オーム社/雑誌局

ISBN4-274-50076-4

C3055 ¥1714E

9784274500763

1923055017146

定価(本体1714円【税別】)

IPA 独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター

SEC-TN05-002

PRINTED WITH SOY INK

R100

SEC BOOKS

経営者が参画する要求品質の確保 ～超上流から攻めるIT化の勘どころ～ 第2版

独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター 編

Ohmsha

SEC BOOKS

経営者が参画する要求品質の確保
～超上流から攻めるIT化の勘どころ～ 第2版

独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター 編

解説・CD-ROM付

Ohmsha

SEC BOOKS

経営者が参画する要求品質の確保

～超上流から攻めるIT化の勘どころ～

第2版

独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　編

解説・CD-ROM付

Ohmsha

第２版改訂にあたって

「経営者が参画する要求品質の確保」（第１版）を発行してから１年が過ぎました。この間に、同書は、大きな反響を得ることができました。

その中には「非常に分かりやすい」、「共感できる」など賛同する意見の他に、「狙いは良いのだが、用語の矛盾や独自の用語が使用され理解を妨げている」や「もうちょっと具体的な内容を知りたい」、「言っていることはわかるが、次に何をするかを示して欲しい」といった意見も頂いております。

そこで、改訂にあたっては、用語の一部を最新の事情にあわせて統合・整理を図ると共に、基本的な用語や説明が必要と思われる箇所に適切な解説を参照可能にしたCD-ROMを添付しました。

また、すべての関係者の心得として、システム開発の超上流フェーズを進めるにあたって気を付けるべきポイントを、17個の原理原則とその基本的な考え方、および行動規範という形でまとめました。

今後もSECではソフトウェアエンジニアリング発展に惜しみない支援を行なってまいります。これを契機に、ますますのご活用をお願いいたします。

2006年4月　開発プロセス共有化部会　一同

はじめに

思い当たることはありませんか？

・IT システムで何がしたいのか、何ができるのか、はっきりした答えがだせない
・はっきりした答えを出せるユーザ企業は多くないと思う
・IT システムの開発要件がはっきりしていない案件がある
・開発に入ったあとで、開発規模が膨れた経験がある
・運用テストに入ってから、大幅な手戻りした経験がある

近年このような経験を見聞きする機会が増えたと思いませんか？

本小冊子は、IT システム開発やソフトウェア開発に関係する人々が日頃悩んでいる問題にずばり解決のヒントを与えるものと思います。

情報化にあたってユーザ企業の経営層や利用部門は何を認識しなければならないのか。ユーザ企業の IT システム開発・調達の現場および、その実現のパートナであるベンダ企業では現在いったい何が起こっているのか。これらを明らかにし、それぞれの問題の本質が事業や業務検討の始まりから要件定義までの、「超上流」工程にあることを突き止めました。

これは、誰がいつ何をしなければならないのかを超上流での関係者（経営層、利用（業務）部門、情報システム部門、ベンダなど）が共通に認識

し、それぞれが果たす役割と責任を明確にして課題解決に当たるというものです。

したがって、本小冊子は、IT システムを駆使して事業を行う経営者から、その実現に協同してあたるベンダまであらゆる関係者に是非読んでいただきたいものです。

本小冊子によって、これまで果たせなかった課題に解決の糸口を与え、プロジェクトの成功に大きく寄与できることを願ってやみません。

2005年 4 月　開発プロセス共有化部会　一同

目　次

● 文章中のアンダーライン部分は、CD-ROM にてクリックすると、解説が表示されます ●

第 2 章　IT システム開発・調達の現場で何が起こっているか

2.1　背　　景

2.2　IT システム開発・調達の現場で起こっていること

第 3 章　要求品質の確保に向けて

3.1　開発プロセス共有化への取り組みにあたって

3.2　開発プロセス共有化部会のねらい

第 4 章　超上流工程でやるべきことと役割分担

4.1　工程の考え方

4.2　システム化の方向性およびシステム化計画工程での役割と成果物

第１章　情報化にあたって経営者は何を認識すべきか

本章では、ユーザ企業側からみて、現在どのような問題が発生しているか、それに対する解決の方策は何かを示すことを目的としています。現状の背景から説き起こし、問題発生のメカニズム、発注側、受注側のそれぞれの役割を認識することが重要であることを説明します。

1.1　背　　景

20世紀の後半から始まったコンピュータの高度活用は、ビジネスの仕組みを根底から変えました。まさに新しい技術が新しいビジネスモデルを次々と可能にし、そのことにより、われわれのビジネスは画期的に発展し、同時に生産性の革新を実現しました。バッチシステム中心の時代から、営々と作り上げられた業務システムは、いまや企業の枠を超えて広がり、ビジネスの隅々にまでいきわたり、社会基盤となりました。

一方、システムの使い方も大きく変わりました。業務の結果をバッチにしてコンピュータに入れて高度活用するといった、今となっては牧歌的な時代は終わり、誰もがコンピュータからのアウトプットで業務を行う、コンピュータが止まるとビジネス、社会の動きが止まる、という時代となったわけです。システム障害が社会の大きな事件として、新聞紙面をにぎわすようになったのは決して偶然ではありません。

ところで、システム開発の現場は、最初から最後まで人間の手作業によ

るところが多く、その道具の発達ほどに進歩していないというのが実態です。プログラム言語を一通り学んだだけで、あとは、勘と、経験と、気合だけといった「３Ｋ開発」が現実…、といった指摘も真っ向否定できる状況にはない中でのシステムの巨大化は、障害発生時の影響を大きくする一方のようにも思われます。

21世紀に入り、ネットワーク化されたシステムに支えられた社会の安全性、生産性を保証していくためには、「システム開発が100％人間技による、という事実を踏まえたシステム開発に関わる品質向上施策強化」を追求することが必要です。

我々は、「システム開発には、たとえ自動化がどんなに進んでも、自動化し得ない部分がある、そしてそれこそが本質なのだ」ということを正しく認識することが大切です。

1.2　問題発生のメカニズム

（１）システム開発は人間技

システム開発の道具は、コンピュータ技術の発展と並行して進歩してきました。カードパンチをして、紙の束に打ち込まれたプログラムをコンピュータに読み込ませた時代は過去のものです。しかし、プログラムをコンピュータに登録し、稼働させるための道具はどんなに進歩しても、プログラムの作成はもちろん、そもそもプログラムに何をさせるかについては、最初から最後まで人間の作業に依っています。

コーディングの作業環境は向上し、キーインミスによる障害は減少しました。しかし、要件定義も、プログラミングも、テストケース作成も、テストも、本番環境設定業務も、本番移行も、相変わらず100％近くが手作業であるのが現実です。そこには人間なら誰でも犯すであろう誤りがついて回ります。

人が普通に仕事をしていて犯すミスは、その作業の0.3％くらいではないでしょうか。例えば、原稿用紙に文章を書くと、誤字、脱字が１文字くらいは必ずあるといえるでしょう。プログラミングについても、さまざまな手続き作業において、やはり同様のことがいえます。プログラミングを一度でも経験された方ならすぐにわかるでしょうが、単純なコーディングミスに自信が失せていく自分を思い出しませんか？

要件定義においては、なおさらミスは構造的に発生するでしょう。なぜならば、それがある意味では答えのない企画業務だからです。

（２）広がるシステム利用と多様化する要求

20世紀、新しい技術は新しいビジネスを可能にする。という信念のもと、我々は次々とフロンティアを文明化（システム化）してきました。その結果、先に述べたように、すべてのフィールドはシステムに覆いつくされ、その利用者は、システムを作る企業の中から外へ外へと広がっていきました。かつてシステムは、企業内で自らの業務を合理化、効率化したい担当者が、自分が使うないし自分の部門が使うニーズを満たすために企画され、発注されました。そのとき、要求はその範囲で明確であり、的確でぶれのないものでした。

21世紀の今日、状況は全く違ってきています。企業の中で業務企画をし、システム構築を依頼する担当者は多くの場合、自らがそのシステムを利用することはありません。業務企画担当者は、いまやシステムの発注者でありながら開発を担う担当者とともに、利用者にシステムを供給する立場にいます。

利用者の要求は多様化し、システム開発における要求定義、要件定義を複雑・不安定にする一方といえます。

（３）ステークホルダの増加

開発に関係するステークホルダ（利害関係者）も増えています。

業務システムが相互に関係を持つようになった結果、ひとつ施策を実現するためには、既存のシステムを広範囲に修正しなければならず、開発に参加するシステム担当者が増えてきました。また、企業と企業が、ネットワークでつながれた結果、企業同士で連携をとって進めなければならない開発も増えました。

アウトソースも盛んに行われています。運用がアウトソースされていれば、その委託先との緻密な調整は不可欠です。また、コールセンタや、ヘルプデスクの担当者も稼働時の必要要件を詰めるため、システム開発では重要な役割を果たします。ネットワークでつなぐ相手がお客様である場合は営業担当もシステム開発プロセスに参加しなければうまくいきません。

20世紀においては、プログラムを徹底的にテストし、完成させ、正しく本番環境に移行設定すれば、それでシステム開発はうまくいきました。しかし、21世紀の開発では、どんなに完璧にプログラムプロダクトを完成させても、例えば本人確認機能といったセキュリティ環境構築に関わる契約が一つ漏れていれば稼働できない事態が起こります。

システム開発に関わるステークホルダを漏れなく洗い出し、担当ミッションを明確にし、開発プロジェクトに参加してもらわないとうまくいきません。要求の明確化も、要件定義も、プロジェクト推進もステークホルダの適切な参加なくしてはうまく機能しない、そういった時代になっています。

（４）開発は調査・修正中心になった

ある大手損害保険会社では、毎年、本番移行するプログラムが、２万本程度あります。その中で修正の占める割合は、おおよそ80%を占めます。残りの20%は、新設プログラムになりますが、このプログラム群も大きく見ると既存のシステムの拡張、機能追加ということであり、これまた広い意味での修正となります。企業の業務が広くシステムに覆いつくされ、相互に密接な関係を持って動くようになってきていますから、何か新しい施策を打つためにシステム開発をすることになりますと、保有するレガシーともいえるシステムに手を入れなくてはならなくなり、その範囲も広く、また量も多い状況になっているわけです。

その結果、たった一つの項目を追加するのに保有システムを数百箇所直す…などということが生じます。

（５）オープン化対応

20世紀の最後の10年にオープン化の波が押し寄せ、大型コンピュータをお釈迦様の手とし、アプリケーションが孫悟空よろしく自由にその上を飛び回っていた環境は大きく変わりました。その技術の進化、変化はきわめて早く、パソコンは数年で陳腐化し、サーバにも次々と新しい技術が登場しています。

アプリケーション開発といった利用技術についても、ハードウェア、ネットワークといったインフラストラクチャ（以下、インフラと略す）の活用についても、業務企画にベストフィットしたしつらえを実現するためには、全体最適の実現、パフォーマンス、キャパシティ、セキュリティ、障害対応、稼働時間、運用監視など、必要な非機能要件を踏まえたオープン化ならではの検討が、その都度要求されます。

しかし、一般のシステム利用企業には、その進歩、変化の早さ、広がりの大きさ、多様さから、この分野についての専門性のある要員を十分に抱え込むことが簡単にはできません。新しい器に対応できる、体制の充実が急務になっています。

1.3　新しい時代の、新しい役割（解決の方向性）

システムは21世紀に入り、ますますその重要度を高めるばかりです。優秀な業務企画担当者が、次々と画期的なビジネス構想を組み立てても、システムによってその形を得られなければ絵に描いた餅にすぎません。

ビジネスに命を吹き込む…、今やシステムは、単なる生産性の向上や、合理化の推進といった役割だけでなく、競争力の強化を実現し、優位性を確保し、そしてなによりも企業の存在価値を形にするために不可欠なものとなっています。

（１）経営者の役割――IT（Information Technology）ガバナンスの実現

① システム部門への重点投資

企業がその事業において価値を創出し、競争優位を獲得し、社会的責任を果たすために、経営者はシステムをその中心において戦略を描く必要があります。すべての業務がシステムの上で動くようになった今日、ビジネス戦略は、システム開発力なくしては実現しません。

IT を担当するシステム部門に重点投資するとともに、価値を創出するバリューパートナとしてのベンダを獲得し、さらに、IT 担当とともにシステム開発を担う業務企画担当者を増やし、開発・発注キャパシティを拡大することで、多様なマーケットニーズに応えていく、これが競争優位実現の鍵となります。

② 選択と集中の実現

しかし、システムの開発力は、自在に増やすことができるものではありません。加えて、システムは作れば作るほど、その保守にも能力あるシステムエンジニアを必要とします。限られた開発力を最大限に生かすためには、開発対象の戦略的選択と集中が不可欠となります。

経営の観点から真に必要な開発を決定する。個別部門の個別最適企画を政治的に処理して開発案件を決定するのではなく、経営の責任で選択し、人、金、時間といった資源を集中していくこと、つまり「選択と集中」、これが IT ガバナンスの要として求められます。

③ 開発品質向上と障害管理

次に、IT ガバナンスのうえで重要な点は、開発品質向上と障害管理です。

すべての業務がシステムの上で動くようになった結果、システムが実現する機能、アウトプットに誤りが発生した場合、その影響は大きく、社会に混乱をもたらし、企業の存続に危機をもたらすことも考えられます。

システムの利用は会社の中から外へ、ネットワークに乗り大きく広がり、その重要さは増すばかりです。30年前のバッチシステムの時代では、システムが止まっても会社業務が止まることはありませんでした。しかし、現在は違います。システムが止まると業務が停滞する会社が増えています。システム化の進んだ大企業ほどその影響が大きくなります。このことは、お客様にも影響を及ぼすことになります。例えば、ATM（Automatic Teller Machine）、為替や証券の取引、交通機関の予約など、万一ネットワークがシステムダウンした場合の社会的影響は計りしれません。

システム開発品質を高め、障害の発生をおさえ込み、万一のときにもじん速に組織を挙げて影響を最小限におさえられるように、そのための組織と体制を整え、そして経営自らが危機管理の要として機能させることです。

したがって、システム障害リスクは経営リスクです。このことを正しく理解し、開発品質向上を実現し、システム障害リスクマネジメントを実現した経営者こそが、社会的責任を果たせる存在であるといえます。

④ 投資効果（ROI：Return On Inverstment）の視点からのマネジメント

従来、システム開発は、そのプロダクトの生産性の範囲で管理されていたともいえます。システム開発のアウトプットは、プログラムであり、その評価といえば、作るところに焦点の当たった規模あたりの開発生産性でした。

しかし、システムを手に入れたい経営者は、プログラムの塊を手にしたいのではなく、プログラムの塊が実現するビジネスの成果を手に入れたいのです。そうであるとすると、システム開発は、そのコストをビジネスの成果との関係でマネジメントすべきです。

ビジネスのサイズにあった、適切なサイズのシステム実現への一歩は、システム開発における投資効果分析および評価にあります。

＊ROI…RETURN ON INVESTMENT

一般的には、投下した資本がどれほどの利益を生んでいるかを測るための指標。システム化の場合、システム化コストに対してどれだけのビジネス上の効果が出ているのかを測ります。

（2）業務企画担当の役割

① 供給責任

かつてシステムを発注するのは、多くの場合、企業の中で業務を企画する部門の担当者であり、完成したシステムを使うのは、発注した担当者自身ないしはその所属する部門、管理する部門でした。そこでは発注者が利用者であり、同時に評価者であったわけです。しかし、21世紀の今日、事情はだいぶ変わりました。

企業が開発するシステムはもちろん、いまだにその多くが企業の中で利用されています。しかし、それ以上に企業の外で利用されるようになってきました。システムは、販売店、代理店や業務のアウトソース先、そして顧客に提供されるようになり、24時間、365日活用されるようになりました。以前のように、利用者が発注者として疑問の余地のない明確な要件を提示でき、でき上がったものに満足すればよい、という状況はなくなりつつあります。

いまやマーケットないしマーケットの近くにでき上がったシステムの利用者がおり、業務の企画、設計者はシステムの提供者として、考えた業務の仕組みをシステムにして供給する役割と責任を、情報システム部門、ベンダとともに担うようになったといえます。業務企画担当者は、アプリケーションの責任者として、また個々のプロジェクトの責任者として、情報システム部門、ベンダとともに要件を確定し、システムの検証をし、利用者

に供給する必要があります。

② ステークホルダ間の合意形成

システム開発に関わるステークホルダが増えてきています。

システムのカバーする範囲が広がった結果、システム開発に関係する部門も増え、利用者がマーケットに広がったことで稼働してからのサービス担当もステークホルダに加わりました。

また、企業間接続によるサービスの提供進展により、相手企業のシステム部門もまたステークホルダとして担当営業などとともにコミュニケーションの対象となりました。システムの運用をアウトソースしている場合は、その委託先もステークホルダに入るでしょう。

プロジェクトを起こした業務企画担当者は、プロジェクト責任者として、これらステークホルダの方針、意見、課題などについて、漏れなく綿密に把握し、できることとできないことをIT担当者、ベンダとともに切り分け、業務要件として取りまとめていく責任を果たす必要があります。

ステークホルダもまた、システムの供給側に立つ場合は、積極的に要件の策定に参加し、利用者ニーズを確実に把握して、正確にシステム機能に反映していくことが必要です。したがって、ステークホルダの果たす役割は、きわめて重要なものになっています。

③ 要件確定責任と説明責任

業務企画担当者は、構築しようとしているシステムの要件を明確に確定させ、承認する責任があります。また開発者に対しその要件を説明する説明責任があります。単なる説明、通り一遍の説明ではなく理解してもらう責任を負っています。

システム開発の受託側から見た原則は「受託した要件として、書いてあるものは実現させる。書かれていないものは作らない。」ことです。システムは決めたとおりに作られ、決めたことに理解の誤りがない限り、正しい

結果を生み出します。もちろん、物事をはじめるときに、すべてが誤りなく責任もって確定できることは神業です。よって、決め事も、それに基づくシステム作成も、人間技である限り、見込み違い、思い込み、決め付け、聞き違い、聞き漏れといったことからくる「誤り、漏れ」と無縁ではなく、システム開発は、そういったことにより発生した「誤り、漏れ」を解消していく過程ともいえます。したがって、以心伝心的な要件伝達は、厳しく戒められなければなりません。

システム開発における万全なる準備は、次工程に向けての正確な要件という情報の伝達であり、それを実現するためには、自分が次工程に伝える必要のある情報について、漏れなく要件確定責任と説明責任を負う必要があります。「要件の行間を読め」ということを要求してはいけません。基本的には当たりまえの前提や例外処理であっても漏れなく伝達する必要があります。

また、同じ言葉を聞いても頭に浮かぶものが異なるのが人間です。かつてメートルとヤードを取り違え、宇宙の塵になってしまった火星探査機がありました。このように発注側、受注側双方の説明責任が、多様化した要求と、複雑化したシステム開発において品質を確保する重要なポイントとなることは間違いありません。

（３）IT 担当者の役割

① 全体最適の実現

システムは作り上げるまでは可能性の塊ですが、でき上がった瞬間に制約と条件の塊となります。そのため営々と作り上げてきたシステムを踏まえることなく次のシステムを実現することはできません。

20世紀に営々と築き上げたシステムは、企業の基盤として大きな役割を果たしていますが、同時に、何をするにもその仕組みに手を入れないと実現しない状況を生み出しました。多くの場合、個別の業務ニーズに対応して構築されたアプリケーションシステムは、それ自体、独自の構造を持つことが少なくありません。特化した業務知識を必要とすることもあり、共通的なソフトウェア知識だけでは解読が難航することも少なくないわけです。「マタギ」や「職人の棟梁」的暗黙知が必要といわれるゆえんです。

アプリケーションを稼働させるシステムインフラも同様です。ネットワーク上に異なった規格の端末が接続展開されている場合などは、維持管理に大きな障害となります。

長大なレガシーともいえるシステムを抱えた企業は、一方で新たなビジネスを実現するために戦略的なシステム開発を進めなければならない状況にあります。IT 担当者は企業のおかれたこのような状況を踏まえ、全体最適の観点からアプリケーションや、インフラを見直さなくてはなりません。そうすることによって、企業が新しい挑戦をする条件を整えていく必要性が生じます。保有システムを可視化し、全体最適の観点から共通的で一貫した技法、構造を定義、活用できるようにすることこそが、これが IT を担当する企業内担当者、ベンダに課せられたミッションといえます。

② 非機能要件の取り込み

プログラムを漏れ誤りなく作成して、コンピュータに登録し、期日どおりに稼働させることで、システム開発が完了する時代は、過去のものとなり、いまではプログラムの完成と同時にプログラムを取り巻く環境を整える必要があります。例えば、端末認証の仕組みの契約が完了していなければシステムは稼働しません。また、ネットワークの容量を増やす必要があればプログラムの開発と並行して必要な手配をする必要があります。

一方、利用者が広がるにつれ、要求は多様化し、不確実なものとなり、保有システムは足かせのように業務企画担当者、IT 担当者にからみつきます。「実現したいこと」と、「実現できること」の引っ張り合い調整は、システムの要件を確定する際、避けては通れない作業といえます。

