第３章　開発

本章では、開発の各プロセスを追って、プロジェクトを上手に遂行していくにはどのようにしていくべきかについて述べている。

ＪＵＡＳでは、プロジェクトの実態を調査し、また、上手な遂行方法はなにかをテーマに議論を重ねてきた。その成果の一部をここに紹介している。

本章の内容は下記のとおりである。

第３章　開発

第１節　開発の実態

システム開発を予定通りの予算、工期、品質で推進することには、相当に大きな努力を伴う。まずは最近の開発実態を眺めてみよう。そこに改善の源泉がある。

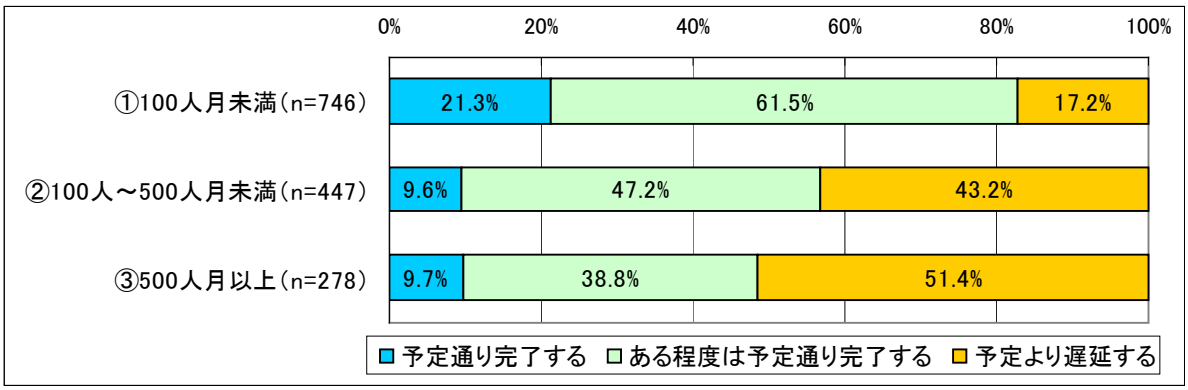
１．プロジェクトの調査
２．遅延理由分析
３．開発ベンダーへの満足度
４．発注者としての反省点

１．プロジェクトの調査

JUAS では、プロジェクトの推進がうまくいっているかどうかについて、工期・予算・品質の視点で、ユーザー企業から調査している。（IT 動向調査 2005）

開発工数が、100 人月以下、100～500 人月、500 人月以上の 3 つの分類して、プロジェクトの規模別に聞いている。

100 人月は数ヶ月から半年の事業部レベルのプロジェクト、500 人月以上は全社プロジェクトに相当すると考えられる。



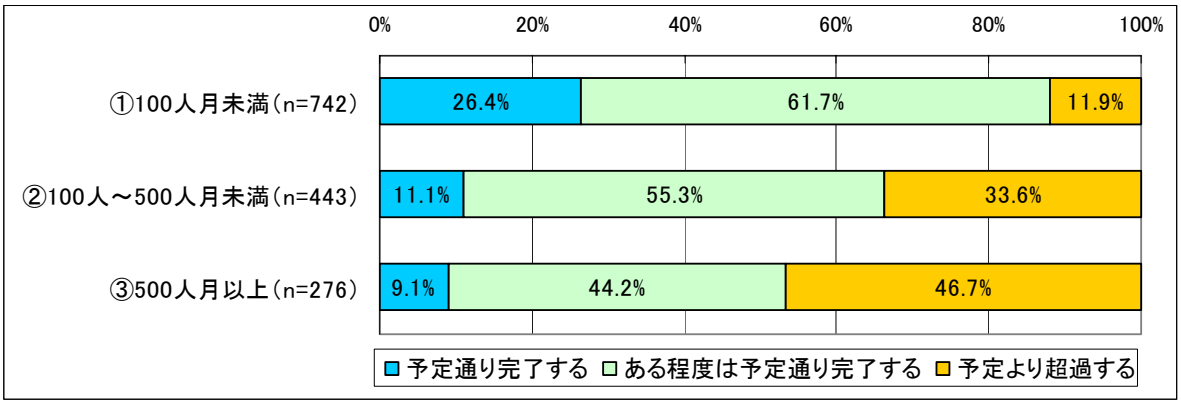
（IT 動向調査 2005）

図表 3-1-1　工期

驚くべきことに、100 人月以下の規模のプロジェクトでも、「予定通り完了している」と答えた企業は全体の 21.3%にすぎない。100～500 人月規模のプロジェクトになると 9.6%、500 人月以上では 9.7%の企業である。逆にいえば、100 人月以上のプロジェクトの 90%あまりは当初の予定どおり仕事が進んでいないということである。

ある程度はうまくいったという答えを除き、予定より遅れてしまったと回答した企業は、100 人月以下のプロジェクトでは 17.2%しかないが、100～500 人月では、43.2%、500 人月以上の大規模プロジェクトでは 51.4%の企業がいつも予定より遅れてしまっている。工期の遅れが日常茶飯事化している現状が浮かび上がってくる結果である。

＜予算＞

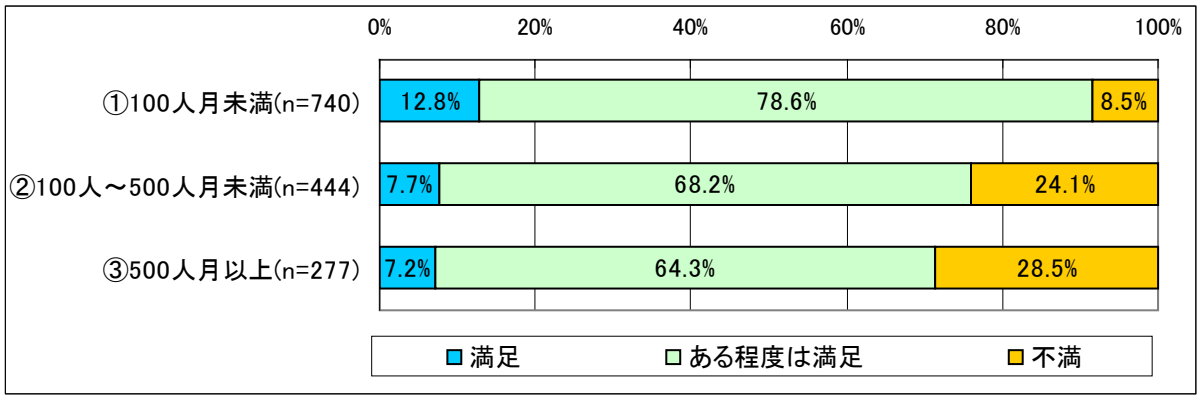


（IT 動向調査 2005）

図表 3-1-2　予算

予算については、100 人月規模のプロジェクトでは、ほぼ予算どおりできていると回答した企業が 88.1%だが、100～500 人月規模になると 66.4%、500 人月以上になると 53.3%の企業しか予定の予算でプロジェクトが完成していない。

＜品質＞



（IT 動向調査 2005）

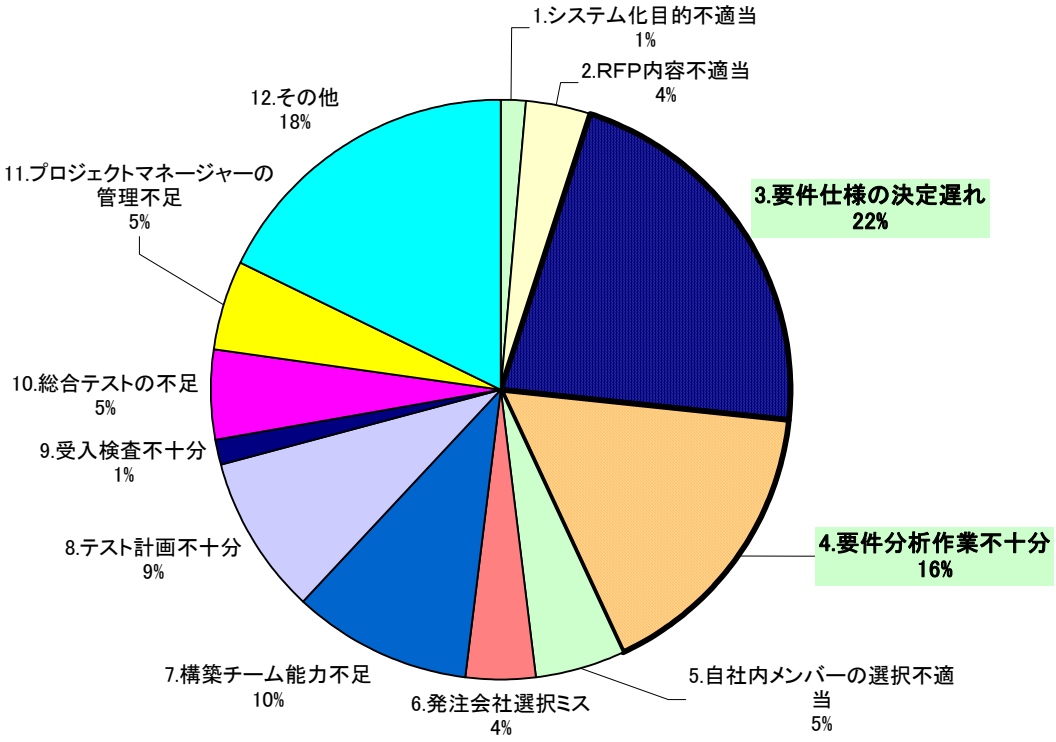
図表 3-1-3　品質

システムが予定どおりの仕上がりであったかどうか、品質という言葉で聞いている。規模により不満度が著しく異なっている。100 人月以下では、90%以上の企業が満足しているが、大規模システムユーザーの不満度は高い。大規模システムは大企業で開発されており、信頼性に厳しいアプリケーションであること、システムに対する利用者の期待が大きいこと、大規模システムを高品質で作り上げることが難しいことなどが影響しているものと思われる。

２．遅延理由分析

（１）業務分析とＲＦＰの作成

ソフトウェアメトリックス調査報告書 2005 年版によると、要求仕様書の問題で約４０％のプロジェクトが遅延している。（他の調査データも同様な結果になっている）



（ソフトウェアメトリックス調査 2005）

図表 3-1-4　工期遅延理由

（２）ＪＩＳＡのデータ

プロジェクトのコスト・品質・納期については、JISA（社団法人情報サービス産業協会）で 2004 年春にアンケート調査[1]を実施している。

その一部を示すと以下のとおりである。２／３は、予算オーバーである。

＜プロジェクトが完結するまでに要したコスト＞

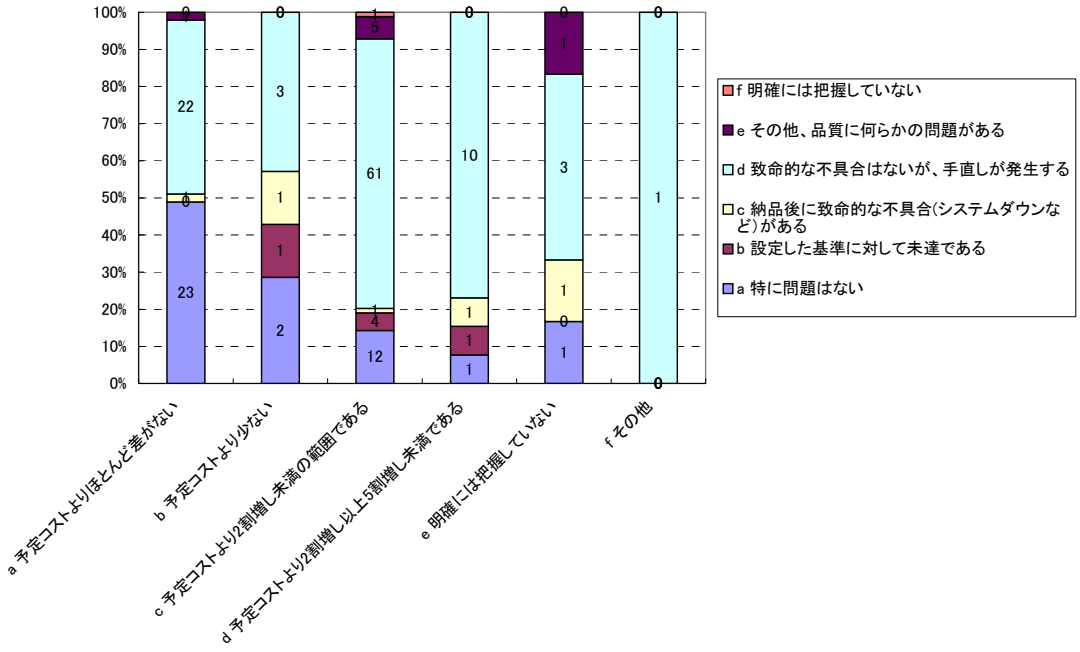
		回答率%
a	予定コストよりほとんど差がない	30
b	予定コストより少ない	4
c	予定コストより 2 割増し未満の範囲である	53
d	予定コストより 2 割増し以上 5 割増し未満である	8
e	明確には把握していない	4
f	その他	1

JISA（社団法人情報サービス産業協会）

図表 3-1-5　プロジェクト完了までに要したコスト

[1]「情報サービス産業における受注ソフトウェア開発の技術的課題に関するアンケート調査 2004」社団法人情報サービス産業協会

＜品質とコストの関係＞

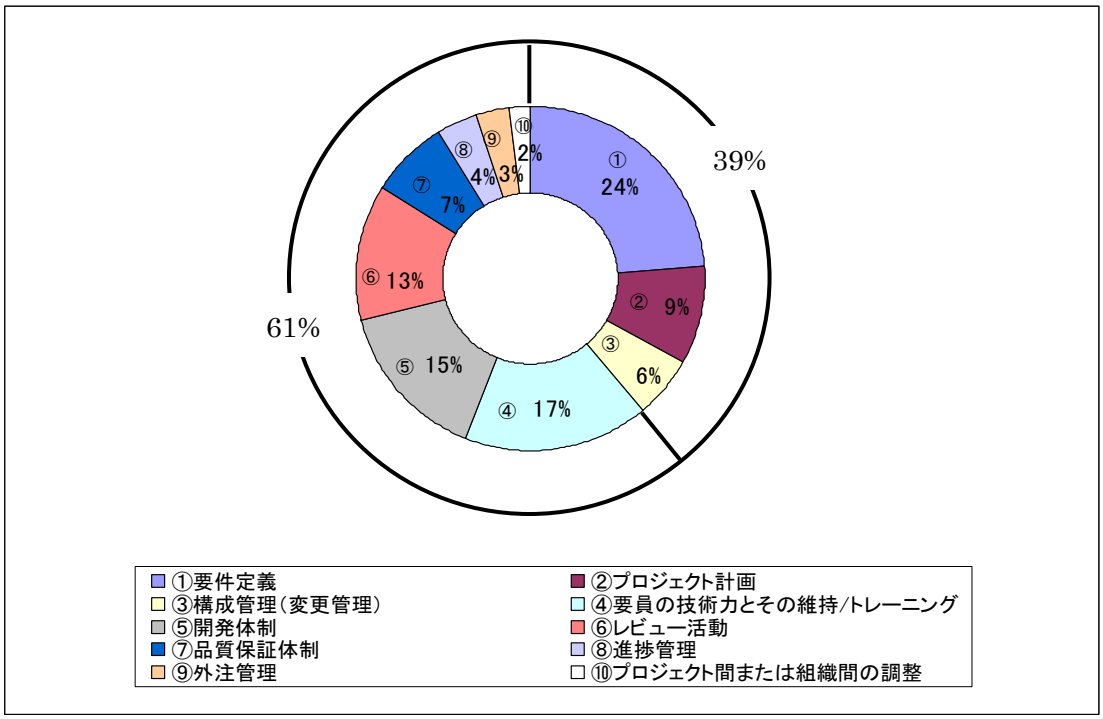


JISA（社団法人情報サービス産業協会）

図表 3-1-6　品質とコストの関係

90%は納期を守っているが、手直しが大きい。

＜品質・コスト・納期の成否に及ぼす要因＞



JISA（社団法人情報サービス産業協会）

図表 3-1-7　品質とコストの関係

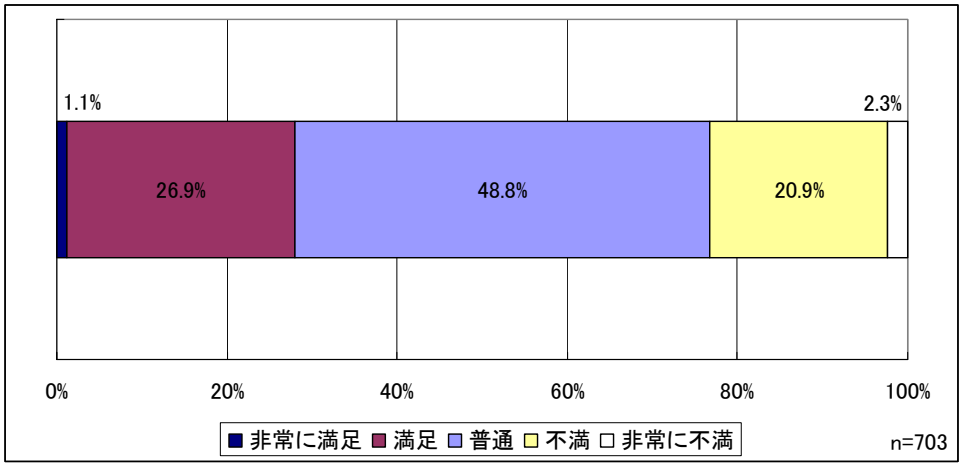
約 60%はベンダー要因であると言える。このこと（ベンダー要因が 60%である）と前述のソ

フトウェアメトリックス調査データにおいて要求仕様に関連する工期遅延の割合が約 42%であることはほぼ同じことであり、興味深い。しかし要求仕様関連はユーザー要因とし、それ以外はベンダー要因とすると、その割合は約 60%であり、ベンダーの更なる努力に期待がかかることになる。

３．開発ベンダーへの満足度

工期遅延だけでなく、開発全体の実態を観察してみる。

（１）開発ベンダーへの不満



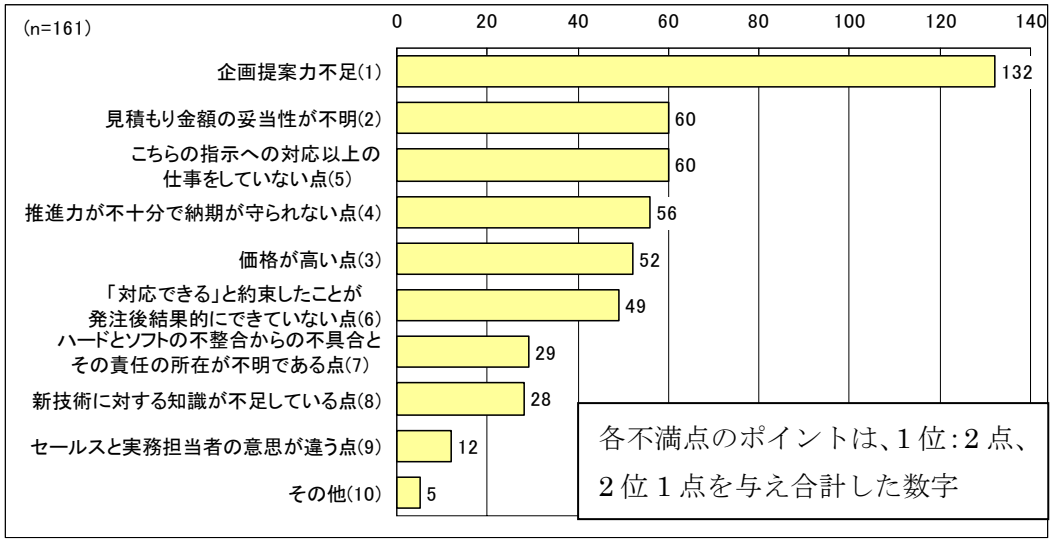
（IT 動向調査 2005）

図表 3-1-8　開発ベンダーへの満足度

まず、開発ベンダーへの満足度を分析してみた。

「非常に満足」「満足」をあわせると 28.0%、「不満」「非常に不満」をあわせると 23.2%となる。（図表 3-1-8）発注者の評価は厳しく、改善が望まれる。

（２）開発ベンダーへの不満理由



（IT 動向調査 2005）

図表 3-1-9　開発ベンダーへの不満点

最も多くの不満は「企画提案力不足」である。回答企業の 80%がこの不満を抱いている。

「顧客は何を期待しているのか？」をベンダーはもっと考えねばならないが、ユーザーも何を期待しているのか？具体的に要望を提示する必要がある。

自分の知恵のないところ、足りないところを、ベンダーから引き出したいならば、的確な質問・要望を投げかけねばならない。

２番目の不満は見積金額の妥当性不明である。５番目の「価額が高い」とあわせるとこれも問題である。しかし一流 SI ベンダーの利益率は決して高くない。

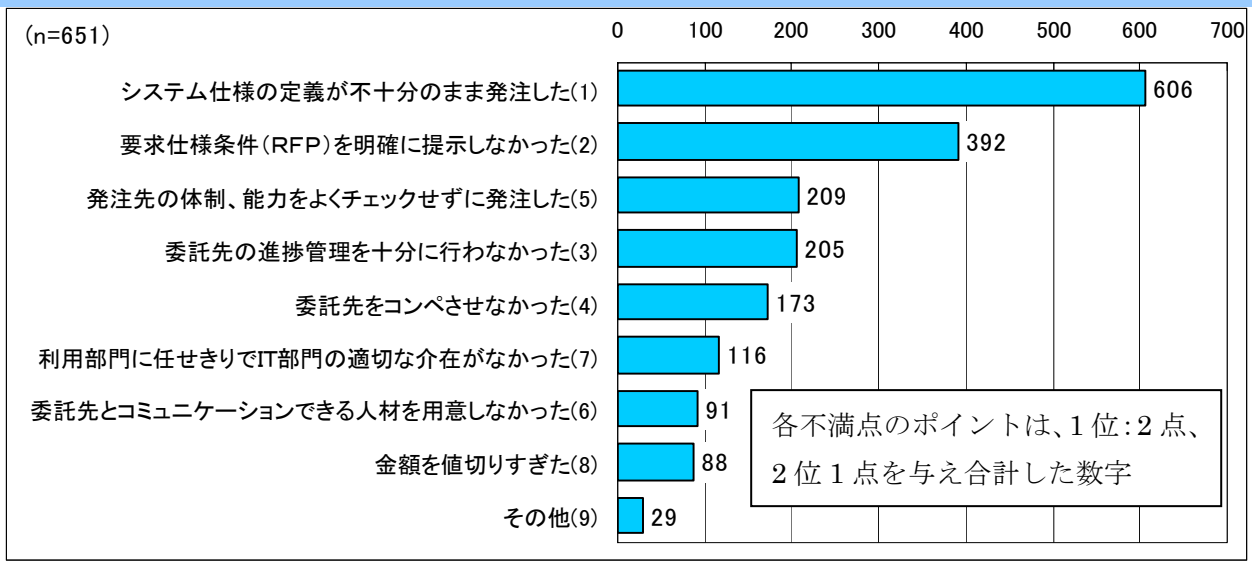
「価額が高いけど儲けてもいない」のは失敗プロジェクトが利益を帳消しにしていることが想定され、産業自体が未成熟で、各種標準化などが不足していたり、作業改善が不十分なことがあると思われる。

価額の透明性については、その第一歩の案を第３節の見積で提示するので参考にしていただきたい。

３番目以降は仕事をする上でのマナーの問題である。「こちらの指示への対応以上の仕事をしていない。」「推進力が不十分で納期が守られない」「対応できると約束したことが発注後結果的にできていない」など不満が多い。

これは発注者であるユーザーの声であるが、ベンダーからの言い分もあると思われる。両者が解決する姿勢を持って話し合わねば前進はしない。2004 年に IPA（情報処理推進機構）にソフトウェアエンジニアリングセンターが設立され、両者の意見交換の場が設定されている。このような場を大いに活用して両者が納得する仕事の仕方を追及せねばならない。

４．発注者としての反省点



（IT 動向調査 2005）

図表 3-1-10　発注者としての反省点

ユーザー自身はプロジェクト開発の失敗の原因をどのように考えているのか？を整理したのが図表 3-1-10 である。

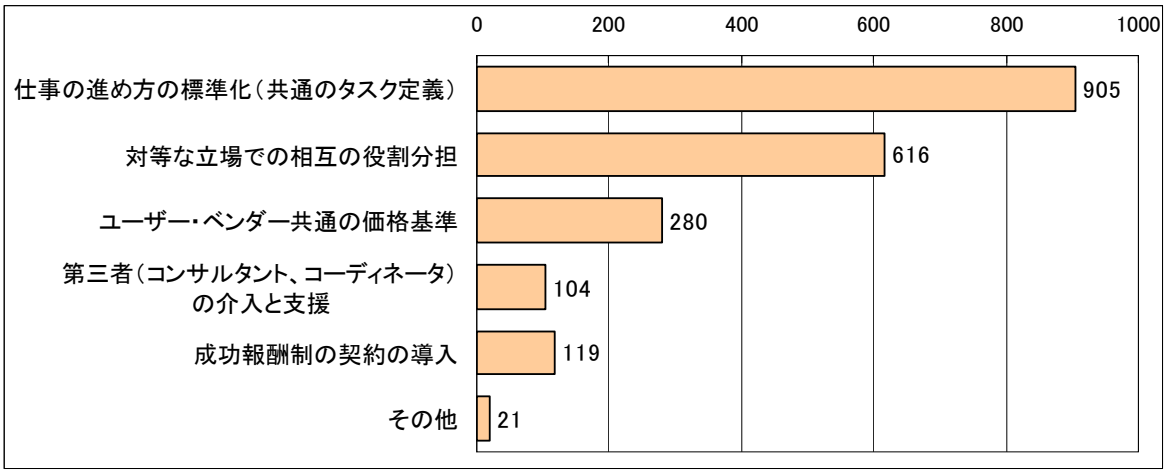
「システム仕様の定義が不十分のまま発注した」「要求仕様条件を明確に提示しなかった」と反省している企業が多い。

この現状は仕方ないとしても、日本全体でどのように改善できるのか？が問われている。毎年同じことを嘆いて、反省しても効果はない。

実はこの「要求仕様条件が不十分とは何を指し、どのように改善すべきか」を明快に解説・指示したガイドはない。

この第一歩として、JUAS では「エンドユーザーによるビジネスシステムガイド」と「システム技術文章化ガイド」を２年間にわたって研究しまとめた。まだまだこの道の奥は深く、問題解消にはほど遠いと思われるので、研究を深めていきたい。

最後に「委託先と円滑な協力関係を築くために重要と考える点」についてのアンケート結果に触れたい。



（IT 動向調査 2005）

図表 3-1-11　委託先と円滑な協力関係を築くために重要と考える点

「仕事の進め方の標準化」,「対等な立場での相互の役割分担」が主たる意見である。
発注者、受注者の IT リテラシーの向上をどのように進めていったらよいのか？
まだまだ問題と対策は両者に数多く残されている。

第３章　開発

第２節　業務分析と見積要求仕様書（ＲＦＰ）の作成

プロジェクトの成功の第一歩はユーザーが何をしたいのか？このシステムに何を期待するのか？を明確にシステム開発者に正しく伝えることである。

ＲＦＰ作成の現状にふれ、ユーザーの業務担当部門が自ら見積要求仕様書を作成することをここで提案する。勿論現場の実態を踏まえた提案でなければ意味がない。

最低限これだけは利用者自らが書いて欲しい（レベル１）から、高いレベル（レベル５）まで、各企業の実力、実態に合わせてまずは「自らの道具についての仕様の明確化」を、勇気を持って作成し始めてみる事の素晴らしさを体験できる手がかりになれば幸いである。

１．見積要求仕様書を取り巻く課題と対策
２．ＲＦＰ作成の現状
３．業務担当部門による見積要求仕様書の作成
４．ＪＵＡＳの考える業務分担（ＳＬＣＰとの対比）
５．利用部門が作成できるＲＦＰ
６．業務システム定義の記述レベルとその構成
７．見積要求仕様書の活用と課題
８．情報システム構築コストダウンの追及

１．見積要求仕様書を取り巻く課題と対策

システム開発プロジェクトが予定通りに進めにくい難問であることは一般に認識されている。

・これはシステム開発が建築工事などと比較して見えにくいものであること、
・仕様が決定したようで、いざプログラムを作成し始めるといろいろな抜けがでてくるようにどこまで詳細に書けばよいのか分かりにくいこと、
・担当者が変わると管理思想が変わりシステムの仕様も変わる流動性の高いものであること
・開発担当者の能力に依存し、設計開発能力に数倍の差があること

などの原因も判明しつつある。

ソフトウェアメトリックス調査 2005 によると、遅延理由の約 40%が見積要求仕様書の問題であることも解明されている。JISA の調査でも同じような結論が出ている。しかし残りの 60%はベンダー自らの問題であるとも反省されている。

この見積要求仕様書をどのように書けばよいのか？のガイドラインは出ていない。

注：システム分野では一般に RFP といえば「見積要求仕様書、あるいは要求仕様書のこと」と漠然と理解されている。建設、プラントなどの世界でのプロジェクトマネジメント理論では RFP と RFQ は使い分けられている。本書の中ではラフな要求の「RFP（見積要求書)」と詳細な定義をしてある「要求仕様書」を分ける意味から「見積要求仕様書」の単語を使用している。本来は「RFQ（見積照会書)」などの単語を使うべきとの議論もあったが、あえて本書では広く慣習に習って見積要求仕様書とした。

２．ＲＦＰ作成の現状

IT 動向調査 2004、2005 によると、見積要求仕様書の作成は発注者であるユーザーの重要業務にもかかわらず、ベンダーまかせが多いと言う結果が出ている。

システム利用者が最低限度の要望、仕様条件を書き、システムの専門家でないと作成できない残りの部分は、企業内の IT 部門の SE またはベンダーに依頼しても良い。しかし、内容の吟味は、ユーザー企業内関係者全員で実施しなければならない。

現実を見てみると、暗黙知のシステム関連の 3 文字単語が並び理解しがたい文章表現が横溢しているドキュメントでは、十分な理解がなされるとはとても思えない。システムの説明は図だけでは説明が完結しないのでシナリオなどの文章表現が多いのも理解を曖昧にさせている。

「承認印をくださいとベンダーSE が言う」ので「仕方なく印鑑は押したが、見たと言うことで理解したと言うことではない」などとユーザー側からの迷言も登場してくる世の中である。

「本来はユーザー企業の中のシステム利用部門が見積要求仕様書の業務部分を定義し、コンピュータ技術に関する部分を IT 部門の SE が記述して、見積要求仕様書を作成した方が、使いやすい、良いシステムができる」との考えに基づき、実現可能なドキュメント体系を整理したのが、本章である。

SLCP98（Software Life-Cycle Process 1998 年版）には、取得プロセスの実施項目として、提案依頼書の作成が規定されているが、これ以上の内容には触れていないので見積要求仕様書作成の参考にはならない。

「ユーザー企業は見積要求仕様書をもっと明確に書いて欲しい。そうすれば、こんなに安くできます」と言って欲しいのだが、そのようなベンダーは殆どいない。

ユーザー企業は最近、IT 企画部門と情報子会社に分かれており、力を持っている利用部門を含めて 3 者の見積要求仕様書の作業分担は各社各様である。

このような背景を踏まえてエンドユーザーの力をもっと活用し、明快な見積要求仕様書ができないものかと 2 年間かけて JUAS では研究、議論してきた。その結果を踏まえた提案を以下に試みる。

３．業務担当部門による見積要求仕様書の作成

この項では、以下の事項に触れる。

・見積要求仕様書の作成には何故業務担当部門の協力が必要なのか？

・どのようにすれば可能なのか？

・この問題について業務担当部門と IT 部門はどのように協力しあえばよいのか？

・その結果の効果はどのように現れるのか？

・実現のための具体的な対策はどうするのか？　など。

（１）業務システム把握分析と改革能力の獲得

若年層の減少に基づく国内消費者数の頭打ち、商品の飽和感などによる国内需要は長期的には横並びか減少が始まっている。そのために、

・顧客と企業を近づけ真の要望を実現するためのシステム、

・販売、生産のビジネスにスピードアップを持ち込むための部門間、企業間の連携システム、

・情報共有を全社的におこない生産性を向上させるためのシステム、

など、企業はさまざまな背景と理由からビジネスモデルの再構築を迫られている。

このような流れの変化を一番感じているのは、業務担当部門〔＝システム利用部門〕であり、そのためのヒントや解決策を考えているのも業務部門である。

また、IT 技術の進歩を肌で感じており、この素晴らしい技術を何とか自分たちのビジネスに活用できないか？と考えているのも業務部門である。詳しい技術的仕組みは分からないが「この道具で何ができるのか？」を業務部門が理解できれば、それを活用するアイデアは生み出せる。

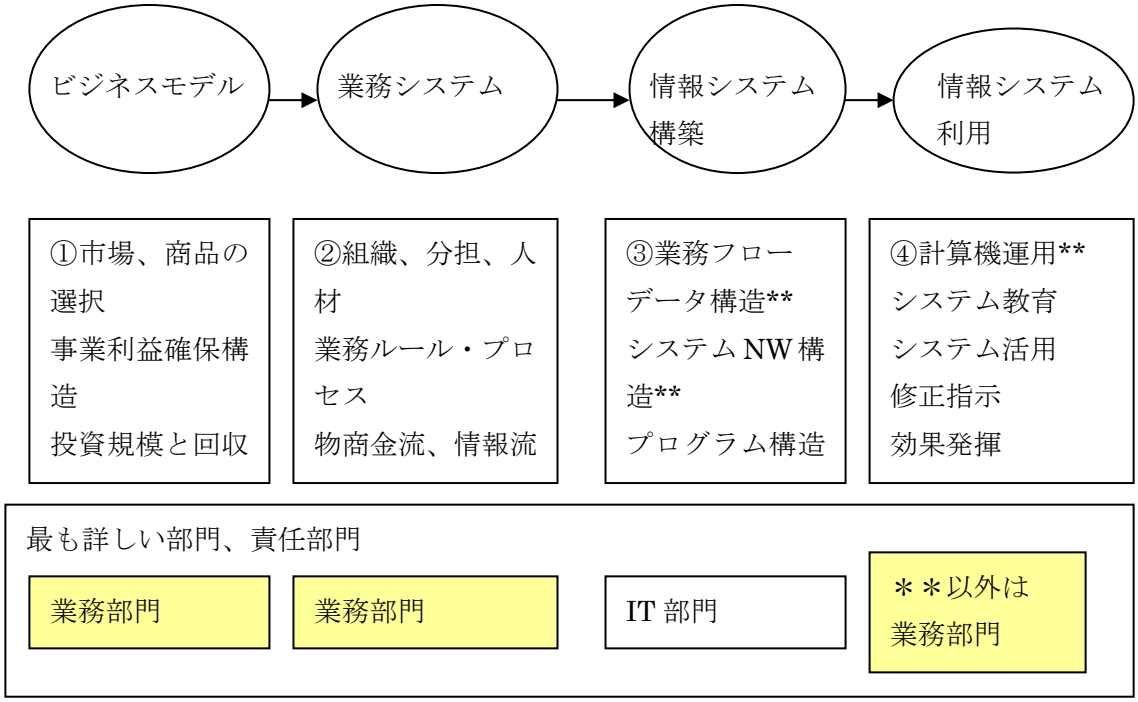
IT なしで毎日の業務が実施できる会社は少ない。程度の差こそあれ IT は身近なものになってきている。

システム開発の前にビジネスモデルをどのように変革するのか？を主として議論し、業務改革を推進する部門は業務部門である。

図表 3-2-1 に表されているごとく、企業の運営はビジネスモデルがまずあり、それをつかさどる業務システムがある。その作業は人手で行うか？機械を使うか？他社に任せるか？などが決められる。このような決定の役割を担う者は業務部門である。

情報システムの構築は IT に詳しい IT 部門に任せるが、できあがったシステムを利活用するのはこれまた業務部門である。

「IT は分からない」などと時代離れをしたことは言っている人に業務部門長は任せられない。積極的に活用する姿勢があって初めて効率の良い企業運営が実現できる。



図表 3-2-1　ビジネスモデルから情報システム運用までの流れと責任

（２）これまでのＲＦＰ方式の反省

自社の業務システムの改革は重要業務であるが、JUAS の調査によると要求仕様書の作成作業のほとんどを自社で実施している企業は 23%であり、残りの企業は、ベースは自社で作成するが細部は委託先が作成している（53%）、すべて委託先が作成（２４％）するとなっている。

その結果の満足度は低くなり、その原因について「要求仕様書を明確に提示しなかったから・・」と反省している。

年々自社で RFP を作成している企業は増加の傾向にあるが、上図にあるごとく、本来一番詳しいのは業務部門であり、この要求仕様書作成の作業を他社に任せて上手く行くはずがないことをまず認識しておいて欲しい。

効率を考えると自分で作成するのが費用も一番安いはずである。他人に任せてもチェック作業は当然自分で実施せねばならず、それほどの作業負荷の軽減にならない。

①②と④は業務部門の仕事であるし、②はそれほど難しくはない作業である。勇気をもって自分で作成することに挑戦して欲しい。

そのための見積要求仕様書作成一式についての提案を以下に説明する。ベンダーの厳しい目で作成した見積要求仕様書をチェックしてもらわねばならない。

従来の「ベンダーの作成した要求仕様書をユーザーがチェックする」作業方式から、「ユーザーが作成した見積要求仕様書をベンダーがチェックする」形に変わった時、システムトラブルの発生や、投資したけど使われないシステムが多発する事態から脱却できるに違いない。

４．ＪＵＡＳの考える業務分担（ＳＬＣＰとの対比）

自己の業務をどのように変えたいのか？を考えるのは、まず業務ラインの仕事である。

従来からシステム開発において最も広く活用されてきた SLCP に基づき利用部門と IT 部門の作業責任を区分してみたのが下の図である。

「SLCP には提案依頼書には何を書きなさい」との記述が 44 項目にわたって整理されているが、「誰がしなさい」とは書かれていない。「システム化の依頼範囲、依頼内容、業務の詳細」を書きなさいとあるが、これらについてそれ以上の記述はない。

また、ベンダーの皆さんが「ユーザーが詳しく書いて欲しい」と願っている、提案依頼書の内容については、記述方式の標準化、詳細度、記述のための前提条件など基本的整理がされないままに数十年経過し、トラブルプロジェクトが多発し、日本の情報産業の優秀さ、真摯さを世界に宣伝する機会を失った。JUAS はこれらのことを反省して、今回ユーザーによるシステム見積要求仕様書の作成について以下のようにまとめた。

「業務部門の利用者がそこまではできない」「これは理想論である」と言われないように、業務部門が担当する範囲を５段階のレベルに分けて現実解に近づけてある。

業務部門ができない範囲は IT 部門が助ければ良いし、場合によってはベンダーに補足してもらっても良い。従来のレベルから一歩でも脱却できるような努力を業務部門に期待したい。

なお、ドキュメント中の個別業務処理定義書の書き方は、現実にはさまざまな形や方式が使われているので、この部分の記述方式は利用者に一任されている。今後、この書き方の研究も進めて行く必要性を感じているが、本年版では利用者の創意工夫に任せている。

基本項目	内容	実行責任部門	
		利用部門	IT 部門
①システムの概要	システム化の目標・方針	◎	○
	狙いとする効果	◎	○
	運用対象者	◎	○
	既存システムとの関係	○	◎
②提案依頼手続き	説明会日程		◎
	対応窓口		◎
	提供する資料	○	◎
	参加資格条件		◎
	提案手続き		◎
	選定方法		◎
③依頼事項	システム化の依頼範囲（業務関係）	◎	○
	（システム関係）	○	◎
	依頼内容、業務の詳細（業務関係）	◎	○
	（システム関係）	○	◎
	システム構成	○	◎
	納期（本番時期）	◎	○
	（S/W、H/W の導入時期、テスト、移行時期など）	○	◎
	必要な技術・技術者の資格		◎
	成果物、納入品		◎
	修正プロセスへの参加	○	◎
	共同レビュー	○	◎
	工程計画	○	◎
	開発推進体制	○	◎
	開発手法、開発言語	○	◎
	支援ツール		◎
	ソフトウェア製品の使用		◎
	品質・性能条件		◎
	保守条件		◎
	運用条件	○	◎
④開発体制・開発環境	役割分担	○	◎
	作業場所		◎
	開発機器・使用材料の負担		◎
	貸与物件・資料	○	◎
⑤保証要件	システム	○	◎
	品質保証基準		◎
	システムの性能		◎
	セキュリティ		◎
	デモ・テスト計画		◎
⑥契約事項	発注形態	○	◎
	検収・支払条件		◎
	瑕疵担保責任		◎
	機密保持		◎
	著作権など		◎
⑦その他	用語	○	◎
	外部委託契約に関する管理	○	◎
	リスクに対する相互認識	○	◎
	仕様変更・機能追加などの条件	○	◎

凡例　◎作業主体（&責任あり）　　○作業支援または内容の吟味・確認
SLCP では 44 項目になっているが、ここでは一部詳細化して 47 項目で分類した。

図 3-2-2　提案依頼書（＝見積要求仕様書）の作成分担

注：業務の詳細については「何をするのか？」は利用部門の責任であるが、「どのようにするのか？」（手順）は SE でないと決めにくいので、両者の責任とした。作成ドキュメントの具体的な分担は、参加者の実力、企業環境、などにより異なるので、５ケースに分けて整理した。

５．利用部門が作成できるＲＦＰ

（１）基本的考え方

企業において毎日実行している実際の業務を一番よく知っているのは業務部門（＝システム利用部門）である。

・「何をシステム化してほしいのか？」「前提条件、実行後の評価方法」は通常の文章で表現すればよく、特別な知識・能力を必要としない。
・プログラムの作成に結びつく業務処理手順を記述することは、専門家でない実務者には、難しい業務である。したがってその部分も、できる人は作成してもらってかまわないが、一般には専門家の SE に依頼することになる。
・システム化する、しないにかかわらず、企業においては「企業／事業の規定」「「業務規定（基本的商習慣などの記述を含む)」を整備しておかねばならない。
・これらを整備しておくことは業務部門の義務である。
・企業の情報システム部門は、開発運用の支援者であり推進者でもある。

常に受身になっているだけでは IT 部門の意味がない、SE 自らが業務部門あるいは部門間の問題を発見し改善するモデルを見つけ提案できなければならない。

業務部門のレベルアップを支援する傍ら、自らが新技術・新手法を学び問題感知力を発揮してさらに上のレベルの提案活動ができるように期待したい。

（２）前提条件

実務にたずさわっている人は、必ずしもコンピュータの専門家ではないので、システムドキュメントを作成することについて限界がある。

したがって、下記のような前提条件を設ける必要があると思われる。

①利用者は自らが必要とする機能が何であるかを開発者に要求する。
ただし、IT に詳しい開発 SE が明確に定義した方が良い項目はそちらに任せる。
例：「画面から入力されるデータには新規、訂正、追加、削除がある」これらの処理は結構複雑で難しいので開発者に任せる。「どのようにするか？」まで記述することは利用者にとって難しい作業である。

②入力データをもとに例外作業が発生する場合は、その例外作業項目を構造図などで必ず記述する。

③この例外作業を具体的にシステムにどのようなプログラムにして取り入れるのか？は開発者に任せる。

④利用者は開発者に対して「文章の裏を読め」と言わないで、「なにが例外作業にあたるのか？」を明確にする。ルール化できない例外作業は「異常作業」としてシステム外の扱いとなる。誤解が生じないように極力具体的な表現方法を取る。

このユーザーとベンダーとの間で、「見積要求仕様書作成の前提条件」を明確にすることは、非常に重要な課題である。

「RFP があいまい」とベンダーSE が言う時があるが、どこまで書かなければならないのか？は今一つ明確ではない。これは、両者が乗り越えてゆかねばならないテーマである。暗黙知を含めてこの前提条件を活用することによって、両者の負担が減少するのであれば、積極的に活用しない手はない。

（３）ユーザーが定義するドキュメント

業務担当者はビジネスの動きを調査分析し、その機能／業務ルール／業務処理フロー／帳票類などを明確にし、業務システム（業務の仕組み）として提示せねばならない。

このために、業務担当者は図表 3-2-3、図表 3-2-4 に示される 14 種のドキュメントを作成しなければならないが、現実の対応力に合わせて５段階レベルで対応することになる。特にレベル１～３の範囲では，IT 部門の SE やベンダーの協力に負うところが大きい。

なお、個別業務処理定義書はさまざまな記述様式が存在する（HIPO、フローチャートなど後述の事例参照)。適材適所で使い分ければよい。

レベル１から５に移行するに従いベンダーへの支払いは減少する。業務内容の吟味や整理を自らがすることによる業務見直しの副産物も大きいので、できるだけレベルをあげて対応することが望まれる。レベルが低いままに広範囲の作業をベンダーに依頼したドキュメントのチェックについてはユーザー企業関係者がじっくり時間をかけ確認しなければならない。全ドキュメントの作成時間をベンダーに確認し、その 10%以上の時間をかけチェックしたシステムには欠陥が少ないことが、JUAS の調査でも確認されている。

（４）業務担当部門とＩＴ部門の作業分担の考え方

業務担当部門は以下の作業を分担する

①ビジネスプロセスの分析整理

②ビジネスプロセスに関連するビジネスルール／業務ルール

（例外処理のルール化）

③ビジネスプロセス間の指示／帳票による連携関係の分析・整理

（正常処理の連携と例外処理の連携を含む）

④ビジネスプロセスに関連するデータ、情報、知識の整理

IT 部門のシステム設計者はシステムの専門家として以下の作業を行う

①ビジネスプロセス内のビジネスロジック

②ビジネスデータのデータベース構造

③情報システム自体の例外事態に対応する処理やルール

④詳細な画面レイアウト,帳票レイアウト

６．業務システム定義の記述レベルとその構成

業務部門だけでは作成が困難なドキュメントもあるので５段階のレベルを設定した。

ここで、レベル－１のビジネス機能提示レベルの定義は、これまでの RFP の記述内容と同等である。

レベル－２のビジネスプロセス提示は、簡単な業務システムや新規の業務システムを記述する際に利用され、IT 部門やシステムベンダーの知見を生かして具体化する場合に採用される。

レベル－３は、標準的な業務システムの定義で使用されるレベルであり、ISO9000 等の認可を受けている企業の業務部門は問題なく記述できるレベルである。システムベンダーのシステム設計者はこのレベルの仕様から具体的な情報システム／アプリケーションソフトウエア設計を行う知見とスキルを持たねばならない。

レベル－４では個別業務処理定義書までを提示する。このレベルの場合、システムベンダーがプログラム設計を中心に実施することとなり、発注側はシステム開発のコストダウンを要求することができる。

レベル－５では個別業務処理定義書とデータ項目定義書までを作成する。システムベンダーは、レベル－４以上にプログラム設計工数の削減を要求される仕様となる。

ケース１とケース２は企画段階の作業から、IT 部門の SE やベンダーに依存する形態をとることになる。

各ケースによりベンダーに依存する範囲、作業量が大きく異なり発注価額も当然異なってくる。ベンダーはユーザー企業が作成した見積要求仕様書を熟読し、理解しがたい部分や誤解しそうな場所、ドキュメント間で整合性が取れていない箇所、別な解法の方が良いと思われるケースは、その旨を発注者側に確認取らねばならない。

どこまでこの見積要求仕様書の内容を読みきれるかが、ベンダーの実力の現れるところである。

注：上図のタイトルを「エンドユーザーによる業務システム仕様書」としたのは，ドキュメントＡ～Ｄはユーザー部門では必要であるが、見積時に必ずしも添付する必要はないので、一連のドキュメント群を業務システム仕様書と名づけた。ドキュメント 1～10 は見積要求仕様書となる。一連のドキュメントの事例を収録してあるので参考にしていただきたい。

仕様の責任と記述項目			レベル−1	レベル−2	レベル−3	レベル−4	レベル−5
			ﾋﾞｼﾞﾈｽ機能提示	ﾋﾞｼﾞﾈｽﾌﾟﾛｾｽ提示	業務ﾌﾛｰ提示	個別業務処理提示	個別業務処理/ﾃﾞｰﾀ項目提示
責任分担	1	ユーザ側責任	RFPﾚﾍﾞﾙでﾍﾞﾝﾀﾞ提案書をﾍﾞｰｽに要求仕様書を明確にしｼｽﾃﾑ構築	既存ｼｽﾃﾑ再構築ではﾋﾞｼﾞﾈｽﾌﾟﾛｾｽ定義と既存ｼｽﾃﾑ仕様提示で発注	As-Isﾍﾞｰｽ機能拡張で業務ﾌﾛｰを通常業務/例外業務処理につき提示	To-Beﾍﾞｰｽの業務処理方法の提示による効果的な発注	To-Beﾍﾞｰｽでのｼｽﾃﾑ構築の発注によるｼｽﾃﾑ構築効率追及
			要求仕様承認の責任	基本設計承認の責任	詳細設計/機能性能の承認	納品仕様/成果物の仕様確認	ｼｽﾃﾑ設計の効率化の推進
	2	ベンダ側責任	RFPのﾋﾞｼﾞﾈｽ機能からTo-Beのあるべき姿のｼｽﾃﾑ機能を提案	提示ﾋﾞｼﾞﾈｽﾌﾟﾛｾｽの改革を実現するｼｽﾃﾑ構築を実現	更なるTo-Be機能への展開とｼｽﾃﾑ設計と納品物のQCD追求	To-Beﾍﾞｰｽのｲﾝﾌﾗ処理方法からの改善改革と構造的で明快なｼｽﾃﾑ構築	To-Beﾍﾞｰｽのｼｽﾃﾑ機能強化と最適化の追求
			納入/契約仕様の実現	設計、納入品の瑕疵責任	ｼｽﾃﾑ設計開発の合理化/ｺｽﾄ低減	綺麗な構造設計/ｺｽﾄ低減	ｼｽﾃﾑ設計開発の合理化/ｺｽﾄ低減
A	ﾋﾞｼﾞﾈｽ機能関連図				IS部門で企業/事業全体機能定義	IS部門で企業/事業全体機能定義	IS部門で企業/事業全体機能定義
B	ﾋﾞｼﾞﾈｽ連携図				業務と対外系/他部門間との連携	業務と対外系/他部門間との連携	業務と対外系/他部門間との連携
C	ﾋﾞｼﾞﾈｽﾙｰﾙ定義書				企業/業務上の戦略ﾙｰﾙ	企業/業務上の戦略ﾙｰﾙ	企業/業務上の戦略ﾙｰﾙ
D	ｼｽﾃﾑ化目標定義書		業務ｼｽﾃﾑ化の目標設定	業務ｼｽﾃﾑ化の目標設定	業務ｼｽﾃﾑ化の目標設定	業務ｼｽﾃﾑのITﾍﾞﾈﾌｨｯﾄ定義	業務ｼｽﾃﾑのITﾍﾞﾈﾌｨｯﾄ定義
1	ﾋﾞｼﾞﾈｽ機能構成表		ﾋﾞｼﾞﾈｽ機能の大分類定義	ﾋﾞｼﾞﾈｽ機能の中小分類定義	ﾋﾞｼﾞﾈｽ機能の細分類定義	ﾋﾞｼﾞﾈｽ機能の細分類定義	ﾋﾞｼﾞﾈｽ機能の細分類定義
2	ﾋﾞｼﾞﾈｽﾌﾟﾛｾｽ関連図			ﾋﾞｼﾞﾈｽﾌﾟﾛｾｽ間の関連定義	ﾋﾞｼﾞﾈｽﾌﾟﾛｾｽ間の関連定義	ﾋﾞｼﾞﾈｽﾌﾟﾛｾｽ間の関連定義	ﾋﾞｼﾞﾈｽﾌﾟﾛｾｽ間の関連定義
3	業務流れ図				業務処理ﾌﾛｰ指示（含む例外処理）	業務処理ﾌﾛｰ指示（含む例外処理）	業務処理ﾌﾛｰ指示（含む例外処理）
4	機能情報関連図					DFD方式での上位DFDとして作成	DFD方式での上位DFDとして作成
5	業務ﾙｰﾙ定義書					業務処理上の社内ﾙｰﾙを定義	業務処理上の社内ﾙｰﾙを定義
6	個別業務処理定義書					各個別の業務処理手順を定義	各個別の業務処理手順を定義
7	画面/帳票一覧				基本的に必要な画面/帳票一覧	基本的に必要な画面/帳票一覧	基本的に必要な画面/帳票一覧
8	画面/帳票ﾚｲｱｳﾄ				画面/帳票ﾚｲｱｳﾄを定義	画面/帳票ﾚｲｱｳﾄを定義	画面/帳票ﾚｲｱｳﾄを定義
9	ﾃﾞｰﾀ項目定義書						ﾃﾞｰﾀ項目の属性を定義
10	運用・操作要件書				業務ｼｽﾃﾑの運用・操作の条件設定	業務ｼｽﾃﾑの運用・操作の条件設定	業務ｼｽﾃﾑの運用・操作の条件設定

