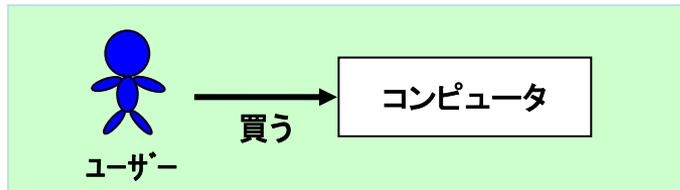
提案書の合意が得られれば、システム開発に入ります。システム開発はプロジェクトチームで進められ、一般に提案型SEがそのリーダーとなります。ここでのテーマは安く、早く、正確に作ることです。このやり方をプロジェクト管理モデルと呼びます。医者でいえば手術の実行です。

医者はこれらのやり方を確立することで専門職としてのステータスを確立したといえます。ソリューションビジネスもやり方の確立がそのステータスを確立し、SEが「やりたい」と思う職種になっていくと思います。

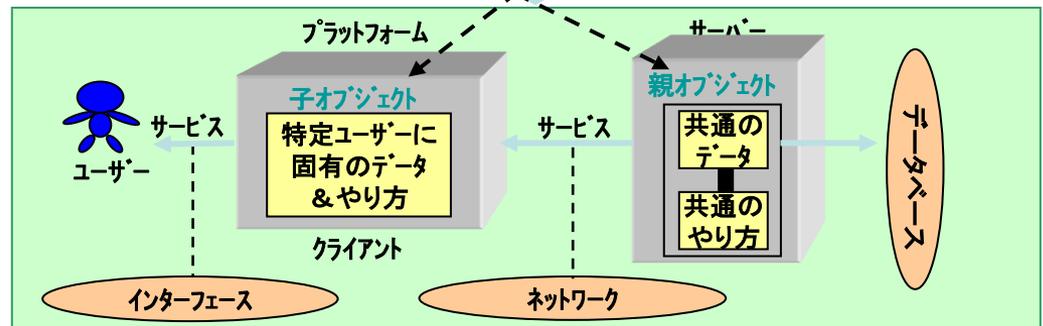
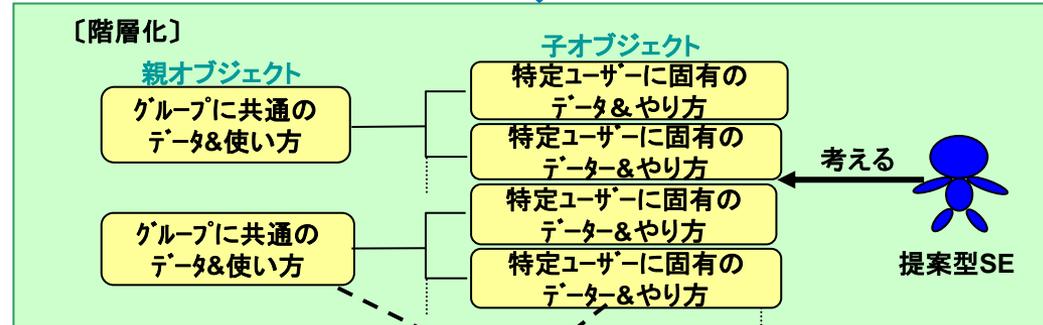
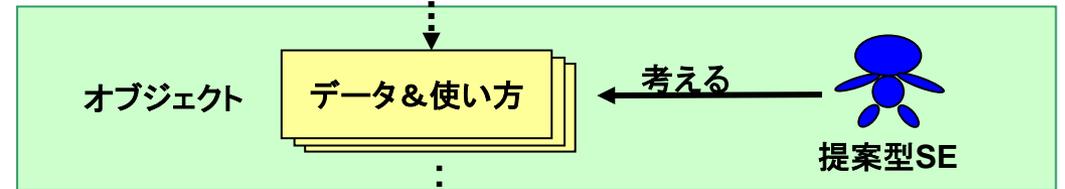
# THEORY3

「プログラムデータ」から  
オブジェクト指向へ

# 従来のシステム開発



# 従来のシステム開発



序章で述べたように従来のシステム開発ビジネスでは、まずユーザーがコンピュータを買い、どんな仕事をやらせるかを考えます。やらせる事が決まると、その仕事のやり方をSEが聞き(これがセオリー1の開発スベック)、そのやり方をコンピュータにコンピュータ語で教えます。このコンピュータ語の仕事のやり方を「プログラム」とよびます。プログラムが出来るとユーザーはそのプログラムを使って必要なデータをコンピュータに覚えさせ、コンピュータはこのプログラムとデータを使って仕事を早く正確にやっています。

ソリューションビジネスではこの「プログラムを教える」という発想を変える必要があります。セオリー2で述べたようにまず提案型SEがユーザーからニーズ、つまり解決してほしい課題をインタビューします。ニーズは「コンピュータに早く、正確に仕事をやらせる」ことによって解決される課題だけでなく、これらを含めてもっと幅広く「コンピュータが人にデータを提供する」ことによって解決できる課題まで広げます。そして提案型SEはこのニーズを解決するために必要な「データとそのデータの使い方」を考えていくようにします。この「データと使い方のセット」のことをオブジェクトといいます。

一般にユーザーは大勢いますので、ユーザーごとにそれぞれ必要なオブジェクトを決める必要があります。しかしこれでは大変なので似たようなオブジェクトはグループ핑します。グループ핑したら、グループの中で共通のデータ・やり方は抜き出して、これを親オブジェクト(スーパークラスという)とします。一方残された特定のユーザー固有のデータ・やり方を子オブジェクト(サブクラスという)とします。子オブジェクトは必要に応じて親オブジェクトのデータ・やり方を利用するよう考えます(これをインヘリタンスという)。こうすれば共通のデータ、やり方を何度も作る手間が省けます。

次にこの子オブジェクト、親オブジェクトを乗せる器(乗るという意味でプラットフォームという)を考える必要があります。従来のようにハードウェア(コンピュータ本体、プリンタ、...)にソフトウェア(基本ソフト、アプリケーションソフト、...)を乗せるということではなく、プラットフォームはパソコン(ハードウェアだけでなく、Windowsなどの基本ソフト、エクセル、ワード、アクセスなどのソフトウェアを含めて)、モバイル端末、携帯電話といったシステム製品を対象とします。

子オブジェクトが乗ったプラットフォームをクライアント、親オブジェクトが乗ったプラットフォームをサーバーといいます。子オブジェクト、つまりクライアント(お客様という意味)が、親オブジェクトつまりサーバーの「データと使い方」のサービスを受けるという意味です。親オブジェクトの「共通のデータと共通のやり方」のセットサービスのことをデータベースといいます。またWWWサーバー(インターネットを使う「やり方」だけをサービス)のようにやり方のみをサービスするサーバーもあります。クライアントとサーバー間は「つながっている」必要があります、これをネットワークといいます。

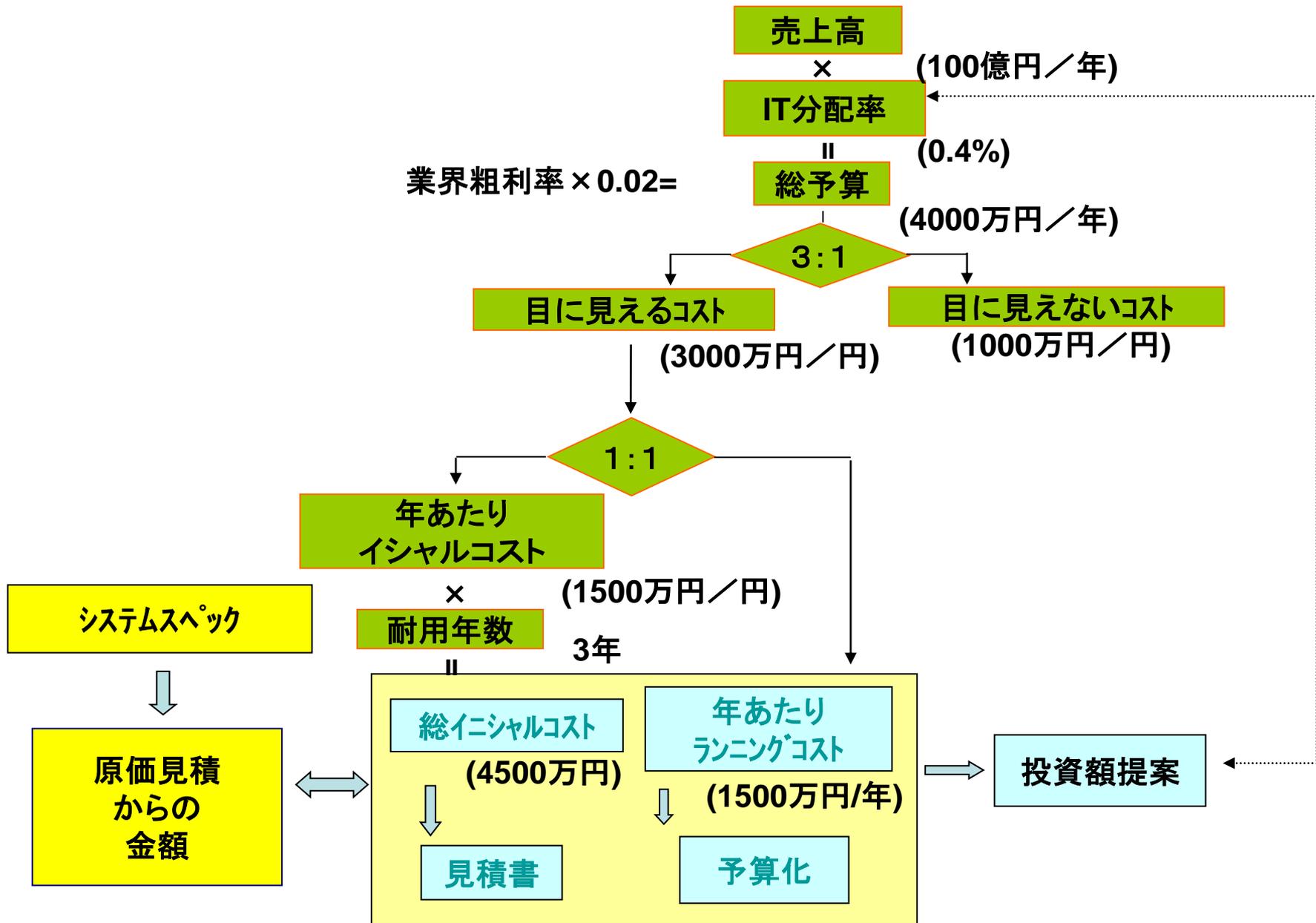
こうしてユーザーはニーズを解決するために、プラットフォームを通してオブジェクトから「データを使う」というサービスを受けることになります。このサービスを受ける部分はインターフェース(マン・マシン・インターフェースともいう)といいます。このようにオブジェクトを中心に考えていくことをオブジェクト指向といいます。

ソリューションビジネスの原点は「ユーザーがニーズを解決するために必要となるデータを提供するサービス」のことであり、このサービスを実現するにはオブジェクト指向という考え方が必要となります。提案型SEは、「プログラムとデータ」という考えから脱皮してオブジェクト指向の手続を理解する必要があります。

ソリューションビジネスがうまくいかない理由の多くは、システム開発時代の理論的バックボーンを変えないでソリューションビジネスに突き進んでしまい、現場のSEが混乱してしまうことにあります。この理論変革がソリューションビジネスの第一歩です。

# THEORY4

原価見積から  
投資金額提案へ



( )は売上高100億円、業界粗利益20%の企業の例です。

ソリューションビジネスにおいて提案するシステムスペックには、何を作るかという開発スペック(What)の他に金額(How much)、スケジュール(Who&When)があります。ここではまず金額について考えましょう。

システム開発における金額は原価見積がベースとなります。ユーザーより提示された開発スペックに対し、プロジェクトリーダーが開発工数(何人で何日くらいかかるか、人日、人月という単位を使う)を見積り、ハードウェア費用などを調べ、原価を積算します。この原価に一定率の経費を乗せ、さらにITベンダーの希望利益を乗せて見積金額を算出します。そのうえでセールスが顧客であるユーザーと、金額折衝またはシステムスペックの調整(値段が高いのもう少しスペックをカット...)をするという形で進めていきます。

一方ソリューションビジネスにおいては、この金額をソリューションベンダーがユーザーの課題解決のためのパートナーとして提案します。したがって「金額折衝」という概念そのものが存在しません。それは「ユーザーにとって最良な投資金額」をソリューションベンダーが提案すべきだからです。この投資金額の提案についてはさまざまな方法がトライされてきましたが、結論は1つに落ちついています。人件費におけるラッカープラン(企業が生み出した付加価値の一定比率を人件費の総額にする。この比率を労働分配率という)同様に、付加価値額(売上総利益、粗利額と考えてもよい)の一定比率をITに投資することです。しかし実際には投資期間内の付加価値を予測することが難しいので、売上高をベースとすることが多いといえます。基本的には以下のように進めますが、このプロセスはすべてユーザーに公開します(または一緒にやります)。ソリューションイメージでは売上高100億円、業界での平均粗利率20%の企業を例としています。

### ①総予算を決める

売上高に一定比率(IT分配率という)を乗じて求めます。IT分配率はユーザーの属している業界の平均粗利率×0.02(粗利対比2%)が平均値といえます。ユーザー側の経営者の感覚では「業界なみに」投資するということです。

### ②目に見えるコストと目に見えないコストに分ける

IT導入には他の項目に隠れてしまう費用(目に見えないコスト)が発生します。例えばハードウェアの場所代、電気代、電話代に隠れる通信費などです。これを個別に見積っていくのですが、目に見えるコストと目に見えないコストの比は3:1位が平均です。

### ③イニシャルコストとランニングコストに分ける

システム開発費、ハードウェア導入費などのイニシャルコストと、保守費、消耗品などのランニングコストに分けます。この比は1:1が標準です。イニシャルコストは年あたりコストですので、耐用年数をかけて総イニシャルコストを出します。耐用年数は法的にはパソコンが4年、サーバー以上が5年、ソフトウェアは5年となっていますが、そんなことはすべて無視して、このシステムを何年位使うか、使えるかということを考えます。この総イニシャルコストと原価見積から計算された見積金額を比較します。またランニングコストはこれをユーザーに予算化するよう提案します。

ソリューションイメージの例では総イニシャルコストは4500万円となります。これに対し見積金額が3000万円の場合は、見積書は3000万円とすればOKです。この場合年あたりランニングコストもこれに合わせて3000万円で予算化します。では逆に総イニシャルコスト4500万円に対し、見積金額が1億円の時はどうしたら良いのでしょうか。従来のシステム開発の場合はこの総イニシャルコストをユーザーがすでに予算として決定していることが多いので、ユーザーとベンダーの商談は決裂となります。こう考えてみると原価と予算が乖離しているシステムは世間にいくらでもあることになります。ソリューションビジネスは見方を変えるとこの仕事を受注することにあるといえます。

イニシャルコスト<見積金額の場合は次の4つの方法しかなく、これをユーザーとパートナーであるソリューションベンダーとの間で話し合っていく(交渉ではなく)しかありません。

### ①IT分配率を上げる

ユーザー企業が業界平均にも増して「ITに力を入れる」ことを意思決定し、見積金額からIT分配率を逆算して求めます。先ほどの例の見積1億円では、売上対比0.9%の投資となります。経営の意思決定といえます。

### ②耐用年数を上げる

総イニシャルコストを上げるには耐用年数を上げる、つまり我慢してシステムを長持ちさせることです。しかしこれだけテクノロジー進歩、環境変化の激しい現代ではなかなか難しいといえます。

### ③システムスペックを落として見積金額を下げる

やや安直ではありますが、もっとも現実的な方法といえます。ソリューションテーマは変えずにシステムスペックを落として、つまりユーザーが多少のことは我慢して見積金額を下げるというものです。家や自動車などの購入ではよくとられる手法です。「本当は都心に一戸建てが欲しいが、通勤時間2時間は目をつぶろう」ここで大切なことは「目をつぶる」スペックをユーザー企業によく理解してもらうことです。

### ④システムスペックを落とさずに見積金額を下げる

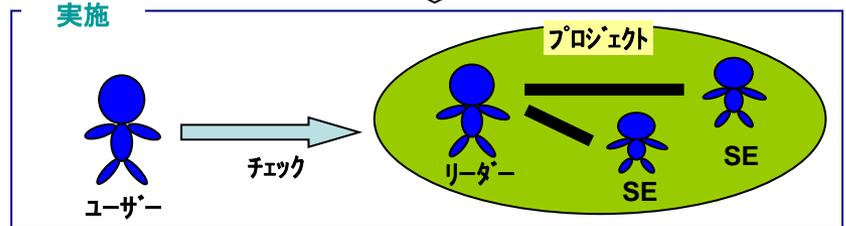
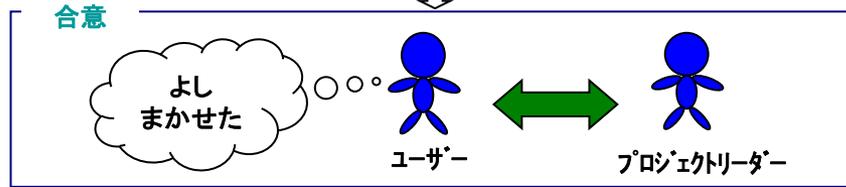
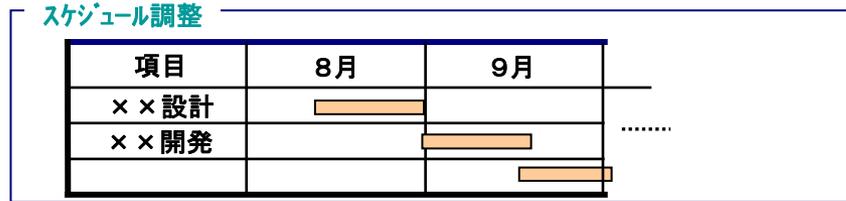
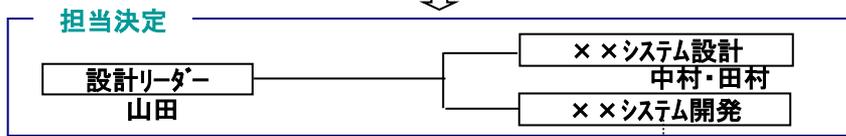
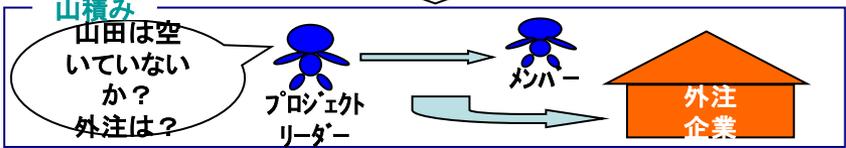
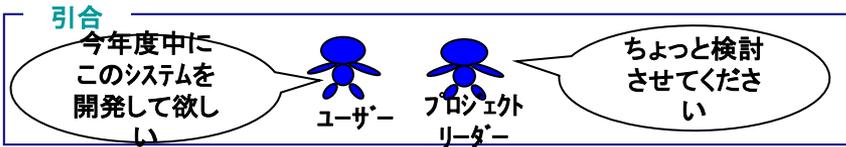
これには2つの方法があります。1つは原価見積を変えずに見積金額を下げることです。利益を減らすディスカウントや、赤字も覚悟するダンピングです。しかしよく考えると原価見積のうちSEなどの人件費は固定費であり、見積は仕事がうまっていることを前提にしています。金額が折り合わないから仕事を断って1円も入らずSEが遊んでしまうくらいなら、仕事をとった方が良く考える企業もあるようです。しかしこれは手抜き開発のもとですし、結果的にはソリューションベンダー、ユーザー双方に不幸をもたらすこととなります。

もう1つは原価見積のダウンです。このテーマに長年ITベンダーは取り組み、出した結論がセオリー36で述べるソフトウェアの再利用です。

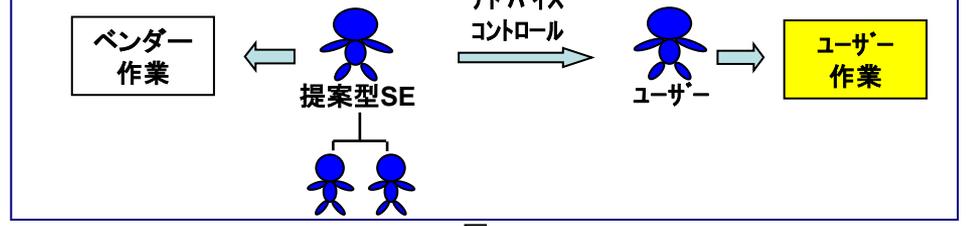
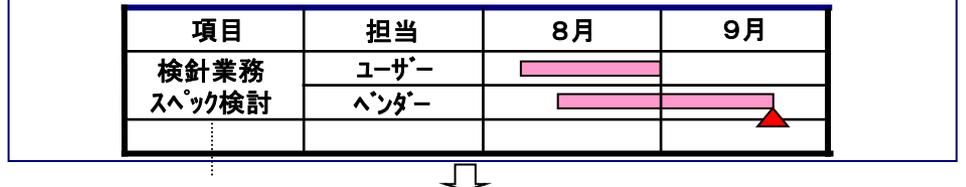
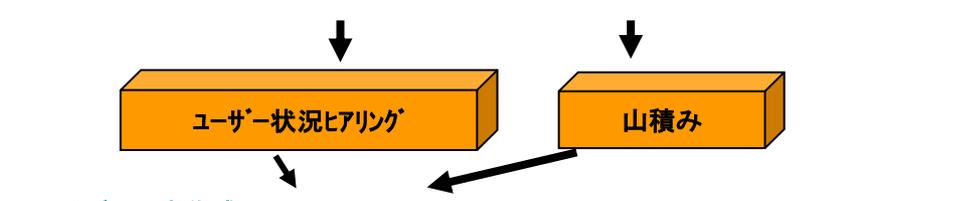
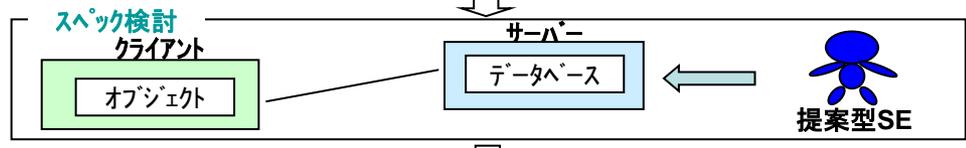
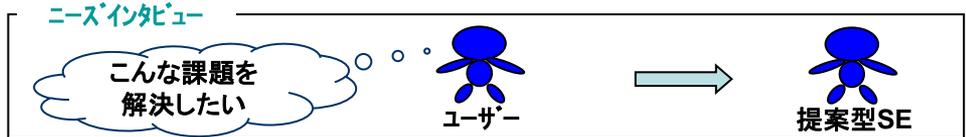
# THEORY5

スケジュール表では  
ユーザー側作業を提案する

# 従来のシステム開発



納期遅れはプロジェクトリーダーの責任



納期遅れは共同責任

従来のシステム開発ビジネスではスケジューリングをソリューションイメージにあるような形で進めてきました。まずユーザーから開発スペックとともに希望納期をインタビューします。自社に戻りこの仕事に必要な工数を考え、メンバーの空き状況、外注状況を調べ、希望納期内に可能かどうかを検討します。(これを山積みといいます。個人別に仕事量を山のように積み上げていくのでこういいます)個々の作業において担当が決定し、リーダーが「行ける」と判断すると、これを作業項目別にスケジュール表にまとめ、ユーザーに提出します。ユーザーの了解を得たら、開発に着手し、ユーザーはこのプロジェクトの進捗を先ほどのスケジュール表をもとにチェックします。もし約束した納期に開発が終わらない時は、もちろんプロジェクトリーダーの責任となります。

ソリューションビジネスにおいては何度もいうように、まずニーズが出発点です。このニーズを解決するために必要なスペックを検討し、オブジェクトを検討します。ここからはシステム開発ビジネスと同じような形になるはずですが、大きく異なる点が1つあります。それはユーザー側が行なう作業について、誰がスケジューリングするかです。システム開発ビジネスではユーザーは「～をやって欲しい」とITベンダーに頼んでいますので、自らの作業を切り分けています。つまり自分の仕事のうち、外注すべきシステム開発という作業を切り分け、他の仕事との関係で希望納期を出し、ITベンダーに打診しています。したがって自分の作業は自分でスケジューリングしています。

一方、ソリューションビジネスではニーズ解決に関するアイデアをすべてソリューションベンダー側が出すのですから、ユーザー側はもちろん何をしたいかわかりません。したがってユーザー側が行なう作業をソリューションベンダーが「提案」する必要があります。ソリューションビジネスで後々トラブルが起きる最大の原因はこの作業の切り分けがなされていないことによるものです。よく両者のエアポケットになってしまう仕事は以下のようなものです。

- ・システム運用開始までに必要なデータの入力作業(これをよくマスタファイルというので、マスタ作成といいます)
- ・NTTなど通信キャリアへのネットワーク手配
- ・新システムに伴なって仕事のやり方が変わるケースが多く、その場合の業務マニュアル(システムの操作マニュアルでなく)
- ・ユーザー側作業の担当者のアサイン

これらはシステム開発をやってきたITベンダーからすれば、何も言わなくても今までどおり「当然」ユーザー側の作業と理解しています。しかしソリューションサービスを受けるユーザー側では「ソリューションベンダーが我々のやることを何も言わないので、すべてベンダーがやってくれる」と思っており、とんでもない事態を迎えます。ベンダーはユーザー側が気づいた時にやれば済むと思っていますが、これらの作業はユーザー側に内部コスト、場合によっては外部コストを発生させてしまいます。

セオリー4で決めた投資金額以外にコストが発生するとすれば、システムを開発すること自体を考え直さなければいけないこともあります。これをユーザー側企業の情報システム部(基本的にはそのシステム開発をやりたいし、失敗すれば社内では自らの責任となる)が「うやむやのうちに無理して」進めてしまい、納期遅延を起こすことも少なくありません。この時社内で矢面に立つのは情報システム部であり、情報システム部は経営者に「この遅延はベンダーの責任である」と主張します。場合によっては訴訟問題にも発展してしまいます。

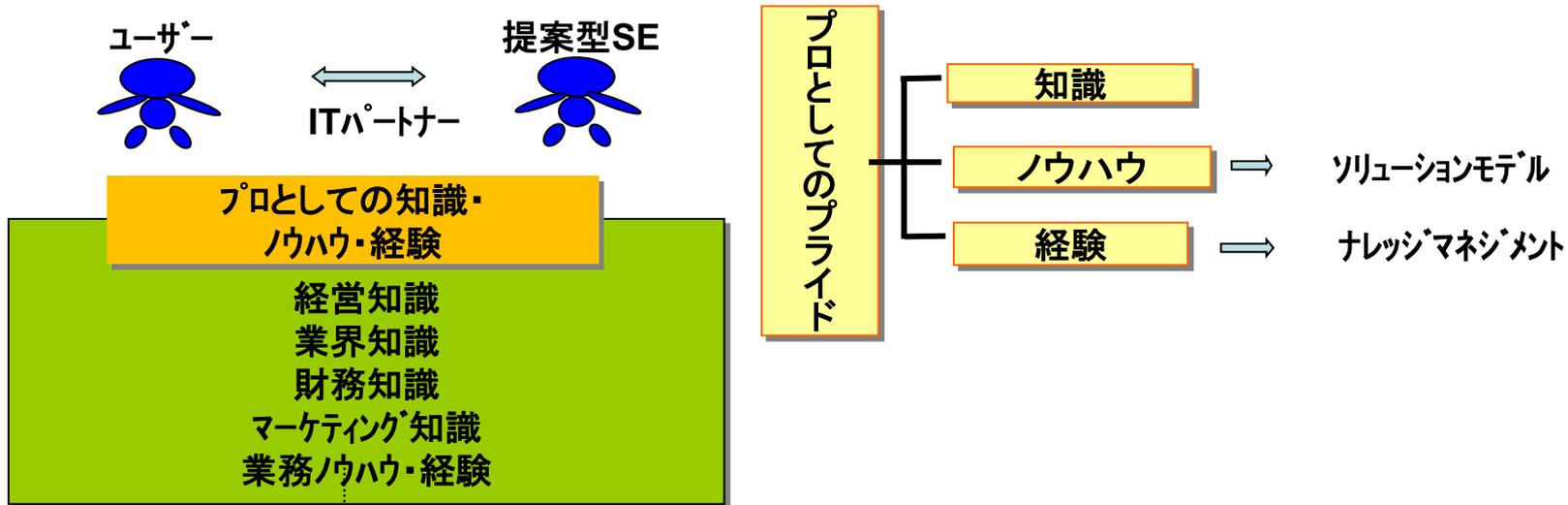
提案段階でしっかり作業分担し、特にユーザー側作業についてはよく仕事の状況(いつが忙しいかなど)をヒアリングして、ユーザー側作業を含めたスケジュール表を提出する必要があります。実施段階では、リーダーである提案型SEがユーザー側をアドバイス、コントロールしながらやっていきます。

これでも納期遅延となれば、いうまでもなくベンダーとユーザーの共同責任であり(ソリューションベンダーの方が責任がやや大きいのですが)、「その対応を共同で話し合っていく」という結論になるはずです。

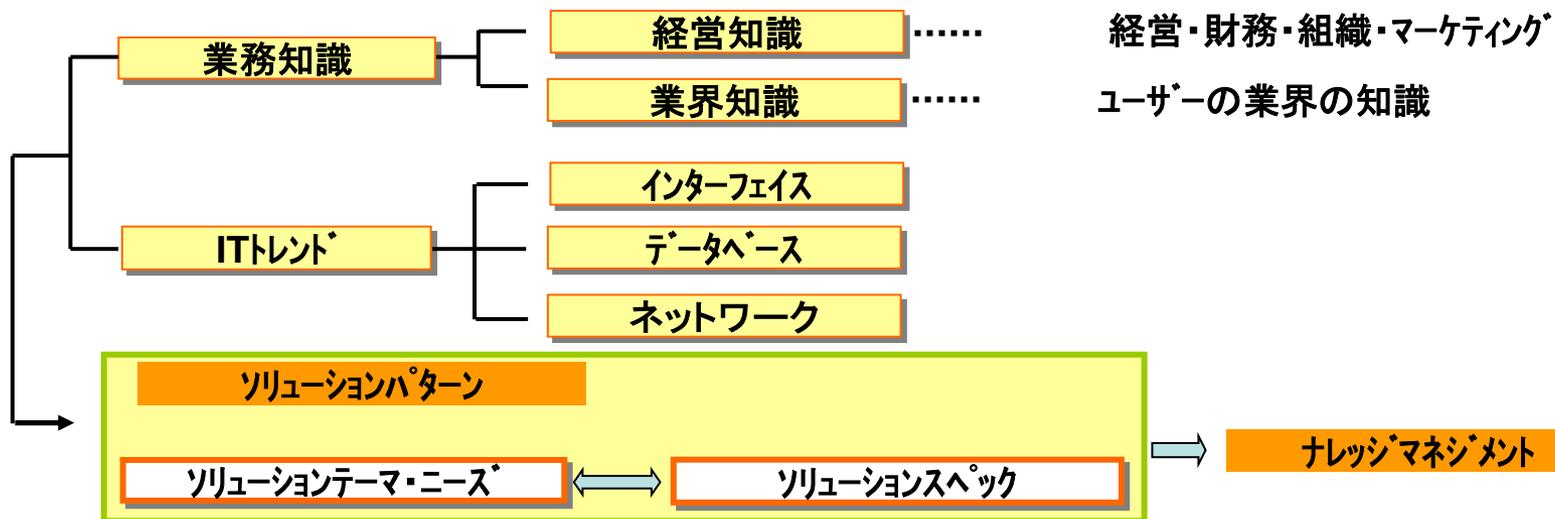
# THEORY6

提案型SEに必要なものは  
知識・ノウハウ・経験

# 提案型SEに求められるもの



# 提案型SEに必要な知識



システム開発ビジネスにおいてSEは、情報システム部のパートナーとして開発パワーを提供してきたといえます。一方、ソリューションビジネスにおいて提案型SEはユーザー企業のITパートナーとして存在することになります。では提案型SEは何を提供しているのでしょうか。ユーザーは特定の仕事のプロ(経営者、経理部、営業部、人事部・・・)であり、プロとしてその分野の知識、ノウハウ、経験を持って、自らの業務の課題を解決しています。しかしこの業務のプロであるユーザーがどうしても自ら解決できないのがIT分野の課題です。IT分野における知識、ノウハウ、経験が不足し、それを解決できないのです。この3つを提供し、そのITに関する課題を解決するのがソリューションビジネスであり提案型SEといえます。

ソリューションビジネスにおいて、ソリューションモデルが確定したら、次にやるべきことは提案型SEに、自分が何の仕事をするプロであるかという自覚と、それに対するプロとしてのプライドを持たせることです。前者の答えは「ユーザーのニーズをITで解決する」ということであり、後者の答えはこの仕事に見合うだけのプロとしての知識、ノウハウ、経験を持つということです。

まず知識ですが、これは大きくユーザー側の業務知識とIT知識の2つに分けることができます。業務知識はその業務のプロであるユーザーと同等の知識を持つということではなく(そんなことは不可能ですし、ユーザーが持っている知識をSEがすべて持っている必要はありません)、ユーザーの言っていることが「理解できる」ことが目標です。業務知識は言い換えれば「常識」であり、「そんなことも知らないのか」とユーザーに言われたい程度の知識です。

常識はビジネスマンとしての最低限の経営常識と、ユーザーの属する業界の常識の2つに分けることができます。前者は「日本経済新聞」を読んで理解できる程度であり、後者は例えば流通業のユーザーのソリューションを行なうのであれば「日経流通新聞」が読める程度ということです。

一方IT知識はプロとしての領域であり、ユーザーが持っていない知識を、ソリューションサービスを通じて有料で提供するものです。ここで必要なのはITに関する詳細な技術知識ではなく、ITの現在の実現化レベル、製品動向、今後の動きといったことで、トレンドと表現したほうが良いものです。分野でいえばインターフェース、データベース、ネットワークの3つに分けられます。インターフェースとは人間とシステムの接点であり、パソコン、携帯端末、ICカード、新しいメモリーなどのハードウェア、音声入力、画像認識などのソフトウェアを指します。これらの動きはシステム設計よりもIT利用に大きなインパクトを与えるものであり、ソリューションの第一キーとなります。POSシステムで実用化したバーコードはその後さまざまな分野で活用され、多くの業務を変革していきました。データベースとはサーバーなどのハードウェア、データベースソフトなどを指します。エクセルなど表計算ソフト(一種のデータベースソフト)の普及がユーザーの業務に大きなインパクトを与えました。ネットワークとは通信キャリア、プロバイダーのネットワークサービスや通信ソフトなどを指します。メール、インターネットもユーザー側の業務に大きな変革をもたらしています。このデータベース&ネットワークのトレンドはシステムのコスト、スピードなどに大きな影響を与え、ユーザー側のソリューション実施の意思決定に大きな影響を与えます。

提案型SEに必要な知識は業務知識とITトレンドという分離されたものではなく、この2つを融合した所にあります。これがソリューションパターンです。つまりどういうテーマ・ニーズの時に(これを理解するのに業務知識が必要)、どういうシステムスペックでソリューションしてきたか、そして今考えればそのスペックはどうあるべきか(これを考えるのにITトレンドが必要)という組み合わせ知識が必要です。

次にノウハウですが、これはセオリー2で述べた仕事のやり方、つまりソリューションモデルを提案型SEが「体で」覚えることです。このモデルを理論的に覚えるのではなく、何回かモデルを使ってソリューションしていくうちに自然に体が反応してくるようになります。

最後に経験ですが、これが最大の難題です。ソリューションビジネスではユーザーは自身の経営・管理・業務の課題解決を相談します。そしてその解決の良否が企業の業績、さらには生死をも決定づけます。こんな大切なことを誰に相談したいかといえば「ソリューションのプロ」です。ソリューションのプロとは「ソリューションをやったことがある人」であり、経験の豊かな人です。しかし、ソリューションをビジネスとして見ると「経験のない人」ができないとすると、一番最初の仕事は誰がやるかということになります。こう考えると以下のようにするしかありません。

- ・まず誰かが擬似的に経験を積み、ソリューションビジネスにトライする。
- ・その経験を企業内で集約してデータベース化する。他の提案型SEはこのデータベースで疑似体験してからソリューションビジネスを行う。さらにその経験をデータベースに追加していく。

前者についてはセミナーなどを使うしかありません。これについてはセオリー44で述べます。後者はいわゆるナレッジマネジメント(企業内の知識、知恵を皆で共有し、皆が擬似体験すること)とよばれるものです。そしてこのナレッジマネジメントにおけるデータベースのイメージは先程知識の項で述べた「ソリューションテーマ・ニーズとシステムスペック」というナレッジの集大成といえます。ソリューションビジネスでは、このナレッジの量が勝敗の分かれ目となります。この作り方についてはセオリー46で述べます。

ソリューションはスポーツのようなものです。ルール、原理・原則を知り(知識)、やり方を覚え(ノウハウ)、そしてやってみる(経験)ことです。ルールを知らずにやったり、やり方を知らずにデタラメにやったスポーツ経験なら、かえってない方がうまくなります。さらにうまくなりたければプロのやっているシーンを見て、自分がスポーツをやっている所をビデオに撮り、比較して見る(ナレッジ)ことが大切です。

## 第2章 ニーズ予想モデル

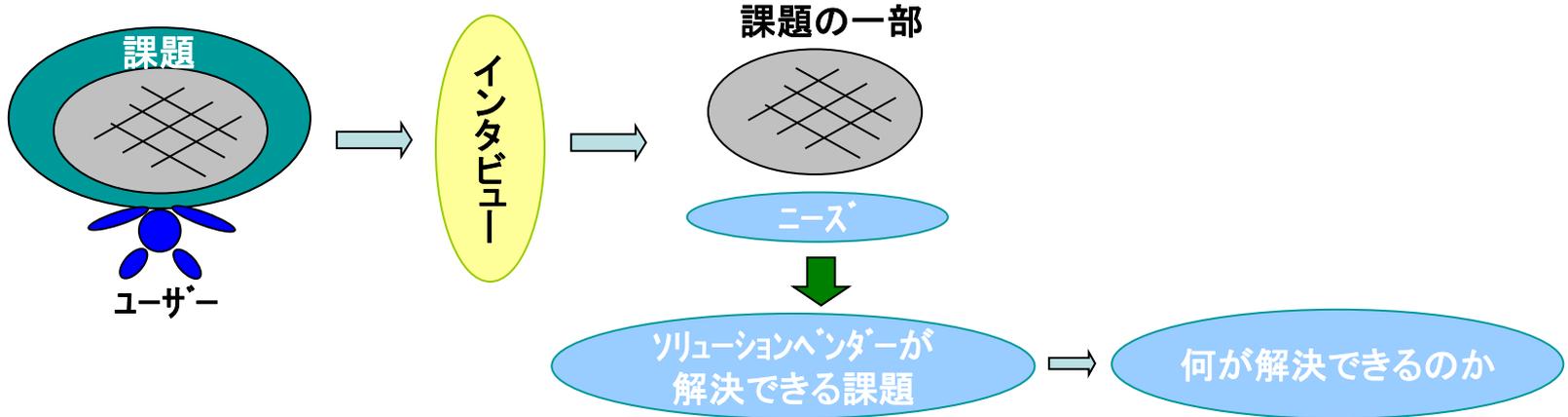
### トータルセオリー

ソリューションビジネスはSEのインタビュー苦手意識をとることからスタートする。それはインタビューすべきことをはっきりさせ、前もって準備しておくこと以外ない。

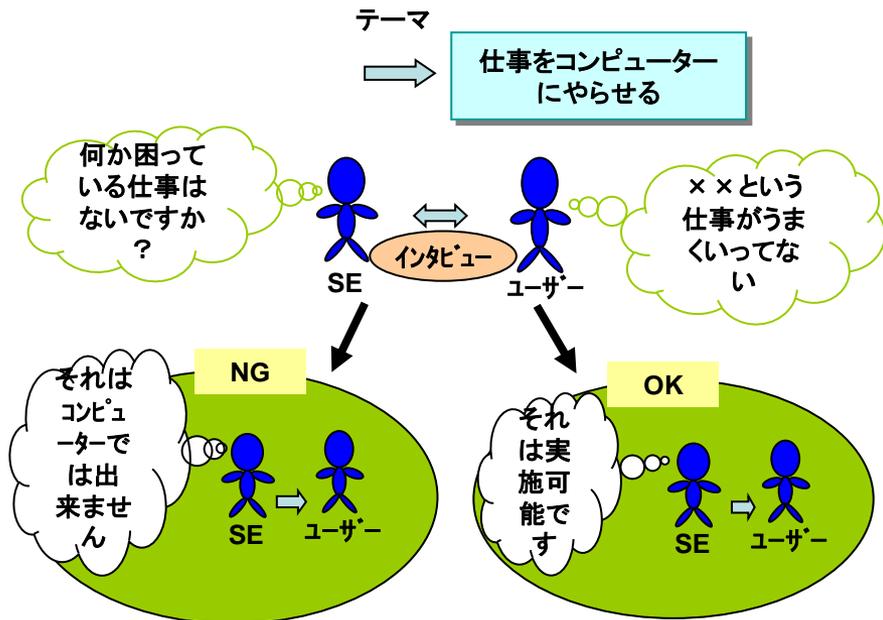
# THEORY7

「何が解決できるのか」  
からスタート

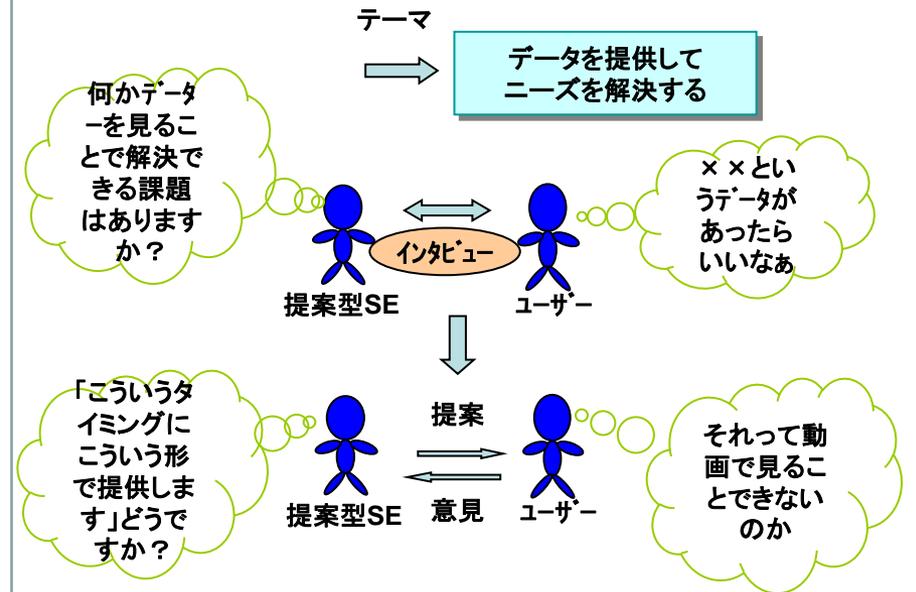
# ニーズとは



# 従来のシステム開発



# ソリューションビジネス



ニーズ予想モデルとは、ユーザーへインタビューする前の準備作業のやり方のことをいいます。驚くべきことにインタビューを苦手としている人ほどこの準備作業をやらずに、出たとこ勝負をねらい、インタビューが得意な人ほどよく準備してから行きます。

SEにニーズインタビューをさせると、よく出る最初の言葉は、「今日はわからない所を教えてください」というものです。SEから見てユーザーの仕事・悩みなどわからないところだらけですし、それをすべてインタビューしたら膨大な作業となってしまいます。SEは他の職種に比べて自らの仕事のコスト・生産性を意識しているといえます。自分の仕事のスピードが企業の損益に直結していることを体で感じているからです。しかし不思議なことにインタビューという作業に生産性を意識するSEは皆無です。何とか早く終らせようとするSEはほとんどおらず、何となく不安なので今のうちに聞けることはすべて聞いておこうというスタンスをとります。

一方ユーザー側もSEと話すときは緊張していることが多く(難しいことを聞かれそうなので)、どんなことを聞かれても自分の答えられることであれば、喜んで答えます。SEに答えができたという喜びで、その質問が何のためになされているのか、何に使うのかといったことにはほとんど無頓着です。

ニーズ予想モデルでまず考えなくてはいけないのは、インタビューは一体何のために行っているのか、何がわかれば提案できるのかということ。「行く前」に考えておくことです。インタビューで聞くべきことはニーズです。ニーズとはユーザーが抱えているさまざまな経営や業務に関する課題のうち、そのソリューションベンダーが解決できる課題のことをいいます。インタビューはユーザーの数多い課題の中からニーズを抽出する作業といえます。インタビュー準備である課題予想モデルの出発点は、そのソリューションベンダーが「そもそも何が解決できるのか」つまり「何をニーズと考えるのか」であり、「自らのサービスは何なのか」ということです。これが今のソリューションベンダーにもっとも欠けている点です。

例えば、人材紹介会社は自分が「採用する」という機能を提供するのですから、インタビューテーマは「採用できない職種はないか」ということになります。計測制御機器を販売している会社は「計測できない所、制御できない所」をインタビューすべきですし、他の課題(例えば採用できない職種)などを聞いていたらインタビュー生産性が低くて仕事になりません。

システム開発ビジネスでは「仕事を早く正確にコンピュータにやらせる」ということがテーマなので、聞くべきことは「遅くまちがっている仕事はないか」です。そのうえでその仕事をコンピュータにやらせることが可能かをSEが検討して回答します。この場合ユーザーは「何か困っていることはありませんか」という質問に対して「遅くまちがっている仕事」を回答できます。

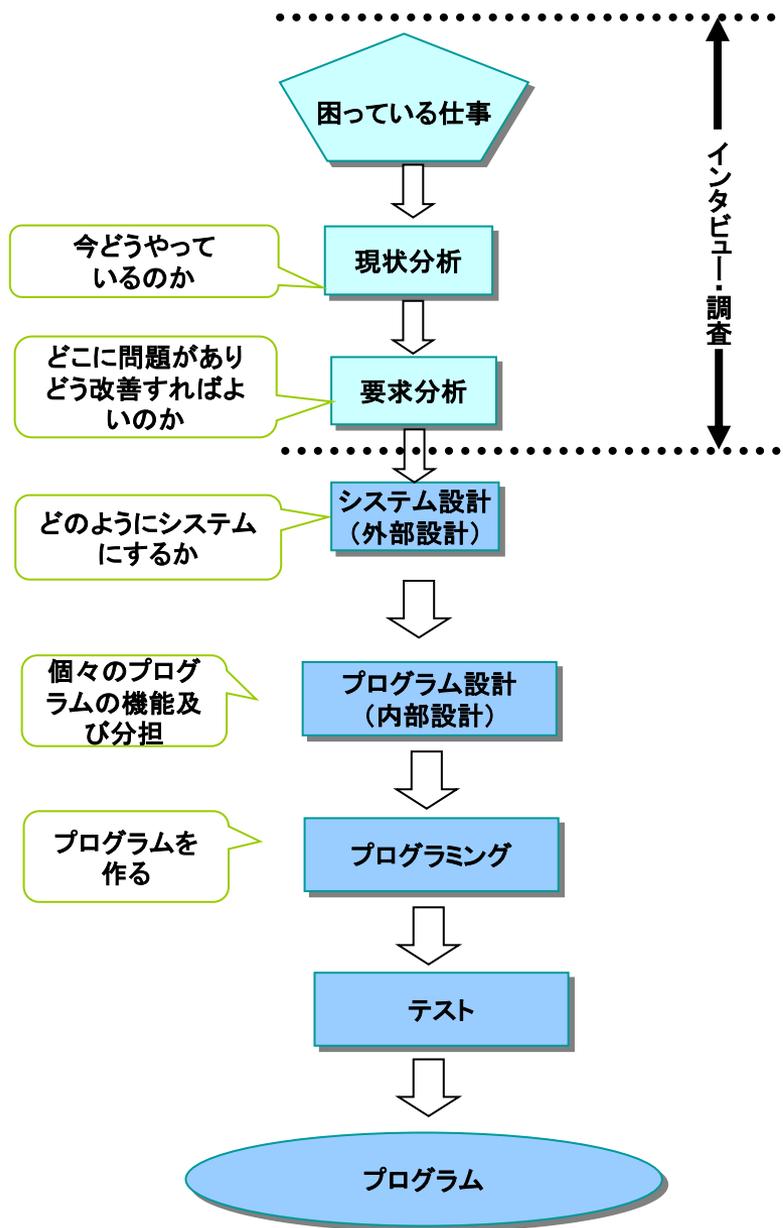
ソリューションビジネスではセオリー3でも少し述べたように「コンピュータが計算したり、覚えたりしたデータを見たい人に、タイミング良く渡す」ことです。したがってソリューションベンダーにおけるニーズとは「ユーザーが抱えている課題の中で、データを提供することで解決できる課題」と定義することができ、これを聞くのがインタビューテーマとなります。

ソリューションビジネスでSEがインタビューをうまくできないのはこれを意識しておらず、従来のシステム開発ビジネスの「くせ」が抜けず、「何か困っていることはありませんか」という質問に終始することです。先ほどのように遅く間違っている仕事は確かに「困っています」が、データ提供は必ずしも「困っている」という問題だけでなく、(もちろん困っていることもあります)「あったらいいなあ」という要望もあるのです。ソリューションビジネスはシステム開発のように「問題点解決」だけでなく、それをも包含して「こんなデータが必要だ」というニーズをとらえることにあるのです。このニーズをもとに提案型SEがデータを提供する方法を提案し、それによってニーズをシステムスペックへと変換していく、これがソリューションビジネスです。

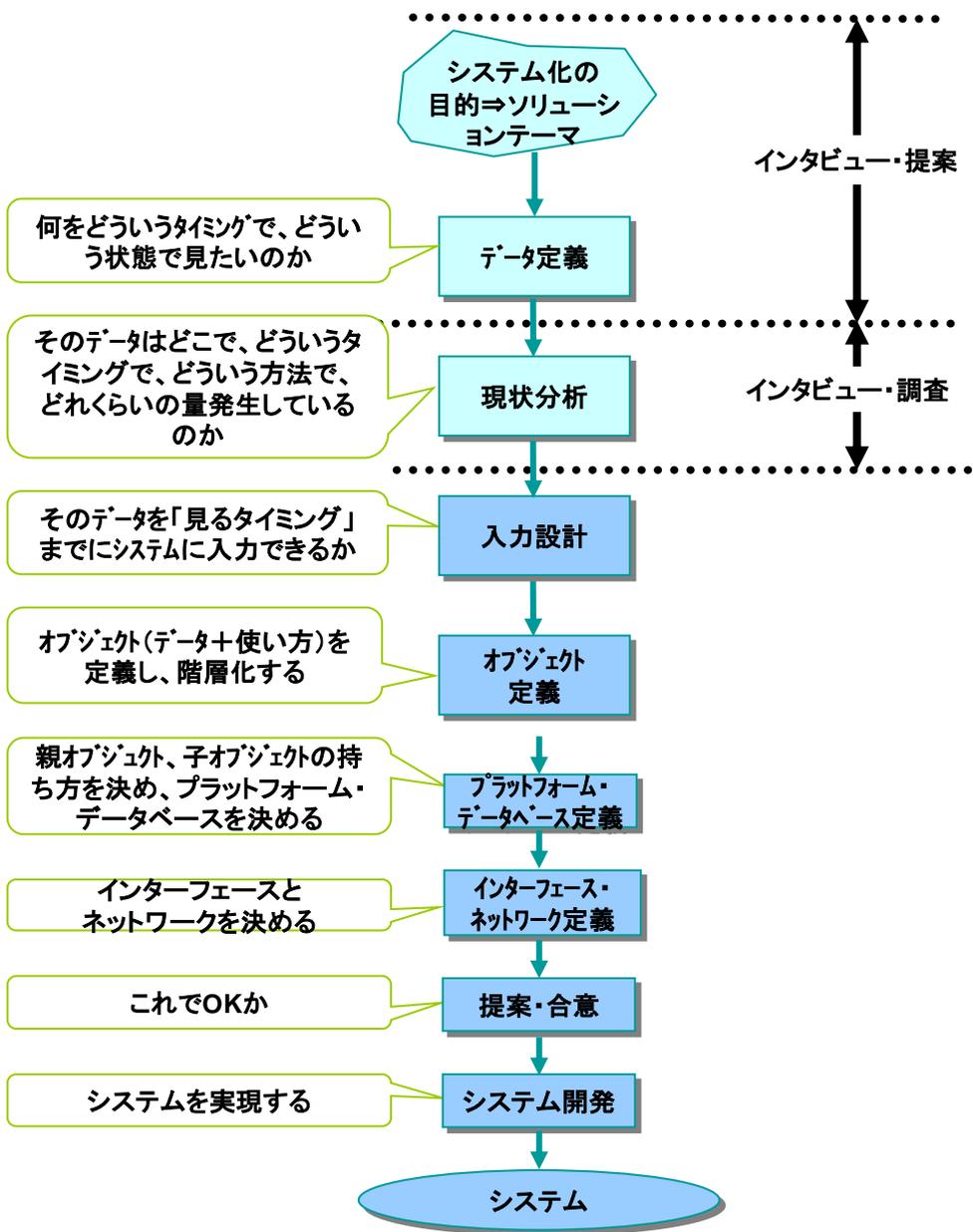
# THEORY8

プロセス志向アプローチから  
データ志向アプローチへ

# プロセス指向アプローチ



# データ指向アプローチ



次にニーズ予想モデルで大切なことは、インタビューの「ソリューションビジネスにおける位置づけ」です。ソリューションビジネスにおいてインタビューはどの工程で行い、何のためにやり、インタビューした後どういう形で作業が進められるかということです。

