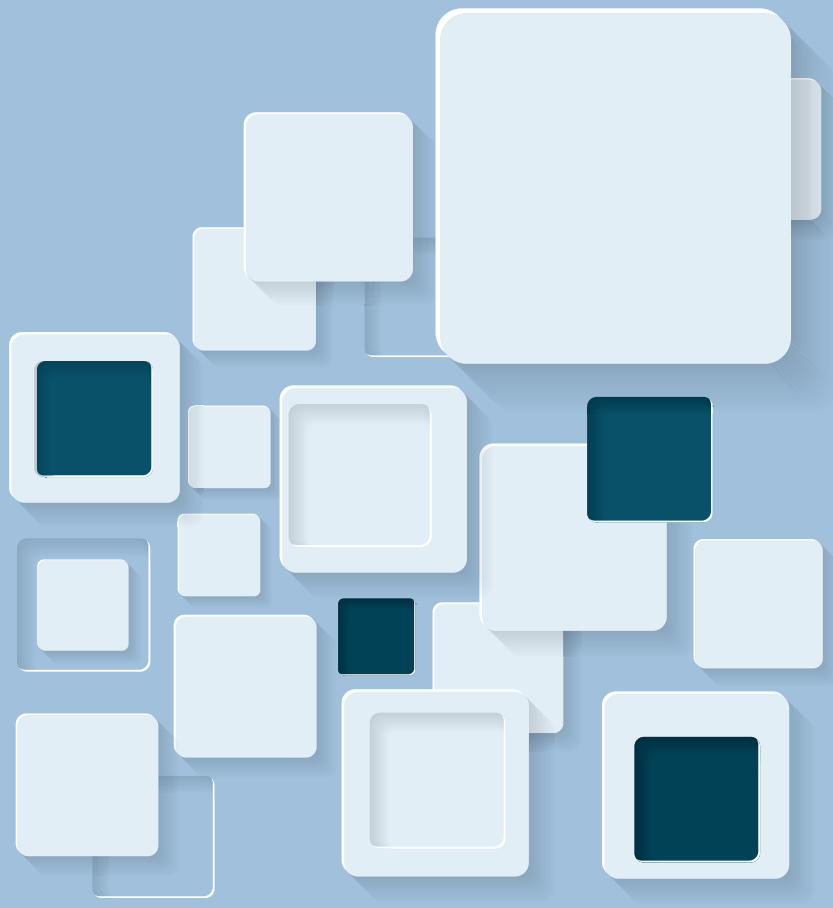
SEC BOOKS

第2版

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～

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



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～

第２版

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

- SAP、ABAP、SAP NetWeaver、および本文書に記載されたその他の SAP 製品、サービス、ならびにそれぞれのロゴは、ドイツおよびその他の国々における SAP AG の商標または登録商標です。
- Oracle と Java は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。
- その他記載の会社名、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）[5]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。

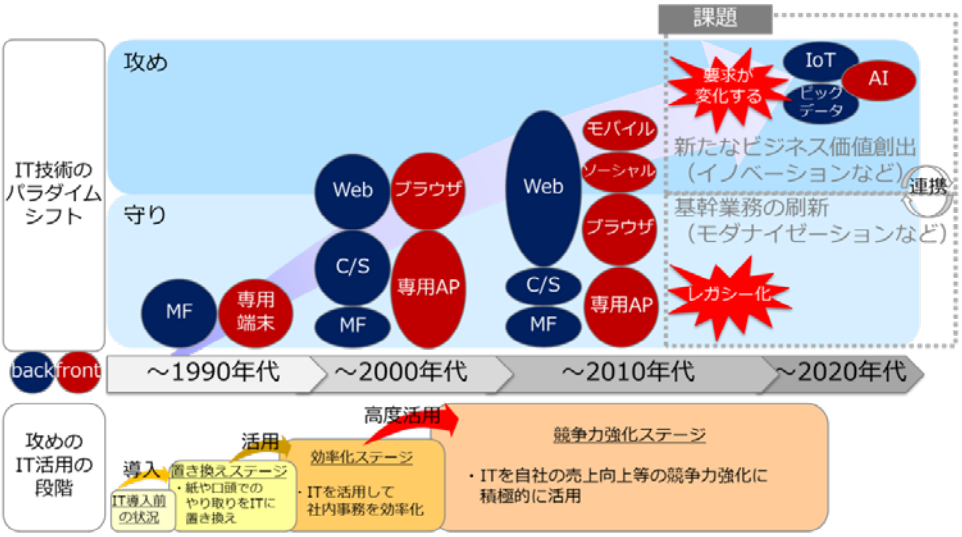


図 1　IT システムに関わる IT 技術とユーザ企業の現在

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。

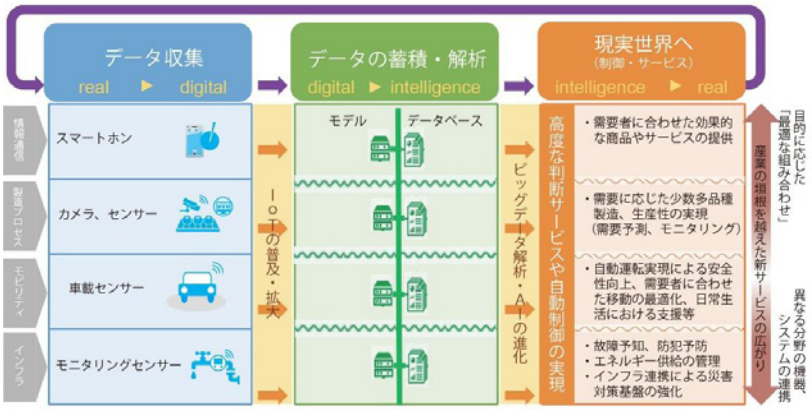


図 2　攻めの分野におけるシステム開発ライフサイクル（例）

（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。

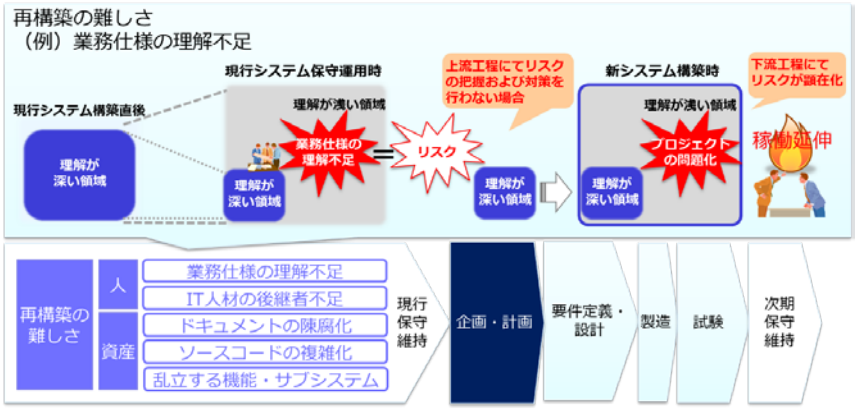


図 3　守りの分野におけるシステム再構築の事例

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。

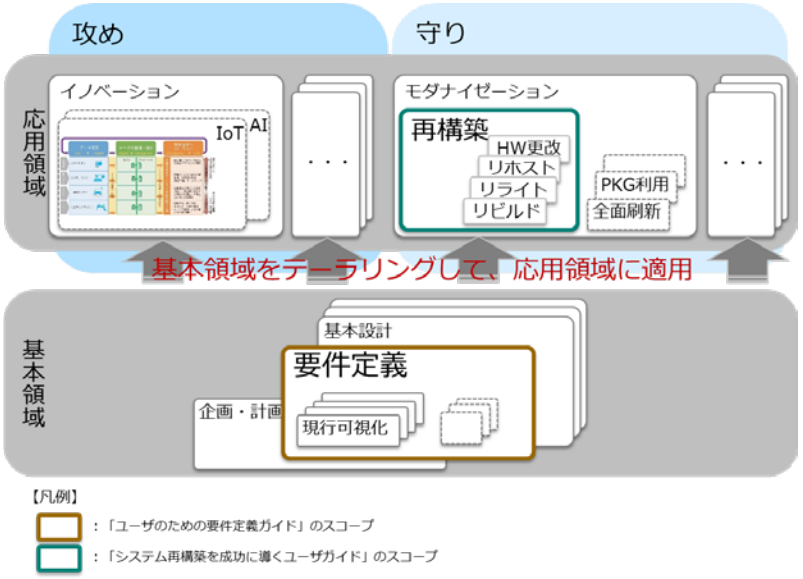


図 4　ソフトウェアエンジニアリングの各領域における取り組み

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～」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～」（以下、本ガイド）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。

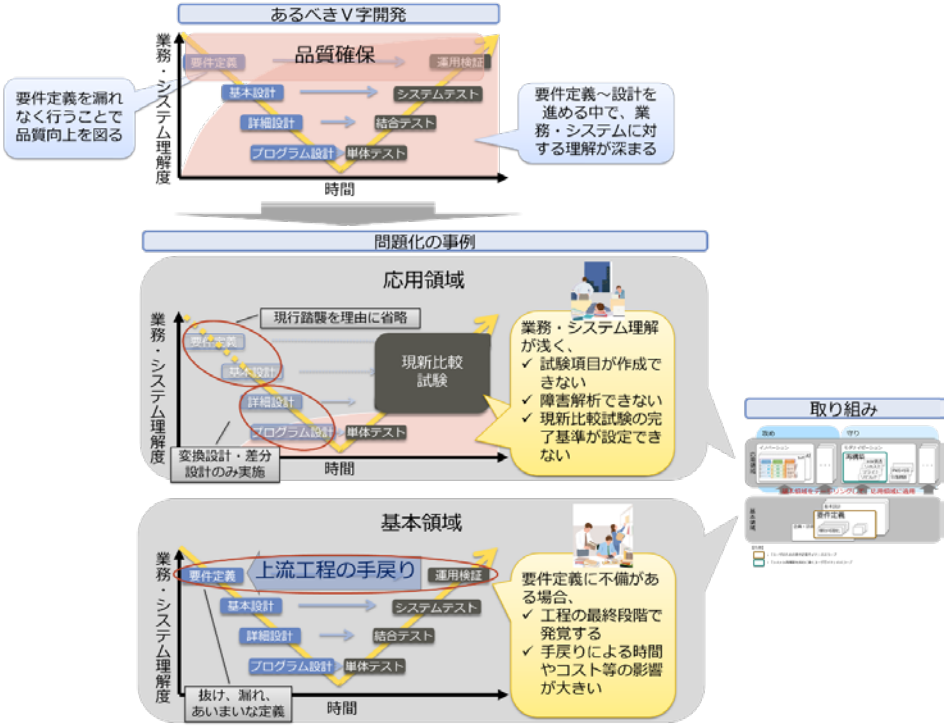


図 5　各領域に取り組む背景

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。

改訂にあたって

本ガイド第 1 版では、現行システムの業務要件を原則変えずに再構築するケースを対象として、安心してシステム再構築に取り組むための指針を示した。一方で、システムを再構築する際に合わせて業務要件を変更する、もしくは新たな業務要件を追加するケースも多く、その重要性が増している。また、システム再構築時にパッケージ製品を導入するケースでは、既存システムがあるために、パッケージ製品を利用するメリットを享受することが難しい側面もある。

そこで、IPA/SEC では、第 1 版でスコープ外とした「業務要件の変更／追加」、および「パッケージ製品の利用」を行うケースを対象に内容を改訂した。さらに、第 1 版の「3.6 品質保証の検討（観点 E)」において、重要な観点として取り上げた「業務継続性」をいかに担保するかについて内容の改訂を行った。

本ガイドを読まれたことがない方もすでに活用されている方も、ぜひ第 2 版をご一読いただき、システム再構築の現場で役立てていただきたい。

なお、主な改訂は以下の通りである。

章	節	改定
第1章　解説編	1.1　システム再構築を取り巻く現状	
	1.2　再構築の現場で何が起こっているか	
	1.3　再構築の問題化を防ぐために	更新
	1.4　再構築にパッケージ製品を利用する場合	新規追加
第2章　再構築手法選択編	2.1　再構築手法選択編概要	
	2.2　現行システムの調査・分析（ステップ1）	
	2.3　新システムの要求事項分析（ステップ2）	
	2.4　再構築手法の選択（ステップ3）	
	2.5　再構築手法の決定（ステップ4）	
	ケーススタディ	
第3章　計画策定編	3.1　計画策定編概要	
	3.2　要求の確認（観点A）	
	3.3　現行踏襲内容の明確化（観点B）	
	3.4　現行資産活用方針の検討（観点C）	
	3.5　現行業務知識不足への対応（観点D）	
	3.6　品質保証の検討（観点E）	更新
	3.7　意思決定プロセスの策定（観点F）	
	3.8　データ移行の計画（観点G）	
	3.9　再構築の計画と見積り（観点H）	
	3.10 業務要件の変更／追加への対応	新規追加
	ケーススタディ	
第4章　事例編	4.1～4.8　各社事例	

第１章　解説編

1.1 システム再構築をとりまく現状

システム再構築のニーズ

長年運用しているシステムを最新の IT 技術に対応させて近代化するモダナイゼーションが注目されている。現在利用しているシステムの基盤を始めとする製品を新しいものに変える取り組みは、マイグレーション、システム再構築などの呼称でかなり以前から行われているが、最近は、システムの老朽化や、メインフレーム技術者など、現行システムの技術者の減少への対応に加え、ビッグデータ、IoT、FinTech など最新のトレンドに対応することを契機に、モダナイゼーションを検討する場合も多い。

現行システムのレガシー化

システム再構築が求められる一方で、企業や組織の基幹システムにおいては、長期間に渡る安定運用を経た後のレガシーシステム化が進んでいる。

以下の図 1.1 は、日本情報システム・ユーザー協会（以降、JUAS と表記）がユーザ企業の基幹システムについて、システムのレガシー化の現状と今後（3 年後）の予測を「①技術面の老朽化」、「②システムの肥大化・複雑化」、「③ブラックボックス化」の 3 つの観点から調査した結果である。この調査によると、現状では約 3 割、今後でも 2〜3 割の企業が、保有しているシステムの多くまたは約半分でレガシー化の課題を持っていると回答している。システムの一部が該当すると回答した企業を含めると、現状では約 7 割、今後でも 6〜7 割がレガシー化の課題を抱えていることが分かる。

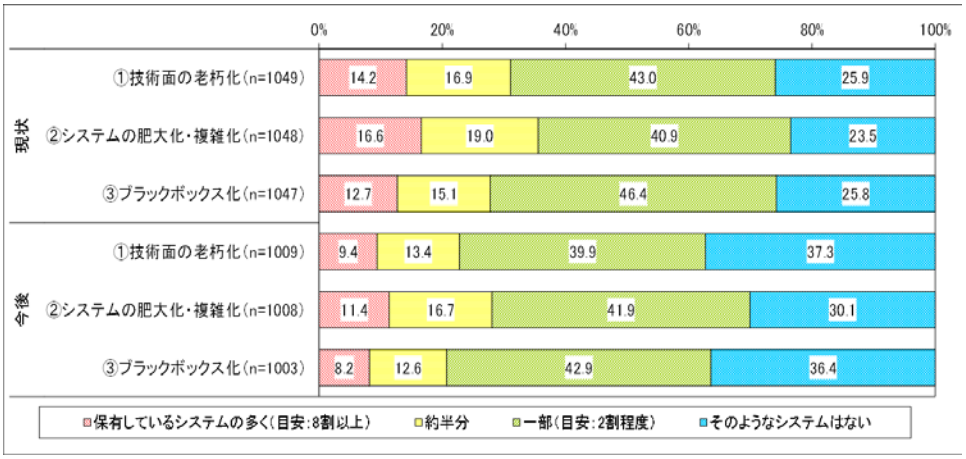


図 1.1 基幹系システムにおける課題システムの保有状況（現在と今後）

（出典）日本情報システム・ユーザー協会「企業 IT 動向調査 2016」[1]

図 1.1 のようなレガシー化の課題は、経営の観点からも深刻と考えられ、何らかの対応が必要と考える企業が多い。

以下に示す図 1.2 は、「①技術面の老朽化」、「②システムの肥大化・複雑化」、「③ブラックボックス化」のユーザ企業が保有するシステムの中での発生度合いごとに、経営上の深刻度の今後の予測を調査したものである。レガシー化の課題がシステムの半数以上にあるユーザ企業では、8～9 割が深刻度「中」以上と回答している。

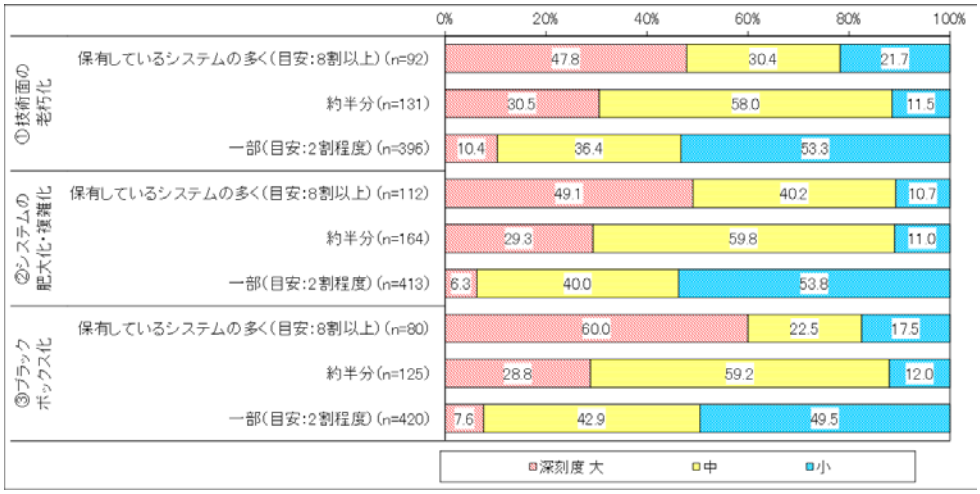


図 1.2 基幹系システムの課題システムの保有状況別　経営上の深刻度（今後）

（出典）日本情報システム・ユーザー協会「企業 IT 動向調査 2016」[1]

レガシー化の課題を抱えるシステムでは、課題の対応のためにシステム再構築が求められる。一方で、そのようなシステムでは、システム再構築において必要となる現行システムの業務知識が失われており、それが再構築の実現にとって大きな阻害要因となる。
そのような再構築の難しさや問題化した事例は、システムインテグレーションに関する学会や情報誌、Web サイトなどでも多く取り上げられており、IT 業界での難題として認知されている。

業務知識が失われる背景

業務知識がなぜ失われるのか、業務知識の存在領域の観点で考えると、次の(1)～(3)で示す背景がある。

(1) 業務知識の細分化
(2) 業務知識の断片化
(3) 業務知識の領域の変化

(1) 業務知識の細分化

大規模なシステムになると、図 1.3 のように開発者とユーザがシステム全体で必要とする業務知識を互いに補い合いながら保持している状態となることが多く、通常、ひとりのユーザや開発者が全ての業務知識を保持することはない。このように、複数のユーザや開発者が業務知識の領域を分けて理解している状態を「業務知識が細分化した状態」と呼ぶこととする。

業務知識の細分化は現行システムの構築当初から発生することが多く、そのようなシステムでは、長年運用を続ける中で(2)で示す業務知識の断片化が起こりやすくなる。

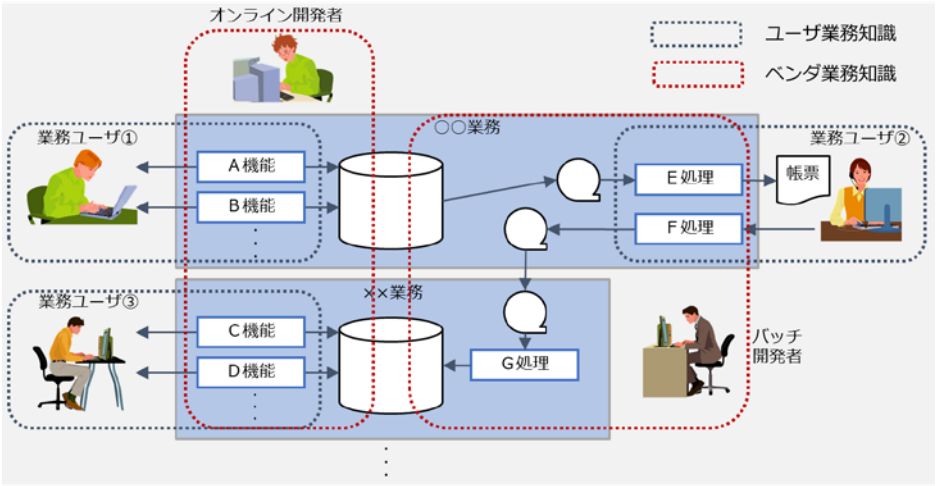


図 1.3 ユーザ企業・ベンダ企業双方における業務知識の細分化

(2) 業務知識の断片化

長期間に渡り運用保守している基幹システムでは、図 1.4 のように一部の業務知識が失われた状態となる。この状態を業務知識が断片化した状態と呼ぶ。

業務知識が断片化する原因としてまず考えられるのが、保守費削減に伴う体制の縮小や、人事異動・定年退職によりシステムに関わる人が交代することである。ある領域の業務知識を持った人がいなくなる際に後任がいない、あるいは後任に知識を伝える過程で漏れが発生することが往々にしてあり、その領域の業務知識が失われ、全体の知識が分断された状態（断片化した状態）になる。

長期間に渡って運用されるシステムであればあるほど関わっていたメンバの異動（昇進や他組織への転出、退職）は避けられないため、断片化は必ず起きる事象である。機能追加・修正やトラブルの改修に対して、保守費削減によるメンテナンス稼動の不足や単純な修正漏れよって、設計書等のドキュメントがメンテナンスされていないと、業務知識の断片化はますます深刻になる。

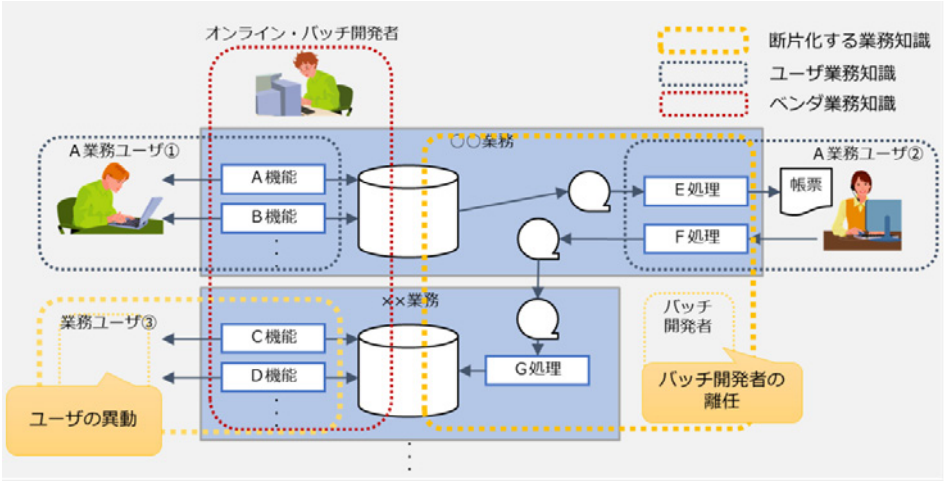


図 1.4 離任などによる業務知識の断片化

また、知識の断片化は業務だけではなく、図 1.5 に示す通り、基盤機能や制御機能でも同様に発生する。特にメインフレームシステムでは、OS やミドルウェア製品が幅広い機能を提供していることで、業務要件上必須ではなくても機能を使用している場合がある。要員交代などによる基盤機能、制御機能の知識の断片化により、業務要件上本当に必要な機能かを誰も把握していない状態となる。

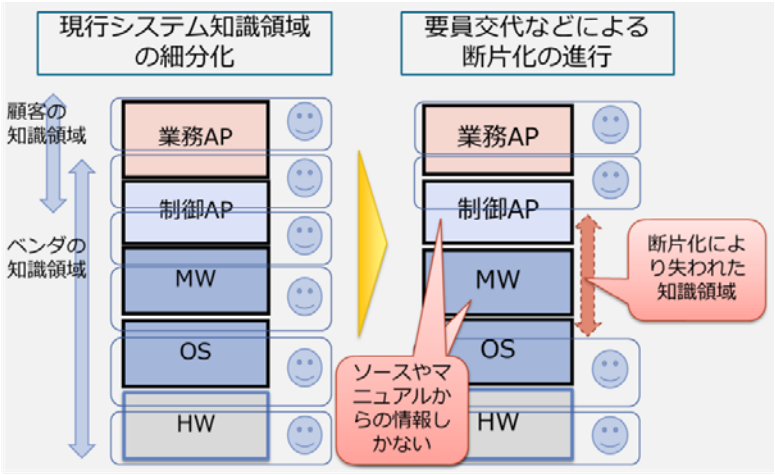


図 1.5 制御機能における断片化

(3) 業務知識の領域の変化

長期間に渡るシステムの運用において、システムは様々な変化に対応しなければならない。

初期構築時に全ての仕様変更を予測することは不可能であり、ビジネス環境の変化や技術環境の変化に合わせて、システムは仕様変更を繰り返す。そのようなシステムに加えられた変化により、業務知識の領域が変化する。

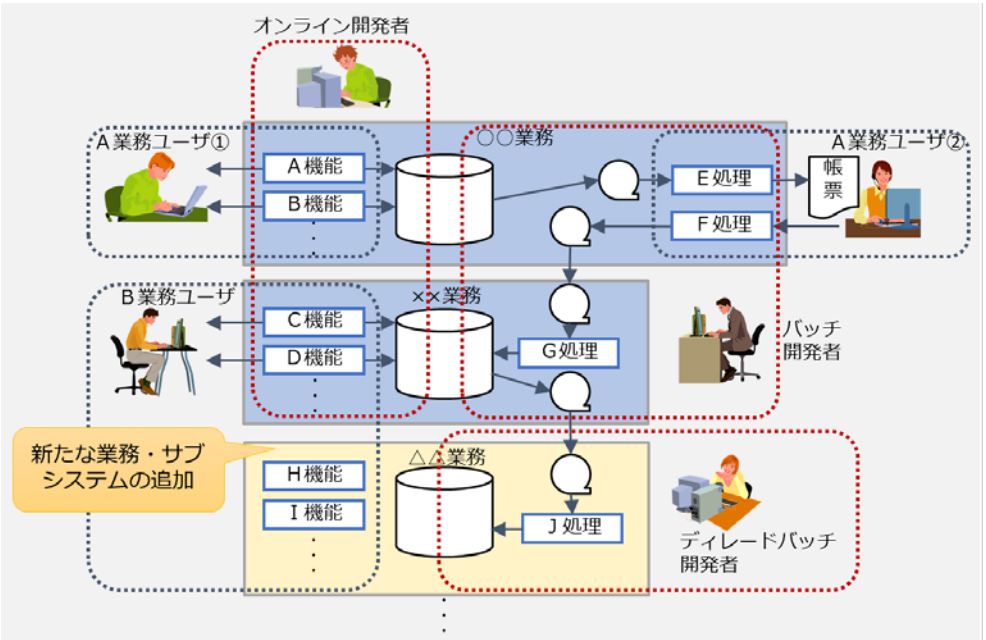


図 1.6 業務知識の領域の変化

1.2 再構築の現場で何が起こっているか

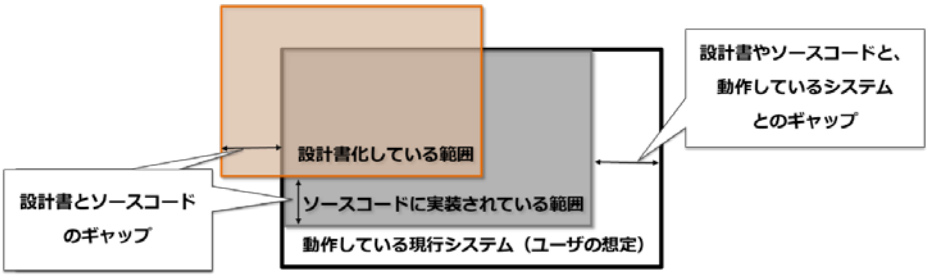
再構築の現場で起こりうる問題

前述のように、ユーザ企業側で業務知識が細分化され、長期にわたる維持保守において業務知識が断片化した状態で、知識領域が変化した影響の見極め不足のまま再構築を進めようとすると、工期・コスト・品質に関わるトラブルにつながる。特に「現行踏襲」と「品質保証」で問題が現れやすい。

