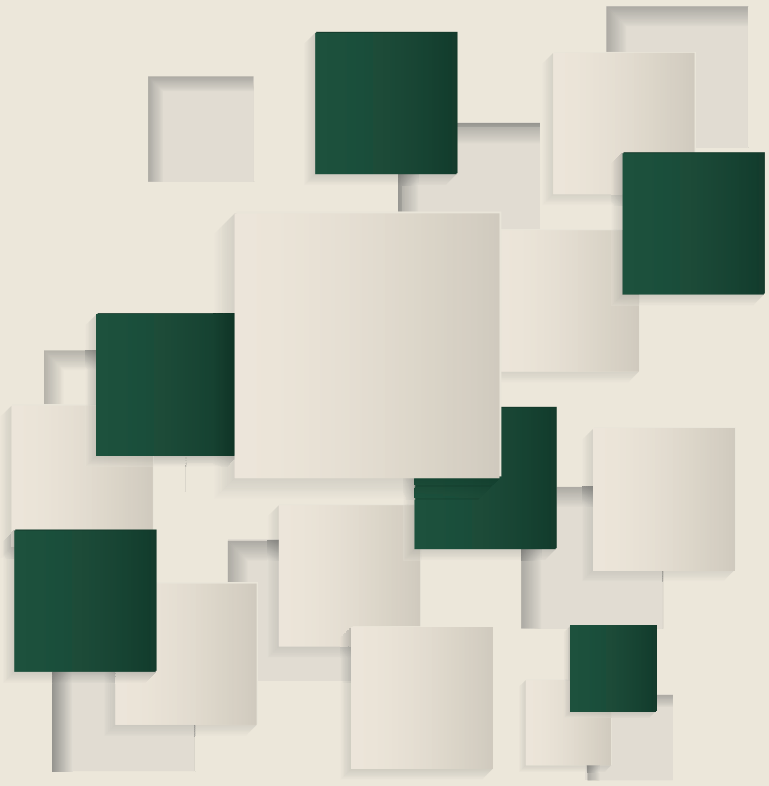
SEC BOOKS

ユーザのための要件定義ガイド

～要求を明確にするための勘どころ～

独立行政法人情報処理推進機構
技術本部 ソフトウェア高信頼化センター　編



ユーザのための要件定義ガイド

～要求を明確にするための勘どころ～

独立行政法人情報処理推進機構（ＩＰＡ）
技術本部 ソフトウェア高信頼化センター（ＳＥＣ）

目次

本書の内容に関して

- 本書を発行するにあたって、内容に誤りのないようできる限りの注意を払いましたが、本書の内容を適用した結果生じたこと、また、適用できなかった結果について、著者、発行人は一切の責任を負いませんので、ご了承ください。
- 本書の一部あるいは全部について、著者、発行人の許諾を得ずに無断で転載、複写複製、電子データ化することは禁じられています。
- 本書に記載した情報に関する正誤や追加情報がある場合は、IPA/SEC のウェブサイトに掲載します。下記の URL をご参照ください。

独立行政法人情報処理推進機構（IPA）技術本部
ソフトウェア高信頼化センター（SEC）
https://www.ipa.go.jp/sec

商標

- 本書に記載する会社名、製品名などは、それぞれの会社の商標もしくは登録商標です。
- 本書の文中においては、これらの表記において商標登録表示、その他の商標表示を省略しています。

はじめに

はじめに

独立行政法人情報処理推進機構　技術本部 ソフトウェア高信頼化センター（以下、IPA/SEC）は 2000 年代に「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）[1]を発行し、IT システムの役割が現場中心からビジネス中心へ、個別最適から全体最適へ、と変化する時代に、経営層が IT システム開発の上流に深く関わる重要性を発信した。以来 10 年が過ぎたが、その間ずっと重要性が指摘されてきたにも関わらず、上流工程の作業不備に起因した開発プロジェクトの失敗や運用後のシステムトラブルは無くなっていない。むしろ、IT システムの大規模化・複雑化が進み、一部ではトラブルの発生数が増大している。今後、システムが予想もしなかった相手とつながり、要求が多様化して複雑に深化すると、一つの不備が広範囲に影響し、社会に与えるインパクトは以前とは比較にならないほど甚大となる。そこで、「複数のステークホルダから要求（What）を抽出し、見極め、要件として定義する」という、上流工程において歴然と変わらないシステム開発における問題への取り組みが大切になる。

ここで、IT システムの変遷に伴う課題について少し考察してみたい。

当初業務支援のために使われてきた IT システムは、その技術自身の進展と時代の要請から企業の基幹事業や社会インフラなど、その利用が質、量ともに拡大してきた。紙や口頭でのやりとりを IT システムに置き換える導入時期から、社内事務を効率化する活用時期には品質が重視されていたが、自社の売上に直結するビジネス戦略上の要として利用されてくると、競争優位性を確保するためにスピードが重視されるシステムも増えてくる。最近では新たなビジネスを創出し、企業価値を向上する経営戦略上の武器と考えられるようになった。

IT システムは、図 1 が示すように基幹業務に必要なデータの転送・蓄積・処理を中心とする「守りの分野」と、顧客とのつながりによる協働を重視し、IoT やビッグデータ、AI などの技術を活用する「攻めの分野」に大別され、それらの連携が重要になる。

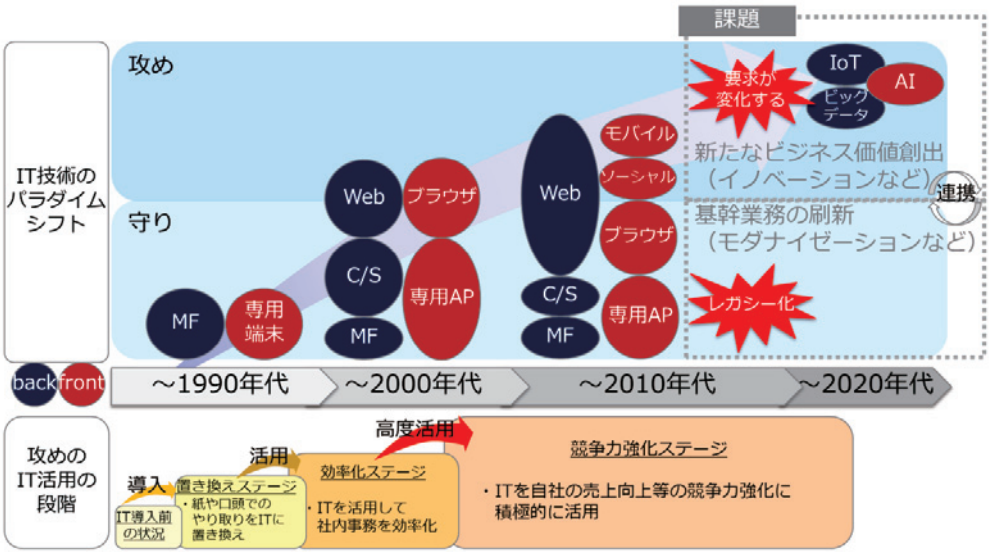


図 1　IT システムに関わる IT 技術とユーザ企業の現在

IT 技術に関する進歩は目覚ましく、今後さらに加速することは明らかである。新たなビジネス価値を創出するための攻めの分野と、基幹業務を確実に遂行する守りの分野を区別し、それぞれに強化することが経営に直結する課題となった。

しかし、攻めの分野においては求められる要求の全てが開発初期に分からないことが多く、IT システムにはサービス開始後に徐々に明らかになっていく要求への対応が常に求められる。変化する要求を要件として的確にシステム化し、迅速にサービスとして市場へ提供し続けることが課題である。一方で、守るべき基幹システムなどは、長期間の運用により、関連する人の減少や資産のレガシー化が進行している。仮にモダナイゼーション（システムの再構築）に踏み切っても、コスト超過や稼動延伸などが発生する場合があり、ユーザ企業、ベンダ企業ともに取り組むべき課題となっている。

攻めの分野では、図 2 が示す通り、産業の垣根を越えて異なる分野の機器やシステムが連携し、目的に応じた「最適な組み合わせ」を導き出して新たなサービスを提供することが求められる。システム化における上流工程の重要性はさらに高まり、これまでのようにユーザ企業が責任を持って要件を定義する、という明確な線引きを行うだけではなく、ベンダ企業とともに協力して、多様なデータを収集、蓄積、解析してサービス化するサイクルを廻すことが重要である。

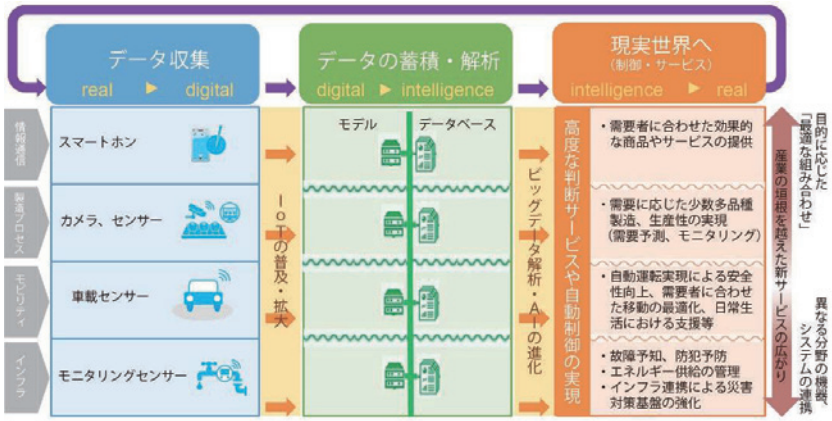


図 2　攻めの分野におけるシステム開発ライフサイクル（例）

（出典）産業構造審議会 商務流通情報分科会 情報経済小委員会「中間取りまとめ～ CPS によるデータ駆動型社会の到来を見据えた変革～」を基に IPA が作成

一方で守りの分野におけるレガシー化の課題には、図 3 が示す通り、長年の保守開発で蓄積された「業務仕様の理解不足」などによる再構築の難しさという問題がある。しかし、その難しさは可視化されておらず、下流工程における大きなリスクとなるにも関わらず、開発着手前に把握することは難しいのが現状である。開発プロジェクトの問題化を防ぐには、上流工程においてリスクを明らかにして、対策をユーザ企業とベンダ企業が合意し、ユーザ企業の経営層を含めてリスクヘッジすることが重要になる。

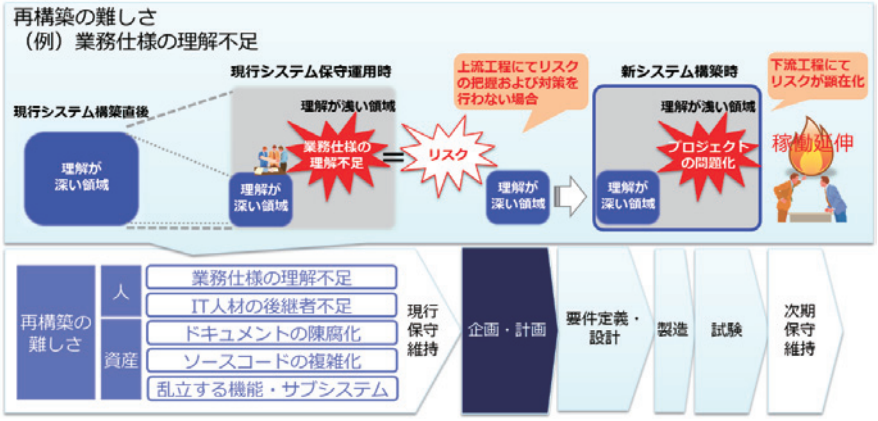


図 3　守りの分野におけるシステム再構築の事例

IT システムの変遷に伴う課題から、システム開発における上流工程の重要性が改めて浮き彫りになる。以上のような状況から、今後も、前述した上流工程における要件定義に関する問題や、再構築時固有の問題に対して、それらを解決するための課題に取り組むことが、「守り」から「攻め」へ転じる足場を固めることになる。

なお、攻めの分野に投資を拡大するには、支える側の守りが大切である。仮に業務仕様を変えずに基幹システムを再構築する場合でも、投資判断が難しい側面はあるが、最新のIT技術を用いることでセキュリティ強化や性能向上が図れるなど、得られる効果がある。経営層は、より一層システム開発の上流工程に関わり、「基幹システムは現行通りに・・・」といったように要求をあいまいにせず、現行通りとは何か、どうしたいかを明確に判断することが経営の競争力強化につながる。

本書では、図4に示す通り、ソフトウェアエンジニアリングにおける領域を大きく「基本領域」と「応用領域」の二つに分類した。基本領域は、システム開発における攻めの分野にも、守りの分野にも共通する内容として、プロセスやドキュメント、人、体制などを整理した領域である。一方、応用領域は、基本領域を取捨選択し、プロジェクト向けにテーラリングして適用する領域であり、攻めの分野であるイノベーションや、守りの分野であるモダナイゼーションなどである。

以上を踏まえ、IPA/SECでは、基本領域の一つである「要件定義」について、正しく伝えるための「How」を整理するという課題へ取り組んだ。そして、上流における要求からどのように要件を決め、表現するか。また、どのように要件定義の不備（抜け、漏れ、あいまい等）を無くし、成果物の品質を上げるかをとりまとめた。

また、応用領域である「モダナイゼーション」の領域で、上流工程におけるリスクを把握し、合意形成する手段を整理するという課題へ取り組んだ。システム再構築に特有の難しさを踏まえた工期・コスト・品質への影響を見える化した上で、要件定義着手前の企画・計画段階で下流におけるこれらの影響について合意して、リスクを減らすための施策（やるべきこと）を明確化した。

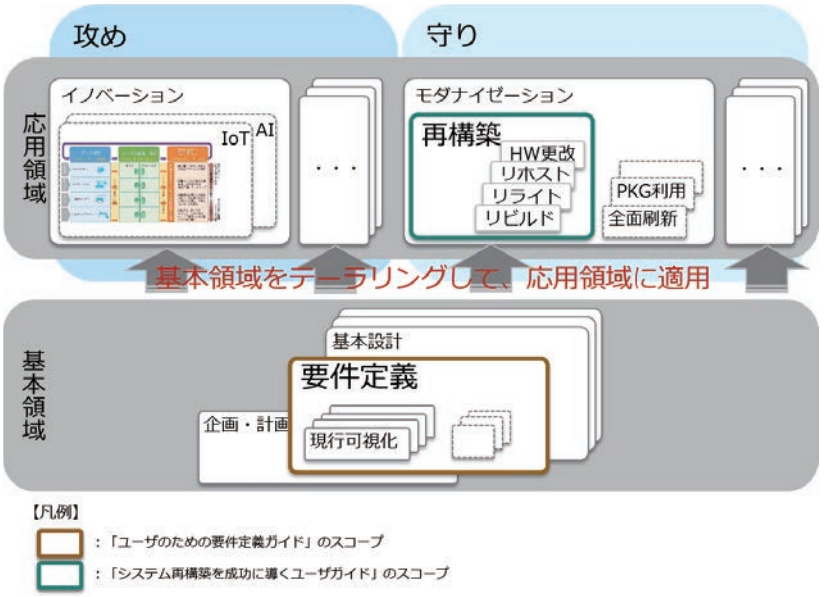


図4　ソフトウェアエンジニアリングの各領域における取り組み

例えば、品質確保の観点では上流工程での取り組みが重要であり、下流工程にかけて業務・システムに対する理解を深めていきながら実現してきた。しかし、問題化する事例には傾向があり、基本領域、応用領域ともに、それぞれに抱える問題と解決すべき課題がある。

基本領域では、要件定義をユーザ企業自身が行う際に、抜け、漏れなく行うことが難しい。「ユーザのための要件定義ガイド　～要求を明確にするための勘どころ～」（以下、本ガイド）を用いて上流工程に起因する手戻りを無くすことで、要件定義の品質向上を図っていただくことを期待している。ユーザ企業は部門間で要求を明確にして、ベンダ企業を含むステークホルダ間で理解し合い、要件定義のインプットとアウトプットの品質を高め、企業の競争力向上につなげる足掛かりとしていただきたい。

応用領域として取り組んだ再構築では、現行システムがあるがゆえ、テーラリング時に上流工程の省略などから陥り易いリスクがある。「システム再構築を成功に導くユーザガイド　～ユーザとベンダで共有する再構築のリスクと対策～」を用いることで、リスク対策を保守・運用までを含めた計画に反映して、レガシー化した基幹システムの再構築に安心して取り組めることを期待している。そして、将来的にモダナイゼーションで得られる価値を得る足掛かりとしていただきたい。

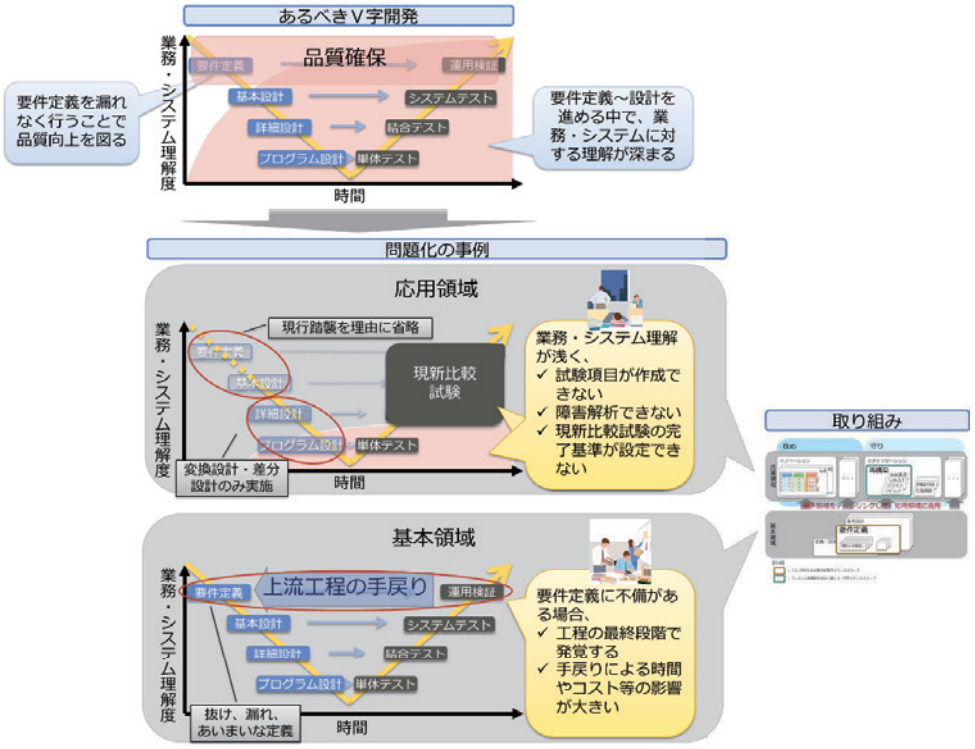


図 5　各領域に取り組む背景

なお、攻めの分野に特化した内容は、今回の内容に含まれていない。基本領域では、攻め、守り、いずれの分野のシステム開発にも共通的な内容を解決することから始めた。また、応用領域では、現在のシステムをそのまま再構築する場合を前提とした内容を整理した。

例えば「イノベーション」などに代表される攻めの分野には、開発手法の違いによる勘どころや、基本領域に追加するプロセスなどが考えられる。また、「モダナイゼーション」などの守りの分野でも、全面刷新やパッケージ導入時などの内容を整理することが求められる。両ガイドブックを踏まえて、今後の取り組みにつなげていきたい。

第１章　システム開発の現状と課題

1.1 システム開発の現状

ここでは、経営層向けに日本のシステム開発の現状からその課題を探ってみたい。その上で、そのうちの要件定義にまつわる課題を詳らかにする。

システム開発は経営要求を含むシステム化構想、要件定義と議論が進められてくるが、ユーザ企業とベンダ企業の境目は要件定義である。

要件定義の問題は IPA でも古くから取り上げられてきて「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）[1]発刊から 10 年が経過しユーザ企業の環境、技術の環境は大きな変化を遂げている。

実際にシステム開発の実態を見てみると、日本情報システム・ユーザー協会（以降、JUAS と表記）の企業 IT 動向調査報告書 2016[2]によれば、工期遵守状況は図 1.1、経営層の品質への満足度は図 1.2 のようになっている。

500 人月以上の大規模システム開発プロジェクトを見てみると、2015 年度は「予定より遅延」したプロジェクトが 42.3%を超し、26%のプロジェクトにおいて経営者がシステムの品質に不満を持っている。規模が小さくなるとこの状況はやや改善されている。
また年々工期、品質は改善されてはいるがまだまだ満足できる値ではない。

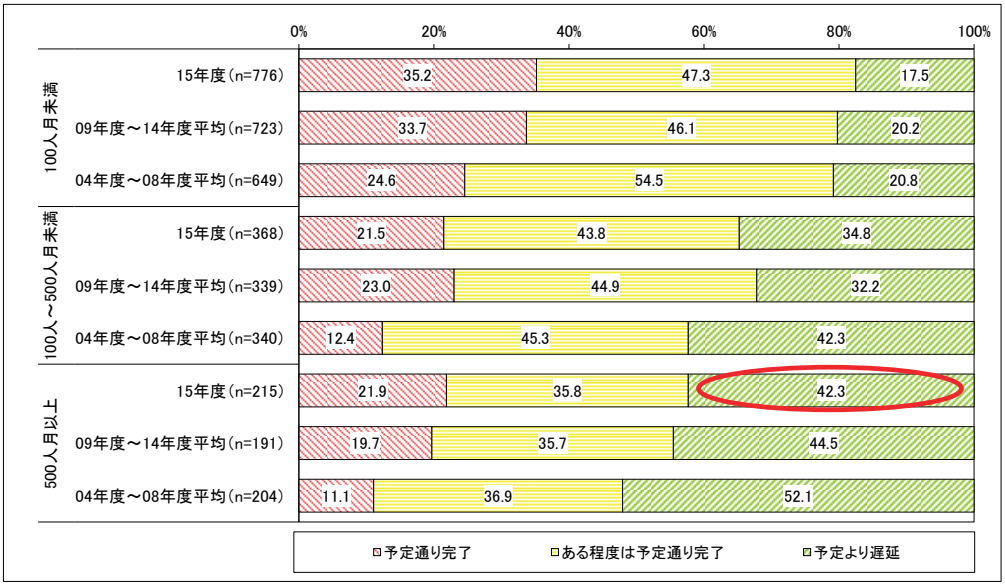


図 1.1　年度別システム規模別 システム開発工期遵守状況

（出典）日本情報システム・ユーザー協会「企業 IT 動向調査 2016」[2]をもとに加筆

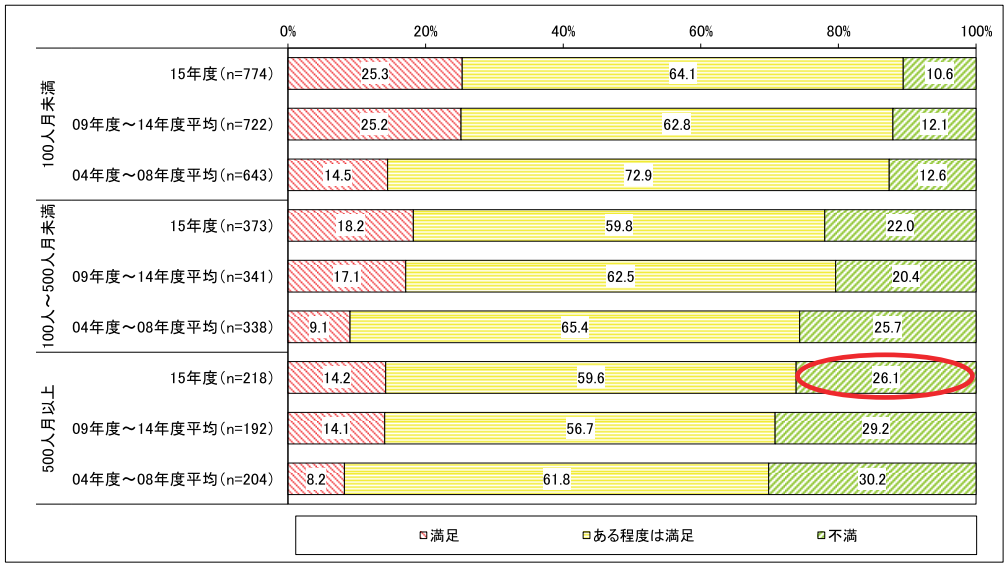


図 1.2　年度別システム規模別・システム開発の品質状況

（出典）日本情報システム・ユーザー協会「企業 IT 動向調査 2016」[2]をもとに加筆

この原因を、工期遅延理由の分析データ（図 1.3）から探ってみよう。

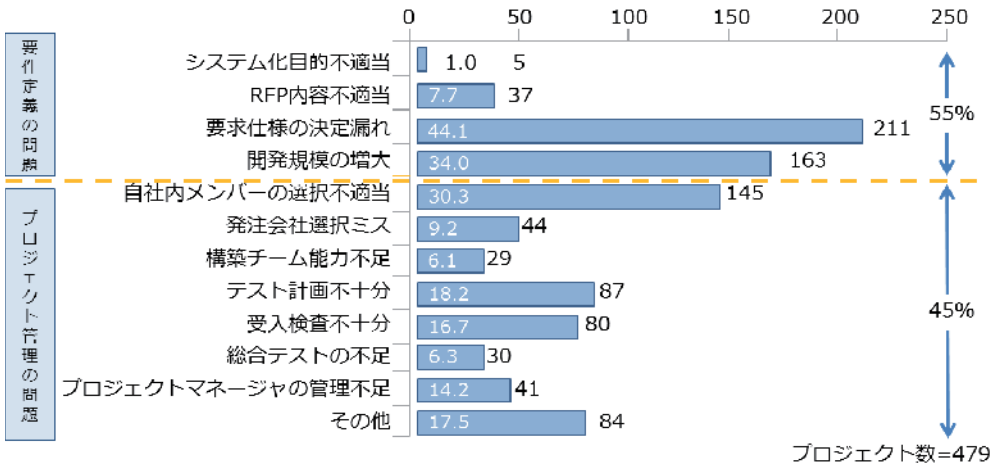


図 1.3　工期遅延理由分析

（出典）日本情報システム・ユーザー協会「ソフトウェアメトリックス調査 2016」[3]をもとに加筆

図 1.3 は、JUAS ソフトウェアメトリックス調査 2016[3]において 479 プロジェクトを対象に工期遅延の理由を担当者にアンケート調査した結果である。この図から、工期遅延の理由の 50%以上が要件定義の問題にあることがわかる。また、同じ調査の中では、予算オーバーや品質不良においても要件定義の問題が原因の多くを占めることが指摘されている。

1.2 日本の IT 投資が抱える課題

日米のシステム開発における環境の差について見ていこう。

SE がユーザ企業、ベンダ企業のどちらに所属しているかの比率を見ると、日本はユーザ企業 25 対ベンダ企業 75 であるのに対して、米国はユーザ企業 75 対ベンダ企業 25 と逆の比率になっている（出典）グローバル化を支える IT 人材確保・育成施策に関する調査）概要報告書[16]。日本では一般に入札等によってシステム開発ベンダを選択できることから、ユーザ企業にとっては安く高品質のシステムを入手できるプラス面がある。しかし、企業内での IT 新技術の採用や技術面でのイノベーションにおいては、米国のようにユーザ企業内に数多くの SE を抱えている方が有利である。

設備投資全体に占める IT 投資の割合の推移に関する欧米との比較を図 1.4 に示すが、日本は欧米に比べて IT 投資の割合が低く、その差は拡大し続けている。ただし、日本で導入できるハードウェアやシステムの調達価格が安価かつ高品質であることが、売上高に対して IT 投資比が少ないことに結びついているとの指摘もある。

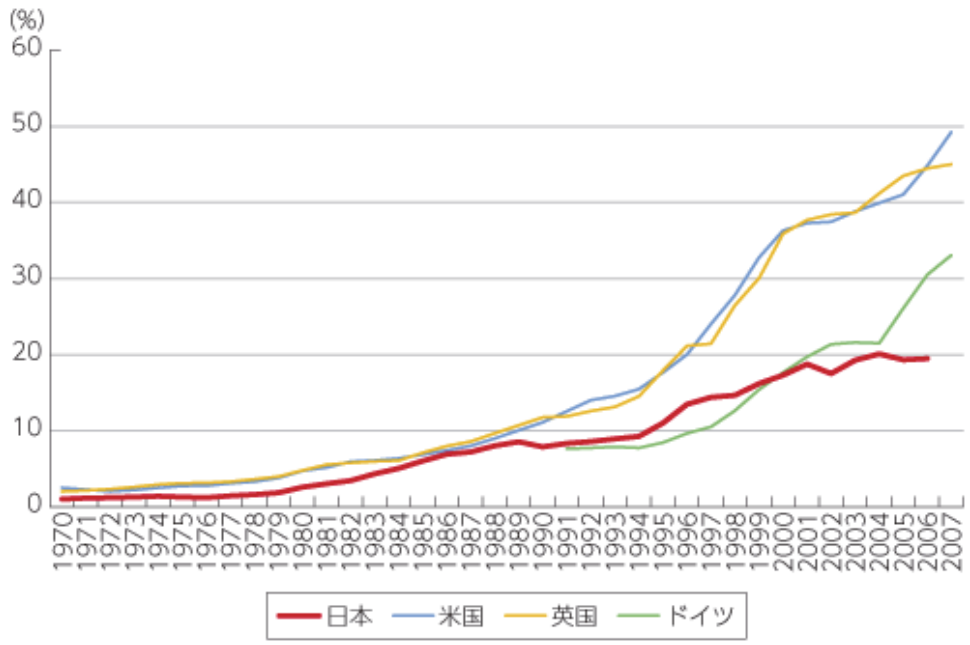


図 1.4　設備投資全体に占める IT 投資の割合

（出典）EUKLEMS より作成、総務省「ICT による経済成長加速に向けた課題と解決方法に関する調査研究」（平成 26 年）

日本と米国の経営者 200 人に、IT についての意識を質問し比較したデータがある。経営者から IT 部門に期待することとして多く選択された項目の順位(多い順に 1 位～11 位)の日米比較を表 1.1 に示す。

表 1.1　IT 部門に期待すること（3 件選択）

IT部門に期待すること	日本(216団体)	米国(201団体)
ITによる業務効率化の提案と実行	39.4%　(1位)	45.4%　(2位)
低コストでのオペレーション	33.8%　(2位)	5.7%　(11位)
要望への迅速な対応	33.3%　(3位)	20.6%　(8位)
ITを通じた業務変革提案と実行	24.5%　(4位)	25.8%　(6位)
ITによるコスト削減提案と実行	24.1%　(5位)	34.0%　(3位)
利用部門の業務内容を理解した対応	20.8%　(6位)	11.3%　(9位)
システムの可用性の向上	18.5%　(7位)	46.9%　(1位)
新規事業・製品・サービス開発の提案	12.5%　(8位)	23.2%　(7位)
IT部門における技術力の強化	9.7%　(9位)	28.4%　(4位)
システムのグローバル化への対応	6.9%　(10位)	6.7%　(10位)
新技術の紹介	2.8%　(11位)	27.8%　(5位)

（出典）平成 25 年 10 月 9 日　一般社団法人電子情報産業協会、IDC 調査報告をもとに作成

これを見ると、日本企業の経営者は低コストでのオペレーションを 2 番目に多く望んでいるが、米国は最後の 11 位の要望である。コストダウンよりも有効活用に目が向いている傾向がある。

日本企業の営業利益率は、米国企業の営業利益率と比較するとおおよそ半分である。平均3%程度の営業利益率の日本企業にとっては、対売上高比 0.5～1%の IT 費用は削減対象の目玉になるがしかし、ほぼ 2 倍の利益率を出している米国企業の経営者にとっては、IT のコストダウンよりは、前向きの IT 活用に目が向きやすいと考えられる。

JUAS の企業 IT 動向調査報告書 2016[2]によれば、図 1.5 に示すように、営業利益率がプラスの企業の場合、営業利益率が高いほど、IT 予算比率も高くなっている。つまり営業利益率が高い企業は、IT 投資にも積極的に投資できる。一方、営業利益率が低い企業は IT 投資を削減し、営業利益率を確保する消極的経営戦略に流れがちになる。

営業利益率と売上高に対する IT 予算の比率（IT 予算比率）の関係を図 1.5 に示す。縦軸は売上高 IT 予算比率、横軸は営業利益率であり、営業利益率を 0%、3%、10%で区切り、売上高 IT 予算比率の統計値を平均値、トリム平均値、中央値で計算している。また、トリム平均値は、除外する割合を 20%と 50%の二通りで計算している。

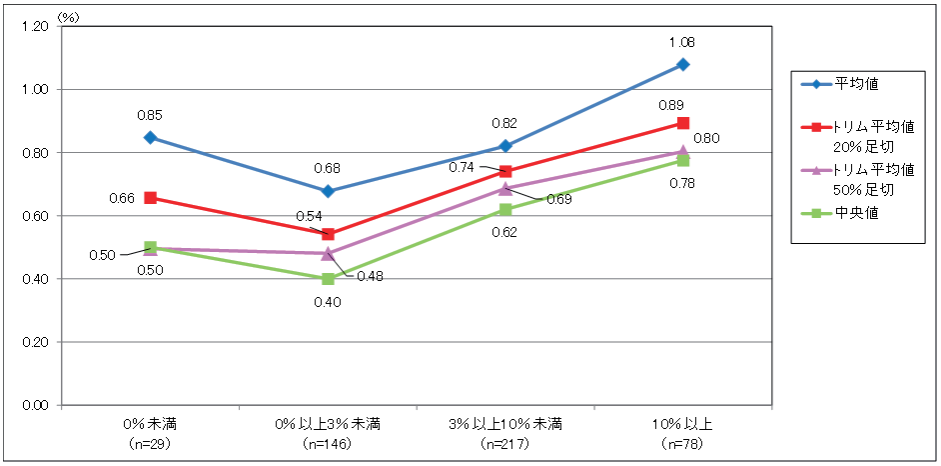


図 1.5　営業利益率別　売上高に占める IT 予算比率（除く金融）

（出典）日本情報システム・ユーザー協会「企業 IT 動向調査報告書 2016」[2]

適切な「攻めの IT 投資」および「守りの IT 投資」を実現するために、ベンダ企業は「顧客企業が求める／期待するシステムとは何か」を第一に考慮する必要がある。一方、ユーザ企業は「自社が求める／期待するシステムを入手するためには何が必要か」について、明確に定義しなければならない。

1.3 要件定義を巡る課題

システム開発の歴史を振り返ると、それは新しい分野の難しい業務であり、解決すべき課題は山積している。システム開発の要件定義の問題を解くにあたって、ユーザ企業およびベンダ企業の双方の委員に「要件定義の問題は何ですか」と質問した。以下がその結果である。

問題として 5 項目が挙げられた。

- 既存システムが存在することが普通であるが、分社化、ベンダ依存などの影響もあり、業務部門、システム部門で既存システムを知っている人が減っている。
- 要件定義を請負契約で実施している発注が未だに 50%もある。経済産業省の「情報システム・モデル取引・契約ガイド（2007・4）」[4]では、要件定義作業は準委任としている。しかし、この不確定要素の多い作業を請負契約で実行していることが、後々の問題を誘発していることは否定できない。
- 計画時の要件定義の工数、期間比率が低いプロジェクトが多い。予定した期間内で要件を決められず基本設計以降の開発作業を圧迫している。
- 確定しないで後続工程に入る（完了基準が不明確）。その結果仕様変更が多発する。要件定義書をどこまで詳しく書けばよいのかは、後続工程である基本設計以降を分担する作業者の納得感で決まる。しかし、この要件定義書の検証については、曖昧なケースが多い。
- ビジネス目的に合致した要求を抽出しきれていない。

指摘された問題から、特に要件定義を巡る課題として取り上げられるべき項目は次の 12 項目であった。この各項目の解決策は第 3 章、第 4 章で詳述する。

課題項目

① ビジネス目的と施策が合致していない
② 手段が先行し、「何のために」が理解できていない
③ 業務の複雑さが増している
④ 合意形成が取れていない
⑤ 膨らむ要求を抑えきれない
⑥ 要件定義書の不備が多い（抜け、漏れ、曖昧、不完全、不整合などへの対策が不十分）
⑦ 非機能要件を決めきれない

⑧　要件定義の記述の粒度や深さの基準が不明、内容の評価ができない
⑨　As-Is の分析、To-Be の可視化が不十分
⑩　業務部門の参画、理解が不十分
⑪　システム部門が要件を引き出せない
⑫　体制、役割分担の不備での失敗

なお、指摘された課題の中で、組織・マネジメントに関わる下記 4 項目については、本ガイドの検討範囲を超えているため、詳細については別の機会の検討を待つこととしたい。

- プロジェクト管理の不徹底（定量化、見える化、ソフトウェアメトリクス活用、リスク管理、変更管理が不十分）
- 業務部門のビジネスアナリスト育成が不十分
- 仕様変更の後続工程での対応不備、後続工程への影響が不明（リポジトリ技術の活用により負荷減少が図れると思われる）
- 新技術、新環境への対応が不十分

1.4 本ガイドの構成

「第 1 章　システム開発の現状と課題」（本章）では、日本の IT 投資の現状とシステム開発の課題を概観し、システム開発を巡る問題を整理した。

「第 2 章　経営者／プロジェクト責任者が考慮すべき要件定義のポイント」は、システム開発において重要となっている要件定義で考慮すべきポイントを、経営者、プロジェクト責任者向けに紹介している。

「第 3 章　昨今直面している要件定義課題を解決するための勘どころ」では、第 1 章、第 2 章で取り上げた要件定義を巡る課題に対し、解決するためのポイントを端的にまとめている。課題を認識しても、実際に要件定義の実施に関わる業務部門やシステム部門、ベンダなどの要件定義実務者が自ら実施できるための方法を提示しないと、なかなか実行には移せない。第 3 章では、「経営や業務に貢献する IT システムをいかに構築するか」「膨らむ要求をいかにコントロールするか」など、昨今特に重要になってきている課題を、一部ではあるが取り上げ、要件定義実務者が実行するためのハウツーを紹介している。

「第 4 章　要件定義成果物の品質向上」は、要件定義の主要な成果物をリストアップし、それらには何をどの程度まで記述すると要件定義の漏れが減り、後続工程以降からの手戻りが少なくなるかについて、要件定義の実務担当者の経験を踏まえてまとめている。成果物に何をどの程度まで記載するかは、これまで明確なガイドが少なかった。要件定義に取り組む実務担当者に活用してもらいたい。また、重要な成果物に対して、その作成上で想定される課題の解決や品質向上のための勘どころについても紹介している。

特に第 4 章に書かれていることは、従来では、基本設計工程（外部設計工程と呼ばれることもある）で実施することが多いことも含まれているが、業務要件／システム要件の漏れをなくし、後続工程からの手戻りを少なくするという観点から、要件定義工程で決めるべきこととしてガイドしている。

「第 5 章　事例」は、第 3 章、第 4 章で説明した項目について、各社ではどのように適用し、実践しているかの事例を紹介する。

また、本ガイドの対象読者としては、図 1.6 に示すように、上記各章ごとに主たるステークホルダを想定している。なお、本ガイドはユーザ企業に向けたものであるが、ベンダ企業の支援が必要なケースも考えられるため、対象読者に含めている。

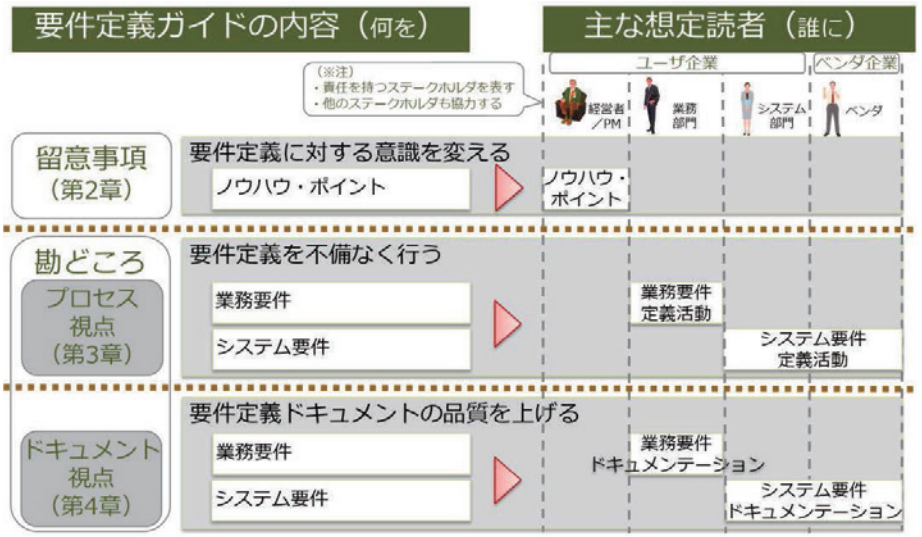


図 1.6　本ガイドの想定読者

なお、プロセスの観点からは、「共通フレーム 2013」[15]によれば、次のように対応付けられる。企画プロセスにおいて明確化した要求に基づき、要件定義プロセスにおいて業務要件の定義を行う。システム要件の定義は、システム開発プロセス内システム要件定義プロセス、及びソフトウェア実装プロセス内ソフトウェア要件定義プロセスに位置づけられている。「共通フレーム 2013」では、プロセスと開発工程（フェーズ）とは独立であるものとしているが、それらのマッピング例として、要件定義フェーズに要件定義プロセスを、開発フェーズの最初にシステム・ソフトウェア要件定義プロセスを、それぞれ位置づけている。開発工程については、企業等の事情に応じて規定されるものである。たとえば、開発フェーズのシステム・ソフトウェア要件定義プロセスを、"外部設計工程"や"基本設計工程"と呼んでいるケースがある。

第２章　経営者／プロジェクト責任者が考慮すべき要件定義のポイント

2.1 当該システムを開発する妥当性を検討する

経営課題に対して適切な位置付けのIT投資であるか

(1) ビジネス構造とその最適化対策－構想段階の準備はできているか

あるグローバル企業のシステム化計画の一部を図2.1に示す。

同図の第1行目に会社で必要とする一般的な機能を記述してある。第2行目以降には、その機能を、関係組織を通じてどのように最適化していくかを示している。

国内だけでなくグローバルを視野に入れて最適化を志向する過程を記述する。「Local」は各国別に、「Regional」は世界の地域ごと、例えば北米地域、アジア地域などのシステムを作り、「集中化」は世界で一つのシステムを使うことを意味している。

日本企業の多くは、子会社を含めて何十年も前から別々にシステムを開発し高価な支出をしてきたが、今後は国内外の関係企業が連携し合って最適化経営をする時代にとなる。時代の変遷に合わせて業務、そしてそれを支援するあるいは担う情報システムを見直す長期プランの原点を、このような形で残しておくことが望ましい。

Global Systemの構造例
業務、情報システムの標準化を進めロスを防ぎ競争力を高める
・グローバル市場は日本市場の10倍

各1区分が右のように細分化される → Local / Regional / 集中化

機能 / 時間	mail/ Internet	財務、人事 設備管理	調達	生産	配送	マーケティング	販売	研究
現状	Local	Local	Local Regional 集中化	Local	Local	Local Regional	Local Regional	地域特化
3年後	集中化	Regional	Local Regional 集中化	Local	Local	Local Regional	Local Regional	集中化
将来	集中化	集中化	集中化	集中化	Regional	集中化	Regional	集中化

※実際は各機能をさらに細分化し、業務および情報システムの両面について、柔軟性をもった、業務と情報システムの単純化、標準化、共通化をさらに追求する必要がある

Local	各国、各工場の事情に任せる
Regional	地域の地理、制度、文化の特性を生かして地域最適化を求める
集中化	全社最適を狙うシステム 全社情報を参照しながら支社が具体的なアクションをする場合もある

図2.1　Global System の構造例

（出典）日本情報システム・ユーザー協会「JIIP3 2014年度報告書」[5]をもとに作成

(2) 自社の新商品・サービス、技術、組織、人事制度、企業風土・文化の将来にわたっての改革案の構想を整理しているか

経営者には、「新商品・サービス、技術、組織、人事制度、企業風土・文化、IT 活用の将来にわたっての中期計画改革案を立案する」ことが求められる。

例えば商品について、現状の商品、3 年後の商品、6 年後の商品などについて機能、デザイン、販売先などをどのように変えていくのかを明記する。実際にそのようにならないことは多いが、まずは基軸となる案を考えてあれば、変化に追随しやすいだけでなく、発想の転換もしやすくなる。

人事制度も「言われたことを確実に実行できる人材の育成」から「新しい商品・サービスを考えることができる人材の育成」、「ワクワクして働く職場を創り出せる人材の育成」など、どのように展開していくのかを描けたら、自社の発展の形が見えてくる。

経営思想についても、CSR[1]経営から CSV[2]経営に切り替えるべきか否か、切り替える場合にはどのように切り替えてゆくのかをまとめておくことも必要である。

このような背景があった上で、情報システムはどう変化していくのかを考える会社と、目先の変化・問題に対応して対策を追いかけている会社との差は大きい。

(3) 企業のグローバルな発展を考慮した場合に、このシステムを今開発することは妥当か

どんなに最先端の技術を駆使したシステムであっても、市場ニーズとマッチしていなければ、ユーザ企業には受け入れられない。大切なのはタイミングだ。売上高も少なく、業務の方法が安定していない時期に大掛かりなシステムを構築することが良いことなのか、もう少し待ってから開発した方が良いか、を十分に考え優先度をつけて開発することに配慮したい。

(4) 商品の性質、顧客、生産販売方式などが従来商品と大きく異なる場合にシステムを独立化して早く立ち上げるなど迅速な対応を考えてあるか

新商品を既存チャネルで販売しようとしてもうまくいかないことは多い。具体的手順、方法の見通しが立たないことは頻繁に起こるからだ。

売り方が分からないような場合に、既存システムを使うことなど考える必要はない。小規模で変化に強い個別システムでスタートした方がよい。安定期になったら全社システムとの整合性を論ずれば十分である。たとえシステムの組み直しになっても、事業開始時期の諸問題に臨機応変に対応できることが先である。

---

[1] CSR（Corporate Social Responsibility） 企業の社会的責任

[2] CSV (Creating Shared Value) （社会的な）共通価値の創造

(5) 現在のシステムに対して各関係者が抱いている不満があるならば、その原因と対策を分析整理してあるか

もともとコンピュータは多くの業務領域において人手よりも処理能力が高いが、使い方によっては望んだ結果が得られずに高い処理能力を無駄にしてしまうことも多く、システム化には難しい側面がある。「もっと良い方法がないか」と関係者全員が一緒になって現状のシステムの棚卸をするきっかけとなるのが、顧客満足度調査である。

(6) IT 投資金額は売上高比、人件費と対比して妥当なものか

この質問を、主要業務部門、システム部門の関係者に行い、総意を出してもらうなどして、IT 投資金額（対売上高 IT 予算、一人あたり IT 予算等）、IT 要員比率を算出する。そして、それを各業界の平均的な値と比較し、妥当かどうか判断する。

JUAS の企業 IT 動向調査報告書 2016[2]による IT 投資金額、IT 要員の業種別比較を表 2.1 に示す。同じ業種でも性格の異なるものが含められていることや、企業規模により IT 活用に差が出てくることから、あくまで参考値であるが一つの目安にはなる。

同報告書では、通常は 5 年償却で、そのあとは残存簿価の 1/10 だけが残るが、大きな企業イノベーションが必要な場合には、表 2.1 の平均値・中央値よりも IT 投資金額は増加すると述べている。このことから、思い切った投資が必要な場合には、思い切って投資し、企業体質を強化することが企業の収益性を高め、企業の寿命を長くさせるという共通認識が理解できる。

なお、実際には保守運用費が開発費用に計上され、表 2.1 にあるように、多くの企業における IT 予算は、売上高の 1%前後の費用である。ただし、金融業界では IT 予算（IT 投資）が他の業種より多い。一般に、金融業界の多くの企業が、他の業種の平均的な企業よりも収益性が高く、企業の寿命が長い傾向にあるからだ。業種によって業務の IT 化のやりやすさに違いはあるが、この傾向は IT 投資の結果を多少とも反映していると言える。

表 2.1　IT 投資金額、IT 要員の比較

業種区分	対売上高 IT 予算		一人当たり IT 予算	IT 要員(注)比率
	平均値（%）	中央値（%）	平均値（万円）	平均値（%）
全体	1.21	0.60	141.3	2.9
建築土木	0.49	0.38	47.7	2.0
素材製造	0.75	0.53	93.3	2.6
機械製造	0.67	0.56	122.5	2.3
商社流通	0.73	0.50	107.7	3.3
金融	7.82	6.43	366.8	4.1
社会インフラ	1.18	0.80	209.0	3.8
サービス	1.25	0.60	36.0	2.7

（注）IT 要員＝IT 部門の要員+事業部門の IT 要員+情報子会社の IT 要員(外販要員を除く)

（出典）日本情報システム・ユーザー協会「企業 IT 動向調査報告書 2016」[2]

経営者の視点での問題認識と対策を十分に盛り込む

(1) コンペティタを上回る戦略とは何か

国内、海外の競争相手が何を仕掛けてくるかを想定し、対策を考えることは企業戦略上欠かせない。IT 投資計画の立案審査のタイミングでこのことについて協議することは非常に重要である。「このシステムが完了した場合に次は何が出てくるのか、何に投資をするのか」を現時点で推測し、議論をしておくことは、関係者各自の意識向上にも役立ち、企業戦略上も極めて重要なことである。

この議論をすると、「このシステム開発は今の企業文化で実施すべきではない」と否定されることも出てくる可能性はある。ただし、議論をすることの価値は大きい。

(2) 自社の社員の実力でこのシステムを使いこなせるか、それを補うための戦術は何か

システムを使用するのは従業員である。高価なパッケージを導入しても使いこなせなければ、導入に掛けたコストは水泡に帰すことになる。利用者の作業水準に見合ったシステムであるかどうかも、システムの導入を判断する一つのポイントである。

技術進歩を予測し、システムを開発するタイミングは妥当か

最新技術の必要性を見通すことは難しい。先行して新技術の実用化に着手すれば、その技術についての知識と経験は企業内に蓄積できる。しかし、より優れた技術が比較的近い時期に利用可能となる場合には、競争相手がそれを導入し、活用を始めれば置いてきぼりを食うことにもなる。

技術の先見性を追究している企業における技術戦略計画・実行の例を図 2.2 に示す。5 年先を予測し、3 年先を見通しつつ直近の 1 年間の実行計画を推進していることを表している。

同図の企業では、毎月各技術について 5 年先のレベル、3 年先のレベルを想像し「今年は何をするのか」を決めている。競争相手は世界中にいて、どの競争相手もこの程度の検討はし続けていると考え情報収集をしたほうがよい。それには企業のシステム部門、あるいは業務部門にこのような情報収集の専門家を確保し、予算を確保する必要がある。

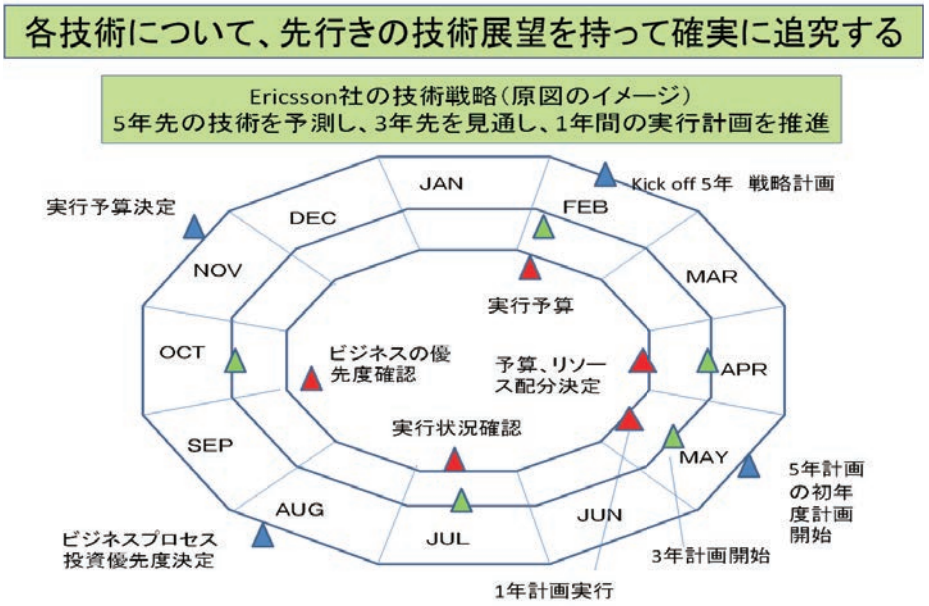


図 2.2　技術戦略の例

（出典）日本情報システム・ユーザー協会「JIIP3　2014年度報告書」[5]

リスク評価を踏まえた非機能要件について、ユーザの要望を明確にする

必要としている機能を明確にすることは、最初に実施すべきことである。同時に、次に挙げるようなリスク評価を踏まえた非機能要件の指示も、重要となっている。

- 最初に配慮すべきは、企業の社会的責任である
- 大地震、津波、大火などのリスクが発生しても企業は責任を果たせるのか

非常事態にシステムが動かなくなった場合、その間は仕事ができない（人手で代替できない）ことが多い。

従来は、システムに関する想定外の事態に対する技術的対応（非機能要件）は、システム部門の専門家に任せておけばよいと考えられてきた。しかし、今やリスク対応について検討することは、経営そのものである。

他社との競争を意識した場合に工期は妥当か

新商品の発売や新サービスの開始時期は他社との関係で決まることが多いため、その関連システム開発には短工期を期待されることが多い。苦労して開発した新システムであっても新商品発売や新サービス開始の時期が他社の後塵を拝したのでは、全く評価されなくなってしまう。

2.2 システム化の前に業務を分析・整理する

業務部門の責任者が現行システムの実態を正しく理解し、要件定義に反映しているか

自部門の業務を IT 化する場合、各作業を人手、機械、システムのうちどれで行うかの組み合わせを考えることになる。特にシステムにどのような機能を乗せるのか、その機能を今後どのように改善すればさらに自部門の効率化が図れるのか、業務担当管理者は理解する必要がある。

一定規模以上の企業では、何もシステムを使用していないことはない。予想された使い方以外の使い方をされている場合もあるし、使われていない機能もある。システムの基本機能がブラックボックス化して、利用者には理解されていないことも多い。利用者からみてシステム部門やベンダ企業に対する満足度が低い場合もある。

そのため、図 2.3 の①に示すように、システム開発の前段階でこの実態を数ヶ月前から時間をかけて、「次期システムには何を期待するのか」「今のシステムで使用されていない機能はないか」「数年前の開発時と異なり現在の環境にふさわしくない機能はないか」を整理、分析する期間を設けることが望ましい。その過程で「もっと効率的な、もっと顧客に喜んでもらえる」機能や仕組みがないかを、関係者にヒアリングすることも重要だ。

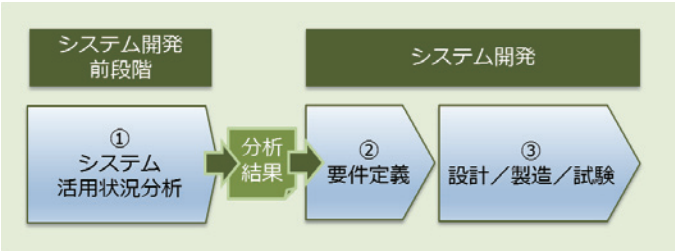


図 2.3　システム開発前段階を含めたプロセス

次にすべきは、同図の②に示すように、分析結果を反映して要件定義作業に入る。その後、③の設計に進む。センサーやシステムの機能は、常にレベルアップしているので、既存システム開発時にはできなかったことが、一定期間経過後にはできるようになっている場合は多い。「既存システム機能の上には、もっとよいシステムがある」のは事実である。この段階こそ経営者の活躍の場である。この段階で大きな期待ができる新機能が見つからなければ新システムにする意義は半減するため、システム開発を見送り、資源をより優先度の高い投資に回すという経営判断を下すことができる。このプロセスを経てから、システム開発に入らなければならない。

全社視点で見て重複作業や無駄を十分に排除したシステムになっているか

顧客情報データベース（以下、顧客情報 DB）には企業名、住所、電話番号、などの情報が蓄えられている。これは全社共通のものである。同一企業でも業務部門ごとのシステムの担当者が異なる場合、よく注視しておかないと、各担当者が別々のデータベースを作り、それをそれぞれ更新していくシステムを作り出す。

また、各システムはお互いに情報を交換するインタフェース機能を持っていることが多いが、これが複雑になってくる。システム間の情報交換に柔軟性がある場合は、システム変更の際に限られたシステムだけを変更すればよいが、システムの作り方によっては全部のシステムに手を入れないと機能変更ができない仕組みになってしまう。

このようにシステム間の連携を簡潔に保つ仕組みがないと、企業運営の柔軟性が欠けるので注意が必要である。特に注意が必要なのは、全国各地に広がる工場のシステムがそれぞれ独自に同様な構成のシステムを作ってしまうことである。図 2.4 の例に示すように、工場内の（a）システムと（b）システム、および工場用のシステムを別のシステムとして作ることが多いが、その場合でも同じ顧客情報 DB を使用することが望ましい。このようなことを監視するのも経営者側の役割の一つである。

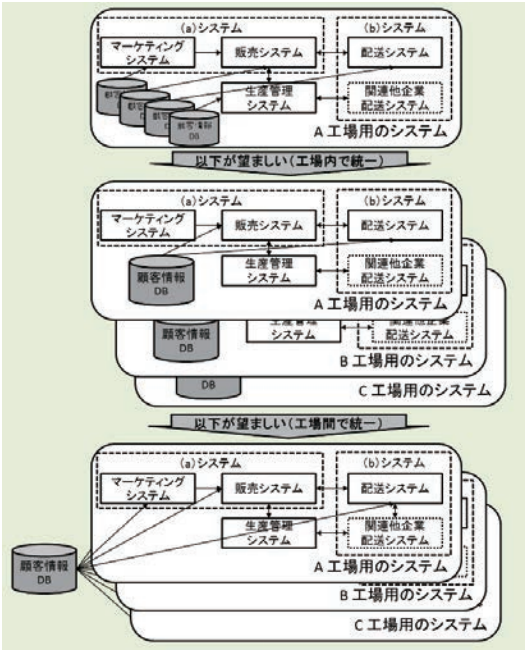


図 2.4　工場用システムの例

新組織を作成した場合や担当者を変えた場合に、上記に挙げたような重複作業や無駄が生じやすい。組織や作業方法を変えた場合、必ず削除すべき機能が生じていると考え、無駄の排除を心掛けなければならない。

2.3 要件定義を確実に効率よく進める

合意形成に時間がかかることを前提としてプロジェクトを計画する

JUAS・ソフトウェアメトリックス調査 2016 によると、要件定義には開発工程全体のうち 20%の期間が費やされている。（後述の表 2.2 参照）それでも要件のすべてを決め切れずに基本設計以降に持ち込む場合が頻繁に起こる。これは、要件の合意形成に時間がかかるためである。

合意形成に時間がかかる原因の一つは、ステークホルダ（システムの利害関係者）の数が多いことであり、その数は従来に比べて増えている。

ステークホルダの例を図 2.5 に示す。プロジェクトスポンサ（担当役員）、プロジェクトオーナ（部門長）を始めとして、プロジェクトメンバ、システムの社内・社外のユーザ、さらには、社内の関連部門、社外の関連部門、ビジネスパートナ、開発ベンダ、など、想像以上に多くのステークホルダが存在していることが分かる。