注：空白部分は企業内 IT 部門 SE やベンダーSE の支援を受けて作成する。

（ビジネスシステム定義研究 2004）

図表 3-2-3　業務担当部門による業務システム仕様書の記述レベル

	記述項目	利用目的	承認者／利用者	見積仕様作成上の利用
A	ビジネス機能関連図	①IS部門が全社のビジネス機能を把握 ②IS部門が全社のビジネス機能のシステム化度合把握	IS部門長 IS企画担当	全社システム上の関係
B	ビジネス連携図	①開発対象に関係する社内外の責任部門との関係把握 ②対象内の責任分担組織間の商流、物流、金流の関係把握	業務担当部門長 業務担当者	ビジネスルールの把握
C	ビジネスルール定義書	①社内外の責任部門とのビジネスルールの把握 ②商流、物流、金流上の企業規範ルール	業務担当部門長 業務担当者	ビジネスルールの整理 （通例、例外のルール）
D	システム化目標定義書	①情報システムの目的と狙いの明確化 ②情報システムの投資効果の発揮方法の作り込み方針	業務担当部門長 IS部門長	情報システム基本設計指針 （契約の基本条件）
1	ビジネス機能構成表	①対象に関連する全てのビジネス機能の整理把握 ②各ビジネス機能のシステム化方針の設定	業務担当部門長 業務担当者	業務システムのIT化対象指定 （見積依頼範囲の指定）
2	ビジネスプロセス関連図	①対象の業務システムの概要、基本構成の整理把握 （詳細に整理する時は、「業務流れ図」または「機能情報関連図」をしようすることが望ましい）	業務担当部門長 業務担当者	基本業務処理の構成指定 （見積依頼範囲の指定）
3	業務流れ図	①対象業務の組織とビジネス（処理）機能の基本関係把握 （業務担当者が現状の業務処理（As-Is）を記述し、業務改革／改善を行いTo-Beを作る）	業務担当部門長 業務担当者	業務システム基本設計条件 （見積依頼範囲の指定）
4	機能情報関連図	①ビジネスプロセス間の基幹ビジネスデータの関係を把握（ビジネスプロセス関連図上に基幹ビジネスデータを追記し作成が可能）（「業務流れ図からIS側が作成）	業務担当部門長 業務担当者	業務システム基本設計条件 （基本システム構造条件）
5	業務ルール定義書	①ビジネスプロセスに準じた社内業務の意思決定ルール把握 ②取引ルール、リソース関連ルール、意思決定ルール明確化	業務担当部門長 業務担当者	業務システム基本設計条件 （納入物検収の基本条件）
6	個別業務処理定義書	①各ビジネスプロセスの業務ルール適用手順の明確化 （業務担当者はHIPO方式、フローチャート方式を利用：情報システム設計側では、UMLシーケンス等を）	業務担当部門長 業務担当者	情報システム開発条件 （納入物検収の基本条件）
7	画面／帳票一覧	①情報システムとして開発依頼する画面／帳票類を明確化 （ビジネスデータの処理、人間がビジネス／業務ルールの判断処理を行なう上の画面帳票、管理統制上の帳票を）	業務担当部門長 業務担当者	情報システム開発条件 （納入物検収の基本条件）
8	画面／帳票レイアウト	①ビジネスデータの詳細を把握整理する基本ベース ②情報システムとして作り込む画面／帳票設計の参照	業務担当部門長 業務担当者	情報システム開発条件 （アプリソフト設計条件）
9	データ項目定義書	①ビジネスデータの属性を定義 （情報システム構築依頼後のシステム構築者の設計スケジュールに合わせて依頼下の業務担当部門が承認）	業務担当者 （システム構築担当者）	情報システム開発条件 （アプリソフト設計条件）
10	運用・操作要件書	①業務システム上の操作要件、データ処理要件を明確化 ②情報システムとしての運用／保守、セキュリティ／安全	業務担当者 IS担当者	情報システム開発条件 （納品物検収の条件）

（ビジネスシステム定義研究 2004）

図表 3-2-4　エンドユーザーによる業務システム仕様書の利用目的と利用者

７．見積要求仕様書の活用と課題

JUAS 業務システム見積要求仕様書を利用した見積／調達方法を採用しようとした場合、下記の現実に直面することが予想される。

①業務部門が業務システム仕様の作成能力に乏しい

②IT 部門の方が業務システムの内容を知っている

③システムベンダーのノウハウを生かす方が良い

今までは、IT 部門が業務部門に成り代わり、業務システムの仕様作成を行い、業務システム設計構築を長く依頼してきたシステムベンダーとの良好な関係を保ちつつ、RFP（提案要求書）をベースとして上記の課題を解決してきた。

今後は、業務部門、IT 部門はお互いの甘えの構造をなくし、企業組織上での分業／協業を効果的に行う役割分担／責任分担を明確にする必要がある。特に、グローバルに事業拠点を展開する企業において経営者は、業務分担／業務責任／業務統括／業務評価を明確に行う能力を発揮しなければならない。

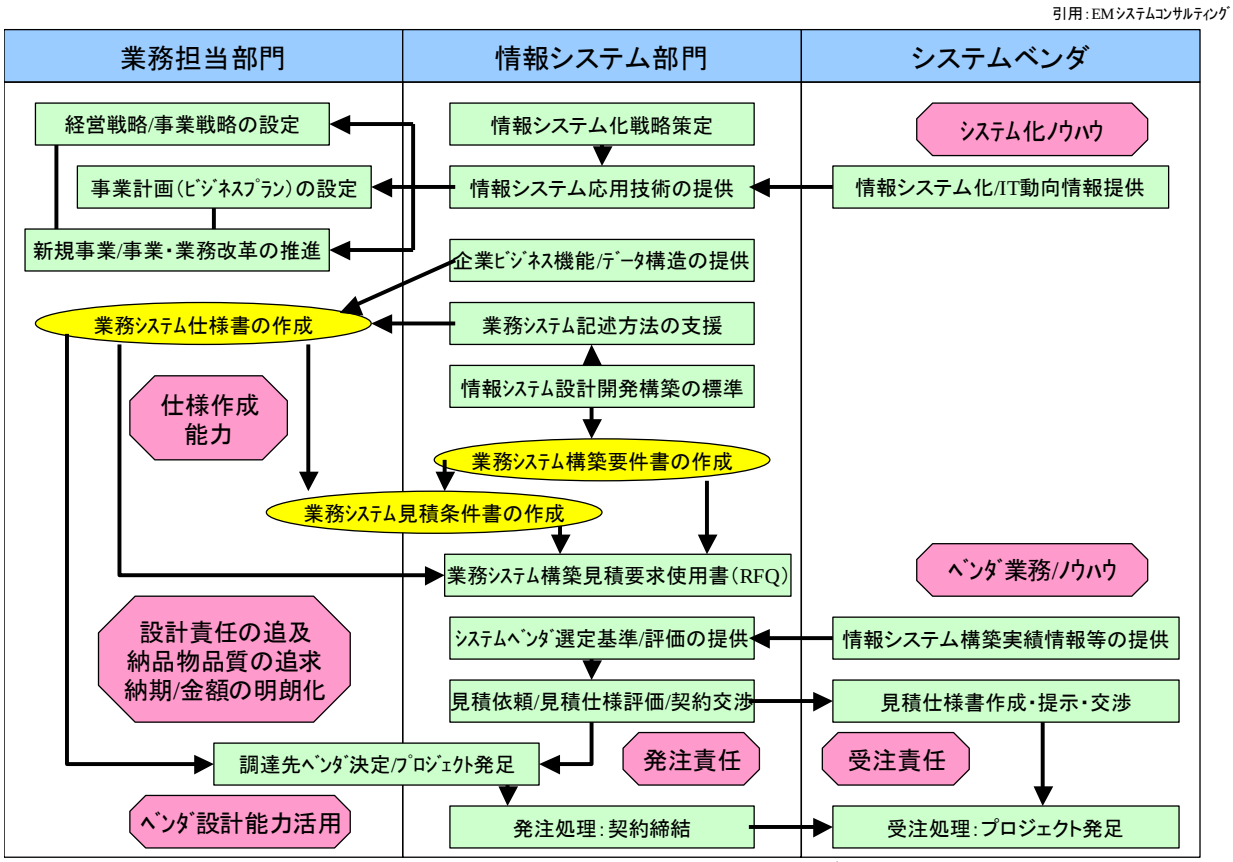
一般的に、この見積要求仕様書によるシステム設計開発構築の調達を行うことは普及していない。すなわち、調達内容を明確に指示できないがゆえにラフな RFP によるベンダー主導の契約管理／遂行を行う方式が採用されており、これを前提にソフトウェア産業や情報処理学会等で「如何に作るか」の議論がなされている。

これまでは、業務部門で明確にできない発注仕様について IT 部門がベンダーとの橋渡しを行ってきた。つまり、IT 部門とベンダーでの多くの重複作業工数をかけて設計を進めてきた。

しかし、一般的な調達において、調達側が概要を指定して発注し、それを受託したベンダーは責任を持って基本設計／詳細設計を行うのが常識である。今後は、企画仕様のレベルでの概要をベースに、基本設計／詳細設計のプロジェクトの進行を管理し、開発責任を全うできないシステムベンダーは、システムベンダーの資格がないと判断すべきである。

赤字プロジェクトが発生した場合に、ベンダー側は「ユーザーが見積仕様書をきちんと書いてくれなかった」と弁明するが、仮にユーザー側が完全に書いてしまったら、残りの作業はコード化作業のみが残り、楽しみのある作業は残らないことになる。

優れたアイデアと創造性の発揮があるからこそ SE の仕事は楽しみがある。前提条件の確認などを踏まえた上でベンダーSE の創造性の発揮も、一つの要因として認められる業務体系の構築が望まれる。

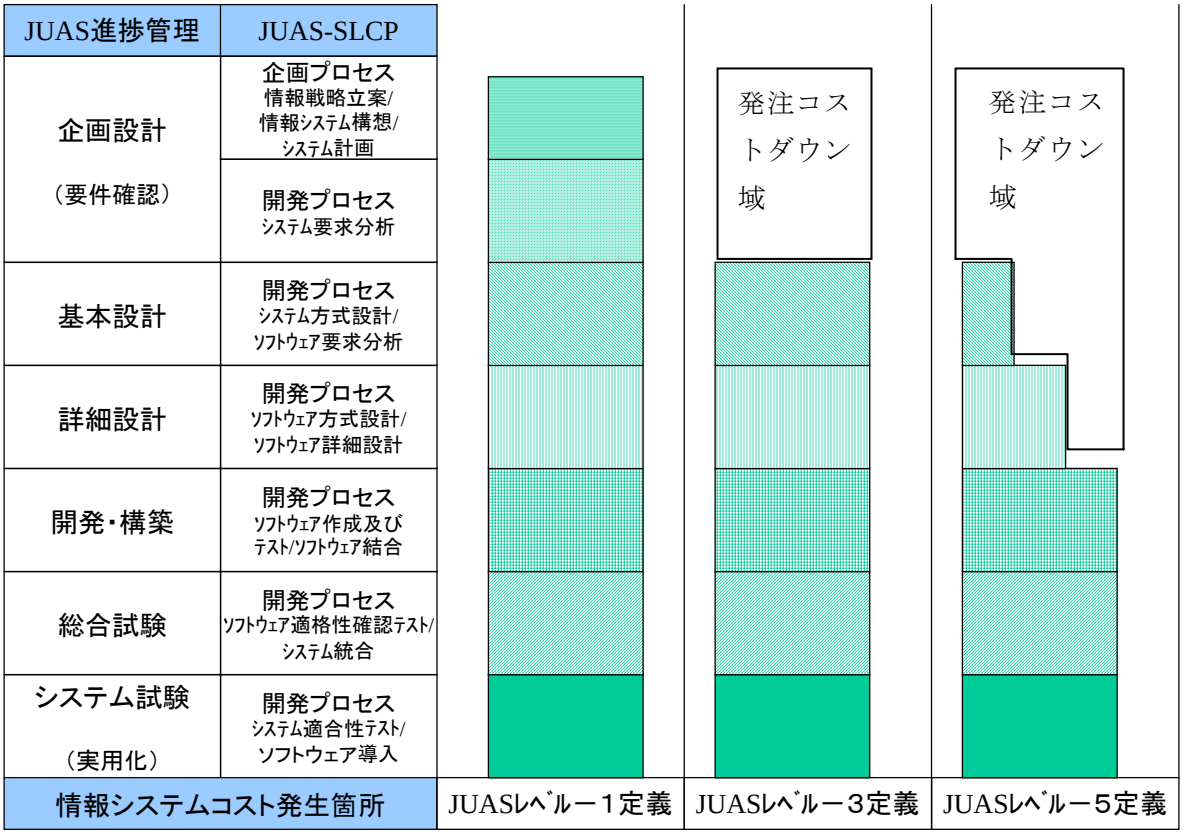


（ビジネスシステム定義研究 2004）

図表 3-2-5　JUAS 業務システムによる見積要求仕様書のいかし方と課題

８．情報システム構築コストダウンの追及

実力のあるユーザーが努力して所望するシステムの内容を詳述すればするほど、本来は高品質、安価なシステムが完成するはずである。図表 3-2-6 にケース別のコストダウン効果の関係図を示す。

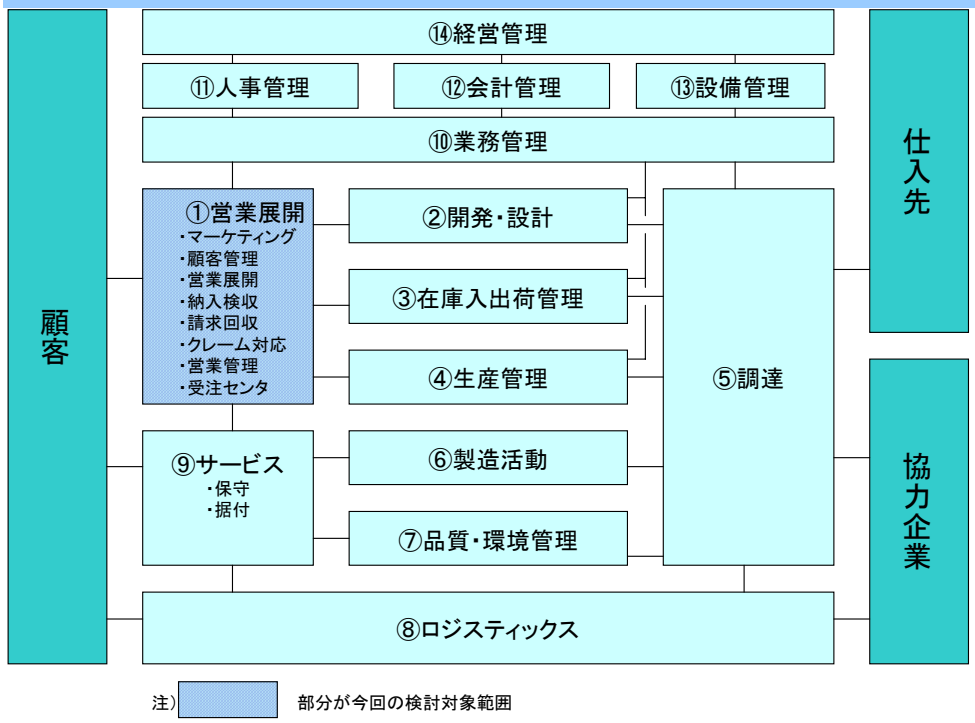


（ビジネスシステム定義研究 2004）

図表 3-2-6　JUAS 業務システム見積照会によるコスト効果

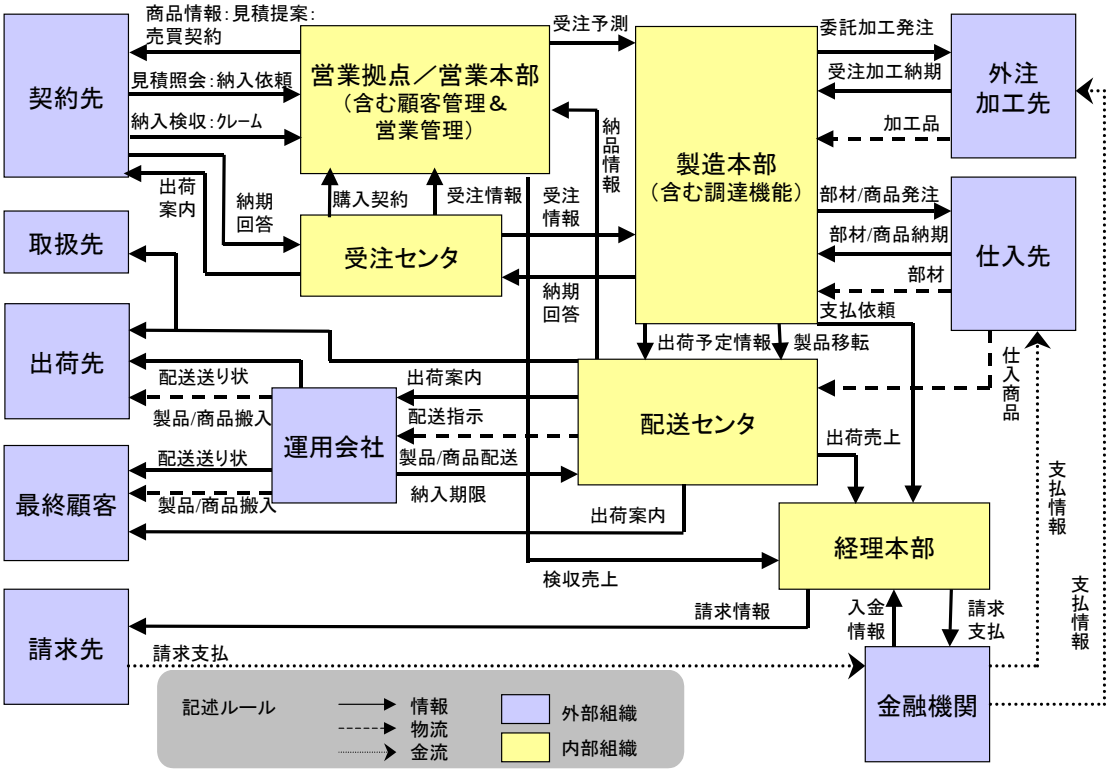
ここではコストダウンに着目したが、システム利用者が協力してシステム開発をすれば、自分たちの使いやすい、高品質のシステムが短工期で入手できるので、幅広い意味での効果は大きい。次章に一連のドキュメントの事例を収録してあるので、参考にしていただきたい。

９．ユーザーによる見積要求仕様書のドキュメント例



（ビジネスシステム定義研究 2004）

図表 3-2-7　ビジネス機能関連図（事例：製造業）全社基幹ビジネス機能



（ビジネスシステム定義研究 2004）

図表 3-2-8　ビジネス連携図（事例：製造業－営業関連機能）

分類	対象		ビジネスルール	システム化上への反映
共通	1	顧客管理	全社顧客分類／顧客識別コード体系に準拠	顧客コードの設計
	2	製品管理	製品分類／製品識別コード体系に準拠	製品コードの設計
	3	製品構成	設計BMに準拠して定義	製品構成／オプション構成の定義
管理	1	営業管理報告	販売管理規定に準拠	営業管理上の帳票一覧
	2	営業活動管理	タイムマネジメント要領に準拠	営業管理上の業務分類(ビジネスプロセス分類)
	3	機密／ｾｷｭﾘﾃｨ	機密管理規定／セキュリティ管理規定に準拠	データアクセス権限の設定
商流	1	見積	積算見積要領に準拠	積算書の作成方法
	2	与信管理	与信管理規定に準拠	取引先の与信限度管理
	3	受注内容確認	品質管理規定(ISO-9000等)の営業管理要領に準拠	受注のQCDの確定処理フロー
	4	在庫生産引当	生産管理要領に準拠	在庫引当、生産計画引当の条件
	5	納品検収支払条件	基本売買契約書に準拠	納品条件、検収条件、支払条件
物流	1	出荷	出荷管理規定に準拠	出荷製品品質確認、出荷処理フロー
	2	入荷検収	入荷管理規定に準拠	入荷製品品質確認、出荷処理フロー
金流	1	請求	販売管理／経理規定に準拠	請求書の作成方法
	2	支払	購買管理／経理規定に準拠	支払請求の処理フロー

（ビジネスシステム定義研究 2004）

図表 3-2-9　ビジネスルール定義書（製造業－営業関連業務）

対象システム名称:新営業展開システム			
システムの狙い		目標	課題
1	受注オーダ処理サービス向上	納期回答のリアルタイム化によるビジネスチャンスの拡大	納期確定方法 （生産の計画／スケジュールの確定引当方法）
		受注入力の精度向上	受注伝票上のデータの参照確認方法
2	受注予測精度向上	案件提案見積交渉の進捗管理 （受注予測）	セールスプロセスの進展と受注確度の連携
3	受注／生産／在庫／出荷上のデータ連結	受注オーダ／生産オーダ／出荷オーダ／請求オーダ間のデータ連携で顧客情報サービスと生販業務連携の強化	各オーダの付番体系の整備と結びつけ
4	全社オンライン情報化	営業／生産／物流上のビジネス情報のオンライン化	オンライン操作環境の整備

（ビジネスシステム定義研究 2004）

図表 3-2-10　情報システム化目標定義書

ビジネス機能							詳細業務活動		システム化方針
大分類		中分類		小分類			細分類		
1	営業								
		1	マーケティング						
				1	商品サービスメニュー				人間系で対応
				2	商品カタログ販促資料				人間系で対応
				3	販売価格情報準備				
				4	広告宣伝展開				人間系で対応
				5	市場／顧客開拓				
				6	販売戦略展開				人間系で対応
				7	販売促進活動				人間系で対応
		2	顧客管理						
				1	顧客データベース化				ビジネス・データ連携
				2	顧客満足度把握				人間系で対応
				3	顧客信用度調査				オンライン処理
		3	営業展開						
				1	案件開拓				オンライン処理（SFA的に）
				2	提案活動				オンライン処理（SFA的に）
				3	見積交渉				見積書作成オンライン化
				4	契約交渉				人間系で対応
				5	受注手配 （オーダ エントリ）				オンライン処理

（ビジネスシステム定義研究 2004）

図表 3-2-11 ビジネス機能構成表（事例：製造業－営業関連業務）その１

ビジネス機能							詳細業務活動		システム化方針
大分類		中分類		小分類			細分類		
		4	納入検収管理						
				1	納期管理				オンライン処理
				2	出荷納品管理				在庫・売上ﾌﾟﾛｾｽと自動化
				3	検収				人間系で対応
		5	請求回収						
				1	請求書発行依頼				バッチ処理
				2	請求回収差異分析				バッチ処理
				3	代金回収				人間系で対応
		6	クレーム対応						
				1	納品前クレーム処理				オンライン処理
				2	納品クレーム処理				オンライン処理
				3	納品後クレーム処理				オンライン処理
		7	営業管理						
				1	受注予測				バッチ処理
				2	受注実績分析評価				バッチ処理
				3	営業活動スケジュール管理				オンライン処理
				4	商品/市場/顧客情報分析				オンライン処理
				5	営業研修				人間系で対応

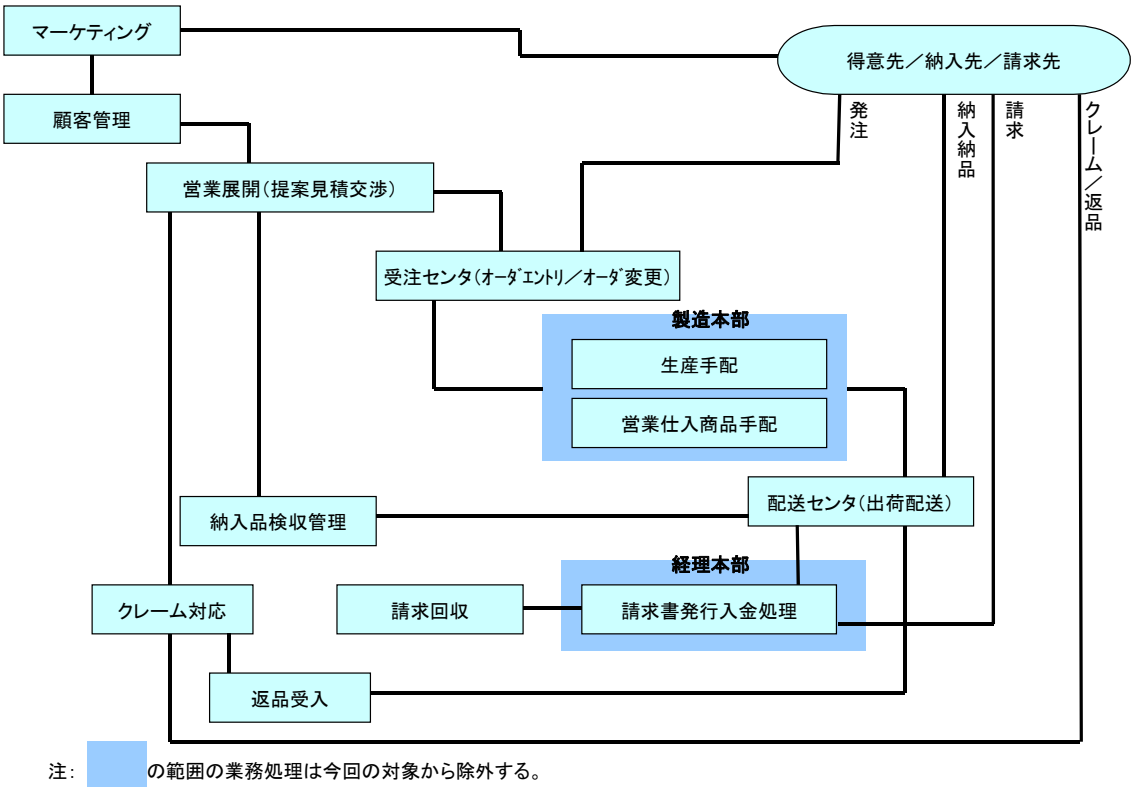
（ビジネスシステム定義研究 2004）

図表 3-2-12　ビジネス機能構成表（事例：製造業－営業関連業務）その２

ビジネス機能						詳細業務活動		システム化方針
大分類		中分類		小分類		細分類		
2	受注オーダ処理（受注センタ業務）							
		1	受注オーダエントリ					
				1	得意先オーダ受付			人間系で対応
				2	受注オーダ入力			
				3	受注オーダ納期回答			
		2	受注オーダ変更					
				1	得意先オーダ変更入力			
				2	受注オーダ変更連絡			人間系で対応
3	出荷配送（配送センタ業務）							
		1	出荷配送（配送センタ業務）					
				1	出荷計画／指示			
				2	梱包包装			
				3	配送手記			
				4	出荷			
				5	出荷基準売上計上依頼			
		2	返品受入（配送センタ業務）					
				1	納品物返品受入			
				2	返品クレーム対応依頼			

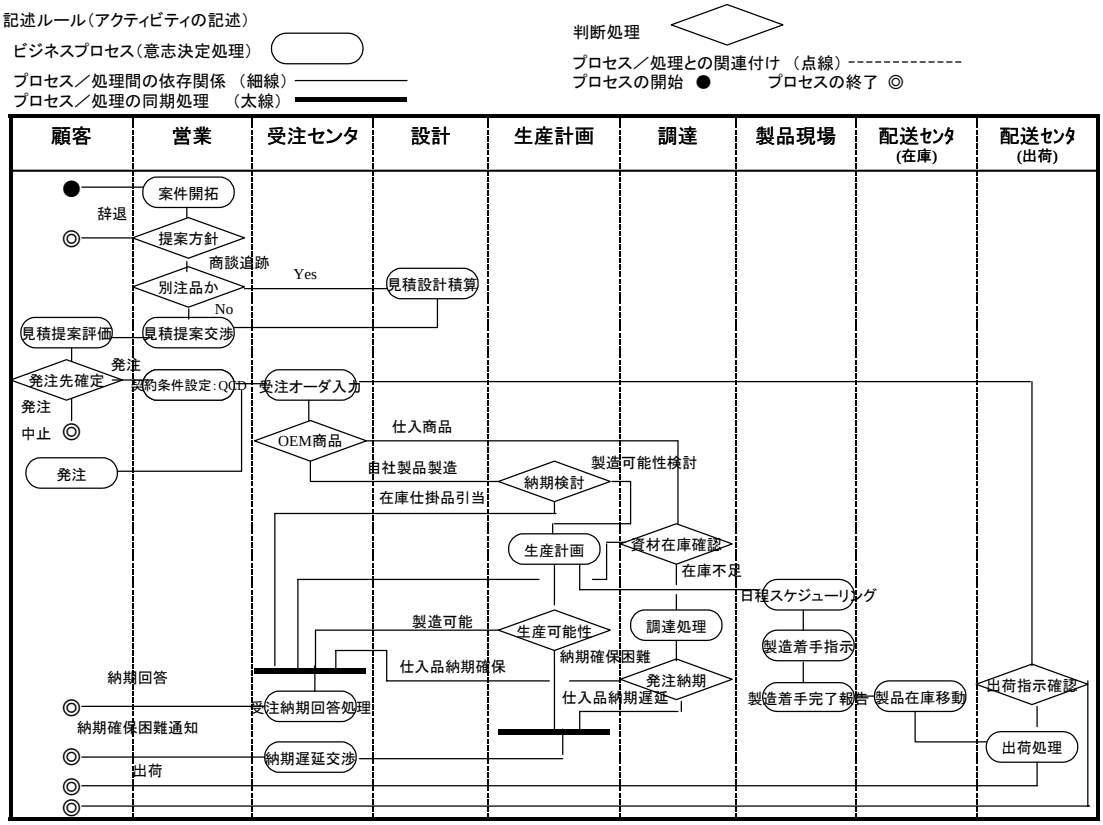
（ビジネスシステム定義研究 2004）

図表 3-2-13　ビジネス機能構成表（事例：製造業－営業関連業務）その３



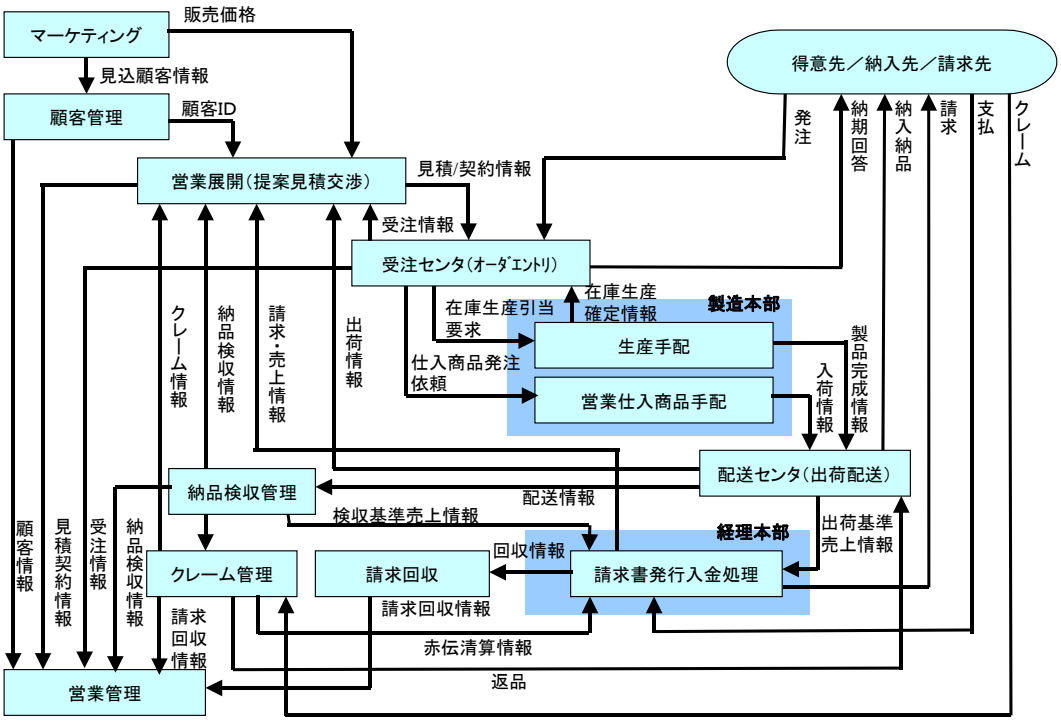
（ビジネスシステム定義研究 2004）

図表 3-2-14　ビジネスプロセス関連図（事例：製造業－営業関連業務）



（ビジネスシステム定義研究 2004）

図表 3-2-15　業務流れ図（事例：製造業－営業：営業展開／受注／納入関連業務）



（ビジネスシステム定義研究 2004）

図表 3-2-16 機能情報関連図（事例：製造業－営業関連業務）

分類	準拠ベース			通例ルール処理基準		例外ルール処理基準			システム化要求
共通									
	1		顧客管理	1	取引先企業／担当者識別コード				
	2		製品商品ｻｰﾋﾞｽ管理	1	製品／商品／サービス識別コード				
商流	1		最終顧客直販						
		1	営業区分	1	営業テリトリ判別 担当営業選定	1		グループ営業展開の指定 グループメンバの選定	最終顧客別営業担当者の設定
		2	見積提案	1	提案見積方針 見積作成依頼発行				
				2	提案内容 （標準／オプション商品）		1	別途設計し製造する商品	
				3	見積単価選定 （顧客別商品別単価選定）		1 2	特別単価の設定 必要コストアイテム見積	
				4	納期提案 （標準納期提案）		1	別途設計・製作品納期	
		3	特例契約条件	1	標準契約交渉 （標準/ｵﾌﾟｼｮﾝ商品）（標準QCD対応）		1 2	事業部決済条件の交渉 企業のリスクで交渉	人間系で対応
		4	与信審査依頼	1	顧客与信限度 (規定による与信限度)		1	将来ビジネス勘案で裁量	
		5	受注伝票	1	受注契約 (標準QCDの受注)		1	別途設計・製作のQCD	受注センタで対応
		6	納期回答	1	納入納期確約 (標準納期で受注)		1	部材、生産枠の不足調整	受注センタで対応
		7	納品検収条件	1	標準品質の工場出荷/検収		1	納品先納入後テスト検収	
		8	請求条件	1	標準品の月間納品物の請求 （月次請求書処理）		1	特注品の進行基準の請求	

（ビジネスシステム定義研究 2004）

図表 3-2-17　業務ルール定義書（事例：製造業－営業関連業務）その 1

分類	準拠ベース			通例ルール処理基準		例外ルール処理基準			システム化要求
共通									
	1		顧客管理	1	取引先企業／担当者識別コード				
	2		製品商品ｻｰﾋﾞｽ管理	1	製品／商品／サービス識別コード				
商流	1		代理店経由販売						
		1	営業区分	1	営業テリトリ判別 担当営業選定				
		2	見積作成	1	見積作成依頼受信 見積作成依頼発行				
				2	提案内容 （標準／オプション商品）		1	別途設計し製造する商品	
				3	見積単価選定 （顧客別商品別単価選定）		1 2	特別単価の設定 必要コストアイテム見積	
				4	納期提案 （標準納期提案）		1	別途設計・製作品納期	
		3	特例契約条件	1	標準契約交渉 （標準/ｵﾌﾟｼｮﾝ商品）（標準QCD対応）		1 2	事業部決済条件の交渉 企業のリスクで交渉	人間系で対応
		4	与信審査依頼	1	顧客与信限度 (規定による与信限度)		1	将来ビジネス勘案で裁量	
		5	受注伝票	1	受注契約 (標準QCDの受注)		1	別途設計・製作のQCD	受注センタで対応
		6	納期回答	1	納入納期確約 (標準納期で受注)		1	部材、生産枠の不足調整	受注センタで対応
		7	納品検収条件	1	標準品質の工場出荷/検収		1	納品先納入後テスト検収	
		8	請求条件	1	標準品の月間納品物の請求 （月次請求書処理）		1	特注品の進行基準の請求	

（ビジネスシステム定義研究 2004）

図表 3-2-18　業務ルール定義書（事例：製造業－営業関連業務）その 2

ビジネスプロセス（業務処理）名	受注オーダ処理

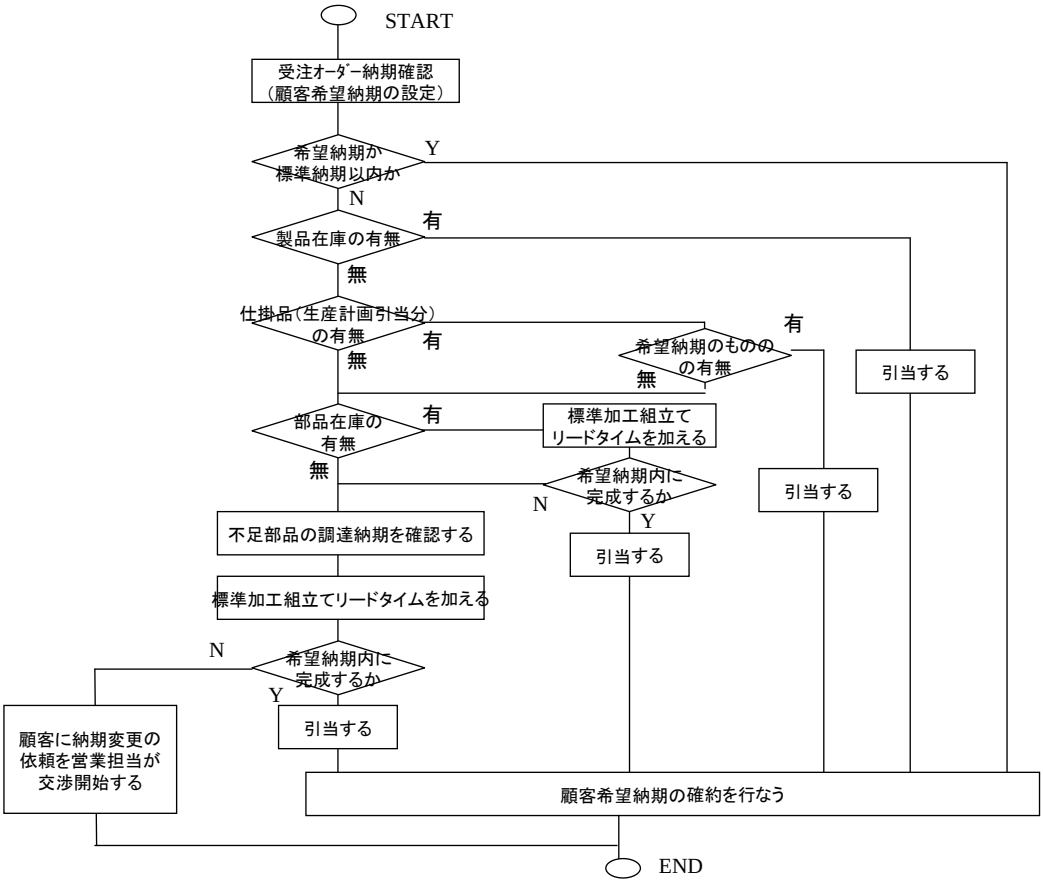
個別業務処理名	納期回答

入力／処理要求	アクティビティ（意思決定処理）	出力情報／指示伝達
（1）受注オーダ	（1）受注オーダ納期確認（顧客希望納期の設定） （2）標準納期以内であれば顧客希望納期で納期確約 （3）在庫製品であれば引当し希望納期で納期確約 （4）在庫切れの製品は仕掛品（生産計画引当分）で希望納期のものがあれば引当し納期確約 （5）部材在庫がある場合は、総標準加工組立に要するリードタイムを取り、希望納期以内に製品が完成する場合は納期確約 （6）部材不足の場合は、調達納期を確認し、総標準加工組立に要するリードタイムを取り、希望納期以内に製品が完成する場合は納期確約 （7）顧客希望納期で、部材手配、生産計画手配が出来ない場合は顧客希望納期変更要求で営業担当が交渉開始することを依頼	（1）納期回答 （2）在庫引当 （3）生産計画引当 （4）生産枠引当 （5）顧客希望納期変更要求

規程／標準／要領	参照情報
（1）受注管理要領書 （2）在庫管理要領書 （3）生産計画要領書	（1）受注実績 （2）在庫量 （3）仕掛品完成予定量

（ビジネスシステム定義研究 2004）

図表 3-2-19　個別業務処理定義書（事例：製造業－営業関連業務）（HIPO 方式による記述）



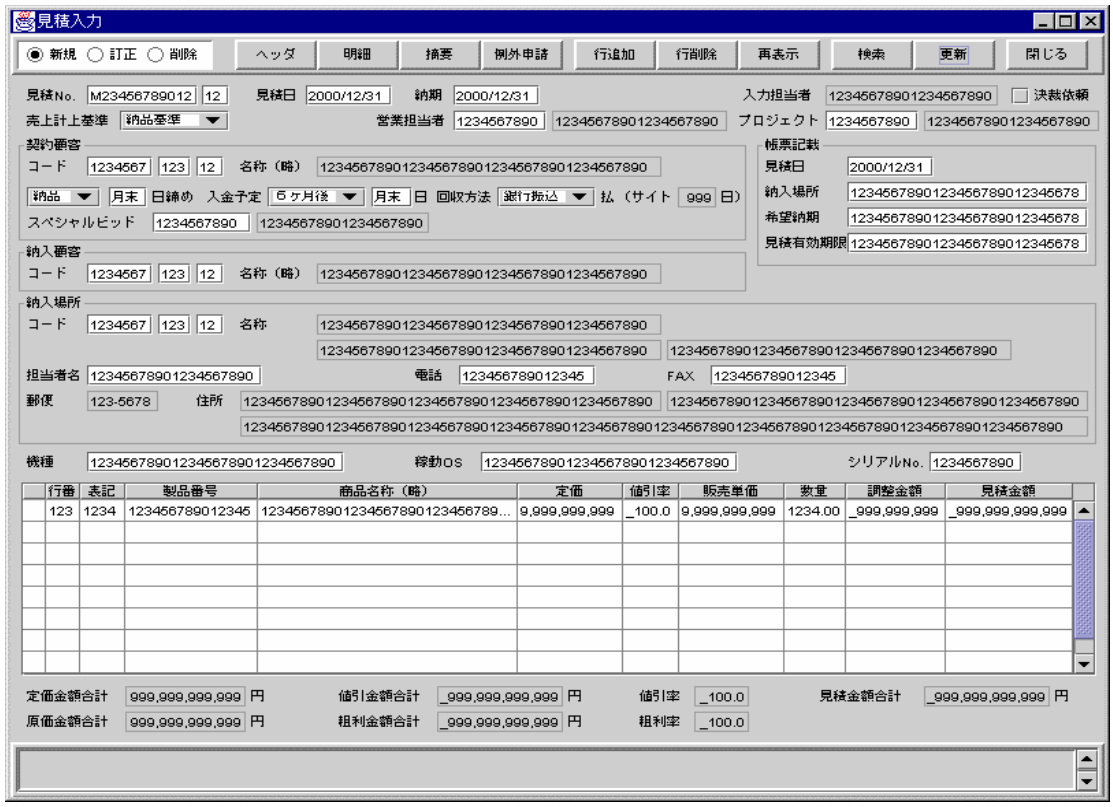
（ビジネスシステム定義研究 2004）

図表 3-2-20　個別業務処理定義書（事例：製造業－営業関連業務）（フローチャート方式による記述）

番号	画面・帳票名	目的・利用用途	利用者	タイプ	サイクル	容量（枚数）および1枚あたりのレコード件数	注記
1	受注伝票	注文内容の確認	受注ｾﾝﾀ	伝票	随時	1枚 (最大5件/枚)	
2	納期回答書	顧客向け納期回答	受注ｾﾝﾀ	伝票	随時	1	
3	出荷案内書	納入先への納入案内	配送ｾﾝﾀ	伝票	随時	1	
4	送り状	運送会社の送り状	配送ｾﾝﾀ	指定ﾌｫｰﾑ	随時	1	
5	納品書	発注者への納品案内	営業管理部門	伝票	随時	1	
6	納品検収書	納品物の検収	営業管理部門	伝票	随時	1	
7	請求書	請求元への請求	会計部門者	伝票	月次	1	月次未払清算請求
8	クレーム伝票	顧客クレーム識別	営業担当者	伝票	随時	1枚 (1件1葉)	返品理由／原因分析

（ビジネスシステム定義研究 2004）

図表 3-2-21　画面・帳票一覧（事例：製造業－営業関連業務）



（ビジネスシステム定義研究 2004）

図表 3-2-22　画面レイアウト

御見積書

下記のとおりお見積もり申し上げますので御査収ください。

記

御見積金額合計　　¥210,000-（消費税込）

見積書番号：XXXXXXXX
御見積日：2004年 3月 15日

AAA株式会社
第一営業部 部長　○○○○
営業担当　　△△△△
TEL　03-1111-2222
FAX　03-3333-4444

納入顧客：BBBB株式会社様
納期：別途ご相談
御支払条件：納品　月末締め　翌月末銀行振込み

納入場所：御社ご指定場所
御見積有効期限：御見積書提出後30日間
対象商品：見積項目どおり

No	項目	標準価格	数量	金額	備考
1	アルミ製品A	¥1,000	80	¥80,000	
2	アルミ製品B				
	小計				
3	…				
	小計				
	中計				
	出精値引き				
	合計			¥210,000	

摘要：

（ビジネスシステム定義研究 2004）

図表 3-2-23　帳票レイアウト

データ項目定義

画面・帳票名	受注伝票	作成方法：

番号	データ項目名	説明（ﾃﾞｰﾀの意味）	全体桁数	表示形式	符号・小数部	編集	注
1	受付年月日	当日カレンダデート	6	数字			
2	受注番号	自動付番方式	8	数字			
3	得意先名	契約先名	30	日本語・英数			
4	得意先コード	顧客コード表検索	8	数字		ｺｰﾄﾞ検索	
5	納入先名	商品出荷先名	30	日本語・英数			
6	納入先コード	顧客コード表検索	8	数字		ｺｰﾄﾞ検索	
7	取扱先名	取扱代理店名（2次店）	30	日本語・英数			
8	取扱先コード	顧客コード表検索	8	数字		ｺｰﾄﾞ検索	
9	品目名	商品／ｺｽﾄｱｲﾃﾑ名称	50	日本語・英数			
10	品目コード	顧客コード表検索	8	数字		ｺｰﾄﾞ検索	
11	オプション指定	オプション機能／寸法	36+16	日本語・英数+英数		ﾃﾞｰﾀ編集	
12	別途注文設計仕様書番号	見積仕様書番号	8	数字		画面連携	
13	数量	品目の受注数量	6	数字	小数点以下2桁		
14	数量単位	受注品目の数量単位	2	数字		ｺｰﾄﾞ検索	
15	金額	受注品目の金額	12	数字			
16	梱包／輸送指定	梱包方式と輸送方法指定	16+16	日本語+日本語		ﾃﾞｰﾀ編集	
17	客先注文書番号	顧客先注文番号	12	日本語・英数+英数			
18	エンドユーザー名	エンドユーザー名称	30	日本語・英数			
19	エンドユーザーコード	顧客コード表検索	2	数字		ｺｰﾄﾞ検索	

（ビジネスシステム定義研究 2004）

図表 3-2-24　データ項目定義（事例：製造業－営業関連業務）

エンドユーザ操作条件						
業務システム操作条件			操作方法	要件（現状）	将来展望	留意事項
1	データ入力操作					
	1	データ入力操作環境	クライアントPC／OS／ブラウザ等 文字記号表現			
	2	トランザクション入力	オンライン入力（ｲﾝﾀﾈｯﾄ/ｲﾝﾄﾗﾈｯﾄ：ﾌｧｯｸｽ等）			
			通例データ・マニュアル入力方法			
			例外データ・マニュアル入力方法			
	3	マスタ・データ入力	マスタ・データ・マニュアル入力方法			
	4	ファイル／DB転送	ファイル／データベース転送方法 （表形式ﾌｧｲﾙ転送/ﾃﾞｰﾀﾍﾞｰｽ転送等）			
2	アプリケーション起動方法					
	1	オンライン・アプリケーション	オンライン操作方法 アプリケーション連携方法			
	2	日次処理アプリケーション	日次処理起動条件			
	3	月次処理アプリケーション	月次処理起動条件			
	4	年次処理アプリケーション	年次処理起動条件			
	5	その他				

（ビジネスシステム定義研究 2004）

図表 3-2-25　運用・操作要件書（事例：製造業－営業関連業務）（その 1）

システム運用要件					
業務システム運用条件		運用処理方法	要件（現状）	将来展望	留意事項
1	オンライン・リアルタイム処理				
	オンラインシステム項目：	オンライン処理起動方法			
		オンラインデータ入力方法			
		オンライン入力レスポンス			
		その他			
2	日次運用条件				
	日次運用業務システム項目：	日次処理起動方法			
		バックアップ処理			
		日次ﾌｧｲﾙ／DB更新／ガーベッジ処理方法			
		日次処理スループット			
		その他			

（ビジネスシステム定義研究 2004）

図表 3-2-26　運用・操作要件書（事例：製造業－営業関連業務）（その 2）

システム運用要件					
業務システム運用条件		運用処理方法	要件（現状）	将来展望	留意事項
3	月次運用条件				
	月次運用業務システム項目：	月次処理起動方法			
		バックアップ処理			
		月次ﾌｧｲﾙ／DB更新／ガーベッジ処理			
		月次処理スループット			
		その他			
4	年次運用条件				
	年次運用業務システム項目：	年度末処理起動方法			
		バックアップ処理			
		年次ﾌｧｲﾙ／DB更新／ガーベッジ処理			
		年次処理スループット			
		その他			

（ビジネスシステム定義研究 2004）

図表 3-2-27 運用・操作要件書（事例：製造業－営業関連業務）（その 3）

データ要件					
データ項目		日単位データ発生量	ピーク時対応条件	ピーク時データ発生量	システム設計要件
1	トランザクション量（発生量）				
1	受注データ	1000件 （ｱｲﾃﾑ数/伝票　5ライン）	9-10時　16時-18時 16時-18時	300件／時間	30名で入力作業
	発注データ	100件	16時-18時	50件／時間	2名で入力作業

	データ項目	ﾃﾞｰﾀ発生量(単位：日/月/年)	保管期間 (期間;月/年)	保存期間 (期間;年/永久)	データ量
2	データ量				
	受注データ	1000件／日 （1KB／件）	1年間 (ディスク上に保管)	1年経過後はアーカイバルのファイル保存	300,000件 (300GB)
	発注データ	1000件／日	3年間 (ディスク上に保管)	3年経過後はアーカイバルのファイル保存	900,000件 (900GB)
	請求データ	1000件／日	1年間 (ディスク上に保管)	1年経過後はアーカイバルのファイル保存	300,000件 (300GB)
				総データ量	1,800GB

（ビジネスシステム定義研究 2004）

図表 3-2-28　運用・操作要件書（事例：製造業－営業関連業務）（その 4）

第３章　開発

第３節　見積

ここではユーザー企業におけるシステム開発を実施する場合の見積作業の解説と注意事項を述べる。

「開発システムの見積価額＝費用＋利益+リスク」と一般的にはなっているが、ベンダーはこの合計金額しか提示せず、中味が分からない。ここではリスクの可視化を基にユーザーベンダーが双方理解しつつ良いソフトウェア開発をするための見積について考える。

・見積とはどのようなステップを踏み、何をするのか？
・見積と契約との関係はどのようにあれば良いのか？
・ユーザーは何をすれば安くて良いソフトウェア開発が可能になるのか？
・ベンダーに何をどのように要求すれば良いのか？
・工期・品質・費用の関係はどのように考えれば良いのか？
・システムライフサイクルを考えた見積がなぜ重要なのか？

などを整理している。

１．見積プロセス
２．見積のタイミングと契約の関係
３．見積手法の実態と有効活用手法
４．規模見積方式とその特徴
５．ソフトウェア開発工期見積
６．品質の見積
７．リスク見積
８．ソフトウェア開発費用の見積
９．システムライフサイクルの費用見積