「現行踏襲」に関する問題

再構築に際して、「現行踏襲」という要求が発生することは多い。「現行踏襲」を実現する上で注意しなければならないのは、「現行踏襲」の「現行」が何か、ユーザ企業とベンダ企業との間でギャップが発生する場合があるという点である。

ユーザは「動作している現行システム」がそのまま踏襲されることを期待している。一方ベンダは実際に開発するためには「現行通り」を仕様化する必要があるため、設計書などのドキュメントに記載されている要件や仕様、またはソースコードに実装されている内容を「現行踏襲」の拠り所の一つとする。このギャップは、以下のような現象として表れる。



設計書とソースコードのギャップ
・設計書が更新されておらず、記載内容が古い状態となっている

設計書やソースコードと、動作しているシステムとのギャップ
・製品の提供する機能で実現されている機能が、設計書には明記されていない
・ソースコードで実装するのではなく、運用保守担当者の手作業や独自ツールなどによって実現しているしくみが設計書に明記されていない
・設計書に記載されているシステムの設計の時限と、実際の運用フローの時限に乖離がある(帳票の出力時間など)

図 1.7 「現行踏襲」の拠り所のギャップ

前述したように、業務知識は時間経過とともに細分化された状態からさらに断片化、領域変化していき、ドキュメントとソースコード、動作しているシステムの間で乖離が発生する。例えば、仕様変更を実施した際に、変更の契機となった業務についてはドキュメントとソースコードの双方を更新したが、関連する業務のドキュメントは更新が必要であることが分からず更新しなかったといったケースや、障害対応時にソースコードを修正したもののドキュメントを修正しなかったといったケースがある。

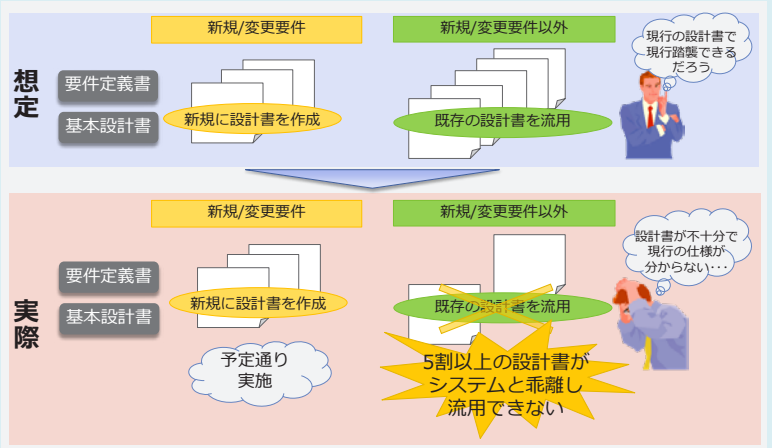
ソースコードに実装されているものとドキュメントに乖離がある場合、どちらの情報で実現したいのかを見極めないと誤った「現行通り」になってしまう。また、システム全ての仕様がドキュメント化されていないことが原因となり、ユーザ企業とベンダ企業の両者が拠り所とする「現行踏襲」にギャップが生じてしまうこともある。ユーザ企業とベンダ企業の間だけではなく、ユーザ企業の中でも、システム部門と利用部門とでは、ギャップが生じるケースもある。

上記の状態で再構築を行うと、トラブルになりやすい。それに加え、再構築と同時に業務要件の変更や追加を行うと、現行踏襲部分と業務要件の変更や追加をした部分との整合性の確保が難しくなる。その結果、品質保証の難易度が非常に高くなり、一層トラブルが生じやすくなる。

＜トラブル事例 1：設計とソースコードのギャップ＞

明確に要件定義したのは新規／変更要件のみで、現行システムの要件定義書・基本設計書が存在することを前提にした「現行踏襲」での開発の方針としたが、蓋を開けると 5 割以上の設計書がシステムと乖離しており、有識者も存在しないことが発覚した。

現行仕様が不明確な状態で開発を進めた結果、現行で使用している機能の実装漏れや、ユーザが期待する現行通りの処理結果との不一致が試験工程で多発し、新システム稼働後も品質改善を継続することとなった。



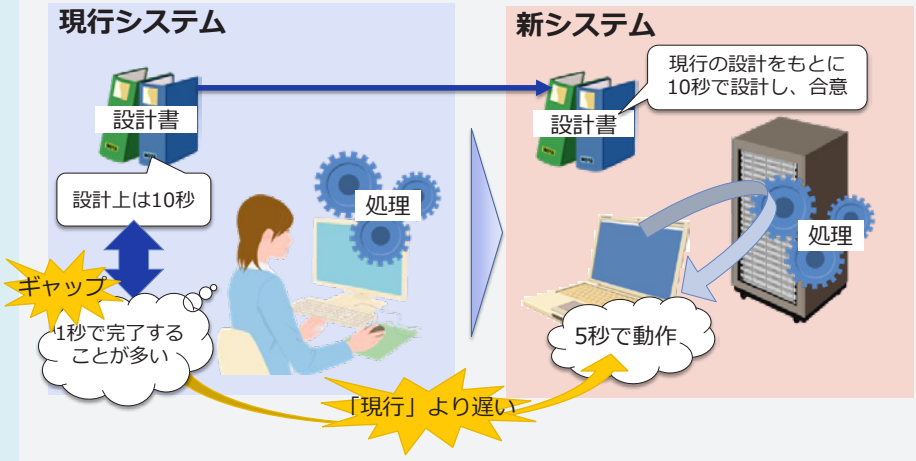
設計書やソースコードのギャップのイメージ

＜トラブル事例 2：設計書やソースコードと動作しているシステムとのギャップ＞

現行システムではクライアント端末上で動作していた処理を、新システムではサーバ上で動作させつつも画面表示や処理内容、処理性能は「現行踏襲」とし、その仕様について利用部門、運用部門、システム部門、ベンダ間で合意した。

処理性能について、現行システムでは 10 秒を目標とした設計となっており各種ドキュメントにもそのように記載されていたが、実運用では運用時間帯によって処理時間にばらつきがあり、実際には 1 秒で処理できることが多かった。

新システムでは、現行システムの性能設計に基づき最大 10 秒で処理が完了する仕様にて、ステークホルダ内で合意した。完成したシステムでは、ピーク時を除くと 5 秒で動作したが、現行システムでは 1 秒で処理できるケースが多かったことから、現場のオペレータは新システムでも 1 秒を期待していた。その結果、現行システムより遅く業務が滞るため改善してほしいという要望が寄せられ、当該処理を作り直すことになり、予定していたサービス開始が延期される事態となった。



設計書やソースコードと動作しているシステムとのギャップのイメージ

「品質保証」に関する問題

再構築の品質保証では新規開発と異なり、「業務継続性の担保」が求められることに特徴がある（「業務継続性」の詳細は後述する）。

しかし、どういう状態になれば業務が継続できると言えるのか、および、どういう確認をどこまで行えば業務継続性を担保できると考えるかは、業務の重要度やステークホルダによって異なり、また再構築プロジェクトに掛けることができる予算や期間によっても左右される。そのため、業務継続性の担保の方針は、再構築対象のシステムのオーナーであるユーザ企業が検討し判断する必要がある。しかし、現行業務知識が失われていると、その検討や判断ができなくなってしまう。

また、品質保証の取り組みとしてテストを実施する際には、以下のような問題が発生する。

① テストをどこまで実施する必要があるかの判断が困難

現行の仕様が把握できていないことから、処理ルートやデータバリエーションの全量が分からないため、新規開発のようにテストの網羅性によって品質を担保することができない。テストをどこまで実施すれば品質を担保できたと言えるのかについて、あらかじめ方針を決めておかないと、どういう状態になればテストを完了してよいのか判断できず、延々とテストを実施し続けるという事態に陥ることがある。

② テスト項目の作成が困難

現新比較テストやシステムテスト以降のテストでは、業務の代表的な処理を中心に確認するケースがある。しかし、テストで不具合が多く発生したために、品質改善のテストを実施することがある。例えば、代表的な処理以外に確認範囲を広げる、または確認観点の粒度をより詳細にすることがある。しかし、現行の仕様が把握できていない場合は、どのようなバリエーションのテストをすればよいかが把握できず、テスト項目を作成することも困難となる。

③ トラブル時に解析が困難

再構築では品質確認手段として、現行システムから取得した実データをインプットに、現行システムと新システムで同じ処理を実行し、そのアウトプットを比較することがある。現行システムと新システムで同じ動作結果となっているかを確認する現新比較テストである。テスト方法はブラックボックステストで行うため、テスト項目作成時や現新比較テストの結果に不一致が起こらない間は現行の仕様は把握できていなくても問題にならない。しかし、現新不一致が発生した場合、現行の仕様が分からないと不一致の原因解析が難航し、テストの遅延につながる。

＜トラブル事例＞

大部分がリホストの前提で、通常の開発のようなV字モデルの品質積み上げを実施せず、現行踏襲部分の品質は現新比較テストで担保する方針とした。現行仕様の分かる設計書・ドキュメント類は存在しない状態だったが、テストに必要となる業務知識、システムの知識は現行保守ベンダが把握しているという前提であった。

テスト着手後、現行仕様から追加・変更した箇所で仕様漏れが発生したことにより、品質強化のために追加のテストが必要となったが、テスト項目のインプットとなる設計書類が存在せず、また期待していた現行保守ベンダもシステム全体にわたる業務知識は有していないことが判明した。そのため、必要なテスト計画が立てられず、数年分のサイクルテストをすることで対応する方針となったが、テスト期間が大幅に延伸し、システムのリリースも延伸した。

参考情報

上記のような背景から、一般社団法人プロジェクトマネジメント学会（以下、PM学会）でも、レガシーシステムの再構築をいかに実現し、品質を確保するかというテーマが度々発表されている。

- 2015年度「レガシーモダナイゼーションプロジェクトにおける提供品質確保の取り組み」[2]
- 2016年度「新旧比較照合テスト自動化による,マイグレーション開発の品質保証」[3]

など

1.3 再構築の問題化を防ぐために

本ガイドの目的

前述のような問題を避けるためには、現行システムの状態を正しく認識した上で再構築の方針を決め、実行計画を立てることが肝要である。

「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～」[5]では、システム開発の成功のためには、事業や業務検討の始まりから要件定義までの、「超上流」工程におけるユーザ企業とベンダ企業の双方の取り組みが重要であることを述べている。

同書では新規開発を想定しているが、再構築でも同様に「超上流」工程が重要である。すなわち、企画段階で現状認識（現行システム資産の量・質、有識者の有無など）を正しく行い、再構築の目的を基に再構築後のシステムに求められる要求事項から求められる要件や機能などを整理し、それらを基に、一定の品質に到達させるために必要な作業を計画し、それに係る工期・コスト・リスクを明確化する。それらについて、ユーザ企業とベンダ企業双方で共有した上で開発を進める。（本ガイドで使用している工程区分は、図 1.9 を参照）

本ガイドでは図 1.8 に示すように、過去の問題化案件の知見から抽出した、再構築の企画段階で検討すべき観点を説明する。各観点で検討すべき内容は、再構築手法（例えば、リホスト・リライト・リビルドなど）によって変わるため、検討の前段として、どういった手法で再構築を実現するかを選定することが重要である。

このため、再構築の目的・要求事項や現行システムの状況から最適な再構築手法を選択しリスクを洗い出す流れを 2 章「再構築手法選択編」に、選択した手法とそのリスクをもとにリスクの予防策を検討する観点を 3 章「計画策定編」に分けて説明する。

2 章および 3 章の内容を企画段階で取り入れることで、要件定義以降の工程で発生する恐れがある問題の予防につなげてほしい。

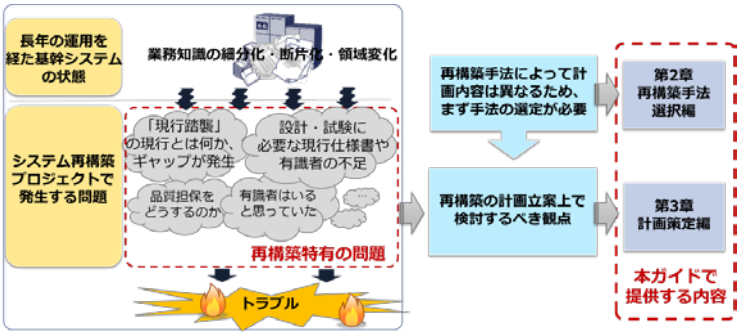


図 1.8 本ガイドの内容

本ガイドの利用について

(1) 利用シーン

システムの再構築の企画段階において、再構築の構想、計画を検討する際に利用する。図 1.9 に示す工程区分（本ガイドで前提とする分け方）の中の企画工程で利用し、この企画工程での計画内容をもとに、RFP の作成や開発ベンダのプロジェクト計画の策定が行われると想定している。

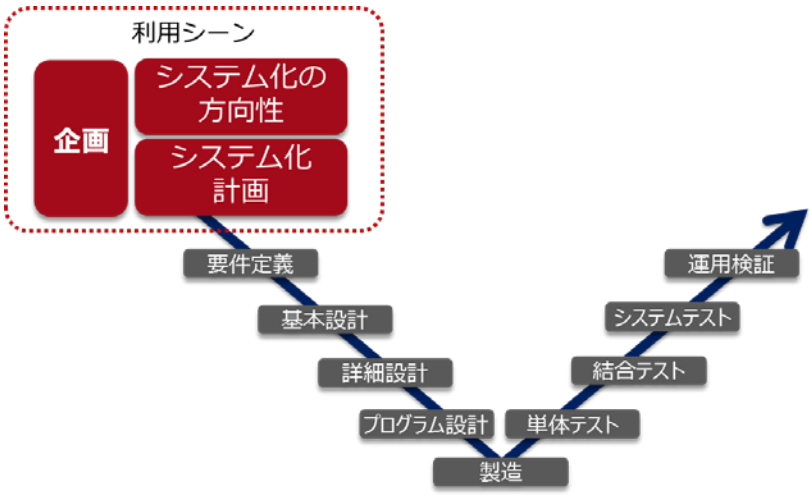


図 1.9 本ガイドの工程区分

(2) 利用者

システム化計画の策定に関わるユーザ企業を対象としている。特に、次期システムを企画する立場の読者（ユーザ企業の経営者、システム部門、利用部門全体）に利用されることを想定している。

再構築の経験がある、または勉強会・セミナなどで再構築に関するノウハウを収集している読者には、再構築で必要となるタスクや難しい点についての再認識や、従前の知識にない情報の取得のために活用いただきたい。

また、再構築の経験や知識があまりない読者には、本ガイドを通じて再構築で必要となるタスクや難しい点について気付きを得てもらえると幸いである。

なお、システム化計画の策定に関わるベンダなども、セルフチェックのために利用できると考えている。

(3) 利用方法

本ガイドで説明する内容は、システム再構築に特有な事項に限定しており、新規開発でも求められるものは触れていない。

実際のシステム化の方向性、システム化計画の立案では、図 1.10 に示すように共通フレーム 2013[7]や PMBOK[8]、自社で保有する標準類などをもとに検討した計画内容に、必要に応じて、本ガイドをもとに検討した内容を追加する形で利用してほしい。

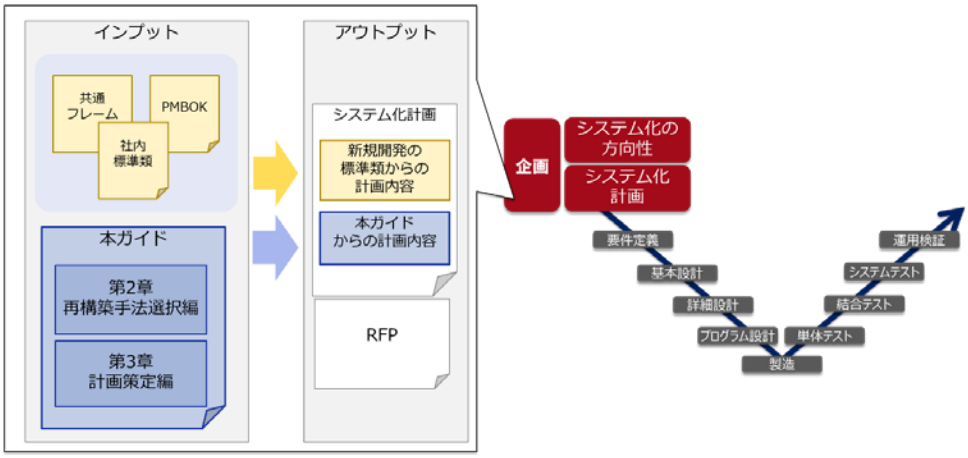


図 1.10 ガイドの利用イメージ

(4) 本ガイドが対象とする再構築のケース

- アプリケーションのみを変更するケースは対象外とし、現行システムの基盤を含め刷新するケースを対象とする。

- 再構築の方向性は、大きく図 1.11 の 3 パターンに分かれる。本ガイドは現行システムを（全部または大部分）踏襲する場合に取り得る「パッケージ製品の利用」、「HW 更改・リホスト・リライト・リビルド」を対象とする。

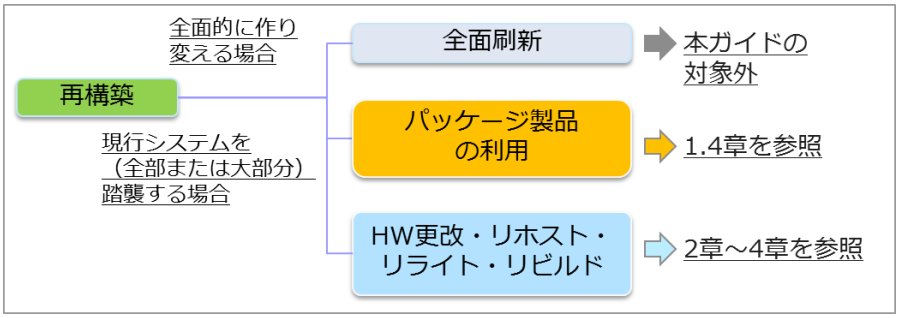
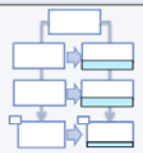
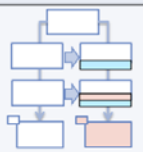


図 1.11 再構築の方向性

「パッケージ製品の利用」は、「HW 更改・リホスト・リライト・リビルド」とは企画・計画段階のタスクを実施するタイミングが異なること、およびパッケージ製品を利用する場合の特徴的なタスクが存在するため、「1.4　再構築にパッケージ製品を利用する場合」にまとめている。

なお、「HW 更改・リホスト・リライト・リビルド」で業務要件の変更／追加を実施するケースのリスク対策については、「3.10　業務要件の変更／追加への対応」にまとめた。2 章および 3.1 節～3.9 節に加えて参照してほしい。

＜注意点＞
「HW 更改・リホスト・リライト・リビルド」の定義については、企業や書籍によって異なるが、本ガイドでは、再構築でのアプリケーションの移行に際して、現行システムのどのレイヤの設計情報を維持するかによって図 1.12 の 4 種類に分類している。

再構築手法	ハードウェア更改	リホスト	リライト	リビルド
概要	ハードウェアの変更を行う。OS、ミドルウェア製品のバージョンが変わる影響は、プログラムの修正が発生する場合もある。	プログラムは現行と同一の言語で、原則そのまま新規プラットフォームへ移行する。	現行のプログラム設計書をもとに、異なる言語で新たなプログラムを実装する。	現行の要件定義書をもとに新規システム構築。業務要件は変えずに、アプリケーションを事実上作り変える。
要件定義 基本設計／詳細設計 プログラム設計 プログラムソース				
要件定義書	現行流用	現行流用	現行流用	現行流用
基本設計書／詳細設計書	現行流用 （OS、ミドルウェア製品のバージョン変更に伴う修正の場合あり）	現行流用 プラットフォーム変更に伴う修正あり	現行流用 プラットフォーム変更に伴う修正あり	再設計
プログラム設計書	現行流用 （OS、ミドルウェア製品のバージョン変更に伴う修正の場合あり）	現行流用 プラットフォーム変更に伴う修正あり	現行流用 プラットフォーム変更に伴う修正及び言語依存、新規機能分再設計要	再設計
プログラムソース	現行流用 （OS、ミドルウェア製品のバージョン変更に伴う修正の場合あり）	現行流用 プラットフォーム変更に伴う修正あり	再生成	再生成
補足	ハードウェアを同一機種、または後継機への乗り換えすることに伴い、OS、ミドルウェア製品のバージョンが上がることはあるが、製品の種類は変更しない	「現行と同一の言語」について、例えばメインフレームCOBOLからオープン系COBOLに変わるケースはこちらに含む	－	－

<凡例> ：非互換の変更 ：再作成

図 1.12 再構築手法の分類

ポイント！

再構築プロジェクトを通じて、新システムでは、業務知識不足や、ドキュメントやソースコード・動作しているシステムの間での乖離の解消が行われると考えられるが、新システムでも再び「1.1 システム再構築をとりまく現状」で述べたようなレガシー化が進むおそれがある。本ガイドの対象は再構築プロジェクトの企画段階で検討すべき内容であるため説明はしていないが、新システム稼動後の再レガシー化を防止するような取り組みを検討することも重要である。

1.4 再構築にパッケージ製品を利用する場合

目的

再構築の計画時に、開発コストダウンや自社保有のアプリケーション資産の削減、業界標準の適用などを目的に、パッケージ製品を組み込む例は少なくない。

パッケージ製品の利用では、製品が標準として提供する機能をできるだけそのまま利用することが望ましい。それによって、「既製品」を利用するメリットである、高品質・短期導入・低コストが実現できる。

とはいえ、パッケージ製品では実現できない業務運用や機能について、事業の遂行上必要なものも存在する。パッケージ製品を拡張して自社独自の業務運用や機能を実現する場合も多い。再構築プロジェクトの企画・計画段階で製品の Fit&Gap 分析が十分に行われず、プロジェクト開始後になって想定以上にパッケージ製品の拡張が必要であることが分かり、計画時よりもコスト・スケジュールが膨らんでしまった事例は少なくない。

上記は新規開発でパッケージ製品を利用する場合でも共通する事項である。再構築では、システム化されている慣れ親しんだ現行の業務運用があるがゆえに、それをパッケージ製品に合わせて変更するという事は難しい。

再構築プロジェクト当初は、パッケージ利用によるコスト削減やスケジュール短縮の目標の下、パッケージに業務運用を合わせるトップダウンの方針で進めていた。しかし、プロジェクトが開始し具体的な設計が進む中で、現行の業務運用と同じにしたいという要望が現場から上がることはよくある。その結果、パッケージ製品の機能拡張規模が増大していくといったケースに陥りやすい。

こうした事態を回避するためには、企画・計画段階で現行の業務運用に対するパッケージ製品の Fit&Gap 分析とユーザ部門への確認を行い、パッケージ製品の利用可否を十分に吟味することが重要である。また、次期システムでの業務運用の変更点について早期にユーザ部門と認識を合わせることも肝要である。

本節では、再構築でパッケージ製品を利用する際のタスクと注意点について説明する。

＜トラブル事例 1：Fit&GAP 分析の不十分さによるトラブル＞

顧客からの振込み・入金通知などを行う業務に対し、コストダウンを目的に、パッケージ製品を導入することになった。業務運用を現行通り実現できるかについては Fit&Gap 分析を十分行ったことで問題はなかったが、非機能面について Fit&Gap 分析が不十分であった。そのため、将来の顧客数の増加を見据えた場合、性能要件が満たせないことがシステムテストにて判明した。

結果的に処理方法を見直すこととなり、設計から試験までを一から実施し、スケジュールが遅延した。また、パッケージ自体の修正を行ったことにより、新規開発と同等の開発コストがかかった。さらに、パッケージを自社でメンテナンスすることになり、保守コストが増大した。

＜トラブル事例 2：現行業務にこだわったことによるパッケージ導入メリットの低下＞

再構築に際し、財務部門の業務に対して会計情報システムのパッケージを導入することとなった。しかしパッケージの基本提供機能だけでは、その会社独自の財務帳票が出力されないことが Fit&Gap 分析により分かり、アドオン開発で対応することとした。開発・維持コスト削減のためアドオン規模は最小限に抑えるという方針から、財務帳票の出力以外についてはパッケージの基本機能を活用し、業務運用フローを変更することとした。

しかし、ユーザテスト中に、帳票出力だけでなく、途中の画面確認などの運用も現行通りにするよう財務部門から要望があがった。従来通りの業務運用を維持したいとの強い要請を受け、画面出力機能のアドオン開発を追加することになったが、その結果、パッケージを導入することで期待した開発・維持コストの縮小が実現できなくなった。

本節の対象

本節は、現行システムを再構築するにあたりパッケージ製品を利用するケースを対象としている。

本節の対象とするパッケージ製品とは、業務運用を実現するためのパッケージ製品を指しており、メインフレーム機能の互換製品といった基盤パッケージ製品は含まない。

Fit&Gap 分析とは

「目的」で述べたように、パッケージ製品の利用にあたっては、企画・計画段階で現行の業務運用に対するパッケージ製品の Fit&Gap 分析とユーザ部門への確認を行い、パッケージ製品の利用可否を十分に吟味し、また次期システムでの業務運用の変更点について早期にユーザ部門と認識を合わせることが重要である。

現行の要件や仕様とパッケージ製品の標準機能を比較して適合度を調査し、適合しない要件や仕様については、実現の方針を検討する。(図 1.13)

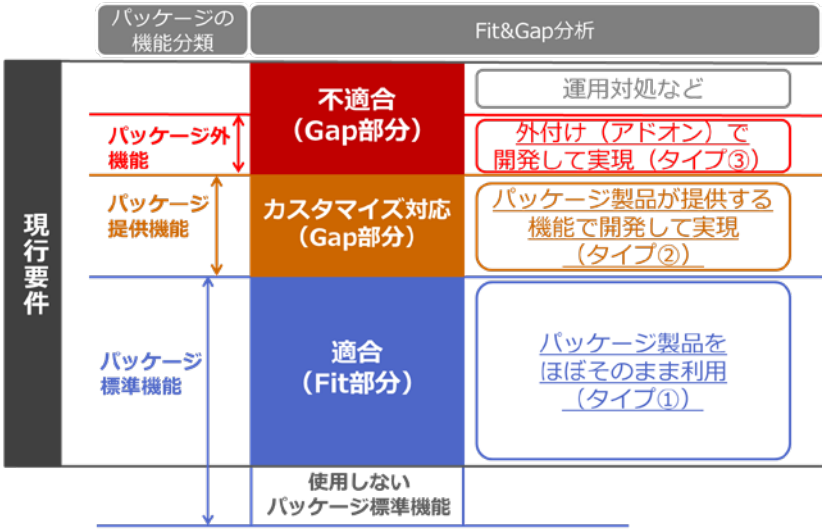


図 1.13 Fit&Gap 分析のイメージ

図 1.13 中のパッケージ製品の利用形態（タイプ①～③）の定義は以下の通り。

- タイプ①：パッケージ製品をほぼそのまま利用する
  製品のパラメタ定義の設定をする程度で、製品をそのまま利用する。

- タイプ②：パッケージ製品が提供する機能を用いてカスタマイズする
  パッケージ製品が提供する開発言語・API・環境を用いて、企業独自の業務ロジックを作りこむ。