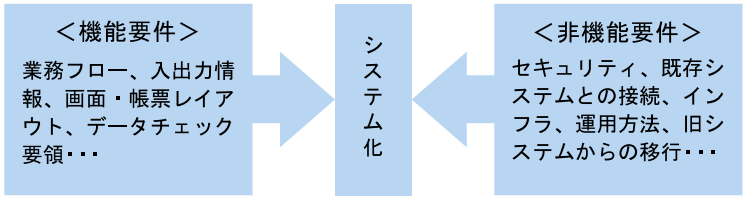


図 1.1　機能要件と非機能要件

図 1.1のようにプログラムのインプット、アウトプットの定義といった機能要件に加えて、現在のシステム開発では、このように多種多様な非機能要件を定義し、開発に取り込んでいかなければなりません。非機能要件自体も多様化しています。個人情報保護やセキュリティ、コンプライアンスといった事項は、すでに有力な機能要件の内法となってきました。しかし、24時間、365日稼働といったことについて、発生する制約を正しく理解していない業務企画担当者，IT 担当者がいることもまた事実です。障害時の対応設計といった重要かつ、基本的な事項について、設計が後回しになるという事例は珍しくありません。

個々のプロジェクトだけに発生する要件もまた存在します。プログラムを共有する他システムに同時に改定が入る場合などは、プログラムの修正要求が機能要件とは無関係に発生します。また、認可事業なら認可条件、各種制度改定、企業の機構改革も同時期に発生した場合、システム開発に関わる非機能要件となりえます。

インフラに関わる改定、例えば、サーバの保守切れ、OS（Operating System）のバージョンアップなどもプロジェクト遂行時に発生すれば、非機能要件として管理対処しなければなりません。ネットワークを使って、注目されている話題性の高い新規のアプリケーションシステムを数多くの利用者に提供を始める場合などは、スタート時の利用アクセスの集中をコントロールする仕組みが必要になることもあります。これまた立派な非機能要件といえます。

このように、企業内情報システム部門の担当者は、業務企画部門、ベンダと協力し徹底的に非機能要件を洗い出し、管理をし、システム開発に反映させる必要があります。機能要件を正確にシステム要件に落としていくのと同じように、あるいはそれ以上に重要な業務として、非機能要件反映管理を役割として実践しなければ、間違いのない開発を実現できません。

③　「精緻主義」の排除

IT 担当者の職業病として知られているのが、条件反射的に発病する「何とかしなければいけない病」です。しかし、多くの場合「頼まれるままに何とかしなければいけないものなのか、対象、内容についてよく考えて着手する必要があるもの」でもあります。

同様に、芸術家的プロ意識がそうさせるのか、作るとなったら最高のものを、という「精緻主義」、「完璧主義」も病気の一つかもしれません。この病気は、業務企画担当者もよくかかる病気ですが、「何とかしなければ病」と重なって全体最適や、ビジネスに適正な仕組みを実現するための阻害要因となります。つまり、普通のタクシー営業にロールスロイスは不要ということです。

品質とグレードを勘違いしてもいけません。確かに、システム開発は「100点以外は 0 点」という世界であり、バグは困ります。品質は最高であるべきですが、しかし、100点の内容はビジネスの内容にフィットしたものでなくては困ります。同じ機能を実現するためのシステムの作り方は多様です。IT 担当者は、業務要件をシステム要件に落としていく際、シンプルで使いやすい設計を実践することが重要です。

課題を解決するソリューションはさまざまです。IT 担当者にはプロとしてシステムを用いた、ビジネスのベストソリューションを提示し、それらのポイントを丁寧に説明する説明責任があり、そのための説明力が必要で

す。

④ 開発品質向上の実現

先に、システム障害リスクは経営リスクであることを述べました。IT担当者は、システム開発のプロとして開発品質を高めることにより経営リスクの極小化を実現することが大切です。したがって、当然ながら、業務企画担当者もまた、システムの実現可能性を把握し、責任を持って業務要件を必須妥当なものに落とし込んでいかなければなりませんが、そのための情報を提供するのは、ITを担う企業内担当者、またはベンダということになります。

加えて、非機能要件のところで指摘したとおり、現在のシステム開発はプログラムに実装される機能要件の充足だけでは完了しません。企業の経営者も、業務企画担当者も、プログラムの塊を手にしたいのではなく、プログラムの塊が実現するビジネスの成果を手に入れたいわけですから、プログラムプロダクトオリエンテッドなシステム開発管理から、プロジェクトオリエンテッドな開発管理にコンセプトチェンジが必要です。

プロジェクトに参加するシステム供給側のステークホルダに関わる役割を明確にし、その作業進捗を管理していくことはIT担当者の重要なミッションです。また、プログラマからキャリアアップしたIT担当者は本能的にプロダクトであるプログラムの完成に没頭しがちです。どんな仕事でも担当者の決め込み、抱え込みは大きなリスクの原因になりますが、現在のシステム開発においては、開発品質を下げる原因の大きな一つになります。したがって、進捗管理、プロジェクトレビューを活用することにより、開発担当者の決め込み、抱え込みを排し、プロジェクトを可視化し、担当者に情報を供給し、タイムリに必要なアクションがとれるようにすることが、了解違い、認識漏れといったことを防止し、プロダクトの品質を上げることになります。

（４）ベンダの役割

ベンダは顧客企業のIT担当者とともに、企業のビジネス戦略実現を担う存在です。共に顧客企業のシステム開発を担い、顧客企業にIT担当者がいない場合は、担当者に代わって、その役割を引き受ける必要があります。加えて、顧客企業のIT担当者が、特定の産業に特化した存在とは違い、ベンダはシステム開発のプロとして、その技術力、構築力を用いて、顧客企業の戦略実現を支える役割を強く担っているといえます。

① バリューパートナ

新しい技術は新しいビジネスを可能にする。この事実はいまだ色あせていません。コンピュータをめぐる技術は、まさに日進月歩であり、システム技術の革新は、ビジネスの革新を推し進めています。

この時代にあって、ベンダは企業のビジネスにおける戦略目標を共有化し、システム技術を活用して、ビジネスの価値を創出する担い手でなければなりません。

そのためにはシステム開発、運用につき、ビジネスの性格、規模に応じた柔軟で適正、合理的な提案をすることが肝要で、開発およびプロジェクトマネジメントについても可視化に努め、高品質を実現しなければなりません。単なるプログラムプロダクトの供給者ではない、顧客企業のビジネスを通じて、顧客企業とともに価値を創出する、バリューパートナとなることが求められます。

② ソフトウェアエンジニアリングの担い手

コンピュータ上で動くプログラム、すなわちソフトウェアは今日、もっとも重要な技術の一つです。システムを用いて業務革新を実現してきた企業も、ベンダが開発する新しいソフトウェア技術を活用して、その構想を実現してきました。そのためベンダは、早く、安く、うまい（使い勝手がよく高品質な）システム開発ができる技法を、技術革新にあわせて提供し

てきました。その役割は現在も変わっていません。むしろ、ますます重要になっているといえます。

システムを開発するために必要な技術は、ソフトウェアエンジニアリングとして、これからもベンダから提供されるでしょう。また、ベンダはその活用についても、プロとして引き続き、利用企業に対し指導力を発揮していくことが期待されます。

③ 見積り手法の充実

ホストコンピュータでバッチのシステムが動いていた時代には、プログラムの作成量がおおむね開発付帯業務を含めた開発業務量と相関関係を保っており、システム開発担当者は、プログラムの作成ステップ数を見積り、計っていれば適正なコスト管理ができていました。

オープン化の時代、開発言語の多様化や画面や画面遷移後の自動コード生成などの新しい開発技術により、プログラムステップ数による規模見積りを困難なものにし、開発計画、とりわけ費用見積りをむずかしいものとしました。加えて、システム開発にはプログラム作成以外のコスト要素がたくさん入り込み、従来システム開発を担ってきた企業所属 IT 担当者の手にあまる状況となっています。新しい時代の、新しい技法にふさわしい見積り手法の開発が期待されます(注)。

④ ビジネス構想の段階から戦略パートナとして参加

システムをどう作るかが、要件定義のほぼすべてを占めていた時代には、システム開発を担うベンダは、何かを作るところから仕事を始めれば、その役割をまっとうできたといえます。しかし、現在のシステム開発では既存システムへの影響、客先や代理店などといった社外のステークホルダの

（注）SEC では、見積り手法に関して小冊子「IT ユーザとベンダによる見積りの勧め」及び「ソフトウェア開発見積りガイドブック」を発行し、プロジェクトを成功に導くための見積りの考え方や手法を示しています。

存在、新技術の活用といったことも考慮しなければなりません。このため、システム化計画のための周到な調査分析と実現システムのシステム設計が必要となります。

企画を実現するには、どのくらいの仕組みが必要で、どのくらいの要員、予算、時間がかかるのか、稼働してからのコストはどうか、こういったことを正しく出していかないと、そのビジネス企画自体が実現しません。システムをどう作るかという要件定義から、システムを作るか作らないか、何を作るかを決めるためのシステム化の方向性の検討が必要となっており、過去事例実績、他事例実績、新技術提案、開発ノウハウなどを踏まえたベンダの提案がおおいに必要とされています。

⑤ プロジェクトマネジメント

システム開発はプログラムを作ること、そのプログラムを作る工程を管理するのがプロジェクトマネジメントです。つまり、プログラムを作り始めてから本番環境に移管するまでです。30年前に IT 担当になったエンジニアは、こういう範囲でプロジェクトマネジメントを学びました。レビューひとつとってみても、プロダクトであるプログラムの品質管理が目的でした。しかし、今日では、システム開発を企画し、開発体制を整え、ステークホルダを含め、その役割と業務分担を明確にし、プロジェクト進捗を可視化、管理制御していかないと、プログラムがどんなに正確に誤りなく完成しても、システムは期待どおり稼働しません。

プログラムを期限どおり、正確に誤りなく作り上げる、これだけを最重要ミッションとして仕事をしてきたシステム担当者が主流であるなか、プロダクトからプロジェクトに視野を広げたシステム開発プロセス管理を実現するには、進捗マネジメントやコストマネジメントは当然のこととして、スコープマネジメント、品質マネジメント、リソースマネジメント、コミュニケーションマネジメント、リスクマネジメント、さらにはこれらを統合

したマネジメントを間違いなくシステム開発に適用していくことが必要です。

　システム開発を行う企業は、ベンダに開発に合ったプロジェクトマネジメントを提案し実施していくことについて、従来以上に力を入れるよう期待しています。能力あるベンダによる高いレベルのプロジェクトマネジメントの実現が、高品質なシステム開発プロセスを実現し、企業が望むビジネスを実現することは間違いないからです。

第2章　ITシステム開発・調達の現場で何が起こっているか

第1章では、ユーザ企業側の経営層や利用部門を中心に述べましたが、本章では、ユーザ企業のITシステム開発・調達の現場および、その実現のパートナであるベンダ企業では、現在いったい何が起こっているのかを示すことを目的とします。第1章と同様に、背景から説き起こし、産業界での取り組み、問題点、どこで、だれが何を決めなければならないのかを説明します。

2.1　背　　景

（1）ネットワークの広がりで見える範囲が拡大

世界中でのインターネット利用者の増大に見られるように、確実にネットワーク社会は広がり、あらゆるところに浸透しつつあり、社会インフラ化しています。

ネットワークの広がりを、固定ネットワークの段階から、やわらかいネットワークの段階へ、さらにユビキタス時代の自在なネットワークによる新しい段階に分けてみると、それぞれに見える範囲が拡大しています。

固定ネットワーク段階は、例えば、サプライチェーンマネジメントでは在庫や製造工程、オーダの変化が見えるようになり、POS（Point Of Sales）では商品の売れ筋、購買層などが見られるようになりました。システム構成もバックオフィス（端末、無停止サーバ、コンピュータなどの構成）か

ら、フロントオフィス（インターネットフロント、Web サーバ、アプリケーションサーバ、データベースサーバ、管理サーバによる構成）へと変化しています。

次に、やわらかいネットワーク段階では、業務の機動性が追求されています。例えば、モバイル（無線 LAN（Local Area Network）、MCA（Multi-channel Access）無線、携帯電話）を利用したシステムのニーズの高まりです。モバイルの即時性で、固定ネットワークの限界を突破する人が増えており、仕事のやり方を変えるようになってきました。

さらに、ユビキタス時代の自在なネットワーク段階では、RFID（Radio Frequency IDentification）、IC カード、携帯などを使って、製品の稼働状況、渋滞状況、生活者の嗜好、モノの動き、人の動きなどが見えるようになり、それをどう経営に活かしていくかが課題となります。それは、マネジメントの範囲の拡大につながっているともいえます。すなわち、経営者は「何を IT で実現したいのか」を明確にしていくことを求められ、「経営と IT の融合」という問題が顕在化し、膨らんでいます。

このようなネットワークの広がりによって、システムの利用者もまた社内のエンドユーザから、社内外のネットワークの先にいる人々までが対象となっていきます。

（2）オープン化と IT システム

メインフレームといわれるコンピュータを使った IT システムと、いわゆるオープン化時代の IT システムとでは、その開発に大きな違いがでています。

メインフレームといわれるコンピュータを使った IT システムでは、専用 OS（Operating System）のもとで、提供元がハードウェアやミドルウェアの組み合わせおよびその整合性をきちんと取っていたので、ソフトウェア開発者は、アプリケーションに集中して開発できました。

しかし、オープン時代になって、ハードウェアやミドルウェアのベスト・オブ・ブリードの組み合せは必ずしも全体最適にならず、かえってシステム開発を複雑にしています。すなわち、システムを構成する要素について、それぞれ個別の機能・価格の上で最適なものを組み合わせたとしても、製品間の相性などの問題のため全体としては必ずしも最適にならないケースも多いのです。本来あるべきソフトウェアの機能、あるいは個別のソフトウェアをつなぐ機能に不足が生じ、この不足する機能をアプリケーションソフトウェアで補完せざるをえなくなり、これがアプリケーション開発費の増加をもたらしました。

さらに、IT システムのインフラ（システム基盤）の設計・構築も、組み合わせ検証・保証を含めて複雑になり、アプリケーションソフトウェア開発とは別の新たな課題を生じさせました。

（3）IT ガバナンスの高まり

あらゆる施策が、システム開発なしには実現しないことや、業務は IT システムに支えられて運用されており、システムが止まると、業務が何もできないことになることを第 1 章で述べました。繰り返しになりますが、システムの停止は、ビジネスの停止につながり、「システムリスクは経営リスク」であり、システム品質の向上は IT ガバナンスの重要課題です。それゆえ、経営課題として、「経営者がシステムに大きく関与していく時代」になっています。

2.2　IT システム開発・調達の現場で起こっていること

（1）産業界での取り組み

ネットワークの広がりとオープン化が進展する中で、IT システム開発や調達の現場で何が起こっているのでしょうか？

IT システムでは、オープン化の時代になって、組み合わせる基本ソフト

ウェア、ミドルウェアは複雑多岐にわたり、また、その上で動作するアプリケーションソフトウェアを見ると開発のやり方、品質管理やプロジェクトマネジメントなど、従来に増してむずかしくなってきました。

そのために、ソフトウェアを中心としたITシステム開発に関連して産業界としての取り組みが必要となり、ライフサイクルプロセス、プロセスアセスメント、品質マネジメントシステム、プロジェクトマネジメント、技法・ツール群の整備などが行われてきました。

図2.1のように、ITシステムでのソフトウェア開発について、ソフトウェア産業界の取り組みをみると、ライフサイクルプロセスとしてはISO/IEC (International Organization for Standardization/the Internatioinal Electro-technical Commision) 12207やそれに基づいた共通フレーム98、プロセスアセスメントとしてはISO/IEC 15504やCMM/CMMI(Capablility Maturity Model/Capability Maturity Model Integration)など、品質マネジメントシステムであればISO 9001、プロジェクトマネジメントであればPMBOK(Project Management Body Of Knowledge)などがあり、工程ごとに技法・ツール群も整備されてきました。従来は、ユーザーがはっきりとした要求を持っていて、正しい要件が明示できることを前提とした作り方中心のソフトウェアエンジニアリングでした。

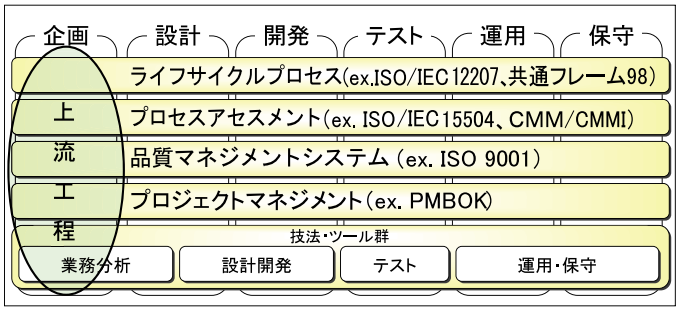


図2.1　産業界でのITシステム開発の取り組み

以降では、そもそも何を作るのか、要件定義に至るまでが対象となる上流工程に注目してみましょう。

（2）事業における業務システム、ITシステムとは

上流工程においての関心は、ステークホルダによって異なりますが、事業そのものをどうするか、あるいは業務をどうしていくか、そのためにITを使ってどのようにシステムを構築し、あるいはソフトウェアを開発していくかでしょう。まず、ここで用いた用語（事業、業務システム、ITシステム、ソフトウェア）の関係を位置づけます。

図2.2に見るように、企業における「事業」のなかに、業務部門などが運用する「業務システム」があります。業務システムの効率化などの要求を満たすために、IT（情報技術を活用した）システムを構築しています。いわゆる従来のコンピュータシステムのことです。さらに、ITシステムは、ハードウェアやソフトウェアから構成されています。

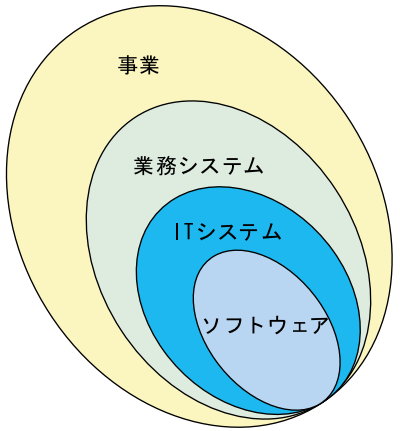


図2.2　ソフトウェアとシステムとの関連

（３）要件定義が正しくないと運用テストで問題が顕在化

図 2.3は（２）で述べた、事業、業務システム、IT システム、ソフトウェアの関係を、開発の流れで見たものです。図の V 字の左側は、仕様を定義することを、右側はそれをテスト・検証することを示し、それぞれが両矢印で示したように「対」になっています。

図 2.3の事業の枠では、システム化の方向性やシステム化計画を練ったものが、事業にとってどのような価値を生み出すか、あるいは生み出したかを評価することを表しています。

業務システムの枠では、計画を業務として運用する上での要件定義や IT システムへの要件定義を行い、できあがったものを利用者や運用者が運用テストで確認することを表しています。

IT システムの枠は、IT システムの仕様がシステムテスト段階でシステム仕様どおりになっているかを検証することになります。同様に、ソフトウェアの枠は、ソフトウェアテストでソフトウェア仕様どおりになっているかを検証している様を概念化して表しています。

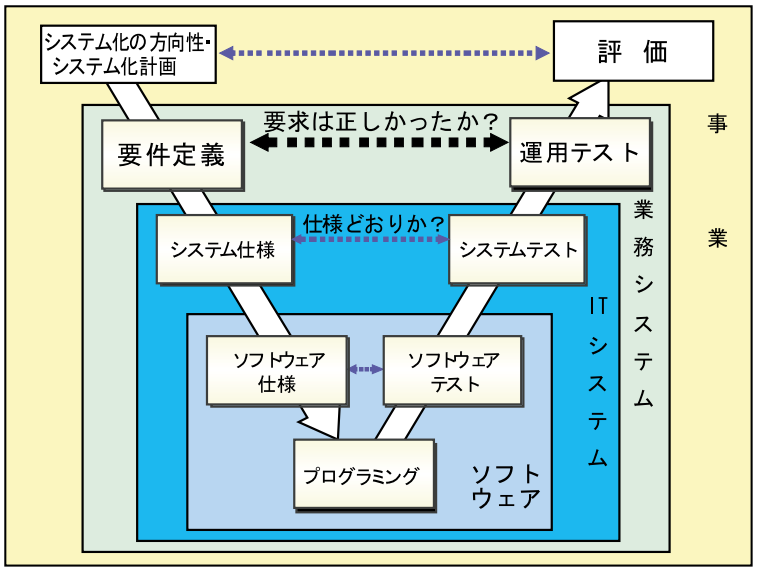


図 2.3　要件定義・仕様とテストの関係

また、事業を含めたおのおのの領域の仕様決定責任を見てみると、事業の領域は事業担当（経営者など）が、業務システムの領域は業務担当が、IT システムの領域は IT 担当(ベンダ企業も含む)が、それぞれ役割を担っているといえるでしょう。

ここで注目したいことは、ソフトウェアや IT システムの開発は、IT 担当によって行われるのに対して、業務システムの運用確認は、開発者ではなく、いわゆるエンドユーザなどの利用者が主担当になることです。

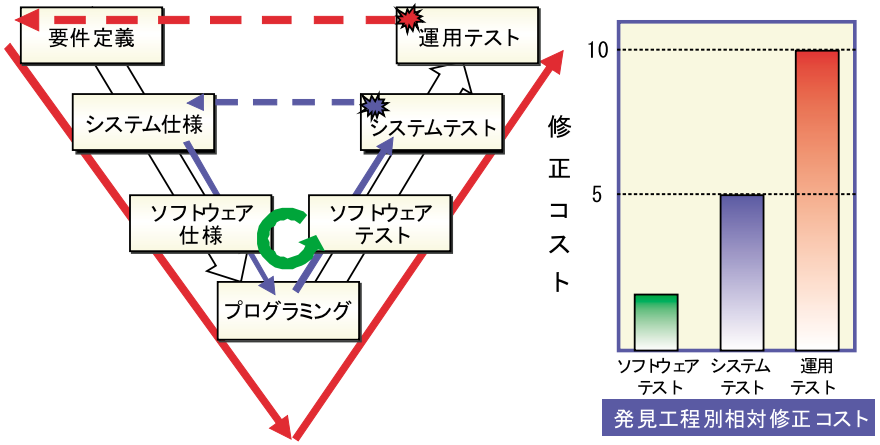
IT システムの開発プロセスに入る前に、サービスの最終利用者であるエンドユーザのニーズが反映され業務仕様がきちんと定義されていないと、開発者が要求仕様どおりに開発・テストしても、運用テストでそれにエンドユーザが気づいて要件定義からのやり直しになりかねません。要件定義自体が正しくなかったからです。