要件定義の期間の大半を目的が曖昧な議論に費やした結果、合意形成が期間内にできず、スケジュール遅延や計画見直しを余儀なくされるケースが多い。要件定義のスケジュールを立てる際は成果物作成だけでなく、合意形成にかかる期間を十分に確保できているかを確認することが望ましい。

そして、全関係者が要件定義作業の優先度を上げて取り組み、要件定義期間内に作業を完了する。そのためには、業務部門の責任者が業務仕様を早く決定する意思を持つことが第一である。また、そのためのプロジェクト管理の工夫も必要である。

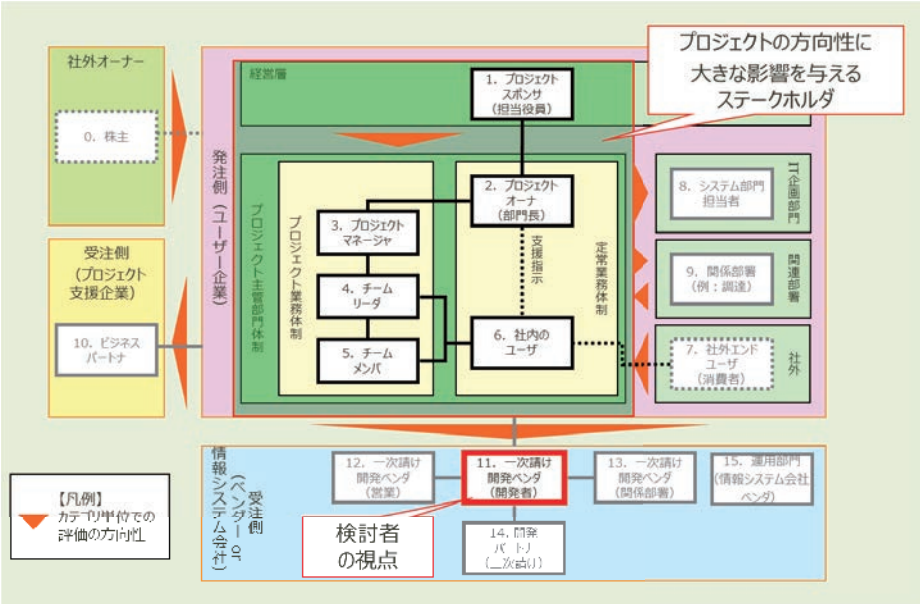


図 2.5　ステークホルダの整理

（出典）日本情報システム・ユーザー協会「サービスサイエンス研究プロジェクト（2011 年度）検討資料」

「要件をすべて決めることは難しいため、早く開発に着手して、手直しをしながら本番を早く迎える方がよい」との指摘もある。確かにどのような販売ルート、方法で販売したらよいか分からないため、業務モデルが決定できずに、開発、運用を繰り返えしせざるを得ない情報システムを構築する場合もある。そのような場合は、反復型開発、あるいはアジャイル開発を採用することもあり得る。

この繰返しのために増える予算は、ユーザ負担となる。また、この方式で開発したシステムは、ソフトウェア構造を工夫すると共に修正方法を適切に管理しないと、機能追加を繰り返し、サボテン型システムになりやすいので注意が必要である。なお、一般には、大規模システムへのアジャイル開発適用は、日本ではそのままでは難しいと言われている。

要求は膨らむことを前提としてプロジェクトを計画する

システムの現状分析を行い、要件定義が進んで、新システムのイメージが構築されつつあると、要求者は次々とアイデアが湧き出て要求が際限なく出てくるケースがある。

計画時には規模の大きさを 2 と見積もって開始したものの、要件定義を通して 4 まで膨らむ。そこで予算に入らないため規模縮小を検討し、最初の 2 にまで戻すが、実際開発完了時には 3 になっている。これは「2:4:2:3 の法則」と呼ばれている。要求は膨らむことを前提に、膨らんだ際の取捨選択基準や意思決定方法を予め計画しておくことが重要である。

2.4 決めるべきことは決めてから次に進む

要件定義工程では、定常作業、例外作業、特例作業などの作業条件の内容は決めることができる。一方、サーバや端末の機種などはこの工程で決まっていないこともあり、システム要件の中には決められない事項もある。この部分は基本設計工程に持ち込まれる。

要件定義期間で決めておかねばならないのが業務要件だ。しかし実際には決められないケースが多い。その理由は「要件定義期間があらかじめ決められているので、次工程に入りましょう」とする悪しき習慣である。期間内に決定できないのであれば、この要件定義工程を延長してでも、決定すべきだ。その方が後続工程からの手戻り負荷を減らし、結果的に、安く、よい品質の開発ができることになる。

要件定義工程で決めるべきことを決めるために十分な工期、工数を割り当てる

通常の情報システムの開発期間は、要件定義、設計、実装、テストの 4 工程に分けられる。この 4 工程それぞれに要した期間の全体工期に占める割合をシステムの規模、新規開発／改修・再開発で分けて調査した結果が表 2.2 である。この表を見ると、要件が複雑化しているにも関わらず、要件定義に割いている工期の比率は、どの規模においても 20%前後である。

表 2.3 には、開発工程別の工数比を示す。新規と再開発でやや様子は異なるが、規模が大きくなり仕様が複雑になってくるにも関わらず、要件定義の投入工数の割合が少なくなってくるのは、問題を抱える前兆である。全体工数における要件定義工程の割合で判断することが望ましい。「規模が大きくなると、おのずと実時間、実工数で要件定義にかかるコストも増える」ということを理解した上で、要件定義工程の工数割合を増やすことが望ましい。

両表を合わせて見ると、全般に、要件定義の工期割合は、投入工数の割合に比べて大きい。また、新規開発の場合、要件定義を含む各工程の工期割合は開発規模によらずほぼ一定であるが、要件定義の投入工数割合は開発規模が増えるに従って小さくなっている。これは、実装やテストでは規模見合いで要員を投入することにより進捗が進むものの、要件定義では要員を増やすことが進捗に必ずしも効果がないことを表している。しかし、再開発の場合には、開発規模による変動はあまり見られない。ここにシステム開発の難しさが潜んでいる。この点をよく考慮した計画が求められる。

表 2.2　開発工程別工期比

全体工数	開発種別	件数	要件定義＋設計＋実装＋テスト工期を100%とした工期の割合			
			要件定義	設計	実装	テスト
500人月未満	新規	194	20.7%	25.6%	28.4%	25.3%
	改修・再開発	159	19.7%	24.7%	27.3%	28.4%
	合計	353	20.2%	25.2%	27.9%	26.7%
500人月以上	新規	30	19.9%	25.0%	26.4%	28.7%
	改修・再開発	20	19.5%	25.1%	29.8%	25.7%
	合計	50	19.7%	25.1%	27.7%	27.5%
合計	新規	224	20.5%	25.5%	27.9%	26.1%
	改修・再開発	179	19.6%	24.8%	27.7%	27.9%
	合計	403	20.1%	25.1%	27.8%	26.9%

（出典）日本情報システム・ユーザー協会「ソフトウェアメトリックス調査 2016」[3]をもとに作成

表 2.3　開発工程別工数比

区分	全体工数	件数	合計を100%とした比率			
			要件定義	設計	実装	テスト
ウォーターフォール型開発 新規	10人月未満	11	20.2%	23.3%	37.0%	19.5%
	50人月未満	64	12.5%	23.5%	43.1%	20.9%
	100人月未満	32	10.3%	25.5%	40.1%	24.1%
	500人月未満	59	11.9%	22.6%	37.4%	28.2%
	500人月以上	25	9.5%	20.2%	39.5%	30.8%
	合計	191	10.4%	21.3%	39.0%	29.3%
ウォーターフォール型開発 再開発	10人月未満	6	11.6%	14.5%	42.5%	31.4%
	50人月未満	56	8.1%	21.7%	40.2%	30.0%
	100人月未満	39	9.4%	20.2%	40.0%	30.5%
	500人月未満	63	10.8%	22.5%	33.7%	33.0%
	500人月以上	21	7.5%	22.9%	40.6%	29.1%
	合計	185	8.9%	22.4%	38.1%	30.7%
合計	10人月未満	17	18.0%	21.0%	38.4%	22.5%
	50人月未満	120	10.2%	22.6%	41.6%	25.7%
	100人月未満	71	9.8%	22.5%	40.0%	27.7%
	500人月未満	122	11.3%	22.5%	35.5%	30.6%
	500人月以上	46	8.7%	21.3%	39.9%	30.1%
	合計	376	9.7%	21.8%	38.6%	29.9%

（出典）日本情報システム・ユーザー協会「ソフトウェアメトリックス調査 2016」[3]をもとに作成

繰り返しになるが、「要件定義で決めるべきことは要件定義工程内で決める」覚悟で臨むことが重要である。表 2.3 によれば要件定義工程には、現在は小規模システムを除くと全体工数の 10%前後の工数しかかけていないが、それ以上の工数をかける計画を最初から作ることがプロジェクトマネジメントを円滑にするコツの一つである。

「まだ要件定義で決めねばならないことが多いが、基本設計以降で決めれば何とかなるのではないか。実装を行うプログラマも準備しているので、先を急いで欲しい」などのベンダの要望もあって、要件定義打ち切りとなってしまうことが多い。

しかし、要件定義工程で気がついた修正に要する作業負荷は少ないが、設計、実装、総合テスト、稼働後に気がついた修正の作業負荷は 2 倍から 30 倍になることに留意し、要件定義に取り組む必要がある。

「1 文惜しみの、百の損」。ここでの工期延長、工数増加を惜しんだ結果、設計、実装、テスト期間や負荷が膨張して予算超過になるケースは後を絶たない。要件定義工程で移行計画、総合テストシナリオとテストケースを作成することにより、要件の不足が発見できることは多い。この効果は総合テストや移行作業の負荷軽減、障害発生防止に大きな効果を発揮する上、開発したシステムの品質向上に役立つ。

要件定義成果物の品質向上のための課題

今まで見てきたように、システム開発の工期遅延、工数増大は、要件定義工程で決めるべきことを決めきれずに、後続工程からの手戻りが発生し、かけなくても良い余計な負荷が増加するために発生する。そのため、要件定義の成果物（ドキュメント）の品質向上が重要となる。

要件定義の成果物については、多くの企業やプロジェクトにおいて、標準化を行ってきた。その標準化の多くは、要件定義の成果物のフォーマットを規定したり、成果物サンプルを準備したりしてきた。しかし、成果物の品質向上には、大きく 2 つの課題がある。

1 つは、作成する要件定義の成果物をどう考え、どのように作り上げていくのか、もう 1 つは、成果物に何を書けばよいのか（漏れ）、どのレベルまで書けばよいのか（粒度や深さ）といった成果物の品質の問題である。

第 1 の課題については、第 3 章で、要件定義を実施する際の課題を取り上げ、その具体的な解決方法の勘どころを説明する。

第 2 の成果物の品質向上については、第 4 章で、主要な 12 の成果物を取り上げつつ、何を書けばよいか（漏れ）、どこまで書けばよいか（粒度や深さ）などについて、成果物作成上の留意事項・勘どころを中心に説明する。

2.5 ユーザとベンダの役割分担を考える

要件定義の契約基本形は準委任契約である

要件定義は準委任契約にすることが経済産業省の「情報システム・モデル取引・契約書（2007・4）」[4]に掲載されている（図 2.6 参照）。その後のシステム設計は請負または準委任契約を推奨すると記載されている。図 2.6 に示す契約モデルが発行される前は、ソフトウェア開発は全工程一括請負が大半であった。そして問題が多発したので、この契約方式に切り替え、開発に関するトラブルは減少した。

システム開発は仕様が明確に決まっていれば、後はそんなに大きな問題が発生することは多くない。もし、基本設計（システム設計）以降の作業管理者が「要件定義仕様決定が不十分なので、基本設計に入れない」と感じた場合には、基本設計も準委任契約で、要件定義での仕様決定が十分であるかを確認する必要がある。

なお、再構築の場合には、現行仕様の理解不足である場合が多い。そのため、要件定義に入る前に現行仕様を調査し、分析する工程を設けるなどの考慮が必要だ。

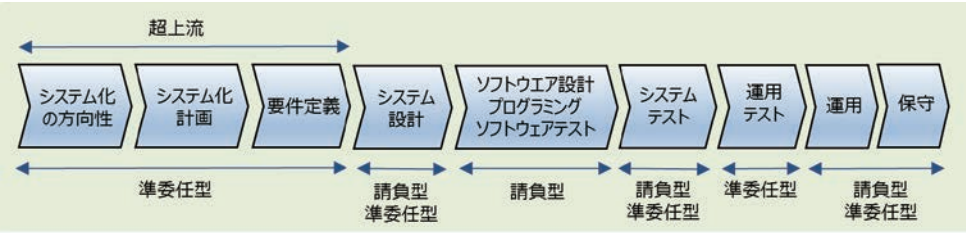


図 2.6　信頼性向上・取引可視化のための標準モデル

（出典）経済産業省：情報システム・モデル取引・契約書（2007・4）[4]より作成

全工程を請負開発で行うことのリスクを示したデータとして、契約方式と発生欠陥数の関係の分析結果を表 2.4 に示す。全工程を自社開発で行った場合と全工程を請負開発で行った場合を比較すると、換算欠陥率の差が特に大きいことが分かる。「すべてお任せ」の請負契約は、最初の要件定義工程からユーザの参加割合が低くなりがちで、最後の総合テスト時点になって「これでは使えない」とユーザが変更を申し出るパターンが多く、その結果プログラム修正に追われて、品質低下に陥るケースが多い。要件定義を自社開発または準委任型契約とすることにより、ユーザに責任の自覚と覚悟が高まり、結果としてプロダクトの品質が高くなる。

表 2.4　契約方式と発生欠陥数の実態

フェーズごとの契約形態			換算欠陥率(注)	
要件定義	設計	実装	件数	平均値
委任	委任	委任	29	0.38
委任	委任	請負	11	0.36
委任	請負	請負	57	0.27
請負	請負	請負	92	0.62
自社開発	自社開発	自社開発	39	0.24

(注)換算欠陥率：欠陥を大中小に分類し、重み付けをした欠陥数／全体工数

(出典) 日本情報システム・ユーザー協会「ソフトウェアメトリックス調査 2012」[17]をもとに作成

したがって、たとえば、「もし未決定事項の作業項目が何件かあり、その負荷が本来の要件定義作業工数の 10%を超える場合は、基本設計は準委任契約とする」などの文言を契約文書に入れることにより、ユーザの自覚を促しシステム開発のトラブルを避ける方法も考えておきたい。

表 2.4 の結果を踏まえ、ユーザ企業からの要求が明確な場合とそうでない場合の契約方式の要点を表 2.5 にまとめた。ユーザ企業の要求内容が明確であるか、要求が漠然としているかで、契約方式を分けることが望ましい。また、ベンダ企業が同種のプロジェクトの経験を持っているかも、重要なポイントとなる。

ユーザ企業の要求が明確で、過去にも取引き経験がある場合には、一括請負契約の選択肢もあり得るが、あくまでも要件定義は準委任契約が推奨される。一方で、ユーザ企業の要求が漠然としていてベンダ企業も経験がない場合には、基本設計まで準委任契約とし、仕様の明確化を図ることが不可欠だ。

表 2.5　システム開発プロジェクトで成功する契約方式

		ベンダ企業 (経験有／リスク低)	(経験なし／リスク高)
ユーザ企業	要求が明確	(要件定義までは) 準委任契約	(要件定義までは) 準委任契約
	要求が漠然	(要件定義までは) 準委任契約	(要件定義～基本設計までは) 準委任契約で仕様の 明確化を図る

システムの効果を評価する責任は業務部門である

要件定義から運用の各段階における責任主管、およびそれぞれの目標を表 2.6 に示す。

表 2.6　要件定義から運用の各段階における責任主管と目標

	要件定義期間	システム開発期間（開発着手）	運用期間（開発完了）
責任主管	業務部門	システム部門	業務部門
目標	システムに求められる要件の定義	システム開発のQCD確保	システム投資の効果評価

システム開発期間は、システム部門が開発の QCD 確保を目標に取り組む。開発が完了し運用に入った段階で経営者からシステムを利用する業務部門に対し、当初狙った効果を捻出するよう指示し、投資効果に対する責任を持たせることが望ましい。そのためにも、開発着手前には、業務部門が運用期間中のシステムに求められる要件を確実に定義することが求められる。また、本稼働後にシステムの適用効果を正しく評価するためには、効果を測定する指標を定め、その目標指標が達成できるようにシステムの要件を定義するとともに、効果を運用中に実測できる仕組みをシステムに組み込むことが必要である。

仕様変更が生じた場合を想定した予算確保

仕様確定は要件定義段階で完璧にできることが理想だが、実態としては難しい。その場合を考慮して、予算をあらかじめ余分に確保しておくことが大切だ。「開発期間中に顧客の要望に変化が生じた」「受注状況が変わる」といった変更を受けて経営方針が変わることは多い。

仕様変更を計画に含めたかどうか（予備予算を確保したかどうか）及び実際に仕様変更が発生したかどうかと、仕様変更が発生した場合の仕様変更費の総開発費に対する割合についての調査結果を表 2.7 に示す。これは、402 のプロジェクトを対象とする JUAS のソフトウェアメトリックス調査 2016[3]によるものである。同表より、仕様変更を見込んで予備予算を確保した 229 プロジェクトでは約 90%で、仕様変更は起こらないと予想し予備予算を確保しなかった 173 プロジェクトでも約 67%で、実際に仕様変更が発生していることが分かる。

なお、この調査では、予備予算を確保した上で実際に仕様変更が発生した 208 プロジェクトでは、ほとんどその予備予算を使いきっていること、予備予算を確保していないにも関わらず仕様変更が発生した 117 プロジェクトでは予算調整に多くの時間を費やしていることも報告されている。

表 2.7　仕様変更の見込みと仕様変更費（総開発費に対する割合）

仕様変更をあらかじめ計画(予算確定)に		仕様変更の発生		合計
		発生した	発生しなかった	
含めた	件数	208	21	229
	割合	90.8%	9.2%	100.0%
	総開発費に対する割合の平均	9.7%		9.7%
含めなかった	件数	117	56	173
	割合	67.6%	32.4%	100.0%
	総開発費に対する割合の平均	8.3%		8.3%
合計	件数	325	77	402
	割合	80.8%	19.2%	100.0%
	総開発費に対する割合の平均	9.2%		9.2%

（出典）日本情報システム・ユーザー協会「ソフトウェアメトリックス調査 2016」[3]をもとに作成

第３章　昨今直面している要件定義課題を解決するための勘どころ

本章では、主だった課題からテーマを選択し、勘どころとして端的にまとめてみた。今回とりあげたテーマは以下の 6 項目である。

また、第 1 章 1.3 節で述べた課題項目との関係も示しており、該当する節にはこれらの課題解決のヒントとなることが記載されている。

(1) 経営や業務に貢献する IT システムの構築（3.1 節）

昨今の IT システムは単なる効率化の道具から経営や業務に直接貢献することが求められている。そのため、要求の価値判断が重要である。

1.3 節の課題との関連：

① ビジネス目的・施策と合致していない
② 手段が先行し、「何のために」が理解できていない
④ 合意形成が取れていない
⑤ 膨らむ要件を抑えきれない
⑩ 業務部門の参画、理解が不十分
⑪ システム部門が要件を引き出せない

(2) 膨らむ要求のコントロール（3.2 節）

要求が膨らむと、要件定義工程が膨らむだけでなく、開発規模や期間などにも影響を与える。いかに要求の膨らみを抑えるかが重要である。

1.3 節の課題との関連：

④ 合意形成が取れていない
⑤ 膨らむ要件を抑えきれない
⑩ 業務部門の参画、理解が不十分
⑪ システム部門が要件を引き出せない

(3) 業務の複雑性を軽減（3.3 節）

変化に柔軟に対応できる IT システムが求められている。システムのアーキテクチャや設計の問題もあるが、要件定義工程では、業務自体の複雑さを少しでも軽減することに着目することも重要である。

1.3 節の課題との関連：

③ 業務の複雑さが増している
⑤ 膨らむ要件を抑えきれない
⑩ 業務部門の参画、理解が不十分
⑪ システム部門が要件を引き出せない

(4) 要件定義工程からの非機能要件定義（3.4 節）

非機能要求は、ややもすると、設計工程に入ってから実施しがちである。昨今は、災害やセキュリティ事故などは経営の重要な関心事となっている。多岐に渡る非機能要求を要件定義工程からどう捌いていくかが重要である。

1.3 節の課題との関連：

⑦　非機能要件を決めきれない

⑩　業務部門の参画、理解が不十分

⑪　システム部門が要件を引き出せない

(5) 多様化するステークホルダとの合意形成（3.5 節）

昨今の要件定義では、関係するステークホルダが多岐に広がっている。それらステークホルダと確実に合意形成を行うことが重要である。

1.3 節の課題との関連：

④　合意形成が取れていない

⑩　業務部門の参画、理解が不十分

⑪　システム部門が要件を引き出せない

(6) 現行業務やシステムの把握（3.6 節）

昨今の IT システム開発では、現行システムが存在していないことが極めて少ない。逆に、業務がシステムに隠蔽されていて見えなくなってきている。いかに現行業務やシステムを可視化し関係者で共有するかが重要である。

1.3 節の課題との関連：

⑨　As-Is の分析、To-Be の可視化が不十分

⑩　業務部門の参画、理解が不十分

⑪　システム部門が要件を引き出せない

3.1 経営や業務に貢献するITシステムの構築

昨今ITシステムは、手作業の機械化すなわち事務作業の効率化から、経営やビジネスに直接貢献することが求められている。ビジネスプロセスが変わらず、箱（ハード）やユーザインタフェースが変わっただけで、多額のIT投資に見合わなかったという経営層の声をよく聞く。

すなわち、要求自体の経営的、業務的価値も要求の品質として捉える必要がある。

一般的に、新規事業を行う際の経営方針やシステム化方針は、構想立案や計画立案で打ち出される。

要件定義工程では、この経営方針やシステム化方針を実現する形で、システム化要件を定義しなければならない。しかし実際のプロジェクトを見ると、手作業の機械化や操作性向上のための要件検討に終始しており、経営方針やシステム化方針が置き去りになっていることがある。

なぜこのようなことが起こるのだろうか。いくつか理由が考えられるが、1つは、担当者が変わるからである。要件定義工程は、開発する側(情報システム部門やベンダ企業)も検討に加わる。開発する側にとっての要件定義工程での最大命題は、作って欲しいもの（仕様）を明確にすることである。したがって、どれくらい企業や業務がよくなるかよりも、どんな機能を作らなければならないかに重きが置かれる。

もう1つの理由は、ステークホルダの価値観が異なるからである。ステークホルダには、経営者、業務部門の長、業務のリーダ、業務の担当者、システムのオペレータ、ITシステムの利用者などさまざま存在する。これらのステークホルダはそれぞれ重要視するポイントが異なる。例えば、ITシステムの利用者であれば、個人視点での作業の改善(ムダ、ムラ、ムリ)に関心が高い。収集した要求も「こんな機能を作って欲しい」「今の機能をこう改善して欲しい」など操作性に関するものが多くなる。これらの要求と、経営方針やシステム化方針とは乖離がある。さまざまな価値観を持ったステークホルダから抽出した要求は、整合を取りながら整理し、最終的に経営方針やシステム化方針を実現する要求を取捨選択していかなければならない。

本節では、経営レベルの要求、業務レベルの要求、現場レベルの要求、IT機能要求などの整合性を確認しながら要求を見極めていく具体的な実施方法を示す。

経営方針やビジネス目的との関係を明確にすべし

【目的と手段の違いを意識せよ】

要求とは、「～したい」と表現できる。「自動集計する機能を作成したい」も要求である。これを作成して何をよくしたいのか。「工数を削減したい」これも要求である。

すなわち、要求には、「～をよくしたい」と「～を実施したい」の２つがある。前者が「目的」であり、後者が手段である。「～を実施したい」だけでなく、「～を作りたい」「～を導入したい」など実際に実行したいという要求である。まずはこの２つ、“目的”と“手段”を区別することが重要である。

【目的を体系化せよ】

経営レベルの目的は、システム開発プロジェクトが開始される前や構想立案時に設定されているのが通常である。しかし、要件定義を終えてみると、経営の意図とは異なるシステムの要件になっている事がある。システムの要件として現場の利用者から要求を吸い上げるため、経営レベルの目的とは異なる、便利で操作性のよいシステムにするための要求（手段としての要求）が沢山あがってくる。これを防止するために、目的を明確にし、手段との関係を押さえる必要がある。

そのためには、まず目的を体系化する。目的には、大きな目的から小さな目的まであり、それらを関連付けする必要がある。企業活動では、経営レベルの目的、業務レベルの目的、現場レベルの目的など立場によって目的のレベルが異なる。これら目的は、トップダウンで降りてくるものと、ボトムアップであがってくるものの両方がある。これらを関連付ける。経営レベルの目的の最上位は、おおむね以下のとおり。

- 売上を上げる
- コストを下げる
- 品質を上げる
- 納期を守る
- 顧客満足度を上げる
- 安全性を上げる
- リスクを下げる
- モチベーションを上げる
- コンプライアンスを守る
- 環境貢献、社会貢献

この経営レベルの目的に対し、業務レベルや現場レベルの目的に落としていく。手作業の機械化の時代の IT システムは、効率化＝コストダウンが主たる目的だった。しかし、一通りシステム化された現在、経営層は、コストダウン以外の価値を求めている。経営や業務の目的との関係を明確にすることが重要になってきている。

【目的と手段を関連付けせよ】

目的と手段を関連付ける。手段にも経営レベルの手段からシステム化手段、人が行う手段などがある。これらがどの目的を達成するためのものかを関連付ける。

関連付けられた目的、手段のイメージを図 3.1 に示す。

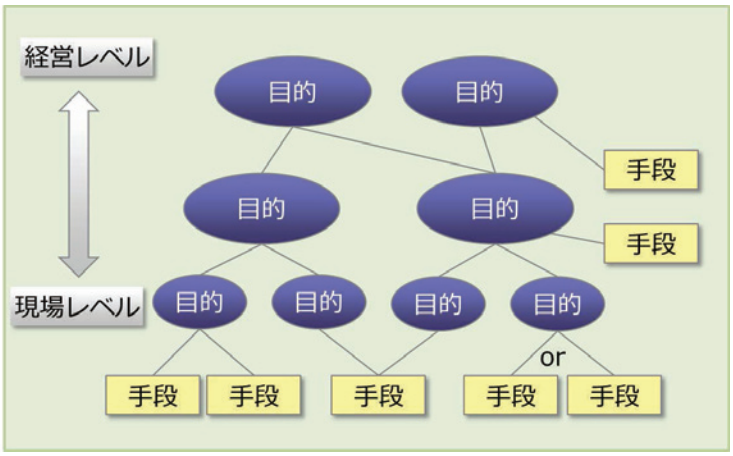


図 3.1　関連付けられた目的、手段のイメージ

上位の目的を達成するためには、下位の手段全てが必要であるといった「and 関係」と、候補が複数ありどれか一つでよいといった「or 関係」がある。どちらかわかるようにしておこう。

【目的の重要性を意識せよ】

民法の契約に関する部分が改正され 2019 年ごろ施行されるようである。その中で、ベンダ企業側は「開発したシステムが、契約の目的通りであったか」の責任を問われるようになる。すなわちシステム構築の目的が業務の効率化であれば、業務が効率化されないと契約の目的を達成したことにならない。前述したように、目的とシステム化手段との関係を明確にし、共有し、合意し、評価をすることが重要になってくる。

曖昧な目的を具体化すべし

【真の目的を明確にせよ】

「コストダウンのために自動集計する機能を作りたい」という要求があったとする。この要求を目的と手段に分けると、「コストダウン」が目的であり、「自動集計する機能を作りたい」が手段ということになる。ここで目的と手段の関係を考える必要がある。

つまり、「目的がコストダウンであれば、それを実現する手段は自動集計しかないか」を問うのだ。その上で「自動集計した場合、どのくらいのコストダウンにつながるか」「自動集計にした場合、従業員ひとり当たりの 1 日の作業時間はどのくらい削減できるか」を数値化する。そして、その金額と数値を見ながら「投資に見合う目的か」を検討する。もし、目的が投資に見合わないのであれば、現場が「自動集計する機能を作りたい」と主張した“本当の”目的を詳らかにする必要がある。例えば、以下のようなやりとりだ。

開発側「なぜ、自動集計しなければならないのですか」
利用者「集計データをいち早く顧客に提供できます」
開発側「早く集計データを提供すると、どんなメリットがあるのですか」
利用者「(顧客の) 機会損失が低減します。ひいては顧客満足度が向上します」

利用者側の目的は「顧客満足度の向上」であったわけだ。
この検討結果を図 3.1 で示した方法で表現したものを図 3.2 に示す。

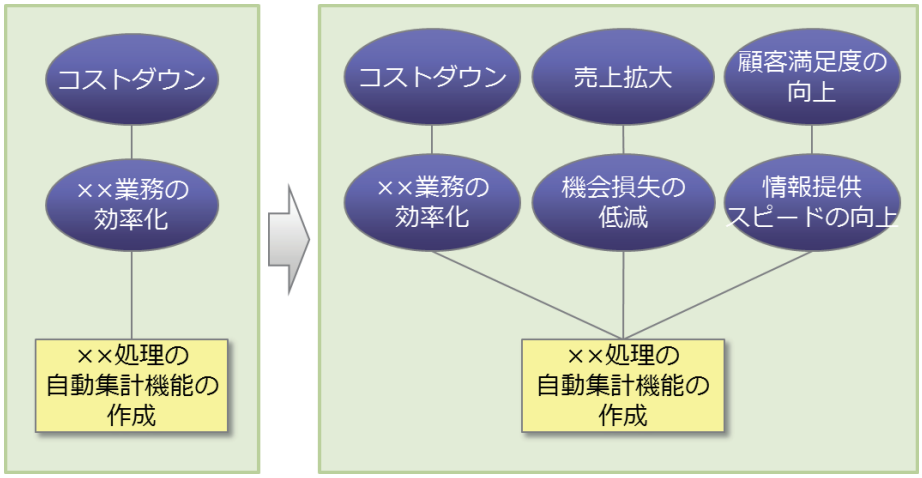


図 3.2　目的を精査した結果のイメージ

【目的と目標の違いを意識せよ】

目的はあくまでゴールであり、目標を設定することが大切になる。目標とは目的を達成するためのマイルストーンである。よって、今はこのくらい（現状値）だが、いつまでに（達成時期）、どのくらいの効果（目標値）を目指すのかを示す指標を設定する必要がある。目的と目標の関係を図 3.3 に示す。

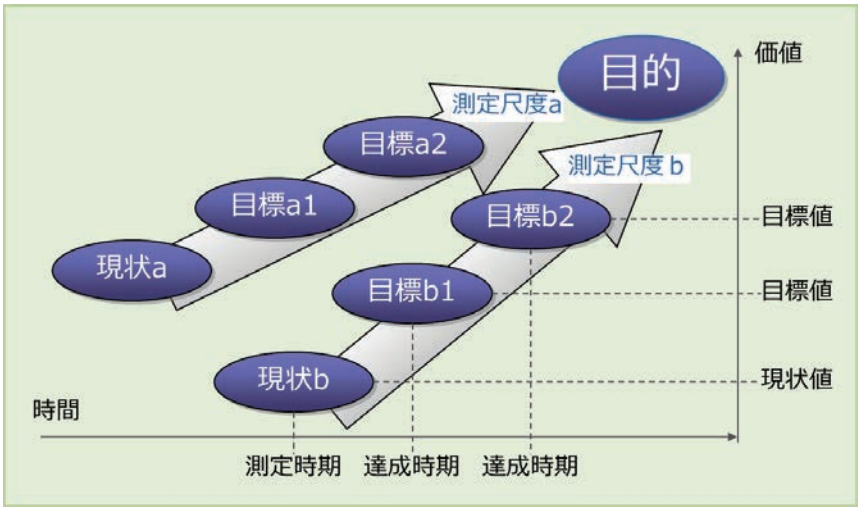


図 3.3　目的と目標の関係

【目的に目標を設定せよ】

現状値や目標値を測る物差しを「測定尺度」と呼ぶ。大切なのは何を測定尺度として設定するかである。例えば、顧客満足度の向上を目的にしたとき、顧客満足度をどのように測定するかが問題になる。顧客にアンケートを出し自ら集計するのもよいだろう。しかし、タイムリーには状況がわからない。そこで、間接的でもよいので、タイムリーに測定できる尺度がないかを検討してみる。「お客様に少しでも早く情報を提供できたら、きっとお客様の満足度は高いはずである、お客様からのクレーム数が減ったら、顧客満足度は高いはずである」と考えることが重要である。漠然としていた「顧客満足度の向上」という目的に対して、具体的な対策が見えるようになる。測定尺度は、間接的でも良いので、定量的に自らが測定可能なものを設定することが重要である。

【稼動時の目標を明確にせよ】

出来上がった IT システムが期待どおりでなかったことはないだろうか。特に昨今の IT システム開発では、早く構築して稼動後徐々に成長させていくという方法がとられることが多い。つまり IT システムの稼動時はあるべき姿には到達していない。稼動時はこのぐらいである、という合意形成ができていないといけない。すなわち期待値のコントロールを行わなければならない。これができていないと、出来上がった IT システムに期待はずれのレッテルが貼られ、利用部門からの信用が失われ、IT システム成長のための協力にも支障をきたす。例えば、稼動時の目標値は、「前日までの情報は翌日提供する（24時間）」で、翌年までには、「半日に 1 回のタイミングで情報を提供する」、と言った具合である。こう考えると、手段も変わってくる。夜間に自動集計処理すればよかったものが、オンライン中（稼働中）の処理での運用方法も検討しなければならなくなる。この関係を図 3.4 に示す。

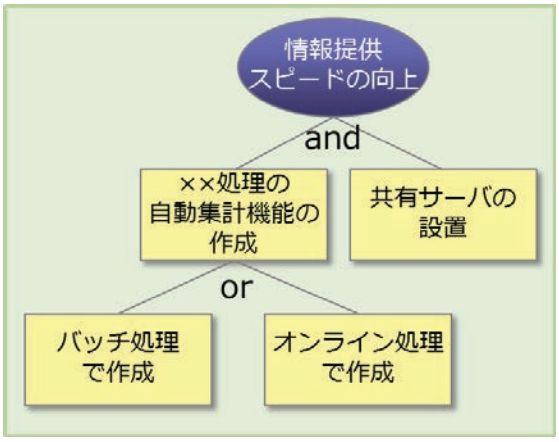


図 3.4　目標を意識した手段

曖昧な手段を具体化すべし

【手段の充分性を検討せよ】

手段はアイデアである。アイデアなので複数考えられる。複数のアイデアの中からどの手段が目的を達成するために最適な方法かを検討する必要がある。

「現状の問題」を仮に「手作業で集計していること」とする。問題を解決するには、自動集計しようとなってしまう。ここで、一度目的に立ち返る。すなわち、問題を解決することでどんな目的を達成したいのかを検討する。仮に「情報提供スピードの向上」が目的ならば、この目的達成に向けてどんな手段が考えられるかを検討することができる。

例えば、図 3.5 のように、「集計前のデータでもよいから公開しては？」「BI ツールを導入し自由に参照できるようにしては？」といったアイデアが出る。大切なのは、目的を達成することである。すなわち、目的を達成するためにこの手段で充分かどうかを検討することが重要である。

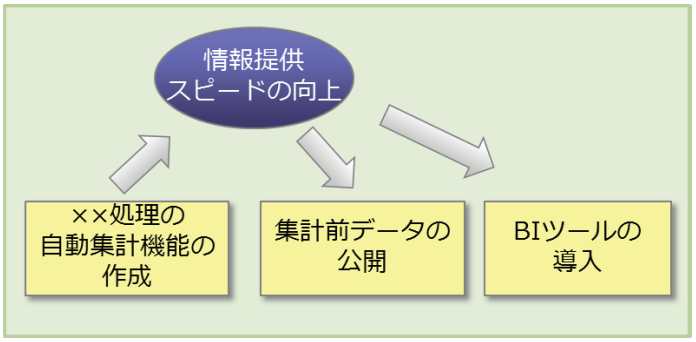


図 3.5　充分性の検討イメージ

【手段を選択するための優先順位を明確にせよ】

手段が複数出てきたら、これら手段のどれを実現すればよいかを判断する必要がある。膨らむ要求を絞り込むためにも重要である。これまでの手順で目的と手段が体系化できており、この手段は目的にどう貢献しているか（妥当性）と、この目的を達成するためにはこの手段で充分か（充分性）が検討できているので、この関係から経営や業務にどれだけ貢献するのかを判断し実現すべき要求を絞り込んでいく。

一般的には、優先順位をつけると言われている、主な判断基準は以下である。

重要性：目的、目標にどれだけ貢献するか（達成効果）
緊急性：急を要するかどうか
費用：実現するのにどれくらい費用が掛かるか
実現性：使用する技術が本当に実現できるのか（人材や実現期間も含め）
新たな問題：この手段を実現したときに発生する新たな問題はないか

上記指標に対し、例えば、重要性であれば「効果が大きい／効果あり／効果が少ない」の 3 段階で判断する。

判断に迷うものは、詳細に調査・分析し正確なデータをもとに判断するという方法が良い。詳細は「3.2 膨らむ要求のコントロール」を参照願う。ここでは、手段を明確にするというところで止めておく。

目的を明確にし、手段を明確にし、関係を押さえることにより、曖昧だった要求がより具体化された。今までの過程での要求の体系化結果を図 3.6 に示す。

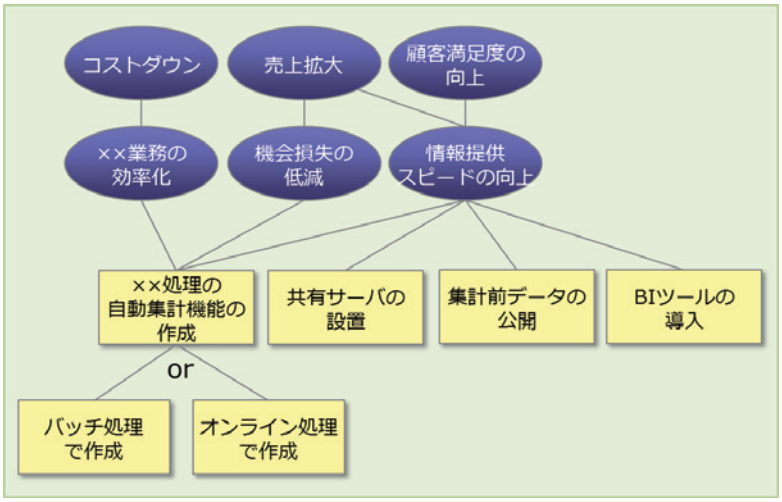


図 3.6　要求の体系化

真の問題を捉えるべし

【要求の源泉を分析せよ】

要求の分析の出発点は図 3.7 に示すように、問題、ニーズ、課題である。問題とは、現状の業務などで起こっている事実である。ニーズとは市場などから求められている要望である。問題から出発するか、ニーズから出発するかは、都度異なってよい。「ニーズを叶えたいが、このような問題がある」「この様な問題があるのでニーズに対応できない」というように、裏腹の関係にあるからだ。昨今では、新たなビジネスやサービスを検討するときは、ニーズだけから発想することもある。問題、ニーズを明確にするということは、要求を定義した理由、すなわち「なぜ（原因）」を明確にすることである。「なぜ」がないと納得感が低く、合意形成しにくい。

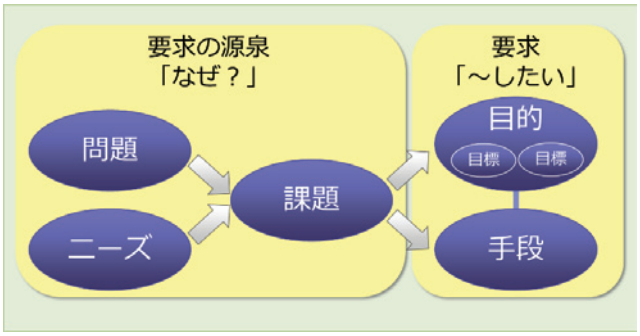


図 3.7　要求分析の基本要素の関係

ニーズや問題が明確になったら課題を設定する。課題とは解決すべきテーマのことである。

この課題（解決テーマ）が設定できれば、目的を明確にして、手段を検討して、現実に即した本質的な解決方法が導かれる。この課題／解決テーマをいかに「適切」に設定できるかがポイントとなる。

【問題と課題の違いを意識せよ】

例えば、問題とは「コストの計画値が 100 の予定であったのに対し、実績値が 200 であり異なった事」であり「事実」である。

一方「課題」とは、この事実に対し「生産性が低いので解決しなければならない」と人の意見・意思が入っている。ここで大切なのは、問題の原因は本当に生産性が低いことなのかである。もしかすると、「生産性は妥当で、見積もり精度が悪い」ことも考えられる。

【『なぜなぜ分析』を行い、真の原因を見極めよ】

真の原因を見極めるためには「なぜなぜ分析」を行う必要がある。

図 3.8 は、「情報系システムが利用されていない」という問題の原因分析の例である。図に示した「探す場所が複数ある」という問題に対処しようとすれば、「サーバ統合」などが考えられる。「情報が多すぎる」に対処しようとすれば、「いかに情報を削減するか」を考えなければならない。両方かもしれない。原因によって対処方法が異なるということがわかる。

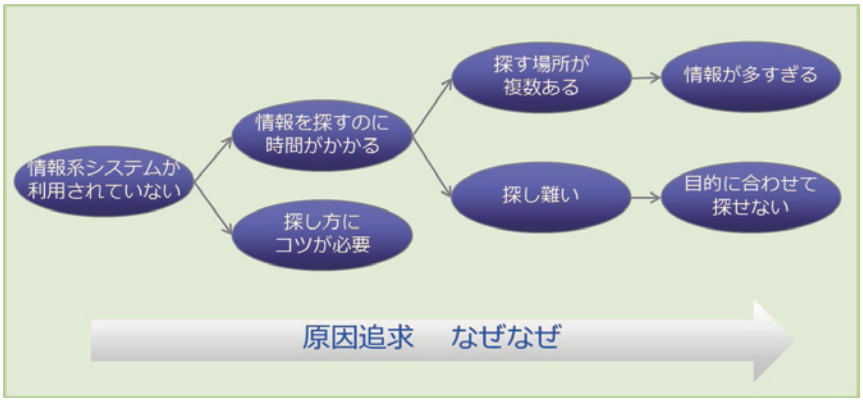


図 3.8　なぜなぜ分析のイメージ

【課題／解決テーマを「適切」に設定せよ】

このような原因分析を経ることにより、本当に解くべき課題を見つけることができる。

全ての原因を解決しようとし要求が膨らむことが無いようにすることが「適切」に設定するポイントの一つである。プロジェクトで本当に解決しなければならない「真の原因」を見極め、何を「課題」として設定するかを分析、検討する。ボトルネックになっている問題、効果の大きな問題などを見極める必要がある。

3.2 膨らむ要求のコントロール

要件定義において要求は膨らむ。昨今様々な仕組みが IT 化され、身の回りにあるシステムから想起されるアイデアが溢れんばかりにある状況で、プロジェクトが開始する。また、実際に要件定義が進むにつれて、現行業務、現行システムに潜在する問題や課題が明確になったり、新業務、新システムのイメージが明確にされたりする。そうなると、さらに様々な要求が出てくるのだ。

一方で工期やコストが有限であることも事実である。そうした中で、必要不可欠な要求を捉えた上で、優先度の低い要求をスコープ外にする必要があるなどの調整が不可欠となる。

加えて、単にシステム部門が業務部門に「優先度が低いので▲▲機能はスコープ外にしました」や「要求が多すぎるので、要求一覧から〇〇件をスコープ外としました」と一方的に伝えたとしても納得性もない。最終的には、多くの要求を実現したい業務部門と要求を絞りたいシステム部門がお互いにわだかまりを抱えたまま譲歩して、要求を落ち着かせる。まさに「2:4:2:3 の法則」を地で行く結果となってしまう。

本節では、要求をいかに漏れなく抽出しながら、かつ納得性をもって絞込んでいくかの具体的方法の例を示す。

要件定義内のフェーズを意識して要求をコントロールすべし

【膨らませるフェーズと絞り込むフェーズを使い分けよ】

要件定義には 2 つのフェーズが存在する。それは、「要求を膨らませるフェーズ」と「要求を絞り込むフェーズ」である。

要件定義開始直後は、業務部門もこれからの業務や次のシステムをどうすべきかが明確でないことが多い。したがって、要求として抽象的なものしか挙がらない。しかし、実現すべき要求を見逃してしまうと、結果として開発工程における仕様変更や稼働後のユーザ満足度の低下につながってしまう。そのため、システム部門としても業務部門から要求を引き出したり、逆に、「～するとよいのではないか？」といった具体的な要求の提案を行ったりする必要がある。その際の要求管理ルールはシンプルにしておく必要がある。

一方で要件定義の後半では、業務やシステムが具体化されていく中で要求も膨れがちになる。そのため、何らかの基準を用いた要求管理ルールに則って、優先度を考慮して要求の採否を判断する必要がある。要求管理ルールも厳格にし、要求として収束に向かわせなければならない。その際、要求の絞り込みを業務部門のみに任せるのではなく、システム部門やベンダが知見や経験を活かし、ファシリテーションをするとよい。

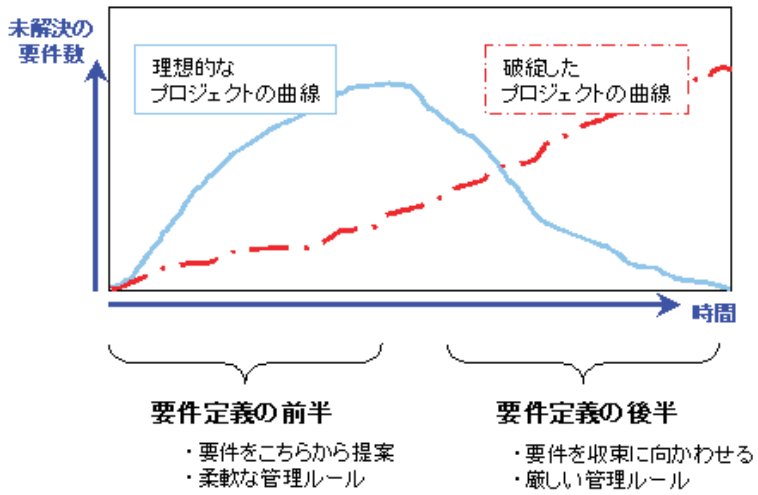


図 3.9 要件定義のフェーズを意識した要求コントロール例

要求の漏れと過剰要求を是正すべし

【要求間の関連を見て充分性を判断せよ】

前述したように、要求は目的を具体化し、手段として詳細化されたものである。そのため、要求間は関連付けられている。この関連性を確認するには、上位から下位を見て、充分な要求が抽出できているかの視点でみることだ。繰り返し確認することで、その充分性を判断できる。

まず、目的の上下関係を確認する。上位の目的を達成するための下位の目的は、抽出されているもので充分か検討する。もし充分でなければ目的を追加する。

次に、目的と手段の間の関係を確認する。対象の目的を達成するために必要な手段が足りているかも検討する。もし足りない場合は新たな手段やアイデアを抽出し、追加する。漠然とした中で手段を抽出するよりも具体的な目的に関連付けたほうが、アイデアを発想しやすくなる。

【要求の価値を検討して妥当性を判断せよ】

逆に、要求間の関連付けを下位から上位に確認することにより、要求の妥当性を検証することができる。

まず、手段から関連づいている目的を確認する。この手段を実施して何がよくなるのか、何に貢献しているのか？を確認する。もし、納得性のある関連でなければ、その手段は誤っているし、目的への貢献度が低いようであれば、目的を達成するための別の手段を検討する必要がある。

次に、目的間の関連付けを確認する。なぜ下位の目的を達成しなければならないのか？の質問に対し、上位の目的が答えとして妥当か確認する。もし、答えとして首をかしげる場合は、問題の本質を見極められていないということであり、今一度深掘りをする必要がある。

【過剰要求を捨てよ】

限られた工期とコストの中ですべての要求を実現することは不可能である。そのため、要求に優先順位をつけて、過剰な要求を捨てる（プロジェクトスコープ外とする）必要がある。優先度付けに使う評価観点の例を以下に挙げる。

- 目的に対する妥当性
  前述の充分性、妥当性を検証していれば、目的に対して妥当でない要求は抽出されていないはずだ。しかし、再度確認し、目的に対して妥当でない要求は優先度を下げるべきである。
- 目的に対する効果
  目的に対する効果も重要な指標である。定量効果、定性効果の観点でそれぞれの要求を評価するとよい。定量効果については、目標の数値だけでなく、その数値の計算方法、算出根拠も必要である。
- 費用
  費用対効果の観点で評価するために、効果だけなく実現にかかる費用、運用時にかかる費用などを評価する必要がある。
- 実現性
  実現できない要求があるということは、裏返せば、要求を実現できない可能性があるというプロジェクトリスクを抱えることになる。したがって、外部動向、内部動向などを踏まえて要求の実現性を評価し、リスクが高ければスコープから外すという選択もプロジェクトを成功に導くための重要なリスク管理である。また、ベンダ企業の知見などを活用し、技術的な観点で実現性を評価することも重要である。
- 必要性
  要求には必要不可欠なものから、あれば嬉しいと担当者が個人的に思っているものまで玉石混交である。したがって、関係者で集まって各要求に対し必要性を確認する必要がある。有名な評価手法として、MoSCoW 分析がある。以下の観点で必要性を評価するものである。
  - ✓ M(Must have)　：必須（これが実現しなければ目的を達成できない要求）
  - ✓ S(Should have)：推奨（目的を達成するために必須ではないが重要な要求である）
  - ✓ C(Could have)　：できれば（あればよいレベルの要求）
  - ✓ W(Won’t have)：不要

優先度を付けただけで終わりではない。現時点で贅沢な要求は、実際に稼働後使ってみて本当に必要であれば実現すればよい。「断捨離」の気持ちで要求を捨てる英断をしていただきたい。

定量化した要求量の中で要求を調整すべし

【要求を定量化せよ】

要求に優先順位をつけて、どこまでも要求を捨ててよいわけではない。関係者間で納得したうえで要求を調整するためには、客観的な情報に基づき、かつ定量的に評価した方がよい。要求を定量化し、プロジェクトに割り当てられたコストの範囲で実現的な要求量を明確にすることが大切だ。

定量的に要求を評価する際に利用できる尺度の例とメリット・デメリットを以下の表3.1に挙げる。

表 3.1 要求の定量化尺度の例とメリット・デメリット

尺度	メリット	デメリット
ファンクションポイント（FP）	・確立したルールに基づいた算出値のため納得性のある数値となる ・粒度にばらつきがない ・生産性指標の参考として外部書籍（ソフトウェアメトリックス調査やソフトウェア開発データ白書など）を活用できる	・計測できる要員を調達、育成する必要がある
機能数	・簡易に計測できる ・ITにうとくても理解しやすい	・粒度にばらつきがあり、精度が低い
画面・帳票数	・簡易に計測できる ・ITにうとくても理解しやすい	・同じ要求でも実現方式により数が異なる

【ベースラインを合意してから要件定義を進めるべし】

たとえ、要求を定量化したとしても、定量化した要求量を使った調整のタイミングも重要であり、考慮が必要である。もし、あなたが業務部門だとして、要件定義の最終局面でシステム部門から「本プロジェクトのコストからすると〇〇FP しか実現できないため、〇〇FP まで減らしてください」と言われたとしたら、素直に「わかりました。減らします」と言えるだろうか？後出しジャンケンで、システム部門の担当者に対し、懐疑的になってしまうのではないだろうか。

これを避けるためには要件定義着手前に本開発で開発する要求量のベースラインを予め合意し、そのベースライン要求量の中でスコープに入れる要求を取捨選択するとよい。その際、プロジェクトのコストと、過去のプロジェクト（もしくは類似プロジェクト）の生産性や外部の生産性を参考値として利用しながら、ベースラインとなる要求量を説明すれば、関係者で納得性をもって合意できるはずだ。

要件定義をしている中で、図 3.10 のように定期的にその時点における要求量を計測すると現時点での状況やその推移から要求量を予測できるようになる。

これにより、業務部門やプロジェクト外の関係者に対しても、プロジェクト状況を可視化することができる。

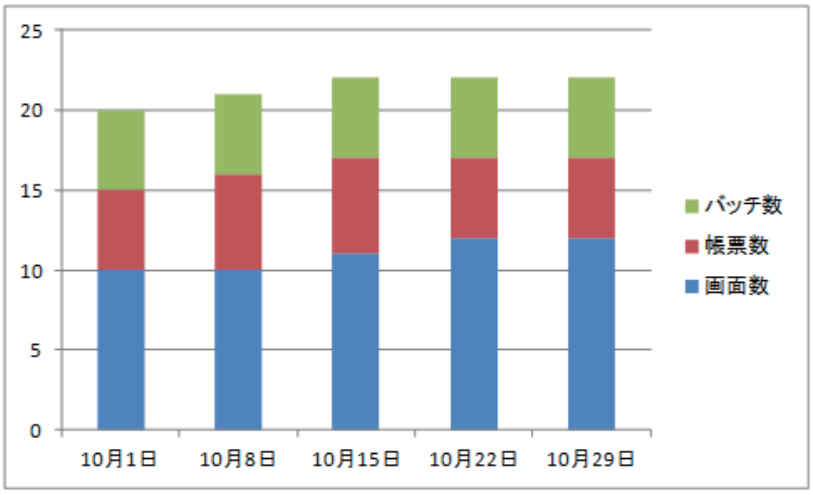


図 3.10 要件量推移の可視化例

3.3 業務の複雑性を軽減

システム企画や構想立案時に、「柔軟で変化にスピーディに対応できるシステムを目指す」というシステム化方針をよく見る。しかし、要件定義書に柔軟なシステムに対する施策として書かれているのは、「SOA、BPM、BRMS などの導入を行う」というものである。これらのアーキテクチャは、システムの変更を容易にできる。要件定義工程では、その導入に向けた準備を実施しているというものだ。いずれも、柔軟なシステム開発のために重要な検討事項であることに違いはない。

一方、システムを複雑、巨大化させている要因には、業務自体が複雑であるということが挙げられる。業務自体を見直すのも要件定義工程の重要なタスクである。しかし、業務の複雑さを積極的に減らす施策が書かれている要件定義書は少ない。システムの柔軟性向上というミッションではシステム部門が中心に検討を進めるからである。もっと業務に踏み込み、業務部門と協力して以下の課題を検討すべきである。

- そもそも業務自体が複雑

  商品の種類や契約の仕方などの業務処理のパターンがたくさんある。その中には、めったに起こらないものも少なくない。一般的な企業では、上位 2 割のパターンで業務量の 8 割をカバーできているという。似て非なる業務も多く存在している。もっと業務自体を標準化し、シンプルにし、システムの大規模化、複雑化を根本的に削減することも考えなければならない。

- 昔からのしがらみで、直せない

  例えば、EDI 区分が昔からあり、今の業務では使用していないが、削除するとシステムがエラーを起こすことがあるかもしれないので残しておこうと決めてしまう。システムが停止してしまうのは問題だか、このようなゴミと思われるものは削除するという英断も必要だ。

- 贅沢システムが故の硬直化

  システムでは、なかなか実装できそうもない、人間だからできるような処理、例えば、リスク判断を行うため、ノウハウを可視化し、必要な情報を大量に入力し、システムにアラートを出させる、といったような機能も実装する。これは、すばらしいシステムかもしれないが、本当にシステム化することがよいことなのか考える必要がある。システムが大規模、複雑になり、修正も容易でなくなるかもしれない。もしかすると人手で対応した方がビジネスの変化にスピーディに対応できるかもしれない。

このように、システムのアーキテクチャ以前に業務自体を見直すことも重要であり、To-Be 業務を創り上げるのは、要件定義工程の重要なミッションのひとつである。

一昔前のシステムに対する価値観は、いかにコンピュータに仕事をさせるかであり、滅多に起こらないことでも、本来人でしかできそうに無い固有のノウハウまでもシステムで対応するように頑張った。結果として、システムは肥大化、複雑化、属人化して硬直化極まりない状態になった。現在の価値観は、人に任せる部分はシステム化せず、いかにシンプルにし、ビジネススピードに対応できるようにするかが主流である。一旦システムから機能を大幅に削減し、「何か起こったら、人手で対応し、対策を考える」と英断した CIO もいた。

管理対象のバリエーションを整理すべし

業務は、管理対象の種類の組み合わせにより複雑になっている。

管理対象とは「顧客」「商品」「オーダー」などである。管理対象の種類とは、例えば「顧客」には「一般顧客」と「得意先」があり、商品には「在庫品」「オーダーメイド品」「サービス商品」があり、オーダーには「買い取り」「レンタル」があるといった具合である。そして業務ルールはこれらバリエーションの組み合わせで決まっていることが多い。「一般客にはオーダーメイド品は注文できない」し、「オーダーメイド品は必ず買い取りでないといけない」と言ったルールである。

ここでは、このような複雑さの元になっている管理対象の種類の組合せを可視化し、整理し、バリエーションを低減する勘どころを紹介したい。

【管理対象の種類を整理せよ】

整理の基本は、「同じか」「違うか」「部分か」である。

図 3.11a の例では、「ユーザ」と「顧客」は別であり、「ユーザ」は「利用者」と同じであり、「販売先」は「顧客」と言われるが「仕入先」は「顧客」とは言わない。「販売先」には、契約の仕方によって、「得意先」がある。「得意先」でない「販売先」を指す言い方を持っていない、ということを表している。

この整理は辞書を引いてもできない。図 3.11b の例では、「仕入先」も「販売先」も取引のある企業は全部「顧客」と呼んでいることを表している。図 3.11a と図 3.11b は異なった定義になっているが、どちらが正解とは言えない。企業の文化によって企業ごとに異なるからである。

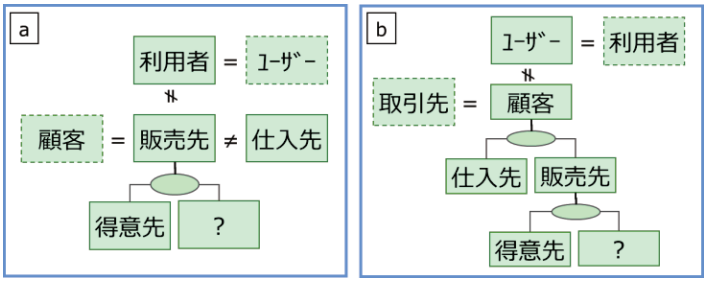


図 3.11　管理対象の分類図の例

【漏れなくダブりなく整理せよ】

ミッシー（MECE：Mutually Exclusive and Collective Exhaustive）になるように分類する。ミッシーに分類することは、相互に重なりがなく、全部集めて漏れがない状態にすることをいう。図 3.12a の例では、「取引先」を「大口」と「小口」に分類したが、実際には他の分類もあった。図 3.12b は「取引先」を「仕入先」と「販売先」と「大口」に分類したが、「大口」は「取引先」にも「仕入先」にも両方含まれていた。このような状態はミッシーになっていない。これらを正しく描くと図 3.12 c のようになる。