- タイプ③：パッケージ製品に外付け（アドオン）機能を開発する
  独自に開発した機能や、別のパッケージ製品を外付け（アドオン）して必要な機能を実現する。

ポイント！

適用先システムが必要とする機能を実現するために、パッケージ製品自体のソースコードを書き換える場合がある。ただし、以下のような理由から本ガイドでは推奨しない。

- パッケージ利用のメリットである高品質・短期導入・低コストが損なわれる。
- 維持保守フェーズでの機能追加や修正の際に、パッケージ製品部分の品質保証を自社で行うことが求められる。
- OS のバージョンアップ対応など、パッケージ製品のバージョンアップを自社で行うことを求められる。

パッケージ製品利用時のタスク

再構築でパッケージ製品を利用する場合の特徴的なタスクを図 1.14 に示す。

フェーズ	タスク	タスク概要
企画・計画 (本ガイドの対象)	1.パッケージ製品利用目的の明確化	パッケージ製品利用の目的を明確にする
	2.パッケージ製品候補、製品提供者の選定	利用するパッケージ製品および製品提供者の候補を選定する
	3.Fit&Gap分析（企画・計画）	パッケージ製品の採否を決定し、見積りを算出するためのFit&Gapを行う
	4.パイロット検証	標準機能によるプロトタイプを作成しユーザとの早期意識合わせを行う
	5.パッケージ製品の確定	Fit&Gap分析、プロトタイプ作成の結果を受け、投資効果や開発スケジュールなどから、利用する製品を確定する
	6.品質保証の考え方、検証計画の検討	パッケージ製品利用時の品質保証の考え方を検討する
	7.利用プロセスの立案	パッケージ製品利用時の運用方法、利用手順整備の計画を立案する
要件定義	Fit&Gap分析（要件定義） パイロット検証（詳細）	開発するアドオン機能、利用するパッケージ製品機能を具体的に検討するためのFit&Gapを行う
設計 製造 試験	パッケージ製品パラメタ設計・設定 アドオン部の設計・製造 試験	左記のとおり

この段階で、利用するパッケージ製品を確定し、概算見積りを算出する

図 1.14 再構築でのパッケージ製品利用時のタスク

企画・計画段階の各タスクについて、以下に説明する。

（タスク 1）パッケージ製品利用目的の明確化

再構築の目的と新システムへの要求事項から、システムの将来像を整理し、それらの実現手段としてパッケージ製品を利用する方針で進めるのか、利用しない方針（リホストなどの手法）で進めるのかを検討する。

パッケージ製品を利用する方針に決定した場合、決定した根拠（目的や期待効果など）を明確化しておくことが重要である。それにより、後続のタスクでパッケージ製品利用の実現性やコストなどの評価を行った結果、当初の目的に合致しない場合に、方針の見直しを行う基準とすることができる。

（タスク 2）パッケージ製品候補、製品提供者の選定

前タスクで整理した再構築の目的と新システムへの要求事項を満たすと想定されるパッケージ製品の候補を、一つまたは複数選定する。あわせて、製品提供者（製品ベンダ、SI ベンダ）を選定する。選定の理由や根拠は、明確にしておく。

（タスク 3）Fit&Gap 分析（企画・計画）

現行の要件や仕様とパッケージ製品を比較し、パッケージ製品を利用して業務を実現することが可能かどうか調査する。

Fit&Gap 分析は、企画・計画工程と要件定義工程のそれぞれで実施する。企画・計画工程では、パッケージ製品を決定し、プロジェクト計画立案と概算見積り算出のための分析を行う。要件定義工程は、企画・計画工程での Fit&Gap 分析結果をもとに設計に向けた内容を検討する。

本ガイドの利用シーンである企画・計画工程で実施する Fit&Gap 分析（企画・計画）のアプローチの考え方について特に説明する。図 1.15 は Fit&Gap 分析のアプローチのイメージを表したものである。ステップ 1、2 の順に現行要件（仕様）の実現可否と対応方針を検討していき、各タイプの割合と規模を概算で見積る。

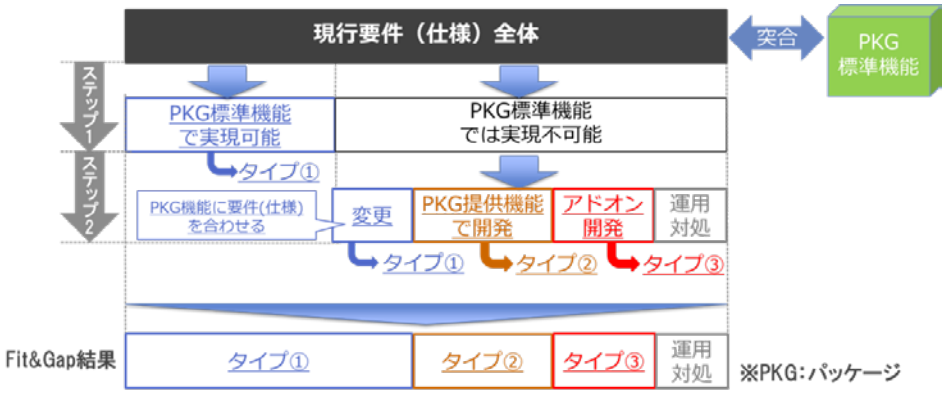


図 1.15 Fit&Gap 分析（企画・計画）のアプローチ

(1) ステップ 1

現行要件・仕様に対して、パッケージ標準機能で実現可能か、机上確認またはパイロット検証により調査する。

この際、パッケージに置き換えるシステム内の調査だけではなく、パッケージによって当該システムと外部システムのインターフェースが変わるか否かも確認すること。インターフェースの変更を外部システム側が許容できない場合、アドオン開発によって対応する必要が生じるためである。

(2) ステップ 2

ステップ 1 でパッケージ標準機能では実現不可能となった要件・仕様に対して、「業務継続性」の観点（「3.6 品質保証の検討（観点 E)」参照）から要否を判断し、以下のように対応方針を検討する。

- パッケージが提供する機能を用いた開発で実現
- アドオン開発で実現
- 製品および作り込みではなく、システムを利用しないオペレータ操作などの運用対処により実現
- 現行から要件・仕様を変更し、パッケージ標準機能に合わせる

パッケージ利用の期待効果とリスク回避には、現在の業務運用を極力パッケージに合わせていくことがポイントとなる。企画・計画段階で十分に調査しないままでプロジェクトを進めたため、要件定義工程の Fit&Gap 分析で精緻な分析を行った結果、カスタマイズやアドオンの規模が増大しプロジェクト期間やコストが超過したといったケースは多い。そのため、企画・計画工程でパッケージの持つ機能をできる限り正確に把握し、Fit&Gap 分析不足による作り込みの膨張を防止することが重要である。

（タスク 4）パイロット検証

ユーザ部門と開発部門の認識相違を早期に解消するため、資料や口頭ベースでの確認だけでなく、標準機能によるプロトタイプを作成し実機で確認することが望ましい。

（タスク 5）パッケージ製品の確定

Fit&Gap 分析（企画・計画)、プロトタイプ検証の結果、およびパッケージ製品や製品提供者に対するその他の評価（図 1.16）をもとに、開発スケジュールや投資効果を検討し利用するパッケージ製品を決定する。

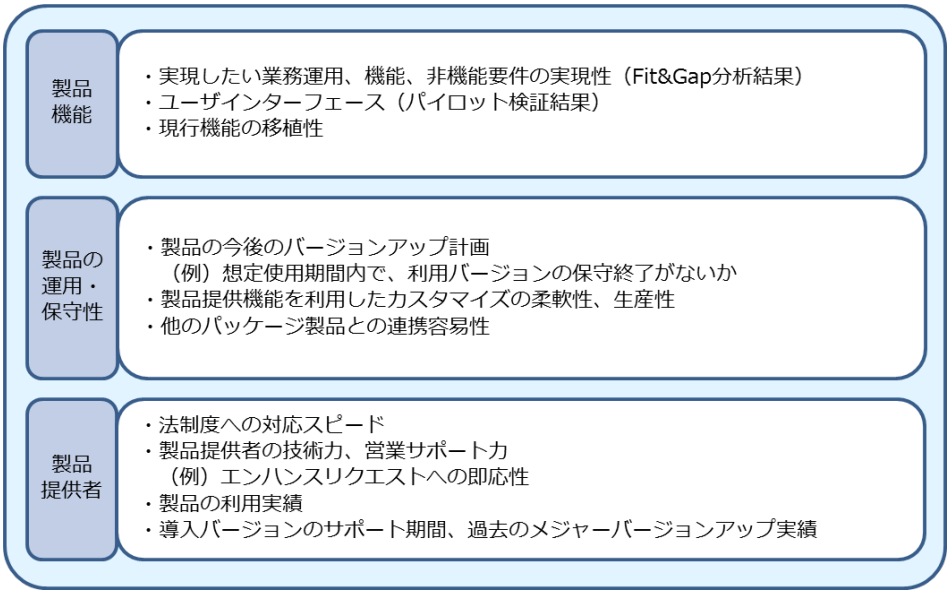


図 1.16 パッケージ製品の評価観点（例）

開発時だけではなく、その後の維持保守も鑑みた検討が必要である。（後述の「パッケージ製品利用時の注意点」を参照）

また、パッケージ製品との「Gap」を埋めるために必要となる追加開発の規模が大きい場合などは、パッケージ製品を利用しないという判断も必要である。その場合、本ガイド 2 章および 3 章を参照し、リホストなどの手法での再構築を検討すること。

（タスク 6）品質保証の考え方、検証計画の検討

基本的に新規開発でパッケージ製品を利用する際と同様である。ただし、現行システムを踏襲する業務運用・機能については、現行と同様であることを確認する必要がある。この点については、「3.6 品質保証の検討（観点 E）」中の「業務継続性の担保」に関する説明を参照のこと。

（タスク 7）利用プロセスの立案

パッケージ製品利用時の運用方法、利用手順整備の計画を立案する。現行の運用方法や利用手順からの変更が見込まれる場合、ユーザとの認識合わせを早期に行い、プロジェクト終盤で認識相違が判明し機能変更などが発生しないようにする必要がある。

パッケージ製品利用時の注意点

上記までの内容に加え、以下のような注意点がある。

(1) 「作り込み」が膨張しすぎないようにコントロールする

パッケージ製品提供の機能を用いた作り込み(タイプ②)やアドオン開発(タイプ③)は、その規模が膨大になると、開発コストやスケジュールが膨らみ、当該システムを作り直しすることと大差なくなってしまう。自社固有の業務運用の実現要求とパッケージ製品を利用するメリットとを比較し、開発規模を可能な限り抑えることが望ましい。

また開発を進める中で、当初は製品機能を利用する予定であった業務や機能について、現行通りとするために追加開発をしてほしいという要求が発生することがある。パッケージ利用の目的や期待効果を達成するために、本当に必要な要求に絞るようコントロールすることが重要である。

製品機能の利用と整理していた業務や機能に追加開発を行う等、プロジェクト着手時に定めた方針等を大きく見直す必要が出てきた場合、プロジェクト関係者だけで要求の取込み可否を判断せず、意思決定プロセスを利用し、積極的に経営層の参画を促して判断することが重要である。(「3.7 意思決定プロセスの策定(観点F)」参照)また当初の方針を変更し追加開発を行う場合は、その決定を行った記録を残すことも必要である。

(2) 開発時だけでなく、維持保守フェーズも考慮した検討をする

維持保守フェーズで追加開発や OS・ソフトウェア類のバージョンアップ対応を行う際に、パッケージ製品提供の機能を用いた作り込み(タイプ②)やアドオン開発(タイプ③)も対応が必要になる。

作り込み量が多く複雑であると、対応が煩雑になってしまったり、対応できるだけの要員を抱え続ける必要が生じたりし、保守コストが増加する可能性があるため、維持保守を考慮した開発を検討する必要がある。

第２章　再構築手法選択編

2.1 再構築手法選択編概要

再構築手法選択編とは

本ガイドでは、再構築手法の選択をシステム再構築企画のスタートラインと位置づけている。システムの稼動状況や再構築プロジェクトの状況に応じてリビルド、リライト、リホスト、ハードウェア更改の 4 つの中から最適な再構築手法を選択するプロセスを紹介している。プロセスは「再構築手法選択編における各ステップの概要」で紹介する 4 つのステップに分かれている。各ステップの作業はユーザ企業を主体に、必要に応じてベンダ企業のサービスや技術支援を受けて進める。

また各ステップでは作業に加えてアウトプットを定義している。このアウトプットには再構築手法を選択した条件や判断情報が含まれている。従って手法選択の経緯、リスクや合意条件などをドキュメントとして残すことができる。

なお、再構築手法の選択は、図 2.1 に示す通り、システム開発の工程区分の中では企画段階でシステム化の方向性を決定する工程に当たる。

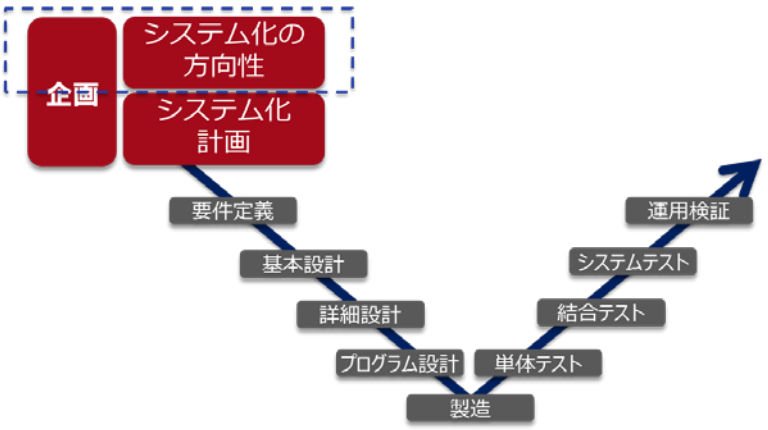


図 2.1 再構築手法選択編の利用シーン（システム開発の工程区分での位置づけ）

狙い

再構築手法の選択条件や判断情報を明らかにして、再構築に伴うリスクやコストをユーザ企業とベンダ企業間で共有し、再構築の目的に適った再構築手法を選択する。

利用シーン

システム再構築の企画工程において、次期システムで採用する再構築手法の選択、選択した再構築手法のリスクを洗い出す際に利用する。なお再構築手法の選択は、各企業の現状やニーズに応じてシステムまたはサブシステムごとに再構築のテーマを決めて再構築手法の選択を行う。

利用者

システム再構築の企画工程において、システム開発計画の策定に関わる経営者、システム部門、利用部門、ベンダ企業などのステークホルダが利用する。

再構築手法選択のプロセス

再構築手法選択のプロセスについて説明する。このプロセスの特徴は、表 2.1 で紹介する再構築テーマを起点に、現行システムの状態と新システムの要件から適切な再構築手法を選択できることである。再構築手法はシステム単位またはサブシステム単位に再構築のテーマを決めて選択する。本ガイドでは 5 つの再構築テーマを提示している。この再構築テーマが該当しない場合は、個別に再構築のテーマを設定する。再構築手法選択のプロセスをフロー図として図 2.2 に示す。

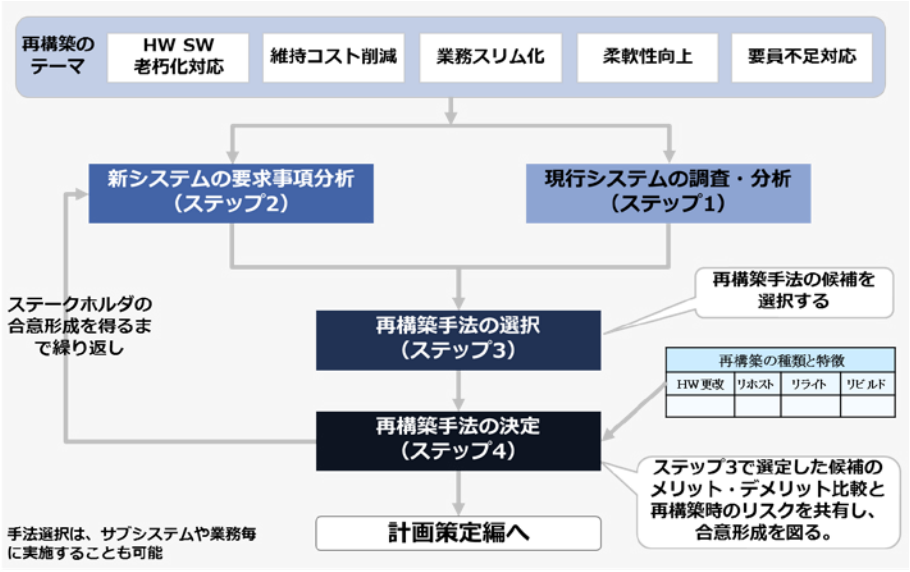


図 2.2 再構築手法選択フロー図

再構築手法選択編における各ステップの概要

本ガイドでは、手法選択の起点となる再構築のテーマを 5 つに絞り込んだ。再構築のテーマ一覧を表 2.1 に示す。また再構築のテーマについては各企業の現状やニーズに応じて、独自の目標や目的を設定し、追加、変更して使用することを可能としている。

表 2.1 再構築のテーマ一覧

No	再構築テーマ	テーマの意味
1	HW SW 老朽化対応	ハードウェア機種やソフトウェアの販売終了、サポート終了などへの対応
2	維持コスト削減	現行システムの維持管理コスト（運用・保守コストなど）の削減
3	業務スリム化	肥大化、複雑化した基幹システムのスリム化、業務改善、運用の改善
4	柔軟性向上	新しいビジネスとの連携や新サービスへの対応を容易とする柔軟性の向上
5	要員不足対応	現行システムの保守要員の高齢化や基幹システムの技術者の減少への対応

以下では各ステップの概要について説明する。

表 2.2 各ステップの概要

2章 再構築手法選択編	目的	インプット情報	アウトプット情報	作業概要
ステップ 1	現行システムの把握	現行ドキュメントや運用・保守の状況、有識者の知識など	現行システムの状態	様々な資産の調査や有識者へのヒアリング
ステップ 2	新システムに求める要求の決定	再構築のテーマ	新システム要求事項一覧	再構築の目的を設定して、要求の優先度を決める
ステップ 3	再構築手法の候補を選択	現行システムの状態および新システム要求事項一覧	再構築手法の候補	業務仕様、基盤、開発言語の変更有無を評価
ステップ 4	再構築手法の決定	再構築手法の種類と特徴	決定した再構築手法と根拠	投資効果確認およびリスク把握

（ステップ 1）現行システムの調査・分析

まず、再構築の際に必要となる現行システムの状態を正確に把握するための調査・分析を行い、現行システムの稼動状況と現行資産の実情を明らかにする。そのための観点は「2.2 現行システム調査・分析（ステップ 1）」で詳しく述べる。

調査・分析作業は、ユーザ企業で実施、あるいはユーザ企業がベンダ企業のサービスなどを活用して実施する。

（ステップ 2）新システムの要求事項分析

再構築の目的を達成するために、新システムへの要求事項の洗い出しと要求事項の優先付けを行う。これにより、新システムへの要求事項が明確になる。

（ステップ 3）再構築手法の選択

新システムへの要求事項（業務仕様や基盤の変更有無）と再構築後の効果および移行期間の組み合わせから、再構築手法の候補を 1 つ（場合によっては複数）選択する。

（ステップ 4）再構築手法の決定

ステップ 3 で選択した再構築手法を対象に、経営者を含む再構築に関わるステークホルダが協議して再構築手法を決定する。協議はステップ 1～3 のアウトプットに加えてステップ 4 で提供する予想されるリスクと再構築手法の特徴を使用して行う。

ここで決定した再構築手法に対するコスト、期間、リスクおよび、業務有識者の関与度合いについては、計画策定編の作業を経てユーザ企業とベンダ企業で合意する。

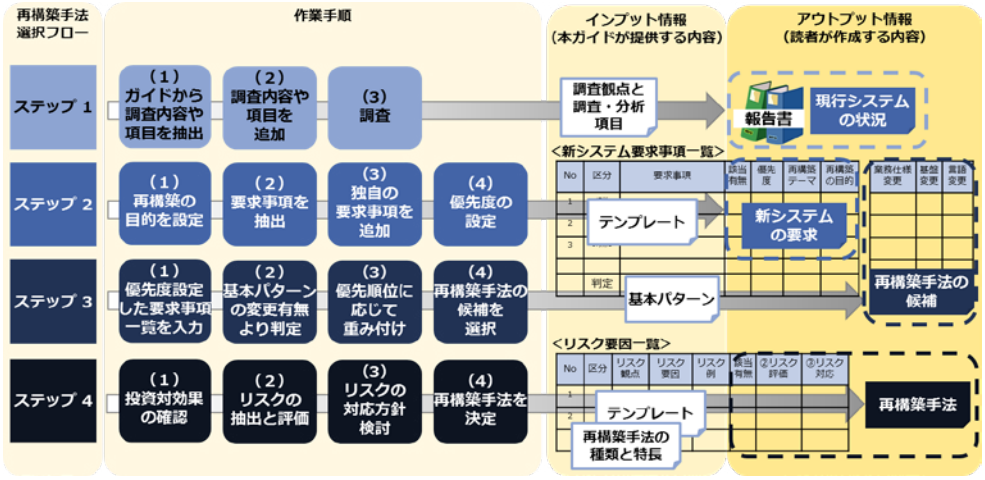


図 2.3 ステップの全体イメージ

インプット情報

各ステップでインプット情報を定義している。再構築手法を選択する情報の種類は、現行システムの稼動状況が分かる運用情報やインシデント情報、システム構成が分かる構成情報や資産情報、開発状況が分かる情報や開発体制、および新システムへの要求事項など広範囲に渡っている。

アウトプット情報

各ステップでアウトプット情報を定義している。再構築手法選択編のアウトプット情報としては、再構築手法だけではなく、その再構築手法に関わるリスク、および再構築手法の選択条件と判断情報も出力される。

2.2　現行システムの調査・分析（ステップ１）

目的

再構築において拠り所となる現行システムの状態を正確に把握するために、現行システムの調査・分析を行う。再構築するにあたっての現行システムの構成や規模などの基礎情報、再構築手法選択に際して必要となる業務知識（有識者やドキュメントなど）の状況を把握するためだ。現行システムの現状を正確に把握することによって、長年の運用によってブラックボックス化したシステムにおいて、これまで気づかなかったものを見える化することもできる。

また、再構築の拠り所となる現行システムの仕様や運用などを正確に把握することは、再構築工程に入った後での現行仕様の認識違いによる手戻りや品質問題化を防ぐことにもつながる。

ポイント！

現行システムの稼動状況や現行資産は膨大である。調査・分析作業を効率的に行うには、再構築のテーマや目的をもとに分類して進める。例えば、老朽化対応であれば、システム構成やソフトウェア構成を中心に調査分析する。維持コスト削減であれば、運用・保守の状況を中心に調査するといった具合だ。再構築のテーマに応じて調査・分析作業することが肝要である。

現行システムの調査観点と具体的な調査・分析項目の例を表 2.2 に示す。

表 2.2 現行システムの調査観点と調査・分析項目（例）

No	調査観点	区分	具体的な調査・分析項目の例
1	資産	システムの規模	業務毎の資産規模、本数、言語種別、画面数、帳票数と属性
2		構成	システム構成、ミドルウェア、データベース管理システム、運用ソフトウェア、文字コード体系
3		特性	アプリケーションの複雑度、類似度
4		稼動/未稼働	現運用で稼動している業務資産、未稼働資産の識別
5	業務知識	要員（質的な面）	有識者（業務仕様、基盤、運用・保守など）のスキルセット
6		要員（量的な面）	再構築時に必要な有識者（業務仕様、基盤、運用・保守など）を揃えられるか、限定されるか
7		ドキュメント	業務ドキュメントの有無（業務フロー、機能仕様書、処理フローなど）
8		ドキュメント	上記のドキュメントの最新性が保たれているか
9	処理方式	オン/バッチ	アプリ動作環境、オンライン処理方式、バッチ処理の形態、DBアクセス方法、帳票出力方法
10		外部接続	外部接続先、メッセージ連携、ファイル連携方法、コード変換の有無、種類
11		バックアップ	バックアップの対象、種類、タイミング、リカバリの方法
12	非機能要件	可用性	現行システムの稼働時間、停止時間、稼働率
13		運用・保守性	運用時間、運用監視、異常時運用、運用スケジュール定義方法、システムオペレーション方法
14		性能・拡張性	オンラインレスポンス、バッチ処理時間、スループット、長時間ジョブ情報
15		信頼性	冗長構成の有無、多重化の対象
16		セキュリティ	システム利用に対する有資格者のチェック方法
17		システム環境	プラットフォームやミドルウェアのバージョン、及びサポート切れになるハード・ソフトウェアとその時期
18	業務データ	データ量	ファイル数、テーブル数、データ容量、トラフィック数、伸び率
19		データ構造	論理データ構造、データレイアウト
20	保守	保守状況	システム保守、アプリケーション保守の計画と実績、業務毎のバックログ数、対応期間
21		品質状況	業務毎のインシデント数、傾向分析
22		保守環境	アプリケーション保守環境（言語、ツールなど）、開発規約、資産管理方法、リリース管理

本ステップ 1 の現行システム調査・分析結果と次のステップ 2 の新システムの要求事項を合わせて、ステップ 3 にて再構築手法の候補を選択し、ステップ 4 にて最終的に決定する。

インプット情報

調査や分析に必要な情報は、現行システムで使用している運用、設計ドキュメントおよびソースコードや定義体などの資産である。この情報だけではわからない場合は、有識者へのヒアリングを実施し補完する。具体的な内容は以下の通り。

- 現行システムの運用、保守情報（稼動ログ、保守における環境やアプリケーションの変更状況など）
- 現行システムのドキュメント（業務フロー、機能仕様書、インターフェース仕様書やデータモデル図、CRUD 図、運用手順書など）
- 現行システムの資産（ソースコード、画面・帳票定義体、運用・環境定義、外部接続定義情報など）
- ドキュメントや資産調査だけでは現状の把握が難しい場合は、有識者（業務有識者や実運用者、エンドユーザなど）へのヒアリング

アウトプット情報

調査・分析で得るべき項目の代表例を、調査観点ごとに以下に示す。これらの調査結果は、再構築手法（特にハードウェア更改以外のリホスト、リライト、リビルド）の決定、および再構築計画の策定に必要な情報となる。

(1) 資産

再構築にあたって、現行システムの構成や規模を把握し、アプリケーションの保守性および再利用範囲を見極めるための基礎情報。

- 現行システムの巨大化、肥大化の度合い（資産規模、言語の種別、定義体数など）
- 新システム構成検討に必要な項目（現行システム構成、適用ミドルウェア、データベース管理システム、文字コード体系、外字の使用有無など）
- アプリケーションの保守や維持における特性（複雑度や類似度）
- 再利用や移行対象の候補となる資産（商用システムでの稼動、または未稼動の区分）

(2) 業務知識の保有状況

業務知識の断片化、細分化の状況や度合いを調査、分析する。

また、再構築手法の選択において、再構築対象となる基幹システムの業務仕様の掌握度合いや連携する他システムへの影響調査、業務継続性を判断できる有識者の関与度合いを判断する際に必要となる情報である。