１．見積プロセス

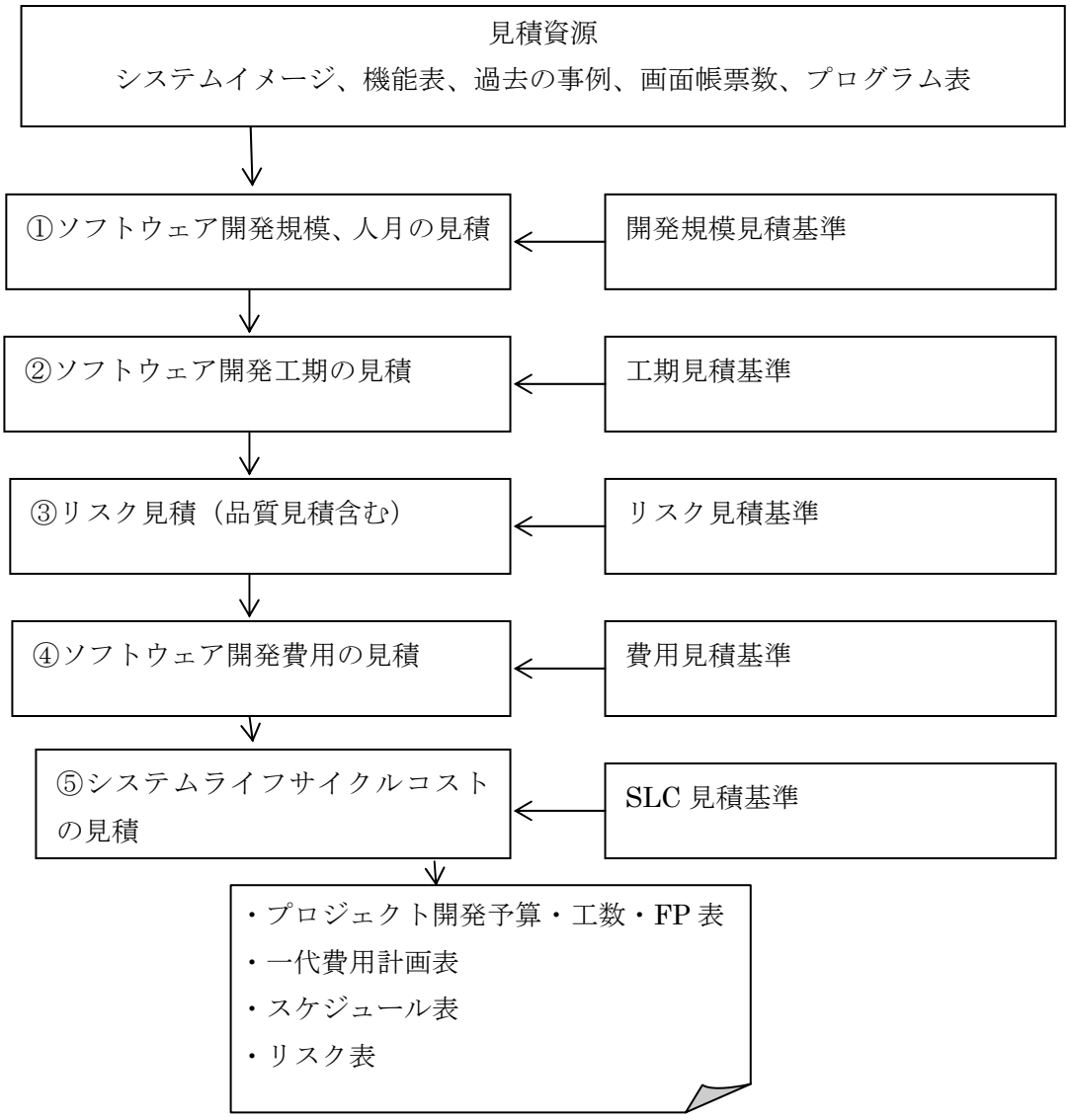
ソフトウェア開発の見積は、以下の 5 つのステップに分けて行われる。

見積資源から開発規模（人月，FP，LOC)、工期を推定し、それにリスクを加えることになる。企画初期はシステムのイメージから見積作業は始まり、開発が進むにつれて徐々に見積資源は詳細に且つ正確になって行く。あわせてリスク見積も徐々にリスクが減少してゆく。このリスク見積はベンダーの協力に依存するところが大きいが、ユーザーとベンダーの間で相互努力して開発負担を下げてゆかねばならない。

そのための透明性を持ったリスク見積基準が必要となる。

その結果で初期費用を見積るが、最終決定はシステム寿命を配慮したシステムライフコストを算出し、投資判断の是非と開発方法、発注先を決定せねばならない。

なお、ハードウェアの見積については別途論じたい。



図表 3-3-1　見積プロセス

２．見積のタイミングと契約の関係

規模、内容に応じて変化するが、一般的には下記の手法となる。大規模システムの場合には開発フェーズ別に予算見積を行い、契約をしなおすことによりユーザーとベンダー間のトラブルを避けることが出来る。ユーザー側はできる限り一括見積でリスクを下げようとし、ベンダー側は「内容が判らないものを見積ることは出来ない」と一括見積を躊躇する。規模内容を一定に抑え込める自信がある場合、「多少のリスクは我慢しても受注する」ことで顧客との関係を維持する場合などの場合以外では、一括見積は避けたほうが後のトラブルは少ない。

小規模システムでは一括契約により契約作業負担を軽減させる。それぞれに特徴があるので長所を活用すれば良い。

	一括請負契約	局面化契約（多段階契約）
前提	システム化の方向性、計画、要件定義はユーザー企業内では完了している。未完の場合は、委任契約などで要件定義を実施すること	システム化の方向など基本的考え方は決まっているが、要件定義から作業を開始しなければならない
基本的考え方	ユーザー企業はベンダーに全責任を持ってもらいたいと思っている。	開発局面ごとに契約をしなおし、ユーザー、ベンダーともにリスクを避ける 機能が確定していないものに値段はつけられない
契約時期	基本設計開始前にリスク含みの完成までの契約を行う	要件定義前、基本設計前、詳細設計前、製作前、テスト計画前、以降作業前など開発局面ごとに契約する
仕様変更	一括契約の中に含まれるとユーザーは理解している 実際は後智恵の分も含まれ、両社の協議になる	局面ごとに協議されて決定されるので、仕様変更は少ない
契約作業負担	小	大
適用プロジェクト	規模が小さめのプロジェクトに適している 大規模プロジェクトを要件が定まっていない段階にこの方式を採用するのはリスク大	大規模開発に適している リスクは下がるが、総契約金額が低下するかどうかは、別な要因で決まる

図表 3-3-2　一括契約と多段階契約

３．見積手法の実態と有効活用手法

では、見積の実態はどのようになされているのか？

JUAS の 2005 年の IT 動向調査報告書による見積方法の実態を次に掲げる。

要件定義策定時、開発着手時「新システムを開発する際、予算をどのように確定しますか？」と複数回答を求めた結果である。その結果を、情報処理業を除くユーザー企業全体 951 社と情報処理業 26 社に分けて差を明確にしてみた。％は回答した企業の割合を示す。

①要件定義策定時は過去の事例を元に算出しているが、開発着手時には具体的な見積手段に変わってくる。ユーザー企業は同じようなシステムを作成する機会は少ないが、情報処理業の会社は類似システムを開発するチャンスは多いので、自社基準などを整備しやすいし、また、持つ必要性も高い。

＜類似事例を参照して見積る割合＞

	要件定義時	開発着手時
全体（情報処理業を除く）	39.6	20.1
情報処理業	65.4	38.5

②要件定義策定時は自社の過去のプロジェクト事例を参考にし、開発着手時にはベンダーに責任を持ってもらうためにも、ベンダー見積に頼る実態が現れている。

③要件定義時からベンダーの見積に頼りすぎると、結局、高価なシステム開発になる可能性が高い。そのためにも自社見積基準を持つ必要がある。そのために業界標準を活用する

①要件定義策定時

	全体（情報処理業を除く）			情報処理業		
過去の類似事例を参照して決める	377	39.6%		17	65.4%	
概算 FP をベースに、自社標準値を元に算出する	92	9.7%		7	26.9%	
概算 LOC をベースに、自社標準値を元に算出する	38	4.0%	29.3%	3	11.5%	53.8%
画面、帳票数をベースに、自社標準値を元に算出する	149	15.7%		4	15.4%	
概算 FP をベースに、業界標準値を元に算出する	27	2.8%		0	0.0%	
概算 LOC をベースに、業界標準値を元に算出する	12	1.3%	11.3%	1	3.8%	15.4%
画面、帳票数をベースに、業界標準値を元に算出する	68	7.2%		3	11.5%	
ベンダーからの見積を元に決定する	502	52.8%		4	15.4%	
WBS	19	2.0%		1	3.8%	
あらかじめ決められた予算枠を元に決定する	197	20.7%		10	38.5%	
その他	9	0.9%		0	0.0%	
N	951			26		
無回答	55			0		

図表 3-3-3　要件定義策定時の見積根拠

②開発着手時

	全体（情報処理業を除く）			情報処理業		
過去の類似事例を参照して決める	191	20.1%		10	38.5%	
概算 FP をベースに、自社標準値を元に算出する	70	7.4%		10	38.5%	
概算 LOC をベースに、自社標準値を元に算出する	46	4.8%	31.8%	4	15.4%	76.9%
画面、帳票数をベースに、自社標準値を元に算出する	186	19.6%		6	23.1%	
概算 FP をベースに、業界標準値を元に算出する	33	3.5%		0	0.0%	
概算 LOC をベースに、業界標準値を元に算出する	16	1.7%	16.3%	1	3.8%	11.5%
画面、帳票数をベースに、業界標準値を元に算出する	106	11.1%		2	7.7%	
ベンダーからの見積を元に決定する	554	58.3%		5	19.2%	
WBS を元に見積もる	56	5.9%		2	7.7%	
あらかじめ決められた予算枠を元に決定する	200	21.0%		9	34.6%	
無回答	63	6.6%		0	0.0%	
N	951	100.0%		26		
その他	8			0		

図表 3-3-4　開発着手時の見積根拠

全体では、自社標準値と業界標準値の活用割合の差は要件定義策定時で 29.3-11.3＝18.0%、開発着手時で 31.8-16.3＝15.5 であり、どちらも自社基準を採用している割合が高い。

自社標準の準備状況を見ると、情報処理業は全体の 2 倍近い倍率で自社標準の準備が進んでいることがわかる。ビジネスの本来の姿とはいえこの面では、一般企業に比較して進んでいることがわかる。

プロジェクトの特徴、品質精度への要求の差、工期の短工期度合い、システム開発における、ユーザー企業の協力度の差、ベンダーの能力差などが各社異なるので、今回提示する業界標準を参考にして、是非、自社実績を蓄積整理し自社標準値を持たれることを望みたい。

④最下段の「あらかじめ決められた予算枠を元に決定する」方法は

「無駄なシステム機能を要求しない」

「小さく生んで大きく育てる」

「システムトラブルを最小化する」

「短工期開発を試みる」

などの効果に結びつくので、是非推奨したい方法である。

４．規模見積方式とその特徴

具体的に何を基準にして規模予算を見積るのか？下記に、各種方法を紹介する。

見積方式	概要と特徴	要件定義策定時	開発着手時
１．システムイメージ、機能表からの見積	超概算予算を計算する	◎	
２．過去の類似例からの見積	自社の過去のプロジェクトの金額を元に算出する	◎	○
３．画面数からの見積	画面数を元に予算を概算する	◎	
４．画面帳票からの見積	画面と帳票の数を元に概算する	◎	○
５．FP の見積	FP を元に概算する		◎
６．LOC の見積	LOC を元に概算する	○	○
７．WBS の見積	WBS を作成し概算する		◎
８．ユースケースからの見積	ユースケースの機能数からシステム規模を算出する		◎
９．個別プログラムからの見積	個別にプログラムを列挙し概算する		◎
１０．データ要素からの見積	データ要素数を元に概算する	○	
１１．ベンダーによる見積	ベンダーからの見積を取る	○	◎

図表 3-3-5　見積方式

１．システムイメージ、機能表からの見積

このシステム開発は 1000 万円で出来るのか？１億円かかるのか？などの判断を経営トップから求められた場合は経験値を基に超概算予算を試算する場合に使われる。プロジェクトを興すかどうか？の社内判断に活用される。

２．過去の類似例からの見積

開発内容の詳細が決まっていない開発着手時に「どのくらいの予算が必要なのか？」と推定する場合に使用する。

自社の事例を分析し蓄積しておくことが前提になる。

３．画面数からの見積

システムにおいて使用する画面数を業務流れ図、機能情報関連図などから推定し、工数、予算規模の超概算を行う。

JUAS のソフトウェアメトリックス調査結果 2005 によると

工数〔人月〕＝画面数となる。

予算〔万円〕＝画面数×90 となるが、規模・内容により修正した方が良い。

４．画面帳票からの見積

JUAS のソフトウェアメトリックス調査結果 2005 によると

工数（人月）＝０．０２×ファイル数＋０．７８×画面数+０．０７×帳票数＋０．０６×バッチ数+32

となる見積式が提案されている。

JUAS の次年度調査では、更なる精度向上を目指している

５．FP の見積

画面数、帳票数、データベース数をもとに、FP を算出する。

JUAS のソフトウェアメトリックス調査結果 2005 によると

1 人月＝22.5FP となった。別の調査でもこの係数は 20.0 から 24.0 の間に収まっているが、FP 法の基準を自社用に修正して活用しているユーザー企業が多い、各社の FP 法の計算精度が問題である、ユーザー企業の 20%程度しか FP 法は活用されておらずデータ数が少ないなどの課題を含んだ標準であることを認識し活用して欲しい。

６．LOC の見積

JUAS のソフトウェアメトリックス調査結果 2005 によると

KStep 単価＝62 万円になる。620 円／step とも読むことができる。

COBOL 言語の使用割合が多いので、言語別生産性の差などは、COBOL と各種言語を換算して活用することになる。一つのシステムで複数言語の活用が一般的に行われていること、ベンダー別に部品化され再利用が進んでいること、言語別の生産性の実態は一次請負ベンダーから二次,三次請負ベンダーへと、作業が流れてゆくので、実態を正確に把握し、分析することは非常に難しい。

７．WBS の見積

全作業を WBS に分解し、各要素の作業数を積み上げる。これに間接工数を 10%（ソフトウェアメトリックス調査 2005）上乗せして工数を算出する。

WBS のブレークダウンに作業負荷がかかる

８．ユースケースからの見積

UML 開発手法のユースケースを確定し、ここのユースケースを構成する機能をもとに工数を集計する。UML が日本ではまだ普及時期であり、事例が少ないが、各社でさまざまな試みが行われ始めている。

９．個別プログラムからの見積

基本設計でシステム概要が把握できた場合に、全部のプログラムを列挙し個別に Step 数を見積る。言語別生産性などの標準値が必要となる。

１０．データ要素からの見積

特定業種の特定業務別にみて、DB の中のデータ要素数とシステム規模が一定の関係がある場合に適用が可能になる。特定ベンダーで試みられている。

１１．ベンダーによる見積

ユーザー企業から、ベンダー各社に見積要求仕様書を提出し、公開見積をする。

業務内容を一番詳しく知っているベンダーが高い見積額になり、業務内容を熟知しないベンダーが、安い見積結果になり落札し、後で暗黙知に泣くことがある。

このあたりは技術評価を確実にユーザー側で実施し、本物を見抜く能力をつけねばならない。

以上、各種の見積手法があるが、一つの見積方法に頼らずに、いくつかの手法で見積作業を実施し、何故差が出たのか？を、確認し、前提条件に立ち戻り、見積精度を向上させることが望ましい。大規模システムの見積は慎重に複数見積を試みることが要請される。

５．ソフトウェア開発工期見積

工期標準式

プロジェクト全体工期（月）　＝　２．７×（人月）$^{1/3}$

JUAS のソフトウェアメトリックス調査 2005 年版の標準工期から計算したものである。これは COCOMO 法に則った形になっている。

「当社の工期はこの式に乗らないからこの方法は適用できない」と捨てないで欲しい。

ある程度の規模がある、50 人月以上のプロジェクトであれば、世の中の標準はおおよそこの式に乗っていることは分析結果で証明されているが、各企業により工期短縮率は相当に異なっている。「この標準工期の半分もあれば実行できる」企業も存在している。

各社各種プロジェクトがこの式に比較して何%短い工期を実施するのか？を工期短縮率として保存しておき、そのときのアクションと緊急作業度、トラブル度の関係を蓄積すれば企業としての貴重な財産になる。

（注）工期短縮率＝（標準工期-希望工期）÷標準工期

希望工期：顧客からの要望工期

工期の設定においては、新商品の販売、対コンペ戦略、株式の上場、企業の統合などのタイミング重視プロジェクトにより必要工期が確保できないことが発生する。このような場合のアクションに工期標準式とアクションの蓄積結果が役に立つ。

実際には金融業のシステム化ではこの標準工期よりも 20～30%長い工期が必要であり、流通・サービス業では逆に 20～30%短い工期の要請が多いことが経験的に判っている。なお、短工期の要請があったとしても、50%以下の工期での開発などは無謀である。

注　WBS：Work Breakdown Structure

RAD: Rapid Application Development

DOA: Data Oriented Approach

６．品質の見積

見積時点で品質を配慮し費用に反映することが難しいこと事態がソフトウェア産業の未熟さを示しているが、これが現実である。非常に良い品質を望めば高価なものになることは、皆理解しているがそれがどの程度になるのかが、不透明である。

JUAS では一つの目安として「受入（＝納入）からカットオーバーを経て安定稼動にいたるまでの間に発見される欠陥数は 500 万円に 1 件以下にしてほしい」と提唱しているが、それではこの目標を「5000 万円に 1 件にした場合、いくら費用を上積みすれば可能ですか？」とベンダー各社に問うとその答えは「20%から 5 倍までにばらつく」のが通常である。

それでは品質を高めるために、どこに費用をかけるのか？

ａ．レビューの密度を上げる

「ドキュメント作成時間の 10%はレビューにかけないと欠陥が十分に見つけられない」とする経験則はあるが、「それ以上時間をかけても見つけにくい欠陥は見つからない」ことも経験則の一つである。

ｂ．テストに時間をかける

これも有効であるが、「どれだけテスト密度を上げると、どの程度品質が向上する」経験則は公開されていない。しかし良い品質を望まれた場合にテスト密度を上げて品質向上を図ることは公知の事実である。

「10 倍のテスト仕様書を作成しテストしたら殆ど欠陥は発生しなかった」ことも経験はしたが、一般的なルールにはなっていない。

ｃ．優秀なプロジェクトマネージャーをつけて厳密な製作を行う。

これも「ソフトウェアメトリックス調査 2005」で証明された。

しかし価格／FP の値は出せていない。

以上の事実を勘案すると、現在の品質の目標を 10 倍あげた場合の費用を実績データから求めるのは、まだまだ時間がかかりそうである。

ベンダーの見積基準を比較する方が一次答案は早く入手できそうである。

JUAS では見積の中にテストの密度、レビューの密度の標準値を見込んである。

これによると、製作費用 20%、テスト費用 50%以上アップで可能となる指標となっている。

実績データからさらに検証精度を高めてゆく必要がある。

７．リスク見積

（１）システム特性を公開する見積

ユーザーからのシステム開発を受託することはベンダーにとって相当なリスクを伴う。

ベンダーにとっては

見積金額＝開発費用+利益+リスク金額

になるが、現状ではこのリスク金額は、発注側のユーザーに取って、

「どのような意味をもっているのか？」

「ユーザーが何を努力すれば安くなるのか？」は殆どの場合に霧の中である。

ほとんどのベンダーは「見積金額一式」で回答してくるので、折衝のしようがない。

ベンダーのリスク見積モデルは２種類あり、その一つはリスクを発注者にも公開して相互にそのリスクを低減する方式であり、もう一つはリスクも見積金額に内包し発注者には見えないようにする方式である。後者が現状の日本では一般的でありユーザー企業からは改善を望まれている。

リスクには、システムの持つシステム特性リスクと現存プロジェクトチームの業務能力リスクとがあり、ベンダー各社の見積りは各リスク要素ごとに試算される。

システム特性リスクとは、次の要素に分けて試算される。

・システムの先進事例　・要求仕様の網羅性　・規模の大小　・業務機能の難易度
・要求機能のレベル　・DB 構成の難易性　・移行条件　・現行保証の有無
・標準化のレベル　・レビューの充実度　・開発・テストの環境　・テスト密度
・制約条件（納期･技術）など

業務能力特性は次の要素に分けて試算される。

・個人の能力　・チームの能力　・外注企業の能力
・ハードウェア，ソフトウェアの経験度　・ツールの経験度

この業務能力特性はベンダーの費用の中に自社開発能力として織り込んでいただければよい。

残りの問題はシステム特性リスクである。このシステム特性リスクを明確にし見積時点でユーザーに公開し相互の努力によりリスクの低減を図ろうとするのが次に述べるリスク管理モデルである。

（２）ＪＵＡＳ推薦の見積モデルとの特徴

ユーザーの求めるリスクが公開される見積モデルを持っており、なお一般に内容を公開できるベンダーを求めていたところ、ジャステック社に協力していただけることになった。

ジャステックの見積りモデルは一つのモデルであり、実プロジェクトへ適用するのには数々の工夫を必要とするが、ここまでリスク管理の内容をオープンにし、可視化されたモデルは世界でも珍しい。

このジャステック社の見積りモデルは非常に優れたものであり、次の特徴を備えている。

ａ．この見積モデルは当然のことながら契約の基盤になるが、開発に入っても進捗管理の中で、このリスク要素はフォローされる。したがってユーザーとの約束の実施状況、両社の努力結果も明確にされるのでリスクとして積んだ環境変化による変動費（リスク費と呼んでも良い）の見直しが可能になる。しかし、一般的には契約金額の中に積み込まれたリスク費用はベンダー

の中での様々な対策に消費され、ユーザーに公開されることはない。

b．このリスク見積費用の中にはベンダーの生産性低下の要素は含まれていない。高価な SE を使う、生産性の低い SE を使用するのはベンダーの内責であって、ユーザーの努力外の事項である。しかし、一般の見積りモデルはベンダーの生産性低下の要素も区分けされずにコストに盛り込まれている場合が多い。その点もユーザーにとって見積り価格の妥当性を分かりにくくしている。

c．このリスク見積りはジャステックでは環境変数値の決定により実践している。さらに環境変数値の予実差異分析はリスク管理以外に、開発プロセス改善に生かされている。ユーザー協力に基づくプロセス改善は、次の見積り費用の低減に結びつく。

この見積りモデルには４０項目以上のリスク項目がある。各リスク項目はユーザーの特性による変動要因（外責と呼ぶ）とベンダーの能力による変動要因（内責と呼ぶ）とに、すべて区分けされている。この外責での変動要因の中で、ユーザーが努力すれば効果が大きく上がる項目は何なのか？

どれに注力すればどの程度の効果があるのか？

を JUAS のシステム開発生産性プロジェクトのメンバーと議論して整理したのが、下表の環境変数である。

「なんだこんな協力をすれば安くしてもらえるのか？」

「こんなことで高くなっていたのか？それなら協力するよ」

と期待するユーザーが多く出現することを期待している。

（参考）

コスト見積りモデルと環境変数

【コスト見積りモデル】

ある工程 i の生産物量をV_i、生産性P_i、標準生産物量をV^B_i、生産性のベースラインをP^B_i、で表現すると、コストC_iは次式で求まる。

$$C_i = V_i \times P_i = V^B_i(1+a_i) \times P^B_i(1+b_i)$$

但し、$a_i:(\Sigma \alpha_{ij})$　$b_i:(\Sigma \beta_{ij})$

【環境変数】

a_i、b_iは、それぞれV^B_iおよびP^B_iに対して開発環境の違いや品質要求の多寡による変動を吸収する「環境変数」と呼ぶパラメータである。
（α_{ij}:生産物量環境変数、β_{ij}:生産性環境変数）

（ジャステック資料より）

図表 3-3-6　コスト見積モデルと環境変数

（３）リスク評価表

このリスク評価表はジャステックの環境変数を基準にして、ＪＵＡＳで議論し、整理したものである。この３章の終わりに、参考資料として「生産性環境変数表」「規模環境変数表」を添付している。

リスクは最終的には、

「作成する設計書、あるいはプログラムなどの生産物の量に影響する」か、

「作業の効率に影響する」かのどちらかである。

前者を「規模環境変数」と呼び、

JIS の品質特性の機能性（４）、信頼性（３）、使用性（３）保守性（３）に関するリスク要因をあらわしている。

後者を「生産性環境変数」と呼び、

業務特性（１）、ハードウェア特性（１）、ソフトウェア特性（１）、コミュニケーション特性（４）、開発環境特性（２）、工程入力情報特性（１）、顧客の協力特性（１）、改造・再構築特性（３）、機能性（４）、効率性（２）、保守性（２）、移植性（４）に関するリスク要因をあらわしている。

「（　）内は副特性の数を表している。」

副特性ごとに評価の観点，特記特例事項を付してある。

重要なポイントは外責評価基準の影響度であり、各工程にこの副特性がどの程度の影響を与えるのかを％で表示してある。例えば「設計工程のある項目のリスクが 10%と評価されれば、その工程にかける負荷を 10%増やして見積ってある」と言う意味になる。これは経験値であり直面しているプロジェクトの特性によって影響度は変わってくる。これはベンダーに依頼してどのように評価しているのかを回答してもらえば良いが、そのための標準的な値とまずは考えていただいて良い。この方式が世の中に普及しデータが集まれば、そのときには又修正する。

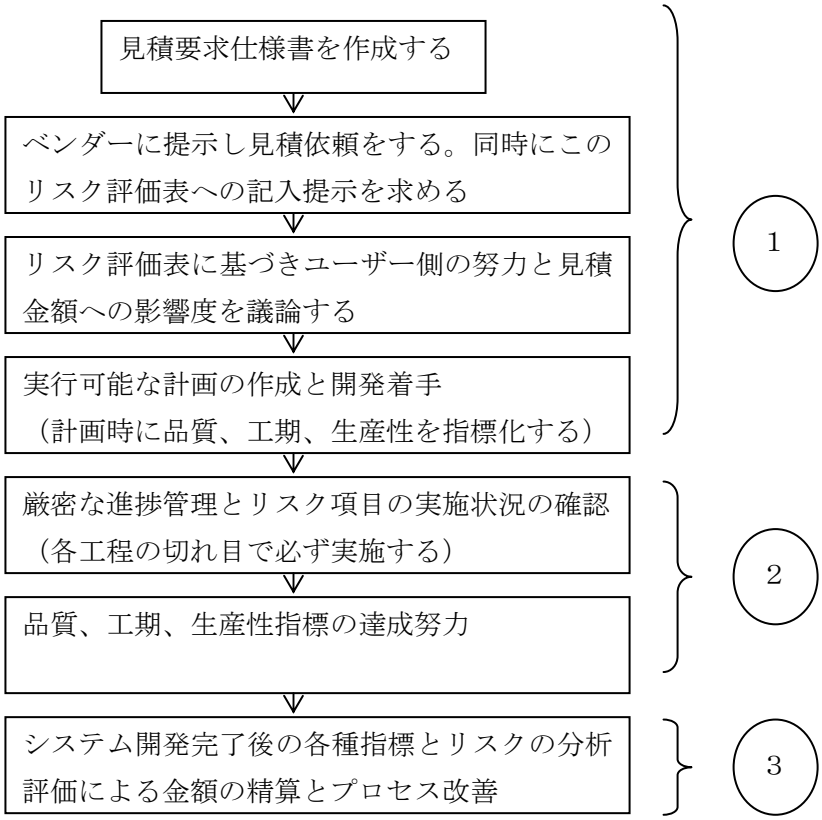
影響度の評価が低い場合は-10%，高い場合は 50%などと表されているが、これは標準的なサンプルである。リスク項目が、この項目の開発負荷に与える影響の大きさを表している。上記事例では 60%の幅があることを示しているが、中には上下の幅が 10%しかない項目もある。影響度の大きい項目を重点的に検討したほうが効率的である。

当然のことながら上流工程の要件定義、設計の段階でのリスク影響度は大きくなる。

ベンダー内の内責評価基準はユーザーと協議しても意味が無いので省いてある。

（４）このモデルの活用方法

このリスク評価表を使用する手順は以下の図（図表 3-3-7）のようになる。



図表 3-3-7　リスク評価表を使用する手順

①ユーザーが作成した見積要求仕様書に基づいて見積もり、プロジェクトの計画を作成し合意する。

このモデルでは、ユーザーが作成した見積要求仕様書を入力にして、ベンダーがリスク評価表に基づいてリスクを評価して、リスク金額とともにユーザーに公開する。

ユーザーおよびベンダーはリスクに対する認識を摺り合わせた上で、相互に協力してリスクを低減するリスク低減策を議論する。リスク評価表に記載するリスク項目は開発コストの変動要素なので、これを利用してより積極的にコストを削減する改善策を議論することもできる。次に、リスク低減策および改善策を適用することを前提として、ユーザーおよびベンダーが協力してリスクを再評価し、対策の実施に投ずるコストと対策による効果とを対比して、対策の妥当性を確認し合意する。結果として、リスク金額は小さくなり、改善策を施すことにより、見積もり金額を削減することができる。

対策の合意に基づいて、実行可能なプロジェクトの計画を作成して開発に着手するが、プロジェクトの進行中に対策の効果を確認できるように、品質、工期、生産性の目標を指標として設定して、ユーザーおよびベンダーの間で合意しておくことが重要である。

通常見積もり行為はこれから基本設計を実施する､あるいは詳細設計を実施するときに行うが、時にはユーザーが作成した見積要求仕様書を見直すこともあるかもしれない。このように、より上流の工程で見積もるときほど不確実性は高くなる。従って、契約を工程毎に区切るなど多段階に契約する方式を採用された方が、契約に含むリスク金額を低く抑えることができる。

②プロジェクトの進行中にリスクをフォローして見直すと共に対策する。

見積もりには不確実性が内在しているので、開発を開始した後でもリスクをフォローする。対策の実施状況の確認はもとより、プロジェクトの計画時に、品質、工期、生産性の目標を指標化しているので、対策の効果をより客観的に確認できる。少なくとも各工程の切れ目でユーザーおよびベンダーが相互に確認することが重要である。

相互に合意した対策が遵守されていないことが確認された場合は、当該工程に残作業が残っている可能性が高い。その時は、ユーザーは別途費用（注）を投じて残作業を片付けるか、別の方法で回避するのかを議論して進めることになるが、放置して後工程で手戻りなどの混乱によるコストが増加するのと比較すれば、安く済むことになる。

このように、ユーザーおよびベンダー双方の努力の結果は開発工程毎あるいはＷＢＳ毎に明確にされ、アクションは早く取られることになる。この効果も大きい。

（注）対策が達成されることを前提とした契約にはこの対策費用は含まれていないので、ユーザーは対策が遵守されない場合に備えて、予算を確保しておくことが肝要である。

③システム開発完了時に実績を評価して、プロセスの改善策を立案すると共に、リスクの実績を評価して精算する。

システム開発完了時には、ユーザーおよびベンダーが協力して、リスクの実績を評価する。プロジェクトの計画で設定した品質、工期、生産性の指標と実績とを対比して、その差異理由をリスク要因による「影響度」の相違および対策の妥当性とで説明する。この結果、当該プロジェクトで実施し効果を確認した対策を標準化して、プロセスの改善に活用される。ユーザ協力に基ずくプロセス改善は、次の見積り費用の低減に結びつく。

最後に、システム開発完了時に評価したリスク金額の実績を精算する。

なお、「コスト＋変動コスト（リスク金額）＋利益」で契約するのか、「コスト＋利益」で契約し、リスクが氷解した時点で、リスク金額を再配分するのかは、ユーザーとベンダーの実力によって使い分けられることになる。いずれにしても、ユーザーはリスク金額の精算に必要な予算を確保しなければならないので、ユーザーの社内稟議は「コスト＋変動コスト（リスク金額）＋利益」で了解を取っておく方が良い。

このように、本見積もりモデルを活用することは、ユーザーとベンダーとが互いの責任と役割を認識して協力することを促進する。結果として、ベンダーの健全な努力競争がなされ、ユーザーはシステム開発コストを削減することに繋がることを期待する。

８．ソフトウェア開発費用の見積

工数が算出されれば、開発費用の算出は以下の手順で行われる。

①工数×平均単価（PM・SE・PG ランク別単価や工程別単価などを掛け合わせる場合もある）

②前章のリスク管理に基づいた、リスク費用を加算する

③プロジェクトチーム内管理費用として 10%上乗せをするなど開発規模に応じて社内標準に基づいた調整をする。

④一般には販売管理費と呼ばれる社内間接要員費用を加算する

⑤利益を一定率加算する

以上の結果が見積費用となる

９．システムライフサイクルの費用見積

（１）総経費の金額で実施可否を判断する場合

2004 年度の JUAS の調査結果では、システムの寿命はスクラッチ開発された企業の基幹業務システムは 17 年、ERP パッケージの場合は 11 年となっている。

この間の保守運用費用に費やされる費用は、減価償却を 5 年間均等償却として以下の項目費用の集計となる。総投資金額をもって実施可否の決裁をする場合には、下表すべてを集計して判断した方が良い。

B：既存アプリケーションシステム利用費用						
		自社資産減価償却費用	賃貸資産使用料	自社社員経費	外部委託保守経費	賃貸資産保守費
B 1	自社アプリ償却	計上				
B 2	自社アプリ保守			計上	計上	
B 3	外部アプリ使用	計上	計上	計上	計上	
B 4	外部アプリ保守					計上

C：ITCインフラ利用費						
		自社資産減価償却費用	賃貸資産使用料	自社社員経費	外部委託保守経費	賃貸資産保守費
C 1	サーバーマシーンハードウェア	計上	計上		計上	計上
	サーバーマシーンソフトウェア	計上	計上		計上	計上
C 2	ストーレージ・ハードウェア	計上	計上		計上	計上
	ストーレージ・ソフトウェア	計上	計上		計上	計上
C 3	ネットワーク・ハードウェア	計上	計上	計上	計上	計上
	ネットワーク・ソフトウェア	計上	計上	計上	計上	計上
C 4	端末系・ハードウェア	計上	計上	計上	計上	計上
	端末系・ソフトウェア	計上	計上	計上	計上	計上
C 5	ミドルウェア	計上	計上	計上	計上	計上

Ｄ　情報システム付帯費用						
	自社資産減価償却費用	賃貸資産使用料	自社社員経費	外部委託保守経費	賃貸資産保守費	自社資産減価償却費用
Ｄ１　オフィス機器						
	コピー／ＦＡＸマシーン	条件下	条件下	条件下	条件下	条件下
	電話交換機・携帯電話機	条件下	条件下	条件下	条件下	条件下
	プレゼン機器	条件下	条件下	条件下	条件下	条件下
Ｄ２　情報システム付帯設備						
	電源設備	計上	計上		計上	計上
	空調設備	計上	計上		計上	計上
		購入費用				教育その他
Ｄ３	消耗品その他	計上				計上
Ｄ４	土地家屋					
	スペース代	条件下	条件下		条件下	条件下

Ｅ　運用費用						
		自社社員経費	外部委託費用	一般経費		教育その他隠れた費用
Ｅ１　自社管理費用						
	システム管理	計上	計上	計上		計上
	コンピュータ運用	計上	計上	計上		計上
	ネットワーク運用	計上	計上	計上		計上
	端末系運用	計上	計上	計上		計上
	ヘルプデスク	計上	計上	計上		計上
	データ入力	計上	計上	計上		計上

		ICTインフラ	アプリ保守･提供	システム管理・保守	システム運用	ヘルプデスク
Ｅ２　アウトソーシング						
	センターコンピュータ・アウトソーシング	計上	計上	計上	計上	計上
	ネットワークシステムアウトソーシング	計上	計上	計上	計上	計上
	デスクトップアウトソーシング	計上	計上	計上	計上	計上

（IT コスト研究プロジェクト 2004）

図表 3-3-8　JUAS　IT オペレーティングコスト詳細構造

（２）ＥＲＰパッケージの活用か自社開発か？

上記のハードウェアなどに関する大半の金額はほぼ同じとみなすと、保守費用の差で判断をすればよいことになる。

（Ａ）ＥＲＰパッケージの場合

ほぼ 20%の保守費用がかかるのでシステム寿命を 10 年間で見た場合は

初期費用+初期費用×0.20×10 年間＝３×初期費用　となる

（Ｂ）自社開発の場合

年間５％の保守費用が一定にかかるとすれば、

初期費用+初期費用×0.05×10 年間＝1.5×初期費用　となる。

初期費用がほぼ同じ場合は半分の経費ですむことになる。

基幹業務システムは簡単に変更出来ないので、10～20 年使用するものとして総経費を考えて開発手法を判断しなければならない。

多くのプロジェクトの場合、IT 動向調査 2005 によれば、プロジェクトの実行に際して、このシステムライフサイクルコストの配慮を必修としている企業はわずか 7.6%である。必修ではないが考慮している企業まで含めても 25.8%程度である。

目先の調達金額だけにとらわれると、システムを使い続ける間に、大きな損失を招くことになる。

選択を間違えないように調達能力を高めねばならない。

第３章　開発

第４節　契約

ソフトウェア開発プロジェクトにかかわるトラブルを追求してゆくと
①顧客との要件定義が曖昧
②計画が甘い
③プロジェクトマネジメント能力不足
④進捗状況が不透明
⑤開発戦力が不足　　その他
等の問題が浮かび上がってくるが、破綻し始めると契約との対比が必要になる。

契約書を作成してなかったことに起因する問題、契約が遅れたために起こった問題、契約が不備なために発生した問題、契約内容をプロジェクトメンバーに周知徹底させてなかったことから出た問題など各ケースに応じて、さまざまな注意事項がある。

実際の訴訟にまで発展した課題を例にとって、ソフトウェア契約についての留意点を述べてみたい。

１．ソフトウェア開発の特徴と契約の種類について
２．契約書作成のタイミング
３．契約書の作り方
４．契約内容について
５．契約内容を守るということ
６．ＩＴ訴訟
７．まとめ

１．ソフトウェア開発の特徴と契約の種類

ソフトウェア開発の特徴の一つは、業界の歴史が浅いことである。しかも、技術進歩が早く変化が激しい。開発方法や用語の定義に関してもあいまいなものが多い。情報処理技術者試験でも、基本情報技術者試験、システムアドミニストレータ試験、プロジェクトマネージャー試験それぞれのカリキュラムで開発工程の用語が異なっている。プロジェクトマネージャー試験での基本設計の工程は、システムアドミニストレータ試験では、基本設計（機能設計と概要設計）と詳細設計、基本情報技術者試験では外部設計と内部設計となっている。成果物はシステム仕様書であると契約書に記載してあったりするが、はたしてどのような内容かは明確でないのである。

特徴の二つ目は、成果物が目に見えないものだということである。完成品の全体像が捕らえにくいし、完成品の品質の確認が難しい、また完成品に瑕疵（バグ）があってもある程度許される業界の風土になっていることである。成熟した業界では、クレームやリコールの対象となりそうな品質でも完成品として納入あるいは販売されても大きな問題にならないのである。

特徴の三番目は、ベンチャー企業が多いということである。実際に開発を行うのは、大手や中堅の企業から下請けとして仕事を受けた零細なベンチャー企業が行っている。ベンチャー企業は、SE やプログラマーなどの技術者を中心に設立されることが多い。技術力はあっても法的対応、特に契約実務にまで配慮が行き届かない。ベンチャー企業は、納期と自社の開発能力との関係から契約書を作成する以前に仕事に取り掛からざるを得ない状況であったり、契約を引き伸ばされてノウハウだけを吸い上げられたり、力関係で不利な契約書を押し付けられたりなどするケースが多いのである。

ソフトウェアの開発で、開発計画の段階では全体の状況が見えにくいため、同じ発注先に何段階かに分けて発注することが多い。そのため最初に基本契約を交わして、その後、計画が固まるごとに個別契約を交わすことが行われてきた。

また、契約書に関しては「ソフトウェア開発契約書」または「業務委託契約書」などとして契約書が交わされるが、契約の内容について、その契約が請負契約としての契約か、準委任（委任）契約、共同開発かがはっきりしていないことも多い。トラブルになった時には、契約の種類によって紛争が有利になったり、不利になったり、解決に大きな影響がある。契約書を作成する場合には、契約の種類をはっきりと意識することが必要である。

委託という言葉は法律用語ではなく取引用語である。法律用語としては委任と請負しかなく、委託という用語が使用されている場合は、契約全体をみて請負契約であるか、委任契約であるかを判断しなければならない。

委任契約というのは、法律行為に使われる。法律行為ではない事務の委託がされた場合は、準委任契約といわれるが、委任契約の規定が準用されるのである。

（１）基本契約と個別契約

業務を受託することは決まったが、内容の詳細が決まらず、内容は逐次固めていくようなケースでは、最初に共通的・基本的な事項に関して基本契約を交わし、具体的な内容が決まる都度、個別契約書あるいは注文書･請書によって契約を進めることが行われている。このような取引は、大企業と小企業の間で交わされるケースに多く見られる。大企業側が作成した基本契約書に小企業がサインするだけといったケースが多いのである。契約書は作成した側に有利な契約書になっているのは当然である。

基本契約書には、全体の金額が示されていないケースもある。全体の価格が決まっている場合でも、個別契約や発注書・請書で進めていくうちに、全体の金額が当初想定した予算を大幅に上回ることになり､開発途中でトラブルとなって契約解除などの紛争になるケースが見受けられる。

基本契約と個別契約の契約方法で、注文書・請書で開発を進める方法では、委託者側は注文書を発送すればよいのである。注文書を受け取った受託者側では、遅滞なくあるいは基本契約で定めた期限内に許諾か否諾かの通知をせねばならない。継続的な取引においては、通知がない場合は、その注文を承諾したものとされてしまう。受託者としては、基本契約を締結したあとは、注文書・請書で開発を進めるとしても個別契約の成立について明らかにしておく必要がある。

注文書・請書で開発を進めているケースでの紛争が多いのは、開発内容の詳細が添付されていないからである。また、トラブルになった時に開発範囲を確定する決め手がないのである。注文書･請書で進める場合には､注文の範囲について､極力詳しく詳細まで確定しておく必要がある。

（２）契約の種類

「申立て人お聞きします。この契約は請負契約なのですか」
「はい、請負契約で受注しました」
「それでは、納期までに仕事を完成させる責任は申し立て人側にありますね」
「納期が遅れたのは、発注者の無理な仕様追加・変更が原因です」
「調停が成立しないで本審に戻された場合は、納期遅延の立証責任は申し当て人である受注者側にあります。無理な仕様追加・変更で納期が遅れたという証拠などが提出できるでしょうか」
「・・・」
「社長さん、代理人の弁護士さんとよく相談なさってください」
紛争で調停が行われた際に、このようなやり取りがしばしば行われている。
どのような契約が行われたかは、重要なことなのである。

ａ．請負契約

請負契約の目的は「定められた仕事を納期どおりに完成させること」である。請負人は発注者に対して仕事を完成させる責任がある。請負契約においては、対価を得るためには、システムを完成させなければならない。納期が遅れたのは、あるいは納品したシステムが予定通り稼働しなかったのは、請負側の責任になるのである。

発注者側は、成果物に問題があれば瑕疵修補請求や損害賠償請求が可能なのである。成果物の瑕疵やその完成遅延に対しては、原則として、受託者側が無過失でも受託者が責任を負うことになる。また、定額の請負においては、原則として経費が増加しても契約金額を変更することはで

きない。

「発注者である当社は、システム開発は素人です。だから、請負契約で全てを専門家である受注者にお任せしたのです」

「システム開発のための関連するデータや仕様固めのためにエンドユーザーにお願いしたのですが、忙しいからと言ってほとんど協力が得られませんでした」

「営業の段階では、エンドユーザーにそれほど負担がかかるという説明は受けていません。すべてを任せてくれということでした」

「でも、それではシステム開発はできません。現場の協力は常識ですよ」

「でも、私達のような素人には、プロの常識はわからないのです」

これも、調停の時によく交わされるやりとりである。発注側も大きな投資をしてリスクの伴うシステム導入の意思決定をするからには、情報システムの勉強をして十分な事前準備をしているはずである。また、業務改革のためのシステム導入なら、実際に仕事を担当している人たちの仕事のやり方が変わるのが当然であり、現場が痛みを伴うであろうことは十分予想されることである。発注側では、このようなことを承知の上で発注するはずである。しかし、発注側の仕様の追加・変更があったり、エンドユーザーが十分な協力をしてくれなかったり、といったことが原因で納期遅れや十分なテストができず品質に問題が発生したりすることがある。請負契約では、それでもトラブルになった場合には受注者側に不利になってしまう。

ソフトウェアの開発では、受注者に全てを任せればよいのではなく、発注者側の協力が不可欠である。受注者は専門家なのだから全てを任せたのであって、発注者の意図したものとかけ離れたものが納入されたとしてトラブルになるケースがまれではない。

請負契約をする場合は、特に契約段階でお互いの意識合わせを行って、合意の上で開発を進める必要がある。エンドユーザーから受注者に仕様についての注文がつけられた場合の対応について充分な詰めが必要である。受注者は、エンドユーザーの要求を客先という意識で対応して、過分な仕様追加や仕様変更に応じてしまうことがないように、プロジェクトの関係者全員に徹底しておかなければならない。後に、納期遅れや追加費用の請求などで関係を悪化させることになった場合、泣きを見るのは請負った受注者側だからである。

ユーザー企業の最終仕様の承認者は誰なのか？その承認者が認めたものだけが開発対象になること、変更管理ルールの扱いなど、基本的な取り決めを両社で取り交わし、両社内に徹底させておく必要がある。

ｂ．委任契約

委任契約とは、法律行為の委任契約のことである。したがって、ソフトウェア開発の委託契約は、民法上の契約の区分からいうと準委任契約になる。

ｃ．準委任契約

準委任契約では、受注者はソフトウェアを開発するために知識や技術といったサービスを提供することが契約の目的となる。指揮命令権は発注者にはなく、受注者にある。準委任契約においては、受注者は要求された仕事を行わなければならないという義務はあるが、システムを完成させることは約束されられていない。ソフトウェアが完成しなくても、今までに行った作業（履行

したサービスの提供）の割合に応じて対価を受けることができる。準委任契約の場合は、ソフトウェア開発のための役務の提供（委任事務の処理）を行った費用は償還請求ができる。

定額の請負契約においては、原則として経費が増加しても契約金額を変更することはできないが準委任契約の場合には、システム開発のための役務の提供のために必要な費用は償還請求ができる。

準委任契約は、受注者が受託したサービスに対する善管注意義務（受託者の業種、業務などに応じて要求される仕事を行わなければならないという義務）のみを負う契約なので、受注者に過失がない場合は瑕疵責任を負わないのである。

d．要員派遣契約

派遣契約は、ソフトウェア技術者の派遣を行い、派遣先の指揮監督のもとに業務に従事することが目的である。労働者を所定の期間、決められた場所に派遣する、労働者派遣法（労働者派遣事業の適正な運営の確保及び派遣労働者の就業条件の整備等に関する法律）に基づく契約である。

受注者は契約で定められた能力を保有する要員を派遣する義務のみを負っている。契約で定められた能力（技術力）がポイントになるので、契約には、技術力を客観的に評価できる方法を盛り込んでおくことが重要である。派遣された技術員の労務管理・監督責任は発注者側にある。指揮命令権が派遣先の発注者側にあるので、成果に対する責任は準委任契約と同様で発注者側にある。

e．売買契約

「システム導入の契約は、この売買契約書で行ったのですね」

「そうです。でもシステムは使えませんでした」

「なぜ？」

「当社の仕組みとは異なった設定になっていたからです。当社の仕組みにあわせて使えるようにするためには、カスタマイズが必要だといわれました。カスタマイズの見積を見たら、パッケージの金額をはるかに上回っているものでした」

「カスタマイズしないと全く使い物にならないのですか」

「部分的には使えるので既に使い始めています」

これでは、返品もできないことになってしまう。

この事件は、相手方（パッケージの納入側）にも、説明が不十分だったということもあって、カスタマイズ費用を割り引くことで調停が成立した。

簡単なパッケージソフトは売買契約で取引されている。売買契約では、売買の対象物の範囲、売買金額と支払方法、売買対象物の品質、解除条件とアフターサービスを決める。カスタマイズを必要としない程度のソフトウェアなら売買契約でも問題はない。

f．共同開発契約

双方が技術やノウハウを提供して、ソフトウェアの開発期間を定めて共同で開発を行う。受注者が成果物のライセンスを第三者へ譲渡するような場合には有効な方法である。

ソフトウェアの開発は、困難度が高く、開発コストの見積も難しい。また、ソフトウェア開発は、何段階かの性質が異なる工程にわかれている。全体を一括して請負契約、あるいは準委託契約にするのは、あまりに不確定要素が多くて危険である。ソフトウェア開発は、請負契約では、ソフトウェア開発で瑕疵のない完全な完成品を納入することが不可能に近い現状では、受注者側が不利になるケースが多い。面倒でも、開発段階によって契約を分けて考えるのが望ましい。また、状況によって、追加の契約をできるようにしておくべきである。要件定義書を作成するまでの開発の上流の段階では、コンサルタントなどを活用する要員派遣契約が適している。開発の場所が発注先の企業の場合は、開発側が場所だけ借りているような場合を除いては、実質的な指揮命令権を見極めて、要員派遣契約にすることも必要であろう。

また、ソフトウェア納入後に仕様追加の要求があり、発注側に納入したソフトウェアに追加したり、変更のために要員を発注側に一定期間常駐せざるを得なかったりするような場合には、準委託契約の追加や発注書・請書で処理するのではなく、要員派遣契約を別途締結して、旅費とか滞在費に関してもはっきりさせておく必要があるであろう。

トラブルが発生し紛争になった場合、指揮命令権の所在や成果に対する責任の所在がどうであったかが重要な問題になる。

２．契約書作成のタイミング

業務システムの企画がなされ、システム開発化計画が決定する。さらに、開発をアウトソーシングすることになれば、開発ツールの検討や委託先の調査が行われ開発委託契約へと進むことになる。どの段階で正式に契約し契約書を交わすか、このタイミングは非常に難しい。早い段階では、システムの全貌が確定しないため、見積りができなかったり、非常にラフな見積りになってしまったりする。発注者側では、予算が立てられない、受注者側ではリスクが大きくなる。

契約書を交わすタイミングが遅すぎると契約前に行う受注側の仕事が多くなり、受注者側に大きな負担がかかったり、ノウハウが流出したりして、受注に至らなかった時の受注者側のリスクが大きくなる。

そこで考えられたのが、基本契約と個別契約に分けて契約を行う方法である。しかし、基本契約では、契約金額が決定していなくて、仕事はそちらに発注するが、価格については個別契約で決定するといったケースも多く見受けられる。個別契約を勧めていく中で、発注側が予想していたより、はるかに多額な金額になりそうになって、トラブルに発展することもある。どのタイミングで、どのように契約を行うかは非常に難しい問題である。

トラブルを避けるためには、面倒でも何段階かに分割して契約を行う方法がある。システム調査の段階では､独立した情報システムコンサルタントと契約して､システムの概要と費用の概算、業者選定等を行う。

システム開発の概要が決定したら、契約方法を検討して数社から見積を提出させ、コンサルタントの意見を参考にして発注先を決定し契約を行う。契約にあたっては、請負契約にするのか、準委任契約にするのか、派遣契約にするのかをシステムの内容によって選択することが重要である。

運用・保守に関して、開発契約と別契約にするかどうかも微妙な問題を含んでいる。発注者側では、保守運用を別会社に依頼することも可能になるからである。開発と運用を別の会社にすることによって､マニュアル類や開発に関するドキュメント類をきちんと整備しておく必要がある。システム開発を行ったベンダーに運用まで委託させることを早めに決定してしまうとシステム開発に緊張感が欠けることになる。操作マニュアルや仕様書の修正に手抜きが生じるのである。

しかし、発注者側にとっては、保守・運用までの一括受注ができれば、手間も省けるし利益も大きくなるはずである。また、受注者側にとっても開発を行って内容までよく理解しているところに保守・運用を任せるメリットは大きい。トラブル発生時に解決までの時間が短いことも考えられる。開発契約の段階で運用や保守に対してある程度の見通しをつけておくことは、発注者にとっても受注者にとっても重要な決断なのである。