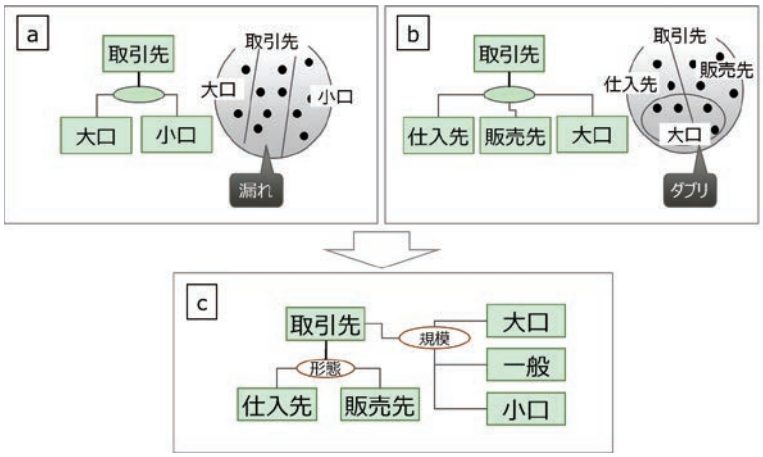


図 3.12　ミッシーに分類するイメージ

【名前のない分類に名前を付けよ】

図 3.12a の例は前述したとおり、「販売先」には、契約の仕方によって、「得意先」があり、「得意先」でない「販売先」を指す言い方を持っていないことを表している。この「「得意先」でない「販売先」」に明確に名前をつけることが重要である。例えば「一般販売先」というように名前を付ける。こうすることにより、明確に使い分けて、仕様書を記述できるようになる。

【用語の定義として共通認識せよ】

このような整理は、用語の定義としても有効である。特に、同じ意味のものは、どちらを仕様書上で使うかを明確にしておくとよい。図 3.12a の例でいうと、「顧客」という用語を使わず「販売先」という用語で統一する、同様に「ユーザ」という用語を使わず「利用者」という用語で統一する、といった具合である。使わないと決めた「顧客」や「ユーザ」は禁止用語とし、仕様書の記述に使わないよう徹底することも、仕様書の曖昧さを低減する上で有効である。

【企業文化に依存するので注意せよ】

図 3.11 の a，b の例のように、管理対象の整理は企業文化に依存するので、結果は企業ごとに異なる。思い込みで整理しないように、確認を取ることが重要であり、結果は全員で共有する必要がある。

このような可視化、整理ができたらバリエーションを削減できないか検討する。

【分類の廃止を検討せよ】

図 3.13 のように、現在使用されていないバリエーション、今後使用しないバリエーションを洗い出し、削除を検討する。

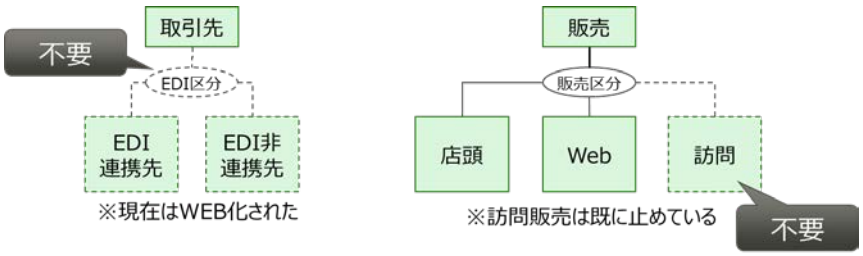


図 3.13　不要な分類を廃止する

【分類の統合を検討せよ】

他の部署や、他のシステムと異なる分類を使っており統一されていない場合がある。分類の統合が可能か検討し統合する。

分類の統合を追求していくと、全社統合マスタを実現しなければならなくなる。これは、大変な労力を要する。そこまでできないときは、範囲を限定し統合を検討する。

業務のバリエーションを減らすための整理として記載したが、このような整理は以下のような効果も生む。

- 業務用語の理解促進、誤解低減
- 業務仕様のモレ防止（バリエーションが明確になっている）
- バリエーションの低減による、システムのシンプル化・柔軟性の向上

業務のバリエーションを減らすべし

業務プロセスも整理する。

似て非なる業務を標準化したい、幹となる業務と枝葉である業務を切り分けたいといったニーズも多くある。これは、業務がシンプルになるだけでなく、業務プロセスの重複の低減や効率の向上にもつながる。前述の管理対象の種類の組み合わせ等により、業務にはたくさんのバリエーションが存在することになるが、このパターンの上位 20%で実際に発生する業務の 80%をカバーできるといわれている。しかし、システム開発では利用する頻度が少ないものでも独立した業務として存在する限りは同等に開発しなければならない。そのまま開発すると、上位 20%のパターンよりも、残りの 80%の例外系のパターンの方が品質確保や開発難易度が高くなるかもしれない。したがって、業務プロセスを標準化したりバリエーションを減らしたりすることは、システムのシンプル化、規模削減に大いに効果がある。

ここでは、業務プロセスや業務パターンを整理し、複雑さを低減する勘どころを述べる。業務プロセスを見直すときに常に意識しておくとよい。

【業務プロセスの廃止を検討せよ】(図 3.14a)

業務をシンプルにするには、不要なプロセスを無くすことが効果的である。「これは本当に必要なのか？要らないのでは？」という観点で検討する。システム化されているプロセスを人手による運用に戻すということも検討する。

【業務プロセスの集約を検討せよ】(図 3.14b)

似て非なる業務プロセスを共通化したり片寄せしたりして一つにする。「同じルールにできないのか？」という観点で検討する。

【業務プロセスの連結を検討せよ】(図 3.14c)

2 つのプロセスを一つにする。「異なる人が実施しているプロセスを同じ人で同時にできないのか？」という観点で検討する。プロセスが分かれているということは、各プロセスを別の人が実行しても良いということである。人が異なれば、その間にインタフェース（業務間インタフェース）が存在するということであり、このインタフェースは、効率を落としたり、コミュニケーションギャップを生みだしたりする。これをなくそうという観点で検討するものである。

【業務プロセスの並行化を検討せよ】（図 3.14d）

順次行われていた業務プロセスを並行で実施できないか検討する。業務プロセスの順序は環境の変化などによって変わることがある。順序が変わっても柔軟に対応するためには、順序の依存関係がないように見直しておくことも必要である。

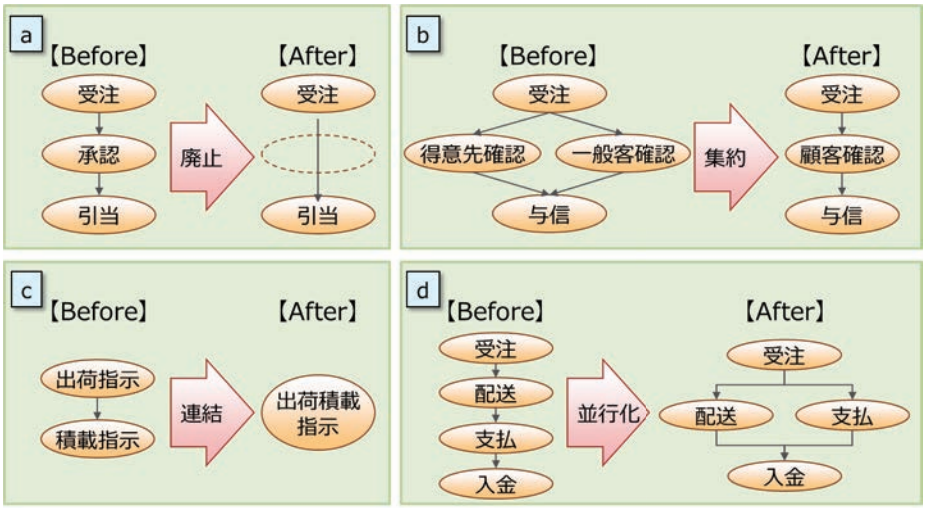


図 3.14　業務プロセスの見直しの観点

【一連の業務処理のパターンの削減を検討せよ】

業務フローを作成した後に、どのような経路を辿るパターンがあるのかを明確にすることは、業務を検証する上でも、運用テストのシナリオを作成する上でも有効である。この一連の業務処理のパターンを削減できないか検討する。

どのような時にどのプロセスを辿るのかをトレースしていく。そうすると、似たようなパターンが見えてきて、前述した業務プロセスの見直しの観点で見直しを行い、パターンを低減していくことができる。そのパターンの業務上の発生頻度や重要度などからパターンそのものを止められないかを検討する。上位 20%のパターンで業務の 80%がカバーできるといわれているので、滅多に起こらないパターンの統合や、業務運用によるカバーを検討する。この分析は業務全体を鳥瞰し、業務の大きな流れを 1 つのパターンとして整理している。そのため、パターンの統合や削減による業務の標準化やシンプル化の効果が高い。

これらの分析は、業務フロー上でトレースする方法や、表で整理する方法がある。業務フローが大量にある場合は、図 3.15 のような表での整理方法が有効である。

シナリオ名	頻度	商品					顧客						オーダー形態			業務の流れ								
		有形	サービス		単品	セット	代理店	法人		個人	国内	海外	一般	レンタル	無償	顧客確認	見積り	在庫引当	出荷指示	直送	出荷	配送	請求	・・・
			定額	従量				大口	小口															
パターン1	30	●			●		●				●		●			①		②	③		④	⑤	⑥	
パターン2	20	●			●			●			●		●			①	②	③	④		⑤	⑥	⑦	
パターン3	2	●			●				●		●	●	●			①		③	④	⑤		⑥	②	
パターン4	1	●			●					●	●	●	●			①		③	④	⑤		⑥	②	
・・・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・

図 3.15　一連の業務処理のパターンの可視化

業務パッケージを適切に利用すべし

業務パッケージの導入は、業務の見直し、標準化には一番手っ取り早い方法である。しかし、業務パッケージ導入の成功率は、低いと言われており、導入には注意が必要である。

欧米では、ビジネスプロセスを大きく見直すためのパッケージの導入が当たり前である。欧米では、現場からの業務改善よりもトップダウンで業務を変えていく場合が多い。その方法としてパッケージ導入が採用される。一方、日本は現場を巻き込んだ改善が多い。ここに注意が必要である。以下に注意点を記す。

【カスタマイズ量に注意せよ】

失敗の原因は、パッケージのカスタマイズ量が多くなることである。カスタマイズが20%以上あると新規開発と変わらないと言われている。

【業務をパッケージに合わせるという意識を強く持て】

業務をパッケージに合わせると明確に宣言できているかが重要である。パッケージの導入目的が単なるITシステム開発の効率化のためのソース流用であると、カスタマイズ量が増え、プロジェクトは失敗する。業務をパッケージに合わせるという意識をプロジェクトメンバに周知させておかないと失敗する。

【業務パッケージの導入に対する現場の反対をコントロールせよ】

現場は、業務をパッケージに合わせることに反対する。慣れ親しんだ業務が変わる、退化するかも知れない、大幅に変えると調整などが大変になるなどの思いからである。結果、今まであった機能が無くなる事に反対し、あれも必要、これも必要となり、カスタマイズ量が増え失敗する。

【経営トップが参画せよ】

業務をパッケージに合わせるという目的での業務パッケージの導入は、今までの業務を変えるという意志表示でもある。新しいやり方に業務を変えるわけなので、経営トップが積極的に参画するか、プロジェクト責任者に業務変更権限を明確に与えないと成功しない。

【パッケージ導入の是非を再度判断せよ】

パッケージ導入時にはFit&Gap分析を行い、適合性を判断していても、要件定義が進むにつれて、上記のような理由で、カスタマイズ量が増える。しかし、パッケージの導入が決定しているので無理にカスタマイズするプロジェクトも多い。要件定義工程で新たな要求が明確になった時点で、もう一度Fit&Gap分析を行い、パッケージ導入が適切であるかを判断し直すことが重要である。

注意は必要だが、業務を大幅に見直したい、標準化したいときには、業務パッケージの導入は有効な手段である。

3.4 要件定義工程からの非機能要件定義

第 2 章でも述べたように、非機能要件の重要さが増してきており、経営や業務の重要な関心ごととなっている。今まで非機能要件は、技術的な要素が強く、システムの専門家が設計工程に入って苦労しながら決めていることが多かった。しかし、要件定義工程で、経営層、業務部門、システム部門やベンダ企業が協力して要件として決めなければならなくなってきた。

非機能要件の定義で役立つのは、IPA が提示している「非機能要求グレード」である[6]。非機能要求グレードは、非機能に対する要件定義を 236 種類の項目に対し、目標値／指標値（メトリクス）を設定することで行うものである。示された項目のメトリクスを決めればよいので、非常にシンプルで分かりやすい。経験の少ない人でも、漏れなく実施しようとしたときに役立つ。基本は、この非機能要求グレードを有効利用していただきたい。

しかし、非機能要求グレードを実際に利用しようとしたときには、適用上の難しさがある。技術的要素が強く、設定項目も多いので、いかに経営部門や業務部門などと合意をとりながら、効率的に進めていくかを考えなければならない。

本節では、非機能要求グレードをベースにした適用上の工夫ポイントを述べる。

工夫ポイントの一つは、要件定義工程で経営層、業務部門、システム部門やベンダ企業などがいかに協調して合意形成していくかである。方針を確認し、経営層や業務部門から引き出すべき要求を明確にし、そこからメトリクスにつなげていくという工夫が必要である。

二つ目は、非機能要求の曖昧性や矛盾を無くすことである。性能の要求を聞くと、「早いに越したことはない」などと言われる。とかく非機能要求はオーバースペックになりがちだ。また、セキュリティ強化のために暗号化を適用すれば、性能が低下する。こうしたトレードオフ関係、優先順位関係に決着をつけなければならないものもある。

非機能要求グレードを活用すべし

まずは、IPA が提示している非機能要求グレードの活用を検討していただきたい。

非機能要求グレードのイメージを図 3.16 に示す。

（注）詳細は IPA/SEC のホームページを参照[6]

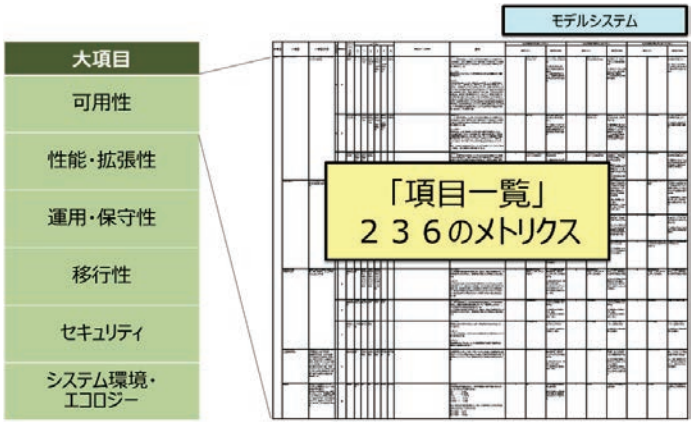


図 3.16　非機能要求グレードのイメージ

非機能要求グレードは、可用性、性能・拡張性、運用・保守性、移行性、セキュリティ、システム環境・エコロジーの 6 領域に対し、要件定義工程で明確にすべき 236 のメトリクスを提示している。

例えば、運用時間（通常）というメトリクスでは、

レベル 0：規定無し

レベル 1：定時内（9 時～17 時）

レベル 2：夜間のみ停止（9 時～21 時）

レベル 3：1 時間程度の停止有り（9 時～翌朝 8 時）

レベル 4：若干の停止有り（9 時～翌朝 8 時 55 分）

レベル 5：24 時間無停止

というガイドを示し、決定を導いている。

非機能要求グレードは、ただ単に 236 もの項目を定義せよではなく、以下に示すような扱い方を提示してくれているのが特徴である。

① モデルシステムの参照

実現したいシステムのイメージに最も近いモデルシステムを選ぶことにより設定したい項目に対するメトリクスの一般値を引用できるようになっている。

② 段階的詳細化

樹系図が用意されている。これは、検討の順番や重要項目を示してくれている。

重要項目では、品質やコストに大きな影響を与えるものを明示している。

基本的な情報を要求として引き出すべし

工夫ポイントの一つ目は、経営層、業務部門、システム部門やベンダ企業などがいかに協調して合意形成していくかである。

機能の要件定義は、利用部門が主体で実施することが基本であるが、非機能の要件定義は、システム部門が決めるメトリクスもある。協調して合意形成していくには、経営層や業務部門に関心の強いものを選別する必要がある。そうすることにより、どのステークホルダから非機能要求を抽出し合意形成すべきかが明確になる。また、非機能要求は、トレードオフ関係になるものがあるので、どちらを優先するかの判断をしなければならない。コストだけではなく、ビジネス目的達成を考慮した判断をするためにも以下の選別は役立つ。

【非機能グレードの項目をステークホルダ別に分けよ】

合意形成しなければならないステークホルダによって、非機能要件の項目を振り分ける。まず、ステークホルダによって大きく 3 つの領域を設定する。

- ビジネス領域：経営層や利用部門の関心の高い領域。ビジネスとしての要求として捉えるものであり、判断に困ったときのよりどころとなる重要な要求。
- ソリューション領域：利用部門や運用部門の関心が高い領域。IT システムが提供するサービスに対する要求。
- テクニカル領域：システム部門が検討し提示していく IT 専門技術要素の強い領域。技術や構成要素に対する要求。

次に、各々の領域に非機能要求の項目を振り分ける。項目の種類を振り分けた例を表3.2に示す。

表 3.2　非機能要求の振分け例

領域	大項目	中項目	備考
ビジネス領域	ビジネス要求/基本情報	業務特性、サービス時間、業務継続、業務連携、業務量、運用方針、内部統制、セキュリティポリシー	非機能要求グレードに追加
ソリューション領域	可用性	業務復旧	
	性能・拡張性	拡張指針、性能目標、性能検証指針	
	運用・保守性	運用監視、運用自動化、システム連携、ライフサイクル、サポート体制、システム保守、システム復旧、システム運用環境、システム運用管理	
	移行性	移行計画、システム移行	
	セキュリティ	情報セキュリティ	
	システム環境・エコロジー	システム環境、設備規格	
	ユーザビリティ	ユーザビリティ目標、ユーザビリティ指針	非機能要求グレードに追加
テクニカル領域	可用性	資源冗長化、データ保護対策	
	性能・拡張性	資源拡張性、帯域保証	
	運用・保守性	バッチ運用指針	
	セキュリティ	セキュリティ対策	
	システム環境・エコロジー	設備環境	
	ソフトウェア保守・移植性	ソフトウェア保守・移植性	非機能要求グレードに追加

（出典）富士通「要件定義手法 Tri-shaping マニュアル」[7]

このように、同じ可用性でも、合意形成しなければならないステークホルダが異なることがわかる。また、非機能要求につながる、ビジネス要求／基本情報も追加している。さらに、昨今非機能として注目されている、ユーザビリティやソフトウェア保守性・移植性なども追加されている。

【聞きだす要求を明確にする】

3つの領域に分けても、236項目を提示しメトリクスを決めていくのは困難である。基本的な「要求」を抽出するところから出発するのがよい。ここでは、最終的に確定したメトリクスなどを「要件」と呼び、それの元となる段階のものを「要求」と呼ぶことにする。各領域に要求抽出シートを用意しておくとよい。要求抽出シートとは、非機能のメトリクスを決めるにあたっての基本情報や重要な意思表示を引き出すためのものである。例えば、「セキュリティ」の「リスク分析範囲」では、個人情報の流出が検知できる対策を講じるかどうかの意思（要求）を聞き出すに留める。ソリューション領域も同様に、例えば、情報セキュリティに対しては、第三者による診断を実施したいかどうかの意思（要求）を聞き出すに留める。その後ネットワーク診断、Web診断、DB診断はどうするかというようにメトリクス（要件）として具体化していく。要求を抽出することにより、その中で、優先順位や矛盾やトレードオフ関係を検討できるようになり、236項目ものメトリクスを直接決めていくことより効率的に進めていけるようになる。なお、サービス利用者数等、非機能要求のメトリクスそのものでもある基本情報もビジネス領域に位置づけておりここで抽出する。

このような「要求」レベルで確認するためのシートを準備しておくとよい。

表3.3にビジネス領域の要求抽出シートの例を示す。

表3.3　ビジネス領域の非機能要求抽出シートの例

要求の種類	確認項目
業務特性／サービス時間	サービス提供先（企業内、企業間取引、コンシューマ向けなど）
	サービス提供範囲（特定地域、国内、国外）
	サービス提供時間（通常・特定日　24時間365日）
	サービス特性（繁忙期・閑散期、月間、年間、年度）
	サービス利用者数
	サービス提供拠点数、利用端末数（人/端末）
業務継続	サービス継続範囲（対象業務範囲、計画停止有無、代替業務運用範囲）
	業務停止の許容度（単一障害、多重障害）
業務連携	提携先企業とのシステム連携有無
	提携先企業とのシステム連携方法や条件
業務量	サービス提供業務・取扱量
	サービス拡張指針（利用者数、対象業務）
運用方針／内部統制	サービス運用指針（運用範囲、場所、内部統制有無）
セキュリティ	リスク分析範囲、順守すべき社内規程・ルール・法令・ガイドライン等の有無

要求の抽出は、ヒアリングにより行うことが基本だ。ただし、セキュリティや業務継続などは、システム開発プロジェクト以前に企業として方針が設定されている。方針書やガイド等を入手し理解し、そこから抽出することも必要になる。

矛盾、トレードオフを解消すべし

工夫のポイントの二つ目は、いかに非機能要求の曖昧性や矛盾を無くし、トレードオフ関係、優先順位関係に決着をつけるかである。

非機能要求はオーバースペックになりがちである。技術や予算や期間や現状のレベルなどを考慮せず、「処理は早いに越したことはない」「セキュリティレベルは高いに越したことはない」となってしまう。そのスペックが無いと本当に業務がまわらないのか、判断に苦しむことも多い。さらに非機能要求はトレードオフ関係になることが多い。セキュリティ強化のために暗号化を適用しようとしたら性能が低下するといった具合である。

以下にオーバースペックやトレードオフを解消するための観点を挙げる。

【制約や前提条件を明確にせよ】

法令、商慣習、予算や特定技術採用、技術人材、再利用素材などビジネス上の制約や技術的な制約を明確にしておく。また、未確定の要素があれば仮定し前提条件として設定して検討する。

【現状を正確に把握せよ】

現状からよくしたいというのが要求のベースにあり、どのくらいよくなったかがシステム化の成果である。まず現在の状況を正確に把握することが重要である。一気にジャンプし過ぎて現場がついていけないという事も注意しないといけない。段階的なステップアップも検討要素にいれないといけない。また、逆もある。現行と同じでよいと言われたのに、現行より悪くなることもある。Web 化したら性能が悪くなるというのは、システム部門としてはしかたないと思うかもしれないが、業務部門にとっては現状より悪くなることは許せないはずである。現状を把握して判断することが重要である。

【優先順位を判断せよ】

制約や前提条件、現在の状況などを鑑みて優先順位を決める。ビジネス領域の要求が、優先順位が高い要求とみるとよい。昨今の非機能要求はビジネス要求達成のための重要なファクターになってきている事は前述のとおりである。

【要求のレベルでトレードオフ関係を判断せよ】

非機能要求の依存関係を明らかにし、依存関係があるものについて、矛盾した要件がないか確認する必要がある。非機能要求間でトレードオフがある場合は、要求度合いや優先順位付けをもとに、どちらの要求を優先して採用するかを判断する。

【一律でメトリクスを決めるな】

性能などでは、目標値が 1 秒であるが、遵守率 90%というような決め方や、ある処理だけは 3 秒を目標とするなど、条件に応じたメトリクスを決める必要がある。一律で決めないことである。業務復旧やセキュリティなども、完全社内に閉じたシステムとか基幹システムでないものまで一律で高い水準のものを求めないことである。システムの種類によっては贅沢な機能になってしまう。

【実際に体感してもらえ】

性能などは、余裕があれば体感してもらう。現行業務のレスポンスが 3 秒で、目標値が 1 秒であったとしても、実際には 2 秒で充分速いと感じることがある。利用者に体感してもらった上で合意形成をとるということも効果がある。難易度の高い方式を検討するよりリスクは下がる。

関心が高い非機能要求設定に関するポイント

(1) 業務継続性

BCP（事業継続計画）とは、企業が自然災害、事故、テロ等の予期せぬ緊急事態に遭遇した場合に、重要業務に対する被害を最小限にとどめ、最低限の事業活動の継続、早期復旧を行うために事前に策定する行動計画である。大きな震災やテロなどが身近にあるという認識が強まり関心を高めている。経営的なダメージも大きく、重要課題として捉えている企業が増えているということだ。電力や通信、金融など、社会的影響が大きい業種だけでなく、製造業などでも業務停止による影響を鑑み設定する企業が増えている。国は中央防災会議において、2015 年度までに、大企業の全てと中小企業の半数以上が設定という目標を出している。

これらの要求は、非機能要求グレードの「可用性」「継続性」の領域の中に「業務継続性」／「目標復旧水準」などとして位置づけられている。

【長時間ダウン時を考慮せよ】

昨今は、システム障害などによるシステムのダウン時にどうするかを決めるだけでは大きな災害に対応できないことがわかってきた。大きな災害の発生を考慮するなら、1 週間や 1 ヶ月という長期間ダウン時にどうするかまで検討しておかなければならない。

【運用を考慮せよ】

バックアップシステムを遠隔地に配備したとしてもそこで業務を遂行するための運用体制がとれなくては業務が停止したままになる。運用方法や体制も検討しておく必要がある。

【訓練せよ】

運用方法を立案しても、実際機能しない場合がある。人が中心となる場合は訓練を実施し運用の見直しも同時に行う必要がある。

(2) セキュリティ

インターネットの普及とともに、情報の盗難やコンピュータシステムの破壊といった犯罪が増えている。システムが止まるだけでなく、情報漏えいなどの事件を起こすと企業としての大きな責任問題になる。経営的なダメージも大きく、企業としての重要課題である。

これらの要求は、非機能要求グレードの「セキュリティ」という領域の中に沢山記載されている。セキュリティは非機能要求として従来から認識の強いものだが、昨今は、セキュリティ犯罪が多様化していて、対応に苦慮しているのが現実だ。

【守り以外も考慮せよ】

ガード、検知、ログ／証跡、分析、対処、教育といった、プロセスの全体を見通した設計を行わなければならない。守りきれないことも想定した対処方法を検討しなければならない。

【対組織だけでなく、対個人も考慮せよ】

組織として仕組みを作るだけでは防げない。アクセス権のある社内の人材が悪意を持って攻撃すれば守りきれない。個人の教育やモラルの向上などの取り組みも必要である。

【IPA の情報セキュリティ読本などを参考にせよ】

IPA セキュリティセンターでは、情報セキュリティ強化に対する対策として、「情報セキュリティ読本[8]」や「情報セキュリティ教本[9]」などを出版している他、ホームページで常に最新情報を発信している。是非読んで参考にしていただきたい。

(3) ユーザビリティ

最近、ユーザビリティに対する関心が高まっている。ユーザビリティとは、「使いやすさ」である。IT システムの画面でいうと、色や配置の工夫等による見やすさ、操作性などである。以前のシステム開発は目的の機能を実現することが最優先で、使い方は、人が訓練して習得すればよいという発想であった。しかし、インターネットの普及やスマートホンなどの普及にともない、IT システムが直接ユーザと触れることが多くなってきたことにより、機能だけでユーザビリティを考慮しなければ、ユーザに見放されてもしかたない時代になっている。さらに、使い易さだけでも競争優位を築けない時代になってきている。「楽しさ、驚き、心地よさ」という UX（User Experience）も脚光を浴びてきている。

これらの要求は、残念ながら現行の非機能要求グレードには記載されていない。非機能要求グレードの基は「製品品質」であり、ISO/IEC25010[10]における「品質モデル」の中で定義されている。ISO/IEC25010 では、品質モデルとして「製品品質」だけでなく「利用時品質」が追加された。「利用時品質」とは、「有効性」、「効率性」、「満足性（実用性、快感性、快適性）」などである。製品（システムやソフトウェア）自体の品質だけでなく、それを利用したときの品質も意識する必要があるといっている。本ガイドではこれら利用時品質の向上方法については割愛する。

【ユーザビリティ、UX を考慮せよ】

ユーザビリティの設計方法や UX 要求を獲得する手法、人間中心設計などは、一般書物として出版されていたり、セミナーが開催されたりしているので、そちらを参考にするとよい。

(4) 性能

性能への関心は未だ強い。

理由の一つは、未だ性能に関する障害や要求との乖離が後を絶たないからである。

二つ目の理由は、上述のユーザビリティの高まりにともない、性能に関する要求も高まっていることである。例えば「今まで、3 画面で遷移していたものを、タブやスクロールを駆使し、1 画面に集約して欲しい」といった要求である。使いやすさは向上するが、性能が問題となる。ユーザビリティと性能はトレードオフになることが多いからである。

① 性能に関する障害や要求性能との乖離による不具合への対策

【条件による違いを考慮せよ】

画面レスポンスは「一律 2 秒以内」と決め込まない。性能は条件によって異なる。利用者数やデータ量などによって異なると共に、一日の時間帯や一年の時期などによっても異なる。詳細は非機能要求グレードの項目を参照願う。ピーク時を見つけ出し、そこの検証と合意形成を行うことが重要である。

【将来の状態を予測せよ】

上記のようなデータを見極めるとき、現状のデータだけでは、問題がある。将来まで使用するシステムとして、稼動時や 1 年後、2 年後と言ったデータで判断しなければならない。将来のデータを見極めるためにも、障害を起こさないためにも、日ごろの運用で、定点観測を含め、定期的にデータの測定を実施し、システムの利用状況の変化を先読みしておくことが大切である。

【想定外を検討せよ】

ピーク時などの最大量を想定して検討するが、想定外が起こった時の対応方法を検討しておく。あくまでも予測であり、異常値の想定ではない。異常値が発生したときの対処方法も検討しておく。これは、性能だけに言えることではない。

【当たり前の思い込みによる齟齬を払拭せよ】

インターネットの普及にともない、Web 画面でのシステム化が多くなっている。システム部門は Web になったら速度が遅くなるのは当たり前だと思っている。しかし利用部門は違う。現行の速度は当たり前で、むしろ速くなると思っている。この齟齬は、特に現行システムと機能的には変わらない時によく起こる。要件定義は、変えるところに注力し変わらないところは現行と同じという手抜きが発生している。機能は良いが、非機能要求は要件定義時に確認しておかないと意図せず変わってしまうことがあるので注意しなければならない。

② 昨今の性能に関する要求の高まりを踏まえて注意すべきこと

【現行システムと比較してみる】

レスポンスタイムを１秒以内と決めて、それが「早い」のだろうか。「早い」というのは人の感覚であり、比較のもとに感じている。すなわち、現行システムや他のシステムより早いかどうかを比較してみることが有効である。たとえば、現行システムのレスポンスタイムが 5 秒ぐらいであり、3 秒であっても十分に満足度が得られるのであれば、3 秒を目標値と設定することは、性能を出す難易度が高い場合の合意形成方法のひとつとして有効である。

【体感する】

感覚による評価によるところが大きいため、実際に速度を体感できる環境を整えると良い。いわゆる、プロトタイピングである。すなわち要件定義工程でプロトタイピングを実施し、体感してもらう。要件定義の計画時にプロトタイピングの実施を意識した進め方、体制を盛り込んでおくことが重要である。

3.5 多様化するステークホルダとの合意形成

要件定義後の開発工程において、要件変更はプロジェクトの QCD に大きな影響を与える。そのような要件変更は要件定義段階で識別していなかったステークホルダから発生することもある。これは、第 2 章でも述べたとおり、ステークホルダの数が多いことに起因する。したがって、関係するステークホルダを抽出することが、成功への第一歩となる。

また、適切なステークホルダ抽出ができたとしても、ステークホルダの問題認識に対し、適切な課題設定や解決策を提示できないために合意形成に時間がかかったり、ステークホルダ間での調整に時間がかかったりして、要件定義の期間を守ることができないケースも多い。したがって、過不足なくステークホルダを抽出した上で、それぞれのステークホルダ自体とステークホルダ間の関連を分析し、適切な要件定義アプローチを取る必要がある。

適切なアプローチを取ることにより、ステークホルダそれぞれが腹に落ちた要求として定義でき、スムーズに要求の合意形成を進められるだけでなく、開発工程における要件変更リスクを低減することができる。

漏れなく、的確にステークホルダを分析すべし

【漏れなくステークホルダを抽出せよ】

目の前にある対象とする業務やシステムに関連するステークホルダに視野が狭くなってしまうことが多い。取り掛かりとしてはよいが、対象業務やシステムとの間で間接的に影響を受ける／影響を与える関係者もステークホルダとして抽出する必要がある。

また、実務担当者だけでは、組織としてのゴールや経営課題が見えていない可能性があり、表面的な意見や個人的な意見が混じる可能性がある。そのため、経営層もステークホルダとして識別する必要がある。

以下に抽出の観点を示す。

- 直接的な関わりを持つ関係者
  - 対象業務の運用者
  - 対象システムの運用者
  - 対象業務・システムの最終利用者（お客様やエンドユーザなど）
  - 業務の結果で評価される人（経営層など）
- 間接的な関わりを持つ関係者
- 対象業務と関連する顧客、取引業者
- 対象業務と関連する業務の運用者
- 対象業務と関連する業務の結果で評価される人（経営層など）
- 対象システムと関連する外部システムの運用者
- 対象業務の主管部門以外の組織の人
- 対象システムの外部システム

あくまで一例であるため、プロジェクト関係者へのインタビューやブレーンストーミングなどを使用して、漏れなくステークホルダを抽出する必要がある。

【プロジェクトへの影響度と関心度でステークホルダを分析せよ】

漏れなく抽出をしても適切な対応をしなければ、限りある時間を無駄に浪費してしまう可能性がある。例えば、プロジェクトへの影響度も高くなく、関心度も低いステークホルダに熱心に対応したとしても、影響度、関心度共に高いステークホルダの一声で決定事項が覆ってしまう事態も発生しかねない。

したがって、プロジェクトへの影響度と関心度の程度を各ステークホルダに対して評価し、その特性に応じて適切な対応をとる必要がある。以下が、影響度、関心度による対応例である。

- 影響度：高、関心度：高
  ⇒より多くの時間と労力を割いて積極的にコントロールする
- 影響度：高、関心度：低
  ⇒ステークホルダの持つ要求を満足させておく
- 影響度：低、関心度：高
  ⇒情報を提供し続けて、プロジェクトへの関心を満足させる
- 影響度：低、関心度：低
  ⇒モニタリングしておくだけでできるだけ労力を割かない

【ステークホルダ間の対立・関連を見極めよ】

ステークホルダを抽出し、課題を分析していく中で、ステークホルダが出す課題間に関連性が発生する。そのような状況をそのままにして、課題定義、解決策の定義を行っても、最優先で解決すべき課題を見落としてしまい、ステークホルダが納得して合意できない場合もある。

そのような場合、ステークホルダの課題をリッチピクチャとして図や文字、絵で表現し、対立する課題や類似する課題を可視化して整理するとよい。

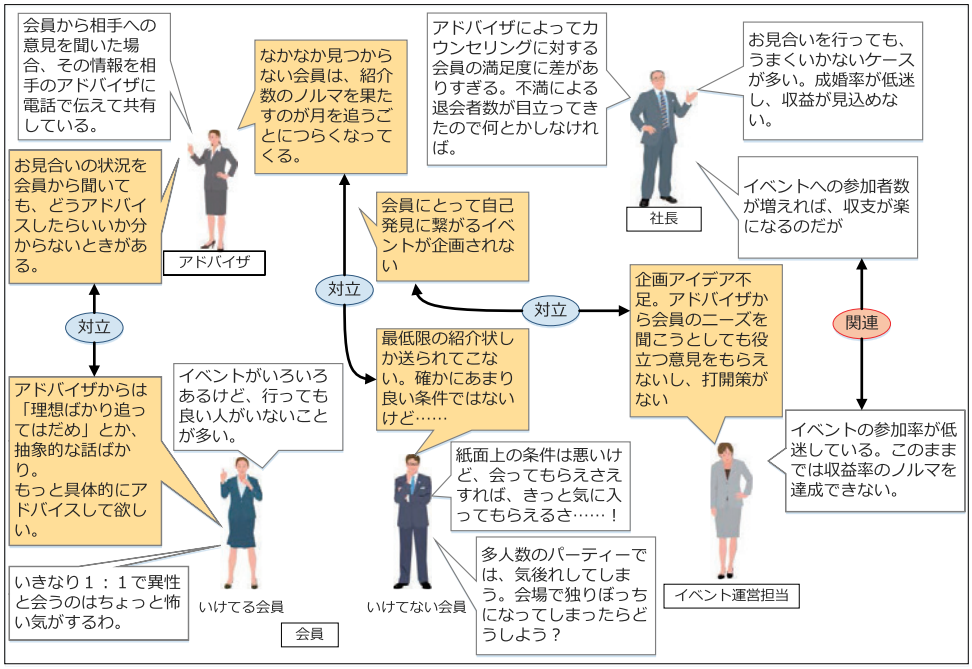


図 3.17 リッチピクチャを用いた課題整理例

対立が多い、提示者が多い課題は解決策の影響が大きく、合意形成が必要である。そのような課題から目的に照らし合わせた重要課題の深堀り、解決策の導出を行うことでステークホルダとの合意形成、調整をスムーズに進めることができる。

当事者意識を持ってレビューをすべし

【問題、ニーズ、課題を一人称で扱うべし】

要求の源泉として、問題、ニーズ、課題を分析すべきであると前述した。プロジェクトメンバ全員が我が事として認識し、納得し、自分の言葉でプロジェクト外の人に説明する責任を持っている。プロジェクト参画者はこれから新しい業務やシステムを、プロジェクト外の人に展開していく中枢になるはずである。展開に際して、業務担当者から「どうしてこのような業務の流れになっているのか？」「どうしてこのような機能が必要なのか」質問を受けるであろう。その際、「××さんが必要だと言ったから」と答えたら、業務担当者は失望するだろう。

したがって、プロジェクト参画者がそれぞれ持つ考えを出し合い、質問し合い、議論し合うことで、相互理解を深めたり、共通認識を構築したりするプロセスを踏んで、一人称で問題、ニーズ、課題を扱えるようになる必要がある。

一定の時間を確保してワークショップを開催し、以下の様なことを行うことにより、相互理解や共通認識構築をしやすくする。

- 問題、ニーズのブレーンストーミング
- 模造紙と付箋を使った問題、ニーズの深掘り、課題の抽出
- 模造紙と付箋を使った目的、手段の構造化、詳細化

【セレモニーでないレビューを行うべし】

成果物を表面的にレビューして、合意書に署名するなどの形式的なレビューは意味が無い。レビューは要求が自分たちの共通認識として構築できているかを関係者全員で確認する最終イベントである。

例えば、業務フローのレビューでは、関係者全員が集まって、それぞれが業務担当者となって、新しい業務フローに沿った業務をシミュレーションすることにより問題が解決できるのか、ニーズを満たすことができるのか、または、例外的な業務でも本フローを運用することができるかなど確認する必要がある。

あくまで、レビューは要求のエラーを摘出する場であり、エラーの解決策を検討する場所ではないため、できるだけエラー摘出に集中することが重要である。

エスカレーションパスを明確にすべし

【プロジェクト外で意思決定できる機関を用意せよ】

プロジェクト当事者では決定できない事項や当事者間の意見の衝突が発生し、合意形成に時間がかかるケースが往々にしてある。そのような場合は、プロジェクト外の上位層で意思決定して、プロジェクトへ指示したほうが早い。したがって、プロジェクト計画時において、プロジェクト外の上位層で意思決定を行う責任者の任命と意思決定者へのエスカレーションパス、決定機関の設置と関係者への事前合意を行うとよい。

決定機関への参加者の例を以下に挙げる。

- プロジェクトオーナ
- プロジェクトスポンサ
- 業務部門の責任者
- システム部門の責任者
- ベンダの責任者

定期開催に加えて、重大な課題が発生した場合や緊急な意思決定を要する場合に関係する責任者のみを招集して実施する臨時開催もできるようにするとよい。

3.6 現行業務やシステムの把握

昨今のシステム開発では、既存システムが存在することがほとんどである。しかし、業務自体がシステムに隠蔽されているため、なぜそのような業務になっているのかを理解している人が業務部門にも少なくなってきている。一方で、システム部門にも業務を理解している精通者がいるわけでもない。真の問題を分析するためには、業務を理解する必要があるが難しいケースが多い。理解が浅いにもかかわらず、「現行踏襲」という言葉を使ってしまい、業務仕様が不明瞭なままプロジェクトを進め、混乱を招くことが多い。

現在は、業務のほとんどがシステム化されている。そのため、現行システムを理解することが、業務理解への第一歩だ。したがって、現行システムを可視化して理解するとともに、その理解のもとに現行業務を可視化し、理解を深める必要がある。

現行業務やシステムを理解することにより、現状の問題やニーズを的確に捉えられ、適切な要求として定義できる。

現行業務／システムを可視化すべし

【システムを可視化せよ】

システムは業務の写像である。現行業務を理解する手がかりがない場合、もしくはドキュメント群が全くメンテナンスされていない場合は、システムを可視化することが現行分析の第一歩となる。以下に可視化できる情報の例を挙げる。

- データモデル（データベースから）
- モジュール構成・クラス構成（プログラムコードから）
- 処理ロジック（プログラムコードから）
- ジョブスケジュール（ジョブ管理ツールの設定から）
- OS、ミドルウェア、ソフトウェア設定（設定ファイルから）
- 接続先システム情報（データ連携ツールの設定から）

【詳細から全体像に概略化せよ】

可視化の目的は、現行業務、システムの理解である。したがって、精通者でない担当者でも理解できるよう、概略化し全体像を文書化する必要がある。あくまでもあるべき業務像を検討するための入力とするために、単に寄せ集めるのではなく、高い視点を持って俯瞰的に整理するよう留意していただきたい。以下に、全体像への概略例を示す。

- システム間関連図
  接続先システム情報、ジョブスケジュール、データモデルなどをもとに、どのシステムからどのシステムへいつ、なにが、どのように連携しているかを可視化する
- システム構成図
  OS、ミドルウェア、ソフトウェア設定などからシステムの構成、サーバ間の関連性を可視化する
- 概念データモデル
  どのような情報が保持され、どう関連しているかを可視化する
- 画面遷移図
  処理ロジックなどから画面遷移を可視化する
- 業務スケジュール
  ジョブスケジュールなどから業務スケジュールを可視化する

フィールドワークを通して現状を理解すべし

現行システムの復元だけでは、システムとして実現していない領域を可視化できない。また、業務担当者の頭の中にだけあり、形式化されていない情報もある。したがって、既存のシステム、運用ドキュメントや現行業務担当者からの再学習（リラーン）を通じて、現行業務、システムの可視化結果の補完を行うとよい。再学習とは、システムを再学習し、システムへの理解を促進する手法である。

現行業務を再学習することで、プロジェクトメンバが単に業務の流れ、システムの使い方を知るだけでなく、潜在する問題、ニーズ、課題を一人称で、あたかも自分事のように考えられるようになる。そのような状況で導かれた課題やその課題を解決する要求は現行業務の担当者からすれば納得感のあるものとして受け入れられやすくなる。

【埋もれた宝（既存ドキュメント）を発掘せよ】

社内にはたくさんの資料があるはずである。まずは、対象の業務、システムや周辺業務、システムに関する資料を探すことから始めるとよい。以下に、ドキュメント例を挙げる。

- 現行業務、システム稼働時の説明資料
- 新人研修用ドキュメント
- 運用マニュアル
- SLA
- 契約書

【可視化した業務を実態と照らし合わせて確認せよ】

既存ドキュメントを使って、関係者全員での理解を深める際に使う、現行の業務シナリオを作成する。ただし、業務シナリオを作成して終了ではない。あくまで可視化は現行理解への第一歩である。したがって、真の理解を行うために、実際の業務担当者に弟子入りする気持ちで実態と照らしあわせて作成した業務シナリオを確認するとよい。実態に照らしあわせた確認方法の例を以下に挙げる。

- 現行システム操作
  まずは、実際の現行システムを操作し、作成したシナリオを確認する。試験環境や擬似本番環境があればそれを使うほうが望ましい。操作をしている中で様々なケースでどのようにしているのか疑問に思う点があるはずなので、記録し業務担当者に確認するとよい。
- 業務担当者へのインタビュー
  作成した業務シナリオを使って、業務担当者にインタビューして確認してもらうとよい。あわせて、前述のシステム操作で出た疑問点を確認する。インタビューを通して、様々なシナリオが抽出できるし、現行業務、システムへの不満、ニーズなども抽出できる。
- 実作業の観察（エスノグラフィー）
  次に、業務担当者の作業環境に同行、陪席し、実作業を観察しながら作成した業務シナリオを確認するとよい。業務担当者は日々の作業で当たり前に感じている部分を第三者の目からすると問題に感じたり、こうしたほうがよくなるのではないかといったアイデアを発想できたりする。
- 業務作業のシャドーイング
  さらに、業務担当者と一緒に、もしくは業務担当者に代わって業務を遂行する方法もある。実際に経験することにより、現行業務を一人称で捉えられるようになるだけでなく、「なぜこのような作業が必要なのか？」といった、ドキュメントでは到底表現できない、背景や文脈を理解する足がかりを掴むことができる。

プロジェクト全員で共通認識を構築すべし

【プロジェクト全員での共通認識として構築せよ】

プロジェクトの一メンバがドキュメントを作成するだけでは不十分である。現状をプロジェクト全員で共通認識として構築しなければ、同じ方向性を持って問題、ニーズ、課題の分析や目的、手段の検討もできない。

そのため、プロジェクト全員で作成した業務シナリオの読み合わせやウォークスルーを通じて関係者間での認識違いを浮き彫りにしたり、お互いの共通認識を再確認したりする。

【業務移行を見据えて現状を理解せよ】

業務シナリオについての共通認識を構築することは重要であるが、現状の業務遂行体制や人材についての共通認識を構築することも重要である。システムライフサイクルからすると開発期間はほんの一部であり、大半は稼働後の運用期間である。したがって、スムーズな稼働だけでなく、安定的に業務を運用するためにいかに業務移行として体制を構築する必要があるか、人材をいかに育成していくかを要件定義において検討し始める必要がある。そのために、現状の体制、人材についての共通認識を構築することも必要である。

第４章　要件定義成果物の品質向上

3章では要件定義プロセスにおける課題に焦点をあて、課題を解決するための勘どころを提示した。ここで提示した勘どころは、要件を成果物としてまとめあげるまでのプロセスに対する提案であり、主に要件定義内容の質的向上（経営や業務に役立つ要求を見極めるなど）を狙ったものである。

4章では要件定義における各成果物に焦点をあて、その品質向上のための勘どころを提示する。

これまで要件定義成果物の品質向上は、各社でフォーマットや書き方、成果物サンプルなどを準備するといった「標準化」施策を中心に進めてきた。標準化により何を書けばよいかは明確にできたが、要件定義にはまだまだ課題が残っている。それらの課題が未解決のままで要件定義を実施することにより、定義内容に抜けや漏れや齟齬が発生し、後続工程に正しく伝わらず、手戻りの原因になるなどの問題が発生している。抜けや漏れや齟齬は、発見が遅くなればなるほど、図4.1に示すように修正の負荷が大きくなる。

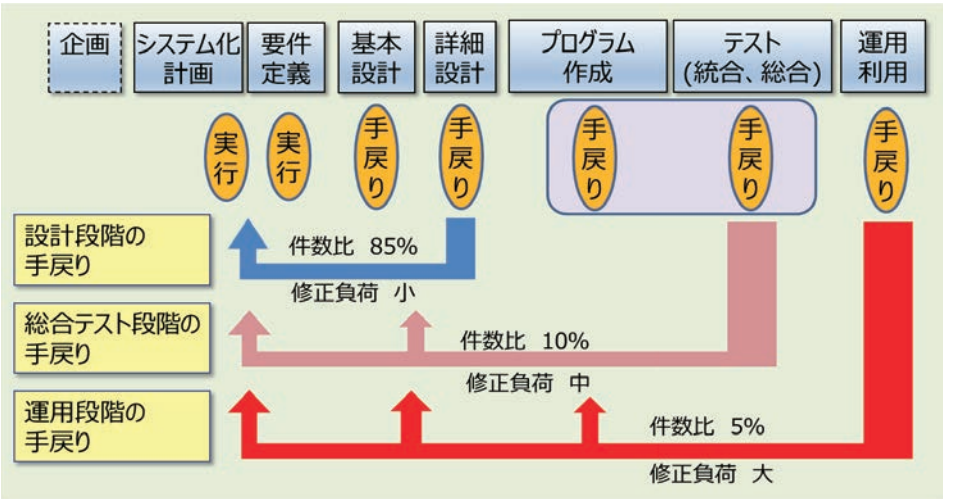


図4.1　システム開発における手戻りの影響度

（出典）日本情報システム・ユーザー協会「ソフトウェアメトリックス調査2014」[18]のデータから作成

要件定義成果物に関する具体的な課題とは、各成果物をどの順で書くのか、どこまで書くのか（例外も含めてすべて書くのか）、どのレベルで書くのか（粗い粒度で書くのか、細かい粒度で書くのか）、何のために書くのか、齟齬なく伝えるためにどのように書けばいいのか、どうチェックするかといったものである。

これらの課題に対して、4.1節では、成果物の作成計画時の留意点を解説する。4.2節では、主要な成果物の書き方や留意事項を、サンプルを交えて解説する。4.3節では、レビューのように成果物に共通する留意点を解説する。1.3節で述べた課題項目との関連は、以下の通りである。

⑥　要件定義書の不備が多い（抜け、漏れ、曖昧、不完全、不整合などへの対策が不十分）

⑧　要件定義の記述の粒度や深さの基準が不明、内容の評価ができない
⑫　体制、役割分担の不備での失敗

なお、要件定義成果物は、一過性の成果物ではないことにも注意してほしい。

要件定義の目的はシステムの要求者（発注者）から開発者（受注者）に要件を正確に伝えることであるが、要件定義に用いられる各種定義書は要件定義に不慣れな業務担当者に対し、分かりやすい形式であることも重要である。その一方で、要件定義の内容を第三者に検証依頼したり、システムへの機能追加や将来のシステム更改の際に初期構築時とは異なる実装者に依頼したりする場合がある。よって、要件定義に用いられる各種の定義書の書式については、なるべく一般的に使用されているものを使用した方がよい。

本章で説明する各要件定義成果物を実際のプロジェクトで作成する際には、こうした上記のバランスにも注意して、プロジェクトに最適な記述形式を選択していただきたい。

4.1 成果物の作成計画時の留意点

本節では、要件定義の成果物を決めるなど、計画時の勘どころを中心に解説する。

4.1.1 成果物を決めるための勘どころ

要件定義において何を成果物としてどのタイミングで作成するかを決定することは意外に難しい。例えば、業務フローは企画工程で作成すべきか要件定義工程で作成すべきか、システム化業務フローは要件定義工程で作成すべきか設計工程で作成すべきか、概念データモデルはどの工程で作成すべきか、など成果物によっては工程の線引きが難しいものがある。

ここでは、要件定義工程で作成すべきドキュメントを決める際に考慮すべきポイントを示す。

【前後のプロセスの目的を踏まえて、プロセスに成果物をマッピングすべし】

要件定義成果物をプロセスにマッピングした例を表 4.1 に示す。同表は、要件定義工程で作成する成果物について、それらと上流工程の各プロセスがどのように関係づけられているかを示している。ここでは構想立案で使用される SWOT 図や要件定義で使用されるステークホルダ関係図など、作成プロセスが明確なものは省略している。プロジェクトの状況や目的によって成果物の作成プロセスは異なるので、これは一つの例として参照いただきたい。

なお、表 4.1 に示すプロセスは、「共通フレーム 2013」[15]が規定するものを用いている。共通フレームが規定する要件定義関連のプロセスと工程とのマッピングについては、付録 1 を参照願いたい。また、システム要件定義の成果物については、「機能要件の合意形成ガイド」[11]にもそのコツがまとめられている。その一覧を付録 2 に示す。

表 4.1　プロセスへの成果物のマッピング例

企画プロセス		要件定義プロセス	システム要件定義プロセス
構想立案	システム化計画立案		（外部設計：※合意形成ガイド）
問題点、課題、ニーズなど一覧	問題点、課題、ニーズなど一覧	問題点、課題、ニーズなど一覧	
要求一覧	要求一覧	要求一覧	
業務機能構成表	業務機能構成表	業務機能構成表	システム化業務一覧
ビジネスプロセス関連図	ビジネスプロセス関連図	ビジネスプロセス関連図	
業務フロー	業務フロー	業務フロー	
システム化業務フロー	システム化業務フロー	システム化業務フロー	システム化業務フロー
	業務処理定義書	業務処理定義書	
		システム化業務説明	システム化業務説明
		画面一覧	画面一覧
		画面遷移	画面遷移
		画面レイアウト	画面レイアウト
		画面入出力項目一覧	画面入出力項目一覧
			画面アクション明細
概念データモデル(ER図)	概念データモデル（ER図）	概念データモデル（ER図）	ER図
		エンティティ一覧	エンティティ一覧
		エンティティ定義書	エンティティ定義書
		CRUD図	CRUD図
外部システム関連図	外部システム関連図	外部システム関連図	外部システム関連図
	外部インタフェース一覧	外部インタフェース一覧	外部インタフェース一覧
		外部インタフェース項目定義	外部インタフェース項目定義
			外部インタフェース処理説明
		バッチ処理一覧	バッチ処理一覧
			バッチジョブフロー
			バッチ処理定義
	帳票一覧	帳票一覧	帳票一覧
		帳票レイアウト	帳票レイアウト
			帳票項目定義
		帳票概要	帳票概要
			帳票編集定義
		データ項目定義書	データ項目定義書
		ドメイン一覧／定義	ドメイン一覧／定義
		コード一覧／定義	コード一覧／定義

表 4.1 は、要件定義の各成果物は複数のプロセスにおいて、それぞれ異なる目的のために作成される可能性があることを示している。同表で例示した成果物のうち複数のプロセスに記載されている成果物について、使用の目的が異なれば、どのプロセスで、どのレベル（範囲、深さなど）で作成するかが変化する。要件定義を実施する際には、前後のプロセスの目的を踏まえて成果物を決定することが重要である。

(1)「問題点、課題、ニーズなど一覧／要求一覧」

- 構想立案
  経営者や事業部門長などの経営的な課題やニーズ施策などを明確にする。
- 計画立案
  構想立案で明確になった方針を実現するための主要な現場やITの課題や施策を追加し、計画立案に盛り込む。
- 要件定義
  承認された計画（方向性）を実現するために、関係するステークホルダから現場の問題や要求も抽出し、実行に移す全ての要件を決める。

(2)「業務機能構成表／業務処理定義書」

- 構想立案
  経営施策がビジネスプロセスの大きな変更である場合は、変更部分の概要を合意するために作成することもある。
- 計画立案
  変更量を想定するため、現場レベルの主要な変更機能を明確にする。
- 要件定義
  承認された計画（方向性）を実現するために、関係するステークホルダがTo-Beとして実現する全ての業務機能を合意形成し実行に移す要件として決定する。

(3)「ビジネスプロセス関連図」

- 構想立案
  今回の構想立案に関係する業務（スコープ）を明確にして、認識の共有を図る。
- 計画立案
  構想立案で明確になった方針を実現するために、インタフェースや対象となる業務プロセスの詳細化により、スコープの精度を向上させる。
- 要件定義
  要件定義スコープを共通認識するために、業務フローや役割分担の目次として利用する。また、要件定義の終了時にはシステム開発スコープの定義文書として内容を更新する。

(4)「業務フロー／システム化業務フロー／概念データモデル」

- 構想立案
  - ✓ As-Is
    現行業務の理解が不足している場合、As-Is の業務フローなどを作成し理解を深める。また、現行の課題の抽出や経営課題のマッピングを行い対策の検討を行うために利用する。
  - ✓ To-Be
    経営施策がビジネスプロセスの大きな変更である場合は、変更部分の概要を合意するために作成する。
- 計画立案
  - ✓ As-Is
    現行の課題の抽出や既存業務課題のマッピングを行い対策の検討を行うために利用する。
  - ✓ To-Be
    経営施策がビジネスプロセスの大きな変更である場合は、主要な現場レベルの変更部分を追加し具体的な変更量を想定する。
- 要件定義
  - ✓ As-Is
    ステークホルダから抽出した業務課題のマッピングを行い、対策の検討を行うために利用する。
  - ✓ To-Be
    ステークホルダにより、新たな業務を立案する。
- 外部設計
  確定した新たな業務のシステム化機能に対し実現性を考慮し再確認する。

(5)「エンティティ定義書／データ項目定義書」

- 要件定義
  概念データモデルの理解を補完する情報として、エンティティに属する主要なデータ項目を明示する。
- 外部設計
  概念データモデルをデータベース設計のインプットとするために、業務で必要とされる情報（データ項目）を洗い出す。

(6)「画面レイアウト／帳票レイアウト」

- 要件定義
  画面や帳票に対するステークホルダの要求として、画面レイアウトのイメージや必要な情報（データ項目）、画面に必要な処理（ボタンなど）などを明確にする。
- 外部設計
  実装する画面や帳票として性能や規約を考慮し再設計し、ステークホルダとの合意を行う。

(7)「CRUD 図」

- 要件定義
  業務フローやシステム化業務フローと概念データモデルを関連付けて、業務フローやシステム化業務フローの漏れや矛盾、概念データモデルのエンティティやデータ項目の漏れや矛盾を発見し是正するために作成する。
  また、データの発生元（データの所管部門）を明確化する。
- 外部設計
  どのシステム機能によってどのエンティティやデータ項目が生成・参照・更新・削除されるのかを明確にし、システム機能の漏れや不整合を無くす。

コラム　～成果物に記載する範囲を決める～

【悩み】

成果物は漏れなく正確に書くことが大前提です。しかし、成果物の記載ルールに「主要な○○を記載する」と書かれていることがよくあります。「主要」だけでは漏れにつながるのではと心配になってしまいます。

【解決案】

「主要」でいい成果物と、「すべて」が求められる成果物を明確にしておくことが重要です。

(1) 一覧系成果物の扱いを明示すべし

一覧系成果物は、漏れのチェックや管理に適しています。そのため、一覧系成果物には「すべて」が求められます。何を一覧表で管理するのかを明確にし、他のどの成果物と関係付けられているのかを明示しましょう。

一覧系成果物と、他の成果物の関係の例を以下に示します。

- 「問題点一覧（すべての問題点を記載）」
  ←→「問題点原因分析図（図で分析が必要な複雑な問題）」
- 「要求一覧（すべての要求を記載）」
  ←→「要求構造図（すべての要求を記載）」など
- 「画面一覧/帳票一覧（処理単位で分割されたすべての画面を記載）」
  ←→「システム化業務フローの画面（抽象化された画面名）」

(2) どのプロセスで作成するかによって記載範囲が変わる成果物を明示すべし

概念データモデルなどは、表 4.1 に示されているように、どのプロセスでも作成される可能性があるものです。このような成果物は、どのプロセスで、どの範囲で記載するのかを明示しましょう。概念データモデルの記載範囲がプロセスごとにどう異なるかの例を表①に示します。

表①　各プロセスでの概念データモデルの記載範囲の例

	企画プロセス	要件定義プロセス	システム要件定義プロセス
エンティティ	主要エンティティ	全エンティティ	全エンティティ
データ項目	主要データ項目	主要データ項目	全データ項目

(3) 後続の成果物で記載を補完する場合のドキュメント関係を明示すべし

「業務フロー」は業務改革レベルの変更を行う業務だけを対象として作成するが、「システム化業務フロー」はシステム化するすべての業務について記載するという、後続ドキュメントで全てを追求するという判断もあります。

【成果物全体の関連を明確にすべし】

成果物を定義する際には、個々のドキュメントの書き方やサンプルを提示するだけでなく、ドキュメント関連図を作成し、要件定義において作成するすべてのドキュメントが相互にどのように連携しているかを明示することが重要である。