① 有識者

有識者については、スキルの深さ（質的な面）と業務エリアなどの習熟範囲、カバー範囲（量的な面）の２つの側面がある。

- 再構築時に必要な有識者（業務仕様、基盤、システム運用・保守担当）の状況
- 運用検証や合否判定ができる体制や要員検討のための、保有スキルセット

② ドキュメント

- ブラックボックス化の度合い（設計ドキュメントが最新になっているか　など）
- 運用検証のシナリオ作成のために必要なドキュメント（業務フロー、機能仕様書など）の有無

(3) 処理方式

リホストなど再構築手法によっては、再構築後も現行と同じシステム機能が必要となるため、現行システムでどのような処理方式が存在するのか、明確にする必要がある。

これらを明確にするため現行システムがどのような処理方式で動作しているのか調査を実施する。

① 再構築手法によって、移行方法や新システムでの処理方法が変わる項目の例

- 新システムで検討が必要な項目の候補（アプリケーションの動作環境、オンライン処理方式、バッチ処理方式、データベースアクセスの方法、帳票の出力方法など）

② 手法の選択にかかわらず新システムへ引き継ぐ必要のある項目の例

- 再構築時に対外的な調整や検証が必要な外部接続先の種類や連携方法（メッセージ連携、ファイル転送方法、文字コード変換の有無と種類など）
- 印刷位置の踏襲が必要な規定帳票やメールシーラなどの指定帳票の有無。

(4) 非機能要件

再構築手法の選択によって、新しいOSやプラットフォームにより基盤を再構築する場合、新旧システムの構成の違いから、運用や性能、信頼性は現行システムと全て同じになるとは限らない。

従って、業務の運用に支障がないように現行システムから引き継ぐ範囲や内容を明確にする必要がある。このため、下記のような非機能要件を観点とした現状調査を行う。

- 現行から新システムへ引き継ぐ情報を整理するための現行運用項目（運用時間帯、運用スケジュール定義、バックアップ、リカバリ方法など）
- 現行から新システムへ引き継ぐ範囲を明確にするための可用性、信頼性項目（現行システムの冗長構成や多重化の対象有無、稼動率、停止時間など）
- 新システムの性能評価のための比較元情報（現行システムの性能情報）

(5) 業務データ

再構築手法の選択により、システム切り替えやデータ移行が発生する場合のために、業務データ量やデータ仕様を調査し、把握する必要がある。

- 新システムの設備量や拡張性の諸元になる、データ容量やデータ伸長率
- データ移行時に考慮すべき、データ構造やテーブル数、データレイアウト

(6) 保守状況

再構築後の保守環境や再構築スケジュールの検討に必要な現行システムの保守状況の調査を行う。

- 再構築による投資効果を評価するための現行システムの運営コストやアプリケーションの保守コスト
- 再構築の際に考慮が必要になる凍結可否や凍結時期の検討のための現行システムのアプリケーション保守状況（変更時期、頻度など）
- 移行元アプリケーションの保守状況や品質状況を確認するための、これから着手する予定の案件やインシデントの発生状況など。

作業手順

(1) 表 2.2 の調査観点や項目例を参考に、現行システム特性に合わせて、調査内容や調査項目を抽出し、選定する。

(2) 表中に該当する観点や項目がない場合は、再構築の目的や要求事項などからテーラリングし、適宜追加する。

(3) 調査項目に応じて、ヒアリングやドキュメント調査、ツールによる資産調査により現状分析を実施する。なお、資産調査やアセスメントを、サービス提供しているベンダ企業のサービスを活用して、実施することも可能である。ベンダ企業のサービスの例は、4 章の事例編に示す。

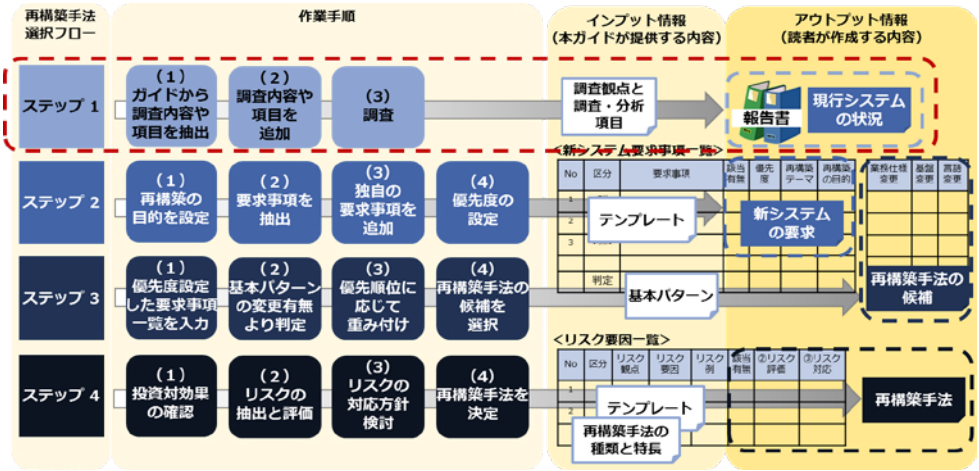


図 2.4　ステップ 1 の作業手順

2.3 新システムの要求事項分析（ステップ２）

目的

システム再構築した先にある最終目的は、経営課題を達成することである。そのために新システムの求める姿を描いたうえで、再構築の目標や目的を設定するところからはじめる必要がある。

再構築の目的がはっきりしないと、再構築に進むべきか、止めるべきかの判断も曖昧になる。再構築の目的と、それを実現するための新システムの要求事項を明確にすることが重要である。

さらに、優先度が高い要求内容―例えば、運用コスト削減や保守性を上げることが優先なのか、業務スリム化や業務柔軟性を向上させることが優先なのか―を明確にして、それに合った最適な手法を選択する必要がある。

本ガイドでは、再構築の代表的な目的や‘きっかけ’である以下の５つを取り上げ、これを「テーマ」として起点に用いることにより、再構築手法の選択や計画策定のスタートラインとしている。

表 2.1　再構築のテーマ一覧（再掲）

No	再構築テーマ	テーマの意味
1	HW SW 老朽化対応	ハードウェア機種やソフトウェアの販売終了、サポート終了などへの対応
2	維持コスト削減	現行システムの維持管理コスト（運用・保守コストなど）の削減
3	業務スリム化	肥大化、複雑化した現状基幹システムのスリム化、業務改善、運用の改善
4	柔軟性向上	新しいビジネスとの連携や新サービスへの対応を容易する柔軟性向上
5	要員不足対応	現行システムの保守要員の高齢化やレガシー技術者の減少への対応

ステップ 2 では再構築の目的を軸にして新システムの要求事項をまとめて、次のステップ３にて再構築手法を選択する。

インプット情報

本ガイドにおける新システムの要求事項分析に必要なインプット情報を示す。

(1) 新システムの要求事項一覧テンプレート

(a) 再構築の起点となる再構築のテーマ

本ガイドで取り扱う代表的な再構築テーマを表 2.1 および表 2.3 に示す。

(b) 新システムの要求事項

本ガイドでは、「JUAS 企業 IT 動向調査 2016（15 年度調査）」[1]などから、再構築に求められる必要性の高い要求事項を例として表 2.3 に掲載した。

表 2.3 は、再構築テーマと要求事項を関連付けたマトリックス形式にしている。

マトリックス形式の整理により、新システムの要求事項が、再構築の目的と合致しているか確認しやすくなり、経営、業務、IT システムに対する要求事項が全体として整合し、方向性が統一されるかなどの検証に活用できる。

(2) 各企業の固有情報

- 経営　　：経営方針、事業目標　など
- 業務　　：現状の業務課題、今後の改善事項　など
- IT システム　：現行システムの課題、問題点一覧　など

表 2.3 に掲載していない独自の要求事項を追加するためのインプット情報として使用する。

新システムの要求事項の例を表 2.3 に示す。再構築テーマ（5 つ）は、「インプット情報」(1)の表 2.1 に対応したものである。新システムの要求事項は、「インプット情報」(2)の例である。

表 2.3 では、起点となる再構築テーマに寄与すると考えられる要求事項例からあらかじめ‘レ’点を付与しているので、要求事項の抽出や優先度設定などの参考にされたい。

表 2.3 新システムの要求事項一覧テンプレート

No	区分	新システムの要求事項	該当有無	優先度	再構築テーマ				
					老朽化対応	維持コスト削減	業務スリム化	柔軟性向上	要員不足対応
1	経営	業務プロセスの効率化(省力化、業務コスト削減)				レ	レ		
2		事業競争力、営業力の強化						レ	
3		迅速な業績把握、情報把握(リアルタイム経営)					レ	レ	
4		開発・運用コストの削減				レ	レ		
5		広域災害対応機能を強化したい(BCP(※1))							
6		セキュリティ機能の強化							
7		ベンダロックインを解消したい			レ	レ			
8		基盤要員の若手を育成し、既存言語の要員減へ対応したい							レ
9		一時的な再構築・移行コストは極力抑えたい				レ			
10	業務	業務開発生産性、アプリケーションの保守性を向上させたい				レ		レ	
11		オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい						レ	
12		バッチ業務は、基幹系処理であり、業務仕様は踏襲したい							レ
13		業務プロセスのスピードアップ(リードタイム短縮など)					レ	レ	
14	IT システム	最新のIT技術や新機能を使って、運用の効率化、自動化を図りたい			レ			レ	
15		現行システムの堅牢な基盤を継続して利用したい			レ				
16		現行システムの老朽化した基盤から、最新のインフラに変更したい			レ	レ			
17		現行システムより可用性や信頼性を高めるため、冗長化を実現した新たな基盤に乗せ換えたい						レ	
18		現行の開発言語の保守要員の定年から、新しい言語に変更したい				レ			レ
19		開発・保守環境は維持したいので、開発言語は変更する必要ない			レ				
20		ホストのセンタオペレーション負荷を削減したい				レ			
21		システム運用や監視方法を簡易化したい				レ		レ	
22		WSの画面・メニュー構成や操作性(ユーザインターフェース)を改善したい						レ	

(例)
再構築テーマと関連性がない要求事項のため、優先度付けから除外

(出典) 日本情報システム・ユーザー協会「企業 IT 動向調査 2016」[1]をもとに作成

(※1) BCP (Business Continuity Plan):事業継続計画のこと。災害や事故など不測の事態を想定して、事業の継続や復旧を図るための計画。

アウトプット情報

アウトプット例を「使用例」手順(1)～手順(4)-1 に示す。また、「使用例」手順(4)-2 では、再構築の目的を本ガイドの再構築テーマから選択し関連付けた（レ点）マトリックス形式ではなく、日本語で説明した例を示す。

＜アウトプット例＞

「新システム要求事項一覧」

新システムの要求事項を経営、業務、IT システムの区分毎に分類し、優先度を付けた一覧表。

作業手順

起点となる再構築のテーマおよび新システムの要求事項の参考情報と各企業の固有情報からアウトプットに至る手順を図 2.5 に示す。

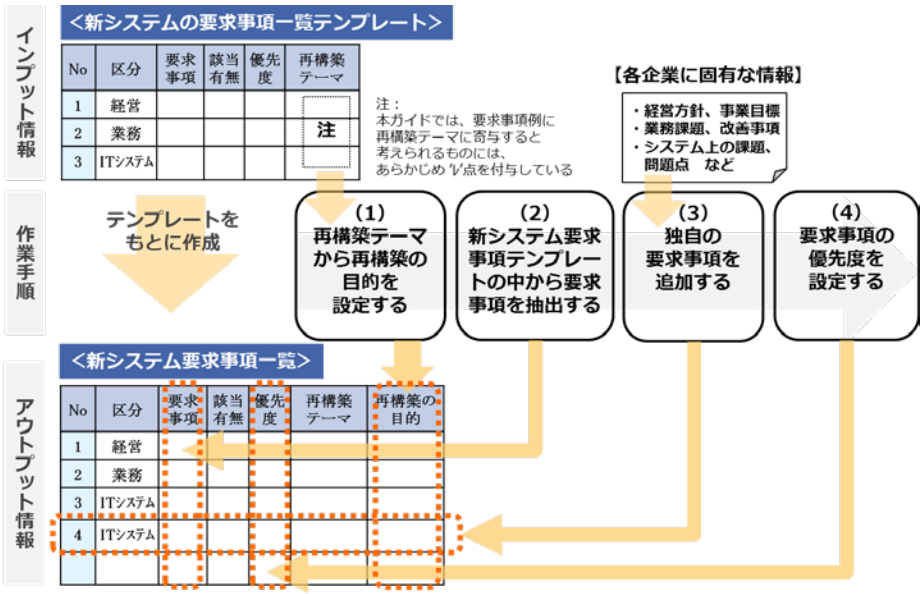


図 2.5　ステップ 2 の作業手順

(1) 再構築目的の設定

再構築テーマの中から、再構築の目的を定める。(複数可) 本ガイドでは、再構築のテーマとして代表的なものをあげている（表 2.1 参照）が、各企業の現状やニーズに応じて独自の目的を設定し、追加、変更して使用することも可能。

(2) 新システム要求事項の抽出

再構築目的をもとに、図 2.5 の要求事項例の中から適合する要求事項を抽出する。

(3) 独自の要求事項の追加

業務部門、システム部門、運用部門などからそれぞれの要求事項を追加し、経営、業務、IT システムに区分し、全体として再構築の目的に合致するか、方向性や整合性を確認する。

- 目的を実現するのに適切な手段か、他の手段はないか
- 手段は目的にどのくらい貢献しているのか

(4) 新システム要求事項一覧の完成

要求内容の重要性、緊急性、必要性などにより、優先付けを行う。本ガイドでは、起点となる再構築テーマに寄与すると考えられる要求事項例にからあらかじめ‘レ’点を付与しているので、優先度設定の参考にされたい。例えば、要求事項ごとに、再構築の目的に合致するか否かを確認し、再構築手法と関連性のない、または再構築手法の選択に影響を与えない要求事項は、選択の優先付けから除外する。

また、優先度を設定する際に、その理由や背景、根拠を記載する必要がある場合は、適宜欄を追加して記述する。(使用例を参照)

ポイント！

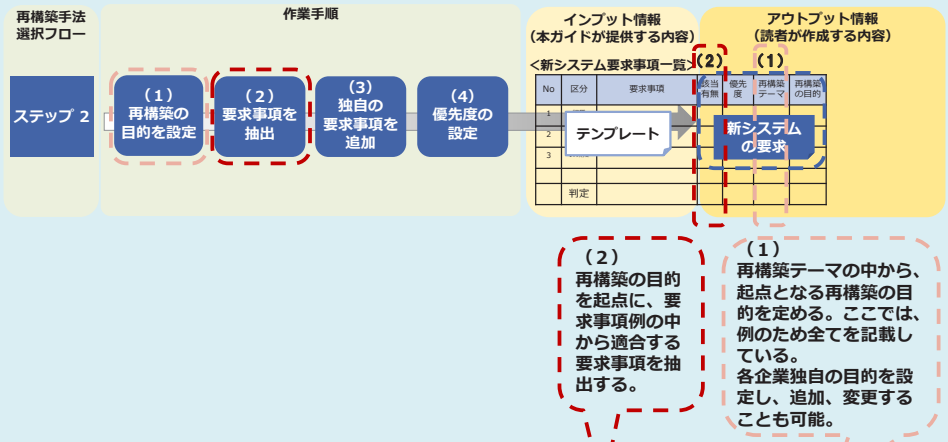
- 要求事項は要求元（業務部門、システム部門、運用部門など）によって、視点や動機が異なるため、再構築の目的に合致するか、方向性が統一されるかなどの確認が必要である。
- 経営、業務、IT システムに対する要求を目的と手段からそれぞれ関連付け、全体として整合しているか検証が必要である。
- 要求事項を再構築の目的と関連付けることで、再構築とは関連性のない、または再構築手法の選択に影響を与えない要件の抽出ができる。再構築手法の選択に影響を与えない要求は、手法選択段階では使用せず、計画策定段階で使用する。(表 2.3　新システムの要求事項一覧テンプレートの No.5,6)

本ガイドでは、上記の留意点の実施タイミングは(3)としているが、各企業の状況に合わせて別タイミングで実施してもよく、どのタイミングでも検証することが重要である。

使用例

「作業手順」に沿った、使用例を以下に示す。

＜手順(1)(2)＞

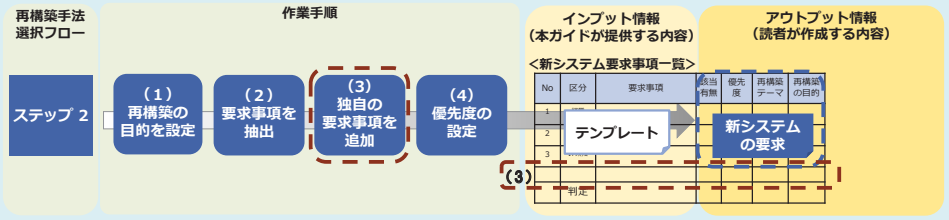


＜凡例＞該当有無(◆:該当あり、－:該当なし)

No	区分	新システムの要求事項	該当有無	優先度	再構築テーマ				
					老朽化対応	維持コスト削減	業務スリム化	柔軟性向上	要員不足対応
1	経営	業務プロセスの効率化(省力化、業務コスト削減)	◆			レ	レ		
2		事業競争力、営業力の強化	◆					レ	
3		迅速な業績把握、情報把握(リアルタイム経営)	－				レ	レ	
4		開発・運用コストの削減	◆			レ	レ		
5		広域災害対応機能を強化したい(BCP)	－						
6		セキュリティ機能の強化	－						
7		ベンダロックインを解消したい	－		レ	レ			
8		基盤要員の若手を育成し、既存言語の要員減へ対応したい	－						レ
9		一時的な再構築・移行コストは極力抑えたい	◆			レ			
10	業務	業務開発生産性、アプリケーションの保守性を向上させたい	◆			レ		レ	
11		オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい	◆					レ	
12		バッチ業務は、基幹系処理であり、業務仕様は踏襲したい	◆						レ
13		業務プロセスのスピードアップ(リードタイム短縮など)	－				レ	レ	
14	ITシステム	最新のIT技術や新機能を使って、運用の効率化、自動化を図りたい	－		レ			レ	

（例）再構築テーマと関連性がない要求事項のため、優先度付けから除外

＜手順(3)＞



(3)
業務部門、システム部門、利用部門などからそれぞれの要求事項を追加し、経営、業務、ITシステムに区分し、全体として再構築の目的に合致するか、方向性や整合性を検証する。

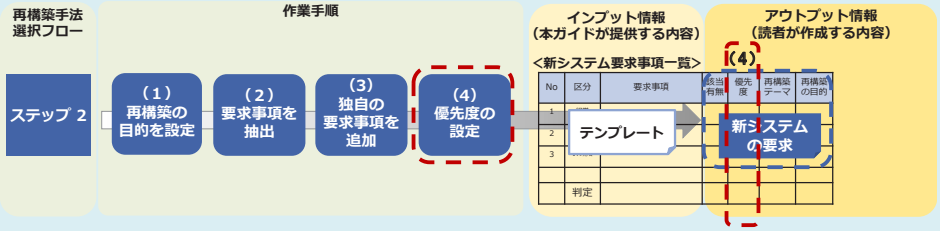
【各企業に固有な情報】
- 経営方針、事業目標
- 業務課題、改善事項
- システム上の課題、問題点

など

＜凡例＞該当有無(◆:該当あり、－:該当なし)

No	区分	新システムの要求事項	概要有無	優先度	再構築テーマ				
					老朽化対応	維持コスト削減	業務スリム化	柔軟性向上	要員不足対応
15	IT システム	現行システムの堅牢な基盤を継続して利用したい	－		レ				
16		現行システムの老朽化した基盤から、最新のインフラに変更したい	◆		レ	レ			
17		現行システム上り可用性や信頼性を高めるため、冗長化を実現した新たな基盤に乗せ換えたい	－					レ	
18		現行の開発言語の保守要員の定年から、新しい言語に変更したい	◆			レ			レ
19		開発・保守環境は維持したいので、開発言語は変更する必要ない	－		レ				
20		ホストのセンタオペレーション負荷を削減したい	－			レ			
21		システム運用や監視方法を簡易化したい	－			レ		レ	
22		WSの画面・メニュー構成や操作性(ユーザインターフェース)を改善したい	◆					レ	
追加	経営	業務プロセスの質・精度の向上(ミス、欠品削減など)	◆				レ	レ	

＜手順(4)-1＞



（4）-1
要求内容の重要性、緊急性などにより、優先付けを行い、アウトプットする。
下表では、３段階（◎：必須、○：推奨、△：できれば実現したい）を例示している。

＜凡例＞該当有無◆:該当あり、－:該当なし
優先度 ◎:必須、○:推奨、△:可能ならば

No	区分	新システムの要求事項	該当有無	優先度	再構築の目的				
					老朽化対応	維持コスト削減	業務スリム化	柔軟性向上	要員不足対応
1	経営	業務プロセスの効率化(省力化、業務コスト削減)	◆	◎		レ	レ		
2		事業競争力、営業力の強化	◆	◎				レ	
4		開発・運用コストの削減	◆	◎		レ	レ		
9		一時的な再構築・移行コストは極力抑えたい	◆	○		レ			
追加		業務プロセスの質・精度の向上(ミス、欠品削減など)	◆	○			レ	レ	
10	業務	業務開発生産性、アプリケーションの保守性を向上させたい	◆	○		レ		レ	
11		オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい	◆	○				レ	
12		バッチ業務は、基幹系処理であり、業務仕様は踏襲したい	◆	○					レ
16	IT システム	現行システムの老朽化した基盤から、最新のインフラに変更したい	◆	○	レ	レ			
18		現行の開発言語の保守要員の定年から、新しい言語に変更したい	◆	○		レ			レ
22		WSの画面・メニュー構成や操作性(ユーザインターフェース)を改善したい	◆	△				レ	

＜手順(4)-2＞

（4）-2
優先度の設定について再構築の目的との関連ではない要因がある場合には、自由記入欄（※1）を追加して、背景や理由などを記載する。

＜凡例＞該当有無:◆:該当あり、－:該当なし
優先度 :◎:必須、○:推奨、△:可能ならば

No	区分	新システムの要求事項	該当有無	優先度	再構築の目的(※1)
1	経営	業務プロセスの効率化(省力化、業務コスト削減)	◆	◎	不要なプロセスを排除し、運用コスト削減のため
2		事業競争力、営業力の強化	◆	◎	事業課題、経営トップの方針
4		開発・運用コストの削減	◆	◎	運用、保守費用の削減
9		一時的な再構築・移行コストは極力抑えたい	◆	○	一時コスト含めたTCO削減のため
追加		業務プロセスの質・精度の向上(ミス、欠品削減など)	◆	○	業務生産性の向上
10	業務	業務開発生産性、アプリケーションの保守性を向上させたい	◆	○	制度改正等の保守費用の削減
11		オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい	◆	○	新業態、新サービスへの柔軟で、迅速な対応
12		バッチ業務は、基幹系処理であり、業務仕様は踏襲したい	◆	○	オープン環境への安全で、確実な移行
16	IT システム	現行システムの老朽化した基盤から、最新のインフラに変更したい	◆	○	現ハードウエアの保守期間、サポート切れ
18		現行の開発言語の保守要員の定年から、新しい言語に変更したい	◆	○	オープン環境への開発要員のシフト
22		WSの画面・メニュー構成や操作性(ユーザインターフェース)を改善したい	◆	△	エンドユーザの利便性の向上

2.4　再構築手法の選択（ステップ３）

目的

ステップ 1 の現状調査とステップ 2 で明確にした要求事項から再構築の"候補"となる手法を選択する。同一システムでも業務やサブシステムの現状や要求によって、最適な再構築手法は異なる場合があるため、ステップ 3 では複数手法を候補として選定することも可能である。

また、手法選択の単位は、現行システムの調査・分析や新システムの要求事項分析の対象範囲に応じて、システム全体とサブシステムや業務単位のどちらにも適用できる。

インプット情報

- 現行システムの調査・分析(ステップ 1)の結果
- 新システムの要求事項分析(ステップ 2)の結果
  （新システム要求事項一覧　など）
- 基本パターン

アウトプット情報

システム再構築手法の候補
「ハードウェア更改」、「リホスト」、「リライト」、「リビルド」のいずれか。(複数可)

作業手順

(1) 基本パターン

再構築の目的や新システムの要求事項から、再構築手法の主要要素である業務仕様、基盤、開発言語の変更が必要か(変更あり)、それとも変更不要か(変更なし)を評価し、候補となる再構築手法を選択する。

再構築手法選択の基本パターンを図 2.6 に示す。

<凡例>（○：変更あり、×：変更なし、－：どちらでも可）

業務仕様	基盤	開発言語	候補
×	×	×	ハードウェア更改
×	○	×	リホスト
×	○	○	リライト
○	○	－	リビルド

図 2.6　再構築手法選択基本パターン

(2) 詳細選択手順

新システムへの要求事項（ステップ 2）について、それらを実現するために、業務仕様の変更、基盤の変更、言語変更の必要有無をもとに、候補を選定する。詳細手順を図 2.7 に示す。

① ステップ 2 の新システムの要求事項それぞれに対して、それを実現するためには、業務仕様を変更する必要があるか、ないかを検討し、該当するものにマーク（レ点など）を付ける。

（注）どちらも該当する場合は両方にマークし、どちらにも該当しない場合はマークしない。

② 基盤についても、同様に行う。

③ 開発言語についても、同様に行う。

④ 全ての要求事項について、①②③の終了後、業務仕様の変更、基盤の変更、言語の変更について、それぞれ「変更あり」、「変更なし」のどちらが多いかを集計の上で、その多いほうを判定し、「図 2.6 再構築手法選択基本パターン」に基づき候補を選択する。その際、優先度に応じて重みを付けて判定する。（例えば、◎：必須は 3 ポイント、○：推奨は 2 ポイントなど）

もし、同数の場合は、複数手法を候補とするか、要求事項の優先度を見直して再評価するかを検討する。

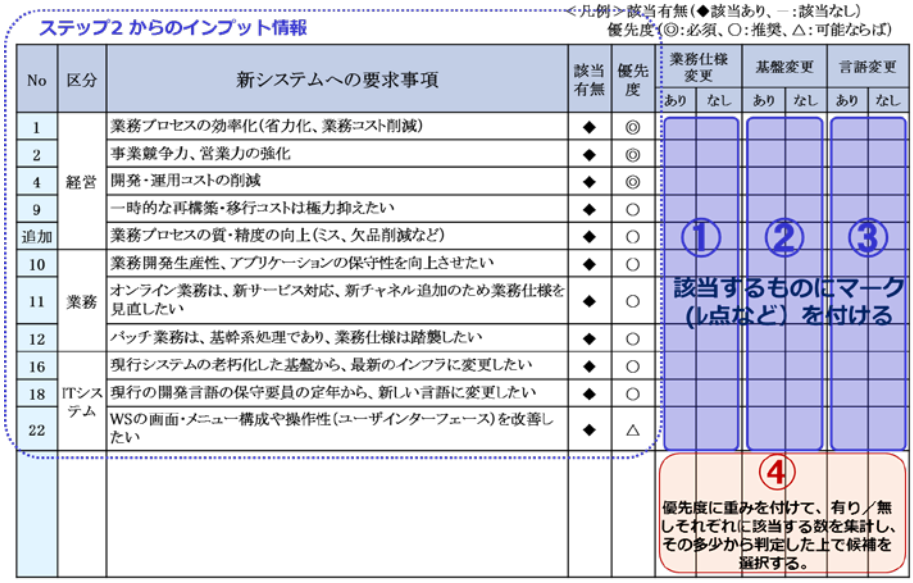


図 2.7 再構築手法選択の詳細

なお、新システムへの要求事項から業務仕様や基盤および言語変更の有無を判断できない場合は、必要に応じて、企業のシステム部門、ベンダ企業などのサポートを受ける。

使用例

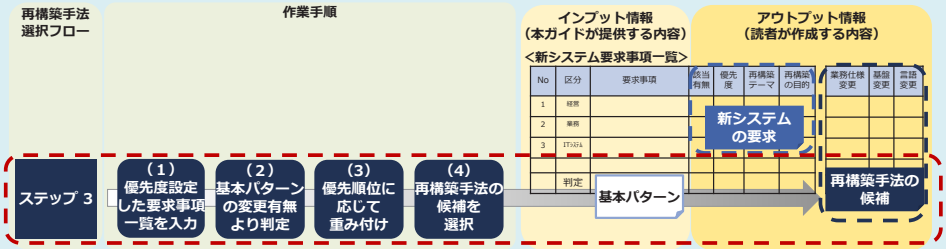
＜例 1：業務単位に手法を選択＞

新システムの要求事項の一つ一つからではなく、業務単位にオンライン業務やバッチ業務の仕様変更の有無と基盤変更のみから、候補を選択する例を以下に示す。

業務仕様	基盤		候補
オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい。	変更する		リビルド
バッチ業務は、基幹系処理であり、業務仕様は現行から変更しない。	変更する		リライト or リホスト