結論としては、システム開発契約は開発段階によって、派遣契約、準委任契約の使い分けが望ましい。ベンダー側からみれば、一括で契約するならば、原則委任型にすべきである。新規の開発で困難度が高く、開発コストの見積もりが困難なような場合には委任型の契約にしておく必要があるからである。ユーザー側から見れば契約を出来るだけ細分化しリスクをなくすあるいは可視化し後々のトラブルに巻き込まれないように注意しておくことが肝心である。紛争に巻き込まれるとその作業負荷は非常に大きい。

３．契約書の作り方

（１）契約書はトラブルを想定して作る

基本契約、個別契約の方式でも、見積書、発注書、請書の形式でも、請負契約でも準委託契約でも、順調に開発が行われ、テスト、検収が済んで支払が終わり、運用が順調に行われれば問題はない。

しかし、システム開発において、開発中に問題が起こらないのは奇跡に近い。十分な準備をして、仕様書も漏れがないように固め、しっかりしたプロジェクトリーダーの元に優秀な開発要員がいて開発を進めても、開発途中で様々な問題が発生するのが普通である。それでも、何とか乗り切って開発が終わり運用にこぎつけるであろう。しかし、開発途中でのリーダーやメンバーの異動、ちょっとした連絡ミスや仕様書の読み違い、思い違いなどから契約解除、紛争にまで発展するケースもまれではない。

紛争になった時のよりどころは文書である。会議議事録、打合せのメモ、電子メールでのやり取りの記録なども証拠にはなりうるが、お互いの内容を確認しあって、責任者同士がサインをした正式な契約書は効力が大きい。

そして、紛争においてもっとも重要視される契約書は、トラブルになった時のことを想定して作成するべきである。両者の関係や双方の立場・背景によって異なるので一概には決められないことも多いが、誠意を持ってとか、信義に基づいてとか抽象的な表現ではなく、できるだけ具体的に解決に至るような内容にすることが重要である。

（２）契約書や文書は自社にとって有利な内容にする

大企業は、法務の専門部署があったり、法律の専門家をかかえていたりして、充分に検討を加えた上で契約書を作成している。どちらかというと法務にまで手の回らないことが多い小さなベンチャー企業が心得ておかなければならないのは、契約書は自社にとって有利に作成されるということである。システム開発の契約は、業界の歴史が浅かったり、IT 技術の進歩が早かったりするために、契約書の雛型が十分に準備されているとは言いがたい。また、大手企業では、当社はこの契約書でなければ契約できないなどと圧力をかけてくるケースもしばしば見受けられる。対抗手段を持たない中小の IT ベンダーは、手間も省けるので大手企業が準備した契約書にサインして印鑑を押すだけで契約してはいないだろうか。

トラブルになってはじめて、こんな不利な条件で契約していたのかとホゾをかむが、それでは遅いのである。紛争は、時間も費用も半端ではなく大変な無駄であることを知って欲しい。特に小さな企業では、企業の存亡に関わることになる。受注が欲しいばかりに、相手の要求を受け入れて、紛争になった時には不利な状況で争わなければならず、後で泣かされるのである。

４．契約内容

契約書の作成にあたって、トラブルを避けるために充実した内容にしようとすると多岐にわたって多くの項目を設けることになる。

トラブルが多いのは、納期、成果物、納品と検収であり、これらに関連して支払がなされない、または支払った費用の返却を求めることで紛争になるのである。

ここでは、問題が少ない項目については割愛し、トラブルが多い項目についてのみ述べることにする。

（１）システム化の範囲

「このカタログでは、ERP の範囲として販売管理は入っていますよ」

「はい。しかし、御社の販売は独自の形態をとっているとの説明があり、契約には販売管理はパッケージではなく、独自のシステムを開発するとのことでした」

「でも、開発費の総額は販売管理を含めたものだと説明しました」

契約書は ERP とカスタマイズ一式となっている。販売管理が ERP パッケージのカスタマイズなのか、独自開発なのかでトラブルとなったのである。

契約時にシステム化の範囲を決める場合、経理処理といったときで売上の集計はどうなるのかといったお互いの用語の確認が必要である。当社ではそれはこちらのシステムで扱っている。いや当社のパッケージではこちらで扱うことになっているということでのトラブルも案外多いのである。

また、最近のトラブル事例では、カスタマイズに関するものが多い。その要求は、パッケージの基幹部分を変えるものだから変更は出来ない。いや、当社のやり方はこうだと説明してあったのだから、カスタマイズの範囲内であるという争いである。受注側では、カスタマイズの定義をはっきりさせておかなくてはならない。また発注側では、カスタマイズは総額、あるいは個別のシステムの価格の 10%以内で行うなど具体的な取り決めが必要である。

（２）納期

納期で問題になるのは納期遅れである。もともと無理を承知で受注して納期遅れをきたすケースと仕様の追加・変更が絡んでの納期遅延のケースがある。納期変更の条件について、きちんと取り決めを行っておく。個別システムごとに納期を設定してある場合には、他のシステムに連動して納期がずれていくケースが多い。そのようなケースを想定して、納期に関する取り決めをしておかなくてはならない。

特に納期遅延でトラブルになったケースでは、開発の進捗管理がしっかり行われていないことが多い。進捗管理の元になる計画表すら作成されていないこともある。納期については、契約内容を受注側、発注側共にプロジェクトメンバーに周知させ、守らせることが必要である。開発責任者は、少なくとも毎週チェックを入れて、進捗状況を把握し、発注者側に連絡を入れておくことが必要である。遅れに関して、発注者側が承知していれば、トラブルになった時に一方的に責任を負わされることはないのである。

（3）成果物

「この仕様書ではシステムを完成させられないと判断して契約解除しました」

「この仕様書では、中間金の支払はできません」

途中の成果物が不満足なもので、システムの完成が出来そうもないので契約解除に至ったという事例が何件かある。

確かに、成果物として、仕様書のボリュームは少ないように見える。しかし、その仕様書でシステムが完成できるかどうかの判断は非常に難しい。

進捗状況表､開発体制表などを製作し双方が納得して仕事を進める必要があるドキュメントは、納品物とは言えないが、成果物である。納品物以外の成果物についてもあらかじめ約束しておく必要がある。

（4）品質と検収

「納品はされました。でも、検収はしていません」

「実際に私どもが開発したシステムを使って仕事をしているではないですか」

「納品されたソフトが使い物にならなかったので、こちらで修正して使えるようにしたのです」

「連絡をくれればすぐにこちらで修正したのですが」

「納期が遅れていたので、立ち上げを急ぐ必要があったのです」

こんなやり取りもある。

本来なら、ソフトウェアであってもバグのない完全な製品を納入すべきであろう。しかし、現状では業界としての許容範囲のようなものが生きている。契約時に、バグとは何か、バグはある単位内にいくつまで許されるのかなど取り決めが必要案状況である。実際、そのような項目を入れている契約書も存在している。

どの程度の品質を望めば良いのかは、本書の「生産性・品質の指標」の章を参考にしていただきたい。

（5）保証と保守契約

「保証期間は一年間ですよね」

「それはバグなどに関してであって、環境の変化に関するものは保守契約が必要です」

「保証期間内なので保守契約が必要だとは思いませんが・・・」

「でも、そちらの要望は保守契約をいただけないと受け入れられません」

保証では、期間だけでなく保証の内容まで詰めておくべきである。保守契約についても同様である。

（6）支払

契約時の一括先払いによる支払、システムが完成して検収後に一括支払、契約時に着手金を支払い中間で中間金を支払い検収時に最後の支払を行うなど様々なケースがある。受注者側が小企業の場合、資金繰りの都合から契約時一括支払を求めるケースも多い。しかし、受注者側に債務不履行がありトラブルになった場合には、負債の回収は非常に困難である。また、システム完成後の一括支払では、支払を受けるために発注者側の無理な要求を受け入れざるを得ないケースが

出てくる。トラブルのことを考えれば、何回かに分割して支払うことが望ましい。支払時期に関しては、カレンダーで日程を決めてしまうのではなくて、成果物に見合った額を成果物が納入された段階で支払うことが望ましい方法である。

（７）その他

上記は、実際にトラブルになったケースについて述べたが、守秘義務、プログラム著作権など契約書に盛り込んでおきたい項目は多岐にわたる。

５．契約内容を守るということ

しっかりした契約書を作成しておけば、トラブルが避けられるのではない。契約書はあくまでトラブルが起きたときの最後の砦である。トラブルを避けるためには、十分検討して契約書を作成し、契約書の内容を開発チーム全員に徹底して、内容を守らせることが重要である。

実例では、しっかりした契約書を交わしておきながら、その内容がプロジェクトのメンバーに周知されていなかったためにトラブルになるケースが多い。仕様の追加・変更は契約書では責任者を通して書面を交換して行うことになっていた。しかし、開発の現場では、エンドユーザーとプログラマーとが画面を見ながら、あるいはメールでのやり取りで仕様の追加・変更が行われ、納期遅れにつながりトラブルとなった。双方の責任者は、報告を受けていないため開発の真の遅れの原因を把握していない。顧客の要望を受け入れることは必要だが、そのためには時間と費用がついて回ることを開発メンバーは意識していない。開発チームには、経験豊富な実力のあるプロジェクトマネージャーが必要とされるのはこのためである。プロジェクトマネージャーに頼れない時には、コンサルタントを活用したい。発注者と受注者に力の差がある場合には、弱い立場の側から直接要求しにくいこともある。計画の初期段階、できれば計画段階からコンサルタントを活用することが望ましい。言いにくいことや言ったら角が立ちそうなことは、コンサルタントから言わせればよいのである。

６．ＩＴ訴訟

（１）ＩＴ関係の訴訟

IT 関係の訴訟件数は、コンピュータ 2000 年問題で急増した。その後、やや落ち着きを見せているものの、年々増加傾向にある。そして、訴訟金額も大きくなっている。大企業では、法務部門があったり、総務部門の中に法務関係の部署があったりする。しかし、一般の企業の法務部門は、IT 関連の知識や経験をもった人を所属させていることはほとんどない。IT 関係の訴訟に関しては、IT 部門が相当協力しなければならない。また、法律にうといベンチャー企業が、法務部門を抱えた大企業と太刀打ちするためには、法律の専門家すなわち弁護士に頼らざるを得ない。さらに、他の事件に比べて、システムは目に見えないものだけに、その全貌を把握することが難しく、解決には長期間かかるために、多大な労力と費用が必要となる。

出来れば訴訟は避けたい。

（２）代理人（弁護士）との関係

訴訟の場合、ほとんどが弁護士に事件の解決を依頼することになる。どのような弁護士を頼むかは重要な問題である。経験豊富で実力があり、なおかつ IT に詳しい弁護士に依頼することができればよい。しかし現状では、IT 関係に詳しい弁護士は、ほとんどが若手である。経験が豊富で力のある弁護士で、なおかつ IT 関係に詳しい人はほとんどいないといってよい。力のある弁護士のいる事務所に依頼して、IT 関係に詳しい若手の弁護士と協力してもらうのがよい。その際でも、証拠の提供や準備書面の作成のため、相当な時間を割かれることを覚悟しなければならない。

７．まとめ

「契約書というのは、訴訟を想定して作成するのです。訴訟になった時に自分の方に有利になるような契約書にするのです」

契約書の基本知識に関する研修を受けた時の講師の言葉である。相手が契約書を作成してきた時は、この言葉をよく噛みしめて契約書を読んでみるとよい。必ず相手に有利にできているはずである。その部分をどうするかは、営業政策、発注者と受注者の力関係などいろいろな状況で変わってくる。自分の側にとって不利な契約になっていた場合、目をつぶって契約をしたほうがよいのか、あくまで対等に主張を行いどこか妥協点を見つけるのかを判断するのである。

次に、重要なことは、契約書の内容を確実に履行することである。発注側、受注側双方のプロジェクトメンバーに契約書の内容を周知させ、全員に契約の内容を守らせることが大切である。なぜ簡単な追加やちょっとした仕様変更でも責任者を通さねばならないのか、なぜ開発日程について厳しくチェックされるのか、担当者がお客様を大切に思うあまり無理な要求を受け入れてしまって、後でトラブルになっては意味がないのである。トラブルを避ける基本は、全員に契約内容を知らせ、メンバー一人一人が契約内容を守ろうとする努力をさせることである。

IT 技術の進歩は急速である。開発期間が長いケースでは、個別に開発していたシステムについて、開発中にパッケージが驚くほど安い価格で売り出されたりする。

契約書は、開発がスムーズに行われ、プロジェクトが成功した時は影が薄いものになる。しかし、ひとたび関係がこじれて、債務不履行、契約解除などの訴訟になった時には、大変力強い武器になることを心に刻み付けておいて欲しいのである。

第３章　開発

第５節　実行計画

システム開発プロジェクトは始めよければ全て良しには、必ずしもならないが、計画がずさんであれば結果は無惨なものになる。

実行計画の重要性から説き始め、実行計画書作成上の注意事項、ユーザーとベンダーとの役割分担、さらに新しい開発方法のＵ字型開発法を紹介する。

ここでは技術、人間、仕事のバランスについてのノウハウが入手できる。

１．実行計画の重要性
２．実行計画の作成
３．ソフトウェア開発における発注者、受注者の業務分担
４．発注者と受注者のプロジェクトマネージャーの役割
５．ＥＡ（Enterprise Architecture）とその活用
６．プロジェクトマネージャーの選抜とプロジェクトへの影響度
７．プロジェクト組織の編成とチェックポイント
８．役割分担を明確・円滑にする仕組み
９．Ｕ字型開発法
１０．ＥＶＭ（Earned Value Management）
１１．日程計画表
１２．システム仕様決定時のスケジュール

１．実行計画の重要性

プロジェクトを成功させるために、プロジェクトの開始に先立って、ユーザーとベンダーがそれぞれに、実行計画書を作成する。

特に、ユーザーとベンダー間の業務分担を適正に行うことが重要である。

IT 動向調査によれば、ユーザー企業のシステム部門のみならず、利用者は、自分たちの要求を正しく伝えるためにも、利用部門が望むシステムを入手するためにも「システムの利用者である利用部門はもっと開発に参画する必要がある」といっている。

２．実行計画の作成

プロジェクトを開始するに際して実行計画書を作成し、以下の項目について関係者の同意を得なければならない。数百万円程度の小規模かつ影響範囲が限定的なシステムの場合を除いて、以下の項目を実行に先立ち確認すること。

確認項目	注意事項
①契約範囲、時期	見積照会書の記述内容に基づきケース１からケース５のどのパターンであるかを確認すること。 基本仕様が確定するまでは委任契約（工数契約）を結び、発注者、受注者が納得できるまで内容の検討を行うこと。 システム開発の範囲、内容が確定した場合は一括請負契約とする
②予算	予算規模の縮小　予算枠の 30%カットから始める ２：４：２：３の法則がある 予算カット担当者の設定
③工期	COCOMO 法、JUAS 標準、自社標準などとの差を確認し過去の自社事例と比較を行う １週間の遅れが判断できる日程計画が作成できているか？ TRM、PAM、WBS、EVM、開発高制度などが準備されているか？
④品質	開発フェーズ別品質目標が設定されているか？ 納入以降安定稼動までの障害目標が決められているか？
⑤リスク分析	両社間でリスクについての共通認識がなされているか？ ユーザー側での努力が結果に跳ね返る仕組みができているか？
⑥PM の実力	ユーザー側プロジェクトマネージャーは業務に精通しているか？社内を動かす信念、実力をもっているか？ ベンダー側プロジェクトマネージャーは過去に同規模、同種のプロジェクトでの成功体験をもっているか？ 問題感知力、達成意欲などの人間性を備えているか？
⑦SE 戦力分析表	必要技術要素とそのレベルに対して、今回の戦力のギャップが認識されているか？ 不足分の補充策はたてられているか？（協力会社の SE，PG 含めての評価） 各人の性格、人間性の概要を把握して配置を決めているか？（ハーマンモデルなどを活用しているか？）
⑧協力会社の実力評価	このプロジェクトを共に推進するのにふさわしい企業か？担当者か？ 社全体としてのバックアップ体制はあるか？
⑨推進体制図	プロジェクトの成功はこの体制図にある。 何が課題であるか確認しているか？

⑩顧客キーマンと規模、工期についての対話	予算厳守、工期厳守のプロジェクトか？ 予算リスクは準備されてあるか？ コストオーバーを支払う習慣のある企業か？
⑪変更管理	変更管理シートごとの価額設定方式か？工数設定方式か？ 一件ごとの承認か？まとめての承認か？
⑫顧客満足度計画	プロジェクト完了時に何をもとに顧客満足度を測るのか？ 顧客満足度計画が作成されているのか？ 工期・品質・価格は基準に対してのバランスはどのような状況にあるのか？ 認識しているか？
⑬効果発生の仕組み	前提条件の確認、効果発生の仕組み、効果目標値 （KPI,ROI、US，BM など） フォローのタイミングと分担別効果発揮責任者
⑭利益計画 （ベンダーのみ）	利益金額、粗利率、営業利益、営業利益率の妥当性 直営付加価値生産性（売上－外部支出費）／直営投入人月 画面帳票別単価、FP 別単価などの計画値と達成対策を準備しているか？

図表 3-5-1　実行計画時の確認項目

・TRM：　task responsibility matrix 業務分担・責任をツリー状に示したもの（後述）

・PAM：　power analyze matrix

　　必要技術とチームメンバーの戦力ギャップをマトリックスで示したもの

・WBS：work breakdown structure　作業をツリー上に細分化したもの

・EVM：earned value management

　　作業を細分化した単位に計画と実行をフォローする仕組み

・KPI：key performance indicator 金額以外のプロジェクト評価指標

・ROI：return of investment 投資利益率、投資利益回収率　利益÷投資金額

・US：user satisfaction ユーザー満足度

・BM：bench mark 他社比較

３．ソフトウェア開発における発注者、受注者の業務分担

システム開発プロジェクトは発注者と受注者の共同作業であるから、業務分担は両者の合意を得て作成するのが望ましいが、基本的分担は下表の通りである。

両者間の契約によって、発注者が主体的により多くの業務を実行する場合もあり、また、受注者がより多くの業務の受託する場合もあるが、開発の工程毎に発注者がその工程の成果を承認して次の工程に進むと言う原則を守ることが肝要である。

項番	主要業務	発注者	受注者
1	無理のない情報化システム計画立案 （範囲、納期、規模、体制など）	◎	
2	しっかりとした要求分析・定義と文書化 （仕様変更の減少）	◎	○
3	業務に精通した開発者の選定	◎	○
4	妥当な見積（特に規模見積の精度向上）		◎
5	明確な契約の締結（発注業務範囲、分担、納期など）	◎	◎
6	正確な設計による仕様の作成	○ 設計承認	◎
7	適切なプロジェクトの推進体制 （含む、発注者側責任者の明確化）	○	◎
8	適切なプロジェクト管理	○	◎
9	データベースの設計	○ 承認	◎
１０	コード設計＆準備	◎	○
１１	品質管理体制の整備	○ 目標の提示	◎
１２	受け入れテスト、研修	◎	
１３	教育訓練資料の作成	◎ 業務関連	◎ 操作関連
１４	データ・コンバージョン作業	○ データ提供	◎ コンバージョンプログラム
１５	移行計画の作成	◎	◎
１６	変更管理	◎ 承認	◎ 整理・申請

図表 3-5-2　ソフトウェア開発における発注者と供給者の業務分担

４．発注者と受注者のプロジェクトマネージャーの役割

役割・作業分担の概要を下表に示す。ケースによって修正、追加して対応すること。

作業項目	ユーザー側プロジェクトマネージャーの役割	ベンダー側プロジェクトマネージャーの役割
開発準備	①ビジネスシステム仕様書の作成 （システム化の目的、狙い、前提条件の変革、業務変更イメージおよび要求機能、運用条件の明確化、概括予算設定） ②プロジェクトチームの編成と発足 ③本プロジェクト予算確定、システムライフ予算の作成 ④契約形態、条件設定 ⑤開発手法、成果物、納入物の決定 ⑥開発環境の設定と確保 （集中開発／分散開発などフェーズ別の環境設定、対象マシーン、ネットワーク、準備すべきソフトウェアなど） ⑦スケジュールの承認 ⑧進捗状況管理方法、レビュー方法、コミュニケーション方法の指定 ⑨主要技術の確定 （品質目標、セキュリティ対策含む）	①システム概要の確認 （システム化の目的、狙い、前提条件の変革、業務変更イメージおよび要求機能、運用条件の確認と補足、予算分解と対策） ②開発チームの選定 （プロジェクトマネージャー、基幹直営メンバーの選定、時期別協力会社の体制確保、適材適所への配慮） ③予算策定と発注者との調整 ④契約内容の確認と契約準備 ⑤開発手法、成果物、納入物の決定 ⑥開発環境の確認と自社側の準備 ⑦スケジュールの作成 ⑧進捗状況管理方法、レビュー方法、コミュニケーション方法の確認と決定 ⑨主要技術の提示 （品質目標、セキュリティ対策含む）
開発作業	①大日程計画の作成、中、小日程計画の承認 ②開発に必要な情報、資料の提供 ③ベンダーからの開発内容・仕様の提案の承認 ④システム仕様要求の調整（不要不急部分のカット） ⑤他システムとのインターフェース、自社分担分の開発推進 ⑥成果物、納入物の確認 ⑦進捗確認 ⑧変更管理の承認	①大日程計画の作成、中、小日程計画の作成と調整 ②開発に必要な情報、資料の要求 ③開発内容・仕様の作成と提案、 ④開発規模、予算の確認、調整、依頼 ⑤計画に則った開発実施と確認調整 ⑥計画通りの成果物、納入物の納入と調整 ⑦進捗管理と遅れ作業への対策 ⑧変更管理手続きの実施
移行準備	①移行計画のイメージ作成と要請 （プロジェクト開始時から） ②移行計画の決定 ③移行計画の分担作業実施 （データ・コンバージョン、利用者教育など） ④運用・保守体制などの計画の策定と推進	①移行計画の発案と予算作成 ②移行計画の詳細準備 ③移行計画の分担作業実施 （データ・コンバージョン、利用者教育など） ④運用・保守体制などの計画の提案
カットオーバーと運用	①トラブルシュート ②運用評価、効果確認と報告、対策実施 ③社内関係者、利用者の評価要請と実施 ④トラブル再発防止策の策定とノウハウの蓄積 ⑤工期・品質・生産性の評価表作成	①トラブルシュート ②運用状況評価 ③ユーザー満足度計画との対比 ④トラブル再発防止策の策定とノウハウの蓄積 ⑤予算対比（コスト実績把握）

図表 3-5-3　役割・作業分担の概要

５．ＥＡ（Enterprise Architecture）とその活用

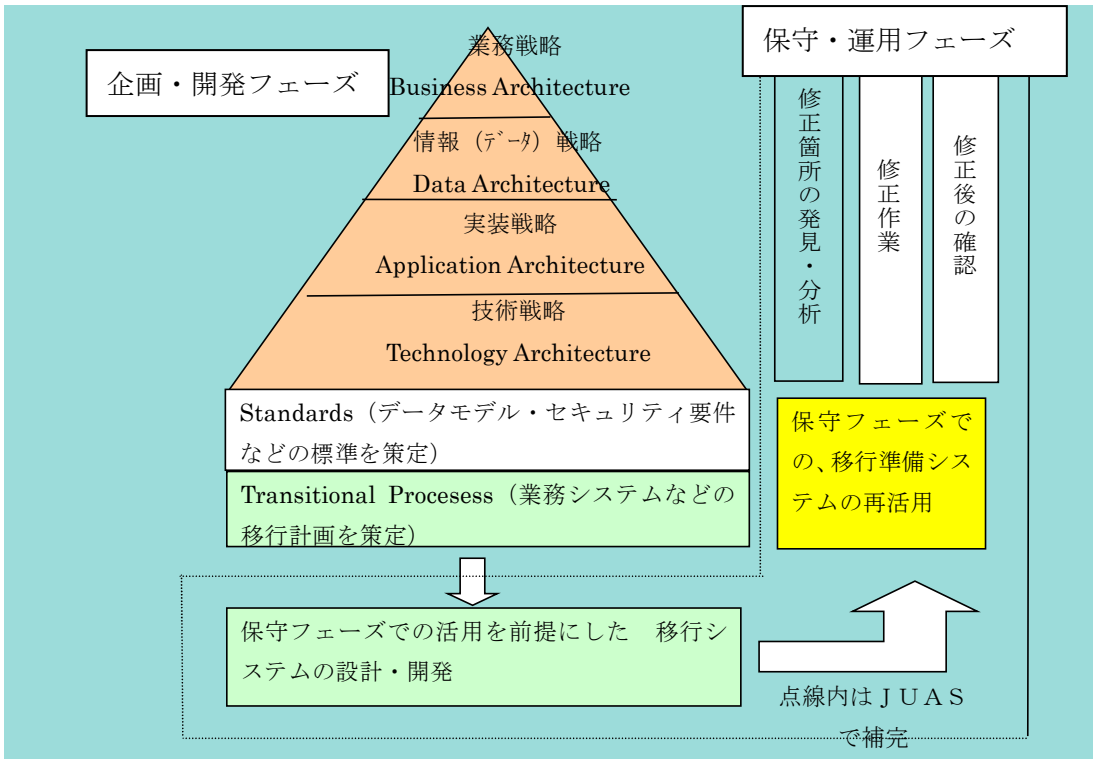
企業の基幹業務システムは、まずビジネスモデルを設定し、それを構成するデータに着目し整理する。次にアプリケーションの構造をもとにした機能をくみ上げる。これらの機能をどのようなソフトウェアで構成するのが最適なのか？将来をも考えてのソフトウェアとハードウェアの選択をする。

これが EA（Enterprise Architecture）のコンセプトであるが、これ以外に下図に示されているベースとなる重要な要素がある。

データモデル、プログラムなどの標準化、セキュリティ確保のためのさまざまな施策と最近のシステムでは非常に重要な移行計画をシステム設計の最初から考えておくことである。信頼性・安定性がシステムに対して強く望まれる時代の背景を踏まえた準備作業である。このＥＡをどのフェーズで、どのように組み合わせて作成してゆくのか？ということが　特に 24 時間稼動のシステムにおいて、問題を起さずに旧システムから新システムへ切り返るためには、この下の２項目が重要となる。

開発会社、発注ユーザー企業の腕の見せどころでもある。

なお、移行計画時に使用したツールやデータは、その後のシステム保守にも活用し、保守段階の欠陥発生防止に役立てる。



図表 3-5-4　EA（Enterprise Architecture）の概念

６．プロジェクトマネージャーの選抜とプロジェクトへの影響度

（１）顧客キーマンの影響度

顧客キーマンの資質はプロジェクトの成否に非常に大きな影響力をもたらす。特に業務仕様の決定に対しての影響力は大きく、業務に精通したユーザー側プロジェクトマネージャーがいる場合は、仕様決定の遅れは出ないことが実証されている。

	顧客キーマンの資質		
	高	中	低
システム開発のオーナー意識	高い	曖昧	無し
契約金額	必要と認めた分は承認	できるだけ安いのがよい	値切りに値切る
システム要求仕様書の定義	完全	曖昧	皆無 後付の智恵の仕様増加
ソフトウェア仕様書のチェック	約束納期厳守 緻密なレビュー	約束納期甘い 曖昧なレビュー	約束納期守れず レビューは無し
開発規模	ほぼ約束どおり	やや増加	大幅増加
仕様変更	ほとんど無し	頻発	多発
開発時の追加支払	不要、有れば、 全額支払	大幅な値切り	要求すれど、 ゼロ回答
カットオーバー	計画通り、 または、早期化	納期確保に苦戦	納期遅れ常態化
粗利率	高い	計画より低下	低い、場合によっては、赤字
運用時のトラブル	ほとんど無し	時々	頻発
運用費用	低コスト	増加しがち	高コスト
キーマンの常用言	当社分の開発責任作業は実施します	できるだけ開発会社が実行してください	当社ではITは判らないから、すべて開発会社にお任せします
完了時の発言	また次の開発もお願いします	曖昧	次は別のベンダーと開発したい
開発担当ベンダーSEの発言	また次の開発もください	（キーマンが変われば考えます）	2度と一緒に仕事をしたくない

図表 3-5-5　顧客キーマンの資質分析

上表はベンダー側から見たユーザー側の顧客キーマンの評価表である。

ベンダー側の期待と見て良い。

顧客側のリーダーとベンダー側のリーダーが気持と力をあわせて協力しないことにはプロジェクトの成功はおぼつかない

（２）プロジェクトマネージャーの影響度

> システムの欠陥率は、製作者側のベンダーのスキルに依存し、ユーザー側のスキルはほとんど影響しない
>
> （ソフトウェアメトリックス調査結果より）

「ソフトウェアメトリックス調査 2005」の調査結果によると興味ある結果が出ている。ここでの欠陥率とは、「受入（納入）検査から総合テストを経由し、カットオーバー、安定稼動にいたる間に発生した欠陥数」を意味する。

以下にその要点を示す。データの詳細は、第３章第７節の「開発生産性指標」を参照されたい。

ａ．プロジェクトマネージメントスキルと欠陥数

ベンダー側プロジェクトマネージャーのスキルが低いと、欠陥率は高くなる。ユーザー側プロジェクトマネージャーのスキルが低くても、プログラム製作には直接タッチしないので、欠陥率には影響しない。

ｂ．業務精通度と欠陥率

ベンダープロジェクトマネージャーの業務精通度が低いと、欠陥率は上昇する。

ユーザープロジェクトマネージャーの業務精通度は、欠陥率に影響しない。

ｃ．技術精通度と欠陥数

ベンダープロジェクトマネージャーの技術精通度は、欠陥率に影響が現れていない。スタッフがサポートしているものと思われる。

ｄ．ユーザー側プロジェクトマネージャーの役割、能力、特性

ユーザー側プロジェクトマネージャーの業務知識スキルが不足していると仕様確定時期の遅延や要求仕様の変更度に影響が出ることが分かる。

（３）結果が出せないプロジェクトマネージャー

「どうもプロジェクトの進捗がはかばかしくない」
「何か大変悪い結果が発生するような予感がする」
「プロジェクトの実態報告がされてこないようだ」
などの問題が発生する可能性がある場合は、思い切って以下の対策をとる。

①別の同階層のマネージャーに変わってもらう

人間には相性がある。「どうもあの人とは馬（性格）が合わない」「あの人の考えていることは、理解できない」などと、呟きが聞こえ始めたら危険な前兆である。

同じクラスのプロジェクトマネージャーに変更してみるのも一つの対策である。

②上位階層の優秀管理者にプロジェクトマネージャーになってもらう。

さまざまな理由により、現在のプロジェクトマネージャーは選出されているが、どうもお互いに良い結果が出ていないと思われる場合は、上位管理者に支援を頼むことである。

特に課長クラスのまだ経験が乏しいプロジェクトマネージャーが責任者になっている場合は、「自分の力で何とかしよう」と努力するので企業の最適なアクションを取ることがどうしても遅れる傾向がある。

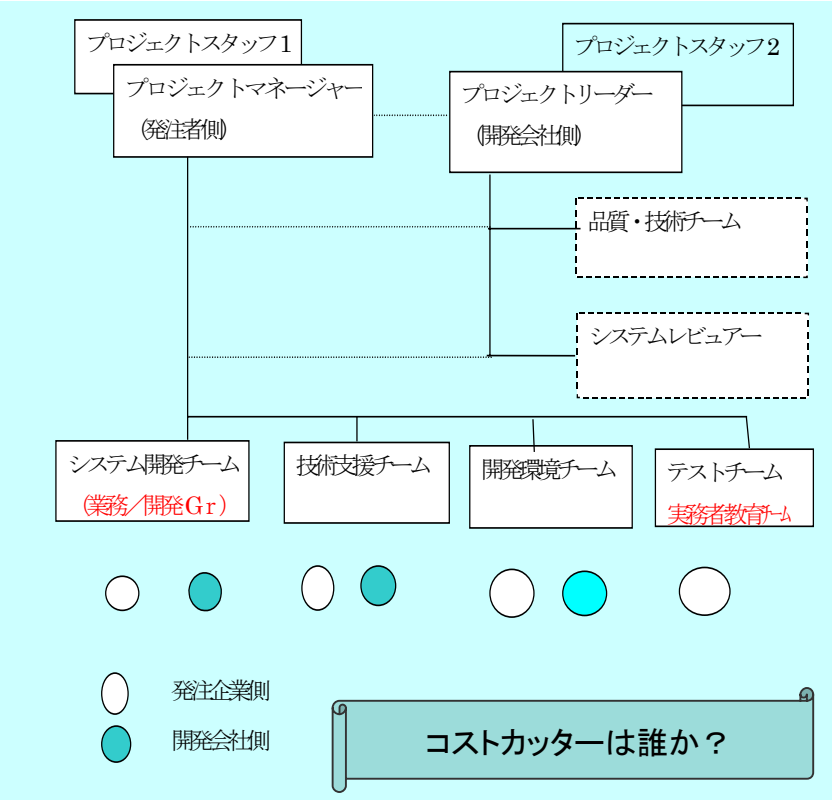
エスカレーション能力が部長、あるいはその上の管理者とは大きく異なる。

上位管理者の支援は、それまでのプロジェクトマネージャーに傷をつけなくてもすむので比較的実施しやすい。プロジェクトマネージャーの上に「総合プロジェクトマネージャー」などと命名して、上位者の出番をうまく演出することで、お互いの名誉も守られ良い結果に結びつくことが多い。

７．プロジェクト組織の編成とチェックポイント

プロジェクトチームの編成は、前記(第３章第２節業務分析とＲＦＰの作成)のレベル４，５のシステム構築が作業の中心になる場合とレベル１，２のシステムの要件定義、基本設計から始める場合とではプロジェクト体制が異なる。詳細設計以降のプログラム作成が中心になる場合は、ほぼ開発会社中心の作業体制になり、責任も明確であるが、仕様決定からはじめる場合は発注者と受注者の共同作業になることが多く、下図のような体制を作るのが一般的である。

受注者は準委任契約を結び設計作業を推進するが、実際はベンダーが豊かな経験を生かしてサポートすることが多い。



図表 3-5-6　プロジェクト体制図の例示

この体制で推進する場合の注意事項は、次のようなことがある。

①発注者側と開発会社側のプロジェクトマネージャーがいかに協力できるかにある。

片方をプロジェクトマネージャー、片方をプロジェクトリーダーと名前を変えただけで上手く行かなかったので、「両者をプロジェクトマネージャーと同じ名前にしたら上手く行きだした」ケースもある。

リーダーシップ論によれば「リーダーとマネージャーとは、大きく性格をことにするもの」であるが日本では、そのような扱いに必ずしも慣れていない。したがって同じ名前で共同作業、共同責任の方が上手く行くようである。

②システムが大規模になってくると、進捗状況の確認や新人の受入などの労務管理、教育、庶務などの作業が増えてくる。場合によってはプロジェクト全体の人数に対して１０％の人数がこの作業にかかることもある。このプロジェクトスタッフの果す役割はプロジェクト推進の潤滑役、指導役であり、重要である。

③各種作業チーム

発注者、受注者お互いに知らざるを補う設計フェーズは、同じチームに両者が属して作業することが多い。そうなると各チームのリーダーには、技術力とともに気配りなどの人間性が問題になる。後述のハーマンモデルなどを使い性格をお互いに知り合うなどの努力も要請される。

④スタッフチーム（システムレビュアーチーム）

仕様が矛盾していないか？十分か？などを検証するには、相当な経験と実力を必要とする。作成者とともにレビューをするので、実力とともにチームの皆さんから信頼される技術力と人柄が要求される。厳しい指摘をしてなおかつチーム全員から愛されるものを持っていなければならない。なお、このために使う時間はドキュメントを作成した時間の 10%程度をレビューにかける必要がある。

⑤利用者側は、ともすれば「あの機能も欲しい」「これも必要」と言いがちである。

そのような場合に「そのような機能に予算は使えない」「それは使いこなしてから必要かどうか考えよう」などと機能をカットする「カットマン」が必要となる。ユーザー企業側プロジェクトマネージャーがその役を果すこともあるが、なかなか言い出せない場合もあるので、そのような時に勇気を持って発言できる人が必要になる。

利用者部門の立場に立って「コスト対効果」を意識して主張をする人を含めて、かつその役割の重要性を認識させておかねばならない。

費用をかけて作ったがほとんど使われていない機能が多きことも事実である。

費用をかけたものは必ず利益者にコスト配布されるので、不要機能を作成しコストをかければ、そのツケは利用部門に回ってくることを忘れてはならない。

８．役割分担を明確・円滑にする仕組み

（１）ＴＲＭ（Task Responsibility Matrix）

TRM とは開発作業を推進する上で「関連する人とその責任および権限を階層型で表現したもの」である。プロジェクト管理をする上で、作業を WBS 展開する必要があるがその前提になるものである。

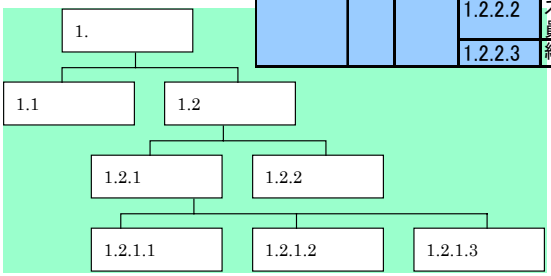
TRM の目的は人材の有効活用である。すなわち、場面に応じてもっとも相応しいメンバーに査閲や承認（中間承認含む）の役割を与えるものである。発注会社、開発会社のプロジェクトメンバーは全員これに明示し責任を分担してもらう。

特に前提条件の提示、作業実施、承認、査閲などの役割を明確しておかねばならない。

また変更管理の基準もここで確立したい。

プロジェクトの役割分担は、単なる職制、経験、学歴、性別、資格などにとらわれず、偏見を捨て、適材適所で決めなければならない。同じ人が責任者になったり単なるメンバーになったりするので、日程計画作成時には時間配分、予定には十分に気をつけねばならない。

WBS番号	作業内容	ﾏﾈｰｼﾞｬ 山口	若松	達川	長島	星野	野村 ﾘｰﾀﾞ 顧客	東尾 顧客	権藤 顧客
1	開発要求内容の妥当性分析	承認					承認		
1.2	関連情報の分析		査閲					査閲	
1.2.1	入力情報の分析			査閲				査閲	
1.2.1.1	スキルに関する情報			実施	実施				実施
1.2.1.2	給与に関する情報			実施	実施				実施
1.2.1.3	家族に関する情報			実施	実施				実施
1.2.2	出力情報分析		査閲				査閲		
1.2.2.1	部門別社員一覧表		実施			実施		実施	
1.2.2.2	スキル別社員一覧表		実施			実施		実施	
1.2.2.3	給与明細表		実施			実施		実施	



WBS と TRM の関係

図表 3-5-7　TRM(Task Responsibility Matrix)の例示

（２）ＰＡＭ(Power Analysis Matrix)

プロジェクトを成功に導くためには「おのれのチームの戦力を知る」ことが、まず成功への第一歩である。

自社の既に気心や実力が十分に分かっている仲間だけでチームを組めることは滅多にない。

大きなプロジェクトになると、チームメンバーの大半が「初顔合わせ」であることも珍しくない。自らのチームの実力も分からずに、新しい仕事を成功させねばならないリーダーの苦労は大変なものがある。

そこで、この PAM を活用することになる。

縦軸に利用技術を列挙し、その必要レベルを確認する。

横軸に担当者を並べ、「必要技術に対してどのレベルなのか？」「必要技術をチームとして質量ともにカバーしうるかどうか？」「不足技術力をどのようにして補うのか？」を検討する。日本的開発風土の、協力会社主体の戦力で開発を成功させるためには必修な分析であると思うが、意外に実施されていない。

ユーザー側、ベンダー側ともにこの表を準備することが、必要であるが、特にベンダー側はシステム構築を激しい技術変化の中で実施するので、この分析は非常に意味をもつ。

各人の評価は,まずは協力会社の代表者や本人との面接で確認することになるが、人は性格により評価基準が異なるので、重要な部分は、本人に作業をさせてみて「品質、生産性」の両方の面から確実に評価することが望ましい。

＜SE 戦力分析表＞

	Gr. 1			Gr. 2		
	必要度	氏名１	氏名２	必要度	氏名１	氏名２
１．OS						
２．ミドルウエア （ネットワーク CASE GUI セキュリティ）						
３．ハード構成力						
４．DB 構築力						
５．言語						
６．パッケージ						
７．業務知識						
８．業種知識						
９．基礎知識						
１０．プロジェクト管理力						

図表 3-5-8　PAM(Power Analysis Matrix)の例示

＜評点方法＞

全体項目について自己申請及び面接結果あるいは確認結果に基づき以下の評点をつける

・必要度　このプロジェクトを推進するに際し必要なレベルを記入しておく。

必要度１：基礎知識があればよい

２：実務で十分に使いこなしたことがある

３：ほかの人を指導する実力がある

・現状での各人の評価標準形

ａ：知識無し

ｂ：知識はあるが経験無し

ｃ：実務経験あり（経験プロジェクト数個数）

ｄ：指導する実力あり

この評価と必要度を対比し絶対レベル不足は、教育で補うのか？他のチームから支援を受けるのか？メンバーの補充をするのか？等の対策を立てる。

＜項目の説明＞

1：OS

汎用機，UNIX、LINUX、WINDOWS、その他の知識と経験

2：ミドルウェア

・開発開始前に確認すべきソフトウェアは？　安定力不足のソフトウェアは？

・今回のプロジェクトで使用するポピュラーでないソフトウェア、安定力不足のソフトウェアに着目する

・必要度とメンバーの実力にギャップがあるソフトウェアについては、あらかじめテストチームを作ってテストを実施し機能性能の確認をする、あるいはプロトタイプを作成し信頼性を高めるなどの準備を行う

3：ハード構成力

a：構成図が書ける、b：エラー解析、対応が可能、c：パフォーマンス分析が可能

4：DB 構築力

a：論理 DB の構成可能、b：物理 DB の構成可能、c：パフォーマンスチューニング可能

5：言語　標準形を利用

6：パッケージ　標準形を利用。特に類似経験プロジェクト数を重視する。

7：業務知識

顧客業務の知識、経験について標準形を活用。

8：業種知識　顧客業務の知識、経験について

評価標準形を活用、業務コンサルができるレベルの人材が必要（この経験者がいないと顧客の業務改善も指導できない&余分機能カットできない）

9：基礎知識が必要な場合は特に注意

理系基礎知識、あるいは経理基礎知識などが根本的に不足しているメンバーのみで専門知識を必要とするシステムの開発は出来ない

10：プロジェクト管理力

（３）ハーマンモデルの利用（チーム内メンバーの性格分析）

開発メンバーの力をフルに発揮してもらうためには、技術力だけに注目しても上手く行かないことが多い。人はスキル、センス、マインドの総合力で動くものである。

初めて顔をあわせたメンバーの技術だけでなく性格・気質を知っておくことは重要なことである。そこでこの手法「ハーマンモデル」を活用する。

わずかな時間、例えば 30 分程度で、20 人程度のメンバーの、おおよその性格を知ることが出来る。この手法の創始者のハーマンはＧＥの人事部長であったが、この素晴らしい方法を編み出した。

まずＡ，Ｂ列の一見脈絡のない２４質問に回答していただく。マーク欄にＡかＢかどちらかを選択しレ点をつけておく。次に２番目の表のＡ，Ｂ，Ｃ，Ｄのスコアー列にレ点をつけた質問番号を探しマークする。Ａ、Ｂ、Ｃ、Ｄの合計マーク数を「チェック数合計」フィールドに記入する。最後にＡ，Ｂ，Ｃ，Ｄの円グラフの軸の上に先に集計した合計点を記す。４軸の点を線で結ぶ。Ａ軸が高い人は論理的、Ｂ軸は計画的、Ｃ軸は社交的、Ｄ軸は冒険的を表している。

総合力を発揮するようにＡ，Ｂ，Ｃ，Ｄの組み合わせを参考にチーム編成を行う。同じ性格の人を組み合わせても、新しい発想は浮かばないし、ウマが合わない組み合わせで仕事をしても成果は出ない。

複数のソフトウェア会社のメンバーが集まって１チームを構成している場合

発注会社と受注会社の混成チーム

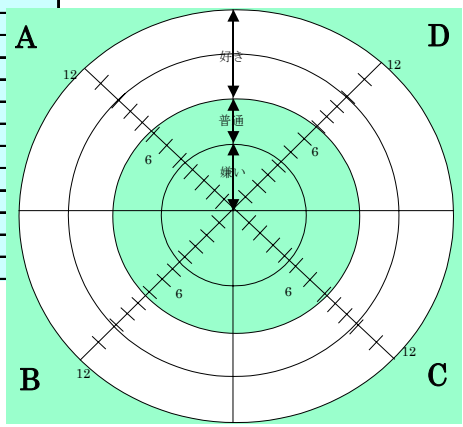
情報システム部門、業務部門と協力会社のメンバーの混在チーム

年齢差の大きいチーム

などの場合にこの手法を適用されると効果が大きい。

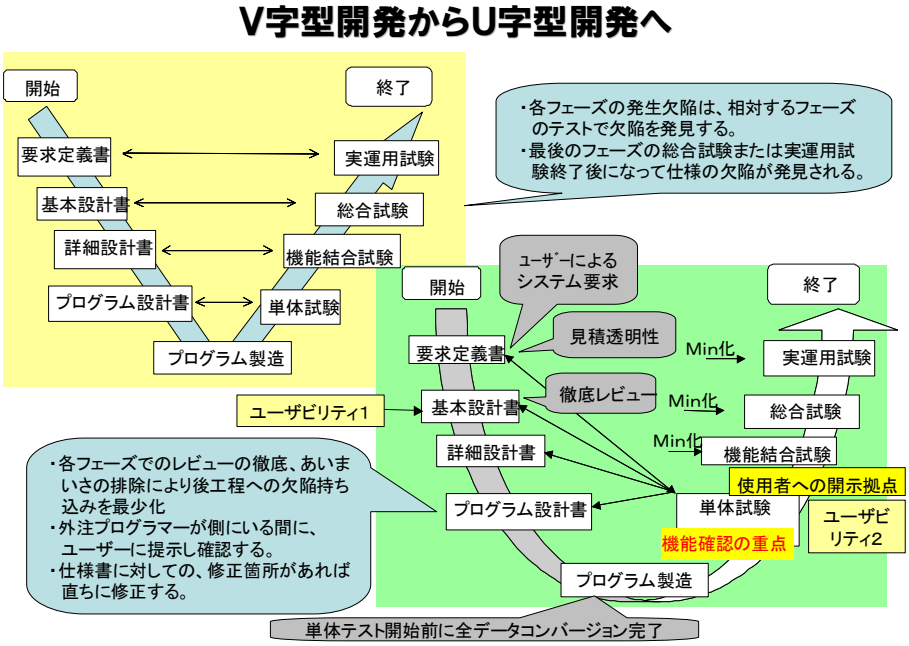
	A	マーク		B
1	決まりやしきたりに従う方だ			他人がどのように感じているかを理解できるほうだ
2	考え方や物事を部分に分け、部分のほうから考えてゆく方だ			考え方や物事の断片を組み合わせ、全体像の方から考える方だ
3	正確な数値を知りたがる方だ			リズムに乗りやすい方だ
4	難しい問題を論理的に解きほぐしながら答えを見つけるのが好きだ			実行する前にきちんと計画を立てるほうだ
5	自分の感情をコントロールできる方だ			自分で考え、新しいものを作り上げることが好きだ
6	普通の人が考え付かないようなことを思いつく方だ			自分の感情を表に出す方だ
7	感覚を大事にするほうだ			考えることが好きだ
8	人と接することが苦にならない			仕事の流れ、物事の構造を一連の流れとして整理するのが好きだ
9	神秘的なものに興味を持つ方だ			自分で考え、新しいものを創り上げることが好きだ
10	物事の細かな部分、項目を十分チェックする方だ			個別のことよりも全体のことを考える方だ
11	アイデアを良く思いつく方だ			アイデアが実現可能かどうか確かめたくなる方だ
12	どちらかと言えば「暖かく友好的に」人に接する方だ			考えや物事を部分に分け「部分」の方から考えてゆく方
13	空想を巡らすことが好きな方だ			仕事が決まった順番にしたがって処理するのが好きだ
14	普通の人が考えつかないようなことを思いつく方だ			新しいものより、古くても実績があり信頼できるものを選ぶ方だ
15	自分で考え、新しいものを作り上げることが好きだ			論理的に考えることが好きだ
16	自分の感情をコントロールできる方だ			自分の感情を表に出す方だ
17	リズムに乗りやすい方だ			物事の細かな部分、項目を十分チェックする方だ
18	2つ以上のことを同時にこなすことが出来る方だ			他人がどのように感じているかを理解できる方だ
19	人とのコミュニケーションを大切にする方だ			具体的な事柄を抽象的な概念に一般化して考えをまとめる方だ
20	科学・技術的事項の理解、技術的スキルを習得することが好きだ			人を大事にする方だ
21	仕事の流れ、物事の構造を一連の流れとして整理するのが好きだ			論理的に考えることが好きだ
22	規則・規律を破るのはあまり好きではない			物事を何かにたとえ得て表現するのが得意な方だ
23	従うべき計画があると安心する			数学的なものを好む方だ
24	科学及び工学の知識を理解し、応用するのが好きだ			自分の考えにしたがって組織を動かすのが好きだ

Aスコア		Bスコア		Cスコア		Dスコア	
A2		A1		A7		A6	
A3		A5		A8		A11	
A4		A10		A9		A13	
A20		A16		A12		A14	
A22		A21		A17		A15	
A24		A23		A19		A18	
B7		B4		B1		B2	
B11		B8		B3		B5	
B12		B13		B6		B9	
B15		B14		B16		B10	
B21		B17		B18		B19	
B23		B24		B20		B22	
チェック数合計		チェック数合計		チェック数合計		チェック数合計	



図表 3-5-9　ハーマンモデルの例示

９．Ｕ字型開発法



図表 3-5-10　Ｖ字型開発から Ｕ字型開発へ

Ｖ字型開発の図は、システム開発の奥義として解説されているし、その説明は間違ってはいない。しかしこれを使いこなすための工夫は数多く存在する。

流れに従って JUAS 流工夫を考えてみよう。

（１）要求定義書

「システム利用者は、毎日の作業をこのシステムを利用して実行するので、システムオーナーでもある」「このシステムは使いにくい、役に立たない」と嘆く前に「新システムに何をまかせるのか？どのような機能を持つシステムを期待するのか？を利用者から文書で要求しよう」と１４種類のドキュメント体系を作成することを提言した。（第３章第２節　「業務分析とＲＦＰの作成」の項を参照のこと）

作成者である利用者のレベルに合わせてレベル１から５段階に分けて、ベンダーとユーザーの分担を整理してある。

「要求仕様書のレベルが低かったので失敗した」とベンダーに嘆かれないように、ユーザーの作成したドキュメントのレベルを明確にしたのである。

（２）見積の透明性

ユーザーは「システム開発費用が高い」不満とともに「見積の透明性がない」ことにも不満を感じている。ではベンダーはシステム開発に伴うリスクをどのように見積もっているのか？の手がかりを、見積の章で示した。（第 3 章第 3 節「見積」の項を参照のこと）

「何をすれば安くて品質の良いシステムを入手できるか？」がユーザーにとって分かるように出来た意義は大きい。ユーザーから「何をすればもっと安く開発できるのですか？」と質問をし、両者のコミュニケーションをとるきっかけができたと考えている。

（３）レビュー

この見積照会仕様書の内容を補足し、あるいは理解し「基本設計書」をベンダーが作成する。この承認・確認をするためにレビューを両者で徹底する。

内容は文章で表現されることが多いが、日本語の持つ曖昧さが以降の開発での障害につながってくる。このようなトラブルや重複作業を避けるために「システム技術文章の書き方」なる参考書を作成している。

ソフトウェアメトリックス調査の中で、このレビュー時間の有効性を確認した。

作成した時間に対して 10%程度のレビュー時間を確保されるのが望ましい。

（４）プログラム製造

しかし現在の日本のシステム開発環境は一般的に、プログラム作成部分はオフショアー開発含めて外注作業として一次請負会社からさらに下請に出されることが多く、機能結合試験、総合試験、実運用試験の時期には各プログラムを作成してくれたプログラマーは既に別の仕事につき、このプロジェクトの近くにはいないケースもある。