図 4.2 にドキュメント関連図のイメージを示す。

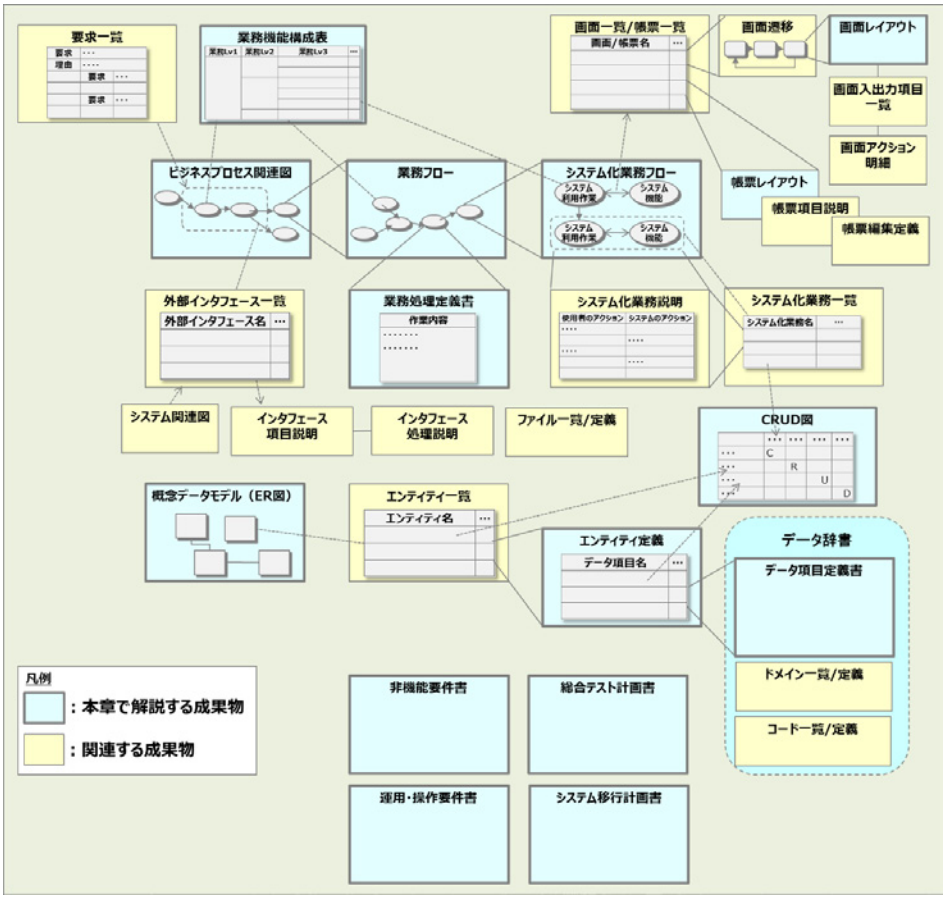


図 4.2　ドキュメント関連図のイメージ

上記のイメージは、前出の表 4.1 で示された各ドキュメントが他のドキュメントとどのような関係になっているのかを図示したものである。

例えば、「業務機能構成表」は「ビジネスプロセス関連図」に記載されたビジネスプロセスと「業務フロー」に記載されたビジネスプロセスと「システム化業務フロー」に記載されたシステム化業務プロセスが矛盾なく階層化されている、という関係を表している。

また、「業務処理定義」は「業務フロー」の業務プロセスに対応付けて記述するものであり、「システム化業務説明」は、「システム化業務フロー」に記述されたシステム化業務に対して記述するものである、と言うことが分かりやすくなる。

コラム　～フロー系成果物のプロセスの粒度を決める～

【悩み】

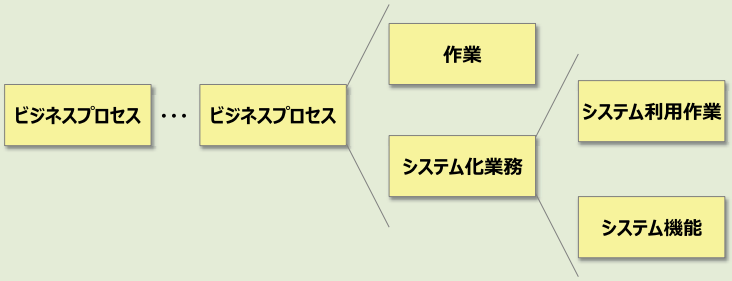
業務フローを作成する時に、書く人によって、大まかになったり、やたら細かくなったりしませんか。業務フローのプロセスの粒度を揃えることに苦労した経験のある人も少なくないでしょう。

【解決案】

粒度を定義し成果物と対応づけて書き方として提示することが重要です。

(1) 業務上のプロセスを定義すべし

図①にビジネスプロセスやシステム機能などの定義例を示します。

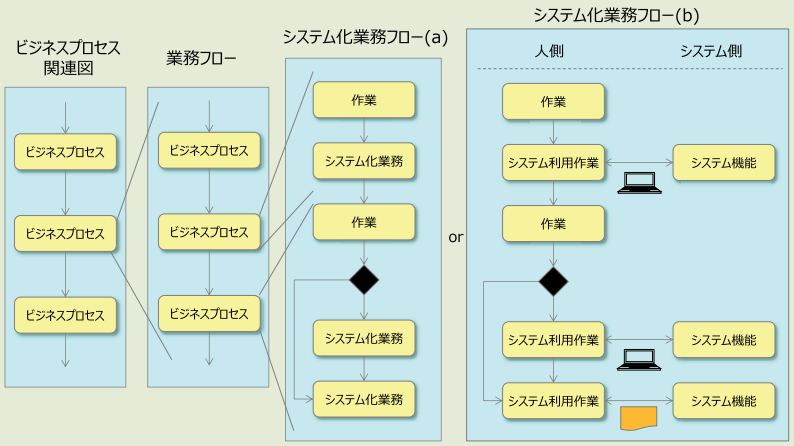


図①　ビジネスプロセスやシステム機能などの粒度の定義例

- 「作業」とは人手によって実施するプロセスを意味する
- 「システム化業務」とは IT システムを利用して実施するプロセスを意味する
- 「システム化業務」は、IT システムを利用する人手の「システム利用作業」とその人に対して IT システムが提供する「システム機能」の 2 つで構成される
- 「ビジネスプロセス」とは、「作業」や「システム化業務」をまとめた業務そのもののプロセスである。ビジネスプロセスには粒度の大きなプロセスや小さなプロセスがあり、それらが階層化される。

(2) プロセスの階層を成果物で固定化すべし

一つの考え方として、業務上のプロセスは「ビジネスプロセス関連図」「業務フロー」「システム化業務フロー」に 3 階層で記載することに決めます。業務の規模や複雑さによっては、さらに業務フローを 2 階層に分けても良いでしょう。ただし、その場合には記載の粒度を明確に提示することが難しくなります。



図②　3 種類のドキュメントとプロセスの定義の関係

図②で示すように、システム機能と関連づけて記載するかどうかによって(a) (b) の 2 通りの記載方法ができます。どちらかを選択して明示してください。

(3) 作成ルールを提示すべし

「ビジネスプロセス関連図」と「システム化業務フロー」はドキュメント記載の粒度を提示するより、以下のルールを提示した方が、作成者による粒度のバラつきを抑えることができます。

「ビジネスプロセス関連図」

ルール：1 枚に全体を記載すること、鳥瞰できること、目次になること。

「システム化業務フロー」

ルール：「システム機能」を業務に提供するサービスと想定し、IT システムを利用した業務のフローを「システム機能」が「システム利用作業」と 1：1 で対応するように記載すること。

(4) 業務フローは、粒度の考え方を提示して浸透させるべし

「業務フロー」

ルール：業務で何をするのかを記載する（How ではなく What を書く）。「作業」と「システム化業務」を区別せずに記載します。詳細は本文の 4.2.3 項を参照してください。

このように、プロセスを 3 種類のフロー図で階層化し、各成果物での分割のしかたを提示することで、作成する際の考え方がシンプルになり、プロセスの粒度を揃えることに悩まずに作成できるようになります。

【プロジェクトの状況や利用目的を意識して成果物を決定すべし】

各プロジェクトにおいて、要件定義で作成する成果物（定義書）を決定する際には、そのプロジェクトが置かれている状況や目的をもとに、前後の工程との作業の配分も調整しながら検討を進める必要がある。

その際には、各成果物の利用目的を明確にする必要がある。利用目的とは、何を表現する成果物を作成するかではなく、今後の作業にどう利用するために成果物を作るのかということを意味している。これは、プロジェクトの目的・状況によって異なるので注意する必要がある。

各ドキュメントの利用目的は、4.2 節で示すが、要件定義工程の目的を簡単に復習する。

要件定義工程では、IT システムを利用した経営や業務に貢献する新たなビジネスプロセスを創出し、確定させるという目的と、その IT システムに対してステークホルダ要求を明確に提示し、漏れなく設計・開発に繋げるという目的の 2 つの目的がある。この 2 つの目的を達成できるように、個々の成果物の目的を決定する必要がある。

また、漏れや齟齬をなくし手戻りを最小限にするためにもっとも効果的なことは、関係者全員が検討対象の業務を十分理解することである。徐々に詳細になっていく業務仕様やシステム化業務仕様は自動的に展開されるわけでなく、人と人との意志疎通の結果をもとに付加されていくものである。対象業務の目的や価値や使われ方などを理解して実施するかどうかは品質に大きく差が出る要因になる。関係者の理解や共通認識を醸成するという観点で成果物を決定していくことも要件定義を成功させる一つの要素になる。

4.1.2 成果物作成上の役割分担を決めるための勘どころ

成果物作成にあたっては、どの部門が主体となって作成するかを明確にすることが重要である。要件定義はステークホルダの要求仕様を要件として定義する作業であり、成果物に記載された内容に対する責任は、ユーザ企業の利用部門（オーナ部門）などのステークホルダが負うのが原則であることから、作成もそれらの部門が主体的に実施することが理想である。

しかしながら、利用部門が要件定義をすべて実施することは、技術的にも時間的にも困難である。また、昨今の業務は IT 抜きには改革できないことから、利用部門、システム部門、ベンダ企業がそれぞれの得意とする作業を分担しあって進めることが現実的である。

どの成果物を利用部門やシステム部門が主体で作成するかは、プロジェクトの状況に応じて異なる。後述する 12 の成果物の役割分担と補足事項（内容補足と状況に応じた代替方法）を表 4.2 に例示する。

表 4.2　主要な要件定義成果物と作成時の役割分担

項番	成果物	役割分担				補足
		ユーザ			ベンダ	
		業務部門	システム部門	運用部門		
4.2.1	ビジネスプロセス関連図	○	△	△	△	ビジネスプロセスが現状と変わる場合は、業務部門が作成する。 ただし、ビジネスプロセスの変更がなく、システム化対象の明示や外部のインタフェースの明示が主である場合、システム部門が作成してよい。
4.2.2	業務機能構成表	○	△	△	△	
4.2.3	ビジネスプロセスフロー （業務フロー／ システム化業務フロー）	○	△	△	△	
4.2.4	画面／帳票レイアウト	△	○	△	△	
4.2.5	業務処理定義書	○	△	△	△	業務自体の目的や業務内容を明示するため、利用部門が作成する。
4.2.6	概念データモデル（ER図）	△	○	△	△	業務部門がER図を理解することが困難な場合は、概念データモデルを説明した補足資料（4.2.6　概念データ コラム参照）を確認する。
4.2.7	エンティティ定義書／ データ項目定義書	△	○	△	△	
4.2.8	CRUD図	△	○	△	△	業務理解促進の意味も含めてシステム部門が作成するのが望ましい。
4.2.9	総合テスト計画書	○	○	○	△	多様な内容が含まれる成果物であり、内容に応じた作成主体を検討し、分担して作成する。また、計画書であるため、内容に応じた全ての関係者に確認する。
4.2.10	システム移行計画書	○	○	○	△	
4.2.11	運用・操作要件書	○	○	○	△	業務運用やシステム運用の要件、操作要件など多様な内容が含まれる成果物であり、内容に応じた作成主体を検討し、分担して作成する。 また、内容に応じた全ての関係者に確認する。
4.2.12	非機能要件書	△	○	△	△	非機能要件にも利用部門の業務要件が大きく関係してくるが、技術的な要素も多いため、システム部門が主体で、利用部門から業務要件を聞き出し、成果物の作成を行うことが効率的である。 また、非機能要件は多岐にわたるため、内容に応じた全ての関係者に確認する。

凡例）○：作成主体、△：作成支援（情報提供、確認（チェック）、アドバイス）

4.2 主要な成果物と作成上の留意点

本節では、要件定義の結果を後続する設計、実装、テスト工程につなぐ主要な要件定義成果物として、以下に示す 12 種類の定義書を取り上げて解説する。

- ビジネスプロセス関連図
- 業務機能構成表
- ビジネスプロセスフロー（業務フロー／システム化業務フロー）
- 画面／帳票レイアウト
- 業務処理定義書
- 概念データモデル（ER 図）
- エンティティ定義書／データ項目定義書
- CRUD 図
- 総合テスト計画書
- システム移行計画書
- 運用・操作要件書
- 非機能要件書

これら 12 種類の成果物の関連は、前述の図 4.2 を参照されたい。

要件定義において作成する成果物は、上記の 12 種類以外にも一覧表やプロジェクト管理用ドキュメント、要件定義の作業中に利用する分析ドキュメントなどがある。ここでは内容に不備があると大きな手戻りの原因になる可能性の高い成果物を対象として、各成果物の目的を含む要点を示すとともに、記述漏れが減り、後続工程からの手戻りが少なくなるように作成するための留意事項などを示す。

なお、業務要件が同じ場合でも、システム要件定義の成果物は、「自社サーバを利用してシステムを構築する」「クラウドサービスを利用してシステムを構築する」など、対象システムの特性によって作成する成果物や記述する範囲が異なる場合がある。また、比較的小規模のシステムの場合には、2 種類の成果物を 1 つにまとめて作成することもある。

4.2.1 ビジネスプロセス関連図

(1) 目的

要件定義対象／システム開発対象の範囲（スコープ）を明確にする。それをもとに、スコープ外とのインタフェースを明確にする。

ビジネスプロセスの改革内容に関する認識を共有する。

(2) 説明

ビジネスプロセスとは、情報・モノ・カネを抽出・集約・加工・移送・記録するプロセスである。情報であれば、特定の条件のデータを抽出し、計算結果を導出し、その情報を必要とする組織や機能に向けて送り出したり、後に利用できるように記録したりする。モノであれば、材料を加工して製品とし、集積場所に向けて発送する。カネであれば、差し引き計算や集計を行い、取引先に送金する。情報・モノ・カネには必ず用途があり、それらを処理するプロセスは、必ず何らかの目的を持つ。

ビジネスプロセス関連図には、情報・モノ・カネの各プロセスで実施されている抽出・集約・加工・移送・記録の内容と、他のプロセスとがどう関連しているのかを記載する。ビジネスプロセス関連図に上記を記載し、図上の各プロセスが果たす目的を理解すれば、ビジネスの全体像を把握することができる。

ビジネスは機械処理だけで構成されるわけではない。組織や個人が人手によって実施する作業もビジネスプロセスにおける重要な構成要素である。したがって、ビジネスプロセス関連図には、組織や人が担当する判断や作業が含まれる。また、人手による処理の際に用いられるルールやマニュアルなどは、それをシステム化する際に参照する基礎資料として、要件定義の際に洗い出しておくことが必要になる。

要件定義においては、ビジネスを構成するプロセスのうち、システム化の対象範囲とするものを切り分け、それらの境界を明らかにすることが必要である。

ビジネスプロセス関連図の例を図 4.3 および図 4.4 に示す。図 4.3 はビジネスプロセスを組織としてとらえて、システム化対象の組織の位置づけおよび他の組織とどのように連携しているかを表し、図 4.4 はシステム化対象を機能としてとらえて、他の機能とどのように連携しているかを表す。

図 4.3 において四角のボックスは組織を表し、内部組織と外部組織で色分けしている。また、矢印は、組織間での情報・モノ・カネの流れを表す。各矢印の説明はその矢印で連携する情報・モノ・カネをあらわす。

図 4.4 において四角いボックスは機能を表し、矢印は機能間での情報・モノ・カネの流れを表す。各矢印の説明はその矢印で連携する情報・モノ・カネをあらわす。

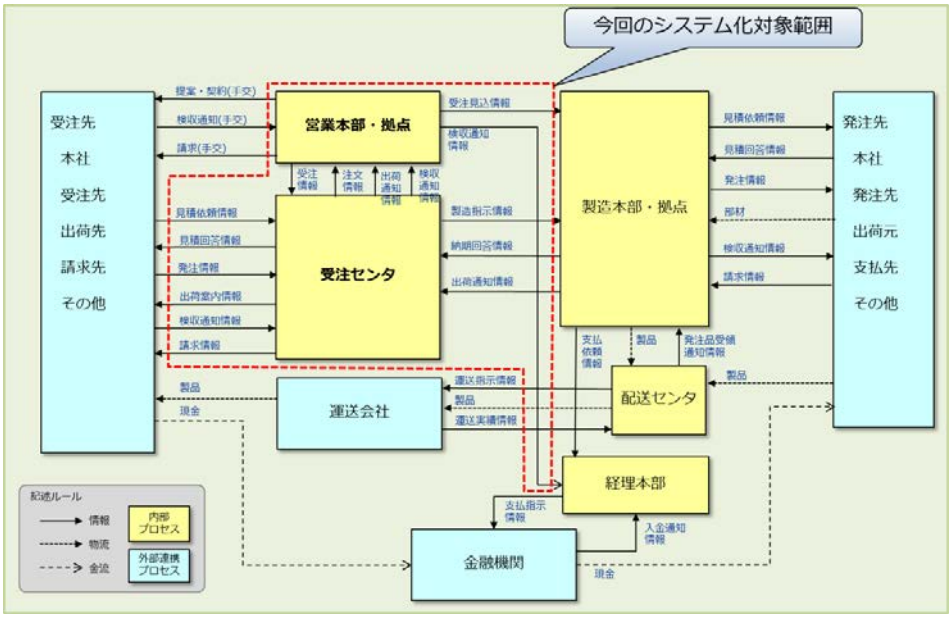


図 4.3　ビジネスプロセス関連図（組織に着目）（例）

（出典）日本情報システム・ユーザー協会「システム・リファレンス・マニュアル（第 1 巻）」[12]をもとに作成

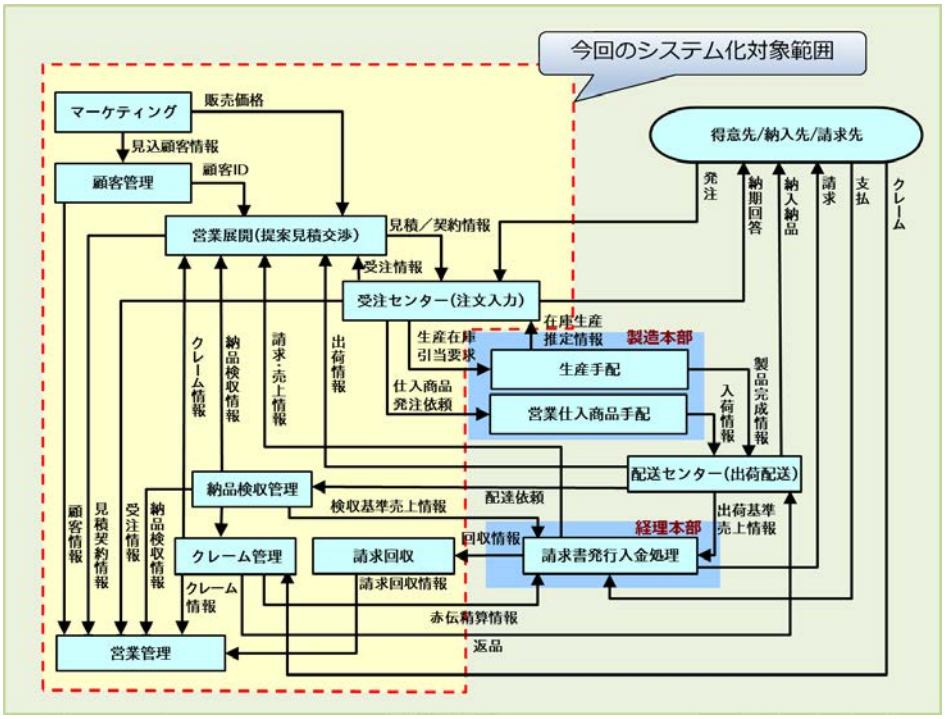


図 4.4　ビジネスプロセス関連図（機能に着目）（例）

（出典）日本情報システム・ユーザー協会「システム・リファレンス・マニュアル（第 1 巻）」[12]をもとに作成

(3) 留意事項

① ビジネスプロセス関連図作成においては、全社最適を考え、全社のシステム化計画との整合性を確保する。

例えば、図 4.3 や図 4.4 と、第 2 章で説明したシステム化計画（図 2.1）をもとに、次の事項について確認する。

- システム化対象機能の範囲が妥当か（関連する部門のビジネスプロセスまでが明確化されているか）。
- 将来の機能追加範囲が妥当かどうか（時間軸）。
- 関連する他部門や外部組織のどこまでを対象とするか、全体最適を視野に入れているか。
- このシステムを開発した場合、他システムへの影響度、業務担当者への影響に配慮し、開発優先度が妥当かどうか。

上記のような確認を行うことにより、同じような機能を持つシステムを複数開発したり、システムを開発後、すぐに大規模な修正が入ったりすることを避けることができる。また、これらの結果は全社システム化計画に反映しておくことが望ましい。

機能 / 時間	mail/ Internet	財務、人事 設備管理	調達	生産	配送	マーケティング	販売	研究
現状	Local	Local	Local Regional 集中化	Local	Local	Local Regional	Local Regional	地域特化
3年後	集中化	Regional	Local Regional 集中化	Local	Local	Local Regional	Local Regional	集中化
将来	集中化	集中化	集中化	集中化	Regional	集中化	Regional	集中化

※実際は各機能をさらに細分化し、業務および情報システムの両面について、柔軟性をもった、業務と情報システムの単純化、標準化、共通化をさらに追求する必要がある

Local	各国、各工場の事情に任せる
Regional	地域の地理、制度、文化の特性を生かして地域最適化を求める
集中化	全社最適を狙うシステム 全社情報を参照しながら支社が具体的なアクションをする場合もある

図 2.1　Global System の構造例（再掲）

（出典）日本情報システム・ユーザー協会「JIIP3　2014 年度報告書」[5]

② ビジネスプロセス関連図は、次の項目に留意して作成する。

- 複数のビジネスプロセス（組織・機能）間を連携する情報の発生元、用途・目的を詳細に確認する。確認の際には、システム化の対象部門（事業部門など）だけでなく、関連する部門（経理・人事・総務・監査、購買、物流など）からもヒアリングする。
- 関連企業を含めて、顧客対応（CRM）、サプライチェーン、バリューチェーンなど、企業活動の全体像からシステム化対象を具体化する。
- 既存業務の変更を伴うシステム化の場合は、As-Is と To-Be の 2 種類を作成する。
- 図形や矢印の色や種類に意味を持たせることにより、直観的に全体像が把握できるようにする。

③ 作成されたビジネスプロセス関連図をもとに現場作業への影響範囲を確認する。
ブロック間、機能間の情報の流れに注目することにより、以下のような検討ができる。

- 図 4.4 で見ると、在庫生産の引当要求は受注センタから実施することになっているが、生産の効率化を考えるとこれが妥当かどうか
- 出荷情報は配送センタからの出荷を契機として営業部門に通知されることになっているが、途中の事故等で顧客に届かなかった場合どうするか

さらに、今回のシステム化範囲は営業システムに限定であるが、あわせて以下の検討もできる。

- 今後、生産管理システムの構築を検討する場合、外注加工先までシステム化の対象に含めるか、配送センタはどうするか、など
- これらを多段階に分けて開発するか、まとめて開発するか

4.2.2 業務機能構成表

(1) 目的

対象の業務全体をまとまりのある機能単位でグループ化し、機能ごとにシステム化対象か人手で処理を行うプロセスかを区分する。

(2) 説明

対象システムの業務機能を担当者の作業レベルまで分解し、各業務に含まれる機能を具体的に把握できるようにするための表であり、展開された個々の機能をシステム化対象にするかどうかを分類するために使用する。中規模の基幹業務システムの場合で 3 階層程度（大分類、中分類、小分類）、大規模の場合は 4〜5 階層程度に階層分けして分解する。

業務機能構成表の例を表 4.3 に示す。この例は、業務に含まれる機能を 3 段階の階層に分けて、機能ごとに、その業務の担当者に必要とされるスキル、作業時間、システム化対象にするかどうかを取りまとめた表の例である。分解された各機能には、重要度、システム化の優先度を併記している。この表を参照すれば、システム化対象の業務に含まれる処理の全体像が把握できる。

表 4.3　業務機能構成表（図書館システムの例）

業務レベル1	業務レベル2	業務レベル3	スキル	担当者	作業時間（分/月）		システム化対象	重要度	優先順位
					現在	3年後			
図書管理	図書購入	図書購入依頼、受取	A	田中	120	同左	人手		
	新規登録	図書登録作業	B	田中	60	同左	○	1	1
	在庫管理	棚卸、除却、修復など	B	田中	120	同左	○	2	2
	貸出期間管理	貸出期間経過の督促など	B	田中	1,200	同左	○	2	1
	在庫照会	図書在庫照会への回答など	C	田中	1,200	同左	既存	−	−
会員管理	入会管理	会員登録	B	山田	1,200	同左	○	1	1
	継続、退会管理	変更、継続	B	山田	600	同左	○	1	1
		退会	B	山田	600	同左	○	1	1
	有料会員管理	請求	B	山田	480	同左	○	1	1
		入金確認	B	山田	120	同左	人手		
		督促	B	山田	840	同左	人手	−	−
貸出管理	貸出	新規貸出	C	鈴木	8,400	同左	○	1	1
		ネット貸出	B	中村	2,100	4,200	○	2	2
	返却	返却、督促など	B	鈴木	2,400	同左	人手	−	−
	期間延長	期間延長受付	B	鈴木	1,200	同左	人手	−	−
ネット照会、予約管理	ネット照会	棚検索、貸出可能照会	C	中村	2,400	4,800	○	3	3
	ネット予約	予約受付、引当通知	B	中村	4,800	9,100	○	3	3
		督促	B	中村	4,800	9,400	○	3	3

必要スキル　A:熟練を要する　B:AとCの中間　C:短時間で習得できる
重要度　1:最重要　2:次に重要　3:重要度低い
優先順位　1:一次　2:できれば一次、二次でも可　3:二次以降で問題ない

(3) 留意事項

① 業務機能の抽出は、ビジネスプロセス関連図でシステム化対象とした各業務に対して、上位レベルから段階的に実施する。業務機能構成表を作成することにより、システム化対象の業務の処理が一覧形式で可視化されることから、それを使用して効果的にシステム化対象の機能を選抜できる。

② 次に現場担当者にアンケート・ヒアリングや、現場観察（エスノグラフィーなど）を実施し、実際に行われている業務や作業と業務機能構成表の照合を行い、抽出した作業の漏れを確認する。抽出対象の業務全体の有識者がいる場合でも、確認のためにこのプロセスは必要である。

③ 機能の抽出は、システム化対象の機能だけでなく、人手により実施される業務機能についても漏れなく実施することが重要である。人手により実施されている業務機能についても、システム化後も継続して人手で実施することが明示されていないと、システム化対象範囲が曖昧になってしまい、機能要件の抽出漏れの原因になる。また、シ

ステム化対象外にした理由も明記しておくと、システム化対象範囲の妥当性を再確認する際に効率化が図れる。

④　業務機能は担当者の作業レベルまで分解し、担当者に要求されるスキルレベル、作業時間を確認する。スキルレベルは

A：熟練を要する業務

B：中間的なレベルの業務（作業）

C：短時間で習得できる業務（作業）

に分類すると、システム化対象業務における処理難度の分布が分かる。1回当たりの作業時間、期間あたりの発生回数および人件費単価を積算することにより各業務にかかる負荷を定量的に測定できる。この作業により、誰がどのような作業にどの程度の時間をかけているかが分かり、さらに、システム化した際の1回当たりの処理時間の予測を行うことにより、システム化することにより効果が期待できる機能と、効果が期待できない機能とを分別できる。

⑤　作業時間は現時点の実績だけでなく、将来の予想も調査する。また、日次のピーク時間帯や月次・年次の集中時期（月末、年度末など）がある場合は、それを明示しておくことにより、上記の効果予測の精度を向上させることができる。

⑥　上記①～④を記載した業務機能構成表を作成することにより、システム化の効果（作業時間の削減他）、業務の重要度、現状と将来の需要予測に基づいた優先順位付けが検討できる。さらにこの表を見て、手作業に分類した業務のうち将来システム化の対象にする機能の候補を抽出でき、次期機能追加の内容の検討ができる。業務機能構成表に将来の処理量の予想も入れておき、それを適宜更新するようにしておけば、レベルアップ（業務改革）を検討する際の基礎資料として活用できる。

⑦　DMM（ダイヤモンド・マンダラ・マトリックス、機能分析表）を用いると、業務機能構成表の業務レベルの分類の偏りをなくすことに役立つ。

4.2.3 ビジネスプロセスフロー（業務フロー／システム化業務フロー）

(1) 目的

As-Is 業務フローを作成して業務の問題点や課題を明確にする。As-Is 業務を関係者で理解する。

新しいビジネスプロセスを創造し、To-Be 業務フローに表して共有する。

(2) 説明

① ビジネスプロセスフロー（業務フロー／システム化業務フロー）（以下、業務フロー）には、対象業務の業務処理機能、業務処理担当部署、処理手段（手作業、コンピュータ処理）を記載する。システムによる処理だけでなく、人手の作業も記載の対象にする。業務フローには、手段に依存しない目的レベルの業務を明確にし、システム化業務フローでは、IT システムを利用した作業を記載し、システムとの関連についても記載するのが望ましい。

② As-Is と To-Be の両側面で業務を分析する。As-Is と To-Be の作成目的が異なるため、それぞれの目的を意識して作成することが望ましい。As-Is 業務フローの作成目的は、現状の業務の問題や課題を発見し、フロー上の作業と対応付けることにより業務を理解することである。一方で、To-Be 業務フローの作成目的は、新しい業務プロセスと施策の関係を対応付けることにより業務プロセス改革内容の合意を図ることである。

③ 業務フローの記述方法には、BPMN（Business Process Modeling Notation）、フローチャート、DFD(Data Flow Diagram)、UML（Unified Modeling Language）のアクティビティ図、PFD（Process Flow Diagram）などがある。

④ 業務フローの書き方を説明する。図 4.5 に、図書館の図書貸出業務の業務フローの一例を示す。

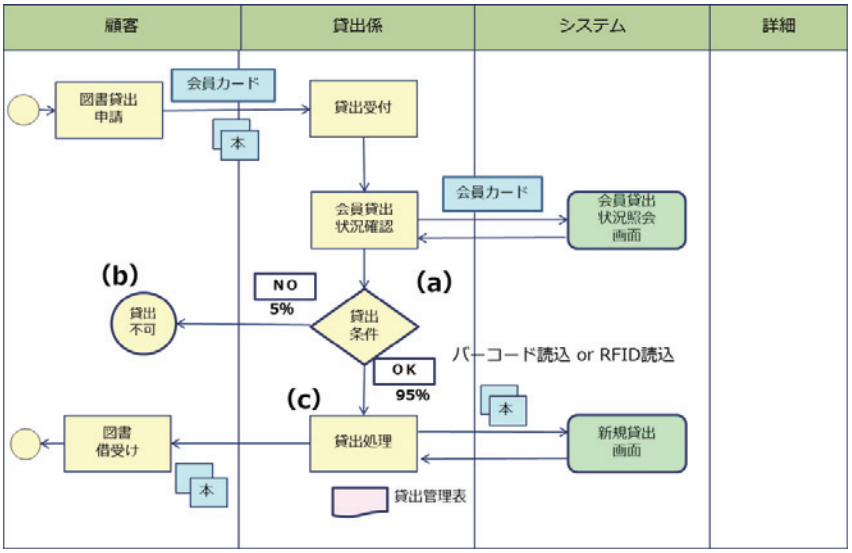


図 4.5　図書館の図書貸出業務フローの例（1）

同図においては、最上部に担当部署（アクタ）およびシステムを記し、各担当部署に対応する業務の流れをその下に記している。流れ図では、黄色い四角のボックスは作業を表し、青い四角のボックスは物理媒体を表す。また、緑の四角いボックスは画面を表す。ひし形は条件判定を表し、サークルは開始、終了を表す。そして、矢印は、業務の順序関係を示す。同図上に表現してある(a)から(c)に対して、書き方のポイントを説明する。

(a) 条件によって処理が大きく変わるものだけを記載する。分岐の一方は別のプロセスにして、できるだけ複雑な図になることを避ける。貸出条件（貸出延長の有無、督促会員の確認など）は詳細欄に記載する。また、詳細欄には記述しきれない量になる場合には、業務処理定義書(4.2.5 項を参照)に記載し、ここには記載しない。その際には、詳細欄には業務処理定義の ID など、関係付けできる情報を記載する。

(b) 貸出不可になった際に継続する業務プロセスがある場合は、業務フローを別に作成し、コネクタを記載する。

(c) 本によって貸出期間に違いがあるなどの貸し出し条件の詳細は、詳細欄にコメントを記載する。また、条件の組み合わせが二段階を超える場合は詳細欄に条件分岐内容を記述し、フローが複雑にならないようにする。

また、図 4.6 に図書貸出業務の業務フローのもう一つの例を示す。同図においては、担当部署（アクタ）、システム、および外部組織の業務処理を、左から右に時系列で記載している。

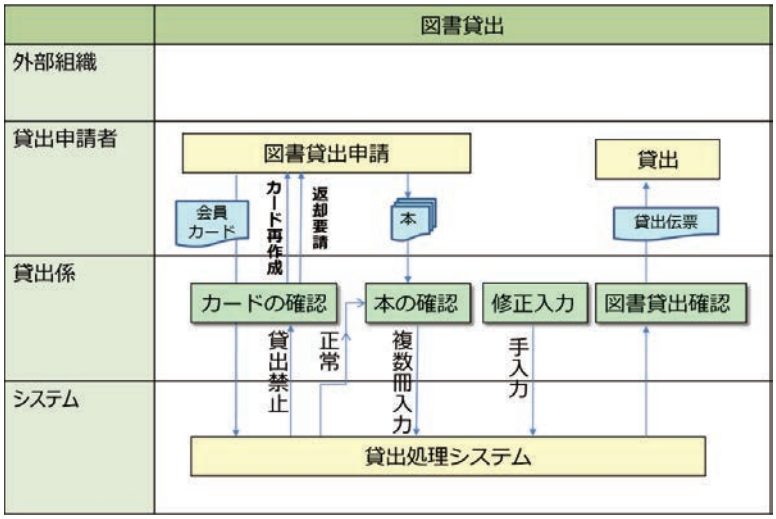


図 4.6　図書館の図書貸出業務フローの例（2）

(3) 留意事項

① 業務フロー作成にあたっては、業務改革の視点を常に意識し、現状追従にならないように注意することが重要である。また、対象業務の目的、期待する効果を記載しておくと、それをもとに開発目的を随時再確認することができ、後続工程で議論が後戻りしない。

② 5W2H に注意して要件を記載すると、要件がより具体的に表現され、漏れや不整合の防止に繋がる。図書館システムの例で見ると次のようになる。

表 4.4　図書館システムにおける確認ポイント

5W2H	確認ポイント
What	・商品やサービス(What)による違いはないか
Who	・相手(Who)による違いはないか
When	・時間、時期、タイミング、順序など(When)による違いはないか ・状況や状態、条件による違いはないか
Where	・場所(Where)による違いはないか
Why	・理由（Why）による違いはないか
How	・やり方や方式（How）による違いはないか
How many（How much）	・量（How many）や金額（How Much）による違いはないか

③　業務フローの作成手順

(a) 通常処理の抽出

まず、各業務機能における通常処理（例えば月次の業務量の大半（例えば 95%以上）を占める業務パターンなど）の業務フローを記載する。

通常処理における取消、変更（一部変更、一部取消などは、業務シナリオによっては例外処理扱いにする）は、通常処理に含めて機能詳細を記載し、システム化対象に含める。対象外にする場合はその理由を詳細欄に記載する。

(b) 例外処理、特殊処理の抽出

(i) 例外処理の抽出

例外処理とは、プログラムがある処理を実行している途中で何らかの異常が発生した場合に現在の処理を中断（中止）して別の処理を行うことであり、その際に発生した異常のことを例外と呼ぶ。業務プロセス上のある処理を実行している際に、処理を継続できないようなエラー（貸し出し状況を確認したら新規に図書の貸し出しを許可できない状態であった等）が発生した際に実施する別処理のことを、本書では例外処理と表現する。この例外処理を漏れなく洗い出すことは業務フロー作成における重要目的であり、その精度が以降のシステム開発の品質に大きく影響する。そのため、例外処理として実施される業務処理の洗い出しには、漏れを極力排除する努力が必要である。抽出した各処理の発生頻度や影響度、重要度も把握する。システム化対象にしないと判断した処理であっても、業務への影響度などからシステム化対象に含める場合もある。その場合は、詳細に業務フローを記載してシステム化対象に含め、システム化対象に変更した理由を記載する。また、システム化すべきかどうかが判断できない場合は課題として管理する。

(ii) 特殊処理の抽出

イベント、キャンペーンなどの不定期に実行される業務処理を、本書では特殊処理と表現する。この特殊処理も業務プロセスとしてすべて抽出する。本作業により抽出した各業務処理をシステム化対象にするかどうかは、それぞれのシステム化に対する投資対効果を別途評価した上で判断する。

(iii) 例外処理、特殊処理の抽出における漏れの防止

例外処理、特殊処理の抽出は漏れることがあるため、入力ミス、決算またぎ、責任者不在、緊急時の事務、サービス時間外の対処など、重点的に確認すべきポイントを事前に洗い出した上で、業務担当者からのヒアリングの実施、抽出結果に対する別担当者による再確認、現場で実施されている操作の実見などにより抽出を行う。不定期の処理、イベントやキャンペーンなどにも注意する。

抽出した例外処理をシステム化する際に、画面や帳票に対して例外処理か

どうかが分かりやすい命名を行うと、業務上のミス防止にもつながる。なお、例外処理、特殊処理の大半は投資対効果からシステム化対象外と判断できるものが多い。その場合は処理の抽出までとする。

(c) 業務処理の組み合わせにおける抽出漏れの確認

複数の処理パターンの組み合わせで処理が分類される業務処理（例えば、特定の会員種別で、特定のカテゴリーの商品を購入した場合など）において、組み合わせによる処理の相違がないかについて、社内の業務部門や可能なら顧客にも協力してもらい確認する。

④ 業務フローは、次の事項に留意して作成する。

(a) 業務機能名には、次の例のように業務機能を具体的に表わす名称を使用すると、どのような業務機能のフローなのかが分かりやすい。

- 「照会」「更新」「出力」「管理」等、何の業務機能か分からない一般的な名前にとどめず、「会員貸出状況照会」などのように業務機能の対象など具体的な処理を想像しやすい単語を付加して書くと分かりやすい。
- 同様に、「与信照会」、「契約入力」でなく、「契約前顧客与信レベル確認」「契約条件登録」とする。

(b) 情報の加工内容（検索、集約、集計、抽出）と利用目的（用途）が具体的に記載されるように業務フローを作成すると、各プロセスにおける業務要件の検討漏れを減らすことができる。また情報を利用する部門における情報利用プロセス抽出の漏れを防止できる。

(c) 分岐が発生する場合、それぞれの発生頻度（滅多に発生しない事象かどうか）、作業負荷、時刻・季節などにより発生頻度に変動がある場合にはその分布を確認する。これを実施すると、システム化対象に取り込むかどうかを決める際に役立つ。開始条件、終了条件などがシステム化の際に重要な意味を持つ分岐は、その条件を明確にする。

(d) 業務プロセスフロー作成時に調査した業務データ締め切りタイミング、データ発生頻度・サイクル等については、非機能要件に処理データ量として記載する。

(e) 業務フローをシステム開発者が作成すると、システム部分だけのフローになりがちである。あくまで業務（仕事）の流れであることを意識し、業務全体の流れについて記載する。例えば、一連の業務を、担当者が一度に実施する作業の単位で

分解し、業務フロー上にそれぞれの単位で業務とシステムの機能を漏れなく記載する。

(f) システム化、データベース化されていない紙の台帳等も省略せずに記載する。

(g) 外部のエンティティ（銀行、顧客、監査人、輸送業者、官公庁など）を含め、それぞれのエンティティでの情報の使用目的に適合するかどうかを検討して業務フローを作成すると、要件の確認がしやすくなる。例えば、顧客が税務申告に使用する資料であれば、確定申告時期まで検索性を高めて保存し、必要に応じて再作成できることが「要件」となる。

(h) 担当部署（アクタ）は、具体的な組織名ではなく、役割（ロール）で記載する。

(i) 権限により業務フローが変わる可能性があるため、作業権限によってどのように業務フローが異なるかについても確認し、必要に応じて業務フローに記載する。

(j) 通常は To-Be の業務フローだけでよいが、As-Is と大きく変わり、As-Is との対比、確認が重要な場合は、As-Is も作成する。作業レベルの業務フローについては、作業レベルの変更が新システムにおける重要な変更ポイントになっている部分、例えば手順の簡素化により作業効率を大幅に高める部分などがあれば、記載する。

(k) 通常処理の処理量、例外処理、特殊処理の発生頻度など主要な業務量は、漏れることがないように非機能要件定義の処理データ量の箇所に記載しておく。

(l) 用紙 1 枚に業務全体が把握できる業務フローも作成すると、業務全体を俯瞰でき、気づきが得られることがある。

(m) 業務が複雑な場合には、フローを階層化して記載することにとり、見やすさを確保することも必要になる。

(n) 昨今は、現行業務がシステムに隠蔽されたり、業務知識が分散しているなどの理由により As-Is が明確になっていない事が多いため、As-Is 業務フローを記載して業務を理解した上で、それを修正しながら To-Be にしていくことが望ましい。

コラム　～新しいビジネスプロセスを作り上げる意識を持つ～

【課題認識】

「ビジネスプロセス改革の期待のもと To-Be フロー図を作成したが、ビジネスプロセスがそう大きくは変わらなかった」こんな経験はないでしょうか？

今日では、企業の大半の業務には何らかの形で IT システムが導入済になっています。その状況で新たな投資をして IT システムを導入するのは、単なる手作業の効率化ではなく、ビジネスプロセス自体の見直しを望んでいるからであるという企業は少なくないでしょう。IT システム再構築によって箱（ハードウェア）を新しくしても、ビジネスプロセスをまったく変化させないのはナンセンスと言われています。

企業におけるビジネスの実態を一番意識しやすいのは、業務フローなどのフロー系ドキュメントです。しかし、フロー系成果物は精度を上げて具体的に記載すればするほど量が多くなり、作成には時間も掛かるため、対象をすべて記載するまでで精一杯になり、内容に対する議論が疎かになる傾向にあります。

【解決案】

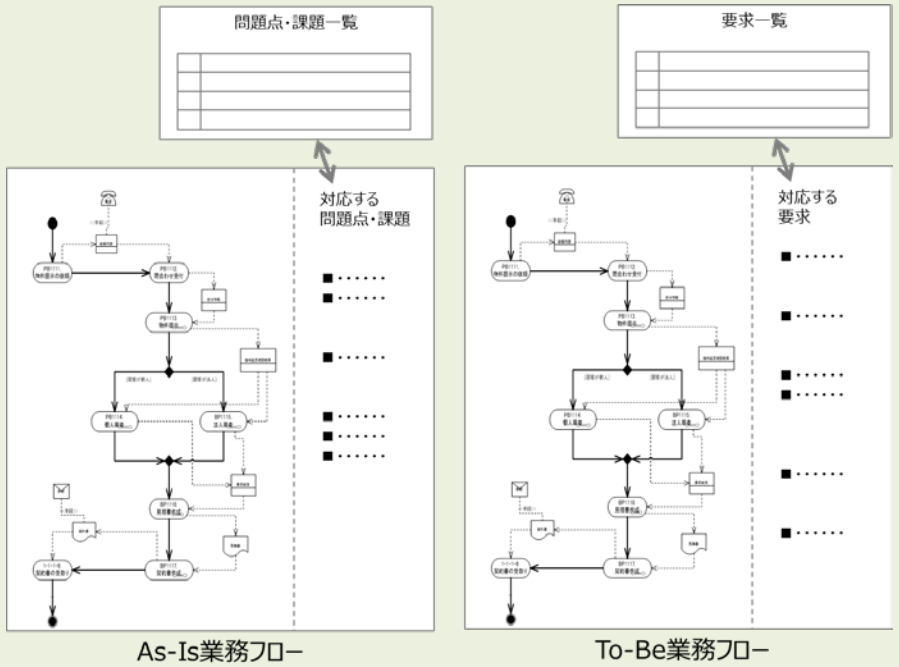
作成者が経営的視点でビジネスプロセスを改革、改善するという意識を持って、業務フロー図などのフロー系成果物の作成に臨むことが重要です。

ビジネスプロセスを大きく改革、改善するような要求は、経営層や事業部門長などから事業戦略や方針として示されます。現場の担当者から課題を収集しても、これらの要求は提示されてきません。それは、現場の担当者にとっては、自身の直面する作業の問題点を解消したいという視点からの要求が日常の関心事であるからです。したがって、フローを検討する際には、トップダウン・ボトムアップ双方から提示される相反する要求からいかにゴールを見出すかがポイントとなります。

改革・改善を意識させるための簡単かつ有効な方法は、フロー図に要求・課題の欄を設け、それらとマッピングさせながらフローの検討を進めることです。

- As-Is フローを作成する
- 経営レベルの課題を As-Is フローにマッピングする
- 現場レベルの問題点を As-Is フローにマッピングする
- 新たな問題点・課題がないか検討し、分析を補完する
- As-Is フロー図にもとづいて問題点・課題の共通認識、合意形成を行う
- 経営レベルの目的・施策を To-Be フローにマッピングする
- 現場レベルの目的・手段を To-Be フローにマッピングする
- マッピングした内容をもとに To-Be フロー図を作成する
- To-Be フロー図をもとに施策、手段の共通認識、合意形成を行う

以下の図①に問題点・課題一覧、要求一覧とフロー図とのマッピングイメージを示します。



図①　問題点・課題一覧、要求一覧とフロー図とのマッピングイメージ

プロセスフローの検討において使用される 3 階層のフロー図（ビジネスプロセス関連図、業務フロー、システム化業務フロー）のうち、ビジネスプロセス関連図では、より経営レベルに近いビジネスプロセスの改善が表現でき、業務フロー、システム化業務フローでは、より現場レベルに近い細かいビジネスプロセスの改善が表現できます。プロセス改革の検討においては、ビジネスプロセス関連図から検討を始めましょう。

要件定義において To-Be フロー図を作成するのは当然ですが、フロー図上に問題点・課題や要求を記載しているプロジェクトは多くありません。

ぜひ、フロー図上に問題点・課題や要求をマッピングしたフロー図を用いて、新しいビジネスプロセスの検討を行ってください。

また、システム部門（またはそこから委託されたベンダ企業）がフロー図を作成すると、現行システムをベースに新たな要件への対応を追加するというアプローチで検討を行った結果、ビジネスプロセスの変革が期待したほどでなかったという結果を招くことになりがちです。それを防止するためにも、フロー系成果物は、利用部門が主体となって検討をしてください。

4.2.4 画面／帳票レイアウト

画面、帳票の標準化

特に一つの企業が複数の業務を個別のプロジェクトでシステム化しようとした場合に、システムの設計から構築、運用に至る各工程の作業について、最初のシステム開発プロジェクトが始まる前に予め決められるものは標準として決めておくと、各プロジェクトで決めるべきことが減り、システム開発の生産性・品質の向上および開発コストの低減に寄与する。

画面については、画面設計、画面遷移を標準化することで、個別に検討する範囲が少なくなる。同様に帳票についても、何を帳票で出すか、その場合のレイアウト等の扱いを事前に決めておく。

業務面では、業務部門からすると、どのシステムの操作性も同じとなり、業務の効率化や操作ミス防止につながる。さらに、部門間の異動で利用するシステムが変わっても、画面の配置、操作はあまり変わらないなど、会社全体としての業務の効率向上が期待できる。

画面レイアウト

(1) 目的

- 業務を遂行するためのデータ項目（入力項目、表示項目）が業務部門の要求と合っているかを確認する。
- 画面操作方法、画面遷移が、業務部門の要求と合っているかを確認し、どのように入出力すると使いやすいかを検討する。

(2) 説明

業務で使用する画面のレイアウト、入出力項目、画面遷移を作成し、業務部門の確認をとる。

図 4.7 に画面レイアウトの例を示す。業務処理メニューは画面の左側に、新規（登録）、検索、変更（更新）、削除といった操作は画面の右側に、エラーメッセージは画面の下に配置している。

画面の左側のメニューは、操作者が業務実行のために必要とする機能のメニューを示している。基幹系システムの場合の処理は、新規（登録）、照会、変更、削除に分かれるため、左側画面の一番上の行にそれらの処理を選択できるようにしている。図 4.7 は明細を一覧形式で表示する事例を示している。

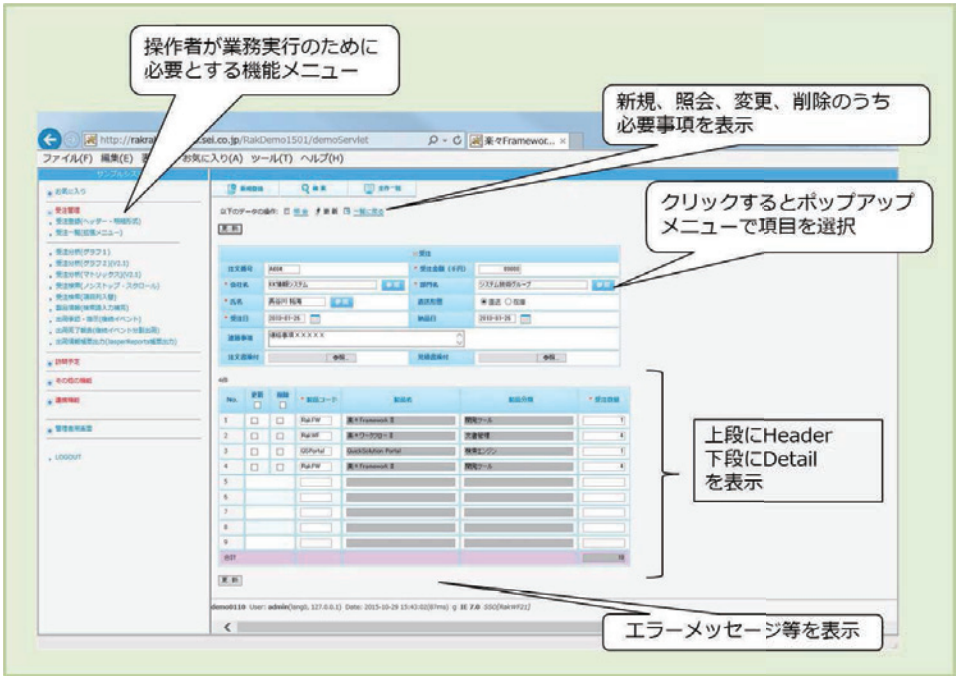


図 4.7　画面レイアウトの例

さらに、主要な画面遷移図も、標準化すると、各プロジェクトで決めるべきことが減少する。図 4.8 に、ある企業の画面遷移の標準例を示す。この画面遷移は、処理のプロセスと機能（登録、照会、変更、削除）の関連を標準化している。照会、変更を例にとると、検索条件を入れると一覧が表示され、さらにその明細を照会できる。変更があれば、変更画面に遷移し、変更するところまで標準化している。

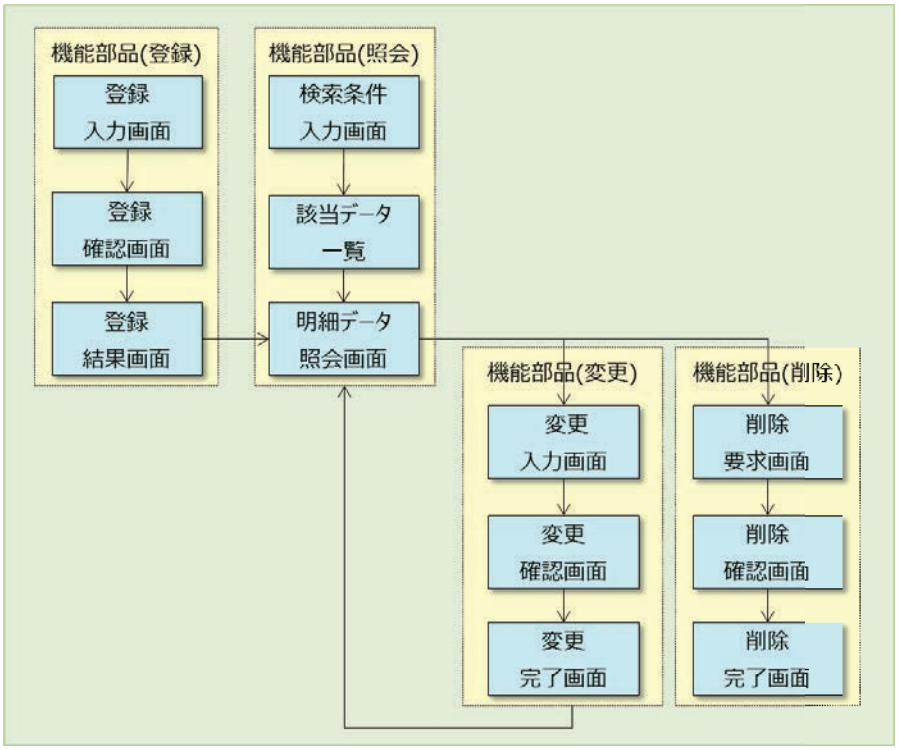


図 4.8　画面遷移の標準例

(3) 留意事項

① 利用目的や使い方を明確にする。

各画面の利用シーン（どの業務でどのような人が使用されるのか、業務の習熟者か、初心者か、顧客かなど）を明確にする。各画面には、当該画面が取り扱うデータの種類や、参照なのか更新なのかなど、目的用途が明確に分かるような名称を付与することを命名規約で定めることが必要である。

また、利用者の権限により、画面に表示する内容や処理などが変わる可能性があるので確認をしておく。想定利用者や検討経緯、利用シーンが明らかになっていれば、基本設計時に、各画面に配置する項目の選択・項目の並び順・読みやすさ・使いやすさの検討での誤りの発生を減らすことができる。

② 利用目的や操作性を考えた画面レイアウトになっているか確認する。

利用目的によっては、画面デザイン時に、企業のブランドカラーや企業ロゴなどに配慮することも必要となる。

- 短時間に多くのデータを入力する業務（例えば、伝票を短時間で 10 キーだけで高速に入力する業務）に、複数の選択肢から選ぶチェックボックスや階層を順次掘り下げるドリルダウンなどのサポート機能を駆使した画面を設計しない。
- 顧客や初心者が使用するために使いやすさが特に重視される画面には、画面レイアウト、画面遷移、入力補助機能を漏らさずに記載し、使いやすさに関してユーザの要件確認の手戻りがないようにする。

③ 画面仕様は要件定義の段階で業務部門と合意する。

- 入力チェック仕様などは、データ項目定義書に記載された条件以上のものがなければ、この段階では不要である。しかし、チェック機能が特に重要な場合は、確認のために記載することが望ましい。また、XXX や 9999 などの桁数やデータ属性表現ではなく、具体的なデータの例で表現することにより、ユーザの理解を確かなものとする。ただし、最大表示桁数の確認のためには、XXX や 9,999 のような表示も必要。
- 画面レイアウトや画面遷移は、ツールを活用してイメージや使いやすさを確認する。ツールを使用すれば比較的容易に簡易図（モックアップ）が作れるので、この段階で、簡易図（モックアップ）で業務部門に画面の操作性、画面遷移、補助機能などを確認してもらうと手戻りが少なくなる。

④ なぜこの画面レイアウトになったかの検討経緯を残す。

画面レイアウトの結論のみの記述の場合、後続工程でなぜこの画面レイアウトになったか、検討経緯がわからなくなり、技術的な問題等で修正の必要が生じた場合に判断がつかなくなり、工期に遅れが生じるといった問題が起きる。検討経緯を画面レイアウトの注釈欄などに残しておくとよい。画面レイアウト決定までの経緯を記録に残しておくと、何の目的のための画面かが確認でき、責任者や担当者（窓口）が変わった時にも担当業務の画面がどうしてこのようなレイアウトになったのかをより正確に伝達できる。

⑤ B2C の業務や社外公開 Web ページなどについては、要件定義の段階で画面に表示する項目がほぼ確定するようであれば、リーガルチェック（法規違反や知財侵害等のチェック）を行う。

帳票レイアウト

(1) 目的

帳票の機能要件について、利用者と開発者の合意形成を図る。

帳票レイアウトに記載された出力項目の情報や業務フローなどをもとに、利用者側は出力帳票がシステム化の目的達成に役立つかを確認する。

(2) 説明

帳票レイアウトを作成し、出力項目の過不足、表示形式、表示順序などについて、業務部門に確認する。帳票はこれまでは紙に印刷していたが電子化が進んでいる。帳票の提供形態を検討する際には、電子化を行うかを含めて検討する。

(3) 留意事項

① 紙で出力するか、電子化するかを考える。

従来は、紙媒体で実施してきた業務処理を、画面を使用したワークフローで済ませ、業務効率化とペーパーレス化を実施するケースが増えてきている。業務の電子化により、業務簡素化、業務処理時間の短縮、出力紙媒体の削減の効果が得られることに加え、帳票をデータとして保管することにより、検索、加工、分析等による業務効率化（高度化）を進めることができる。

② 帳票については、独自フォーマット等による完全な電子化以外にも、コストを意識して別の選択肢を取ることも検討する。

(a) 画面のハードコピーで済ませられるものはそれで済ませるくらいに割り切る

(b) ビジネスの変化にあわせて業務で必要とする情報が変化すると、業務部門からは効率化のためにこのような情報を出して欲しい、配置をこうして欲しいなどの変更要求が発生する。これに対しては、帳票データを CSV などの形式でファイルに出力し、業務部門で適切な形式に加工してもらうなどの対応を検討する。これにより、利用環境の変化に柔軟に対応することができ、帳票設計、実装の大幅な工数削減も同時に実現できる。

③ 利用目的や利用シーンを想定して帳票設計をすること、また用途が明確に分かる帳票名を付与する。

(a) 現場での利用シーンに合わせて帳票を設計する必要のある例

(i) 拠点別営業業績の管理帳票 ： 拠点別の営業業績を、明細⇒小計⇒中計⇒総計、という管理帳票で出力することにした。これを業務部門のマネージャに見てもらったところ、「この帳票リストは、現場ではまず総計を見て、例えば近畿

地方の売上は前年比でどうかとまず確認する。前年比で下回っていると、どの支店の売上が減少しているか、その支店を商品別に見るとどうかと、掘り下げて行く。」との使用方法であり、総計⇒中計⇒小計⇒明細の順で見られるように情報の出力順番を決定する必要があることが分かった。

(ii) 部門での配布の便宜を考慮して、部門別／個人コード別のリストを配布する場合、部門で個人ごとの切り取りが必要になるが、控えの保存性を高めるため、個人ごとの改ページは行わない」などの注釈があると分かりやすい。