＜例 2：新システム要求事項から選択＞

ステップ 3 詳細選択手順の実施例として、新システム要求事項一覧（ステップ 2 のアウトプット例）から選択するパターンを以下に示す。要求事項毎に業務仕様の変更、基盤の変更、言語の変更について変更あり／なしのどちらが多いかを集計の上で、その多いほうを判定し、再構築手法を選択する。以下は、優先度を◎：必須は 3 ポイント、○：推奨は 2 ポイント、△：可能ならばは、1 ポイントで集計した例である。



ステップ2 からのインプット情報

＜凡例＞該当有無（◆該当あり、－：該当なし）
優先度（◎：必須、○：推奨、△：可能ならば）

No	区分	新システムへの要求事項	該当有無	優先度	業務仕様変更 あり	業務仕様変更 なし	基盤変更 あり	基盤変更 なし	言語変更 あり	言語変更 なし
1	経営	業務プロセスの効率化（省力化、業務コスト削減）	◆	◎	レ					
2	経営	事業競争力、営業力の強化	◆	◎	レ		レ			
4	経営	開発・運用コストの削減	◆	◎			レ			
9	経営	一時的な再構築・移行コストは極力抑えたい	◆	○		レ		レ		レ
追加	経営	業務プロセスの質・精度の向上（ミス、欠品削減等）	◆	○	レ					
10	業務	業務開発生産性、アプリケーションの保守性を向上させたい	◆	○			レ		レ	
11	業務	オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい	◆	○	レ					
12	業務	バッチ業務は、基幹系処理であり、業務仕様は踏襲したい	◆	○		レ				レ
16	ITシステム	現行システムの老朽化した基盤から、最新のインフラに変更したい	◆	○			レ			
18	ITシステム	現行の開発言語の保守要員の定年から、新しい言語に変更したい	◆	○					レ	
22	ITシステム	WSの画面・メニュー構成や操作性（ユーザインターフェース）を改善したい	◆	△	レ		レ		レ	
		◎：必須（3ポイント）、○：推奨（2ポイント）、△：できれば（1ポイント）で集計した場合			11	4	11	2	5	4

上記では、「業務仕様を変更する」、「基盤を変更する」、「言語を変更する」が多いため、「図 2.6 再構築手法選択基本パターン」をもとに選択する候補は、「リビルド」となる。

＜凡例＞（○：変更あり、×：変更なし、－：どちらでも可）

業務仕様	基盤	開発言語	候補
×	×	×	ハードウェア更改
×	○	×	リホスト
×	○	○	リライト
○	○	－	リビルド

なお、再構築手法の候補選定によって、実現できなくなった要求事項（数の少ない方）があれば、それが何か（上記では No. 9, 12）、どう取り扱うのかなどを記録しておくことも重要である。

2.5 再構築手法の決定（ステップ４）

目的

現行システムの状況と新システムへの要求事項から再構築に最も適した手法を決定する。まず、ステップ 3 にて候補として選定した再構築手法について、メリット・デメリットや投資対効果を検討する。次に、再構築において発生する可能性のあるリスクについて、ステークホルダ（経営者、業務部門、システム部門、運用部門など）で共有を図り、最適な再構築手法を決定する。その際には、参照した情報や判断の根拠とした情報を記録として残す。

なお、ステップ 3 で選択した手法にて、期待した効果が得られないためステークホルダ間での合意が得られない場合は、手法の再選定、または新システムへの要求事項や優先度の見直しを行い、再度ステップ 2 からの手順を実施する。

インプット情報

(1) 現行システムの調査・分析(ステップ 1)の結果
「現行システム調査・分析結果報告書」　など

(2) 新システムの要求事項分析(ステップ 2)の結果
「新システム要求事項一覧」　など

(3) ステップ 3 において選定した、再構築手法の候補
「ハードウェア更改」、「リホスト」、「リライト」、「リビルド」のいずれか 1 つ、または複数。

(4) 「表 2.4 再構築手法の種類と特徴」
再構築手法に応じて構築にかかるコスト・時間や品質確保に必要な有識者の関与度合いと保守性の向上など再構築後の効果を対比した手法決定のためにステークホルダが参照する情報。

(5) 「表 2.5 再構築におけるリスク要因テンプレート」

(6) 「表 2.6 再構築におけるリスク対応テンプレート」

アウトプット情報

(1) 決定した再構築手法（後続の計画策定のインプット）

(2) 再構築手法決定の根拠、ステークホルダ間で意識合わせを行った合意形成の記録
＜アウトプット例＞

- 再構築手法の投資対効果の比較評価
- リスク評価一覧、リスク対応方針
- 再構築手法は何を元に決定したのか、誰の判断で決まったのか、合意形成した決定会合の議事録　など

ポイント！

再構築手法の決定などシステム化の企画は、1 回の決定会合で決まるとは限らないため、十分な期間を設けて論議していく必要がある。決定に至った経緯や根拠をトレースし、振り返ることができるように、記録を残しておくことが肝要である。

作業手順

ステップ 3 にて候補選定した再構築手法から、最適な再構築手法を決定するまでの手順を以下に示す。

(1) 再構築手法の投資対効果（メリット・デメリット）の確認
(2) 再構築リスクの共有
- 選択した再構築手法において、再構築時に発生しうるリスクの抽出と評価
- リスク対応方針の検討

(3) 再構築手法の決定

以下、この流れに沿って各手順を説明する。

(1) 再構築手法の投資対効果（メリット・デメリット）の確認

再構築手法にはそれぞれ特徴があり、メリット・デメリットや再構築時に必要なコストと構築後の効果は異なるため、表 2.4 を参考に、選択した手法におけるメリット・デメリットや投資対効果を確認する。

① 再構築に必要な投資
- 再構築時に当たっての検証方法、品質確保の方法
- 再構築時に必要な業務有識者の関与度合い
- 再構築に必要なコスト、期間
- 必要な体制（リソース配置）

② 再構築後の効果

- 新しいプラットフォームの活用
- 運用コストの削減
- アプリケーション保守性の向上や保守コストの削減度合い
- 業務の効率化や新規業務への柔軟な対応
- 新サービスを提供したい目標時期に合っているか

表 2.4 再構築手法の種類と特徴

比較項目			再構築			
			ハードウエア更改	リホスト	リライト	リビルド
業務仕様の変更			変更なし	変更なし	変更なし	見直す
基盤の変更（アーキテクチャなど）			変更なし	一部変更あり（アーキテクチャをエミュレート又は非互換と関連部分の再設計）	一部変更あり（アプリ構造、基盤は再設計）	変更あり（再設計）
プログラム資産の変更			変更なし	変更小（自動変換＋手修正）	異言語で書き換える部分は変更中～大（アーキテクチャに合わせてリライト）	変更大（再設計）
再構築に必要な投資	検証、品質確保の方法		疎通テストレベル	プロトフォームや言語非互換を中心にしたブラックボックステスト（現新比較テスト）	左記に加えて、言語書き換え（COBOL→java等）の部分はホワイトボックステスト	再設計後の設計書に基づいたホワイトボックステスト
	業務有識者の関与度合い	量的な面	小	小～中	中	大
		質的な面	現行運用・保守の範囲内	業務シナリオ作成、テスト結果の妥当性判断	業務シナリオ作成、テスト結果の妥当性判断	現行運用、業務仕様の網羅的な把握、業務シナリオ作成、テスト結果の妥当性判断
	再構築コスト		小	小～中	中	大
	再構築期間		短期間	短～中期間	中～長期間	長期間
再構築後の効果	新しいプラットフォームの活用		不可	可能	可能	可能
	運用コストの削減		限定的	取り組みは可能	取り組みは可能	可能
	新規業務への柔軟な対応／業務の効率化		不可	不可	不可	可能
	アプリケーションの保守性		向上しない（現行の状態を引き継ぐ）	向上しない（現行の状態を引き継ぐ）	基本的に向上しない、言語書き換え部分は一部向上する	向上する（再設計でスリム化・最適化）
	保守コストの削減		不可	不可	部分的な取り組みは可能	可能

ポイント！

- リビルドは、アプリケーションの保守性や業務柔軟性をリホスト／リライトより向上させることが可能であるが、再構築にかかるコストや業務有識者の関与度合いは、より高く求められる。
- リホスト、リライトは、現行資産を活用することで業務仕様の再現性を高めることが可能であるが、アプリケーションの保守性や業務柔軟性は、現行システムと同等である。

(2) 再構築リスクの共有

① 想定リスクの抽出と評価

ステップ 3 で選定した再構築手法を対象に、リスク要因と再構築時に発生しうるリスクをステークホルダで共有し、認識合わせを行う。

システム再構築時に問題化する可能性のある、リスク要因とリスク例を表 2.5 に示す。

表 2.5 のリスク要因とリスク例の中から、当てはまり発生しうると考えられる（該当する）リスクを特定・抽出し（①該当有無）、ステップ 3 で選定した再構築候補を対象に、②リスク評価（顕在化の度合いなど）を行う。

ポイント！

- 候補となる手法が複数の場合は、手法間での相対的な評価を行う。
- 例にないリスクが想定される場合には、追加のうえ評価する。

表 2.5 再構築におけるリスク要因テンプレート

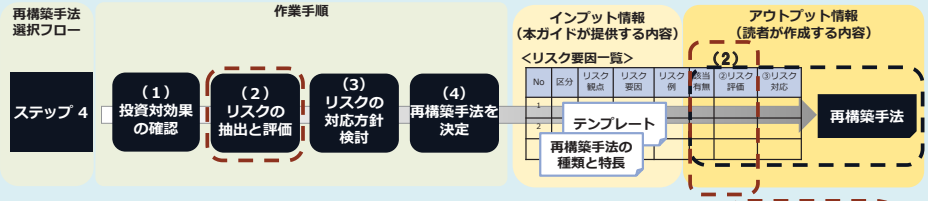
（注）ステップ3で選定した再構築手法が対象

No	区分	リスク観点	再構築におけるリスク要因	リスク例	①該当有無	②リスク評価（注）			
						HW更改	リホスト	リライト	リビルド
1	要求	現行踏襲（機能）	• 現行から受け継ぐべき業務運用や画面・帳票、データ仕様、外部インターフェースなどの対象や範囲は明確か？	• 作り込み品質の不良 • サービス開始の延伸 • 作業工数の増大					
2	要求	現行踏襲（非機能）	• 移行先のシステムの運用方法や性能、操作性、可用性、信頼性などの非機能要件は明確か？	• テスト、運用段階になり、利用者から「使えない」、「レベルダウン」とのクレームが起こる					
3	要求	再構築サービスの選定	• 適用するサービスやソリューションは、現行と同じ機能、運用を全て実現できるか？（特にリホストの場合） • 役割分担や制限、制約事項は明確になっているか？	• 想定どおり動作しない不具合の多発による手戻り • サービス開始の延伸 • 作業工数の増大					
4	要求	アーキテクチャ	• 最新技術の適用や初めてのアーキテクチャ採用時には、充分な適用検討と技術検証を行っているか？ • アーキテクチャが混在することはないか？（例. 現新混在）	• 想定どおり動作しない不具合の多発による手戻り、対応工数の増大 • 保守要員の増加 • 保守環境の複雑化					
5	現状	システムの規模	• 現行システムは肥大化・複雑化していないか？ （例えば、未稼働資産が多数存在する）	• 業務仕様の把握不足による手戻り • 作業工数の増大					
6	現状	業務知識不足（要員）	• 現行業務有識者は、後継者不足になっていないか？（現行システム仕様は掌握できるか） • 再構築時の要員は確保可能か？	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • 試験による検証の長期化 • サービス開始の延伸 • 作業工数の増大					
7	現状	業務知識不足（ドキュメント）	• 業務ドキュメントとアプリケーションは一致しているか？ • ブラックボックス化により影響調査の精度が低下していないか？	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • サービス開始の延伸 • 作業工数の増大					
8	現状	資産管理	• 現行稼動中資産と移行資産のアンマッチ（例. 実行モジュールとソースコードのアンマッチ） • 未検証資産の移行（開発中資産）は、ないか？	• 想定どおり動作しない不具合発生による手戻り、対応工数の増大					
9	現状	データ移行	• データ仕様の把握不足によるデータ変換の誤り（フォーマット、文字コード） • 異常データの混入は考えられないか？	• 現行システムと新システムで結果が一致しない • 移行データ調査や対応工数の増大					

＜再構築におけるリスク要因テンプレートの使用例＞

- 表 2.5 リスク要因とリスク例の中から該当するリスクを特定する。(①該当有無)
- ステップ 3 で選定した再構築候補を対象に、選択した手法間でリスク顕在化の相対評価を行う。(②リスク評価)

以下の使用例では、No.1,2,6,7 をリスクとして認識したため、①該当有無を「該当あり」としている。この例では、ステップ 3 にて再構築手法のハードウェア更改は選択されなかったため、リスク評価の対象からは削除し、候補となったリホスト、リライト、リビルドを対象としている。



No	区分	リスク観点	再構築におけるリスク要因	リスク例	①該当有無	②リスク評価		
						リホスト	リライト	リビルド
1	要求	現行踏襲(機能)	• 現行から受け継ぐべき業務運用や画面・帳票、データ仕様、外部インターフェースなどの対象や範囲は明確か？	• 作り込み品質の不良 • サービス開始の延伸 • 作業工数の増大	あり	△	○	◎
2	要求	現行踏襲(非機能)	• 移行先のシステムの運用方法や性能、操作性、可用性、信頼性などの非機能要件は明確か？	• テスト、運用段階になり、利用者から「使えない」、「レベルダウン」とのクレームが起こる	あり	◎	○	○
3	要求	再構築サービスの選定	• 適用するサービスやソリューションは、現行と同じ機能、運用を全て実現できるか？(特にリホストの場合) • 役割分担や制限、制約事項は明確になっているか？	• 想定どおり動作しない不具合の多発による手戻り • サービス開始の延伸 • 作業工数の増大	なし	—	—	—
4	要求	アーキテクチャ	• 最新技術の適用や初めてのアーキテクチャ採用時には、充分な適用検討と技術検証を行っているか？ • アーキテクチャが混在することはないか？(例. 現新混在)	• 想定どおり動作しない不具合の多発による手戻り、対応工数の増大 • 保守要員の増加 • 保守環境の複雑化	なし	—	—	—
5	現状	システムの規模	• 現行システムは肥大化・複雑化していないか？(例えば、未稼働資産が多数存在する)	• 業務仕様の把握不足による手戻り • 作業工数の増大	なし	—	—	—
6	現状	業務知識不足(要員)	• 現行業務有識者は、後継者不足になっていないか？(現行システム仕様は掌握できるか) • 再構築時の要員は確保可能か？	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • 試験による検証の長期化 • サービス開始の延伸 • 作業工数の増大	あり	○	○	◎
7	現状	業務知識不足(ドキュメント)	• 業務ドキュメントとアプリケーションは一致しているか？ • ブラックボックス化により影響調査の精度が低下していないか？	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • サービス開始の延伸 • 作業工数の増大	あり	△	○	◎
8	現状	資産管理	• 現行稼動中資産と移行資産のアンマッチ(例.実行モジュールとソースコードのアンマッチ) • 未検証資産の移行(開発中資産)は、ないか？	• 想定どおり動作しない不具合発生による手戻り、対応工数の増大	なし	—	—	—
9	現状	データ移行	• データ仕様の把握不足によるデータ変換の誤り(フォーマット、文字コード) • 異常データの混入は考えられないか？	• 現行システムと新システムで結果が一致しない • 移行データ調査や対応工数の増大	なし	—	—	—

②リスク評価の凡例：リスクが顕在化する相対的な度合い◎：高、○：中、△：低

② リスク対応方針の検討

選択した再構築手法についてリスク要因と発生しうるリスクに対して、リスクを顕在化の予防のために、回避、軽減、受容、転嫁の観点で、リスク対応方針や考え方の共有を行う。

前項の、リスク要因とリスク例の中から、発生しうるリスク要因を特定・抽出し、選定した再構築手法におけるリスクへの対応方針を検討する。(③リスク対応)

表 2.6 に前述のリスク要因に対する一般的なリスク対応のテンプレートを示す。下表は、「表 2.5 再構築におけるリスク要因テンプレート」のリスク要因に対して、一般的なリスク対応(回避、軽減、受容、転嫁)を記載できるようにしたテンプレートである。リスク対応方針の検討結果などにより表現方法はテーラリングして使用されたい。

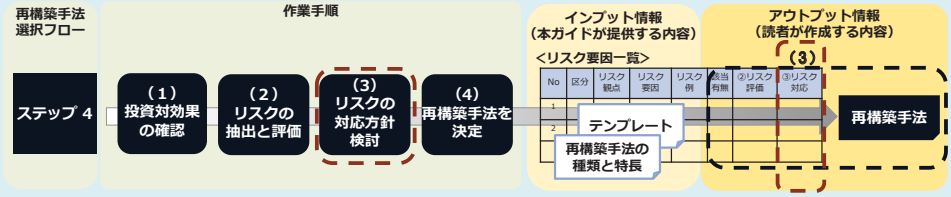
表 2.6 再構築におけるリスク対応テンプレート

No	区分	リスク観点	再構築におけるリスク要因	リスク例	①該当有無	②リスク評価				③リスク対応			
						HW更改	リホスト	リライト	リビルド	回避	軽減	受容	転嫁
1	要求	現行踏襲(機能)	• 現行から受け継ぐべき業務運用や画面・帳票、データ仕様、外部インターフェースなどの対象や範囲は明確か?	• 作り込み品質の不良 • サービス開始の延伸 • 移行作業工数の増大									
2	要求	現行踏襲(非機能)	• 移行先のシステムの運用方法や性能、操作性、可用性、信頼性などの非機能要件は明確か?	• テスト、運用段階になり、利用者から「使えない」、「レベルダウン」とのクレームが起こる									
3	要求	再構築サービスの選定	• 適用するサービスやソリューションは、現行と同じ機能、運用を全て実現できるか?(特にリホストの場合) • 役割分担や制限、制約事項は明確になっているか?	• 想定どおり動作しない不具合の多発による手戻り • サービス開始の延伸 • 作業工数の増大									
4	要求	アーキテクチャ	• 最新技術の適用や初めてのアーキテクチャ採用時には、充分な適用検討と技術検証を行っているか? • アーキテクチャが混在することはないか?(例. 現新混在)	• 想定どおり動作しない不具合の多発による手戻り、対応工数の増大 • 保守要員の増加 • 保守環境の複雑化									
5	現状	システムの規模	• 現行システムは肥大化・複雑化していないか? (例えば、未稼働資産が多数存在する)	• 業務仕様の把握不足による手戻り • 作業工数の増大									
6	現状	業務知識不足(要員)	• 現行業務有識者は、後継者不足になっていないか?(現行システム仕様は掌握できるか) • 再構築時の要員は確保可能か?	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • 試験による検証の長期化 • サービス開始の延伸 • 作業工数の増大									
7	現状	業務知識不足(ドキュメント)	• 業務ドキュメントとアプリケーションは一致しているか? • ブラックボックス化により影響調査の精度が低下していないか?	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • サービス開始の延伸 • 作業工数の増大									
8	現状	資産管理	• 現行稼動中資産と移行資産のアンマッチ(例. 実行モジュールとソースコードのアンマッチ) • 未検証資産の移行(開発中資産)は、ないか?	• 想定どおり動作しない不具合発生による手戻り、対応工数の増大									
9	現状	データ移行	• データ仕様の把握不足によるデータ変換の誤り(フォーマット、文字コード) • 異常データの混入は考えられないか?	• 現行システムと新システムで結果が一致しない • 移行データ調査や対応工数の増大									

＜再構築におけるリスク対応テンプレートの使用例＞

- 前述したリスク対応テンプレートのリスク要因とリスク例の中から、発生しうるリスク要因を特定・抽出する。(①該当有無)
- 選択した再構築手法におけるリスク要因への対応方法の方針を検討する。(②リスク評価)

下表の例では、特定したNo. 1, 2, 6, 7のリスク要因ごとに、リスク対応の方針や考え方を検討している。本例では、リスク対応としてポピュラーな観点である‘回避’‘軽減’‘受容’‘転嫁’としているが、リスク対応方針の検討結果により、テーラリングして使用する。



(3) リスク対応の観点や区分は、テーラリングして使用する

＜凡例＞ リスク評価:リスクが顕在化する相対的な度合い(◎:高、○:中、△:低)
リスク対応:●検討対象とするリスク対応方針)

No	区分	リスク観点	再構築におけるリスク要因	リスク例	①該当有無	②リスク評価			③リスク対応			
						リホスト	リライト	リビルド	回避	軽減	受容	転嫁
1	要求	現行踏襲(機能)	• 現行から受け継ぐべき業務運用や画面・帳票、データ仕様、外部インターフェースなどの対象や範囲は明確か？	• 作り込み品質の不良 • サービス開始の延伸 • 移行作業工数の増大	あり	△	○	◎	●	●		
2	要求	現行踏襲(非機能)	• 移行先のシステムの運用方法や性能、操作性、可用性、信頼性などの非機能要件は明確か？	• テスト、運用段階になり、利用者から「使えない」、「レベルダウン」とのクレームが起こる	あり	◎	○	○	●	●		
3	要求	再構築サービスの選定	• 適用するサービスやソリューションは、現行と同じ機能、運用を全て実現できるか？(特にリホストの場合) • 役割分担や制限、制約事項は明確になっているか？	• 想定どおり動作しない不具合の多発による手戻り • サービス開始の延伸 • 作業工数の増大	なし	—	—	—				
4	要求	アーキテクチャ	• 最新技術の適用や初めてのアーキテクチャ採用時には、充分な適用検討と技術検証を行っているか？ • アーキテクチャが混在することはないか?(例. 現新混在)	• 想定どおり動作しない不具合の多発による手戻り、対応工数の増大 • 保守要員の増加 • 保守環境の複雑化	なし	—	—	—				
5	現状	システムの規模	• 現行システムは肥大化・複雑化していないか？ (例えば、未稼働資産が多数存在する)	• 業務仕様の把握不足による手戻り • 作業工数の増大	なし	—	—	—				
6	現状	業務知識不足(要員)	• 現行業務有識者は、後継者不足になっていないか？(現行システム仕様は掌握できるか) • 再構築時の要員は確保可能か？	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • 試験による検証の長期化 • サービス開始の延伸 • 作業工数の増大	あり	○	○	◎		●		
7	現状	業務知識不足(ドキュメント)	• 業務ドキュメントとアプリケーションは一致しているか？ • ブラックボックス化により影響調査の精度が低下していないか？	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • サービス開始の延伸 • 作業工数の増大	あり	△	○	◎		●	●	
8	現状	資産管理	• 現行稼動中資産と移行資産のアンマッチ(例.実行モジュールとソースコードのアンマッチ) • 未検証資産の移行(開発中資産)は、ないか？	• 想定どおり動作しない不具合発生による手戻り、対応工数の増大	なし	—	—	—				
9	現状	データ移行	• データ仕様の把握不足によるデータ変換の誤り(フォーマット、文字コード) • 異常データの混入は考えられないか？	• 現行システムと新システムで結果が一致しない • 移行データ調査や対応工数の増大	なし	—	—	—				

(3) 再構築手法の決定

再構築手法のメリット・デメリットや投資・効果および発生しうるリスクの共有と対応方針、考え方について、ステークホルダ（経営者、業務部門、システム部門、運用部門など）で合意形成を図り、再構築手法を決定する。主な協議内容は以下の通り。

① 現行システムの調査・分析結果による、現状の認識合わせ
② 新システムの要求事項の優先度が再構築の目的と方向性が合致しているか、選択した再構築手法と整合しているか
③ 再構築手法のメリット・デメリットや再構築への投資と構築後の効果
④ 再構築において発生しうるリスク要因の共有とリスクへの対応方針

ポイント！

決定に際して共有した情報や根拠となった資料を後で振り返ることができるように記録として残す。なお、決定時のプロセスは計画策定編「3.7 意思決定プロセスの策定（観点F)」が参考になる。

もし、ステークホルダ間で現状の認識、新システム要求事項の優先度や決定する再構築手法およびコストや期間などで合意形成に至らない場合は、新システムの要求事項分析（ステップ 2）の優先度の見直しやそれによる再構築手法の選択（ステップ 3）の見直しを行い、合意形成が得られるまで繰り返す。

図 2.8 に再構築手法決定時に合意形成を図る場面のイメージを示す。

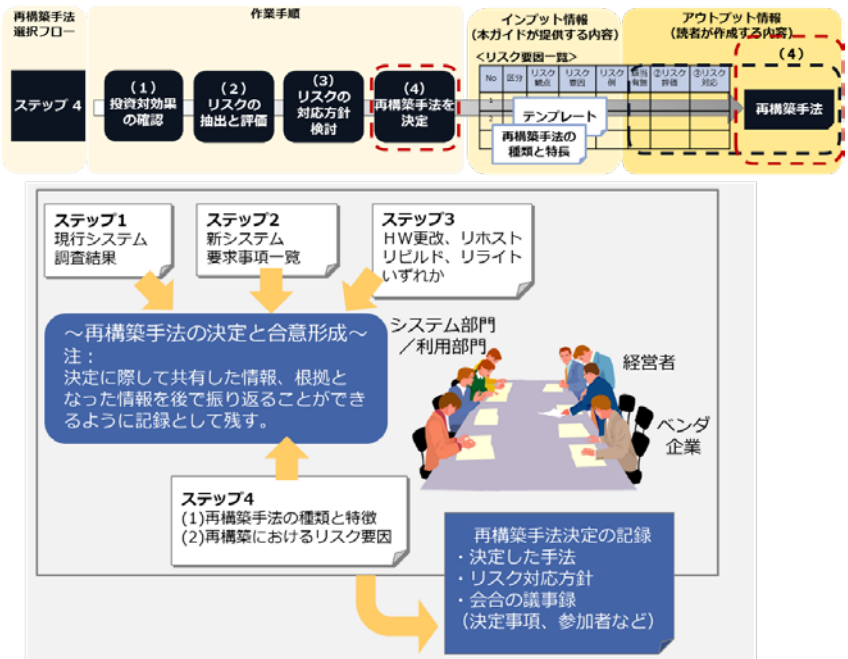


図 2.8 再構築手法決定場面イメージ

ケーススタディ

はじめに

2章の各ステップ全体を通した使い方の参考として、システム化企画段階の例題に基づいたケーススタディを示す。本ケースを各企業の状況に置き換え、本ガイドの使い方の参考にされたい。