このソリューションビジネス全体の流れはデータ指向アプローチという形で表現され、システム開発ビジネスにおけるプロセス指向アプローチとははっきりと分けて考える必要があります。

システム開発ビジネスで用いられるプロセス指向アプローチ(滝のように上から下へ向かっていくのでウォーターフォールモデルともいいます)では、まず困っている仕事を見つけ、現在その仕事はどうやっているかを調べ(現状分析)、どこが問題で、どう改善すべきかを考えます(要求分析)。ここまでのアプローチをユーザーへのインタビューと書類調査などにより行ないます。そのうえで、どのようなシステムにするかを決め(システム設計:外部設計ともいう:本来これをやる人をSEと呼んだ)、個々のプログラムの機能とその開発担当を決め(プログラム設計:内部設計ともいう)、プログラムを作り(プログラミング:これをやる人をプログラマーと呼んだ)、テストします。最終成果物はプログラムです。プログラムつまり仕事のやり方を調べ、コンピュータに覚えさせるという意味で、プロセス指向アプローチといえます。

一方、ソリューションビジネスの出発点はシステム化の目的であり、ソリューションすべきテーマです。まずこれをインタビューを通じてはっきりさせます。つまりデータを提供することで解決できるテーマをインタビューで探し出します。次にこの提供データの属性をはっきりさせます。属性とは項目(What: 何を見たいか)、タイミング(When: 毎日1回見たい、いつでも見たい...)、状態(How: 数字で見たい、グラフで見たい、音で聞きたい...)の3つを指します。データ属性はインタビューできることも(ユーザーが見たいデータをはっきり言えるとき)ありますし、ぼんやりとテーマだけがあって提案型SEがテーマから提案することもあります。ぼんやりとテーマしかない時の方がソリューションビジネスは高付加価値サービスとなります。

見たいデータが固まったら、次にそのデータがどこで、どのように発生しているかを調べます(現状分析)。これは次の入力設計に使います。データ提供はそのデータがシステムに入力することさえできれば、何とかできます。逆にいえば入力していないデータは見ることはできません。システムがうまく動かない理由のほとんどは「データの処理の仕方」が悪いのではなく、そもそも「見たいタイミング」までにデータが入力されていないことがほとんどです。現状分析はこの入力設計が可能となる情報だけをインタビューまたは書類調査します。入力設計に必要なものは発生場所、発生タイミング(毎日コンスタント、月に1回...)、方法(電話がかかってくる、他システムで入力される...)、量(ピーク時にどれくらい発生するのか)の4つです。

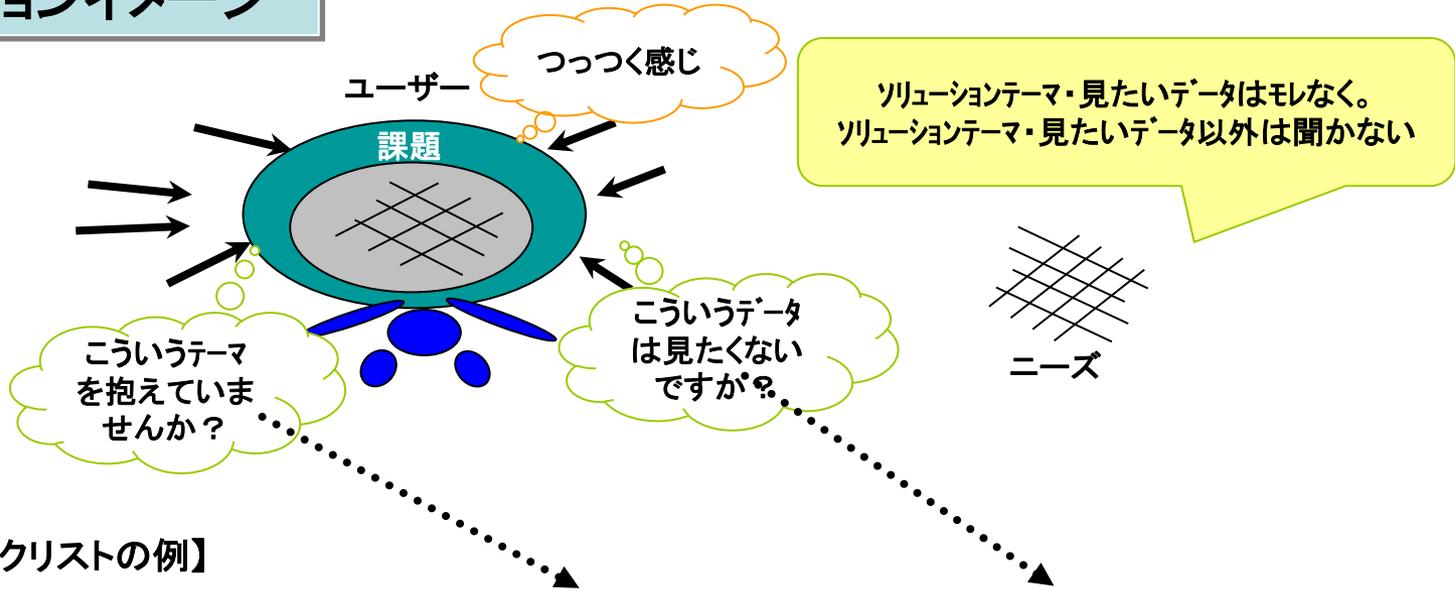
インタビュー及び調査は現状分析までです。このフェーズで大切なことは提案型SEはソリューションテーマ、見たいデータ、データ発生状態以外のことに興味を持たないことです。医者は病状とその治療に関すること以外のパーソナル情報(患者の出身地、性格、年収...もちろん病気治療に必要ななら別です)は興味を持ちませんし、持ってはいけないといえます。提案型SEも同様でこれがインタビューの生産性を高めることになるばかりでなく、かえってユーザー側の信頼を得ることになります。

インタビュー・調査後は入力設計、オブジェクト定義(必要なデータと使い方を決め、グルーピングして階層化する)、プラットフォーム・データベース定義(クライアント、サーバーなどのプラットフォームを決め、データベースを設計する)インタフェース・ネットワーク定義(ユーザーとクライアント、クライアントとサーバーのつなぎ方の検討)をします。こうしてデータ提供の実現を保証し、それをユーザーに提案し、ソリューションテーマが解決できることをユーザーに確認してもらいます。そのうえでこれらを実際に開発します。

# THEORY9

タイプ別のニーズ  
チェックリストを作る

# ソリューションイメージ



## 【製造業チェックリストの例】

区分	タイプ	予想ソリューションテーマ	予想される見たいデータ
生産形態 から見て	見込生産	<ul style="list-style-type: none"> <li>・製品の需要予測がうまくできない ⇒</li> <li>・在庫過剰による廃棄損、保管コスト増大 ⇒</li> <li>⋮</li> </ul>	<ul style="list-style-type: none"> <li>製品別売上の予測値、実績値</li> <li>在庫データ、保管コスト</li> <li>⋮</li> </ul>
	受注生産	<ul style="list-style-type: none"> <li>・部品と受注の間のギャップが大きい ⇒</li> <li>・見積ミスによる赤字が出る ⇒</li> <li>・納期遅れが発生する ⇒</li> <li>⋮</li> </ul>	<ul style="list-style-type: none"> <li>部品別出荷高</li> <li>見積と実績のかい離が大きい仕事ラインごとの進捗データ</li> <li>⋮</li> </ul>
製品種類から 見て	少品種多量生産	<ul style="list-style-type: none"> <li>・特定製品の売れ行きに業績が左右される ⇒</li> <li>⋮</li> </ul>	<ul style="list-style-type: none"> <li>製品ポートフォリオ</li> <li>⋮</li> </ul>

データ指向アプローチには1つ大きな問題点があります。それはニーズが「もれやすい」ということです。システム開発のプロセス指向では「困っている仕事」がインタビューテーマでしたので、まずもれることはありません。しかしデータ指向アプローチのインタビューテーマは「見たいデータ」です。

読者の皆さんもユーザーの1人ですので自分のケースで考えてみてください。貴方の目の前にSEが来て「今困っている仕事を挙げてください」といわれた時、多くの方は答えることができますし、もれも少ないといえます。仮にもれても「気がつかない程度に困っている仕事」なので大丈夫かもしれません。しかしSEから「データを見ることによって解決できる課題や見たいデータを教えてください」と質問され、「まず第一は需要予測という仕事が予測データを提供してもらえれば解決されます。次に...」「私の見たいデータは、大切な順に1番目は××2番目は××...」とすぐに答えていく人はまずいないでしょう。困っている仕事は毎日困っていることを意識しますが、見たいデータは毎日「見たい」「見たい」と考えているわけではありません。

ニーズにもれが生じるとどうなるでしょうか。システムが出来上がってから「このデータが見たい」と突然言われても、入っていないデータは見ることはできませんので入力設計を見直す必要があります。入力設計を見直せばオブジェクト定義を見直し、プラットフォームを見直し、データベースを見直し、ネットワークを見直し...これなら最初からやり直したほうが楽かもしれません。

データ指向アプローチではニーズもれは致命傷となることが多くなります。このニーズもれの防止策は次の3つがポイントです。

## ①ユーザーに認識してもらう

インタビューのニーズもれはシステムが出来上がってからわかってしまったことをユーザーに理解してもらうことです。ソリューションビジネスは「ユーザーのために」共同でシステムを考えていくものです。ニーズもれによる手直しはソリューションベンダーよりもユーザー企業に不幸をもたらすことを理解してもらいます。極端に言えばソリューションベンダーはやり直しをして、その料金をもらえば良いのですが、ユーザー企業はそのコストを負担しなくてはなりません(インタビュー段階でこういう話をすれば、ニーズもれはユーザー責任であり、コスト負担すべきことはわかってもらえます。これをシステム開発後話すとユーザーは納得してくれません)。

そのうえで「何がもれると大変か」をユーザーにわかってもらいます。それはセオリー8で述べた提供データの3つの属性であり、なかでも項目、タイミング、状態の順にもれると大変だということをわかってもらいます。見たい項目がなければもっとも大変、次に「夕方見たい」が「いつでも見たい」に変わったら大変、そして「数字をグラフにしたい」は大したことはないということを提案型SEがきちんと説明することです。それでももれたら大変なことが起こると予測される時は、この後さらにプロトタイピング([セオリー40参照](#))という手法を用います。

## ②インタビューで「つつく」

ニーズインタビューの感覚は「つつく」という表現がフィットします。医者が患者に手を当てたり、熱をはかったりという感じです。決してニーズを「うまく当てる」と思わず、「こういうテーマを抱えていないか」「こういうデータは見たくないか」と軽く「つつく」感じます。もしユーザーがそのデータを「見たい」といえばGoodですし、もし「見たくない」といわれてもがっかりすることはありません。「見たくない」というニーズをとらえただけでなく、それによって別のニーズを聞くことができるかもしれません。「顧客別の売上高を見たくないですか」と質問すれば「それはどうでもいいけど、顧客別の粗利の方が見たいなあ。そうだ、もっと見たいのは同じ商品売って、粗利を出している顧客と出していない顧客の対比ができれば良いな」となっていくこともあります。「ユーザーニーズをあてる」のは難しいのですが、「つつく」のは質問を多く用意すれば容易です。

### ③ニーズ予想チェックリスト

そう考えると、顧客のタイプ別にニーズ予想チェックリストを作れば良いことに気づくと思います。これはソリューションイメージにあるようにまず業種単位(製造業、金融業...あるいはもっと細かく家具製造業、さらには部署単位など)に表を作り、タイプ別(見込生産か受注生産か、素材加工か組立加工か...)に予想される「ソリューションテーマ」および「見たいデータ」の一覧表があれば便利だと思います。

ここまで話すと、ソリューションベンダーの経営者は「これをコンピュータにデータベース化し、顧客の属性を入れると質問表が出るようにしよう」と言います。筆者は賛成できません。ニーズインタビューでは「何故そのような質問をするのか」を質問する側が理解しておく必要があります。そうでないと質問される側に「何でこんな質問するのだろう?何を考えているのだろうか」といった不信感が生まれてきますし、質問者である提案型SEはユーザーの答えに応じて、臨機応変に質問を修正・追加・削除することができません。

筆者は年に数回、新人のSEに会うことがあります。彼らに「見込生産の製造業の経営課題をあてなさい」というと10人が10人とも「いくつ作ったら良いか」つまり「需要予測」と答えます。ところがSEになってから10年間、自治体のシステム開発だけをやってきたSEに聞くと「知りません」と答えます。今度は10年間製造業だけをやってきたSEに聞くと「ケースバイケース」と答えます。さらに「新人SEでもあてられる」と言うと、自治体をやってきたSEはあてます。しかし製造業をやってきたSEは「ケースバイケース」といいはります。

タイプ別のテーマ、見たいデータの予想は冷静に考えれば誰でもわかりますし、はずしても質問した根拠を述べればユーザーと会話になります。また質問事項がどうしてもわからなかったらインタビューする前に、誰かに教えてもらって、どうしてそういう質問をすべきかを理解しておくことです。

# THEORY10

データベースのニーズは  
「共有化」と「捨て方」

# データベースニーズ

## 共有化ニーズ

- ・データが重複している
- ・不整合
- ・他人のデータが見られない  
使えない

## 共有化

### 共有化の問題点解決

使い勝手が  
悪い

操作性

壊れる  
危険性

バックアップ

ぬすみ見  
される

セキュリティ

## データ量ニーズ

- ・データが多すぎてどこに  
何があるのかわからない
- ・データが多すぎてレスポンスが悪い
- ・どう整理していいのかわからない

## データタイミングの定義

### 捨てる提案

見るタイミングによって持ち方  
を考える

捨てるタイミングを決める

ソリューションニーズには「見たいデータ」のほかに「データのため方」、つまりデータベースに関するものや、「つなぎ方」つまりネットワークに関するもの(セオリー11~13で述べます)があります。

データベースに関するニーズは次の2つのものがあり、これをインタビュー前に予想し、かつインタビュー後の提案の方向を知っておく必要があります。

### ①共有化ニーズ

データを各人が持っている、データの二重入力による重複や不整合、他人が作ったデータが見られない、利用できないといったことがニーズとして挙がってきます。

この場合そのデータを皆で「共有化しよう」と考えるのが普通です。共有化において提案型SEが考えなくてはいけないのは「共有化が良いのがわかっていて、なぜ今まで共有しなかったのか」ということです。すぐに浮かぶのは「共有化する」という仕事が皆のための仕事であるため、一体誰の責任かがわからず、誰もその方法を考えず、誰もやらなかったということです。この問題は提案型SEがアウトソーシングを受け、考えて実行すればかたづきます。

共有化にはもっと本質的な問題があり、共有化すると必ず次の3つのことが露見してきます。これらを前もってユーザーに話すことがニーズを掘り起こすことになります。共有化は「する」ことが難しいのではなく、「した後その状態を維持していく」ことが難しいのです。

## ・使い勝手が悪い

個人で自分のデータを持っていた時と比較すると、共有化すれば必ず使い勝手は悪くなります。机の中のデータをセンターファイルの棚に置けば、そこに取りに行き探るのが大変です。共有化すれば他人のデータを見ることが可能ですが、自分のデータを自分で使う時はかえって不便になることをユーザーに理解してもらいます。そのうえでデータの操作性についてユーザーとよく相談します。これが理解されないと共有化が進まないだけでなく、仮に進んでもまた元に戻って(いつの間にか机の中にデータがある)しまうことになります。「皆の幸福は各人が少しずつ不便になること」といえます。

## ・壊れる危険性

共有化すれば壊れる危険性(誰か1人が壊してもすべてが壊れる)が高まり、壊れた時のダメージが大きく(壊した人だけでなく皆が困る)なることをユーザーに理解してもらいます。これにはデータのバックアップをとることで、バックアップにはコストがかかり、データが壊れないと「ドブにお金をすてた」と同じだということを理解してもらいます。そのうえで壊れる危険性と壊れたときのダメージ、そしてそのためのコストをはかりにかけてバックアップを取るよう提案します(セオリー13で述べるセキュリティ対策と同じです)。

## ・ぬすみ見

共有化すれば、悪い人もぬすみ見しやすくなります。これについても[セオリー13](#)を参照してください。

## ②データ量ニーズ

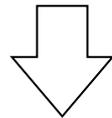
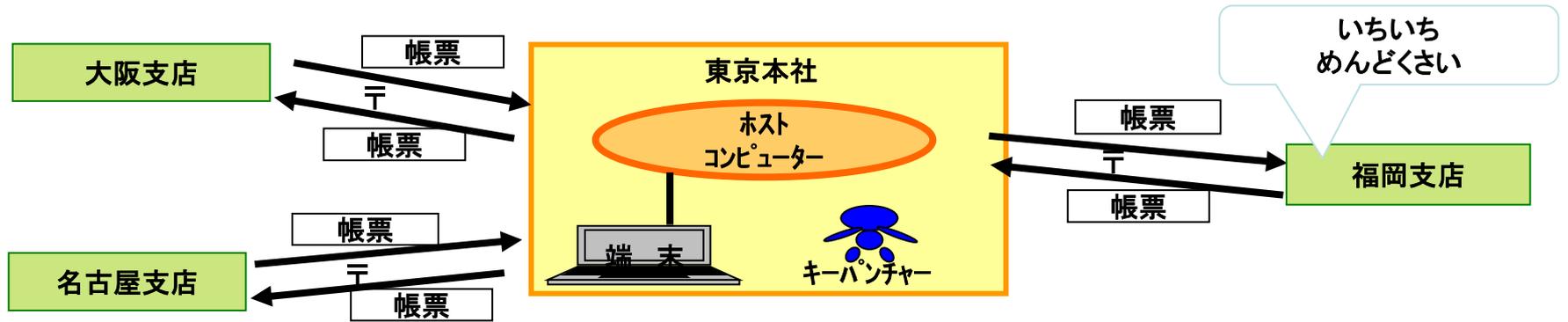
パソコンを中心としたクライアント・サーバー型の情報システムを入れて、2年もたてば、ほとんどすべての企業で「データ量が多すぎてどこに何があるのかわからない。そのためデータ操作のレスポンスが落ちて遅い。しかしどう整理したらいいかわからない」というニーズが生まれてきます。

この答えはわりと単純です。「いらないものを捨てる」ということです。それにはまずデータ定義のうちの「タイミング定義」をきちんとやり、見たいタイミングによってデータの持ち方を変えることです。例えば同じ「データ項目」である売上でも「毎日見る売上データ」、「月に1回見る売上データ」、「年に1回見る売上データ」はタイミングがちがうので、持ち方を変え、それぞれ「毎日捨てる」「毎月捨てる」「毎年捨てる」ルールをはっきりとさせることです。そのうえでバックアップデータ(決められた時以外に見るもの)でそれ以外の「イレギュラーなタイミングで見るニーズ」に対応します。

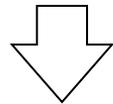
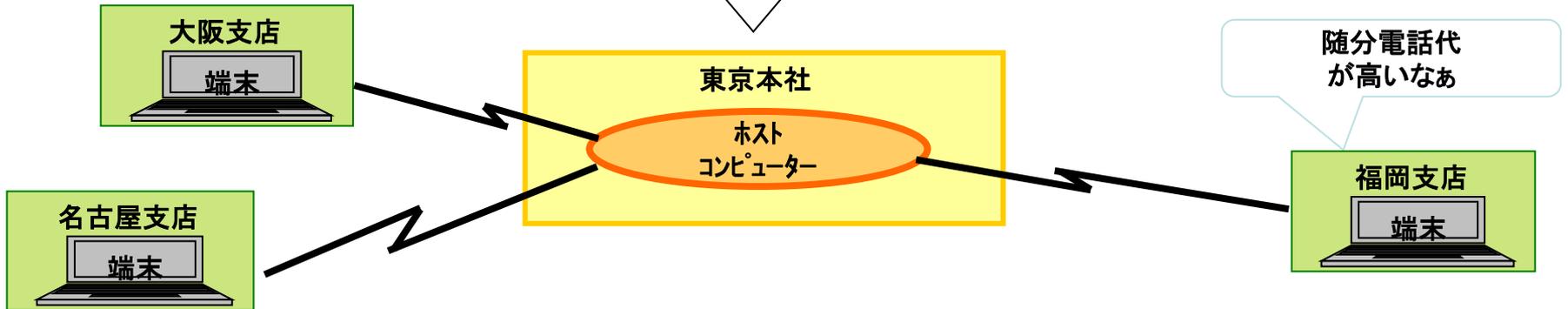
# THEORY11

常時接続は  
システムを変える

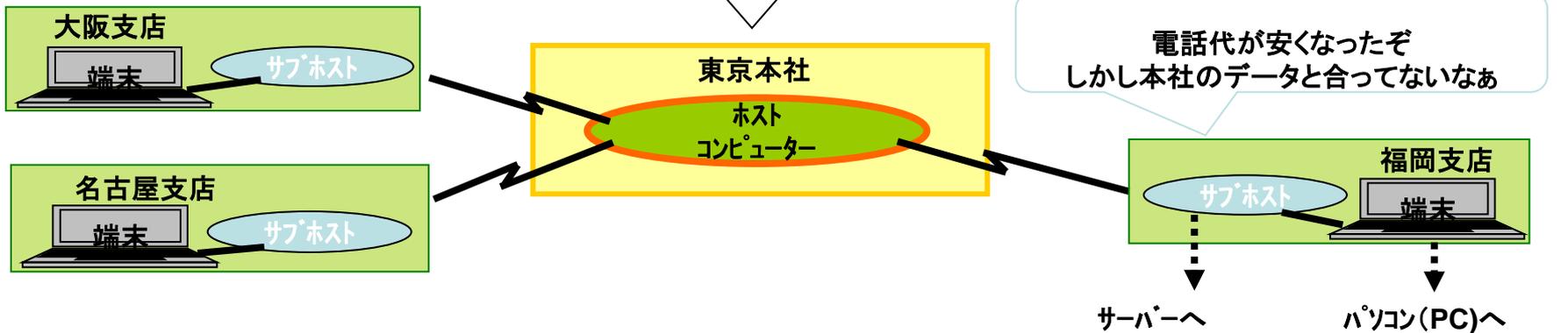
# 常時接続はシステムを変える

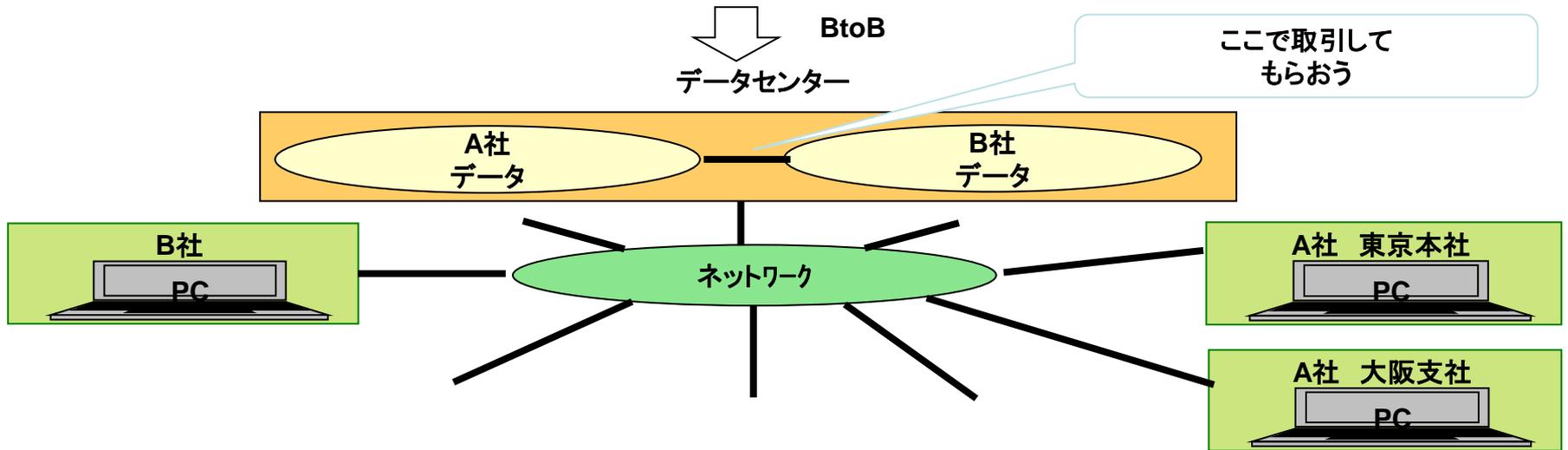
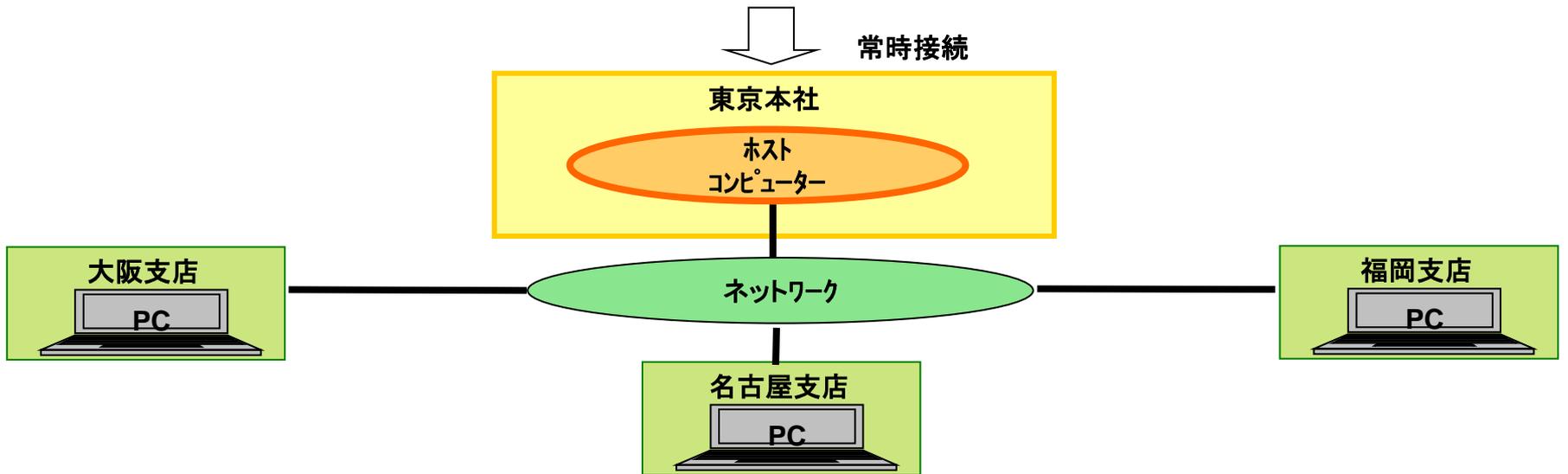


オンライン化



分散化





ネットワークに関するソリューションニーズは近年増大しており、ネットワークソリューションという言葉も生まれています。規制緩和、インターネット・携帯電話の普及・低価格化などで利用環境が変わったことがその理由といえます。ネットワークニーズは常時接続、インターネット、セキュリティという3つの分野に分けることができます。ここでは常時接続について考えます。

ネットワーク化はネットワークサービスの変化とともに進展してきました。①コンピュータ導入

大企業に大型のホストコンピュータが導入され、支店など遠隔地とのデータのやり取りは、帳票・郵送で行なわれました。郵送された帳票はホストコンピュータの端末からキーパンチャーというプロの入力者によって入力され、プリンターからの出力帳票はやはり郵送で遠隔地に届けられました。

②オンライン化

帳票を郵送でやっているるとタイムラグが出るので、遠隔地間のデータのやり取りが、電話のために作ったネットワーク線を使って行われるようになりました(これはオンライン化と呼ばれました)。しかし電話のためのネットワークを使っているので、電話用の課金ルール(つなぐ距離とつながっている時間で料金が決まる)が適用され、遠隔地の通信には膨大な費用がかかりました。

### ③分散化

この電話代というネットワークコストを節約するために、遠隔地にサブホストと呼ばれるコンピュータが置かれました。各支店では通常は支店内のサブホストコンピュータを使って処理し(内線電話や市内通話なので安い)、必要に応じて東京のホストコンピュータに電話をかける(市外通話をこの時だけ使う)ようにしました。このサブホストがサーバーへと進化していきます。

### ④常時接続

インターネットの普及でネットワークサービスの中心が音声からデータへとシフトしていく中で、次第に「どこからどこに何分かけたか」という課金ルールでは対応できなくなってきました(10倍のスピードのネットワークを開発すると料金は10分の1になってしまいます)。またそもそも接続という概念も怪しいものとなってきました(電話とちがって、相手が不在でもとりあえずどこかに送っておける)。そして誕生したのがインターネットを前提とした常時接続サービスです。これはネットワークに「つなぐ」という概念はなく(今までのネットワークはまず「つながる」ことを可能にして、普段は「つながらない」ようにしておきます。これを解除したときに課金するものです。)、いつもネットワークに「つながっている」という本来のネットワークの姿にし、課金も1ヶ月いくらというように固定料金にしたものです。つまりどこからどこへ送ろうと、何を送ろうと受けようと定額というものです。

常時接続の世界では「サブホスト」という中継地が不要となり、ネットワークの変身をせまります。これが近年のネットワークニーズです。

## ⑤BtoB

一方で企業間がネットワークされ、そこで取引がなされ、BtoB(Business to Business)とよばれています。

BtoBの問題点はセキュリティであり、企業間ネットワーク(送っている時)がその弱点となっていました。しかし、常時接続のネットワーク下で考えれば、BtoBを行なう複数の企業のコンピュータは1台でいいはずで、このコンピュータ内で取引をすれば安全性が高まるといえます。こう考えていくと取引や処理のコンピュータは1ヶ所に集中するほどよく、これをデータセンターといいます。個々の企業にホストコンピュータは不要で、データセンターからさまざまなサービスを受ければよいことになります。これがASP(Application Service Provider)と呼ばれるものです。

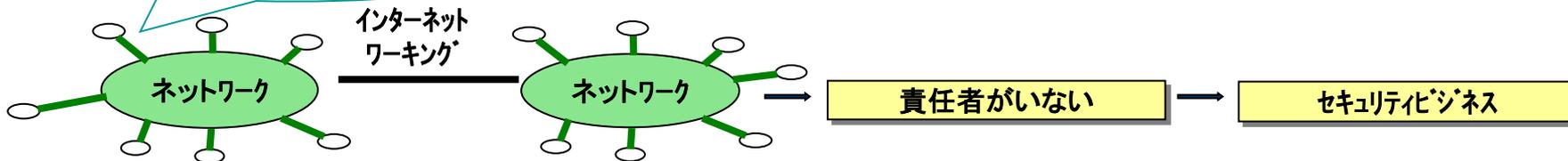
データセンター、ASPは単独企業が利用するものではないので、第三者が設立する必要があり、ソリューションベンダーが行なうべきビジネスといえます。ニーズ予想・インタビューでもこのニーズを意識して行う必要があります。

# THEORY12

**インターネットはセキュリティ、Web技術、  
プロモーションの3つに分けて考える**

## セキュリティの世界

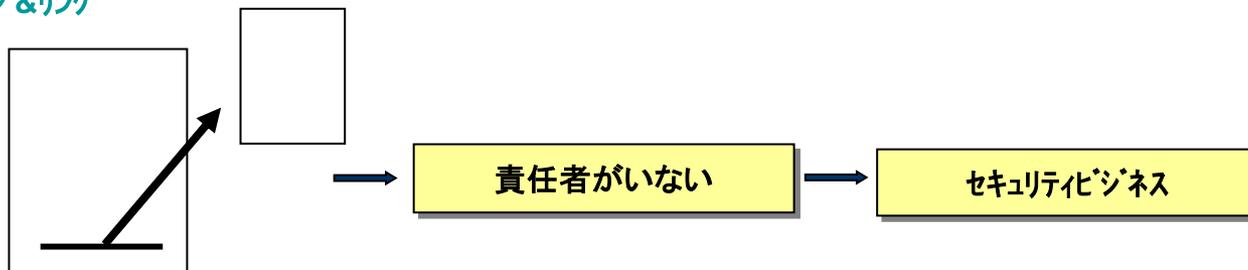
となりのネットワークのメンバーとも  
通信できる



## Web技術の世界

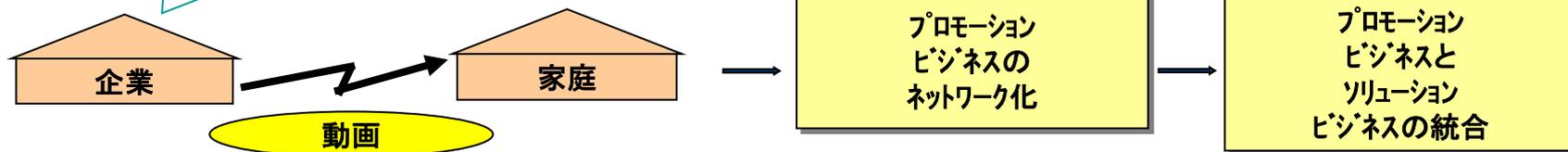
ホームページ&リンク

プロに頼むまでもなく自分で画面  
を作成できる



## プロモーションの世界

TV局に高い金を出して頼まなくても、自分でCMが打てる



ソリューションニーズから見た「インターネット」は次の3つの分野に分けることができます。

### ①セキュリティの世界

インターネットにはネットワークとネットワークがつながるといふ「インターネットワーキング」という意味があります。インターネットワーキングすれば、つながったネットワークのメンバー同士は自由に通信できます。しかし「自由に」というのは見方を変えれば、悪意を持った人とも自由につながり、かつその悪意を排除する責任者がいないネットワークになるということの意味しています。単独のネットワークサービスでは、必ずそのネットワークを作り、料金を徴収する主宰者がおり、これが責任者です。このネットワークで何かあればすべてこの主宰者の責任です。インターネットワーキングの世界では何かあっても責任者を特定することは難しいといえます。現在のインターネットは世界中がインターネットワーキングされたものであり、もっとも無責任なネットワークといえます。

ユーザーから見ればそんな無責任なネットワークでビジネスをやる気などなく、誰かに責任を取って欲しくなります。これがソリューションビジネスにおけるセキュリティニーズといえます。この対応については次のセオリー13で述べます。

## ②Web技術の世界

従来の情報システムではインタフェース、つまり画面などの設計にプロの技術が必要とし、これをSEが担当していました。しかしインターネットのWeb技術の世界では、誰でもホームページという画面を作ることができ(筆者の中学生の娘も作っています)、かつ画面の遷移などもリンクという技術を使い、クリック1つで可能にしています。これはシステム開発ビジネスから見れば大きな脅威といえます。セオリー11で述べたキーパンチャーがいつの間にか消え、入力という仕事はユーザーの仕事になっていったように、インタフェースを作るという仕事はユーザーの仕事になりつつあります。

しかしユーザーが勝手気ままに作っていると、セオリー10のデータベースの項で述べた共有化と同じ問題が起きてきます。他人の作ったインタフェースとうまくつながらない、データベースとつながらない...といったことです。Web技術の進展とともに、ソリューションニーズは画面作りから、画面と画面、画面とデータベース間のネットワーク(これをよくインフラといいます)の設計にシフトしていくことになります。これには従来のSEのシステム設計(各仕事の関係や分担をきちんと決めることがシステム設計といえます)という技術が活かされることになり、このニーズをとらえていくことが大切となります。

### ③プロモーションの世界

インターネットがブロードバンド化(ネットワークのスピードが上がり動画も自由に送れるようになること)が進むと何が起こるでしょうか。一番大きく変化するのは、企業のプロモーションスタイル、特にコマーシャルの世界といえます。コマーシャルに動画は必須であり、現在はテレビ局しか家庭との間の動画ネットワーク機能を果たせません。企業はここに膨大なプロモーションコストをつぎこんでいます。テレビでサッカーのワールドカップ、オリンピック、ニュース、コンサートなどがすべて無料で見られるのは、スポンサー企業がコマーシャルコストを負担しているからです。

インターネットをブロードバンド化して家庭へ自由に動画を送ることができれば、テレビ局よりも使い勝手が良い(いつでも送れる、双方向で相手の反応もわかる)ので、一気にインターネットへコマーシャルがシフトしていくと思われれます。このコマーシャルニーズが新しいソリューションニーズといえます。こうして広告代理店、テレビ局というプロモーションビジネスと、ネットワークを中核としたソリューションビジネスは統合してボーダーレスになっていくと考えられます。そしてソリューションニーズは仕事の合理化、高度化からプロモーションへとその軸足を変えていくこととなります。

# THEORY13

万全なセキュリティはない

財産1億円を守りたい



第1のセキュリティテーマ

いくら金庫を買えば良いのか

たとえば

500万円の金庫を買う

かぎが壊れたら1億円損する

合いかぎを作る

合いかぎを使ってあけられる

でも万全の金庫はない

どの程度のセキュリティが必要か



第2のセキュリティテーマ

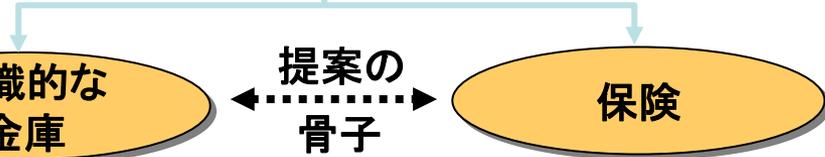


同じテーマになる

常識的な  
金庫

提案の  
骨子

保険



セキュリティというニーズはインターネットをはじめとするITが生んだものではなく、古来人間が誕生した時からあるニーズといえます。そう考えればインターネット、データベースなどのセキュリティは人間が何百年とかけて考えた「結論」を使えば良いことになります。1億円の現金(ITでは例えば投資して作ったデータ)を守る例で考えてみましょう。

1億円の現金(データ)を守るには金庫(ファイアウォールなどのセキュリティシステム。つまり壁を作って入口を1つにする。)とかぎ(その入口でその人が悪い人でないかをチェックする。パスワードなどを使う)を用意するしかありません。まず最初に困るのは1億円の現金(データ)を守るにはいくらの金庫(セキュリティシステム)を買うのが妥当かということです。難しいセキュリティの本によれば「リスクの発生確率(なくなったり壊れたりする確率)×リスクによって受けるダメージ(その時の損害額)」で「リスクの期待値」を出し、その金額以内に金庫(セキュリティシステム)の値段を押さえると書いてあります。しかしそんなものはソリューションビジネスでは使えません。そもそもリスクをすべて予測できないし、できても発生確率なんてどうやって計算したらよいかわからないし、損害金額も「なくては困るデータ」であれば計算のしようもありません。このソリューションニーズに対してソリューションベンダー、提案型SEは答えを用意しておく必要があります。

。

とりあえず500万円の金庫を買ったとします。次に考えるのは「かぎが壊れた時」です。かぎが壊れると1億円の現金(データ)が使えなくなり、1億円損してしまいます。かぎがなくても金庫があくなら、それはかぎとは言いません。500万円の金庫を買ったら1億円損するというリスクを抱えることになります。この答えは、「合いかぎ」を作っておくことしかありません。そしてこの合いかぎはノーマルなかぎを持っている人でなく(かぎを2つ持っていたら2つとも同時になくしてしまいます)、保管業者や、壊れた金庫を直すことのできる人、つまり金庫(セキュリティシステム)を作った企業が持つべきだといえます。残念ながら多くのセキュリティシステムはノーマルなかぎでなく、この合いかぎであけられてしまいます。

合いかぎのことを考えればわかるとおり、世の中に万全の金庫つまりセキュリティシステムなんてあるはずもありません。万全のセキュリティシステムがないとすれば、ユーザーのセキュリティニーズはどの程度のセキュリティが妥当かということになります。まずはっきりしているのは価格が高いほど強い金庫であり、守る現金が多いほど高い金庫にすべきということです。c

こうなると「いくらぐらいの」金庫(セキュリティシステム)を買うかという最初の課題に戻ります。この答えは1つです。それは「普通」つまり「常識的な金庫」です。1億円の現金(データ)を守るなら、普通どの程度の金庫(セキュリティシステム)を買うかという平均値のようなものが必要となります。そして結論はシステム投資額(守るべき財産)の一定比率をセキュリティにかけるべきということになります。筆者はソリューションベンダーには「こわれたら困るシステムはバックアップなどのインテグリティ(過失でこわれる)コストを含めて投資金額の10%を当てるべき。」とコンサルティングしています。そしてセキュリティ事故は「起きる」という前提で考え、保険に入っておくしかありません。保険は従来の保守サービスとも考えられ、ソリューションビジネスの一部となります。

セキュリティは見方を変えると従業員に対する経営者のマナーといえます。セキュリティ犯罪の多くは社内犯罪です。それは日本に悪い人が増えたのではなく、「セキュリティ犯罪をやりやすい環境」が整ってきているのです。経営者は従業員に対し、セキュリティ犯罪を「やりにくい」環境(「できない」環境ではない)を作る責任があるといえ、これがセキュリティに対する経営者の基本的ポリシーといえます。

提案型SEはユーザーに対して個々のセキュリティシステムを提案していく前にこの基本的考え方、ポリシーを理解してもらう必要があります。

## 第3章 インタビューモデル・ニーズ発見モデル

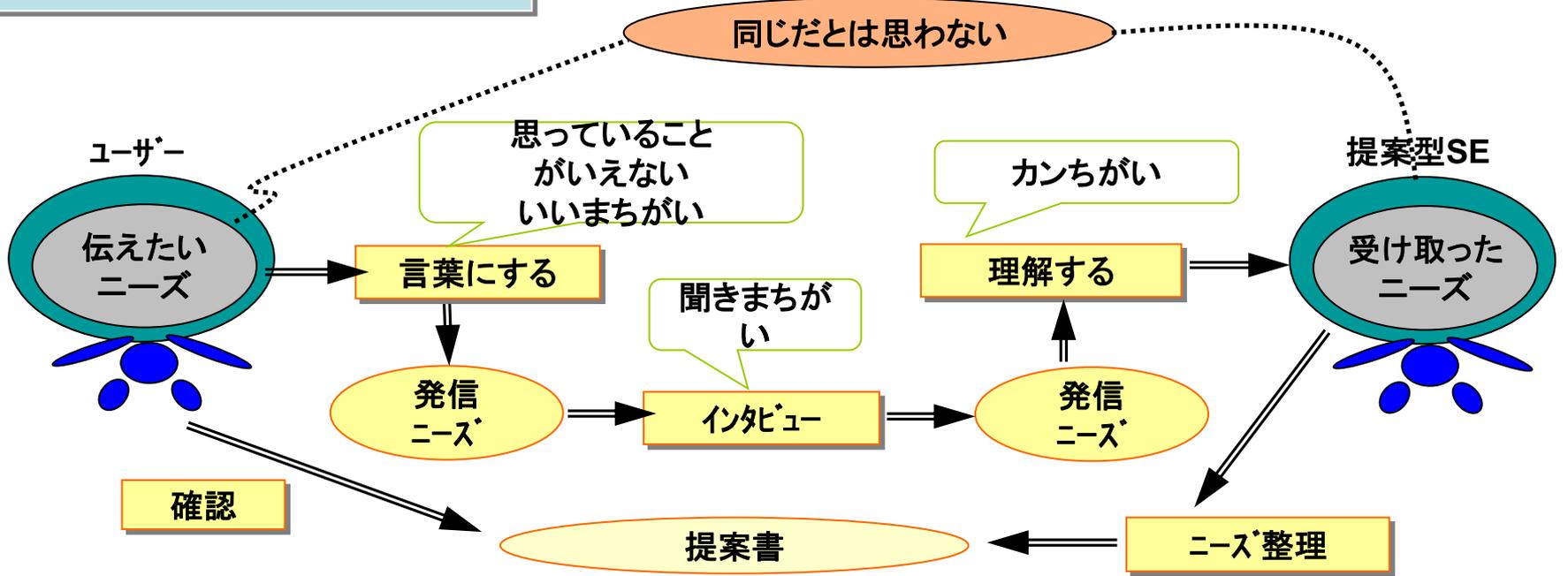
### トータルセオリー

インタビューで受け取ったニーズは整理して、ユーザーにフィードバックする。整理のポイントは「言ったとおり」「グルーピング」「重み」の3点。

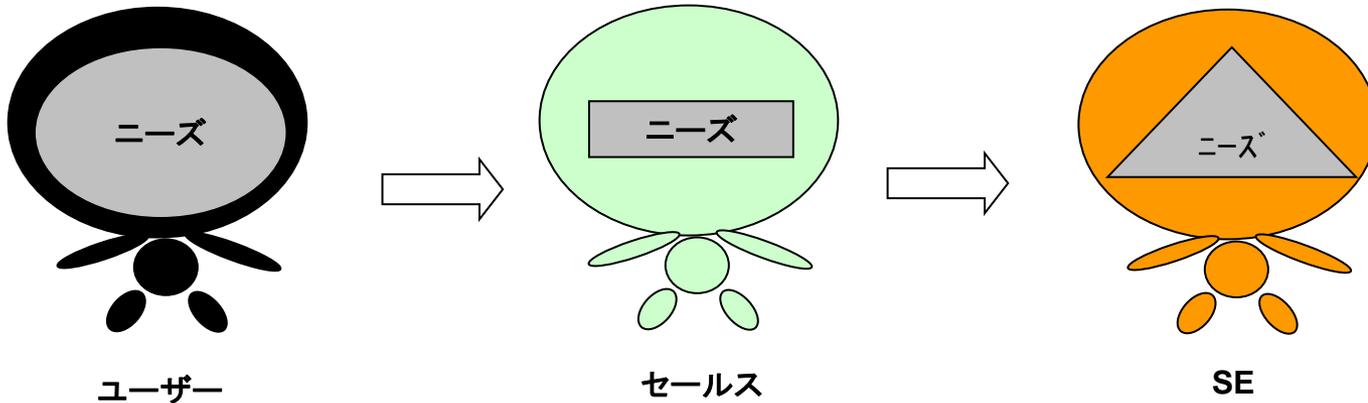
# THEORY14

「伝えたいニーズ」と「受け取ったニーズ」  
は違っている

# ユーザーインタビューの構造



# また聞き



ソリューションベンダーの経営者は、ソリューションビジネスがうまくいかない理由の1つにインタビューを挙げます。「うちのSEは口下手なのでインタビューがうまくいかない」「インタビューテクニックを教育して欲しい」

口下手な人はインタビューが下手なのでしょうか。インタビューはヒアリングであり、スピーチではありません。一般にしゃべるのが上手な人は人の話を聞くのが苦手ですし、しゃべるのが苦手な人は人の話を聞くのが上手です。

ニーズインタビューがうまくいかないとは「ユーザーニーズがうまく伝わらない」という状態を指していると思います。その原因はSEのインタビューテクニックのレベルが低いからでしょうか？そもそもインタビューテクニックとは何なのでしょう？「インタビューテクニックが高いレベルにある」とは、きっと「相手の話したくないこと、言い出しづらいことをうまくしゃべらせる」ことや、「自分の思う方向に相手をしゃべらせる」ことなどを指すと思います。しかし、ニーズインタビューにおいてユーザーがSEに話したくないことなんてあるのでしょうか。ユーザーにSEが望む方向のことを無理にしゃべらせて、後々良いことがあるのでしょうか。どうもニーズインタビューを勘違いしている人が多いといえます。ニーズインタビューの構造を考えてみましょう。

ニーズインタビューではユーザーはSEに「伝えたいニーズ」を持っており、隠しておきたいことなどありません。これが出発点です。この「伝えたいニーズ」をなんらかの形でメッセージ化（インタビューでは「言葉」というメッセージに）します。この時「思っていることが言えない」「いいまちがい」というエラーが発生します。思っていることを100%正確にモレなく話せる人など世の中にいません。ニーズ把握に文書でなくインタビューを用いるのは、文書では「思っていることを書けない」というエラーが発生し、文書のエラーの方がインタビューより発生確率が高いからです。

この「発信ニーズ」をSEがヒアリングし、「受信ニーズ」となります。もちろん「聞きまちがい」があります。そのうえでこの「音」の状態にある受信ニーズを、SEが頭で一生懸命理解しようとしてします。もちろん「カンちがい」は誰にでもあります。さらにこの時SEはこの受信ニーズを「忘れないために」メモを取ります。メモを取ることで気が散ると同時に、メモを取った安心感、さらには理解したことをまちがってメモるなどして、カンちがいは増幅します（我々コンサルタントはインタビュー結果を録音することもあるのですが、それでもカンちがいをします）。そしてやっとSEはニーズを受け取ります。

これだけエラーが起きるチャンスがあって、ユーザーが「伝えたいニーズ」とSEが「受け取ったニーズ」が「同一のものである」と考えるのは乱暴ですし、SEとよばれるエンジニアがやる仕事とはいえません。システムがトラブルが起きた時、いつももめるのは「言った、言わない」です。これは両者を同一として考えているために起こるといえます。

提案型SEは頭に入ってきた「受け取ったニーズ」は、ユーザーが「伝えたいニーズ」とはどこかがちがっているものとして仕事を進めていくべきです。しかし、まるっきり伝わらないと考えるならインタビューに行く意味はありません。経験則から、「インタビューで正確に伝わるのが1/3、忘れ去るのが1/3、誤解されるのが1/3」とよくいわれています。

こう考えれば、インタビュー後、「受け取ったニーズ」をユーザーにフィードバックして確認してもらうことが大切となります。フィードバックする時もう一度口で伝えるのでは意味がないので、SEの論理性を活かして、この受け取ったニーズをきちんと整理して、文書でフィードバックすべきといえます。このテクニックがセオリー16から述べるニーズ発見モデルです。

ニーズインタビューで決してやってはいけないのは「また聞き」です。セールスがユーザーからニーズを聞いて、セールスからそれをSEが聞くというものです。たださえゆがみやすいニーズがさらにゆがんで(セールスの都合の良いように)とんでもないものになってしまいます。