そうなるとプログラム作成技術には疎い SE が、他人の作成したプログラムを修正することになる。効率は数分の一に低下するので結果的に費用が増加する。品質も劣化するし、修正箇所がそこだけで済んでいるのかどうか自信も持てない。

このような事態を避けるために、「単体テストが完了したらほぼ完成している。プログラムの修正作業は機能結合試験以降にはほとんど発生しない」方式を採用した方が、すべての面で効率的である。

そのために単体テスト開始時には、従来総合試験に間に合わせて作成していた「現行システムからの切り替えのためのデータベースコンバージョン作業を単体テスト開始時には間に合わす」ことを目標に作業する。

この指示を聞くと、従来からの作業に固執する多くの SE は「そんなことは出来ない」と否定するので

「そのために DOA は存在する」

「近代開発法はこれであるとデータ・コンバージョンチームに頼め」

「百点満点でなくとも 95 点でも良い。後で 100 点にレベルアップすればよい」

「データベースの中の異常データや仕様の抜けも発見できる」などと説得に励むことになる。

しかし、一度この方法に慣れるとそれ以降のプロジェクトはすべてこの方式を採用するのが普通である。

勿論単体テストではテストデータを別に作成し、プログラムの全ルートを経由させ品質保証をする必要がある。

この方式の効果を整理してみる。

（ａ）プログラマーが側にいる間に大量のデータに基づく作業が出来るので、品質が向上する。とくにバッチプログラムは完全なテストデータを新しく作成しがたいので、「現行システムでの結果と新システムでの結果が比較できる」意義は大きい。

（ｂ）新システムは、切り変え当初に、旧システムで処理した結果を引きついで処理をする必要があるケースが多いが、この処理が十分かどうかの検証がしやすい。

（ｃ）データ・コンバージョン作業にも、ミスが伴うが、単体テスト完了時から多くの人の目でチェックがなされるので、自然と精度が向上してくる。システム切り替え時のコンバージョンミスのためにプログラムは正しかったが、結果に障害が出て後でその修正に苦労した経験を持つ方も少なくないのではないか。

（ｄ）大量データを使ってのテストが可能になり、内部テーブルの容量不足などのテストも可能となる。またこのための処理時間の把握も可能になるので、リレーショナルデータベースのインデックスの持ち方を後の総合テストなどで変更するなどの作業も不要となる。バッチシステムの処理時間、オンラインシステムのレスポンスタイムの把握も可能となり対策を早めに講じることが出来る。

（５）目標の設定と遵守（機能結合、総合、実運用試験）

単体テストでのテストケースが充実し品質が高まると、上記のテストフェーズは、サブシステム相互のインターフェースや、実機との接続性に重点を置いたテストを行えばよい。

しかし、ソフトウェアメトリックス調査によると、設計：構築：テストの期間比率は、

3：3：4になっており、プログラムは出来たが、内容確認やユーザーの習熟に時間をかけシステムの品質向上に努力している実態が現れている。

しかしこの期間に実施している作業内容をみると

（ａ）単体テストの不十分さを補うテスト

（ｂ）ユーザビリティ向上のための修正

などの作業も行われている。

（ａ）についての対策はプログラム作成者と受入検査する人の間で「プログラム作成についての品質目標を設定すること」である。

JUASはユーザーに分かりやすいように、ユーザーへの受入検査で「発注金額500万円に対して１件の割合で障害を抑えること」との目標値を提唱してきたが、この目標を達成するためには

「500万円に対して単体テスト終了時には3件程度の障害」に抑えてないと、その後の修正作業が増大し品質を維持することが出来ない。勿論FP単位に換算しても良いし、LOCベースに基準を設定しても良い。粗くても良いので目標値を設定してあることが,品質確保の第一歩である。

（ｂ）に対してはまだ研究中であるが、以下のように考えている。

（６）ユーザビリティの向上

Web システムが開発され、かつての汎用機時代の画面設計からは大きくユーザビリティは飛躍した。しかしこのユーザビリティを確保するための手順の研究は大きく遅れている。

詳しくはユーザビリティの章で解説するが、基本設計段階でまずはユーザビリティのレビューをすることである。そのための規範のガイドが必要となる。

「このシステムが使いやすいか・どうか？」は多少の見解の差はあるが、誰でも判断は出来る。問題は基本設計段階でユーザビリティの確認レビューをしているかどうかである。

この段階でレビューをしておけば、単体テスト完了時のユーザビリティのための修正作業は少ない。

以上のような要素をＶ字型開発に取り込み、単体テスト完了時にはほとんどの障害は解決されている、日本的風土にあったシステム開発の体系としてまとめたのが、Ｕ字型開発方式である。つまり単体テストの段階、プログラマーがまだ近くにいて作業をしている段階でシステムの完成した姿が現れる方式である。日本情報システムユーザー協会（JUAS）にちなみユーザーのＵを取ってＵ字型開発と名づけた。

「Ｕ字型開発はウォーターフォール型開発なのか？イテレーション型開発なのか？」と疑問を持たれた方がいた。

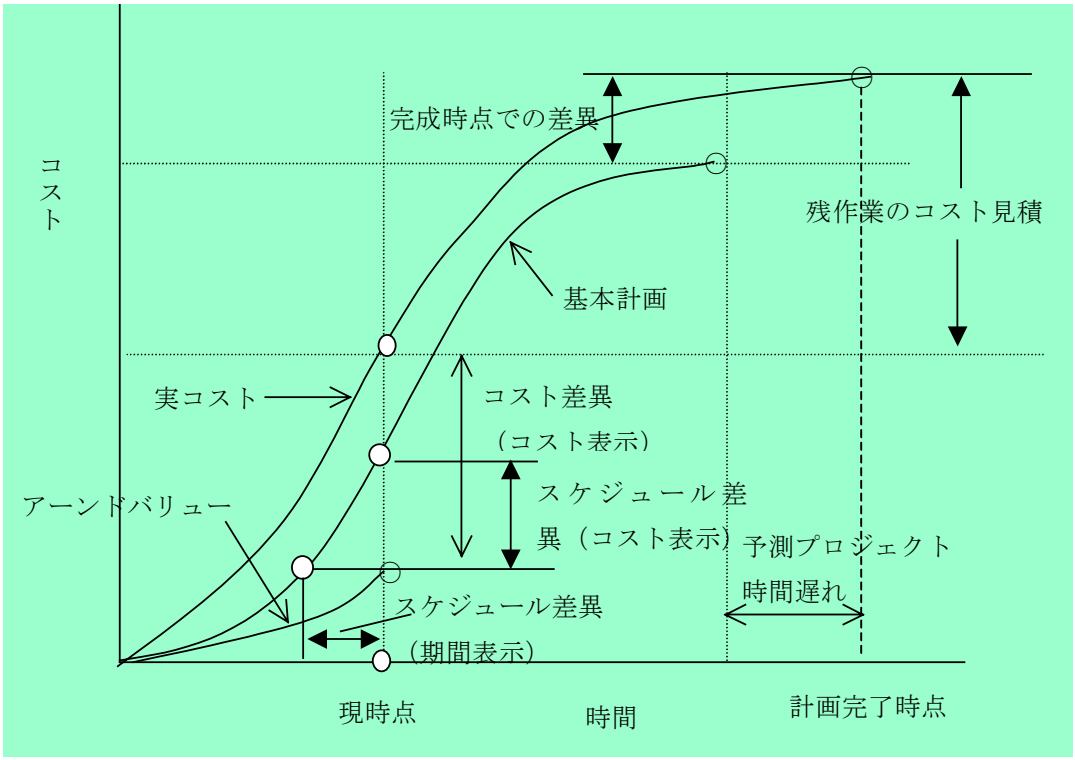
大きな流れはウォーターフォール型に見えるが、基本設計の段階でユーザビリティをプロトタイプアプローチで短期間に「繰り返し繰り返し」使用性を確認しているので、この部分はイテレーション型開発法とも言える。

単体テストで不都合な点は直ぐに修正をかけられるのもイテレーションの特徴に近い。

イテレーション法の良さとウォーターフォール法の両者の良さを限りなく取り込んだ開発法と言うことが出来る。

実際に適用してみてこの方法の効果を味わうとともに、更なる改善をされたい。

１０．ＥＶＭ（Earned Value Management）



図表 3-5-11　EVM（Earned Value Management）

＜EVM の手順＞

（１）ＷＢＳの作成（受注者による計画の提出）

プロジェクト作業計画を、WBS として作成。

ID	作業名	請負	開始予定日	終了予定日	PV 出来高計画値
5	開発	■	2002/1/22	2002/2/20	3.096
5･1	サブシステムA開発	■	2002/1/22	2002/2/14	998
5･1･1	共通機能開発	■	2002/1/22	2002/2/5	343
…	…	■	yyyy/mm/dd	yyyy/mm/dd	….
5･2	Bサーバー開発支援	□	2002/2/2	2002/4/20	974
…･･	….	□	yyyy/mm/dd	yyyy/mm/dd	….

図表 3-5-12　受注者による計画

・請負：該当作業が請負作業なのか？支援を含むその他の作業なのかの区別を記入する

・PV 出来高計画値：計画時に各作業に割り当てられた出来高

（２）進捗実績表（受注者による進捗状況の定期的報告）

	1月22日	3月12日	3月19日	3月26日	4月2日	4月9日	4月16日	4月23日
PV	41	2.41	3.058	3.453	3.518	3.669	3.709	BAC 完了までの予算
EV	35	2.11	2.708					
AC	35	2.771	3.44					
SV	−6	−300	−350					
CV	0	−661	−732					
SPI	0.86	0.87	0.89					
CPI	1	0.76	0.79					
EAC	3.889	5.109	4.703					

図表 3-5-13　進捗実績表

・開始後現在までの実績提出

・記載すべき項目の値：進み／遅れの程度

PV：出来高

EV：出来高実績値

AC：コスト実績値

SV：スケジュール差異

CV：コスト差異

SPI：スケジュール効率指数

CPI：コスト効率指数

EAC：現時点で見積もった完成までの総コスト予測

EVM の作成手順は、すべての作業をツリー状に構成し WBS に落とす。一つの作業単位は大きくても 1 週間程度の大きさに細分化することが望ましい。（細分化された作業単位は work package　と呼ばれる）この work package について作業開始予定日、終了予定日、出来高計画値（この計画作業負荷は予算ではいくらに相当するのかを全体予算をもとに割り出しておく）を設定する。作業完了報告を元に「この work package が完了したから作業費がいくら売りあがったことになるか」を集計し開発実績の状況を確認していく。

理論的ではあるがすべての作業を WBS に落とす、実態に応じて見直すことは相当な作業負担であり、ここまで時間をかけ、作業負荷、予算をかけて、計画を細分化できないとあきらめるプロジェクトや企業が多い。

納期厳守のプロジェクトはこの EVM を採用したほうがプロジェクトの進捗について安心感を持てる。

ここまではできないプロジェクトや企業は後述の開発高算出法を検討されると良い。

１１．日程計画表

（１）日程計画表の要点

項目＼計画種類	大日程	中日程	小日程
①作成タイミング	工程毎の実行計画 （月次単位見直し）	工程毎・ｻﾌﾞｼｽﾃﾑ毎の実行計画 （週単位見直し） 「一週間の遅れが判定できる基準を盛り込む	ｻﾌﾞｼｽﾃﾑ・担当毎の実行計画 （日単位の見直し）
②狙い、目的	全体進捗管理 （本番ｽｹｼﾞｭｰﾙ調整）	ｻﾌﾞｼｽﾃﾑ毎の進捗管理 （同左）	個々のアクション毎の進捗管理 （SE/PGRの要員調整）
③管理メッシュ （フォローのタイミング）	ﾌﾟﾛｼﾞｪｸﾄ外からの応援体制	ﾌﾟﾛｼﾞｪｸﾄ内（ｸﾞﾙｰﾌﾟ間）の応援体制	１週間以内の遅れは自己調整 上記以外の遅れは応援体制 （ｻﾌﾞｼｽﾃﾑ・ｸﾞﾙｰﾌﾟ内）
④作成担当者⇒レビュ	PL⇒顧客	SPL⇒PL	SE⇒SPL PGR⇒SE
⑤要点・他	顧客の要望を尊重	大日程に合わせた日程／体制を配慮	実務担当者の意思を尊

図表 3-5-14　日程計画表の要点

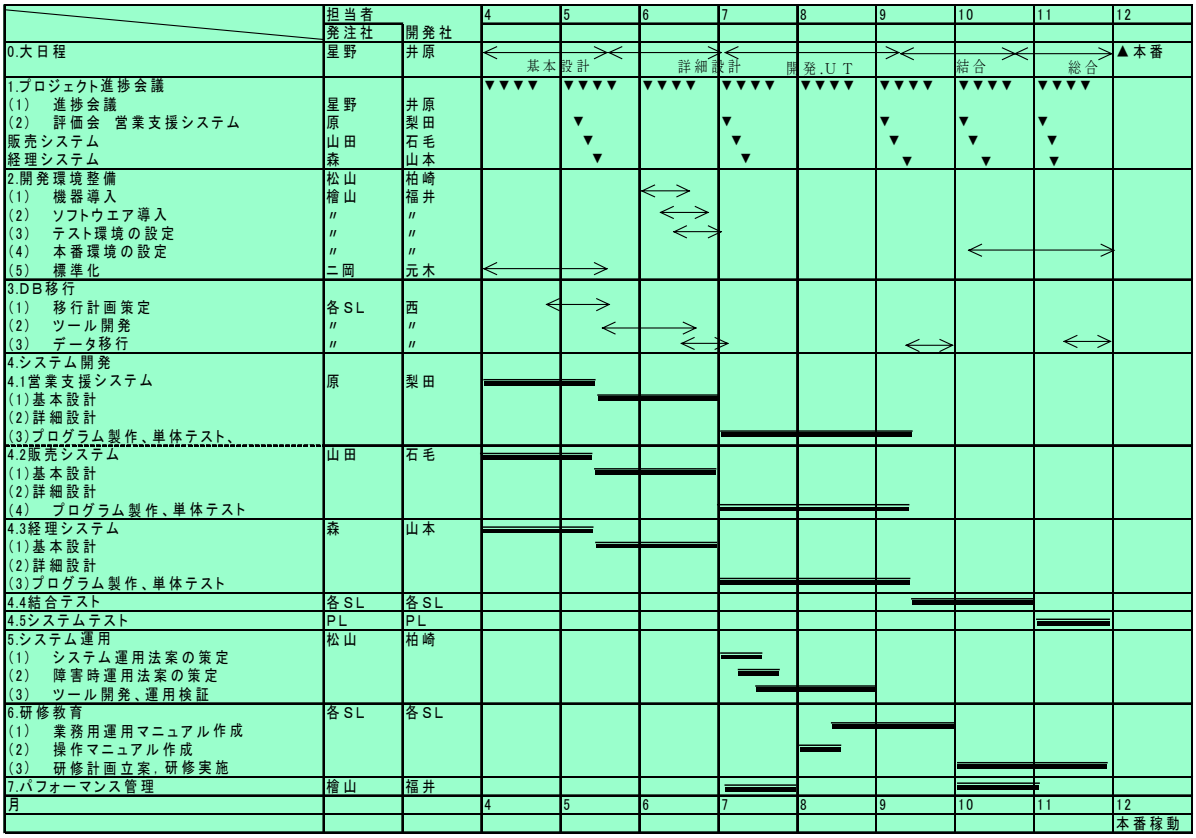
「バーが書いてあるだけでは日程計画を作成した」とは言えない。

営業活動段階でベンダーが提示したバーチャートは大日程計画のひとつではあるが、これでスケジュールの作成が完了したと誤解しないで欲しい。

作成タイミング、狙い、管理メッシュ、レビューの仕方などについて、大日程計画から中日程計画に移り、最後は日毎の作業計画である小日程計画になる。

（２）大日程計画の立て方

大日程計画表に例示



図表 3-5-15　大日程計画表

（３）中日程計画の作成方法

中日程計画表に例示

NO	詳細設計・開発作業項目	担当者				開発スケジュール														
		A	B	C	G	4月				5月				6月				7月		
						1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	他システムIF　整理，確定	○	○	△																
2	DB設計の確定	○	△	△																
3	項目・コード・定義表作成	○	○	△																
4	検索画面設計	△			○															
5	画面レイアウトの決定	△			○															
6	画面変遷の確定	△		○	△															
7	帳票レイアウトの確定	○			○															
8	詳細検討条件整理(データ対象、編集条	△	○		○															
9	プログラム仕様作成(帳票、画面、その他)		△	△	○															
10	プログラム製作説明(標準、仕様概要、エ		○	△	○															
11	DB登録・プログラム作成					○														
12	単体テスト準備(テストケース，データ)			○																
13	プログラム単体テスト実行	△		△	○	△														
14	営業支援システム内結合テスト			○	○															

○実行　△確認

＜凡例＞　－　概要決定・＝　確定

図表 3-5-16　中日程計画表

（４）小日程計画の作成方法

小日程計画表に例示

NO	Pgno	プログラム名	DB	規模	担当者	3 月	4 火	5 水	6 木	7 金	8 土	9 日	10 月	11 火	12 水	13 木	14 金
1	AA110	顧客情報入力	A	1000	仁志	―				レビュー日			＝	＝	＝		
2	AA120	顧客訪問記録	A	500	清水	―							＝	―			
3	AA130	月次目標入力	B	500	清水	―								＝	＝		
4	AA140	作業時間入力	B	1000	仁志		―									＝	＝
5	AA150	保守運用情報	A	500	後藤	―							＝	―			
6	AA200	優良顧客分析	A	500	〃	―								＝	＝		
7	AA210	地域顧客分析	A	1000	〃		―									＝	＝
8	AA220	保守分析	A	2000	仁志			―	―								
9	AA230	インセンティブ計	B	1000	清水		―	―								＝	＝
10	AA240	月次売上処理	C	500	〃				―								
11	AA250	生産性分析	D	2000	後藤			―	―								＝

＜凡例＞　－　詳細仕様の理解・分析、＝　コーディング＆テスト

図表 3-5-17　小日程計画表

（５）日程計画表の使用時の注意点

「スケジュールとは、この道しるべ通りに開発してゆけば、プロジェクトは成功する」ために作るものであるから、その期間にその作業が完了する根拠を持たねばならない。

さて、根拠に基づく日程計画表が書かれているだろうか？各日程計画表の作成と使用法の注意点を述べる。

ａ．大日程計画表

・ユーザーと開発者が最初に「大まかにはこのような予定で開発しましょう」と約束を交わすのが、この大日程計画表である。
・これは設計、製造〔構築〕、テストの期間を、一定の比率で分けて算出する。
・JUAS の調査では、上記期間比率は 3 : 3 : 4 であったが、システムの性格や環境の差により４：３：３あるいは 2 : 2 : 4 でも使い分ければよい。
・この期間内に投入できる人月を総予算に応じて押し込むことになる。
・縦軸に開発体制を配慮しながら大きな作業区分ごとに、基本設計、詳細設計、開発、テスト（結合、総合）など工程ごとの予定を、チーム相互間に矛盾が生じないように、割り当てる。
・一つの作業単位は月，旬にまとめておく。
・全体の予定と進捗を確認するためであるから、一表に収められていることが望ましい。
・この表を基にした進捗会議の予定も、あらかじめ定めておく。
・短工期開発の場合は、稼働日から設定して、各開発フェーズがどこで終了しておかねばならないか？を考えて予定を作る場合もある。この表で個別の作業を管理することは難しい。むしろ中日程計画を作成するための準備表の意味ももっている。
・大規模システムの場合、複数開発会社に分散して発注する場合は、ユーザー企業のプロジェク

トマネージャーが作成責任を持つか、開発会社の中の代表企業にプロジェクトマネージメント責任を持たせて全体責任を持たすなどさまざまな形態がある。
・いずれの場合もユーザー企業はシステム遅れの影響を大きく受けるので、実態把握に努めねばならない

ｂ．中日程計画表

・大日程計画に基づいた、工程ごと、サブシステムごとの作業実行計画であり、投入人月と成果物の対応がついている必要がある。週単位に作業の進捗状況を確認するために使用される。「一週間の遅れが判定できる基準を持った計画でなければならない」
・縦軸はサブシステム内の作業区分に細分化されてくる。この後は小日程計画表となるので、そのプログラム単位の設計、構築作業を集約した区分となる。
・スケジュールは週単位に区切られて管理される。
・この予定表に対して実績が記入され、担当者ごとの進捗が確認される。
・サブリーダーごとに情報提供の責任を持つ。ユーザー企業の作業負荷よりもベンダーの方が大きいので、ベンダー企業が責任を持って作成することになる。
・しかしユーザー企業の協力状態が悪く、ユーザー作業が遅れがちになる場合は、思い切って、ユーザー企業側の作業を上段にまとめ、ベンダー側作業を下段にまとめて、一目瞭然、どちらの作業遅れが大きいか？を示す方法もある。

ｃ．小日程計画表

・一週間内の毎日の作業を「何をこの日の中で完了させないといけないか？」を意識させる計画である。その日の中で作業を完了させることを約束する計画であるから、各担当者に精神的束縛を与える。したがって上位者は一週間の作業量を与え、後は作業者個人に仕事の日割りを任せるのが良い。上位管理者が作成して与えれば「ここまでやらねばならない」との拘束になるが、自分が予定を作成すれば、「ここまで作業を進めることに自分で決めた」との目標になる。
・作業者も感情を持った人間であるから、当然体調の良、不良も出てくるし予定外の用事も入ってくる。「今日は予定どおりいかなかったが、明日はがんばるぞ」との目安にしてほしい計画表である。
・あくまで、小日程は作業者自身の作成する目標管理用である。
・小日程→中日程と見直しされた結果が大日程に上がってくる。

１２．システム仕様決定時のスケジュール

仕様決定段階のスケジュールは、システム開発時のスケジュールに比較して、予定どおり進めがたいものである。進捗遅れの大部分はこの仕様決定の段階に発生すると思ってよい。

この段階を予定通り進めるために、このような計画表を作成し透明性を高めるのが良い。

各仕様を決定するためにはこのような６段階の過程をへる。プロジェクトによっては関係者が多くさらに多くの方の承認を必要とする場合もある。

利用部門の責任者は、実務をしながら、承認作業を行うので「システムの承認を予定どおりに実施しないと関係者皆に迷惑をかける」認識を持って相当な努力をしないと、予定は確保できない。このような図を提出されれば、どこが停滞しているのか？約束を守れていないのか？明瞭である。

タスク		役割分担		予定	4月				5月				備考
項目番号	対象	顧客	当社	実績	506070809	12131415	1920212223	2627282930	304050607	1011121314	1718192021	2425262728	
1	画面設計												
1.1	注文書登録	野村	山本	予定	①①①①②	②②③③③	④④④④④	⑤⑥⑥⑦					
		◎	●	実施	①①①②	②②②③③	③④④④④	⑤⑥⑥⑥⑦					
1.2	注文内容調整	達川	東尾	予定	①①①①②	②②③③③	④④④④④	⑤⑥⑥⑦					
		◎	●	実施	①①①②	②②②③③	③④④④④	⑤⑥⑥⑥⑦					
2	帳票設計												
2.1	注文結果リスト	星野	仰木	予定		①①①②②	③③④④④	④④④④④	⑤⑤⑥⑦⑦				
		◎	●	実施	①	①①②②③	③③④④④	④⑤⑤⑥⑥	⑦				
2.2	代理店別注文明細	同上	同上	予定		①①①②②	③③④④④	④④④④④	⑤⑤⑥⑦⑦				
		◎	●	実施	①	①①②②③	③③④④④	④⑤⑤⑥⑥	⑥⑦				
3	論理ファイル設計												
3.1	注文DB	川上	大下	予定				①①②②③	④⑤⑤⑥⑥	⑦			
		△	●	実施				①①②②③	④⑤⑤⑥⑥	⑦			
3.2	在庫DB	同上	同上	予定				①①②②③	④⑤⑤⑥⑥	⑦			
		△	●	実施				①①②②③	④⑤⑤⑥⑥	⑦			
4	機能仕様詳細												
4.1	注文書処理												

図表 3-5-18　仕様決定段階のスケジュールの例示

＜凡例１＞

◎：主管チェック承認　●：作業主体　○：支援協力　△：チェック承認に分けて明確化。

＜凡例２＞

①資料提供および顧客側要求機能まとめ　②一次案作成　③一次レビュー

④顧客内レビュー(何箇所の承認を得れば良いのか？確認する)

⑤プロジェクトチーム内レビュー　⑥修正　⑦確認承認（決定）

１３．予算規模の縮小

プロジェクト予算には、２：４：２：３の法則がある。

最初の予算を２で想定し、システム検討を始めると「あの機能も欲しい」「これも必要」と希望が出て、そのまま開発すれば倍の４の予算が必要になる。

「それでは絞ろう」と元の予算規模にまで大議論のすえ機能削減を実施し、開発をはじめるが「やっぱりこの機能がないと上手く動かない」「この機能・条件を忘れていた」と３の予算規模に戻る。

「これがシステム開発の普通の現象である」とする法則である。

その結果苦労して開発したのに、後から考えてみれば「あの機能は別の機能でもカバーできた」「あの機能は不要であった」と使われない機能が続出し不良資産となる。

「システムは小さく生んで大きく育てる」方が予算効率は良い。

以下に、予算規模を縮小しなければならない時の手段を列挙する。

・思い切って予算は半分、あるいは 30%カットの機能に収めることを前提に検討を始めるのも一つの手である。
・あるいは絶対に必要とする機能以外は、「次のフェーズまわし」としておき、システムを稼動し始めてから、やむを得ない場合のみ機能追加をする、手もある。
・担当者から「この機能は絶対必要」と主張された機能について、上司の管理者に判断を求めたところ「１年間で何回その機能を使うのか？　そのためにいくら費用がかかるのか？」確認されて、「その作業は手作業ですましてくれ」と決裁されたことがあった。「絶対に必要な機能か」「あれば便利な機能か」「なければないで何とかなる機能」なのかを見極め、上手に説得するのも、SE や管理者の仕事である。
・システムにはプログラム保守作業がつきもので、常に成長するものである。その中の SE やプログラマーが余裕のあるときに改善してもらえば良い程度の機能は最初から準備する必要はない。思い切ってスリムにした機能で稼動開始するのが良い。
・「ほとんど使われていないので不良資産として削除した」機能がなぜ開発されてしまったのか？を分析するのは、過去の話でもあり、分析するのは難しいが、企業の文化に関係するところであり、CIO はチェックしてみる価値がある。

第３章　開発

第６節　プロジェクト管理

プロジェクト管理は人と人との Struggle 闘争である。「大規模システムの開発は、代表的な経営総合活動である」といわれているが、本当に各種のプロジェクト開発には、個別の問題が生じてくる。この問題を技術だけで解こうとしてもなかなか解けない。プロジェクトマネージャーの人間性で解決した方が簡単に解ける問題もある。この章では実践的な解決方法を紹介してみたい。

１．仕様決定段階のプロジェクトマネジメント
２．開発テスト段階のプロジェクトマネジメント
３．システムの切り替え段階のプロジェクトマネジメント
（参考）イテレーション型開発（反復開発）の注意点

１．仕様決定段階のプロジェクトマネジメント

確実迅速な仕様決定を行うためには、技術的問題解決のほか、コミュニケーションに注意し全知全能を傾けた対策をとらねばならない。

（１）プロジェクトのキックオフ前の打ち合わせ

ユーザー、ベンダーのキーマンが顔をあわせ、プロジェクトのキックオフ前に下記の項目について打ち合わせをするのが望ましい。一度も顔をあわせたことのないキーマン同士がプロジェクト・キックオフの場にのぞみ、個性、主張を突きあわせても上手く行くはずがない。プロジェクトを開始するキックオフミーティングに先立ってキーマン同士が以下の項目について意見交換をしておく必要がある。

①発注者、受注者のキーマン同士の性格把握、社内での実績を語り、お互いを知りあっておく。
公式、非公式な場を設け今回のプロジェクトについて「目的、予算規模、範囲、工期、品質」などの希望や課題を語り合い成功を誓い合う。
キックオフの場で「リーダー同士は相当に心が通い合っているな」「お互いにこのプロジェクトの成功に人生をかけているな」などの印象をプロジェクトメンバーに与えることが出来れば、少々の不満は担当者同士が話し合って解決するものである。
②互いの企業文化について情報を得る。企業には外から見えない企業風土・文化をもっている。この影響は計り知れないものがあるので、それを前提にしたプロジェクト推進や進捗管理を心がけねばならない。
③顧客の経済環境、事業環境を理解
どのような背景があってこのシステムを開発する必要が出てきたのか？
このシステムの開発が遅れるとどのような問題が発生するのか？
収益状況はどうか？
周囲の協力体制はどうか？
など十分にこのシステムを取り巻く環境を認識しておかねばならない
④システム効果について
効果が発生する仕組み、効果確認の時期、効果責任者などを確認しておく
⑤稼動後の運用体制・条件について確認しておく
追加予算の考え方について認識する。
初対面で追加予算の話をするのは気が引けるところがあるが、開発者側は顧客の過去のプロジェクトがどのような扱いになっているか？確認したほうが良い。追加予算は一切認めない顧客もあるので、そのような顧客にはシステム規模、優先度の考え方、次のフェーズへのシステム開発内容の持ち越しなどについて了解を得ておく必要がある。

以上③、④、⑤はベンダーからの質問である。
⑥基本的な発注者、受注者の考え方の差をまとめると次のようになる。

項目	発注者（顧客）側	受注者（開発会社）側
１：工期	短期開発を希望	必要工期は確保したい
２：品質	良ければ良いほど良い	適切な目標の達成を望む
３：コスト	支払金額は安ければ安いほどよい	受注金額は高く欲しい
４：投資効果	早期達成を望む	顧客側で責任を持って欲しい
５：開発マナー	素直に受け入れて欲しい	予算内での対処が原則
６：プロジェクトの成功	成功させたい	成功させたい

図表 3-6-1　発注者、受注者の考え方の差

（２）確実迅速な仕様決定

ベンダー側が最終仕様を作成し、確認をユーザーにもらうことが多いので、主としてベンダー側のプロジェクトマネージャーは、仕様決定の工期を守るために以下の事項について配慮して作業を行う。

ａ．基本設計の前提条件の確認をしておくこと

・システム化の目的、対象、範囲、このシステム開発を依頼した実質キーマンの意向、エンドユーザーの要望、解決策との関係などの条件を今一度再確認する。これらの事項が発注企業内で意外にあいまいなことが多い。

・ハードウェア、ソフトウェア、ネットワーク、OS、ミドルソフト、データベース、ケースツール、開発言語などについて、相互確認する。

・このシステムの最終利用者は誰か？、主要データの入力者は誰か？、仕様決定者は適任か？、最終納入検査者、承認者は誰か？

・承認行為は何をもってなされるのか？、ドキュメントは章ごとの承認か、ページごとに承認するのか？、報告単位の承認か？

・TRM（Task Responsibility Matrix：実行計画の８．１項参照）、PAM（Power Analysis Matrix：実行計画の８．２項参照）は準備できているか？

・ユーザー側がプロジェクトリーダーは社内関係者をリードできる立場、人か？

・ベンダー側リーダーはこのプロジェクトを任せて十分な実力経験をもっているか？

ｂ．ヒヤリング・打ち合わせの準備

・業種、業務知識を取得する。（参考書を読む、過去の類似システム経験者からの情報入手など）

・システム業務知識（業務規定、過去の開発記録などを基に）習得しておく。

・ヒヤリング前の準備（参加者、時間、目的、予定表、会議室、資料などを出来るだけヒヤリング前に配布）をして関心を持ってもらう。

・現状業務フローは、書ける範囲でユーザーに準備してもらう。頂いた資料についてはよく読んで質問表も準備しておく。

・宿題であった資料は締め切り前に、督促電話をして確実に間に合わせる。

・ヒヤリングは単に意見聴取の場ではなく、「意見・知識・人間性の挑戦の場・相互研鑽の場である」。

ｃ．ヒヤリング・打ち合わせ中の配慮

・今何をすべきか？、残作業は何があるか？、「Priority」をつけて周知徹底させる。（フェーズ別作業アイテムをあらかじめブレークダウンしておく）
・発言を文字、表、図に変える努力をする。（発言を要領よく整理して黒板・紙に書く、パソコンに入れて投影し共感をうるなど）これにより議論の理解を深め、後戻りを防ぐ。
・Active Listening 積極的に身体全体で聞く姿勢を見せる。（アイコンタクト、合槌、なるほど、たしかに）。相手から好感をうる。
・同意，感嘆できたときは、身体全体で喜びを表す。（すばらしい、非常に良いご意見）
・仕様の説明は資料をなるべく読まずに（下を見ているだけでなく）、相手の反応をみながら説明する。聞き手の表情を見よ。理解できたかどうか質問をしてみる。
・仕様を決定しやすい形で提案する。（相手の問題は自分の問題として考える）
・利用者のレベルに合わせたシステムデザインを留意する。（使われなければ何の意味もない）
・本筋をはずさせないための誘導を行う。

セッション・ルール：会議の効率化、活性化、創造化のためのルール （IBM の CPS セッション・ルールより） 目的：時間の有効利用と計画の効率的達成
1:Listen　(相手の立場、主張、意見の理解、話を中断しない) 2:Think　（集中、クリエイティブに） 3:talk net　（簡潔に、同じ話を繰り返さない） 4:State Clearly　（適語表現） 5:1-Subject,1-Time　（テーマをスリップしない） 6:Be Decisive　（自分が意思決定、論拠を持つ、示す） 7:Unanimous Agreement　（全員の合意・対案なければ賛成）

図表 3-6-2　CPS のセッション・ルール

ｄ．打ち合わせ結果のまとめ方の注意

・打ち合わせ完了後は「決定事項の再確認、保留事項の次回までの検討計画、次回の打ち合わせ計画」をプロジェクトの進行段階（通常は、初期の形成期、混乱期、確立期を経て定着期に至る）に応じて、スケジュール、担当者を確認する。
・否定的なまとめ方をしない、簡潔にまとめる。会議のけじめをつける。準備不足だと感じたら再開に持ち込み作戦を立て再挑戦する。

（３）検証設計（自らの再確認）

（主として開発者が実行するが、条件提示段階で発注者側が試みることもある。）

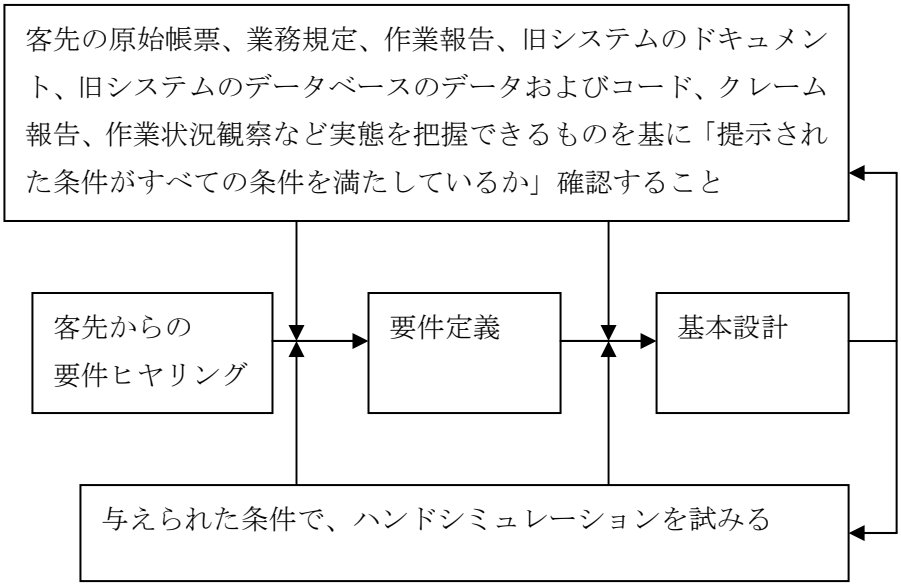
見積照会書に書かれた条件がすべての要件をみたしているか？、このままプログラムを作成した場合に、利用者が喜んで使ってもらえるシステムができあがるのか？、と疑いを持って仕様書を眺めて見る必要がある。この条件の「抜け」をいかにしてプログラム作成前にみつけるのか？

下図のように、

①実態を出来るだけ調査してみること

②提示された条件でハンドシミュレーションをして確認すること

の二つの手法がある。



図表 3-6-3　検証設計

＜注＞

１件のデータ入力で１件の情報が出力される場合は答の検証は入力に対する出力結果を確認すればよい。（ただし、処理条件のすべてが、要求仕様書に記載されているかどうかは？現場を視察調査し、使われている伝票や記録表を数日分チェックするなどの配慮が必要である。）

しかし、データ入力が複数回、あるいは長時間行われた結果が正解であるかどうか？を問われる場合は、その検証にはさらなる工夫が必要である。

さまざまな形の製品を積み上げた結果が出荷しやすい形になっているか？は積み上げるロジックが問題になる。複数の入力に対して結果も複数になる処理ロジックの検証はシミュレーションをしてみないと分からない。

複数のトラックが到着し荷物を複数の排出コンベヤーに流すロジックなどの検証にもこのシミュレーションが効果的である。本番でトライをするのには問題が多すぎるのであらかじめ紙のカードを作り処理通りのロジックを手で並べて確認するなどの慎重さが必要である。

ほんの少し注意深く「気配り」をするだけで、プログラム作成後のテストフェーズの修正がミニマムに抑えられる。

客先がすべての条件を整理し語ってくれていると勝手に思ったら、テスト段階やカットオーバー後の修正多発に直面する。

（４）仕様書・設計書の記述・確認のしかた（行間を読む方法）

以下の事項は、

・ユーザーが要求仕様や設計条件を定義する場合

・開発側がそれらを注意して読み取る場合

・そして、設計書に落とす場合

それぞれの場合に、注意しなければならない事項である。

①ユーザーは何をすればよいか？は記述するが、「どのように処理する手順は書かない」。

したがって不十分な記述が多い。入力フィールドの説明は通常は、新規のみ配慮してある。したがって訂正、取り消し、削除はベンダーまかせとなる。

②例外作業条件については、ユーザーはすべての条件を列挙する。その処置手法はベンダーとの相談事項となる。ルール化できない条件は異常作業であり、システムには取り込まない。

③名詞について：単数か複数か？を確認すること。

例：「出来た製品を倉庫に入れる」この文章を見て、

・倉庫は一つか、複数か？

・製品は一つ一つ入れるのか？ロットにまとめるのか？

などの確認をせねばならない

④形容詞、副詞について

あいまいな形容詞、副詞は、すべて確認する

＜例１＞

「十分な在庫が倉庫にあった場合は、その在庫から出荷する。」と仕様書に書かれているとして、「十分なとはなにを指しているか」の吟味が必要である。

（吟味）注文を 10 個受けた場面。

在庫が 20 個あった場合は倉庫の在庫から出荷する。

在庫がゼロなら、問屋に発注して入荷を待つ。

在庫が 7 個の場合はどうするか？7 個、取りあえず出荷してよいのか？

あと 3 個入荷するまで待ってから出荷するのか？

処置は複雑である。このような条件が文章の奥に含まれていることを、SE は見抜かなければならない。

＜例２＞

適切なレスポンスタイムを確保すること

厳密に整理すると、レスポンスタイムは次のどのレベルを望んでいるのか？を確認する必要がある。

ａ.「レスポンスタイム」の用語例

最も重要で複雑な一つのアプリケーションを動かして、このレスポンスタイムを一定限度の中に入るように要求する

ケース１：「受注登録プログラムを動かし、３秒以内に処理完了すること」
端末一台で入力作業をすべておこなう場合はこれでよいが複数台の端末から入力する場合は、入力作業が重なり、この条件だけでは満足できないことになる。

ケース２：「同時にＮ台の端末から入力を行い、90%以上が 3 秒以内で処理完了すること」
Ｎ台をいくつにするかはそのシステムに入力される端末総台数によって決まる。
仮にＮが 5 台だとすれば、3 秒以内に 5 入力が入る訳であるから、
１分間では 60÷3×5＝100 入力となり、１時間では 6,000 件処理できる相当に大きなシステムである。

ケース３:「プログラムＡ、プログラムＢの処理をしながら、同時にＮ台の端末から入力を行い、90%以上が 3 秒以内で処理完了すること」
通常は他の処理も平行的になされるので、その条件を明確にする必要がある。
プログラムＡ，Ｂの負荷影響度にもよるが、OS の優先度処理機能を活用しても、DISK へのリード、ライトが重なると、ケース２に比べて相当な、厳しい条件となる。

ケース４：「通常多種多様なプログラムが流れる中で、最も重要な入力処理の 90%以上が 3 秒以内で処理完了すること」

ケース３まではテストケースの設定が比較的簡単であるが、ケース４は条件設定が明確ではない。したがってどのプログラムとどのプログラムが重なった場合に遅れが発生するなど、条件の設定と解明が難しくなる。

ケース１～ケース４までの、どの条件が契約条件になるかによって、準備すべきハードウェア、ソフトウェアが著しく異なる。
本当にその条件が必要なのか？、テストをする場合に、証明するための条件設定が可能なのか？を確認して要求せねばならない。
開発者にソフトウェアだけを発注する場合は、複数データの同時処理をそのベンダーに確保させることは出来ない。OS、ネットワーク、データベースなどの条件が判明し、テスト条件を与えることが出来た場合のみ、設計条件として認められるので注意を要する。

⑤その他の表現注意事項

・出来るだけ能動態で書き、受動態を避ける。主語、述語が明確な方が誤解されない。

・要求をネスト形式の条件文で表現してはならない。

・用語を一貫して用いる。異なった人々が内容をまとめるときは、データ・ディクショナリーが有効になる。

・データ要素が「どこで作成され、どこで参照され、どこで削除されるか」を明確にするためには設計段階の CRUD 分析が有効である。

（５）進捗報告書の見方（真の実態を見通す方法）

進捗報告書を読み、その中からプロジェクトの問題を汲み取るのは相当な経験が必要となる。人には自分を格好よく見せよう、他と比較して自分だけ悪いと言われたくないなどの潜在意識がある。「問題です」と、正しく情報を上司にあげてくるケースはむしろ稀であると考えるほうが良い。だからと言って、進捗報告書をやめてしまえと言うことにはならない。プロジェクトの進捗状況の実態を読み取るコツを

ａ．問題に気がついている

ｂ．問題に気がついていない可能性がある

ｃ．問題に気がついているが、あえて報告しない

の３ケースに分けて整理したのが下表である。

報告書だけに頼らず、作業状況の観察や日常会話の中から関係者の表情を読み取り、実態を汲み取ることが必要である。複数の階層の関係者との会話を増やすことが肝要である。

課題区分	課題	原因・対策
ａ．問題には気づいている	①「やや遅れ」なおの曖昧な記述が多い	・１週間の遅れが発見できない計画を作成してある ・甘い管理姿勢がある
	②前回の報告と同じコメントが出てくる	・「進捗遅れ」を取り戻す施策や努力が足りない
	③何時になったら遅れを取り戻せるのかの回答の記述がない	・本人だけの努力だけでは回復は難しいとみてサポートをする必要がある
	④達成率をプログラム本数でカウントして難易度を見過ごしている。最終局面に高難易度の仕事が残り結局遅延する可能性がある	・難易度まで含めて進捗達成率をカウントする方式に変更する ・WBS ベースの進捗把握をする
ｂ．問題に気づいていない可能性がある	①問題の存在にすら気づいていない	・上位管理者は別ルートから問題を見つける工夫が必要 ・ATAM の採用
	②成果物の内容を十分に検査していないので問題が発見できない	・レビュー制度の充実と一定比率のレビュー時間確保 ・重点査読、抜き取り検査の実施など
ｃ．問題の存在に気がついてはいるが報告しない	①問題報告をすると追求・処罰が厳しいので最後の局面まで報告を避ける	・企業の風土・文化の改革が必要 ・契約方式を見直す必要がある （作業委任契約の場合は時間が長引けば支払額が増加する） ・プロジェクト管理者は自ら問題発見をする努力が必要

図表 3-6-4　進捗報告書の見方

（６）見積予算に入らなくなった場合の優先度付け

投資効果と投資費用が明確であれば、優先度付けは簡単であるが、実際の開発段階でこの試算を試みるのは、至難の技である。しかしおおよその費用、投資効果はシステム利用者、開発者が協議すればそれなりの順序づけは可能であり、経験は大きく狂っていることは少ない。

システムを設計し始めると、「あの機能も欲しい、これも必要」と利用者からはさまざまな要求が出て、当初の予算には入らないことになってくる。このような場合の調整に便利な方法がある。実際の投資金額、投資効果が分からなくても、機能の優先度付けにより、調整が出来る方法を紹介する。

各要求をいくつかの重み付けされたプロジェクト成功判定基準で評価して得点を計算して、要求の優先度付けをランクつける方法

（１）優先度付け会議の参加者
プロジェクトマネージャー：予算権を持つ顧客側のリーダー
利用者代表者　　：利益、不利益の格付けをする
開発代表者　　：費用とリスクの格付けをおこなう

（２）計算手順
・ユースケースから機能を取り出す。同じプログラムで実行できるものは集約し、分かりやすい説明文をつける
・次ページの表に評価点を 10 点満点で記入する。
・相対的利益、相対的不利益について相対的評価点を 10 点満点で記入する。
・リスクについても相対的リスク評価点をつける。
・重み付け基準にもとづき評価点を計算する。
・優先順位を計算する。

$$\text{優先順位} = \frac{\text{価値\%}}{(\text{費用\%} \cdot \text{費用重み}) + (\text{リスク\%} \cdot \text{リスク重み})}$$

・計算された優先順位の降順で機能一覧をソートする。一覧の最上位の機能は、最も有利な価値と費用とリスクのバランスが高いとみなされる。
・モデル予測した優先順と貴方が正しいと思う優先順位に著しい隔たりがある場合には重み係数を修正して再計算しモデル調整をおこなう。

（３）優先度付けマトリックスの例（次ページ）

図表 3-6-5　優先度付け手法[2]

[2] ＜出典＞Pardee 1996　総合的品質管理手法　TQM（Total Quality Management）

JUAS のセミナーの受講生募集システムを例にとって作成してみた。

この結果優先順位⑦の一部と⑩は後回しになった。

相対的 業務機能	相対的利益	相対的不利益	合計価値	価値%	相対的費用	費用%	相対的リスク	リスク%	費用＋リスク×0.5	優先順位１	優先順位２
相対的重み	2	1			1			0.5			
受講案内の送付	10	10	30	16.7	6	9.2	5	9.2	13.8	1.2②	1.8①
受講案内の検索	3	5	11	5.9	3	4.6	2	3.7	6.4	0.9③	1.3③
受講者情報の登録	10	10	30	16.5	8	12.3	8	14.8	19.7	0.9③	1.3③
受講者の紹介機能	5	5	15	8.2	10	15.4	8	14.8	22.8	0.4⑦	0.5⑦
受講者情報の訂正	10	10	30	16.5	8	12.3	6	11.1	17.9	0.9③	1.3③
受講者自身での登録内容の訂正	5	5	15	8.2	10	15.4	8	14.8	22.8	0.4⑦	0.5⑦
請求書の発行	10	8	28	15.4	6	9.2	3	5.6	12.0	1.3①	1.7②
受講者の分析	3	1	7	3.8	6	9.2	8	14.8	16.6	0.2⑩	0.4⑩
セミナー実績の登録	3	5	11	6.0	4	6.2	3	5.6	9.0	0.7⑥	1.0⑥
セミナー結果の分析	2	1	5	2.8	4	6.2	3	5.6	9.0	0.3⑨	0.5⑦
合計			182	100.0	65	100.0	54	100.0			

優先順位１：リスク配慮の優先順位、○内は順位

優先順位２：価値%／費用%を配慮の優先順位、○内は順位

＊直接金額まで評価しないが、そこそこの優先順位が得られるのがこの方法の特徴

図表 3-6-6　優先度付けマトリックス（JUAS の例）

（７）ＡＴＡＭ（Architecture Analysis Method）

リスクを回避するために、システム開発の計画の時点で ATAM を実施することを奨める。「このシステム開発を成功するためには何がリスクになるか？」をシステム開発の早期に見つける手法の一つである。下記のように、ATAM にはトップダウンとボトムアップの２つのアプローチがあるので、特長を活用して使い分ける。

ATAM1 は、システム開発の技術的な問題を見極める方法である。
プロジェクトマネージャーが主催者になり
「このシステム開発おいて一番問題になりそうなものは何か？」をシステム技術、アプリケーションのベテランを集めて議論し、あらかじめこのプロジェクトの実行上課題を議論し、整理して対策を採る。
初めて使用する技術、リスクの多い技術など心配ごとを早めに認識し、上手く使いこなすためのポイントを確認しておくのがこの ATAM１である。

＜ATAM１：トップダウン　アプローチ＞

＊このアプローチの目的は、アーキテクチャーに潜むリスクを発見することである。
下記の参加者がそれぞれの役割を担当する。

＜参加者＞	＜役割＞
プロジェクトマネジャー	全体調整役
ビジネス側代表	システム化の目的、機能概要、実施シナリオ等の説明 情報共有をはかる
アーキテクト	アーキテクチャーの説明
評価者（４人程度）	主として、アーキテクチャーに関する質問を投げかけ、 アーキテクチャーを分析評価する
書記	

＊進め方は以下のとおり。
・最初に全員で ATAM を理解するための講義を受ける
・次に、ビジネス側代表がシステム要件、ビジネスゴールなどシステムの内容を説明し、新機能の実現方法、ハードウェアとソフトウェアの構成、システムの品質目標と実現方法などを全員で理解する
・その結果、システムの品質に関しての心配事、関心事を列挙して、それをツリー図に整理する
・その際、重要性、難易度、トレードオフの関係について重点的に整理する
（例）パフォーマンス向上は→メモリー増設→コスト高
（例）リスクが存在するのに書いてない→その理由は？

図表 3-6-7　トップダウン　アプローチ

ATAM2 は、このシステムに関係する全部の部門から代表を集め各人が考えているリスクを列挙し、問題認識を共通化する。ともすれば「あれも欲しい、この機能も必要」と各部門代表は要求しがちであるが、リスクを発見するのが第一目標であり、機能要求会議ではないことを認識させリスクの議論に徹する。

＜ATAM２：ボトムアップ　アプローチ＞

＊このアプローチの目的は、全システム関係者から、このシステムについてのリスクを聞き出すことにある。
＊下記の参加者は、利用者、需要家、マーケッター、その他のステクホルダー、および多数の評価者（３０人に及ぶこともある）である。
＊進め方は以下のとおり。
・最初に全員で ATAM を理解するための講義を受ける
・次にビジネスの説明、アーキテクチャーの説明を行い、全員で理解する
・その際、要望事項の聴取ではなく、リスクの発見を第一の狙いとして、いかなる質問、シナリオ、リスクに関する自発的な発言を歓迎して進める
・また、常に事業目標、品質属性、技術的手段に照らし合わせて理解を促す。
・理解が進んだ状況になった後に、重要性、難易度を投票によって決める
・ATAM 評価の総括としての結果説明には十分な時間をかける

図表 3-6-8　ボトムアップ　アプローチ

この ATAM 手法は JUAS の若手メンバーが CMU を 2002 年に訪問し、講義を受け取得した方法である。

（８）ＡＤＤ（Attribute Driven Design Method）

ADD は、システムの品質属性の視点で、システム・アーキテクチャーをレビューする手法である。

システムを構造分析した後、六つの品質特性を各サブモジュールの中にどのように組み込むかを順番に議論し明確にする。相互に矛盾した要求になることもあるので、何回かこの特性議論を繰り返して、矛盾をなくす。

サブシステム分割が済んだからと言って、直ぐにコーディングに入るのではなく、上記特性をどのように組み込むのか？を議論し、例えばシステム変更が頻発するモジュール、サブシステムはどこか？そのサブシステムはどのような構造にして変更容易性を満たすのか？などの議論を行うので、品質の高いシステムが出来上がる。

下記のステップを繰り返し実施ことによって、目標とする品質属性を実現できるアーキテクチャーであるかどうかを吟味できる。

ADD（Attribute Driven Design Method）

STEP1：アーキテクチャーの設計要素の選択
検討対象のシステムをサブシステムに徐々に分解して対象にする。
STEP2：設計要素の分割、実現方法の選択
①アーキテクチャー・ドライバーの選択
プライオリティの高い**品質属性**を対象に取り上げる。
②タクティックス（tactics：実現方法）の選択
アーキテクチャー・ドライバーを達成する「実現方法」を選択する。
③要素分解：機能を達成するために**機能分割**を行う。
④分解要素の制約条件ならびに品質属性要件の見直し
一度の分解では解決せず、何段階かにわたって徐々に解決する品質属性もある。このような場合は、分解要素に新たな制約条件や、品質属性が付加される可能性がある。