【ケーススタディの概要】

大規模な基幹システムを短期間で再構築する場合の手法選択のケーススタディで、最終的に以下の(1)～(3)の内容で開発が進められた。

(1) 「システム運用のコスト削減を目的として、基幹システムのメインフレーム保守契約満了を期に、現行システムの資産・運用・ハードウェアをオープン化」

～再構築の目的～

- 運用コストの削減
- オープン環境への開発要員のシフト
- 運用の継続性を重視した短期間で安全なオープン化
- BCPサイトの構築（手法選択には関連性なし）

(2) メインフレームから、オープン（Unix）への移行

(3) オンラインはリビルド、バッチはリホストの2つの手法を選択

実施内容

（ステップ 1）現行システムの調査・分析

システム再構築例題における現行システム調査・分析のアウトプットである「現行システム調査結果報告書」を本ガイドの調査・分析観点に沿って記載したケースを図 2.9 に示す。

I. 現行資産

現行資産の調査結果を下表に示す。資産の調査・分析結果としては、長期に渡る運用・保守を経て一万本以上の資産になっており、大規模な再構築例である。

項　目	移行元	移行先	移行資産規模
OS	メインフレームOS	UNIX	
データベース	RDB,VSAM,DAM	Oracle	650ファイル
	順ファイル(PSファイル)	COBOL行順ファイル	7,000ファイル
言語	COBOL85	オープンCOBOL	14,000本(5Ms)
	アセンブラ	C or オープンCOBOL	90本(新規開発)
	JCL	Shell	12,000本

II. 業務知識

- 現行システムの操作マニュアルや設計ドキュメントは存在するが最新化されていない。
- 業務知識の範囲やカバレッジは現行保守要員（協力会社）を含めて、網羅している。（量的な面）
- 業務スキルの深さ（質的な面）は、初期開発時からかなり期間が経過していることもあり、充分とは言えないが、複数人で分担してカバーしている。

III. 処理方式

- メインフレームの ISV 製品によるシステム固有の基盤機能を使用して実現している。（JOB 実行基盤、帳票配信、圧縮ファイルなど）
- メインフレーム固有の媒体(CMT)や全銀手順など外部システム連携については、取引先との他方式への変更調整が必要となる。
- バックアップは、メインフレームに特有な世代管理ファイルを多く使用している。

IV. 非機能要件

- 現行メインフレームシステムと同等の処理能力を持たせる。
- 信頼性を保持するために処理分散、2 重化などを図る。
- 基幹サーバは、24 時間 365 日の稼動を前提とする。(監視体制を含む)

V. 業務データ

- 主要マスタ類は約 1000 万件、ピークトランザクション件数は約 150 万件/H。

VI. 保守状況

- メインフレームの COBOL85 をベースとした、開発・保守環境にて、アプリケーション保守を行っている。
- 現行システムの改修やレベルアップは、期初や期末など年に数回実施している。

図 2.9 「現行システム調査結果報告書」の例

（ステップ 2）新システムの要求事項分析

① 再構築の目的

本ケースを本ガイドの再構築テーマの中から選定すると以下の 3 つ。(手順(1))

(a) システム運用のコスト削減

⇒ 維持コスト削減

(b) 基幹システムのメインフレーム保守契約満了

⇒ ハードウェア、ソフトウェアの老朽化対応

(c) オープン環境への要員のシフト

⇒ 本ケースでは、要員不足対応と読み替える

② 新システムへの要求事項の分析

例題における以下の新システム要求事項とシステム化の目標を「各企業に固有な課題、要求事項」としてステップ 2 の要求事項例と突き合わせ、ないものは追加する（手順(2)(3))。

「新システム要求事項」

(a) 短期間で安全にオープン化する。(再構築期間は約 1 年 6 カ月） ⇒ 追加

(b) 仮想化・分散処理技術などの新 ICT を活用する。 ⇒ No.16

(c) 高性能かつ安価な機器を選定する。 ⇒ No.9

(d) 現行運用が継続でき、内部統制へ影響を与えない方式を採用する。 ⇒ 追加

(e) メインフレーム固有の入出力やインターフェースは、オープンでの代替方法に見直す。 ⇒ 追加

(f) 基幹システムを BCP 化し、業務継続性の確保を行う。 ⇒ 追加

「システム化の目標」

(a) 運用コストの低減 ⇒ No.4

(b) オープンシステムにおける画面(JavaServlet/JSP)、バッチ(shell/オープン COBOL)開発保守ノウハウの習得 ⇒ 追加

(c) オープンシステムにおける、システム(OS/DB)運用ノウハウの習得 ⇒ 追加

これらを再構築手法選択編のステップ 2 手順に沿って実施すると、以下のようになる。なお、本ケースでは、再構築の目的に BCP サイトの構築（No.5）もあるが、手法選択には影響しないため優先度の設定からは除外している（次の計画策定段階で対応計画などを検討する)。

- ステップ 2 手順(1)～(3)

No	区分	新システムへの要求事項	該当有無	優先度	再構築テーマ				
					老朽化対応	維持コスト削減	業務スリム化	柔軟性向上	要員不足対応
1	経営	業務プロセスの効率化（省力化、業務コスト削減）	◆			レ	レ		
2		事業競争力、営業力の強化	―					レ	
3		迅速な業績把握、情報把握（リアルタイム経営）	―				レ	レ	
4		開発・運用コストの削減	◆			レ	レ		
5		広域災害対応機能を強化したい（BCP）	◆						
6		セキュリティ機能の強化	―						
7		ベンダロックインを解消したい	―		レ	レ			
8		基盤要員の若手を育成し、既存言語の要員減へ対応したい	―						レ
9		一時的な再構築・移行コストは極力抑えたい	◆			レ			
10	業務	業務開発生産性、アプリケーションの保守性を向上させたい	◆			レ		レ	
11		オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい	◆					レ	
12		バッチ業務は、基幹系処理であり、業務仕様は踏襲したい	◆						レ
13		業務プロセスのスピードアップ（リードタイム短縮など）	―				レ	レ	
14	ITシステム	最新のIT技術や新機能を使って、運用の効率化、自動化を図りたい	―		レ			レ	
15	ITシステム	現行システムの堅牢な基盤を継続して利用したい	―		レ				
16		現行システムの老朽化した基盤から、最新のインフラに変更したい	◆		レ	レ			
17		現行システムより可用性や信頼性を高めるため、冗長化を実現した新たな基盤に乗せ換えたい	◆					レ	
18		現行の開発言語の保守要員の定年から、新しい言語に変更したい	◆			レ			レ
19		開発・保守環境は維持したいので、開発言語は変更する必要ない	―		レ				
20		ホストのセンタオペレーション負荷を削減したい	―			レ			
21		システム運用や監視方法を簡易化したい	―			レ		レ	
22		WSの画面・メニュー構成や操作性（ユーザインターフェース）を改善したい	◆					レ	
追加	経営	短期間で安全にオープン化する（再構築期間は約1年6カ月）	◆		レ				
追加	経営	オープン環境の開発保守ノウハウやシステム運用ノウハウを習得する	◆		レ				レ
追加	業務	現行運用が継続でき、内部統制へ影響を与えない方式を採用する	◆						
追加	ITシステム	ホスト固有の入出力やインターフェースは、代替方法に見直す	◆		レ				

図 2.10 ステップ 2 手順(1)～(3)

- ステップ 2 手順(4)

(4) 要求内容の重要性、緊急性などにより、優先付けを行い、アウトプットする。
ケーススタディでは、2段階（最重要、重要）を例示している。

＜凡例＞優先度(◎:必須、○:推奨、△:可能ならば)

No	区分	新システムへの要求事項	該当有無	優先度	再構築の目的				
					老朽化対応	維持コスト削減	業務スリム化	柔軟性向上	要員不足対応
1	経営	業務プロセスの効率化(省力化、業務コスト削減)	◆	◎		レ	レ		
4		開発・運用コストの削減	◆	◎		レ	レ		
9		一時的な再構築・移行コストは極力抑えたい	◆	◎		レ			
追加		短期間で安全にオープン化する	◆	◎	レ	レ			レ
追加		オープン環境の開発保守ノウハウやシステム運用ノウハウを習得する	◆	○					レ
10	業務	業務開発生産性、アプリケーションの保守性を向上させたい	◆	○		レ		レ	
11		オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい	◆	○				レ	
12		バッチ業務は、基幹系処理であり、業務仕様は踏襲したい	◆	○					レ
追加		現行運用が継続でき、内部統制へ影響を与えない方式を採用する	◆	○		レ			
16	IT システム	現行システムの老朽化した基盤から、最新のインフラに変更したい	◆	◎	レ	レ			
18		現行の開発言語の保守要員の定年から、新しい言語に変更したい	◆	○		レ			レ
22		WSの画面・メニュー構成や操作性(ユーザインターフェース)を改善したい	◆	○				レ	
追加		ホスト固有の入出力やインターフェースは、代替方法に見直す	◆	○	レ	レ			

図 2.11 ステップ 2 手順(4)

（ステップ 3）再構築手法の選択

本ケースは、オンラインとバッチでシステムの特性が異なる。そのため、それぞれサブシステム毎に分けて再構築手法の選択要素に基づき評価し、候補選定する使用例を示す。

① 使用例 1（オンライン）

ステップ 2 の新システム要求事項一覧をオンライン関連のみとし（バッチ業務の要素 No. 12 を除外）、再構築手法の選択要素に基づき評価したものを、表 2.7 に示す。

表 2.7 再構築手法の候補選択評価（オンライン）

ステップ2 からのインプット情報　　＜凡例＞優先度(◎:必須、○:推奨、△:可能ならば)

No	区分	新システムへの要求事項	該当有無	優先度	業務仕様変更		基盤変更		言語変更	
					あり	なし	あり	なし	あり	なし
1	経営	業務プロセスの効率化(省力化、業務コスト削減)	◆	◎	レ					
4		開発・運用コストの削減	◆	◎	レ		レ			
9		一時的な再構築・移行コストは極力抑えたい	◆	◎		レ		レ		レ
追加		短期間で安全にオープン化する	◆	◎		レ	レ			レ
追加		オープン環境の開発保守ノウハウやシステム運用ノウハウを習得する	◆	○			レ			
10	業務	業務開発生産性、アプリケーションの保守性を向上させたい	◆	○			レ			
11		オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい	◆	○	レ					
12		バッチ業務は、基幹系処理であり、業務仕様は踏襲したい	◆	○		レ				
追加		現行運用が継続でき、内部統制へ影響を与えない方式を採用する	◆	○		レ				
16	ITシステム	現行システムの老朽化した基盤から、最新のインフラに変更したい	◆	◎			レ			
18		現行の開発言語の保守要員の定年から、新しい言語に変更したい	◆	○					レ	
22		WSの画面・メニュー構成や操作性(ユーザインターフェース)を改善したい	◆	○	レ		レ		レ	
追加		ホスト固有の入出力やインターフェースは、代替方法に見直す	◆	○	レ		レ			
		◎:必須(2ポイント)、○:推奨(1ポイント)として集計			7	5	10	2	2	4

ケース1では、「業務仕様を変更する」、「基盤を変更する」、「言語変更は変更なし」手法 ⇒ 「リビルド」

上記の結果から、ケース 1（オンライン）では、「業務仕様を変更する」「基盤を変更する」「言語変更はなし（任意）」であるため、前述の基本パターンをもとに判断する手法は、図 2.12 の通り「リビルド」となる。

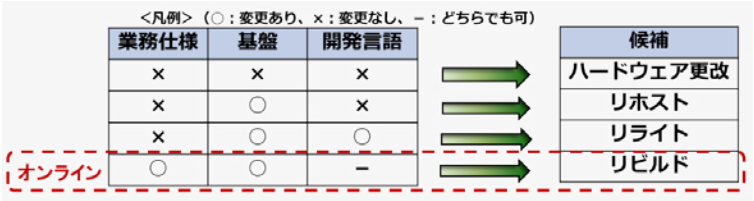


図 2.12 再構築手法の選択（オンラインの例）

② 使用例 2（バッチ）

ステップ 2 の要求事項一覧をバッチ関連とし（オンライン関連の No. 11、22 を除外）したものを表 2.8 に示す。

表 2.8 再構築手法の候補選択評価（バッチ）

ステップ2 からのインプット情報　　＜凡例＞優先度(◎:必須、○:推奨、△:可能ならば)

No	区分	新システムへの要求事項	該当有無	優先度	業務仕様変更		基盤変更		言語変更	
					あり	なし	あり	なし	あり	なし
1	経営	業務プロセスの効率化(省力化、業務コスト削減)	◆	◎	レ					
4		開発・運用コストの削減	◆	◎	レ		レ			
9		一時的な再構築・移行コストは極力抑えたい	◆	◎		レ		レ		レ
追加		短期間で安全にオープン化する	◆	◎		レ	レ			レ
追加		オープン環境の開発保守ノウハウやシステム運用ノウハウを習得する	◆	○			レ			
10	業務	業務開発生産性、アプリケーションの保守性を向上させたい	◆	○			レ			
11		オンライン業務は、新サービス対応、新チャネル追加のため業務仕様を見直したい	◆	○	レ					
12		バッチ業務は、基幹系処理であり、業務仕様は踏襲したい	◆	○		レ				
追加		現行運用が継続でき、内部統制へ影響を与えない方式を採用する	◆	○		レ				
16	ITシステム	現行システムの老朽化した基盤から、最新のインフラに変更したい	◆	◎			レ			
18		現行の開発言語の保守要員の定年から、新しい言語に変更したい	◆	○					レ	
22		WSの画面・メニュー構成や操作性(ユーザインターフェース)を改善したい	◆	○	レ		レ		レ	
追加		ホスト固有の入出力やインターフェースは、代替方法に見直す	◆	○	レ		レ			
		◎:必須(2ポイント)、○:推奨(1ポイント)として集計			5	6	9	2	1	4

ケース2では、「業務仕様は変更なし」、「基盤を変更する」、「言語変更は変更なし」手法 ⇒ 「リホスト」

上記のケース 2（バッチ）では、「業務仕様は変更なし」、「基盤は変更する」、「言語変更は変更なし」であるため、前述の基本パターンをもとに判断する手法は、図 2.13 の通り「リホスト」となる。

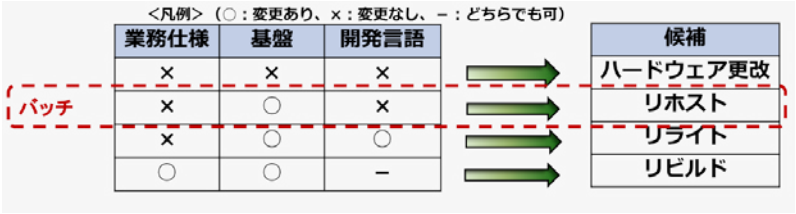


図 2.13 再構築手法の選択（バッチの例）

（ステップ 4）再構築手法の決定

① 投資と効果やメリット・デメリットの確認

本ケースの再構築による効果と投資をあげると、以下のような内容になる。

＜再構築による効果＞

- 大幅なコスト削減
- 将来のクラウド化、サーバ統合、データベース統合などによる更なるコスト削減の可能性
- メインフレーム系技術者涸渇からの解放、オープン技術、パッケージ利用などによるシステム開発費の削減
- 業務、使途に合わせたスケーラビリティーの増加（BCP サイト構築）･･･（手法選択には関連なし）

＜再構築に必要な投資＞

- オープン系技術研修、作業体系の再構築
- オンライン処理プログラムの組み換えを通じて新技術研修を実施する
- サーバ系の開発ツールを導入し、開発・運営効率を確保する

② 再構築リスクの共有

＜本ケースにおける想定リスク＞

- スケジュールが短期間であり、本稼動時期の遵守が可能か（スケジュール遅延リスク）
- オンライン（リビルド）の開発品質、バッチ（リホスト）の資産変換品質の確保(現行踏襲の充分性)
- オープン運用技術の適用範囲と開発範囲からのコスト増懸念（アセンブラや運用代替機能の見極め）
- 大規模な再構築（リビルト＆リホスト）を行う上での技術リスク

(a) リスクの抽出

リスク要因例の中から、上記の想定リスクに該当する内容を抽出する。表 2.9 に抽出した内容を示す。

表 2.9 再構築におけるリスク要因例

＜凡例＞ リスク評価:リスクが顕在化する相対的な度合い(◎:高、○:中、△:低)

No	区分	リスク観点	再構築におけるリスク要因	リスク例	①該当有無	②リスク評価		
						リホスト	リライト	リビルド
1	要求	現行踏襲(機能)	• 現行から受け継ぐべき業務運用や画面・帳票、データ仕様、外部インタフェースなどの対象や範囲は明確か？	• 作り込み品質の不良 • サービス開始の延伸 • 作業工数の増大	あり	○		◎
2	要求	現行踏襲(非機能)	• 移行先のシステムの運用方法や性能、操作性、可用性、信頼性などの非機能要件は明確か？	• テスト、運用段階になり、利用者から「使えない」、「レベルダウン」とのクレームが起こる	あり	○		○
3	要求	再構築サービスの選定	• 適用するサービスやソリューションは、現行と同じ機能、運用を全て実現できるか？(特にリホストの場合) • 役割分担や制限、制約事項は明確になっているか？	• 想定どおり動作しない不具合の多発による手戻り • サービス開始の延伸 • 作業工数の増大	なし	—		—
4	要求	アーキテクチャ	• 最新技術の適用や初めてのアーキテクチャ採用時には、充分な適用検討と技術検証を行っているか？ • アーキテクチャが混在することはないか?(例.現新混在)	• 想定どおり動作しない不具合の多発による手戻り、対応工数の増大 • 保守要員の増加 • 保守環境の複雑化	あり	○		○
5	現状	システムの規模	• 現行システムは肥大化・複雑化していないか？ (例えば、未稼働資産が多数存在する)	• 業務仕様の把握不足による手戻り • 作業工数の増大	あり	○		△
6	現状	業務知識不足(要員)	• 現行業務有識者は、後継者不足になっていないか？(現行システム仕様は掌握できるか) • 再構築時の要員は確保可能か？	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • 試験による検証の長期化 • サービス開始の延伸 • 作業工数の増大	あり	△		◎
7	現状	業務知識不足(ドキュメント)	• 業務ドキュメントとアプリケーションは一致しているか？ • ブラックボックス化により影響調査の精度が低下していないか？	• 業務仕様の把握不足による手戻り • 作り込み品質の不良 • サービス開始の延伸 • 作業工数の増大	あり	△		◎
8	現状	資産管理	• 現行稼動中資産と移行資産のアンマッチ(例.実行モジュールとソースコードのアンマッチ) • 未検証資産の移行(開発中資産)は、ないか？	• 想定どおり動作しない不具合発生による手戻り、対応工数の増大	なし	—		—
9	現状	データ移行	• データ仕様の把握不足によるデータ変換の誤り(フォーマット、文字コード) • 異常データの混入は考えられないか？	• 現行システムと新システムで結果が一致しない • 移行データ調査や対応工数の増大	なし	—		—

(b) リスク対応方針の検討

前記のリスク要因に対する対応方針を、ステークホルダ間で検討する。

リビルド（オンライン）、リライト（バッチ）それぞれに対する、対応方針の検討結果を表 2.10 に示す。

表 2.10 再構築におけるリスク対応例

＜凡例＞ リスク評価、リスク対応：リスクが顕在化する相対的な度合い(◎:高、○:中、△:低)

No	区分	リスク観点	再構築におけるリスク要因	リスク例	①該当有無	②リスク評価		③リスク対応			
						リホスト	リビルド	回避	軽減	受容	転嫁
1	要求	現行踏襲（機能）	・現行から受け継ぐべき業務運用や画面・帳票、データ仕様、外部インターフェースなどの対象や範囲は明確か？	・作り込み品質の不良 ・サービス開始の延伸 ・移行作業工数の増大	あり	○	◎	[リビルド◎][軽減策] 開発技術者の育成度評価を行い、第3者チームを加え共同でオンライン開発の推進可否判断を実施する。 [リホスト○][軽減策] 設計内容反映した変換ツールによるプロトタイプ検証を実施することによる変換仕様品質を確保する。 ※スケジュールリスクについては、設計工程完了時に本稼働の見直し判断を実施する。			
2	要求	現行踏襲（非機能）	・移行先のシステムの運用方法や性能、操作性、可用性、信頼性などの非機能要件は明確か？	・テスト、運用段階になり、利用者から「使えない」、「レベルダウン」とのクレームが起こる	あり	○	○	[リホスト、リビルド○][軽減策] 設計工程で当初想定のオープン系運用ツール、機能をエンドユーザに説明し、納得性を高めていく。			
3	要求	再構築サービスの選定	・適用するサービスやソリューションは、現行と同じ機能、運用を全て実現できるか？（特にリホストの場合） ・役割分担や制限、制約事項は明確になっているか？	・想定どおり動作しない不具合の多発による手戻り ・サービス開始の延伸 ・作業工数の増大	なし	－	－				
4	要求	アーキテクチャ	・最新技術の適用や初めてのアーキテクチャ採用時には、充分な適用検討と技術検証を行っているか？ ・アーキテクチャが混在することはないか?（例. 現新混在）	・想定どおり動作しない不具合の多発による手戻り、対応工数の増大 ・保守要員の増加 ・保守環境の複雑化	あり	○	○	[リホスト、リビルド○][軽減策] ・設計段階で、処理方式、運用方式の検証を目的とした、パイロット検証を実施する。			
5	現状	システムの規模	・現行システムは肥大化・複雑化していないか？ （例えば、未稼働資産が多数存在する）	・業務仕様の把握不足による手戻り ・作業工数の増大	あり	○	△	[リホスト○][軽減策] ・商用運用の実績を再度調査し、未使用資産の廃止などスリム化を検討する。			
6	現状	業務知識不足（要員）	・現行業務有識者は、後継者不足になっていないか？（現行システム仕様は掌握できるか） ・再構築時の要員は確保可能か？	・業務仕様の把握不足による手戻り ・作り込み品質の不良 ・試験による検証の長期化 ・サービス開始の延伸 ・作業工数の増大	あり	△	◎	[リビルド◎][軽減策] オンラインは、リビルドする過程の設計～テストの中で、有識者から業務仕様を継承し、把握していく。			
7	現状	業務知識不足（ドキュメント）	・業務ドキュメントとアプリケーションは一致しているか？ ・ブラックボックス化により影響調査の精度が低下していないか？	・業務仕様の把握不足による手戻り ・作り込み品質の不良 ・サービス開始の延伸 ・作業工数の増大	あり	△	◎	[リビルド◎][軽減策] オンラインは、リビルドする過程の再設計の中で、ドキュメントの整備を図る。			
8	現状	資産管理	・現行稼動中資産と移行資産のアンマッチ（例.実行モジュールとソースコードのアンマッチ） ・未検証資産の移行（開発中資産）は、ないか？	・想定どおり動作しない不具合発生による手戻り、対応工数の増大	なし	－	－				
9	現状	データ移行	・データ仕様の把握不足によるデータ変換の誤り（フォーマット、文字コード） ・異常データの混入は考えられないか？	・現行システムと新システムで結果が一致しない。 ・移行データ調査や対応工数の増大	なし	－	－				

③　再構築手法の決定

本ケースにおける再構築手法の決定に際して、共有した情報と主に協議した内容、判断の根拠となった情報は以下であり、これを記録として残す。主な協議内容は以下の通り。

(a) 現行システムの調査・分析結果による、現状の認識合わせ

(b) 新システム要求事項の優先度が再構築の目的と方向性が合致しているか、選択した再構築手法と整合しているか

(c) 再構築手法のメリット・デメリットや再構築への投資と構築後の効果

(d) 再構築において発生しうるリスク要因の共有とリスクへの対応方針

ステークホルダ間で共有した資料、記録として残す資料は以下の通り。

(a) 現行システムの調査・分析（ステップ 1）の結果

- 現行システム調査結果報告書

(b) 新システムの要求事項分析（ステップ 2）の結果

- 「表 2.7　再構築手法の候補選択評価（オンライン）」の新システム要求事項一覧

  （注）ステップ 2 手順（4）のアウトプット

(c) 再構築手法の選択（ステップ 3）の結果

- 「表 2.8　再構築手法の候補選択評価（バッチ）」の再構築手法の候補選択評価

(d) 再構築による投資対効果確認およびリスクの共有

- 再構築による効果と必要な投資
- 「表 2.9 再構築におけるリスク要因例」のシステム化リスク要因一覧

(e) 再構築手法の決定とリスク共有会合の議事録

- 日時、場所、参加者（経営者、利用部門、業務部門、システム部門、ベンダ企業の代表者）
- 決定事項

  再構築の手法は、「オンライン業務はリホスト、バッチ業務はリホストを採用する」
- 共有したリスク対応方針

  「表 2.10 再構築におけるリスク対応例」のシステム化リスク対応方針

システム再構築リスクの共有と手法決定時の合意形成の場面イメージを図 2.14 に示す。

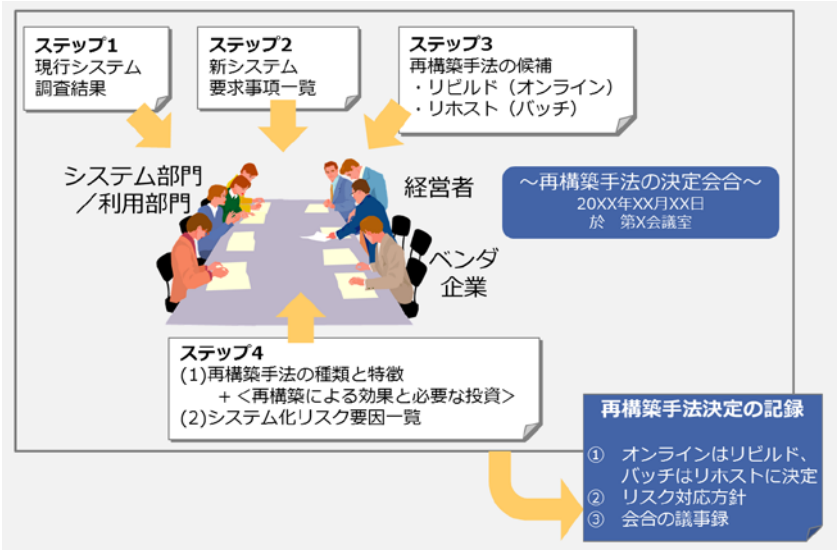


図 2.14 再構築手法決定会合の場面イメージ

第３章　計画策定編

3.1 計画策定編概要

計画策定編とは

システム再構築のシステム化計画の策定時に検討し当初から対応施策項目を計画すべき観点を説明する。観点は8つ（A～H）に分けて定義している。各観点の検討・実施内容が再構築手法によって異なる場合は、各観点内で手法ごとに説明する。

8つの観点には、「2.5 再構築手法の決定（ステップ4）」で挙げたリスクの防止策となるものと、ステップ4のリスクに関係なく検討することを推奨するものがある。ステップ4のリスクに対応する観点がどれにあたるかについては、「3.1 計画策定編概要」の「再構築手法選択編リスクとのマッピング」で記載する。

なお、業務要件の変更／追加を行うケースの対応については「3.10 業務要件の変更／追加への対応」にまとめた。8つの観点に加えて参照してほしい。

狙い

再構築に特有な工期・コストおよび品質保証に影響するリスクの予防策をシステム化計画時に検討・実施することで、開発工程での手戻りや工期・工数の見積りの甘さを防止する。