(b) 用途の分かる名称の例

「契約明細リスト」ではなく「当月未入金契約者抽出リスト」にすると、用途や利用シーンが分かりやすい。

④ 帳票のレイアウトの確認においては、現実のデータに近いデータ例で表示することにより利用者との理解の齟齬をなくすことができる。

全部の帳票レイアウトを作成することが望ましいが、以下の内容でもよい。

(a) 新設帳票の場合は、帳票レイアウト、項目説明をすべて記載する

(b) 既存帳票の場合は、既存帳票のレイアウト図を利用して、変更部分の説明を加える

⑤ お客様先など外部に提供する帳票については、要件定義の段階で帳票に表示する項目がほぼ確定するようであれば、リーガルチェック（法規違反や知財侵害等のチェック）を行う。

⑥ なぜこの帳票レイアウトになったかの検討経緯を残す。

最終帳票だけが成果物として残ると、検討経緯がわからなくなるため、何故この名称になったかなど、検討経緯も分かるようにする。

⑦ 帳票設計の標準化

帳票を作成する場合は、その扱いの標準を事前に決めておく。表 4.5 に帳票設計の標準化をしっかり行っているユーザ企業の例を示す。ここまで標準化（ルール化）しておくと、要件定義の漏れの防止や利用者の業務効率向上に役立つ。

表 4.5　帳票設計の標準化の例（1/2）

区分	標準化の内容
全体	(1) 使用目的・使用方法に沿った項目の配置や位置等を考え、業務が効率的に行われるようにする。 (2) 事務の流れに沿った項目配置(押印欄・記入欄) 担当者 1⇒担当者 2⇒承認者⇒決裁者　の事務の流れであれば、例えば上から下に記入欄が埋まって行き、承認に必要な記載項目を読み終える位置に押印欄(承認ボタン)を配する。 担当者 1 記入欄　担当者 1 確認欄 担当者 2 記入欄　担当者 2 確認欄 承認者付記　　承認欄 認証欄　等 (3) 顧客・ユーザからみて、不快な感情を持たないよう感受性に配慮した用語を用いる。 例：×：受付不可、受付不能　でなく、「再度ご連絡ください」など (4) 定義および範囲の明確な用語を用いる。 (5) ファイリング・郵便等規格に配慮する。 ・棚や倉庫、格納箱などの社内規格に基づき、「A4 縦左綴じ標準」「A3 の場合、横置き左綴じ」「両面の場合は長辺方向」「パンフレットや顧客用資料は A4 二つ折り、綴じなし」など。 ・郵便は、「定型 X 版に三つ折りで格納できるサイズ」や、「X 枚で n グラム以内の重さの用紙」「定型はがき(料金後納)のこの標準書式」など。 (6) 帳票コストの削減に配慮（耐用期間と緊急再実行（リラン）を配慮した在庫管理含む)する。 ・専用紙でなくてもよいものは、専用紙にしない。 ・帳票の重要度・緊急度に応じて、「全量 1.5 回分の再作成を考慮して在庫量を決める」等。例えば支払調書なら、もしミスがあれば短時間で再作成することを要するが、業績統計なら、帳票用紙再発注後の再作成でも問題ない等。 ・一方、カプセルのり帳票や香りつき帳票など、使用期限が数か月程度のものもあるので、在庫量が多すぎると大量廃棄につながるなど。

表 4.5　帳票設計の標準化の例（2/2）

区分	標準化の内容
デザイン	(1) 罫線：文字との間隔、罫線種類、罫線・網掛けの使い分け。 (2) フォント：使い分け、1 帳票内種類制限、項目種類別標準文字属性等。 (3) 文言：使用禁止用語、文体(例. 顧客向け帳票で「～すること」は不可。「～してください」と、お願い調にする)、数量・金額・日付形式を標準化しておく。 (4) 項目配置：同時使用(照合)伝票と順序を合わせる、検索・計算項目を右下に配置など。 例：伝票をめくりながら、伝票の金額と帳票の金額をチェックするため、金額を右下に配置する
その他	(1) 推奨出力枚数(1 枚がベスト)。 (2) 用語：用語(標準用語、顧客向け用語、禁止用語)に従う。 例：顧客向けの帳票に、申請、徴収などの表現は使わない (3) 標準出力：内容と順序。 ・処理名（プログラム、プロセス)、印刷年月日・時間等 ・再出力や抜けの確認のためのシリアル番号の付与 (4) 見出しの書き方や文体を決めておく。 例：体言止めする、ですます調にするなど

4.2.5 業務処理定義書

(1) 目的

業務理解を促進させるために、各業務処理機能の内容を明確にする。業務改革や改善の詳細内容の認識を共有する。

(2) 説明

- 業務フローの各業務処理機能の内容を明確にする。
- 業務フローは業務機能の流れを示した図であり、個々の業務機能におけるシステムの処理には言及していない。業務処理定義書は、業務フロー上の個々のボックス(業務機能)単位に作成され、業務機能とシステム機能を関係付ける成果物となる。

(3) 留意事項

① 業務の背景、目的、取り扱い範囲・制約事項を明確にしておく。

ユーザのニーズや、特別な業務処理が存在する背景や理由を理解しておくと、漏れや不整合を摘出しやすい。そのため、業務処理には理由を記載しておく。また、発生数や発生頻度の片寄りなどがあれば、その理由も記載しておく。業務処理が存在する理由を書くことにより、この業務は何のために行う業務かを確認でき、要件定義漏れ(システムの仕様誤り)の防止に繋がる。

② 重要な時間や日付、発生・完了タイミングを確認する。

業務処理が発生するタイミングや完了するタイミングや、他の業務の発生との整合性などに重要な違いや留意すべき点があるケースは、業務の実施時間やタイミングを業務処理定義書に記載する。

③ 重要な業務上のチェック処理も確認する。

要件定義段階では、システム例外、データ例外、運用例外などアプリケーションの処理詳細に関係する例外は除外してもよい。業務系の例外(誤発注を防止するために、発注数量が発行済株式数の 20%を超えないことなどの業務上のチェック処理)については、それらがシステム化の目的やニーズになっている重要な処理である場合は、業務処理定義書に記載する。

④　表形式にまとめると漏れや不整合を発見しやすい。

業務仕様をまとめる際には、表 4.6、表 4.7 に示すように表形式を活用するとよい。表 4.6 は図書館システムの会員種別によるサービスの違いを示す。有料会員と無料会員のサービスの違いを文書で説明するより明確に伝えることができる。表 4.7 は、会員種別と会費納入状況、返却日延長回数による返却延期、および追加貸出の可否の判断を示す。

このように表形式にすることにより、業務仕様を条件別に明確に区別して確認でき、漏れ、不整合を発見しやすくなる。なお、3 次元以上の要素がある場合は、1 次元目を分けて 2 次元の表で作成する。それ以上になる場合はデシジョンテーブルなどを使って、網羅するのがよい。

表 4.6　図書館システム–会員種別によるサービスの違い(1 次元の例)

(判定条件)

会員種別	会費	貸出冊数	延長回数	ネット予約	ネット延長	延長時の貸出制限
無料会員	なし	3 冊	1 回まで	不可	不可	あり(不可)
有料会員	あり	5 冊	2 回まで	可	可	1 回まではなし

表 4.7　図書館システム–会員の状況と貸出延長回数による新規貸出制限(3 次元の例)

(判定条件3)

(判定条件1) (判定条件2)

会員種別	会費納入状況	判定項目	現在貸出中の図書の返却日延長回数			
			なし	1回	2回	3回以上
無料会員	–	返却日延長可否	延長可		延長不可	
		追加貸出可否	合計3冊まで貸出可能	追加貸出不可		
有料会員	滞納なし	返却日延長可否	延長可			延長不可
		追加貸出可否	合計5冊まで貸出可能		追加貸出不可	
	会費督促中	返却日延長可否	延長可		延長不可	
		追加貸出可否	合計3冊まで貸出可能	追加貸出不可		

⑤　業務の要件定義において、業務担当者から「この部分は現行どおり」と言われることがある。しかし業務側も業務の詳細部分を把握していないことが多く、現行システムのドキュメントも最新でない場合も多い。「現行どおり」と言われても、要件定義においては明確な仕様として記述しなければならない。

⑥　改造の場合は、新旧の違いが分かるように、補足説明、または参考資料として旧設計書を残す。

⑦　セキュリティ上の制約（例えば、顧客情報など特定の情報に対するアクセス制限、アクセスログの要否等）を確認する。IT 統制や業務監査の証跡として残す日付やデータに留意すべきものがあれば、記載する。

⑧　障害時の代替方法も、あらかじめ決定しておくのがよい。

⑨　管理機能を確認する。
使う人（操作者、利用者など）の視点だけでなく、管理者の立場からの機能の組込みが必要か否かを確認する。

⑩　定義された業務要件は、検討経緯を補足説明または参考資料として残しておくと、後続工程での検討に役立つ。

⑪　定義された業務要件には、各々ID 番号を付与し、システム機能の ID 番号と関係付けした一覧表を作成しておくと、業務機能とシステム機能の関連の把握に役立つ。

参考

USDM は、要求と仕様を階層化して表形式で表現し、各要求と理由（要求の存在理由）をペアで記述する業務処理定義書の作成方法である。表 4.8 に、業務処理定義書の作成例を示す。階層化された表形式で要求と仕様が紐付き、さらに理由を書くことにより、理由が適切でない要求仕様が明確化される。これにより、仕様漏れ・誤りを減少させられる。

表 4.8　業務処理定義書の作成（表 4.7 の業務処理を定義した例）

<table>
<tr><td rowspan="30">貸出延長可否判定</td><td rowspan="3">要求</td><td>KS01</td><td colspan="3">貸出延長請求、追加貸出請求を受理してもいい状態かどうかの判定と追加貸出可能冊数の算出を、会員情報、図書貸出情報をもとに実施する。</td></tr>
<tr><td>理由</td><td colspan="3">本の貸出期間の延長可否や追加貸出可能冊数を会員向け貸出規約にもとづいて確認したい。</td></tr>
<tr><td>説明</td><td colspan="3">会員向け貸出条件は以下のとおり
<table><tr><th rowspan="2">会員種別</th><th rowspan="2">会費納入状況</th><th rowspan="2">判定項目</th><th colspan="4">現在貸出中の図書の返却日延長回数</th></tr><tr><th>なし</th><th>1回</th><th>2回</th><th>3回</th></tr><tr><td rowspan="2">無料会員</td><td rowspan="2">-</td><td>返却日</td><td colspan="2">延長可</td><td colspan="2">延長不可</td></tr><tr><td>貸出</td><td colspan="2">合計3冊まで貸出可能</td><td colspan="2">追加貸出不可</td></tr><tr><td rowspan="4">有料会員</td><td rowspan="2">滞納なし</td><td>返却日</td><td colspan="3">延長可</td><td>延長不可</td></tr><tr><td>貸出</td><td colspan="2">合計5冊まで貸出可能</td><td colspan="2">追加貸出不可</td></tr><tr><td rowspan="2">会費督促中</td><td>返却日</td><td colspan="2">延長可</td><td colspan="2">延長不可</td></tr><tr><td>貸出</td><td colspan="2">合計3冊まで貸出可能</td><td colspan="2">追加貸出不可</td></tr></table></td></tr>
<tr><td rowspan="3">要求</td><td>KS01.1</td><td colspan="2">会員IDをもとに、有料/無料の区別、会費納入状況、貸出中図書の冊数、貸出中の図書に対する返却期日延長状況を取得する。</td></tr>
<tr><td>理由</td><td colspan="2">貸出延長を実施できる状態かどうか、および追加貸出可否および追加貸出可能冊数を判定するために、必要な情報を取得したい。</td></tr>
<tr><td>説明</td><td colspan="2">ー</td></tr>
<tr><td colspan="4"><判定に使用する情報の取得></td></tr>
<tr><td></td><td colspan="3"><会員情報取得></td></tr>
<tr><td>□□□</td><td>KS01.1-01</td><td>カードから読み取った会員IDをキーに会員マスタから以下を取得する。
・有料／無料の区分
・会費納入状況を取得する</td></tr>
<tr><td></td><td colspan="3"><貸出状況取得></td></tr>
<tr><td>□□□</td><td>KS01.1-02</td><td>カードから読み取った会員IDをキーに図書貸出テーブルから貸出中図書の情報を取得する
・貸出中図書の冊数
・各貸出中図書のID
・各貸出中図書の返却期日延長回数</td></tr>
<tr><td></td><td colspan="3"><返却期日延長回数取得></td></tr>
<tr><td>□□□</td><td>KS01.1-03</td><td>貸出中の図書の返却期日延長回数のうち、もっとも多い回数を取得する</td></tr>
<tr><td rowspan="3">要求</td><td>KS01.2</td><td colspan="2">貸出延長請求を受理できる状態かを判定する</td></tr>
<tr><td>理由</td><td colspan="2">取得した情報と会員向け貸出規約から、本の貸出期間延長可否を知りたい</td></tr>
<tr><td>説明</td><td colspan="2">ー</td></tr>
<tr><td colspan="4"><貸出期日延長可否判定></td></tr>
<tr><td></td><td colspan="3"><可否判定></td></tr>
<tr><td>□□□</td><td>KS01.2-01</td><td>無料会員かつ貸出中の図書に対する返却期日延長が1回以下 → 延長可能</td></tr>
<tr><td>□□□</td><td>KS01.2-02</td><td>無料会員かつ貸出中の図書に対する返却期日延長が2回以上 → 延長不可</td></tr>
<tr><td>□□□</td><td>KS01.2-03</td><td>有料会員かつ会費延滞なし、かつ貸出中の図書に対する返却期日延長が2回以下 → 延長可能</td></tr>
<tr><td>□□□</td><td>KS01.2-04</td><td>有料会員かつ会費延滞なし、かつ貸出中の図書に対する返却期日延長が3回 → 延長不可</td></tr>
<tr><td>□□□</td><td>KS01.2-05</td><td>有料会員かつ会費延滞あり、かつ貸出中の図書に対する返却期日延長が1回以下 → 延長可能</td></tr>
<tr><td>□□□</td><td>KS01.2-06</td><td>有料会員かつ会費延滞あり、かつ貸出中の図書に対する返却期日延長が2回以上 → 延長不可</td></tr>
<tr><td rowspan="3">要求</td><td>KS01.3</td><td colspan="2">追加貸出可否の判定と追加貸出可能冊数の算出を行う</td></tr>
<tr><td>理由</td><td colspan="2">取得した情報と会員向け貸出規約から、追加貸出可否と貸出可能冊数を知りたい</td></tr>
<tr><td>説明</td><td colspan="2">ー</td></tr>
<tr><td colspan="4"><追加貸出可能冊数算出></td></tr>
<tr><td></td><td colspan="3"><貸出可能冊数取得></td></tr>
<tr><td>□□□</td><td>KS01.3-01</td><td>無料会員かつ図書に対する返却期日延長が0回 → 上限3冊まで貸出可</td></tr>
<tr><td>□□□</td><td>KS01.3-02</td><td>無料会員かつ図書に対する返却期日延長が1回以上 → 追加貸出不可</td></tr>
<tr><td>□□□</td><td>KS01.3-03</td><td>有料会員かつ会費延滞なし、かつ貸出中の図書に対する返却期日延長が1回以下 → 合計5冊まで貸出可</td></tr>
<tr><td>□□□</td><td>KS01.3-04</td><td>有料会員かつ会費延滞なし、かつ貸出中の図書に対する返却期日延長が2回以上 → 追加貸出不可</td></tr>
<tr><td>□□□</td><td>KS01.3-05</td><td>有料会員かつ会費延滞あり、かつ貸出中の図書に対する返却期日延長が0回 → 合計3冊まで貸出可</td></tr>
<tr><td>□□□</td><td>KS01.3-06</td><td>有料会員かつ会費延滞あり、かつ貸出中の図書に対する返却期日延長が1回以上 → 追加貸出不可</td></tr>
<tr><td></td><td colspan="3"><追加可能冊数取得></td></tr>
<tr><td>□□□</td><td>KS01.3-07</td><td>追加貸出が可能と判定された場合、以下の式により貸出上限冊数を計算する
追加貸出可能冊数 = 上記により取得した上限冊数 - 貸出中図書の冊数</td></tr>
</table>

4.2.6 概念データモデル（ER 図）

(1) 目的

実業務の管理対象や業務ルール等を分析・モデル化し、企業のビジネス活動の構造を理解する。

(2) 説明

① ER 図（Entity-Relationship Diagram）

データモデルの図法には ER 図を用いる。

ER 図は、受注、顧客、商品といった業務を構成する「もの」や「こと」であるエンティティ（Entity）、エンティティ同士の関連（Relationship）、およびエンティティを構成するデータ項目（Attribute）を図示したものであり、業務の実態（業務で扱うデータ構造）を表現する。

従来の考え方では、ER 図はデータベース設計のための手法として基本設計以降に登場するが、要件定義段階で概念レベルの ER 図を導入し業務の実態を把握しておくと、基本設計段階で要件定義工程に遡ることを少なくすることができる。

② ER 図の作成手順

業務システムは、業務の管理対象を扱うものであり、業務で扱う管理対象であるデータやそのデータに関する業務ルールを調査し、エンティティとその関連として構造化した ER 図は実業務の管理過程をよく表している。ER 図の作図工程はある程度ルール化されているため、システマティックに進めることが可能となる。このように、ER 図を作成することで、業務システム全体を鳥瞰でき、自然言語で要件を記述する場合に比べ、あいまいな記述や漏れを少なくし、要件定義の精度を向上させることができる。

ER 図を描くにあたっては、業務ルールや業務実態のヒアリング、および既存の画面、帳票やドキュメント（業務マニュアル、システム設計書等）を収集する。そこから収集した情報をもとに、ER 図を描くことで、業務の管理対象や業務ルール等の事実関係を確認していく。

As-Is と To-Be の両側面で業務を分析する。As-Is と To-Be の作成目的が異なるため、それぞれの目的を意識して作成することが望ましい。As-Is 概念データモデルの作成目的は、現状の業務の問題や課題を発見し、モデル上のエンティティ、関連と対応付けることにより業務を理解することである。一方で、To-Be 概念データモデルの作成目的は、新しいモデルと施策の対応関係を対応付けることにより業務プロセス改革内容の合意を図ることである。

③　ER 図作成にあたってのチェックポイント

ER 図はシステム化の管理対象や業務ルール（What）を表現した静的モデルである。What は実現のための機能・処理や手順（How）検討するための前提であり、What の分析・設計に専念することでビジネスの本質を網羅的に検証できる。具体的には下記のような点を検証する。

(a) 描かれているエンティティが実業務の中で管理対象として管理されている事実があるか。1 つのエンティティに扱いの異なる複数の管理対象が含まれていないか。

(b) 描かれているリレーションの多重度（1 対 1、1 対多、多対多）を確認する。

(c) 描かれているリレーションの対象について全てのデータか漏れなく関係しているか、関係していないデータがあるか。

(d) 描かれていないエンティティ間のリレーションが本当に存在しないか。

これらを精査することにより、担当者からの業務実態の確認漏れをチェックすることができる。

④　ER 図のチェック方法

企業の組織・従業員を例に、要件の漏れのチェック方法を説明する。

(a) 当初の組織・従業員の ER 図は、図 4.9 であった。

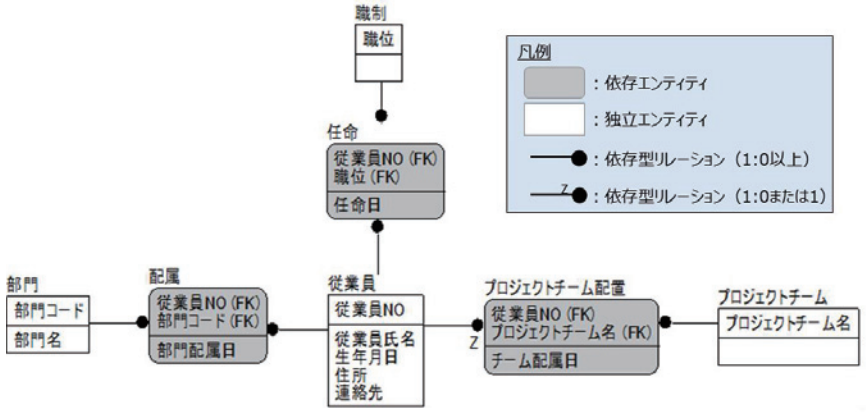


図 4.9　当初の ER 図

(b) 図 4.9 の各エンティティ間の多重度に着目して、業務の関連を確認する。

- 職制と従業員の関係は、1：多なのか、従業員が 1 つ以上の職位に任命されることはないか。
- すべての従業員に職位が付与されているが、パート社員、アルバイト社員などの短時間労働者に職位はあるのか。
- 配属の履歴を保持する必要があるか。

これらの確認を行うことで、図 4.10 の(ⅰ)～(vii)に示すように、次の事実関係が分かる。

(i) 従業員には正社員とパート・アルバイトがある。正社員は職位の付与、部門への配属、プロジェクトへの配属があるが、パート・アルバイトはない。

(ii) 従業員は現在在籍している従業員に加えて退職した従業員もスコープに含めて把握したい。

(iii) 正社員は同時に複数のプロジェクトに従事できる。プロジェクトチームにはプロジェクトコードが付与され、プロジェクトコードでプロジェクトチームが管理されている

(iv) 従業員は本務に加えて他部門への兼務がある。

(v) 職位には資格と役職があり、それぞれ辞令発令がある。役職とは、部長、課長、係長等を表し、資格とは技術スタッフ、1 級、2 級などを意味し、基本給のベースとなる。

(vi) ある部門では部長だが、別の部門では課長というように部長兼課長のような兼任がある。

(vii) 社内組織（部門）は事業部の下に部、部の下に課等の階層関係が存在する。

(c) 上記の変更を反映した ER 図は、図 4.10 のようになる。

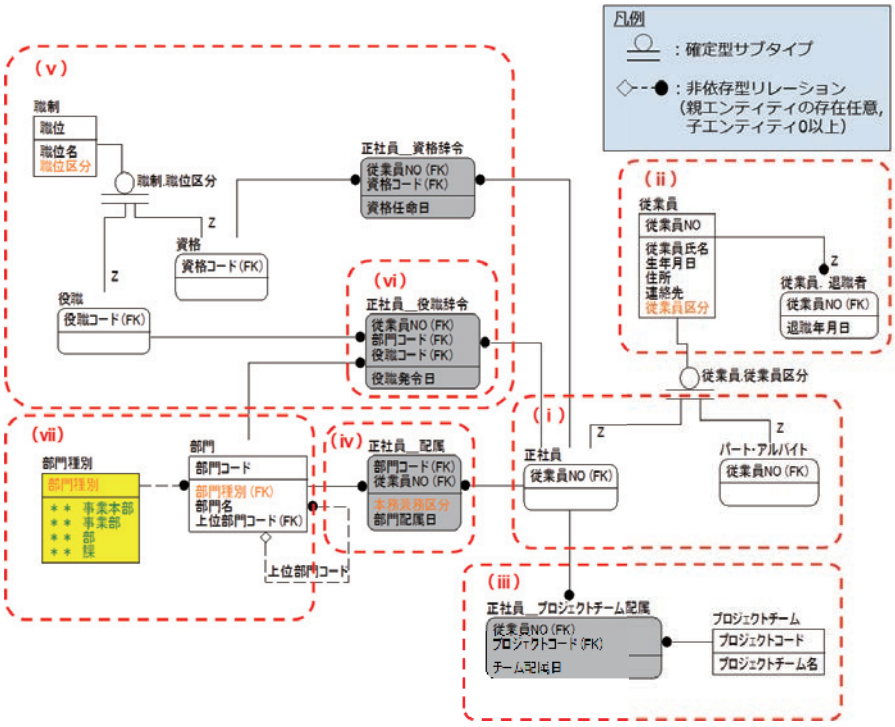


図 4.10　変更を加えた ER 図

(3) 留意事項

① As-Is 概念データモデルから作成する。
概念データモデルのエンティティは業務フローとは異なり、安定しており、フローほど大きく変わらない特徴がある。よって、As-Is 概念データモデルを書いた上で、その後に To-Be 概念データモデルとして修正する手順が効率的である。また、概念データモデルは、短時間で正確に業務を理解するのに適したモデルである。よって、作成担当者は、まず業務を理解するという意味でAs-Is概念データモデルから書くとよい。

② 現状を表す ER 図は業務の実態、事実に忠実に描く。
エンティティを描く場合は、実業務でそのデータが管理対象として運営されていることを確認した上で描く。関連についても確認したものだけを描く。SE の経験からこうあるはずだと描いてしまうと、事実とは違う都合のよい ER 図ができてしまう。

③ 作成した ER 図には必ず漏れ、誤りがあるという前提で網羅的に検証を実施する。
業務部門からのドキュメント（画面、帳票、業務ルール）やヒアリングは現状をすべてカバーしていることはまずない。一旦、描いた ER 図を前述 (2) ③の 4 つのチェックポイントに従い網羅的に検証して事実確認の漏れを摘出する。

④ ER 図の読み方は業務部門にも説明し、両者で確認する。
ER 図の作成は、原則としては、システム部門だが、確認は業務部門でも実施できるよう、読み方を覚えてもらう。

⑤ 現在の業務実態にはなくても To-Be として必要ないかという観点でも検証する。
現在は存在しないエンティティとエンティティの関連を検討することにより、業務担当者に気づきを与えることができ、要件の漏れの防止につながる。

⑥ 具体的なデータを入れて検証してみる。
個々のデータ項目に具体的な値を入れて検証すると、抽象的な図では気づかなかった事項が明らかになることが多い。

⑦ イベント（トランザクション系）のデータも ER 図に表現する。
イベント（トランザクション系）のデータも ER 図に表現することができる。

⑧ 属性は別成果物で定義する。
個々のデータの制約条件やアクセス権限などの属性は、ER 図には記述できないのでデータ項目定義書として別途まとめる（4.2.7 節で説明）。

コラム　～概念データモデル（ER 図）を業務部門に確認する～

【悩み】

概念データモデルを業務部門の人に確認するのは難しい。そう嘆いている人も少なくないでしょう。

要件定義で作成する概念データモデルはデータベースの設計図ではなく、本文にもあるように実世界（業務そのもの）を写像したモデルです。本来、業務部門が主体となって業務をモデル化し、概念データモデルとして定義することが理想ですが、概念データモデルの作成は、学習と実践を繰り返さないとなかなかうまくできるようにならないのが現実です。多くのプロジェクトでは、概念データモデルの作成にはシステム部門の有識者か外部の有識者がアサインされていることと思います。

そこで発生することは、作成する側は概念データモデルを作成できるが、業務には精通していないことです。さらに、業務部門側は業務を理解しているが、作成された概念データモデルがそれを正しく写像しているかどうかを確認することができないという状況です。

【解決案】

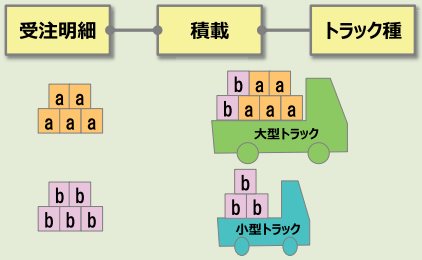
(1)業務部門にヒアリングしながら概念データモデルを作成する

業務部門から業務内容をヒアリングしながら、理解した内容をフローや概念データモデルとしてホワイトボード等に書き示していくことにより、業務部門の人にも概念データモデルが表現している意味を分かってもらいながら作成を進めます。

(2)概念データモデルそのものではなく補助説明資料を用意して確認する

①構造を絵で表現して説明し、内容理解の妥当性を確認していきます。

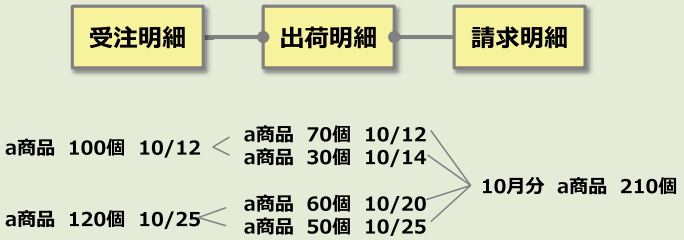
イメージを図①に示します。



図①　概念データモデル補足説明（絵）

この例は、一回の注文で受注した商品を複数のトラックに小分けして積載・配送する処理があるかどうか、そして、量が少ない場合には運送手段を経費の安い小型トラックに変更するような運用が認められているかなどを、絵と概念データモデルで表現したものです。

②構造をインスタンス（値）で表現して確認します。イメージを図②に示します。

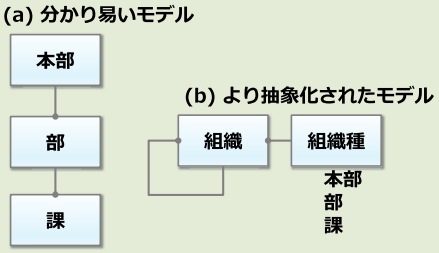


図②　概念データモデルの補足説明（インスタンス）

この例では「一回の受注で受けたａ商品が分割されて出荷され、出荷実績は月別に集計して請求する。しかし 120 個注文しながら当月には 110 個しか出荷しないパターンがあり、そのパターンでは月別の受注と出荷、請求の数量は一致しない」という業務ルールを、インスタンス（実数値例）で表現するシーンを表しています。

(3)抽象化（最適化）を急がない

概念データモデルの理解を難しくしている原因のひとつに、抽象化によるモデルの最適化という作業があります。実ビジネスを概念データモデルに正確に描き出すためには避けて通れない作業ですが、その検討は難しく、出来上がったものも業務部門には直感的に理解されにくいものになります。例を図③に示します。



図③　概念データモデルの抽象化レベルの違い

この例では、「本部」「部」「課」で階層化されている組織をモデル化しています。 (a)は組織の構造を実体どおりの縦の関係で表現しているのに対して、(b)は組織の構造を繰り返しで表現し、各組織が本部・部・課のどれなのかは「組織種」に持たせるという抽象化を行っており、(b)の方が組織変更に強いモデルになっています。概念データモデルの検討においては、(a) を(b)にするよりも先に、(a)の状態で業務部門とモデルを確認してから抽象化を実施するとよいでしょう。

4.2.7 エンティティ定義書／データ項目定義書

(1) 目的

管理対象エンティティの管理内容、管理ルールを項目レベルで把握する。

(2) 説明

① エンティティ定義書では、管理対象エンティティのレイアウトと従属するデータ項目の以下のような項目内容を定義する。

- キー情報
- データ型
- 桁数・文字数

表 4.9 にエンティティ定義書の例を示す。

表 4.9　エンティティ定義書の例

エンティティ名	正社員_役職辞令
説明	正社員に発令した配属、役職任命の履歴を保持する。

No.	属性名	主キー	データ型	桁数 文字数	説明
1	従業員NO	○	文字列	6	従業員を一意に識別する番号
2	部門コード	○	文字列	4	部門を一意に識別するコード
3	役職コード	○	文字列	2	役職を一意に識別するコード
4	役職発令日		日付	-	役職任命を発令した年月日

② データ項目定義書では、エンティティ定義書に記載された各データ項目について、次のような属性や管理ルールを定義する。表 4.10 にデータ項目定義書の例を示す。

- データ属性（項目タイプ、桁数、小数点桁数、最大値、最小値など）
- 初期値、Null 値
- 前提条件、制約条件　など

表 4.10　データ項目定義書の例

データ項目の属性	説明
データ項目の名称	部門配属日
データ項目が属するエンティティ	正社員_役職辞令
データ型、桁数	日付型(他に、文字型、整数型、桁数、小数以下桁数、最大値、最小値など)
Nullを許容するか	許容しない
初期値	なし
前提条件(制約条件など)	部門に配属されると、従業員NOと部門コードと同時に発生する。
参照･更新権限	登録権限は人事にだけ付与
･･･	

(3) 留意事項

① データ項目を使用するのは業務部門であるため、主要なデータ項目については該当の業務部門が中心になってまとめる。複数部門で共同利用する場合はシステム部門が主幹業務部門を決定するとよい。

② データ項目定義書は、次に示すことに留意する。

(a) 初期値の有無、データの範囲、扱いを明確にしておく。

(例) コード値の設定

性別：1（男）、2（女）だけでなく、3（不明）、4（その他特記あり）など考慮

商品種別：001～700、700～800　新商品のための予約、900 番台は使用しない、等

(b) 項目名称の付け方をルール化して、同名異義（ホモニム）・異名同義（シノニム）、等、項目名称を整理して、極力統一し、数を減らす。

(例)「日」「YMD」「年月日」　⇒「年月日」に統一

「発令日」「着任日」「異動日」⇒「発令年月日」に統一

「管理項目 1、管理項目 2」　⇒　番号つき項目禁止

③ 区分、記号、コード、表示などの違いを明確に定義し、関係者間で合意する。

④ 一つの項目に対し複数の意味を定義しない。

(a) 一つの部門コードの上２桁は「部」、下２桁は「課」と分けている定義している例がある。

このような場合、異なる意味である部と課に強い依存関係が発生してしまうため拡張性が低くなる。部コードと課コードを別項目にすることで、拡張性が保たれる。

(b) コードと数値、数量を区別し、混同しない。

数値と数量の混同例：「期間月数」が未定の時に期間月数＝9999（未定）は使用しない。この場合は、「期間区分=1:有期　2;無期　3:未定」といった区分を別途設け、3 を期間区分の初期値とする、等

⑤ データ、コードの管理部門（データ管理者）を決めておく。

- データ項目の変更権限や複数の関連部門にまたがるコードの主幹部署については、その管理部門とシステム部門が調整をすると、関連する業務、およびシステムへ影響を把握できる。
- 桁数等のデータ属性、コード項目の値等を定める時は、連動先や使用者・用途などを視野に入れ、齟齬が出ないようにする。

4.2.8 CRUD 図

(1) 目的

主要エンティティ、データ項目について、どの機能で作成、更新、消去されるかを確認し、その主要データ項目のライフサイクルを把握する。

(2) 説明

主要なデータがどこで作成され（C、登録、Create）、どこで使用され（R、照会、Read）、どこで更新され（U、更新、Update）、どこで削除される（D、削除、Delete）かを表す CRUD 図を作成する。CRUD 図の作成を通して、主要データのライフサイクルと機能との間の関係性を確認し、データ、機能の抜け漏れ、矛盾を検証する。

CRUD 図記載の粒度にはエンティティレベルとデータ項目レベルがある。エンティティレベルは、主に CRUD 図に記載すべきエンティティや機能の抜け漏れ、エンティティと機能との関連性の矛盾を検証するために作成する。一方でデータ項目レベルではデータ管理上の矛盾を検証するために作成する。表 4.11 に CRUD 図作成を通して検証できるエンティティ、データ項目、機能の抜け漏れ、矛盾の例を示す。

矛盾の例：先行して実行される機能で更新しているデータ項目が後続で実行される機能で登録されている。

表 4.11　CRUD 図作成により検証できるエンティティ、データ項目、機能の抜け漏れ、矛盾の例

記載粒度	抜け漏れ／矛盾	例
エンティティ	機能の抜け漏れ	あるエンティティを更新、削除する機能があるにも関わらず、登録する機能がない。
	エンティティの抜け漏れ	更新機能であるにも関わらず、更新・削除対象のエンティティがない。
	矛盾	先行して実行される機能で参照しているエンティティが後続して実行される機能で登録されている。
データ項目	矛盾	先行して実行される機能で更新しているデータ項目が後続して実行される機能で登録されている。

図 4.11 にエンティティレベルの CRUD 図の例を、図 4.12 にデータ項目レベルの CRUD 図の例を示す。

	受注	生産手配	進捗管理	倉庫管理	出荷	売上計上	財務処理	削除処理
受注	C	U	U	U	U	U	U	D
出荷			U	U	C	C	U	D
・ ・								

凡例：C（登録、Create）、R（照会、Read）、U（更新、Update）、D（削除、Delete）

図 4.11　CRUD 図（エンティティレベル）の例

	受注	生産手配	進捗管理	倉庫管理	出荷	売上計上	財務処理	削除処理
受注番号	C							D
受注数量	C							D
単価	C					U	U	D
受注金額	C				U	U	U	D
主要仕様（色など）	C	U			U	U		D
生産数量		C	U	U	U	U		D
・ ・								

凡例：C（登録、Create）、R（照会、Read）、U（更新、Update）、D（削除、Delete）

図 4.12　CRUD 図（データ項目レベル）の例

(3) 留意事項

① 要件定義段階で、主要データについて CRUD 図を作成するのは、情報を登録・参照・更新・削除する機能の配置を確認し、機能面、データ面での抜け漏れを検証するためである。すべてのデータについて、CRUD 図を作成する必要はなく、企業活動の主要データ、複数の機能から更新されるような項目だけでよい。

例として、受注システムにおける主要なデータとは、「受注」などであり、複数の機能から更新されるような項目とは「数量」「金額」「主要仕様」などである。受注後、生産を開始した時点まで、システムで変更を受け付けることを認めるかどうかなど、エンティティレベルやデータレベルの情報のライフサイクルについては概念データモデルなどで確認する。

② 項目を作成、更新するタイミング（更新してはいけないタイミング）を整理の上、制約がないかを確認する。

D（削除）については、業務的な削除（論理削除）とシステム的な削除（物理削除）がある。要件定義ではデータの業務的なライフサイクルを確認することが目的であるため、業務的な削除を考慮して検討する。

③ データ項目を正しく定義する。

データ項目レベルで CRUD 図を書くことにより、エンティティレベルでは確認できないあいまい性を排除できる。

コラム　～要件定義の CRUD 図の役割～

【悩み】

「CRUD 図は作成するのに時間を要するので作成しない」というプロジェクトは少なくないことでしょう。模造紙のような大きな紙に定規をあてながら苦労して CRUD 図を作成したことはありませんか。100 プロセス×1,000 データのマトリクスなら 100,000 の要素が存在し、CRUD が存在しないのが正しいのかを含めてすべての要素を確認しなければなりませんが、これは容易なことではありません。

【解決案】

要件定義で CRUD 図を作成する目的を明確にしてから作成します。

設計工程では機能とデータの整合性の検証を目的として大きなマトリクスで検証をすることがありますが、要件定義における CRUD 図作成の目的は、新しいビジネスプロセスが業務として成立しているかどうかの検証です。CRUD 図はそれを検証するために作成すれば目的を達成します。

要件定義では、業務の実世界を業務フローや概念データモデルを使ってモデル化し、可視化された業務の上で、課題分析や新しい業務の検討、業務の理解・共通認識を行っています。業務フローなどは機能モデル、概念データモデルはデータモデルになり、機能とデータの関係は下図に表すように、クロスするような関係になっています。

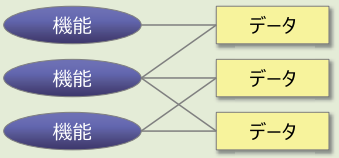


図　機能とデータの関係

CRUD 図は、この両者の関係を表形式でモデル化したものであり、相互関係のモデルなどと言われています。機能とデータの相互関係のモデル化は、状態遷移図などでも実施できますので、状態遷移図を用いる方法もあります。

CRUD 図の利点は、縦軸、横軸の各要素を軸に網羅的に内容を検証できるところです。、縦軸、横軸を検証の目的に合わせて選択することにより、マトリクスを必要最小限に絞り込んで機能とデータの相互関係をモデル化することが、CRUD 図作成のコツになります。要件定義での軸の候補としては、以下の 2 つが考えられます。

- データ側：エンティティ、主要なデータ項目
- 機能側：ビジネスプロセス、システム化業務

横軸はプロセスであり、大きなレベルではビジネスプロセスになり、詳細なレベルではシステム機能になる。データのライフサイクルが長い、例えば企業全体に及ぶような場合は、大きなプロセスでライフサイクル全体を可視化できるようにビジネスプロセスを横軸に選択した方が良い。

縦軸はエンティティにするか、データ項目にするかである。エンティティはデータ項目に比べると圧倒的に数は少ないのでエンティティを縦軸に選択した方が良い。しかし、作成者がエンティティ自体を正確に理解できていない場合が少なくない。この様な場合は、主要なデータ項目を厳選し、縦軸に設定すると良い。具体的な項目で検証できるので理解し易い。また、データ項目で CRUD 図を作成することでエンティティ自体を明確化することができる。

縦軸（プロセス）、横軸（データ）のレベルを考慮したとして、プロセス、データ双方漏れなく列挙し CRUD 図を作成しなければならないのか。

冒頭に述べたように、業務フローやシステム化業務フローの漏れや矛盾、概念データモデルのエンティティやデータ項目の漏れや矛盾を発見することが目的であり、全てを漏れなく検証しなくても良い。例えば、オーダを中心とした、販売、製造、倉庫、出荷、請求業務の全体最適システムへの見直しが課題であれば、それに関係した部分だけを検証できれば良い。課題が複数あれば、無理に一枚の CRUD 図にせずに、課題ごとに分けてもよい。

全業務を検証できるわけではないが、この様に的を絞って検証するのもひとつの方法である。

また、CRUD 図は、例えば以下のようなテーマを設定して、それが検証できるビジネスプロセスやデータに絞って作成するのも良いでしょう。

- 「オーダのビジビリティの追求」
- 「販売計画から生産計画の立案から予実管理」など

すべての業務を検証できるわけではありませんが、このように的を絞って検証するのもひとつの方法です。

本文にもあるように、業務フローやシステム化業務フローの機能の抜け漏れや、概念データモデルのエンティティの抜け漏れ、ビジネスプロセスとデータの関係の矛盾を発見できることが CRUD 図作成の効果です。どのように作成すれば効果が得られるかを検討した上で、プロジェクトに必要十分な範囲で CRUD 図を作成し、有効に活用して下さい。

4.2.9 総合テスト計画書

(1) 目的

要件定義で決められたことが、「システムの機能として不足や誤りなく実装されているか」、「定義した要件で実際の業務が成立するか」、「例外処理も含めて、業務が正しく実行できるか」を確認する。

(2) 説明

① 総合テスト計画書は検討開始が遅くなりがちで、要件次第では業務設計、運用方法などに非常に大きな影響を与える。総合テスト計画書の記載事項のうち、要件定義の成果から決定できる部分は作成しておくことにより、後続工程に入ってからの手戻りを少なくできる。

（注）要件定義、基本設計等、それぞれの設計段階でテスト仕様を固め、漏れの早期発見と後続工程からの手戻り防止を図る方法はWモデルと呼ばれる。

② 総合テスト計画書には、表 4.12 に示すように、総合テストの目的、総合テストの概要に加え、一連の業務の流れに沿った総合テストシナリオ、締め処理（日、月、年次など）、他システムとの連携、障害回復テストなどを記載する。

表 4.12　総合テスト計画書（例）

総合テスト計画書(例)
1. 総合テストの目的
2. 総合テスト概要 (1) 対象、範囲 (2) 前提条件 (3) 環境(ハード、ソフト、ネットワーク) (4) 期間、スケジュール (5) 体制
3. 総合テスト仕様書 (1) 総合テストシナリオによる一連のつながりの確認(テスト) ・通常処理 ・例外処理 ・特殊処理 (2) サイクルテスト(日回しのテスト) (3) 締め処理(日、月、年次) (4) 他システムとの連携テスト (5) 性能負荷テスト (6) 障害回復テスト (7) 移行リハーサルテスト (8) ユーザテスト
4. 総合テスト終了条件

③ 総合テストシナリオの作成は以下を前提として開始する。

- 要件定義書の根幹部分(システム化業務フロー、業務機能構成表など)がほぼ確

定していること。

- 全システム一括本稼働一斉切り替えなのか、部分的に移行していくのか、などの、総合テスト後の移行方式が決まっていること。

(3) 留意事項

① 総合テスト計画書には正常系だけでなく、例外系、特殊系を含め、一連の業務が正しく実行できるかを確認するための総合テストシナリオを記載する。

(a) ビジネスプロセス関連図や業務流れ図（業務フロー）を活用して総合テストシナリオを可視化する。

(b) システムが複数のサブシステムから構成される場合、全サブシステムを通して業務が成り立つかというシナリオを作る。

(c) テストケース作成については、通常業務で一般的に発生する業務、利用頻度の高い業務を優先し、特殊ケース、難易度の高い機能の確認を後回しにする。

② 総合テストシナリオの作成には、全体の業務を熟知した担当者を当たらせる。それに加え、これらのシナリオが作成できるスキルのある（全体業務を把握している）技術者をアサインする。

③ 本格的な総合テスト実施前の事前検証として、全体の新システム構成を集約した環境で、少数の基本的な業務データの流れテスト（参照する基準値テーブルおよびシステム動作環境も含めて設定する）を実施することにより、技術的環境の動作検証も含めて一連の基本データの流れを検証できるので、致命的な初期動作不良を除去できる。これにより、以降の詳細なシナリオに沿った総合テストを円滑に実施することが出来る。総合テストでのテストケースを要件定義で作成すると、単体テスト、結合テストでも参考にできテストケースの抜けが減少する。

④ 要件定義の段階で検討されていない事項は、未決事項として管理する。

⑤ 要件定義と並行して総合テスト計画を作成することは難しいが、総合テスト計画書を作成することにより発見できる要件の漏れは多い。要件定義作業の最後にこの作成を義務付けるとよい。

⑥ 総合テストの完了判断基準を決定しておく。

(a) 期待する品質目標値（例．不具合件数／規模）を記載しておく。

(b) 期待するレスポンスタイム（例．秒／件）や、バッチ処理時間等を記載しておく。

4.2.10 システム移行計画書

(1) 目的

現行システムから新システムへ、業務、データ、システムを問題なく移行する手順を確認する。

(2) 説明

① 近年システム規模が大きくなり、現行システムからの移行をトラブルなく実施できるかが、システム構築の成否の大きな比重を占める。要件次第では、業務設計、システム基盤設計、稼働日などに、非常に大きな影響を与える場合がある。しかし、システム移行の検討は、システム実装後へと後回しにされることが多く、移行方法を検討した結果、機能の見直しが発生する場合がある。要件定義の段階で、システム移行計画のうち作成できる部分は作成しておくことと、それらを基本設計以降の各工程において順次見直すことが重要である。

② 次の項目は、要件定義で検討できる部分は検討しておく。特に（d）は漏れやすいので注意すること。

(a) 移行の対象

- データ
- アプリケーションシステム、OS の移行、ハードウェア、ネットワーク
- 業務

(b) 過去のトランザクションの移行範囲

(c) 業務の切替方法、移行作業負荷および時間、平行運転期間

(d) 障害対処や切り戻し判断（移行中止判断とその後の作業）

(e) エンドユーザ教育関係の方針

(f) 移行体制

(3) 留意事項

① システム全体の利用者を対象にして一斉に全機能を切り替えるか、できるだけ対象を狭めてスモールスタートするか、を確認する。スモールスタートの方が、危険、障害が少ないため、そのための仕組みを要件定義で組み入れることを推奨する。

② 並行稼働の有無を確認する。並行の場合は、現時点で想定する並行稼働の対象システムと想定期間を明記する。また、旧システムの運用凍結時期と新システムへの移行時期を明確にしておく。

③ データ移行（コンバージョン）を早い段階で検討しておくと、4.2.8（CRUD 図）では想定していなかったデータの存在を確認でき、移行時だけでなく、要件定義、設計段階で、早期に対応を検討できる。さらに、移行対象データはできる限り絞り込むと良い。5 年に 1 回使われる程度のデータのデータ移行は行わないことなどを確認する。

④ データベースの中から誤りや重複を洗い出し、異質なデータを取り除いて整理する、データクレンジングが重要であり、次の点に留意することが必要となる。

- データクレンジングの責任者を明確にしておく
- この作業を、ベンダ任せにしないこと
- 対象データの取り扱いを明確にしておく
- 修正履歴を保管する

⑤ 移行しないデータ、手修正するデータ、確認用に紙または別ファイルに保管するデータを確認する。少量のデータ移行や組織独自で管理している情報で、手作業で移行できるものは、むやみに移行プログラムの数を増やさない考慮も必要である。

⑥ 移行リハーサルを計画に入れておく。

⑦ 障害対応や切り戻しを含めて検討しておく。
移行作業に入ってからの予期せぬトラブルに対応するため、システム移行作業途中の検証ポイントを明確にし、システム戻しの手順を検討しておく。

⑧ 体制面等での考慮事項

- 移行チームを独立設置し、運用部門も巻き込み、開発チームより先に立ちあげる。
- 業務／システム／データに分けて検討する。データクレンジングを検討しておく。

⑨　次の項目は、後回しになることが多いため、この段階で方向性を検討し、要件定義に反映する。

(a) 訓練、研修について、次の検討を行っておく。

- 訓練・研修の要否、訓練・研修時の要件（個人情報は表示、印字しない等)、時期および対象機能
- システム切替前に研修開始するのであれば、環境準備、開発スケジュール等
- システム運用に携わる現場要員の教育訓練・習熟計画を忘れない

(b) 機械処理系以外の判断基準(教育・習熟度、監査部門の承認、経理処理、企業内部への広報、セレモニー、資源準備、組織発令など)を検討する。

⑩　要件定義の段階で検討できていない事項は、未決事項として管理する。

4.2.11 運用・操作要件書

(1) 目的

このシステムを業務が実際に運用、操作でき、それによって業務が遂行できるための要件を定義する。

(2) 説明

業務の遂行要件、システムの運用要件・障害時の対応、操作要件のそれぞれについてまとめる。

① システムの運用要件書

次の項目についてまとめる。

(a) システムのサービス時間、メンテナンスなどの休止に関する要件

(b) 年次、月次、週次、日次の運用スケジュール、特殊日のスケジュールなどを確認

(c) 地域による違いや、グローバルの運用の場合のサービス時間帯や、運用要件の違い

(d) ヘルプデスク・サービスの運用要件

画面操作でどうしてよいか分からなくなった場合の電話対応の意義は大きい。親切に対応できるように、ヘルプデスク担当者の十分な教育等を実施することが重要。

(e) インシデント管理、キャパシティ管理等

(f) 関係する他システムの運用要件との整合性、制約条件、スケジュール等

(g) 障害発生時の業務継続の条件を明確にし、対応方針を検討、記載

(サーバ復旧、データの整合性、システム間の整合性)

② 業務の運用要件書

業務部門で、当該システムを運用するための要件を記載する。

システムの運用要件以外で、決めておくべきことがあれば決める（例：システム障害時の業務代替手段等)。

③ 操作要件書

実際の操作手順書の作成は後続工程でよいが、要件定義段階では、次の点に留意した操作要件書を作成し、後続工程で修正していく。

(a) 目的に合った操作性か

(b) 重要な業務は、要件定義時に、画面イメージやプロトタイプなどで確認する

(3) 留意事項

① 運用・操作要件書の作成にあたっては、運用部門を検討に巻き込む。

② 要件定義の段階で、運用・操作要件を検討する。
運用・操作要件書の検討は、システム実装後の後回しになることが多いが、要件次第では、システムの基盤設計、運用保守体制やアプリケーション設計に大きな影響を与える場合も多い。要件定義の段階で運用・操作要件を作成することで、運用、操作に関する要件の漏れや手戻りを少なくすることができる。

③ 操作要件書は、次の点に留意する。

(a) 業務目的に合った操作性か

- 使い方をすぐに習得できるか
- 誤操作しにくいか
- 高齢者や障がい者など、誰もが利用可能なユニバーサルデザインか
- 実作業の操作（例えば、工場での実作業）と端末操作が、共存できるような操作になっているか。

(b) 事前に、使い勝手を確認しておくこと
詳細、4.2.12（非機能要件書）の使用性の項目で説明する。

(c) 習得のしやすさや誤操作の防止は、画面上の操作ガイドやエラーメッセージの記述レベル、マニュアルの記述レベルによって大きく変わる。そのため、画面の操作ガイドやエラーメッセージの出し方に注意を払うことが重要となる。
(例) エラーメッセージを「不正な日付が入力されました」（原因）ではなく「×月×日以降の日付を入力下さい」（指示）とするだけでも大きく変わるので、どこまで記述するかについて取り決めておく。

(d) セキュリティ機能、安全性の機能も盛り込む。
詳細は、4.2.12 項（非機能要件書）のセキュリティの項目で説明する。

4.2.12 非機能要件書

(1) 目的

業務機能を実現するために必要となるシステムの非機能要件を定義する。

(2) 説明

当該プロジェクトの要件定義工程で決めておくべき非機能要件を決める。

(3) 留意事項

① 次の観点で非機能要件を定義する。

(a) データ量

(i) 業務量は、利用ユーザ数（稼働当初、最大）、商品数（稼働当初、最大）として、稼働当初、予想される最大量を確認する。
（注）最大量は、システムのライフサイクルを考えた最大量

(ii) トランザクション量（入力量）は平常時、ピーク時、緊急時、障害時に予想される量について確認する。

(iii) データ保存量は、稼働当初、予想される最大量を確認する。

(b) 求められるレスポンスタイム

(i) レスポンスタイムは、通常時、ピーク時、将来について確認する。

(ii) レスポンスタイムについては「1 分間に最大 500 件の入力があった場合に、2 秒以下で返答できる確率を 95%にして欲しい」などと定義する。

(iii) 他の業務とサーバ、ネットワークを共同使用する場合が多いので、（ii）の条件の評価は実稼働環境でないとわからないことをユーザもベンダも理解しておくことが必要である。

(iv) 端末内処理、端末からサーバへの通信、サーバの処理、サーバから端末への通信、端末内処理などの計画時間を提示し、それが実際稼働した際に計画時間と異なる場合は、どの資源を強化するか説明できるように整理することが望ましい。

(v) ネットワークを利用するシステムでは、ネットワーク品質(処理能力、遅延率など)を考慮する。

(vi) 業務部門の要求の意図・目的を明らかにする。例えば、移り気なネットユーザを確実に情報提供画面に導くため、画面遷移は最長でも 2 秒以内とする、といったことを明らかにしておく。

(vii) 前提としているハードウェア構成を明確にすること。そのような部分でハードウェア構成が変更になった場合は、前提が崩れるため、性能要件の見直し、あ

るいは、性能要件をそのままとするのであれば、アプリケーションとしての設計の見直しが発生する。

(c) 使用性

(i) 使い勝手は、数値化が難しい。そのため、画面イメージやプロトタイプで確認する。要件定義時に、操作性を確認できるツール類を使用し、利用者が納得してから設計に取り掛かれば、手戻りを減らせる。

(ii) 使い方の習得のしやすさや誤操作の防止は、画面上の操作ガイドやエラーメッセージ、マニュアルの記述レベルによって大きく変わる。例えば、エラーメッセージを「不正な日付が入力されました」ではなく「×月×日以降の日付を入力してください」とする。どこまで詳細に記述するかについて、非機能要件として取り決めておく。

(d) 稼働率、稼働品質

(i) システムの稼働率、稼働品質の評価には、表 4.13 に示すようにいくつかの評価指標がある。重要インフラシステムか、基幹業務システムかの違いや、そのシステムの特質により、稼働率、稼働品質の評価基準は異なるため、表 4.13 のような表を準備し要件定義の段階で確認する必要がある。

表 4.13　稼働率、稼働品質

区分	評価項目	評価式	評価	参考目標
稼働率	稼働率	実績稼働時間/計画稼働時間	1 に近いほどよい	99.999%(5 分停止/年)以上のような厳しい目標の場合は特別な対策が必要(99.5%程度は通常確保)
	延べ稼働率	(延べ時間-計画停止時間-障害停止時間)/延べ時間	1 に近いほどよい	顧客の使用可能時間の評価(99.95%以下になると不満が出るなど)
稼働品質	業務停止回数	業務停止回数/年	0 に近いほどよい	基幹業務システムは 0.06 件/1 年間で運用費程度は確保される
	規定時間外停止回数	規定時間以上に停止した回数/年	0 に近いほどよい	重要インフラシステムで特に重視(15 分以上の停止は 0 回/年が目標)
	オンライン平均応答時間	規定内応答回数/全応答回数	1 に近いほどよい	顧客からの直接受注入力など主要な入力データに限って管理するのが良い
	バッチ処理異常終了率	異常処理回数/年	0 に近いほどよい	障害報告の迅速性などの目標も必要

(ii) 現行システムなどと比較して、要件を具体化する。例えば、夜間処理は今の半分の 2 時間で終わらせ、オンラインサービス遅れの発生を年間 10 日から 2 日以内にする。

(iii) 業務システムのエラー時の対応の大方針を決めておく。例えば、業務システムのエラー時に、処理そのものを止めるのか、エラーログを出して続行するか、などを決めておくと、基本設計以降からの手戻りが減る。

(iv) 既存のサーバに相乗りする場合は、対象サーバの制約を確認する。

(v) バックアップを含めた総コストを見積り、投資判断に加えることも重要である。

(vi) クラウド活用のシステム構成の場合、オンプレミスのシステム構成とシステムの挙動が変わって、思わぬ障害に遭遇することがある。事前にクラウド環境のシステム動作に慣れておくことが望ましい。

(e) セキュリティ

(i) 利用者の認証、権限管理、アクセス権管理、ログ管理、暗号化機能などの要件を明確にする。

(ii) データや業務へのアクセス権限を適切に設定し、許可されていない者がアクセスできないようにする。万が一不正アクセスが発生しても、情報漏洩や妨害行為などの事象が引き起こされた事実が証明できる仕組みにしておきたい。悪意ある相手によるデータの窃盗や改ざんに対し、これを防止してデータが健全な状態を保つ方策が必要となる。これには専門的な知識が要求されるため、場合によっては外部の専門機関の活用を検討する。

(iii) 業務の監査ログ、証跡の保持、各種アクセスログの採取と保持、ユーザ ID とパスワード以外の利用者本人を確認する認証方法についても、検討する必要がある。

(f) 保守性

システム改修が自分たちでできない、あるいは少しの改修に高い費用がかかるといった問題はよくある。例えば、「受注登録画面に新たな入力項目を追加したい」など、稼働後の機能追加要求は必ずある。これらの要求に、短時間、低コストで対応するには、プログラムを簡単に修正できるようにする必要があり、そのためにはプログラムの構築方法まで踏み込む必要がある。要件定義では、保守性を向上させる手段として、適当な開発ツールやフレームワークを検討するのも有効である。最近の開発ツールには保守ドキュメントを自動的に出力できる製品も多い。

(g) 移植性

ハードウェアや OS、データベースなどのミドルウェアは保守切れに伴うバージョンアップが避けられない。その場合もアプリケーションは変更せずにすませたい。

国内で作成したシステムを海外展開する場合も、言語や OS の違いを吸収できるような考慮が必要となる。特に、機能が特定のハードウェアや OS に大きく依存すると移植性が下がる。できるだけ、世の中に普及している標準的な技術を組み合わせるのが好ましい。

② 非機能要件を決めるにあたっての考慮事項

(a) コスト、セキュリティ、使い勝手などのトレードオフ関係を明らかにし、多くのステークホルダに判断してもらう。ステークホルダには、経理部門、管理部門、法務部門など上記の判断に関係する部門も加える。

(b) ユーザ企業は、要件定義段階でユーザが提示すべき必要最低限の要求を定義すること。できればユーザが簡単に答えられる事項に簡略化してベンダから提示してもらうことが望ましい。

(c) 顧客ニーズの変化、技術進歩により、非機能要求や実現費用は変わる。ユーザ企業各社では非機能要件の標準を決めているが、標準は定期的に見直すこと。標準を見直す時は、現有共通基盤などの実態だけでなく、自社業務の本来の要求をもとに実施する。

③ 非機能要件の見積もり方法

非機能要件の管理項目、目標値の確認は実行段階に入ってからの実績値に基づくものが多く、検証の難しさはあるが、このプロジェクトで守るべき目標値を要件定義時点で定めていないとシステム予算に大きな影響を及ぼす。期待値でよいので必要項目を提示することが重要である。ただ、他システムとの共存で決まる要素も多いことに注意が必要となる。非機能要件の費用対効果の検討には、表 4.14 のようなものを準備しておくと、要求の妥当性と価格の比較が可能となる。