ましてや、システム化の方向性やシステム化計画などの早い段階で出てきたニーズや Wants に基づくシステム開発では、なおさらあやふやなものになってしまうでしょう。

（４）ソフトウェア開発における手戻りとコストの関係

図 2.4をもとに、ソフトウェア開発における手戻りとコストの関係を見てみます。

図の左側は、各テスト段階で発見された誤りの源泉が、その対になった仕様定義にあることを矢印で示したものです。例えば、運用テストで発見された誤りは、要件定義自体の誤りであるために、要件定義作業に戻って再定義されることになり、再定義された仕様に基づいたシステムの見直しになります。システムテストの誤りはシステム仕様に戻り、またソフトウェアテストの誤りは、ソフトウェア仕様に戻ることになります。



出典：B.W.Boehm “Soft. Eng.” IEEE Trans. com (Nov ‘76)

図２.４　手戻りとコストの関係

図２.４の右側のグラフは、誤りが発見された工程別の相対修正コストを示しています。ソフトウェアテスト段階で発見された誤りの修正コストに比べ、システムテスト、運用テスト段階で発見される誤りの修正コストは非常に大きくなることがわかります。

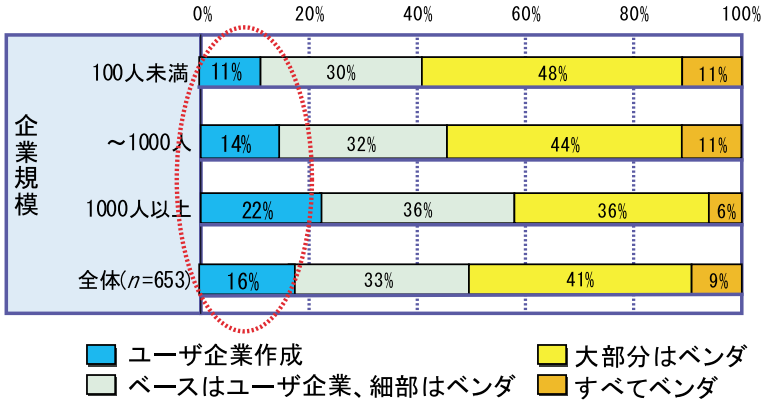
このことから、開発に入る前の要求品質の確保とその課題解決は、ITシステム開発の重要成功要因であるといえます。

（５）要求仕様書は書けなく（書かなく）なった？

では、ITシステム開発の現場で、開発プロセス以前の実態はどうなっているのでしょうか？

図２.５は、日本情報システム・ユーザー協会（JUAS）が平成15年12月に出した、ユーザ企業IT動向調査の中の「要求仕様書の役割分担」についてのデータです。縦軸が企業規模、横軸では要求仕様書を誰が作成するかを示しています。

企業規模によって若干の差はありますが、上から３段目のグラフでは、企業規模が1 000人以上のいわゆる大企業のデータですが、その大企業にお



出典：JUAS 平成15年12月ユーザ企業IT動向調査調査概要報告書

図２.５　要求仕様書の役割分担

いても要求仕様書のユーザ企業作成は22%、ベースはユーザ企業、細部はベンダ作成が36%、大部分ベンダ作成が同じく36%、すべてベンダに作成させる企業が６％というデータが出ています。

ユーザ企業が現在、要求仕様書を書けなく、あるいは書かなくなっていることが分かると思います。この調査では、「システム仕様の定義が不十分のまま発注した」、「発注先に要求仕様条件を明確に提示しなかった」という事項が発注者の反省点の第１位、第２位を占めていることも明らかになりました。

開発に入る前の要求品質が確保されていない状況では、できあがるシステムが満足できるものにならないのは必然といえるでしょう。これはまた、開発担当の努力だけでは開発は成功しないことでもあります。

注）この課題に対して、JUASでは、平成16年４月「エンドユーザによるビジネスシステム定義の進め方～業務システム構築見積照会書の作成と調達～」をまとめました。

（６）ものづくりの流れとソフトウェアエンジニアリングの取り組み

ユーザ企業は、なぜ要求仕様書を書けなくなったのか。その理由を考えてみましょう。

これまで、コンピュータ化といえば、業務規定書に代表されるように、現実の業務をコンピュータに写像してシステム化することでした。

しかし、ネットワークの広がりなどに伴い、いままで見たことがない業務をITで実現する、という場面も増えてきました。未知・未経験の業務やあいまいなWantsは、それを欲する利用者にも、はっきりとは分かっておらず、それをもとに業務要求仕様書を作成することは非常に困難です。また、機器やソフトウェアの構成も非常に複雑になってきており、開発着手時にそれらのすべてが明らかになっているわけではありません。

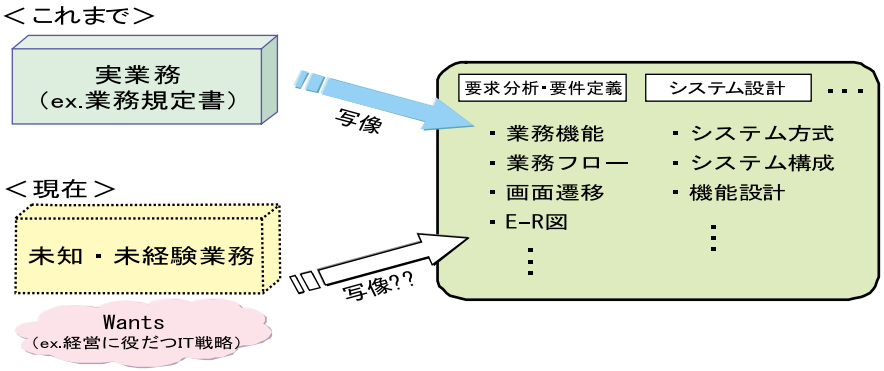


図2.6　ものづくりの流れとソフトウェアエンジニアリングの取り組み

（７）“ITシステムを求める人”と“ものづくりの人”

図2.7のように、経営者やエンドユーザなど、ITシステムを求める人は、経営に役だつ機能、スピード、費用、品質などを求めています。

逆に“ものづくりの人”、すなわち、情報システム部門やベンダなど開発側の人は、システム開発の視点から仕様の固まり具合、ステークホルダの

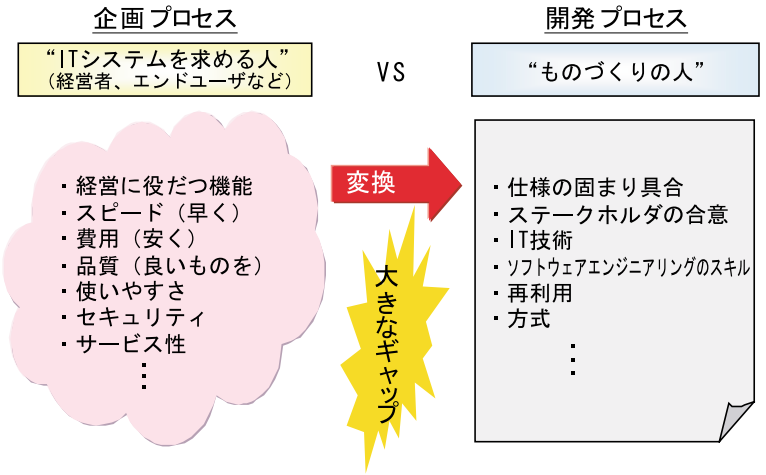


図2.7　“ITシステムを求める人”と“ものづくりの人”

合意、IT技術、ソフトウェアエンジニアリングのスキルなどを求めています。両者の橋渡しをしなければなりませんが、その間には大きなギャップがあり、両者を埋めるためにはいくつかの手順を踏まなければなりません。

（８）ギャップを埋めるための手順の認識

図2.8に見るように、Wantsと開発プロセスの間のギャップを埋めるためには、ステークホルダの要求定義、システム分解や業務分析、業務やシステムの範囲や深さを設定して開発プロセスの入口となる要件定義書とするなど、どの段階で、誰が、何を決めなければならないかを可視化して明らかにすることが大切です。それも、経営者やエンドユーザにも、ソフトウェアやシステム作りを正しく理解できるように、家造り（コラム参照）などになぞらえて、分かりやすく見せる努力が必要です。

例えば、ITシステムを365日24時間運用するという要求は明らかであっても、システムが停止したときを想定した復旧のやり方までは厳密に決められていないことも多いようです。復旧時間の選択肢のなかで、かかるコストと効果との関係で、そこまではしなくてもいいから、セキュリティ対策

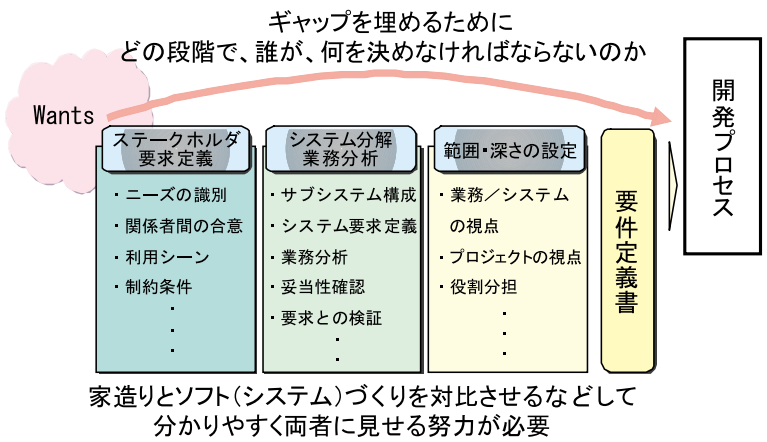


図 2.8　ギャップを埋めるための手順の認識

をむしろ強化したい場合もあります。その意味では、ユーザ企業の情報システム部門やベンダ企業も、いろいろな要求に関して、その他の案件とのバランスで予算内に収めるために、何を優先し、何を削減するかを施主（経営者やエンドユーザ）が判断できるよう、わかりやすいカタログやメニューを作って提示していけるとよいでしょう。こうすれば、システム開発も家造りのようなスタイルになっていくでしょう。

（9）規模が膨張する力学

ニーズや Wants の分解が十分でないと設計段階になって開発規模が膨らみ、それが以降の開発に問題を起こす原因になります。SE（Systems Engineer）の経験則として2-4-2-3の法則（図 2.9）というものがあります。要求の全容が見えない中での見積り規模を 2 とすると、設計段階で明らかになるに従って倍になり、開発時にそれを当初の規模に抑え込もうと努力しますが、結局最後は 3 で終わるというものです。利用者から見れば機能が 1 足りず、開発者から見ればコストが 1 多く、互いに不満が残るというものです。

上流での要求が当初はっきりしていなかったことによる規模の膨張だけではなく、次のような別の力学が存在しています。図 2.10のように、これまで、現業部門（含むエンドユーザ）と開発部門（ユーザ企業の情報システム部門やベンダ）の関係は、現業部門の「やりたいこと」と開発部門の「できること」で考えられていました。経営ニーズは把握するものの、操作性や、現場のクレームを重視した結果、現業部門の「やりたいこと」が最優先されがちでした。その結果、際限なく機能追加と仕様変更が発生して後工程で費用超過してしまい、大きな問題となるケースが発生するのです。

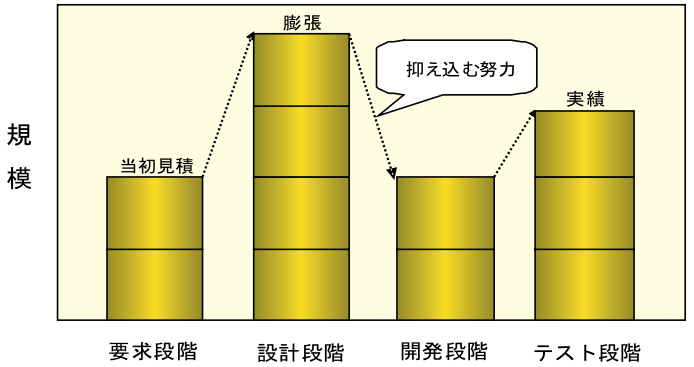


図 2.9　2-4-2-3の法則

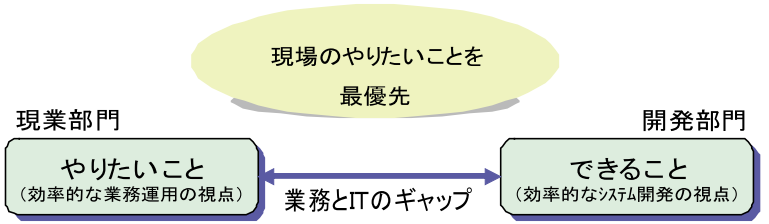


図 2.10　規模が膨張する力学

第３章　要求品質の確保に向けて

本章では、第１章、第２章で述べました問題点の解決策に対応して、開発プロセス共有化の観点から本部会ではどのような取り組みをしているかを示します。特に事業や業務検討の始まりから要件定義までの、「超上流」工程が重要であるとの認識を持ち、ユーザ企業の積極的な参画がプロジェクトを成功に導くと説明しています。

3.1　開発プロセス共有化への取り組みにあたって

システムやソフトウェアの作業手順を可視化して、それに関わるステークホルダや関係者が、共通のことばで語れる取り組みは、ソフトウェアライフサイクルプロセスの国際標準化や日本工業標準、あるいはその拡張である共通フレーム98などで実績があります。

また、ソフトウェアの上位に位置するシステムの作業手順の可視化も進み、システムライフサイクルプロセス（ISO/IEC 15288；JIS X 0170）として標準化されています。

これらのプロセス標準は、意味定義、作業の位置づけ、取引やサービス、あるいはプロセスアセスメントなどを行ううえで不可欠なものです。また、これらのプロセス標準は、企画・開発・運用・保守・管理などに関わるすべての人の基盤で、いろいろな課題解決に役だつものに肉付け・改善していく必要があります。

平成16年度下期から始まった開発プロセス共有化部会では、二度の合宿を含めて、現状抱える問題や、解決案の提示、あるいはその課題に取り組んだ事例紹介など、真剣な討議が進んでいます。それが第１章で述べた「情報化にあたって経営者は何を認識すべきか」であり、第２章の「IT システム開発・調達の現場で何が起こっているか」です。これらのことを部会のメンバが共通認識し、その中で、取り組みにあたってのスタンスとして次のようなことを基本としていこうという合意ができつつあります。

（１）基本スタンス

「問題の本質をしっかりとらえ、実務をベースにした課題解決」をしていくことや、課題解決に共通の悩みを持つなかで、「お互いのノウハウや知恵を共有し、産業界全体を改善」していく思いがあります。

（２）開発に入る前の要求品質確保に焦点

この開発プロセス共有化部会では、これまで述べてきたように、ライフサイクルプロセスのなかでもっとも大切な入口にあたる、超上流での要求仕様をはっきりさせる段階に焦点を当てて、「開発に入る前の要求品質確保」に平成16年度から取り組むことにしました。このため、開発に入る前を「超上流」プロセスと命名しました。図 3.1のように、超上流プロセスは、「システム化の方向性」、「システム化計画」、「要件定義」の三つの段階から構成されています。

（３）ユーザ企業参画の業界ルールづくり

これまで述べてきたように、この課題の解決はベンダ企業だけの努力ではなしえません。システムやソフトウェア開発は、経営者やネットワークの先にいるエンドユーザまでを含めた開発になってきていることから、「ユーザ企業参画での業界ルールづくり」が大切だと認識しています。この部会には JUAS からの参画が多く、期待しています。

システムがおりなすサービスをユーザ企業自身が提供していくわけです

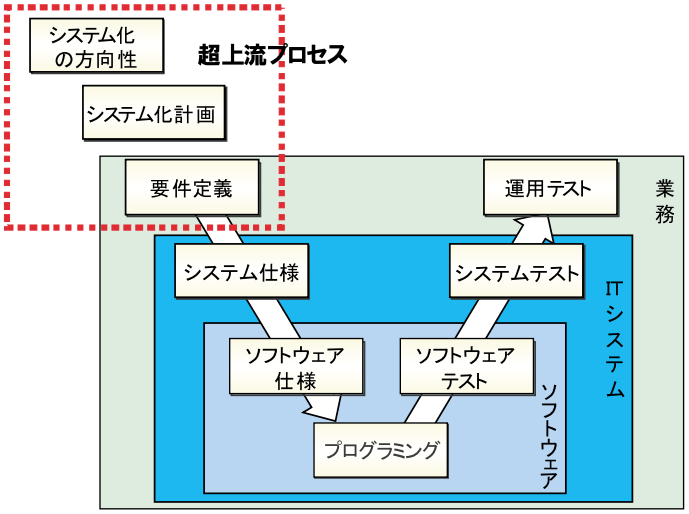


図 3.1　超上流プロセス

から、ユーザ企業の情報システム部門からは、「システムのトラブルをなくすために、もっと自分たちのシステム要件をはっきりさせていきたい」、「情報システム部門・ベンダには開発責任があり、ユーザにはサービス供給責任があることを共通認識していきたい」などという声が出ています。

（４）「経営者の参画」を訴求

前に、システムが止まると業務運用ができないことになり、「システムリスクは経営リスク」といえると述べました。システム品質の向上は IT ガバナンスの重要課題であるために、経営問題として、「経営者がシステムに大きく関与していく時代」になっています。それを反映するために、「経営者の役割と責任分担」を明確にしていくことにしました。

現業部門と開発部門の規模がふくらむ力学の話をしましたが、これを解決するのも経営者の参画です。すなわち、現業部門の「やりたいこと」が、経営戦略実現の視点から本当に「やるべきこと」であるかを検証し絞り込む、経営者の視点と参画が求められます。

この視点は、IT システム開発の軸からシステムで何が得られるかを論じるだけではなく、経営の軸からビジネス上の価値を最大にするには、何をなすべきかを両軸からぶつけ合い、検討し、合意のとれたシステム投資効果（ROI：Return On Investment）を評価するうえでも大切なことだといえます。

さらに、適切なサイズのシステム化の実現は、開発期間や費用の削減にもつながるもので、それはまた品質管理も楽なものに変えていける可能性をもちます。

（５）関係者の役割と責任分担を明らかに

経営者の参画と役割分担を述べてきましたが、これと同じように、ほかのステークホルダ（業務部門、エンドユーザ、情報システム部門、ベンダなど）にも同じようなことがいえるわけで、システムに関係するあらゆる関係者の役割と責任を明らかにしていくことになりました。

（６）どこまでできていると要求仕様は固まったといえるのか

システム開発の超上流、すなわち、ニーズや Wants を要求仕様として定義するまでの一連の主要な活動項目をあげるとともに、その成果として何ができていればいいのかを、ステークホルダのおのおのにわかりやすい形で見せていく必要があります。そのためには、作業項目でのアウトプットあるいは成果（outcomes：その作業が首尾よくいったときにある状態になること）を明示していくことにしました。

（７）システムに対するニーズ（要求）と要件の違いを明らかに

システムに対するニーズ（要求）と要件とは違うものだということをわかりやすくしていくことも部会のテーマです。経営の目標・システムへのニーズや Wants は要求と呼び、システムの開発要求を要件（requirements）と呼ぶようなことです。これらはさらに検討を重ねて業界標準として打ち出していきたいと考えています。

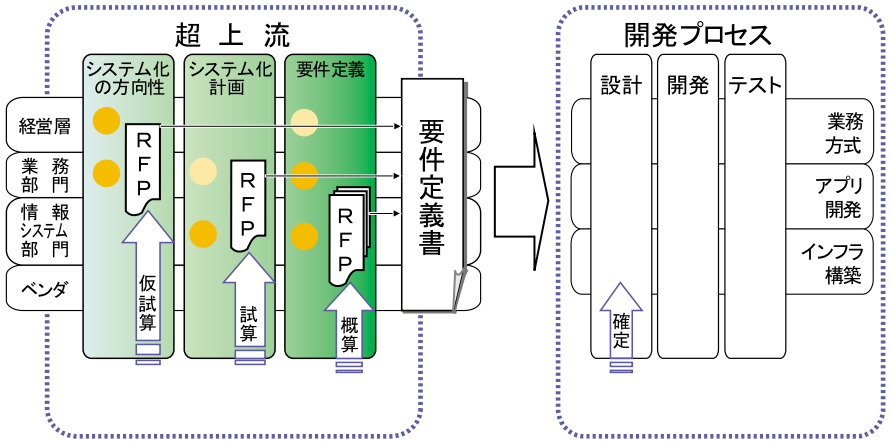
3.2　開発プロセス共有化部会のねらい

開発プロセス共有化部会は、超上流でのプロセスを明らかにすることで、プロジェクトを成功に導くことができることをねらいとしています。そのために、ステークホルダ全員にとって分かりやすく役だつガイドラインを作成し、経営者にも理解してもらえるものを提供することにしています。

まだ、検討を開始してから日が浅いこともあり、すべて合意されている段階ではありませんが、いくつかのコンセンサスをみた部分もあります。それを以下に紹介します。

（１）超上流のプロセス定義と役割・責任をマトリックス化

その一つは、超上流のプロセス定義と役割・責任をマトリックス化し可視化することです。すなわち、図 3.2の左の超上流のプロセスに見るように、時間軸でいえば、システム化の方向性を決める段階、次にシステム化計画を立案する段階、続く要件を定義する段階、この三つのプロセスを念頭におく。その時間軸を横軸にとり、縦軸に、経営層、業務部門、情報システ