# THEORY15

タイプ別にインタビュー  
スタイルを考える

親分型

特徴

- ・明るい、前向き
- ・話すの大好き
- ・アイデア・ヒラメキがある
- ・時間を気にしない
- ・ほめ言葉に弱い

インタビュースタイル

- ・話しやすい環境を作る
- ・メモを取る
- ・相手に思いきりしゃべらせる
- ・問題点ではなく、アイデアを聞く
- ・システム化した後のイメージを聞く

バリバリ型

特徴

- ・仕事の成績が良い、やり手
- ・目標・達成が大好き
- ・時間に厳しい
- ・人に指示するのが好き
- ・ピリピリしたムードが好き

インタビュースタイル

- ・インタビュー時間を守る
- ・インタビューすることを文章にして渡す
- ・結論から言う
- ・案はいくつか用意しておく
- ・意見にさからわない

まじめ型

特徴

- ・おとなしい、無口
- ・冷静沈着
- ・論理的
- ・安全第一主義
- ・整理がうまい

インタビュースタイル

- ・インタビューに使う資料を前もって送る
- ・他社の事例を用意する
- ・結論を急がない
- ・リスク分析をしっかりやる
- ・インタビューの議事録を取る

SEが提案という仕事をやりたくない理由の1つに、「同じニーズを持っている企業に同じ提案」をしても、その提案を評価する人によって「GO」となったり「NG」となったりすることがあります。設計やエンジニアリングの世界では一定の性能のものは一定のパフォーマンスを発揮します。ここが大きく違う所です。なかでもインタビューは「する相手」つまりユーザーによって、うまくいくか、いかないかが大きく依存します。

ここで大切なことは人間には自分と異なるタイプ、異なる価値観を持つ人間がいることをSEが知り(SEの職場は多くの場合、同タイプ・同価値観の人が集まっているといえます。下のタイプでいえばまじめ型がほとんどです。)、そのタイプに合わせたインタビューを心がけることです。こう思うだけで精神的にも随分楽になります。インタビューという局面で考えると、相手をソリューションイメージにあるような3つのタイプに分けるのが妥当です。

## ①親分型

特徴はソリューションイメージにあるような人で「あの人どんな人」とその人を知っている人に聞くと、「あの人は思っていることと言っていることが同じ」と言われるタイプです。創造力、決断力(決める力というよりも有無をいわず決めてしまう力といえます)、実行力を持っており、上昇志向もあり、意思決定者にも多いタイプです。起業型の社長などはほとんどこのタイプです。

このタイプへのインタビューの基本は「相手に思い切りしゃべらせる」ことで、そのためにはなるべくフランクな環境で行なうと良いといえます。会議室よりも応接室、喫茶店などが良いといえます。そのうえで必ず「メモをとる」ことです。このタイプは人に「ほめられる」ことに弱いのですが、一般にSEは人をほめるのが下手です。そこでメモをとるのです。メモはなるべく大きなノートに、相手に見えるようにしてとります。こうして「貴方の話はタメになる。だからメモをとる」という相手をほめる姿勢をとります。聞く内容はその企業の問題点などマイナス面ではなく、アイデアです。夢と未来への希望を語らせるようにします。決してそのアイデアの理由や根拠は聞かないことです。このタイプは「強気のSE」よりも「無口でまじめなSE」を求めています。

## ②バリバリ型

まわりからは「やり手」「仕事ができる」といわれてる人で、実際仕事の成績も良いといえます。このタイプは「仕事を効率良く仕上げ、結果を出す」ことに大きな関心があります。そのため生産性を強く意識し、周囲にもそれを要求します。時間に厳しく、人に厳しく、仕事に厳しく、彼のまわりはピリピリしたものとなります。

このタイプへのインタビューは「要注意」です。1度嫌いになるとそのSEと2度と話すことさえしません。インタビューする前に十分に質問事項を準備し、それをペーパーにまとめ、インタビューの席に座った瞬間にそれを渡します。あいさつは不要であり、ましてや世間話など禁物です。(もちろんこのタイプの人でも世間話などしません) そのうえで結論から述べ、それに至った経緯はペーパーなどで渡します。また意思決定局面ではSE側で案を決めず、選択肢を用意し、相手に決めてもらいます。「人の決めたこと」はきらいで「自分で決める」ことが好きです。いわれたことには基本的には逆らわず、どうしても反論のある場合はインタビュー後に良く考えてからペーパーで渡します。

## ③まじめ型

おとなしく、無口でまじめなタイプです。SE自身に最も多く見られます。そういう意味ではSEがもっともつき合いやすいタイプといえます。しかしこのタイプの問題点は、慎重でなかなか意思決定しないことです。

インタビューではまず前日までに話す内容をまとめ、メールなどで送っておきます。そうすると相手はじっくり考えて当日には答えをすでに用意しています。彼にとって最大の関心事は新システムに関するリスクであり、セオリー22で述べるリスク分析をしっかりとやるしかありません。またインタビュー内容は議事録としてまとめると喜ばれます。

# THEORY16

システムを考える前に  
インタビュー結果を整理する

ニーズ列挙

グルーピング

重みづけ

スペック作成

インタビューワークシート

タイトル	ニーズ	重み	ソリューションスペック	その他対策
業務課職員の残業削減	<ul style="list-style-type: none"> <li>・業務課の職員の残業が多く、月末～月初は特にひどい、やめる人間も出て困っている</li> <li>・何時間もパソコンに向かって数字を入れていると頭が痛くなるし、間違いも多くなる</li> <li>・請求書出力に時間が膨大にかかってしまい、その間、パソコンが使えない</li> <li>・入金消し込みの手間が非常にかかる</li> </ul>	9	③から⑥で対応	
		5	④入金消し込みはネット資金サービスを受ける	
		3	③から⑥で対応	
		8	⑤で対応	
検針作業の効率化	<ul style="list-style-type: none"> <li>・検針員の検針モレや、メーターの引き算がまちがっていることが多いので手戻りがあって大変だ</li> <li>・雨の日に検針すると、台帳をぬれないようにするのが大変で、検針に時間がかかってしまい、業務課の人に遅いと怒られる</li> </ul>	9	③検針員にモバイル端末を持たせ現場でメーター値を入力する	<ul style="list-style-type: none"> <li>・請求書が検針時に出力されることを公示する</li> </ul>
		4	④中央センターで検針員の検診状況をリモートコントロールする ⑤請求書をモバイルから出力する ⑥新規メーターについてはテレメーターサービスを受ける ⑨モバイル端末に防水機能をつける	
窓口業務のサービス向上	<ul style="list-style-type: none"> <li>・大口ユーザーから窓口対応が悪いという評判を聞いている</li> <li>・最近、料金の間違ひに関する問合せが多い</li> <li>・払ったはずなのに督促状がくるといクレームも多い</li> <li>・月末はパソコンが混んでいてユーザーからの問合せに即答できない</li> </ul>	10	①社名、個人名を入力するとすべてカード形式で顧客台帳がディスプレイに出力される	<ul style="list-style-type: none"> <li>・大口ユーザーのWeb導入状況を調査する</li> <li>・対応者をテンポラリースタッフにする</li> </ul>
		8	②大口ユーザーについてはWeb上で自社の状況がすべてわかるようにする	
		8	③、⑥で間違いが減る	
システム修正の対応	<ul style="list-style-type: none"> <li>・パソコンのプログラムを変えようと思っても、開発会社の担当SEがやめてしまったとかで全然直せない。アフターサービスが悪すぎる</li> </ul>	8	①、②で対応	<ul style="list-style-type: none"> <li>・現システムのドキュメントを提供してもらう</li> </ul>
		7	⑧窓口の対応には1人1台のクライアントを用意する	
システム修正の対応	<ul style="list-style-type: none"> <li>・パソコンのプログラムを変えようと思っても、開発会社の担当SEがやめてしまったとかで全然直せない。アフターサービスが悪すぎる</li> </ul>	3	⑩システム保守契約を結ぶ	

インタビューが終了するとその結果を整理するニーズ発見モデルに入ってきます。このセオリーではまずニーズ発見モデルの意味および全体の流れについて述べます。そのうえで以降のセオリーでニーズ発見モデルの各作業について個別に説明していきます。

ユーザーが提案書をまるつきり読もうともしないケースはたった1つです。それは完璧に「ねらいをはずし」たからです。自社のねらいにあっていない提案書など見たくもありません。どんなに性能が良く、安く、信頼性が高いシステムでも採用しません。ユーザーがねらいをはずした提案書を見て思うのは「言葉が通じないSEとは一緒に仕事をやりたくない」ということです。経営者などが自社の情報システム部を通して、外部のSEと話したがるのは「言葉が通じない」からです。

この言葉が通じない理由は決してSEが「ユーザーの言っていること」がわからないからではありません。もしそうであればわかるまでユーザーに質問し、その後提案書を出すことで解決できますし、まじめなSEならそうしているはずです。提案書をユーザーに出して「はずした」のは「ユーザーの言っていること」とシステムの連携が取れてない場合がほとんどです。この連携をとるのがニーズ発見モデルの第一のポイントです。

SE側から見るとインタビューは苦手な分野です。この苦手の理由をつきつめてみると、インタビューという作業の本質がわかっていないからです。インタビューの本質はインタビューしている時にポイントがあるのではなく、その準備と整理にあるといえます。準備がニーズ予想モデルであり、整理がニーズ発見モデルです。インタビューした後の作業イメージがわかれば、何をどうやってインタビューすべきかが見えてきます。インタビュー後の作業がいつ誰がやっても同じやり方であれば、インタビューしている時に自分がインタビュー後やるべき事が頭に浮かび、自然と何を聞けば良いかがわかってきます。この「インタビュー後の作業を標準化する」ことがニーズ発見モデルの第2のポイントです。

ここではまず次のような簡単な例を使って、ニーズ発見モデルの流れを説明しましょう。セミナーなどでニーズ発見モデルの練習をする時も、現実的でややこしい例でやるよりも、まず単純で簡単な例でやってから本格的に練習した方が良いと思います。野球でいえば打ち方を習って、すぐにスピードボールを打たないで、まずトスバッティングをやって感じをつかむ所からスタートするという事です。

## ケーススタディ

### 〔A社の概要〕

A社は、地方都市をサービスする小規模ガス会社である。A社では、現在、10名の検針員が検針台帳をもって、各家庭のメーターを見て回り、結果を台帳に記入している。その結果は、毎日、業務課に集められ、業務課職員がそれをパソコンに入力している。毎月末、パソコンより請求書を発行し、各家庭へ送付する。

支払いは、窓口持参又は銀行口座からの自動引落としにより行ない、入金を確認されると、職員がパソコンのマスタに消込みを行なう。

### 〔A社からのインタビュー内容〕

経営者 「業務課の職員の残業が多く、月末～月初は特にひどい、やめる人間も出て困っている。大口ユーザーから窓口対応が悪いという評判を聞いている。」

業務課 「検針員の検針モレや、メーターの引き算がまちがっていることが多いので手戻りがあつて大変だ。何時間もパソコンに向かって数字を入れていると頭が痛くなるし、間違いも多くなる。

請求書出力に時間が膨大にかかってしまい、その間、パソコンが使えない。入金消し込みの手間が非常にかかる。

パソコンのプログラムを変えようと思つても、開発会社の担当SEがやめてしまったとかで全然直せない。アフターサービスが悪すぎる。」

検針員  
(外注員) 「雨の日に検針すると、台帳をぬれないようにするのが大変で、検針に時間がかかってしまい、業務課の人に遅いと怒られる。」

ユーザー問合せ  
窓口 「最近、料金の間違いに関する問合せが多い。また、払ったはずなのに督促状がくるというクレームも多い。月末はパソコンが混んでいてユーザーからの問合せに即答できない。」

さあインタビューが終わりました。ここで多くのSEはすぐにシステムを考えてしまいます。ということは上のインタビュー結果を使っていないということです。インタビュー内容がすべて頭に入っているわけではなく(上の内容を目を閉じて言うことはできないと思います)、それでシステムが提案できるということは、インタビュー内容を使っていない(もともとガス会社のシステムを知っていたか。実はこれが最悪のケース。これならインタビューせずにシステム案を先に持って行って意見を聞くべき)か、自分にとって都合の良いごく一部(例えば「引き算ミス」など)だけを使っているかのどちらかです。こうして提案にインタビュー内容が反映されなくなっていくます。

システムを考える前にインタビュー結果を整理する。これが基本です。インタビュー後の作業は大きく次の4ステップから成ります。

①ニーズ列挙(セオリー17参照)

まずインタビューで得た、ニーズを列挙します。

②グルーピング(セオリー18参照)

列挙されたニーズをグルーピングします。この段階でニーズをワークシートへ転記し、グループにタイトルをつけておきます。

③重みづけ(セオリー19参照)

ニーズの1つ1つに重みをつけます。

④スペック作成(セオリー20参照)

重みの高いニーズからシステムスペックを考えます。

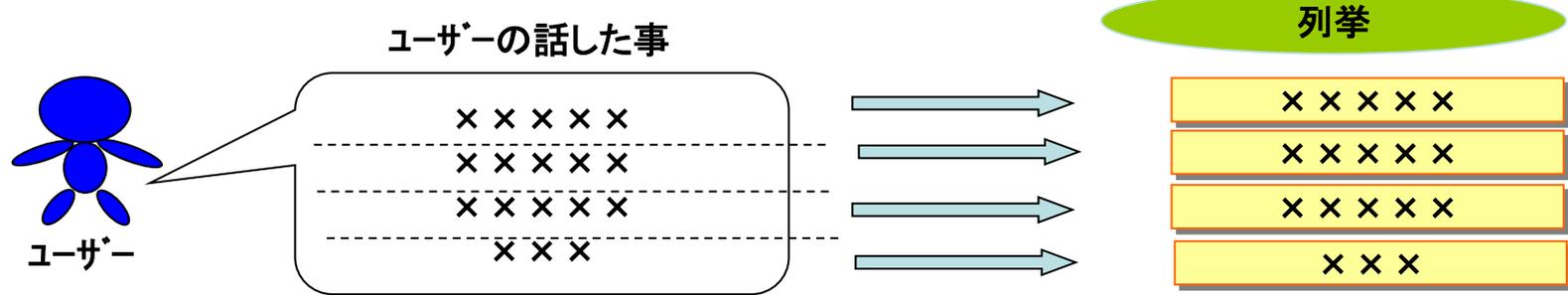
以降のセオリーでその細かいやり方を説明していきます。先ほどのケーススタディについてこのやり方を忠実に実行すれば、ソリューションイメージのようになります。本書ではこの整理に使うシートのことをインタビューワークシートと呼びます。

上の①～③のステップはニーズ発見モデル(③の重みづけで真のニーズがやっと発見できます)であり、④はニーズ解決モデルの第1歩となります。

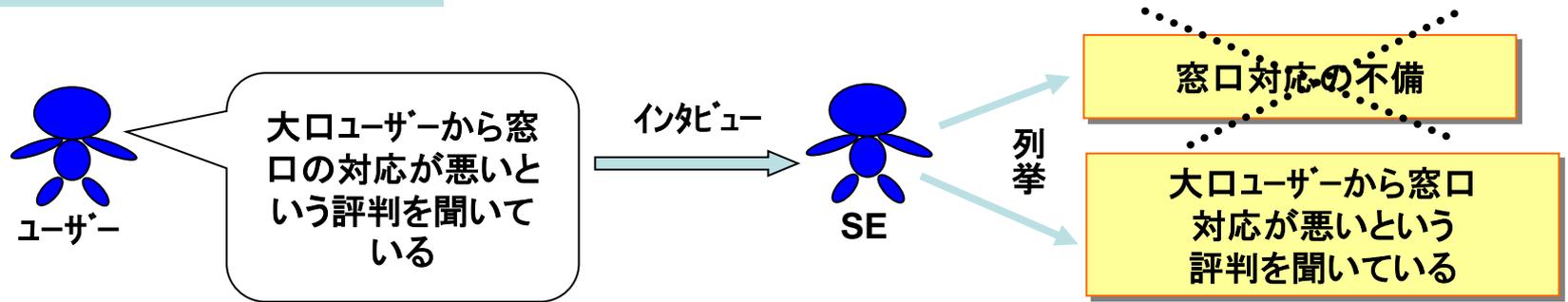
# THEORY17

ユーザーの言ったとおりに  
列挙する

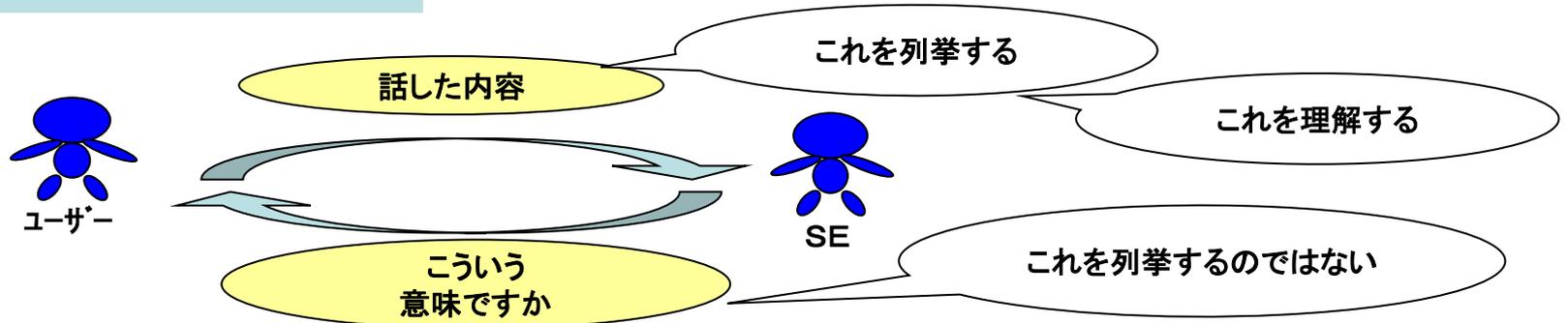
# 列挙のイメージ



# 言ったとおりに書く



# 言いかえない



ニーズ列挙のフェーズは次のようなステップで進めます。

### ①ニーズらしきものを列挙する

インタビューが終わったら、でき限り速やかにこのステップに入ります。ここでのポイントは「らしき」と「列挙」です。

この段階ではどうしてもユーザーから聞いたニーズを分析したくなります。「このニーズは大きすぎないか」「そのニーズとこのニーズは原因と結果の関係だ。こちらが諸悪の根源だ」「そのニーズとこのニーズは少し重なり合っているぞ」「このニーズは一体どういう意味だ」……

これらのことを一切意識せず、ユーザーから聞いたことをただ「列挙」、つまり箇条書きにしていきます。「これはニーズではない」といってはすたがる人がいますが、これが非常に大切な情報であることも多いといえます。そこで「ニーズ」ではなく「ニーズらしきもの」を列挙する、つまり聞いたことはすべて列挙するようにします。列挙する媒体は後でこのニーズの順番を考えたりしますので、シールのついたラベル、ポストイットなどがあれば便利です。また1人でやる時はパソコンのワープロソフトかプレゼンテーションソフトを使えば、順番変え、コピーなどが便利です。

## ②モレがないか

次に列挙されたニーズにモレがないかをチェックします。この時人間は色々なことが気になって、色々な作業を一度にやろうとしますが、モレ以外のチェックはしてはいけません。インタビューを思い出し、メモを見て「モレがないか」をチェックします。ここでもれると二度とニーズは復元できません。

## ③ユーザーがわかるか

次に挙げられたニーズらしきものが「わかるか」というチェックをします。ここで大切なことは「誰がわかるか」をチェックするかです。SEが列挙したのですから、自分で書いたことはわかるはずですが、もしSEとしてわからない所があるならば、このステップで考えるのではなく、インタビューの時にわかるまで質問するしかありません。よくインタビューの時に「こんなことを聞いたら恥ずかしい」と思い、後で調べてみようと思うSEがいますが、これはもっともいけないことです。「無知で恥ずかしい」と思う気持ちは大切に、これが人間を成長させるのですが、「恥ずかしいから聞かない」のではなく「恥ずかしいけど聞くしかない」のです。

では誰がわかるかという主語はユーザーです。この列挙したニーズはセオリー14で述べたようにユーザーへフィードバックします。そこでわかるかです。「ユーザーがわかる」ように書く最良の手段は「ユーザーが言ったとおりに書く」ことです。人間はどういうわけか「言ったとおりに書く」ことをきらい、何とかまとめようとします。まとめても何の幸せもなく、少し「情報が歪む」だけです。「言ったとおりに書く」のは非常に難しい仕事です。言ったことなど正確に覚えてないし、言ったとおりになどともメモれません。仮に録音してもそのニュアンスがわからなくなってしまいます。それでもがんばって「言ったとおりに書く」努力をします。

よくインタビューでユーザーの話した内容が理解できない時「それはこういう意味ですか」と聞き、ユーザーが「そうです」と答えることがあります。この時コンテンツが同じで、表現の違う情報が2つあることになります。列挙では、はじめの「ユーザーの話した内容」をそのまま書きます。そしてこの内容をSEは理解します。自分のわかりやすいように表現を変えてはいけません。

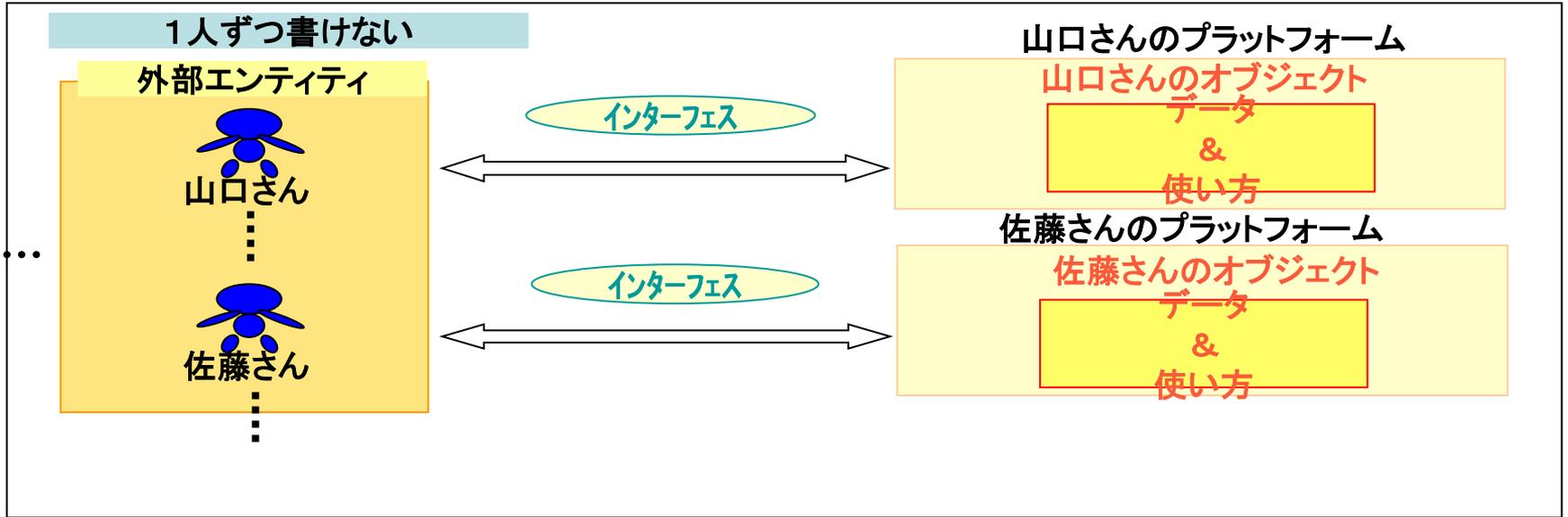
#### ④前後関係なしでわかるか

最後に、列挙されたニーズが「それだけで」わかるかをチェックします。この後でニーズの順番を変えてしまうので、単独でわかるか、細かく切りすぎているかをチェックします。もし単独ではわかりずらかったり、誤解する可能性のあるものは他のニーズとくっつけるようにします。ただし「なるべくくっつける」のではなく、①の「なるべく離す」のが大切で、どうしても切り離すとわかりづらいもののみをくっつけるようにします。

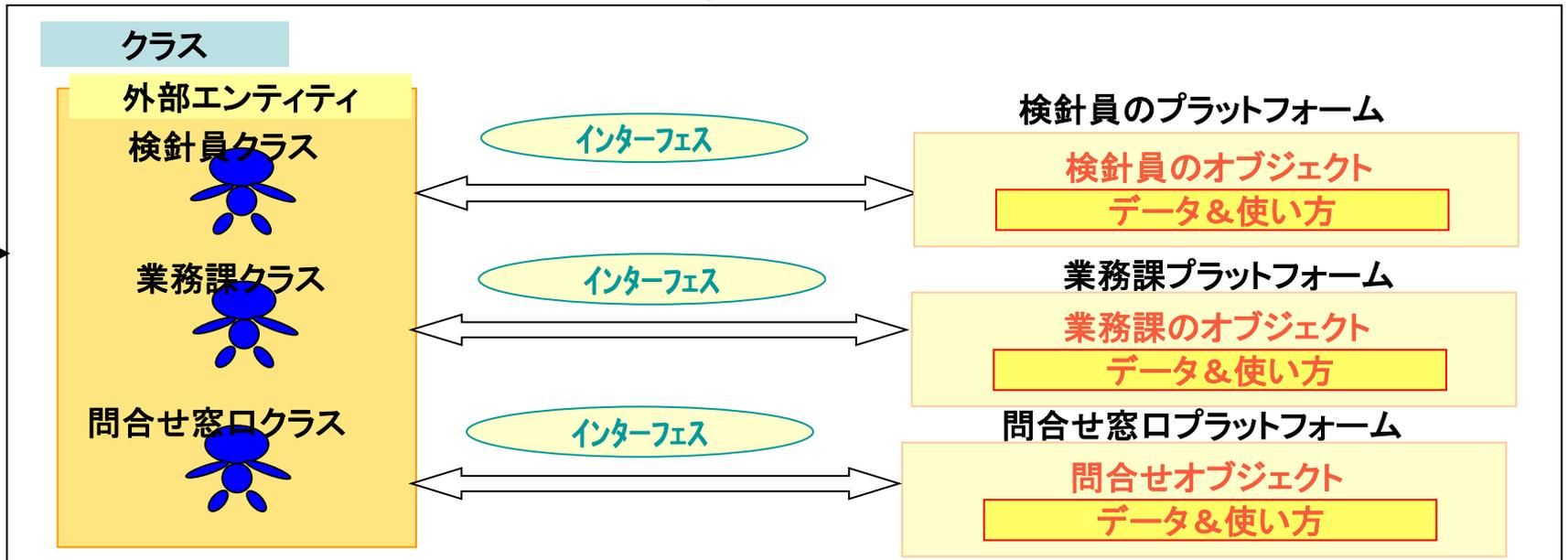
# THEORY18

グルーピングは外部エンティティの  
クラスを作ること

グルーピング



グルーピング



ニーズの列挙が終わったらこれをグループに分けます。グルーピングは関連のあるニーズを同一グループにしていくということではなく、必ず業務単位(業務の分担である組織単位といっても良い)にするということを体で覚えてしまうことです。普通は業務単位にニーズをグルーピングしません。例えば「セールスが売上データをデタラメに入れているので、経理部が売上が合わなくて困っている」とニーズをインタビューしたとします。この結果「セールスが売上データをデタラメに入れている」と「経理部が売上が合わなくて困っている」という2つニーズが挙げられますが、これを同じグループにしないことです。セールスと経理部は全く違う業務ですし、全く違う組織です。

なぜ業務別にグルーピングするのかを説明しましょう。ニーズインタビューの目的は「ユーザーの業務を理解し、ユーザーの課題を分析し、最適な業務方法を確立し、業務改善しよう」というものではありません。提案型SEはユーザーの業務を「やったことがない」ことが前提ですし、やったことのない仕事の業務改善など提案できるわけもないし、仮にできたとしてもその新しい業務方法に責任がとれません。

ニーズインタビューの目的は「システムスペックを作ること」だけです。システムスペックを作ることのできないユーザーのために、システムスペックを作り、そのシステムスペック通りのシステムを提供するのがソリューションビジネスです。

システムスペックはセオリー3で述べたようにオブジェクトとプラットフォームから成ります。ニーズインタビューではこれを直接聞きたいのですが、それができないので、かわりにニーズを聞いて、そのニーズからこのシステムスペックを提案型SEが提案するわけです。またユーザーに提案型SEが考えたシステムスペック(オブジェクトとプラットフォーム)でどうですかと確認しても、ユーザーは判断できません。そこでそのオブジェクト、プラットフォームの使用感つまりインタフェースを確認してもらい、そのインタフェースでニーズが解決できるかをユーザーに判断してもらうことが提案といえます。

そう考えると対象となるユーザー1人1人に対してすべてインタフェースを作る必要があります。したがってインタビューでまず大切なことは、一体誰が今回のシステムの対象ユーザーかということを確認することです。これを「システム化の領域」といいます。この領域の外にあるユーザーや他システムなどを「外部エンティティ」といいます。

しかし、外部エンティティ1つ1つにインタフェースを定義するのは大変なので、似たような外部エンティティは1つにまとめてグルーピングし、グループの共通部分と各メンバーの固有部分に分けて考えるのが自然といえます。このグループをクラスといえます。クラスはなるべく同一のインタフェースを持っている外部エンティティにする必要があります。この同一性を考えると同一業務をやっている外部エンティティ(ユーザー、外部システム)をグループ化のキーにすればよいことになります。この同一業務をやっているグループが組織ともいえます。

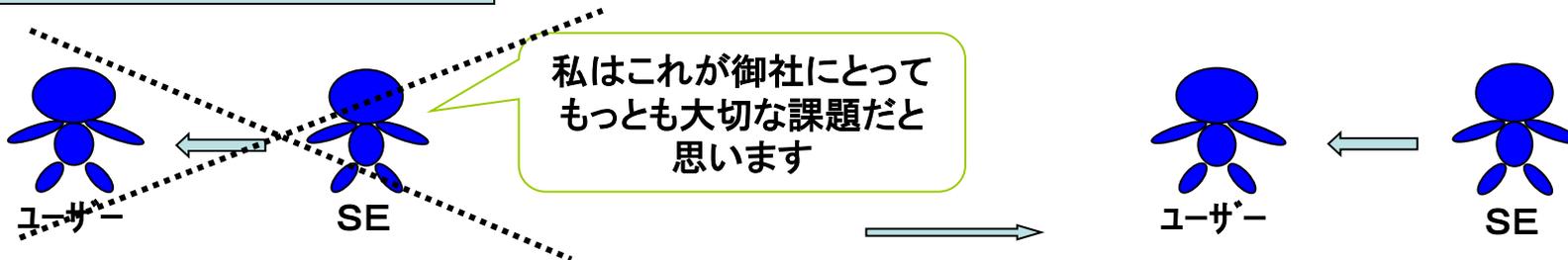
先ほどのガス会社の例で言えば、従業員や検針員1人1人のインタフェースを作るのではなく、検針員クラス、業務課クラス、問合せ窓口クラス...としておいて、このクラス単位にインタフェースを作ります。こうしてユーザーである検針員の山口さんに必要なインタフェースは、検針員クラス共通のものはそれを使い、それ以外の山口さんだけ固有ものは個別に考えるというものです。インタフェース、オブジェクト、プラットフォームをクラス別に考えるなら、それを考える原点となるニーズも当然クラス別に整理すべきです。

さらに、検針員・業務課・問合せ窓口に共通なニーズは共通クラスを作ります。共通クラスは各クラスの上位にあたるもの(スーパークラス)であり、検針員・業務課・問合せ窓口の各クラスに共通のサービスを提供するオブジェクトに関するニーズになります。つまりサーバーのサービス内容になるわけです。

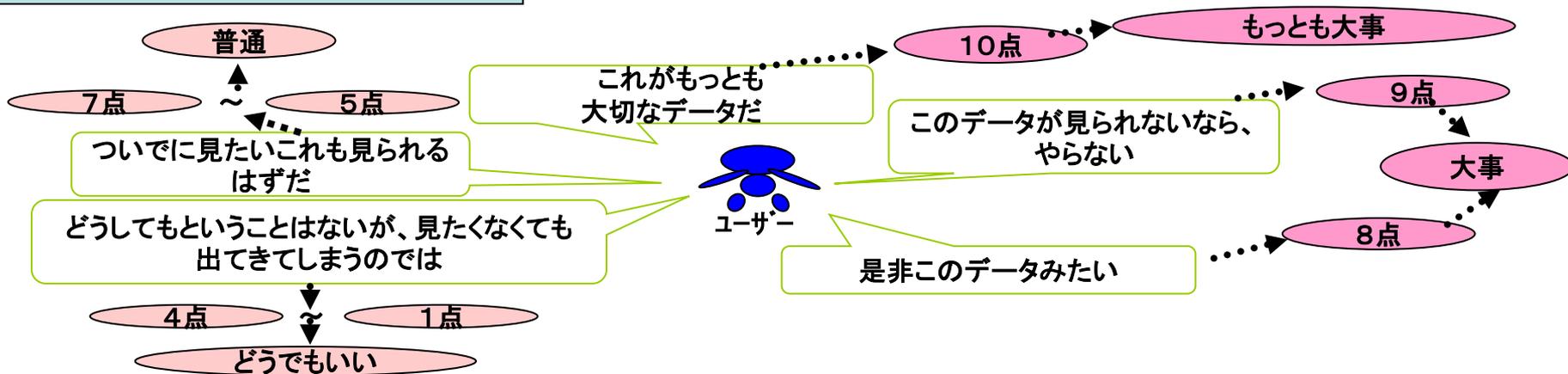
# THEORY19

重みづけは  
「インタビューでのニュアンスの定量化」

## Who & When & Where



## How & What



## Why

10点の対策が打てないなあ



SE



まあいいや  
やれるものを  
提案しよう



もう1度ユーザーに相談しようこれが結論でるまでに9点にはいかない

グルーピングが済んだら列挙されたニーズ「1つ1つ」について(グループ単位ではなく)重みをつけていきます。

### ①Who...誰がつけるのか

「ニーズの重みづけ」つまりどのニーズが大切かの判定ができるのは、冷静に考えればユーザー以外にいません。多くのユーザーがいたらその対象ユーザーたちの責任者です。全社がシステム化対象であれば社長、営業部のシステムであれば営業部長です。

### ②When & Where...いつどこでつけるのか

したがって重みをつけるのはニーズインタビューの時です。ここでニーズを聞くだけでなく、その重み、大切さを聞かなくてはなりません。インタビューが終わり、帰社してから上司と相談して重みづけをするSEがいますが(重みをつけるだけ良い方です。重みという概念を持っていない人もいます)、ナンセンスです。上司がいくら経験豊富であっても、やったこともない仕事(ユーザーの仕事)に関するニーズの重みなどつけられるはずもありません。重みづけとはインタビューでの「大事、少し大事、どちらでもいい・・・」といったニュアンスの定量化だということを理解してください。

### ③How...どうやってつけるのか

重みは基本的には点数なのでしょうが、何点法がよいのかを筆者は随分試行錯誤しました。多くの問題解決技法では3段階のものが多いといえます。感覚的には「大事、普通、どうでもいい」というのは人間の言葉のニュアンスと一致しています。しかしニーズインタビューでは粗すぎます。後に述べるように粗いとどうしてもSE側でやりたい所、できる所から考えてしまいます。人間にはマジック7というものがあり、7つを越えることは一度に認識できないといわれています。しかし7点法というのは直感的ではありません。実際にやってみて出した結論は以下の通りです。

- ・10点法でつける(10点がもっとも大事)
- ・まずすべてのニーズから1つだけ10点を選ぶ(相対評価:10点は1つ)
- ・残りを大事(9、8)、普通(7~5)、どうでもいい(4~1)の大きく3段階にし、かつ「どうでもいい」ほど細かくつけ、実質的に7段階程度におさめる(絶対評価:9点のニーズはいくつあってもよい)

#### ④Why...なぜつけるのか

重みづけでもっとも大切なポイントはこの「なぜつけるか」です。これは次の2点のためです。

(i)どのニーズからシステムスペックを考えていくのかを決める。

(ii)どのニーズにコストをかけるのかを決める

(i)はこれを決めておかないとSEは必ず「できること」から手を打っていきます。ニーズを20個聞いて19個について対策を考えたので「大体良いだろう」とします。しかし残った1つのニーズが10点でもっとも大切なことかもしれません。そして10点対策を打つのが難しいニーズであることが多いといえます(だからユーザーの方で解決できずに残っていたのでしょう)。そうだとすればユーザーの意見は当然「この10点が解決しないのなら、そのシステムをやってもしょうがない」となってしまう。

(ii)は「何に金をかけるか」はニーズの大きさに依存しているということです。セオリー4で述べたように、ソリューションにおいては投資金額を決め、その範囲内でやるべきものを順にやっていくという形になります。そしてこの順序はニーズの重さに依存していると考えられます。ユーザーは10点の結論が出ないうちに(「やらない」、「できない」も結論です)、9点を考えないということです。

#### ⑤What...何につけるのか

ニーズの「何に」重みをつけるかですが、オブジェクトの中のデータ項目単位が妥当でしょう。そう考えるとインタビューニュアンスの定量化ですので、ソリューションイメージのような「感じ」になると思います。

## 第4章 ニーズ解決モデル

### トータルセオリー

提案型SEはユーザーにニーズ解決のためのベストな案を提示することがテーマではない。案の選定に関する情報、リスク、解決のための考え方を提示し、ユーザーとともに考えていくことが基本。

# THEORY20

ソリューションズペックは  
稼働責任を負う

# 責任の所在

ユーザーの全責任

中身を確認し  
ユーザースペック  
責任を負う

ベンダーは  
稼働責任

ユーザーの責任  
のもとに実行

ユーザーが言ったこと

ベンダーの  
ビジネスとしての提案

ベンダーのビジネス  
以外の提案

インタビュ  
ークシート

タイトル	ニーズ	重み	ソリューションスペック	その他の対策
------	-----	----	-------------	--------

# ニーズ同士の関連

10点のニーズを解決するには

これらのニーズを解決すればOK

その解決策を  
ここに書く

インタビュ  
ークシート

タイトル	ニーズ	重み	ソリューションスペック	その他の対策
—	×××××	9		
	×××××	10		
	×××××	7		
	×××××	6		
	⋮	⋮		

インタビュ  
ークシート

タイトル	ニーズ	重み	ソリューションスペック	その他の対策
—	×××××	9	①で対応	
	×××××	10	①××××× ②××××× ③×××××	
	×××××	7	②で対応	
	×××××	6	①②で対応	
	⋮	⋮		

重みづけが終わるとスペック作りに入ります。ここからが課題解決モデルですが、第1段階のスペックの基本的方向を決める作業では、引き続きセオリー16のインタビューワークシートを使います。ポイントは以下の通りです。

### ①責任の所在

インタビューワークシートは大きく2つの領域に分かれます。左側(タイトル、ニーズ、重み)はユーザーの「言ったこと」が「言ったとおり」の言葉で書いてあり、すべてユーザーの責任です。後でユーザーに「そんなことは言っていない」と言われても困ります。右側は提案型SEの「考えたこと」つまり提案です。提案はさらに次の2つに分かれます。

#### (i)ソリューションスペック

ソリューションベンダーが有料で行うサービスに関するシステムスペックを書きます。ソリューションベンダー側はこのシステムスペックについては最終的に稼動責任(このスペックどおりに動かなかったらソリューションベンダーが動くまで責任をとる)を負います。ただしスペックの中身、つまりこのスペック、有料サービスで良いかをユーザーが確認し、その内容(そのスペックでニーズが解決できるか)についてはユーザーが責任を負います。

## (ii) その他の対策

提案型SEが考えたものですが、ソリューションベンダー側としてはビジネスの対象としないもの、つまりその責任をとれないものです。提案型SEが「このシステムをスムーズに運用していくためには、情報システム部の要員を現在の10人から20人に増員する」という提案を考えたとします。これがソリューションベンダーのビジネスの対象であり、例えば「組織コンサルティング料として50万円下さい」というのであれば、これはソリューションスペックです。この場合は「20人に増員して何か問題が生じた場合、ベンダー側でユーザーが納得するまでフォローする」という約束をしているのと同じです。もし「増員したらどうですか？ 我社のビジネス外のことですが」というなら「その他の対策」として、もちろんユーザーから料金はいただきません。

その他の対策については、ユーザー側の責任のもと実行(もちろん実行しなくても可)されます。

## ②10点から考える

セオリー19で述べたように重みづけの1つの意味は考える順番を決めることでした。したがってソリューションスペックは10点のニーズから考えていきます。10点のニーズを解決するスペックがうまく浮かばなかったら、そこで作業を一旦止め、その旨をユーザーに伝え、どうしたら良いかをユーザーと考えます。10点のスペック作りが終わらない限り9点のニーズにいきません。

10点のニーズを解決しようと思ったら、9点以下のいくつかのニーズを解決すれば10点のニーズが解決できるということはよくあります。この時、ニーズ同士の関連を分析して、「これが諸悪の根源だ。」「これとそれは原因と結果の関係になっている」・・・と分析していくのではありません。ソリューションビジネスでは業務分析などをやっている場合でなく、(ユーザーもそんなことは期待していません)なるべく効率的にシステムスペックを作るべきです。それがコストを落とすことになり、ユーザーに大きなメリットをもたらします。

この場合は(10点を解決することになる)9点以下の解決策を10点の欄に書き、それぞれのスペックに番号をふっておきます。9点以下のニーズを解決しようと思ったのではなく、10点のニーズを解決しようと思ったはずです。9点のニーズにきたら、その解決策を考え、すでに10点のスペックで解決しているなら、そのスペックの番号を書き、例えば「②で対応」とします。(セオリー16のインタビューワークシートを参照して下さい)

### ③システムスペックの記述方法

システムスペック作りは、セオリー8のデータ指向アプローチに沿って進められていきます。その第一ステップとして現状調査を行いながらデータ定義を固めて、入力設計、オブジェクト定義へと進んでいきます。インタビューワークシートではデータ定義によってもたらされるインターフェースをイメージして書いていきます。ここでの留意事項は以下のとおりです。

#### ・ソリューションシーンが浮かぶこと

ユーザーがそのスペックを聞いて、ソリューションシーンが浮かぶかどうかポイントです。SEから見ればそのニーズを解決しているシーンを絵に書けるかどうか「スペック」と「notスペック」の分かれ目です。よくどこまで細かく書くのかとSEから質問を受けますが、その答えは「ユーザーがそのソリューションシーンが浮かび、それならニーズを解決できる、またはできない」ということが判断できるまでです。

#### ・目標はダメ

「notスペック」の代表例が「目標」です。例えば「窓口対応が悪い」というニーズに対して「窓口対応を良くする」は目標であり、そのシーンの絵が書けません。これに対し「社名、個人名を入力するとカード形式で顧客の利用実績を見ることができる」はそのシーンを絵で書くことができるので、スペックといえます。

## ・タイトルはダメ

次に「notスペック」に多いのはシステム名を決めるだけのものです。先ほどの例でいえば「窓口対応システムの構築」というものです。これは何かを決めているようで何も決めてません。

## ・自動化はダメ

スペックの表現によく自動化という言葉を使いますが、これも不可です。先程の例では「窓口対応を自動化する」といったものです。自動化にはピンからキリまであります。ユーザーはもっとも高度な自動化(顧客の問合せにコンピュータが無人で応答してくれる)をイメージし、SEはもっとも低コストの自動化(顧客番号を入れると「自動的に」その顧客データが出る)を考えています。これが後々システムトラブルのもととなります。

## ・データベース化はダメ

データベース化も不可です。そもそもデータベース化とはファジーな言葉であり、共有化なのか、コンピュータ化なのか、データの操作性を上げることをいうのかなどさまざまな意味を持っています。これでは何も決まっていなくて、SE、ユーザーが決まっているような気持になっているので、もっと悪い状態といえます。多くの場合SEとしては「コンピュータを使いましょう」位の気持で「データベース化」と言っていますが、これでは提案とはいいません。そもそもユーザーはコンピュータを使うつもりでSEにニーズを話しています。ユーザーはコンピュータをどう使っているかわからないので、SEを呼んで提案を求めているのです。

#### ④説明しない

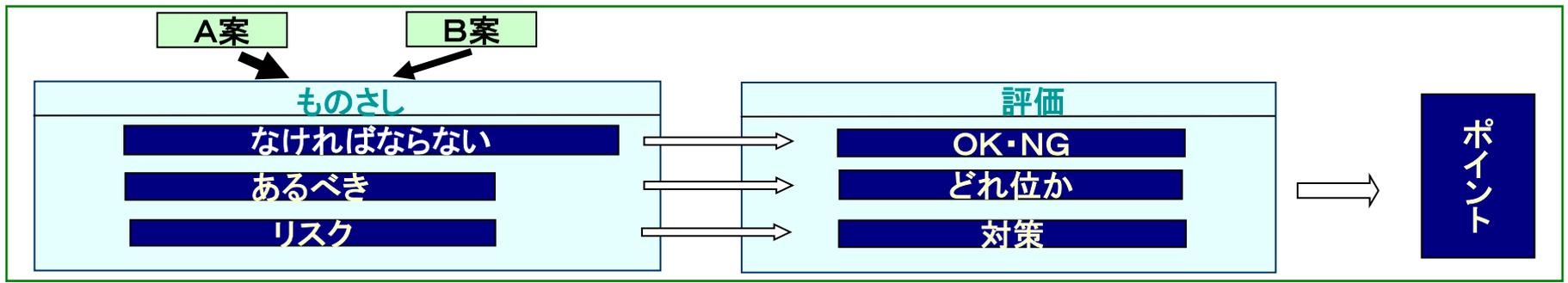
この段階では何人かのSEが集まって知恵を出し合うことも多いといえます。この時、口で一生懸命説明し、その後でインタビューワークシートに書こうとしますが、逆です。まずインタビューワークシートに書いてから説明させます。(媒体は紙だけでなく電子白板、パソコンどれでも同じです)。そのうえで書いた内容と説明していることが違ったら、書いた内容を変更します。これから多くの人携わってシステムを作っていきますが、この後の作業では、このワークシートだけを使っていきます。後に残らないものは言ってもしょうがないし、トラブルの元です。

#### ⑤結論を出すのではなく、アイデアを出す

この局面は提案するシステムスペックを決定しているのではなく、ソリューションのアイデア、あるいは他ベンダーに負けないアイデアを出し合っているのです。アイデア列挙では「人のアイデアを否定する」のは禁じ手です。「人のアイデア」を参考にしてもっと良いアイデアを出していくことです。アイデアはANDでもORでももちろんOKです。アイデアを出すのは大変ですが、それをまとめるのは簡単です。

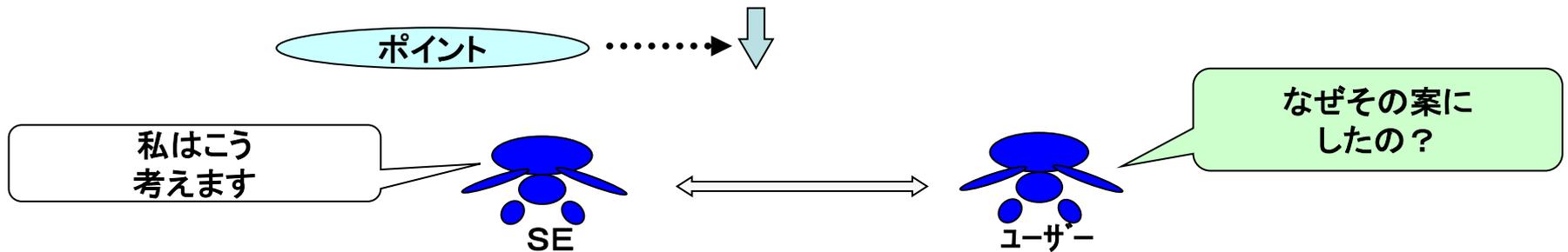
# THEORY21

案の選定では  
「なぜ選んだのか」を用意する



比較表

ものさし		自社サーバー利用		データセンター利用	
なければならない	初期費用2000万円以内	1800万円	OK	500万円	OK
	ランニングコスト100万/月以内	20万円/月	OK	80万円/月	OK
	⋮	⋮	OK	⋮	OK
あるべき	バックアップ体制	自社で行なう 自社ですべてコントロールする 必要あり	△	3世代で管理 センター利用については高いが 外部ネットワーク部分にやや 問題あり	◎ ○
	セキュリティ	⋮	×	⋮	
リスク		サーバーがダウンして基幹系が止まる		データセンター側の課金が増大する	
		⋮		⋮	



提案とは案を提示することであり、多くの場合考えられる案が複数あります。セオリー20のアイデア出しで「OR」が出た場合です。「案の選定」という仕事はソリューションビジネスにおいて骨子となる所といえます。案選定のポイントは、「ベストな案を選ぶ」ということではなく「なぜ選んだか」をユーザーにきちんと説明できるかです。

筆者は今ユーザー側の立場でSEが作った提案書を見るが多くなっています。例えばSEがクライアント・サーバー方式で案を持ってきたとします。この時筆者が、「どうしてこの案にしたのか、サーバーなしのピア・トゥ・ピアでも大丈夫なのでは」と聞くとSEはどのように反応するでしょうか。10人中5人位のSEは「ではピア・トゥ・ピアにしましょう。」といます。SEに「どちらの案が良いか？」を聞いているのではなく(もちろんSEがクライアント・サーバーが良いと思っているのはわかっています)、その案を持ってきた理由を聞いているのです。先ほどのように答えれば、それは提案ではなく単なる他社での実施例であり、厳しい言い方をすれば「思いつき」です。

残りの5人くらいの強気のSEはどうするかといえば「どんなにクライアント・サーバーの案が良いか、どんなにピア・トゥ・ピアに問題があるのか」を訴えます。この時ユーザーが思うのは「ああこのSEは数ある情報の中からクライアント・サーバーの良い所とピア・トゥ・ピアの悪い所だけを選び、他の情報は隠して、私を説得しようとしているのだなあ」と思い、逆に「だまされないように気をつけよう」と思うだけです。そして、他のソリューションベンダーにも聞いてみようということになります。

案の選定は次の4つの情報から成り立っています。

①案自体

②ものさし

案を評価するものさしです。ものさしは大きく2つに分けることができます。それは「なければならない」と「あるべき」です。前者はそれを満たしていない場合はその案を棄却するというものです。後者は「そうであって欲しい」というものです。情報システムの場合はもう1つ「リスク」というものを加えると良いと思います。（「あるべき」の裏返しでもあるのですが。）これについては次のセオリー22で解説します。

### ③評価

それぞれの案をものさし1つ1つを使って、SEが評価した結果です。「なければならない」の評価は「そうである＝OK」と「そうでない＝NG」の2つに分かれ、NGが1つでもあるとその案は採用されません。すべてNGのために案が1つもなくなってしまったらどうするかです。やってみればわかるのですが、ソリューションでは結構こうなることがあります。この場合はこの状態でユーザーに持っていくことです。そして「案はありません」と答えるのではなく(案はあるのですから)、「すべての案が『なければならない』条件にひっかかってしまいました」というべきです。この時ユーザーのとるべき対応は2つあります。1つは「あきらめる」であり、もう1つは「なければならない条件を変える」ということです。この条件を変えることができるのはユーザー(というよりも意思決定者)だけです。例えば予算500万円という条件ですべての案がNGとなった場合「あきらめる」か「予算を増やす」かどちらからであり、その選定はSEではなくユーザーの仕事です。

次に「あるべき」のものさしを使って評価します。評価は○、×、△などでも点数(5点法、10点法)でも何でも大丈夫です。ユーザーのタイプに合わせるべきといえます。セオリー15の親分型なら○×、バリバリ型、まじめ型なら点数、超まじめ型ならものさし自体にも重みをつけ「重み×評価点の合計」で評価すべきといえます。

リスクの評価については次のセオリー22で解説します。

## ④ポイント

③の評価情報を見てSEが案を選定します。正確にいうと「実施案を決める」のではなく、「ユーザーにお勧めする案」を決めます。このとき、必ずユーザーに「なぜ」と聞かれることを想定して、ポイントを明確にします。例えば「セキュリティをポイントとする」といったことです。

ユーザーには①～③を一覧表にして説明し、最後にポイント、つまり「プロとしての意見」をいいます。

ここでユーザーが「それはちがうと思う」といったらどういう対応をとるかです。多くのSEは「しまった、ちがっていたか」あるいは「素人のくせに生意気いな」と思ってしまいます。

「ちがう」といつてくれたらSEは「良かった」と思うことです。つまりユーザー側に何か意見があり、自分の意見と「ちがう」所がはっきりしたと思うべきです。

「ちがう」ということは、上の①～④のどれかがちがっているのです。例えば「①案自体」であれば「案がまだあったのか」、「②ものさし」であれば「こんなものは『なければならない』条件ではない」「『あるべき』条件がまだある」、「③評価」であれば「これで○はおかしい。△ではないか」、「④ポイント」であれば、「私はポイントがちがうと思う」・・・とこれだけ「ちがう」ということがあるのです。そしてその①～④の情報についてどこがちがうのか、どう変えればよいのかをユーザーと話し合っていくのです。これがソリューションであり、ユーザーのパートナーになるということです。

案選定におけるSEの立場は、案選定に関するすべての情報をユーザーに代わって収集し、整理し、自らの意見や見積を付け加えて、ユーザーの意思決定を効率的かつ効果的にしていくというものです。

# THEORY22

新システムのリスクは  
消えない

リスクを挙げる

検針員が誤った  
メータ値を入力する

検針データベースが  
壊れる ...