表 4.14　非機能要件の見積もり方法（案）

大区分	中区分	小区分	梅	竹	松
信頼性	稼働率	サーバ、電源、 運転費用 費用の比率 （以下、比率）	99.5%未満 1	99.5%以上99.9%未満 2	99.999%以上 3
	障害許容性	ネットワーク（比率）	1	1	1.2
	回復性	障害時業務継続性 （比率）	単一構成 1	コールドスタンバイ 2	ホットスタンバイ 5
	回復性	BCP（災害対策） （比率）	1Wで回復 1	3日で回復 2	1日で回復 5
効率性	レスポンス タイム	プログラム＋HW （比率）	3秒以内（画面） 1	2秒以内（画面） 1.2	0.1秒以内（主要項目） 1秒以内（画面） 2
使用性		画面（比率） 帳票（比率）	1 1	1.2　操作性 1.1	2.0　操作性 1.2
		運用サポート時間 （比率）	9時～17時 1	9時～21時 1.5	24H／365日 3
保守性	プログラム	外部テーブル （比率） リポジトリー（比率）	1 1	1 1.2	1.2 1.5
移植性	プログラム	（例）ホスト→クラサバ クラサバ→クラウド	1 1	1.1 1.1	1.1 1.1
セキュリティ	攻撃プロテクト	導入ウィルスソフト （比率）	定期更新（～2個） 1	定期更新（5個） 1.5	常時更新（6個～） 2.0

上記例は仮設定、数値は概算費用の相対比率

4.3 成果物に共通の留意点

本節では、要件定義で作成する成果物に共通の勘どころを中心に解説する。

4.3.1 要件定義成果物のレビュー

(1) 目的

要件定義成果物の抜け・漏れおよび成果物間の整合性不備などに関して、熟視を主体とする公式なレビューによる検査によって品質を担保する。

(2) 説明

成果物のレビューと一口に言っても、レビュー手法には非公式、公式を含め、多くの種類が存在する。レビュー手法の特徴を知ってもらうために、表 4.15 は各レビュー手法に期待される主な効果を示す。

表 4.15　レビュー手法に期待される主な効果

効果 / 手法	欠陥製品の発見	仕様との適合性チェック	標準との適合性チェック	製品の完全性と正しさの検証	理解性と保守性の評価	クリティカルまたは高リスクのコンポーネントの品質の実証	プロセス改善のためのデータ収集	文書品質の測定	製品に関する他のチームメンバーの教育	対処法についてのコンセンサスの決定	変更またはバグ修正の正しさの確認	別の対処法の検討	プログラム実行のシミュレーション	レビューコストの最小化
アドホック・レビュー	●													●
ペア・プログラミング	●				●				●	●		●		
ピア・デスク・チェック	●	●	●									●		●
ピア・レビュー	●	●	●	●					●	●	●			
パス・アラウンド	●	●	●		●				●					
チーム・レビュー	●	●	●		●		●		●	●	●			
ウォークスルー	●			●					●	●	●	●	●	
インスペクション	●	●	●	●	●	●	●	●						

（出典）SECBOOKS「高信頼化ソフトウェアのための開発手法ガイドブック」[13]

プロジェクトの特性などにもよるが、特に高信頼ソフトウェアの要求および要件定義成果物のレビュー方法を検討する場合、インスペクションは重要な手法になる。よって、本稿ではインスペクションに焦点をあて説明する。

インスペクションは ANSI/IEEE 標準において体系が定義されているもっとも厳格で公式なレビュー手法である。インスペクションは他の手法に比べ、実施コストは高くなるが、欠陥の発見に最も効果が高い手法と言われており、その目的を次のように定めている。

- 成果物（ソフトウェア要素）が基本仕様、指定された品質属性、顧客要求を満たしているかの検証
- 成果物（ソフトウェア要素）が関連する基準、規制、規則、計画、手順に適合しているかの検証
- 発見された欠陥とインスペクションに使用した時間に関する評価指標の供給
- 表現方法の良し悪しや文体に対する検査は、行わない

インスペクションの特徴の一つに役割をはっきり分けることがあげられる。役割は作成者、進行・まとめ役、記録役、説明役、レビューを行う役の 5 つの役がある。特に注意しなければいけないこととして、作成者は説明役以外を兼任できないという点がある。インスペクションでは第三者の目（客観性、論理性、具体性、実現性など）の評価からみた事実を基本とした指摘やコメントを受け、これに対して回答することで、作成者の思い込みを廃し、可能な限り多くの欠陥を取り除くことを目指している。指摘やコメントは記録役により迅速に収集される。インスペクションの対象にもよるが、インスペクションでは記録役、進行・まとめ役の負担もかなり高いため、これら 2 つの役も兼任を避けるべきである。

表 4.16　インスペクションでの役割

役　割	作業内容など
進行・まとめ役 (Moderator)	インスペクションの主催者である。
	インスペクションを熟知している必要がある。インスペクション・メンバーの選抜やスケジュール調整、成果物の受領と配布、さらには作業進捗の管理を行う。
	計画段階で、ある条件が揃った場合にインスペクションの中断か継続可能かのチェッポイントを設けておく。
	最終的には当該インスペクションの結果報告責任者になる。
レビューを行う役 (Inspector)	欠陥検出の担当者で、当該チームの作業内容について既知であるか否かを問わず、高い欠陥発見能力を持つ人が望まれる。
	インスペクションでは、対象成果物ごとの関連性を考慮して、レビューを行う役などを決める。
	インスペクションはオーバーヘッドなど非効率性を鑑み、最適人数を決定する。 （経験則として３名～４名まで）
	インスペクションに関する正式な教育を受けていることが必要である。
説明役 (Reader)	進行に合わせて対象成果物の該当部分を他に説明する。一般的には作成者が担当する。
記録役 (Recorder)	指摘された欠陥と問題を分類、記録する。
作成者 (Author)	インスペクション対象に関わる成果物の作成者または保守者が該当する。

（出典）SECBOOKS「高信頼化ソフトウェアのための開発手法ガイドブック」[13]

インスペクションでは手順も定められており、一般的に図 4.13 に示すように 7 つの段階を踏む。

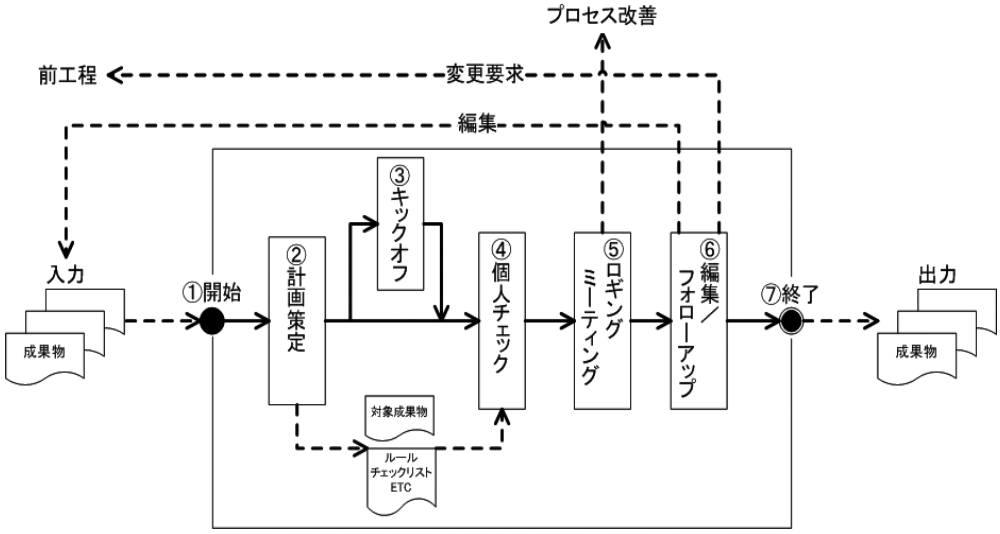


図 4.13　インスペクションのプロセス

（出典）SECBOOKS「高信頼化ソフトウェアのための開発手法ガイドブック」[13]

(3) 留意事項

要件定義成果物のインスペクションを実施するにあたって、以下に留意事項を示す。

① 要件定義成果物を対象にしたインスペクションの導入は、ユーザ要求への適合性検証が主眼に置かれるため、ビジネス目的・施策を各役割担当者間で共有しておく必要がある。さらに、品質を重んじるチーム文化の構築、各種スキル（知識含む）の共有・移植促進およびステークホルダ間の合意形成などの工夫努力により、チーム全体の結束を高める効果が副産物として生まれることを認識すること。

② 特に進行中の成果物をレビューする場合は、インスペクション開始時に成果物の作成者に未完成部分と完成部分、さらには完成部分でも不安要素が残る部分などといった成果物全体の出来栄え状況をヒアリングしておくことで、事前にレビューポイントを押さえておくことができる。

③ 品質の観点から要件定義成果物に対するチェックリストを具備することが必要である。インスペクションでのロギングミーティングの開始前に誤字脱字や形式の不整合など、一見して分かる欠陥は、チェックリストの配布などにより単独チェックにて検出しておくこと。さらに、ISO/IEC 25010「品質モデル」など、何らかの品質基準を基にチェックリストなどを作成し、観点のバラツキをなくす工夫も必要である。

④　テスト工程ではデグレード確認とかリグレッションテストといったアクティビティが存在する。レビューでも同じように、ある欠陥の修正時に誤って別の欠陥を作り込む場合がよくある。同様に、レビューを行う役やまとめ役は誤修正がないかを確認しておくことが必要である。

⑤　鉄則として詳細から検証するのでなく全体を俯瞰した検証から実施すること。つまり、全体を俯瞰できる成果物に欠陥があれば詳細を検証しても効率が悪い。また、レビュー時間が足りなくなった場合に影響度が大きい欠陥を先に検出できているため、詳細はサンプリングでも足りる場合もある。さらに、インスペクションで重要視すべきは、本来要求（目的・施策）と適合しない仕様および後続工程での課題（検出しにくい不具合や手戻りコスト増大）を発生させるような仕様間の不整合／不統一の検出がある。

そこで、要件定義成果物としてレビューおよび総合テスト仕様書の作成にも参考となる中間生産物の作成を推奨したい。以下に、いくつかの事例を紹介する。

（事例 1）

4.2.8 項　CRUD 図は機能とエンティティに抜けや漏れがないかを分析、検証するのに有効である。但し、業務部門を巻き込む場合、業務部門に分かり易い表現にするなどの工夫をすることも必要である。

（事例 2）

新旧画面(論理)対比表による旧画面に対する新画面の対応方法の確認（図 4.14）

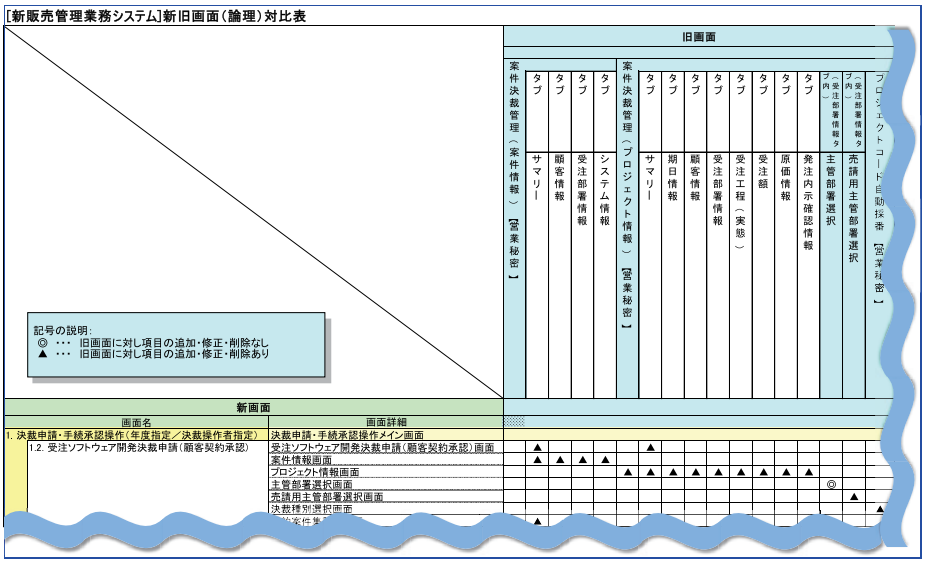


図 4.14　新旧画面対比表の例

（出典）ジャステック社資料

（事例 3）

業務とワークフローステータス対比表による相互の関連を確認（表 4.17）

表 4.17　業務とワークフローステータスの対比表の例

［新販売管理業務システム］業務とワークフローステータスの対比表

業務名称	ワークフローステータス										
	作成	作成確認	依頼	受付	起案	申請	仮決裁	決裁	結果入力	内容確認	決裁確認
①　(1)受注ｿﾌﾄｳｪｱ開発-①取引開始承認	●			●	●	●	●	●	●	●	●
②　(1)受注ｿﾌﾄｳｪｱ開発-④顧客契約承認	●	●	●	●	●	●	●	●	●	●	●
③　(4)対外提出物	●	●	●	●	●	●	●	●	●	●	●
④　(5)顧客との交際	●	●	●	●	●	●	●	●	●	●	●
⑤　(6)営業-①売上計上報告	●				●	●	●	●	●	●	●
⑥　(6)営業-②売上連携報告	●				●	●	●	●	●	●	●
⑦　(6)営業-③企業ランク見直し	●				●	●	●	●	●	●	●
⑧　(7)請求書対外提出	●				●	●	●	●	●	●	●
⑨　(8)売掛債権回収管理	●									●	●

（出典）ジャステック社資料

⑥　インスペクションの結果分析の一つとして、レビュー・メトリクスの予実評価による変動幅の小さいベースラインの確保がある。簡単に変動幅の小さいベースラインの確保と言うが、手順、手法および規約を整備し、有効なデータを蓄積・評価し組織標準として確保することになる。メトリクスにはレビュー密度、欠陥指摘密度、レビュー指摘効率および手戻り量などがある。さらに機能分類ごと、重要機能および重大欠陥などの傾向分析を考慮することで、より欠陥の偏在性向などを掴むことができ、是正や予防につながる。

⑦　その他として、検出と除去（対策）は同時にしがちであるが分離して行うこと、インスペクションで得た指摘事例等は整理して実事例集（症例ガイダンス等）として横展開すること、およびインスペクションで欠陥が検出された場合にも、その欠陥を作りこんだ当事者に対する人的評価には加味しないことなどがある。

4.3.2 要件定義における未決定事項の取扱い

(1) 目的

要件定義の出来栄えを評価し、今後の対策を確認する。

(2) 説明

- 要件定義工程の最後で、要件定義が十分になされているかどうか、次工程（基本設計）に入ってよいかの確認を行う。
- 要件定義書の完成度の測定は、未決事項を確認し、その未決事項を決めるための補充作業負荷量を測定することによって、要件定義書の完成度が評価できる。

(3) 留意事項

① 未決事項の整理と出来栄えの評価

要件定義工程の最後で、未決事項を整理し、その軽重を判断して、条件付きで次工程に入ることの承認を得るとともに関係者にそのリスクを認識してもらう必要がある。未決事項の整理は、表 4.18 に示す内容などを対象として実施する。

- 未決事項の内容（成果物のレビューの結果、既決事項とされていたが未決事項として残る事項を含む）
- いつまでに決定するか
- 誰が責任者で、誰が協力して決定するか
- 決定するために必要な条件があるか（業務条件、技術条件、工数の負荷条件、担当者の能力など）
- 上記に関する各ステークホルダとの合意
- 予算枠確保のための残件対応に要する工数・費用見積もり

表 4.18　未決事項確認表

未決事項	いつまでに決定するか	決定の責任者	決定のための条件	決定までの予想工数

未決事項確認表を用い、各項目は、誰が責任者で、いつまでに何回打ち合わせすれば決まるのかを想定し、決定までの予想工数を算出する。そして、「（決定までの合計予想工数）÷（あらかじめ設定された要件定義の予算工数）＝追加工数割合」の値を確認する。検討においては、単に残件を決定するまでの費用・期間を見積るだけでなく、残件への対応を後工程で実施した結果、作成済の要件定義成果物に修正が発生するリスクと、リスクが現実のものになった際の対応に要する費用、期間、体制、工程遅延に対するリカバリ計画の評価が必要である。

② 対策

未決事項の決定時期が決まらない場合や時期が大きく遅れる場合は、未決のまま次工程に進んだ場合のリスクと対策をプロジェクト内で承認した上で、未決事項として課題化し、次工程以降で完了まで管理する。経営者やプロジェクト管理者は、未決事項を認識したうえで、このまま次工程に進んでも対応できる範囲であるかどうかを判断する。一番避けたいのは、要件定義工程に積み残した未決事項を十分に認識せず、要件定義が完了したと思うことである。

要件定義の残件の発生状況に応じて、基本設計工数、工期の調整での対応や、要件定義期間の延長、基本設計以降の追加予算と追加工期の確保をセットにした対応を検討する。残件が多い場合には要件定義工程の失敗を想定して体制を変更してやり直すなどの根本策をとる必要がある。

③ 要件定義、基本設計の契約形態と課題、対策

要件定義が確実に行われることが望ましいが、実態は基本設計に入ってから、上記の評価をすることが多い。その場合、既に契約形態が決まっていることが多い。そのため、ここで契約形態と対応についてまとめる。

表 4.19 は、要件定義が終了し、基本設計に入る前に、基本設計担当者による要件定義書の評価とその対策をまとめたものである。タイプ 1 は、要件定義書の修正が少ないと評価された場合、タイプ 2 は、修正が大きいため、このまま基本設計に入れない場合、タイプ 3 は、修正が大きいが、基本設計工程で要件定義を見直しながら進める場合、を示している。タイプ 1 の要件定義書の修正が小さい場合は、不足部分を補充して基本設計作業を進める。タイプ 2 では、要件定義期間内にこの点検期間を設けるか、要件定義期間を延長してこの確認作業を実施する。タイプ 3 は、ユーザ責任で要件定義を見直しながら基本設計を進めるが、どの程度要件定義が不足しているかを評価しておかないと、基本設計が予定通り進まないことになる。

表 4.19　基本設計担当者による要件定義書の評価と対策

タイプ	要件定義書の評価	対　策		基本設計契約形態
		ユーザ	ベンダ	
1	修正が小さい	基本設計の工数、工期を調整	修正が小規模ならば、基本設計工数、工期を調整し、受注する	請負
2	修正が大きい	要件定義期間内にこの要件定義書の点検期間を設けるか、要件定義期間を延長してこの確認作業を実施する	ベンダの担当者は、要件定義書の点検に参画し、要件がまとまったところで、契約形態を判断する	請負 または、 準委任
3	修正が大きい	ユーザ責任で要件定義を見直しながら、基本設計を進める	ユーザの指示に従う	準委任

④　表 4.19 のとおり、ユーザ企業とベンダ企業は、要件定義工程の最後に要件定義書の点検期間を設け、双方の合意を得て次工程（基本設計）に進むのがよい。この時点で決まっていない事項は、基本設計では変更事項として扱うことになる。

4.3.3 要件定義文章の品質向上

(1) 目的

文章の曖昧さにより要件定義が正しく伝わらないことを避ける。

(2) 説明

適切な要件定義文章の書き方を取りまとめる。

「SE を極める　仕事に役立つ文章作成術」(JUAS 編、福田修著)［14］を参考にした。

(3) 留意事項

自身で考えて記述した表現が受け取り手、読み手に意味が正しく伝わるかを注意し、書いた文章を必ず数回推敲する。以下に示すように、要件定義文章の記述規約をあらかじめ決めておき、周知したうえで要件定義書を作成すると、曖昧な文章が少なくなる。

具体的に、要件定義文章（ソフトウェア文章）の書き方について説明する。

① 抽象的な単語は定義を明確にして使用する。特に「ｘｘ管理」は何をすることか、明確に定義する。辞書を引いても出てこない業界の専門用語、企業固有の専門用語、また事業部門固有の言葉、並びに辞書等と異なった意味で使われている用語をまとめておき、関係者間での齟齬をなくす。

② 短文主義、一文一義で書く（長すぎる文は読みにくい文章となる）

一文 40 文字平均が読みやすいと言われている。長い場合でも 1 文 200 文字以内、3 行以上の文は書かない。文には 3 種類ある。単文と重文それに複文である。

単文：A は B である。A が B する。

重文：A は B であり、C は D である。

複文：A が B であるため、C は D である。

技術文章は単文主義が原則である。

③ 否定文、部分否定文、二重否定は、くれぐれも注意し、できるだけ使わないのがよい

④ 冗長、回りくどい、あいまい、嘲笑的な表現を排除する

⑤ 技術文章では、「である調」を用いるのが作法である

「である調」、「ですます調」、「体言止め」を混在させない。

⑥ 書けるものは、箇条書きで書く

⑦ 表にできるものは表にする

⑧ 視覚化する

図やイラストで、視覚化して伝える。

⑨ かなと漢字の比率は、2：1 が読みやすい

漢字の比率が 5 割を超えると読みにくくなる。

⑩　形容詞や副詞を使用した場合は、補足説明が必要である

［形容詞］　大量のハードディスク容量→100GB 以上のハードディスク容量

［副詞］　高速で動く　→入力後 1 秒以内に応答がある

⑪　助詞の使い方を間違えない

助詞を連続させると読み辛くなる。理由として助詞は述部が表している「ありよう」や「ふるまい」を整えるために用いるため、これを連続させると助詞の「ありよう」や「ふるまい」が不明瞭になるからである。

(a)　「は」を連続させない

（誤）今日はテストは楽だった。

（正）今日のテストは楽だった。

(b)　「が」を連続させない

（誤）進捗会議が時間が来るまで続いた。

（正）進捗会議は時間が来るまで続いた。

(c)　接続助詞「が」を用いない

接続助詞「が」には以下の四つの機能があり、これを用いると読み手に四つの内どの意味なのかの選択を求め負担をかける。別の言い方をすると接続助詞「が」は書く立場からは便利な助詞であるがゆえに読む立場に負担をかける。

(i)　二つの事柄について接続詞の役割をする。共存か時間的推移を表す。

（誤）重要な結果が報告されたが、現実には大きな矛盾が萌芽してきた。

（正）重要な結果が報告された。その後、現実には大きな矛盾が萌芽してきた。

(ii)　題目・場面などを持ち出し、それらについて叙述する。

（誤）開発の件ですが、五年前とは状況が変化しています。

（正）開発の件については、五年前とは状況が変化しています。

(iii)　補充的説明の添加

（誤）機能障害はたびたび起きるものだが、多くはバグによるものである。

（正）機能障害はたびたび起きる。その多くはバグによるものである。

(iv)　内容を対比させる。前件に拘束されず後件が存在することを表す。

（誤）文章には自信のある私であるが、できるだけ正確に書くよう心がけている。

（正）文章には自信がある私であっても、できるだけ正確に書くよう心がけている。

(d) 「の」を連続させない

格助詞「の」を連続して使うと文の意味が複雑になり、分かりにくくなる。

(誤) 演算命令の再度の実行の場合は、データの処理速度の時間の確保のための方法・・・

(正) 演算命令を再度実行した場合は、データ処理速度時間を確保するための方法・・・

(e) 「を」を連続させない

(誤) PC を保守などをする安全装置と呼ばれるソフトウェア

(正) PC に対して保守等をする安全装置と呼ばれるソフトウェア

(f) 「に」を連続させない

(誤) 明日までに移行作業に支障がでないよう。

(正) 明日までに移行作業上支障がでないよう。

(g) 「まで」を連続させない

(誤) 夜 9 時までテストが終わるまで待った。

(正) 夜 9 時にテストが終わるまで待った。

⑫　話し言葉と書き言葉

話し言葉を口頭言語と言い、書き言葉を書記言語と言う。口頭言語と書記言語とは本来異質である。すなわち口頭言語は目前の相手に対して当事者的関係においてする伝達の言語である。これに対して書記言語は思考や感覚を自己の内面において客観化する認識の言語である。

表 4.20　口頭言語と書記言語

分類	口頭言語	書記言語
音便・省略	じゃない	ではない
	話している	話をしている
	そんな	そのような
	やっぱり	やはり
	みんな	みな
	話すなんて事は	話すなどということは
	いろんな	いろいろな
	人間なんである	人間なのである
自立語	お父さん、お母さん、僕	父、母、私
	すごく	とても、非常に
	ちゃんと	きちんと、はっきりと、正しく
	一発で	一度で、一回で
接続詞	でも	しかし
	なので	だから、このため
用言語尾	知らなく	知らず
	見れる	見られる
	移らさせて	移らせて
	書かして	書かせて
	みたい	ようだ
	というのは	というものは
助詞	おもしろいなと思った	興味深く感じた
	開くけど、開くけれど	開くが
	たら	なら

第５章　事例編

はじめに

5.1「ビジネスに貢献する要求を見極める」～ビジネス成果と IT 施策の整合性をとる～
サントリーシステムテクノロジー株式会社

5.2「要件量を可視化する」～定量化した要件量を使ったスコープ管理～
株式会社 NTT データ

5.3「業務バリエーションの整理」～業務バリエーションを鳥瞰的に把握し整理する業務シナリオマトリクス～
富士通株式会社

5.4「非機能要求の進め方」～各ステークホルダが協力しながら進めていく方法～
富士通株式会社

5.5「ステークホルダ分析」～多様化するステークホルダといかに合意形成を図るか～
セイコーエプソン株式会社

5.6「要求の定量化による合意形成と膨らむ要求の制御」
株式会社ジャステック

5.7「手戻りコストを抑制する要件定義書のレビュー」
株式会社ジャステック

5.8「要件定義文章の品質向上」～図表を使った記述技法～
株式会社 NTT データ

表 5.1　各社事例と本編との関連

事例	会社名	第3章 昨今直面している要件定義課題を解決するための勘どころ						第4章 要件定義成果物の品質向上 4.3 成果物に共通の留意点		
		3.1	3.2	3.3	3.4	3.5	3.6	4.3.1	4.3.2	4.3.3
5.1「ビジネスに貢献する要求を見極める」～ビジネス成果とIT施策の整合性をとる～	サントリーシステムテクノロジー	✓								
5.2「要件量を可視化する」～定量化した要件量を使ったスコープ管理～	NTTデータ		✓							
5.3「業務バリエーションの整理」～業務バリエーションを鳥瞰的に把握し整理する業務シナリオマトリクス～	富士通			✓						
5.4「非機能要求の進め方」～各ステークホルダが協力しながら進めていく方法～	富士通				✓					
5.5「ステークホルダ分析」～多様化するステークホルダといかに合意形成を図るか～	セイコーエプソン					✓				
5.6「要求の定量化による合意形成と膨らむ要求の制御」	ジャステック		✓		✓	✓	✓			
5.7「手戻りコストを抑制する要件定義書のレビュー」	ジャステック							✓		
5.8「要件定義文章の品質向上」～図表を使った記述技法～	NTTデータ									✓

（注）各社ごとに特有の用語・表現が存在するため、事例編では本編と異なった用語・表現を用いている場合がある。

5.1 「ビジネスに貢献する要求を見極める」の事例 ～ビジネス成果と IT 施策の整合性をとる～ サントリーシステムテクノロジー株式会社

取り組み背景

システム開発スタート時のコンセプト検討段階で狙ったビジネス成果があったにもかかわらず、いざ要件定義を終えてみると、当初のコンセプトとは異なる要件・施策になっていることがままある。それでは、開発したシステムをいくら利活用してもビジネス成果を獲得できない。この課題を解決すべく、サントリーグループでは、リザルトチェーンを描いて、ビジネス成果に直結する IT 施策を明らかにしている。

本編との関連

「3.1　経営や業務に貢献する IT システムの構築」と関連している。

リザルトチェーン

リザルトチェーンとは、獲得したいビジネス成果と、それに必要な IT 施策との関連を明らかにする整理ドキュメントである。このドキュメントを作成することで、売上増大、業務コスト削減、企業価値向上、リスクヘッジなど自ら定義した最終ビジネス成果と、その成果に直結する IT 施策を見える化し、ステークホルダの開発合意ができる。

図 5.1 は、サントリーグループの SCM に関するリザルトチェーンである。一番右側に「業務工数を○○億円削減する」など計 3 つの最終ビジネス成果を配置している。一方、一番左側に「需要計画を取り込む」「計画修正の比較ができるための情報を入力する」などの IT 施策を配置し、その施策が「計画修正検討・比較ができる」という機能を実現できる。次にその機能が「在庫計画・生産配分の調整時間を短縮できる」という中間成果を生み、「業務コストを○○億円削減する」という最終ビジネス成果に直結する。

このリザルトチェーンの凡例は、図 5.2 のとおりである。リザルトチェーンは基本的には「成果」「施策」「前提・制約条件」の連鎖で表される。また、それぞれの因果関係は「貢献」と言う。

「成果」は、リザルトチェーンの左から右へ、①構築する機能、②その機能が業務に与える好影響、③その影響による活用（中間成果）、④その活用による恩恵（最終ビジネス成果）に分類される。「成果」の中には、業務へマイナスの影響を与えるものがあり、それらに対してはマイナスを打消す施策が必要である。

「施策」は、成果をもたらす活動であり、IT による施策だけではなく、BTOPP（ビジネス戦略、技術（IT）、組織・文化、業務プロセス、人）の観点で施策を抽出し、記載する。

「前提・制約条件」は、施策がうまく機能するための条件である。

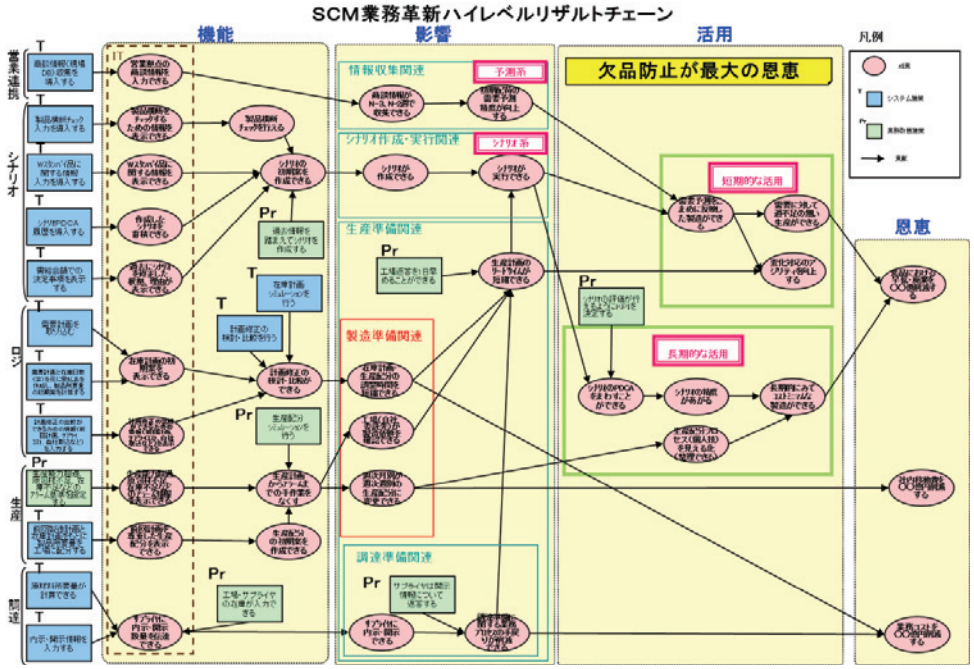


図 5.1 リザルトチェーンの事例

記号	概要
成果	プログラムにおいて実現すべき成果 （最終および中間成果から構成）
施策	1つ以上の成果をもたらす活動 （BTOPP全てを考慮して定義）
前提・制約条件	施策が上手く機能するための条件 （リスク）
貢献	因果関係 （上位施策に対する下位施策の貢献度）

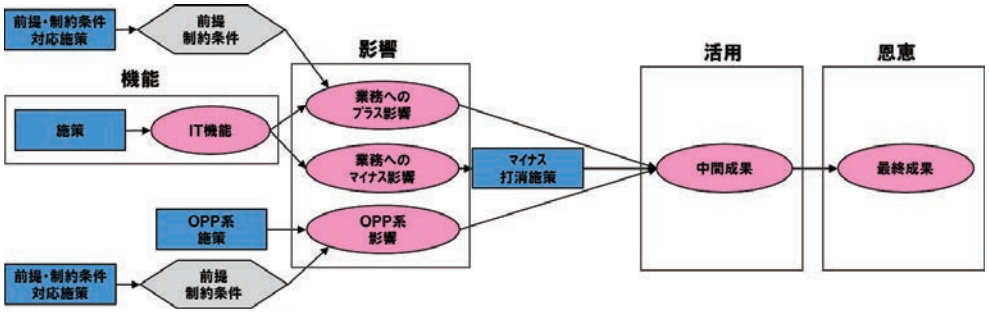


図 5.2 リザルトチェーンの凡例

これらを踏まえて、リザルトチェーンの作り方を、以下図 5.3（STEP1～5）に示す。このリザルトチェーンを作成することで、捉えた真の問題を元に、曖昧な目的や手段を具体化し、経営方針やビジネス目的と IT 施策との関係を明確にしている。

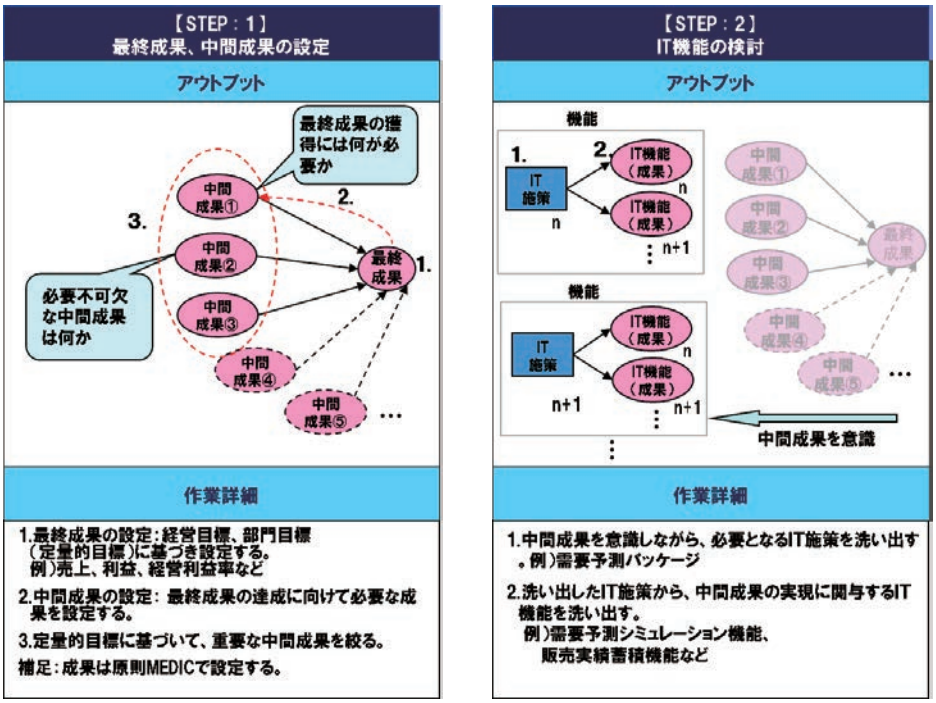


図 5.3　リザルトチェーンの作り方（1/2）

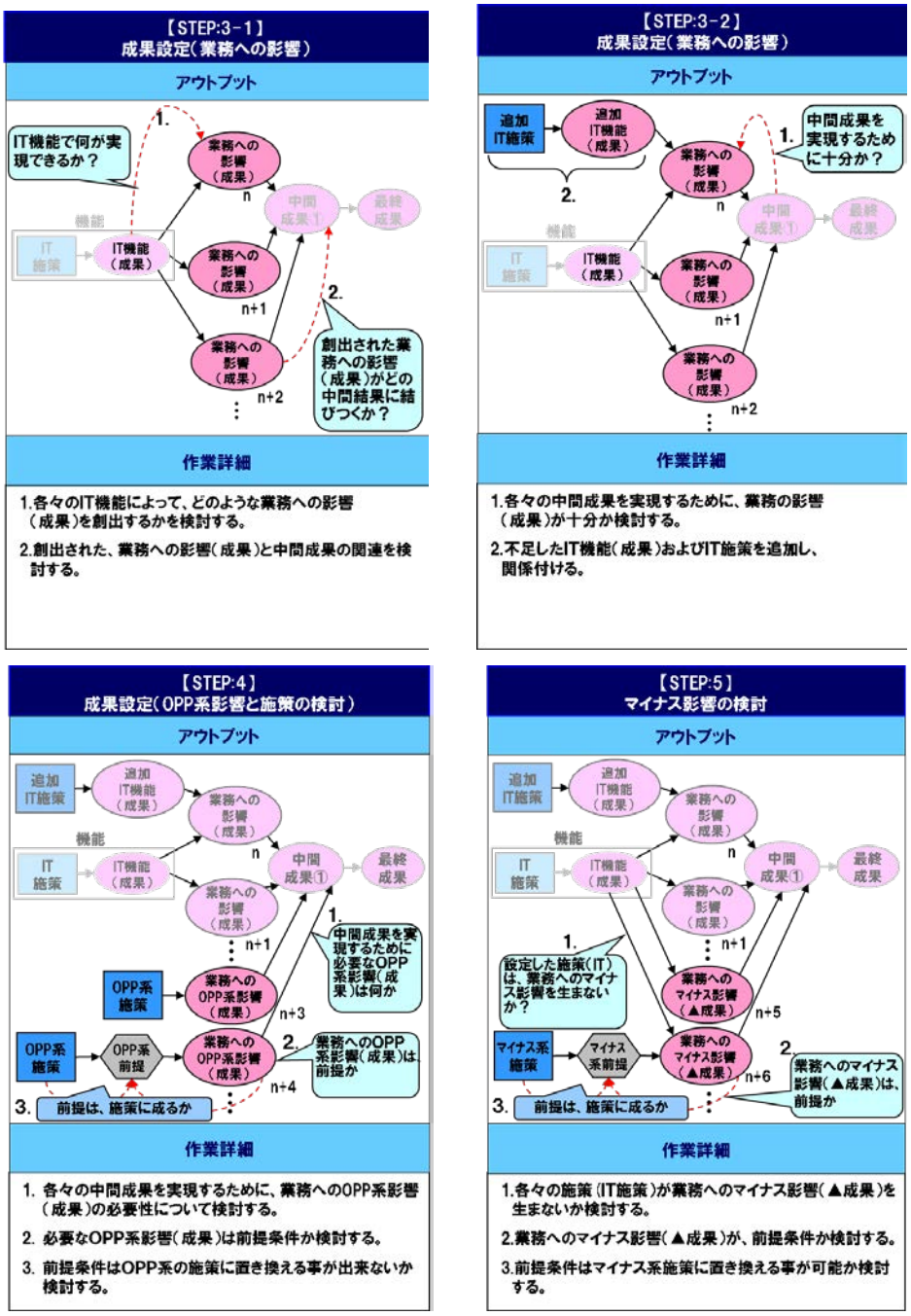


図 5.3　リザルトチェーンの作り方 (2/2)

5.2 「要件量を可視化する」の事例 ～定量化した要件量を使ったスコープ管理～ 株式会社 NTT データ

取り組み背景

近年のシステム開発プロジェクトでは、ビジネス環境、技術環境の変化が早く、システム要件の変化も早い。しかし、スコープ（要件量）が増大していることに気づかないまま要件定義を行ってしまい、問題化するプロジェクトが散見される。そのようなプロジェクトではスコープ増加についてユーザと認識の共有ができておらず、すべての要求を受け入れてしまったり、全要件量を把握しておらず、適切なスコープ調整ができていなかったりする。

このような背景のもと、NTT データでは上流工程の早い段階でスコープを定量化し、プロジェクトライフサイクルを通じて、スコープ増減を監視、管理するマネジメント手法をグループ内で展開している[1]。本手法を適用することにより、ユーザとのスムーズなスコープ調整、要件量増減の早期捕捉による手戻りの減少などの効果を確認している。

本稿では、要件量管理の手法を概説する。

本編との関連

「3.2　膨らむ要求のコントロール」と関連している。

要件量管理

要件量とは、システム化対象とした要件を特定の規模尺度（例として、ファンクションポイントや機能数など）で定量化したものである。本手法では機能要件を対象として、要件量を用いて定量的にスコープの増減を管理する。

(1) 要件量管理手法の概要

本手法の全体フローを図 5.4 に示す。本手法は以下に示す、2 つの特徴がある。

- 要件定義着手前に見積もった要件量による基準（ベースライン）の設定
  予算に基づく開発中に受け入れ可能な要件量（許容ライン）やスコープ増減の基準を定める。ベースラインを初期にユーザと合意することで、その後のユーザ調整に役立てる。
- スコープに対する変更要求と検討状況の監視

ベースラインを設定したあと、要件量の変動を管理する。これにより、要件全体をコントロール可能であり、上流工程における要件量の増大を防ぐことができる。

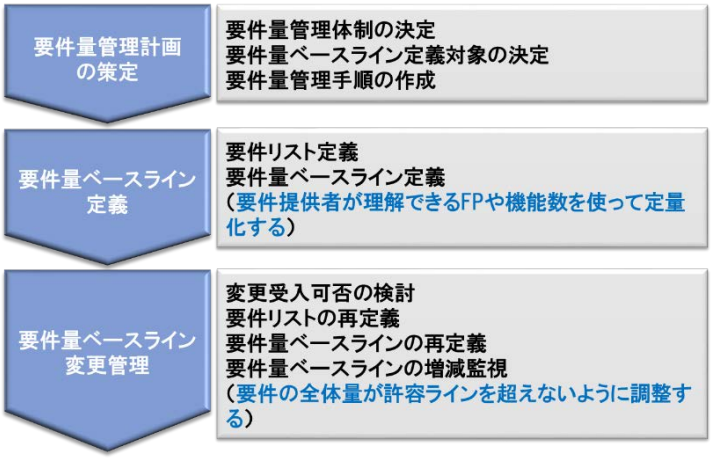


図 5.4 要件量管理手法の全体フロー

(2) 要件量管理の適用例

本手法の適用例を示す。

- 要件量定義

  ユーザからのシステム要求に基づき、機能要件の一覧を作成する。一覧に記載した機能要件を事前にユーザと合意した規模尺度を用いて定量化し、要件量を算出する。また、各機能要件の検討状況を記載する。定期的（週次、日次）にその時点での要件量を把握できるよう工夫する。

- 要件量監視

  上記の機能要件一覧を用いて、要件量を監視する例を図 5.5 に示す。

  図の左側は、全体の要件量の増減と要件の検討状況を可視化しており、A は検討中、B は検討済みの要件を表している。

  一方、図の右側は要件に対する変更要求の件数を示しており、C は変更要求の総件数、D は検討済みの変更要求件数である。

このように変動を要件量と要求件数の両面で可視化することで、スコープの変動を直観的に把握することが可能となる。

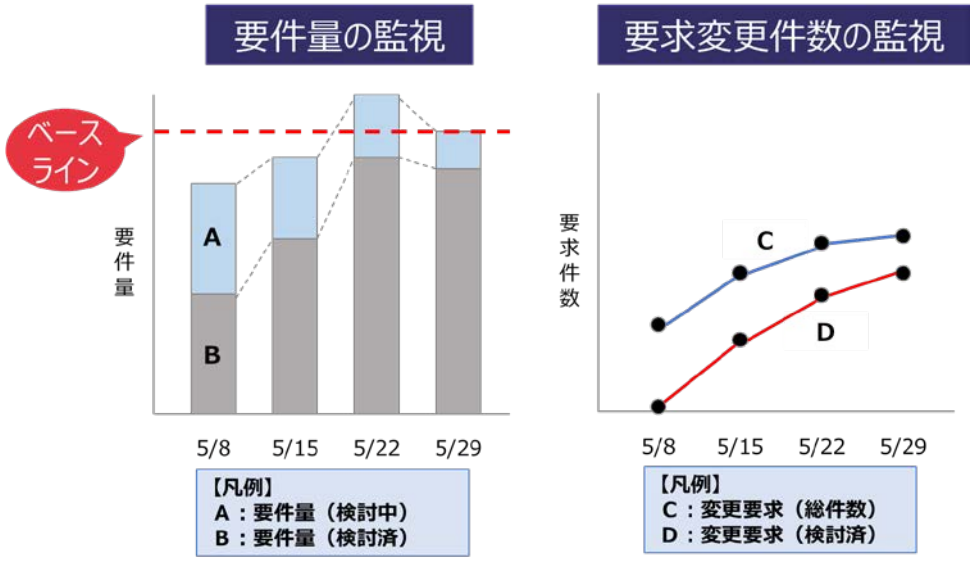


図 5.5 要件量推移の可視化例

参考文献

[1] 長谷川 智亮，池田 訓子，柴原 英明，山中 啓之，藤貫 美佐 ： 上流工程での要件定量化によるスコープマネジメント， プロジェクトマネジメント学会 第 26 回秋季研究発表大会 2015

5.3 「業務バリエーションの整理」の事例

～業務バリエーションを鳥瞰的に把握し整理する業務シナリオマトリクス～

富士通株式会社

取り組み背景

企業において、各部署に存在する似て非なる業務を標準化できないかという悩みがある。富士通の要件定義手法 Tri-shaping[7]の中に上記のような類似業務パターンの整理方法として業務シナリオマトリクスがある。業務シナリオマトリクスは、業務の流れのパターンを鳥瞰的に把握し、業務プロセスの標準化や複雑さの削減につなげるものである。ここでは、その使い方を紹介する。

本編との関連

「3.3　業務の複雑性を軽減」で利用できる方法の事例。

業務シナリオマトリクスとは

業務の流れを表現するためのドキュメントとして、業務フローが存在する。従来は業務フロー上で、例えば大口顧客の場合はこのような業務の流れになる、個人顧客の場合はこうなる、と色分けして流れをトレースしていた（図 5.6）。このトレースパターンは、業務を検証するだけでなく、システムテストのシナリオ（業務の大きな流れを表したもの）を抽出する際にも利用価値が高かった。

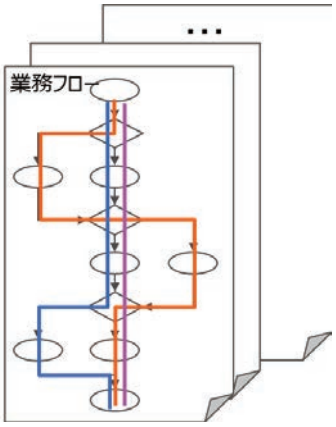


図 5.6　業務フロー上でのトレースイメージ

最近では個別最適から全体最適へといったシステム化対象範囲の広がりや、商品や顧客の多様化により業務パターンが増加し、業務フローを記述したら箱ファイル 1 冊になることもある。また、自社の業務がどれだけのパターンを持っているかが、もはや明確に把握しきれなくなっていないことも多い。

そこで、Tri-shaping では、業務フローを表形式で記載することを提案している。それを業務シナリオマトリクスと呼んでいる。

業務シナリオマトリクスのイメージを図 5.7 に示す。

シナリオ名	頻度	商品					顧客						オーダー形態			業務の流れ								
		有形	サービス		単品	セット	代理店	法人		個人	国内	海外	一般	レンタル	無償	顧客確認	見積り	在庫引当	出荷指示	直送	出荷	配送	請求	・・・
			定額	従量				大口	小口															
パターン1	30	●			●		●				●		●			①		②	③		④	⑤	⑥	
パターン2	20	●			●			●			●		●			①	②	③	④		⑤	⑥	⑦	
パターン3	2	●			●				●		●	●	●			①		③	④	⑤		⑥	②	
パターン4	1	●			●					●	●	●	●			①		③	④	⑤		⑥	②	
・・・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・	・

図 5.7　業務シナリオマトリクのイメージ

【業務シナリオマトリクスの表記】

［シナリオ名］：横一列のパターンに名前を付けたもの

［重み］　　：業務における各シナリオの重み(業務に占める重要度)を表したもの。例として「頻度」を上げているが売上高などで表現しても良い。

［管理対象のバリエーション］：管理対象の種類の組み合わせを表現したもの。図 5.3.2 の「商品」から「オーダー形態」のように管理対象の種類の組み合わせによって業務のパターンが異なる。

［業務の流れ］：大きな業務の流れを表したもの。数字は、業務フロー上の処理順番を示している。順番が異なるだけのパターンもある。

【書き方のコツ】

「シナリオ名」は最後に命名する。最初から全業務のパターンが洗い出されているわけではないため、パターンを洗い出した後、パターンごとの特徴を適切にあらわす名前を付与する。

「重み」は、正確性を追求しない。大まかでよい。

「管理対象のバリエーション」は、初めからその業務処理の条件のすべてを正確に描こうとしないことがコツである。重要な管理対象からまず記載する。その上で、区別したい条件が浮かび上がってきたら、その都度バリエーションを追加する。そのような手

順でこの表を作成することにより、当初認識できていなかった業務の流れを左右する重要な管理対象のバリエーションの存在が認識できるようになる。

「業務の流れ」は、あまり詳細なレベルではなく、「見積り」や「在庫引当」といった大まかな単位の業務プロセスを使用して表現する。

業務シナリオマトリクスでの分析

いきなり To-Be の業務シナリオマトリクスを作成するのではなく、まず As-Is の業務シナリオマトリクスを作成して現在の業務を表現するのが良い。そうすると、自社の業務のパターンの多さに気がづき、標準化や統廃合による業務シナリオパターンの削減がどれほど必要性であるかを実感できる。

【統廃合の観点の例】

- 業務プロセスの統廃合
  実施される頻度の少ない業務パターンがあるなら、その業務パターンを廃止する。
- 管理対象のバリエーションの統廃合
  管理対象が異なるだけで他の業務パターンとほとんど同じパターンである場合には、類似するパターンと統合して、管理対象のバリエーションを削減する。
- シナリオパターンの統廃合
  複数の業務パターンにおいて、ほとんど同じ業務プロセスになっている場合には、同じ業務パターンとして業務シナリオを標準化する。

滅多に発生しない業務パターンは人手による処理でカバーするなどの切り分けによって、業務システムはより単純化することができるので、ぜひ業務パターン整理の実施を検討していただきたい。これまでの実施においては、As-Is で 500 を超えるパターンを To-Be では 400 弱まで削減した事例がある。これは、大きな業務改革であるとともに、業務システム機能の大幅な単純化につながっている。

このように、大局的な視点で業務の標準化や統廃合を実現できれば、次期開発予定業務の整理にも効果は大きいものとなる。また、開発対象のシステムのテストシナリオを検討するときにも非常に役立つ。

5.4 「非機能要求の進め方」の事例 ～各ステークホルダが協力しながら進めていく方法～ 富士通株式会社

取り組み背景

富士通の要件定義手法 Tri-shaping[7]では、要件定義工程での非機能要求の進め方をガイドしている。IPA/SEC から提示されている非機能要求グレード[6]を基本におき、要件定義工程において経営層や業務部門と協力しながら重要な要求を引き出し、そこを中心に最終的に非機能要求グレードに示しているメトリクスを決めていくプロセスをガイドしている。

非機能要求を要件定義工程において実施するためのポイントは、3.4 節で述べている。ここでは、非機能要求が合意形成にいたるまでの全体の流れを紹介する。

本編との関連

「3.4　要件定義工程からの非機能要件定義」で利用できる事例。3.4 節では勘どころを述べているが、ここでは全体の進め方を述べる。

非機能要求グレードの拡張

Tri-shaping は非機能要求グレードを非機能要求の基本に置いているが、要件定義工程で円滑に合意形成ができるように工夫している。

- 最終結果のメトリクスの提示だけではなく、そのメトリクスに行き着くまでのプロセスを可視化している。
- 非機能要求グレードはシステム基盤をスコープに 236 のメトリクスを設定しているが、Tri-shaping では業務アプリケーションも含めたシステム全体をスコープとしていることから 337 のメトリクス設定に増えている。

非機能要求の進め方

図 5.8 に Tri-shaping の非機能要件定義の流れを示す。

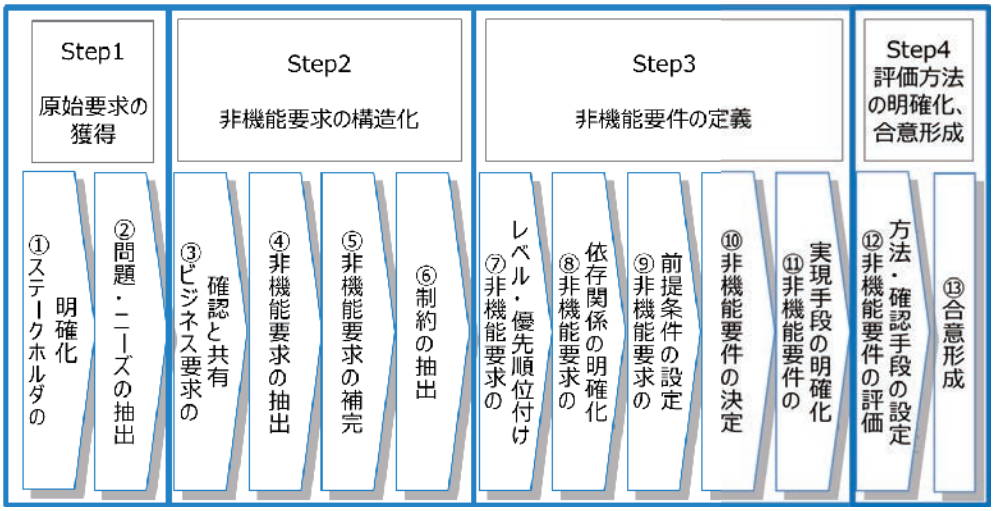


図 5.8　Tri-shaping の非機能要件定義の流れ

非機能要件定義には大きく 4 つのステップがあり、それらは 13 のタスクに分類される。以下に、各ステップおよびタスクの概要を説明する。このうち特にポイントとなる Step2、Step3 については、イメージを図 5.9、図 5.10 に示す。

【タスクの概要】

Step 1

①　ステークホルダの明確化

業務の実現の観点だけでなく、業務を遂行するための一連の作業（業務運用）や IT サービス提供、システム運用・保守の観点で、要求を抽出するステークホルダを洗い出し、各ステークホルダの役割、権限、関係などを明確にする。

②　問題・ニーズの抽出

システムに関わるすべてのステークホルダの満足度を高めるために、業務運用の観点だけでなく IT サービス提供、システム運用・保守の観点で、各ステークホルダの様々な関心事から問題・ニーズを洗い出す。

Step 2

③　ビジネス要求の確認と共有

経営ビジョン・経営方針、部門ビジョン・部門方針、システム化方針、スローガン、経営目的・経営施策、業務目的・業務施策などの情報から、数量・時間・場所などを観点に、要求抽出シート（3.4 節で紹介）を使って、ビジネス要求に含まれている業務特性・業務継続・業務連携・業務量・運用方針・セキュリティなど、トップダウンの非機能要求を確認し、その結果をステークホルダ間で共有する。

④　非機能要求の抽出

要求抽出シートを使って、非機能特性である可用性、性能・拡張性、運用・保守性・移行性などを観点に、ボトムアップの非機能要求を抽出する。

⑤　非機能要求の補完

現行システム仕様や類似システム仕様、あるいはソリューション仕様など、ステークホルダが参照し共有できるモデルを具体的に例示し、明示的に示されなかった暗黙の非機能要求を抽出することにより、ボトムアップの非機能要求を補完する。

⑥　制約の抽出

法令や組織規定、提携先とのインタフェースなど、要求を実現する上でのビジネス観点の制約や、利用するサービスや適用する製品など、技術観点の制約を抽出し、システム化や業務・システム運用に制限を与える条件を明らかにする。

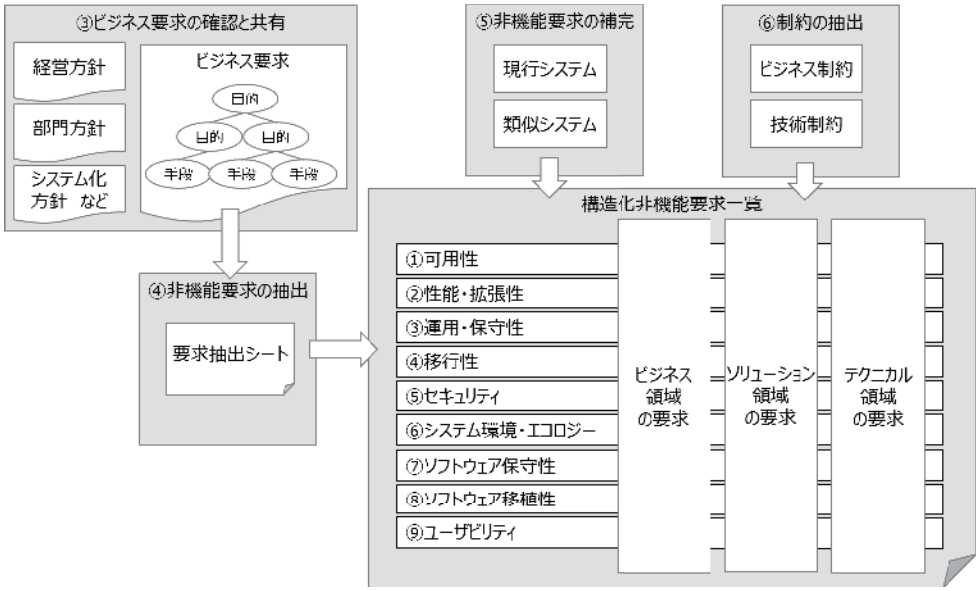


図 5.9　Step2 のイメージ

Step3

⑦　非機能要求のレベル・優先順位付け

制約による非機能要求の実現可否や、要求の重要度や緊急性などから「Must」「Need」「Want」「Hope」のように実現の優先度を決めていく。その際、一般的にはビジネス領域にある非機能要求の優先度は高いものと判断する必要がある。

⑧　非機能要求の依存関係の明確化

制約とともに非機能要求相互の依存関係を明らかにする。依存関係があるものについて、矛盾した要件がないか確認する。非機能要求間でトレードオフがある場合は、要求度合いや優先順位付けをもとに、どちらの要求を優先して採用するかを判断する。

⑨　非機能要求の前提条件の設定

システム化や業務・システム運用に必要な条件にもれがないかどうかを確認し、要求や制約として抽出されていなかった非機能要求を要求として設定する。また、非機能要求を達成するときに前提となる条件があれば設定する。設定した前提条件は、リスクとして管理する。

⑩　非機能要件の決定

抽出された要求と制約、前提条件を整理し、非機能要件定義書に、目標となるメトリクスを設定していく。

⑪　非機能要件の実現手段の明確化

非機能要件の実現性や技術的根拠を示すために、非機能要件を実現する手段を明らかにする。その際は、システム構成要素ごとに、実現手段の実装範囲に他との隙間や重複がないように実現性を検討する。なお、具体的な実現方式は設計工程で具現化する。