また、開発工程で発生しがちな、要件や品質保証に関するユーザ企業・ベンダ企業間の認識齟齬の防止にもつなげる。

利用シーン

3章の内容は、図3.1に示すように、システム再構築の企画工程において、システム化計画を立案する際に利用する。

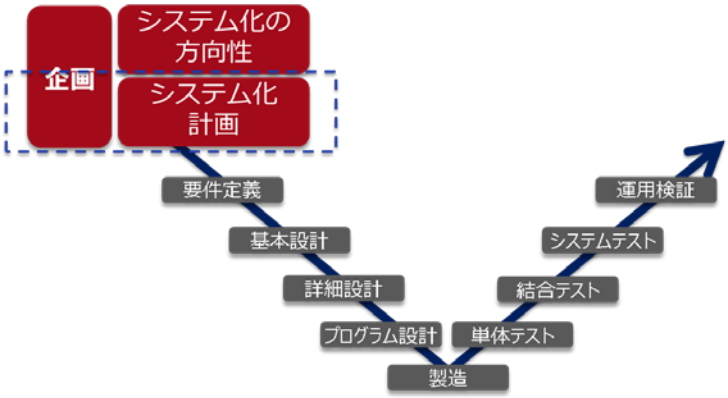


図 3.1 利用シーンのイメージ

利用者

3 章の内容は、システム再構築の企画工程において、システム化計画の策定に関わるシステム部門、利用部門、ベンダ企業などのステークホルダが利用する。

利用方法

利用者は、3.2 節以降の観点 A～H の内容、および業務要件の変更／追加を行う場合は 3.10 節を元に必要な検討を実施し、システム化計画内容に反映する。

＜注意点＞

本ガイドではシステム再構築に特有な検討・実施観点を記載しているため、新規開発でも共通する観点については、共通フレームや利用者の自社が保有するシステム化計画策定に関する規定などの一般的な標準をもとに、別途検討が必要である。

「2 章 再構築手法選択編」との関連、および「3 章 計画策定編」を利用したシステム化計画立案の流れを図 3.2 に示す。

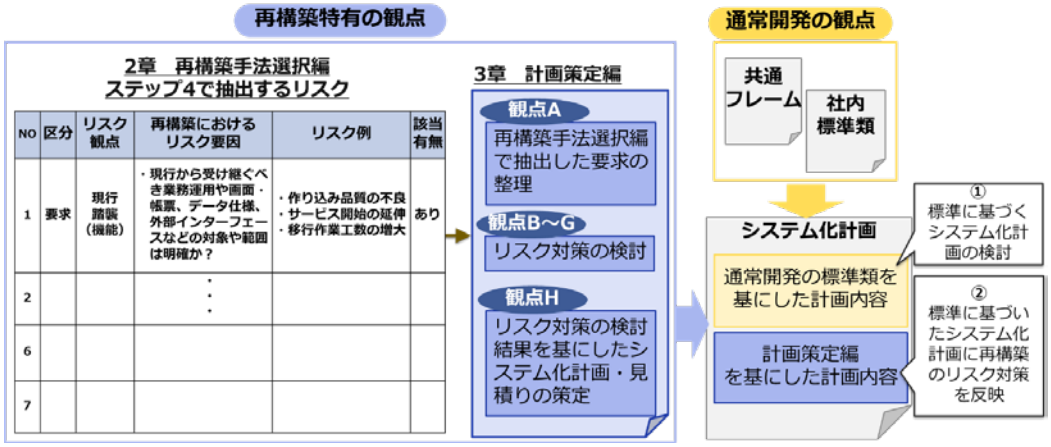


図 3.2 計画策定編の利用イメージ

インプット情報

「2 章 再構築手法選択編」ステップ 1～ステップ 4 のアウトプット情報
現行システムのシステム構成、業務、機能に関する情報
新システムの開発に利用する技術、製品に関する情報　など

アウトプット情報

システム化計画に反映するリスク予防策の内容
（具体的な内容は、後述の各観点の説明を参照）

再構築手法選択編リスクとのマッピング

「2.5 再構築手法の決定（ステップ 4）」で抽出したリスク「表 2.5 再構築におけるリスク要因テンプレート」の対応が計画策定編のどの観点に対応するかについて以下の図 3.3 に示す。

なお、観点 A と H については、特定のリスク対応策のための観点ではないため、マッピングの対象外としている。

		計画策定編　観点							
再構築手法選択編 ステップ4「再構築におけるリスク要因（ハザード）」		A	B	C	D	E	F	G	H
リスク観点	再構築におけるリスク要因（ハザード）	要求の確認	現行踏襲内容の明確化	現行資産活用方針の検討	現行業務知識不足への対応	品質保証の検討	意思決定プロセスの策定	データ移行の計画	再構築の計画と見積り
現行踏襲（機能）	現行から受け継ぐべき業務運用や画面・帳票、データ仕様、外部インターフェースなどの対象や範囲は明確か？		○			○			
現行踏襲(非機能)	移行先のシステムの運用方法や性能、操作性、可用性、信頼性などの非機能要件は明確か？		○			○			
再構築サービスの選定	運用するサービスやソリューションは、現行と同じ機能、運用をすべて実現できるか？（特にリホストの場合）		○			○			
	役割分担や制限、制約事項は明確になっているか？						○		
アーキテクチャ	最新技術の適用や初めてのアーキテクチャ採用時には、十分な適用検討と技術検証を行っているか？		○			○			
	アーキテクチャが混在することはないか?（例：新旧混在）		○			○			
システムの規模	現行システムの肥大化・複雑化していないか？				○				
業務知識不足（要員）	現行業務有識者は、後継者不足になっていないか？（現行システム仕様は掌握できるか）				○				
	再構築時の要員は確保可能か？				○				
業務知識不足（ドキュメント）	業務ドキュメントとアプリケーションは一致しているか？		○	○	○				
	ブラックボックス化により影響調査の精度が低下していないか？				○				
資産管理	現行稼動中資産と移行資産のアンマッチはないか？（例：実行モジュールとソースコードのアンマッチ）		○	○	○				
	未検証資産の移行（開発中資産）はないか？			○					
業務データ移行	データ仕様の把握不足によるデータ変換の誤り（フォーマット、文字コード）や異常データの混入は考えられないか？							○	

図 3.3 再構築手法選択編で抽出したリスクと対応策のマッピング例

3.2 要求の確認（観点 A）

目的

再構築手法選択で抽出した新システムへの要求を、システム化計画立案にあたり再度整理し、ユーザ企業内で合意することを目的とする。

再構築手法選択編において、手法決定に至るまでに新システムの要求事項分析（ステップ 2）～再構築時のリスク共有と合意形成（ステップ 4）が繰り返され、当初の要求事項に対し、取捨選択（変更・追加を含む）・優先度設定が実施される。その結果、図 3.4 に示すように手法決定時点の要求事項は当初より集約されたものとなることが予想される。

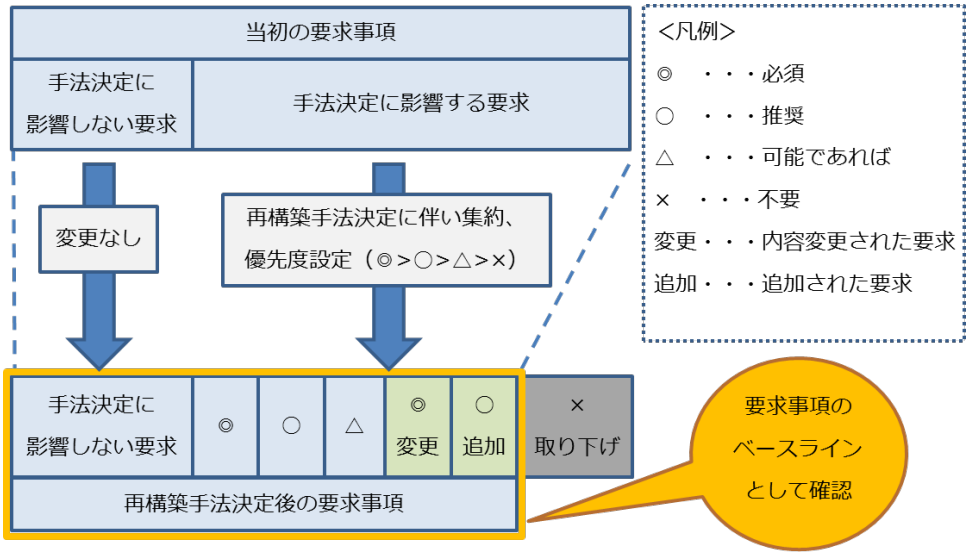


図 3.4 要求の集約(再構築手法選択編)のイメージ

よって、手法決定時点では、ユーザ企業内のステークホルダ間で認識齟齬が発生している可能性がある。ステークホルダ間で認識齟齬が存在したままとなった場合、後々、「こんなはずではなかった」、「これも入れてほしい」などの要求事項の変更や追加の依頼が各ステークホルダからバラバラと発生することが想定される。その発生がプロジェクト開始後ともなると、大幅な手戻りとなり、さらにはシステム化計画から見直しが必要となる場合もある。

そのような事態を避けるため、RFP 作成に向けてユーザ企業内の各ステークホルダ全員が、最終的に確定した全ての要求事項の内容と集約結果に認識齟齬がないか確認する。そして、

その時点での要求事項が、新システムに対する要求事項のベースラインである旨を合意する必要がある。

タスク

下記(1)に「要求の確認（観点 A)」の観点におけるタスクとその該当工程を示す。

(1) 要求事項確認の場を設定

企画	要件定義	設計	製造	試験	リリース後
(1)					

タスクの内容

(1) 要求事項確認の場を設定

再構築手法選択後、RFP 作成前に、ユーザ企業内で改めて要求事項を確認するための場を設定する。

要求事項の確認において検討事項が発生した場合は、RFP 作成前に解決しておく。検討事項によっては手法選択に戻っての検証も考慮する必要がある場合もある。

要求事項の確認が完了し、ユーザ企業内の各ステークホルダ間で合意に達したならば、その記録を文書として残す。

タスクのアウトプット

本タスクの実施により、システム化計画向けに期待されるアウトプットは以下の通り。

- 合意済みの要求事項一覧

見積りに反映すべき項目

「要求の確認（観点 A)」の観点においては特になし

3.3　現行踏襲内容の明確化（観点 B）

目的

「現行踏襲」の内容について曖昧な部分があるために、その部分の現行踏襲内容をベンダ企業が認識できず、試験工程になって初めて実現できていないと判明し問題プロジェクト化するケースが多い。

現行の業務を継続するために「現行踏襲」することは必然的な要求であるが、実際に新システムで「現行」を再現するためには、現行システムの「何を」「どのような状態に」したいのかを明確化する必要がある。

ユーザは「動作している現行システム」がそのまま踏襲されることを期待するが、ベンダは実際に開発するために「現行通り」を仕様化する際に、設計書などのドキュメントに記載されている要件や仕様、ソースコードに実装されている内容を「現行踏襲」の拠り所の一つとする。

しかし、ソースコードに実装されているものとドキュメントに乖離がある場合、どちらの情報が正しいのかを見極めないと誤った「現行通り」になってしまう。また、システム全ての仕様がドキュメント化されていないことが原因となり、ユーザ企業とベンダ企業の両者が拠り所とする「現行踏襲」にギャップが生じてしまうこともある。ユーザ企業とベンダ企業の間だけではなく、ユーザ企業の中でも、システム部門と利用部門とでは、ギャップが生じるケースもある。これらのギャップのイメージを図 3.5 に示す。

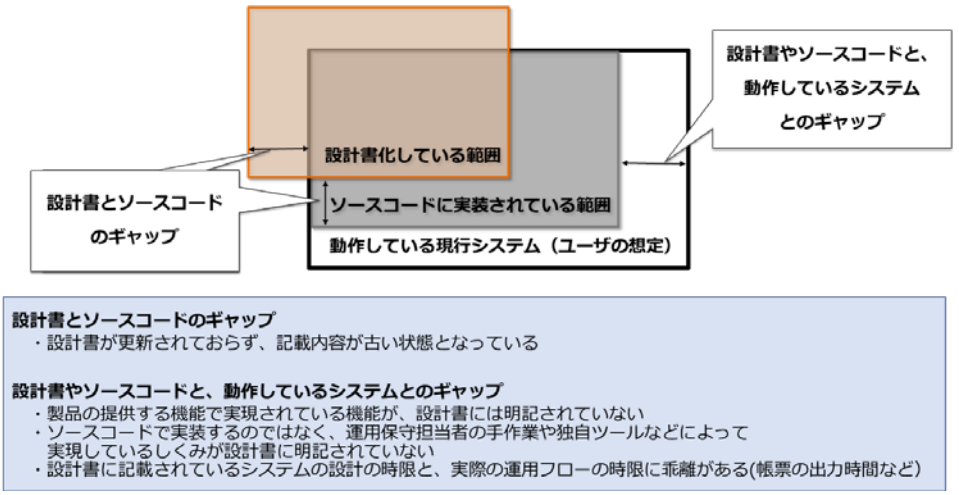


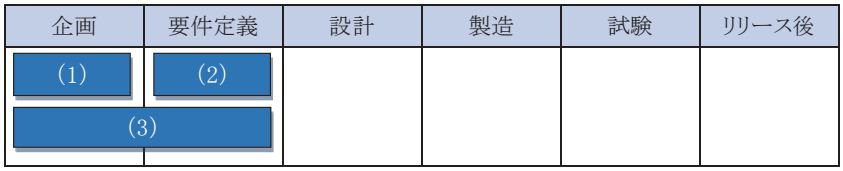
図 3.5 「現行踏襲」の拠り所のギャップ

そのため本節では、曖昧さを残した現行踏襲内容による問題プロジェクト化を防ぐため、ユーザ企業とベンダ企業の「現行踏襲」のギャップを踏まえたうえで、段階的に現行踏襲内容を明確化し曖昧さを排除する考え方を説明する。

タスク

下記(1)～(3)に「現行踏襲内容の明確化（観点 B)」の観点におけるタスクとその該当工程を示す。

(1) 企画工程での明確化
(2) 要件定義工程での明確化の計画
(3) 追加コスト・時間がかかる部分への対応判断のためのプロセス



タスクの内容

(1) 企画工程での明確化

企画工程で、機能要件、非機能要件に対して現行踏襲内容（現行から変えるのか、変えないのか）を明確化する。明確化の検討対象となる要件と内容の例を下表に示している。検討結果を RFP やシステム化計画ドキュメントに記載する。

その際、図 3.5 の各ギャップで示したように、設計書と実装に乖離があったり、ステークホルダによって認識する「現行」が異なる可能性があることから、どのような状態になることを期待するのかも明確にする。

現行業務知識のブラックボックス化などの理由で、現行踏襲内容明確化が困難な場合には、意思決定プロセスで新システムでの仕様を決定するという対応も必要となる。（「3.8 意思決定プロセスの策定（観点 F)」参照）

それでも企画工程で明確化ができない部分が残ってしまう場合には、そのような部分が存在する点と、要件定義工程での解消方針を RFP に記載することが重要である。

ポイント！

- ユーザ企業やベンダ企業といった各ステークホルダと調整・合意した内容を明確化すること。
- 機能要件のみではなく、必ず非機能要件も対象とすること。
- 本節の取り組みによって、設計書やソースコード、システム間の情報の乖離を是正（適切な「現行」を明確化）する。再構築プロジェクトにて整理した情報について、プロジェクトの終了後に再び乖離が発生することは多いため、それを防ぐような対策を検討することも重要である。

表 3.1　明確化対象となる要件の例（1/2）

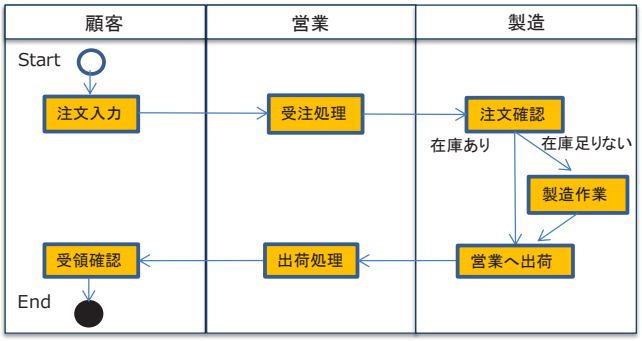
要件		内容
機能要件	業務フロー	・機能の流れと処理概要 ・各機能の入力データ・出力データは一致 ・各機能内の処理実装内容 （例）システム化業務フロー 処理の流れと入力・出力データは「変えない」などを検討 
	画面	・登録・更新系ボタン押下時の処理 ・画面項目の内容、レイアウト（項目の配置）デザイン ・タブ順、IME 制御、画面初期表示位置
	帳票	・レイアウト、フォント、印字位置 ・印刷時間、PDF の表示時間
	データ	・データレイアウト ・外字、入力可能文字の取り扱い
	外部インターフェース	・他システムとやり取りする電文、帳票のフォーマット ・文字コードの種類 ・送受信の時間帯、タイミング

表 3.1　明確化対象となる要件の例（2/2）

要件		内容
非機能要件	性能	・バッチ処理実行時間　※現行実行スケジュールで完了 ・オンライン処理のレスポンス　※複数同時使用時も含む
	信頼性	・冗長構成 ・サーバ片系停止時の切り替え

(2) 要件定義工程での明確化の計画

(1) で明確化できなかった部分を解消するために、ベンダ企業などと協力して対象の現行仕様を調査し、現行踏襲内容を明確に要件定義書に記載する。

（現行仕様の調査は「3.5 現行業務知識不足への対応（観点 D)」参照）

(3) 追加コスト・時間がかかる部分への対応判断のためのプロセス

設計工程で検討の詳細化が進むにつれて、現行踏襲にあたっては追加コスト・時間が必要となることが判明する場合がある。例えば、その場合、該当箇所の現行踏襲の重要度とのトレードオフを考慮して対応を判断する必要がある。

① 実施判断は意思決定プロセスに従う

現行踏襲を追加コスト・時間をかけて実施するかどうかの判断は、意思決定プロセス（「3.7 意思決定プロセスの策定（観点 F)」参照）で実施するものとする。判断においては必ず利用部門のステークホルダを体制に含めておき、認識齟齬が発生しないようにすること。

対応方針としては、以下の表 3.2 に示す 3 通りが想定される。

表 3.2 対応方針の例（1/2）

対応方針(例)	対応を選択するシチュエーション
現行踏襲しない	画面レイアウトやフォントの変更など、運用にあまり影響がない場合など (例)現行システムの帳票で使用していたフォントは新システムでは使用できないが、運用上影響がないため、新システムで利用可能なフォントを採用した。
現行踏襲しない （運用を変更する）	運用に影響はあるが、許容可能な範囲で運用を変更することで対応可能な場合や、追加コスト・時間が許容できない場合 (例)新システムでは現行システムとバックアップ・リカバリの方式が変更となるため、運用ジョブ・手順を変更した。

表 3.2 対応方針の例（2/2）

対応方針(例)	対応を選択するシチュエーション
コストと時間をかけて現行踏襲を実現する	外部システムとのインターフェース部分など、現行踏襲が実現されなければ業務の継続が困難になる場合、かつ追加コスト・時間が許容できる場合 (例)帳票ソフトウェアの変更に伴い、新システムで出力する顧客向け帳票のレイアウトが現行システムと異なることが判明したが、変更に伴う影響が大きいことから現行レイアウトで出力できるようにカスタマイズを実施した。

② 該当部分の早期発見への取組みを計画する

プロジェクトへの影響を最小とするには、要件定義から設計工程までの設計レビューでの発見が最も効果的であり、まずはそちらに注力することが望ましい。

しかし、試験工程で実際に実行してみないと発見できないものも存在する。それらの発見時期を早めるには、現新比較試験などの現行踏襲を確認できるテストを早期に実施することが効果的であると考えられる。

そのため、プロジェクトへの影響が大きい「変えない」部分を中心に、現行踏襲を確認可能なテストデータ・テストケースを早めにベンダ企業に提供しておき、テストの早期実施を可能とする対応を計画することが有効であるといえる。

ポイント！

- **現行踏襲内容は設計ドキュメントに仕様として明記**

ユーザ企業内の各ステークホルダやベンダ企業との間で、現行踏襲内容に認識齟齬が発生しないように、現行踏襲内容は必ず設計ドキュメントに仕様として記載し、その内容を正とする旨を各ステークホルダで合意しておく必要がある。

- **レビュー・承認時の注意点**

ユーザ企業側は主に設計ドキュメントのレビュー・承認を行うこととなるが、その記載内容がすなわち現行踏襲内容となり以降のテストのよりどころとなるため、レビュー実施と承認にあたっては責任を持って実施すること。

- **利用部門との認識齟齬を軽減**

利用部門に現行踏襲内容の確認を行っていたとしても、運用検証で利用部門が実物を触る段階になって、「こんなはずではなかった」という認識齟齬が明らかとなるケースがある。その認識齟齬を軽減するために、現行踏襲内容が変更になる際の利用部門へのレビュー実施や、サンプルプログラム（プロトタイプ）の早期提供による使用感の確認実施などの取組みをシステム化計画に組み込むことが有効である。

タスクのアウトプット

本タスクの実施により、システム化計画向けに期待されるアウトプットは以下の通り。

(1) 現行踏襲内容の検討結果
- 現行踏襲を明確化した要件、仕様
- 現行踏襲が曖昧または不明なところはどこか

(2) 追加コスト・時間がかかる部分への対応判断のためのプロセスの検討結果
- 意思決定プロセス体制
- 早期発見のための対応計画

(3) 利用部門との認識齟齬軽減のためプロトタイプ早期提供などの計画

見積りに反映すべき項目

本タスクで見積りに反映すべき項目は以下の通り。

- 企画工程以降での現行踏襲内容を検討するタスク
- 決定した現行踏襲内容の実現にかかるプロトタイプ検証などのタスク
- 策定した意思決定プロセスを遂行するためのタスク（「3.7 意思決定プロセスの策定（観点F)」の見積りに反映すべき項目と共通）

3.4　現行資産活用方針の検討（観点 C）

目的

限られたスケジュール・コストの中で、品質を担保しつつ効率的に再構築を実施するためには、既に品質が確保されている現行資産の活用を検討することが有効である場合がある。

ここでは、再構築手法によって活用可能な範囲が異なる点を踏まえた上で現行資産の活用方針を検討する。

現行資産の定義（活用範囲検討の対象）

ここでは現行資産の内訳として、以下の 2 点を定義する。
各々について、活用方針の検討を実施する。

- ドキュメント系資産
  現行システムの設計、運用情報に関連する資産
- 実装系資産
  プログラム、ソフトウェア製品の定義体、実データ　などの資産

ポイント！

- 現行資産の活用方針検討にあたっては、どの時点の資産を原本とするかを決定するため、「資産凍結タイミング」を決定しておく必要がある。
- 各資産のバージョンが一致していることを確認し、その後も継続的に資産間のバージョン管理に充分に留意すること。

タスク

下記(1)～(2)に「現行資産活用方針の検討（観点 C)」の観点におけるタスクとその該当工程を示す。

(1) 活用方針の検討（ドキュメント系資産）
(2) 活用方針の検討（実装系資産）

企画	要件定義	設計	製造	試験	リリース後
(1)	(2)				

タスクの内容

(1) 活用方針の検討（ドキュメント系資産）

現行システムのドキュメント資産について、新システムの開発に活用可能な範囲を決定する。

ソースコードと同期しており、記載内容が充分な現行ドキュメントが存在する場合、可能な限り活用し、設計作業のボリュームを削減する。

上記のような現行ドキュメントが存在しない場合、まずは現行ドキュメントの整備が必要である。

（「3.5 現行業務知識不足への対応（観点 D)」参照）

① 活用の定義

以下を「活用」と定義する。

- 仕様部分の記述はそのままに、ヘッダ部分などの修正のみで新システムのドキュメントとして現行ドキュメント資産を流用すること
- 若干の非互換による修正は予想されるが、作業ボリューム削減のため、現行ドキュメント資産をベースとして流用すること

② 検討対象の資産

プロジェクト状況により対象は異なるが、図 3.6 に示す。

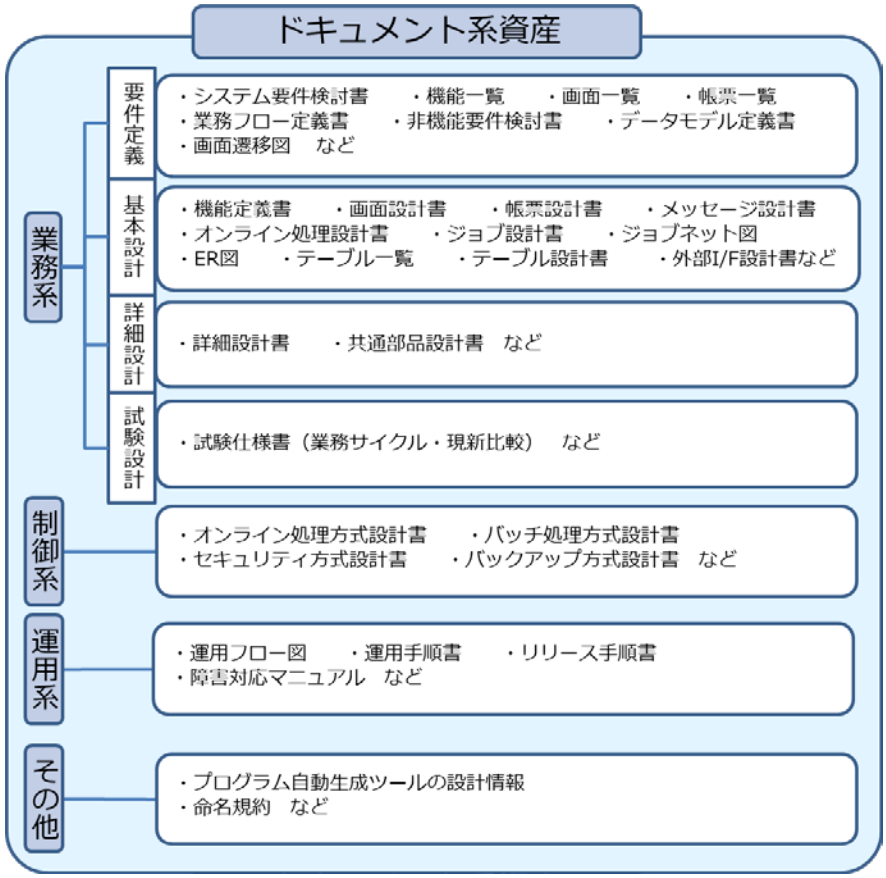


図 3.6 ドキュメント系資産の対象例

③ 再構築手法ごとの活用方針

再構築手法とプロジェクトによりドキュメント活用範囲は異なるが、図 3.7 に示す。

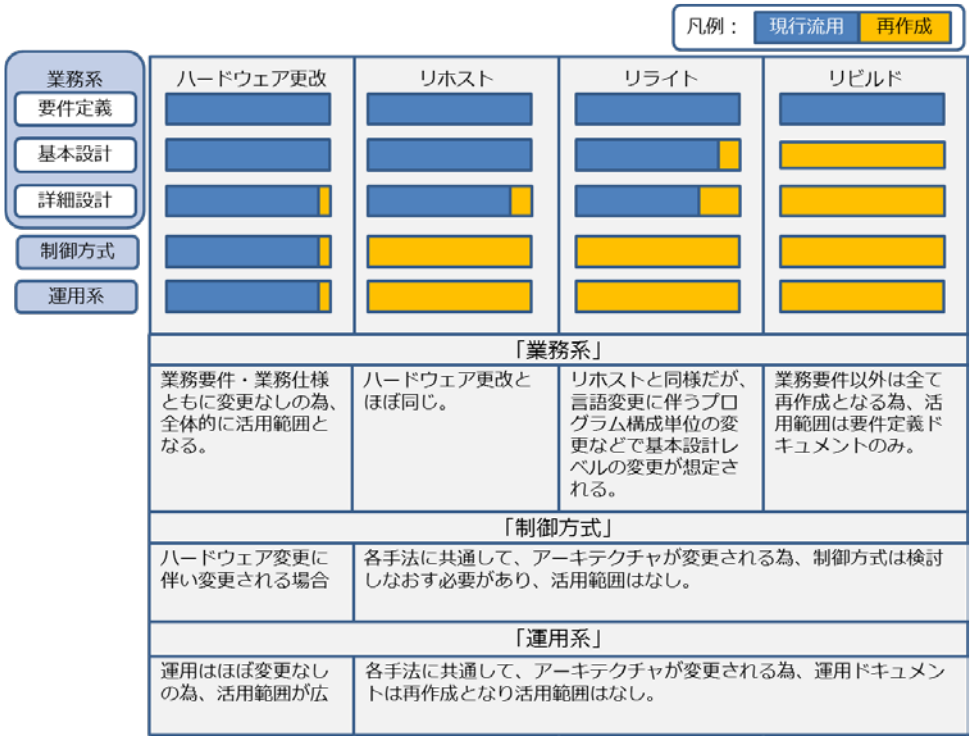


図 3.7 ドキュメント系資産の活用範囲の例

(2) 活用方針の検討（実装系資産）

各プログラム資産の種類（画面定義体・データなど）ごとに、どこまでを活用し、どこを開発対象とするのかを検討して作業ボリュームを把握し、システム化計画に反映する。

① 活用の定義

以下を「活用」と定義する。

- そのまま新システムで流用すること
- ツール変換や手修正の基となり、新システムで利用されること

② 検討対象の資産

プロジェクトにより対象は異なるが、図 3.8 に一例を示す。

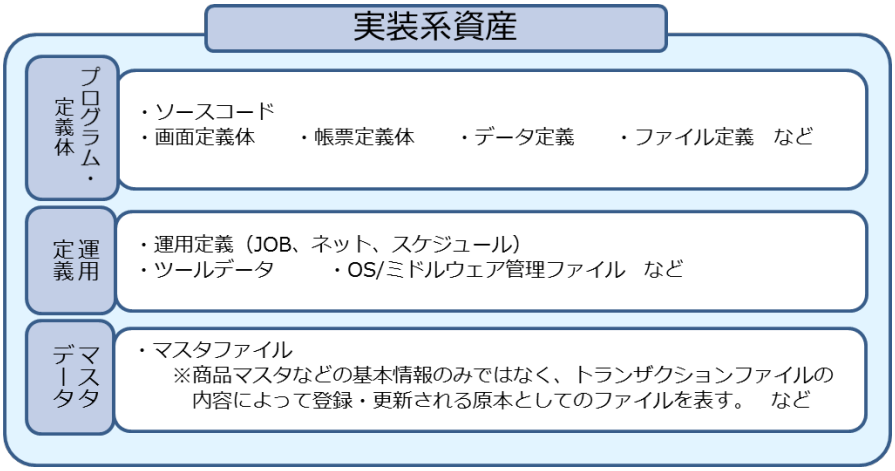


図 3.8 実装系資産の対象例

③ 再構築手法ごとの活用方針

活用方針は、再構築手法ごとに異なる。活用範囲検討の参考として、表 3.3 に手法ごとの方針の例を記載する。

※各手法ともに「マスタデータ」は移行対象となるので、活用対象となる

表 3.3 各再構築手法の方針の例

再構築手法	方針の例
ハードウェア更改	基本的にハードウェアのみの変更のため、活用範囲は資産全体とする。OS やミドルウェアのバージョンアップに伴う非互換部分について、開発対象とする必要あり。
リホスト	業務アプリケーションにはなるべく手を加えず、基盤を変更するため、プログラム・定義体は全体的に活用対象となる。運用定義は変更先の製品により活用範囲が異なる。メインフレーム→オープンの場合、アーキテクチャ変更による非互換をプログラムの業務アプリケーションではなく制御アプリケーションの作り込みで吸収する方針となるため、制御アプリケーションの活用範囲が低くなる傾向あり。
リライト	リホストとほぼ同様。ただし、言語変換により詳細仕様が変更となる場合、プログラムは開発対象となる。
リビルド	一からシステムを再作成する特性上、活用範囲はほぼ存在しない場合が多い。