重みをつける

可能性	影響度
大	中
・雨の日は特に入力が難しい	・仮に間違えても翌月調整でき、トータルでは迷惑かけない

可能性	影響度
小	大
・ハード的にクラッシュする危険はかなり低い	・請求書が発行ができなくなる ・銀行へ引落とし依頼ができなくなる

手を打つ

予防	発生時対策
・入力時に使用量を計算し、前月と20%以上違う場合はワーニングを出す ...	—

予防	発生時対策
—	・毎月MOIにバックアップをとる ・モバイル側のランザクシオンデータは1ヶ月間保管する

セオリー21で述べたリスクへの対応について解説します。まずリスクの定義づけですが、起きていることや絶対に起きないことはリスクとは言いませんし、良いこともリスクとはいいません。リスクとは「起きそうな、起きるかもしれないトラブル」となります。日本語ではよく「脅威」と訳しますが、「不安」と言う言葉の方がピッタリです。この不安について考えることをリスク分析といいます。人間は不安を持つ唯一の動物といわれ、このリスク分析はかなり前に人間社会で確立された理論です。保険ビジネスなどにも適用され、人間があみ出した「ワザ」の中でも、極めてすぐれたものだといえます。

このリスク分析を提案、ソリューションビジネスでは「ユーザーが新システムに対して、良いのはわかったが、(あるいはこの案でいきたいが、)何となく不安だ」というときに使います。新システムにはリスクはつきものであり、ほとんどすべてのケースでこれが必要となります。このリスク分析という考え方、手法が存在していること自体を知らないユーザーが多く、ソリューションビジネスの1つの柱と考えることもできます。

リスク分析のポイントは「リスクをすべて消すことはできない」ということをSE、ユーザーともに理解することです。新システムを仮に5年間使うとして、「5年間に起きうるトラブルを全部解消しないとやらない」とユーザーが言った場合は「それは新システムをやらない」といっているのと同じです。5年間に起こりうるトラブルを前もって全部消すことなんてできません。

実例ですが、ある企業で「紙幣を数える機械」の導入を検討していました。ある機械を導入することをほとんど決め、最後にこのユーザーが製造メーカーに「この機械は数えまちがえませんか？」と質問しました。何と答えたらよいでしょう。「数えまちがうことはない」というのは技術者のいうことではありません。「この機械が数えまちがえないことをどうやって証明するのか」を考えればわかると思います。ダイクストラという学者の有名な言葉に「システムにエラーがあることは証明できても、エラーのないことは証明できない」というものがあります。これがリスク分析の基本の基本です。もし「この機械は数えまちがえるからいやだ」というなら、数えまちがえのない方法があるかを聞くべきです。今人間が手で数えているなら、もちろん数えまちがいはあります。どちらの確率が低いかというだけだと思えますし、数えまちがいの確率だけで案を決定するわけにはいきません。このときはセオリー21に戻り、「手で数えるか」「機械で数えるか」について情報を収集し、案の選定をすべきです。ユーザーにこのことを理解してもらわないで、リスク分析をやっていけば果てしない作業となり、いつまでたってもシステム開発に入れませんし、リスク分析のために膨大なコストがかかることになります。

リスクがすべてには消えないとすれば、セオリー13で述べたセキュリティ同様に「どこまで手を打つか」ということになります。リスク対策にはコストがかかります。無限のコストを使うわけにもいかず、リスク対策コストとそれによって受ける効果の見合う所というのが妥当な線です。ユーザーはリスク分析ではどういうわけかこの費用対効果を考えなくなってしまう。ここまでユーザーが理解してくれれば、もうリスク分析の仕事の半分は終わっています。

リスク分析の具体的なやり方を説明しましょう。

### ①リスクを挙げる

リスクの対象分野(新システムなど)を決め、それに対する起きそうな、起こるかもしれないトラブルをユーザーとともに列挙していきます。このとき、この作業が終わってもリスクはすべて挙げきっていないことをユーザーに理解してもらいます。起こるかもしれないトラブルを全部挙げることなど不可能です。

## ②重みをつける

リスク1つ1つに重みをつけていきます。重みは次の2つについてつけます。

- ・可能性⇒起きる度合
- ・影響度⇒起きたときのダメージ

これはセオリー13で述べたように「可能性をその期間内の発生確率で表わし、影響度を被害額で表わし、これをかけリスクの期待値を計算し、この範囲内でリスク対策を行なう」というのが基本です。しかしこれが使えるのはそのリスクがコンスタントに起きているもの(例えば交通事故のようなもの)だけです。ソリューションビジネスのように新システムを提案するケースではほとんど使えません。発生確率など出しようもないし、被害額などどのように計算してよいかわからない場合がほとんどです(ソリューションイメージの例でいえば、検針員が誤入力をする確率なんて、初めてやる仕事なのでわからないし、誤入力したときの被害額など見積りようもありません)。

ここでは可能性について大(起きそう)、中(起きるかもしれない)、小(めったに起きない)、影響度について大(起きたら大変)、中(イヤな感じ)、小(たいしたことない)の各々3段階くらいでつければ十分といえます。

### ③手を打つ

リスクに対して何らかの手を打つのですが、ここでは②の重みを使って2つの手を打ちます。1つは「予防」です。可能性が「大」のリスクから順に、それが「中」や「小」にならないかを考えます。つまり「なるべく起きないように」に考えるということです。リスクの期待値が計算できないときは、リスクの均一化(すべてのリスクを「中」以下にする。信頼性を強く要求されるときは「小」以下にする)を図ることで、これが最も合理的です。

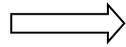
同様に影響度が「大」のリスクから順に、それが「中」、「小」にならないかを考えます。つまり「起きてても大丈夫なようにする」というもので、「発生時対策」といいます。

そして最後にユーザーとSEの間で確認すべきことは「リスクはまだ残っている」ということです。トラブルは起きると思っていれば起きて驚かず冷静に対応できます。起きない、万全と思っていて、起きるとパニックになります。

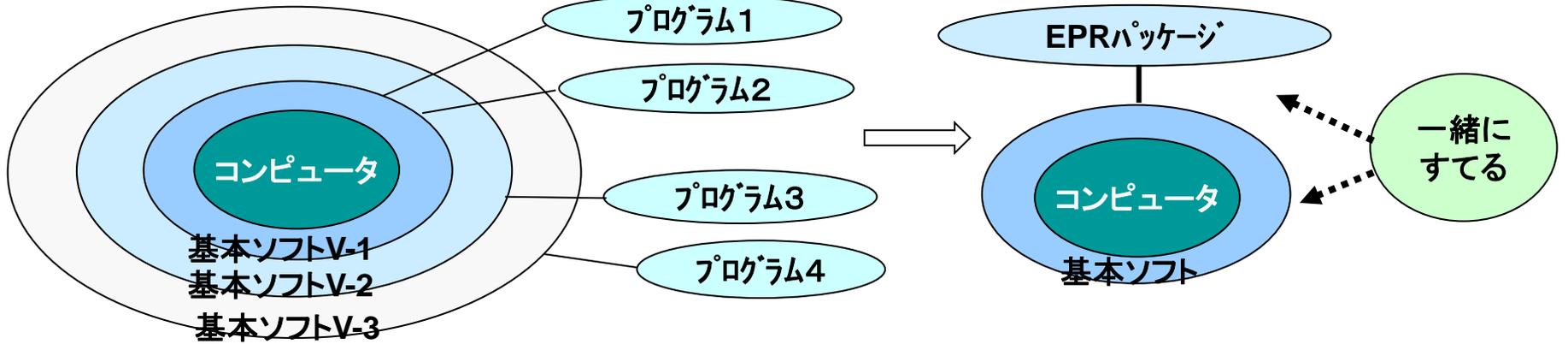
# THEORY23

基幹系・情報系・ネットワーク系に  
分けて提案する

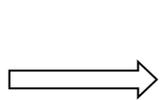
## 基幹系システムの提案



### 賞味期限の見極め

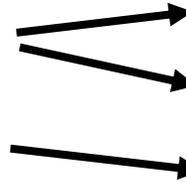


## 情報系システムの提案



### データ活用方法の提案

### コンピュータリテラシーの提案

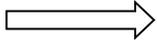


セオリー24

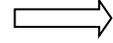
セオリー25

セオリー26

## ネットワークシステム提案

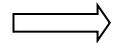


BtoB



自らがeマーケットプレイスを構築し、  
ユーザーに参加を求める

BtoC



ユーザー企業とアライアンスして会社を作る

ソリューションテーマに対し、どのような方向で提案を行うのかはシステムを次の3つのパターンに分けて考えます。

- ・基幹系システム・・・企業の根幹となるシステムであり、なくてはならないもの。このシステムが止まると企業が致命傷となることも多い。(経理システム、販売システム、生産管理システム・・・)
- ・情報系システム・・・企業活動を効率的、効果的に行うためにあった方がよいシステム。これが止まっても企業の致命傷になることは少ない。(商品売れ行き表、セールス成績表・・・)
- ・ネットワーク系システム・・・企業がネットワークを通じて外部と接するシステム(E C、メール・・・)

ニーズ解決を考える時は、それが3つのうちどのシステムに該当するかを考え、基本的には以下のように対応していきます。

### ①基幹系システム

基幹系システムはどんな零細企業でもある程度は進められています。そしてどんな企業でも一気に作るのではなく、プライオリティの高いシステムから順に開発・導入していきます。

一方コンピュータには基本ソフト(オペレーティングシステム、OSともいう、Windows、UNIXなどが代表例・・・)というものがあり、これがシステムのベースとなっています。基本ソフトは時とともに改善が進められ、バージョンアップがなされていきます。この時、新バージョンの基本ソフトは常に旧バージョンの基本ソフトを包含し、旧バージョンの基本ソフトで動いたプログラムはすべて新バージョンの基本ソフトで動くように考えます。

これを繰り返していくとソリューションイメージのようにだんだんシステムは太っていきます。基幹系システムを順に導入し、さらに改良を進めれば進めるほどシステムは複雑になり、わけがわからなくなってきました。家の増築・改築を繰り返していくことを考えればわかると思います。

こういった環境下にある基幹系システムへの提案スタンスは「さらに増築する」ではなく、いつ壊して立て直すか、つまりシステムの賞味期限の見極めがポイントとなります。今のシステムの捨てる時期を決めたら、次期システムもまたいつか捨てるのですから、その時期も始めから考えておくべきです。つまりセオリー4で述べたように耐用年数(これが賞味期限)を設定すべきです。そして耐用年数を迎えたら、「基本ソフト、プログラムをまとめて」捨てることを考えるべきです。

そう考えると基幹系システムは、使い勝手よりもできるだけ安く、信頼性の高い(なるべく止まらない)ものが必要といえます。そしてその答えがERPパッケージ(Enterprise Resource Planning: 企業全体の基幹系システムを1パッケージで提供するもの)です。つまり多くの企業が共同で共通の基幹系システムをコピーして使うというものです。そしてその責任はそれを選定したソリューションベンダーがとるべきです。

## ②情報系システム

情報系システムの提案ポイントはデータ活用方法とコンピュータリテラシーです。これはセオリー24～26で述べます。

## ③ネットワーク系システム

ネットワーク系システムはECのような基幹系システム(止まったら企業の致命傷になる)とメールシステムのような情報系システム(止まっても致命傷にはならない)に分けることができます。後者はやはりセオリー24～26を読んで下さい。前者の基幹系システムはユーザー企業がつなぐネットワークの相手先によってBtoB(企業)とBtoC(消費者)の2つのパターンに分かれます。

### ( i ) BtoB

BtoBでは企業と企業が取引するのですから、そのネットワークを誰が構築するかということがテーマとなります。ユーザー企業同士が共同で作るとするのは次の理由でうまくいきません。

- ・社長が何人もいるので、投資の決定、スペック決定を誰がするのかわからない。仮に出来てもネットワークの稼働責任が不明確となる。
- ・発生コストを誰がどうやって負担するのかを決めるのが難しい(「取引量に応じて」としたいが、誰から誰にそのコストを支払っていいかわからない)

こう考えるとユーザー側の結論は1つです。誰かが作ったネットワークに後から入って、使った分だけ料金を支払うというものです。したがって誰かが作るのを待つしかありません。これがBtoBが進まない理由です。

ソリューションベンダーから見れば答えは1つです。自らがネットワークを作り(これをeマーケットプレイスという)、多くのユーザー企業に参加してもらい、料金を払ってもらって、このマーケットの開設費・運営費を回収するというものです。これがネットワークビジネスの基本です。したがってBtoBでの提案骨子は「自社のネットワークへの参加」です。

## (ii) BtoC

例えばネットワークで花を消費者に売ることを考えてみましょう。これは純粋な花屋とは言い難く、見方を変えればネットワークビジネスともいえます。つまり花屋がやる(花屋がネットワーク構築費を負担する)には、ネットワークノウハウがなく、ソリューションベンダーがやる(ソリューションベンダーがネットワーク構築費を負担する)には花屋のノウハウがなく、どちらにしてもうまく行くとはいえず、どちらもやりません。これがBtoCがうまくいかない理由です。

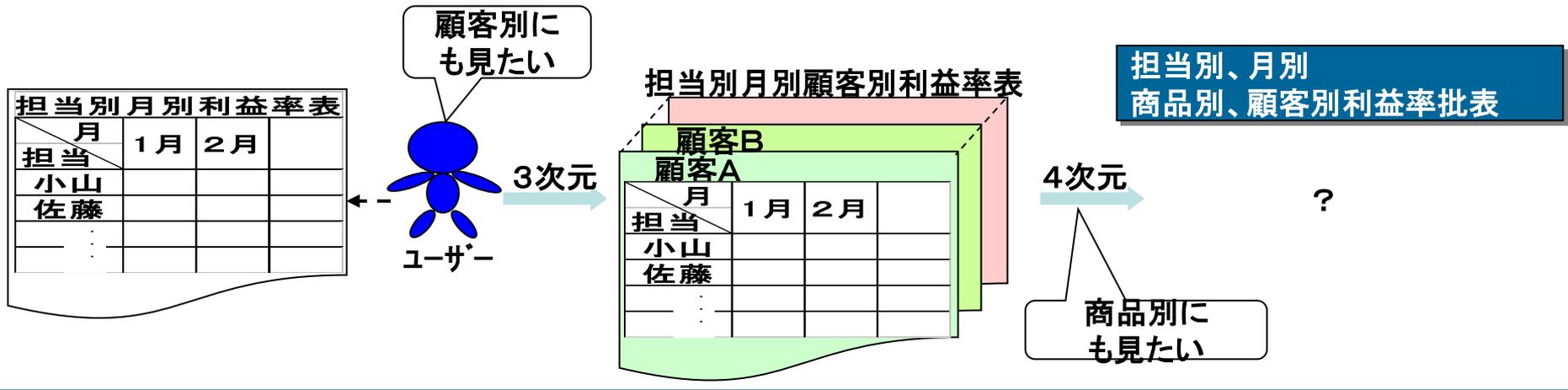
これも答えは1つです。花屋とソリューションベンダーがアライアンスしてやるしかありません。この時2社が共同でやるとすると、(i)のBtoBと同じように、複数の意思決定者とコスト負担という同様の問題が発生します。ここではユーザー企業とソリューションベンダーが共同出資で会社を作り(社長を1人作り)、ネットワーク花屋をやるしかありません。

したがってBtoCでの提案骨子は「アライアンス企業の設立」です。

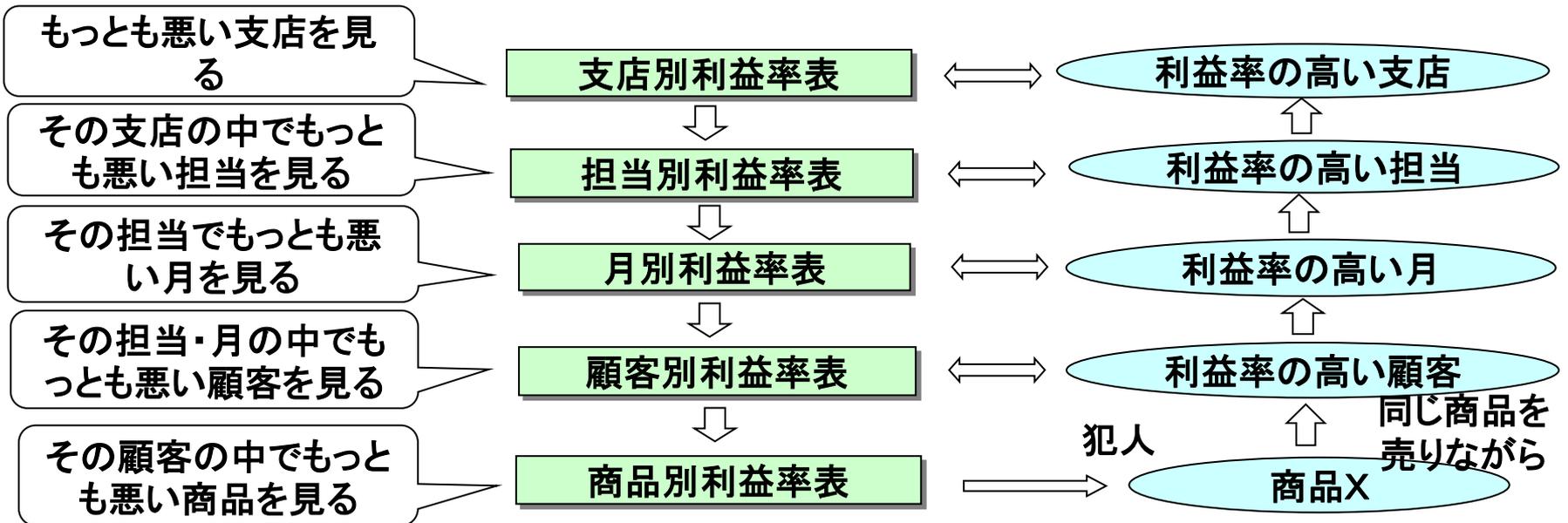
# THEORY24

データ分析は  
マクロからミクロへ犯人探し

多次元ツールは使えない



マクロからミクロへ犯人探し



情報系システムの提案の中心となる、コンピュータを使ったデータ分析のやり方はユーザー、SEとも苦手な分野です。またソリューションビジネスのように分析する人(ユーザー)とその方法を考える人(提案型SE)が分かれている時は特に難しいといえます。その理由は次の2つにまとめられます。

- ・ユーザーはコンピュータがデータを分析してくれると思っている
- ・SEはユーザーがデータ分析手法を提示してくれると思っている

前者はユーザーに、分析するのは人間で、分析するにあたって、必要となるデータを取り出しやすくしてくれるのがシステムだということを理解してもらいます。後者はSEがデータ分析の基本的考え方を持って、それをユーザーに提案することで解決します。本セオリーではこの「考え方」の例を挙げます。

SEが例えば担当別・月別の利益率表をエクセルなどの表計算ソフトを使って2次元の表で提案すると、ユーザーは必ずといっていいほどクロスするキーを追加しがります。「顧客別にも見ることもある」というものです。これならソリューションイメージにあるように、3次元の表ですので何とか頭にそのイメージを浮かべることはできますが、さらに「商品別にも見たい」といって担当別×月別×顧客別×商品別となると、4次元の表となり頭に描くことさえもできません。2次元の紙への出力が中心ですので、果てしないページ数となってしまいます。データマイニングツール(データ分析用のソフトウェア)の代表格ともいえるn次元分析ツール(しかしn次元とは想像もつきませんが)はこの果てしないクロス(××別××別××別...の表)を実現し、ユーザーニーズを満たそうとしたものです。

しかしユーザーは頭に描くこともできないデータ間の関係表を使って分析することなどももちろんできません。SEはユーザーからどうやって分析したいか(分析手法)を聞くのではなく、どんな課題を解決したいのかを聞いて、分析方法を提案すべきです。

多くのユーザーは現状の経営、業務の中から、問題点となる部分の抽出をコンピュータに求めることが多いといえます。そこでのポイントは「マクロからミクロへ犯人探し」です。犯人とは「問題点＝悪い所」という意味です。悪い所は線(1次元)や面(2次元)ではなく、点である必要があります。売れ筋・死に筋リストというのは売れ筋という欠品を起こす元となる悪い点(特定の商品そのもの)を探すことですし、死に筋という在庫ロスを出す悪い点を探すというものです。これをいきなり細かい表で見ないで、マクロからミクロへ少しずつたどっていくわけです。先ほどのようなn次元の膨大な表ではどこに犯人がいるかわからず、線や面でさえもとらえることができません。

ソリューションイメージの例では、まず単純に支店別の利益率を見ます(これなら1次元です)。この中で最も利益率の悪い支店を探します。その最も悪い支店の中で、最も利益率の悪い担当者を探し、さらにその最も悪い担当者の中で、最も利益率の悪い月を探します。次に最も悪い月の中で、最も利益率の悪い顧客を探します。この最も悪い顧客の中で、最も利益率の悪い商品を探します。これが犯人(点)です。

犯人が見つかったら、「その同じ担当が同じ商品売りながら利益率の高い顧客はいないか」、「月はないか」「利益率の高い担当者はいないか」と先ほどの逆のルートをたどっていきます。次にこれと犯人を比較して、なぜ犯人だけが利益率を落としているのかを「人間」が考えます。これが仮説です。

次にこの仮説が正しいとすると、その仮説どおりの結果にデータがなっているかを確認します。これを仮説の検証といいます。場合によっては仮説の検証のために手を打ってみます。例えば利益率を落としているのは「Aという顧客とXというセールスの相性が悪い、どうも価格折衝でうまくやられている」という仮説を立てます。次にそのとおりにデータが出ているか(Xというセールスが他の顧客では利益率が高いか...)、実際に担当セールスを変えることで利益率の変化を見るということです。

これがデータマイニングです。マイニング(mining)とは鉱山で金などを「採掘」するという意味です。犯人という金を採掘するのがマイニングです。

# THEORY25

意思決定支援は  
シミュレーションモデル

# 特売商品の決定

## レポート出力

カテゴリー	商品名	定番販売			特売販売		
		平均単価	平均日販数	平均日粗利	平均単価	平均日販数	平均日粗利
トイレトペーパー・ティッシュペーパー	Aトイレト	350	22	2420	268	364	10,192
	Bトイレト	200	36	2800	148	45	1,170

## 提案

このゾーンは  
特売やる  
必要なし

B  
トイレト

月粗利額

このゾーンは  
特売効果あり

(円の大きさは売上高)

A  
トイレト

C  
ティッシュ

特売比率

E  
ティッシュ

F  
トイレト

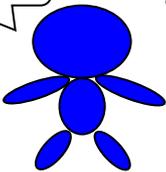
このゾーンは特売をやっ  
ていないかわからない

D  
ティッシュ

このゾーンは特売をやっ  
ても、うまくいかない

## シュミレーション

よし  
やるか!



ユーザー

Aトイレトペーパー  
を特売で298円  
にしたら

意思決定データ

コンピュータ

データベース

評価データ

粗利予測額は  
82,000円

情報系システムのデータ分析の行きつく先は、企業の各層で行なうさまざまな意思決定を情報システムが支援し、その意思決定を効率化、高度化できないかというものです。この意思決定支援システムには3つのパターンがあります。「スーパーで特売商品を選ぶ」というケースで考えて見ましょう。

### ①レポート出力

過去の意思決定内容とその結果をレポートし、それを見て人間が意思決定するというものです。意思決定支援システムというよりも情報提供システムといったほうが良いと思います。多くのSEがこのタイプを提案しますが、意思決定の対象が多くなり、その結果が複雑となってくるとあまり使えません。そのため結局は人間のカンだけに頼るという形になっていきます。

### ②提案

コンピュータが主に意思決定対象の提案などを行うものです。時間とともに変化していくデータについては「そろそろ意思決定してはいかがですか？」という提案を情報システムでできます。例えば「入荷してから3日間売れていない商品」を在庫の多い順に出力して、これを特売の対象にしたらどうかと提案するものです。あるいは意思決定対象をいくつかのグループに分け、対象の選定を助けることもできます。この場合ソリューションイメージにあるように2つの指標でプロットし、4つの象限(グループ)に円(大きさがもう1つの指標)を書くというのが一般的です。例えば右上のゾーンにあるものは「特売効果が大きいのでやってみよう」左下は「やったことがないのでやってみる価値はある」といったことです。

### ③シミュレーション

その意思決定を行ったらどうなるのかを予測するのが意思決定シミュレーションです。これはコンピュータに過去の意思決定内容・結果などのデータベースを持ち、意思決定データを入力し(意思決定したら)、評価データを出力する(どうなるか)というものです。

ソリューションイメージではAトイレットペーパーを特売で298円にしたら(意思決定データ)、日あたりの粗利予測額は82,000円(評価データ)とコンピュータが計算し、この82,000円でOKであれば意思決定をOK(つまり298円にする)とし、評価データが気に入らなければ、意思決定データを変える(318円にする、Dティッシュにする...)というものです。

このシミュレーションを一步進めると、コンピュータがさまざまなケースをすべてシミュレーションして、最良な結果となる意思決定データを出力するという事も可能な時があります。これをオペレーションズリサーチといいます。例えばコンピュータが「Aトイレットペーパーを特売対象にし、298円で売るのが最適です」と出力するものです。もちろん人間はこれに従う必要はなく、これを参考にして自分で判断すれば良いことになります。

オペレーションズリサーチのインタフェースを少し工夫して、人間が「特売をどうすればいい？」と聞いてコンピュータが「明日くらいにAトイレットペーパーを3日間298円で特売したら」と答える感じにすると、エキスパートシステムと呼ばれる人工知能のイメージになります。

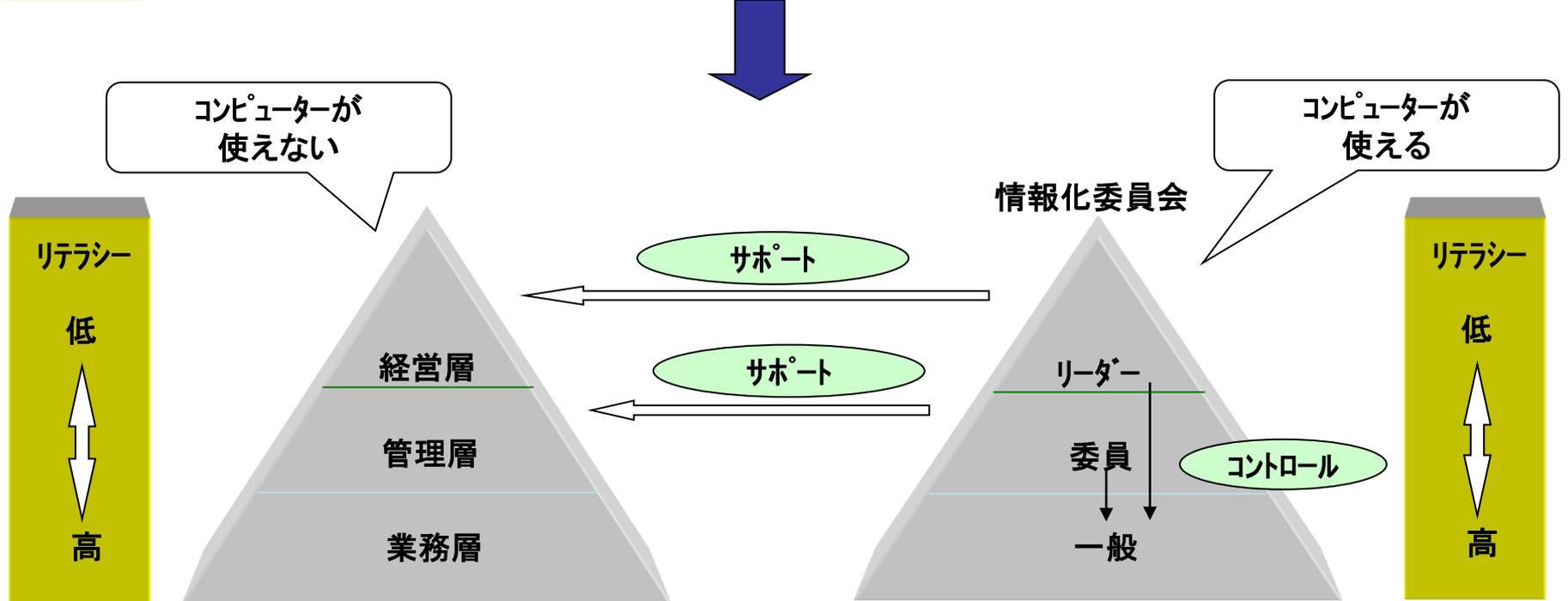
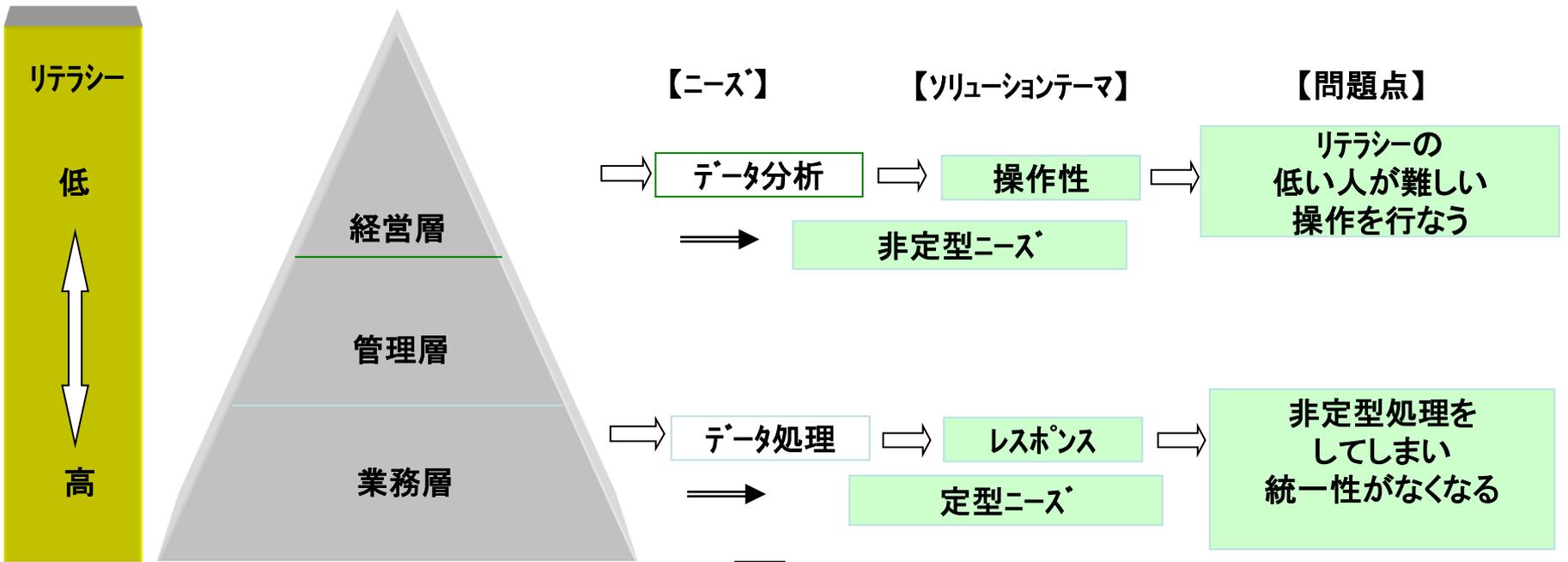
オペレーションズリサーチもエキスパートシステムもシミュレーションが可能かどうかにかかってきます。シミュレーションシステムには次の2つの問題点があり、これをユーザに理解してもらうことが大切です。

- ・シミュレーションはさまざまなパターンの意思決定を試して、その意思決定と実績データがたまらないと、良い評価データを出力することはできない。
- ・シミュレーションは評価データの予測であり、実績データがたまるだけでなく、何度も予測→実績→差異分析をくり返すことで予測精度が高まる。

つまりさまざまな意思決定を試し、結果を残し、そのうえで何度もシミュレーションして、その結果と予測の差異を分析することで実用化できるということです。「あたらないからやらない」ではなく、「あたるためにやり続ける」ということをユーザに理解してもらいます。

# THEORY26

非定型ニーズと  
リテラシーは逆行する



企業は一般に経営層(社長、経営陣)、管理層(部長、課長、マネジャー、リーダー)および業務層(一般職、プレイヤー)の3つに分けることができます。そして上位層に行くほど、さまざまなデータをさまざまな形で(非定型に)分析します。この上位層の情報系システムでソリューションすべきテーマは、そのシステムの操作性、つまり使い勝手です。分析の非定型さを増せば(色々なデータ分析をできるようにすれば)「今日はどういう形で見えるか」をコンピュータに伝えなくてはならず、操作の難しいシステムとなります。提案型SEとして大切なことは「非定型さを増しながら操作性を良くする、つまり使いやすくすること」などできないということをユーザーに理解してもらうことです。まんがやグラフなどを使って、優しく(「易しく」ではなく)することはできますが、それはコンピュータへの抵抗感を少なくしているだけです。「非定型さ」と「操作性」は常にトレードオフであることをよく理解してもらう(SE自身も理解する)ことです。

一方、企業の上位層に行くほど年齢階層が上がっていくのが普通です。一般に年齢が上がっていくほど、コンピュータリテラシー(コンピュータ操作能力)が下がっていくのも事実です。つまり操作能力が低い人ほど難しいシステムを操作して、非定型で複雑なデータ分析をしなくてはなりません。提案型SEはこのテーマから逃げてはいけません。「操作教育をする」という手で簡単にすまそうとしますが、自分がその年齢に達すればわかりますが、教育してもなかなかリテラシーは上がらず、むしろ低下を何とか止められる程度です。

また、企業の下位層に行くほど、データをタイムリーに更新して、いつも同じ見方でデータを処理していくようになります。いつも同じやり方なのでデータを更新さえすれば、ボタン1つで結果は出るようになります。このシステムのソリューションテーマはいかに早く結果が出せるかという「レスポンス」になります。ところがこの層は一般にコンピュータリテラシーが高いので、非定型な分析ツールを使いたがり、会社としての統一性を失います。エクセルなどの表計算ソフトなどは典型的な非定型ツールですが、これを使うことによって、私のエクセル、君のエクセル・・・が出来て、エクセル同士が不整合を起こし、ドッキングできず、合計が出ず、数字の意味がわからず、そもそも必要な表がどこにあるかもわからない・・・という問題を引き起こします。

この2つのコンピュータリテラシーの逆転、不統一を解決するのは、本来であれば情報システム部なのですが、以下のような問題を抱えています。

- ・情報システム部はITベンダーとの調整で手一杯で、エンドユーザーをサポートする時間がない(それを仕事とっていない)
- ・サポート業務は仕事の波が大きい
- ・中小企業ではそもそも情報システム部というスタッフは持てない

これを解消するのが情報システム委員会です。現場の業務を持ちながら(このように複数のタスクを持つ組織をマトリクス組織という)、コンピュータに興味があり、コンピュータリテラシーの高い人を各セクションから選び、さらにその中から全社リーダーを選びます。情報システム委員会は通常の仕事と同様に階層性をもちます。情報システム委員会のリーダーは経営層を、委員は管理層をサポートし、彼らのコンピュータリテラシーを補います。一方、業務層はこの委員会のサポートを受けるのではなく、コントロールを受け、システムの使い方に制約を受けます。「表はこうやって作る」「ファイル名はこうつける」・・・とルールが決められ、その指導に従ってデータ処理を行います。ソリューションベンダーが全部これをサポートするには、ユーザー企業に常駐する必要がありますが、そんなことは一般の企業ではコスト的に不可能です。

提案型SEはこうしたコンピュータリテラシー、組織についても提案を行うことで、ユーザーのパートナーとして機能していくとともに、情報化リーダー・委員のサポーターとして機能しています。またこの情報化委員会と定期的に打ち合わせすることで、新しいソリューションテーマが発見され、新しい提案・システムアイデアを得ることが可能となります。

## 第5章 プレゼンテーションモデル

### トータルセオリー

プレゼンテーションはわかりやすさがすべてに優先する。押しつけず、情報を正確に提供することを心がける。

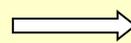
# THEORY27

**提案書はSEが作り、説明する**

# 提案書の作成目的

(意味)

ソリューション方法の提示



スペック確定

買って欲しい

+

販売条件書

## 買って欲しい

どんなに  
この提案がすばらしいか

他のシステムがどんなに  
問題あるか



提案書



誰が作るんだろう？

本当かなあ  
他の会社にも聞いてみよ  
う

私はこう考えます

この点は問題がるが了承  
して下さい



提案書



この人が作るのか

信頼できる

## 販売条件

こんなこともできますか



提案書



誰が作るんだろう？

それはできません。  
こういう方法ではどうで  
すか？

ここはどうなっているの  
ですか



提案書



細かいことはご契約後S  
Eがやります

それはこのようになります

本セオリーではプレゼンテーションモデルのポイントを述べます。

### ①提案書の作成目的

ソリューションビジネスにおける提案書の目的は「ユーザーにそのシステムを買ってもらおう」ということであり、いってみればユーザーごとに個別に作る製品パンフレットのようなものです。「買ってもらおう」ことが目的だとすれば販売金額の提示(いわゆる見積書)が必要です。

SEから「提案の段階で見積書をつけるのですか」という質問をよく受けますが、答えは「YES」です。システム開発ビジネスを長い間やっている、自分たちが「システムを売っている」という感覚がなくなり、「その企業のためにシステムを作っているのだから、ユーザーはかかった分を払ってもらわなければならない」という意識になりがちです。この論理が成り立たないことは自らが家を買う時などをイメージすればわかるでしょう。「この家を買って下さい」「いくらですか」「まだ決まっていません」こんなビジネスが成り立つはずもありません。

「工数を見積ることができなくても、金額を出さなくてはいけないのですか？基本設計終了後別途見積ではダメですか？」という質問も受けます。見積ることができなかつたら、金額提示ができないのですから提案書を出せません。金額提示は提案書の一部です。もしそのビジネスを受注したいのであれば、見積ることができる状態まで持って行って、提案書を出すしかありません。

その見積のための工数がかかりすぎるといふなら、やるべきことは2つしかありません。1つは「提案しない」です。それがいやなら工数をかけずに見積る、つまり見積作業の生産性を上げるしかありません。これについてはセオリー35で述べます。「基本設計終了後別途見積」も自分が家を建てることをイメージすればわかると思います。家を新築しようか、そのまま住んでいようか悩んでいる時、建築士から「こんな家はどうですか」という提案を受けたとします。価格を聞いたら「基本設計終了後別途見積です。ただし基本設計料として100万円頂きます」と言われたらどうするでしょうか。「家を新築する」という意思決定ができるでしょうか。しかもシステムは家とは異なり、作り方が標準化されていないので、基本設計をした企業にそれ以降も頼むしかありません。後でとんでもない金額を言われそうで、ユーザーがなかなか「GO」を出せない気持ちはわかると思います。

こう考えていけば提案書には2つのことを書く必要があることがわかると思います。1つは「ユーザーに買って欲しい」という目的のための「ユーザーニーズのソリューション方法」です。もう1つは見積書の但し書きとしての販売条件書であり、見積のためのスペック確定というものです。

## ②提案書は誰が作るか

ITベンダーのうちコンピュータメーカー、システムディーラーのようにハードウェアを販売してきた企業はセールス部隊が大勢いて、彼らがハードウェア販売のための(ソフトウェアがあっても「おまけ」、オプションのような感じといえます)提案書を作っていました。従来セールスが存在していなかった(いても機能していなかった)ソフトハウスなどはソリューションビジネスのためにこれをまねしてセールス部隊を作り、彼らに提案書を作らせようとしています。しかしソリューションビジネスの提案書をセールスが作ったらどうなるでしょうか。ソリューションビジネスはハードウェアのように目に見えるものではなく、これから作っていくシステムを売るわけです。セールスは「売ること」がすべてで、その後の「作業」「動かす」ことにはあまり興味がありません。

セールスはソリューションニーズを解決することよりも、他社に負けない「世界一機能が高く」「世界一スピーディで」「世界一オシャレで」「世界一安い」システムを提案したがります。そして社内には「今回は赤字覚悟で無理してでも受注して、後でゆっくり回収するようにしましょう」といいます。この論理がまかり通ると、世間を騒がせる「1円入札」となってしまいます。こうした行為を行うITベンダーの経営者は「1円で設計を受注し、他の会社が開発できないように設計し、1億円で開発を受注し、さらにランニングコストを上乗せして受注し、回収する」ということを考えているのでしょうか。しかしこれがまともなビジネスといえるのでしょうか、そしてこんなビジネスが本当に続くと思っているのでしょうか？

だからといってセールスを教育して、「受注後」に興味を持たせるというわけにはいきません。これはモラルの問題はなく仕事の分担(売る、作る)という企業の仕組みによるものです。提案書はその提案したシステムを受注後に設計、開発、運用していく責任者が作るべきです。つまり受注後にできるプロジェクトのリーダーであるSEが作るべきです。冷静に「できること」「できないこと」を切り分けられるSEが作ることが、ベンダー・ユーザーの双方に幸せをもたらすことは少し考えればわかることです。

### ③提案書は誰が説明するか

提案書の目的は2つですので、この判断基準もその2つの理由で検討すべきです。

i 買って欲しい⇒誰が説明すれば、もっとも買ってもらう確率が高いか。

ii 販売条件書(ユーザーに誤解されるのを避けるためのもの)⇒誰が説明すれば誤解が少ないか。

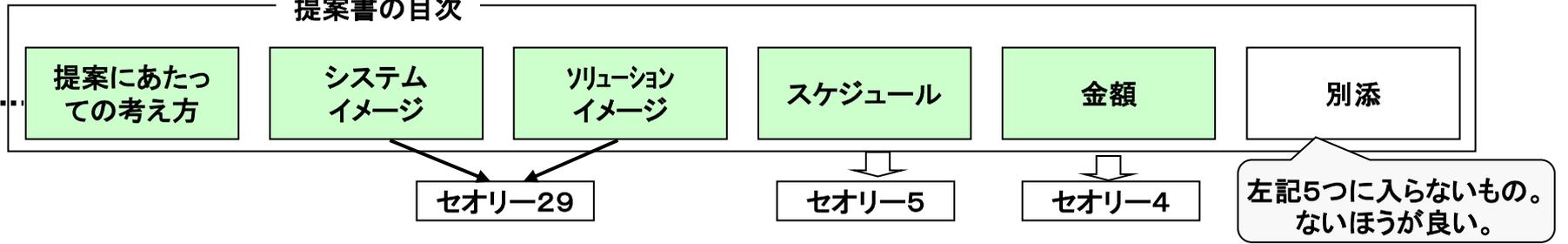
ちょっと考えると(i)の適任はセールス、(ii)の適任はSEとなるでしょう。しかし(i)は本当にセールスでしょうか。これは買う立場で考えてもらえばわかると思います。あなたが自分の家の提案書を説明してもらう時「人当たりは良いが、買うと決まったらこの先もう来ないだろうセールス」と「まじめで人当たりはあまり良くないが、この先一生つき合うかもしれない設計者」のどちらが説明してくれたら「買う」という勇気が出せるかです。

「提案書は、SEがユーザーをインタビューし、SEが考え、SEが作って、SEが説明する。そしてそのSEが責任を持ってシステムを作り、稼働責任を負う」ことがセオリーです。これが提案型SEという仕事です。

# THEORY28

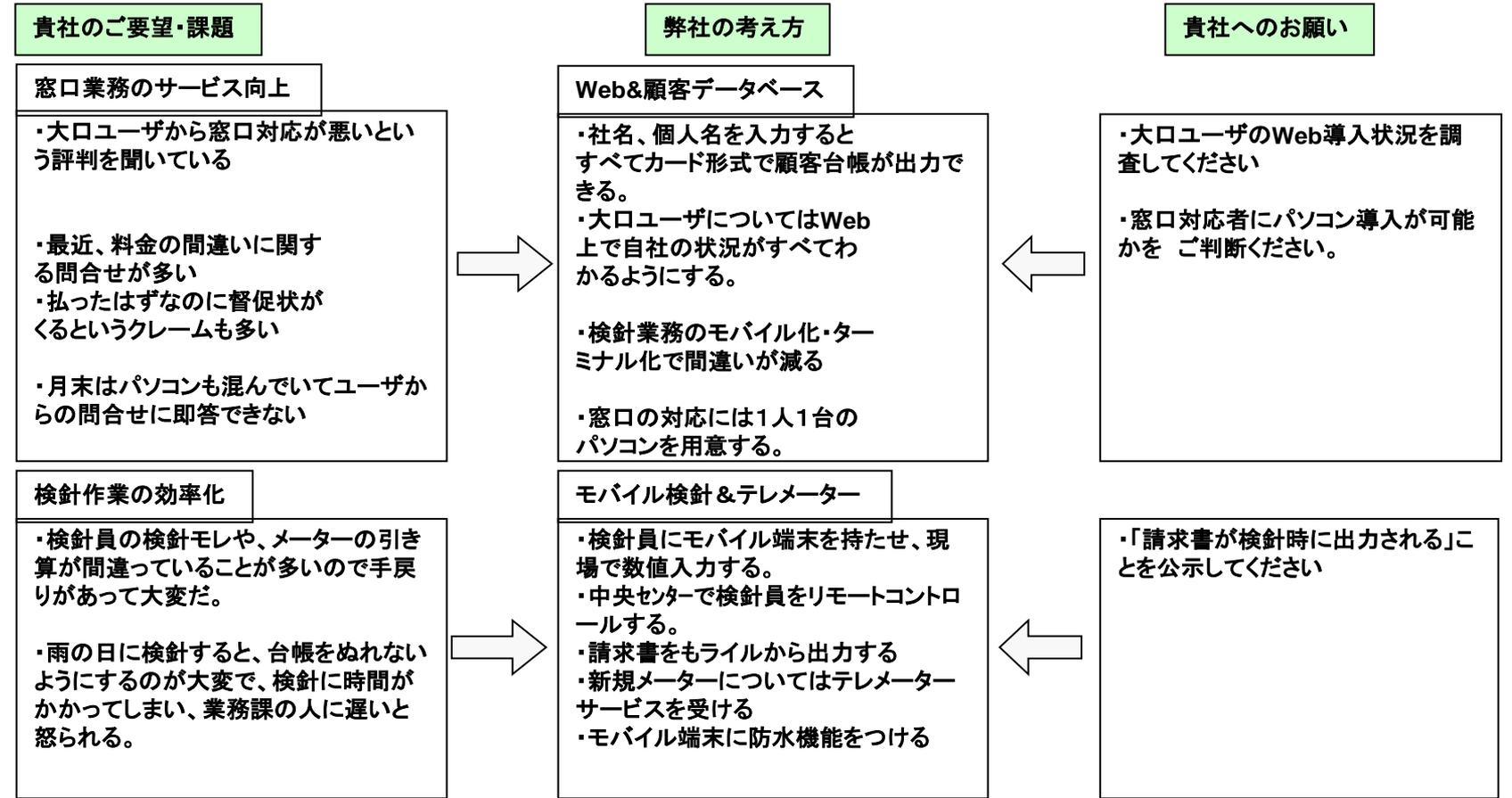
ユーザーニーズはまとめずに、  
そのまま書く

提案書の目次



〔提案書の例〕

1. 提案にあたっての考え方



提案書はソリューションベンダー内で目次を統一しておく必要があります。目次を決めておかないとどんな目次にするかで話し合いがなされ、膨大な時間がかかるだけでなく、目次が決まるとできたような気になってしまいます。提案書は目次を競うのではなく、中身を競ってください。

提案書の目次はソリューションイメージにある5つの項目と別添に固定すれば、どんな提案書でも作ることができます。別添は「おまけ」であり、ないほうが良いものです。「まえがき」などを書くことが多いのですが、「何のために書いているか」を考えると不要だということがわかれると思います。その目的は「提案の機会を与えてくれたことへのお礼」などでしょうが、どうもシステム開発ビジネス・外注作業時代の発想が抜けきれていないと思います。ITのプロとして、将来のパートナーとして提案すると考えればいらないでしょう。医者や弁護士などプロになればなるほど、いちいち「今日は来てくれてありがとう」とはいわないと思います。

本セオリーでは第1項の「提案にあたっての考え方」について述べます。「システムイメージ」、「ソリューションイメージ」については[セオリー29](#)で、「スケジュール」、「金額」についてはそれぞれ[セオリー5](#)、[セオリー4](#)を参照してください。

提案書でまず書くべきことは提案の基本的コンセプト、つまりソリューションしていく方向です。これが「提案にあたっての考え方」です。ニーズ発見モデルで作ったインタビューワークシート([セオリー16参照](#))をベースにして書いていきます。基本的にはインタビューワークシートをそのまま書きます。ワークシートの「ニーズ」欄が「貴社のご要望、課題」であり、「ソリューションスペック」が「弊社の考え方」、「その他の対策」が「貴社へのお願い」です。もちろんこの名前は変えても構いません。貴社のご要望・課題の見出しは、インタビューワークシートのタイトルを使います。「見出し」となるようワークシートでもその業務(クラス)でのニーズのキーワードを入れるようにします。こうすればユーザーが見る「気力」が湧きます。「弊社の考え方」にも見出しをつけ、見易くします。

この項を書くときのポイントは以下の通りです。

①10点になったニーズを先頭にする。

もっとも大事なニーズ(10点のニーズ)は先頭に書きます。(したがってセオリー19で述べたように10点は1つです)ニーズ間の関連や仕事の流れ(検針業務⇒検針管理⇒請求業務⇒窓口対応・・・といったもの)などはすべて無視します。この部分はユーザーにニーズをフィードバックして確認してもらうためのものであり、ユーザーのニーズを分析したり(そんなことはソリューションするうえで不要ですし、ユーザーも期待していません。ニーズの中身はユーザーの方がよくわかっています)、ニーズを理解してもらう(ユーザーの方が作っているSEより理解しています)ものではありません。

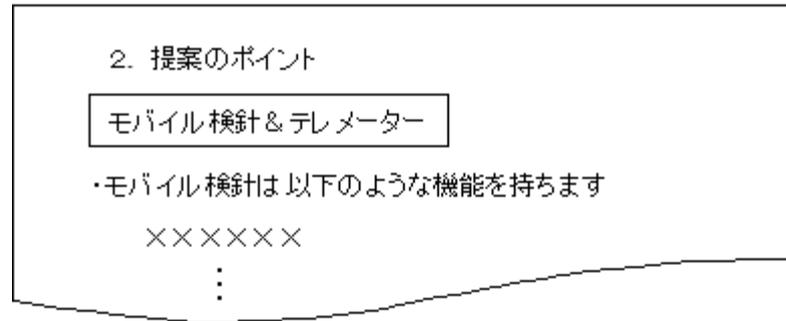
ではなぜ10点のニーズが先頭かというと、10点のニーズが下の方にあると、先頭のニーズを説明しているとき、ユーザーはその話を聞かず10点のニーズを見ているからです。「ユーザーの見る順番に提案書を書く」ことがセオリーです。そしてこの10点のニーズがはずれていなければ、ユーザーはこの提案書を「読もう」という気持ちになります。