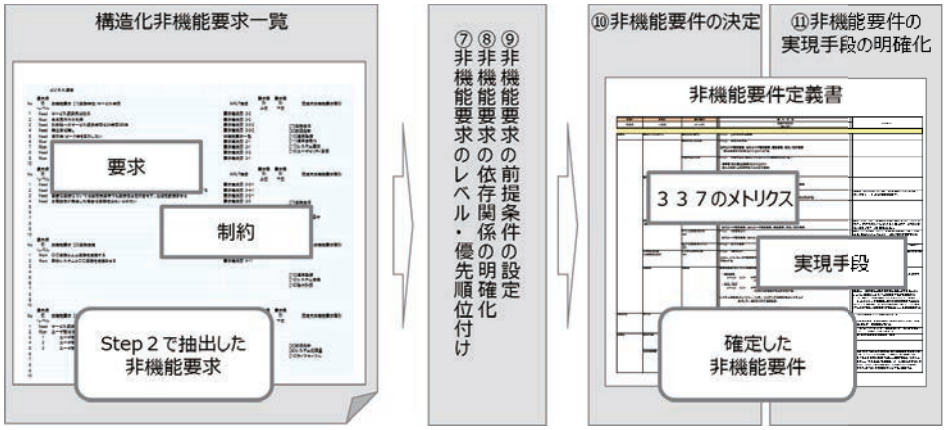


図 5.10　Step3 のイメージ

Step4

⑫　非機能要件の評価方法・確認手段の設定

非機能要求が満たされていることの妥当性を確認するため、測定する項目や評価方法、評価タイミングを決める。定性的な内容の要件など、ステークホルダによって認識が異なる可能性があるものは、要件定義段階であらかじめ評価項目を定めておき、ステークホルダ間で合意する。また、評価実施のために必要なツールなどの確認手段を明らかにする。

⑬　合意形成

非機能要件定義書、評価方法・確認手段について、ステークホルダ相互の意見の一致を図る。なお、未確定要素がある場合は、前提条件を見直すほか、次工程に影響をおよぼさないよう、非機能要件を確定する時期をステークホルダと合意する。

非機能要求の進め方について、本編では詳細な説明はできなかったが、イメージを掴んでいただけたと思う。Tri-shaping は、非機能要件であるメトリクスを決めていくまでのプロセスをガイドし、そこに必要な道具を用意している。また、ビジネス要件、システム化機能要件を中心とした本編とここで紹介した非機能要件編の両方のガイドを提示している。

5.5 「ステークホルダ分析」の事例
～多様化するステークホルダといかに合意形成を図るか～
セイコーエプソン株式会社

取り組み背景

セイコーエプソンでは、事業環境の変化やメガトレンドなどを踏まえ、中期経営計画として長期ビジョン「Epson 25」を制定した。この「Epson 25」実現に向け、IT 基盤をグローバルに統一・強化するミッションに取り組んでいる。

その要となる情報システムについては、業務オペレーションの全体最適化に向け、社内 IT 環境と業務システムの計画的な企画・設計・導入に取り組んでいる。本取り組みでは、これまで事業毎に使用してきた異なるシステムを、業務領域毎に共通化・標準化する必要がある。そのため、既存システムのような単一事業に関わるステークホルダのみならず、各事業間や関係会社、経営層、取引先も含めた広範囲なステークホルダと合意形成を図っていく必要がある。

本事例は、そのような多様化するステークホルダといかに合意形成を図るかについて、セイコーエプソンの取り組みを紹介するとともに、読者の何らかの手助けとなることを目的としている。

本編との関連

「3.5　多様化するステークホルダとの合意形成」と関連している。

ステークホルダの特定

多様化するステークホルダと合意形成を図るために、まず、ステークホルダの特定をする。ステークホルダとは、開発対象のシステムから直接、または間接的に便益を得る組織や人であり、プロジェクト発足時（企画段階）の活動の中で特定する。ステークホルダの特定に当たっては、対象のシステム開発に関係するステークホルダを漏れなく特定（識別）することが重要である。ステークホルダの特定や分析が不十分な場合、以下のような問題が発生する。

- キーパーソンでないステークホルダへのヒアリング実施による変更要求の手戻り発生
- ステークホルダの特定漏れによる要求（機能）漏れの発生

上記のような問題を未然に防止するためには、要求の源泉となるステークホルダを特定・分析し、合意されたステークホルダを中心に要件定義の実施、ならびに合意形成を図ることが重要である。ステークホルダの特定では、対象システムがどのステークホルダに対して価値・サービスを提供するのかに着目して、漏れのないようにステークホルダの特

定を行う。その際には、対象のシステムによって、価値・サービスを受ける直接的な受益者だけではなく、間接的に影響を受ける関係者についても、漏れなく特定する。

どのステークホルダが、価値・サービスを直接的に受ける受益者なのか、または間接的に影響を受ける関係者なのかは、対象システムによって異なるが、その一例を以下に示す（表 5.2）。

例えば、経営情報システムの場合、経営判断に必要な経営情報をマネジメントコックピットとして提供する。その場合、価値・サービスを直接的に受ける受益者は経営層であり、間接的に影響を受ける関係者は各事業・関係会社や事業管理部門となる。また、経営情報システム構築に関わるプロジェクト関係者として、システム部門やベンダを抽出する。

さらに、場合によっては、事業管理部門と連携して業務を行うステークホルダ（生産管理部門など）や、事業管理部門に情報提供などの支援をするステークホルダ（経理部門など）もステークホルダとして特定した方がよい場合もある。

最初は、ステークホルダの漏れを防ぐ観点で広めにステークホルダを特定し、プロジェクト体制の整備や要求を明確化する中で、実際にどの範囲のステークホルダまでをヒアリングや合意形成の対象とする必要があるかを判断するとよい。

表 5.2 ステークホルダの特定

対象ｼｽﾃﾑ / ｽﾃｰｸﾎﾙﾀﾞ	経営情報ｼｽﾃﾑ	生産・販売ｼｽﾃﾑ	顧客向けCMS	ｺｰﾎﾟﾚｰﾄｼｽﾃﾑ	ｺﾗﾎﾞﾚｰｼｮﾝｼｽﾃﾑ
顧客	ー	ー	◎	ー	ー
経営層	◎	△	△	△	◎
各事業・関係会社	△	△	△	△	◎
業務部門	△	◎	◎	◎	◎
利用部門	ー	△	ー	◎	◎
システム部門	□	□	□	□	□
ベンダ ※委託ありの場合	□	□	□	□	□
取引先	ー	△	△	△	△

（凡例）◎：価値・サービスを直接的に受ける受益者
△：間接的に影響を受ける関係者
□：プロジェクト関係者
ー：概ね関係なし

ステークホルダの分析

ステークホルダを特定する中で、「ステークホルダリスト」を作成し、ステークホルダを分析する（表 5.3）。ステークホルダリストでは、組織／個人の視点で、次のような事項を明確にする。

- 組織の視点
- 会社／事業部／部門：社内外のステークホルダの組織名
- ステークホルダ：ステークホルダの種類（顧客、経営層、各事業・関係会社、業務部門、利用部門、システム部門、ベンダ、取引先など）
- 個人の視点
- 氏名：個人名（キーパーソンや役職者中心にヒアリングや合意形成の対象者を絞り込む）
- 役職：役職名（役員、部長、課長など）
- 役割：プロジェクトでの役割（業務部門担当者、システム部門担当者、プロジェクト管理者、プロジェクト推進責任者、ステアリングコミッティ、システムオーナなど）
- 影響度：プロジェクト活動における影響度合い
- 対立関係：ステークホルダ間のコンフリクト

表 5.3 ステークホルダリスト

組織		個人				
会社／事業部／部門	ステークホルダ	氏名	役職	役割	影響度	対立関係

前記事項のうち、「役割」、「影響度」、「対立関係」は特に重要な事項である。

役割については、実行担当者、意志決定者、相談者、報告者に分類し、合意形成を図る手順に活用する。

例えば、大きな業務変革を伴う場合、まず、相談者である社内外の有識者に意見を聞いた上で、意志決定者であるプロジェクト推進責任者やステアリングコミッティに説明を行い、方向性について同意を取り付ける。次に、実行担当者である業務部門担当者に説明し、報告者であるシステムオーナに進捗状況を報告することで、円滑な合意形成を図ることができる。なお、これらの役割は、ステークホルダの特定・分析の段階ではまだ決まっていないことも多いが、プロジェクト体制を整理する中で明確にする。

また、影響度については、ステークホルダの発言や行動によって影響を受ける可能性で判断し（プロジェクトが円滑に進む良い影響の場合と、悪い影響の場合がある）、その度合いによって、プロジェクトへの巻き込み方やステークホルダへの接し方（事前確認を行うなど）を変えるようにする。

なお、対立関係については、後述の「コンフリクト（対立関係）の解消」で述べる。

コンフリクト（対立関係）の解消

プロジェクト活動では、異なる立場のメンバが同時に活動するため、業務部門、システム部門、ベンダなどの各ステークホルダの立場によって関心事や優先事項が異なり、コンフリクトが発生する。コンフリクトの存在は、合意形成の障壁となることが多い。

コンフリクトが起こりやすいパターンを以下に示す（表 5.4）。

表 5.4 コンフリクトが起こりやすいパターン

ｽﾃｰｸﾎﾙﾀﾞ / 合意内容	業務部門	システム部門	ベンダ
業務要求	他部門よりも自部門の要求を優先したい	シンプルかつ標準化したい	ー
システム要件	手厚い仕様にしたい	最小限の仕様にしたい	契約通りにしたい
システム変更要求	システム仕様はいつでも変更したい	早く仕様を確定したい	早く開発に着手したい
システムの QCD（品質/コスト/納期）	あまり関心がない	QCD 達成は必達	契約通りにしたい

業務部門、システム部門、ベンダとの間で、上記のような複雑なコンフリクトが存在するため、ステークホルダ分析の中でコンフリクト解消に向けた対策を打つ必要がある。

コンフリクトを解消するためには、相互理解を深めることにより一致点と相違点を層別し、対立の原因（論点）を明確化する。また、対立の原因（論点）について、プロジェクトの目的に立ち返り、より高い視点やより広い視野から話し合い、互いに協力しながら第 3 の案を創造し、Win-Win の関係になるよう努力する必要がある。

しかし、必ずしもいつも第 3 の案が創造できるとは限らない。また、システム要件やシステム変更要求の場合、コストやスケジュールが超過しないように、要求を抑え込もうとするシステム部門のプロジェクトマネージャと業務部門との対立はよく発生する。そのような場合、強引に結論を急がず、何を優先するかについて、参加者全員が納得する評価基準をあらかじめ決めておくとよい。

どの要求を優先するかは、対象システムが何を重要視するかによって異なるが、例えば、法令順守、リスク回避、プロジェクト目標達成への寄与、効率化といった観点で要求の優先度付けをすることで、ステークホルダの納得感を高め、コンフリクトを解消するようにする。

ステークホルダに合わせた合意形成

多様化するステークホルダと合意形成を図るには、前述の「ステークホルダの分析」、および「コンフリクトの解消」を考慮して、ステークホルダに合意を取り付ける際の方法（シナリオ）を事前に決めておくことが重要である。ステークホルダリスト（表 5.3）で、ステークホルダとその役割、影響度、対立関係の分析を行った結果と、合意を取り付ける内容を基に、合意形成の方法をあらかじめ決めておき、確実に合意形成を図る。以下にその事例を示す（表 5.5）。

表 5.5 ステークホルダに合わせた合意形成

ステークホルダ	合意を取り付ける内容	合意形成の方法
業務部門 システム部門	業務要求・ システム要件	(1)要件定義 10 箇条 および、要件定義レビュー
	システム変更要求	(2)変更要求管理
	システムの QCD	(3)システム DR（デザインレビュー）
ステアリング コミッティ	ハイレベルな プロジェクト課題	(4)ステアリングコミッティの設置
各事業・関係会社	各事業・関係会社 への影響	事業連絡会、グローバルミーティング
経営層	経営方針との整合	経営会議

例えば、業務部門に対して、システム要件について合意を取り付ける場合には、要件定義 10 箇条、および要件定義レビューの合意プロセスを経ることによって、確実に合意形成を図る。

ステークホルダに合わせた合意形成（表 5.5） のうち、以下の合意形成方法について詳細を示す。

(1) 要件定義 10 箇条
(2) 変更要求管理
(3) システム DR（デザインレビュー）
(4) ステアリングコミッティの設置

なお、要件定義レビューについては、5.7 手戻りコストを抑制する要件定義書のレビューの事例を参照されたい。

(1) 要件定義 10 箇条

セイコーエプソンでは、要件定義の抜け・漏れ・誤りにより手戻り防止と、業務部門との確実な合意形成を目的に、要件定義で何をするべきかをまとめた「要件定義 10 箇条」を作成した（表 5.6）。

業務要求、およびシステム要件は、本 10 箇条に沿った活動と要件定義レビューにより、業務部門と合意形成を図る。各項目のポイントを以下に例示する。

表 5.6 要件定義 10 箇条

項目	分類	ポイント
1. 要望と要求の切り分け	業務要求	・要望と要求を区別し、不必要な要望は検討の対象から外す ・要求では、実現すべき目的を明確にする
2. 現状業務の把握・理解		・業務要求の対象となる業務プロセスの現在の姿を理解する ・具体的には、業務の目的、業務フロー、使用する帳票、現在使用しているシステム、法令などの制約
3. 業務要求の理解		・業務要求を漏れなく洗い出すための To-Be 業務の詳細化と、経営上の問題／課題の解決に寄与するかの確認をする ・業務要求一覧と業務フローを作成する
4. 5 ゲン主義に基づく整理		・5 ゲン主義（3 現主義「現場」「現物」「現実」に「原理」「原則」を加えた考え方）で業務要求を整理し、俯瞰的な視点で要件定義を実施する
5. 業務上の問題・課題の明確化		・業務上の問題・課題を確認できる一覧を作成する ・対応の有無、実現手段（システム or 業務変更）を決定する
6. システム化範囲の明確化	システム要件	・システム構成図を作成し、システム化範囲を決定する ・As-Is と To-Be の両面から明確化する
7. 要件定義への落とし込み		・システム要件定義書を作成する ・要求、目的／理由、システム仕様、制約などを明記する
8. システム化の品質・コスト・納期		・システムの QCD のバランスに配慮し検証する ・無理な納期設定による品質低下、実現する要求に対する費用対効果、過剰品質による納期設定を検証する
9. 業務部門との合意形成	合意形成	・システム化範囲、システム要件、システムの QCD などを業務部門と合意する
10. 調整会議の効率化		・ヒアリングを効率的に行うために、会議の目的、出席者、会議の進め方、ファシリテーション方法、会議サイクルなどを事前に会議設計する

(2) 変更要求管理

システム開発途中の要求の変更は、システムの QCD に与える影響が大きいため、事前にルール化しておくことが重要である。すでに承認された要求・要件については、ベースラインを設定し、以降の変更は変更要求管理の手順を通してのみ変更できるルールとすることで、変更要求に関して、業務部門とシステム部門との間で合意形成を図る。

変更要求においては、第 4 章の問題で述べているような要件定義の不具合に起因する後工程からの手戻りと同様に、開発工数の増大や開発工期の遅延に繋がることが多いため、慎重な判断が求められる。その際には、事前にルールを決めておき、できるだけ機械的に判断できるようにしておくと、業務部門の納得性が高く、合意形成を図りやすい。

具体的には、以下のような変更要求管理フローとなる（図 5.11）。

①課題リストの中で、変更要求に繋がる項目を「変更要求リスト」に記載し、一覧管理する（表 5.7）。

②変更要求の実施可否を評価する。その際、以下のような点がポイントとなる。

- 合目的性：システム構築の目的と合致しているか
- 効果への貢献度：効果創出への貢献度はどれ位か
- システムの QCD への影響：プロジェクト全体の QCD などの制約事項への影響度合いはどれ位か。変更による既存システムへの影響範囲の特定とその見積りを行う
- 実施タイミング：どのタイミングで実施するのが適切か

③上記評価内容をプロジェクト管理者（PM）以上の責任者が総合的に判断し、各変更要求の実施可否を承認する。

④実施可として承認された要求については、関連ドキュメントを更新するとともに関係者に周知し、ベースラインに取り込む。

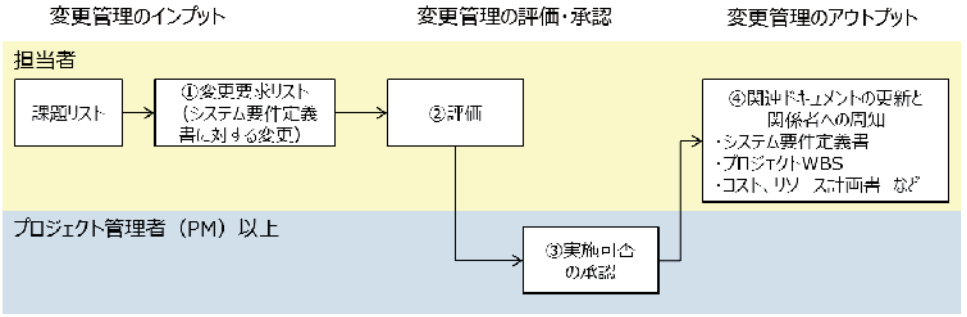


図 5.11 変更要求管理フロー図

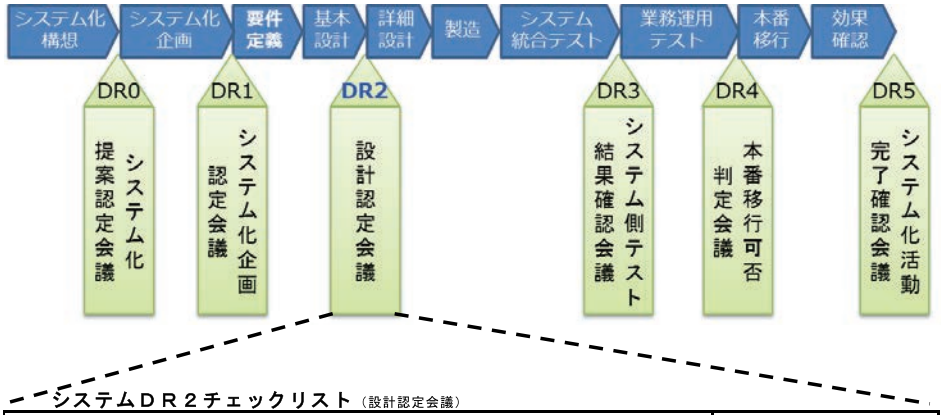
表 5.7 変更要求リスト

ID	起票日	起票者	変更内容	変更理由	影響	対応可否	承認者	完了日

(3) システム DR（デザインレビュー）

セイコーエプソンでは、システム DR（デザインレビュー）基準を制定し、必須の実施事項としている。システム DR は、システム構築における各工程での成果物を、複数のステークホルダで確認することで、システム構築におけるシステムの QCD 確保を目的としている。具体的には、システム化構想から本番稼働後の効果確認までの開発ライフサイクルの中で、節目となる地点で、チェックリストのレビュー項目を基に、各チェックポイントに従った確認を行う（図 5.12）。

デザインレビューの参加者が、成果物について指摘し合うことにより、対象工程での問題・課題を業務部門とシステム部門の双方で共有し、合意形成を図る。その際には、各ステークホルダが当事者意識を持って活発な議論を行い、単なるセレモニーとしないようにすることが重要である。



システムＤＲ２チェックリスト（設計認定会議）

システム名称：	実施日： 年 月 日（ ）

レビュー項目 【対象ドキュメント】	チェックポイント	判定 ※1 ○△×－
1．要件定義 【システム要件定義書】 【システム化フロー図】 【機能階層図】	1) 業務要求は、短期的な視点ではなく、中長期的視点も考慮されている。 2) ITシステム化の前提となる業務改革の姿が描かれている。 3) ITシステム・IT基盤整備の全体像・対象範囲が具体的に描かれている。 4) システムオーナー、業務キーパーソンなどのビジネス要件を十分に配慮し、要求仕様のレビューが行われている。	
2．システムアーキテクチャ設計 【システムアーキテクチャ設計書】	1) 全てのシステム要件がH/W、N/WおよびS/Wの構成に割り当てられている。 2) 実現・実装可能なH/W、N/WおよびS/Wの構成となっている。 3) システム稼動後の保守・運用フェーズでの効率性・将来性・体制を考慮した合理的なシステム構成となっている。 4) 本番稼動後、リソース不足にならないよう適切なサイジングがされている。 5) 仕様として設定された要求の実現可能性に問題がない構成となっている。	

～ 省略 ～

総合判定	総合的に判定して認定できるか （○：認定、△：条件付き認定、×：認定不可） レビュー項目に×がある場合、総合判定は×認定不可とする	

特記事項

※1 ○：問題なし / △：一部問題あり、コメント記入要 / ×：不可 / －：対象

認定	合議
合意形成結果の証憑	

図 5.12 システム DR（デザインレビュー）プロセスとチェックリスト

(4) ステアリングコミッティの設置

プロジェクト当事者では解決できない事項やステークホルダ間のコンフリクトにより意思決定に時間がかかる場合がある。そのような場合は、プロジェクト外の意志決定機関として、ステアリングコミッティの設置を行う（図 5.13）。

ステアリングコミッティには、プロジェクト推進の主体者である業務部門責任者、システム部門責任者、プロジェクト推進責任者のみならず、システムオーナ、経営層といったプロジェクト上位関係者も交えた体制とし、有事の際のエスカレーションパスを明確にする。

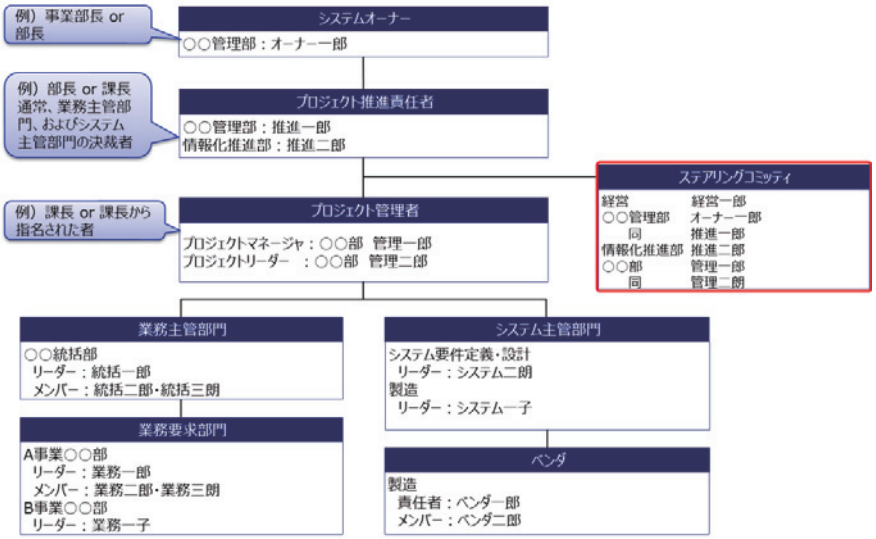


図 5.13 ステアリングコミッティの設置

以上のように、本稿「ステークホルダに合わせた合意形成」では、主に業務部門、システム部門、ステアリングコミッティの各ステークホルダとの合意形成の方法を例示した。

おわりに

昨今のシステム構築は、システム化対象業務が広範囲にわたり、ステークホルダも多様化している。そのような多様化したステークホルダといかに合意形成を図るかについて、ステークホルダの特定と分析、およびステークホルダに合わせた合意形成を切り口に、セイコーエプソンの事例を紹介した。

参考文献

[1] Project Management Institute：プロジェクトマネジメント知識体系ガイド(PMBOK ガイド)第 5 版，2014

[2] 一般社団法人情報サービス産業協会(JISA)：要求工学実践ガイド，近代科学社，2014

5.6 「要求の定量化による合意形成と膨らむ要求の制御」の事例
株式会社ジャステック

取り組み背景

一般的に要件定義工程の見積もりは、要件未確定を理由に、「要件定義工程の成果物を作成できる要員が何人ほど必要で、いつまでに作るから何人月です」といった感覚的な見積もりが見受けられる。その結果、期限切れによる要件積み残しの発生および過度の執行による取り違いなどにより、後工程での当該工程の成果物の誤り、漏れが発覚し、手戻りコストの増加などに繋がる原因の一つになっている。さらに、見積もりと出来高予実管理がカウンタブルな指標値として連動していないなどにより、膨らむ要求の制御が後手に回るケースも見受けられる。

こうした弊害を是正すべく、弊社では要求の定量化を行うために要件定義工程の成果物量（本稿では生産物量という）を代用指標として、当該工程に定量的プロジェクト管理を導入している。

本稿では以下に示す弊社の取り組み事例を紹介する。

- 要件定義工程に定量的プロジェクト管理（見積もり含む）を適用するための前提条件、非機能要求などを吸収した生産物量と生産性に基づく見積り方式および生産物量変換係数による開発全体の概算見積もりを通して醸成される合意形成についての事例を紹介する。
- 見積もりと出来高予実管理の指標値が連動した妥当値と成行値を使用し、膨らむ要求の制御を行う事例を紹介する。

なお、本稿のいくつかの事例は、弊社の新販売管理業務システムの構築で作成したものである。

本編との関連

「3.2　膨らむ要求のコントロール」、「3.4　要件定義工程からの非機能要件定義」、「3.5 多様化するステークホルダとの合意形成」、「3.6　現行業務やシステムの把握」と関連している。

要求の定量化による合意形成

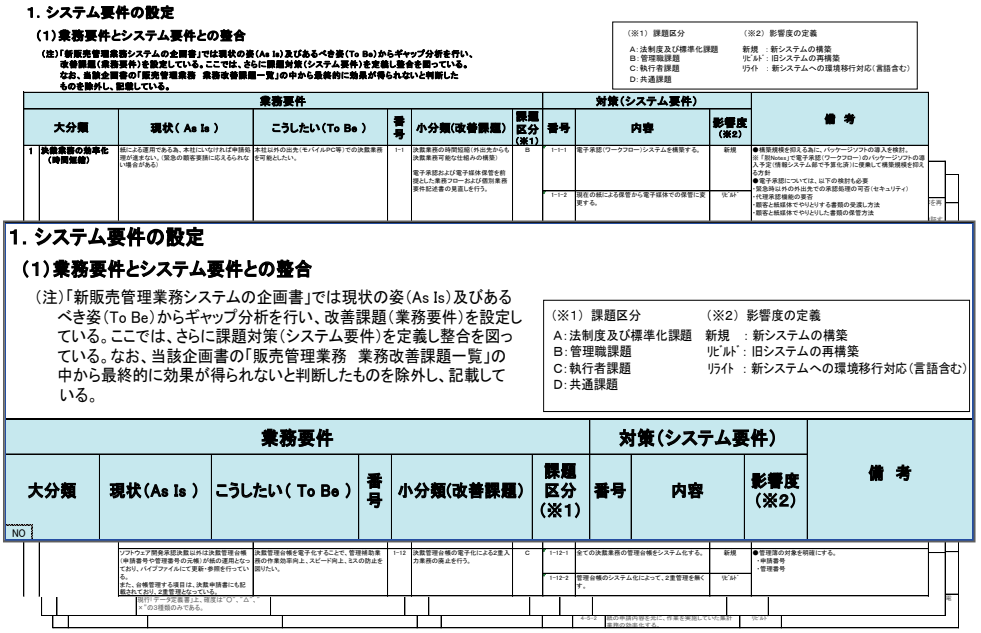
(1) 当該工程での定量的プロジェクト管理（見積もり含む）を適用するための前提条件

要件定義工程のインプットとしてシステム企画書（目的・施策、ROI、優先度、予算等）および組織標準（開発プロセス、生産物と記述水準、各種規約、体制、役割分担、責任等）などが存在し、ステークホルダ間で合意していることが前提である。

本稿では要件定義工程の見積もりを行うにあたり、特に、前提条件として弊社が留意「確認と合意」している事例を以下に紹介する。

① 業務要件とシステム要件との整合

一般的に要求（ビジネスの目的・施策）は階層構造をしているが、最上位の要求から見積もりを可能とする方式は弊社でも課題となっている。しかし、ビジネス目的を反映した業務要件の As Is（現状）→To Be（こうしたい）から改善課題を析出し、析出した当該課題対策の中に IT 施策としてのシステム要件が存在する。本稿の要件定義工程での見積り方式を適用するにあたり、少なくとも業務要件とシステム要件の整合がドキュメント化され合意が得られていることが前提である。以下にそのドキュメント事例を紹介する。



1. システム要件の設定

（1）業務要件とシステム要件との整合

（注）「新販売管理業務システムの企画書」では現状の姿（As Is）及びあるべき姿（To Be）からギャップ分析を行い、改善課題（業務要件）を設定している。ここでは、さらに課題対策（システム要件）を定義し整合を図っている。なお、当該企画書の「販売管理業務　業務改善課題一覧」の中から最終的に効果が得られないと判断したものを除外し、記載している。

（※1）課題区分
A：法制度及び標準化課題
B：管理職課題
C：執行者課題
D：共通課題

（※2）影響度の定義
新規　：新システムの構築
ﾘﾋﾞﾙﾄﾞ：旧システムの再構築
ﾘﾗｲﾄ　：新システムへの環境移行対応（言語含む）

業務要件						対策（システム要件）			備考
大分類	現状（As Is）	こうしたい（To Be）	番号	小分類（改善課題）	課題区分（※1）	番号	内容	影響度（※2）	
1 決裁業務の効率化（時間短縮）	紙による運用である為、本社にいなければ申請処理が進まない。（緊急の顧客要請に応えられない場合がある）	本社以外の出先（モバイルPC等）での決裁業務を可能としたい。	1-1	決裁業務の時間短縮（外出先からも決裁業務可能な仕組みの構築） 電子承認および電子媒体保管を前提とした業務フローおよび個別業務要件記述書の見直しを行う。	B	1-1-1	電子承認（ワークフロー）システムを構築する。	新規	●構築規模を抑える為に、パッケージソフトの導入を検討。 ※「Notes」で電子承認（ワークフロー）のパッケージソフトの導入予定（情報システム部）で検討して構築規模を抑える方針 ●電子承認については、以下の検討も必要 ・緊急時以外の外出先での承認処理の可否（セキュリティ） ・代理承認機能の要否 ・顧客と紙媒体でやりとりする書類の受渡し方法 ・顧客と紙媒体でやりとりした書類の保管方法
						1-1-2	現在の紙による保管から電子媒体での保管に変更する。	ﾘﾗｲﾄ	

1. システム要件の設定

（1）業務要件とシステム要件との整合

（注）「新販売管理業務システムの企画書」では現状の姿（As Is）及びあるべき姿（To Be）からギャップ分析を行い、改善課題（業務要件）を設定している。ここでは、さらに課題対策（システム要件）を定義し整合を図っている。なお、当該企画書の「販売管理業務　業務改善課題一覧」の中から最終的に効果が得られないと判断したものを除外し、記載している。

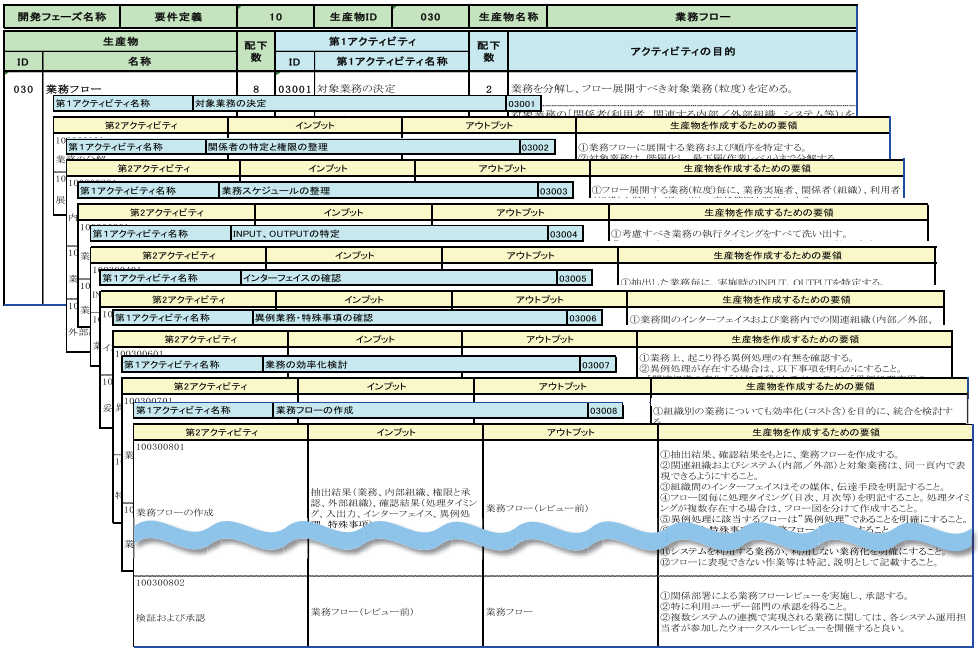
（※1）課題区分
A：法制度及び標準化課題
B：管理職課題
C：執行者課題
D：共通課題

（※2）影響度の定義
新規　：新システムの構築
ﾘﾋﾞﾙﾄﾞ：旧システムの再構築
ﾘﾗｲﾄ　：新システムへの環境移行対応（言語含む）

業務要件						対策（システム要件）			備考
大分類	現状（As Is）	こうしたい（To Be）	番号	小分類（改善課題）	課題区分（※1）	番号	内容	影響度（※2）	
NO									
	ソフトウェア開発承認決裁以外は決裁管理台帳（申請書号や管理番号の台帳）が紙の運用となっており、バインダファイルにて更新・参照を行っている。 また、台帳管理する項目は、決裁申請書にも記載されており、2重管理となっている。 （執行者：予定着工と、確度は「○」、「△」、「×」の3種類のみである。	決裁管理台帳を電子化することで、管理補助業務の作業効率向上、スピード向上、ミスの防止を図りたい。	1-12	決裁管理台帳の電子化による2重入力業務の廃止を行う。	C	1-12-1	全ての決裁業務の管理台帳をシステム化する。	新規	●管理項目の対象を明確にする。 ・申請番号 ・管理番号
						1-12-2	管理台帳のシステム化によって、2重管理を無くす。	ﾘﾗｲﾄ	

② 要件定義工程の生産物および記述水準

弊社の見積りモデルは生産物量と生産性が基準指標となるため、要件定義工程（各開発工程含め）の生産物にどのようなドキュメントを適用し、どのような記述水準にするかを合意することが前提である。以下に生産物ごとの第 1、第 2 アクティビティおよび生産物の作成要領を定義している標準化事例の一部（業務フロー）を紹介する。



③ 非機能要求の要求レベル

弊社の非機能要件定義は見積り方式の‘環境変数’の一部に相当し、環境変数には‘品質特性’と‘環境特性’があり、両特性の因子ごとに要求レベルを 5 段階評価している。

環境変数は後述の(2)②および「本稿末項」に記載した‘添付別紙 1(1)’に示す。

見積もりの前提として、環境変数のインプットになる非機能要求確認書（非機能要件設定）を作成している。以下に事例を示す。

(3)非機能要件設定

(3)－1. 非機能要件設定（環境変数「共通版」新規開発、改造型開発、再構築開発」）

非機能要件に関しては、当社の環境変数因子ごとに要件を設定する。
当該要件は非機能要件仕様及び見積もりの環境変数（生産物量、生産性）の入力とする。
（非機能要件仕様は2.（3）、見積もりは、「4.（3）費用見積り」参照）

新販売管理業務システム「V1版」

環境変数 品質特性			要件（内容）
機能性	合目的性（含む環境配慮機能数）	①	利用者の広がり⇒段階処理本部の一部が利用することとなるが、影響は軽微
		②	ステークフォルダーの増加⇒現行と同じ
		③	コンティンジェンシーに対する機能要求⇒現行と同じ
	正確性	①	ジャスチック標準「検証」「レビュー時間」、「テスト項目数」に準拠する。
	時機性	①	他システム（「会計システム」「構築管理システム」）とのインターフェースは、現行と同じ
		②	ワークフロー（市販パッケージ）とのインターフェースは、新たにコード及びフォーマット変換を検討し、非機能要件仕様として定義する。
	セキュリティ	①	ジャスチック標準「情報セキュリティ開発・保守基準」に準拠する。
信頼性	成熟性	①	現行システムと同等水準とする。
	障害許容性	①	ジャスチック標準「開発・保守に関わる信頼性基準」に準拠する。
	回復性	①	ジャスチック標準「開発・保守に関わる信頼性基準」に準拠する。
使用性	理解性	①	新たに、営業本部にてマニュアル（標準準拠）を作成するが、開発発案とは別に作成する。
	習得性	①	同上
	操作性	①	現行システムと同等水準とする。
効率性	実行効率性	①	先行開発（第1フェーズ「対外提出決裁等」、第2フェーズ「売上請求開始等」）にて試行検証を行い、パフォーマンスが厳格な第3フェーズに反映する。
	資源効率性	①	最大同時利用人数を配慮し、CPU仕様を情報システム部の協力を得て決定する。
保守性	解析性	①	ジャスチック標準「開発・保守に関わる信頼性基準」に準拠する。
	変更作業性	①	新たな保守用ドキュメントは作成しない。（当該開発生産物をもって、変更作業を行う。）

環境変数 環境特性			要件（内容）
業務特性	業務ナレッジ	①	現行業務を担当する営業員（販売課員含む）が参加する。
		②	旧システムを作成した営業員が一部参加する。
		③	①、②を前提に、企画、要件定義、基本設計（一部パッケージ設計含む）、テストを営業本部が行う。
HW特性	安定度／信頼度使用実績	①	当社で使用実績のあるハードウェア（OS含む）を利用する。
SW特性	安定度／信頼度使用実績	①	市販パッケージは、当社で使用実績がほとんどないものを利用することになる。（但し、プロトタイピングにより、当該生産性悪化を補う。（後述の「開発手法／開発環境」参照））
コミュニケーション特性	顧客窓口特性	①	開発（製造工程）は情報システム部にお願いするが、上位工程作成者（営業本部）を窓口とする。
	工期の厳しさ	①	標準工期＝2．5×（380人月$^{(1/3)}$）＝18ヶ月だが、開発工期は24ヶ月（7（2）「マスタースケジュール」参照）とする。
	コミュニケーション基盤	①	開発（製造工程）は情報システム部にお願いするが、上位工程作成者（営業本部）が全面的に支援する。
	レビュー体制	①	上位工程（企画、要件定義、基本設計（一部パッケージ設計含む）及びテストは、システムと業務を熟知している営業本部が行う。
		②	開発工程（製造工程）は情報システム部にお願いするが、システムと業務を熟知している営業本部がレビューする。
	業務／他システム資料現行・標準化版	①	現行販売管理システムの設計書については、一部不足しているとともに、最新状態ではない。
		②	一方、設計においては、既存資料、設計書を流用し、活用する。
開発環境特性	開発手法／開発環境	①	プロトタイピング手法により、以下の生産性向上を図り、SW特性での生産性悪化を補う。
		②	後工程の手戻り抑制「先に問題点を顕在化する。」
		③	習熟度を高める。
	テスト要領書水準	①	ジャスチック標準「テスト」「記述項目」、「記述水準」」を準拠し作成する。

④ 現行（現行業務と既存システム）の可視化

要件定義工程での生産物量と生産性の見積もりを行う上で、現行を調査し把握することは重要である。弊社の環境変数‘再構築特性’には、現行の可視化「業務の可視化度合、システムの可視化度合‘データ含む’、トレーサビリティ」の三つの副特性が存在し、因子ごとに具備状態を 5 段階評価している。見積もりの前提としては、環境変数のインプットになる上記の三つの副特性以外の副特性を含めて確認書を作成している。

環境変数は後述の(2)②および「本稿末項」に記載した‘添付別紙 1(2)’に示す。

なお、5 段階評価結果でのレベルに応じた生産物の復元量は、開発時の生産性との兼ね合いで合意を得る必要がある。この場合の復元作業は要件定義工程での付帯作業として見積もっている。

a）開発生産性との相互関係を勘案した現行の可視化

現行の可視化度合いによる現行生産物の復元量は、現行の可視化に関係する生産性環境変数（現行ドキュメントの品質「解析性」、既存システムの錬度「業務、システム」、ツール具備「調査」）の評価レベルを勘案する。また、他の生産物量環境変数と生産性環境変数との相関関係（例：棚卸しと移植性など）を含め、最適なアルゴリズムが存在する。ここでは紙面の都合上、アルゴリズムの紹介は省略する。

■ 現行の可視化に関係する生産物量環境変数*1

主特性	副特性	基準生産物量からの変動率(%) 要件定義	設計	製作	テスト
現行（業務＆システム）の可視化	業務の可視化度合	0～35	–	–	–
	システムの可視化度合（データ含む）	0～30	0～38	0～15	0～10
	トレーサビリティ	0～5	0～10	0～15	–

*1:付帯作業としての復元生産物量を対象

■ 現行の可視化に関係する生産性環境変数*2

主特性	副特性	基準生産性からの変動率(%) 要件定義	設計	製作	テスト
現行ドキュメント等の品質	解析性	0～80	0～80	0～30	–
現行の錬度	既存業務	0～150	0～60	–	0～70
	既存システム（旧新アーキテクチャ差異含む）	0～125	0～170	0～130	0～100
ツールの具備度合	既存システムの調査ツール	0～15	0～20	0～20	–

*2:付帯作業(復元生産性)と当該開発工程作業(開発生産性)を対象

b）見積もりを行う上での必要な現行（要件定義工程の生産物）の可視化

現行の全ての生産物が可視化されていることが望ましいが、実態は全ての生産物が品質の保証含め可視化されていることは稀である。そこで、弊社では見積もりを行う上で、以下に記載した三つの現行の生産物を、現行の可視化に関係する生産性環境変数との兼ね合いを勘案し、可視化（復元）することの合意を得るようにしている。

- 業務フロー（現行業務の流れ‘組織、手段、手順’の把握）
- ER 図（現行の‘データ、業務ルールなど’の管理過程の把握）
- 個別業務処理定義書（現行各業務機能と現行システム機能との対応付け）

なお、前述の(1)①「新業務要件と新システム要件の整合表」が存在していること、また、実態上において、画面／帳票レイアウト、データ項目（棚卸しを留意）は存在するので利用可能である。よって、可視化対象から外している。

(2) 生産物量と生産性に基づく見積りモデル

① 要件定義工程における見積もりの基本アルゴリズム（新規開発見積り方式）

要件定義工程(i=1)での、あるアクティビティ（生産物）’K’のコスト C_k は、生産物量を V_k、生産性を P_k、基準生産物量を V^B_k および基準生産性を P^B_k で表現すると、次式で求まる。

$$C_k = V_k \times P_k = V^B_k(1+a_i) \times P^B_k(1+b_i)$$

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

添字(i,j,k)
i:工程
j:環境変数因子
k:工程生産物

（注）KC：K .Character　H：時間　C_k：円 or 工数　V_k：KC　P_k：円/KC or H/KC　KC は Page 変換有

要件定義工程(i=1)のコストは当該工程で作成する各生産物に対応するコストの総和である。よって、コスト C_1 は次式で求まる。

$$C_1 = \Sigma C_k$$

② 非機能要求を定量化した環境変数

■a_i 、b_i は V^B_i および P^B_i に対して品質要求の多寡や開発環境の違い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。(a_i は生産物量環境変数、b_i は生産性環境変数)

■a_i 、b_i は‘品質特性’と‘環境特性’から影響される独立した変動要素（α_{ij}、β_{ij}）から構成している。$A_i(\Sigma\alpha_{ij})$と $b_i(\Sigma\beta_{ij})$は‘添付別紙１(1)’の共通編（新規開発、改造型開発、再構築開発）を参照のこと。なお、改造型開発と再構築開発は共通編に加え‘改造型特性’と‘再構築特性（新規開発と改造型開発の一部を共有)’がある。‘添付別紙１ (2)’の固有編を参照のこと。

③ その他の見積り方式について

a) 改造型開発見積り方式

CP(Cost of Positional Analysis)、**CN**(Cost of Net)、**CR**(Cost of Regression)

$$C_i = CP_i + CN_i + CR_i$$

$$CP_i = VP_i \times PP^B_i(1+b_i+bP_i)$$
$$CN_i = VN^B_i(1+a_i) \times P^B_i(1+b_i+bN_i)$$
$$CR_i = VR^B_i(1+a_i) \times P^B_i(1+b_i+bR_i)$$

改造型固有の環境変数「$bP_i: (\Sigma\beta P_{ij})$ $bN_i: (\Sigma\beta N_{ij})$ $bR_i: (\Sigma\beta R_{ij})$」

b) 再構築開発見積り方式

CD(Cost of Differential Analysis)、**CRe**(Cost of Re-build)

$$C_i = CD_i + CRe_i$$

$$CD_i = VD_i \times PD^B_i(1+b_i+bD_i)$$
$$CRe_i = VRe^B_i(1+a_i+aRe_i) \times P^B_i(1+b_i+bRe_i)$$

再構築固有の環境変数「$bD_i: (\Sigma\beta D_{ij})$ $aRe_i: (\Sigma\alpha Re_{ij})$ $bRe_i: (\Sigma\beta Re_{ij})$」

参考事例 1）要件定義工程の生産物量と生産性の見積もりの事例

工程	生産物	分担	生産物量（KC）						
			妥当量	基準生産物量	生産物量環境変数		分担率	許容要求変更量	許容要求変更率
						（率）			
				V	$V*a_1$	a_1	RB	$(V*(1+a_1))*\omega$	ω
要件定義	ビジネス関連図	営業本部	16.459 KC	14.250 KC	0.713 KC	5.0 %	95.0%	1.496 KC	10.0 %
		システム部	0.867 KC	0.751 KC	0.038 KC	5.0 %	5.0%	0.079 KC	10.0 %
		小計	17.326 KC	15.000 KC	0.750 KC	5.0 %		1.575 KC	10.0 %
	業務一覧	営業本部	28.678 KC	23.750 KC	1.188 KC	5.0 %	95.0%	3.741 KC	15.0 %
		システム部	1.509 KC	1.250 KC	0.063 KC	5.0 %	5.0%	0.197 KC	15.0 %
		小計	30.188 KC	25.000 KC	1.250 KC	5.0 %		3.938 KC	15.0 %
	業務フロー	営業本部	45.885 KC	38.000 KC	1.900 KC	5.0 %	95.0%	5.985 KC	15.0 %
		システム部	2.415 KC	2.000 KC	0.100 KC	5.0 %	5.0%	0.315 KC	15.0 %
		小計	48.300 KC	40.000 KC	2.000 KC	5.0 %		6.300 KC	15.0 %
	概念レベルのER図	営業本部	54.079 KC	42.750 KC	4.275 KC	10.0 %	95.0%	7.054 KC	15.0 %
		システム部	2.846 KC	2.250 KC	0.225 KC	10.0 %	5.0%	0.371 KC	15.0 %
		小計	56.925 KC	45.000 KC	4.500 KC	10.0 %		7.425 KC	15.0 %
	個別業務処理定義書	営業本部	53.130 KC	44.000 KC	2.200 KC	5.0 %	80.0%	6.930 KC	15.0 %
		システム部	13.283 KC	11.000 KC	0.550 KC	5.0 %	20.0%	1.733 KC	15.0 %
		小計	66.413 KC	55.000 KC	2.750 KC	5.0 %		8.663 KC	15.0 %
	画面/帳票レイアウト（画面変遷を含む）	営業本部	66.853 KC	55.250 KC	5.525 KC	10.0 %	65.0%	6.078 KC	10.0 %
		システム部	35.998 KC	29.750 KC	2.975 KC	10.0 %	35.0%	3.273 KC	10.0 %
		小計	102.850 KC	85.000 KC	8.500 KC	10.0 %		9.350 KC	10.0 %
	：	：	：	：	：	：	：	：	：
要件定義工程計		営業本部	370.916 KC	301.000 KC	24.080 KC	8.0 %	86.0%	45.836 KC	14.1 %
		システム部	56.991 KC	49.000 KC	3.430 KC	7.0 %	14.0%	4.561 KC	8.7 %
		合計	427.908 KC	350.000 KC	27.510 KC	7.9 %		50.398 KC	13.4 %

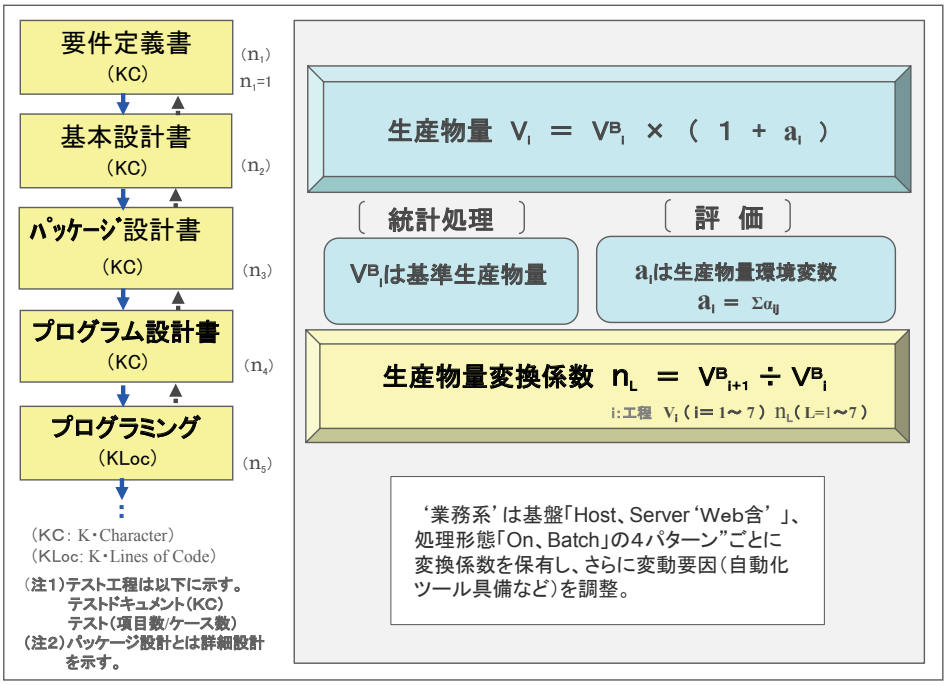
生産性（円／KC）
境変数
（率）
bi
5.0 %
5.0 %
5.0 %
8.0 %
8.0 %
8.0 %
8.0 %
8.0 %
8.0 %
8.0 %
8.0 %
8.0 %
5.0 %
5.0 %
5.0 %
0.0 %
0.0 %
0.0 %
：
7.2 %
5.8 %
6.5 %

（注）妥当量、妥当生産性および許容要求変更量‘率’は後述の「膨らむ要求の制御」(1)を参照

なお、見積もりをお客様と合意するために、お客様の標準見積り指標（FP、機能数等）に準じる場合がある。その場合は弊社見積もり基準指標値（生産物、生産性）から変換する方式を用意している。

(3) 生産物量変換係数による開発全体の概算見積もり

要件定義工程の生産物量から各工程間の生産物量変換係数を使用して、開発全体の概算見積もりを算出している。なお、弊社では要件定義工程が完了していない見積もりを概算見積もりと呼んでいる。以下に各工程間の生産物量変換係数の概要を図 5.14 に示す。さらに、生産物量変換係数を使用して、各工程の概算見積もりを算出した事例を表 5.8 に示す。



(注) 変換係数に使用する要件定義書はシステム要件定義書を対象

図 5.14　生産物量変換係数の概要説明

表 5.8　生産物量変換係数を使用した開発全体の概算見積もり事例

工程生産物	生産物量変換係数 (n_L)	(量×量単価「円/生産物量」) 当該事例は要件定義書'1KC'とした場合						
		要件定義	基本設計	パッケージ設計	プログラム設計	プログラミング	統合テスト	システムテスト
要件定義書 (Kc)	$n_1=1$	¥?0,000	¥?0,000	¥?0,000	¥?0,000	¥?0,000	¥?0,000	¥?0,000
基本設計書(Kc)	$n_2=1.44$	¥0	¥?04,244	¥?04,244	¥?04,244	¥?04,244	¥?04,244	¥?04,244
パッケージ仕様書(Kc)	$n_3=7.60$	¥0	¥0	¥?87,969	¥?87,969	¥?87,969	¥?87,969	¥?87,969
プログラム設計書(Kc)	$n_4=9.53$	¥0	¥0	¥0	¥?8,131	¥?8,131	¥?8,131	¥?8,131
プログラミング(KLoc)	$n_5=0.93$	¥0	¥0	¥0	¥0	¥?36,215	¥?36,215	¥?36,215
統合テスト仕様書(テスト項目)	$n_6=65.55$	¥0	¥0	¥0	¥0	¥0	¥?72,184	¥?72,184
システムテスト仕様書(テスト項目)	$n_7=18.69$	¥0	¥0	¥0	¥0	¥0	¥0	¥?3,411
合 計		¥?0,000	¥?84,244	¥?72,213	¥?10,344	¥?46,559	¥?18,743	¥?002,154

(注) 基盤'Server'で処理形態'0nline'のパターンを適用(見積もり値の一部をマスク)

膨らむ要求の制御

(1) 要件定義工程の妥当値と成行値による出来高予実管理

進捗度（出来高把握）は原価比例法やEVM(Earned Value Management)を使用した方法が一般的である。当社では見積もりと出来高（実績+残予測）を連動させ、精度を高めるために、生産物量（生産性含む）の妥当値*3と成行値*4の考え方を取り入れて実践している。なお、EVMを使用する場合は、PV(Planned Value)とEV(Earned Value)は妥当値を、AC(Actual Cost)は成行値を適用する。

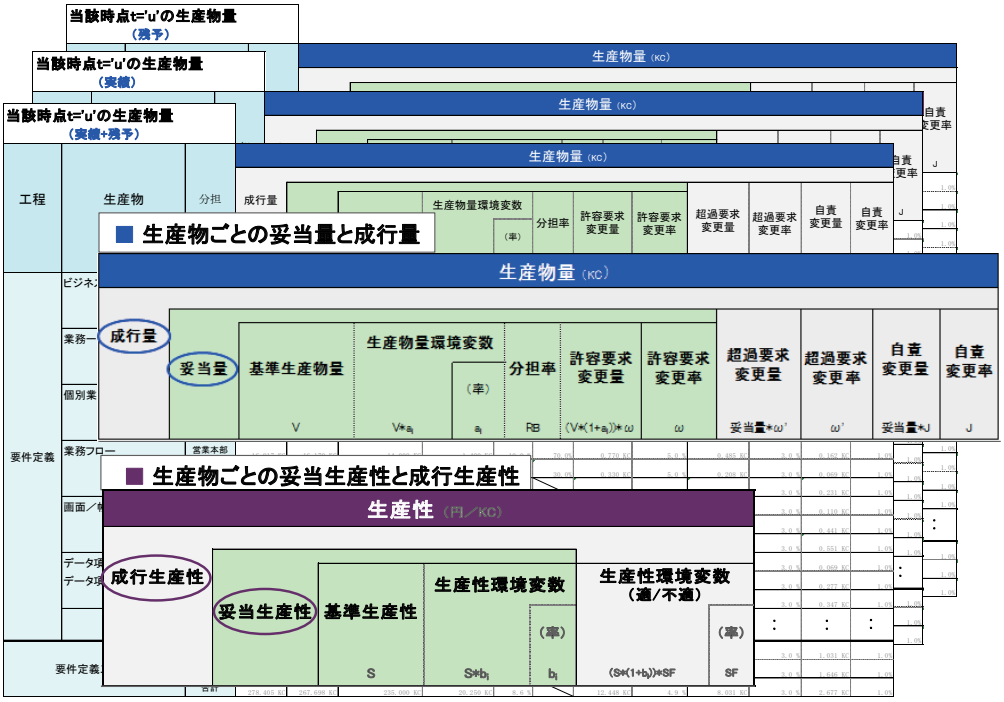
＊3：妥当量（基準生産物量、生産物量環境変数、分担率、許容要求変更量‘率’）と妥当生産性（基準生産性、生産性環境変数）から構成している。お客様への見積もりは妥当値を適用する。妥当生産性の基準生産性は最速な生産性を指す。

＊4：成行量（妥当量に超過要求変更量‘率’と自責変更量‘率’を加算）と成行生産性（妥当生産性に自責生産性‘適/不適’を加算）から構成している。

なお、弊社の標準では「仕様変更量‘率’＝　許容要求変更量‘率’+　超過要求変更量‘率’」として妥当量に仕様変更量‘率’を組み入れている。新販売管理業務システムの構築では、要件定義工程における膨らむ要求を制御するために、許容要求変更量‘率’（妥当量）と超過要求変更量‘率’（成行量）とに分けて設定している。

表5.9は生産物量（妥当量、成行量）と生産性（妥当生産性、成行生産性）の項目および当該時点t=’u’（実績、残予測、実績+残予測）での生産物量の出来高管理表を示す。

表5.9　要件定義工程での生産物ごとの妥当値と成行値



(2) 出来高の予実差異分析による膨らむ要求の制御

妥当量の許容要求変更量は、要件定義工程（生産物量）の完了時に許容できる要求の変更量であり、成行量の超過要求変更量は許容要求変更量を超過した変更量である。

表 5.9 にある（当該時点 t=’u’「実績、残予測、実績+残予測」）は生産物量を対象にした事例である（出来高をコストとする場合は生産物量×生産性「円/量」になる）。

図 5.15 は当該時点 t=’u’での出来高の予実差異分析（妥当値、成行値）から超過要求変更量を検知し、代替案を含む改善提案と調整により過剰な要求量を制御した事例である。

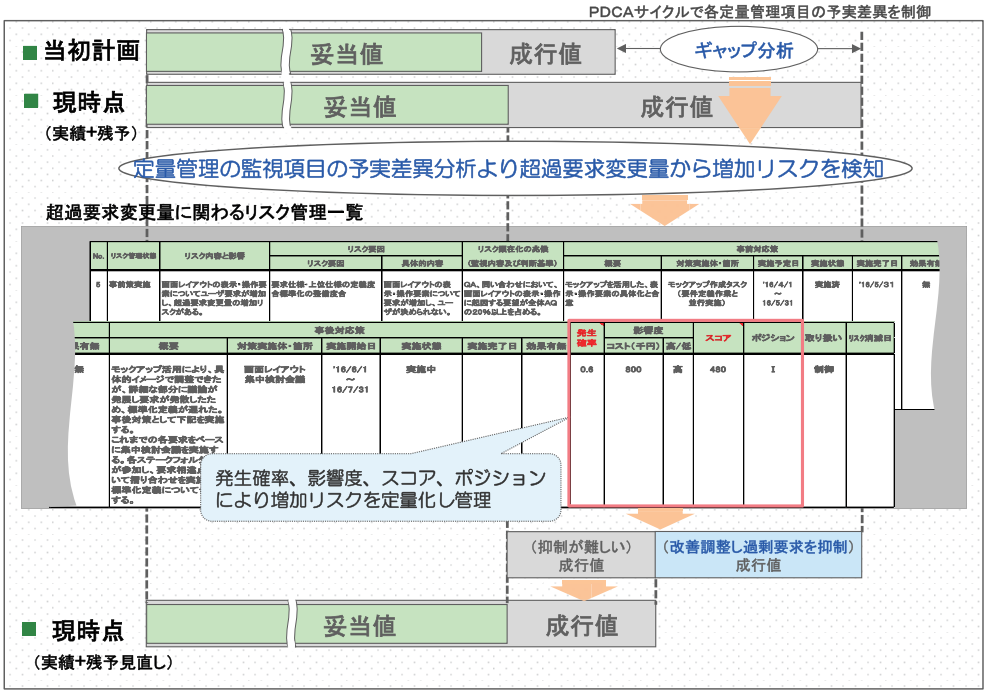


図 5.15　出来高予実分析による膨らむ要求の制御事例

■ まとめ

本稿で紹介した見積りモデルと出来高予実管理方式は、基準指標値（生産物量、生産性）が前提になっている。繰り返しになるが、見積もりは見積もりのためだけでなく、見積もりを‘予’とした予実管理を可能にするためでもある。当該予実管理の事例として、「膨らむ要求の制御」を紹介したが、定量的プロジェクト管理の真髄である各指標値を精錬させ、変動幅の小さいベースラインを確立させることが重要である。