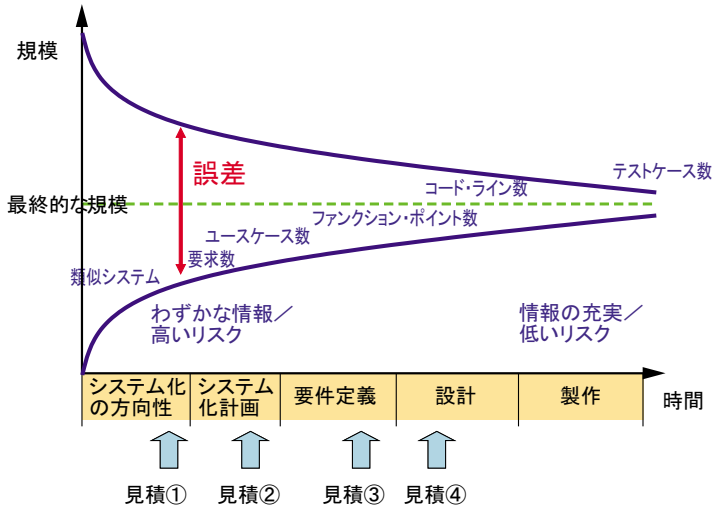
RFP : Request For Proposal

図 3.2　超上流プロセスと役割分担

ム部門、エンドユーザ、ベンダ、パートナなどのステークホルダをとり、その両軸によってできるマトリックスで、役割と責任の主体を位置づけます。例えば、図の中の時間とそれぞれの関係者の交点の濃い丸印が責任主体を表し、また薄い丸印が従たる責任者ということになります。このマトリックスで本来誰が何をどの時点でなすべきかを明確にしていきます。それにはそれぞれが作成すべき具体的なアウトプットを明示すると理解を助けます。

（2）要求の固まり具合と見積りレベルの関係の明確化

超上流や開発プロセスのどの段階での要求仕様かで、その固まり度合いや、見積り対象の深さなどに違いが出てきます。このような超上流や開発プロセスが明らかになると、要求仕様の固まり具合（時間軸、内容＝アウトプット、役割・責任など）と図3.2に示すような見積りレベルとの因果関係などを発注側の経営者にも認識できるものと期待できます。すなわち、



（注）文献：Barry Boehm 著の“Software Engineering Economics（Prentice-Hall 社）”の図に基づき SEC 作成

図 3.3　見積り時期とリスク

超上流での見積り内容は、仮試算、試算、概算レベルだとか、開発プロセスの設計に入って、確定の見積りとなる会話が、日常的に取り交わされることが期待できます。あいまいさがある段階（**図3.3**　見積①～③）での見積りが最後まで開発側（情報システム部門、ベンダ）の束縛になってプロジェクト成功の阻害要因になっている現状から、あいまいさがある段階の見積りを、はっきりした段階で見積り直せるルールづくりなどが産業界にとって新たなプロジェクト成功の鍵となるはずです。

（3）システムにおける非機能要件の重要性の認識

図 3.2の右の開発プロセスにみられる業務方式、アプリケーション開発、インフラ構築の三つのカテゴリについて、少し触れておきます。

業務方式とかアプリケーション開発は、いわばシステムの機能要件（業務を中心として機能）の実現の方法を指したカテゴリです。インフラ構築は、システムの非機能要件（性能、セキュリティ、信頼性、拡張性、運用など）を実現する方法を指したカテゴリです。

超上流の要件定義にいたるまでの要求や要件とは、第 1 章で述べたように決して機能だけを指したものではなく、非機能や既存のシステムの接続やプロジェクト特有の要件も含まれます。

これまで、要求仕様の課題解決のテーマは、むしろ機能要求、すなわち業務機能を明確にするための方法論・手法・表記法、ツールなどが多かったように思います。この開発プロセス共有化部会が、このシステムの非機能要求を避けて論じるわけにはいかないのも事実です。開発プロセスで非機能要求のシステムインフラ構築まで論じることはないとしても、超上流でのシステム仕様には、この非機能要件を検討の俎上にあげておきたいと考えています。

（4）委託した要求仕様の承認と変更への対応

ネットワークの広がりによって、あるいは IT 技術の進展によって、未知・

未経験の業務をシステム化していくことが必要となるなかで、要求仕様作成がユーザ企業が単独でできず、ベンダ企業に作業委託する場合が増えてきたことは前述のとおりです。

その場合でも、要求仕様の確認をベンダまかせにするのではなく、その要求仕様自身を承認（責任）し、それをもとに開発プロセスに入っていくことが、業界のルールに定着することを考えています。

承認された要件が前提となって開発が進み、要件自身の間違いがあった場合は、要求仕様の変更として当初予算とは別の手当てがなされるのがあたりまえとなるような業界ルールです。共通フレーム98のもとになったソフトウェアライフサイクルプロセス国際標準は、この点をはっきり打ち出したものとして知られています。

第4章　超上流工程でやるべきことと役割分担

これまでの章では、システム開発を成功させるために、超上流工程におけるユーザ企業とベンダ企業双方の取り組みが重要であることについて述べてきました。

本章では、ユーザ企業とベンダ企業の役割分担を説明することで、要求から要件を明確にしていく過程や、経営者やシステム部門長が知る必要がある具体的な役割（ロール）や成果物を示していくことを目的としています。そのために、まず、工程の考え方、そして超上流工程（システム化の方向性、システム化計画、要件定義）での役割と成果物の順に説明します。

4.1　工程の考え方

まず本節での説明の範囲を明確にします。工程の考え方は、次の四つの観点で述べられています。一つ目は要件の定義内容と役割（ロール）の関係、二つ目はシステムの開発パターンと今回説明の対象範囲、三つ目はシステム規模の判断、四つ目は要件を確定していくまでの工程図を示します。

これらに共通の考え方は、役割（ロール）に視点をおいてユーザ・ベンダ間およびそれらを取り巻くステークホルダの関係図を模式的に示し、役割の所在と実施時期を分かりやすく述べることです。

ちなみに役割（ロール）とステークホルダの違いを述べますと、役割（ロール）は「何をするか」であり、ユーザであれベンダであれ、誰かが実施し

なければならない行いのことです。それに対して、ステークホルダは、業務やシステムを構築するうえで、関わりあう発言力や影響力を有する関係者のことをいいます。

従来のシステム開発は、業務マニュアルなどによって業務手順が詳細に定義され、システム化する範囲も明確に決まっているケースが多く、情報システム部門やベンダが、業務部門から要望を聞いてつくることができました。しかし、ビジネス環境の変化が早い昨今では、新規事業の立ち上げや関係会社などとの業務連携など、ビジネスのあり方そのものから検討するケースが増えています。

このようなケースでは、システム開発の前に、まずビジネス環境を分析し、業務範囲や業務分掌などを定めたうえで、必要な業務手順を決める必要があります。そして、その役割（ロール）は従来のように情報システム部門やベンダ主体ではなく、業務部門や経営層主体であるべきだといえます。

図 4.1および表 4.1に示すように、システム開発に必要な要件としては、

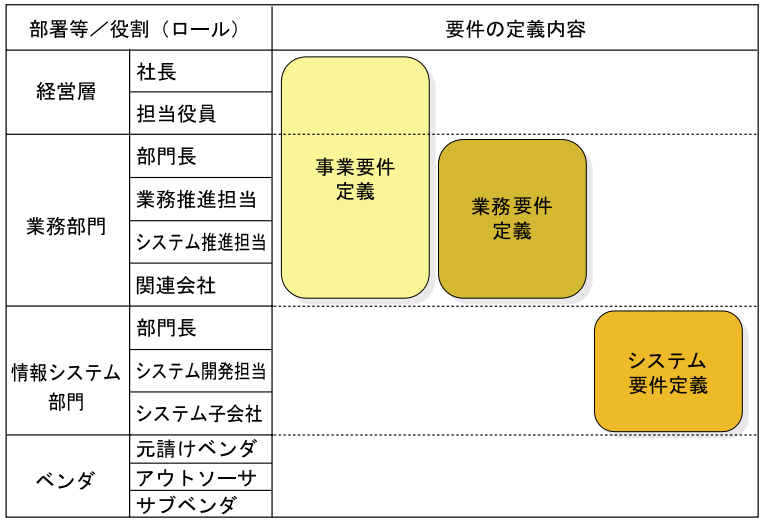


図 4.1　要件の定義と役割（ロール）

表 4.1　要件の定義

名 称	項 目	内 容
事業要件定義	ビジネスモデルの検討 等	新規事業／社外連係／組織改編／部門間業務分掌変更／現行業務踏襲、セキュリティなど
業務要件定義	業務モデルの検討 等	業務内容（手順、責任・権限など）、業務形態（ピークなど）、業務品質、性能目標 、運用、移行要件、セキュリティなど
システム要件定義	システムモデルの検討 等	システム構成、業務アプリケーション（構造、ＤＢ・ファイル構造など）、運用、移行要件、セキュリティ、機密情報保護対策など

事業要件、業務要件、システム要件などがあり、それぞれに対応した役割（ロール）が考えられます。ビジネスのあり方そのものから見直すような場合には、事業要件からシステム要件までを明確にしなければならず、検討と意思決定に携わるべき関係者も多くなります。

これらの要件は機能要件と非機能要件（後述）の二つに大別して分類することもできます。機能要件とはシステムに実装する機能に関する要件であり、非機能要件とは運用要件、移行要件、性能要件、セキュリティ、機密情報保護対策など、機能以外の要件のことをいいます。

本小冊子が対象としているシステム開発は、予算規模にして 5 千万円から 1 億円程度をベースとしていますが10億円クラスの開発にも適用し、社内の要員だけではなく、外部の専門業者を活用するシステム開発を想定しています。

システム開発の方法としては、検討項目やステークホルダが多く、各企業が苦労している「新規開発」などを想定しています（図 4.2）。

システム開発の方法には「スクラッチ開発」のように自社主体で、一から開発する方法のほかにもいろいろあります。ベンダなどが提供する既製の業務パッケージを利用したり、ASP（Application Service Provider）のように社外サービスを利用する場合もあります。また、作ったシステムの保守と運用の設計などは、今回の対象に含まれないものとします。

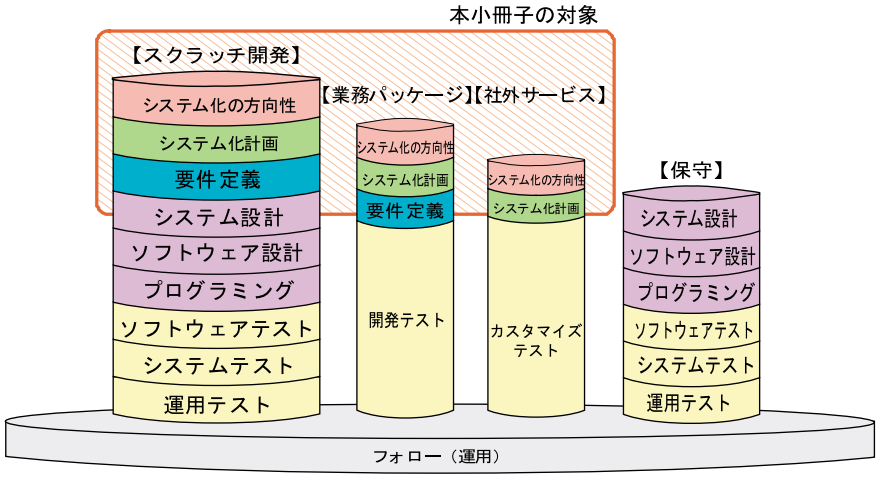


図 4.2　対象としているシステム開発

いずれの開発方法をとる場合も、事前に「システム化の方向性」や「システム化計画」が明確になっている必要があるといえます。

さらに「新規開発」を掘り下げて考えますと、システム開発にかかるコストや期間による違いがあり、この違いはシステムに処理させる業務の手順や判断の複雑さが影響します。業務手順や判断基準が明確になっていない場合では、業務の確定度と複雑さ、ステークホルダの数、役割（ロール）の明確さの度合いなどからシステム開発の規模を想定することができます。

図 4.3ではシステム開発の規模を大、中、小に分類しています。また、現行業務を踏襲するケースとして、既存システムの構成を変更する「設備変更」、効率性や利便性を高めるために、既存プログラムを改修する「改良」、既存プログラムには手を加えないものの、機能を維持するために必要な措置を講ずる「維持」に分類しています。あわせて、それぞれの分類ごとに実施すべき工程と意思決定のタイミングを示しています。

次に、実際に要件が確定していくまでの上流段階においての基本的な工程図を図 4.4、図 4.5に示します。

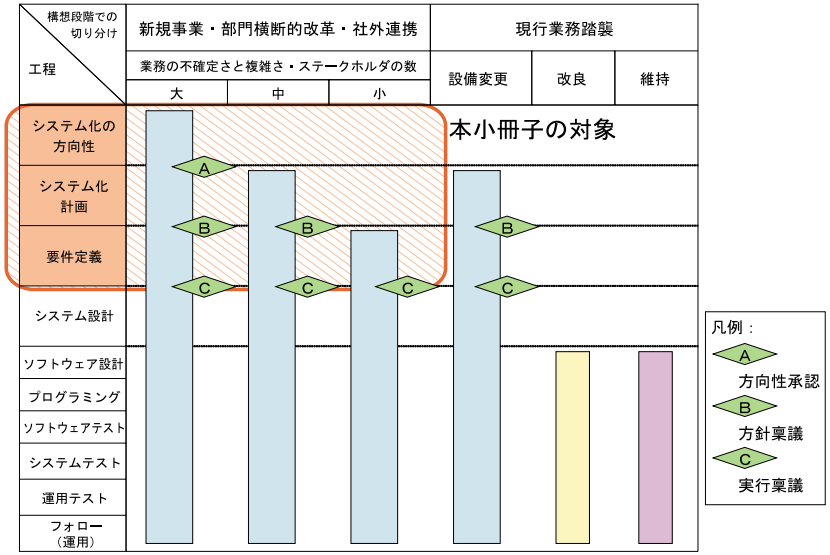


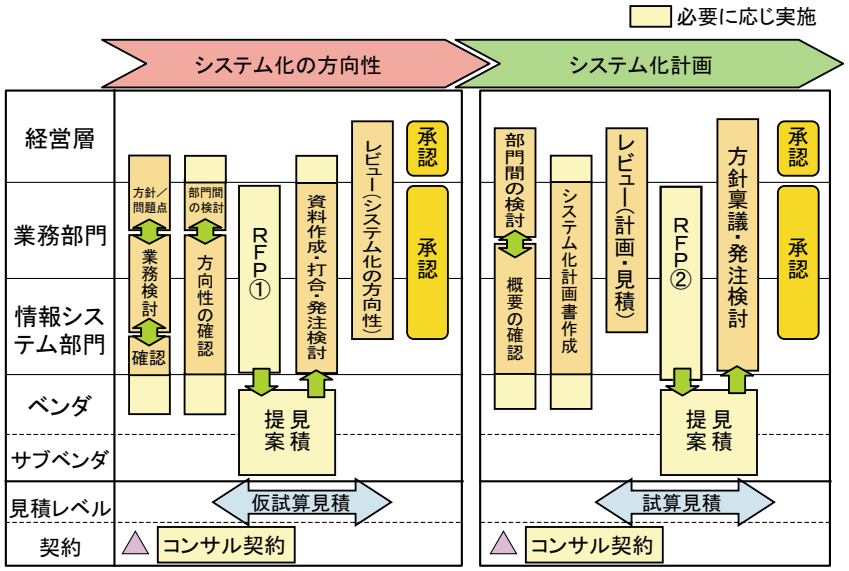
図 4.3　システム化の規模と工程との対応

（1）システム化の方向性

システム化の始まりは前述したように、経営の方針や事業部門からの業務上のニーズあるいはシステムの課題など、いろいろな状況や場からの要求により始まります。当然、各ステークホルダ間での利害関係も異なり、お互いに共通の認識合わせが必要になります。したがって、十分な検討を繰り返し、そのシステム化の方向性をまとめ上げていくことが大切です。

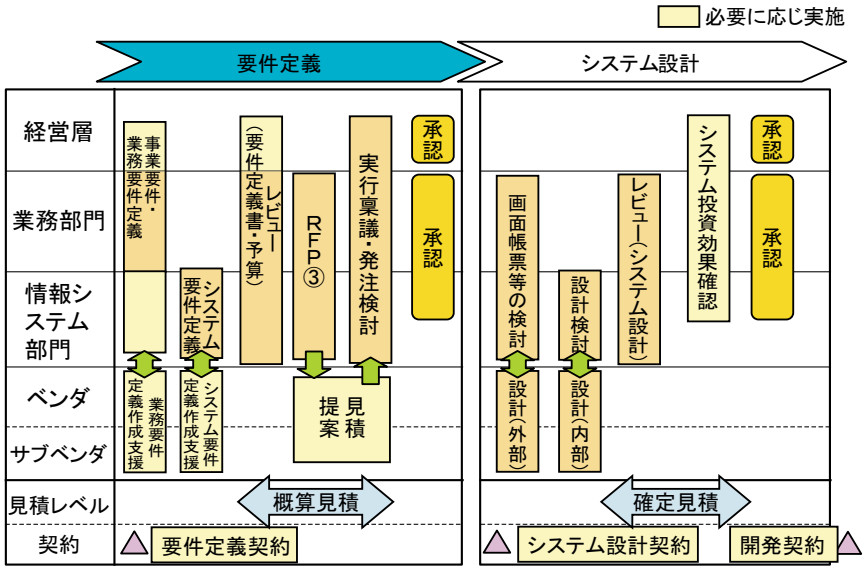
（2）システム化計画

システム化計画はシステム作りの基本です。前の工程で確認したシステム化の方向性を具体的にするために要求分析を実施し、そのための体制、スケジュールの目処をつける計画書を作成します。試算での見積りを実施するのもこの段階です。経営は方針稟議として承認のための判断をしなければなりません。



RFP : Request For Proposal

図 4.4　工程図の例（その 1）



RFP : Request For Proposal

図 4.5　工程図の例（その 2）

（3）要件定義

業務改革やシステム改善を「このようにしたい」という要求の結果を分析して、事業要件、業務要件、システム要件、非機能要件として定義することで、実現方法や機能が要件として明確になり、コンピュータシステムとして構築することが可能になります。つまり、要件定義書は、システムの実現性を担保するための設計書になります。この段階での実行稟議が経営として判断できる最終段階といえます。

（4）システム設計

超上流工程の後工程に位置づけられるシステム設計は、画面や帳票など、インターフェースを決める外部設計と内部ロジックや物理データ構造を決める内部設計に分かれます。情報システム部門はこの設計書により開発工程に進むことができるか？の判断を実施します。図 4.4および図 4.5の工程図は基本形を示したものですので、実際のシステムを構築する上流段階では、この工程図を基準にしながらカスタマイズして利用してください。

4.2　システム化の方向性およびシステム化計画工程での役割と成果物

（1）経営戦略と IT 戦略の乖離

第 1 章でも述べたように、かつて単なる生産性の向上や合理化の推進といった役割が中心だったシステムは、いまや企業のビジネスの中核を成すものとなりました。このビジネスとシステムの関係性の深化に伴って、システムに対する要求は、より複雑化・高度化しています。また、それに応じて IT 投資額も伸長著しいのは、皆さんもご承知のとおりです。

企業価値創造を前提とした経営戦略・事業戦略があり、その実現のために IT 戦略を立案し、それに則ってシステム化を計画する・・・、企業における IT 投資は、そのような筋道を踏んだうえで判断されるべきです。そう

であれば，IT 戦略は経営戦略と整合しているはずです。

ところが、経営層から「IT 投資が効果に結びついて利益を生んでいるのかどうかが見えにくい」といった声が聞かれます。

なぜ、このようなことが起きるのでしょうか？

一見明白な経営戦略・事業戦略からシステム化計画への筋道が、実は想定したとおりに連鎖していなかったのかもしれません。仮に以下の事項が不明確なままシステム導入が行われたとしたらどうなるでしょう。

・期待する事業上の効果とは何か

　売上増、経費削減、顧客満足度の向上など

・IT 投資が生みだす情報システムの効果とは何か

　登録会員数や資料請求数の増大、注文受付から納入までの期間短縮、在庫の減少、書類作成時間の短縮など、事業上の効果に貢献すると想定される結果

・効果の有無を何で測るのか

　具体的に上記の効果を評価するために収集可能な情報で、目標との対比で達成度合を評価できるもの

これらが不明確な場合、IT 投資の効果を実感する以前の問題として、システム開発の目的が不明確だったはずです。評価に困るプロジェクトでは、大なり小なり、これに似た状況にあるのではないかと推察されます。

伝言ゲームでよく見られるように、発信した内容は、発信者から他者へと伝わる過程で、少しずつ変化していきます。これは、他者の意思の介在によって表現が変わったり、別の解釈が入り込んだりするからです。もし、メモ用紙に内容を書いて回したら、何人の手を通ったとしても内容が変わることはありません。

経営戦略や IT 戦略の立案、システム開発に対しても同様のことがいえます。要求の多様化に伴ってシステム開発に関わるステークホルダも増加しています。さまざまな人間の意思、思惑、目論見などのなかで、同じゴールを目指さなければなりません。そこで最も必要となるのは、発信者が意図(＝目的）を明確に伝えること、適切なコミュニケーションにより補完(＝認識）すること、そして伝わる内容を明文化し、ステークホルダ間で共有(＝報知）することではないでしょうか。