「9点はいくつあっても良い」と述べましたが、これは10点の次に9点を書くことができないからです。ニーズは業務単位(クラス)に大切なもの(点数の高いものを含んでいるクラス)から書いていきます。

## ②ニーズはまとめない

人間はどうしてもインタビューワークシートのニーズをそのまま提案書に書かずに「何とかまとめよう」とする習性を持っています(何も言わないでやらせると必ずまとめます)。こんなことをしても何も良いことはありません。少しニーズが歪むだけです。ニーズをダラダラ書いてあるとユーザーが読んでくれないと思うなら、それは誤解です。ニーズはどんなにたくさんあってダラダラしていても心配いりません。なぜならユーザーが「言った言葉を言ったとおり」に書いてあるからです。これならユーザーの頭にスーと入っていきます。むしろダラダラ書いていると伝わらないのはソリューションスペック(弊社の考え方)の方です。逆に、書いたSEはソリューションスペックの方が頭に入りやすく、ニーズの方が頭に入らないので恐がってしまいます。ソリューションスペックがユーザーに伝わりづらいので、これを絵にしてソリューションイメージを書くわけです。

どうしてもソリューションスペックがダラダラした感じが出たら、短文にまとめ、タイトルをつけ、ページを変えて「提案のポイント」という項目を作って詳しく書きます。こうしてユーザーニーズからスペックが離れないように気をつけます。例えばソリューションイメージでの「モバイル検針 & テレメーター」について、もっと詳しく書きたいときは、「弊社の考え方」はソリューションイメージ程度にとどめておいて、ページを改め、「2.提案のポイント」を作って下記のようにします。



### ③ソリューションスペックはダブって書く

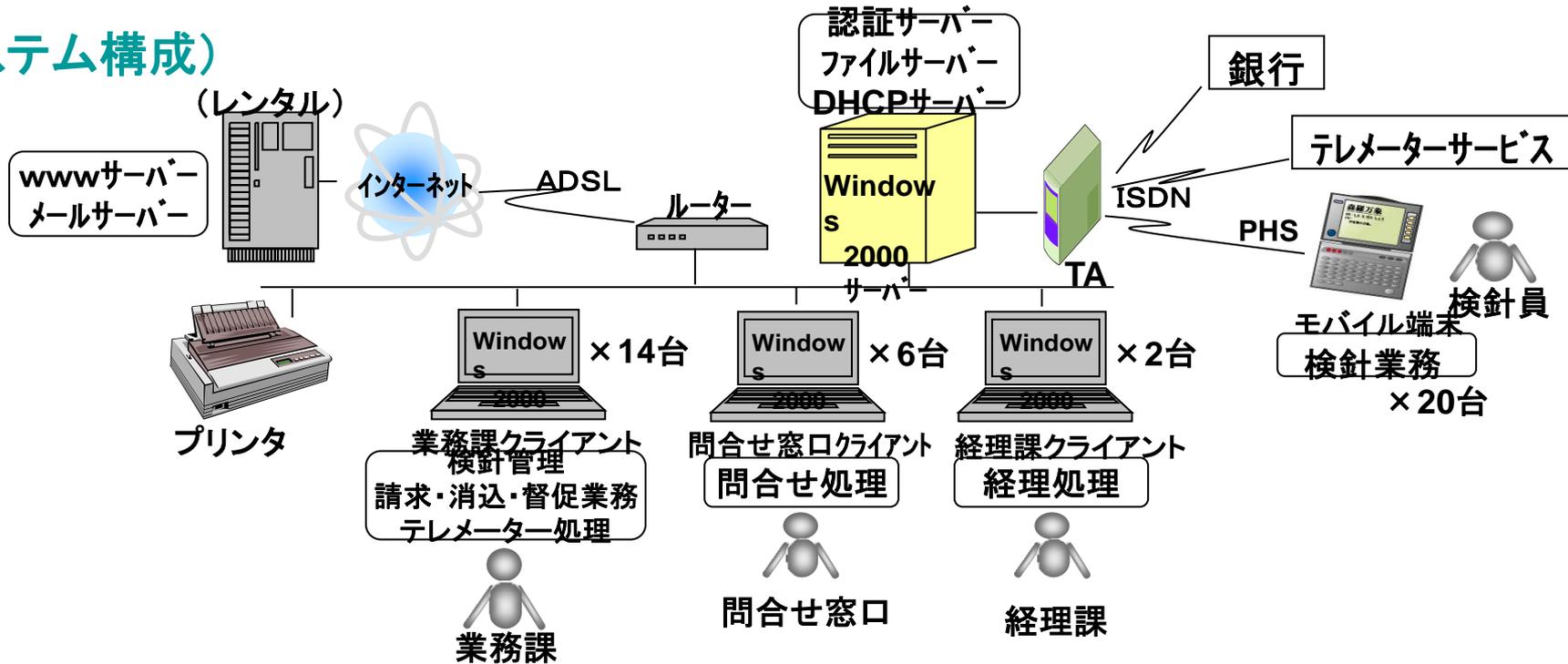
インタビューワークシートに「①で対応」と書かれたことはこのままではなく、その行を読んだだけでもわかるようにします。例えば「検針のモバイル化で対応」といった感じです。戻らなくてもつまり業務別に見てもわかるようにしておきます。ユーザー側は場合によっては提案書を組織別つまり業務別に自分の関係する部分だけについて見ることもあります。

# THEORY29

システムイメージと  
ソリューションイメージを分ける

# システムイメージ

## (システム構成)



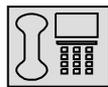
## (ソフトウェア構成)

処理業務	ソフトウェア	開発方法
検針業務	サーバーアクセス	パッケージ
	検針値入力・チェック	パッケージカスタマイズ
	検針ルート管理	オーダーメイド
検針管理	検針チェック	パッケージカスタマイズ
	日報・月報処理	パッケージ
	⋮	⋮

# ソリューションイメージ

## 問合せ処理

××社ですけど、今月の使用量をメーター別に知りたいのですが



会社名入力

メーターNo

使用量

××社

- ・使用量
- ・使用料
- ・属性情報
- ・対応履歴
- ・

××社 ×月別利用量 月指定

メーターナンバー	7月	6月	5月	4月
366263	426 (443)	384 (426)	225 (364)	326 (285)
366264				

( )内は前年同月

戻る 料金へ 属性へ 対応履歴へ

基本的なシステムの方角を書いたら、次はシステム自身をどのように提案書に表記していくかです。提案書には2つの目的があるのですから、2つのシステム表記が必要です。

- ・「買って欲しい」⇒ソリューション方法の提示⇒「ソリューションイメージ」で書く
- ・販売条件書⇒スペック確定⇒「システムイメージ」で書く

この2つを提案書に乗せる順序ですが、システムイメージ⇒ソリューションイメージの順が良いと思います。ユーザー側が見たいのは無論ソリューションイメージであり、SE側が確定したいのがシステムイメージです。ソリューションイメージが先にあると、どうしてもユーザー側はシステムイメージをあまり見なくなり、確認もれの可能性が高いといえます。システムイメージ⇒ソリューションイメージの順にして、ユーザーにソリューションベンダーが求めているのは「システムイメージのシステムスペックを確認する」ことだということを理解してもらいます。このシステムスペックの良否を判断するために「このシステムでどのようにソリューションできるか」というソリューションイメージを提示していることを理解してもらいます。

## ①システムイメージの書き方

システムイメージはこれから売ろうとするシステムの部品・性能諸元を書いたものであり、自動車やパソコンのパンフレットの裏についている細かい性能表と目的は同じです。つまり、ユーザーにあなたが買うものは「これ」であり、購買後「違う」といっても、もう遅いということを確認してもらうものです。

よく「システムイメージはどこまで細かく書くのですか」と聞かれますが、これは「どこまで書けばSEとして恐くないか」というのが答えです。提案書には金額が入っています。その見積条件書なのですから、SEにとっては提案書の中でもっとも大切な部分です。さらにいえばこのシステムイメージを作ることができる人だけが提案書を書くべきであり、SEが提案書を書く理由です。一方、多くの場合提案書は無料で作ることも多いといえます。この場合提案書を書くにはコストが発生するにもかかわらず、もし他社に受注されたり、ユーザーが「これではソリューションできない。やっぱりやめる」といえば、提案書作成のために発生したコストは返ってきません。「受注後のリスク」(ユーザーはこう思っていた、SEはそう思っていない、ということ提案書ではっきりさせていないために起こるトラブル)と「提案書作成コスト」を天びんにかけ、ぎりぎりの所で止めておくというのが基本です。そしてそのリスクと作成コストの両者を見積もることのできる人、つまりリーダーSEが提案書を作るべきです。これが提案型SEの本当の意味です。

書く内容は例示したようなものですが、フォーマット、コンテンツとも各々のソリューションベンダー内で統一しておけば良いだけで、こうしなくてはいけないというのはありません。市販ソフトやパッケージソフトの組立中心の開発スタイルであればソリューションイメージの例のように、オブジェクト指向型の開発環境であれば、デ

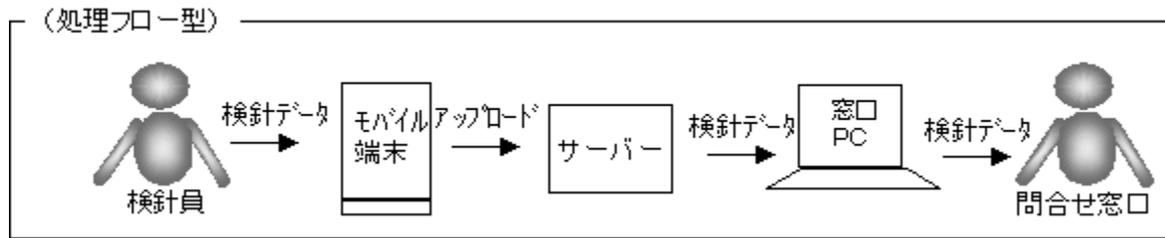
## ②ソリューションイメージ

ソリューションイメージでは、インタビューで聞いたニーズが提案するシステムを使うことで、「具体的に」どのように解決するのかをイメージで表します。

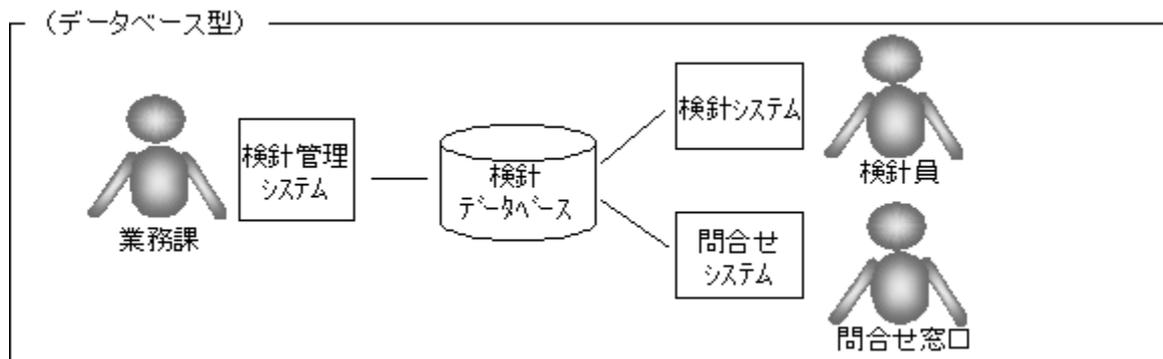
まずソリューションしているステージ(対象ユーザーは誰か、今どこにいるのか。ガス会社の例でいえば問合せ窓口担当が顧客から問合せを電話で受けているステージ)をはっきりさせます。そのステージにおいて「ユーザーはどのようにシステムを使っているのか」つまり「何を入れているのか、何を見ているのか」を書きます。

このソリューションイメージをユーザーに見てもらい、そのステージで(電話がかかってきたとき)本当にそのデータを入れることができるのか(会社名を入れることができるのか、そんな時間はあるのか)をまずチェックしてもらいます。そして、そのうえで「見ることのできるデータ」で自分のニーズがソリューションできるか(大口ユーザーの問合せに回答できるか)を考え、確認してもらいます。

SEは従来データフローや処理フロー、さらには操作マニュアルなどで似たような絵を描いているので、その経験を頼りに次のようなものを書こうとします。



こういう絵を書く人はこの「データの流れ」を知り、誰が何に使うのか、何のために書いているかをあまり考えていません。どうしても心配で、こういう流れをユーザーに確認したいのであれば、販売条件ですのでシステムイメージに入れておきます。



上のようなパターンも多いといえます。これも同様にユーザーに何を確認しているのかがわかりません。すべてのドキュメントは「何のために作っているか」という目的が大切です。ソリューションイメージは「このシステムでソリューションできるか」をユーザーに確認してもらうことが目的です。

# THEORY30

ドキュメントを提案書を中心に  
データベースする

# 提案にあたっての考え方

## 窓口業務のサービス向上

- ・大ユーザーから窓口対応が悪いという評判を聞いている
- ・最近、料金の間違いに関する問合せが多い・払ったはずなのに督促状がくるというクレームも多い
- ・月末はパソコンも混んでいてユーザーからの問合せに即答できない



## Web&顧客データベース

- ・社名、個人名を入力するとすべてカード形式で顧客台帳が出力できる。
- ・大口ユーザーについてはWeb上で自社の状況がすべてわかるようにする。
- ・検針業務のモバイル化・ターミナル化で間違いが減る
- ・窓口の対応には1人1台のパソコンを用意する

## 検針作業の効率化

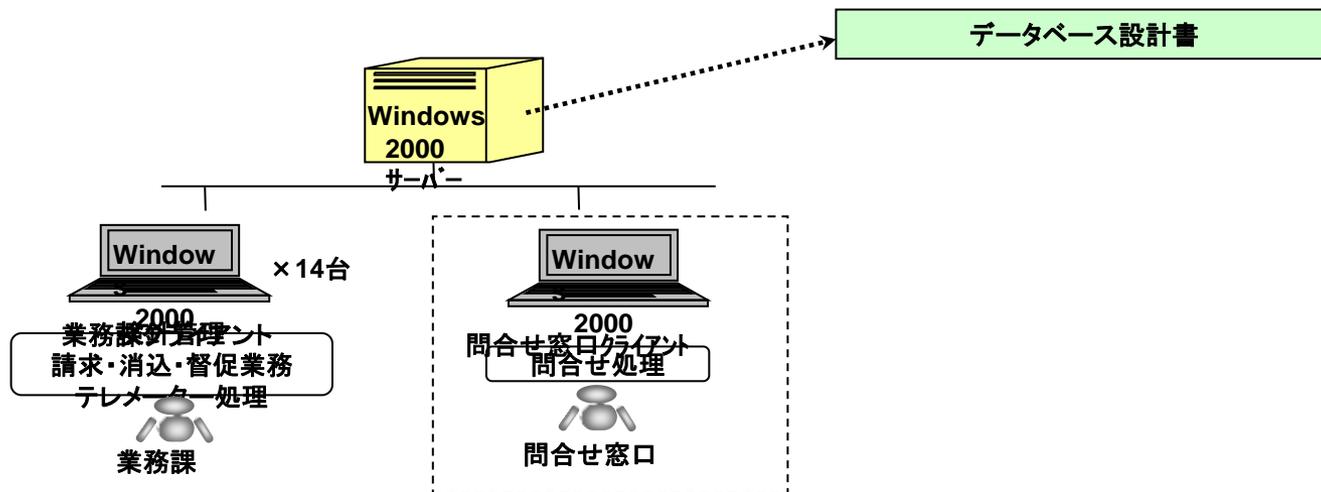
- ・検針員の検針モレや、メーターの引き算が間違っていることが多いので手戻りがあって大変だ
- ・雨の日に検針すると、台帳を濡れないようにするのが大変で、検針に時間がかかってしまい、業務課の人に遅いと怒られる。



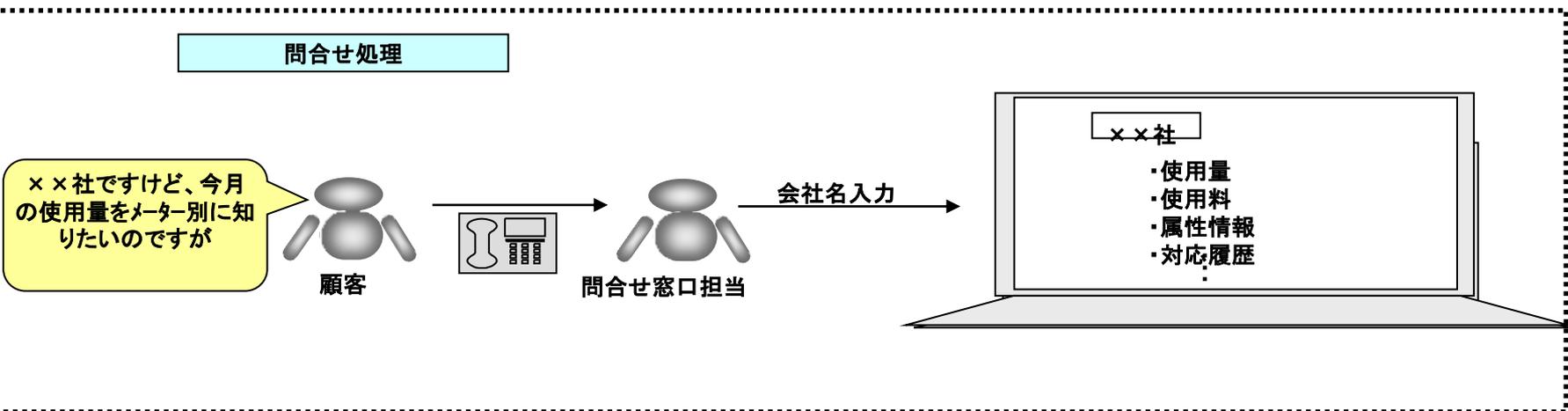
## モバイル検針&テレメーター

- ・検針員にモバイル端末を持たせ、現場で数値入力する。
- ・中央センターで検針員をリモートコントロールする。
- ・請求書をもライルから出力する
- ・新規メーターについてはテレメーターサービスを受ける
- ・モバイル端末に防水

## システムイメージ



## ソリューションイメージ



SEが作る提案書以外のドキュメント(システム設計書、内部設計書、テスト設計書、運用マニュアル...)は体系的に整理され、その責任は一般に設計・開発プロジェクトのリーダーが負っています。またそれぞれのドキュメントの全体の中での位置づけ、各ドキュメント間の関連が設計・開発手法の中で明確に定義されています。

一方提案書はプロジェクト立ち上げ前に作るため、誰が作るのか、その責任は誰が負うのか、他のドキュメントとの関係...といったことがすべてあやふやのうちに作られていきます。しかしよく考えれば最上流工程のドキュメント(一番最初に作り、かつこれから作るすべてのドキュメントの出発点)ですので、もっとも大切なものといえます。運用段階で発生しているトラブルの原因もすべてとっていい位、この最上流工程である提案段階にあるといえます。また仮にそのトラブルがシステム設計段階で未然に防止できても、ユーザー側は「最初に言っていたことと話が違う、このベンダーはもうけようと思って、システムの仕様を変えたに違いない」と誤解されてしまいます。

提案書は作る前に責任者(以降のプロジェクトのリーダーとなる提案型SE)、位置づけ(第一段階のドキュメント)、体系づけ(他のすべてのドキュメントの概要版)をはっきりさせる必要があります。

次に提案書の構造をはっきりさせます。提案書の「提案にあたっての考え方」は、業務・組織単位つまり外部エンティティおよび共通部分にグルーピングされています。システムイメージは無論、外部エンティティ(クライアント)及び共通部分(サーバー)に識別されます。ソリューションイメージは仕事のステージごとですので、ユーザー単位つまり外部エンティティ単位に書いてあります。こう書いておけば、この3つのページは外部エンティティ(ユーザー、業務)単位に串刺しで見ることができます。

ではスケジュールはどうでしょうか。システム開発のスケジュールの多くは次のように書かれます。

見積 項目	担当		7月	8月	9月
	貴社	弊社			
仕様凍結	○	○	■		
インタフェース設計		○		■	
インタフェース確認	○			■	
データベース設計		○			■

これは各項目(仕様凍結、インタフェース設計…)がシステム開発のプロセスに沿って書いてあります。SEにとっては自分の仕事中心であり、見やすいでしょうが、ユーザーにとってどういう意味があるのでしょうか。こう考えればスケジュールもユーザー中心にユーザーグループごとに、つまり外部エンティティ単位に作っていくべきでしょう。またこうすれば開発側も開発プロセス中心から、外部エンティティ、それに対応するオブジェクト中心へと考え方を考えることができます。

見積 業務	担当		7月	8月	9月	
	貴社	弊社				
<b>問合せ窓口</b>						
インタフェース確認	○	○	■			
プロトタイプ作成		○	■	■		
プロトタイプ方式用 ：	○			■		
<b>検針業務</b>						
モバイル端末選定	○	○	■			
パッケージ仕様確認 ：	○	○	■			

では金額はどうでしょうか。システム開発ビジネスにおける、見積書の多くは次のように書かれます。

(株)××様

新システム開発に関するお見積書

項目	機器	単価	金額
1. ハードウェア			
サーバー	1台	380,000	380,000
クライアントPC	30台	160,000	4,800,000
⋮			
2. ソフトウェア			
パッケージ改良	2人月	1,200,000	2,400,000
			⋮

人月表示(1人で1ヶ月やる量が1人月、1人月あたりの単価を決めておいて、これによって作業金額を見積る)は論外です。ソリューションビジネスでは何を売っているのでしょうか。いくら何でも人間ではないはずですが。こんなことをやっているとう作業効率を上げ、システム開発の生産性を高めると売上はどんどん減っていきます。(以前、下請製造業が自らのコストダウンで自らの売上を減らしていったのと同じです)

ソリューションビジネスで売っているものは「ニーズを解決するサービス」です。ハードウェアもソフトウェアも部品であり、部品別の金額を出す必要などありません。ただユーザーは欲しいと言ってきます。何のために欲しいかといえば値引き折衝の材料に使いたいからです。ベンダーも値引きされるのがわかっているので少し高めに設定しておいたりします。こんなかけひきには何の意味もありません。

ソリューションビジネスに内訳があるとなれば、それはサービスの「部分」であり、ソリューション単位といえます。これをグループ化するなら業務、つまり外部エンティティ単位となります。

金額内訳		
1. 問合せ窓口業務	1式	4,500,000
・クライアントPC	5台	
・ミドルソフト	1式	
・ソフトウェア開発		
	⋮	
2. 検針業務	1式	3,200,000
・モバイル端末	30台	
	⋮	

こうすれば「提案にあたっての考え方」「システムイメージ」「ソリューションイメージ」「スケジュール」「金額」がすべてユーザー、業務、外部エンティティ単位に串刺しで見られるようになります。

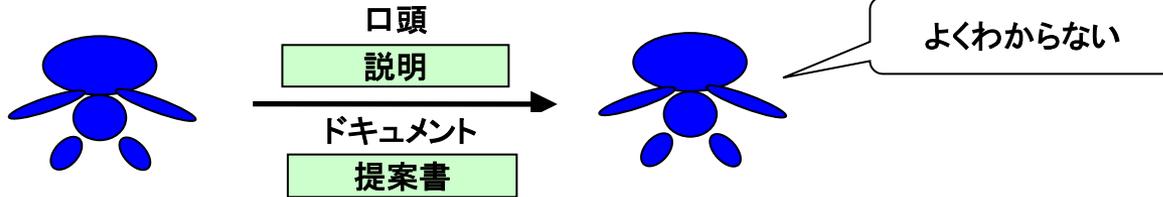
システムイメージはシステム設計の第一段階のドキュメントになります。データベース設計をしたら、その設計書はシステムイメージのサーバー部分とリンクして、問合せ窓口システムの設計書はシステムイメージの問合せ窓口部分とリンクし…と詳細化していきます。

こう考えると提案書を中心としたドキュメントは紙ベースの2次元のものでは不可能となります。画面上で、目次に沿ってページ単位に見たり、「問合せ窓口」だけを串刺しに見たり、サーバーの設計内容を掘り下げて見たり…という形が必要となり、複数のキー(外部エンティティ、設計・開発工程…)を持ったデータベース構造にすべきといえます。もしSEが「ソリューションビジネスのドキュメント」をプロとして設計するとしたらこういうデータベースになるはずです。

# THEORY31

「読んでわかる提案書」  
を作る

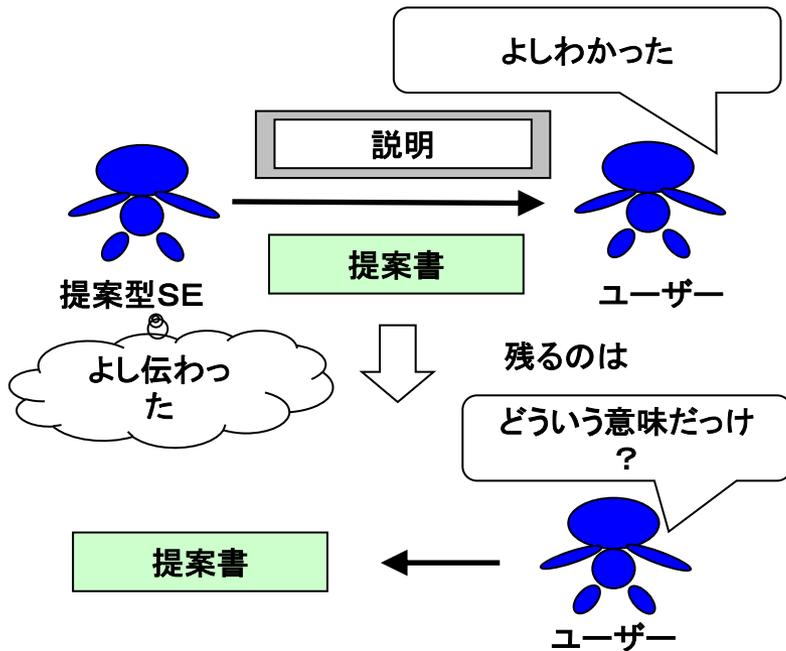
# プレゼンテーションの準備



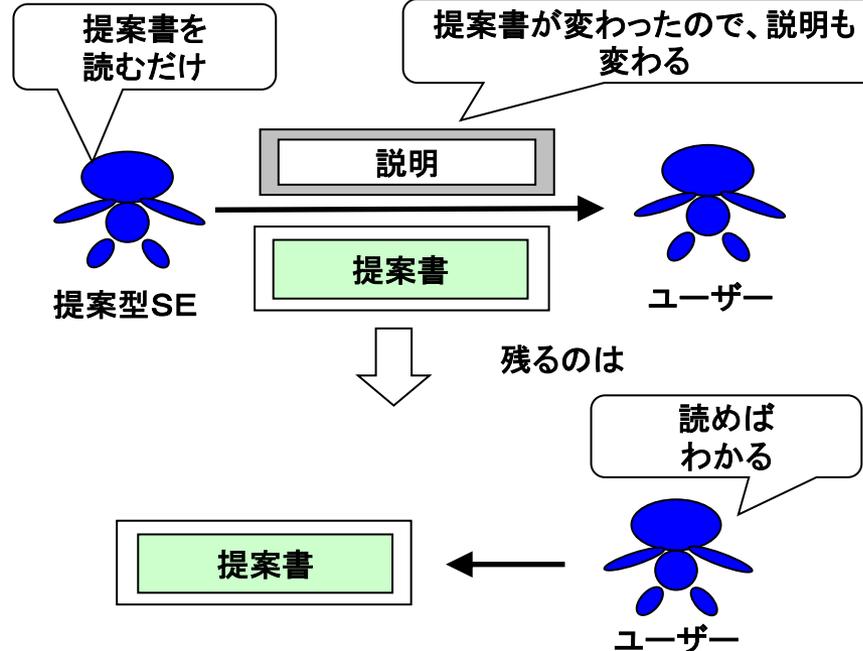
説明の仕方が悪い

提案書が悪い

説明だけを変える



提案書を変える



プレゼンテーションはインタビューとともに、SEが悩むことの多い仕事です。多くのソリューションベンダーの経営者は「うちのSEはプレゼンテーション能力がない」といい、プレゼンテーション研修を受けさせたりします。

しかし、プレゼンテーション能力とはどういう能力のことをいうのでしょうか。「相手にインパクトを与えるプレゼンテーション」といいますが、SEがユーザーにインパクトを与えてよいのでしょうか。もし「インパクト」という言葉の意味が「提案書に書いてあること以上に強い印象」を持つことだとしたら、本当にそんなことをして、仮にこの提案が通って受注できたとしても良いことがあるのでしょうか。トラブルのもとではないでしょうか。プレゼンテーション能力とは、どう考えても「正確にもれなく」考えていることを相手に伝える能力のことでしょう。内容(コンテンツ)と伝達(プレゼンテーション)と考えるなら、内容通りに伝達されることを考えるべきです。逆にいえば内容以外のこと(誤解)が伝わらないことも大切です。

プレゼンテーションはセオリー27で述べたように提案型SEがやるしかありません。「うちのSEはまじめでやや暗く、ハツタリがきかないから、プレゼンテーションに向いてない」とよく言いますが、「まじめでやや暗い人間」は正確性、網羅性を大事にするタイプであり、最もプレゼンテーションに向いていると思います。以下にSEが提案書をプレゼンテーションする時のポイントをまとめておきます。

## ①読んでわかる提案書を作る

提案書をユーザーに実際にプレゼンテーションする前に、SEがリハーサルで誰かに提案書を説明したとします。SEが提案書への追加説明を一切せず、ただ提案書を読むだけで終わりました。プレゼンテーションを受けた側がよくわからず、提案の主旨が伝わりませんでした。この時何をすべきでしょうか。多くの人は「説明が悪かった、不足していた」と考え、説明のやり方を直そうとします。しかしSEが正確に提案書を読んで、相手に伝わらないとしたら、それは提案書が悪いと考えるべきです。したがって提案書を直し、「読んだだけでわかる提案書」にすべきです。口で説明したことは残りませんし、後で確認のしようがありません。この提案書はこれからの設計・開発・運用の各段階で使っていく販売条件書です。システムトラブルのもととは、いつも「言った、言っていない」であり、「書いてある、書いてない」ではありません。

さらにいえば本番のプレゼンテーションで、何らかの事情で書いてあること以外の説明を追加した場合は、提案書はその説明が入っているものに差し替えることをSEに徹底させる必要があります。

## ②所要時間を守る

あたりまえの話ですが、ユーザーが「30分で終わらせてくれ」といったらプレゼンテーションは30分でやめます。31分目にやることがあるのです。そちらの仕事が気になって、以降のプレゼンテーションは聞いていません。聞いてくれないことが恐いのではなく、聞いてくれないで受注して、後でトラブルが起きることが恐いのです。これは「30分ですべてのプレゼンテーションを終えろ」ということではありません。広告代理店のプレゼンテーション競争のようにプレゼンテーションの時間精度やそのテクニックを競っているではありません。30分経ったら、プレゼンテーションをやめて、まだプレゼンテーションが終わっていないことをユーザーに伝え、延長戦をお願いすることです。

## ③バランスをとる

SEが説明すると、どうしてもシステムイメージ、すなわち「どんなシステムか」を中心に説明しようとしてします。ところがユーザーはそれに興味を示さず、延々と続く細かいそして難しいシステムの説明にうんざりし、次のソリューションイメージにたどり着いたころには、だんだん聞くのがイヤになっています。

「提案にあたっての考え方」「システムイメージ」「ソリューションイメージ」の3つは時間的にバランスをとり、もっとも時間のかかるシステムイメージは販売条件書なので、「詳しくは提案書をよく読んで欲しい。そのうえで疑問があればいって欲しい」という形にすべきです。システムイメージの細部のことで、後々トラブルが起こるということはありません。システムイメージの誤りの多くはソリューションイメージで発見できます。(システムイメージが違っていればソリューションイメージも違っています)

#### ④説明と視線を合わせる

プレゼンテーションをしていると、説明と違う所を見ているユーザーがいます。(結構意思決定者に多いといえます。)これは聞いてくれないから残念なのではなく、後になって「事前に聞いてない」というクレームが出るのが怖いのです。説明とユーザーの視線が合うように努力します。説明している所以外はユーザーに見せないようにできるプレゼンテーションツール(パワーポイントなど)を選べば、この問題は解決します。そのため提案書はプレゼンテーション時には配布せず、パワーポイントなどでの説明後、最後に配布するようにします。

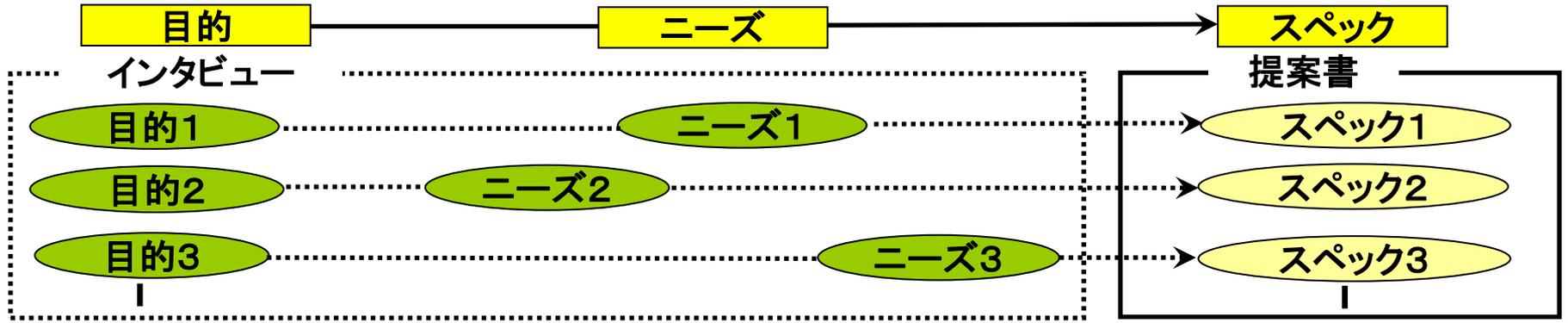
#### ⑤メモをとらせない

提案書を配布しない理由はもう1つあります。それはSEの説明を聞いて、ユーザーがメモをとることを避けるためです。ユーザーの作ったメモにSEは責任をとれません。「私のメモには×月×日に君がこういったと書いてある」これが一番怖いといえます。書いていること以外にプレゼンテーションしないのですから、ユーザーにとってメモは不要です。もし追加事項があればSEがメモをとります。

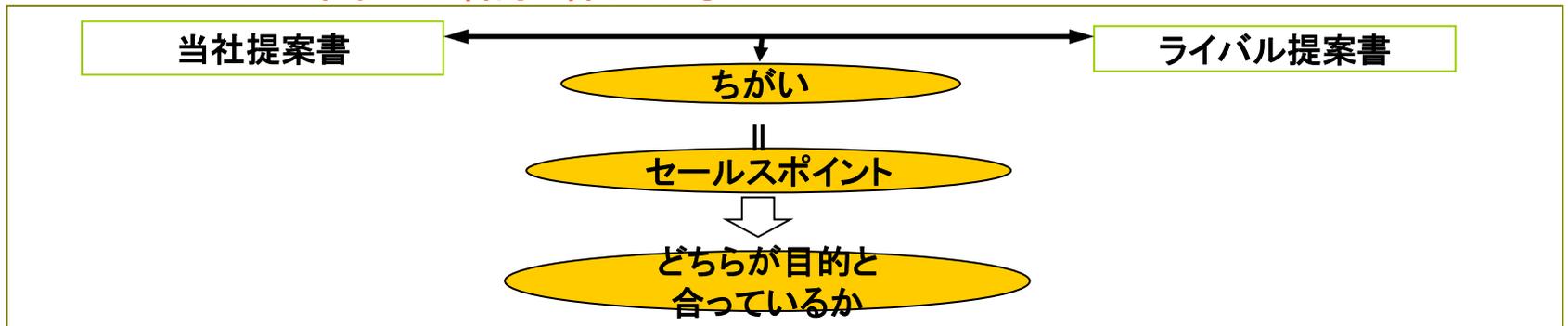
# THEORY32

目的・ニーズ・スペックの  
ベクトルが合っているか

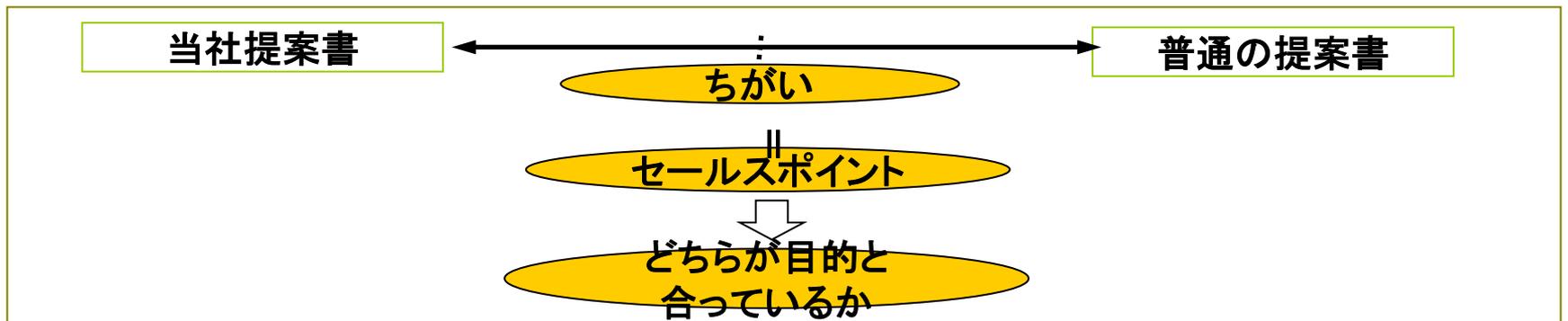
# ベクトルは合っているか



# セールスポイントが目的に合っているか



他社提案書はわからない



提案書はユーザーにプレゼンテーションする前に、ソリューションベンダーの社内でチェック(これをドキュメントレビューという)することが大切です。この提案書のドキュメントレビューではユーザーへプレゼンテーションするのと同じ環境で、SEがユーザーにプレゼンテーションする立場でやり、他の人に見てもらうことが大切です。

このセオリーではレビュー時に何をチェックすれば良いかを挙げておきます。次の6点は①から順に大切であり、先の条件を満たしていないと後の条件を満たしていても意味がないものといえます。

### ①わかったか、わかりやすいか

提案書では「わかりやすさ」がすべてに優先します。ユーザーにわかってもらえれば何とかかなりです。提案書で一番怖いのが、プレゼンテーション終了後にユーザーから「大体良い」といわれることです。「大体良い」というのは「全部は良くないが、どこが悪いかをうまく指摘できない」状態の時に使われます。このちょっとした悪い点が、後でとんでもないトラブルに発展することが多くあります。この悪い点の多くは、提案書という文書ではなく(そうであれば××ページの××行目がおかしいといえます)、口頭で説明した部分にあります。前にも述べましたが、「書いてないことは言わない」のが基本です。レビューの時に提案書が読みづらく、口頭で説明する必要がある場合は、その場でその説明を提案書に追加します。

## ②ニーズをとらえているか

ソリューションビジネスの出発点である提案書作成という仕事のインプットはニーズです。インプットが違っていれば、アウトプットである提案書も当然違っており、すべてやり直しです。提案書ではシステムスペックをレビューしがちですが、もっと大切なことはニーズが**あっているのか**、**もれていないか**をチェックすることです。こう考えるとユーザーへのインタビューは複数で行く必要があるといえます。人間は難しいニーズ、やりたくないニーズははずそうとします。インタビューで聞いたニーズは解決しようが、**しまいが提案書にすべて書きます**。「解決できない、しない」というのも提案です

## ③ニーズの表現

ニーズはインタビューから生まれるので、原形は「音」です。この音を提案書という2次元の用紙に書こうとすれば、必ず歪みます。そして歪みはいつもSEIにとって都合良く、ユーザーにとって都合の悪いものです。これをユーザーが提案段階で指摘してくれれば良いのですが、不思議と見過ごされます。何とかインタビューで聞いた音に近づけないかを考えます。

#### ④目的・ニーズとスペックのベクトルが合っているか

ニーズのもっとも原始的な姿は「目的」つまり「何のためにシステムを導入するか」といえます。そしてニーズのもっとも具体的な姿は「システムスペック」つまり「どんなシステムにするか」です。このシステムスペックが固まれば次の設計工程へ進むことができるのです。

システム化において目的はいくつかあり、したがって1つのシステムにもニーズはいくつかあります。あるニーズは目的そのものであり、どんなシステムにするかと聞いてもあやふやなものであり、あるニーズはほとんどスペックになっているという形で連続分布で存在しています(システム開発ビジネスに慣れ親しんだSEはニーズがシステムスペックに近いほど喜びますが、ソリューションビジネスでは、ニーズが目的に近いほど高付加価値サービス、つまり魅力的なサービスとなります)。

ソリューションビジネスとは何度も述べているようにこのニーズをスペックに変えることです。したがってレビューのポイントは目的、ニーズ、スペックが一直線に並んでいるか、つまり、目的を解決するスペックになっているか、インタビューしたニーズとスペックに整合性があるかということです。

ソリューションのものさしは常に「目的」であり、合目的性が大切なチェックポイントです。

## ⑤一貫性

これは提案書の最初と最後でつじつまが合っているかというようなことではありません。そんなことはすぐに発見できるし、後々トラブルの原因になることもありません。

提案書作成はソリューションビジネスの1フェーズです。このフェーズの目的は「ユーザーにGOと言ってもらう」ことです。提案はビジネスですから、生産性も大切です。生産性の高い提案書・プレゼンテーションとはユーザーに早くGOと言ってもらうものであり、生産性が低いとはなかなか言ってもらえないものといえます。ここでいう一貫性とはいかにGOといいやすいかです。これについてはセオリー30で述べたように、あらゆる角度からチェックでき、特に個々のユーザーが自分の業務単位にチェックできるかが大切といえます。

## ⑥セールスポイントが目的と合っているか

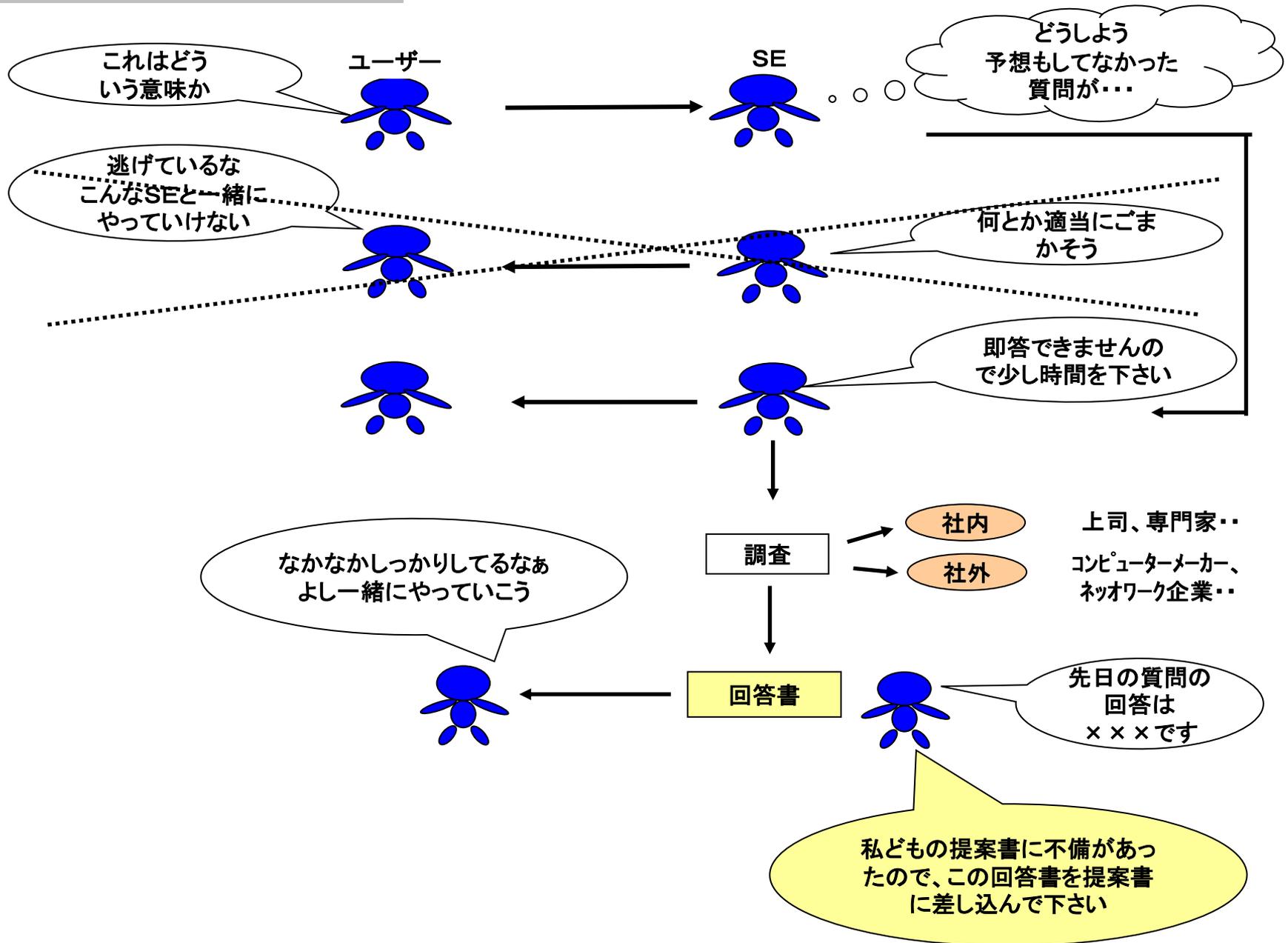
セールスポイントとはファジーな言葉ですが、本書ではこれを「他社とのちがい」という意味で使います。したがって、ソリューションベンダーが1社しか提案書を出さないのであれば、セールスポイントはありません。またベンダーがライバルと競合し、セールスポイントを持てば、両社の「ちがい」ですので当然ライバル企業にもセールスポイントがあることになります。したがってセールスポイントを持つことが大切なのではなく、そのセールスポイントのどちらが目的と合っているかであり、合った方の勝利といえます。

しかしライバルの提案書はセオリー37で述べる著作物であり、競合企業の手に入ることはあり得ませんし、あってはいけないことといえます。ではどう考えたらよいでしょうか。それは「普通」です。こういうニーズの場合「普通」どういう提案をするのかを考え、その「普通」と提案システムがちがっていれば、それをセールスポイントと考えることです。先ほど述べたように、ちがいを出すことが大切なのではなく、そのどちらが(今回の提案か普通か)目的に合っているかを考えます。

# THEORY33

プレゼンテーションでの  
質問を恐れるな

# 答えられない質問が出たら



提案書を「読んでわかるよう」に作るとすれば、一体何のためにプレゼンテーションをするのでしょうか。プレゼンテーションは「提案書を説明する」ためではなく、「提案書を見てユーザーがわからない所、提案書がまちがっている所」をユーザーに教えてもらうために行っているのです。もちろん提案書はわかるように正確に全力を込めて作ります。しかし作った人と見る人は違う人なので、わからない所、まちがっている所は必ずあるはずです。ユーザーがわかりづらいところをあらかじめ予想して口頭で説明するのではなく、提案書からわからないところを自分としてはなくてして、そのうえでユーザーにわからないところを聞きにしているのです。

提案書は以降、SEとユーザーが共同で仕事を進めていくための確認書です。SEがユーザーの業務をわからないのは当然です。その業務を効率化したり、高度化するためのシステム提案ですから、まちがっているところはあるはずです。この提案書は「きっとまちがっている」と思ってプレゼンテーションを行うべきです。あまり自信を持って堂々とプレゼンテーションが行われると、それを受けるユーザー側は気後れして、まちがっているところを指摘することをためらったり、忘れたりしてしまいます。プレゼンテーションはユーザーにまちがっている所を見つけてもらうために行っているのです。人間はまちがっている所を指摘されると、がっかりしたり、不愉快になる動物です。ユーザーからまちがい指摘されたら、「よかった。こんな早いタイミングでまちがいがわかったので、リカバリーが最小限にすんだ」と思うべきです。

よくSEがプレゼンテーションをいやがる理由に「突っ込まれるといやだ」というものがあります。しかしプレゼンテーションは「突っ込まれる」ために行っているものであり、まちがいの指摘も質問も出なかったら、わざわざプレゼンテーションに行った意味がないといえます。

提案書のプレゼンテーションではよくオープンプレゼンとして、そのシステムの関係者すべてが自由に出席できるような形をとります。この時必ず来るのがコンピュータに妙に詳しい人です。このコンピュータ通の人はプレゼンテーションのQ&Aタイムで必ずとっていいほど質問をします。提案システムの本質から離れた技術的な細部の質問を数多くします。彼らは「わからないこと」ではなく、「わかっていること」、「知っていること」を質問します。質問の目的は提案書を理解することではなく、自分の知識レベルが高いことを周囲に認めてもらうことにあります。「知っていること」を聞くなれば無限に質問は続きます。一方質問を受けるSEも当然コンピュータに詳しいのでその質問に答えることができます。またよせばいいのに詳しく説明したくなります。こうして技術的な質問・回答が延々と続くことになります。これが終わるとSE側はガッツポーズです。「よし。嵐のような質問をすべてさばいたぞ。これでユーザーの評価も高いだろう」

しかし、このコンピュータ通の人がシステム投資の意思決定者であることはまずありません。このような質問・回答を聞いていた意思決定者である社長などの反応は「あのSEはなんだかうちのパソコンおたくと波長があっていた。彼に任せられないからプロに頼もうと思ったのに、プロのSEもあの程度か。今の段階ではあのような技術的なことではなく、もっとシステムの本質について話すべきだろう。」というものです。

ではどうしたら良いのでしょうか。方策は2つあります。1つはその場では答えず、いくつあっても良いから質問をすべて聞いて「技術的な質問ですので、別途文書で回答します」と言って「逃げる」ことです。しかしこうするとこのコンピュータ通の人に後々、意地悪される可能性があります。「せっかく質問したのに答えようとしなない。いやな奴だ」というものです。SEにとってこのコンピュータ通の人が必要となる時が必ず来ます。それはシステムの運用段階であり、このとき彼が現場のシステムインストラクター・ヘルプデスクとしての機能を果たしてくれます。これが彼に嫌われると意地悪されて、逆に運用を妨げられる危険があります。