＜６つの品質属性＞

- 可用性（Availability）
- 変更容易性（Modifiability）
- 効率（Performance）
- 安全性（Security）
- テスト容易性（Testability）
- 利用容易性（Usability）

＜機能分割＞

モジュール分解（Module decomposition）、同時並行性（Concurrency）および関連性（Deployment）の視点で設計を進めドキュメント化し、設計要素間のインターフェースを明確にする

STEP3：次の設計要素を検討対象にして、STEP1、STEP2 を繰り返す。

図表 3-6-9　ADD（Attribute Driven Design Method）

この ADD 手法は JUAS の若手メンバーが CMU を 2003 年に訪問し､講義を受け取得した方法である。

２．開発テスト段階のプロジェクトマネジメント

（１）局面化開発

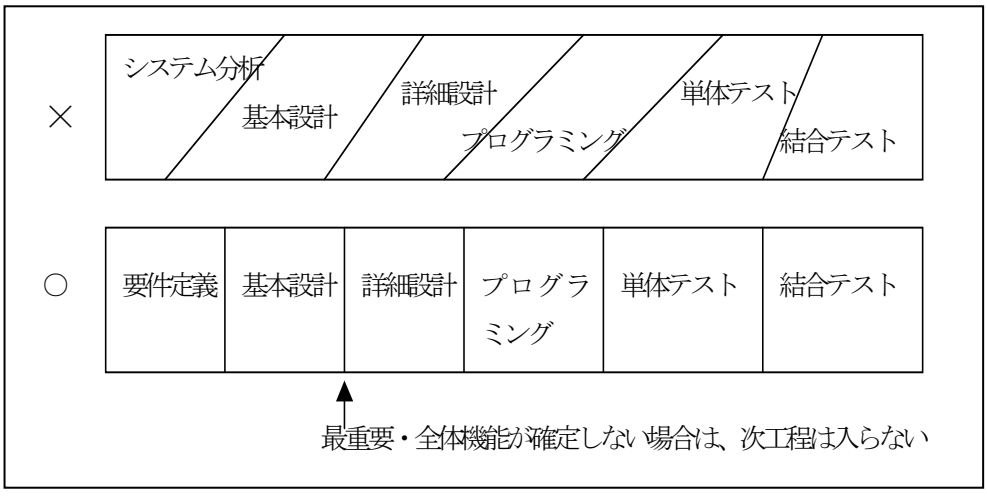
大日程計画表あるいは中日程計画表が、下図の上段のように「一つの工程が全作業について終了していないのに、ある作業は次工程に入っている」状況にしないようにするのが、この極意である。決定が遅れている部分の仕様が、先行している部分の内容に影響する時、先行作業に修正が発生するのを防ごうとする仕組みである。これを局面化開発と呼ぶ。

特にプログラム作成（コーディング）作業に入った後、仕様の変更、追加が起こるとその負担は大きくなるので、他のグループのメンバーが遅れている部分の設計支援に入って設計を急ぐなどの柔軟なスケジュール調整とプロジェクト推進が要求される。

後工程での欠陥発見は、数百倍の修正負荷！ できるだけ、前工程で発見を！

＜要点＞

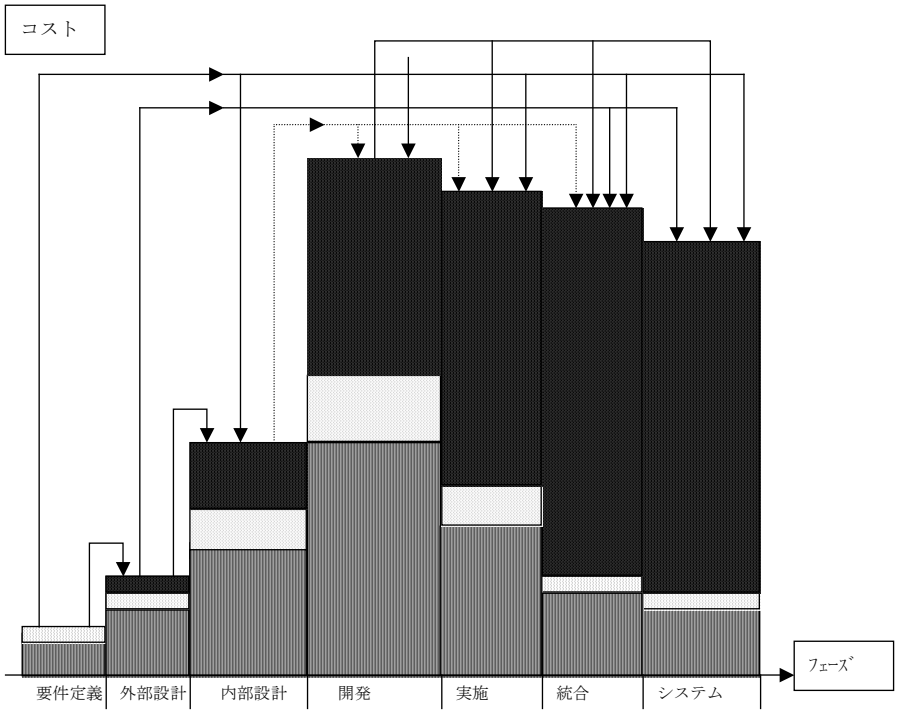
・プロジェクトを実行可能かつ管理可能な大工程に分割
・大工程の境界では、全サブシステムが、工程内の作業が完了しないと次工程開発に入らない
・局面毎の品質確保
・試行錯誤の防止
・併行作業の円滑化



図表 3-6-10　局面化開発

＜手戻りコストの概念図＞

日本アイ・ビー・エム株式会社では、各開発工程における手戻りコストの概念図を準備しているので以下に掲載しておく。



図表 3-6-11　手戻りコスト概念図

（２）進捗状況の確認

進捗遅延の典型的状況とその原因をまとめたものが下表である。プロジェクトの開始時に進捗管理の計画を作成する際のチェックリストとして利用すれば、事前に予防対策を採ることができる。

段階	作業項目	典型的な進捗遅延状況とその原因	対策
開発準備	ビジネスシステム仕様書の作成	・仕様の細部が決まらない ・社内関係先の担当者が期限を守らない	徹底的な WBS による作業の細分化とフォロー
	プロジェクトチームの編成と発足	・社内体制が決まらない ・社内関係先の業務調整が遅れる	ユーザー側プロジェクトリーダーの職責階層を上げて推進体制を見直す
	契約形態､条件の設定	・同じ議論が繰り返される ・ユーザー側の希望が曖昧	同上
	開発手法､成果物の特定と決定	同上	同上＋スタッフの充実
	開発環境の設定と確保	同上	同上
開発作業	開発実施	・特定のサブチームの進捗が遅れる ・配置要員の欠落が出る。	メンバーの補充、編成替えを行う
		・特定のサブチームの進捗が遅れる ・仕様変更が多発している	重点レビューの実施、メンバーの入れ替え
		・他システムとのインターフェースの開発が遅れる ・他システム関係者が期限を守らない	リーダーによる重点管理
	成果物の確認	・修正のフィードバックが遅れる ・成果物確認担当者が期限を守らない	小日程表の作成と徹底
	進捗管理	・同一の原因による遅延が発生する ・進捗管理の状況情報が伝えられていない	事前に情報共有の仕組みを作り、日常的にそれを作動させる
移行準備	移行計画(要件定義時)	・同じ議論が繰り返される ・ユーザー側の希望が曖昧	U字型開発法の採用、リーダーによる方針決定
	移行計画の決定	・同じ議論が繰り返される ・ユーザー側の希望が曖昧	同上、EA の参照と徹底
	移行作業	・データコンバージョンが遅れる ・担当者が他業務と兼務	U字型開発法の採用、データコンバージョンチームを別に設定
		・利用者教育が遅れる ・利用者の時間が取れない	教育チームを別の編成し準備に当たらせる

	運用･保守体制の決定	・同じ議論が繰り返される ・ユーザー側の希望が曖昧	保守運用チームのキーマンを指定し開発の最初から参画させる
カットオーバーと運用	トラブルシュート	･トラブルシュートが決着しない ・分担範囲が曖昧	トラブル情報をキーマンに集中させてアクションの透明性・迅速性を高める
		・トラブルが頻発する ・前工程の管理が杜撰	本来はこのような状況が発生しないようにすべきであり、もしこのような状況が発生したならば、ユーザー、ベンダーともに社を上げてベストの緊急体制を敷き、当事者だけの対策としない

図表 3-6-12　進捗状況の確認

一般的に、プロジェクトの規模が大きくなるほどプロジェクト管理の難易度は指数的に増大するといわれているが、工程進捗の管理に関しては、上表の対策の欄にあるように、遅延を自覚してからではなく、予防的に事前に遅延対策を実施することでカバーできる。

（３）品質目標の提示

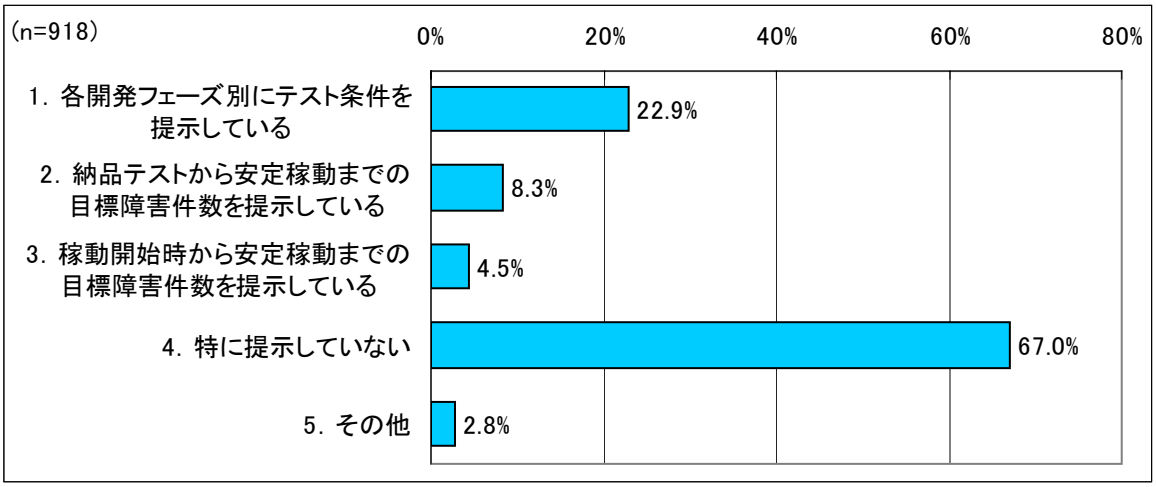
日本の製造業の各社はハードウェアの製造について品質目標を持っており、例えば「百万個の部品について欠陥は一個以下におさえよう」などと言われているシグマ戦略なる目標も出ている。

ではソフトウェア産業における品質目標はどのようになっているのだろうか？ISO、CMM などの考え方が示され「開発の各工程においてこのようなことをしなさい」と造り込みのための作業は規定されている。「何をしなさい」は定められていても「誰がしなさい」「目標値をここにおきなさい」などの具体的項目は規定されていない。それも影響してか、ソフトウェアの品質についての目標値を規定した資料はない。

「CMM のレベル５を取りました」とさる有名な､外国のソフトウェア会社がやってきた。「500 万円に一個以下の潜在欠陥に抑えて納入してくれるなら発注しますが、目標を守れますか？」と質問したところ、やや考えてから「そのような注文には応じられません。注文はいりません」との答えであった。「よくわかっているな」と私は逆に評価した。自社の実力を計数化し評価していないとこのような即答は出来ない。

ベンダーに「潜在欠陥 XX%以下にして納入して欲しい」と発注条件の一つに品質目標を明記するユーザー企業が増えれば、日本のソフトウェア品質は高くなりシステムトラブルは減少し、システムへの信頼性は向上してくる。

「品質の目標管理をしているか」についてのデータがある。

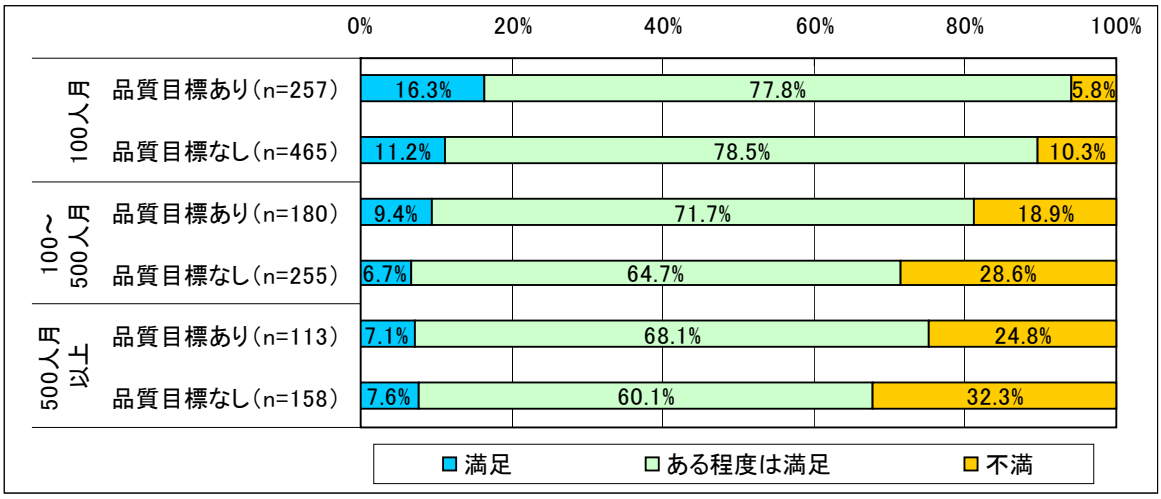


（IT 動向調査 2005）

図表 3-6-13　システム開発の外部委託先に対する品質目標の提示

特に設定をしていない企業は全体の２／３の 67.0%であった。テスト条件を決めているのが 22.9%、障害件数のような定量的な品質の目標を決めているところは、わずか 10%以下にとどまった

では、こうした品質目標は、実際のプロジェクトの仕上がりの満足度に役に立っているのだろうか。



（IT 動向調査 2005）

図表 3-6-14　品質目標と提示と品質満足度の関係

何らかの品質管理の目標を持っている企業が、仕上がりに不満を持つ比率は、100 人月以下の小規模プロジェクトで 5.8%、100～500 人月の中規模プロジェクトで 18.9%、500 人月以上の大規模プロジェクトで 24.8%なのに対し、目標を持っていない企業では、それぞれ 10.3%、28.6%、32.3%と明らかに不満が多い

満足度と言う主観的な物差しではなく、実際の工期や予算はどうだろうか。データは割愛するが、やはり目標を持っていない企業の方が、プロジェクトの工期が遅れることが多いと回答している。

一方、予算では、品質目標を持っていると答えた企業の予算の超過の割合は、小規模プロジェクトでは 6.6%で、持っていない企業の 15.2%に対し明らかに少ないが、中規模になるとその差は小さくなり、大規模ではむしろ逆転している。これは、プロジェクトの規模が大きくなればなるほど、ソフトウェアの開発の進捗管理だけでは予算を制御しきれないことを考えれば納得がゆく。

いずれにしても、何らかの品質目標を持っている企業のほうが、プロジェクトをうまく進行させて満足度も高いという裏づけが得られた。

しかし、こうした定量的な物差しを持った管理を目指している少数の先進的な企業は例外として、大多数の企業の開発プロジェクトの現状は、科学的なプロジェクト管理を云々する以前の状況にあり、ベンダーに任せきりになっていることがこの一連の調査で明らかになっている。

（４）品質管理目標

IPA（独立行政法人　情報処理開発機構）では、ソフトウェアの各テスト工程ごとにテスト項目設定割合の基準値とテストで検出される欠陥数割合の基準値をプロジェクトマネージャーの研修用テキストに下記のように掲載している。

＜テスト項目設定の基準値＞

テスト工程	基準値
単体テスト	50項目/実効KS　以上
総合テスト	20項目/実効KS　以上
システムテスト	10項目/実効KS　以上
運用テスト	3項目/実効KS　以上

＜テスト評価（欠陥数）の基準値＞

テスト工程	基準値
単体テスト	15件/実効KS
総合テスト	5件/実効KS
システムテスト	2件/実効KS
運用テスト	0.1件/実効KS

＜実績値と基準値の比較による品質判定＞

	検出バグ数＜テスト評価基準値	検出バグ数＞テスト評価基準値
実施テスト項目数＜テスト項目設定基準値	？　検出バグ数が少ないのは実施テスト項目数が不足しているのが原因と考えられるため、テスト項目を追加して継続する	X　実施テスト項目数が不足で、かつ検出バグ数が基準を超えているのは明らかに品質不良であり、項目追加の上テストを継続する
実施テスト項目数＞テスト項目設定基準値	◎　テスト実施項目も十分であり、検出バグ数も基準を満たしているので品質は良好だと思われる。バグの収束予測を行い、テスト終了の可否判定を行う	？　一概に品質不良であるとは言えない。バグの収束予測を行い、テスト終了の判定を行う

＜注意＞

開発手法、言語、データベース、ケースツールの採用などによって、上記の値は異なるので、その都度修正して活用すること

図表 3-6-15　欠陥数割合の基準値

（５）最短工期の出し方

システムの開発工期は、開発作業量、開発作業の生産性と投入する開発人員に依存する。何かの理由でカットオーバー時期が限定される場合には、その時期に合わせて上記の３つの要素を調整することになる。

開発作業量は、実現する機能の範囲により変動し、同一の機能範囲であっても、スパイラルアプローチ、プロトタイプアプローチ、フェーズ分けをしたウォーターフォールアプローチなど採用する実現方式およびプログラムの難易度によって変動する。

また、作業量を表現する単位は、ファンクションポイント数（機能実現の難易度を加味した機能数表現）または、作成されるプログラムのライン数が代表的な単位であるが、既成のパッケージを利用する場合の評価はそれらとは異なるなど多様である。

開発作業の生産性は、投入１人月あたりの開発作業量をもって表現される。開発作業量がファンクションポイント数によって表現されるのであれば、その作業量を生産性（投入１人月あたりのファンクションポイント数）で除すことにより所要の投入工数（人月）が導かれる。また、開発作業量が作成されるプログラムのライン数によって表現されるのであれば、同様に、その作業量を生産性（投入１人月あたりの作成されるプログラムのライン数）で除すことにより所要の投入工数（人月）が導かれる。

一般的には設計・製作の作業工期は、上で求められた投入所要工数の総和を投入可能な開発人員で除して、月単位の所要工期として求めることができる。しかし、この工期は最短工期の目安のひとつである。

実際には、作業項目と所要期間については、WBS（Work Breakdown Structure）や
PERT（Program Evaluation and Revue Technique）を用いてその実現性の確認をすることを奨める。

特別に指定される者でないと開発できないプログラムが含まれている場合や設計・製作の作業以外に工期を律速する要件（たとえば、ハードウェアの納期）が有る場合には、プロジェクトの工期はそれに従うことになる。

（６）テストの効果を高めるポイント

開発工期の中でも、テストの工期の比重は無視できない。テスト工期を有効に活用するための工夫がある。下記を参考にしてほしい。

＜計画準備段階の鉄則＞
（ａ）テスト計画の策定はシステム設計時に着手せよ
（ｂ）テストケース作りには現場を巻き込め
（ｃ）テストもメリハリが大事、優先順位をつけよ
（ｄ）Web 系の製品はバージョンに、レガシー系は既存機能に落とし穴
（ｅ）稼動後に見つかったバグをテストケース作りに生かせ（再発防止）
（ｆ）総合テスト用の DB のコンバージョン・プログラム作成を単体テスト開始時に間に合わせよ
（g）単体テスト完了時に、顧客に結果を提示せよ。

＜実施段階の鉄則＞
（ａ）プログラムを動かす前の見なおしも効果がある
　　目視再検査は有効である
（ｂ）進捗管理の効率化は必修、ツールの導入も一考せよ
　　実施漏れや精度の確認が可能になる
　　全員に変更管理条件などが徹底される
（ｃ）見逃し防止、精度向上のためにマークペンを活用せよ
　　単純ミスは黄色、重要なミス、テストケース不足は赤など色を使い分けて
　　変化を持たせてテストを楽しめば、テスト結果の確認精度が上がる

（７）上手くいっています→ドボンの回避策

大規模システムにおいては、システム開発の最後の局面になってから、「予定通りになっていない」ことが発見されるケースが時々起こる。さまざまな進捗状況報告制度を作って、システム開発の実態を知りたいと努力しても実態を把握するのはなかなか難しいものでる。

「こんな月次報告制度を作った」「厳密に週間報告を求めた」にもかかわらず、最後の総合テストに入ったら、「このシステムは使いものにならない」「全く使い物にならないレベルの品質である」などの実態が暴露されて結局システム開発が失敗した経験者も多いのではあるまいか。つまり、どのような報告制度を作っても実態とは異なる報告であれば、とんでもない事態に陥ることになる。

建築などのプロジェクト管理と違って実態が見えにくいこともそのような事態に陥る一つの原因ではあると思うが、「実態を見抜く努力を報告制度に求めたプロジェクトマネージャーにも問題はある」ことを正しく認識して欲しい。

ところでプロジェクト関係者全員が「このまま進めていったら大変な問題になる」と感じてい

ないとは思えない。いくら見えにくいシステム開発といえども「必ず誰かは、このままでは危ない」と警鐘を鳴らしているものである。問題はそれを素直に聞く耳を持っていたかどうか？である。

プロジェクトの関係者は次の 10 種類存在している。

①顧客内上級管理者、②顧客窓口、③顧客内利用者

④開発担当 SE、⑤プロジェクトマネージャー、⑥SE 部門マネージャー、

⑦他の開発グループメンバー⑧レビュースタッフ、⑨システム営業

⑩外注会社の管理者，SE

プロジェクト管理の問題は「報告書から見つけるのではなく、誰も発言はしていないが、ここが問題になることを感じ取ることが出来る」すなわち「自分でチームメンバーの行動から問題を感じ取る」リーダーがあってこそ成功するものである。報告書に載ってきてから問題であると言い出すマネージャーは三流である。

（１） 10 階層による進捗状況のチェック
「誰かが不安、不信を感じている」
「アラームは鳴っている。PM に聞こえるかどうか？素直に聞けるかどうか？」
プロジェクトマネージャーの上司は，顧客のトップ層、外注会社の上司に定期的に会い「自分の把握している実態と彼らの実態把握感が一致しているか確認をせよ」

（２） 計画との対比
「１週間の遅れがチェックできる計画が作成してあるかどうか？」
「作成結果の中味を吟味してあるかどうか？」
「開発進捗率の明細定義とデジタルな実績把握がなされているか？」

（３） 五感を使った進捗度把握を実施せよ
「週報などの報告書は、真の異常発見には役に立たない。
管理者自らが、現場で問題の種を見つけよ」

（４） プロジェクトマネージャーの経験規模を重視せよ
1 億円規模のプロジェクト管理の経験者が 5 億円のプロジェクト管理を出来る確率は、１／５に低下する。

（５） 新規顧客に接する場合は２倍の神経、情報網、感度が必要

（６） 桶の法則を作用することを理解せよ
ほとんどのプログラムが OK であっても、何処か一つのプログラムが不良でシステムがダウンすることはある。完璧を期せ。

（８）仕様変更管理

開発が長期間にわたると、業務そのものが環境に合わせて変化してくる。検討を重ねてくると、ユーザー側にも知恵がつき「もっと良い方法が見つかったので採用したい」など、局面化開発を推進してきたが、なお仕様変更をしたい要望が多発してくるのが、一般的である。これには次のように三つの方法で対処したい。

（ａ）仕様変更が発生しないような仕様決定方法を採る
（ｂ）新しい要望が発生した場合には、変更を最小化して対応できる方法を考える
（ｃ）仕様変更への戦略的対応をする

（ａ）仕様変更が発生しないような仕様決定方法を採る
２→４→２→３の法則
最初の約束を顧客リーダーに伝え、ムダなヒヤリングを避ける
実態の状況を調査
他社での開発経験を活かす
既存ファイルのデータ内容をチェック
ペーパプロトタイプ開発
画面シミュレーターの採用
DOA（Data Oriented Approach）の活用
現状データのコンバージョンを単体テスト開始時完了
＆　そのデータを単体テストフェーズにフル活用

（ｂ）新しい要望が発生した場合には、変更を最小化して対応できる方法を考える
承認業務の階層数を減らし、業務をシンプル化
CRC（Class Responsibility Collaboration）カード
（重複業務、重複開発を避ける）
画面操作性に必要以上こだわらない。
画面とロジックの分離（フレームワークの活用・外部テーブル）
変更に強い開発手法採用（ただし特徴をとらえた活用を）
簡単なプログラムであれば、思いきって再作成する

（ｃ）仕様変更への戦略的・組織的対応をする

＊仕様変更ルールを設ける

（必要不可欠なものに絞ってプロジェクトリーダーが許可したもののみ対応し、担当者の個別承認は認めない）

仕様変更は一件ごとに予算も配慮して採用可否をユーザー側は承認するが、実行側は、その都度変更すべきもの（設計段階にはこの種の問題が多い）とまとめて変更処置をした方が良いもの（プログラム開発に入ってからはこのアクションの方が効果的なものが多い）に分けて対応する

一つ一つの変更をせず、まとめて変更処理期間を設ける

（アッセンブリーやコンパイルの効率を図る）

プログラマーが傍らにいる間に、簡単な変更は依頼する

（単体テスト後に利用者に画面を見せる）

（９）開発高管理

「今月このプロジェクトの基本設計に 10 人投入したので 10 人月の生産が出来たハズ」とする進捗管理方式を採用しているケースが多いのではあるまいか？

「私は何時間作業しました。したがってＸＸ円予算を使いました」

「では成果は予定通り上がったの？」「？？？分かりません」

「予算は使いきったけど、仕事はまだ半分も終わっていない」

「残業時間が多すぎて赤字になるが、どこまで赤字が増えるのかわからない」

など、よく聞くせりふである。

ほとんどどが「一週間の遅れを見つけることが出来ない計画を作成している」ことに起因しているが、すべてのプロジェクトで WBS を完全に作成することは大変な努力を要する。

WBS を完全に作成できなくても、問題が発生していることを会計制度からも早期に発見できる仕組みを持つことは必要なことである。

簡単に投入人月と出来高の関係を把握でき、早期問題発見が出来る仕組みが、この開発高管理である。

手順：

①開発予算あるいは受注金額から、設計フェーズ別に投入可能人月と予算を算出する。

（これをフェーズ別計画開発高と呼ぶ)。さらに投入可能人月を月ごとに割り振る。

②当月消費した人月を予算に換算して製造原価に計上する。

SE 個人別の単価を採用するか？SE 単価をランク別に分けて計上するか？平均予算単価を使うか？は、各社の環境に合わせて使い分けすればよい。

直営 SE，外注 SE の単価を別な方法で算出することもある

③当月の開発高は次式により求める。

当月フェーズ別開発高＝（当月累積進捗率―前月累積進捗率）×フェーズ別計画開発高

④当月累積進捗率はプロジェクトマネージャーがフェーズ別に実態を把握して指定する。

WBS に基づいて算出しても良いし、WBS が無ければ実際の進捗を作成ドキュメント量や作成プログラム数から推定する。

慣れてくればプロジェクトマネージャーのサジ加減も多少は効くが、サジ加減が出来るようなプロジェクトマネージャーは一流である。

予定通りと思っていたが、見直した結果進捗率が逆戻りした場合には、当月フェーズ別開発高は、マイナスになって表示される。

プロジェクト完了時には、当月フェーズ別開発高の合計が受注金額になる。

フェーズ	人 月 区分	ＳＥ、ＰＧランク工数					Ｃ フェーズ 製造 原価	Ｐ フェーズ別計画 開発高	Ｊ 前月迄 累計 開発高	Ｋ 当月迄 累計 開発高	Ｊ－Ｋ 当月 開発高
		Ａ	Ｂ	Ｃ	Ｄ	Ｅ					
基本設計	当月計画										
	当月実績									当月進捗率Ｍ	
	当月累積								前月 累積 進捗率	当月 累積 進捗率	
詳細設計											
Ｐｇ											
テスト											
フォロー											

ＳＥ単価表}			
ＳＥランク	Ｇｒ1	Ｇｒ2	Ｇｒ3
Ａ			
Ｂ			
Ｃ			
Ｄ			
Ｅ			

SE ランク工数×SE ランク単価＝製造原価
単価にはプロジェクトの人件費
直接経費及び SE 部のスタッフ費用が含まれる

・　受注価格×Ｃｎ／ΣＣ＝Ｐｎ（フェーズ別計画開発高）

ΣＣプロジェクト全費用，Ｃｎ　フェーズ別費用

・　Ｐｎ×（当月累積進捗率－前月累積進捗率）＝当月フェーズ別開発高

（当月累積進捗率－前月累積進捗率）は、後戻りすれば、－になる場合あり

・　Σ当月フェーズ別開発高＝当月開発高

図表 3-6-16　開発高管理

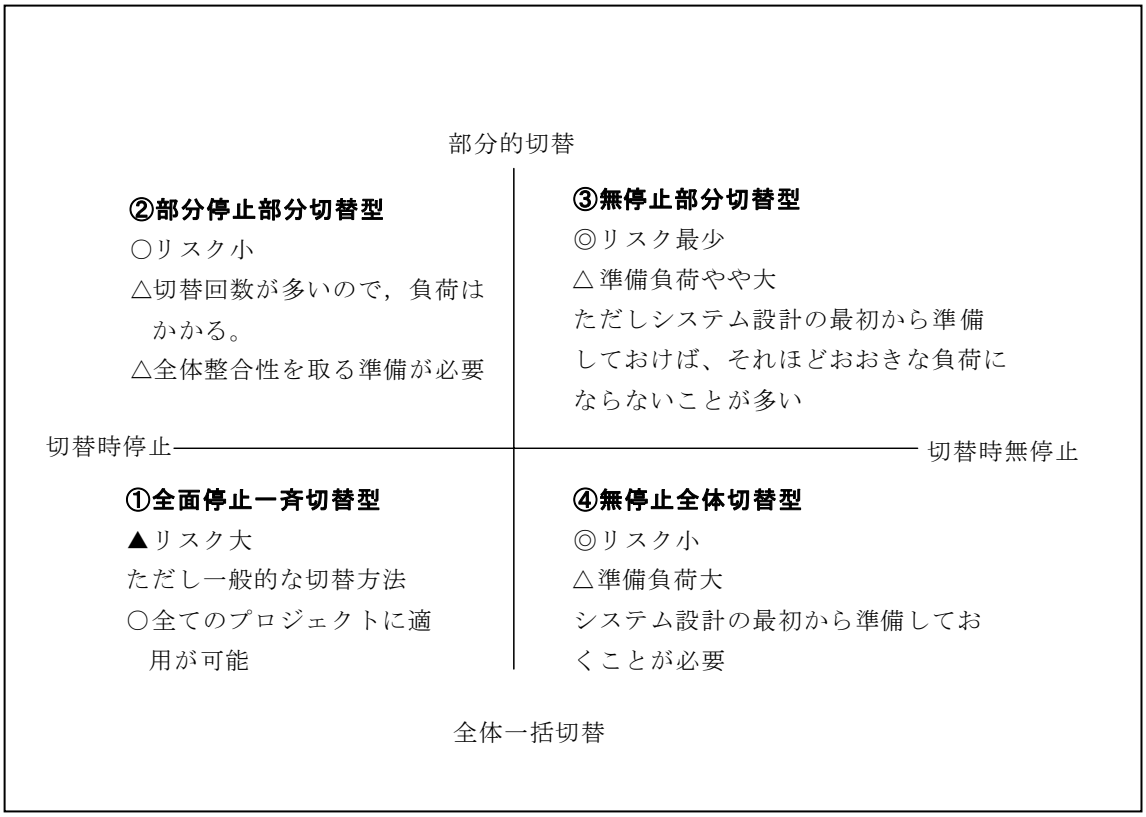
＜WBS と開発高管理方法との違い＞
①必ずしも WBS が準備できなくても良い
（新規プロジェクトの場合は、全ての WBS を準備するのは大変）
②ある作業工程の中間でも進捗率を指定することにより吸収することが出来る
③進捗率の申請はプロジェクトマネージャーの「さじ加減」が可能であり、開発高を貯金することもできる。逆に問題があるのに、正直に申請しないと、問題発見が遅れる
④精緻な管理を必要とする場合は WBS をもとにした EVM(Earned Value Management)を活用のこと

３．システムの切り替え段階のプロジェクトマネジメント

（１）カットオーバー（サービスイン）の条件

（ａ）プロジェクト開始時（又は、基本設計完了時）に移行計画および稼動開始条件を設定する。
・品質目標を設定し、実績を管理する
・システム／ユーザー部門／運転部門の各々の稼動条件を明確化する
（ｂ）プログラムの全ルートにテストデータを通す。
できれば「カバレージモニター」で確認してあることが望ましい。
（ｃ）インターフェースを含めて関係システムとの正確性、処理性（処理時間、レスポンスタイムなど）を確認する。
・日次処理がその日に終わることを確認する
・実機環境とシミュレーション環境でのストレステストを実施する
（ｄ）ハードエラーを人為的に起こさないと確認できない場合の対策を検討する。
（ｅ）出力書類を実際に使用して、運用上の問題が発生しないことを確認する。
（ｆ）万全の対策を取ったつもりでも、問題が発生することはあり得るので、その準備をする。
（ｇ）SE は、稼動開始した日は定時（残業しない）で帰宅できるという自信をもって作業に当たる。
（ｈ）プロジェクトの責任者は部下に具体的に質問しその結果を聞いて稼動開始可否を判断する。
（ｉ）それでも問題が発生した場合には、「問題は３倍根深い」と考えて対処する。

（２）システム切り替え方法



図表 3-6-17　システムの切替方法

システム切替方式は、切替時に他のシステムを含む全体を停止するか、停止しないか、および、新システム全体を一括して切替えるか、部分的に切替えるか、の組み合わせによって４通りの方式がある。

①全面停止一斉切替型
②部分停止部分切替型
③無停止部分切替型
④無停止全体切替型

それぞれの得失は、上の図に書かれているとおりである。どの方式が相応しいかについて事前に検討し、それに応じた準備をすることが必要である。

無停止切替型については「そんなことは出来ない」と反論がベテラン SE から出そうである。実際に実行してみると、「こんな楽な切り替えは始めて・・・」と愁眉を開くことになる。いつでもテストが出来、品質が満足するレベルに至った時に切り替えればよいのであるから、精神的な負担は少ない。ただし多少のシステム切り替えのためのプログラムの準備は必要である。

すべてのシステムでこの方式が採用できるとは考えにくいが、システムの再構築をする場合に、一度は議論して欲しい方法である。

（参考）イテレーション型開発の注意点

スパイラル型、あるいはプロトタイプ型を一歩進めて整理した開発法である。

＊次の４つのフェーズに分けて開発する

・方向付けフェーズ（Inception）

・推敲フェーズ（Elaboration）

・実装フェーズ（Construction）

・移行フェーズ（Transition）

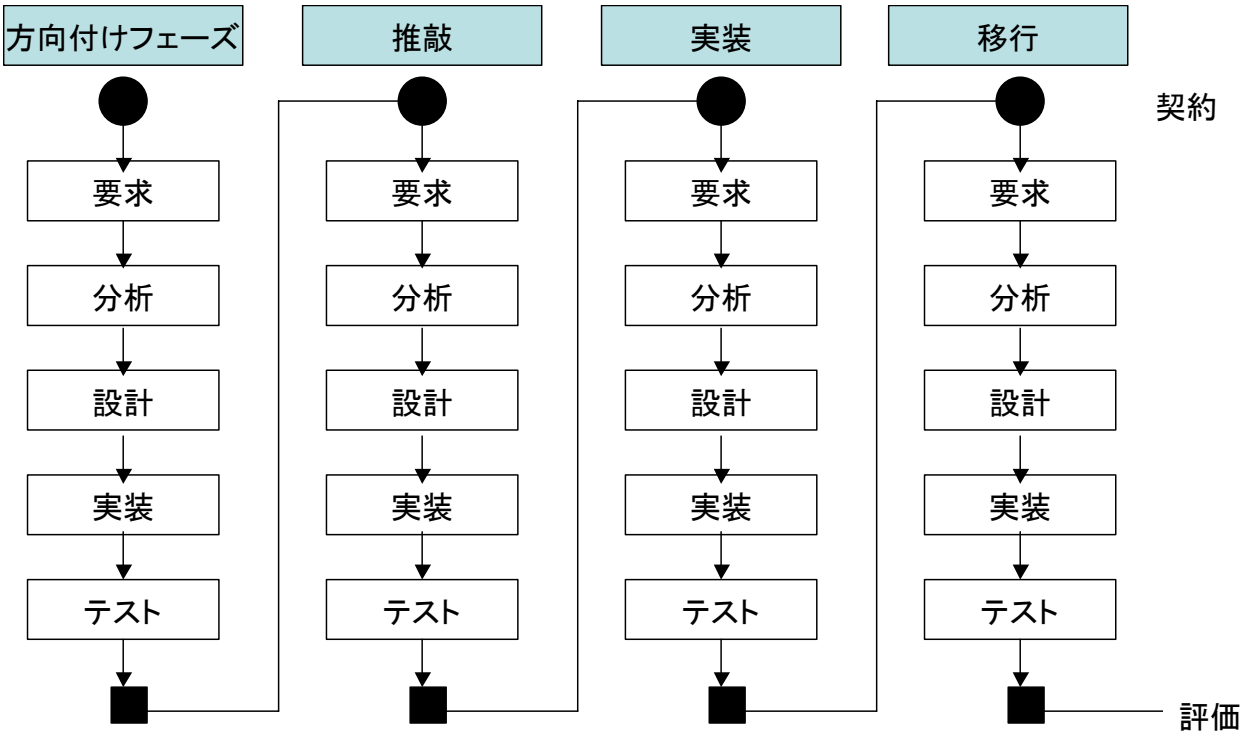
＊各フェーズに、要求、分析、設計、実装、テストの作業を順次進める

＊実装作業があるので、問題を確認し解決しながら開発を進める

＊ユーザー要求、解決法などが明確でない、効率が明らかでない場合に有効である

（たとえば、まったく新しいビジネスモデルを創造するとき、新技術を採用し、

その機能、性能、品質などが不明のとき）



●は契約を意味する。各フェーズの最初にゴール地点を明確にして進めること。

■は評価を表す。各フェーズの最後に評価を実施し、契約に対する到達地点の確認をしてから、次の契約に入ること

図表 3-6-18　イテレーション型開発

第３章　開発

第７節　開発生産性指標

システム開発の世界で良いシステムを作成する方式として次の二つは有名である。

①開発の局面で作りこみのための作業を実施する。（例ＣＭＭ,ＩＳＯなど）

②プロジェクト管理技術を向上させる（例ＰＭＢＯＫなど）

しかし製造業などではＰＤＣＡに見られるがごとく、実行結果をチェックし対策をとる考え方が一般的に採用されている。

最近ソフトウェア開発の世界にも、ＥＡＳＥ（Empirical Approach to Software Engineering ）にみられるように、

③実績を評価し対策をとる方法の重要性が叫ばれてきた。

ＪＵＡＳでは、その必要性を３年前から提唱してきたが、2004 年にやっと陽の目を見ることになった。経済産業省の後押し、ＩＰＡとの協調があり、ユーザー企業におけるシステム開発の実績分析が実施できた。その結果、興味ある有効な知見が得られたので、ここに報告し、解説したい。

１．システム開発の生産性指標の意義

物には品質に応じた価額の相場がある。

システム開発商品の相場観とは何か？を整理してみたい。

ユーザー満足度を議論する中で、基本サービスの指標には３つの要素（品質・工期・生産性）があることがわかった。

良いシステムには優れた機能・品質が備わっており、工期、価額も合格点を取る必要がある。ではこの合格点とはどのようなものか？

この３要素を測る標準値を一般企業のプロジェクト実績から求めようとした。その結果をソフトウェアメトリックス調査報告書として 2005 年 4 月にまとめた。この中の要点を品質・工期・生産性毎に紹介してみたい。

これには、下記の活用法がある。

①各社において「開発したプロジェクトの工期・品質・生産性は、世の中の標準と比較してどのレベルにあるのか？」の目安を知ることができ、活用することができる。

②ベンダーにこの目標値を提示して、ユーザーとベンダーとの共通目標として活用できる。

③得られた各種知見をプロジェクトマネジメントに利用できる。

２．４つの項目の使い分け

（１）ベンダーとユーザーのデータの一致

	ユーザー	ベンダー
①1 社あたりのプロジェクト数	1 社あたり大型プロジェクトは数件／年程度しかない。なおかつ実績を詳細に集めている企業は少ない	1 社あたり、数百万以上のプロジェクトがある。なおかつ各社に品質データ収集組織を持ち集めやすい
②１プロジェクトあたりの規模	今回の平均は 2.1 億円 中央値は、3663 万円	一次請負企業のプロジェクト金額は大きいが、二次以下になると細分化され小規模となる
③プロジェクトデータの収集	企業数を増やさないとデータ数は集められない 今回：40 社 133 件	企業数が少なくてもプロジェクト数は集められる 今回：15 社 1009 件
④プロジェクト予算	明確であり、計画投入人月とともに提示可能である	ユーザーの予算はわからない
⑤開発過程の作業詳細データ	RFP 提示後は請負が多く、すべてベンダー任せとなる。自社を除けば、開発の詳細はわからない	開発過程の詳細データの収集は可能
⑥ノウハウの提示姿勢	開発ノウハウを社外に出すことについてはオープンな企業が多い。出さなければ得るものも少ない	開発ノウハウを社外に出すことについてはクローズである
⑦ユーザー満足度	評価把握が可能である	明確にはわからない

図表 3-7-1　データ収集についてのベンダーとユーザーの差

調査に当たっては次の２点が議論となった。

①図表 3-7-1 に見られるように、ユーザー企業のデータとベンダーのデータの性格がこんなにも異なっていたことは新鮮な驚きでもあった。

②調査項目については当初できる限り一致させようと努力した結果、質問数の増加になってユーザー企業には回答の負担を強いることになった。

また、努力してはみたけれど、全く同じ質問に作り変えられたのは、全質問の半分も無かった。

（２）ＦＰ、ＬＯＣ、人月、金額の有効活用

比較項目		FP	LOC	人月	価格
①この機能の価格はいくらか？	実績のあるスクラッチ	◎　概算評価　画面数、帳票数を基に試算可能	○過去の実績からの推定	○　過去の実績からの推定	○　過去の実績からの推定
	実績の無いもの	(GUIの複雑性の評価に難あり)	○　画面数、帳票数を基に試算可能(GUIの複雑性	△LOCから試算可能	△人月から試算可能
	パッケージ	×ユーザーは評価困難 FP，LOCはベンダーしか判らない	×ユーザーは評価困難 FP，LOCはベンダーしか判らない	×ユーザーは評価困難 ベンダーのマネージメントしだい	○横並び評価は可能
②工期試算		◎FPから人月さらに工期(COCOMO法など)	・LOCから人月換算	◎人月→工期	△過去の実績からの推定
③生産性評価		○　総FPと投入人月の概算評価は可能 ○　詳細設計～UT迄は個別評価も可能	○総LOCと投入人月の概算評価は可能	○FP/人月、LOC/人月の概算評価は可能	¥/FPあるいは¥/LOC
④品質評価	スクラッチ	◎欠陥数/FPが可能	◎欠陥数/ベンダー指定言語のLOCが可能	◎欠陥数/人月が可能	◎欠陥数/価格が可能
	パッケージ本体（ユーザーの立場で）	×自社で見つけた欠陥数は可能(部分的評価)	×自社で見つけた欠陥数(部分的評価)	×自社で見つけた欠陥数(部分的評価)	○欠陥数/価格で概要評価
	パッケージの追加、修正	◎欠陥数/FPが可能	○欠陥数/ベンダー指定言語のLOCは可能	◎欠陥数/人月が可能	◎欠陥数/追加分のための価格で概要評価
スケジュール管理	基本設計～完了	×作業計画に反映しがたい	×作業計画に反映しがたい	◎WBSで人月使用可能	◎EVMで人月、価格あわせて使用可能

図表 3-7-2　見積要因比較

当初の議論では「FP を基にデータを集めて整理すべし」なる主張が強く出されたが、

・「JUAS での調査では FP 法の利用は 20%以下」となっており、ユーザーにおける活用範囲が限られていること

・FP 法を使って日程計画表は作成しないこと

・FP 法も絶対的ではなくプログラマーの作業量は LOC で測った方が現実的なこと

・ユーザーはお金のデータはわかるが、FP，LOC は必ずしも判っていないこと

などを考え図表 3-7-2 のように４要素（FP、LOC、人月、価格）別に見積要因を整理して、特徴に応じて正しく使い分けることを主張した。

これは結果的には正しい判断であったと思っているが、FP 法を活用しないとどうしても解けない課題があることも指摘しておきたい。

「この機能がこの値段で高いか？安いかは機能を基に判断した方が良い」

「工期が短かったのか？長かったのか？を判断するのも、今回は人月をもとに判断しているが、FP 法をもとに判断する方法もある」

「FP 法は、パッケージには適用がし難い、スケジュール作成に直接利用し難い、変動要因の吸収が各社別であることなどの課題がある」

しかし、FP 法でないと解けない問題があることも事実である。別途の計測手法の開発とともに FP 法の進歩を期待したい。

３．調査データの概要

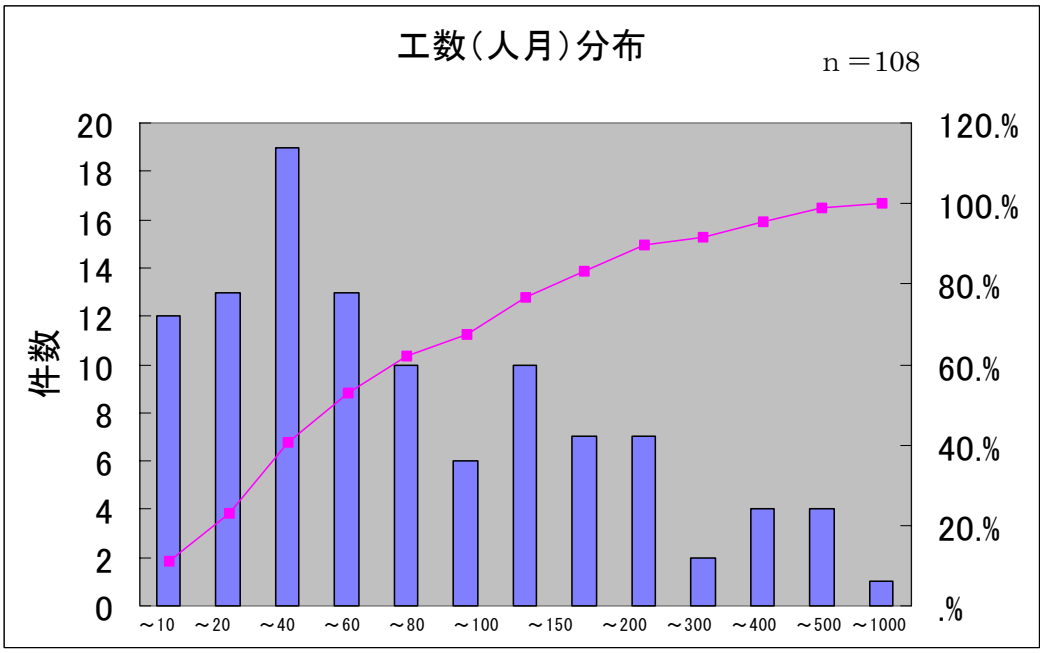
調査に先立って JUAS に 16 社の協力を得て開発生産性プロジェクトを立ち上げた。

このプロジェクトメンバーには

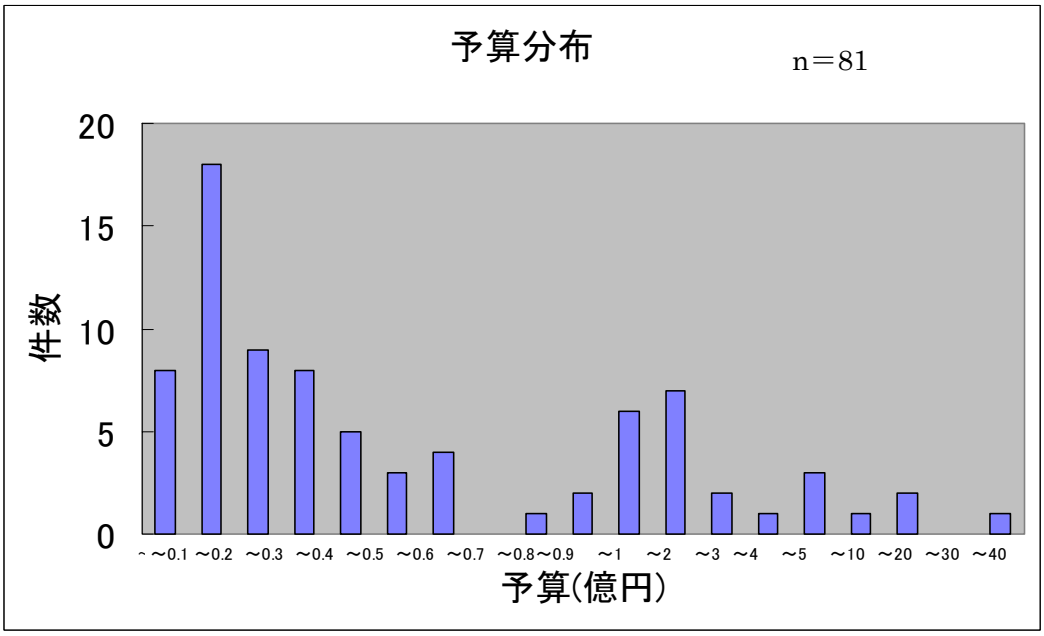
①質問の妥当性の評価　②仮説の評価　③質問表の評価　④データの提供⑤結果の分析の評価　などにご協力いただいた。

優秀なメンバーに支えられ、10 回にわたる議論を通して有意義な知見を頂いた結果は、素晴らしい調査としてまとめられた。

調査によって収集されたプロジェクトデータの工数分布と予算分布は下図のとおりであった。



図表 3-7-3　基本統計量と分布　工数（人月）分布

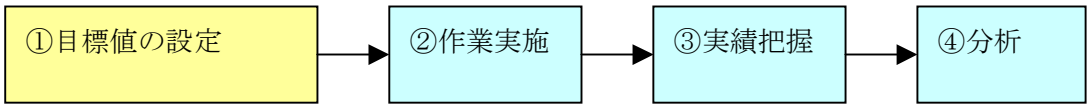


図表 3-7-4　基本統計量と分布　予算分布

４．調査分析についての考察

（１）目標値の設定

品質、工期、生産性について目標値をもって作業した場合と、特に目標値を持たない場合とでは、結果において大きな差が出てくる。



今回は各社の実態を知るためにデータ収集を行っているので、目標値をもっている企業も持っていない企業も混在している。したがってデータを分析してみると、当然のことながら、大きなバラツキが発生してくる。

目標値をもって作業した場合と持たないで作業した場合の結果の比較を後述してある。目標値の設定により、実績が向上するとともに、評価値のバラツキが減少することを期待したい。