以上はトップダウンの場合ですが、このような問題は、ボトムアップの場合にも起こり得ます。

経営層は、業務部門がおかれている状況や、抱えている課題に対する理解が不足しがちです。正しい経営判断を引き出すためには、それを補うための情報をきちんと提示しなければなりません。「大変だから改善したい」ではなく、「この IT 投資の起案は真に会社の利益につながるのだ」ということを定量的、定性的に主張し、理解を取りつける努力が必要です。

一方、経営層は先入観を持つことなく、業務部門の声に耳を傾け、フラットな視点で判断する義務があるといえます。ヒト・モノ・カネという限りあるリソースを、どこに対して投下すべきなのかは、経営層が投資効果を軸に判断する以外にありません。材料不足によるかたよった判断は、当初の目的の達成はおろか、別の弊害を生むことは推して知るべしです。

（２）超上流工程が『かぎ』を握る

システム化の方向性およびシステム化計画工程は、経営層、業務部門、情報システム部門というステークホルダが、きちんとした合意形成を図る場です。それぞれの視点から「何をすべきか」、「どのようにすべきか」を考え、合意の結果、結論を導き出します。それにもまして重要なのは、第１章の事例にもあるように、きちんと役割分担を行い、相互に連携をとりながら、自分たちが実現したいビジネスモデルと実現方法を十分に議論検討すること、そしてオープンな過程を経て、経営層が明確な意思決定を行うとともに、それに対する責任を負うことです。

これによって、投資の背景（必要性）や目的、投資効果が明確になり、より納得感の高い合意形成が得られます。また、業務部門、情報システム部門の担当者は、経営層の明確な意思決定をよりどころに、各論での検討もスムーズに入っていくことができます。

そして、これらの確認は口頭で済まさず、きちんと明文化し、共有しておくことが大切です。口頭のままでは、各人の思惑や認識のずれが発生していても気づくことができません。プロジェクトが進めば進むほど、このずれは致命的になっていくのは言うまでもありません。

プロジェクトの目的の確認は特に重要です。プロジェクトが混乱した場合は、再度ここに立ち戻って判断することになります。また、これが定まっていないと、要件定義の段階で議論が発散したり、最終アウトプットが何を解決することになるのかあいまいになってしまいます。

このように、本工程での議論の深さや、関係者全員の共通認識の有無など、合意形成がいかにしっかりとできているかが、プロジェクトの成功を左右するといっても過言ではありません。それほど、本工程の位置づけは重要なものになっています。

（3）本工程を進めるうえでのポイント

背景や目的の共有が重要であることは先に述べました。ここでは、その後、どのような点に留意して検討を進めていけばよいのかを、重要な七つのポイントとともに紹介します。

ポイント1	最低限やりたいことを考える
ポイント2	システム化ありきで進めない
ポイント3	優先順位を決めてとりかかる
ポイント4	シンプルな業務設計を心がける
ポイント5	死守すべきQCD(Quality Cost Delivery)を決める
ポイント6	プロジェクト体制の穴を見る
ポイント7	責任の所在を明確にしておく

ポイント1　最低限やりたいことを考える

プロジェクトの背景や目的が明確になったら、その目的に応じてシステム化の範囲を決定します。どうせならと、別の範囲を含めたりするケースがありますが、本来の目的を見失う結果につながりますので絶対に避けるべきです。欲張ることなく、最低限やりたいことは何なのかを考えましょう。また、新規ビジネスほど不確定要素が多く、ビジネス環境や事業戦略の大きな変化が予想されます。それに柔軟に対応できるよう、初期投資をおさえて、段階的に大きくしていくことを考えましょう。

既存システムがある場合は、保守の必要性や規模、システム間インターフェースの有無などが気になるところです。ポンチ絵などを用いて範囲を示し、影響箇所をきちんと把握しましょう。

ポイント2　システム化ありきで進めない

次に、プロジェクトの目的達成に向けて取り組むべき課題を明文化します。目的と課題の関係性が分かるように記述することで、目的外の要素が入り込まないようにします。それぞれの課題に対する解決策は、システム化による解決策もそれ以外のものも、できるだけ多く抽出するのがよいでしょう。業務によっては、手作業で行ったほうがメリットがあるものもあります。システム化ありきで考えず、その必要性を冷静に見きわめることが重要です。

システム化による解決策についても複数の実現案を出しておき、メリットとデメリットを勘案しながら、どれにするかを決定します。また、後の工程で議論が再燃しないよう、採用された案には決定理由を明示しておくのがポイントです。

ポイント 3　　優先順位を決めてとりかかる

検討が進むにつれて要件がふくらみ、予算オーバになることもしばしばですが、コストやリソースが青天井でよいプロジェクトはありませんから、何かをあきらめなくてはなりません。そのような事態に備え、取り組むべき課題には、あらかじめ優先順位をつけておくことが大切です。

システム化による解決策（システム化機能）についても、「目的達成のために欠かせない機能」と、「できれば欲しい機能」に分けておきます。これにより、要件をすばやく絞り込むことができるばかりでなく、何をあきらめるかの議論に時間がかかりませんので、プロジェクト運営もスムーズです。したがって、一定の制限の中で最大の成果を上げるためには、早い段階で優先順位を明確にしておくことが重要です。

ポイント 4　　シンプルな業務設計を心がける

周囲からは「業務内容が繁雑」「業務手順が複雑」と評されるのに、現場の業務遂行者には、違和感や課題認識が全くないということがあります。業務内容の経年変化に合わせて最適化しているつもりでも、いつのまにか業務のための業務を作っていたり、無理に既存機能を使うことによるひずみが生じたりするのです。

このような場合は、ともすると複雑な状態のままシステムに置き換えようとするので注意が必要です。システム化は業務を見直す非常にいい機会でもありますので、それまでのやり方に捕らわれることなく、むだな業務や非効率な手順を客観的に評価し、新業務をゼロベースで設計することを心がけることが大切です。

ポイント 5　　死守すべき QCD を決める

実行計画は、プロジェクトマネージャやプロジェクトリーダを中心に策定しますが、事業戦略や事業計画に基づいたプロジェクトであれば、すでに大まかな全体計画があるはずです。それとの整合性を意識しながら実行計画を立案しましょう。

同時に、品質（Q）、費用（C）、納期（D）のうち、死守すべきものはどれなのかを決定します。品質優先であれば、納期を遅らせてでも品質向上に努める可能性があること、納期優先であれば、開発要員の追加投入による費用超過を受け入れることなどをステークホルダ間で確認しましょう。また、QCD それぞれに対して具体的な目標を設定したり、許容範囲を明示したりするのも効果的です。

ポイント 6　　プロジェクト体制の穴を見る

プロジェクトの推進体制を組むときは、まずは必要なメンバの名前を漏れなく書き入れた体制図を作成して、すべてのステークホルダが現れているかどうか、体制的に弱い面はないかなどをチェックします。同時に各担当者の役割、責任範囲、アウトプットと、意思決定者（意思決定機関）も明確にします。

プロジェクトメンバを割り当てる際は、必ず当該メンバの業務状況を確認し、本人にも確実に認識させるようにします。他の業務に時間を取られた場合、期待どおりの働きをしてもらえないばかりか、プロジェクト上のクリティカルパスになることもあります。きちんと人事発令するのも重要な施策のひとつです。

ポイント7　　責任の所在を明確にしておく

どの段階で「誰が、何を、どの範囲で承認するのか」を決めておくことも重要です。主要な検討項目には必ずマイルストーンを設定し、その期日までに完了できなかった場合にどうするか、ということも明確にしておくことが必要です。

ひととおりの検討が終了して要件定義を始められる段階になったら、経営層を交えて検討結果を共有し、要件定義開始の承認を取ります。費用対効果やKPI（Key Perfomance Indicator）の確認に止まらず、システム導入後のどのタイミングで効果測定を行うのか、どのように測定するのか、そして測定結果に対して責任を負うのは誰かを明確にします。当然ながら、プロジェクトを承認した経営層も責任を負わなくてはなりません。

（4）役割分担と成果物

確認したこと、決めたことは、必ずドキュメント化しておきます。図4.1「要件の定義と役割（ロール）」をもとに主体を当てはめてみると、例えば次の表4.2のようになります。

表4.2　役割分担と成果物例

※◎：主担当

名称（大項目） / 名称（小項目）	目　的	内　容	主体
システム化方針の確認			
事業戦略/事業計画	事業戦略（商品戦略）や事業運営計画などを明確にする	事業戦略（商品戦略）、事業方針、売上計画、利益計画、要員計画などを明確にする	業務部門長（◎） 担当役員
中長期システム化戦略	事業戦略、既存システムのライフサイクル、システム技術動向などの視点でシステム化戦略を立案する	事業やシステムの課題、IT動向などをもとにシステム施策を立案し、3年程度のレンジでスケジュールを引く	システム推進担当
単年度システム開発計画	単年度の開発案件を明確にし、並行する案件や事業方針との整合性、優先順位などを把握する	案件名、開発内容、優先順位、希望納期、担当者、代替策、予測工数（超概算）などを一覧化する	システム推進担当
事業システム全体図	既存システムの全体像とそれぞれの機能の関連、業務カバー範囲、保有データなどを把握する	事業全体の既存システムと主要機能の関連、主要DBとデータの流れ、業務のカバー範囲（システム概要）などを分かりやすく図解する	システム開発担当

名称（大項目） / 名称（小項目）	目　的	内　容	主体
プロジェクトの背景・目的の共有			
プロジェクト概要	プロジェクトの背景、必要性、目的、目標、スコープを明確にする	背景、理由、必要性、目的、スコープなどを、必要に応じて図を用い、分かりやすく記述する	業務推進担当
プロジェクトの課題	プロジェクトの目的達成に向けて取り組むべき課題を明確にする	目的からブレイクダウンされた取り組み課題を、前提条件、優先順位とともに記述する	業務推進担当
システム化範囲の選択と集中			
解決策へのアプローチ	取り組むべき課題に対する複数の解決策を検討し、最適案を採択する	解決策を具体的に検討できるレベルまでブレイクダウンし、メリット/デメリット、採択理由とともに記述する	業務推進担当（◎） システム開発担当
システム化機能一覧	採択された解決策のうち、システムによる解決策について、開発内容と優先順位を明確にする	システム開発が必要な解決策をピックアップし、実現に必要な機能と概要を一覧化する	システム開発担当
システムに求められる要素の確認			
業務概要図	プロジェクト対象の業務またはサービスの全体概要とシステム機能との相関関係を明確にする	システム導入のポイントが分かるように、業務やサービスとシステム機能の関連を図で表現する	システム開発担当（◎） 業務推進担当
現状業務フロー/新業務フロー（ラフ）	業務内容や組織形態、役割などが、現状業務と新業務でどのように変化するのかを明確にする	「誰が・いつ・何を・どのようにして」に着目し、現状と新業務フローの対比が可能となるように記述する	業務推進担当
システム全体図	システムによって実現される機能の整合性や他システムとの相関関係を明確にする	システム機能同士の関係、機能と主要DBの関連、他システムとのインターフェース内容などを図解する	システム開発担当
インフラ構成概要	システムが必要とするハードウェアなどの概要や他インフラとの相関関係を明確にする	必要機器、基本／ミドルソフトやネットワークの種類と構成、他システムや他インフラとの関連、ネットワーク接続を図解する	システム開発担当
サービスレベル想定	想定されるユーザ数、処理量、データ容量をもとに、性能目標や運用サービスレベルを確認する	オンラインサービス時間帯、レスポンス、障害対応、稼働率、セキュリティなどへの要望レベルを明記する	システム開発担当（◎） 業務推進担当
他システムへの影響	他システムの保守の必要性、修正方針、規模、金額、スケジュールなどを明確にする	対象システム、保守内容、想定規模（工数/金額）、スケジュール、既存インフラへの影響などを記述する	システム開発担当

名称（大項目）	名称（小項目）	目　的	内　容	主体
	既存システム流用検討結果	パッケージやASPサービスの利用、社内類似システム、インフラの流用を検討したかどうかを確認する	アプリケーションの流用、インフラの流用、データの流用に分類し、検討結果とその理由を記述する	システム開発担当
プロジェクト実行計画の策定				
	プロジェクトのQCD目標	品質（Q）、コスト（C）、納期（D）の目標値と優先順位を確認する（プロジェクト遂行上の判断基準となる）	例えば、品質はバグ出現頻度や重要障害発生数、コストは投資回収期間や投資額、納期はカットオーバー日など、具体的な目標値を設定する	システム開発担当(◎) 業務推進担当
	マススケジュール	開発に必要十分な期間があるか、カットオーバーがビジネス上必要とされる時期と合致しているかを確認する	日程を横軸に作表し、重要会議体などのイベント、マイルストーン、カットオーバー日などを漏れなく、明記する	システム開発担当(◎) 業務推進担当
	開発の特徴	プロジェクトにおいて不可欠な技術や特定のスキル、業務関連知識の深さなど、考慮が必要な特徴を確認する	システムの特徴的な設計方針や機能、複雑なビジネスロジック設計のための深い業務知識などについて記述する	システム開発担当
	検討内容の自己評価	事業戦略や中長期システム化戦略との整合性、検討内容の成熟度合などに対する自己評価を行う	各種戦略と整合性、他システムへの影響、業務変更範囲、サービスレベル想定、システム流用などの検討内容に対する客観的評価を記述する	業務推進担当(◎) システム開発担当
投資効果による評価				
	開発工数見積り	システム開発に関するコストとその見積りの根拠を明確にする	開発工数および金額をフェーズ別、社内/社外別、アプリ/インフラ別に算出し、見積り根拠とともに記述する	システム開発担当
	費用対効果（定量効果予測）	プロジェクトで実現する定量効果とその根拠を明確にする（金額換算可能な効果を金額で設定する）	見込まれる金額換算された効果数字を経年で記述し、算出根拠、算出ロジックも明記する	業務推進担当
	費用対効果（定性効果予測）	プロジェクトで実現する定性効果とその根拠を明確にする（可能な限り定量効果にするのが望ましい）	作業負荷の軽減、作業ミス件数削減など、見込まれる金額換算が困難な効果を記述する	業務推進担当
	費用対効果（収支予測）	プロジェクトの収支予測を経年で行い投資回収期間を明確にする	C/FもしくはP/Lベースでの収支予測を経年（3～5年）で記述し、併せて投資回収期間の予測を行う	業務推進担当

名称（大項目）	名称（小項目）	目　的	内　容	主体
要件定義フェーズの計画と承認				
	要件定義フェーズのQCD目標	要件定義フェーズを推進する上でのQCD目標を明確にし、プロジェクトのQCD目標との整合性を確認する	例えば、品質は設計以降の仕様変更件数、コストは要件定義に掛かる金額、納期は要件定義終了日など、具体的な目標値を設定する	システム開発担当
	詳細スケジュール	作業計画をタスク単位で明確にし、マススケジュールとの整合性、作業に必要な期間などを確認する	タスク、責任者、担当者、作成ドキュメント、作業予定期間、作業工数、作業に関連するイベント（レビューなど）を記述する	システム開発担当
	開発体制図・役割分担	検討を進める上で必要な役割の人材が配置できているか、意思決定者が明確になっているかを確認する	意思決定者（決定機関）、責任者、統括者、担当者、チーム編成、役割、予測工数などを記述する	システム開発担当(◎) 業務推進担当
	マネジメント方法	進捗管理、品質管理、課題管理などの実施方法や、実施に必要な運営ルールを定める	進捗管理、品質管理、課題管理などをどのような方法（会議体、頻度、各種管理表）で行うのかを記述する	システム開発担当
	プロジェクトリスクメモ	未確定事項や技術面での実現リスクなど、プロジェクトの成功を左右する要因とQCDへの影響を確認する	リスク内容、対応策（回避/代替策）、リスク顕在化時のプロジェクトQCDへの影響を記述する	システム開発担当
	取り組みの工夫	プロジェクトの特徴や体制をどのように捉え、どんな工夫や戦略を立てようとしているのかを明確にする	開発の特徴に挙げた内容に対する具体的な取り組み、体制構築上の工夫点、プロジェクト運営方針などを記述する	システム開発担当
	投資決裁書	最終的なシステム化の範囲、経済的費用対効果、各種目標を確認し、要件定義フェーズ開始の承認を得る	投資の目的、システム化の範囲、経済的費用対効果、QCDの優先順位、緊急度、KPI目標などを記述する	業務推進担当(◎) 担当役員

（5）要件定義に向けて

システム化の方向性およびシステム化計画工程で行うべきことと、その必要性について述べてきましたが、いかに重要な工程であるか、ご理解いただけたと思います。

要件定義に踏み込んでいるようにも見えますが、この段階でプロジェクトの目的との整合性や費用対効果を勘案しながら、解決の方向性について

しっかりと議論検討することにより、本当に必要な要件だけに絞り込むことが可能になります。アウトプットはRFP(Request For Proposal)として利用できるのはもちろんのこと、要件定義に引継ぐことで、要件のぶれも極少化できます。併せて見積り精度の向上も大いに期待できるでしょう。

プロジェクトの規模に応じて検討事項や作成すべきドキュメントの範囲は変わりますが、この工程が果たすべき役割はなんら変わらないのだということを、最後にお伝えしておきます。

4.3　要件定義工程での役割と成果物

要件定義工程とは、設計・開発行為に入る前にシステム化のための要件をすべて洗い出し、確定する工程です。一般的には多くの企業が要件定義終了の段階で、確定した要件をもとにシステム化予算を固め、システム設計以降の工程に入る意思決定を行います。つまり、

「要件定義終了時」=「要件の確定時」=「予算の確定時」

であるはずです。

（１）要件と予算が確定しきれない実態

要件定義工程が実施されたのにも関わらず、

・要件が変わる、新たに増える

・システム化予算が増える

のが実態です。

図4.6に示すように、ユーザ企業とベンダ企業では、おのおの異なる「想い」があります。

ユーザ企業は、予算は前倒しで固めたいが、要件の確定は後回しにしたいベンダ企業はその逆の想いを持っています。要件と予算は密接な関係にあるため、両方を同じタイミングで固めるべきものを、「片方だけ後回し」にしたくなるものです。また、要件定義終了時に100%の確定はむずかしいものです。そのため、要件と予算の増加は実は是認されます。

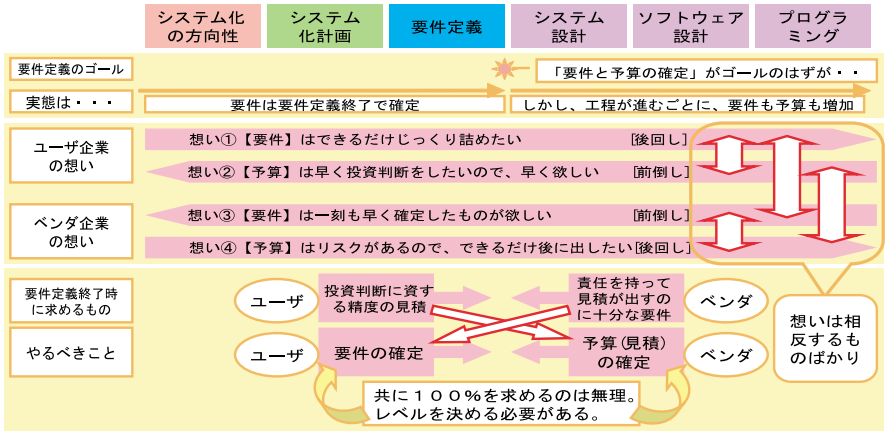


図 4.6　おのおの異なる想い

（２）要件定義工程のゴール

しかし、要件も予算も増えれば、少なからずトラブルになります。ただし、「程度」の問題です。要は、その増える程度がどの「程度」なのかが問題であり、その「程度」の共通認識がありません。

本資料では「程度」を意識し、要件定義のゴールを以下のように定義します。

「要件定義終了時」

=「ユーザ企業が投資判断を行うのに耐えうる精度の予算算定時」

=「ユーザ企業が求める見積り精度に対し、ベンダ企業が責任を持って見積りを出すのに十分な要件が確定する時（ユーザ企業が自ら見積もる場合も同様）」

以降は、このゴールを実現するための役割分担と成果物を定義します。

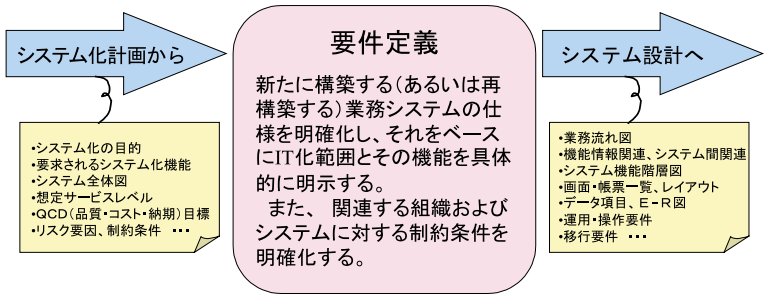
（３）要件定義工程と前後工程の関連

「要件定義」の前工程である「システム化の方向性」、「システム化計画」

では、このシステムがどのような目的で必要とされるのか、そのためにはどのような機能が実現されなければならないか、そしてどのような制約条件があるのかといった、システムに対する大枠の要求事項が、ステークホルダ間で合意されています。逆に考えれば、システムに対する目的や必要性が不明瞭な場合は、「要件定義」に入ることができません。

一方、「要件定義」の後工程となる「システム設計」には、「要件定義」からシステム化すべき機能について、より具体的な情報をインプットとして渡す必要があります。IT 化によって変化する新たなビジネスプロセスを明確にし、実現するシステム化機能の細目を見きわめ、周辺システムとの関係を明らかにし、またシステムの顔となる画面や帳票のイメージを固め、それらを表現するために必要となる情報（データ項目やデータベース）を整理します。また、企業内の IT 化方針や既存システム環境、マシンセンタ運用条件など制約条件も明示しておく必要があります。