# THEORY34

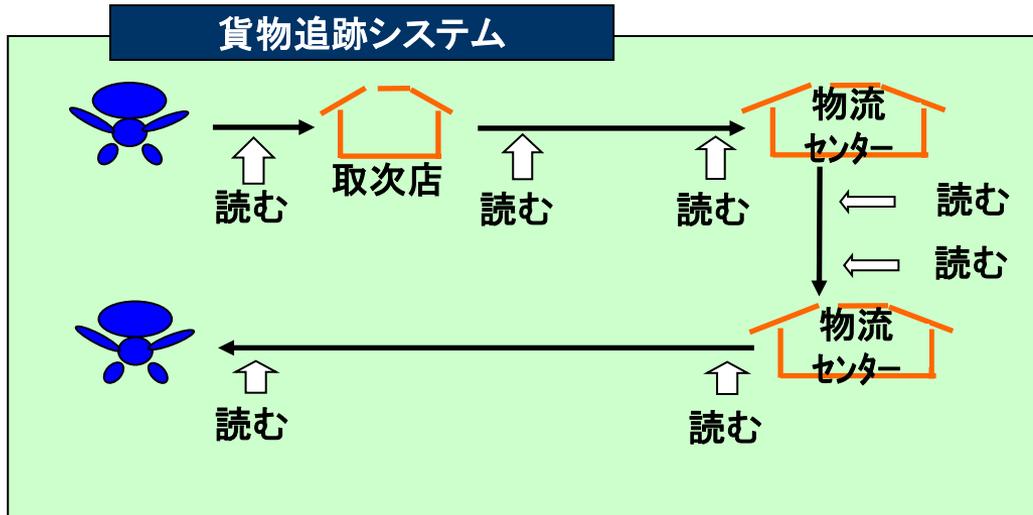
プレゼンテーションでの  
反対を恐れるな

# 現場の反対

荷物がどこにあるかわかってもしょうがない



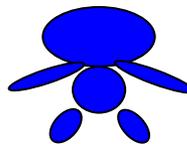
ドライバー  
ドライバー部長



荷物がどこにあるかわかり、取次店へのサービス向上となります



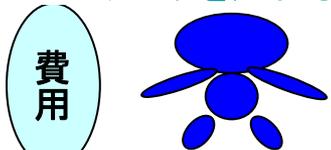
# 意思決定



(共通の上司)

決定

データを入れる



データを見る



ヤマト運輸は「貨物追跡システム」で宅配便業界を勝ち抜きました。これは宅配便の伝票にバーコードを打っておいて、それを受取りから移動、引渡しまでの各段階でスキャンして読むものです。このバーコードをスキャンした回数をコンピュータで見れば、荷物が大体どこにあるかをわかるようにしました。このシステムによってコンビニエンスストアをはじめ多くの取次店を獲得することができ、売上を急速に伸ばしていきました。

仮に、まだこのシステムがなかった時代にソリューションベンダーのSEがこれに気づき、ある宅配便企業にこの貨物追跡システムを提案したとします。この時宅配便企業のドライバーにプレゼンテーションなどで意見を聞けば、まちがいでなくこのシステムに反対するでしょう。「荷物がどこにあるかわかって何がうれしいのか。目の前にあるじゃないか。こんなことをやれば配送効率が落ちるし、時間だって守れなくなる。」ではそのドライバーの上司である、ドライバー部長はどうでしょうか。当然反対です。ドライバーの利益代表者であるドライバー部長がドライバーのために反対しなければ何のためにいるのかわかりません。「バーコードスキャンなどの余計な入力作業をドライバーにやらせれば、優秀なドライバーは他社へ転職してしまう」と発言するでしょう。

ソリューションベンダーのSEはこれにどう対応すべきでしょうか。今までのITベンダーの営業スタイルでは、ドライバーに「どんなに操作が簡単か」を説明し、理解を求め、根回しし……と進め、結局理解は得られず、提案は実現できません。

こういった反対に対してSEがとるべき手段は、その反対は「社長に言って下さい」ということです。どう考えてもこのシステムを導入すればドライバーの仕事は負荷が増えます。しかし、「ドライバーの負荷が増えるのでやらない」という結論になるとは限りません。このシステム的意思決定者はドライバーでもドライバー部長でもないからです。

現代では上のように「データを入れる」という人と「データを見る」という人が企業内の組織上で遠く離れているシステムがうまく実現できません。それは「データを入れる人」には何のメリットもなく、ただ入力作業が増えるだけだからです。見方を変えればこういうシステムが残されており、ソリューションビジネスのメインターゲットといえます。

よく考えればこの「データを入れる人」と「データを見る人」の上司をどんどん上にたどっていくと、いつか社長など1人の人にぶつかるはずです。つまりこの2人の共通の上司であり、「データを入れる仕事」と「データを見る仕事」の両方をコントロールし、冷静な目で見ている人がいるはず。こういうシステムは現場の意見や多数決で決めるのではありません。一般に「データを入れる人」は多数で「データを見る人」は少数です。場合によっては「データを見る人」も、見たからといって自分自身の仕事が合理的、効果的になるわけではなく、会社全体としてのメリット（顧客が増える。売上が伸びるなど。こうなるとデータを見る人もさらに仕事が大変になる）が上がるだけの場合も多いといえます。つまり社長などの意思決定者が「データを入れる」費用と「データを見る」効果をはかりにかけて「効果が重ければやる。費用が重ければやらない」という決定を下すべきといえます。

プレゼンテーション、提案書における提案型SEの立場はこの意思決定者が、はかりにかけて判断をするため正確な情報（費用見積、効果見積）を提供することにあるのです。この情報を意思決定者が持っていないので、意思決定をためらっているのであり、情報提供がソリューションベンダーに求められていることを肝に銘ずべきです。

SEがもっともやってはいけないのは「費用を少な目」に見積り、「効果を大き目」に見積って意思決定者へ渡すことです。セールスが作った提案書がユーザー側社長から信頼されないのは、何とかしてシステムを受注したくて、どんなにシステムが素晴らしいかを訴え、入力などの作業が大変なことを隠しているからです。

## 第6章 プロジェクト管理モデル

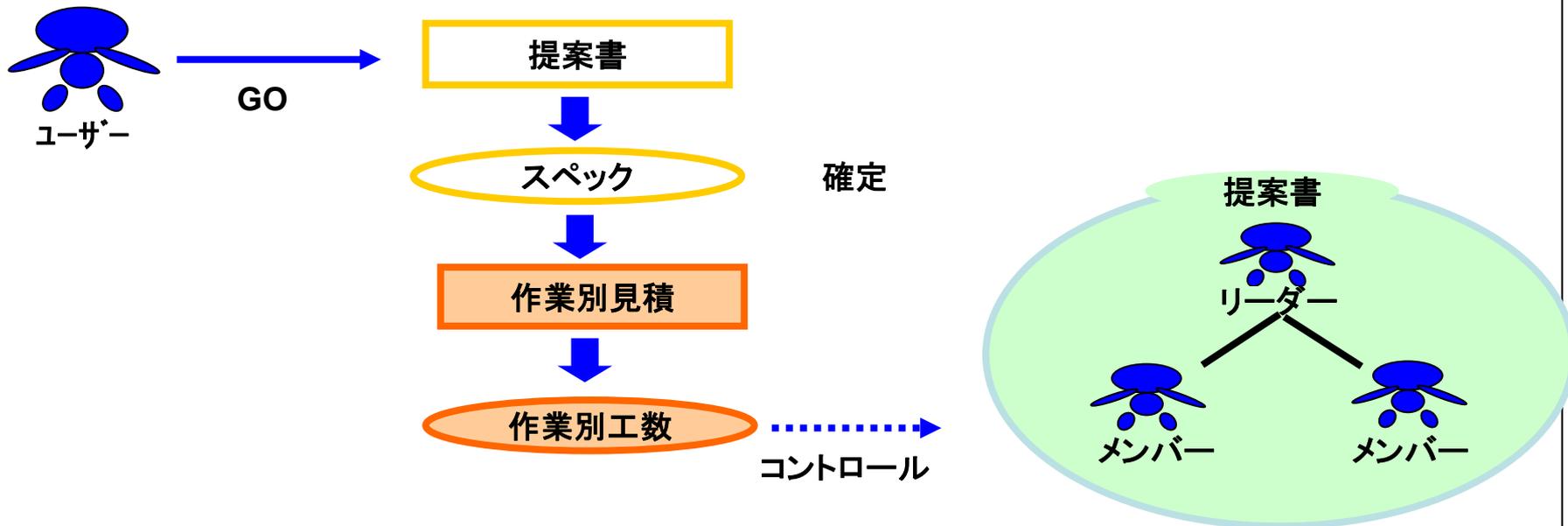
### トータルセオリー

プロジェクトリーダーは法律などのルールを理解し、ヒト、モノ、カネ、トキを管理する。エラー、工程遅延はなくすのではなく減らし、起きたときどうするかを考える。

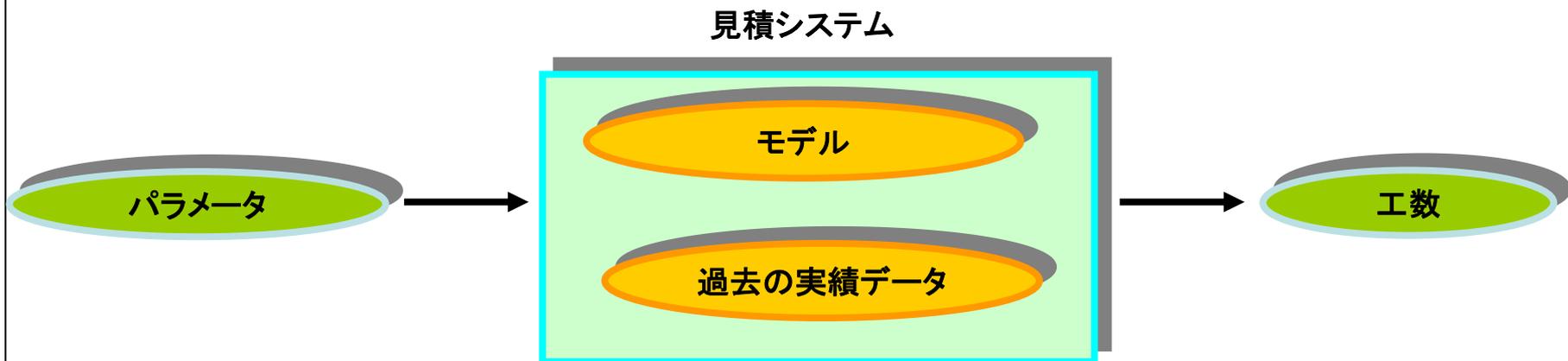
# THEORY35

見積は過去の平均値

## 見積の意味



## 見積システム



提案書にユーザーからGOが出るとそのシステムを開発するためのプロジェクトが立ち上がります。このプロジェクトのリーダーには一般にユーザーとスペックをつめてきた提案型SEが成ります。

提案書にGOが出されるとこの時点でシステムスペックが確定されます。見積は提案書を出すときにも無論行われますが、最終スペック確定とともに最終的な見積が行われ、各作業別に工数(人日、人月といった単位。人日は1人で1日できる量)が計算されます。

従来のシステム開発ビジネスでは工数を積み上げ、原価を出し、利益をのせてユーザーへの販売価格を決定するという形で進めてきました。ソリューションビジネスではセオリー4で述べたように妥当な投資金額を計算し、それを枠組みとしてスペックを提案していくという形をとります。しかし、だからといって工数、原価が軽視されるということではありません。

最終スペックに基づく作業別の工数がプロジェクトをコントロールしていくことになります。見積工数内におさえて作業を終了させることがこのプロジェクトの命題であり、これを達成すれば成功プロジェクトといえます。

積をシステムとして考えると、インプットは工数以外のデータ(パラメータ)であり、アウトプットはプロジェクトで発生する作業別の工数といえます。そしてこのパラメータを工数に変換するには、「過去の工数実績」と「工数予測方法(モデル)」が必要となります。システム開発ビジネスの初期の頃はこの過去の工数実績、モデルが熟練プロジェクトリーダーの頭の中にあり、いわゆる経験とカンと度胸で工数を見積っていました。システム開発ビジネスにおいてはその後、過去の実績データがたまっていくにつれ、この見積作業をコンピュータで実現しようという動きが出て、ファンクションポイント法、COCOMOなどの見積モデル(見積の方法論)が登場しました。これらはいずれもパラメータはスペックおよびその周辺情報(開発環境など)であり、これを回帰分析(過去の実績から考えてもっとも近い値を計算する)して工数を求めるというものです。まさにソリューションビジネスにぴったりのツールといえます。

しかし多くのITベンダーはこの見積システムの導入をためらってきました。その理由は次の2つです。

## ①あたらない

コンピュータで回帰分析した結果である見積工数が実際工数と異なるという理由です。回帰分析とは「2つ以上のデータ(スペックと工数)の関係を結果から類推してパターン化・数式化して、片方のデータ(スペック)がわかったとき、片方のデータ(工数)を予測計算しよう」とするものです。いいかえれば回帰分析は「過去の平均値」のようなものです。つまり「こういうスペックのシステムを作るとすれば、過去は平均してこれ位の工数がかかっていた」ということをパターン化して計算するというものです。過去の平均値に「あたり」、「はずれ」はありませんし、過去の平均値を使いたくないというプロジェクトリーダーもいないはずです。もしこの平均の出し方(モデル)がおかしいと思うなら、使わないのではなく、モデルつまり計算式を考えればよいことになります。

## ②責任がとれない

プロジェクトリーダーがよく言うのが、「そんなコンピュータが予測するような工数に責任がとれない」(しかしコンピュータのSEがこんなことをいうのも変ですが)というものです。プロジェクトがこの工数どおりに終わらなければ納期遅延、赤字プロジェクトになる可能性もあります。しかしだからといって、自分の目標(工数)を自分で自分に都合よく立てるというのも変です。過去の平均的プロジェクトと今回のプロジェクトのパターンが違うなら、それを見積に考慮すれば良いことになります。

逆に見積システムを導入するメリットを挙げればキリがありません。

①スペックの変更が工数に与える影響がよくわかる

ユーザーと提案段階で話している時、スペックを変えると金額や納期にどのような影響を与えるかを簡単に説明できます。

②どうすれば見積ることができるかがわかる

「見積が出来ない」という理由で受注できないケースもあります。セオリー27で述べたように「基本設計終了後別途見積」ではユーザーは納得しません。基本設計などしなくても見積はできるはずです。見積システムを導入すれば「何がわかれば見積ることができるか」がわかります。さらに見積という作業自体にどれ位工数がかかるかもわかり、提案書作成のための工数もわかるようになります。

③見積の生産性が上がる

見積という仕事は熟練のプロジェクトリーダーにとっても大変な作業です。この作業自体の生産性が上がります。

#### ④公平になる

見積工数は見方を変えると、プロジェクトへのメンバー配分や、各作業ごとの目標原価・納期設定などにも使われます。声の大きい人は大きい目の見積工数を出し、人を確保し、時間を確保し、利益を確保します。まじめで気の弱い人はその逆です。これでは現代的ビジネスでなく弱肉強食の原始時代です。同じスペックなら誰が見積っても同じ見積工数になるようにすべきです。

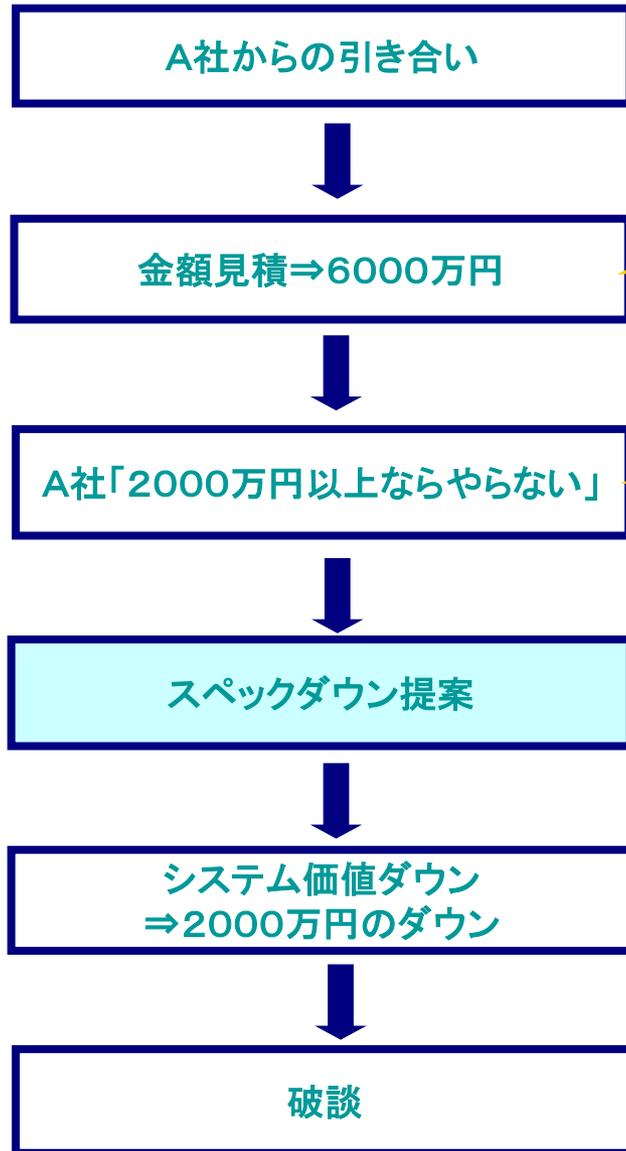
#### ⑤ナレッジ

予測値である見積はあてることも大切です。あてるためには予測値である見積と実績値である実績工数をデータベース化し、差異分析していくことが大切です。あてるためには予測しなくてはなりません。これには長い時間を必要とし、そのためソリューションビジネスの企業としての体力となっていくます。体力のない企業(見積システムが弱い)がいきなり全力で走ろうと思っても無理です。近年天気予報が当たるのは、天気の実績がたまっただけではなく、天気を予測し、ハズし、その理由を考え、モデルを変えて当たるようにしてきたからです。

# THEORY36

システム開発では  
部品流用する

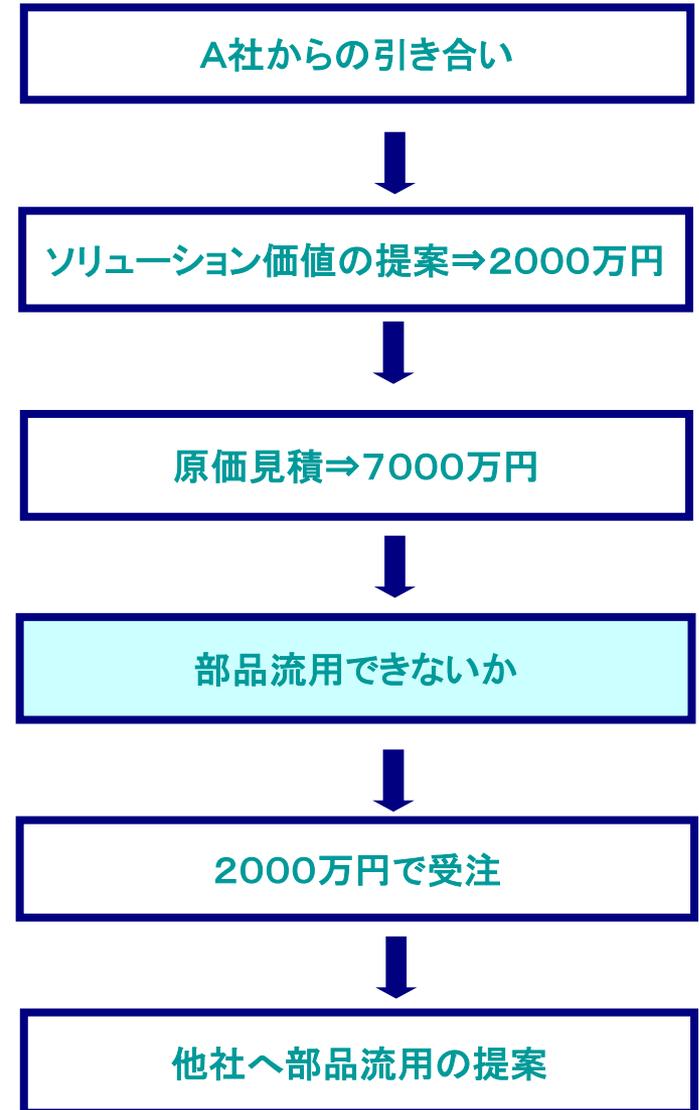
## 従来のシステム開発ビジネス



売っても  
いい値段

買ってもい  
いギリギリ  
の値段

## ソリューションビジネス



従来のシステム開発ビジネスではセオリー4でも述べたように顧客(A社)からシステム開発の引合をITベンダーが受け、システムの原価見積を行い、利益を加味して6000万円と見積書を出します。これはいってみれば、ITベンダーがシステムを「売ってもいい値段」といえます。

どんな商品にももう1つ「買っていい値段」があります。これがA社にとって2000万円だとすれば、ITベンダーは仕方がないのでスペックを落として原価を下げようとします。しかしそれに伴ってシステム価値2000万円も下がってしまい、結局破談となってしまいます。見方を変えると皆がこのシステム開発を断ってきたので、このように原価とリターン、売ってもいい値段と買っていい値段がかけ離れたシステムは、まだまだたくさん残っているはずです。

ソリューションビジネスではこれを何とか2000万円を受注する努力をし、新しいマーケットを築き上げるべきです。そしてその答えは1つです。それはソフトウェアがコピーして流用できるという特徴です。つまりA社のシステム開発で利益が上がらなくても、A社と同じシステムを望んでいる企業が他にあれば、その企業にこのシステムをコピーして売れば良いことになります。

そのためには、A社のシステムを作る時に各社共通と思われる部分(部品と考える)とA社だけの個別だと思ふ部分に分けて開発する必要があります。つまり部品の再利用というあらゆる製造業がやってきたことをソリューションビジネスでもやればよいわけです。しかも自動車など他の製品は部品を再利用するのにその製造コストがかかりましたが、ソフトウェアはほぼ無料です。(この部品のこともオブジェクトといい、部品流用の開発手法のこともオブジェクト指向といいます。しかしセオリー3で述べたデータのオブジェクト指向と混乱するので、本書では部品流用型の開発環境とよんでおきます。)しかし、各社が完全に共通で使えるシステム(経理、給与計算など)はすべてパッケージプログラムが出来て普及していますし、全然ちがうシステムでは流用できません。やや同じ、似たようなシステムを求めている企業群ということになります。

ここまでは誰もが納得するのですが、多くのベンダーはこの部品流用型の開発環境を、以下のような理由でなかなか導入できません。

①個別のプロジェクトで考えると、部品流用型の開発環境の方が原価アップとなる(プロジェクトごとの個別損益計算を行なっており、それがプロジェクトの成績となる。これを落とすことをプロジェクトリーダーが反対する)。

②新しい開発手法にプロジェクトリーダーから反対が出る(「やったことがない」のでやりたくない。「トラブルが怖い」)

### ③部品流用型の開発環境がどこにも売っていない

しかし逆に考えるとライバル企業にこれをやられてしまうと、ライバルはどんどん部品をため、体力をつけ、価格競争力を増していくことになります。つまり他社がやる前にやるしかないという結論になるはずです。

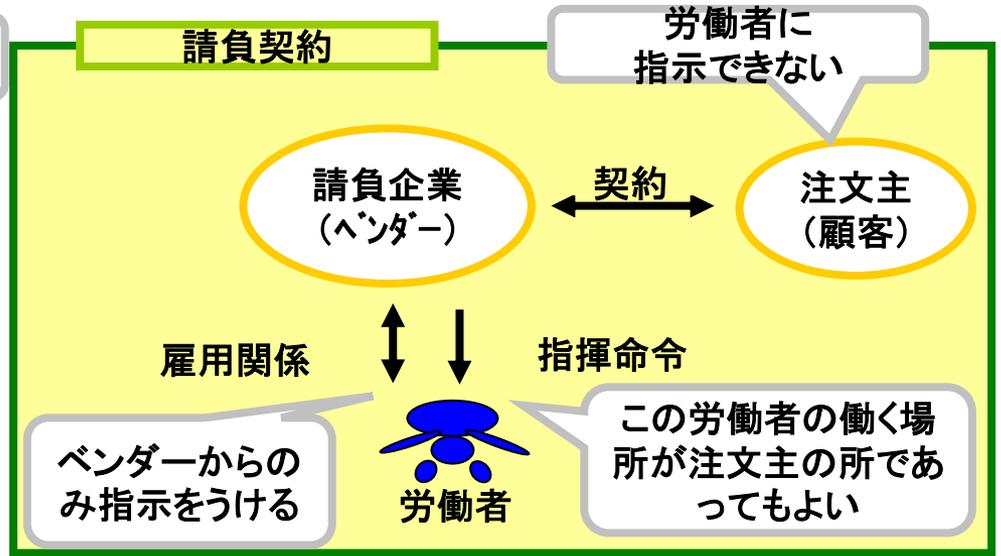
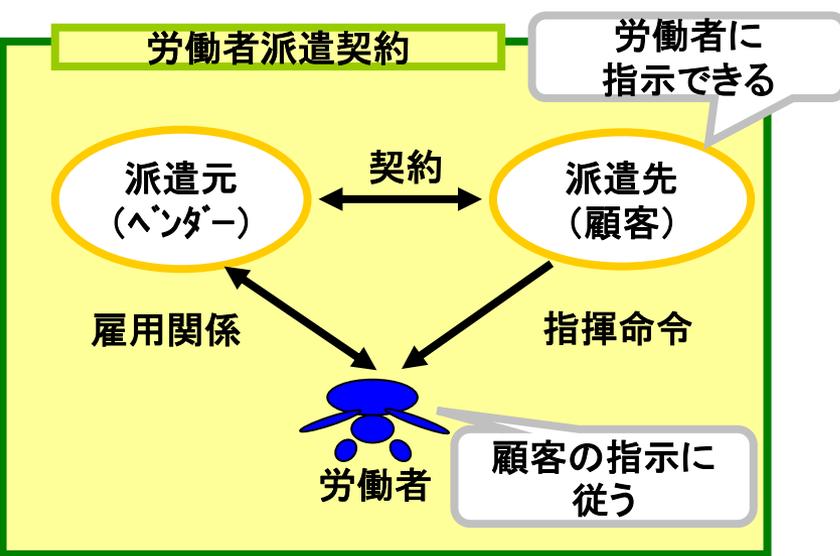
①については個別のプロジェクトの目標原価と販売価格を切り離すしかありません。そのうえで目標原価はプロジェクトリーダーの責任、販売価格による利益責任は経営側が引きとるようにすべきです。つまりプロジェクト管理手法の変更です。②については「やるべきか、やらざるべきか」という経営者の意思決定しかありません。一言「やれ」というだけです。③についてはソリューションベンダーが自社で作るしかありません。自社にSEがたくさんいるはずで、自動車の部品システムが作れて、ソフトウェアの部品システムが作れないはずがありません。設計の単位、開発の単位、ドキュメントの単位をすべて部品をキーとして考え、部品自身(ソフトウェア、ドキュメント...)と部品属性(いつ、誰が作り、どのような機能か)を合わせてデータベース化していくというもので、極めてノーマルなデータベースシステムです。

部品流用型の開発環境は、ちょうど製造業が手作りから機械導入に向かっていくようなもので、現場の技能工や班長は反対します。経営者から見れば製造業同様に当初投資コスト(工場を作るコスト)が生じ、それをゆっくり回収(生産性が上がる)していくものといえます。

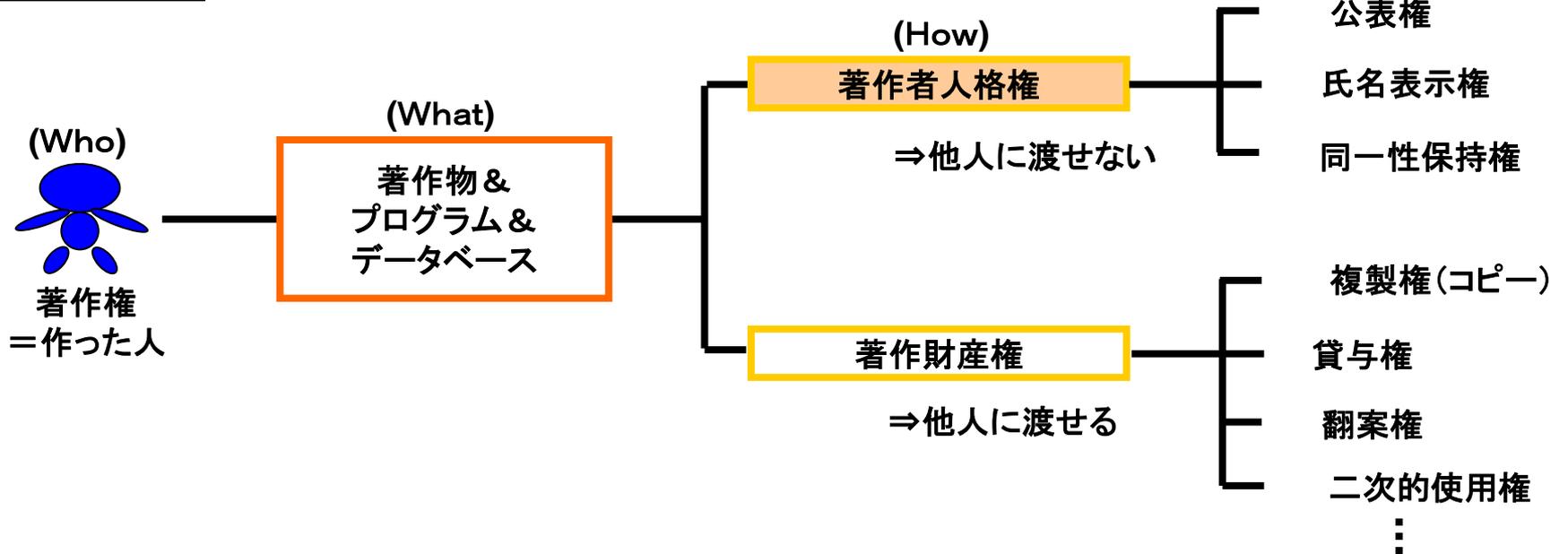
# THEORY37

システム開発は  
外注しない

# 労働者派遣事業法



# 著作権法



プロジェクトリーダーとなる提案型SEが必ず知っておくべき法律は次の2つです。労働者派遣事業法はプロジェクトリーダーとして顧客との契約、およびメンバーの労働環境のために、著作権法はソリューションビジネスの成果物であるシステムがその保護を受けるからです。

### ①労働者派遣事業法

従来システム開発は多くのソフトウェア企業(元請、下請企業)が協力して行ってきました。またSEがユーザーのコンピュータルームでユーザーとともに仕事をしたり、通常の勤務とは異なる形態をとることが多かったといえます。そのため指揮命令系統がはっきりせず、ユーザーがSEの上司のような形で仕事が進められることもありました。

1986年に労働者派遣事業法が施行され、システム開発の契約スタイルは法的には派遣契約と請負契約の2つに分かれることになりました。派遣契約はこの労働者派遣事業法で規制され、同法ではこれを次のように定義しています。

「自己の雇用する労働者を、当該雇用関係の下に、かつ、他人の指揮命令を受けて、当該他人のために労働に従事させることをいい、当該他人に対し当該労働者を当該他人に雇用させることを約してするものを含まないものとする。」

この定義にあたる労働者派遣をこの法律の対象として、一定の制約の下にこれを認めるとともに、それ以外の労働者派遣を逆に認めないということです。

ここでは労働者派遣の対象の労働者を「派遣労働者」、労働者派遣を業として行うことを「労働者派遣事業」、労働者を派遣する企業を「派遣元」、派遣を受ける企業を「派遣先」といいます。

先ほどの労働者派遣の定義を分解してみると次のようになります。

- ・「自己の雇用する労働者を」⇒自社の従業員以外を派遣してはいけない。二重派遣は禁止（A社がX社へ労働者派遣を行うとき、B社より派遣労働者を受け入れて、その労働者をA社からX社へ派遣してはいけない）
- ・「当該雇用関係の下に」⇒派遣労働者は派遣元と雇用関係を維持したまま。
- ・「他人の指揮命令を受けて、当該他人のために」⇒本来派遣元がもっている指揮命令権を派遣先の企業へ移し、その派遣先の企業の仕事を行うということ。指揮命令というのは「この仕事をやりなさい、終わったら次にこれをやりなさい」という仕事の命令をさしている。
- ・「当該他人に対し、当該労働者を当該他人に雇用させることを約してするものを含まない」⇒労働者派遣事業は職業紹介業や労働者供給事業とはちがう。

従来は派遣という形で実際は仕事を行っている場合でも、業務請負という形をとっていましたが、この法律のよって派遣と請負がはっきりと区別されることになりました。請負契約の場合には、労働者を指揮命令できるのはその労働者を雇用している請負業者（労働者派遣の派遣元にあたる。システム開発契約ではITベンダー）だけであり、注文主（労働者派遣の派遣先にあたる。システム開発契約では顧客）にはその指揮命令権はありません。一方、労働者派遣では、その指揮命令権が派遣元（ITベンダー）から派遣先（顧客）へ移されるという違いがあります。したがって、請負契約でシステム開発を外注し、かつその仕事をユーザー側の企業で行う場合でも、ユーザーはベンダーのSEに「あれしろ、これしろ」と命令してはならないのです。

この法律が施行されて、多くのITベンダーは派遣事業ではさまざまな規制(管理台帳の作成など)を受けるので、システム開発の契約には請負契約を選択しました。これによってITベンダーはソフトウェアという成果物に企業として責任を負うことになりました。(派遣契約では成果物責任はありません)

ソリューションビジネスにおいては無論請負契約がされますので、この方式は当然引き継がれ、ユーザーとは常に企業と企業として対応していくことが必要となります。またソリューションビジネスについては、様々なことをユーザーと打ち合わせしていく必要があります。請負契約ではソリューションベンダー側は窓口を一本化する必要があります。その任を担うのが提案型SEであり、プロジェクトリーダーといえます。

## ②著作権法

著作権法とはもともと小説、絵画、音楽などのいわゆる著作物を保護するために、作られた法律でしたが、何度か改定されコンピュータのプログラム、データベースもその対象となりました。

著作権法では著作権はすべて著作者にあると定めています。著作者とは著作物を作った人、プログラムでいえば開発者がこれにあたります。個人が単独で著作したときはその著作者は明らかなのですが、それ以外の場合には次のように定められています。

## ( i ) 法人著作物

法人著作物とは次の3つの条件がそろったときに、個々のプログラム作成者などでなく、その法人(企業など)を著作者と認めるものです。

- ・法人が企画、立案していること
- ・法人の従業員などが職務で作成していること
- ・契約、勤務規定に定めがないこと

## ( ii ) 共同著作物

共同著作物とは2人以上(法人を含む)が共同で作成した著作物で、著作者各人の寄与を明確に分離できないものをいいます。共同著作物は関係者全員が共同で1つの著作権をもつと考え、共同著作者の1人であっても勝手にプログラムを使用できず、著作者全員の承諾が必要になります。たとえば、A社とB社で共同でプログラムを開発し、かつその寄与が明確に区別できない(この部分はA社、この部分はB社と区別できない)ときは共同著作物となり、A社といえどもB社に断りなく勝手にコピーしたり、販売したりはできません(ただし、B社は正当な理由なしにこれを拒むことはできません)。

### (iii) プログラム作成を外注したとき

プログラムの著作者はそれを著作した者という明確な定義がありますが、その作成を外注した場合には解釈が難しくなります。つまり、その外注した部分が、全工程を一括外注したのか、プログラム設計以降なのか、プログラム作成だけなのか、などいろいろなケースが考えられるからです。

著作権には次のような考え方があります。

- ・依頼者やテーマを与えた人は著作者ではない
- ・命令どおりに動いた人は著作者ではない

この2つから考えると、派遣契約ではなく請負契約で一括外注の場合は受注会社(元請と下請なら下請企業、ユーザーとベンダーならベンダー企業)に著作権があり、その他の場合はケースバイケースといえます。

また著作者を保護している著作権はソリューションイメージのように大きく2つの部分に分かれます。ここで大切なことは著作者人格権(誰が作ったかなど基本的な権利)を契約などによって動かすことができないということです。著作財産権にあたるコピー権やレンタル権などは契約で移行できますが、著作権にはどうしても移行できない部分があるということです。

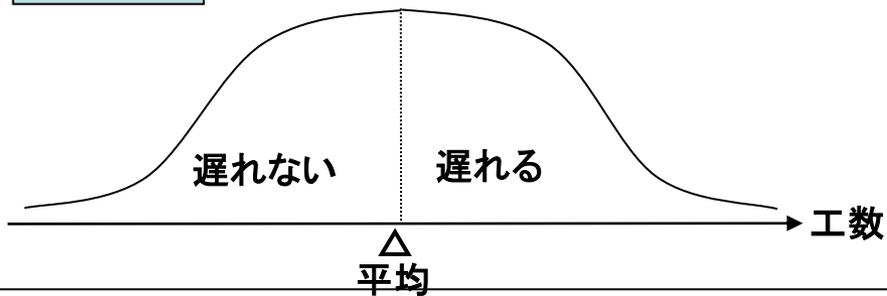
セオリー36のようにプログラムを部品として再利用することを考えた時、顧客との契約ばかりが気になりますが(他の顧客に使っていいかなど。顧客との共同開発でない限り法的には問題なく、契約だけの問題)、もっと大切なことはこの開発をベンダーが他社に外注した時です。

多くのベンダーが従来のように下請へどんどんシステム開発を発注しなくなったのは、ここにも理由があります。部品の著作権を「完全に」自社のものとするにはどうしても自分たちで開発し、自分たちが著作者になる以外ないといえます。著作者でなくとも(プログラムを外注していても)、著作物(プログラム)を自由に使う権利は契約で得られますが、著作者にはなれないということです。開発パワーが足りない時はむしろ派遣社員を使うことの方が安全といえます。見方を変えれば、他社に営業を代行してもらい、下請でシステムを開発するのは難しくなり、自分で自分の仕事を見つける、つまりソリューションビジネスへのシフトが求められるわけです。

# THEORY38

遅れないようにするのではなく  
遅れたときのダメージを小さくする

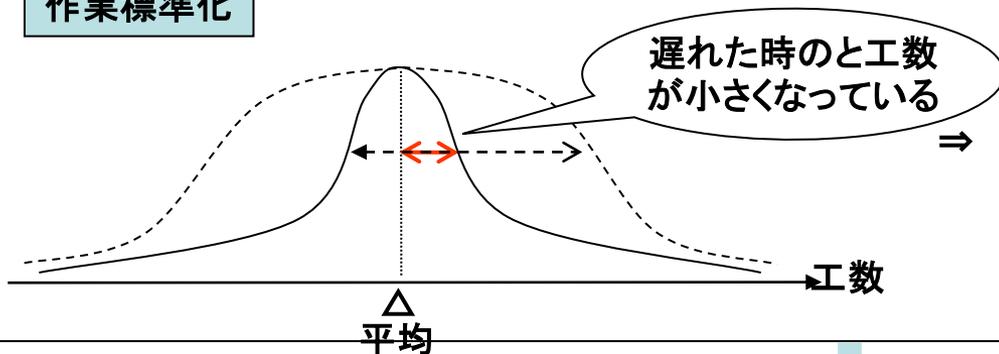
### 工数見積



平均で見積ると

⇒  
遅れる確率50%  
遅れない確率50%

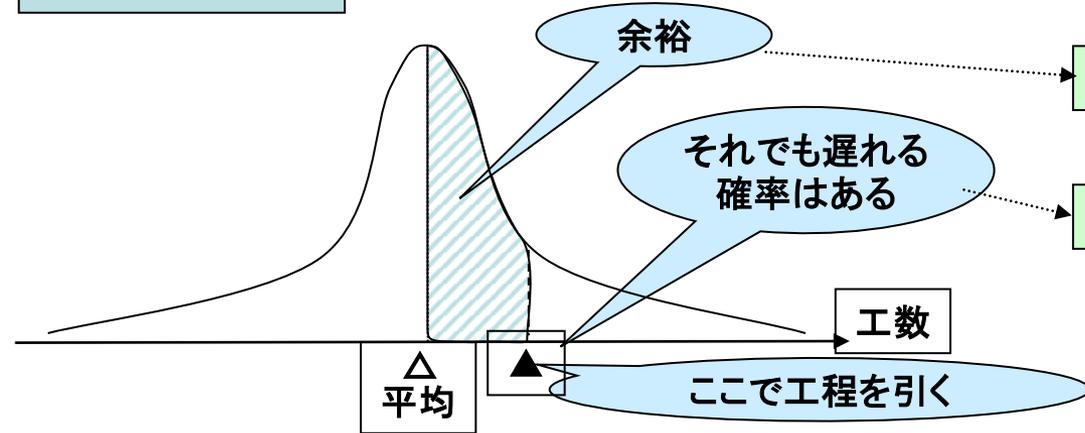
### 作業標準化



⇒  
・平均は変わっていない  
⇒生産性を上げるのではない  
・遅れなくなっているのではない  
⇒遅れる確率は50%

⇓  
遅れてもダメージが小さい

### 遅れた時を考える



⇒ コストアップとのトレードオフ

⇒ バッファを考慮しておく

セオリー1で述べたようにプロジェクト管理でもっとも難しいのは工程管理であり、もっとも恐いのは「工程遅延」です。プロジェクトリーダーを一度でも経験したことのある人であればわかると思います。この工程遅延をプロジェクトリーダーとして経験すると、その後遺症はなかなか消えません。また遅れるのではと思い、工数見積を上げ、企業の価格競争力を落とし、そして遅れることが怖くて、踏み込んだ提案(リスクではあるが、ユーザーにとっては魅力的な提案。ソリューションベンダーから見れば他社に勝ち抜く提案)ができなくなります。SEが新しい仕事をやりたくない、新しい手法を使いたくないと考える理由もすべてここにあります。

プロジェクトリーダーの工程管理におけるセオリーは3つです。

## ①工数見積に平均値を使えば50%の確率で遅れる

工数見積はセオリー35のように数学的に考えれば確率分布(「工数」という母集団があり、そこから特定の標本値が抽出されるということ。数学が苦手な人はソリューションイメージのようなグラフだけをイメージして下さい)となります。同じ仕事を同じようにやっても、場合によってかかる工数が異なり、一定の確率でそれが決まると考えます(というよりも考えるべきです)。そしてよくわからない事象については、コンサルティングイメージにあるような正規分布(もっともデタラメに事象が発生すると考える。つまりどうなるかもっともわからない時に使う)という左右対称の形になると考えます(考えましょう)。

工数見積は平均値ですので、山の頂上になります。これは2回に1回は遅れること(平均値より工数が大きくなること)があるということを意味します。例えば過去、同じ仕事を8人月、9人月、11人月、12人月でやっていけば、平均値は10人月と計算されます。もし工程表を10人月をベースとして引けば2回に1回(11人月、12人月の時)は工程遅延になります。それでは「怖い」というので12人月で見積もっても、今度のプロジェクトでは過去の最高よりもかかることも考えられるので、これでも「遅れない」とはいえません。つまり遅れないプロジェクトなどないということです。SEはこれを理解するだけで心が落ち着きます。

プロジェクトリーダーはこのことをよく顧客に理解してもらうことです。ソリューションビジネスでは「幸いなことに」顧客であるユーザーとソリューションベンダーが共同で仕事を行ないますのでわかってくれるはずですが(ユーザー側の仕事だって遅れることもあるはずですが)。「このプロジェクトは遅れるか？」というシビアな(つまらない?)質問には「遅れることもあります」と答え、「遅れないようにできないか？」という質問にははっきりと「できません」と答えることです。そしてその意味をしっかりとユーザーに理解してもらうことです。

## ②作業を標準化する

こう考えていくと「遅れないようにする」(そもそも遅れるようにと思って仕事をする人がいませんし、いつも遅れている人とはチームを組まないようにすることです)ことよりも「遅れた時のダメージを小さくする」ことが大切だとわかつて思います。これに最も効果的な方法は作業の標準化です。プロジェクト内の作業の標準化は、工程管理から見ると生産性の向上(決して作業が速くなるわけではない)でも、遅れる確率を落とす(相変わらず50%)ものでもなく、遅れた場合の遅れる工数を小さくするものなのです、標準化つまり質の同じやり方で仕事をやるということは、工数バラツキ(グラフでいえば平均からの距離、難しくいえば標準偏差)を小さくして、遅れたときのダメージ(何日遅れるか)を小さくするものなのです。

### ③遅れたときのことを考える

ダメなプロジェクトリーダーほど「遅れた」時のことを考えずにプロジェクト管理を行います。遅れないようにチェックし、遅れないことを祈り、遅れるとがっかりします

プロジェクトリーダーはまずプロジェクトの立ち上がり時点で、そのプロジェクトが遅れることを想定し(できれば遅れる確率を考え)、遅れた場合はどうするかを考えておきます。ガントチャート(バーで引いていく工程表)、PERT(仕事の前後関係をはっきりさせたもの)、マイルストーン(工程の進捗を測るものさしを決めること)といった工程管理の3種の神器も「遅れない」ようにするツールではなく、「遅れたこと」を早期に発見し、どの位遅れ、どの仕事に影響を与えるかを分析するツールです。

こう考えると対応は2つしかありません。

#### ・工数余裕

そもそも工程表に工数の余裕を見ておくことです。先ほどのように平均10人月の時に12人月で見積もっておくことです。この2人月分は遅れても表面化しません。しかしこのやり方の致命的な欠陥は12人月で見積もると、10人月で終了する可能性を消してしまうことです。やってみればわかりますが、システム開発プロジェクトは予定より早く終わることはまずありません(早く終わってもテストなどをしていきます)。この余裕はシステム開発の生産性を落とし、そして、見積原価を上げ、ソリューションビジネスの競争力を低下させてしまいます。

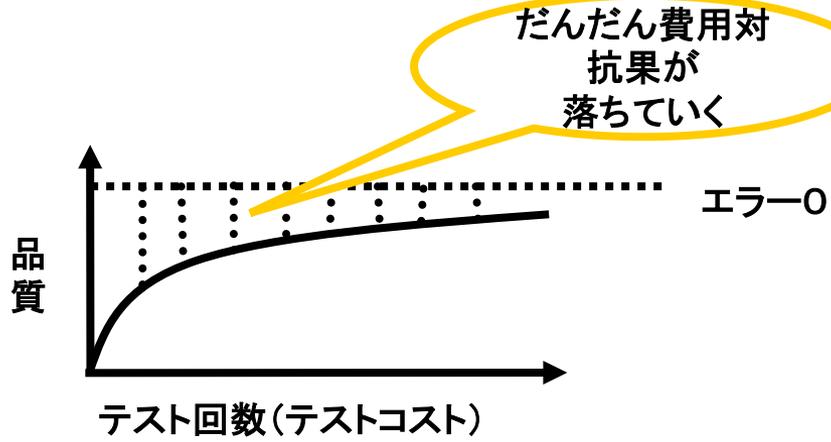
## ・バッファ

遅れた場合のバッファとなるパワーを考えておくことです。バッファとしてまず考えられるのは残業、緊急外注ですが、これは企業としての体力を落とす(モチベーションの低下、生産性の低下、品質の低下など)ことにつながります。使うべきバッファは、プロジェクトリーダー、サブリーダーというマネジメントを行なっている人たちのパワーです。マネジメントの基本は例外の原則(リーダーは定常業務以外の仕事を行う)です。工程遅延のリカバリーはまさに例外の仕事であり、リーダーの仕事といえます。その意味でリーダーは万能プレイヤー(プロジェクト内の仕事はほとんどできる)である必要があります。これに残業、緊急外注を組み合わせるのが基本でしょう。

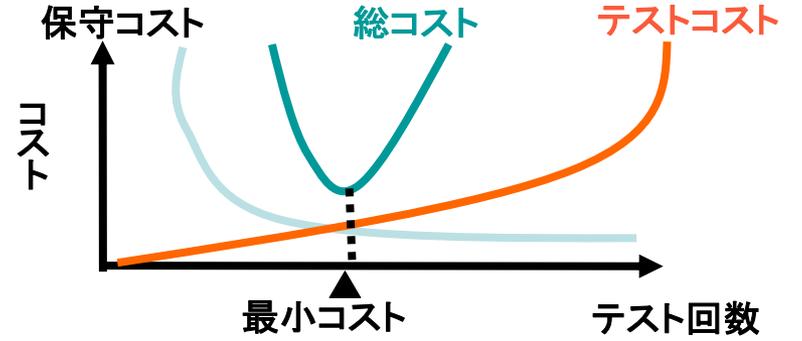
# THEORY39

エラーのない  
システムはない

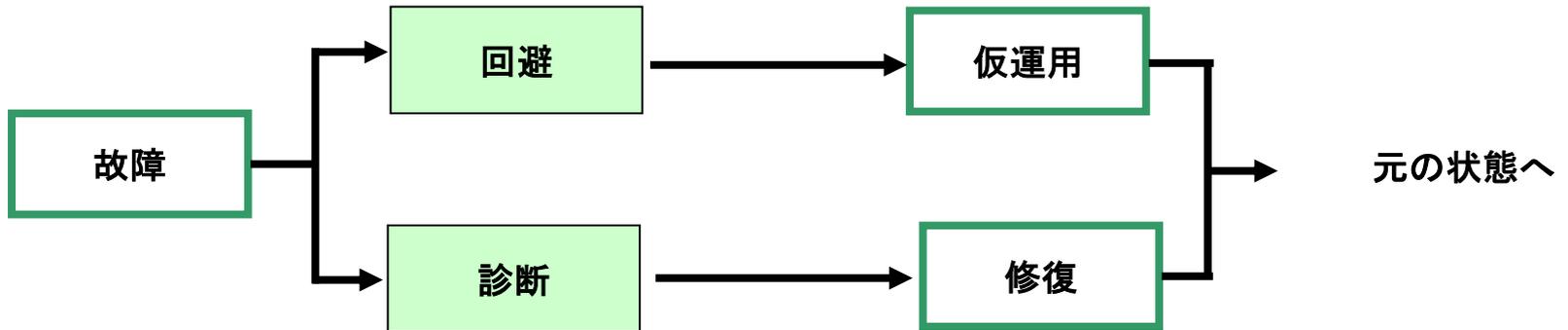
### テストコストと品質



### テストコストと保守コスト



### 保守のフロー



プロジェクトの品質管理でポイントとなるのはシステム開発の後に行われるテストです。システムテストについては顧客であるユーザーに以下の4点を理解してもらう必要があります。

### ①エラーは消えない

セオリー22で述べた「システムにエラーがあることは証明できても、エラーのないことは証明できない」というダイクストラの言葉を思い出して下さい。どんなシステムにもエラーはあり、普段使っているときにはそれが現れない(その意味で潜在エラーといいます)で、何かの拍子に現れて故障したり、あるいは最後まで現れなかったりするということです。

そう考えればシステムテストの役割はエラーをゼロにすることではなく、「潜在的なエラーをテストで見つけて除去し、より潜在的エラーの少ないシステムにする」ということになります。ユーザーの「エラーがすべてなくなってからシステムを納品して欲しい」という要望には、SEは残念ながら「ノー」と答えるしかありません。

## ②どこまでやるか

エラーが消えないとすれば、システムテストの課題はどこで終わりにするかになります。テストにかかるコストと品質の関係は、ソリューションイメージの左上図のような曲線であると考えられます。品質はエラー0の線に向かって無限に近づいていく(品質向上していく)ことにはなりますが、その直線と交わることはありません。つまりエラーは0になりません。このグラフを見ればわかる通り、テストの費用的効果(品質の向上/テストの費用)は時間とともに、テストをやるほど、だんだん落ちていきます。セオリー23で述べた基幹系システム(高い信頼性が要求される)に膨大なコストがかかるのはこのテストコストのためといえます。

このテストをやめる時期は、テストをやりながら、システムの状態を見ながら考えていったり、時間が来たらやめる(実はこれが今までのシステム開発ビジネスでは一番多いといえます。しかしこれではいくら何でも品質が安定しません)というわけにはいかないと思います。品質を安定させるためにソリューションベンダーとしては「システムの潜在エラーを予測し、それが何%まで除去できたらやめる」と考えるしかありません。予測方法には、バグ累積曲線(バグとはエラーのこと、見つけたエラーを指数曲線と考えてグラフ化して全体のエラー量を予測する)、回帰分析(過去のシステムの平均値を出す)、埋め込みバグ(人為的にエラーを入れておいて、それが一定量見つかったらやめる)などが考えられています。