但し、弊社のモデルと方式は他に比べ面倒であるといった側面もある。しかし、日本人は製品への緻密な取り組みや既成概念に捉われない品質へのこだわりが、世界に伍した製品を生み出してきた。グローバル化が進展する状況下、ソフトウェア業界もグローバライズされて行くのは自然であるが、一方で、その国民性を生かした取り組み姿勢を忘れてはならない、という思いがある。

添付別紙１　環境変数の概要説明表

(1)　共通編

① 生産物量環境変数（新規開発、改造型開発、再構築開発）

対象見積りモデル：新規開発(C)、改造開発型(CP、CN、CR)、再構築開発(CD、CRe)

特性タイプ	主特性（対象コストモデル）	副特性	評価の観点（概略内容）	基準生産物量からの変動率（%） 要件定義	設計	製作	テスト
品質特性	使用性	理解性					
		習得性					
		操作性					
	(C、CN、CRe)						
	保守性	解析性					
		変更性					
		試験性					
	(C、CN、CRe)						

対象見積りモデル：新規開発(C)、改造開発型(CP、CN、CR)、再構築開発(CD、CRe)

特性タイプ	主特性（対象コストモデル）	副特性	評価の観点（概略内容）	基準生産物量からの変動率（%） 要件定義	設計	製作	テスト
品質特性	機能性	合目的性	利用者/利害関係者の広がり、コンテンジェンシー対応、不正移行データ対応等の該当事象数	0～50	0～50	0～50	0～50
		正確性	正確性(検証)に関わる標準テスト密度を基準にしたテスト項目量への要求水準	—	—	0～20	0～50
		接続性	他システムとの接続によるコード変換、フォーマット変換数	0～5	0～20	0～20	0～20
		セキュリティ	対応が必要なセキュリティ実現機能数、但し機能要件に定義されている部分は除く	0～20	0～20	0～20	0～20
	(C、CN、CRe)						
	信頼性	成熟性	故障低減に必要な実現機能数	0～5	0～10	0～10	0～10
		障害許容性	異常検知に必要な機能数	0～5	0～10	0～10	0～10
		回復性	再開処理に必要な実現機能数	0～5	0～10	0～10	0～10
	(C、CN、CRe)						

② 生産性環境変数（新規開発、改造型開発、再構築開発）

対象見積りモデル：新規開発(C)、改造開発型(CP、CN、CR)、再構築開発(CD、CRe)

特性タイプ	主特性（対象コストモデル）	副特性	評価の観点（概略内容）	基準生産性からの変動率（%） 要件定義	設計	製作	テスト
環境特性	業務特性	業務					
	(C、CP、CN、CR、CD、CRe)						
	ハードウェア特性	安定使用					
	(C、CP、CN、CR、CD、CRe)						
	ソフトウェア特性	安定使用					
	(C、CP、CN、CR、CD、CRe)						
	コミュニケーション特性	顧客					
		工期					
		コミ基盤					
		レビ					
	(C、CP、CN、CR、CD、CRe)						
	開発環境特性	業務規約					
		開発					
		テス					
	(C、CP、CN、CR、CD、CRe)						

対象見積りモデル：新規開発(C)、改造開発型(CP、CN、CR)、再構築開発(CD、CRe)

特性タイプ	主特性（対象コストモデル）	副特性	評価の観点（概略内容）	基準生産性からの変動率（%） 要件定義	設計	製作	テスト
品質特性	機能性	合目的性（要求仕様の網羅性）	要求の記述水準および網羅性。要求定義については新規性、方針明確性、ステークフォルダの多様性等を考慮	0～100	0～30	—	0～10
		正確性	正確性(検証)に関わる標準レビュー工数(各工程8%)を基準にした要求水準	0～5	0～5	0～3	0～5
		接続性	基準単位(100KS)に対する社内／社外システムとのインターフェス先の数	0～20	0～10	—	0～10
		整合性	整合をとる社内／社外の規格・基準の数、全体適合性やグローバル化対応も含む	0～15	0～15	0～8	0～8
	(C、CN、CRe)						
	効率性	実行効率性	実行効率に対する一般的要求水準(既知)の最適事例を基準にした要求水準	0～5	0～10	0～5	0～10
		資源効率性	資源効率に対する一般的要求水準(既知)の最適事例を基準にした要求水準	0～5	0～10	0～5	0～10
	(C、CN、CRe)						
	保守性	解析性	ソースコードの解析性をコード化規約に定めるコメントに対する要求水準から評価	—	—	0～5	—
		安定性	ソフトウェア変更に対しシステム品質維持可能とする水準をライフサイクル目標年数の長さにより評価	0～8	0～10	0～8	—
	(C、CN、CRe)						
	移植性	環境運用性 移植作業性 規格準拠性 置換製	ソフトウェアをどの程度、多様な環境に移すことができるかに対する要求の水準	0～28	0～28	0～12	0～25
	(C、CN、CRe)						

（注）移行、教育、保守、運用作業は生産性環境変数の適用対象から除いている

(2) 固有編

① 生産物量環境変数（再構築開発）

対象見積りモデル：再構築（再構築開発「CD、CRe」）

特性タイプ	主特性（対象コストモデル）	副特性	評価の観点（概略内容）	基準生産物量からの変動率（%） 要件定義	設計	製作	テスト
再構築特性	現行（業務＆システム）の可視化 *1 （CD、CRe、C1、CP）	業務の可視化度合 *1	・現行業務の目的・施策および業務内容を含めた全体の現行業務生産物（業務フロー等）の具備状態を評価	0～35	－	－	－
		システムの可視化度合 *1（データ含む）	・現行システムのハードウェア、ソフトウェア、主要DB（ファイル）、ネットワークの各構成および関連を明確にした、現行システム全体俯瞰資料の具備状態を評価、 ・現行システムおよびアプリケーションの構造を明確にした、シーケンス図、モジュール構成図、クラス図などの具備状態を評価。 ・データの目的およびデータ間の関連を明確にした、業務データ関連図、概念データモデルなどの具備状態を評価	0～30	0～38	0～15	0～10
		トレーサビリティ *1	・業務要件とシステム要件、システム要件と基本設計（外部設計生産物）、外部設計生産物とパッケージ設計生産物、パッケージ設計生産物とプログラムの紐付けに関する整備度合いを評価	0～5	0～10	0～5	－
	移 行 （CRe）	移行ツール	・移行ツール機能（'プログラムand/orデータ'、検証用コンペア等）の具備要求度合いを評価	0～5	0～10	0～20	0～15
		棚卸し *1	・移行元リソースの品質向上を鑑みた既存システムの棚卸し（プログラムのスリム化、データクレンジングなど）作業の要求度合いを評価	0～2	0～5	0～20	0～10
		移行パターン	・旧新（新旧）とリソース「データ、プログラム、環境」の組合わせは、6パターン（論理上'8'）存在する。当該再構築がどのパターンに該当するかを判断し、移行にあたり「データ、環境 *1」がプログラムに影響（修正対応）する量を評価。 ＊1：ハードウェア、OS、DBMS、ネットワーク等の基盤系を含む	0～10	0～25	0～25	0～25
		追い付き差分取り込み	・再構築開発と並行保守開発している現行システムの並行保守開発期間（凍結タイミング）を鑑みた、差分取り込みの多寡を評価	0～5	0～10	0～20	0～20

＊1：'現行の可視化'と'棚卸し'は、付帯作業の生産物量である。よって、新規開発'C1'と改造型開発'CP'も対象になる。

② 生産性環境変数（改造型開発、再構築開発）

対象見積りモデル：改造開発型（母体調査分析「CP」、正味規模改造「CN」、テスト巻込み「CR」）

特性タイプ	主特性（対象コストモデル）	副特性	評価の観点（概略内容）	基準生産性からの変動率（%） 要件定義	設計	製作	テスト
改造型特性	既存母体調査（CP） 既存母体錬成（CP、CN、CR） 既存母体品質（CP、CN、CR） 既存テスト（CN、CR）						

（注）改造型開発（

対象見積りモデル：再構築（差異調査分析「CD」、再構築開発「CRe」）

特性タイプ	主特性（対象コストモデル）	副特性	評価の観点（概略内容）	基準生産性からの変動率（%） 要件定義	設計	製作	テスト
再構築特性	現行ドキュメント等の品質 *2（CD、CRe、C1）	移植性 *2	・移行元全リソース（データ等）の不備や不整合度合いなどを評価	0～5	0～20	0～80	0～50
		解析性 *2	・既存業務＆システム全体のドキュメント、ソースコード等、成果物の解析容易性を評価（保守状態'可視化度合も含む'）	0～80	0～80	0～30	－
		環境適応性 *2	・多様なハード、ソフト、運用環境への旧新適用度を評価	0～15	0～20	0～20	0～30
	現行の錬度（CD、CRe、C1）	現行業務 *2（既存システム運用含）	・既存業務（システム運用含む）有識者の関与度合「工程毎の有識者関与割合'標準との対比'」を評価	0～150	0～60	－	0～70
		既存システム（旧新アーキテクチャの差異を含む）	・調査分析経験（回数）と調査分析対象規模（既存システム全体との割合）を評価 ・既存システム開発者は影響度'0'で評価（但し、既存システム全体との割合を考慮） ・アーキテクチャ使用and/or開発経験（回数）を評価「旧新差異の取り込み時の生産性への影響」	0～100	0～170	0～130	0～100
	ツールの具備度合（CD、CR）	既存システムの調査ツール	・調査ツール機能（絞込み、モニタリング、リバース等）の数で評価	0～15	0～20	0～20	－
		移行ツール	・移行ツール機能（'プログラム言語、データタイプ'、コンペアツールなど）の数で評価	－	0～5	0～25	0～40

＊2：'現行業務に関係するドキュメントの品質'と'現行業務錬度'は、新規開発'C1'も対象になる。

5.7 「手戻りコストを抑制する要件定義書のレビュー」の事例 株式会社ジャステック

取り組み背景

要件定義工程での成果物の誤り、漏れ（積み残しを含む）などが、それ以降の開発工程で摘発され、当該摘発工程により手戻りコストは増減する。特に開発工程ではシステムテストでの摘発が最大になるが、具体的な数値で捉えられているだろうか。また、要件定義工程での成果物のレビューは、どのような方法が効果的なのかを考えられているだろうか。

本稿では、そのような疑問や課題をユーザ、システム部門（情報関連会社含む）およびベンダが認識を一致させ、知見を広める上での手助けになることを目的としている。

ソフトウェア製品の検証[*1]および妥当性確認[*2]は、各々、弊社のレビュー[*3]およびテストの両方に該当する。本稿では、特に要件定義工程のレビューを中心に据え、弊社のレビューに関する取り組み事例を紹介する。なお、以下にその概要を示す。

- 要件定義工程で混入した欠陥を当該工程のレビューで摘発した場合と以降の開発工程にて摘発した場合での手戻りコストに関わるROI（Return On Investment）を紹介する。
- レビューに関する振る舞いを定義したレビュープロセス成熟度および実チームごとのレビュープロセス成熟度調査から、効果的な要件定義工程のレビュー方法を紹介する。
- レビュープロセス成熟度と手戻りコストに関わるROIを使用し、要件定義工程のレビュープロセス成熟度向上による手戻りコストの改善事例を紹介する。

＊1：「仕様どおりであることの確認」を行うこと。
＊2：「ユーザの仕様意図が、十分に実現されていることの確認」を行うこと。
＊3：レビューは各開発工程の成果物‘＊1’および製品‘＊2’に関するレビューを指す。

本編との関連

「4.3.1　要件定義成果物のレビュー」ではレビュー手法（インスペクション）および実施上の留意事項(勘どころ)を取り上げて説明したが、本稿ではその背景などを含め関連づけている。

欠陥摘発工程の違いによる手戻りコストから捉えたROI

一般的に、混入した工程独自欠陥の手戻りコストは、経験的にレビューとテストでの摘発工程の違いにより増減することが分かっている。しかし、どのくらいの増減値になるかを定量的に捉えた文献資料は極めて稀である。

本稿では要件定義工程独自欠陥を要件定義工程レビューで摘発した場合と、それ以降の開発工程（基本設計～システムテスト）で摘発した場合の手戻りコストの増減値を定量的に捉えることで、費用対効果を算出している方法を紹介する。具体的には手戻り量の算出式と各工程間生産物量変換係数（各工程生産物量単価を含む）を使用し、手戻りコストを求めるとともに、工程ごとの実績収集データと試算値とを突き合せ照合し、精度を高めている。

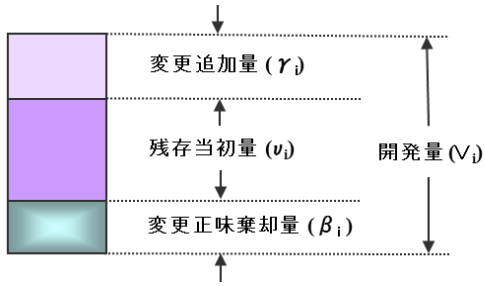
(1) 手戻り修正量の捉え方

① 欠陥混入工程と欠陥摘発工程を鑑みた手戻り量

ある工程 i の成果物に混入した工程独自欠陥による手戻り量と開発量との関係は以下のとおりである。

■ 開発量 ＝
残存当初量 ＋ 変更追加量 ＋ 変更正味棄却量

■ 手戻り量 ＝ 変更追加量 ＋ 変更正味棄却量



しかし、ある工程 i に混入した工程独自欠陥の手戻り量は、必ずしも工程 i の成果物対応だけに留まらない。つまり、手戻り量は欠陥混入工程と欠陥摘発工程との関係で増減する。その欠陥混入工程と欠陥摘発工程との関係を考慮した手戻り量は、欠陥混入工程を要件定義工程（変更追加量 γ_1、変更正味棄却量 β_1 ）とすると、以下の算出式で求めることができる。

$$手戻り量=\Sigma\,(n_L\times\gamma_1)+\Sigma\,(n_k\times\beta_1)$$

n は各工程間の生産物量変換変数
L は欠陥混入工程からシステムテスト工程までの添字
欠陥混入工程(要件定義)の場合、L＝1～7、$n_1=1$
k は欠陥混入工程から欠陥摘発工程までの添字
欠陥混入工程(要件定義)、摘発工程(基本設計)の場合、k ＝1～2、$n_1=1$

事例 1：要件定義工程独自欠陥を要件定義成果物のレビューで欠陥摘出したケース

開発工程 / 量	要件定義 (i＝1)	基本設計 (i＝2)	パッケージ設計 (i＝3)	プログラム設計 (i＝4)	プログラミング (i＝5)	統合テスト (i＝6)	システムテスト (i＝7)
変更追加量 (γ_i)	γ_1	$n_2\times\gamma_1$	$n_3\times\gamma_1$	$n_4\times\gamma_1$	$n_5\times\gamma_1$	$n_6\times\gamma_1$	$n_7\times\gamma_1$
残存当初量 (v_i)	v_1	$n_2\times v_1$	$n_3\times v_1$	$n_4\times v_1$	$n_5\times v_1$	$n_6\times v_1$	$n_7\times v_1$
変更正味棄却量 (β_i)	β_1	—	—	—	—	—	—

留意）n_L は生産物量変換係数である。 手戻り量＝$\Sigma(n_L\times\gamma_1)+\beta_1$

事例 2：要件定義工程独自欠陥をシステムテストで欠陥摘出したケース

開発工程 / 量	要件定義 (i＝1)	基本設計 (i＝2)	パッケージ設計 (i＝3)	プログラム設計 (i＝4)	プログラミング (i＝5)	統合テスト (i＝6)	システムテスト (i＝7)
変更追加量 (γ_i)	γ_1	$n_2\times\gamma_1$	$n_3\times\gamma_1$	$n_4\times\gamma_1$	$n_5\times\gamma_1$	$n_6\times\gamma_1$	$n_7\times\gamma_1$
残存当初量 (v_i)	v_1	$n_2\times v_1$	$n_3\times v_1$	$n_4\times v_1$	$n_5\times v_1$	$n_6\times v_1$	$n_7\times v_1$
変更正味棄却量 (β_i)	β_1	$n_2\times\beta_1$	$n_3\times\beta_1$	$n_4\times\beta_1$	$n_5\times\beta_1$	$n_6\times\beta_1$	$n_7\times\beta_1$

留意）n_L は生産物量変換係数である。 手戻り量＝$\Sigma(n_L\times\gamma_1)+\Sigma(n_k\times\beta_1)$

手戻り量は欠陥混入工程と欠陥摘発工程との関係で増減するが、前事例（事例 1、事例 2）からも分かるように、その増減する値は、既に作成された成果物の変更正味棄却量（β）の多寡に依存する。事例 1 と事例 2 での差分値は ’ $\Sigma n_{k+1} \times \beta_1$（k＝1～7）’ である。よって、レビューとテストでの手戻り量（コスト）に関わる ROI を考察する場合には、追加量（γ）の多寡は考慮する必要はない。

② デグレードを考慮した手戻り量

欠陥摘発工程がテスト工程である場合は、デグレード（改造型開発のテスト巻き込み）を考慮する必要がある。例えば、デグレード率（d）によるテスト追加量を取り入れた手戻り量は、以下の算出式になる。

デグレードを考慮した手戻り量＝$\Sigma (n_L \times \gamma_1) + \Sigma (n_k \times \beta_1) + \Sigma (d \times n_j \times \upsilon_1)$

（j は単体テスト工程から欠陥摘発テスト工程までの添字）

(2) 欠陥混入工程と欠陥摘発工程の違いによる増減する手戻りコストの比較事例

手戻りコストは、生産物量変換係数（n）*4 を考慮した手戻り量に工程別生産物量単価（m）を乗算することで算出することができる。欠陥混入工程（要件定義）と欠陥摘発工程との違いにより増減する手戻りコストは以下のようになる。

＊4：各工程間の生産物量変換係数は 4 つの基本パターン（基盤と処理形態の組み合わせ）が存在する。

増減する手戻りコスト＝$\Sigma((n_{k+1} \times \beta_1) \times m_{k+1}) + \Sigma(d \times n_j \times \upsilon_1 \times m_j)$

以下に生産物量変換係数「基盤‘Server 系’、処理形態‘Online’」のパターンについて、要件定義書‘1’とした場合の生産物量変換係数をご紹介する。

パターン ＼ 開発工程生産物	要件定義書	基本設計書	パッケージ設計書	プログラム設計書	プログラミング	統合テスト項目	システムテスト項目
基盤　：Server 処理形態：Online	1	1.44	7.6	9.53	0.93	65.55	18.69

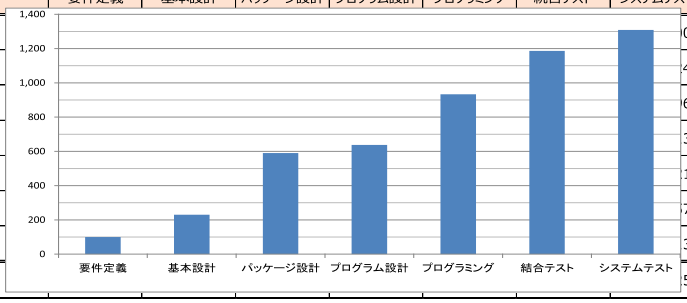
注)テスト工程生産物量は組数(ケースなど)を使用しているが、ここでは一般的に使用されているテスト項目数で表記している。

上記の手戻りコスト算出式から、要件定義工程独自欠陥を要件定義工程で欠陥摘発した手戻りコストを‘1’とした場合、要件定義工程以降の開発工程での各欠陥摘発工程の違いによる手戻りコストのコスト倍率事例を以下に示す。

事例 3：基盤‘Server 系’、処理形態‘Online’のケース

要件定義	基本設計	パッケージ設計	プログラム設計	プログラミング	統合テスト	システムテスト
1.00	**2.30**	**5.90**	**6.38**	**9.33**	**11.86**	**13.09**

摘発工程 / 修正対象生産物	生産物量変換係数 (n_L)	要件定義工程混入欠陥に対する摘発工程別の手戻り修正コスト（量×量単価「円/生産物量」）						
		要件定義	基本設計	パッケージ設計	プログラム設計	プログラミング	統合テスト	システムテスト
要件定義書（Kc）	$n_1=1$							00
基本設計書(Kc)	$n_2=1.44$							44
パッケージ仕様書(Kc)	$n_3=7.60$							69
プログラム設計書(Kc)	$n_4=9.53$							31
プログラミング(KLoc)	$n_5=0.93$							15
統合テスト仕様書(テスト項目)	$n_6=65.55$							70
システムテスト仕様書(テスト項目)	$n_7=18.69$							31
計								59



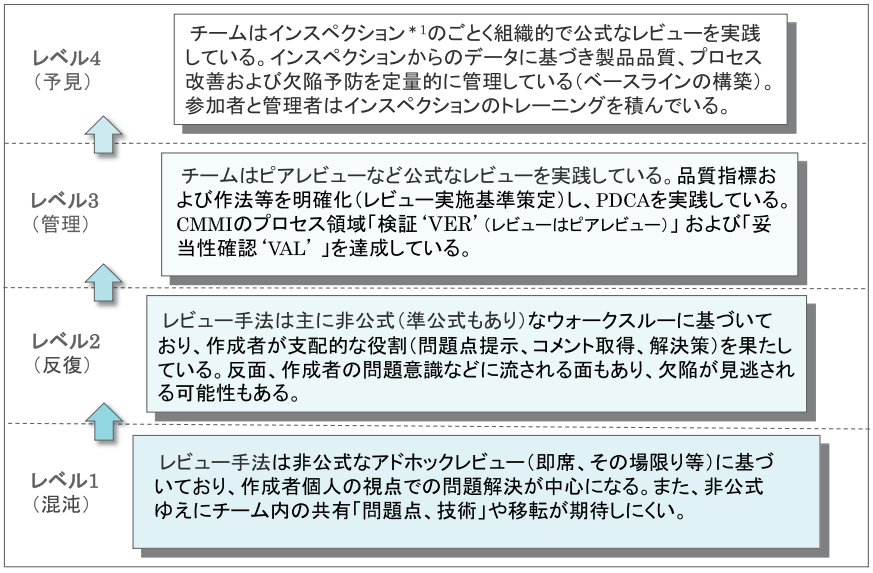
（注）デグレード率は+25%

なお、事例 3 は基盤 'Server 系'、処理形態 'Online' のケースであるが、基盤 'Host 系'、処理形態 'Online' のケースでは、要件定義工程独自欠陥を要件定義工程で欠陥摘発した手戻りコストを '1' とした場合、生産物量変換係数の違いによりシステムテストでは '13.70' になる。

レビュープロセス成熟度と品質指標

(1) レビュープロセス成熟度とは

ソフトウェア成果物を早期にかつ効率的に欠陥を取り除くには、組織のレビューに関するプロセス能力を高めることが必要である。本稿ではレビューに関わるプロセス能力を、組織の振る舞いの観点からレビュープロセス成熟度（レベル 1～レベル 4）として定義する。その概要を図 5.16 に示す。



＊1 ：ANSI/IEEE 標準 1028-1998で体系化され定義されており、厳格で公式なレビュー方法です。
他の手法に比べ、欠陥摘発にもっとも効果が高い手法の一つです。

図 5.16 レビュープロセス成熟度の概要

なお、弊社では全ての開発プロジェクトチームに関して、定期的にレビュープロセス成熟度を測っている。その結果を踏まえ、当該プロジェクトチームの成熟度レベルを向上すべく、目標設定と課題改善に取り組んでいる。その取り組み概要と改善効果に関する事例は、後述の「レビュープロセス成熟度の向上による手戻りコスト改善事例」で紹介する。

(2) 工程別のレビュープロセス成熟度と品質指標

各チームのレビュープロセス成熟度を評価するとともに、チームごとの品質指標（レビュー密度と欠陥指摘密度）を実績データ収集し統計分析した結果に基づき、その相関を得たものを図 5.17 に示す。

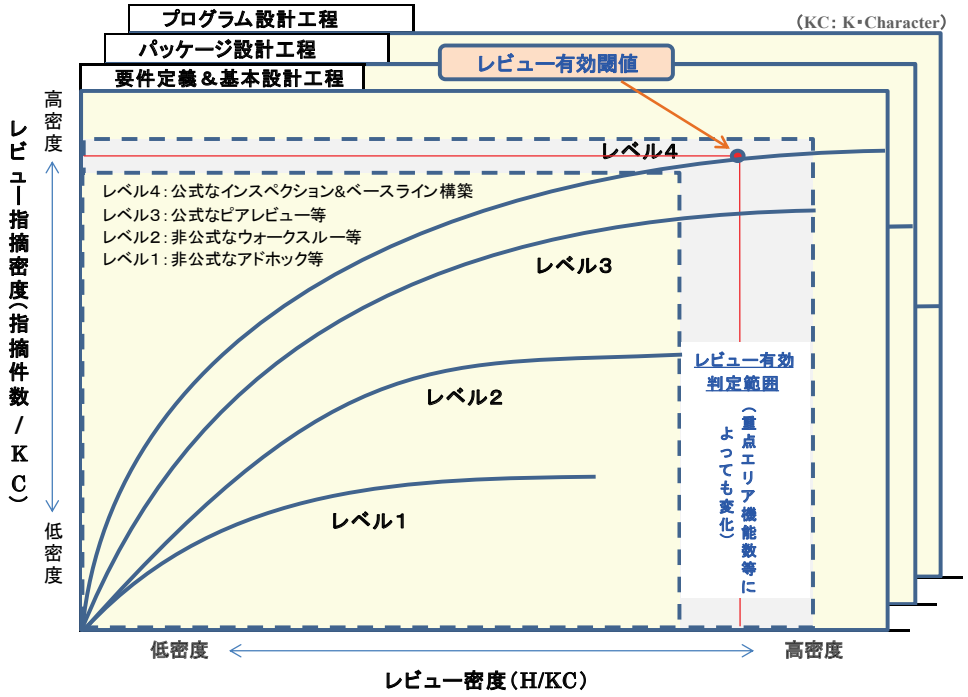


図 5.17 工程別のレビュープロセス成熟度と品質指標

相関関係は要件定義＆基本設計工程～プログラム設計工程までを調査している。

図 5.17 には要件定義＆基本設計工程の事例を表記しているが、実績データ収集から得られた相関関係は、パッケージ設計工程、プログラム設計工程と工程が進むにつれて、レベル 1～レベル 4 の回帰曲線は寝てきており、かつ、レベルの差は縮む傾向にある。つまり、上流工程（要件定義工程＆基本設計）ほど、レビュープロセス成熟度の高レベル効果が現れることになる。よって、弊社では、特に上流工程での高レベル化を図っている。

レビュープロセス成熟度の向上による手戻りコスト改善事例

(1) レビュープロセス成熟度の違いによる手戻りコスト差異

要件定義に混入した要件定義工程独自欠陥が、どの工程でどのくらいの割合（%）で欠陥摘発されるかを、チーム（レビュープロセス成熟度）ごとに実態を調査している。以下にレビュープロセス成熟度レベルごとの欠陥摘発割合分布（平均値）を示す。

レビュー・プロセス成熟度＼開発工程	要件定義 摘発比率%	基本設計 摘発比率%	パッケージ設計 摘発比率%	プログラム設計 摘発比率%	プログラミング 摘発比率%	統合テスト 摘発比率%	システムテスト 摘発比率%	計 摘発比率%
レベル4	94.2	2.3	1	0	0	0.9	1.6	100
レベル3	88.5	4.9	1.6	0	0	1.7	3.3	100
レベル2	73.4	9.2	3.8	0.8	0.1	4.4	8.3	100

参考1）純粋な欠陥（誤り、漏れ等）以外に、後工程での積み残し分（仮置き含む）の対応も含まれる。
参考2）JUASソフトウェアメトリクス調査2016では工期遅延の50%が要件定義工程が不十分との調査結果もある。

ここで、上記表での工程別の欠陥摘発割合および前述の「欠陥摘発工程の違いによる手戻りコストから捉えた ROI」(2)の事例 3（基盤：Server 系、処理形態：Online）での手戻りコストのコスト倍率を使用し、レビュープロセス成熟度レベルの違いによる手戻りコスト差異を試算している。

その結果、手戻りコスト差異はレベル 4 を‘1’とすると、レベル 3 は 1.3 倍に、レベル 2 は 2.1 倍のコスト増になるという結果を得た。

(2) レビュープロセス成熟度の向上による手戻りコスト改善効果事例

一つ目は、前述の(1)のレビュープロセス成熟度レベルの違いによる手戻りコスト差異の試算事例から、レビュープロセス成熟度レベル向上による手戻りコストの改善効果が得られる。以下にレベル向上の 3 パターンを想定した改善効果を示す。

- レベル 2 からレベル 3 への向上による手戻りコストの削減効果は 39%
- レベル 3 からレベル 4 への向上による手戻りコストの削減効果は 21%
- レベル 2 からレベル 4 への向上による手戻りコストの削減効果は 52%

二つ目は、要件定義工程に特化した事例ではない。先に、弊社の全開発プロジェクトチームに関して、定期的にレビュープロセス成熟度を測っていることを述べた。ある年度（改善前）のプロジェクトチーム別のレビュープロセス成熟度レベルの実績から、チーム別に成熟度レベルの向上目標を宣言（改善後）し、向上による組織全体のコスト改善効果を得る事例である。ここでは、紙面の関係で詳細は割愛するが、直近の年度予算において、改善前と改善後のコスト改善比率 8.1%を目標としている。但し、成熟度レベル向上に関わる費用（トレーニングや資料整備などの費用、管理費用など）を除いている。

5.8 「要件定義文章の品質向上」の事例 ～図表を使った記述技法～ 株式会社 NTT データ

取り組み背景

要件定義工程で定義する機能要件はその記述内容に応じて、標準的な技法（ER 図、UML、BPMN など）を使って記載する。しかし、ビジネスルールに関しては、標準的な技法が定着している状況ではない。このため、ビジネスルールをフリーフォーマットで記述してしまう傾向にある。

ビジネスルールを自然言語で記述する場合、以下のような問題がある。

- 曖昧な記述のため、抜けや矛盾を含みやすい
- 作成者によって記述粒度、品質のばらつきが生じやすく、可読性が低くなる

など

NTT データでは、要件定義書、設計書の作成時に、ビジネスルールをその内容に適した記法で記述できるよう、記法の使い分けと記述のコツを体系化し、グループ内のみならず、一般書籍化[3]し展開している。

本稿では、本内容の一部を紹介する。

本編との関連

「4.3.3　要件定義文章の品質向上」と関連している。

ビジネスルール記述技法

本項では、ビジネスルールの記述内容に対する記法の使い分けと、一部記法の概要を説明する。

(1) ビジネスルールの記述内容に合わせた記法の使い分け

ビジネスルールの記述に内容に合わせた記法の使い分けを下表に示す。

表 5.10 記法の選択（1/2）

ビジネスルール分類	記述内容	記法
『事実』	データモデル	ER 図
	用語の定義	表（用語定義）
	項目の列挙	箇条書き
	分類の指定	表（分類）
	項目が存在する条件の指定	表（項目の存在条件）

[3] NTT ソフトウェアイノベーションセンタ、NTT データ：ビジネスルールを可視化する 要件定義の図解術，日経 BP 社， 2015

表 5.10 記法の選択（2/2）

ビジネスルール分類	記述内容	記法
	複数の値に対する関係の指定	計算式
	文字（記号、符号）列のフォーマットの定義	図（フォーマット）
		表（フォーマットの適用項目）
	値の有効範囲の指定	数直線グラフ
		表（値のセット）
	条件による事物などの定義	表（事物の成立条件）
	事物などの関係（関連と多重度）の定義	階層図、クラス図
	行為に対する責任の割り当て	責任分担表
『契機』	アクター毎のアクティビティの定義	アクティビティ図、BPMN
『制約』	事物などに対するアクター毎の操作の制限	CRUD 図
	条件による行為の制限	デシジョンテーブル
		ステートマシン図
		表（状態毎の行為の可否）
		表（対象毎の行為の可否）
	事前に実施する行為の指定	アクティビティ図、BPMN
	期間や期限の指定	期限チャート
	値の増減に関する制限	折れ線グラフ
	空間的な制約の指定	空間配置図
『推論』	事物などの状態の変化の指定	ステートマシン図
	時間帯の指定	時間帯グラフ
	条件の組み合わせによる出力パターンの指定	例示
		デシジョンテーブル
『計算』	計算式の実行順序の指定	アクティビティ図
	四則演算及び切り上げ切り捨て	計算式
	入力値によって変化する計算式の指定	階段グラフ
		計算式
		デシジョンテーブル

(2) 「期限チャート」をつかった「期間や期限の指定」の記述例

ある事由に対する実施内容とその期限が指定されている場合、事由と実施内容と期限の時系列関係を図示することで、条件に対して、実施する行為の期限が、時間の長さとして視覚的に理解しやすい。

さらに、具体的日付を入れた例も合わせて記述することによって、条件や実施する行為の時間的な制約として具体的な始点と終点を把握しやすくなる。

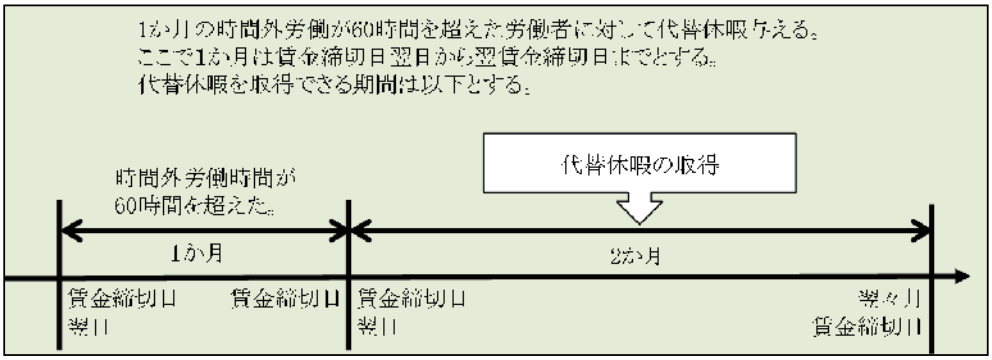


図 5.18 「期限チャート」をつかった「期間や期限の指定」の記述例

(3) 「デシジョンテーブル」を使った「条件の組み合わせによる出力パターンの指定」の記述例

条件の組み合わせと判定結果（動作）を表形式で記述することにより、可読性の低下や、条件の組み合わせの漏れや誤りを見つけやすくなる。

①受験者の確認
試験会場（研修室）入り口で受験者持参の「写真付き証明書」で本人確認をする。
・「写真付き証明書」がない ⇒ 社員証、健康保険証等で本人確認ができれば例外的に受験可能。
・本人でない ⇒ 受験不可。
・受験日が違った ⇒ 受験不可。
・「受験票」がない ⇒ 受験票のコピーと照らし合わせ本人確認ができれば受験可能。

②受験者への説明
受験者に受験票を返し、オリエンテーション開始までに必ず着席するよう伝える。また、試験中はトイレ休憩がないため、事前に行くようあわせて伝えること。

③遅刻者の対応
試験受付時間内に受付をしない場合、受験することができない。但し、交通機関の遅延に限り、遅延証明書があり試験開始時刻を過ぎてない場合のみ受験を認める。
・交通機関の遅延（遅延証明書がない） ⇒ 受験不可
・その他の理由による遅刻（体調不良等） ⇒ 受験不可
・交通機関の遅延（試験開始時刻を過ぎた） ⇒ 受験不可
・交通機関の遅延（遅延証明書がある、試験開始時刻を過ぎていない） ⇒ 受験可能

図 5.19 ビジネスルールを自然言語で表現した例

条件	写真付き証明書の有無	あり	Y	–	Y	–	–	–	–	–
		なし	–	Y	–	Y	–	–	Y	–
	写真付き証明書で本人確認	本人である	Y	–	Y	–	–	–	–	–
		本人でない	–	–	–	–	Y	–	–	–
	受験日	合っている	Y	Y	Y	Y	–	–	–	–
		違っている	–	–	–	–	–	Y	–	–
	受験票の有無	あり	Y	Y	–	–	–	–	–	–
		なし	–	–	Y	Y	–	–	–	Y
	社員証、健康保険証等で本人確認	一致	–	Y	–	Y	–	–	–	–
		不一致	–	–	–	–	–	–	Y	–
	受験票のコピーと照らし合せ本人確認	一致	–	–	Y	Y	–	–	–	–
		不一致	–	–	–	–	–	–	–	Y
動作	受験可能		X	X	X	X	–	–	–	–
	受験不可		–	–	–	–	X	X	X	X

凡例
【条件】　Y：条件が真　–：条件に無関係
【動作】　X：動作が生じる　–：動作が生じない

図 5.20　ビジネスルールをデシジョンテーブルで表現した例

おわりに

おわりに

本書では、システム構築の上流工程における諸作業を適切に行うことにより、開発プロジェクトの失敗を減らし、構築するシステムに対する品質要求に応え、対象システムにより実現されるビジネスや新たなサービスに、より高い価値をもたらすことを目的とした。

現場で発生する事例をもとに問題を洗い出し、分析する中で解決すべき課題を明らかにして、優先度の高い内容に対してノウハウを集結したので、参考にしていただきたい。

ただし、皆さまが持つ様々な問題は、一朝一夕では解決しないことと想定される。長期的なスパンでの取り組みが求められる方々も多くいらっしゃるかと思われる。一方で、IT 技術の昨今のパラダイムシフトは目覚ましく、求められるスピードも一層早くなると、取り組むべき内容や優先度も変化するかもしれない。

その中で、上流工程の重要性に改めて理解を深めて、ステークホルダがともに手を取り合い、地道な改善を続けていくことが目指すべき方向ではないか。

冒頭で述べたように、IT の役割は、従来の"Support Business"から、今日では"Do Business"へと変わってきている。このような状況において競争優位を維持し続けるためには、IT システムのユーザは、ビジネス環境の変化に機敏に対応するとともに、自らそのシステムにビジネス要件を的確に反映することが必須である。そのために、本書が少しでも役立てば幸いである。

付録

付録１　共通フレームが規定する要件定義関連のプロセス

要件定義で作成すべき成果物を決めるためには、要件定義や前後の工程の目的を理解する必要がある。ここでは「共通フレーム 2013」[15]に従って解説する。

共通フレーム 2013 は、発注者と開発者の双方の視点から情報サービス産業の標準的なライフサイクルプロセスを定めている。これは、情報システムの開発や活用を通じて発注者と開発者双方が共に成功を得るための規範となっている。また、共通フレームでは標準とされる成果物を規定しておらず、役割の観点でまとめたプロセスをアクティビティ、タスクに詳細化して提示している。

共通フレーム 2013 が規定する要件定義関連のプロセスを付図 1.1 に示す。また、同図に記載された各プロセスの目的を以下に示す。

(1)「企画プロセス」

①「システム化構想の立案プロセス」

経営上のニーズ、課題を実現、解決するために、新たな業務の全体像とそれを実現するためのシステム化構想を立案する。

②「システム化計画の立案プロセス」

システム化構想を具現化するために、運用や効果等の実現性を考慮したシステム化計画、プロジェクト計画を具現化し、ステークホルダの合意を得る。

(2)「要件定義プロセス」

利用者及び他のステークホルダが必要とするサービスを提供できるシステムに対する要件を定義する。(業務要件定義)　ステークホルダのニーズおよび要望を分析し、以下の対応を行う。

- 相互作用を表現する
- システムがニーズを満たすことを確かめる
- 合意を取る

(3)「システム開発プロセス」

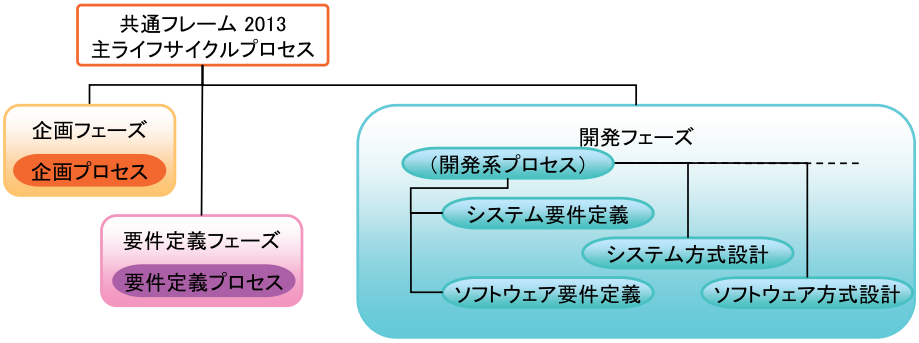
①「システム要件定義プロセス」

定義されたステークホルダの要件をシステムの技術的要件へ変換する。(システム要件定義)

(4)「ソフトウェア実装プロセス」

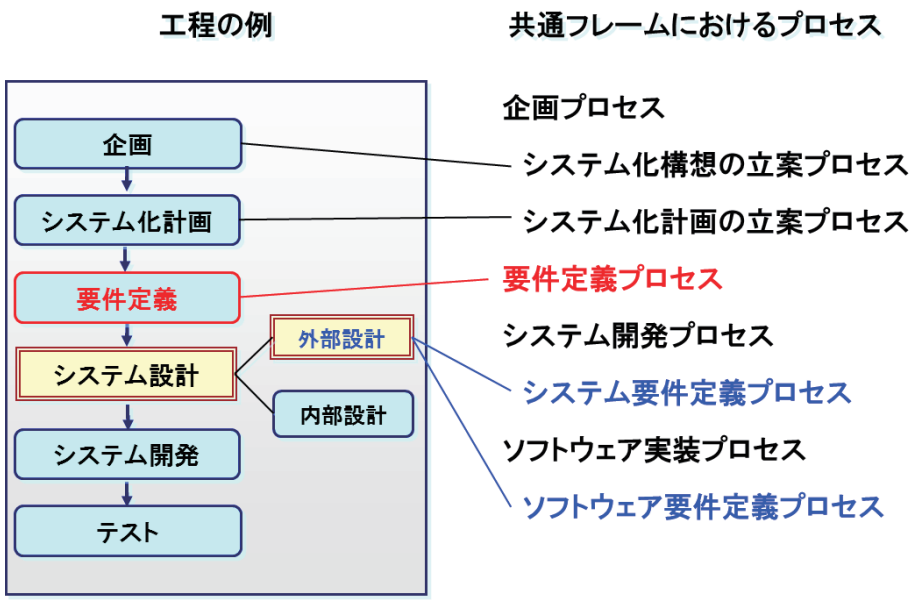
①「ソフトウェア要件定義プロセス」

システムのソフトウェア要素の要件を確立する。(システム要件定義)



付図 1.1 共通フレーム 2013 における要件定義関連プロセス

ここで注意すべきことは、共通フレームでは、一般に使用される「工程」を、要件定義プロセスというように「プロセス」と表現しているということである。これは、システム開発を役割の観点からまとめているためであり、「プロセス」は期間や順序を規定したものではないことを意味している。「工程」は、一般に企業ごとにその呼称や定義が異なる。各企業は必要に応じ、自身の定義する「工程」と共通フレームの「プロセス」とのマッピングを行っている。このような工程とプロセスとの対応例を付図 1.2 に示す。



付図 1.2 工程とプロセス（共通フレーム）との対応例

プロジェクトの状況や目的によって、どの工程にどのプロセスを配置するか、工程の目的を達成するためにどのような成果物を選択するかが異なる。

例えば、プロジェクトの目的が「経営上のニーズや課題を解決するための業務プロセスの改善」であれば、企画工程で業務フローを作成し、業務プロセスのニーズや課題を業務フロー上で確認することになるかもしれない。また、プロジェクトの目的が「ステークホルダが必要とするサービスを提供できる IT システムの明確化とそれによる業務の変化の明確化」であれば、要件定義工程でシステム化業務フローを作成し、設計工程では作成したシステム化業務フローを使って開発部門と「ユーザと IT システムの役割分担」について合意形成することになるかもしれない。

また、一般的にシステム化業務一覧やシステム化業務説明は、システム機能の外部仕様をまとめる成果物として開発プロセスに位置づけられる。しかし、プロジェクトの特性によっては「利用部門が主として参画している要件定義工程で実施した方が効率的である」、「見積もり精度を上げるために要件定義工程で作成する」などと判断して、作成する工程を変更してよい訳である。

各プロジェクトでは、それぞれが置かれている状況にもとづいてプロジェクトの目的を明確にし、作業プロセスを配置し、作成するドキュメントを決定していくことが求められる。

付録２　システム要件定義のコツをまとめた「機能要件の合意形成ガイド」

IPA/SEC が公開している「機能要件の合意形成ガイド」[11]は、システム要件定義に関わる成果物作成上のコツが示されているので参考になる。

機能要件の合意形成ガイドは、実現したい情報システム像を伝える側（発注者）と、それを設計し開発する側（開発者）との間で起こる認識の齟齬を防止することを目的としたガイドであり、発注者と開発者が仕様を確認する最後の段階である外部設計（共通フレームの用語では、システム・ソフトウェア要件定義）での発注者-開発者間の合意形成のコツを紹介している。このガイドには、成果物に対し「何を書けばよいか」「どう書けばよいか」についても多く記載されている。

機能要件の合意形成ガイドは、外部設計工程での活用を想定して記載されており、そこには、システム化する機能の視点での合意形成のコツが書かれている。例えば、システム化業務フローでは、システム化する業務の範囲や機能、システムとの接点の合意形成のためのコツが書かれており、システムを利用した業務がどれだけ業務改善に貢献できるかという視点とは異なる。しかし、要件定義（業務要件定義）の成果物は外部設計へのインプットであり、外部設計へのつながりを意識して要件定義を実施することが重要であることから、合意形成ガイドはぜひ参考にしていただきたい。参考として、要件定義に関連する機能要件の合意形成ガイドの成果物に対するコツの概要を「付表 2.1」に掲載した。詳細は、IPA/SEC のホームページからダウンロードできる。

また、このガイドで提示している成果物は、外部設計で合意形成をする必要があると説明しているが、必ずしも外部設計で作成すべきであるとは述べていない。要件定義プロセスや企画プロセスで作成すべき成果物もあるので注意が必要である。

付表 2.1 本ガイドの成果物で活用可能な「機能要件の合意形成ガイド」の施策（コツ）の例
＜システム振る舞い編＞

工程成果物名	コツ分類	コツID	施策（コツ）
システム化業務一覧	書き方	02D101	システム化業務一覧の中のシステム化業務をグルーピングして、階層的に記述する。
		02D102	システム化業務一覧は、要件定義で作られた業務一覧と業務フローを使って系統立てて記述する。
		02D103	システム化業務一覧のそれぞれのシステム利用作業に、対応する利用者の記述を追加する。
		02D105	システム化業務一覧に記述する機能一式は、業務処理の流れに沿って整列させる。
		02D106	システム化業務一覧に記述された各機能に対して、対応する業務を関係付ける。
		02D107	システム化業務一覧に記述された各システム化業務に対して、その振舞いの規定や制約などの特徴を関係付ける。
	確認	02C201	システム化業務一覧を使って、システム化業務の名前が一意であることを確認する。
	レビュー	02R201	システム化業務一覧を使って、業務とその業務に関連する機能との対応を確認する。
システム化業務フロー	書き方	02D201	システム化業務フローは、業務フローの1業務に対応させる形で記述する。
		02D202	システム化業務フローの詳細を隠蔽した、大きな流れを記述する。
		02D203	システム化業務フローの各業務を、部門などの役割に基づいた区画に記述する。
		02D204	システム化業務フローにおいて、システム化する部分を、システム化業務を表すアイコンを使って記述し、システム化しない業務と区別する。
		02D205	システム化業務フローにおいて、システム化する範囲を、区画の1つを使って記述し、区画の境界を通してシステム化の内と外を区別する。
		02D206	システム化業務フローにおいて、システム化業務の流れ、データの流れを表す矢印は、それぞれ異なる線の種類を用いる。
		02D207	システム化業務フローの矢印を交差しないようにする。
		02D208	システム化業務フローの全体を俯瞰できるように、図表1枚に記述するシステム化業務を階層化する。
		02D209	システム化業務フローにおいて、開発対象と開発対象外のシステム化業務を、異なるアイコンを使って記述し、両者を区別する。
		02D210	システム化業務フローに説明文を併せて記述する。
		02D211	システム化業務フローに書かれているシステム化業務と、その各々を詳細に説明する文章・図表を互いに関連付け、お互いに辿れるように記述する。
		02C101	システム化業務フローで、使用される画面や帳票の対応関係を確認する。
	レビュー	02R101	発注者は、社内他部門、社外関係者など、当該業務に関与する関係者（利）の業務処理内容、処理・作業順序を、業務フローとシステム化業務フローおよびシステム化業務一覧などと比較・検証する。
		02R202	システム化業務フローを使って、業務上のタイミングを確認する。
		02R203	システム化業務フローを使って、分岐の境界条件を確認する。
		02R301	レビューは、要件定義書に登場する主要な登場人物（利用者、関係者）を確認する。
		02R402	最初に、サブシステム間、開発対象と開発対象外部、システム化業務と作業、などの境界に着目し、境界部分の入出力データをレビューする。
		02R403	システム化業務一覧とシステム化業務フローで、異なる利用者から使われる同じ名前のシステム利用作業に着目して、それらが本当に同じであるかを確認する。
		02R407	画面・帳票をあげながら、業務とシステムを対応付けて、システム化業務フローをレビューする。
		02R408	重要な日付／期間に応じたシステム化業務を確認する。
		02R501	システム化業務フローに手順や手続きを明記し、順序を意識しながらレビューする。
		04R201	システムが扱う情報を俯瞰できる資料を作成し、システムが扱う情報の範囲を確認する。
システム化業務説明	書き方	02D301	システム化業務説明に、システム化業務の実行前後の条件を記述する。
		02D302	システム化業務のシナリオを、利用者とシステムのそれぞれのアクションに分けて記述する。
		02D303	1つのシステム化業務の振舞いに対して、基本・代替・例外の3種類のシナリオを考える。
		02D304	システム化業務のシナリオの中の個々のアクション（箇条書き）の先頭に、連番ではなく枝番を使用する。
	レビュー	02R404	システム化業務説明のシステムのアクションにより、要件定義書の現行システムの課題が解決していることを確認する。
		02R405	利用者の役割・権限に着目して、システム化業務説明の基本シナリオのバリエーションを確認する。
		02R406	システム化業務フローに書かれている、システム化の対象範囲外の作業間で引き渡されるデータの種類と状態を確認する。
		02R502	画面一覧と帳票一覧を用いて、画面や帳票を通して利用者が関わるシステム化業務が何かを確認する。
		02R503	画面レイアウトと帳票レイアウトを用いて、画面や帳票を通して利用者が関わるシステム化業務がどう動くかを確認する。
		02R504	システム化業務説明の例外シナリオに関係する画面レイアウト（エラー処理、エラー表示）を確認する。
		02R505	システム化業務説明の事前条件・事後条件・例外シナリオが、要件定義書の業務レベルの条件や制約に対応していること、および考慮されていないケースがないことを確認する。

＜画面編＞

工程成果物名	コツ分類	コツID	施策（コツ）
画面レイアウト	書き方	03D301	画面レイアウト上で、画面部品などの表現を整理・工夫する。
		03D303	画面レイアウト上の操作手順に、操作に対するシステムのアクションも記述する。
	確認	03C301	構成要素の配置やボタン押下時における簡単な動作のみの記述に留まるのではなく、入力情報や制約条件等の詳細情報も併せて記述する。
		03C302	概要に情報項目を記載する際には、なるべく情報項目群の単位で記述する。
	レビュー	03R205	レビューの際にエラー表示を具体的に説明する。
		03R401	画面イメージを提示して、実際の配色を確認する。
		03R402	権限や条件と対応して画面レイアウトを説明する。
画面レイアウト(標準化)	書き方	03D600	個別画面に依らず、共通して合意すべきことは、画面レイアウト共通ルールとして定義する。
		03D601	画面遷移・レイアウト共通ルールの１つとして、あらかじめページの構成要素を定義する。
		03D602	画面遷移・レイアウト共通ルールの１つとして、あらかじめ画面遷移パターンを定義しておく。
		03D603	画面遷移・レイアウト共通ルールの１つとして、あらかじめエラー表示方法・エラーメッセージを定義する。
		03D604	準拠すべき社内外の基準・標準、ガイドライン、特筆すべき要件、開発コンセプト等があれば明記しておく。
		03D606	複数の画面で表示される一組の入出力項目については、レイアウトを共通ルールとしてパターン化する。
		03D607	制約を示す項目やルールは可能な限り数値化する。
	確認	03C401	各エリアの役割を明記されているかを確認する。
		03C402	各エリアの名称に加えて、配色やレイアウト例を施した全体イメージが示されているかを確認する
		03C403	色名称に加え、実際の色で着色したフォントなどを併記しているかを確認する。
		03C404	使用できるカラーの範囲、基本的な配色を定義されているか、また、色彩の使い方について一定の法則が設けられているかを確認する。
		03C405	フォント名に加えて、フォント表示の具体例を示しているかを確認する。
		03C406	共通にするべき範囲が明確にされていることを確認する。

＜データ編＞

工程成果物名	コツ分類	コツID	施策（コツ）
概念データモデル	書き方	04D101	エンティティを分類でタイプ分けして、色、網掛け、枠囲みで表現する。
		04D102	ER図中のエンティティにインスタンスの作成、および更新の時期を記述する。
		04D103	業務単位にER図を作成する。
		04D104	イベント系エンティティをデータの発生順で記述する。
	レビュー	04R301	データモデルに、実データを当てはめて確認する。
		04R302	画面レイアウト、帳票レイアウトとER図の対応づけを確認する。
		04R307	新規要件に伴うER図の変更は、変更前／後を比較して効果を説明する。
		04R308	旧システムと新システムのデータベース変更の方針について業務内容の観点で図示して説明する。
		04R309	業務とエンティティのまとまりの対応を示し、データモデルの読み方を説明する。
データ項目定義書	書き方	04D204	現行システムなどの具体的エンティティおよび属性の抽出元を記述する。
CRUD図	書き方	04D301	Create,Read,Update,Deleteの記載欄を明確に分けて（固定させて）表現する。
		04D302	CRUD図のエンティティをイベント系エンティティとリソース系エンティティに分離して記述する。
		04D303	CRUD図のビジネスプロセスを実際の業務の時系列順に並べる。
		04D304	CRUD図のイベント系エンティティを、エンティティを作成する順番に沿って並べる。

＜帳票編＞

工程成果物名	コツ分類	コツID	施策（コツ）
帳票レイアウト	書き方	07D001	印刷量によって印字する行数が増減する場合、改ページ条件を考慮した帳票レイアウトを作成する。
		07D002	帳票レイアウトに印字される種類（テキスト、表部品、画像、バーコード）を明示する。また、具体的な部品の属性を明示する。
	レビュー	07R002	実業務で用いている紙の伝票と帳票項目説明及び帳票レイアウトの対応付けを確認する。
		07R003	プレ印刷部、固定文字列及び可変文字列の識別をし、それに基づいて発注者に確認する。
		07R004	帳票レイアウトに関する詳細な要件を確認するために具体的なイメージを提示しそれを基にレビューする。
		07R006	利用目的や利用者・利用シーンと照らし合わせ、実際の利用者を交え帳票のレビューを行う。

参考文献

[1] IPA/SEC,「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)，2006,
https://www.ipa.go.jp/sec/publish/tn05-002.html

[2]日本情報システムユーザー協会（JUAS)「企業 IT 動向調査報告書 2016」JUAS，2016

[3]日本情報システムユーザー協会(JUAS)「ユーザー企業　ソフトウェアメトリックス調査 2016」JUAS，2016

[4]経済産業省・情報システム・モデル取引・契約書（2007・4)，2007
http://www.meti.go.jp/policy/it_policy/keiyaku/model_keiyakusyo.pdf

[5]日本情報システムユーザー協会（JUAS）JIIP3　2014 年度報告書，JUAS,2015

[6]IPA/SEC，非機能要求グレード，2010,
https://www.ipa.go.jp/sec/softwareengineering/reports/20100416.html .

[7]富士通株式会社，要件定義手法 Tri-shaping，2011,
http://pr.fujitsu.com/jp/news/2011/02/9-2.html

[8]IPA，情報セキュリティ読本，実教出版，2013,
https://www.ipa.go.jp/security/publications/dokuhon/2006/

[9]IPA，情報セキュリティ教本，実教出版，2009,
https://www.ipa.go.jp/security/publications/kyohon2/index.html

[10]ISO/IEC 25010，Systems and software engineering -Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models，2011.

[11]IPA/SEC,「機能要件の合意形成ガイド」IPA/SEC，2010
https://www.ipa.go.jp/sec/softwareengineering/reports/20100331.html

[12]日本情報システムユーザー協会（JUAS)「システム・リファレンス・マニュアル（第 1 巻)」JUAS，2006

[13]SECBOOKS「高信頼化ソフトウェアのための開発手法ガイドブック」IPA/SEC，2011

[14]JUAS 編,福田修編著「SE を極める　仕事に役立つ文章作成術」日経 BP 社，2005

[15] IPA/SEC,「共通フレーム 2013～経営者、業務部門とともに取組む「使える」システムの実現～」，2013

[16]IPA,グローバル化を支える IT 人材確保・育成施策に関する調査）概要報告書，2011,
https://www.ipa.go.jp/jinzai/jigyou/global-report.html

[17]日本情報システムユーザー協会（JUAS)「ユーザー企業　ソフトウェアメトリックス調査 2012」JUAS，2012

[18]日本情報システムユーザー協会（JUAS)「ユーザー企業　ソフトウェアメトリックス調査 2014」JUAS，2014

執筆（敬称略）

システム構築上流工程強化部会
「システム化要求 WG」

主査	岩佐　洋司	一般社団法人アドバンスト・ビジネス創造協会
委員	太田　忠雄	株式会社ジャステック
	後藤　将夫	一般社団法人アドバンスト・ビジネス創造協会
	坂巻　雅貴	東京ガス株式会社
	崎山　直洋	株式会社エヌ・ティ・ティ・データ
	桜井　新	新日鉄住金ソリューションズ株式会社
	清水　淳史	セイコーエプソン株式会社
	髙橋　康介	大日本印刷株式会社
	髙橋　実雄	サントリーシステムテクノロジー株式会社
	中村　伸裕	住友電気工業株式会社
	藤本　礼久	全日本空輸株式会社
	森田　功	富士通株式会社
事務局	山下　博之	独立行政法人情報処理推進機構
	山本　英明	独立行政法人情報処理推進機構
	村岡　恭昭	独立行政法人情報処理推進機構

監修（敬称略）

システム構築上流工程強化部会

主査	山本　修一郎	国立大学法人名古屋大学
委員	岩佐　洋司	一般社団法人アドバンスト・ビジネス創造協会
	大山　宏	株式会社エヌ・ティ・ティ・データ
	小野　修一	株式会社エヌ・ティ・ティ・データ
	加藤　一郎	日本電気株式会社
	﨑本　壮	株式会社日立製作所
	澁谷　裕以	株式会社日本取引所グループ
	西村　光司	一般社団法人日本情報システム・ユーザー協会
	細川　泰秀	一般社団法人アドバンスト・ビジネス創造協会
	森田　功	富士通株式会社

SEC BOOKS

ユーザのための要件定義ガイド

～要求を明確にするための勘どころ～

平成 30 年 1 月 22 日　1 版 2 刷発行

監修者　独立行政法人情報処理推進機構（IPA）技術本部
　　ソフトウェア高信頼化センター（SEC）
発行人　松本 隆明
発行所　独立行政法人情報処理推進機構（IPA）
　　〒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
　　https://www.ipa.go.jp/

ISBN 978-4-905318-48-4　Printed in Japan

ISBN978-4-905318-48-4
C3055 ¥1574E

定価：本体1,574円+税

独立行政法人 情報処理推進機構
技術本部 ソフトウェア高信頼化センター

SEC-TN16-008