これらの情報がそろって、はじめてシステム開発に着手することが可能となります（図 4.7）。



QCD : Quality Cost Delivery

図 4.7　要件定義工程と前後工程の関連

（4）役割分担と成果物

では、要件定義工程では、具体的に誰が何をすればよいのでしょうか？すでに述べたように、要件定義工程のゴールは、「ユーザ企業の投資判断に耐えうる精度の予算算定が可能であること」と、「ベンダ企業が責任を持って見積りを出すのに十分な要件が確定すること」です。このゴールを達成するためには、ユーザ企業とベンダ企業が、ターゲットとなるシステムに関する価値観を共有することが大前提となります。つまり、何が必要なのか、なぜ今それが必要なのかなどをお互いが納得し、協力しあってシステムの「出来姿」を描ききる努力をすることが重要です。この場合、ユーザ企業内では、業務推進担当者とシステム推進担当者の役割分担・責任範囲を明確にし、仕様の最終決定は、ユーザ企業自身が行うという意識づけが必要です。

一方、ベンダ企業側は、ユーザ企業の要件定義のための、専門分野における支援を惜しまない姿勢が求められます。

「要件定義」というと、何かむずかしい作業を想像しがちですが、例えば、上司への作業状況報告となんら変わりません。つまり、５Ｗ１Ｈを明確にすることによって、ほとんどの必要な情報は伝えることができます。

要件定義工程における５Ｗ１Ｈは、以下のように整理できます。

・When ：処理契機・処理間隔・頻度
・Where：処理所属・入出力場所
・Who ：処理主体・権限
・What ：処理データ・インターフェース（画面・帳票など）
・Why ：処理根拠・法令・規約
・How ：処理方法・ロジック・操作・運用

実際のところ、これらをすべて表現できれば、ドキュメントの種類に関して神経質になる必要はありません。数々の事例から、最も効率的、効果的に上記を表現できるドキュメントを「機能要件」と「非機能要件」に大別し、さらに体系的に分類・整理したのが表 4.3です。また、図 4.8にこれらのドキュメント間の関連を示します。

表 4.3　役割分担と成果物例

名称（大項目） / 名称（小項目）	目　的	内　容	主体
機能要件（プロセス）			
機能情報関連図	業務機能間の情報（データ）の流れを明確にする	業務機能間を流れる情報（データ）と向き（方向）を関連図で表現する	業務推進担当
業務流れ図	業務がどのような組織・手段・手順で処理されるかを明確にする	業務処理機能、業務処理担当部署（担当者）、処理手段（手作業・コンピュータ作業）、処理手順（流れ）をフロー図で表現する	業務推進担当
業務処理定義書	業務流れ図上の各業務処理機能の内容を明確にする	業務処理機能ごとの ・インプット ・プロセス（業務処理のルール、手順、内容など） ・アウトプット を HIPO もしくはフローチャートなどで定義する	業務推進担当
システム機能階層図	業務機能を実現する情報システムの機能を明確にする	情報システムの機能を大分類、中分類、小分類の階層で定義する	システム推進担当 システム開発担当
機能要件（データ）			
概念 E-R 図	情報システムにおける概念レベルのデータ構造を明確にする	情報システムを実現するためのデータ構造をエンティティとリレーションシップで表現する	システム推進担当 システム開発担当
データ項目定義書	データ項目の属性を明確にする	エンティティごとにデータ項目名、説明、データ型、けた数、編集方法（コード参照など）を一覧表で定義する	システム推進担当 システム開発担当
機能要件（インターフェース）			
システム間関連図	検討対象のシステムと、既存システムまたは周辺システムとのデータの流れを明確にする	各システム間で受け渡されるすべてのデータの流れをフロー図で表現し明確にする	システム推進担当 システム開発担当
システム間 インターフェース定義書	検討対象のシステムと、既存システムまたは周辺システムとのデータのやり取りを明確にする	各システム間で受け渡されるデータごとに以下を明確にする ・主なデータ項目 ・データ量 ・受け渡し手段、媒体など	システム推進担当 システム開発担当
画面・帳票一覧	検討対象のビジネス機能で必要となる画面・帳票を業務フローごとに洗い出し、画面・帳票一覧として整理し、基本的なビジネスデータの所在を明確にする	画面・帳票ごとに以下を明確にする ・画面・帳票名 ・利用目的 ・利用者 ・様式 ・利用頻度など	業務推進担当 システム推進担当
画面・帳票レイアウト	各画面・帳票のレイアウトサンプルを集め、整理し、基本的なビジネスデータを収集することで、画面・帳票を処理する業務システムの設計条件を明確にする	既存の画面・帳票については、修正の有無をレイアウト上で明確にする 新規の画面・帳票については、レイアウトサンプルを作成することが望ましい	システム推進担当 システム開発担当
非機能要件			
品質要件	システムに対する品質に関する要件	以下の項目（JIS X 0129による）目標値 ・機能性（相互運用性、セキュリティ、標準適合性など） ・信頼性（障害許容性、回復性など） ・使用性（理解性、習得性など） ・効率性（時間効率、資源効率：レスポンスタイム、資源使用量など） ・保守性（解析性、変更性など） ・移植性（環境適応性など）	システム推進担当 システム開発担当
技術要件	ソフトウェアの開発、維持管理、支援および実行のための技術・環境に関連した要件	例えば、以下の項目 ・システム実現方法 ・システム構成 ・システム開発方式（言語等） ・開発基準・標準 ・開発環境	システム推進担当 システム開発担当
その他の要件 運用・操作要件	安定したシステム運用を行うための、検討対象のビジネス機能を実行するシステムについての運用要件と操作要件	例えば、以下の項目 ・システム運用形態 ・システム運用スケジュール ・監視方法・基準 ・SLA（障害復旧時間等） ・災害対応策(DR)、業務継続策(BC) ・保存データ周期・量 ・エンドユーザ操作方法、など	業務推進担当 システム推進担当 システム開発担当

その他の要件	移行要件	現行システムから新システムへの移行対象、移行方法などの移行に関する要件	例えば、以下の項目 ・移行対象業務 ・移行対象プログラム ・移行対象ハードウェア ・移行手順 ・移行時期、など	業務推進担当 システム推進担当 システム開発担当
	付帯作業	システム構築に付帯する作業に関する要件	例えば、以下の項目 ・環境設定 ・端末展開作業 ・エンドユーザ教育 ・運用支援、など	業務推進担当 システム推進担当 システム開発担当
	その他	上記に該当しない要件	例えば、以下の項目 ・コスト、納期の目標値 ・電力量 ・作業環境 ・フロア面積、など	業務推進担当 システム推進担当 システム開発担当

（注）SLA : Service Level Agreement

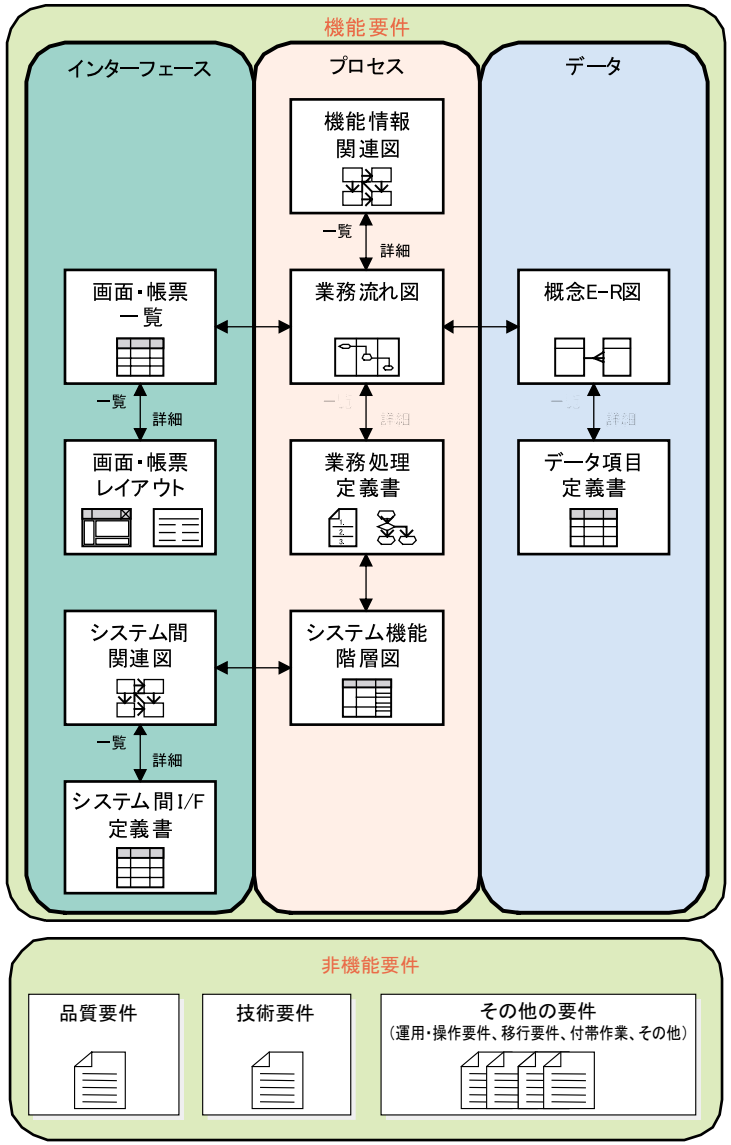


図 4.8　要件定義工程での成果物の相関

（5）解決への提案

① 要件定義におけるユーザの心がまえ

ユーザ企業の実務担当者が、新システムを利用することにより、自分達の業務が、どのように変わるのかをエンドユーザオペレーションやシステムの運用イメージとして、はっきりと描けているか、これが要件定義をうまく進めていくための基本です。このことは逆にいえば、ユーザの想いがあやふやのままでは、いつまでも要件を確定することはできず、ベンダへの想いも届きません。

② ユーザが見落としやすいポイント

画面や帳票などのユーザインターフェースや、ビジネスプロセスを表す業務流れ図などは、ユーザ企業においても重要性を認識され、細部についてよく議論されることと思います。一方、「概念データの整理」や、「他システムとの関連」、また「既存システムからの移行要件」などは、検討が後回しになってしまうことが多いのではないでしょうか。

しかし、要件定義においては、これらも業務流れ図と同様、きっちり要件として洗い出しておく必要があります。これらを軽視して、後工程になって、手痛い設計の見直しや検討の追加工数を発生させないためです。

「概念データ」などは、現業部門にとって、とっつきにくい存在と思いますが、要件定義工程において精度の高い見積りを行うためには、この整理が欠かせません。ここでいう「概念データ」とは、業務上必要なデータという意味です。これらはたいていの場合、画面や帳票上に入力データや出力データとして表現されています。この画面や情報上に散らばって出現する「概念データ」をデータモデルとして整理することにより、例えば、A 画面の「商品名」と B 帳票の「製品名」が同じものを指すのか、あるいは違いを明確化していくための基本情報となるわけです。

③ ユーザ企業、ベンダ企業間の認識のずれ

ユーザが望んでいた仕様が、実はベンダが理解した仕様と異なっている場合があります。最悪の場合は、思い違いのまま設計・開発を終え、テストの段階になって、大幅な設計の見直しや開発費用の積み増しに発展してしまうケースもあります。

このようなケースを防ぐには、ユーザ、ベンダ双方がお互いの理解を深めるため、以下のような配慮が重要です。

【ユーザからベンダへの配慮】

- ユーザはベンダに要件定義完了前から、ベンダが欲しい情報を事前に調査しておく。
- RFP の提出とあわせて、ユーザからベンダに要件の説明会を行う。また、Q&A の機会を設ける。
- RFP はなるべく業界の専門用語は避ける。また、専門用語を使う場合は、注釈や用語集などで、その意味が理解できるようにしておく。

【ベンダからユーザへの配慮】

- ベンダはすべての案件にまんべんなく注意を払い、案件の意味合いを確認する。要件記載の文書がシンプルで短くても、実際には裏にユーザが複雑で手間がかかるようなことを希望していることもある。
- ベンダは、ユーザに提出してほしい情報を事前に準備しておく。ユーザがイメージしやすいように、成果物のイメージや見本を用意する。
- 非機能要件のようにユーザでは情報収集に手間どったり、抜けが発生するような情報については、ベンダ側が誘導的に要件を引き出す。

④ 費用増加への対処方法

要件を確定させることは、要件定義工程においてすべて行うことが理想ですが、どうしても後工程に伸びる要件もあり、また要件の追加なども後工程で発生することが常です。仮りに綿密に要件定義を終えても、企業環境の変化は激しく、それにあわせ IT への要件も変化する可能性があります。

そのため、予算の超過や仕様変更について、事前に取り決めておくことが重要です。

【予算超過時の配慮】

・要件定義終了後に増えてもよい予算の範囲をあらかじめ決めておく。この範囲に収まる要件定義を行う。

・予算が当初予定をどれだけ超過したらプロジェクトを取り止めるかを決めておく。あわせて、消費済みの予算が、その時点時点で把握できる予算の管理をしておく。

【仕様変更、要件追加のルール】

・ユーザ・ベンダ間において、要件の確定ルール（いつまでに確定させるのか）、および、要件の変更ルール（決定プロセス、費用面の扱いなど）を契約段階において決めておく。

【仕様変更や要件追加の際限ない発生への対処】

ある程度の要件追加や仕様変更は、どのようなプロジェクトにも見られますが、際限なくこれらが続く場合は、基本的に要件定義が失敗したことを意味します。したがって、要件定義は個々人の能力に頼るのではなく、組織として確実に行う必要があります。そのためのポイントをいくつか示します。

・要件定義の段階で、ステークホルダとなる社内関係者をすべて巻き込む。また、可能であればベンダも要件定義段階から本格的に参画させる。

・要件定義に入る前に要件定義終了のゴールをあらかじめステークホルダ内で決める。要件定義終了後に増加する案件の許容範囲を決めておく。例えば＋10%など。そのゴールを達成するのに、どれだけ緻密な要件定義を行うべきかを共有化しておく。

・必要に応じ、組織にとって初のチャレンジとなるようなシステムのキーを握るポイントについては、作業を部分的に先取りして実施する。例えば、新しいユーザインターフェースの評価など。事前に具体化・詳細化をしておいたほうが要件の拾い漏れを防ぎやすい。

⑤ 非機能要件・付帯サービス・プロジェクト計画の立案

本小冊子では詳しく述べませんでしたが、インターフェースや処理方法、データなど、システムに対する要望を直接的に表現する機能要件とならんで、それらを実装する環境を定義する非機能要件、システムを実現するにあたっての教育など、なくては成り立たない付帯的なサービスへの配慮も重要な要件定義時のポイントです。

また、要件定義はすでにシステム開発プロジェクトの導入部であり、プロジェクトスタイルで作業をスタートさせている企業も多いことと思います。ここではコスト的影響が大きいにもかかわらず、要件定義時は軽視しやすいポイントを示します。

・性能要件や信頼性要件、セキュリティ要件に厳しい制約がある場合はその条件を明確化しておく。これらによって、ハードウェアや基本ソフトウェア、ミドルウェアの価格は数倍になることもあり、また設計・導入支援費用も大きくなる。

・新しい業務の仕組みの中で、システムをどのように運用するのか、主要機能だけでなく、他システムとの連携、バックアップ、自動運転、セキュリティ管理などの補助・管理機能までを見すえて検討する。

・システム運用（併行）テスト計画、移行計画、教育・展開計画などをスケジュールに盛り込むとともに、可能な限り上流段階で推進組織や費用などを具体化しておく。

このように、第4章では、超上流工程でやるべきことと、役割分担について述べてきました。その詳細は、さらに検討を進め、具体化していきます。

コラム　ー医療とITシステム構築ー

～　医療が教えるITシステム構築成功の鍵　～

（1）イニシアティブを握る発注者

医療現場における治療行為のきっかけには大きく二つのケースがあるように思います。一つは定期健康診断などで特定の病変が発見され、医療側からの提案で始まるケースであり、もう一つが患者側からの愁訴によって始まるケースです。この二つを数で比較しますと後者のほうがきわめて多いのではないでしょうか。

ITシステムの構築においても、ベンダの提言で始まるケースもありますが、発注者側がイニシアティブをとるケースが圧倒的に多いのは同様のことと思います。

（2）発注者が持つ多様性への対応

上記を踏まえますと、医療患者の多様性と同様に、ベンダは発注者の多様性にも十分考慮する必要があります。

医療患者の場合、多くは自己の病状を伝えられる成人でしょうが、ただ泣くだけの小児患者もいますし、愁訴も漠然とした不定愁訴の場合が多いかと思います。

ITシステム構築におき換えますと、ビジネス活動へのIT活用を検討するものの、その方法や手段について不慣れな発注者の存在や、これも多々見られることですが、ベンダへの要求があいまいな場合がこれに該当するように思います。

こういった場合、ベンダには小児科医やカウンセラと同様に、きめ細かいコミュニケーション能力、発注者の立場に立った問題追求能力などが強く要求されることを認識する必要があります。

（3）発注者側の責務

一方で、発注者側も安穏（あんのん）と構えているわけにはいきません。自己の症状を予断偏見なしに医師に伝えることができればよし、それができない場合、患者の振る舞いや表情などから状況を代弁できる付き添いが必要となるでしょう。

ITシステム構築におき換えますと、発注者が自己の目的・目標を論理立てて説明する能力、許容コストの把握能力、ITシステム構築各工程で発注者に要求される事項への対応能力などを保有していることが理想となるわけですが、こういった能力を保有していないと判断される場合には、社内外の専門家をよき理解者・仲介者として利用することが発注者には必要となります。

（4）初期段階での相互理解

最後に、ITシステム構築におけるインフォームドコンセントについて考えてみましょう。

ITシステム構築におけるインフォームドコンセントとは、的確な費用対効果の判断材料を発注者に提供することを目的に、対象となるITシステムへの認識を発注者とベンダが共有、ITシステムの概要明示、システム化に際しての選択肢提示、正確な見積り提示、発注者側にて行わなければならない対応内容の提示などをベンダが行うものではないかと思います。

こういった作業を双方が可能な限り早期に確認・実施しておくことで、目的・目標の実現には不適当な、または過剰なITシステムの構築、ITシステム構築費用のぶれといった問題が改善されるに違いありません。

コラム ー家造りと IT システム構築ー

このコラムでは、家造りと IT システム構築の違いについて次のような観点から考えます。それは概算費用、構築物のイメージ、資格、設計図などです（図 C.1参照）。

家を建てる場合「床面積と坪単価」などを設定すれば、発注者と受注者は早い段階で新しい家の概算費用を共有することができます。このとき、お互いの頭のなかでは予算に対する大きな食い違いは起きません。一方、IT システム構築においては、早い段階での見積りがむずかしいといわれています。

家造りでは、カタログやサンプル、時には実物により、構築に用いる部材が目に見えるため、イメージを共有化しやすいことがあります。一方、IT システム構築では、完成形が目に見えることはまれで、同じ開発要求・要件に対して発注側ではビルを、受注側では一般の住宅をイメージすることも珍しくありません。

また、家造りに携わる人は、一級建築士、測量士など細分化、専門化され、しかも資格を所持しなければ作業自体許されないことも少なくありません。一方、IT システム構築でも、資格に関しては、情報処理技術者試験や各種ベンダ主催の資格がありますが強制力はなく、取得していなくても作業はできますし罰則もありません。これは相手から見れば保証がない、信頼以外に頼るものがないことを意味します。

家造りでは、その工程や内容が可視化できるために、ユーザにとって確認や判断が直感的にできる利点もあります。IT システム構築では、それこそ開発工程の終盤にならなければ確認すらできない状況とは大きな違いです。例えば、変更において、家造りでは、壁紙を貼ったあとに配管や配線に手をいれなければならない場合、変更はむずかしく、また変更してもコストの増加や納期が大幅に遅れることはユーザにも容易に理解できます。一方、システム構築では、ちょっと大げさですが、システムテストや運用テスト後の修正であっても、ワープロで文章を直す程度にしか見えず、困難さが伝わりにくいのです。

家造りでも、要求や要件は、間取り図、設計図をはじめ、段階ごと、相手ごとに数多くのドキュメントを作成します。そのうえこれらは「誰が見ても分かる」ように、決められた内容が適切な粒度（細かさ・深さ）で記述されています。ミリ単位での寸法、部材ひとつひとつの型番まで詳細に記載されています。ところが IT システム構築で必要になるドキュメントは簡単なものであったり、個性的であったりすることが少なくありません。

これでは、正しいシステムができるはずがありません。

文章や図表などいわゆる標準的なドキュメントとして表さなければ相手に伝わりません。口頭では伝言ゲームになり混乱を大きくするだけです。また、暗黙値を用いることも同様です。暗黙値は自分に都合よく解釈されてしまうため、お互いの想いは正しく伝わりません。ドキュメントを作成したとしても、「行間を読む」ことを当然のこととして作成したのでは同じ結果をまねきます。

家造りにおいては、それぞれの立場や観点、進行のタイミングなどなど、それぞれに必要なものが必要なときに存在する。全体の中における相手の役割と自分の責任が明確になっている。スタートからゴールまでの道程での現在位置が共有できる。これら当たり前のことを IT システム構築においても実現することが、ハッピーエンドで終わるために必要なことではないでしょうか。