### ③残ったエラーをどうするか

ここまで考えるとプロジェクトにおける品質管理では、テストよりもプロジェクトが終了し、システムが運用段階に入った段階での品質管理の方が大切なことがわかります。くどいようですが、ユーザーにはエラーが顕在化する可能性はある、つまり「故障する可能性があること」を理解してもらいます。もちろんどんなエラー、故障が起きるかはわかっていません(わかっていればその修正をしています)。そう考えれば自動車や家電などと同じように、故障したときどうするかを前もって考えておく必要があります。これをシステムの保守といいます。

テストをしっかりとやって、エラーをより多くとればテストコストがかかりますが、故障の可能性が落ちて保守コストは下がります。一方テストコストを落とすと保守コストは上がっていきます。概念的にはその2つのコストの和はソリューションイメージの右上図のようなグラフと考えられ、最小値を持つことになります。ユーザーにはこの計算式ではなく、この考え方を理解してもらうことです。テストはもちろん一生懸命やりますが、どこかでやめ、後は故障した時にコストをかけるというのが基本です。

#### ④2つの保守

システムに故障が起きると、2つの作業を同時にスタートさせます。

##### ・回避

故障が起きた時、その修理をしなくても、何とかシステムを止めずに動かしていくことを考えます。これがデータのバックアップであり、ハードウェアの予備機を準備しておくというものです。この回避のコストは一般に固定費(故障が起きても起きなくても発生する)であり、故障が起きないと無駄なコストとなります。

##### ・診断

システムの故障は「システムが使えなくなってしまう」という形で見つかり、「どこが故障したか、ソフトかハードか・・・」は素人にはなかなかわかりません。逆にそれがわかればすぐに修理できるといえます。この「どこが故障したか」を知ること、つまり故障診断は難しい仕事であり、プロの手を借りる必要があります。ソリューションビジネスのポイントはこの故障診断をどうするかにあるといっても過言ではありません。ソフト、ハードを含めたシステムの故障診断をできるエンジニアを作ることが必要であり、このサービスをユーザに理解してもらうことです。具体的には遠隔確認(一般にヘルプデスクという)と現場診断(行って調べる)の組み合わせになります。

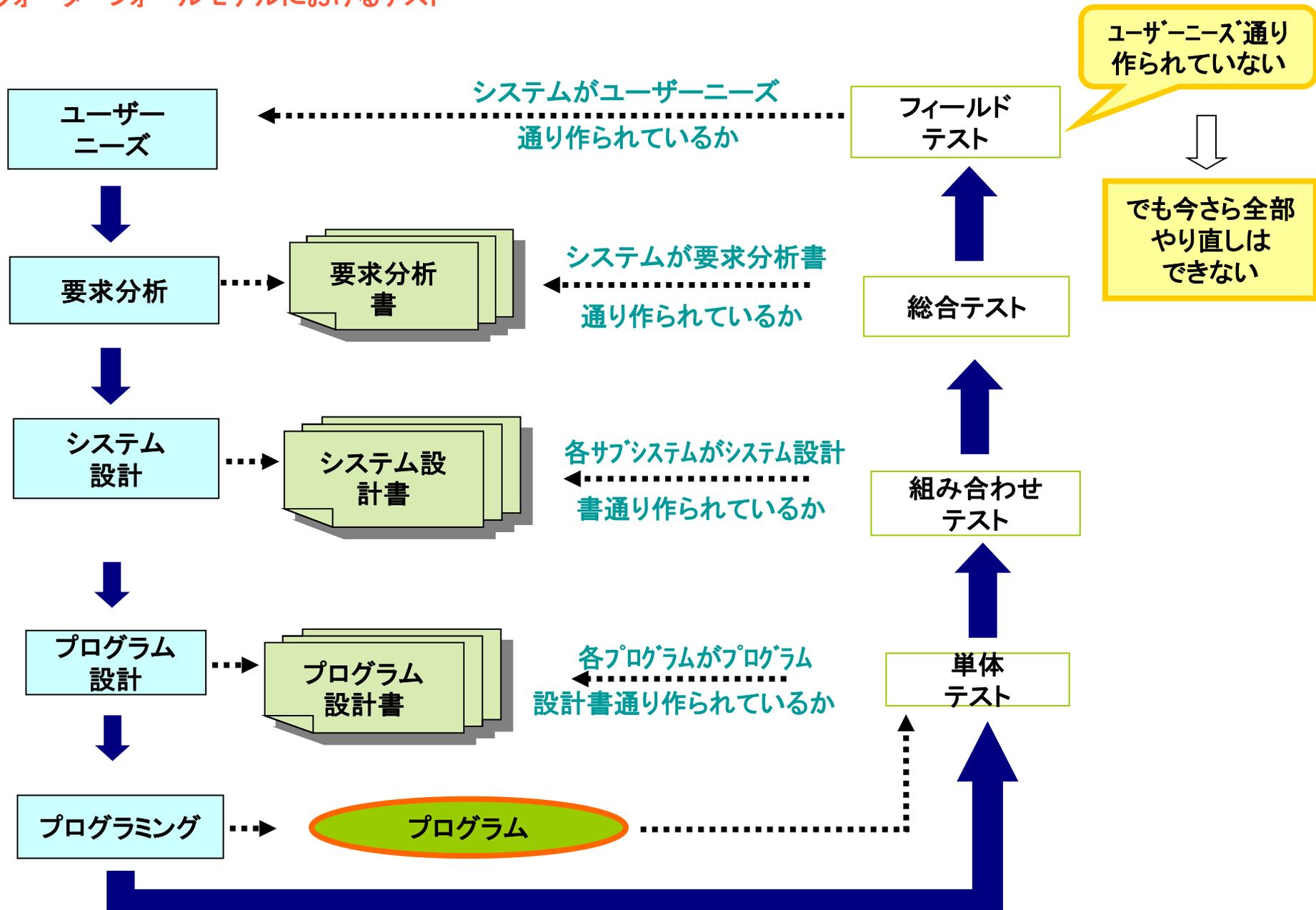
診断のためにかかるコストはベンダー側から見ると固定費(ヘルプデスク要員、保守員)であり、ユーザー側から見ると変動費(故障して修理した分だけ)にしたいものです。これについては他の製品保守サービス(機械、自動車など)同様に、定期点検(月1回程度システムをチェック。これで固定費と理解してもらう)とオンコール対応(電話でトラブルに対応)をセットで月あたり、年あたりの固定費にするのが普通といえます。

保守ビジネスでは提供するシステムの品質が上がれば、要員を減らすことができ、コストダウンとなります。つまり品質が上がればコストが下がるというわかりやすい形となります。これで品質向上はプロジェクトリーダーから見るとコストアップという考えを払拭することができます。

# THEORY40

プロトタイプで  
スペックを凍結する

# ウォーターフォールモデルにおけるテスト



システムテストの実施上の問題点は、「致命的なエラーほど最後に見つかる」ということです。これはどんな開発手法をとっても同じです。一番わかりやすいセオリー8で説明したプロセス指向アプローチにおけるウォーターフォールモデルで説明しましょう。

ウォーターフォールモデルでは、要求分析⇒システム設計⇒プログラム設計⇒プログラミングと作業が進められていきます。このプログラミングが終わるとシステムテストに入っていきます。システムテストはこの流れを逆行する形で進めていきます。まずはじめに行うものを単体テストといい、プログラム1つ1つ(これを単体という)がプログラム設計書通りに作られているかをチェックし、異なるものは直します。次にプログラム同士を組み合わせサブシステムとして使い、システム設計書通り作られているかをチェックし、直します(これを組み合わせテストという)。次にシステム全体を使い、要求分析書通り作られているかをチェックし、直します(これを総合テストといいます)。最後にユーザーにこのシステムを実際に使ってもらってチェックし、直します(これをフィールドテストといいます)。

こう進めていくとユーザーニーズ通りに作られているかは最後に検証され、もし違っていたら、要求分析以降すべてやり直しとなります。しかしこの頃になると、システムの運用開始の時期が迫っており、「今さらそんなことがわかって…」となります。見つかったらリカバリーが大変なエラーは最後に見つかるのです。セオリー8で述べたデータ指向アプローチでもこれは同じです。どんな開発手法をとってもマクロからミクロへと設計・開発をトップダウンで進めていく限り、テストはミクロからマクロへというボトムアップとなり、トップのエラーは最後に見つかります。

この問題点を解決しようと思って考え出されたのがプロトタイピングという手法です。これは開発ステップの途中で一旦スペックを確定し、そのスペックをプロトタイプ(試作品)にしてユーザーに見せて、誤りがないのを見てもらってから次の工程へと進むものです。多くのものはインターフェースの設計が終わった段階で、ユーザーにそれを取りあえず見てもらい、直す所は直してからデータベース、ネットワークなどの設計に入っていくというものです。

プロトタイピングにおいて大切なことは「テストではない」ということです。テストは後でももちろんやります。プロトタイピングをやればその分どうしてもコストがかかることになります。したがって「最後にわかったらとんでもないことになる」ものについてのみ絞り込んでやることです。SEはどうしてもテストと混乱して、「あれも、これも」とやり、オーバーヘッド(プロトタイプを作成したことによって余分に増える作業)が増えて、開発生産性を落とし、原価を上げ、利益を圧迫するとともに、納期遅延の原因となり、長期的には見積原価を上げてしまい、ソリューションビジネスの価格競争力を落としています。

プロトタイピングにはもう1つ意味があります。それはスペックを確定してからシステムを使うまでの間にタイムラグがあり、この間にユーザーニーズが変化してしまうことです。近年これだけ経営環境、ネットワーク環境が変わっていれば顕著になっています。しかしこれではソリューションビジネスが成り立ちません。

この解決方法は2つです。1つは早く作ること、つまりタイムラグを小さくすることです。そのためには極力オーダーメイドをやめ、既存のパッケージ(出来合いのソフトウェア)、部品を使うことであり、ユーザーに多少使い勝手が悪くてもその理解を求めることです。もう1つはこのニーズ変更によるスペック変更がシステムに大きな影響をもたらし、コストが大きく発生することをユーザーに理解してもらうことです。そしてそのためにプロトタイプによる検証を行います。ユーザーには「このインターフェースが後で変わると大変であり、コストが発生する」ということを体で理解してもらうためにプロトタイプ検証を行います。そのためにもプロトタイプは「あれも、これも」ではなく、「これだけはダメ」というものにすべきといえます。

## 第7章 ソリューションを実施する仕組みづくり

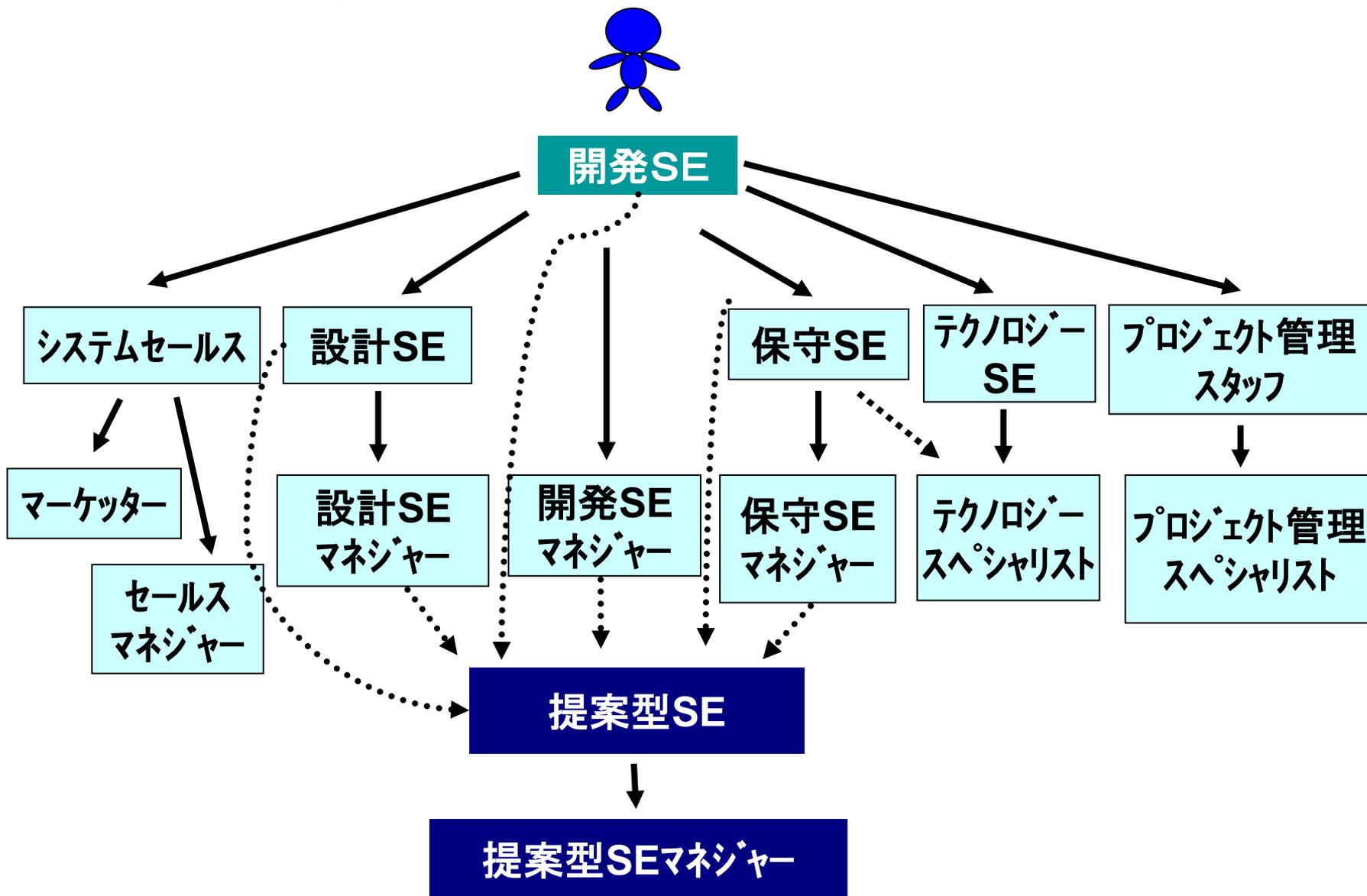
### トータルセオリー

ソリューションビジネスを成功させるポイントは提案型SEの位置づけ。提案型SEを最終像にキャリアステップを作り、早く提案型SEを育て、提案型SEをプロジェクトリーダーとして顧客の前面に出していく。

# THEORY41

**SEのキャリアステップを  
明確にする**

# キャリアステップの例



従来システム開発ビジネスではプログラマー⇒システムエンジニア⇒システムアナリストといったキャリアステップが考えられ、これを実践してきた企業も数多くありました。プログラマーの経験を積んでシステムエンジニアに、そしてシステムエンジニアのエリートがシステムアナリストに……。

しかし実際にやってみると気が付くのですが、「プログラミングが上手な人が良いSEになるのか？」「SEとシステムアナリストの仕事は直接は関係ないのでは？」「これでは皆がゼネラリストになってしまう。」「データベースの専門家も必要なのでは」といった疑問がわいてきます。このシステム開発ビジネスに携わる人を対象とした認定試験である情報処理技術者試験(セオリー43で述べる)もいつの間にか、13種類の試験をやるようになっていました。

ソリューションビジネスにおいては組織を作る前に、まず社員のキャリアステップを作るべきです。多くのソリューションベンダーはこれをうまく作れず悩んでいます。筆者がSEに「あなたの職業はSEですか？」と聞くと多くの方は「まあいってみればSEかな」と答えます(システムセールスの人は「セールスです」と答えます)。きっと設計・開発・保守さらには運用などさまざまな仕事をしているうちに「自分のやっている仕事」と「自分が描くSE像」の間にギャップが生まれているのでしょう。これではプロとしてのプライドを持つことはできません。優秀な人ほどプライドを求めて他社へ移ってしまいます。プライドが持てなければ、後は給料くらいしか考えることがありません。少し給料の条件が良いとやはり他社へ移ってしまいます。

ソリューションベンダーは自社独自のキャリアステップを作り、社員そしてこれから入社する未来の社員に提示すべきです。このキャリアステップはその企業のビジネスモデルであり、その企業が求める像であり、目指してほしい姿です。ソリューションベンダーは他社との人材獲得競争において、給与などの労働待遇や企業ブランドではなく、このキャリアステップを競うべきです。

ソリューションイメージには典型的なキャリアステップを挙げています。キャリアステップ作成では自社のビジネスの構造をはっきりさせるべきです。このイメージでは引合⇒提案⇒設計⇒開発⇒保守というシステムライフサイクルを、すべて自社で行うソリューションベンダーを想定しています。

## ①開発SE

設計以降の工程である開発・テストを担当するSEです。この工程は生産性を上げることが比較的容易であり、そのためソリューションビジネスにおいてもっとも利益を生むフェーズといえます。従来から設計を仮にユーザー側で行ったとしても、この開発部分だけはプロのプログラマーが担当してきました。利益面ではソリューションビジネスの中核部分ともいえます。

いわゆる事務職(経理など)を除くライン系のすべての従業員が、入社してすぐは開発SEをまず担当すべきといえます。小売業に就職した人がまず店員(ここで利益を生む)を必ず担当し、それから店長、バイヤー、マーチャンダイザー、ディストリビューターと職種分化していくのと同じです。もしこの部分をすべて外注している企業でも、外注先に出向させてでも一度は担当すべき職務です。開発SEを最低1年担当した後、次のキャリアに進みます。

また、このうちの一部の人はこのまま開発SEにとどまり、開発SEのマネジャーへと昇格していきます。開発SEに向いている(続けた方がよい)タイプは以下のような人です。

- ・まじめ
- ・持久力がある
- ・時間を守る
- ・チームワークを保つ

## ②システムセールス

マーケット分析、ニーズ掘り起こし(デモンストレーション、セミナーなどの開催)、見込み客の発掘、受注条件の折衝、顧客の契約窓口・相談窓口などを担当する職種です。セールスは販売職として、SEとは別に採用することもあるかと思いますが、それでも半年でも良いから開発SEを一旦は担当させます。そのうえで適性を見てセールスへ職種を変えます。システムセールスに向いているのは次のようなタイプです。

- ・人と接するのが好き
- ・性格が明るい
- ・システムよりも顧客に興味がある

システムセールスはさらにリーダータイプとスペシャリストタイプに分け、前者にはセールスマネジャー、後者にはマーケター(会社全体のマーケティング戦略を考える)という2つのキャリアステップを準備します。

### ③設計SE

従来からいわれる、いわゆるSEであり、システムの設計を担当します。これに向いているのは次のようなタイプです。

- ・論理的である
- ・情報をまとめるのがうまい
- ・正確に仕事を行う

設計SEは無論設計SEのマネジャーへと昇格していきます。

### ④保守SE

システムの運用後、何かトラブルが起こった時リカバリーしたり、修正したりして顧客の運用窓口となるSEです。中小企業では開発SEと同一キャリアにしても良いと思います。これに向いているのは次のようなタイプです。

- ・システムを作るより動かすことが好き
- ・機械が好き
- ・人と接するのが苦手ではない

保守SEも基本的には保守SEのマネジャーになっていきます。その中でリーダーになりたいか、「腕の良い保守SE」はテクノロジースペシャリストへの道も用意します。

## ⑤テクノロジーSE

データベース、ネットワーク、プラットフォームなど特化した分野において技術ウォッチングしていく職種です。これに向いているのは次のタイプです。

- ・1人で仕事をするのが好き
- ・1つのことにのめり込んでいく
- ・(あたりまえのことですが)その技術が好き

テクノロジーSEはチームを組む必要がなく、したがってリーダーも要りません。そのためマネジャーと同格でスペシャリスト(その道のプロと企業が認める)としてのキャリアステップを用意します。

## ⑥プロジェクト管理スタッフ

プロジェクト環境およびツール(開発環境、品質管理、工程管理、見積システムなど)の選定、リーダーへのマネジメント支援などを行うスタッフ部門です。これは向き、不向きというよりは、開発SEをやっていく中で、いつの間にかこういう仕事をチーム内に担当している人が誕生しています。本人もそれがあまり嫌いではないというタイプです。(旅行の幹事などをやるとイキイキしている人はよくいます。)小規模の企業ではテクノロジーSEの一部としても良いと思います。テクノロジーSE同様にプロジェクト管理スペシャリストというキャリアステップを用意します。

## ⑦提案型SE

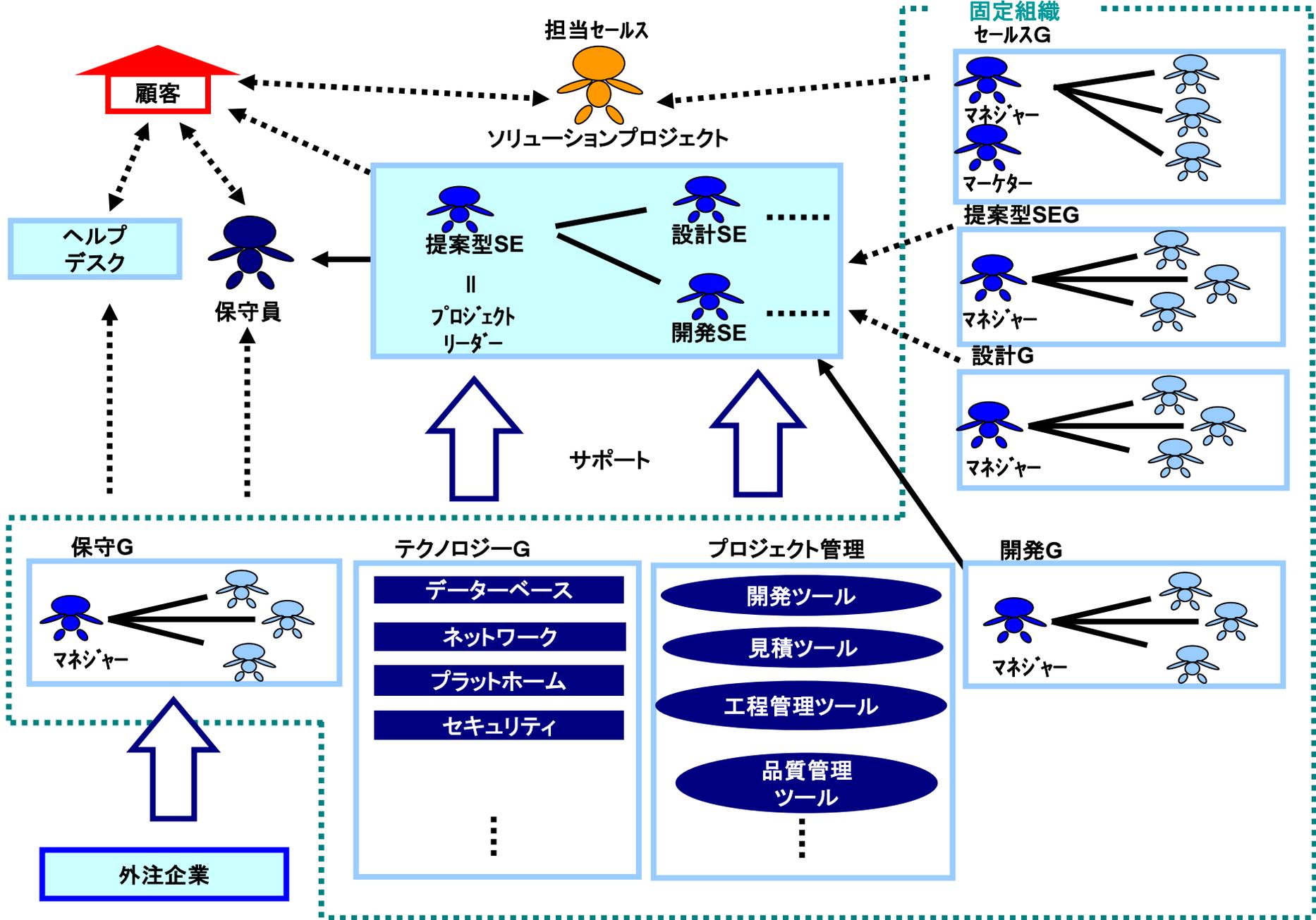
基本的には設計SE・マネジャー、開発SE・マネジャー、保守SE・マネジャーの中から選定します。これについてはセオリー44で述べます。

このキャリアステップで大切なことは各職種のうちどれが「えらいか」というランクはないということをはっきりさせることです。ソリューションはチームプレーです。プロ野球でいえばピッチャーもキャッチャーも外野手も同じプロであり、どれがえらいということはなく、その人の適性で決まり、チームの状況によりコンバート(ポジション変更、つまり職種変更)があります。

# THEORY42

提案型SEを  
プロジェクトリーダーにする

ソリューションイメージ



ソリューションビジネスにおいてはキャリアステップにもとづいて固定的な組織を作ります。セオリー41のようにキャリアステップを決めれば、セールスグループ、提案型SEグループ、設計グループ、開発グループ、保守グループ、テクノロジーグループ、プロジェクト管理グループという形になります。各グループにはマネジャーがいて人事責任、損益責任を負います。マネジャーは部長・課長という階層性を持ってOKです。テクノロジー、プロジェクト管理グループは、フラットな組織にし、グループ長(プレイングマネジャーが一般的)が人事権だけを持つようにします。仕事のイメージは次の通りです。

### ①マーケティング

セールスマネジャーのマーケターが市場分析などをして、どのようなマーケットをどういう形で攻めるかを決定します。例えば「病院市場へ製薬メーカーとジョイントしてソリューションビジネスを展開する」といったことです。

### ②セールス

当該マーケットの担当セールスが決められ、セミナー、デモンストレーション、個別訪問などにより、見込客の開拓を行います。セールスクロージングは提案型SEに委託します。セールスの目標・予算は受注金額であり、これをセールスグループのマネジャーが管理します。

### ③ソリューションプロジェクト

見込客が固まると、セールスグループのマネジャーが提案型SEグループのマネジャーに対して、セールスクローリング、つまり提案の依頼を行います。提案型SEグループのマネジャーは担当する提案型SEを指名し(場合によっては複数)、彼が提案～受注までを担当します。また提案書を作るうえでテクノロジーグループのサポートを受けます。スペックが固まってきたら、プロジェクト管理グループの見積ツールを利用して工数見積を行い、それにもとづいて、設計グループ、開発グループのマネジャーに人員アサインを要求します。両グループのマネジャーは人員に対する原価提示を行い、提案型SEと交渉します。その上で顧客に提案し、受注が決定すると、このアサインされたメンバーでソリューションプロジェクトが発足します。このプロジェクトのリーダーは提案型SEであり、プロジェクトの損益責任を負います。つまり提案型SEはプロジェクトの損益で評価を受けることとなります。やればすぐ気がつくのですが、プロジェクトの損益は提案書でほぼ決まってしまうことがわかります。

設計グループ、開発グループのマネジャーは先ほど提示した「原価」に対して責任を負い、プロジェクトメンバーが見積原価どおりのパフォーマンスを発揮することに責任を負います。

つまりプロジェクトリーダーである提案型SEは受注窓口として受注金額に対する責任つまり損益を負い、設計・開発グループのマネジャーはそのサブコントラクトとして原価責任を負います。セオリー36で述べたようにプロジェクトによっては、当該プロジェクトは赤字でもやるというケースが生まれてきます。この判断を行うのが提案型SEグループのマネジャーであり、長期的な損益責任を負います。この赤字でも行うプロジェクトでは、提案型SEは目標原価(マネジャーと決める)に対する責任を取ります。

プロジェクトは必要に応じて、テクノロジーグループ、プロジェクト管理グループのサポートを受け、このサービスに対しプロジェクトはサービス使用料(人員の工数やツール使用料に応じて)を払います。テクノロジーグループ、プロジェクト管理グループはコストセンターとなり、経費予算を組み、その範囲内で最大のサービスを行ないます。

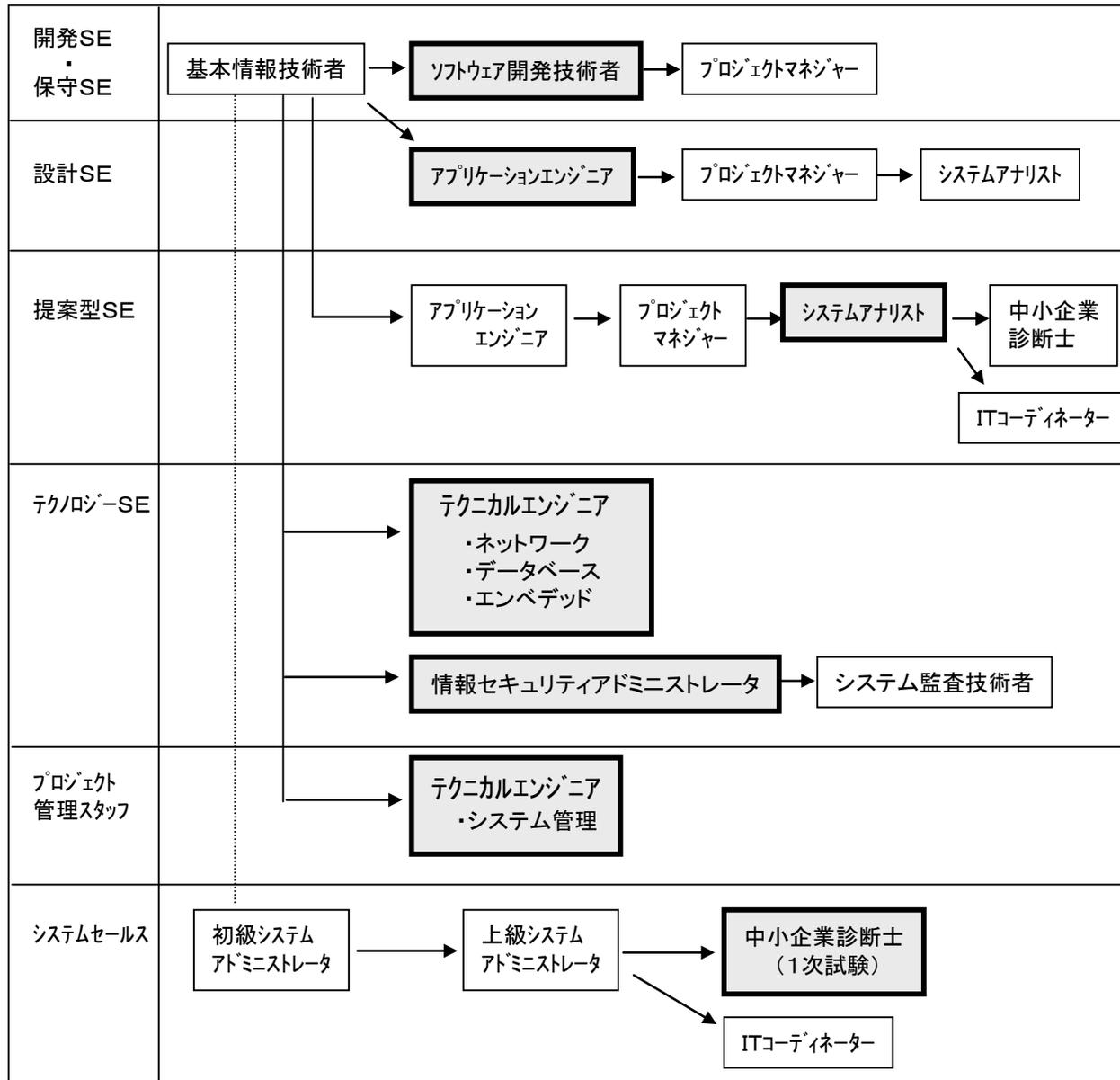
#### ④保守

ソリューションプロジェクトが終了すると、個別保守契約(その顧客を担当する保守員がつく)かヘルプデスク契約(電話対応のみでそれ以外は別途契約)を結びます。個別保守の担当は保守グループのマネジャーが指名し、損益責任はすべてマネジャーが負います。ヘルプデスクは保守グループまたは外注企業との契約により行い、これも保守グループのマネジャーが損益責任を負います。

# THEORY43

資格試験を  
職種認定に使う

(職種ごとの資格ステップイメージ)



キャリアステップは職務を遂行していくグループ、つまり組織化の他にもう1つ意味があります。それはプロとしてのプライドを従業員に持たせることです。プロとして、その職種における仕事は一定のレベルで遂行できることにプライドを持って欲しいのです。プロとしてのレベルに達していることをどうやって確認するかですが、経験年数だけでは「やる気」が起きませんし、人事査定では「不公平」感がつのるといえます。

本来なら各ソリューションベンダーが各職種について職務レベルを決め、それについての認定試験をやりたいところです。しかし個別企業がこのような認定試験を行うと問題作成、評価などにどうしても不公平感が出ます。

そう考えるとやはり国家試験に頼らざるを得ないといえ、経済産業省認定の情報処理技術者試験がその中心的役割を果たすと思います。この試験は当初2種（初級プログラマー）、1種（上級プログラマー）という区分だけでしたが、特種（システムエンジニア）、システム監査技術者試験...とSEの多様化に伴ない追加、修正が繰り返され、現在は13種の区分に分かれています。13種もあって選びやすいというメリットもありますが、職種区分、難易度などがからみ合っ少しわかりづらくなっているのも事実です。多すぎてわかりづらいことで企業、従業員とも混乱してしまって（何のための試験か？）、うまく活用できていない企業がほとんどです。この試験をベースに職種の資格認定（プロとして認めること）を行うのであれば、次のように整理すべきです。

## ①開発SE・保守SE

事務職(ここにも資格認定が必要なら初級システムアドミニストレータが適当)以外はすべて1年間は一且、開発SEに配属されます。そこでこの1年間で「基本情報技術者試験」(従来の2種、プログラマー対象、合格率10~20%、年2回)を合格させるようにします。

開発SEおよび保守SEとしての資格認定はこれではなく、「ソフトウェア開発技術者試験」(従来の1種、上級プログラマー、合格率10~15%)とします。経験年数があつて基本情報技術者に合格していない人は、直接これを受けます。つまりこれをとれば、プロとして認めます。さらにマネジャーを目指す人は「プロジェクトマネジャー試験」(プロジェクトリーダー向け、合格率7~8%)を認定対象とします。

開発環境によっては各ソフトベンダー(マイクロソフト、オラクル...)が行う特定商品の活用レベルを認定する試験なども受けさせます。しかしこれは資格認定ではなく、開発ツールの勉強という位置づけとします。

## ②設計SE

開発SEを1年間担当してきた人は基本情報技術者試験を受けています。設計SEの資格認定は「アプリケーションエンジニア試験」(以前の特種、いわゆるSE向け、合格率5~8%)で行います。さらにマネジャーを目指す人は「プロジェクトマネジャー試験」、「システムアナリスト試験」(システムアナリスト向け、情報処理技術者試験ではトップ、合格率6~8%)を対象とします。

### ③提案型SE

提案型SEになるには「アプリケーションエンジニア試験」、又は「プロジェクトマネジャー試験」の合格を最低ラインとして、資格認定は「システムアナリスト試験」で行います。さらにここでのマネジャーを目指す人は「中小企業診断士試験」(やはり経産省の行う経営コンサルタントの能力認定試験。合格率は1次30～50%、2次10%程度)や「ITコーディネーター」(NPOのITコーディネーター協会が実施している資格認定試験、資格としての位置づけが今ひとつはっきりしないが、そのねらいは提案型SEにあっている)などを目指させ、これによって資格認定というよりも外部コンサルタントなどとの交流を深めさせます。

### ④テクノロジーSE

設計SE同様に基本情報技術者試験は新人の内に受けさせます。ここでの認定は少しテクノロジー分野が粗いのですが、「テクニカルエンジニア試験」(ネットワーク、データベース、システム管理、エンベデッドシステムの4分野にて行っている。合格率6～13%)のネットワーク、データベース、エンベデッドシステム(マイクロプロセッサやLSI技術者向け)、「情報セキュリティアドミニストレータ試験」(セキュリティのプロ向け。合格率13%程度)の中から該当分野を選ばせます。

## ⑤プロジェクト管理スタッフ

「テクニカルエンジニア試験」のシステム管理を認定試験とします。開発ツール担当は先ほどのソフトベンダーの試験をまず受ける「義務」を負います。

## ⑥システムセールス

初級システムアドミニストレータ試験(利用者側の認定資格。合格率30%程度)を必須資格として、上級システムアドミニストレータ(初級の上。利用者側のリーダーをイメージしている。合格率6~8%)、ITコーディネーターなどを目指させます。認定資格は適当なものがないのですが、中小企業診断士試験の1次試験(8科目マークシート式で経営・管理全分野の試験)あたりがよいかもかもしれません(もちろん2次試験までを目指させますが、この2次合格を認定とするのでは、他の試験に比べやや難しすぎるといえます)。

これらの資格はその仕事に携わっていなくても、取得することが可能です。開発SEであってもシステムアナリスト、テクニカルエンジニア、中小企業診断士などほとんどのものが「勉強すれば」とることが可能です。

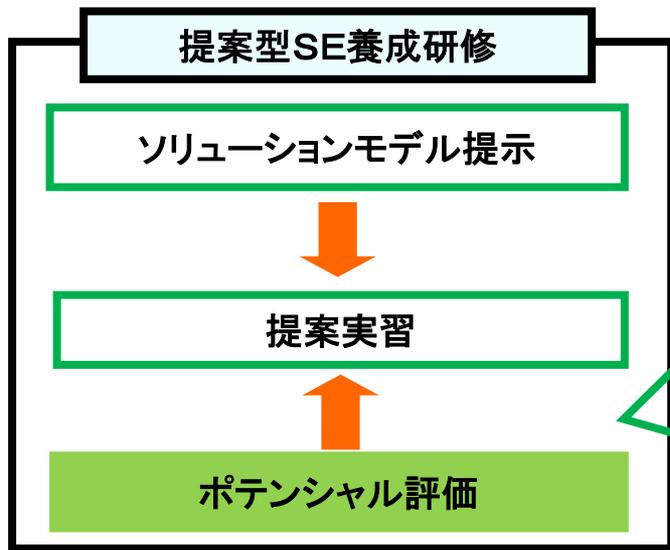
これらの資格試験はその職種におけるプロとしての認定の他に、自分が職種変更(開発SEからテクノロジーSEなど)を望むなら、その認定資格を取得すればその職種になることが可能というルールにしておけばよいと思います。

こう考えていくと、資格取得のための勉強は自己啓発で行い、企業はその支援(勉強ツールの選定、グループ勉強会などの支援...)などに徹するべきといえます。「どの資格をとるべきか」までがソリューションベンダーの企業としての義務であり、その取得は個人の自由といえます。

# THEORY44

提案型SEは  
研修で選ぶ

# 【提案型SEを選ぶ】



個人ポテンシャル評価

ポテンシャル評価

氏名 \_\_\_\_\_

評価項目		評価					コメント
		A	B	C	D	E	
知識	業務知識						
	IT知識						
能力	創造力						
	企画力						
コミュニケーション	インタビュー						
	プレゼンテーション						
マインド	コンサルティング						

総合評価

\_\_\_\_\_

過去の人事評価

ポテンシャル評価

【経営常識の整理】

企業システム研修

セオリー45

【ITトレンドの整理】

IT利用研修

セオリー46

【ナレッジの理解】

ソリューション  
ケーススタディ研修

セオリー47

ソリューションビジネスの組織において、その中核となるのは提案型SEといえます。この提案型SEを誰がやるか、そしてどうやって育てるかが大切なテーマといえます。ソリューションビジネスの立ち上げ時点では、プロの提案型SEが企業に存在していないので、OJT(On the Job Training 仕事をやりながら教育)というわけにはいかず、Off-JTのセミナーに頼るしかありません。そのメインとなるものが提案型SE養成研修です。提案型SE養成研修は次の3つを領域・目的として、標準的には3日間で行います。

### ①ソリューションモデルの提示

提案型SEとしての基本的ノウハウである、その企業としてのソリューションモデルを提示することが一番大きな目的です。「仕事のやり方の提示」は企業の最低限の、そして最大の義務です。やり方を標準化しても、中身は個人の力量で異なります。この「提案型SE養成研修を実施する」と決めれば、「やり方を決める」しかなくなり、自ずと仕事の標準化がされていきます。簡単な例をまじえながら本書の第2章から第6章までのニーズ予想モデル、インタビューモデル、ニーズ発見モデル、ニーズ解決モデル、プレゼンテーションモデル、プロジェクト管理モデルなどを一通り1日半位で学習します。このモデルをマニュアル化した場合は一通り事前学習をさせておきます。

## ②提案実習

残り1日半で提案型SE候補の受講生に、上記モデルを使って実際にソリューションをケースによって実習します。基本的やり方は次のとおりです。

- ・3チーム～4チーム(1チーム5名が限度で、少ないにこしたことはないが、人数が少ないと研修のコストパフォーマンスが落ちる)で提案書による競合を模擬的に行う。

- ・講師(プロのコンサルタント、セミナーインストラクター、企業内の提案型SEなどが考えられる)が提案を受ける対象企業の社長となり、受講生は彼にインタビューし、プレゼンテーションし、各チームが提案書で競う。

- ・プレゼンテーション終了後、受講生同士で提案書をレビューし、最後に講師が各チームの講評をユーザーの立場で行う。

研修で行うケーススタディ対象企業はなるべく、そのソリューションベンダーが経験したことのない業界が良いと思います。受講生の中にその業界で経験のある人がいると、チームがその人に頼ってしまい、実力が発揮できません。また、やってみるとわかるのですが、「やったことのない企業」の方が既成概念がなく「ユニークな」提案、つまり競争力が高い提案がなされ、受講生の真の実力が出るといえます。

またこの実習を行なうことで、ソリューションモデルの問題点などがわかり、必要に応じてモデルを改良することができます。

### ③ポテンシャル評価

この研修のもう1つの目的は「誰を提案型SEにするか」というテーマを解決することです。選定条件は次の2つです。

- ・提案型SEをやってみたい
- ・提案型SEに向いている

「やってみたい」が第1条件であり、提案型SE研修への参加応募を通して、その意思を確認します。提案型SEを経験年数や人事評価によって選抜してから研修するのではなく、研修参加者は公募します。人数を絞る必要があるときは上司または人事部が応募者の中から選定するしかありません。第2の条件は「向いているか」です。この「向いているか」は相対的なものであり、その企業が提案型SEを何人作りたいたいのか、作る必要があるかに依存しています。受講生を相対評価して向いている人の上位から選んでいきます。

「向いているか」の評価は非常に難しいといえます。現在の職務で「提案」という仕事をやってないことが前提ですので、この場合今の仕事(提案以外の別の仕事)だけでその適性を見るのは乱暴です。今の仕事の人事評価(提案という仕事以外のリーダーSEとしての適性度)に加えて、提案という新しい仕事の適性を評価すべきです。そういう意味でポテンシャル評価といえます。

ポテンシャル評価はこの研修で行う提案実習での受講態度(意欲がわかります)、インタビュー、グループディスカッション、プレゼンテーション、レビューというすべての局面で行います。評価者はこの研修の講師を中心とし、場合によっては何名か(提案型SEを現在やっている人、上司、人事部など)がオブザーバーとなります。評価項目はソリューションイメージにあるようなものが一般的であり、各項目の評価基準は以下の通りです。この評価基準がソリューションベンダーが考える提案型SEへの適性といえます。この基準は無論、研修受講者に公開します。

このポテンシャル評価と今までの人事評価と合わせ、提案型SE(正確にいうと候補)を選抜します。そのうえでセオリー45~47にあるような教育を短期間で行ない、プロの提案型SEを養成します。

またこの提案型SE養成研修を外部の専門機関に委託し、ポテンシャル評価を受けると、そのソリューションベンダーのSEの能力がソリューション業界全体の中でどの程度のものかを知ることができます。(少し宣伝になりますが筆者の会社では約2500人のポテンシャル評価のデータベースを持っています)。

	評価項目	評価内容
知識	業務知識	外部スペックを作る力を見る。ユーザの業務内容または業界を知っているかどうかではなく、「情報システム化のセオリー」を知っているかどうかを評価する。
	IT知識	内部スペックを作る力を見る。DB・ネットワーク・開発ツールなどの選定能力、およびこれらのトレンド、コストパフォーマンスを見る目を評価する。
能力	創造力	他社の提案書との差別化を図ろうとするマインドおよびその力を評価する。
	企画力	提案内容をまとめ挙げ、これをユーザーに正確にわかりやすく伝える力を評価する。
	信頼性	提案書のリスクの低さ、つまり、トラブル発生を未然に防ぐ力を評価する。
コミュニケーション	インタビュー	研修時点での課題予想モデルの適応力、およびニーズコミュニケーション力を評価する。
	プレゼンテーション	プレゼンテーションおよびプレゼン時の評価内容（レビュー能力）を評価する。
マインド	コンサルティングマインド	ユーザをコンサルティングしていこうという意欲を評価する。

# THEORY45

**提案型SEは企業を  
システムとして理解する**

## 企業システム研修カリキュラム

### <1日目>

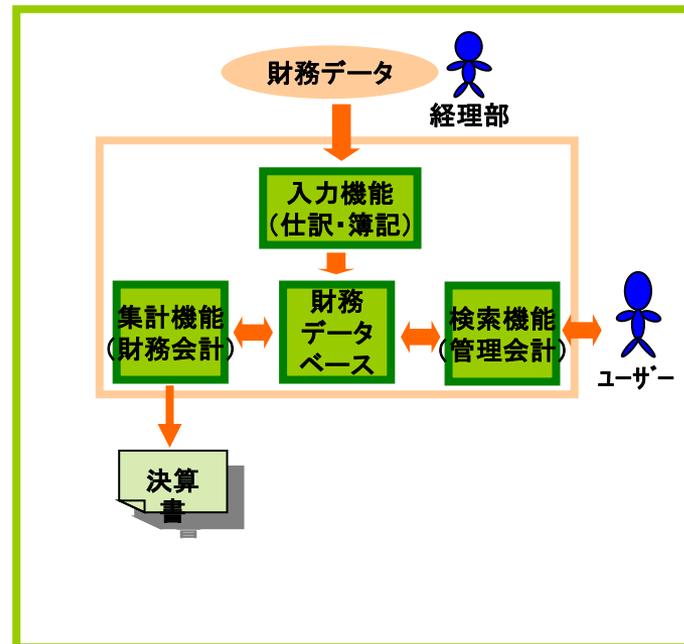
時間		内容
9:00~9:30	①オリエンテーション(L)	<ul style="list-style-type: none"> <li>・ソリューションベンダーはどうあるべきか</li> <li>・知識の欠如は何をもたらすのか</li> <li>・提案型SEに求められる知識</li> </ul>
9:30~11:00	②株式会社をシステムとして見る(L, GD)	<ul style="list-style-type: none"> <li>・株式会社の要素と関係</li> <li>・日本の株式会社の特徴</li> <li>・系列・取引をネットワークで考える</li> <li>・パフォーマンス評価としての利益と株価</li> </ul>
11:00~12:00	③財務会計のアルゴリズム(L)	<ul style="list-style-type: none"> <li>・税法、商法、証券取引法と財務会計</li> <li>・決算書を集計表としてみる</li> <li>・勘定科目は集計キーとして見る</li> <li>・資産、負債、資本、収益、費用は集計区分</li> </ul>
13:00~15:00	④財務会計のデータフロー(L, GD)	<ul style="list-style-type: none"> <li>・仕訳の意味とフロー</li> <li>・利益とは何か、キャッシュフローとの違いは？</li> <li>・減価償却、在庫、連結決算、時価法、…の意味</li> </ul>
15:00~17:00	⑤管理会計と財務データベース(L, GD)	<ul style="list-style-type: none"> <li>・利益計画、キャッシュフロー、計数分析の意味</li> <li>・財務データをデータベース化する</li> </ul>

### <2日目>

9:00~10:30	⑥計数分析のアルゴリズム(L)	<ul style="list-style-type: none"> <li>・収益性、生産性とROI、交差比率</li> <li>・流動性、安全性と在庫、キャッシュフロー</li> </ul>
10:30~12:00	⑦利益計画と予算のデータフロー(L)	<ul style="list-style-type: none"> <li>・CVP分析で損益計算分のグラフを考える</li> <li>・目標利益から予算へ</li> </ul>
13:00~15:00	⑧決算書でデータ分析の基本を知る(GD)	<ul style="list-style-type: none"> <li>・マクロに数値をとらえる</li> <li>・数値の違いから仮説を立て検証する</li> </ul>
15:00~16:30	⑨現代の経営をシステムとして理解する(L, GD)	<ul style="list-style-type: none"> <li>・エクイティファイナンスとデットファイナンス</li> <li>・ワラント、コールオプションと企業システム</li> <li>・企業の統合、分割</li> <li>・コーポレートガバナンスと企業の社会的責任</li> </ul>
16:30~17:00	⑩まとめ(L)	<ul style="list-style-type: none"> <li>・提案型SEのあり方</li> </ul>

L:講義 GD:グループディスカッション

## 企業システムのイメージ



提案型SEの選抜が終了したら、彼らには次に「企業システム研修」を受講させます。ポテンシャル評価をやってみればわかりますが、評価項目のなかの業務知識(特にセオリー6で述べた経営常識)が極めて低いことに気づき、多くのソリューションベンダーの経営者はがく然とします。利益、在庫、キャッシュフローといった基本的な経営用語の意味を理解していないということです。ユーザー側の企業から見れば、この人が自分の経営を左右するシステムを提案するのかと考えると「ゾっ」とします。

この経営常識が低いことが「提案がうまくいかない」もっとも大きな理由といえます。この知識は一朝一夕には身につきませんが、身につけようとする気持ちを持つ「キッカケ」が大切です。自分が知識を身に付けなくてはならない理由とその術を知ればSEは「必ず」自ら学習していきます。

企業システム研修では提案型SEは「何を知るべきか」「なぜそれが必要か」「知らないとどうなるか」を徹底して理解させます。彼らはこの経営常識を新人教育などで「会社のしくみ」といったことですでに学習していることが多いといえます。そしてすべて忘れていきます。使わない知識は忘れるのが当然ですし、使い方さえもわからなくなっています。彼らにもプライドがあり(SEは特にプライドが高い)「何で今さら会社の仕組み」と感じて、学ぶことを「恥」だと思ってしまう。企業システム研修は、この問題をクリアするために新人向け研修のような「会社のしくみ」などといったタイトルは避け、「提案型SEのための企業システム研修」といったタイトルにします。しかし内容は初歩の初歩から、場合によっては新人研修よりも易しく(新人時代よりも頭は硬く、このような知識を学ぶことから遠ざかっている)します。

こうして15～20名くらいの単位で2日間で実施していきます。基本的カリキュラムはソリューションイメージにあるようなものです。

### ①オリエンテーション

実はもっとも大切なところですが、できれば社長などのトップに来てもらって、「当社にとって提案型SEがいかに大事か、どうしてこのような知識が必要か、知識を持ったらどうなる、持たないとどうなる」といったことを話してもらいます。これを外部講師が行う時は、当社を「業界」におきかえて、「この業界でなぜ提案型SEが必要となっているか、ユーザーはSEの知識のなさをどう思っているか、知識がないSEが考えたシステムはどのような結果を迎えているか」を例をまじえて話します。提案型SEになりたい受講者がこれを納得すれば90%は終わったようなものです。自然に知識を求めてきます。

### ②株式会社をシステムとして見る

株式会社をシステム(要素とその関係から成るもの)として考え、システム内の各要素の意味、関係を「システムチックに」説明します。系列、取引といったことはネットワーク理論(ノードとリンク)で、利益、株価はシステムのパフォーマンス評価として理解させます。このフェーズでは簡単なグループディスカッションを行います。テーマとしては「系列は何をもたらしたか」「株価が下がると企業はなぜ困るか」といったことを話し合わせます。

### ③財務会計のアルゴリズム

財務会計は単純なキーブレイク&集計のアルゴリズムであることをわからせませす(場合によってはレコード設計をさせませす)。これを話すと多くのSEは財務会計の意味が「一発で」理解できませす。

### ④財務会計のデータフロー

財務会計においてはアルゴリズムに沿ってどのようにデータが各業務をフローしていくかを説明し、各自にデータフローを作らせませす。ここであわせて仕訳アルゴリズムの矛盾を説明し、減価償却などのイレギュラー処理が必要なことを理解させませす。これにあわせて「減価償却費はなぜ入ってくる金か」などをディスカッションさせ、損益計算書のグラフ化などを通して理解させませす。