（２）仮説と設問

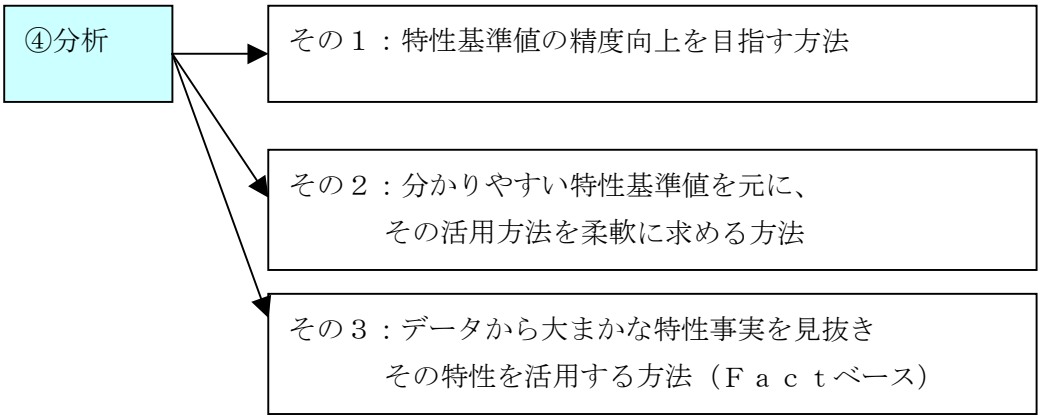
調査アンケートの設問の裏には、仮説が存在している。「プロジェクトマネージャのレベルとプロジェクトの成功の間には相関関係がある」「優秀な経験豊かなプロジェクトマネージャーが担当したプロジェクトは品質も良く、ユーザー満足度が高い」などの意見は一般には存在するがデータで示されたものはない。

これを証明するためには「品質データ」「ベンダーのプロジェクトマネージャーの経験度」「ユーザーのプロジェクトマネージャーの経験度」「ユーザー満足度」などのデータをクロス分析する必要がある。あらかじめ仮説を重んじすぎると、重要な要素を見失う可能性もある。

これらの要素を考え、JUAS のシステム開発生産性研究プロジェクトでは、あらかじめこの質問集で問題が無いか？仮アンケートを行い確認した後に本番アンケートを実施した。いくつかの反省を取り入れたおかげで、設問のレベルは向上したが、まだまだ修正すべき箇所は本番調査で判明した。

（３）分析方法

分析方法には、３つの考え方がある。



その１：特性基準値の精度向上を目指す方法

Ａ＝ｂ・c^{x}　などの仮説式を立てて係数を求める方法である。仮説を立て、データを解析し特性を解明する。

工期と投入工数の関係においては次のような式が一般に使用されている。

工期Ａ＝2.7×（人月）$^{0.318}$　の 0.318 が適しているのか？それとも 0.351 の方が適しているのかを、データの分散分析に基づき追求する方法である。

ソフトウェア工学でもこの手法が良く採用されているが、特定の集団が、いつも定められたメンバーが開発を実施する場合ならともかく、常に新しいテーマを、その場その場で集められたメンバーが、特別な目標も与えられず、毎回異なる仕様に基づき開発している現状データを、詳細に分析すればするほど、混乱し悩みが深くなり、泥沼に陥る可能性がある。

このプロセスは必要ではあるが、的を絞らずに、一般から広くデータを集め解析する場合には、大まかな特性分析で満足する程度でよい。

企業別に、分野を絞り、特定の集団の実績分析を行うならば精度向上の意味が出てくる。ソフトウェア開発の品質、生産性に及ぼす要因は非常に多く、なおかつそれが個々に目標値も無く作業した結果は「ばらつく」のが当然であり、このようなデータをもとに上記係数の精度向上を検討するよりは大まかな特性を捉えてその活用法を柔軟に求めて行くことが肝心である。

- <u>その２：分かりやすい特性基準値を元に、その活用方法を柔軟に求める方法</u>

その１で求められた、何らかの分析結果を基準におき、各プロジェクトでは、その基準との差を意識して利用する方法である。「基準が無いよりは何かあれば一つの目安になる」との見解で基準を利用する方法である。

前出の式は、

工期Ａ＝2×（人月）$^{1/3}$　として使いやすくする。

「標準工期は投入工数の立方根の２倍」と覚えやすく、かつ、計算しやすくする。「1000 人月のプロジェクトは 10 の 3 乗であるから、10×2＝20 ヶ月を標準とする」のように計算すればよい。慣れれば暗算で行うことも出来る。

ユーザー企業で実用化するには、このセンスが必要となる。「システム開発の工期とは、お客がいつまでに開発してほしい」との要望に基づいて決定される。「標準式で計算すれば 20 ヶ月必要となるが、お客の要望が 15 ヶ月であるならば、25%短いことに着目し、前回 20%短いプロジェクトを開発した時の対策より、もう少し何か対策を増やさないと上手く行かない、とみて対策を強化する必要がある」などの、一つの目安として活用できる。

実はこの JUAS が提唱している上記の立方根の法則は Boem の COCOMO 法から借用したものである。べき乗の精度を求めず、むしろこの標準からの差で難易度を判断する考え方にすれば、COCOMO 法も使いやすいものになる。

「COCOMO 法は当社のプロジェクトには適してない」と判断する前に、このように使いこなして欲しい。

その３：特性を活用する方法（Fact ベース）

因果関係を統計解析し原因と対策の関係を追求するだけでなく、基本的特性を見抜きその結果を利用する方法である。

上記工期の例でいえば、「当社では標準工期よりも 50%短いプロジェクトは破綻するのでそのようなプロジェクトは実施しない」などと活用することである。大まかなデータ分析からでも、このような事実を発見できる。

「ベンダー側のプロジェクトマネージャーが未経験な場合はシステム品質が悪い」「ユーザー側プロジェクトマネージャーの経験度はシステム品質に影響しない」などの事実を正しく認識し広く役立てれば良い。

「数値解析にのみ頼らず、知見を見つけ出し、そのノウハウを活用する」ことも有効な対策の一つである。

５．工期分析

（１）工期の標準

工期については、プロジェクト全体工数と、全体工期がともに記入されている 105 プロジェクトについて、工期の 3 乗根と工期の関係をグラフ化し、回帰直線を引いた結果、

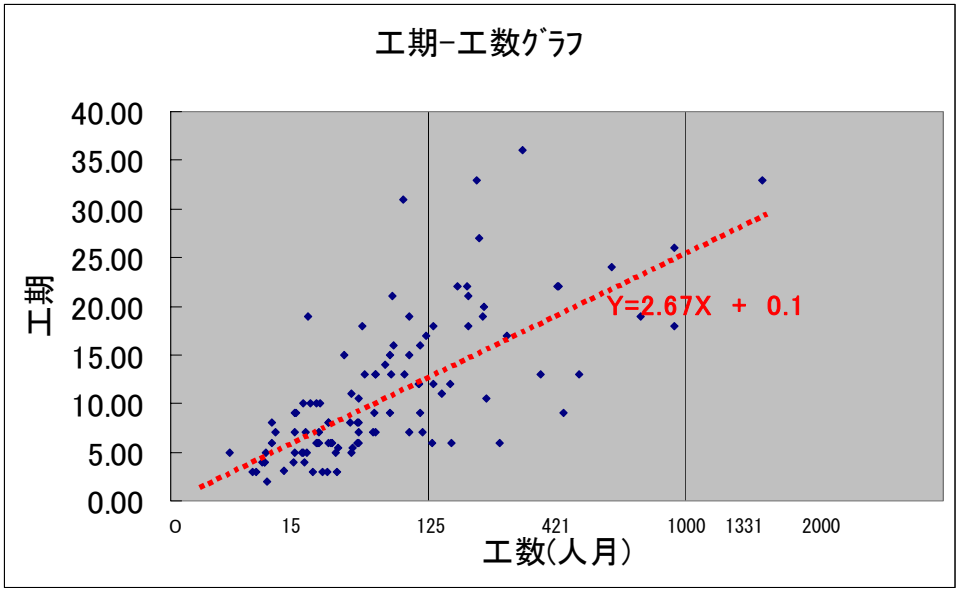
工期＝2.67×（投入人月の立方根）+0.1 となった。

切片を除いた近似式は

工期＝2.7×（投入人月の立方根）となり COCOMO 法とほぼ同じ結果が出た。

日本の工期はもっと短いと当初は予測していたが、諸外国並みの結果となっている。

プロジェクト全体工数と、全体工期がともに記入されている 105 プロジェクトについて、工数の 3 乗根と工期の関係をグラフ化し、回帰直線を引いた。工期・工数共に、実績の回答がある場合には実績の工期・工数を、計画しかない場合には計画工期・工数を採用している。その意味では、ほぼ実績ベースの分析と言える。



図表 3-7-5　工数分布

その結果、回帰の有意性が確認され、回帰直線が、Y=2.67X+0.1 と求められた。

Y 切片をゼロとして回帰をし直すと、Y＝2.69X となり、ほぼ **Y＝ 2.7X** である。

（X は工期の 3 乗根）

（２）工期過不足率

次に顧客から要請されている工期と標準工期の比率（工期過不足率）を求める。

工期過不足率＝1-（顧客からの要請工期÷標準工期）

この過不足率を基に対策を考える。プロジェクトの標準工期に対する過不足率と対策の関係、工期過不足率と品質の関係、プロジェクト終了時のユーザー満足度の関係などを、蓄積しておけば、各企業でのプロジェクト失敗は少なくなる。工期短縮率対策の例は、第 5 章第 1 節の「ユーザー満足度」を参考にしていただきたい。

（３）工期遅延率

工期の計画値、実績値がともに記入されているプロジェクトは 133 件中 95 件であった。

（計画工期　－　実績工期）　／　計画工期　を工期遅延度と定義してプロジェクト規模別の遅延度分析をおこなった。

			遅延度							遅延度
			予定より早い	予定通り	10%未満	20%未満	50%未満	それ以上	総計	20%以上の割合
規模（工数）	～10人月	件数		11					11	
		比率	0.0%	100.0%	0.0%	0.0%	0.0%	0.0%	100.0%	0.0%
	～50人月	件数	1	28	3	2	1	3	38	
		比率	2.6%	73.7%	7.9%	5.3%	2.6%	7.9%	100.0%	10.5%
	～100人月	件数		8	4	1	1		14	
		比率	0.0%	57.1%	28.6%	7.1%	7.1%	0.0%	100.0%	7.1%
	～500人月	件数		19	2	1	3	1	26	
		比率	0.0%	73.1%	7.7%	3.8%	11.5%	3.8%	100.0%	15.4%
	500人月以上	件数		3	1	2			6	
		比率	0.0%	48.0%	16.0%	36.0%	0.0%	0.0%	100.0%	0.0%
	総計	件数	1	69	10	6	5	4	95	
		比率	0.6%	68.9%	11.7%	8.9%	6.7%	3.3%	100.0%	10.0%

図表 3-7-7　プロジェクト規模別の遅延度分析

そのうち、予定通りの工期を確保できた割合は、**70%**であった。大規模プロジェクトほど遅延度が高いとは言い切れず、なすべきことをなせば予定確保が可能である。

工期遅延理由の上位２つ、**44%**が「要求仕様決定の遅れ」と「要求分析作業不十分」という要求定義フェーズの原因であった。要求仕様の内容充実と要件定義の工期確保が必要である。失敗したプロジェクトのデータが調査票にて報告されていない事も考えられる。

（４）工期遅延理由

その他を除くと、上位 2 つが、要件定義フェーズに原因があると回答している。（全体の４割は要件定義に問題があって遅延した。）要求仕様書の内容充実と要件定義の工期確保が必要。上位工程での不具合が、全体工期の遅延につながる恐れが最も多いことがわかる。残りはチーム編成と戦力の問題が多い。

工期遅延理由別の件数

工期遅延理由	規模(工数)						
	10人月未満	50人月未満	100人月未満	500人月未満	500人月以上	記入なし	合計
1.システム化目的不適当		1					1
2.RFP内容不適当		1			1	1	3
3.要件仕様の決定遅れ	1	5	2	6		3	17
4.要件分析作業不十分		6	1	4	1	1	13
5.自社内メンバーの選択不適当		2			2		4
6.発注会社選択ミス		1	1	1			3
7.構築チーム能力不足		3		2	1	2	8
8.テスト計画不十分		3	2	1	1		7
9.受入検査不十分				1			1
10.総合テストの不足		2	1		1		4
11.プロジェクトマネージャーの管理不足		2		1		1	4
12.その他	2	3	3	3	1	2	14
合計	3	29	10	19	8	10	79

図表 3-7-8　工期遅延理由別の件数

（５）基本設計、構築、テストの工期比率

規模別工期比率については、**設計工期：実装工期：テスト工期＝3:3:4** となり、テスト比率が高い結果となった。

長いテスト工期を短縮する方法の第一の方法は、EA(Enterprise Architecture)図の下部に「システム設計の最初から本番への切り替え方法を検討すること」と解説されているごとく、切り替え方法を踏まえてのテスト手法を検討しておくことである。

その２は「ユーザビリティを基本設計で検証し、単体テスト完了時に微調整を行う開発手法を確立すること」である。（第３章第５節　「実行計画」のU字型開発法を参照）

つまり総合テスト時に、使い勝手や運用の容易性の修正作業をしなくても良いように、前工程で準備完了しておくことが肝心である。

その３は、「本来単体テストで発見すべき欠陥は、確実に単体テストで取り除いておくこと」である。旧システムから新システムへの切り替えをするために準備するコンバージョンシステムを単体テストの開始時期までに、早めに作成してコンバージョンされたデータも活用すると、単体テスト結果は良い品質のものになると同時に総合テスト期間も短縮できる。

プロジェクトで、設計、実装、テストにそれぞれどの位の比率で工期を配分しているかを見るために、プロジェクト規模別に、フェーズ別工期の比をみるための、各フェーズ別平均工期の分析を行った。

ここで、

設計工期＝要件定義＋外部設計

実装工期＝内部設計＋製作

テスト工期＝結合テスト＋

総合テスト１(ベンダー内テスト)＋総合テスト２(顧客内の総合テスト)

と定義した。

ＰＪ規模（工数）	工期/件数	設計工期	実装工期	テスト工期	テスト比率
10 人月未満	平均工期(月)	2.17	1.83	4.00	50.0%
	件数	3	3	3	
50 人月未満	平均工期(月)	3.84	3.17	4.94	41.3%
	件数	24	24	24	
100 人月未満	平均工期(月)	3.10	4.50	6.35	45.5%
	件数	10	10	10	
500 人月未満	平均工期(月)	5.54	6.97	9.81	43.9%
	件数	18	18	18	
500 人月以上	平均工期(月)	13.00	9.67	12.00	34.6%
	件数	3	3	3	
記入なし	平均工期(月)	5.69	4.75	6.63	38.8%
	件数	8	8	8	
総計	平均工期(月)	4.76	4.83	6.96	42.1%
	件数	66	66	66	
	比率	3	3	4	

図表 3-7-9　フェーズ別工期

（６）ユーザーＰＭ業務精通度と工期遅延率

		遅延率							
PM 業務(U)	データ	-1	0	1	2	3	4	総計	3+4
1	件数	1	36	3	1	2		43	
	割合	2.33%	83.72%	6.98%	2.33%	4.65%	0.00%	100.00%	4.65%
2	件数	1	28	5	5	2	2	43	
	割合	2.33%	65.12%	11.63%	11.63%	4.65%	4.65%	100.00%	9.30%
3	件数		3	2	1	2	2	10	
	割合	0.00%	30.00%	20.00%	10.00%	20.00%	20.00%	100.00%	40.00%
記入なし	件数		5			1		6	
	割合	0.00%	83.33%	0.00%	0.00%	16.67%	0.00%	100.00%	16.67%
合計	件数	2	72	10	7	7	4	102	
	割合	1.96%	70.59%	9.80%	6.86%	6.86%	3.92%	100.00%	

PM 業務精通（選択肢）	
1.十分精通していた	3.精通していたとはいえない
2. ある程度のレベルまでは精通していた	4.全く経験も知識もなかった

遅延率（係数）			
-1	予定より早い	2	遅延 20%未満
0	予定通り	3	遅延 50%未満
1	遅延 10%未満	4	遅延 50%以上

図表 3-7-10　ユーザーPM 業務精通度と工期遅延率

ユーザープロジェクトマネージャーの業務精通度が高いと、工期遅延が発生しにくい。

６．品質分析

（１）品質目標の必要性

日本の製造業の各社はハードウェアの製造について品質目標を持っており、例えば「百万個の部品について欠陥は一個以下におさえよう」などと言われているシグマ戦略なる目標も出ている。

ではソフトウェア産業における品質目標はどのようになっているのだろうか？ISO、CMM などの考え方が示され「開発の各工程においてこのようなことをしなさい」と造り込みのための作業は規定されている。「何をしなさい」は定められていても「誰がしなさい」「目標値をここにおきなさい」などの具体的項目は規定されていない。それも影響してか、ソフトウェアの品質についての目標値を規定した資料はない。

「CMM の 5 をとりました」とさる有名な、外国のソフトウェア会社がやってきた。「500 万円に一個以下の潜在欠陥に抑えて納入してくれるなら発注しますが、目標を守れますか？」と質問したところ、やや考えてから「そのような注文には応じられません。注文はいりません」との答えであった。「よくわかっているな」と私は逆に評価した。自社の実力を計数化し評価していないとこのような即答は出来ない。

ベンダーに「潜在欠陥 XX%以下にして納入して欲しい」と発注条件の一つに品質目標を明記するユーザー企業が増えれば、日本のソフトウェア品質は高くなりシステムトラブルは減少し、システムへの信頼性は向上してくる。

（２）品質の定義と実績

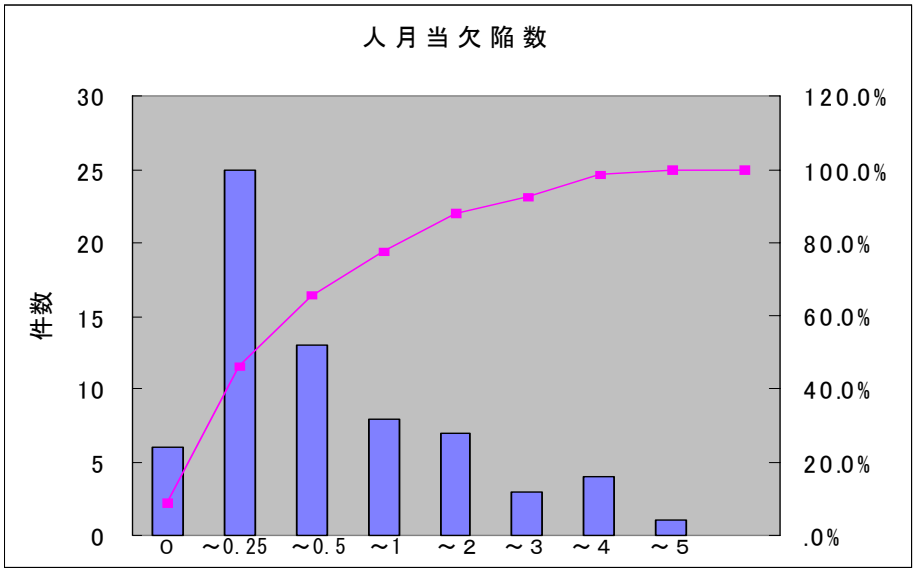
ユーザー側から見た品質の定義は、「ユーザーが発見した欠陥数の密度＝受入テストから安定稼動までの期間に発生した欠陥数」である。今回、欠陥率が計算できたデータは、67 プロジェクトであり、欠陥数の平均値は、**１人月あたり 0.7 件**であった。つまり、5 人月あたり 3.5 個のバグということになる。

およそ 500 万円あたり、1 件以内に納まっているプロジェクトの比率は、40%程度である。これは一つの目安値として、活用できる。

ユーザー側から見たソフトウェア品質の尺度として、「受入検査から総合テストを経て、安定稼動に至る間に、ユーザーが発見した欠陥数の密度」という考えに基づき、欠陥数 PER_工数（人月当欠陥数）という概念を導出した

欠陥率＝欠陥数 PER_工数＝総合テスト２～フォローのフェーズ(受入テスト～安定稼動まで)で発見された不具合の数÷プロジェクト全体工数

欠陥率が計算できたプロジェクト（不具合数、工数ともに記入されている回答数）は 133 件中 67 件であった。その基本統計情報と、分布を以下に示す。



欠陥率(人月当欠陥数)	
平均	0.700659701
標準誤差	0.125086246
中央値(ﾒｼﾞｱﾝ)	0.2806
最頻値(ﾓｰﾄﾞ)	0
標準偏差	1.023875052
分散	1.048320122
尖度	4.662189817
歪度	2.143158971
範囲	5
最小	0
最大	5
合計	46.9442
標本数	67
信頼区間(95.0%)	0.249742658

図表 3-7-11　人月当欠陥数

平均値は**１人月あたり 0.7 件の欠陥数**である。（5 人月あたり 3.5 個のバグ）上記分布を鑑み、今後の分析において、プロジェクトの品質のランクを次のように分類することにする。

欠陥数 PER_工数＝0
欠陥数 PER_工数＝0.25 未満
欠陥数 PER_工数＝0.25～0.5 まで
欠陥数 PER_工数＝0.5～1 まで
欠陥数 PER_工数＝1～3 まで
欠陥数 PER_工数＝　3 以上

	欠陥率(欠陥数PER_工数)						
	0	0.25未満	0.5未満	1未満	3未満	3以上	計
件数	6	25	13	8	10	5	67
比率	9.0%	34.3%	22.4%	11.9%	14.9%	7.5%	100.0%

0.25(人月)以下、およそ 500 万円あたり 1 件以内に納まっているプロジェクト比率は、43%程度である。

（３）品質目標の有無とその効果

133 プロジェクトの中で、品質基準を持って開発にあったっている割合は、43.6%である。

（JUAS の IT 動向調査 2004 では､67%の企業が品質目標を提示して発注していないことも分かっている）

今回はレベルの高い企業の回答が多かったと推察される。

更に、品質基準の有無と欠陥率を比べてみると、品質基準を持っていないプロジェクトでは、**欠陥率が 1.42 倍**になることが判明した。目標を持って取り組むことが、品質アップにつながることを物語っている。

欠陥率の計算できた 67 プロジェクトについて、品質基準の有無と欠陥率の関係を調べた。

	品質基準			
	有り	無し	記入なし	計
件数	38	28	1	67
比率	56.7%	41.8%	1.5%	100.0%
平均欠陥率	0.60	0.85	0.33	0.70
最大欠陥率	3.13	5.00	0.33	5.00
最小欠陥率	0.00	0.00	0.33	0.00

図表 3-7-12　品質基準の有無と欠陥率

品質目標を持っていたプロジェクトと目標が無いプロジェクトでは、発生欠陥率において平均 0.25 件/人月の差があった。

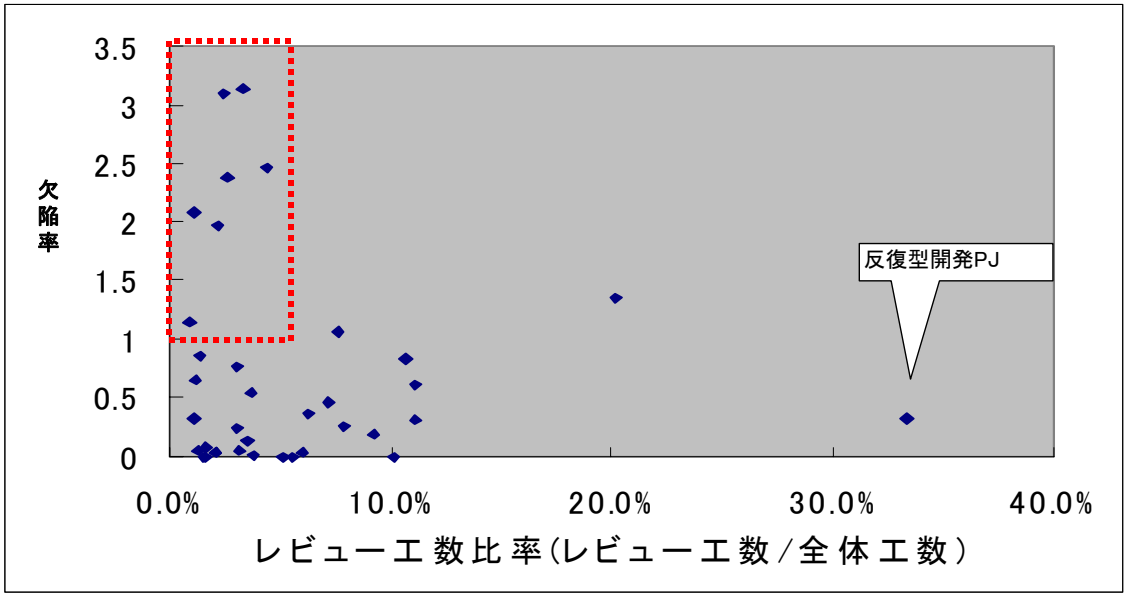
（４）レビューと欠陥率

レビューと欠陥率について、35 プロジェクトについてレビュー工数比率が計算できた。その結果、5%以下のプロジェクトでは欠陥率が 1 個/人月以上になる確率が 30%程度発生する。逆に、レビュー比率を 8%以上確保したプロジェクトには、欠陥率の多いものは少ない。

レビュー工数比率＝レビュー工数÷プロジェクト合計工数にて、換算した。

全抽出サンプル（n=36）

欠陥率が計算できた 67 プロジェクトのうち、36 プロジェクトについてレビュー工数比率が計算できた。



図表 3-7-13　レビュー工数比率

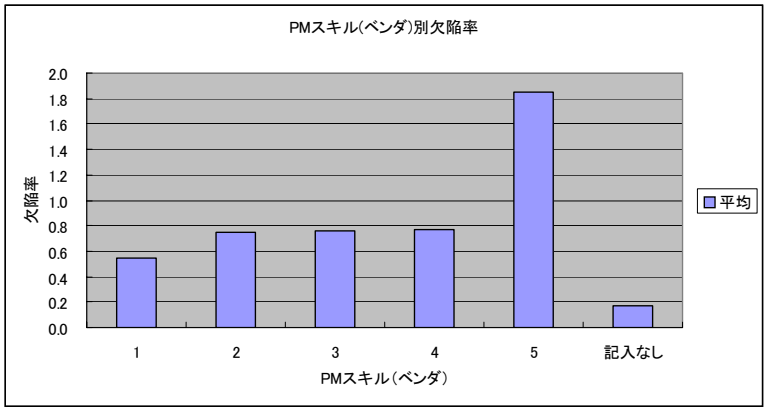
（５）ベンダーＰＭのスキルと欠陥率

プロジェクトマネジメントのスキルが高いベンダープロジェクトマネージャーが担当したプロジェクトでは、ほとんど欠陥率が 0.25 以下に収まっている。

ユーザーは発注先企業を選択するのみならず、経験豊かで優秀なプロジェクトマネージャーを選択せねばならないことになる。

全体を通して、ベンダープロジェクトマネージャーのスキルは、品質に大変影響を与えるが、ユーザープロジェクトマネージャーのスキルは、品質に与える影響が少ない。しかし、ユーザープロジェクトマネージャーのスキルが低いと、工期遅延が発生しやすい。

ａ．PM（ベンダー）スキル

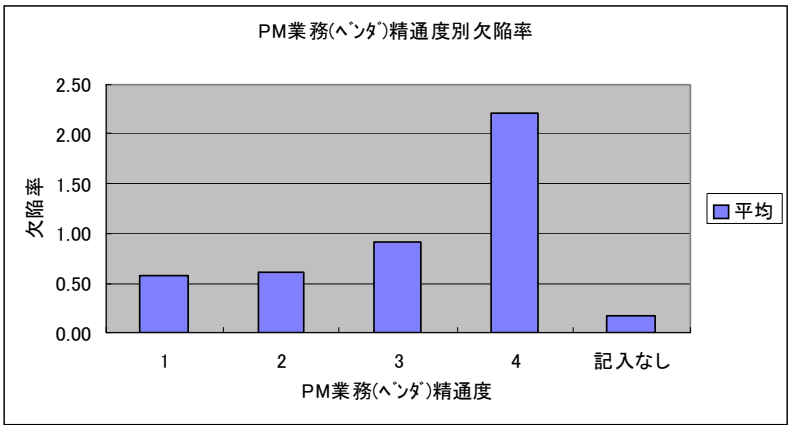


		PM（ベンダー）スキル						
		1	2	3	4	5	記入なし	計
欠陥率	件数	23	10	24	6	2	2	67
	平均	0.55	0.75	0.76	0.77	1.85	0.17	0.70
	最大	3.10	3.13	5.00	1.97	3.29	0.33	5.00
	最小	0.00	0.01	0.00	0.04	0.40	0.00	0.00

PM スキル(選択肢)	
1.多数の中・大規模プロジェクトの管理を経験	3.多数の小・中規模プロジェクトの管理を経験
2.少数の中・大規模プロジェクトの管理を経験	4.少数の小・中規模プロジェクトの管理を経験
	5.プロジェクト管理の経験なし

図表 3-7-14　PM（ベンダー）スキル

b．PM（ベンダー）業務精通度



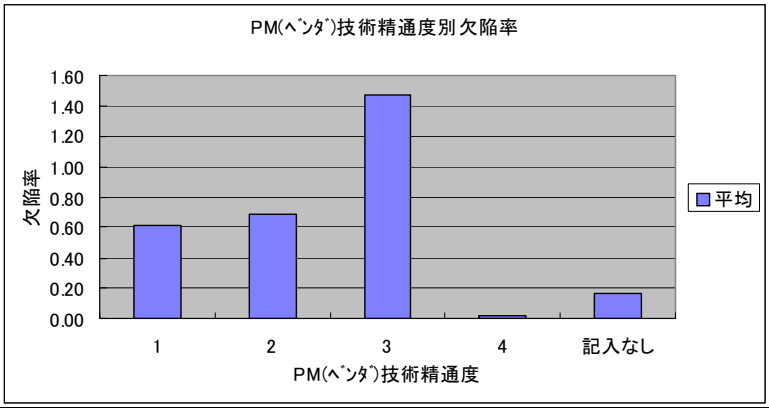
		PM（ベンダー）業務精通度					
		1	2	3	4	記入なし	計
欠陥率	件数	21	27	15	2	2	67
	平均	0.57	0.62	0.91	2.20	0.17	0.70
	最大	3.13	3.10	5.00	3.29	0.33	5.00
	最小	0.00	0.00	0.05	1.12	0.00	0.00

PM 業務精通度（選択肢）	
1.十分精通していた	3.精通していたとはいえない
2.ある程度のレベルまでは精通していた	4.全く経験も知識もなかった

図表 3-7-15　PM（ベンダー）業務精通度

ベンダープロジェクトマネージャーの業務精通度が低いと、欠陥率は上昇する。

ｃ．PM（ベンダー）技術精通度



		PM（ベンダー）技術精通度					
		1	2	3	4	記入なし	計
欠陥率	件数	29	29	6	1	2	67
	平均	0.61	0.69	1.47	0.02	0.17	0.70
	最大	3.13	5.00	3.29	0.02	0.33	5.00
	最小	0.00	0.00	0.04	0.02	0.00	0.00

PM 技術精通度（選択肢）	
1.十分精通していた	3.精通していたとはいえない
2.ある程度のレベルまでは精通していた	4.全く経験も知識もなかった

図表 3-7-16　PM（ベンダー）技術精通度

ベンダープロジェクトマネージャーの技術精通度は、欠陥率に影響が現れていない。スタッフがサポートしているものと思われる。

（６）ユーザーＰＭの業務精通度と要求仕様変更度

ユーザーPM 業務精通度と要求仕様変更度

		要求仕様変更発生				
PM 業務(U)	データ	1	2	3	4	総計
1	件数	4	33	9		46
	割合	8.70%	71.74%	19.57%	0.00%	100.00%
2	件数	2	39	15	1	57
	割合	3.51%	68.42%	26.32%	1.75%	100.00%
3	件数		4	6		10
	割合	0.00%	40.00%	60.00%	0.00%	100.00%
記入なし	件数		1	1		2
	割合	0.00%	50.00%	50.00%	0.00%	100.00%
合計	件数	6	77	31	1	115
	割合	5.22%	66.96%	26.96%	0.87%	100.00%

PM 業務精通（選択肢）	
1.十分精通していた	3.精通していたとはいえない
2. ある程度のレベルまでは精通していた	4.全く経験も知識もなかった

仕様変更発生（選択肢）	
1:変更なし	3.大きな変更が発生
2:軽微な変更が発生	4.重大な変更が発生

図表 3-7-17　ユーザーPM 業務精通度と要求仕様変更度

「ユーザープロジェクトマネージャが業務を良く知っていれば、大きな仕様変更を避けることが出来る。業務を熟知した経験者で決断力のある人をプロジェクトマネージャに選ぶ必要がある」

（７）欠陥率とユーザー満足度

欠陥率と顧客満足度との関係は、欠陥率が 0.25 未満（1 人月あたりのバグが 0.25 未満）のプロジェクトでは、**0.25 以上のプロジェクトよりも、プロジェクト全体の満足度は高い**が、品質の満足度ではその傾向が見られない。**欠陥率が 3 以上であっても、迅速に修正するアクションをすれば満足点をもらえる**ことがある。

欠陥率（欠陥数 PER_工数）と顧客満足度（プロジェクト全体）

欠陥率		顧客満足度(プロジェクト全体)				
		満足	やや不満	未回答	計	満足率
0	件数	1	2	3	6	33.3%
	平均	0.00	0.00	0.00	0	
0.25未満	件数	18	2	5	25	90.0%
	平均	0.09	0.14	0.09	0.09	
0.5未満	件数	5	4	4	13	55.6%
	平均	0.32	0.34	0.40	0.35	
1未満	件数	4	3	1	8	57.1%
	平均	0.68	0.68	0.65	0.67	
3未満	件数	6	3	1	10	66.7%
	平均	1.77	1.48	2.08	1.72	
3以上	件数	5			5	100.0%
	平均	3.51			3.51	
計	件数	39	14	14	67	73.6%
	平均	0.87	0.58	0.34	0.70	

図表 3-7-18　ユーザーPM 業務精通度と要求仕様変更度

7．生産性の評価

（１）予算と工数の関係

特異点を除き分析すると、90 万円/人月になるが、規模が大きいものは単価が安くなっている。

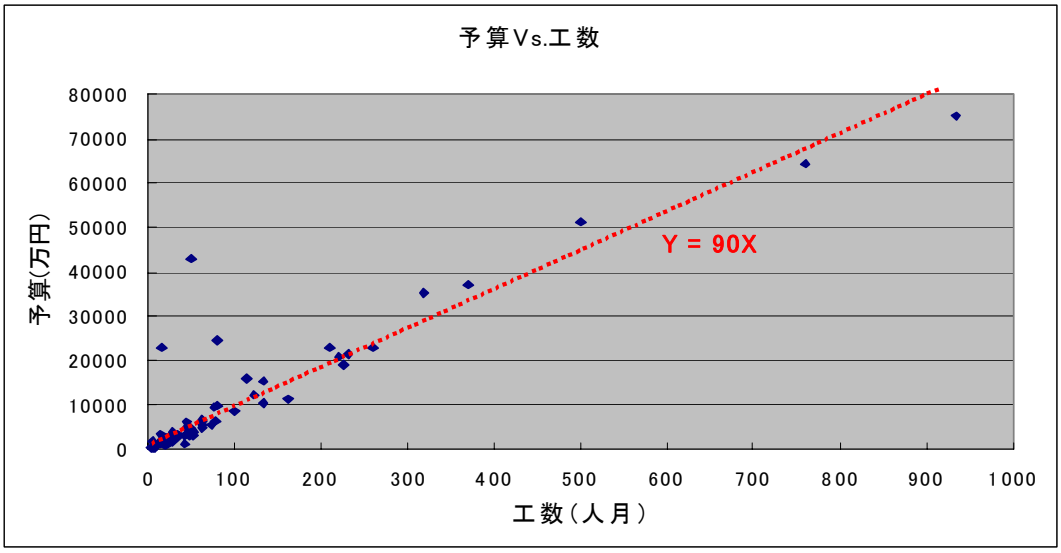


図 3-7-19　予算 Vs.工数分布（特異点を抜く）

回帰は原点を通るように行い、回帰式は　Y＝90X　となった。

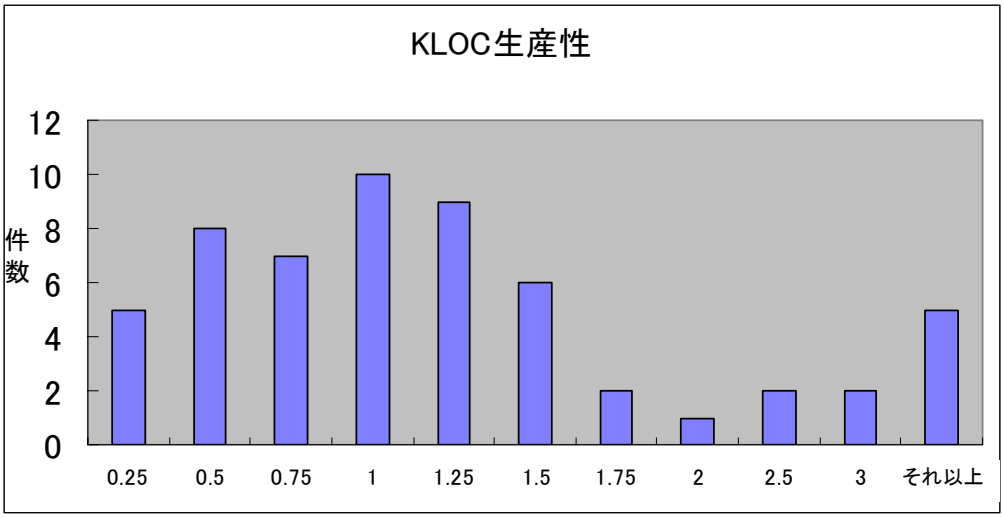
傾き＝人月単価＝約 90 万円ということになる。

	工数区分						総計
	記入なし	～10人月	～50人月	～100人月	～500人月	500人月～	
件数	7	8	32	13	12	3	75
平均予算	37892.3	661.4	2609.6	7820.7	20411.3	63480.0	11881.2
最大予算	226472.0	1800.0	6260.0	24460.0	37100.0	75000.0	226472.0
予算/人月(万円)	-	109.4	98.4	107.3	98.6	89.0	101.0

図表 3-7-20　予算と工数の集約

（２）ＬＯＣあたりの生産性

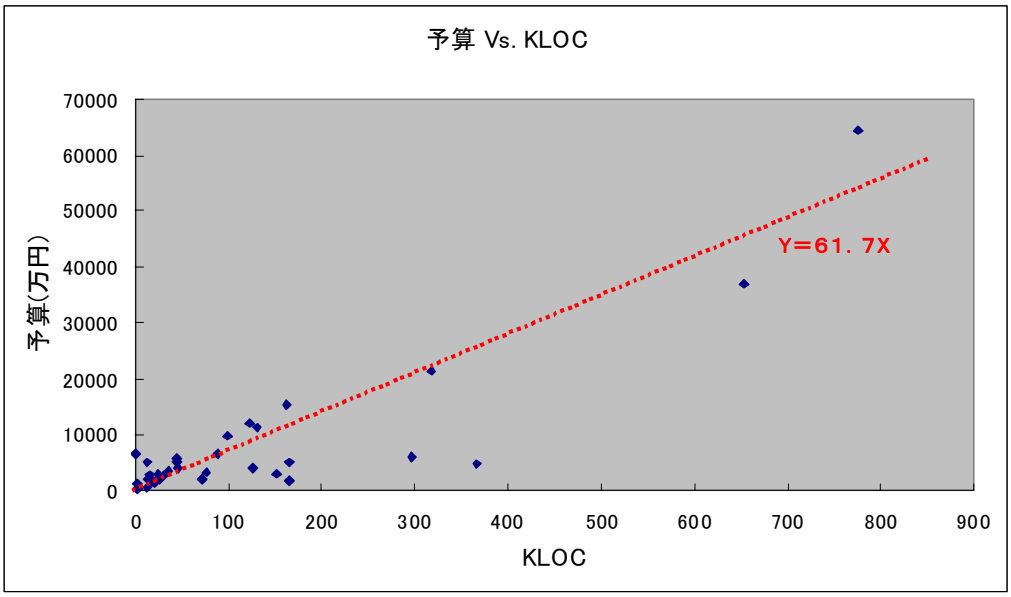
全体工数が記入されている 108 件のうち、LOC の記入があった 57 件について、KLOC あたりの生産性を規模別に計算した。全体の平均値は人月あたり 1.4KLOC であった。工程別に生産性が計算できるものに関しては、工程別にも計算を行った。全体の KLOC 生産性の分布は以下のとおり 500 人月以上のプロジェクトは明らかに生産性が低下している。



工数区分		プロジェクト全体	設計工程	実装工程	テスト工程
～10人月	件数	4	2	4	3
	KLOC/人月	1.0	2.3	3.3	11.4
～50人月	件数	21	14	15	15
	KLOC/人月	1.9	14.9	4.9	5.9
～100人月	件数	10	8	7	7
	KLOC/人月	1.4	12.7	3.0	4.1
～500人月	件数	16	12	13	13
	KLOC/人月	1.2	10.8	2.5	5.3
500人月～	件数	6	2	2	2
	KLOC/人月	0.6	3.2	4.6	2.1
計	件数	57	38	41	40
	KLOC/人月	1.4	11.9	3.6	5.6

図表 3-7-21　　KLOC 生産性の分布

システムが複雑、多岐にわたるので、設計工程とテスト工程に関して、生産性低下が著しい。



図表 3-7-22　予算とプログラム行数（LOC）の分布

回帰は原点を通るように行い、回帰式は　Y＝6.17X　となった。

	工数区分						総計
	未記入	～10人月	～50人月	～100人月	～500人月	500人月～	
件数	1	3	16	7	5	1	33
平均予算	5000.0	437.0	2689.8	5741.4	19466.8	64340.0	7612.5
予算/KLOC(平均値)	438.6	140.3	129.2	69.0	81.1	82.9	118.1
予算/KLOC(最大値)	438.6	192.3	860.7	128.3	100.0	82.9	860.7
予算/KLOC(最小値)	438.6	47.3	10.0	20.3	56.7	82.9	10.0

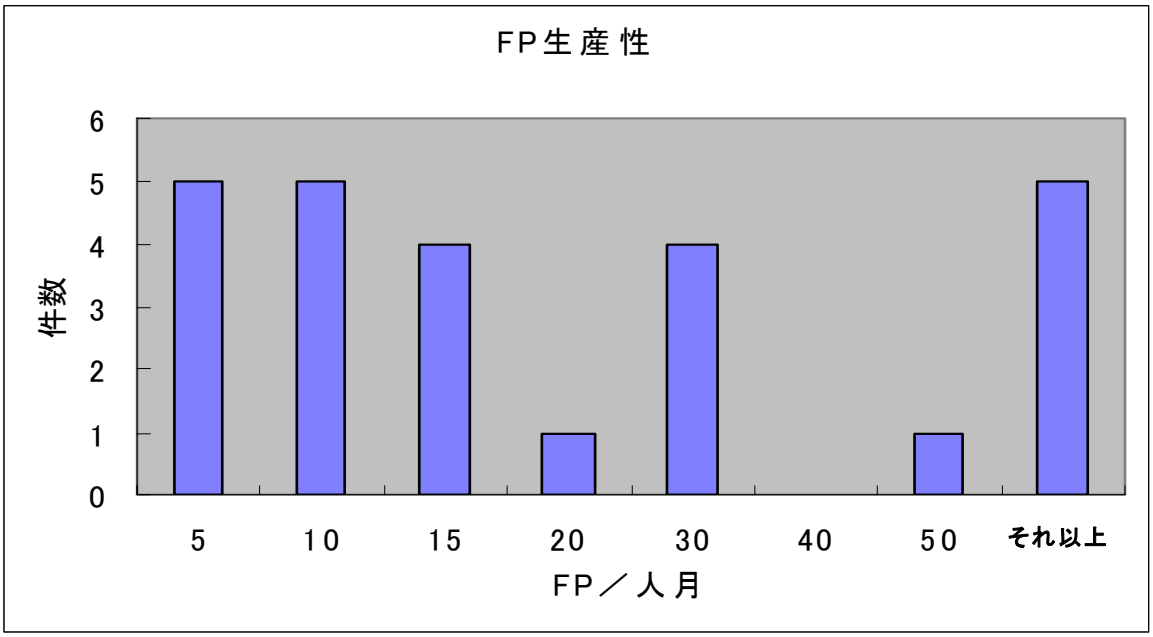
図表 3-7-23　予算とプログラム行数（LOC）の集約

規模の小さいプロジェクトは KStep あたりの予算額が多くなり、規模の大きいプロジェクト程、KStep あたりの予算額が少ない傾向が見える。上表の総計の予算/KLOC が、回帰式の傾きと異なるのは、総計の予算/KLOC は、人月単価の場合と同様単純平均値であるからである。予算が少ないプロジェクトに高い単価のものがあるため、単純平均は総平均よりも高くなる。

加重平均は、約 61 万円／KLOC である。

（３）ＦＰあたりの生産性

全体工数が記入されている 108 件のうち、FP の記入があった 25 件について、FP あたりの生産性を規模別に計算した。全体の平均値は**人月あたり 22.5FP** であった。工程別に生産性が計算できるものに関しては、工程別にも計算を行った。全体の FP 生産性の分布は次のとおりである。



図表 3-7-24　FP 生産性

このバラつきから見ると、FP の計算方法精度に問題があるかと思われる

工数区分		プロジェクト全体	設計工程	実装工程	テスト工程
～10人月	件数	6	3	4	4
	FP/人月	36.1	223.3	101.0	312.4
～50人月	件数	7	2	2	2
	FP/人月	18.4	155.1	71.2	122.0
～100人月	件数	3	3	2	2
	FP/人月	24.6	155.1	8.9	22.8
～500人月	件数	9	5	5	6
	FP/人月	15.8	66.0	33.3	146.4
500人月～	件数	0	0	0	0
	FP/人月	－	－	－	－
計	件数	25	13	13	14
	FP/人月	22.5	136.6	56.2	172.7

図 3-7-25　FP 生産性（工程別）

（４）予算オーバー率

予算オーバー率の基本統計量と分布

中央値、最頻値ともに 0（計画どおり）である。

工数区分		計画未満	計画通り	予算オーバー	合計
記入なし	件数	2	4	1	7
	割合	28.57%	57.14%	14.29%	100.00%
～10 人月	件数	1	3	4	8
	割合	12.50%	37.50%	50.00%	100.00%
～50 人月	件数	4	14	12	30
	割合	13.33%	46.67%	40.00%	100.00%
～100 人月	件数	6	2	3	11
	割合	54.55%	18.18%	27.27%	100.00%
～500 人月	件数	6		6	12
	割合	50.00%	0.00%	50.00%	100.00%
500 人月～	件数	1		1	2
	割合	50.00%	0.00%	50.00%	100.00%
計	件数	20	23	27	70
	割合	28.57%	32.86%	38.57%	100.00%

図表 3-7-26　規模別予算超過状況

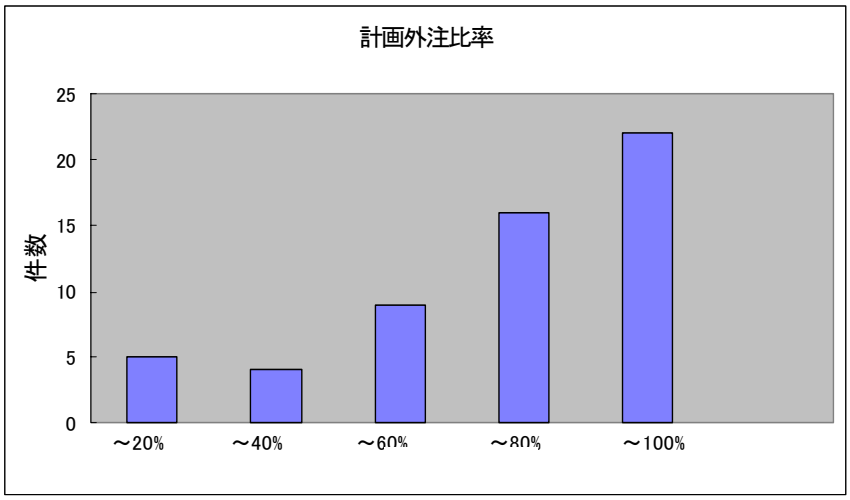
70 件中、予算超過は 27 件（39%）、予算どおりは 23 件（33%）、予算未満は 20 件（29%）であった。6 割以上のプロジェクトは計画通りの予算に収めている。予算超過のプロジェクトの割合は、10 人月以上 100 人月未満のプロジェクトで少ない。規模が大きいからといって、予算超過になりやすいとは限らないことになる。

（５）外注予算分析

ａ．計画外注予算比率

平均値は 68.35%であった。約 3 分の 2 以上の予算を、外注に出している事になる。

（ａ）分布



（ｂ）規模別計画外注比率

	工数区分						総計
	記入なし	～10人月	～50人月	～100人月	～500人月	500人月～	
件数	7	8	26	7	6	2	56
計画外注比率(平均)	73.0%	43.7%	71.7%	64.0%	78.4%	92.0%	68.3%
計画外注比率(最大)	100.0%	100.0%	100.0%	92.5%	100.0%	92.9%	100.0%
計画外注比率(最小)	40.0%	0.0%	12.4%	13.5%	47.6%	91.1%	0.0%

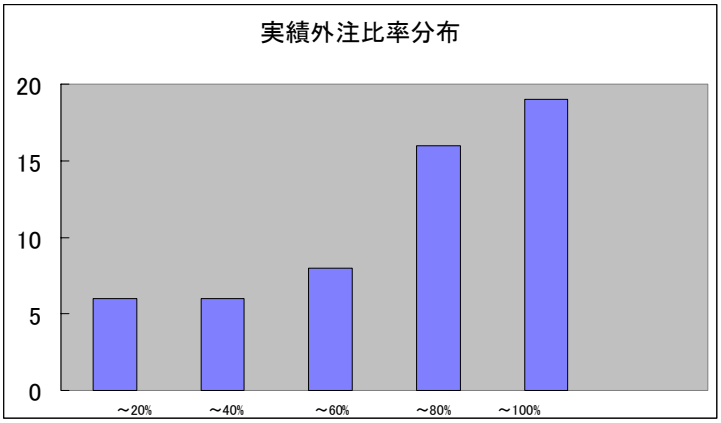
図 3-7-27　計画外注比率

規模が大きいほうが、計画外注比率が高い傾向にあるようだ。大規模システムでは、一括してベンダーに委託するケースが多いからと想定される。

ｂ．実績外注予算比率

平均値は 64.4%であり、計画値よりやや少ないが、比率のレベルでは、計画と実績にそれほど差があるとはいえない。

（ａ）分布



図表 3-7-28　実績外注比率分布

規模別実績外注比率

	工数区分						総計
	記入なし	～10人月	～50人月	～100人月	～500人月	500人月～	
件数	7	8	22	8	8	2	55
実績外注比率(平均)	76.7%	42.0%	68.4%	55.9%	66.3%	93.3%	64.4%
実績外注比率(最大)	100.0%	100.0%	100.0%	91.0%	99.5%	95.0%	100.0%
実績外注比率(最小)	40.0%	0.0%	10.4%	12.5%	33.3%	91.7%	0.0%

図表 3-7-29　規模別実績外外注比率

ｃ．外注予算の計画と実績の対比

計画・実績対比

外注比率が、計画値から実績が増えているか減っているかに関して集計をした。外注比率が計画よりも増えたか否かと、総予算が超過したか否かに関してクロス集計を行った。外注比率については、計画値よりも±5%以上変動した場合、上昇した／下降した　とみなした。予算については、計画値よりも±10%以上変動した場合、増えた／減ったとみなした。

			外注比率			
			計画値より下降	計画値どおり	計画値より上昇	計
総予算	計画未満	件数	1	1	3	5
		割合	20.0%	20.0%	60.0%	100.0%
	計画通り	件数	2	27	4	33
		割合	6.1%	81.8%	12.1%	100.0%
	予算超過	件数	3	8		11
		割合	27.3%	72.7%	0.0%	100.0%
	計	件数	6	36	7	49
		割合	12.2%	73.5%	14.3%	100.0%