	家造り	システム構築
歴史	数千年	数十年
発展速度	Evolution（進化）	Revolution（革命）
規格	確立（あらゆる部品） ex．ネジ１本から規格化 使用部材は型番で管理されている。 単価も含めて明確。 一部の作業には資格必要。	ISO/IEC 15504、ISO 9001、SLCP、共通フレーム98、CMM/CMMI、PMBOK など規格化されているが使用は任意で手順レベルは統一されていない。
評価尺度	法規などにより客観性あり。 複合基準、強制力がある。 ex．建築基準法	定量化・共通化されていない。単体（MISP）、申告制（当社比）または顧客満足度など主観的。
要件	受注者が要件を聞き、具体化する。 カタログなどにより細部まで共有化されている。	発注者が具体化するが、十分でない。必ずしも共有化されていない（非共有→失敗へ）。
価格	機能（性能）価格	稼働（人月）価格

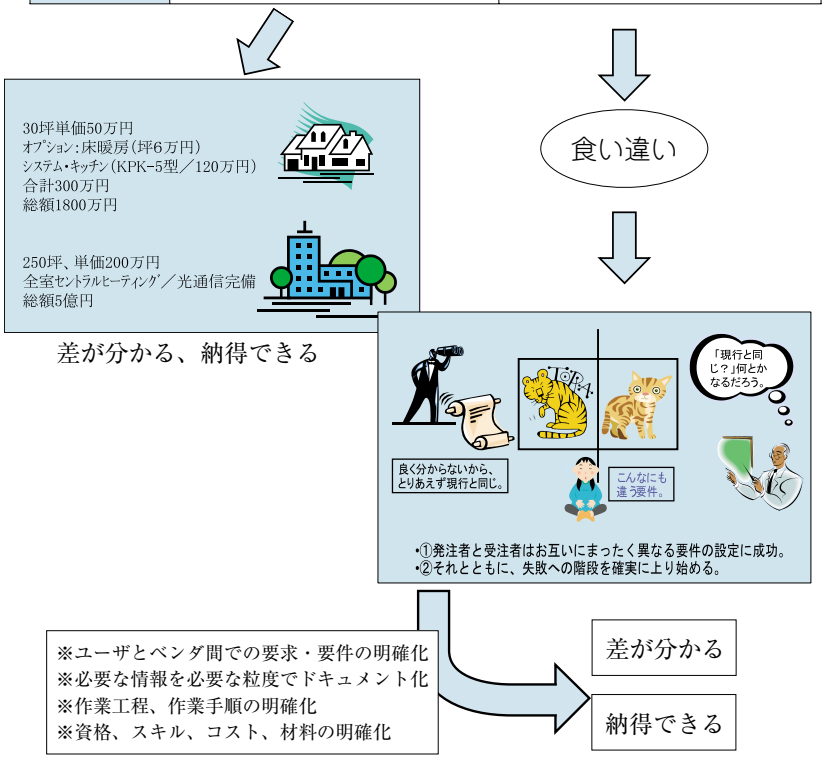


図 C.1　家造りと IT システム構築

第5章　おわりに

本書では、三つの項目

① 経営層はシステム開発責任の当事者

② 役割（ロール）の設定

③ 具体的な対応策

について述べてきました。その中の「経営層はシステム開発責任の当事者」では、経営層がシステム開発と深く関わることにより、要求・要件を明確にさせ、業務およびITシステムに反映させ、事業を推進することで、効果的な経営を達成できることを述べました。

言い換えると、要求品質を確保することが事業を推進する鍵になります。

このことは本書をお読みいただく中で充分にご理解いただけたものと思います。

他の二つの項目については、今後検討を継続していきます。また、「第2版改訂にあたって」でも述べたように、本書の第1版には賛否両論が寄せられております。特に、「もうちょっと具体的な内容を知りたい」、「言っていることは分かるが次に何をするかを示して欲しい」といった意見を多く頂いております。

SECでは、これに応えるため、また、これを広く利用していただけるように「超上流から攻めるIT化の事例検索システム」をWeb上に構築し、また新たに「超上流から攻めるIT化の原理原則17ヶ条」をWeb上に展開しまし

た。

これらの公開は、平成18年度上期を予定しております。詳細に関しましては、SEC の Web サイトをご覧ください。

なお、SEC の Web サイトでは、SEC の事業概要、活動報告、イベント情報、出版情報（＊1）など、さまざまな情報を発信しておりますので、ご活用ください。

（URL : http : //sec.ipa.go.jp/）

（＊1）本書「経営者が参画する要求品質の確保～超上流から攻める IT 化の勘どころ～」を含む SEC BOOKS の情報や SEC journal などの紹介・購入について。

用 語 集

＜あ＞

インフラ：インフラストラクチャを省略した呼称。

インフラストラクチャ：システムに関連する基盤のこと（狭義に解釈）。

エンドユーザ：利用者の項参照。

＜か＞

開発規模：本書で想定している開発規模。

① 開発費用：5 000万円を下限として 1 億円以上。

② 開発工数：100人月以上。

③ ハードウェア費用は実際に開発する状況によって異なる。

④ 対象となる開発工程は実際に開発する状況によって異なる。

（補足）本書での開発規模について

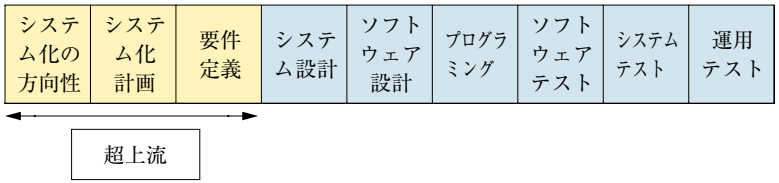
本書では、5 000万円～ 1 億円のシステムをベースとしているが、10億円クラスの開発にも適応可能な開発工程を設定している。これは、ソフトウェア開発において、開発規模の大小という差異はあっても「行うべきことは同じ」であり、省略することはできないからである。しかし、実施するための期間や人数には大きな違いがある。例えば、システム設計まで一人（全ロールを兼任）で行い、以降の作業を外部委託する場合などでは、ここまで（システム化の方向性/システム化計画/要件定義/システム設計）に要する期間は、関係者

が多く複雑な業務を開発する場合に比べて格段に短いことになる。

この手順を踏んだうえで見かけ上工程を省略することは可能である。また、開発規模は、特定の大きさをすべての企業で「○○規模」と規定するのではなく、各企業での規模観に従い、大規模、中規模、小規模として判断することが適切である。

開発工程：開発工程は以下の工程を設定。

事業や業務検討の始まりから要件定義までを「超上流」と呼ぶ。この工程が問題解決の/開発プロジェクトを成功に導くためのキーポイントとなる。



開発範囲：単品、新規での開発を対象とし、開発工程の全工程を対象とする。

「プロジェクトを成功させるには＝役割設定ガイドライン＝（仮）」の対象範囲は、最終的には超上流を含む全工程を対象とする。

本書では、超上流を中心に上流工程を対象としている。

機能要件：利用者の要求を満足するためにソフトウェアが実現しなければならない利用者の業務および手順をあらわす。（JIS X 0135-1：1999の「利用者機能要件」より）

基　盤：広義のインフラストラクチャ。例えば、社会基盤など。

業務要件：担当する業務を遂行するうえでの新規対応や問題点への対策を起因とする追加・改善のための要件。業務手順の変更、性能目標の向上などがある。

業務部門：企業の中で機能別に分けられた内部組織。大抵の場合、階層化されている。

経営層：企業などの経済的活動を、継続的・計画的に運営（遂行）する人または上部の内部組織。

＜さ＞

サービス：ここではITシステムにより提供される用務のこと。

事　業：ビジネスと同意。商業上の取引。

事業要件：新しいビジネスモデルや新規事業を起因とする追加・改善のための要件。情報システム部門（システム要件）はもとより、業務部門（業務要件）、ベンダ（非機能要件）まで、広い範囲に影響する。

システム：ここでは、狭義に定義しコンピュータシステムを指し、ITシステムという表現をしているものと同意。

コンピュータシステムに限定しない場合には、（人間系を含む）業務システムやネットワークシステムなどと表記する。

システム要件：主にコンピュータシステムそのもの（運用を含む）を起因とする追加・改善のための要件。システム構成の変更、DB・ネットワークの変更（HW・SW含む）などがある。

ステークホルダ：利害関係者のこと。

情報システム部門：企業の中の業務部門のひとつ。主に、コンピュータシステムの構築を業務（直接構築を行ったり、間接的な支援や仲介、またはアドバイスを行うなど、いくつかの方法がある）とする。

＜た＞

超上流：開発工程の項参照。

提案要求書・提案依頼書：RFP参照。

＜は＞

非機能要件：機能要件以外のすべての要件。

プロセス：互いに関連をもった作業の集合で、入力を出力に変換するもの。例えば、開発プロセス、運用プロセス、保守プロセスなどがある。

ベンダ企業：ユーザ企業の項参照。

＜や＞

ユーザ企業：本書では、システムを発注する側の企業を指す。これに対し、受注する側の企業は「ベンダ企業」と表す。

要　求：NEEDS。
「ビジネス構築、業務改善を遂行するうえで、ステークホルダが必要と思っている事項」

要　件：REQUIREMENT。
「ビジネス構築、業務改善、システム構築の目的実現のために必要な事項をまとめたもの」

＜ら＞

利用者：エンドユーザと同意であり、システムを実際に利用する人。

ロール：役割と同意。
開発プロセスにおける、責任、承認、実施、支援、確認を担う人の定義。ただし、今回は図 4.1で定めたものを使用。例えば、業務推進担当、システム開発担当、元請けベンダ、運用ベンダなど。
また、表記は「役割（ロール）」とする。

＜英＞

IT ガバナンス：企業が IT を導入・活用するにあたり、目的と戦略を適切に設定し、その効果やリスクを測定・評価して、理想とする情報システムを実現するメカニズムをその企業の中に確立すること。

RFP：Request For Proposal の略。IT システムを導入するにあたって、ユーザ企業が納入を希望するベンダ企業に提供する、導入する IT システムの概要や、調達条件を記述した文書。

RFP ①：システム化の方向性工程での内容・粒度による RFP。

RFP ②：システム化計画工程での内容・粒度による RFP。

RFP ③：要件定義工程での内容・粒度による RFP。

ROI：Return On Investment の略。一般的には投下した資本がどれだけの利益を生んでいるかを測るための指標。システム化の場合、システム化コストに対してどれだけのビジネス上の効果が出ているのかを測る。

超上流から攻めるIT化の原理原則17ヶ条

＜原理原則17ヶ条のねらい＞

1. 産業界の共通認識として
2. プロジェクトを成功に導く羅針盤として
3. IT化における定石として
4. ユーザ、ベンダの役割分担の規範として
5. いつでも立ち戻れる原点として

原理原則17ヶ条の説明

本17ヶ条は、開発プロセス共有化部会において検討してきた、「超上流」フェーズを発注側、受注側の双方がうまく進めるためのポイントをまとめたものです。「経営者が参画する要求品質の確保（第２版）」、「超上流から攻めるIT化の事例検索システム」と密接に関係しますが、本17ヶ条は、重要なポイントを短い言葉でまとめることにより、常に活用していただくことを主眼においています。

重要と考えられる17の原理原則について、基本的な考え方と行動規範にまとめています。

原理原則は、「超上流」において必要とされる事柄を、格言のように短く表現したものです。

基本的な考え方は、原理原則を理解しやすくするため、原理原則の基になる考え方を説明したものです。

行動規範は、原理原則に基づいて、受注者・発注者のそれぞれが具体的にどのように行動すべきかを示したものです。

本17ヶ条は、理解しやすさ、覚えやすさを重視して、重要なポイントを17に絞っていますが、活用する側の状況、特性などにより、17ヶ条以外に必要なポイント（18ヶ条以降）があり得ると考えています。そのようなポイントの追加などにより、それぞれのソフトウェア開発プロジェクトにおいて有用な形で活用されることを、作成者一同、望むものです。

原理原則17ヶ条の利用方法

本17ヶ条は、ソフトウェア開発の現場で広く利用してもらうことを目的としています。その利用（追加／削除などを含めて）は、原則自由とします。

ただし、以下の内容を守ることを前提とします。

（１）17ヶ条は原文をそのまま使用する。

（２）17ヶ条を使用・参照する場合出典を明記する。

・出典：IPA－SEC「超上流から攻めるIT化の原理原則17ヶ条」

（３）17ヶ条の順序の変更は自由とする。

・ただし、使用・参照の出典を明記すると共に本来の順序について明示する。

・出典：IPA－SEC「超上流から攻めるIT化の原理原則17ヶ条（第 n 条）」

・削除などにより17ヶ条の一部ポイントを表記しない場合には、その旨を明示する。

・出典・変更の明記などは、脚注、欄外など、同一ページに明記する。

ただし、全体が17ヶ条関連である場合には巻末などに、まとめて明記してもよい。

原理原則17ヶ条

原理原則[1] ユーザとベンダの想いは相反する
原理原則[2] 取り決めは合意と承認によって成り立つ
原理原則[3] プロジェクトの成否を左右する要件確定の先送りは厳禁である
原理原則[4] ステークホルダ間の合意を得ないまま、次工程に入らない
原理原則[5] 多段階の見積りは双方のリスクを低減する
原理原則[6] システム化実現の費用はソフトウェア開発だけではない
原理原則[7] ライフサイクルコストを重視する
原理原則[8] システム化方針･狙いの周知徹底が成功の鍵となる
原理原則[9] 要件定義は発注者の責任である
原理原則[10] 要件定義書はバイブルであり、事あらばここへ立ち返るもの
原理原則[11] 優れた要件定義書とはシステム開発を精緻にあらわしたもの
原理原則[12] 表現されない要件はシステムとして実現されない
原理原則[13] 数値化されない要件は人によって基準が異なる
原理原則[14] 「今と同じ」という要件定義はありえない
原理原則[15] 要件定義は「使える」業務システムを定義すること
原理原則[16] 機能要求は膨張する。コスト、納期が抑制する
原理原則[17] 要件定義は説明責任を伴う

原理原則[1]

ユーザとベンダの想いは相反する

基本的な考え方

IT システムの企画・開発の現場では、ユーザ企業とベンダ企業の相反する想いがあります。例えば、ユーザ企業は、要件はできるだけじっくり詰めたいし、予算は早期の投資判断を求められるので最終費用を早く確定してほしいとの想いがあります。他方のベンダ企業の想いはまったくその逆です。これがお互いにとってそもそもの不幸の始まりとなります。

「発注内容が固まらないうちに開発作業が開始される」といったことや、その結果としての「赤字案件の発生」といった問題もここに起因しています。

さらに、発注側業務部門も、システム開発のプロセス、システム開発のために必要な情報、例えば、プログラムの作成や変更には、コスト・期間・

ユーザ企業・ベンダ企業の相反する想い

工数が掛かること、品質確保にも時間と労力が必要なことを知らなくてはなりません。

開発規模（工数）に見合った、最低限の工期を確保できなければ顧客満足を満たす開発はできません。受注者には開発規模に見合った工期を主張することが求められます。

行動規範

・発注者・受注者は、お互いの責任、義務、想いを知る。
・発注者は受注者との役割分担を明確にし、プロジェクトに積極的に参画する。
・発注側業務部門も、“システム開発”を理解する。

原理原則[2]
取り決めは合意と承認によって成り立つ

基本的な考え方

全ての取り決めは、承認ルールを明確にして、実施しなければなりません。誰が承認するか、どのように承認するかを明確にするとともに、承認の証跡を残すことです。

証拠のない口約束のように、決まったと了解していることが、それ以降の都合で無責任に変更となり、残念な思いをする、ということはよくあります。

決め事は可能な限り文章に残し、承認ルール（主体と方法）の確認をして、信頼度を高めなければいけません。

承認は合意に基づいていることが必要です。

この場合、受注者は、専門用語や業界用語の多用を避け、発注者が理解しやすく合意を得やすい提案を心がけるべきでしょう。

行動規範

・発注者・受注者は、合意プロセスと承認ルールを明確にし、それに基づいて行動する。
・受注者は、良否判断を仰ぎやすい提案を心がける。

原理原則[3]
プロジェクトの成否を左右する要件確定の先送りは厳禁である

基本的な考え方

要件定義は開発全体にかかる工数で言えば１～２割の作業ですが、開発の成否を左右するという意味で言えば７～８割はここまでの工程に依存します。特に、ユーザ企業の経営層、業務部門の関与はここまでで一旦終わるケースが多く、ここまでで十分に要件が練られていないと問題は運用テストまで発覚しません。

システムの出来を左右する要件に高いリスクを抱えたまま、プロジェクトを進めることは危険です。あせってベンダに開発を依頼しても、先に進めず、かえって時間・コストがムダになることもあります。

解決の目処が立つまでは、先に進まない勇気も必要です。

行動規範

・発注者は、未確定要件の先送りは厳禁であり、現工程を延ばしてでも確定させる。
・受注者は、主要要件の実現の目処がたたないままプロジェクトを進めない。
・発注者・受注者は、未確定要件によるリスクを早期に低減する施策を打つ。

原理原則[4]
ステークホルダ間の合意を得ないまま、次工程に入らない

基本的な考え方

システム開発に関わるステークホルダが増えてきています。

システムの対象となる範囲が広がった結果、システム開発に関係する部門も増え、利用者がマーケットに広がったことで稼働してからのサービス担当もステークホルダに加わりました。

また、企業間接続によるサービスの提供進展により、相手企業のシステム部門もまたステークホルダとして担当営業などとともにコミュニケーションの対象となりました。システムの運用をアウトソースしている場合は、アウトソーサもステークホルダに入るでしょう。

プロジェクトを起こした業務企画担当者は、プロジェクト責任者として、これらステークホルダの方針、意見、課題などについて、漏れなく綿密に把握し、できることとできないことをIT担当者、ベンダとともに切り分け、業務要件として取りまとめていく責任を果たす必要があります。

ステークホルダもまた、システムの供給側に立つ場合は、積極的にシステム開発要件の策定に参加し、利用者ニーズを確実に把握して、正確にシステム機能に反映していくことが必要です。したがって、ステークホルダの果たす役割は、きわめて重要なものになっています。

行動規範

・発注者は、ステークホルダの合意確認を自らの仕事と心得る。
・受注者は、合意を得ないまま開発に入ると、要件定義自体がひっくり返るおそれがあると心得る。
・受注者は、合意確認の作業支援はできるが請負（責任）はできないことを明示する。
・双方は、ステークホルダが誰か、漏れはないかを確認する。

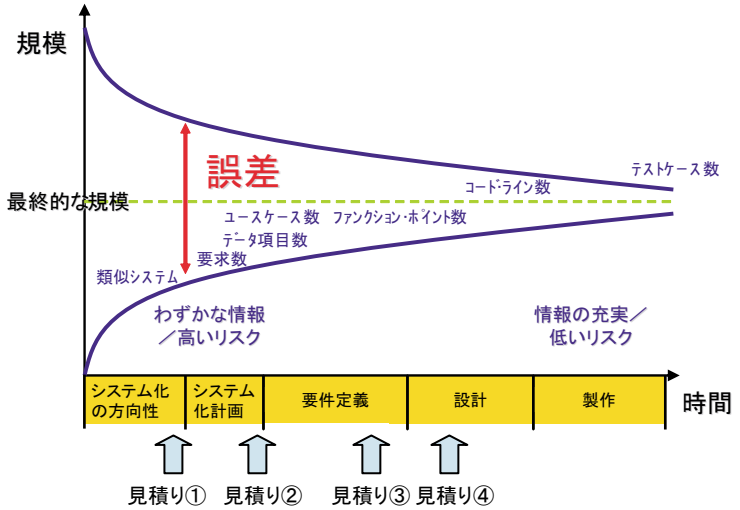
原理原則[5]
多段階の見積りは双方のリスクを低減する

基本的な考え方

開発プロセスのどの段階での要求仕様かによって、その固まり度合いや、見積り対象の深さなどに違いが出てきます。超上流での見積り内容は、仮試算、試算、概算レベルであり、システム設計に入って、確定となります。

にもかかわらず、あいまいさがある段階での見積りが最後まで開発側（情報システム部門、ベンダ）の束縛になってプロジェクト成功の阻害要因になっている現状があります。

不確定要素が多い中での見積りをプロジェクトの目標値として設定すべきではありません。



（注）文献：Barry Boehm 著の“Software Engineering Economics（Prentice-Hall 社）”の図に基づき SEC 作成

見積り時期とリスク

あいまいさがある段階の見積りを、はっきりした段階で見積り直せるルールづくりなどがプロジェクト成功の鍵となります。

要件の不確定さやプロジェクトの特性・リスクに応じて、適切な契約方式（多段階契約、インセンティブ付契約など）を選択することにより、発注者・受注者の双方にメリットが生まれます。

また、非機能要件に対する見積り技術が確立していないため、発注側が一方的に要求を提示しても、要件定義段階では、受注側で保証できないものもあります。

多段階とは、受注先をその都度変えるということではなく、固まり具合に応じて見積り精度をあげていこうということです。

行動規範

・発注者は、事業リスクを低減するためにも、多段階見積りを活用する。
・受注者は、見積りリスク回避のため多段階契約を活用する。
・受注者は、要件定義段階では非機能要件に保証できないものがあることを説明する。