### ⑤管理会計と財務データベース

管理会計を分析とキャッシュフローに分けて説明し、前者には未加工のデータベースが、後者にはデータフローとしてとらえることが必要なことを理解させませす。

### ⑥計数分析のアルゴリズム

分析手法のアルゴリズムを紹介させませす。

### ⑦利益計算と予算のデータフロー

利益と売上・費用の関係をCVP分析(Cost-Volume-Profit)という1次関数のグラフでとらえ、損益計算と予算の意味を直感させ、企業にとって販売予測が大切なことを理解させませす。また先ほどの損益計算書のグラフとCVPの2直線の関係を理解させ、グラフ化の本当の意味を肌で感じさせませす。

## ⑧決算書でデータ分析の基本を知る

自社、ライバル、顧客などの決算書を実際に用意し、そこで起きていることから、この企業の経営戦略、マーケティング戦略、組織戦略(財務戦略ではなく)を予測させます。これを通してデータ分析におけるユーザー側の立場を理解します。あわせてこれだけこのデータを多目的に利用するには、未加工のデータベースと、非定型のマイニングツールが必要なことを、「データベース化の提案」というディスカッションを通して理解させます。

## ⑨現代の経営をシステムとして理解する

その研修時点で新聞をにぎわしているキーワードを選び、その意味を講師が解説します。ここまで理解すれば日本経済新聞を読むことができ、またこれがスポーツ観戦などと同じように結構「おもしろい」ことを実感させます。

## ⑩まとめ

今後提案型SEとしてどのように経営系の知識を習得していけばよいかを話します。

この研修が終わると多くのSEは「目からウロコが落ちた」とよくいいます。SEは企業をシステムで理解する力は誰よりも強いのに、そのやり方を誰も話してくれなかったことに気づきます。

# THEORY46

提案型SEはITを  
パターン化しておく

# IT利用研修カリキュラム

## <1日目>

時間		内 容
9:00～9:30	①オリエンテーション(L)	<ul style="list-style-type: none"> <li>・ナレッジマネジメントの必要性</li> <li>・IT活用の意味</li> <li>・IT活用のパターン</li> </ul>
9:30～12:00	②ビジネスプロセス型IT利用の事例(L, GD)	<ul style="list-style-type: none"> <li>・メインフレーム、クローズドOSがもたらしたもの</li> <li>・システム開発のピークがもたらしたもの</li> <li>・BPR、ERP、データセンター、ASPの意味</li> </ul>
13:00～15:00	③メーカーの戦略型ITの事例(L, GD)	<ul style="list-style-type: none"> <li>・IEから生産管理システムへ</li> <li>・FMS、MEの時代</li> <li>・カンバン、CIM、CALS、SCMの時代</li> <li>・製販一体化、カテゴリーマネジメントの時代</li> <li>・ダイレクトマーケティング(メーカーPOS、ECR、QR、SCM、CALS)の時代</li> </ul>
15:00～17:00	④小売業の戦略型IT利用の事例(L, GD)	<ul style="list-style-type: none"> <li>・日本型GMSが作ったPOS&amp;EOS</li> <li>・アメリカ型DSのCVP分析</li> <li>・ハウスカード、RFM、FSP</li> <li>・CVSの店頭在庫削減</li> <li>・SCMによる流通在庫削減</li> </ul>

## <2日目>

9:00～10:30	⑤卸売業の戦略型IT利用の事例(L, GD)	<ul style="list-style-type: none"> <li>・SCM型 物流コスト管理 &amp; 物流支援システムの事例</li> <li>・リテールサポート型 リテールサポートシステムの事例</li> </ul>
10:30～12:00	⑥その他の戦略型IT利用の事例(L, GD)	<ul style="list-style-type: none"> <li>・見せるシステムの事例</li> <li>・困うシステムの事例</li> <li>・リストラ型の事例</li> </ul>
13:00～16:00	⑦ニューテクノロジーがもたらすもの(L)	<ul style="list-style-type: none"> <li>・インターネットの3つの意味</li> <li>・インターネット利用環境の変化</li> <li>・インターネット型IT利用のパターン</li> </ul>
	⑧ニューテクノロジー利用型の事例(L, GD)	<ul style="list-style-type: none"> <li>・BtoC型の事例とポイント 小売型、VC型、卸型</li> <li>・サービス型の事例とポイント</li> <li>・配信型の事例とポイント</li> <li>・プロモーションツール型の事例とポイント</li> <li>・イントラ型の事例とポイント</li> <li>・BtoB型の事例とポイント</li> </ul>
16:00～17:00	⑨まとめ(L)	ソリューションビジネスの構造

L: 講義 GD: グループディスカッション



次は提案型SEのプロとしての知識である「ITトレンド」を習得させます。これがIT利用研修の第1の目的です。

セオリー6で述べたように、提案型SEはソリューションパターンをいくつ持っているかが勝負です。ソリューションベンダーはこれをナレッジとしてデータベース化するのですが、その意味、使い方が理解できないと宝の持ち腐れです。このソリューションパターンを理解し、ケーススタディで身につけていくのがIT利用研修の第2の目的です。

ナレッジマネジメントのためのソリューションデータベースには大きな問題があります。それは「ナレッジを入れた人」はこのナレッジをすでに持っており、データベースに入れても本人にとってはあまり幸せがないことです。皆がこう考えるとデータベース作成初期の頃は誰かが入れるにしても、徐々にデータベースに誰も入れなくなってしまう。IT利用研修の第3の目的はこの研修を通して講師および受講生がシステム事例を調査し、研修カリキュラム作成・実施の一貫としてまとめ、それをデータベース化していくことです。この研修をやるには事例の準備が必要であり、それを講師が準備し、受講生がディスカッションすることでナレッジデータベースとなっていくということです。

さらに研修の主旨を社内に徹底させることで、このデータベースを運営・管理していくのは、彼らの上司である提案型SEのマネジャーの仕事であることを理解してもらいます。これが第4の目的です。

研修カリキュラムはソリューションイメージのようになります。

## ①オリエンテーション

ここでのポイントは、これから提案型SEになる人にとって、いかにこのソリューションパターンのデータベースが必要か、そしてこれがこの企業の体力となることを理解させます。

## ②ビジネスプロセス型IT利用の事例

ここから先はどのようにやっても良いのですが、IT利用の事例をいくつかのパターンに分類していくことが大切です。この分類キーがソリューションデータベースの検索キーの1つとなります。例では、ビジネスプロセス型IT利用(ITをビジネスの一貫として、インフラとして使っているパターン。主に基幹系システムがこれにあたる)、戦略型IT利用(ITを戦略の1つとして使っていくパターン。主に情報系システムがこれにあたる)、ニューテクノロジー利用型(インターネット、ブロードバンドなどのテクノロジートレンドを利用して、他社との差別化をしていくパターン。主にネットワーク系システムがこれにあたる)になっています。この他、プログラム型IT利用、データベース型IT利用、ネットワーク型IT利用といった形でも構いません。

ビジネスプロセス型IT利用では、主にIT自身のトレンド(ハード、ソフト、OSなど)を述べてから、それが現在どのように利用され、どこに問題があるかを成功例、失敗例を使って理解させます。そのうえで、「ITに関しユーザーが抱えている課題は何か」つまり「どこにビジネスチャンスがあるか」を考えさせ、BPR、ERP、データセンター、ASPといったトレンドの意味(言葉の意味ではなく、なぜそれがトレンドなのか)を理解させます。

### ③メーカーの戦略型IT利用の事例～⑥その他の戦略型IT利用の事例

戦略型IT利用ではとりあえず業種別に分類すると良いと思います。そのうえで、その業種に今までのパターンの提案をするのではなく、異業種にそのアイデアを提供できないかを研修で考えます。セブンイレブンのシステムはトヨタ自動車のカンバンシステムがヒントであり、ヤマト運輸の貨物追跡システムはPOSシステムがヒントになっています。例えば新たなマーケットとして病院を攻めるのであれば、自社が得意としてきた製造業の在庫管理システムや小売業のPOSシステム・カードシステムの事例を活用すればよいということを理解させます。

研修準備としては、企業内システムについては提案書、システム設計書などドキュメントを通して調査します。他社のシステムは雑誌、テレビなどから収集し、業種別に分類しておきます。

研修では目的・使い方をキーとしてこれらをパターン化することをテーマとして、グループごとにディスカッションし、発表します。このグループディスカッションを通して、分類されたシステムの目的・ねらいと機能の関係、実際どのような形で使われているか、異業種に提案できるかを考えさせます。発表はパワーポイントなどで行ない、それをソリューションデータベースとして登録していきます。

## ⑦ニューテクノロジーがもたらすもの

インターネット、次世代携帯電話、ブロードバンドなどニューテクノロジーを講師が整理・分析し、このトレンドを技術ではなく、製品・ビジネスとしてとらえ、当社にどんなビジネスチャンスがあるかを講義します。

## ⑧ニューテクノロジー利用型の事例とポイント

研修準備として講師がニューテクノロジーを利用した事例をパターン化し、その例を調べ、受講生に紹介します。その上でグループディスカッションでそのアイデアが自社の顧客に活用できないかを話し合います。例えばBtoC型(インターネットを使って消費者に販売する)、サービス型(教育、予約などを新しいスタイルでサービスする)、配信型(情報配信ビジネス)、プロモーションツール型(インターネットなどをプロモーションツールとして使う)、イントラ型(Web技術などを使った新しいスタイルの企業コミュニケーションシステム)、BtoB型(企業間取引の新しいスタイル)などといったパターンです。

## ⑨まとめ

最後に自社が利用できるITパターンと業種・業態との関係を整理し、ソリューションビジネスをどういうスタイルでやっていくかについて、グループディスカッションし、さらに個人レポートをまとめさせます。この最後のパターン分析がセオリー9のニーズ予想モデルにおけるニーズチェックリストの元となっていきます。

# THEORY47

ケーススタディで  
戦略を立案する

# ソリューションケーススタディ研修カリキュラム

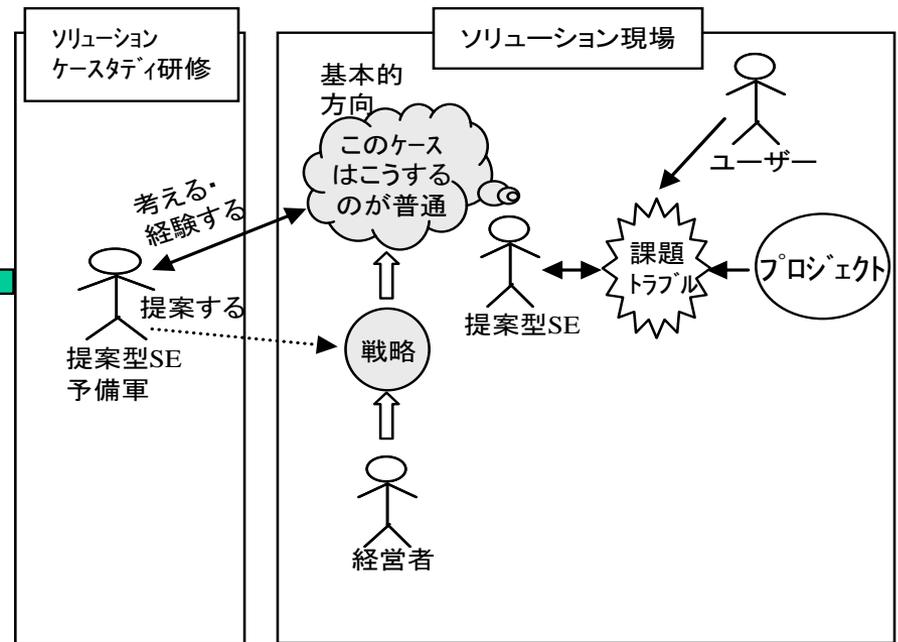
## <1日目>

時間		内容
9:00~10:00	①オリエンテーション(L)	・今、マーケットで起きていること ・ユーザーが望んでいること ・他のソリューションベンダーの考えていること
10:00~12:00	②SWOT分析(GD)	・経営環境をとらえ、当社の強み、弱みを考える ・CSFを抽出する ・CSFへのアプローチを考える
13:00~15:00	③提案・受注に関する ケーススタディ(L, GD)	・提案内容について考える ・受注条件について考える ・価格競合について考える
15:00~16:30	④ビジネスモデルに関する ケーススタディ(L, GD)	・アウトソーシング・アライアンスについて考える ・ビジネスモデルのあり方・リスクについて考える
16:30~ フリー解散	④見積に関する ディスカッション(L, GD)	・見積モデルに関する解説 ・モデル適用可否について考える ・開発手法が見積システムに与える利益を考える ・見積手法と利益の関係について考える

## <2日目>

9:00~10:00	続き	・ディスカッション結果の発表
10:00~12:00	⑤SIに関する ケーススタディ(L, GD)	・SIのあり方について考える ・SIにおけるリターン・リスクについて考える ・SIリーダーか、サブコントラクターかを考える
13:00~15:00	⑥プロジェクト管理ツールに 関するディスカッション (L, GD)	・管理ツールに関する解説 ・ツール適用可否について考える ・ツールと組織について考える ・緊急対策の打ち方を考える
15:00~16:30	⑦システムトラブルに関する ケーススタディ(L, GD)	・トラブルの原因について考える ・トラブルの防止策について考える
16:30~18:00	⑧当社のソリューション ビジネスについて考える (L, GD)	・当社のソリューションモデルの問題点 ・当社のプロジェクト管理・組織の問題点 ・今後のあるべき姿

L: 講義 GD: グループディスカッション



提案型SE養成の一連のセミナーはこのソリューションケーススタディ研修で終わります。これはそのソリューションベンダーで起きうるケースを用意し、その局面で、自身ならどうするかを考え、グループでディスカッションし、その結果をグループごとに発表し、企業としての方向を導き出すというものです。ケースの内容は実際にその企業で起きたものでは受講者に関係者がいたり、利害関係があって、逆にディスカッションがやや歪んでしまいます。その企業で起きそうでまだ起きていないケースや、起きたケースを少し状況設定を変えたほうが良いと思います。ケース作成は外部委託し、ケース作成のプロがさまざまなパターンを用意すべきといえます。また2回目以降の研修では各ケーススタディの前回までのディスカッション結果も講師が紹介し、それらも研修中に共有していくと良いと思います。これはいわゆるMBA(経営学修士)方式であり、考えるタイプのプロフェッショナルを作るための常とう手段です。

この研修の目的は以下の3点です。

## ①ソリューション現場での基本的方向を考えておく

ソリューションビジネスはシステム開発と異なり、判断に迷うケースが多く、それがプロジェクトリーダーである提案型SEに迫られることが多いといえます。この研修でその判断の基本的方向(一般的にこれを戦略という)を、起きる前に、冷静に話し合っておき、場合によっては経営者の了解を得ておくことで、判断ミスをなくすようにします。

## ②戦略を自ら作る

この判断の集大成が、ソリューションビジネスのための戦略そのものとなります。提案型SEに戦略を自分たちで作っていくという意識が生まれ、企業へのロイヤリティ、経営参加意欲が高まるとともに、戦略と現場リーダーの考えが一体化します。

### ③ 擬似体験

提案型SEにとって経験は最重要ファクターであり、このケーススタディで数多く擬似体験し、経験不足を補います。

研修ではソリューションイメージのようにケースをソリューションプロセスごとに分類し、その単位ごとに進めていきます。基本的には提案局面(プロジェクト発足前)とプロジェクト管理局面(プロジェクト発足後)に分けます。提案局面のケースでは、システム開発の時のようにコントロールされたチーム内の仕事ではなく、ソリューションビジネスではユーザーという顧客を中心に、ライバル、ITトレンドなどを意識して、戦略・対応を考える必要があるということを教えます。プロジェクト管理局面ではトラブルを中心にディスカッションしますが、これらがすべて提案段階に課題のあることを意識させます。つまり結論は「提案の良し悪しでプロジェクトの成否が決まる」という方向にもって行きます(講師が意識しなくても普通はこうなります)。

4チーム20人くらいで2日間で次のようなカリキュラムを進めていきます。

### ①オリエンテーション

やはりこのソリューションベンダーのトップが上記3点の目的を説明します。この研修を外部委託するときは、講師が顧客の立場から見たソリューションビジネス、他のソリューションベンダーの考えていることなどを解説します。

### ②SWOT分析

グループディスカッションで自社についてのSWOT分析を進めていきます。SWOT分析とは自社の強み(Strength)、弱み(Weakness)、及び経営環境におけるビジネスの機会(Opportunity)、脅威(Threat)を整理していくというもので、経営戦略立案の基本の基本です。ここで挙げたものについてプライオリティづけし、最重要項目をCSF(Critical Strategic Factor: 主要戦略的要因)とします。さらにこのCSFについて当社としてどのようにアプローチをしていくかをグループで話し合わせ、結果を発表させ、全員で自社としての課題およびその解決の方向を共有します。これが以降のディスカッションの基本的方向(戦略)となります。

### ③提案・受注に関するケーススタディ

提案・受注プロセスの進め方、考え方に関するもので、主なケースパターンは次のようなものです。

- ・自社のユーザーが他社のリプレイス攻勢にあったケース
- ・逆に他社のユーザーをリプレイスできそうなケース
- ・新規物件で他社がダンピングしてきたケース
- ・情報システム部とエンドユーザーの意見が分かれたケース
- ・スペックが全く見えないで、提案書・見積書を求められたケース

### ④ビジネスモデルに関するケーススタディ

顧客から新しいスタイルのビジネス、ビジネスモデルを求められた場合の対応などを考えるというものです。主なケースパターンは次のようなものです。

- ・ソリューション、システム開発ではなく、顧客よりインターネットビジネスなどの共同出資会社の設立を迫られたケース
- ・顧客のビジネスモデル自身の提案を求められてきたケース
- ・ハイリスクなビジネスモデルについて意見・提案を求められたケース

## ⑤見積に関するディスカッション

ここでは、まず講師が自社の見積モデルと他社の見積モデルなどを提示します。そのうえで以下のようなテーマについてディスカッションします。

- ・ソリューションビジネスにおいてはどのような見積モデルが必要か
- ・見積原価と販売価格を当社では誰がどうやって決めていくべきか
- ・ERPパッケージ、ASP、部品流用型開発環境が見積モデルに与える影響

## ⑥SI(システムインテグレーション)に関するケーススタディ

他のベンダーと共同で提案またはビジネスを行なっていくケースについて考えるもので、主なパターンは次のようなものです。

- ・未経験のビッグプロジェクトの提案で、SIリーダー(元請)となるかサブコントラクト(下請)になるかを考えるケース。
- ・SIリーダーになった時のプロジェクトリーダーの機能に関するケース
- ・サブコントラクトになった企業で起こる問題点に関するケース

## ⑦プロジェクト管理ツールに関するディスカッション

プロジェクト管理ツールの選定に関し、自社と他社を比較し、話し合います。ここにはプロジェクト管理スタッフも参加してもらおうと良いと思います。これは意見が分かれ、結論は顧客のニーズによっていくつかの手法が必要ということになると思います。そして受講者の意見は設計・開発グループはこのツール単位に組織化し、提案型SEは提案時にこのグループを自由に選べるようにするというように収束すると思います。

## ⑧システムトラブルに関するケーススタディ

プロジェクトにおけるシステムトラブルに関するケースを用意し、緊急対策のとり方、原因と防止策などを話し合います。主なパターンは次のようなものです。

- ・提案書の内容が不備でスペックを全面的に見直す必要に迫られたケース
- ・システムテストと品質に関するケース
- ・顧客のために工程遅延になったケース
- ・トラブル続出で大赤字になったケース
- ・残業が日常化し、退職者が続出したケース

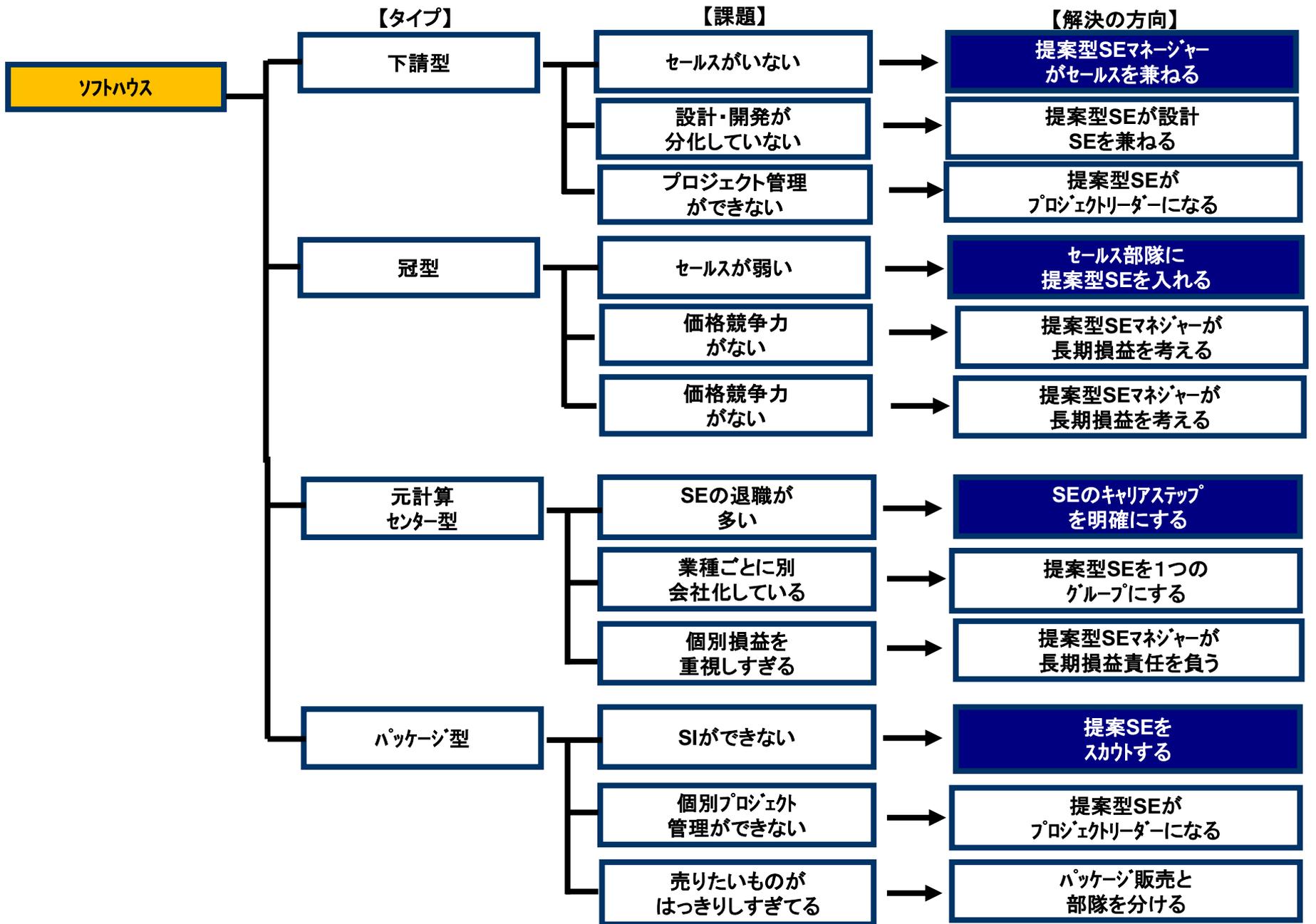
## ⑧当社のソリューションビジネスについて考える

当社のソリューションモデル、プロジェクト管理、組織の問題点を考え、研修前のSWOTと合わせて、今度どのようにこのソリューションビジネスを進めていくか(つまり戦略)を考えます。

さらに受講生にこのテーマ(戦略)について個人別に卒業レポートを作成させ、経営者へ提出させます。場合によっては一同に集合し、経営者にプレゼンテーションをさせます。この卒業レポートをもって提案型SEは終了し、現場へ配置されます。

# THEORY48

ソフトカバーでは足りない職種を  
提案型SEでカバーする



セオリー48～50では、ソリューションベンダーのタイプ別に、組織化の問題点とその解決の方向を考えていきます。自社のタイプについてチェックするだけでなく、ライバルである他のタイプのベンダーの問題点、解決の方向を理解し、戦略の参考として下さい。本セオリーではソフトハウス(ソフトウェア開発が本業)について見ていきます。

ソフトハウスに共通して言えることは、セールスが弱く、当然のことですが、開発パワーが強みということです。従来はシステム開発の仕事はいくらでもあり、「仕事を取りにいく」という気持が弱かったからです。注文が安定しており、危険を冒す必要がないため、リスクの高いニューテクノロジーの導入などは遅く、開発生産性向上の努力に欠け、新開発ツールの導入などもされてこなかったといえます。これらが影響し、価格競争力(そもそも価格競争しようという気持がない)が極端に弱いといえます。

このタイプでは、ソリューションベンダーへの変身の際に、セールス部隊をまず作ろうとします。しかし指名されたセールス(元SE)は何から着手していいかわからず、結局は契約窓口となってしまいます。まずセオリー44にあるように、全SEに提案型SE研修を実施し、SEの10%程度を思い切って提案型SEに指名し(当面は現在の仕事を持ちながら)、そのグループを作り、マネジャーを指名します。彼らにセオリー45～47の研修を実施し、形だけでもいいですから、提案型SEとして働くという気持ちを持たせます。

以降はソフトハウスのタイプ別に見ていきます。

## ①下請型

今までコンピュータメーカーなどの元請システムインテグレーターから安定的な注文を受けていたソフトハウスです。企業数でいえばソフトハウスの中で最も多いタイプといえます。元請が営業してくれますので、セールスは不要です。元請は開発のみを外注しますので、多くの場合設計機能も持っていませんし、プロジェクト管理もやったことがありません。ここから脱皮しようと思えば、まず自分の力で注文を取るしかありません。それにはマーケット分析→ニーズ掘り起こし→個別セールス→クロージングと地道にやっていくしかありません。しかしそのためのマーケティング・セールスの部隊を作るのは困難です。これは今まですべての従業員が売上・利益を生んできた人、つまりSEですので、売上・利益を生まない間接員のセールスを作ると、すぐに業績が悪化してしまうからです。今まで売上・利益をあまり生んでいない、つまり成績の悪いSEをセールスにすると、全くといっていいほど注文はとれません。ここではまず前述のように現業を持ちながら(売上・利益を生みながら)、提案型SEという名前のみを変えた人を作り、この提案型SEのグループのリーダーであるマネジャーが1人でマーケティング・セールスを担当します。そしてある程度見込み客が見えてきたら、提案型SEの担当を決め提案、すなわちクロージングに入ります。こうして徐々にソリューションビジネスへと変身していきます。今までなかったり、不足していた設計機能も提案型SEが担当します。設計手法は今オブジェクト指向型に大きく変わろうとしているので、設計経験の無さはそれほど心配することはありません。(他社も皆経験がないので、そういう意味でこの変身は急ぐ必要があります)

受注後のプロジェクト管理はもちろん提案型SEにやらせるしかありません。この経験不足も従来元請とともにやってきたプロジェクトメンバーとしての経験に加えて、プロジェクト管理の概念を学び、管理ツールをそろえれば心配いりません。そのうえで、徐々に設計グループ、セールスグループ、テクノロジーグループ、プロジェクト管理グループ、保守グループを作っていきます。

## ②冠型

大企業の情報システム部がスピンアウトしてできたソフトハウスであり、社名にその元の企業の名前が入っています。この冠である大企業名の信用力を使って、親企業以外からも受注していくタイプです。金融系では××総合研究所となっている所の多くがこれにあたります。(このタイプは「研究」よりも、「開発」が受注のかなりの部分を占めているものが多い)

このタイプの第1の問題点はセールス力が弱いことです。多くの場合、セールス部隊を持っていますが、何をしたいかがわからなくなっています。ここではまず、下請型同様に提案型SEを指名・育成します。そのうえでセールス部隊にこの提案型SEを入れ、彼から以前の「黙っていても来る親企業の仕事」を取ってしまいます。このタイプは基本的には企業体力がありますので、こういう思い切ったことができるはずですが。この時提案型SEは自分で仕事を見つけるしかなくなり、また優秀で賢い人を指名すれば企業名についての「冠」という信用力で必ず受注を取ってきます。

2つ目の問題点は価格競争力がないことです。今まで開発にかかった分だけもらってきましたので、製造原価しか興味がなく、競争、販売価格という概念さえありません。これは提案型SEのマネジャー(セールスマネジャーを兼務します)に長期的損益を考えさせ、「勝負するときは勝負する」という気持を持たせます。そしてそれを経営がバックアップします。

3つ目はどうしても親企業やその業種に注文が偏りがちとなり、他業種のシステムアイデアが入ってこないことです。一般にこのタイプは親企業に対応する事業部を中心に、それ以外の顧客についても業種別に事業部を作り、各事業部にセールスを置くという形をとっています。このセールスを1つにし、そこに入れる提案型SEも1つになり、彼らに業種特化という気持を捨てさせます。その上で、提案型SEのマネジャーがマーケット分析をしていきます。

### ③元計算センター型

メインフレームが高かった時代に共同利用型のコンピュータセンターとして誕生し、ネットワークの普及でVAN会社になり、バブル期にシステム開発需要の到来でいつの間にかソフトハウスになっていったものです。このタイプは①の下請型と違い、セールス・設計の弱さ、プロジェクト管理といった問題はあまり抱えていません。

最大の問題は、「優秀なSEがやめていく」という企業風土が出来ていることです。SEがやめる理由の多くは「プロジェクト責任やシステムの稼働責任を負うのに、そのつらさが評価されない」ということです。これはできるだけ早い時期にセオリー41のようなキャリアステップをSEに明示し、特に提案型SE(実はこのタイプの企業には従来からこういう仕事をする人がいて、そう呼ばれていなかった)がこの会社をリードしていくということを全従業員に知らせ、提案型SEにプライドを持たせることです。

またこのタイプは大企業が多く、実質的には業種ごとに別会社化しており、その悩みは②の冠型より大きいといえます。これは②と同様に提案型SEを「ソリューションセンター」などと称して、1つのグループにしてしまうことです。

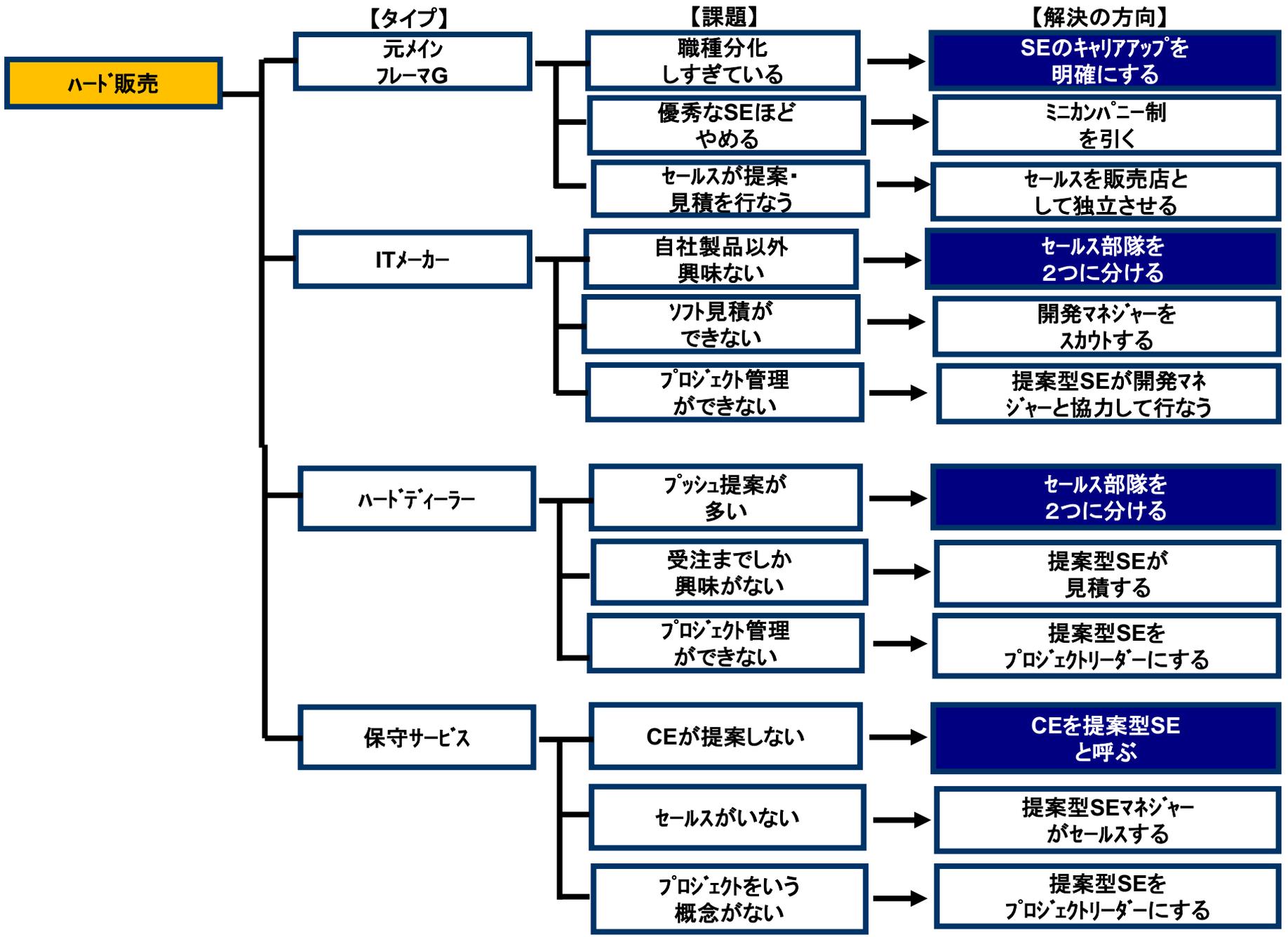
このタイプは計算センター時代から稼働責任を負っていたため、ハイリスクなビジネス構造であり、個別プロジェクトの損益責任が明確となっています。しかし逆にこれがマイナスの経験となり、思い切った提案ができないともいえます。これも②と同様に提案型SEのマネジャーが長期損益責任を負うことで解消していきます。

#### ④パッケージ型

パッケージソフトの開発・販売を行なってきた企業が業積を安定化させたい、あるいは高付加価値化をめざすためにソリューションビジネスを目指す例が増えています。このタイプの最大の問題点は顧客と1対1でシステムを開発し、システムを取りまとめたことがないということです。これは選抜・教育だけではどうしようもありません。提案型SE兼プロジェクトリーダーを外部よりスカウトし、彼を中核として部隊を作っていくしかありません。そのうえでパッケージ開発・販売とソリューションビジネスを分離して(できれば別会社化し)、組織、損益計算、キャリアステップをすべて別のものにする必要があります。こうしてソリューション部隊はパッケージを売るのではなく、ソリューションが仕事だということを社内に告知します。

# THEORY49

**ハード販売企業では  
提案型SEに提案させる**



ハードウェアを販売してきた企業はハードウェア製品の低価格化、ライバルとの競争激化、ネットワーク化による標準化によって他社と差別化できないという厳しい経営環境を迎えています。そのためソリューションビジネスへ進出する(「変身する」という表現が好まれるようです)という企業が増えています。販売業からサービス業への変身なのですが、それを従業員が理解できず(場合によっては経営者も理解できず)、提案などの高付加価値サービスを従来のセールスプロモーションやハードウェアの付加サービスとしか考えていません。これがこのタイプの第1の問題点です。

このタイプではセールス部隊が確立しており、企業内で大きな力、場合によっては全権を持っていることもあります。そのためSEがセールスから頼まれた仕事をやるという意識(ひどいSEは「やってあげる」と言う気持ち)を持っています。これがこのタイプの第2の問題点です。

このタイプでは従来セールスが提案を行っており、これを提案型SEに移管することが上の2つの問題点を同時に解消します。提案型SEを企業を中心に設け、ハードウェアを売るのではなく、彼のやる仕事が企業のコアコンピタンスだということを企業内に知らしめます。そして提案型SEには自ら顧客を訪ね、プレゼンテーションすることで主体性と責任感を持たせます。

以下にタイプ別に見ていきます。

## ①元メインフレームグループ

従来のコンピュータ、システム開発ビジネスを支配してきたメインフレーム(メインフレームと呼ばれる大型コンピュータを製造・販売しているメーカー)を中核とする企業グループです。メインフレームを中心に数百社単位でグループを作り、ソフトハウス、販売会社、データセンター、ネットワークベンダー、保守サービス、さらには教育ベンダー、物流会社、部品工場・・・といったコンピュータに関するあらゆるタイプの企業が存在しています。この企業グループは従来はメインフレームの利益によって成り立っていましたが、近年メインフレームが利益を生まなくなり迷走中です。そのため数百社ある関連企業はメインフレームからの自立を求められ、すべてとっていいほどの企業が自社製品・サービスをベースとしてソリューションビジネスを志向しています。そして場合によってはグループ内競争し、混乱し、顧客から見るとどの会社がどういう特徴があり、どういうことを目指しているのかがわからなくなっています。現在はこの交通整理に手一杯という状況です。しかしこの整理を急がないと、次のような問題がSEに発生し、崩壊の危機を迎えているといえます。

これだけのビジネスを抱えているので、職種がかなり細かく分化されており、どの職種がリーダー職種かという意識もなくなっています。そのためソリューションビジネスを引っ張り、責任の重い提案型SE(ITコンサルタント、システムコンサルタントと呼ぶ事も多い)も他の職種と同等に扱われています。その結果、優秀な提案型SEほどやめて、コンサルティングフォームなどへ転職しています。また、それを引き止めようとして能力給・業績給へシフトしていますが、つなぎとめることができません。

これにはセオリー41で述べたようなすっきりとして単純で、提案型SEがリーダーであることがはっきりとわかるキャリアステップを示し、今この企業グループが求めている(不足している)職種が提案型SEであり、腕を試すには良いチャンスであることをSEに告知する必要があります。

このタイプでは提案型SEだけで大変な数になりますので、これらを階層化し、その中にもキャリアステップを作っていく必要があります。単純に提案型SEのマネジャー、ゼネラルマネジャーというステップだけでなく、本当にプロの提案者(つまりコンサルタント)になろうという人には、提案型SE、システムアナリスト、システムコンサルタントなどといった専門職、プロフェッショナルへの道も用意しておきます。

ここではプロジェクトを一步進めて、提案型SEのミニカンパニー(ソリューションセンターなど)を作り、設計、開発は他の事業部に外注し、その事業部も損益を持っていくという形にします。場合によっては各事業部を子会社というよりもフランチャイズ的(親会社がノウハウを子会社にサービスし、対価を得る)に分社化していくことを考えていくべきです。

もう1つの問題はセールスが提案・見積を行うことです。これはセールスを今も各地にある販売代理店的な独立損益体として、先ほどのミニカンパニーが代理店を支援するという形にします。

## ②ITメーカー

パソコン、プリンター、通信機器などのメーカーがソリューションビジネスへ変身していくケースも増えています。プリンターメーカーが、「プリンターを中核として、オフィスソリューションを」というものです。

このタイプの第1の問題点は自社製品(例えばプリンター)以外への興味がなく、自社製品を売るために付加価値サービス(提案、ソフトウェア、ネットワーク、保守・..)をつけているという印象が強いことです。そのため顧客はどうしてもナショナルブランドの中で最も安いメーカーから購入することを選択し、その「おまけ」として各種サービスを求めてきます。売っている側も買っている側もおまけに品質は求めません。これにはセールス部隊を「ハード販売」と「ソリューションビジネス」の2つにはっきりと分けることです。そのうえでソリューション部隊はハードウェアを売るのではなくソリューション方法を考え、実行していくことがミッションだということを意識づけします。この部隊をさらに提案型SEとマーケターに分けます。マーケターは対象市場別の基本的ソリューションスタイルを考え、提案型SEは個別のソリューション、つまりクロージングを担当するようにします。

このタイプはシステム開発という仕事をやってなかったり、重要視してきませんでした。そのためソフトウェア見積やプロジェクト管理ができません。見積は経験者を開発マネジャーとしてスカウトするしかありません。プロジェクト管理はこの開発マネジャーの指導の下で提案型SEがやるようにします。設計・開発SEは、ハードウェアを作ってきた工場などにいる別職種 of 自社従業員から、人事異動して育てていきます。

### ③ハードディーラー

コンピュータなどを仕入れて売るハードディーラーは、②のITメーカーと同じような傾向にあります。セールスが行う提案が「買って欲しい」の一点張りのこと、受注にしか興味がないこと、プロジェクトという概念そのものがないことの3点がこの特徴です。1点目は②同様にセールス部隊を2つに分けます。2点目は提案型SEがプロジェクト全体の見積を行ない、3点目は②同様にマネジャーをスカウトし、提案型SEがプロジェクトリーダーとなります。

#### ④保守サービス

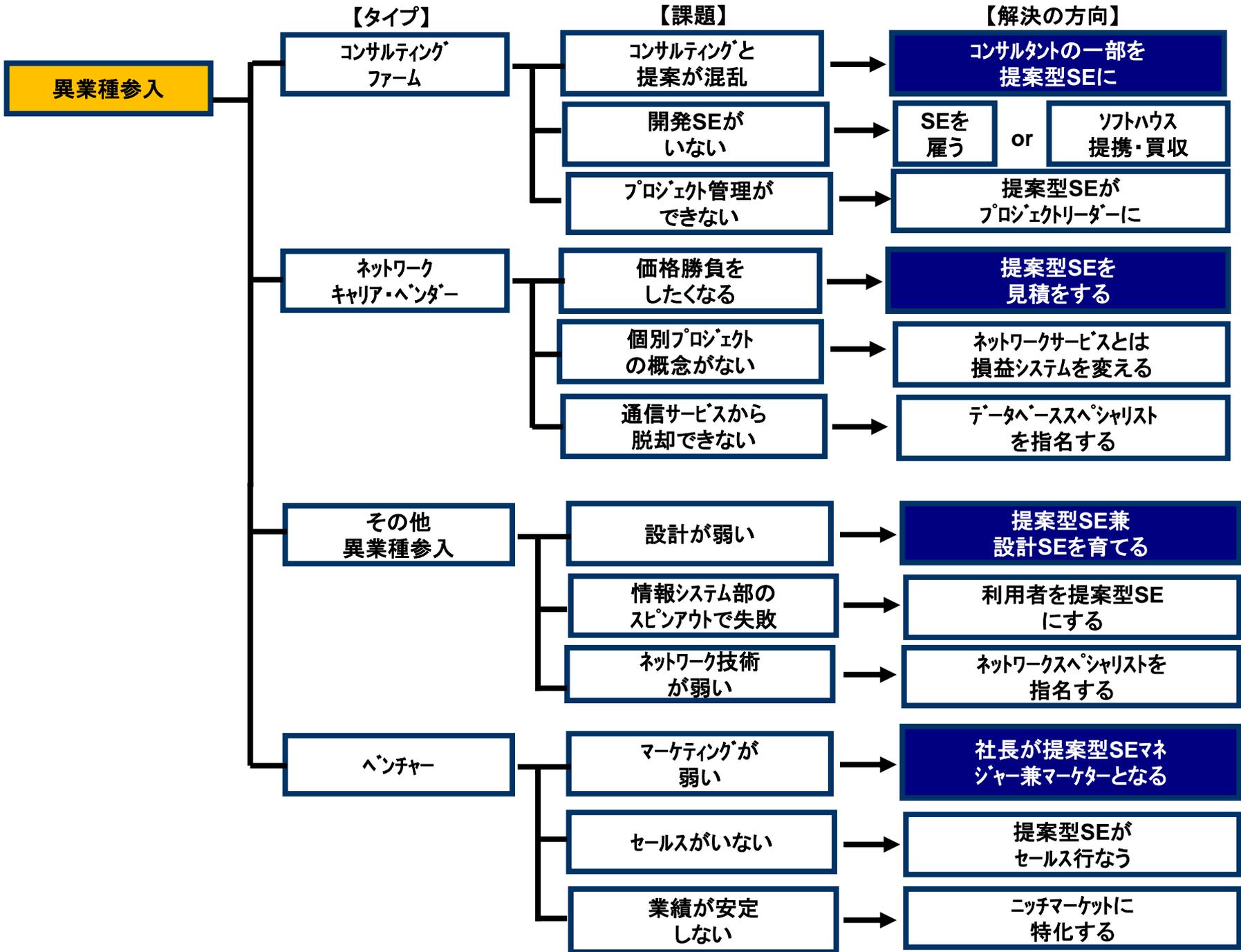
ハードウェアの保守サービスを専門に行なっている企業が注目を集めています。提案→設計→開発→保守というステップが実はフィードバックサイクルであり、最後の保守の次に最初の提案が来ることがわかったからです。つまり保守が提案の上流工程と考えられ、保守が提案の責任を負うべきといえます。

またシステムのライフサイクルが短くなっていく中で、保守をやりながら次のシステムを提案していくことがベストだと考えられるようになっていきます。

つまり保守をやってきた人(カスタマーエンジニア、CEということもあります)が提案型SEに最もフィットしていることになります。保守サービス会社ではCEは効率性のみを要求されているので、顧客に提案する時間まではありません。このタイプではまずCEのうち何名かを提案型SEとして指名し、その人にとっては提案が本当の仕事で、そのアイデアを得るために保守をやっていると意識づけします。これだけでムードがガラッと変わります。保守サービスはハードウェア会社が顧客を紹介してくれていたため、セールスがいません。このセールス機能は提案型SEのマネジャーが担います。プロジェクト概念がない点も②、③同様に開発マネジャーをスカウトし、提案型SEがプロジェクトリーダーになります。

# THEORY50

**異業種参入では  
提案型SEを育てる**



異業種参入

【タイプ】

コンサルティング  
ファーム

【課題】

コンサルティングと  
提案が混乱

開発SEが  
いない

プロジェクト管理が  
できない

【解決の方向】

コンサルタントの一部を  
提案型SEに

SEを  
雇う

or

ソフトハウス  
提携・買収

提案型SEが  
プロジェクトリーダーに

ネットワーク  
キャリア・ベンダー

価格勝負を  
したくなる

個別プロジェクト  
の概念がない

通信サービスから  
脱却できない

提案型SEを  
見積をする

ネットワークサービスとは  
損益システムを変える

データベーススペシャリスト  
を指名する

その他  
異業種参入

設計が弱い

情報システム部の  
スピアウトで失敗

ネットワーク技術  
が弱い

提案型SE兼  
設計SEを育てる

利用者を提案型SE  
にする

ネットワークスペシャリストを  
指名する

ベンチャー

マーケティングが  
弱い

セールスがない

業績が安定  
しない

社長が提案型SEマネ  
ジャー兼 marketer となる

提案型SEが  
セールス行なう

ニッチマーケットに  
特化する

最後は異業種参入、つまりソリューションビジネスの主要部をなすITやシステム開発とは直接関わりのない世界からの参入です。このタイプに共通しているのは「提案型SE」には目を向けず、システム開発パワーを何とか確保しようと思っていることです。ソリューションビジネスの中核は提案型SEであり、これがビジネスのコアであることを経営者が認識するのが第一歩です。

異業種参入は次の4つのタイプに分かれます。

### ①コンサルティングファーム

経営コンサルティングやITコンサルティングをやってきた企業が、売上・利益の増大のためにシステム開発を含めて受注しようという動きが生まれています。外資系コンサルティングファームなどにその動きが顕著であり、その企業ブランド力から優秀なSEを集め、良い顧客をゲットしています。

このタイプの最大の問題点は、コンサルティングと提案が混乱していることです。コンサルティングではその提案のみが仕事で、実行は別企業ですので「べき論」中心です。ソリューションにおける提案では、それを自らが実現しなくてはなりません。最近、このビジネスにチャレンジした企業が多く、まだその実施結果が必ずしも出ていませんが(彼らがコンサルティングするほどですので、大規模かつ長期のプロジェクトといえます)、潜在的トラブルを抱えたプロジェクト・システムは多いと考えられます。今からでも遅くないので、コンサルティングビジネスとソリューションビジネスを完全に分離することです。そのうえでソリューションビジネスの中核となる提案型SEをコンサルタントの中から指名し、評論家的コンサルタントとは分けて育てるべきです。

コンサルティングファームがソリューションベンダーに変身しようと考えた時、悩むのはプロジェクトリーダーや設計SE、開発SEがいないことです。そのためシステム開発をどうしても外注に頼るのですが、外注ソフトハウスもソリューションビジネスへの変身を図りつつあり、いつかライバルになるかもしれないというリスクを抱えます。外注をやめ経験者のSEを雇い、これをベースに地道に育てるか、他のソフトハウスと提携または買収するしかありません。無論このタイプにはプロジェクト概念がないのですが、このSEが経験していますので、彼を頼りにコンサルタント出身の提案型SEがプロジェクトリーダーになるしかありません。

## ②ネットワークキャリア・ベンダー

ネットワークビジネスの規制緩和および価格競争激化を受けて、ネットワークキャリアやネットワークベンダーのソリューションビジネスへの進出が顕著です。このタイプの最大の特徴はネットワークビジネスが固定費回収型（顧客が増えても変動費があまり発生しないので、価格は安くてもよいから顧客は1人でも多い方がよい）であるため価格競争が激しく、これに慣れているためソリューションビジネスでもつい「価格勝負に出る」ということです。ソリューションビジネスは個別プロジェクトによる損益回収という概念がベースです。セオリー通り提案型SEを指名し、彼がプロジェクトごとの見積を行い、そのうえでプロジェクト管理を行うというルールに変えます。しかし逆の見方をすれば「長期的に損益を考える」という他タイプにない強みを持っているといえますので、これは提案型SEのマネジャーに残しておきたい所です。

もう1つの問題点はネットワークビジネスから脱皮できないということです。これはシステム開発ノウハウが欠けているということではなく、データベースのノウハウがサービスに加味されていないという意味です。これについては社内でユーザーの立場でデータを利用してきたエンジニアの中からデータベーススペシャリストを指名することで解決します。

### ③その他異業種参入

メーカー、商社など従来IT・情報システムの利用企業が、利用ノウハウをベースとして他社にソリューションを行うということも増えてきています。このタイプの最大の問題は「システム設計」という基本技術の欠如です。これは外部からプロのSEを雇うということも考えられますが、セオリー48で述べたようにこの設計技術が今大きく変化していることを考えると、むしろ提案型SEを指名し、彼に設計技術も勉強させる方が得策です。

実際に参入するケースのほとんどは「情報システム部をスピンアウトして別会社にする」というものですが、なかなかうまくいきません。それでは従来のソリューションベンダーと差別化できないからです。このタイプの強みは「利用ノウハウ」にあるはずですので、ソリューションビジネスを行なう子会社に情報システム部だけではなく利用者を入れ、利用者を提案型SEとすべきです。

技術面ではネットワークが利用側からは完全にブラックボックスでしたので、これが全くないという状況です。これはネットワークスペシャリストを外部から雇うという手もありますが、そうするとどうしてもこの人が会社の中で浮いてしまって、技術が社内に伝わりません。従業員の中で若くてインターネットなどに興味のある希望者を、ネットワークスペシャリストに指名して、彼に学習させるべきです。

#### ④ベンチャー

ITベンダーなどに勤務していたSEが独立して会社を作るという場合がほとんどです。このタイプは超優秀なSEが社長ですので、仕事さえあればどんなソリューションでもできるのですが、マーケティング、セールスが弱く、良い仕事を取れず、結局システム開発の下請、保守など便利屋になってしまいます。

このタイプは社長が提案型SEマネジャー兼マーケター兼セールスマネジャーになり、社内で一番優秀な人たちを提案型SEに指名して(現在もっとも利益を生んでいる人でも何とかがんばって選びます。そうでなければ起業した意味がありません)、彼にセールスをやらせるしかありません。そして業積を安定させるために、皆がやりたがらない小さなマーケット、見過ごしているマーケットをターゲットとし、そこでのシェア100%を目指します。