図表 3-7-30　規模別対計画外注予算

予算超過する場合は、直営がカバーにまわっているケースがある。

外注予算が、計画値から実績が増えているか減っているかに関して規模別に集計をした。

規模		外注費:計画値－実績値			
		予定以内	予定通り	超過	総計
記入なし	件数	2	5	2	9
	割合	22.2%	55.6%	22.2%	100.0%
	平均超過額	−138.0		7523.5	1641.2
	計画値からの割合	−5.3%		8.5%	7.0%
10人月未満	件数	2	8		10
	割合	20.0%	80.0%		100.0%
	平均超過額	−5.0			−1.0
	計画値からの割合	−1.4%			−0.2%
50人月未満	件数	7	13	8	28
	割合	25.0%	46.4%	28.6%	100.0%
	平均超過額	−825.0		644.9	−22.0
	計画値からの割合	−29.9%		16.6%	−0.3%
100人月未満	件数	3	1	4	8
	割合	37.5%	12.5%	50.0%	100.0%
	平均超過額	−374.7		697.5	208.3
	計画値からの割合	−9.5%		16.2%	5.6%
500人月未満	件数	4	2	3	9
	割合	44.4%	22.2%	33.3%	100.0%
	平均超過額	−1999.3		833.3	−610.8
	計画値からの割合	−10.6%		6.2%	−4.2%
500人月以上	件数			2	2
	割合			100.0%	100.0%
	平均超過額			5350.0	5350.0
	計画値からの割合			9.0%	9.0%
合計	件数	18	29	19	66
	割合	27.3%	43.9%	28.8%	100.0%
	平均超過額	−843.4		1905.1	318.4
	計画値からの割合	−13.5%		9.4%	3.0%

図表 3-7-31　規模別外注費（計画対実績）

外注費が超過したプロジェクトは、規模が大きいほど多い。全体の 44%は、実績支払金額と計画支払金額が等しい。約 30%に関しては、計画額を上回る金額を支払ったことになる。（平均 10%の超過）**外注費は 10%程度余裕を見ておくことが必要。**

（６）ファイル数・画面数・帳票数・バッチ数と開発総数の関連

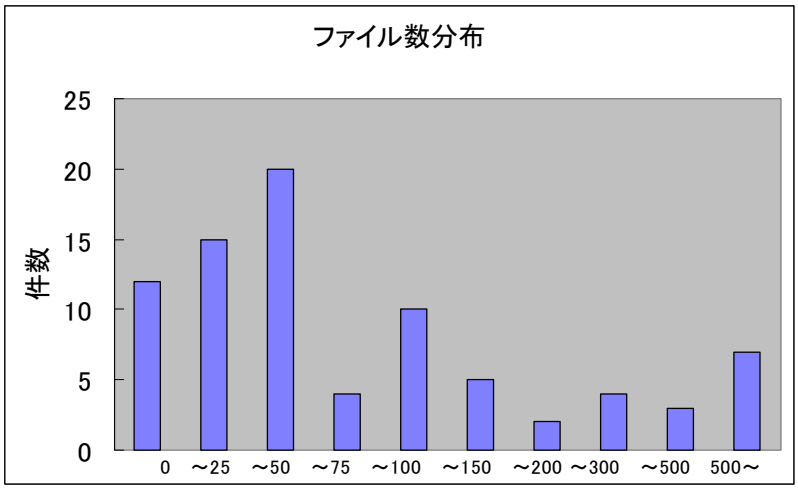
ファイル数・画面数・帳票数・バッチ数と開発総工数との関連

回答のあった 82 プロジェクトについて、ファイル数・画面数・帳票数・バッチプログラム数（バッチ数）と、開発総工数との関連分析を行った。FP とファイル数・画面数・帳票数・バッチ数との関連については、FP 計測方法の精度の問題で、分析は行わなかった。

ａ．基本統計量と基本分布

ファイル数・画面数・帳票数・バッチ数の基本統計量と基本分布は以下の通りとなった。

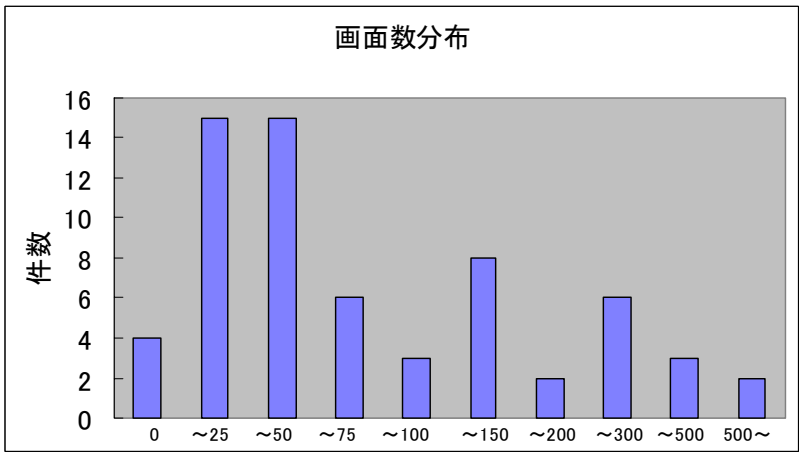
（ａ）ファイル数



ファイル数	
平均	270.6707317
標準誤差	158.1070089
中央値(ﾒｼﾞｱﾝ)	42.5
最頻値(ﾓｰﾄﾞ)	0
標準偏差	1431.719859
分散	2049821.754
尖度	79.18352594
歪度	8.82955038
範囲	12970
最小	0
最大	12970
合計	22195
標本数	82

図表 3-7-32　ファイル数分布

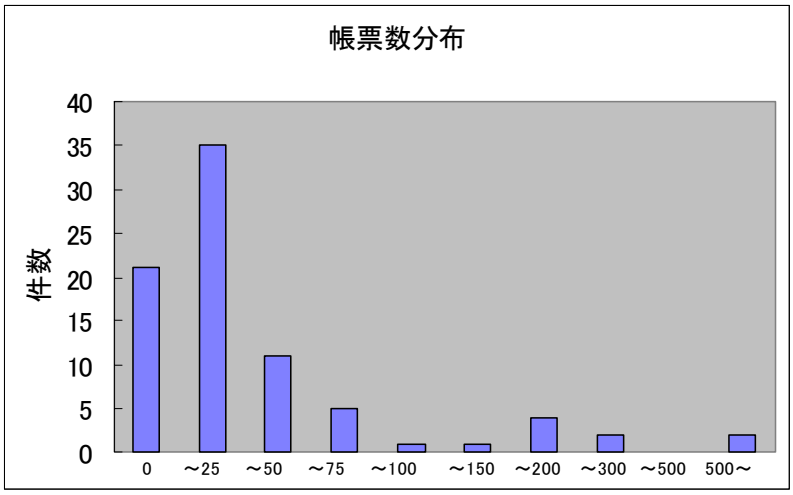
（ｂ）画面数



画面数	
平均	114.3292683
標準誤差	17.69315101
中央値(ﾒｼﾞｱﾝ)	49.5
最頻値(ﾓｰﾄﾞ)	0
標準偏差	160.2182967
分散	25669.90259
尖度	7.961731964
歪度	2.628438283
範囲	897
最小	0
最大	897
合計	9375
標本数	82

図表 3-7-33　画面数分布

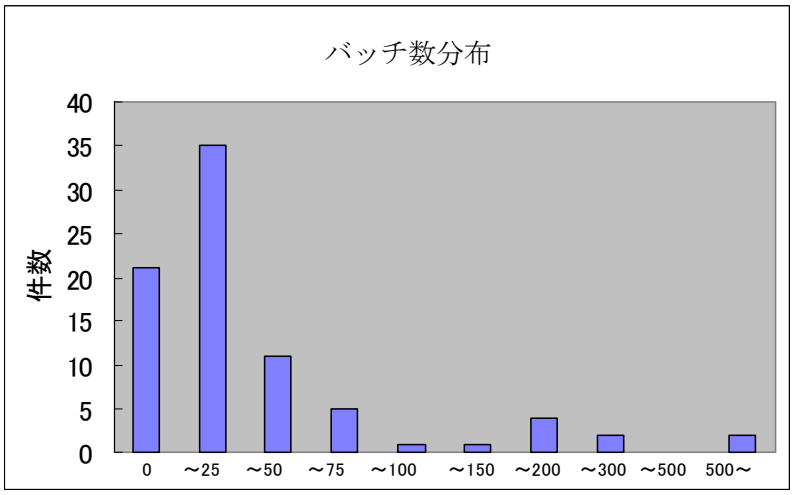
（ｃ）帳票数



帳票数	
平均	51.5
標準誤差	15.64548386
中央値(ﾒｼﾞｱﾝ)	14
最頻値(ﾓｰﾄﾞ)	0
標準偏差	141.6758821
分散	20072.05556
尖度	32.03873214
歪度	5.394098799
範囲	1011
最小	0
最大	1011
合計	4223
標本数	82

図表 3-7-34　帳票数分布

（ｄ）バッチ数



バッチ数	
平均	241.804878
標準誤差	65.59467263
中央値(ﾒｼﾞｱﾝ)	42.5
最頻値(ﾓｰﾄﾞ)	0
標準偏差	593.9850237
分散	352818.2084
尖度	22.16787297
歪度	4.393246658
範囲	4000
最小	0
最大	4000
合計	19828
標本数	82

図 3-7-35　バッチ数分布

バッチ数は、平均値は画面数に比して大きいが、逆に中央値は小さい。これは、極端に多いプロジェクトが一部にあるためである。

ｂ．相関行列と回帰式

ファイル数・画面数・帳票数・バッチ数で総工数が説明できるか分析を試みた。

（ａ）相関行列

ファイル数・画面数・帳票数・バッチ数と総工数間の相関行列は以下の通りであった。

	ファイル数	画面数	帳票数	バッチ数	工数
ファイル数	1				
画面数	0.023796569	1			
帳票数	0.182871979	0.527583786	1		
バッチ数	0.407410814	0.0811118	0.351975298	1	
工数	0.205820947	0.551027206	0.38621562	0.251510438	1

図表 3-7-36　４要素の相関

画面数と帳票数の間には何らかの相関関係がありそうである。工数と最も相関が高い変数は画面数である。

（ｂ）４変数回帰分析

総工数を目的変数に、ファイル数・画面数・帳票数・バッチ数の 4 変数を説明変数にして、回帰分析を行った結果を以下に示す。

回帰統計	
重相関 R	0.601486255
重決定 R2	0.361785715
補正 R2	0.328631726
標準誤差	199.9260157
観測数	82

分散分析表　　F(4,60)の1%点＝3.65

	自由度	変動	分散	観測された分散比	有意 F
回帰	4	1744673.806	436168.4516	10.91228318	4.62631E-07
残差	77	3077721.704	39970.41174		
合計	81	4822395.511			

＊＊＊ 回帰分析は1%有意であった。

	係数	標準誤差	t	P-値	下限 95%	上限 95%	下限 95.0%	上限 95.0%
切片	32.93413523	28.49264429	1.15588202	0.251303376	-23.8020363	89.67030678	-23.80203632	89.67030678
ファイル数	0.021763444	0.017022205	1.278532611	0.204901975	-0.01213214	0.055659025	-0.012132137	0.055659025
画面数	0.784070959	0.164782129	4.758228122	8.95522E-06	0.455947426	1.112194493	0.455947426	1.112194493
帳票数	0.070222646	0.198634143	0.353527572	0.724659077	-0.32530894	0.465754232	-0.32530894	0.465754232
バッチ数	0.058894785	0.043308261	1.359897235	0.177831025	-0.02734308	0.145132655	-0.027343085	0.145132655

図表 3-7-37　４変数回帰分析

この結果から、工数(人月)　＝　0.02 x ファイル数　＋　0.78 x 画面数　＋　0.07　x　帳票数　＋　0.06 x バッチ数 ＋32、(重相関係数=0.601/寄与率=0.36)となったが、画面数以外の偏回帰係数は、有意ではない。

（ｃ）２変数回帰分析

画面数と帳票数、ファイル数とバッチ数の変数間にはある程度の相関を見られそうである事、工数と最も相関があるのは、画面数である事の理由から、今度は説明変数を画面数とバッチ数の 2 変数に絞り込んで回帰分析を行った。回帰式は Y 切片=0 となるように行った。

回帰統計	
重相関 R	0.5793414
重決定 R2	0.335636458
補正 R2	0.314831914
標準誤差	200.1194569
観測数	82

F(2,60)の1%点＝4.98

分散分析表

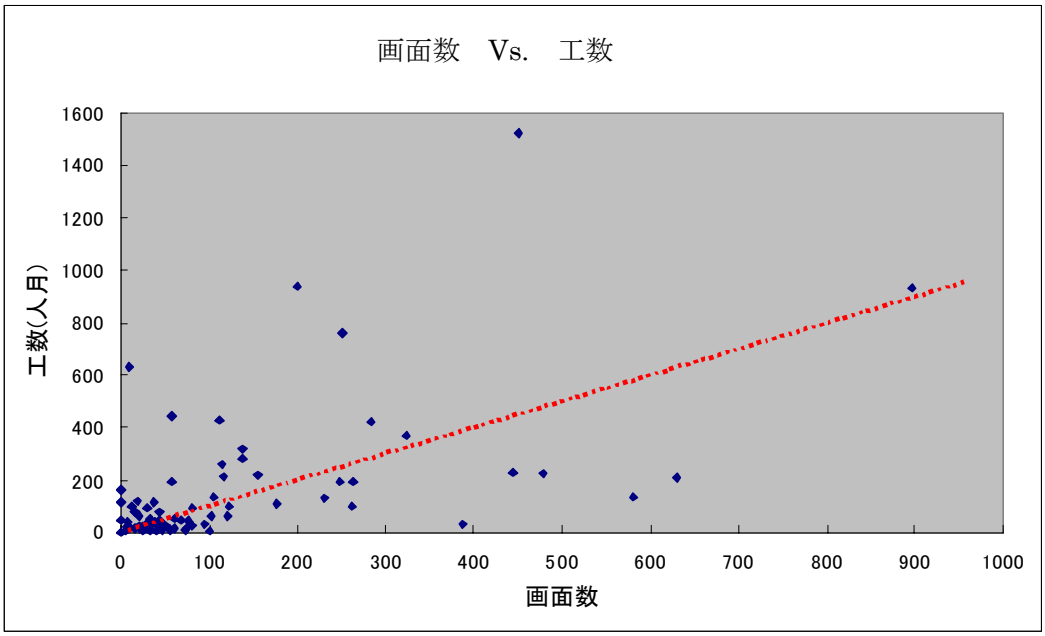
	自由度	変動	分散	測された分散	有意 F
回帰	2	1618571.748	809285.8741	20.20799979	8.17148E-08
残差	80	3203823.762	40047.79703		
合計	82	4822395.511			

	係数	標準誤差	t	P-値	下限 95%	上限 95%	下限 95.0%	上限 95.0%
切片	0	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
画面数	0.899627721	0.117504967	7.656082503	3.80273E-11	0.665785157	1.133470284	0.665785157	1.133470284
バッチ数	0.097446164	0.03610697	2.698818666	0.008486688	0.025590934	0.169301394	0.025590934	0.169301394

図表 3-7-38　２変数回帰分析

結果は、工数(人月)　＝　0.9ｘ 画面数　＋　0.1　ｘ バッチ数　となった。重相関係数は 0.57（寄与率＝0.34）と低下したが、画面数、バッチ数のともに相関係数は 1%有意になった。

（ｄ）画面数と工数の分布



図表 3-7-39　画面数 Vs 工数

工数と最も高い相関を示した画面数との関連を以下に示す。

今回のアンケートでは、ひとつの画面に登録・更新・削除・照会の機能が混在した時の画面カウント方法など、詳細に関して定めていないため、バラつきが大きくなったと考えられる。画面、帳票、ファイル数の相関をどのように解釈するかも含め、今後の課題である。

ｃ．規模別集計

82 プロジェクトに関して、ファイル数・画面数・帳票数・バッチ数を、プロジェクトの規模別に集計した。

プロジェクト規模	件数		ファイル数	画面数	帳票数	バッチ数
10人月未満	8	平均	17.3	41.1	3.0	2.5
		最大	50.0	101.0	14.0	8.0
50人月未満	34	平均	53.1	50.0	13.0	214.4
		最大	605.0	387.0	50.0	4000.0
100人月未満	11	平均	80.8	74.8	52.2	130.2
		最大	205.0	262.0	244.0	475.0
500人月未満	24	平均	701.8	196.5	84.0	354.2
		最大	12970.0	630.0	732.0	2320.0
500人月以上	5	平均	503.8	361.4	233.8	516.8
		最大	837.0	897.0	1011.0	893.0
合計	82	平均	270.7	114.3	51.5	241.8
		最大	12970.0	897.0	1011.0	4000.0

図表 3-7-40　プロジェクト規模と４変数

８．総合評価

工期、品質、生産性の間には、相互に影響する要因が絡んでいる。
下の表を参照されたい。

クラス	工期 1-(実工期/標準工期)＝工期短縮率	品質 納入以降に発見された障害数/基準量(FP、LOC、人月、金額)	生産性 FP／人月 KLOC／人月 金額／人月
1	20%以上の短縮	3倍以上の向上	20%以上の向上
2	20%以下の短縮	3倍以下の向上	20%以下の向上
3	基準値	基準値	基準値
4	20%以下の延長	3倍以下の低下	20%以下の低下
5	20%以上の延長	3倍以上の低下	20%以上の低下

図表 3-7-41　工期、品質、生産性関連表

※工期、品質、生産性について、該当プロジェクトの計画時の目標と実績の評価を上記評価表に記入する。これを基に、原因分析と対策追求を行い、次回以降のプロジェクト実施時についてのノウハウを残す。

「工期が極端に短いので、十分なレビューやテストが出来ずに、品質が低下した」、「品質を確保するために、生産性を落として、十分なテストを実施した」などの関係が存在する。この三つの要因を確認するのが、上記表の目的である。

評価尺度は企業によって異なるので、それぞれが変えてよい。つまり上記表で、「工期が 20%以上短いのはランク 2 などと仮に規定してあるが、20%ではなく 50%を境とする基準に変えても良い。

「残業規制が非常に厳しい会社は、どうしても工期は延びがちになる」し、「工期確保のためには、残業時間数などは問わない」ならば、短工期も可能になる。

この三要因の関係を企業ごとに、標準として整理できれば、アクションが明確になり、システム開発に伴うトラブルは減少する。

この三要因のほかに「要求仕様書の明確度」や「利用者代表の参画度合い」なども合わせて評価の対象として評価データを保存し、企業のノウハウとする方法もある。

９．アンケート調査表

下記に、本調査のアンケート調査表項目を表記する。

調査表項目

Q1 利用局面

Q1.1　業務種別

開発アプリケーションの対象とする業務の種類をください。(複数選択可)

1.経営・企画　2.会計・経理 3.営業・販売 4.生産・物流 5.人事・厚生 6.管理一般
7.総務・一般事務 8.研究・開発　　9.技術・制御 10.マスター管理 11.受注・発注・在庫
12.物流管理　　13.外部業者管理　　14.約定・受渡 15.顧客管理
16.商品計画（管理する対象商品別）17.商品管理（管理する対象商品別）
18.施設・設備（店舗）19.情報分析 20.その他（　　）

Q1.2　要件決定者の人数

要求仕様定義における実質的なキーマン（要件決定者）の人数を記入ください。
（　　）人

Q1.3　対象端末数

開発システムに接続する端末数を記入ください。

1.特定ユーザーの特定端末からの使用を想定しているため利用できる端末には制限がある・・・・
（　　）　台

2.WEB による EC サイト等不特定多数ユーザー向けであり利用できる端末に制限はない

Q1.4　同時利用者数

開発システムの平均と最大の同時利用者数を記入ください。
平均　　（　　）　台
最大　　（　　）　台

Q2　システム特性･開発方法論

Q2.1 開発種別

プロジェクトの開発種別[注]を選択ください。

1.新規開発　　2.再開発・改修

注：今までその企業に存在しない、新しいシステムを開発する場合を新規開発
既存システムが存在し、そのドキュメント、プログラムの一部を、修正、追加し開
発する場合を再開発、改修と呼びます。

Q2.2 新規作成作業の負荷割合

プロジェクトの成果物を作成する上で、ゼロから新規作成した作業の負荷割合、既存のものを利用（コピー&モディファイ等）して作成した作業の負荷割合、および、既存のものをそのまま変更せずに使用した部分の作業負荷[3]割合を記入ください。作業負荷の割合は、合計が100%になるように、ドキュメントとプログラムに分けて記入ください。

	ドキュメント		プログラム	
	作業負荷の割合	工数は Q3.7 に[4]	作業負荷の割合	工数は Q3.7 に
新規作成	%	1.含む　2.含まない	テストありの割合　%	1.含む　2.含まない
既存のものを利用して作成	%	1.含む　2.含まない	テストありの割合　%	1.含む　2.含まない
			テストなしの割合　%	1.含む　2.含まない
既存のものをそのまま使用	%	1.含む　2.含まない	テストありの割合　%	1.含む　2.含まない
			テストなしの割合　%	1.含む　2.含まない
合計	100　%	ー	合　計　100　%	ー

Q2.3 業務パッケージを利用しての開発

業務パッケージ[5]を利用しての開発であったか否かを選択ください。

1.　Yes　　2.　No

Q2.4 カスタマイズの度合い

Q2.3 が Yes の場合の質問です。No の場合は次の設問にお進みください。

利用したパッケージの名称と、パッケージ金額に対するカスタマイズ費用（アドオン費用を含む）の割合を記入ください。

割合は、（カスタマイズ金額・アドオン金額）÷（パッケージ金額）ｘ１００（%）で回答ください。

パッケージ名称　（　　）

カスタマイズの度合い　（　　）%

Q2.5 開発プラットフォーム

開発したシステムのＯＳを選択ください。クライアントとサーバが異なる場合はサーバのＯＳを選択ください。（複数選択可）

1.メインフレーム　2.オフコン　3.UNIX　4.Windows　5.LINUX　6.その他（　　）

Q2.6 システムアーキテクチャ

開発したシステムのアーキテクチャを選択ください。（複数選択可）

1.　汎用機アーキテクチャ　　2.　Ｃ／Ｓアーキテクチャ　　3.　WEB システム
4.スタンドアロンシステム　　5.その他（　　）

Q2.7 主開発言語

主たる開発言語（および開発ツール）について、規模の大きい順番に最大３つまで選択し、番号を（　）に記入ください。

[3]作成はしないが、調査・テスト等に要する作業負荷

[4]当該作業の工数は、Q3.7 で回答頂く開発工数に含まれている場合は 1.を、含まれていない場合は 2.を、選択ください

[5] ERP パッケージなどカスタマイズ等して使うものを指し、ツール的に使用する。通信パッケージ(HULFT 等)等は該当しません。

1）（　　）　2）（　　）
3）（　　）

1.アセンブラ	2.COBOL	3.PL/I	4.Pro*C	5.C++
6.Visual C++	7.C 言語	8.VB	9.Excel(VBA)	10.PowerBuilder
11.Developer2000	12.InputMan	13.PL/SQL	14.ABAP	15.C#
16.Visual Basic.NET	17.Java	18.Perl	19.Shell スクリプト	20.Delphi
21.HTML	22.XML	23.その他言語（　　）		

Q2.8 DBMS

開発において使用した DBMS を選択ください。使用していない場合には「なし」を選択ください。
（複数選択可）

1.Oracle	2.SQL Server	3.PostgreSQL	4.MySQL	5.Sybase
6.Informix	7.ISAM	8.DB2	9.Access	10.HiRDB
11.IMS	12.その他 DB（　　）			

Q2.9 上流ケースツールの利用

開発において上流ケースツールを使用したしたか否かを選択し、利用した場合はその名称を記入ください。

1.利用した 名称（　　）
2.利用していない

Q2.10 統合ケースツールの利用

開発において統合ケースツールを使用したしたか否かを選択し、利用した場合はその名称を記入ください。

1.利用した　　名称（　　）
2.利用していない

Q2.11 コードジェネレータの利用

開発においてコードジェネレータを使用したしたか否かを選択し、利用した場合はその名称を記入ください。

1.利用した　　名称（　　）
2.利用していない

Q2.12 開発ライフサイクルモデル

開発において使用したライフサイクルモデルについて選択ください。反復型の場合には、繰り返し数の実績値を記入ください。

1.ウォーターフォール型　2.反復型（　　）回　3.その他（　　）

Q2.13 開発方法論

開発において使用した開発方法論を選択ください。（複数選択可）使用していない場合は「なし」を選択ください。

1.構造化分析設計　2.オブジェクト指向分析設計　3.データ中心アプローチ
4.その他（　　）　5.なし

Q2.14　設計書再利用率

設計書[6]の再利用率（再利用したページ数／全体のページ数）を%で記入ください。再利用していない場合には「0」と記入ください。

設計書	再利用率(%)
システム化計画書	
要件定義書	
基本設計書	
詳細設計書	

Q2.15 ソースコード再利用率

ソースコードの利用率（再利用した SLOC／全体の SLOC）を%で記入ください。再利用していない場合には「0」と記入ください。

（　　）%

Q2.16 コンポーネント再利用率

ソフトウェアコンポーネント（汎用サブルーチン、プログラムライブラリ等）の利用率（再利用した機能規模／システム全体の機能規模）を%で記入ください。
再利用していない場合には「0」と記入ください。

（　　）%

Q2.17　テストケース再利用率

テストケース[7]の再利用率（再利用したテストケース数／全体テストケース数）を%で記入ください。再利用していない場合には「0」と記入ください。

テストフェーズ	再利用率(%)
結合テスト	
総合テスト(ベンダー確認)	
総合テスト(顧客確認)	

Q3 規模・工期・工数・コスト

Q3.1 FP 値

計画、実績の FP 値を記入ください。計画値は、外部設計終了時の値を記入ください。

項目	計画/実績	FP値
FP値（調整係数適用前）	計画	
	実績	
FP値（調整係数適用後）	計画	
	実績	

Q3.2 FP の計測手法

FP の計測手法を選択ください。

1. IFPUG　2. SPR　3. MKII　4. NESMA 試算　5. NESMA 概算
6. COSMIC-FFP　7. 自社基準　8. その他（　　）

[6]各設計工程の内容については別表1を参照ください。
[7] テスト工程の内容は別表1を参照ください。

Q3.3 FP の詳細値(IFPUG の場合)

Q3.2 の回答が「IFPUG」の場合の質問です。それ以外の場合は次の設問にお進みください。

Q3.1 で記入いただいた FP の 5 つの計測要素(EI、EO、EQ、ILF、EIF)の 複雑度別の個数（機能数）と FP 値を記入してください。計画値と実績値の両方について記入ください。計画値は、外部設計終了時の値を記入ください。

			機能数			FP	
			大	中	小		
トランザクションファンクション	EI	計画					※ FP＝大×6＋中×4＋小×3
		実績					
	EO	計画					※ FP＝大×7＋中×5＋小×4
		実績					
	EQ	計画					※ FP＝大×6＋中×4＋小×3
		実績					
データファンクション	ILF	計画					※ FP＝大×15＋中×10＋小×7
		実績					
	EIF	計画					※ FP＝大×10＋中×7＋小×5
		実績					

注 EI ： 外部入力 （External Input） EO ： 外部出力 （External Output）EQ ： 外部照会 （External Inquiry）
ILF ： 内部論理ファイル （Internal Logical File） EIF ： 外部インターフェイスファイル （External Interface File）

Q3.4 FP の詳細値(IFPUG 以外の場合)

Q3.2 の回答が「NESMA 試算」、「NESMA 概算」、もしくは IFPUG 法に準じた「自社基準」の場合の質問です。該当しない場合は次の設問にお進みください。

Q3.1 で記入いただいた FP のトランザクションファンクション数、データファンクション数の合計個数（機能数）と FP 値を下表に記入ください。計画値と実績値の両方について記入ください。

		機能数	FP 数
トランザクションファンクション数[8]	計画		
	実績		
データファンクション数[9]	計画		
	実績		

Q3.5 SLOC 値・プログラム本数

[8] トランザクションファンクション数　　IFPUG の EI、EO、EQ に相当

システムの SLOC 値（Source Line Of Code）、プログラム本数について記入ください。計測が困難な場合には、「不明」と記入ください。

記入値にコメント行および空行を含むかどうかに関しても記入ください。

Q2. 7 主開発言語で回答した開発基本言語別に、見積時と実績の両方について記入ください。

項目		計画値		実績値		SLOC 値にコメント行を	SLOC値に空行を	プログラム修正時のLOC 換算係数[10]
		SLOC値	プログラム本数	SLOC値	プログラム本数			
合計値						1.含む　2.含まない	1.含む　2.含まない	（　　）倍
言語別（上位5つ）	1.（　　）					1.含む　2.含まない	1.含む　2.含まない	（　　）倍
	2.（　　）					1.含む　2.含まない	1.含む　2.含まない	（　　）倍
	3.（　　）					1.含む　2.含まない	1.含む　2.含まない	（　　）倍
	4.（　　）					1.含む　2.含まない	1.含む　2.含まない	（　　）倍
	5.（　　）					1.含む　2.含まない	1.含む　2.含まない	（　　）倍

[9] データファンクション数 IFPUG の ILF、EIF に相当

[10] 例：1000ステップのプログラム中100ステップを修正した場合、新規開発に対して修正に係る負荷（換算数）を3倍とみなし、開発 SLOC = 100 x 3 = 300 としてカウントしている。プログラム修正時に、ステップ数を換算してカウントいる場合にはその換算係数（本例の場合は 3.0）を記入ください。

Q3.6 DB、画面、帳票、バッチ数

システムのファイル数、画面数、帳票数、バッチ数[11]を難易度別に下表（表 1）に記入ください。

難易度ポイントは、低難易度を１、中難易度を３、高難易度を５としますが、特別難度のものがある場合、当該ファイル、画面、帳票、バッチの難易度ポイントも同時に記入ください。

計画と実績に分けて記入ください。

表 1. ＤＢ、画面、帳票、バッチ数

		ファイル				画面				帳票				バッチ			
		特別難度（特殊）	難易度3（項目多）	難易度2（項目中）	難易度1（項目少）	特別難度（特殊）	難易度3（難しい）	難易度2（普通）	難易度1（易しい）	特別難度（特殊）	難易度3（難しい）	難易度2（普通）	難易度1（易しい）	特別難度（特殊）	難易度3（難しい）	難易度2（普通）	難易度1（易しい）
計画	難易度ポイント		5	3	1		5	3	1		5	3	1		5	3	1
	ファイル、画面、帳票、バッチ数																
実績	難易度ポイント		5	3	1		5	3	1		5	3	1		5	3	1
	ファイル、画面、帳票、バッチ数																

[11]DB再編成等オンラインリアルタイム処理以外のものは、バッチとみなしてください。バッチ的にキックするストアドプロシージャやCGI(Common Gateway Interface)もこれに含めてください。
バッチ数は、汎用機でいうところのJCLのジョブステップ数の単位、オープンシステムでのEXE（実行プログラム）単位でカウントしてください。

Q3.7 体制・工期・工数・コスト

プロジェクトの体制・工期・工数・コストの概要について下表（表 2）に記入ください。
Q2. 12 で「反復型」と回答した場合、工期、工数、コストのフェーズ[注1]別詳細には、記入しなくて結構です。

表 2. 体制・工期・工数・コスト

分類	項目	計画/実績		プロジェクト全体		フェーズ別詳細								
				プロジェクト合計	フェーズ共通	システム化計画	要件定義	基本設計（外部設計）	詳細設計（内部設計）	製作	結合テスト	総合テスト1（ベンダー確認）	総合テスト2（顧客確認）	フォロー（運用）
契約形態開発体制	開発体制（社内/外注）[注2]	実績		1. 2. 3.		1. 2. 3.	1. 2. 3.	1. 2. 3.	1. 2. 3.	1. 2. 3.	1. 2. 3.	1. 2. 3.	1. 2. 3.	1. 2. 3.
	要件決定者ソフトウェア経験[注3]	実績		1. 2. 3. 4.										
	要件決定者関与度[注4]	実績		1. 2. 3. 4.		1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.
	要求仕様の明確さ[注5]	実績		1. 2. 3. 4.										
	要求仕様変更発生[注6]	実績		1. 2. 3. 4.		1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.	1. 2. 3. 4.
工期[注7]	時期/工期	計画	時期	年　月 ～ 年　月		年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月
			工期	月		月	月	月	月	月	月	月	月	月
		実績	時期	年　月 ～ 年　月		年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月	年　月 ～ 年　月
			工期	月		月	月	月	月	月	月	月	月	月
工数[注7]	開発工数[注8]	計画		人月	人月	人月	人月	人月	人月	人月	人月	人月	人月	人月
		実績		人月	人月	人月	人月	人月	人月	人月	人月	人月	人月	人月
	管理工数[注8]	計画		人月	人月	人月	人月	人月	人月	人月	人月	人月	人月	人月
		実績		人月	人月	人月	人月	人月	人月	人月	人月	人月	人月	人月
	その他実績工数[注8]	実績		人月	人月									
	レビュー工数(内数)	実績		人月	人月	人月	人月	人月	人月	人月	人月	人月	人月	人月
コスト	予算[注9]	計画		万円		万円	万円	万円	万円	万円	万円	万円	万円	万円
		実績		万円		万円	万円	万円	万円	万円	万円	万円	万円	万円
	外注コスト	計画		万円		万円	万円	万円	万円	万円	万円	万円	万円	万円
		実績		万円		万円	万円	万円	万円	万円	万円	万円	万円	万円

注1：各フェーズの内容に関しては、別表1を参照ください。

注 2：開発体制（外注化したか、社内開発か。および外注に出した場合は、その契約形態）を以下から選択ください。
（複数選択可）
（**1.** 委任契約　**2.** 請負契約　**3.** 自社開発）

注 3：要件決定者のソフトウェア経験度
（**1:**十分に経験　**2:**概ね経験　**3:**経験が不十分　**4:**未経験）を選択ください。

注 4：要件決定者の関与度（プロジェクト全体、フェーズ別）
（**1.**十分に関与　**2:**概ね関与　**3:**関与が不十分　**4:**全く関与していない）を選択ください。

注 5：要件決定者の要求仕様の明確さ
（**1:**非常に明確　**2:**かなり明確　**3:**ややあいまい　**4:**非常にあいまい）を選択ください。

注 6：要件決定者の要求仕様の変更発生
（**1:**変更なし　**2:**軽微な変更が発生　**3.**大きな変更が発生　**4.**重大な変更が発生）を選択ください。

注 7：工期／工数
工期は「時期（FROM/TO）」、「工期」のいずれか管理しているほうで記入ください。工程の途中で中断があった場合には両方を記入ください。工期は月数、工数は人月で共に小数点第一位まで記入ください。

注 8：開発工数／管理工数／その他実績工数
開発工数は開発SE／PGや開発チーム内の業務設計者等の工数を記入ください。
管理工数は基本ソフト等技　術サポート要員、ホスト・サーバ周辺システムオペレータ等の技術スタッフの工数および労務管理スタッフ、進捗管理スタッフ等の事務スタッフの工数を記入ください。
フェーズ別に分解されている場合はフェーズ別欄に、フェーズ別に分解できない工数はフェーズ共通欄に記入ください。
上記のいずれにも入らない工数はその他実績工数欄に記入ください。

注 9：予算は、ソフトウェア開発に係わる人件費・外注費、業務パッケージのコストを回答ください。
（自社内のハードウェア、ネットワーク等の費用および環境構築費用は除く）

Q4 目標設定

Q4.1 品質基準の有無

Q4.1.1　プロジェクト品質を計画する際に、ベースとした社内基準値はありましたか？下記より選択ください。

1.　Yes　　2.　No

Q4.1.2　Q4.1.1 の回答が Yes の場合の質問です。No の場合は次の設問にお進みください。
基準値の値と、その単位を（　　）に記入ください。

基準値（　　）　単位（　　）

Q4.2 生産性基準の有無

Q4.2.1　プロジェクト生産性を計画する際に、ベースとした社内基準値はありましたか？下記より選択ください。

1.　Yes　　2.　No

Q4.2.2　Q4.2.1 の回答が Yes の場合の質問です。No の場合は次の設問にお進みください。
基準値の値と、その単位を（　　）に記入ください。

基準値（　　）　単位（　　）

Q4.3 工期基準の有無

Q4.3.1　プロジェクト工期を計画する際に、ベースとした社内基準値はありましたか？下記より選択ください。

1.　Yes　　2.　No

Q4.3.2　Q4.3.1 の回答が Yes の場合の質問です。No の場合は次の設問にお進みください。
基準値の値と、その単位を（　　）に記入ください。

基準値（　　）　単位（　　）

Q4.4 計画品質の評価

Q4.4.1　プロジェクトで計画した品質水準を評価し、下記より選択ください。

1.　厳しすぎた　　2.　適当だった　　3.甘すぎた

Q4.5 計画生産性の評価

プロジェクトで計画した生産性を評価し、下記より選択ください。

1.　厳しすぎた　　2.　適当だった　　3.甘すぎた

Q4.6 計画工期の評価

プロジェクトで計画した工期を評価し、下記より選択ください。

1.　厳しすぎた　　2.　適当だった　　3.甘すぎた

Q5 信頼性

プロジェクトの信頼性について下表（表３）に記入ください。

表 3. 信頼度概要

	フェーズ別詳細								開発時合計	フォロー（運用後1ヶ月）
	システム化計画	要件定義	基本設計	詳細設計	製作	結合テスト	総合テスト1（ベンダー確認）	総合テスト2（顧客確認）		
レビュー[12]回数						－	－	－		－
レビュー指摘数						－	－	－		－
テストケース数										
報告不具合件数[13]（大）										
報告不具合件数（中）										
報告不具合件数（小）										
発生不具合件数（合計）										

[12]要件決定者が参加したレビューの事で、内部レビューは含みません。

[13]不具合（大）＝システムにとって致命的で緊急対応を要する障害
不具合(中)＝システムにとって致命的ではないが緊急対応を要する障害（大でも小でもない障害）
不具合（小）＝軽微で緊急対応の必要がない程度の障害

Q6 差異分析

Q6.1 工期差異分析

Q3 で、計画工期に対して実績工期が遅延していた場合の質問です。遅延していない場合は次の設問にお進みください。

Q6.1.1 工期遅延理由

工期遅延の理由と思われるものを選択ください。(複数選択可)

1. システム化目的不適当　2. RFP 内容不適当　3. 要件仕様の決定遅れ　4. 要件分析作業不十分
5. 自社内メンバーの選択不適当　6. 発注会社選択ミス　7. 構築チーム能力不足 8. テスト計画不十分
9. 受入検査不十分　10. 総合テストの不足　11. プロジェクトマネージャの管理不足
12. その他（　　）

Q6.1.2 工期遅延責任

工期遅延の責任の所在と思われるものを選択ください。

1. 責任は要件決定者側にある　　2. 責任は開発者側にある　　3. 責任は両者にある
4. いえない・分からない

Q6.2 規模差異分析

Q3 で、計画規模に対して実績規模が増大していた場合の質問です。増大していない場合は次の設問にお進みください。

Q6.2.1 規模増大理由

規模増大の理由と思われるものを選択ください。(複数選択可)

1. 見積もりの甘さによる仕様増加　2. 発注時の仕様詳細検討不足
3. 検討時の仕様増加　4. 発注時と運用開始時期の環境の変化による増加
5. 見積もりのカウント方法の差　6. その他（　　）

Q6.2.2 規模増大責任

規模増大の責任の所在と思われるものを選択ください。

1. 責任は要件決定者側にある　　2. 責任は開発者側にある　　3. 責任は両者にある
4. いえない・分からない

Q7 要員スキル

Q7.1 PM(ユーザー側)スキル

ユーザー企業側のプロジェクトマネージャ（PM）のスキルについて選択ください。

1. 多数の中・大規模プロジェクトの管理を経験 2. 少数の中・大規模プロジェクトの管理を経験
3. 多数の小・中規模プロジェクトの管理を経験 4. 少数の小・中規模プロジェクトの管理を経験
5. プロジェクト管理の経験なし

Q7.2 PM(ベンダー側)スキル

ベンダー企業側のプロジェクトマネージャ（PM）のスキルについて選択ください。

1. 多数の中・大規模プロジェクトの管理を経験　　2. 少数の中・大規模プロジェクトの管理を経験
3. 多数の小・中規模プロジェクトの管理を経験　　4. 少数の小・中規模プロジェクトの管理を経験
5. プロジェクト管理の経験なし

Q7.3 PM(ユーザー側)の業務精通度

ユーザー側 PM が、システム化対象業務に精通していたか否かについて選択ください。

1. 十分精通していた　2. ある程度のレベルまでは精通していた　3. 精通していたとはいえない
4. 全く経験も知識もなかった

Q7.4 PM(ベンダー側)の業務精通度

ベンダー側 PM が、システム化対象業務に精通していたか否かについて選択ください。

1. 十分精通していた　2. ある程度のレベルまでは精通していた　3. 精通していたとはいえない
4. 全く経験も知識もなかった

Q7.5 PM(ユーザー側)のシステム技術精通度

ユーザー側 PM が、開発システムのシステム技術に精通していたか否かについて選択ください。

1. 十分精通していた　2. ある程度のレベルまでは精通していた　3. 精通していたとはいえない
4. 全く経験も知識もなかった

Q7.6 PM(ベンダー側)のシステム技術精通度

ベンダー側 PM が、開発システムのシステム技術に精通していたか否かについて選択ください。

1. 十分精通していた　2. ある程度のレベルまでは精通していた　3. 精通していたとはいえない
4. 全く経験も知識もなかった

Q8 顧客満足度[14]

Q8.1 プロジェクト全体

プロジェクト全体の満足度について選択ください。

1.　満足　　2.　やや不満　　3.　不満
その理由（　　）
例:やや不満（当初の目的は達成したが、ビジネス環境が代わり使いにくくなった）

Q8.2 ソフトウェアの機能

ソフトウェアの機能に対する満足度について選択ください。理由についても記入ください。

1.　満足　　2.　やや不満　　3.　不満
その理由（　　）
例:不満（当初の目標を達成するための機能が不足していた）

Q8.3 工期

ソフトウェア開発の工期に対する満足度について選択ください。理由についても記入ください。

1.　満足　　2.　やや不満　　3.　不満
その理由（　　）
例: 満足（納期前に完了した）

[14]原則として、要件決定者から見た満足度を意味します。(以下同様)

Q8.4 品質

ソフトウェアの品質に対する満足度について選択ください。理由についても記入ください。

1.　満足　　2.　やや不満　　3.　不満
その理由（　　）
例:不満（納入時のバグが多すぎる）

Q8.5 ユーザービリティ

ソフトウェアのユーザービリティに対する満足度について選択ください。理由についても記入ください。

1.　満足　　2.　やや不満　　3.　不満
その理由（　　）
例: 不満（使用法が難しすぎる）／不満（何度も同じことを入力する必要がある）

Q8.6 開発コスト

ソフトウェアの開発コストに対する満足度について選択ください。理由についても記入ください。

1.　満足　　2.　やや不満　　3.　不満
その理由（　　）
例:不満（機能に対して割高）

Q8.7 開発マナー

ベンダー担当者の開発マナーに対する満足度について選択ください。理由についても記入ください。

1.　満足　　2.　やや不満　　3.　不満
その理由（　　）
例: やや不満（開発関係者間のコミュニケーションの不足）／満足（適切な情報提供があった）

アンケートへのご協力を有難うございました。下表に貴社、貴事業部の概要をご記入ください。

会社名・事業部名称	（ﾌﾘｶﾞﾅ）		
会社・事業部コード		プロジェクト連番	
業　種[15]		従業員:　　人	売上高:　　百万円
プロジェクト名			

[15]別表2産業分類から1つ選択し、該当する番号を記入ください

第３章　開発

第８節　利用部門の参画

＜利用部門はシステム開発にどのように関与すべきか＞

システム開発のプロセスにおいて、利用部門が果たすべき役割は大変重要であることは、第 3 章の各節に述べられている。本節には包括的に下記を述べる。

１．利用部門の関与の必要性
２．利用部門の関与の現状

１．利用部門の関与の必要性

（１）システム開発の各段階における利用部門の業務

システム開発案件の企画・仕様作成・開発・テスト・運用・保守の各段階において利用部門はどのような役割を果たすべきか。案件には利用部門のほか、IT 部門・ベンダー（場合によっては情報子会社）・その他の部門が参画する。利用部門でなければ果たせない役割があり、もしその役割の果たし方が不十分であれば、段階が進んだ後に何らかの不具合が発見されて、段階を遡って改めてその役割を果たすことになる。以下に各段階における必須の利用部門の役割について述べる。

段階	利用部門の役割	注意事項
企画	企画部門が新しいビジネスモデルを考案したときには、それをレビューする。 自ら、ビジネスモデルの改革をする。 案件がもたらす効果（メリット）を想定し、コミットする。 効果を実現するための条件を洗い出し、条件整備の実現性を検討する。	自らビジネスモデルを革新することは利用部門の本来的役割である。 利用部門でないと、実現性の高い革新のシステムを考案しがたい。 利用部門自らが新しいビジネスモデルを起案する。もし智恵不足であると感じたら、社内のスタッフやベンダーに智恵を借りること。
仕様作成	案件に沿ったビジネスルールを明確にする。 ビジネスプロセスを明確にする。 取り扱うデータを厳密に定義する。（データの移行条件を含む）	利用部門でないと、例外的なルールを設定することが難しい。 ルール、プロセスの明確化およびデータ定義を行う。
開発	データ、テーブル、ファイルなどの設計書をレビューする。 プロトタイプのシステムにより、仕様の適合を確認する。	機能だけではなく、性能についてもレビュー対象にすることが肝要である。 後出しの仕様変更は最小限に抑えることが必要である。
テスト	テスト計画書をレビューする。 テストデータを準備する。 テスト結果をレビューする。	テストの作業負荷の程度もレビュー対象である。 テストデータは単体テストの開始に間に合わせる。
運用	システムの移行条件を事前に準備する。（オペレーションの習熟等） システムの移行時期を意思決定する。	移行の事前準備が十分でないと、混乱して元に戻れなくなることがある。
保守	仕様の改良を提案する。	仕様改良はメリットを明確にして提案する。

図表 3-8-1　利用部門の役割

（２）利用部門によるビジネスシステムの定義

システム開発案件の仕様開発段階で利用部門は、次の役割を果たさなければならない。これらは、利用部門でないと、適格に実施できない。

・案件に沿ったビジネスルールを明確にする。
・例外的な条件や作業を列挙する。
・ビジネスプロセスを明確にする。
・取り扱うデータを厳密に定義する。（データの移行条件を含む）

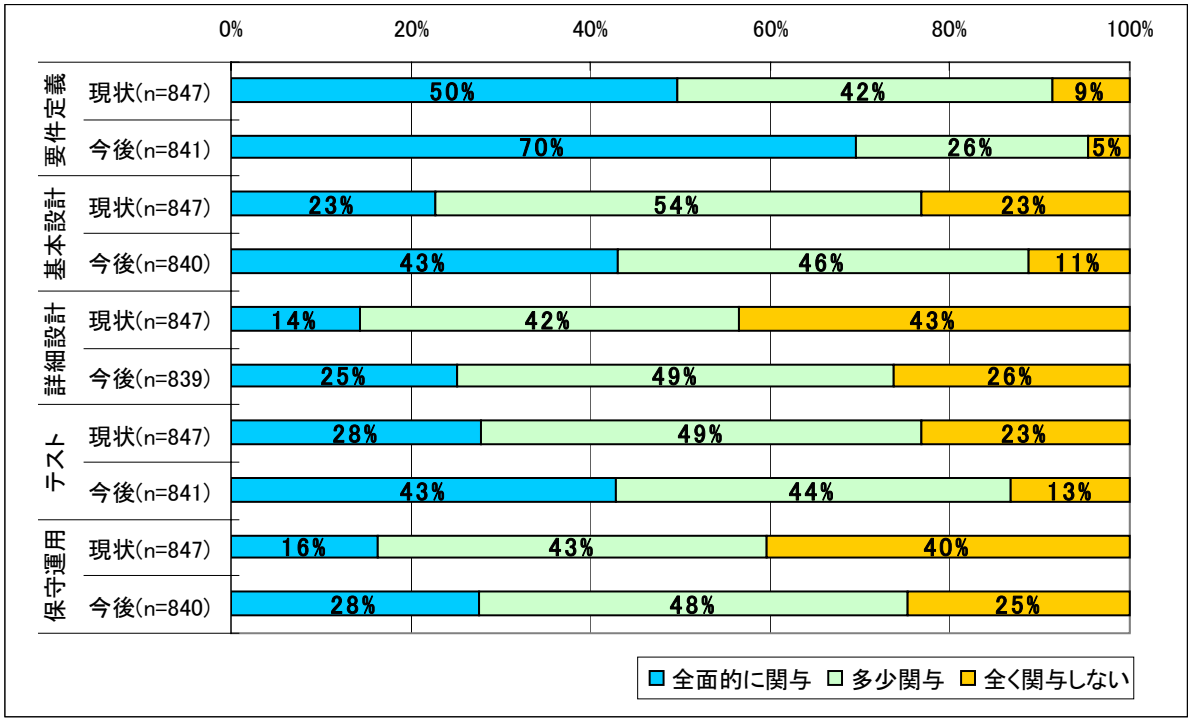
手順	内容	説明	ドキュメント
①前提条件の確定	対象機能範囲	全社のビジネス機能内における位置づけを明確にする。	ビジネス機能関連図
	対象の商流・物流・金流	社内外の機能との関係を商流・物流・金流の側面から明確にする。	ビジネス連携図
	ビジネスルール	商流・物流・金流等の分類ごとに準拠する基本的なルールを明確にする。	ビジネスルール定義書
	システム化の目標	システム化の狙い（効果）を明確にすると共に、効果を実現するための条件を明確にする。	システム化目標定義書
②ビジネスプロセスの定義	対象範囲のビジネス機能構成	ビジネス機能を細分化し、階層図を用いて機能（例外処理を含む）の階層構造を明確にする。	ビジネス機能構成表
	重要なビジネスプロセスの関連	ブロック図を用いてビジネス機能（中分類程度）に関連する部署の関係を明確にする。	ビジネスプロセス関連図
③業務フローの定義	ビジネスプロセスと関連部署の関連	ビジネスプロセスをビジネスルールに従って処理する手順を関連部署間のフローとして明確にする。	業務流れ図
④業務ルールの定義	ビジネスプロセス間の情報	ビジネスプロセス関連図に機能情報を付加して、情報の流れを明確にする。	機能情報関連図
	業務の処理基準	ビジネス機能（細分類）ごとの業務処理基準を準拠するビジネスルールの規定によって明確にする。	業務ルール定義書
	業務処理手順	ビジネス機能（細分類）ごとの業務処理手順を HIPO またはフローチャートにより明確にする。	個別業務処理定義書
⑤ビジネスデータの表現	画面・帳票	ビジネス機能（細分類）ごとに、画面と帳票の名称および用途等の属性を一覧表の形式によって明確にする。	画面・帳票一覧
		画面および帳票ごとにレイアウトを明確にする。	画面レイアウト 帳票レイアウト
	データ項目	画面および帳票に含まれるデータ項目について、項目名および説明などの属性を一覧表の形式で明確にする。	データ項目定義書
⑥操作要件の定義	操作条件	エンドユーザーの操作方法を明確にする。	運用・操作要件書
	運用条件	システムの運用方法およびレスポンスおよびスループット等の性能を明確にする。	
⑦データ要件の記述	データ要件	トランザクション発生量および保管データ量を明確にする。	データ要件書

図表 3-8-2　利用部門によるビジネスシステムの定義

２．利用部門の関与の現状

「IT 動向調査」によれば、ユーザー企業のシステム部門のみならず、利用者は、自分たちの要求を正しく伝えるためにも、利用部門が望むシステムを入手するためにも「システムの利用者である利用部門はもっと開発に参画する必要がある」といっている。

ユーザー企業の中のシステム部門のみならず、システムの利用部門がもっと協力したいという意欲を持っているのは心強い。



（IT 動向調査 2004）

図表 3-8-3　利用部門の関与

現状では、全面的に関与している企業の割合は、要件定義フェーズで 50%、基本設計フェーズで 23%、詳細設計フェーズでは 14%、テストフェーズで 28%、保守運用フェーズでは 16%となっている。

今後の方向性としては、「全面的に関与」したいという企業がどの段階でも増加している。要求定義フェーズでは 70%、基本設計、詳細設計フェーズでは、43%、25%、テストフェーズでは 43%、保守運用フェーズではまた 28%である。いずれにせよ、現状に比べ、今後の方向性としては利用部門の関与を深めるべきであると認識されている。

一方、現状では、要件定義、基本設計に利用部門が全く関与しない割合がそれぞれ 9%、23%あるとの結論が出ている。業務に役立つ情報システムを構築するためには、「何を作るべきか」を決定しなければ、この目的の達成は困難である。ましてや、システムの信頼性を向上させることもできない。

また、テストに関しても、利用部門がより深く関与することによって、業務実態に即したテストが実施可能と思えるし、利用部門自身も必要性を認識していると思われる。いずれの段階でも今後は「全面的に関与」したいという企業が増加している。（IT 動向調査 2004）