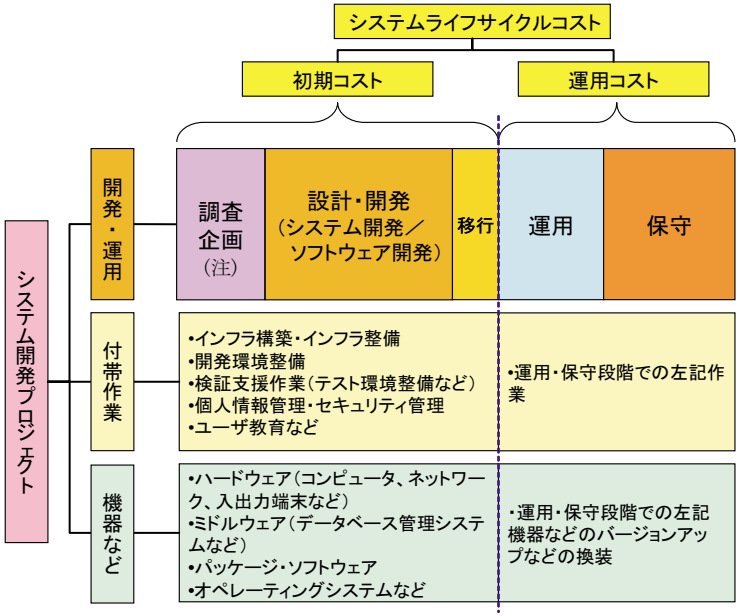
原理原則[6]
システム化実現の費用はソフトウェア開発だけではない

基本的な考え方

見積り範囲がソフトウェア開発のことだけを指しているのか、インフラ整備（システム基盤整備）などのような付帯作業も対象にしているかなど、スコープを明確にしていくことが大切です。

特に、教育や旧システムからの移行にかかる費用などは見落とさないようにしましょう。

また、連携する周辺システムとのインタフェースのための費用や、新システムの変更内容をユーザに周知してもらうための費用なども考慮しておく必要があります。



（注）システム化の方向性から要件定義

システム開発プロジェクトの構成要素

発注者は、何をお願いし、何を自分で行うのか、一方、受注者は自分の提供する作業やサービスはどの範囲なのかをお互いに明確にしておくことが重要です。

開発から運用・保守への引き継ぎをスムーズに行うことがシステムの安定稼働には重要です。そのため、要件定義の段階から、システム稼働後の運用・保守を見据えた計画・体制作りを行うことが必要となります。

行動規範

・発注者は、依頼する範囲、内容を漏れなく洗い出し、提示する。
・受注者は、見積りに含まれる内容と根拠を明確化する。
・発注者は、運用・保守も見据えた計画・体制を作る。

原理原則[7]
ライフサイクルコストを重視する

基本的な考え方

開発コスト、運用・保守コストのバランスを考えなければなりません。大切なことはライフサイクルコストを意識することです。

業務パッケージを採用する場合は、カスタマイズを前提とすることは避けましょう。カスタマイズ費用の予想外な増加を招いたり、パッケージのバージョンアップ時にそのまま適用できないなどの問題が発生する可能性があります。

プロセス主導、データ主導、オブジェクト指向には、それぞれに適した業務があります。対象業務と扱う情報を分析し、それに合った開発技法を適用すべきです。

ミドルウェアも含めた製品の採用にあたっては、実績、信頼性を十分評価し、保守性・運用性を高めることも考慮して、採用すべきです。

例えば、運用性・保守性を高めるポイントとして以下があります。

－メンテナンスフリー
－拡張性の容易さ確保
－モニタリング・トレーサビリティの確保
－障害発生時の調査、リカバリーが容易な設計
－OS・ハードウェアのバージョンアップ対応

行動規範

・発注者は、システムのライフサイクルにわたって投資対効果(ROI)を算定する。
・発注者は、業務パッケージを採用する場合は、カスタマイズを前提としない。
・受注者は、対象システムの特性をよく見きわめて開発技法・環境・ツールを適用する。
・受注者は、運用性・保守性を高める提案をする。

原理原則[8]
システム化の方針・狙いの周知徹底が成功の鍵となる

基本的な考え方

情報システムを入手しようと思う者は、その目的を明確にしなければなりません。さらに、システム化の方向性を示すとともに、関係者で共有しておくことが肝要です。

超上流のフェーズで、システム化の方針・狙いを浸透させておかないと、各人が勝手気ままに要件を考えるため、仕様の統一に時間がかかり、最初の構築だけでなく、その後の維持・保守においても費用と時間が増大することになります。

システム化の目的はコンピュータやプログラムではなく、事業目標を達成するための情報システムの構築なのです。

行動規範

・発注者は、情報システム構築の目的を明確にする。
・発注者は、情報システム構築の方針・狙いをステークホルダに周知徹底する。
・受注者は、方針・狙いを理解して、情報システムを構築する。

原理原則[9]
要件定義は発注者の責任である

基本的な考え方

要件定義とは、どのようなシステム、何ができるシステムを作りたいのかを定義することです。それはあくまでも発注者の仕事であり、発注者の責任で行うものです。要件定義があいまいであったり、検討不足のまま、受注者に開発を依頼した場合、その結果として、コスト増、納期遅れ、品質低下を発生させるおそれがあります。その責任を受注者に負わせることはできません。

要件定義作業は発注者の業務部門と IT 部門が二人三脚で進めます。また発注者によっては、人的資源、経験、スキルなどの問題で、独自で実施できない場合もあります。このような場合、受注者をうまく活用し、不足しているシステム知識を補うことが有効であり、受注者に一部委託し、支援を受けることもあります。その上で受注者は発注者の側に立った支援を提供します。ただし、受注者が支援する場合であっても、要件定義で作成した成果物に対する責任は発注者にあります。

行動規範

・発注者は、「我々が要件を決め、責任を持つ」という意識を社内に浸透させる。
・発注者は、業務部門と IT 部門が、二人三脚で要件定義を進める。
・発注者は、要件定義段階で受注者をうまく活用する。
・受注者は、発注者の側に立った支援を提供する。

原理原則[10]
要件定義書はバイブルであり、事あらばここへ立ち返るもの

基本的な考え方

ベンダ企業を含むステークホルダ間の合意のベースとなるのは常に要件定義書です。設計工程以降よりも、むしろ、要件定義の合意形成時点での吟味が重要です。「決定先送り型」の要件定義では、あいまいな海図に基づく航海のようなもので、早晩プロジェクトが破綻します。

ステークホルダ間の合意は、名目的な合意ではなく、実質的な合意であることが不可欠です。そのかわり、一旦きちんと決めれば、それに沿った運用をすればよく、最初に苦労するだけの価値はあります。

行動規範

・発注者は、安易に変更できない「重み」を認識して要件定義書を提示する。
・受注者は、安易に回避できない「責任」を認識して要件定義書を受託する。
・受注者・発注者とも、以降の変更はすべて要件定義書をベースとして議論する。

原理原則[11]
優れた要件定義書とはシステム開発を精緻にあらわしたもの

基本的な考え方

要件定義工程では、業務要件を整理・把握し、その実現のためのシステム機能要件をしっかり固めます。あわせて性能、信頼性、セキュリティ、移行・運用方法などの非機能要件、既存システム接続要件、プロジェクト特有の制約条件も洗い出します。また、将来の方針を見込んで稼働環境を定めることが大切です。流行に流されず、ルールを定めることです。

業務担当部門と IT 部門とが協力し合って、決めるべきことをきちんと決めることです。

それがシステム開発工程以降のコスト超過を最小限に抑えるとともに、開発工期の確約、要求品質の確保にもつながります。

結果として、システム開発の契約は基本設計、開発と多段階になるとしても、発注者としては、要件定義後にシステム総費用を把握し予算化するため、すべてを漏れなく洗い出す必要があります。

行動規範

・発注者は、機能要件、非機能要件などを漏れなく洗い出す。
・受注者は、特に非機能要件の定義で専門家としての支援をする。
・双方の協力で、システム開発の総費用を固める。

原理原則[12]
表現されない要件はシステムとして実現されない

基本的な考え方

この原則は、建築における施工主と工事業者の関係にあるように、発注と受注における常識です。しかし、情報システム開発においては往々にしてこの原則が成立しない場合があり、「行間を読め」、「言わなくても常識」、「言った言わない」など表現されない要件が、両者のトラブルの原因になります。

行動規範

・発注者は、文書・モックアップなどの手段を講じて、要件を表現しつくす努力をする。
・受注者は、行間を読むのではなく、きっちり確認をとって進める。

原理原則[13]
数値化されない要件は人によって基準が異なる

基本的な考え方

要件定義では、定量化できるものは、極力、数値化します。数えられないものは定義できません。「大きい、小さい、速い」だけでは、人によって「ものさし」が異なります。

例えば、障害発生時の復旧については、「すぐに」、「速やかに」といった表現は避け、想定障害を明記した上で「５秒以内に復旧」、とか「１分以内」、「翌日オンライン開始まで」ということを、双方が協力して定義します。

また、数値化されていても誤りはあります。例えば、使用する単位が違えば結果は大きく変わります。単位まで含めて確認し、決めなければなりません。

行動規範

・発注者・受注者は、協力して、定量化できる要件は極力数値化する。

原理原則[14]
「今と同じ」という要件定義はありえない

基本的な考え方

「今のシステムと同じでよい」という要件定義は、トラブルの元です。

「同じ」という言い方が正しく伝わるのは、具体的なプログラム、コード体系、テーブルなどそのとき存在する個別の形を持ったものについてです。実現機能レベルで同じと言う言葉を乱発しないようにしたいものです。

「今と同じ」でも要件定義は必要です。そもそも同じでよいなら再構築する必要はありません。よくないから再構築するというところから発想したいものです。

現行システムの調査をする場合は、システムの機能を洗い上げ、新システムの実像を明確にするだけでは不十分です。現行システムをどう使っているか、という点から調査をしなければなりません。例えば、データの再利用、アウトプットの二次加工、客先提供などの使われ方について調べて把握しないと、新システムの機能は不十分なものになってしまう可能性があります。

受注者は、発注者の「今と同じ」という要件を、そのまま受け入れてはいけません。

「そもそも今の要件はどうなっているのか」を問い直し、場合によっては具体的な要件にまで導くことも必要です。

行動規範

・発注者は、現行システムと同じ機能の実現であっても、要件定義を実施する。
・発注者は、既存機能だけを見て要件とするのではなく、使われ方まで十分調査し、要件とする。
・受注者は、「今と同じ」要件を具体要件まで問い直す。

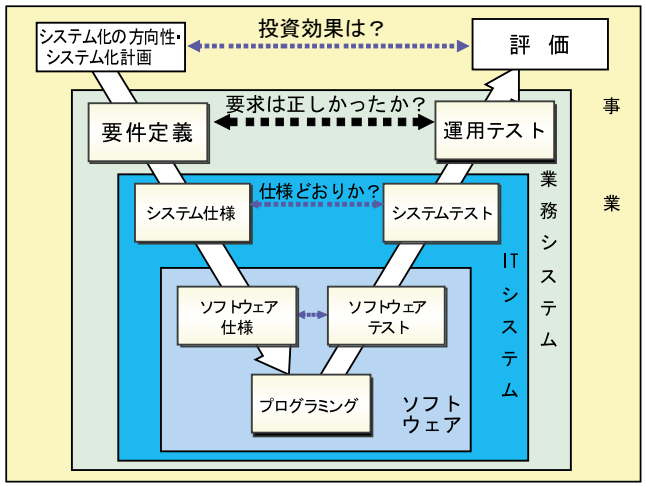
原理原則[15]
要件定義は「使える」業務システムを定義すること

基本的な考え方

要件定義は、業務にとって「使える」、「役に立つ」、「運用できる」システムを定義することです。

発注者は、それまでのやり方にとらわれることなく、むだな業務や非効率な手順を客観的に評価し、新業務をゼロベースで再設計することが大切です。ともすると、業務の複雑な部分を複雑なままシステムに置き換えようとするので、そうならないように注意しなければなりません。

また、ビジネスニーズからシステムの実現機能に落とし込んだ後、その機能が本来のビジネス要求を満たしているものか、立ち戻って検証することが重要です。IT 化が自己目的化して、何のために実現したかったのかを



要件定義・仕様とテストの関係

見失うこともよくあります。これに運用テスト段階で気が付くのでは悲劇です。業務要件があいまいであると判明した場合には、常に業務部門と調整し、システムの合目的性を高いレベルに保つことが必要です。

そして、定義した新たな業務、新たなシステムが運用できるのかどうかの検証も重要となってきます。

要件定義の場に参加して、議論が横道にそれたり、枝葉末節に陥らないように助言するのは受注者の役割です。また、受注者は、要件として定義したものが、システム化計画で想定したコストや期間と比べて過剰なものや、逆にあまりに多くの費用を要さずとも実現可能な要件は勇気を持って変更を進言しなくてはなりません。

行動規範

- 発注者は、常にビジネス要求の視点から、システム要件の妥当性を検証する。
- 発注者は、シンプルな業務設計を心がける。
- 発注者は、運用要件を要件定義の中で定義する。
- 受注者は、オーバースペックを是正し、コストショートを進言する。

原理原則[16]
機能要求は膨張する。コスト、納期が抑制する

基本的な考え方

限られた「期間」と「コスト」の中で必要な「機能を実現」して、初めて評価されるということを忘れてはいけません。システムの開発においては、実現する機能とコストと納期のバランスが重要ですが、このバランスを保つのは非常に困難なことでもあります。プロジェクトメンバーはともするとシステム開発に没頭し、本来同時に達成すべきコストと納期をおろそかにしがちです。自分の家を建てるときには予算や引っ越しの時期との折り合いをつけるのに、システム開発では自分の懐が痛まないので、どうしても、機能＞コスト・納期の関係になりがちです。また、多くの場合、システム開発のコストは実現する機能ではなく、工数に比例しますから、どのくらいの作業が残っているのかをきちんと把握しながら、機能との折り合いをつけて作業を進める必要があります。このバランス感覚をプロジェクトメンバー全員が持っていなければ意味がありません。

新規ビジネスほど不確定要素が多く、ビジネス環境や事業戦略の大きな変化が予想されるため、システムも初期投資をおさえて、段階的に大きくしていくことを考えなければなりません。プロジェクトの背景や目的に応じたシステム化の範囲を検討し、「ついでにこの範囲も」という考え方は本来の目的を見失うので絶対に避けましょう。

要件の検討は、工程の進度に応じ、自由に発想する段階と、現実的にMUST要件に絞り込む段階を使い分ける必要があり、要件がある程度の粒度に詳細化された段階で、優先順位付け、コスト評価・リスク評価を行い、予算や期間などプロジェクトの制約の中で絞り込む必要があります。

納期が守れないと単にペナルティなどの経済的損失だけでは済まず、社会的信用すら失われることもあります。その場合は、段階的システム稼働の提案など、確実に実現できる機能に絞り込むことも必要となります。

行動規範

・発注者は、必要最低限のシステム構築からスタートする。
・発注者は、要求を抽出する段階と、要件として絞り込む段階を分ける。
・発注者は、要件の優先順位付けをする。
・受注者は、納期限界を超える開発量と判断したら、段階的稼働を提案する。

原理原則[17]

要件定義は説明責任を伴う

基本的な考え方

システム開発における万全なる準備は、正確な要件という情報の次工程に向けての伝達です。自分が次工程に伝える必要のある情報について、要件確定責任だけでなく説明責任を負う必要があります。

システム開発の受託側から見た原則は「受託した要件として、書いてあるものは実現させる。書かれていないものは作らない。」ことです。システムは決めたとおりに作られ、決めたことに理解の誤りがない限り、正しい結果を生み出します。もちろん、プロジェクトのスタート地点で、すべてを誤りなく責任をもって確定することはできません。決め事も、それに基づいてのシステム構築も、人間である限り、見込み違い、思い込み、決め付け、聞き違い、聞き漏れはなくなりません。システム構築は、そういったことにより発生した「誤り、漏れ」を解消していく過程ともいえます。

「要件の行間を読め」ということを要求してはいけません。基本的には当たりまえの前提や例外処理であっても漏れなく伝達する必要があります。

また、同じ言葉を聞いても頭に浮かぶものが異なるのが人間です。発注者、受注者双方が説明責任を果たすことが、多様化した要求と、複雑化したシステム開発において品質を確保する重要なポイントとなることは間違いありません。

行動規範

・発注者は、受注者に要件を正しく説明する。
・受注者は、要件を理解して、理解した内容を発注者に確認する。

○**執筆者**（敬称略）

開発プロセス共有化部会

主　　査：	**村上　憲稔**	富士通株式会社
副 主 査：	**菊島　靖弘**	株式会社アイネス（東京海上日動火災株式会社）
委　　員：	**荒生　知之**	株式会社野村総合研究所
	石川　貞裕	株式会社日立製作所
	岩見　好博	オリンパス株式会社
	太田　進一	東京電力株式会社
	小野原泰光	東京ガス株式会社
	尾股　達也	社団法人情報サービス産業協会(JISA)
	加藤　光明	株式会社 CRC ソリューションズ
	角田　千晴	社団法人日本情報システム・ユーザー協会(JUAS)
	寺田　尚弘	清水建設株式会社
	内藤　裕史	日本アイ・ビー・エム株式会社
	芳賀　達夫	みずほコーポレート銀行
	橋本　恵二	東京国際大学
	端山　　毅	株式会社 NTT データ
	福田二三雄／深瀬光聡	新日鉄ソリューションズ株式会社
	森下　哲成	株式会社リクルート
	山崎　剛／野村伸介	日本電気株式会社
	若杉　賢治	富士通株式会社
	石谷　　靖	ソフトウェア･エンジニアリング･センター（株式会社三菱総合研究所）
	小林陽二郎	ソフトウェア･エンジニアリング･センター（株式会社 CSK システムズ）
	新谷　勝利	ソフトウェア･エンジニアリング･センター
	室谷　　隆	ソフトウェア･エンジニアリング･センター(TIS 株式会社)
	安田　　守	ソフトウェア･エンジニアリング･センター（株式会社野村総合研究所）
	祝谷　和宏/山田　圭吾/風間　博之/坂本　教晃/上原　智	経済産業省

CD-ROM について

※：本 CD-ROM 上のファイルは、CD-ROM 上での利用を前提に作成しています。

1：解説付き CD-ROM の構成

- ￥本文．pdf　：解説へのリンクを埋込んだ本書(＊1)の PDF ファイル
- ￥comment.html：解説表示用 html ファイル(＊2)
- ￥img　：画像ファイル
- ￥conf　：解説表示用 html ファイル設定
- ￥readme.html　：本文書

（＊1)「経営者が参画する要求品質の確保
～超上流から攻める IT 化の勘どころ～（第２版)」

（＊2）本書の pdf 版ファイルより読み込まれる解説用ファイル。

2：解説付き CD-ROM の使い方

① CD-ROM 内の本書の pdf 版ファイルを開く。
② CD-ROM 内の本書の pdf 版ファイルの本文中にある該当箇所(＊3)をクリックする。
③該当する解説用ファイルが別ウィンドウで開かれる。

（＊3）該当箇所：本書中に青下線で示された用語または文章。

3：解説付き CD-ROM の稼働環境

・動作確認は、Acrobat Reader(Ver.7.0.7 Windows 版)、Internet Explorer(Ver.6 SP 1)、Windows XP SP 2 で行いました。お使いにあたっては同等環境でのご利用をお勧めします。PDF 制作には、Acrobat Professional 7 を使用しました。
・ハードディスク等にコピーする場合は、お使いのシステムで動作のできるパスに気をつけて、配置してください。Web サイトに配置する場合も同様です。
　なお、パスにカナ文字、漢字、半角スペース等が含まれていると、解説表示用 html ファイルの呼び出しエラーになる場合があります。

4：その他

本 CD-ROM をご利用されたことによるすべての損害については、独立行政法人情報処理推進機構、編者・著作者・著作権者、株式会社オーム社はその責めを負いません。

5：CD-ROM 内提供資料の保護について

Web 事例検索システム、本書(＊1)、解説付き CD-ROM の資料保護のために、以下の文言を記載する。

（1）公開資料の利用規定（利用者への制限）

SEC を通じて提供した資料を利用者が通常のソフトウェア開発において使用することに制約はない。
資料は、販売するなど直接営利目的で使用しない。
資料の利用者が資料から派生した二次的資料について他者（特に資料提供者およびその関係者）の使用を制限することは禁止する。
資料は、国内での使用（二次使用を含む）のみが許される。ただし、海外のベンダ企業へのソフトウェア開発の発注等において資料として利用することができる。
JUAS 作成の資料は、出所を明記して利用する。

（2）成果物に記載する資料保護文

（本来は、全ての文書に個別に明記することが望ましいが、運用を考慮し使用時に明示するという方法を用いることとして、上記のそれぞれに、少なくとも 1 箇所以上記載する。）

「本ソフトウェア開発で利用された資料のうち、SEC 提供にかかる資料は、SEC が著作権を有するものである。したがって、SEC 提供資料を利用した著作物にはその旨を明記すること。」

以上

※Adobe Reader は下記 URL より入手（ダウンロード）できます。
（URL : http : //www.adobe.com/jp/）

編 者 紹 介

独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター

2004年10月に独立行政法人 情報処理推進機構（IPA）内に設立されたソフトウェア・エンジニアリング・センター（SEC）は、エンタプライズ系ソフトウェアと組込みソフトウェアの開発力強化に取り組むとともに、その成果を実践・検証するための実践ソフトウェア開発プロジェクトを産学官の枠組みを越えて展開している。

［所在地］ 〒113-6591 東京都文京区本駒込2-28-8
文京グリーンコート センターオフィス
電話 03-5978-7543，FAX 03-5978-7517
http://sec.ipa.go.jp/index.php

SEC BOOKS
経営者が参画する要求品質の確保 第2版
～超上流から攻めるIT化の勘どころ～

平成17年4月25日 第1版第1刷発行
平成18年5月25日 第2版第1刷発行

編 者 独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター
発行者 佐 藤 政 次
発行所 株式会社 オーム 社
郵便番号 101-8460
東京都千代田区神田錦町3-1
電 話 03(3233)0641(代表)
URL http://www.ohmsha.co.jp/

印刷・製本 報光社
ISBN 4-274-50076-4 Printed in Japan