※リホスト／リライトは、品質保証対策として、さらに不要／未稼働資産の排除検討の必要あり

ポイント！

不要・未稼働資産を資産から排除

リホスト・リライトの場合、対象資産から不要／未稼働資産を排除しておくことが望ましい。不要／未稼働資産が開発対象に残ったままとなると、開発工程の全てに渡って余分なコストが発生するだけではなく、試験工程では現行システムを稼働できず、正解のないテストを実施することとなる（リビルドは、要件定義に伴い不要部分は明らかにされるため、検討不要）。

タスクのアウトプット

本タスクの実施により、システム化計画向けに期待されるアウトプットは以下の通り。

- 再構築手法に応じた現行資産（ドキュメント系資産・実装系資産）の活用方針

見積りに反映すべき項目

本タスクで見積りに反映すべき項目は以下の通り。

- 現行資産（ドキュメント系資産・実装系資産）の活用方針に従って発生する追加タスク

3.5　現行業務知識不足への対応（観点 D）

目的

業務要件は変更せず、アプリケーションをそのまま移行するため、現行業務知識が不足していても問題ないとの判断で、現行業務知識不足のまま再構築プロジェクトを進めると、図3.9に示すような様々な問題が発生して問題プロジェクト化する事となる。

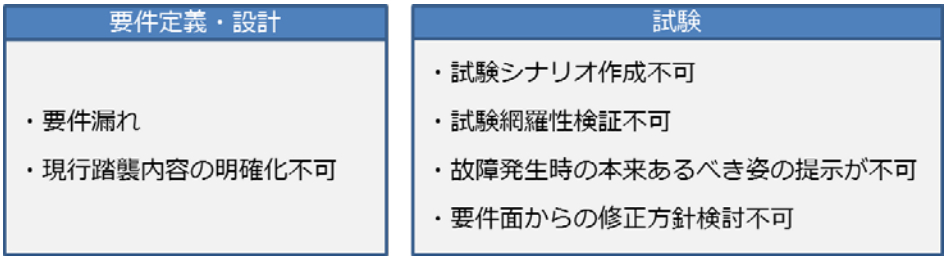


図 3.9 現行業務不足により引き起こされる問題例

再構築プロジェクトを成功させるために、現行業務知識不足への対応は決して避けて通れない。

よって、企画工程で「不足部分の整備計画」と「整備不可の部分については意思決定プロセスで要件判断を行う方針を決定」しておき、現行業務知識不足に備えることが重要となる。

(1) 現行業務知識とは

ここでの現行業務知識とは、ドキュメントと有識者のことを示す。

表 3.4 現行業務知識の対象

対象	概要
ドキュメント	現行システムの仕様(運用・業務の流れと各機能の概要)が確認できる資料のこと。
有識者	現行業務システムの運用・業務の流れなどに関する知識と経験を持ち、担当する一定範囲の要件定義・運用検証シナリオの作成など自律的にプロジェクト実行が可能である人材のこと。

(2) 再構築手法ごとに必要な現行業務知識

手法ごとに必要となる現行業務知識（ドキュメント・有識者）は異なる。一般的にハードウェア更改・リホスト・リライトはテスト（特にシステムテスト・運用検証）の計画やテスト項目を立てるための現行業務知識が必要となり、リビルドでは業務要件を明確にし、それに基づき再設計を行うための現行業務知識が求められる。ただし、プロジェクトの状況によっても異なるため、個々の状況に応じて必要となる現行業務知識を検討すること。

タスク

下記(1)～(2)に「現行業務知識不足への対応（観点 D)」の観点におけるタスクとその該当工程を示す。

(1) 不足部分の整備を計画

(2) 整備不可部分の判断は意思決定プロセスに従う

企画	要件定義	設計	製造	試験	リリース後
(1) (2)					

タスクの内容

再構築手法ごとに必要となる現行業務知識を対象に、不足部分を補うための対応を計画する。

(1) 不足部分の整備を計画

「2.2 現行システム調査・分析（ステップ 1)」の調査結果より、不足している範囲を確認し、ドキュメントと有識者を整備するための対策を計画する。

① ドキュメント不足への対応

- 処理設計書レベルでは、ソースコードからの再作成
- 基本設計書～要件定義レベルについては、有識者・利用部門からのヒアリングで再作成

ポイント！

ドキュメントは作成するだけではなく、その後に要員が内容を理解してプロジェクトを実行可能な状態となるように計画すること。

② 有識者不足への対応

不足している範囲に対し、カバーする有識者を運用部門や現行システム開発時のメンバなどの中からアサインする。

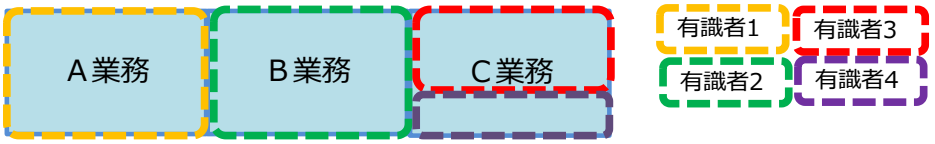


図 3.10 有識者のカバー範囲のイメージ

ポイント！

- **有識者の質は要確認**
  アサイン対象の有識者が、期待する範囲／レベルの現行業務知識を有しているか必ず確認すること。

- **有識者の担当範囲の規模に注意**
  各有識者が担当する範囲が大きくなり過ぎないように留意すること。担当範囲が適切な規模を超えると、作業過多となり有効に機能しなくなり、ボトルネックとなる可能性がある。有識者は適切な一定規模を担当範囲とし、それを基準に必要な人数を計画・準備すること。

- **プロジェクト終了後について検討することも重要**
  再構築プロジェクトの終了後、プロジェクトを通じて得た知識やノウハウが再び失われていくリスクを軽減するための対策を検討することも重要である。

(2) 整備不可部分の判断は意思決定プロセスに従う

(1) で整備できない現行業務知識の不足部分に対して、どのように開発を進めていくのかという「判断」が必要となる場合がある。それに備え、不明部分の判断は意思決定プロセス（「3.7 意思決定プロセスの策定（観点 F)」参照）で行うよう、企画工程で事前に定めておくことが重要である。

【判断の対象となる主な事例】

- 要件定義時に現行仕様が不明な場合
- 現新比較テストで出力結果が不一致となったが、現行要件が不明で正解がわからない場合

タスクのアウトプット

本タスクの実施により、システム化計画向けに期待されるアウトプットは以下の通り。

- 現行業務知識不足への対応計画

見積りに反映すべき項目

本タスクで見積りに反映すべき項目は以下の通り。

- 現行業務知識不足への対応について計画されている追加タスク

3.6　品質保証の検討（観点 E）

目的

再構築の品質保証は、新規開発に比べて容易とみなされがちであり、現行システムで動作しているアプリケーションを変換しているのだから設計の確認は省略してテストさえすればよいと考えたり、ともすればテストもほとんど必要ないと考えられたりすることがある。実際には、再構築でも品質を確保するためには、各工程で品質保証の施策を行って品質を積み上げる必要がある。

再構築手法によって、システム品質の保証上確認しなければならない観点が異なるため、その点について説明する。

さらに、再構築の品質保証では新規開発と異なり、「業務継続性の担保」が重要となる。しかし、どういう状態になれば業務が継続できると言えるのか、どういう確認をどこまで行えば業務継続性を担保できると考えるかは、業務の重要度やステークホルダによって異なる。また再構築プロジェクトによって求める品質保証レベルが異なる。

そのため本節では、「業務継続性の担保」を実現するための、再構築のプロセス全体を通じた品質保証の取組みを検討する観点についても説明する。

業務継続性とは

ここでの「業務継続性」とは、システムの可用性のことではなく、現行で行っていた業務や業務運用（以下、業務）を新システムでも継続して実施できる度合いのことを指しており、求められるのは、「業務プロセスが変わらない」ことである。具体的には、「操作性、帳票の出力される時間、レイアウト、性能、時間内に処理が終わる」といった目に見える要件（外部仕様）が担保されることである。

「業務継続性の担保」の実現にあたっては、問題となりやすい点が 2 点ある。

1 点目は、「業務継続性」を目に見える要件からシステム要件に落とし込む際、どういう状態になれば業務が継続できる状態であると言えるのかが、業務の重要度や、ステークホルダ（ユーザ企業やベンダ企業、ユーザ企業の中でもシステム部門や利用部門など）によって異なることである。

例えば、「日次処理 A が現行と同様に実施できること」が業務継続性の一条件である場合、これを噛み砕くと、以下のようになると考えられる。

- 日次処理 A の入出力が現行で動作した場合と同じであること
- 処理時間（時間帯）が所定の時間内に行われること

（しかし、以下のような詳細なレベルを期待する場合も考えられる）

- 日次処理 A についてエラーデータを入力した際のエラー処理が現行と同じであること

2 点目は、どういう確認方法でどこまで確認すれば業務継続性を担保できるかについても、定石がないことである。例えば、以下のようなものが考えられる。

- データバリエーションをどれだけ確認するか（何日、何年分のデータで確認するか）
- 処理ルートをどれだけ確認するか（正常系を確認すればよいか、異常系のケースを確認するか）
- 数ある業務処理のうち、全てを同レベルで確認するのか、重要処理に確認の比重を置くのか

イメージを図 3.11 に示す。

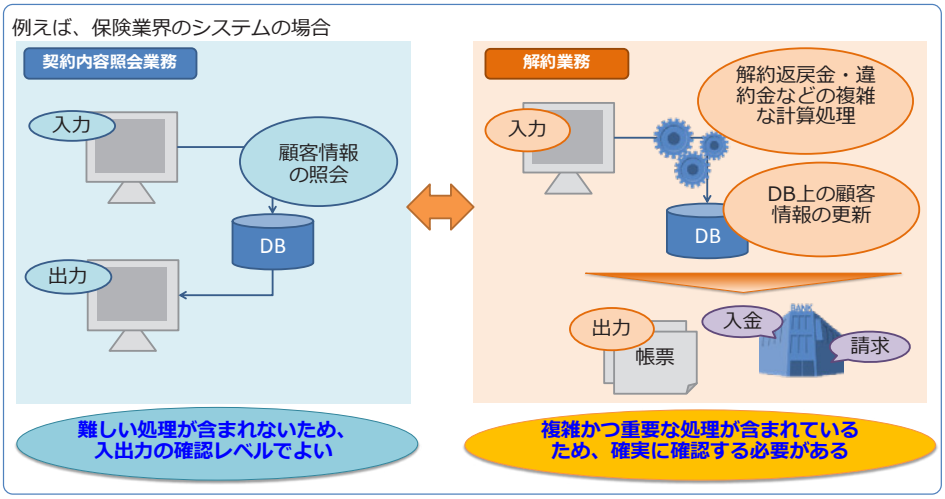


図 3.11 業務継続性の担保の取組みの違い

タスク

下記(1)～(2)に「品質保証の検討（観点 E)」の観点におけるタスクとその該当工程を示す。

(1) 品質確保の内容の検討

(2) 実施方針の検討

企画	要件定義	設計	製造	試験	リリース後
(1) (2)					

タスクの内容

(1) 品質確保の内容の検討

「(a)再構築手法ごとに特徴的な観点」と「(b)業務継続性担保の観点」それぞれについて内容を検討する。(図 3.12 参照)

なお、観点は(a)(b)の 2 種類あるが、観点ごとに実施方法（テストなど）や工程を分けなくてもよい。(後述の「(2) 実施方針の検討」を参照)

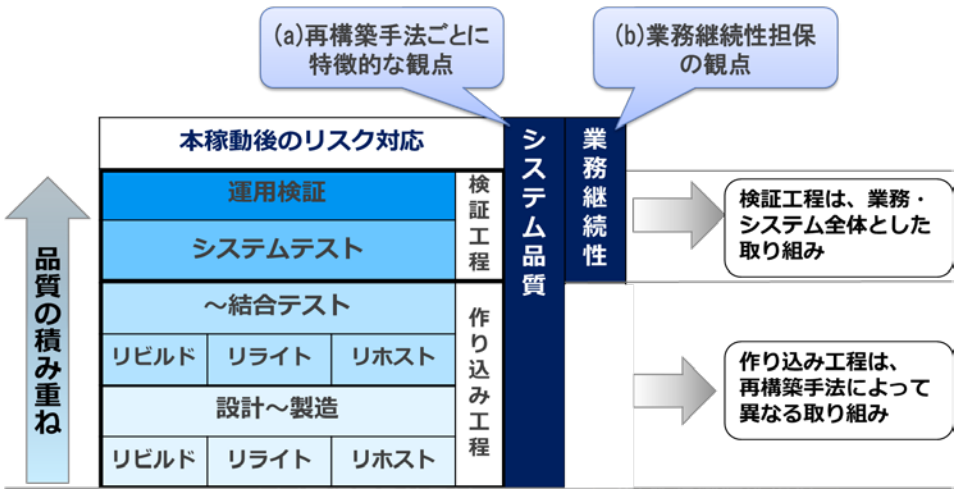


図 3.12 品質確保の内容の検討

(a) 再構築手法ごとに特徴的な観点

ハードウェア更改

現行システムと新システムでのハードウェアの違いを考慮し、変更内容を確認する。

（例）

- 図 3.13 のように OS やミドルウェア製品のバージョン変更によって製品の非互換が発生し、アプリケーションに変更が必要となる場合、アプリケーションの非互換修正内容が適切であることを確認する。

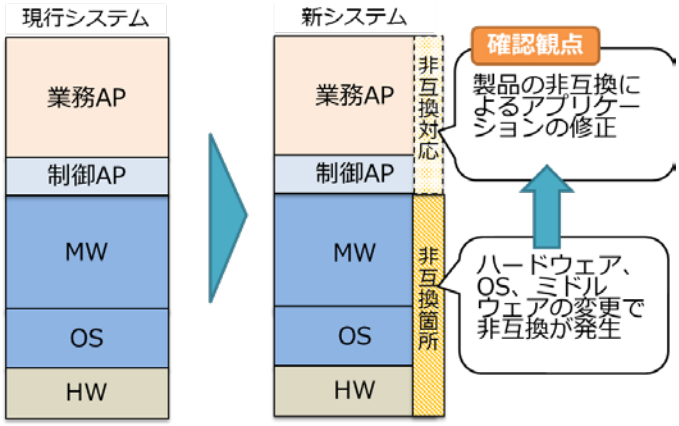


図 3.13 ハードウェア更改の確認観点の例

リホスト、リライト

OS やミドルウェア製品の変更に伴う対応を中心に確認する。

（例）（図 3.14 参照）

- 業務アプリケーションの変換設計が正しいこと
  （メインフレーム COBOL からオープン系 COBOL や、COBOL から Java にすることによる言語の文法の差異の変換　など）
- 基盤アーキテクチャの非互換の対応設計が正しいこと
  （丸め誤差の差異　など）
- OS やミドルウェア製品の機能差異を補完するための機能を作り込む場合、設計内容が正しいこと
  （メインフレームが提供していたユーティリティ など）

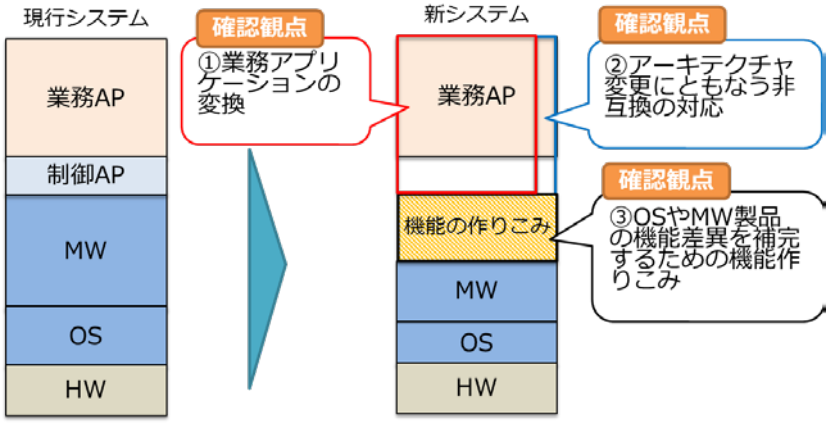


図 3.14 リホスト、リライトの確認観点

ポイント！

OS やミドルウェア製品の変更やアーキテクチャ変更に伴う非互換により、外部設計のレベルで変更が発生する場合、設計変更箇所の明確化と、変更内容のステークホルダ間での早期確認が必要。

（例）

画面の見え方が変更になる（ベンダ製品から Web 画面に変更）。

文字コードが EBCDIC から S-JIS になったことでソート順が変わり、帳票のデータの出力順序が変わる　など

リビルド

図 3.15 のように、現行踏襲する対象について、現行要件が漏れなく、ユーザが期待する内容で取り込まれていることを確認する。

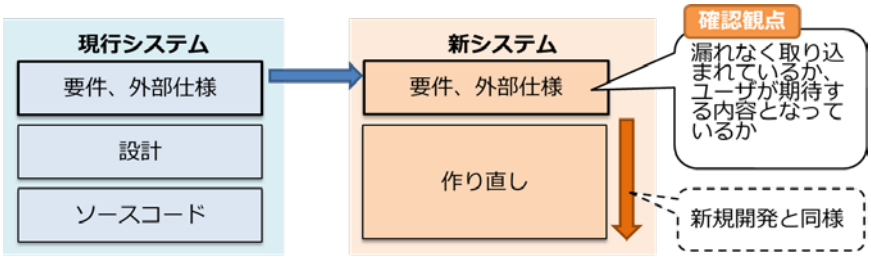


図 3.15 リビルドの確認観点

ポイント！

現行要件の取り込みについては、現行の設計書だけでなく、業務運用マニュアルやシステム運用マニュアルにも新システムの設計に反映すべき観点が存在する可能性があるため、注意すること。

(b) 業務継続性担保の観点

どういう状態になれば業務が継続できるか、および、どういう確認をどこまで行えば業務継続性を担保できるかを、業務の重要度、再構築プロジェクトに掛けることができる予算や期間を考慮して図 3.16 のように具体化する。具体化した内容は、再構築対象のシステムに関わるステークホルダの間で合意すること。

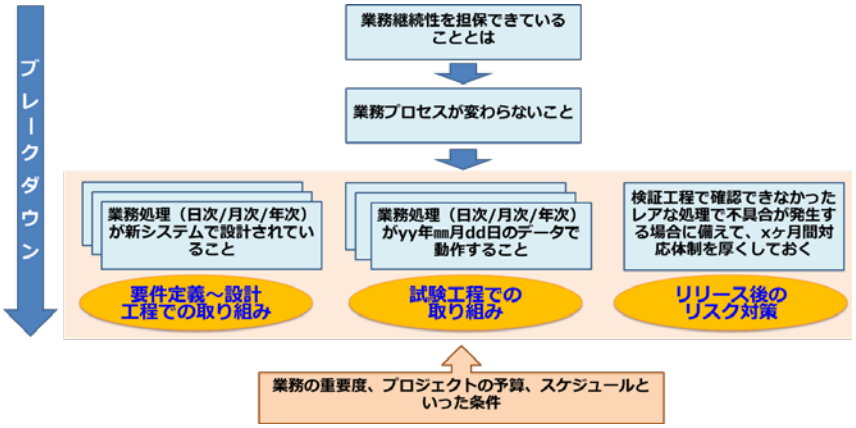


図 3.16 業務継続性担保の具体化のイメージ

図 3.17 および表 3.5 は業務継続性担保の対象と検討の観点を表したものである。IPA/SEC「機能要件の合意形成ガイド[10]」および「非機能要求グレード[11]」を参考に、モダナイゼーション WG に参画したユーザ企業の実務経験をもとに整理したものである。実際に利用する際はユーザ企業の事業や業務の特性に合わせ、改めて業務継続性担保に必要な観点を検討すること。

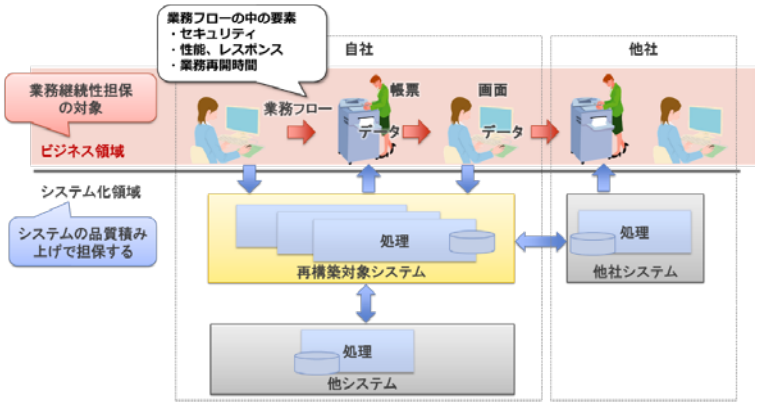


図 3.17 顧客のビジネスにおける業務継続性担保の観点

表 3.5 業務継続性担保を検討する際の観点（例）

観点	説明
業務フロー	・ 随時／日次／月次／年次で行う業務運用（正常時運用、および異常時運用を含む。システム化している業務運用、およびシステム化されていない業務運用を含む。） ・ 端境期で行う業務運用 ・ 再構築対象のシステムのアウトプットのうち、他（社）システムが業務運用を行うために必要なシステム間連携 I/F
画面	・ 画面レイアウト（罫線、入出力項目、項目内容、位置、配色など） ・ 画面遷移 ・ 画面アクション（ポップアップ／ロールオーバーなど） ・ 文字フォント、サイズ ・ 文字コード（外字の扱いなど） ・ 表示内容（入力内容を受け計算処理された結果など）
帳票	・ 帳票レイアウト（罫線、項目、項目内容、位置、配色など） ・ 用紙サイズ ・ 印字フォント、サイズ ・ 文字コード（外字の扱いなど） ・ 印字内容（計算処理された結果など） ・ 印字内容の並び順（明細など）
保有データ	・ システム切替え前後の静止ポイントで、顧客データや商品データなどの業務に必要なデータが移行できていること
セキュリティ	・ システムへのアクセスルート（ログイン後の画面に直接アクセスが禁止されているなど） ・ 不正アクセスに対する対応動作 ・ 利用者権限の適切な設定 ・ 既知のセキュリティ攻撃や脆弱性に対する対処
性能、レスポンス	・ オンライン処理性能（画面レスポンスタイムなど） ・ バッチ処理性能（処理時間など） ※オンライン、バッチ処理単体の時間よりも、処理の結果を受け、ユーザが次の業務を行うまでの時間に注目する
業務再開時間	・ システムメンテナンスからの業務再開時間 ・ 障害、災害発生によるシステム停止からの業務再開時間

観点に優先度をつける例を以下に示す。

＜確認する範囲に優先度をつける例＞

（例 1）ユーザ（顧客／社内業務部門）によって優先度をつける場合

- 顧客が利用する、あるいは顧客に影響する業務運用については全業務ルートを確認するが、社内に閉じた業務運用の確認は正常運用のみ確認する。
- 対顧客帳票はレイアウト詳細まで現行と新規で一致することを求めるが、社内帳票については印字内容が正しいことのみ確認する。

（例 2）取り扱う情報の種別によって優先度をつける場合

- 業務運用への影響が少ない統計情報を扱う機能は、代表パターンの確認のみ行う。

＜確認の実施時期に優先度をつける例＞

（例 3）機会によって優先度をつける場合

- 数年に一度しか実施されない業務運用については、確認時期を後ろ倒しにする。
- 災害時のみしか実施されない業務運用については、確認時期を後ろ倒しにする。

(2) 実施方針の検討

(1) について各工程でどのような取組みを実施するかの方針を検討する。要件定義～設計工程、製造工程、試験工程の各工程と、場合によってはリリース後のリスク対策を含めた再構築のプロセス全体で取組みを検討すること。

なお、要件や設計漏れが後工程で判明すると手戻りが大きく、プロジェクトのコスト、スケジュールに与える影響が大きいことから、要件定義～設計工程での品質確保は重要であるが、どの程度行うかはプロジェクトの条件によるため個々に判断が必要である。

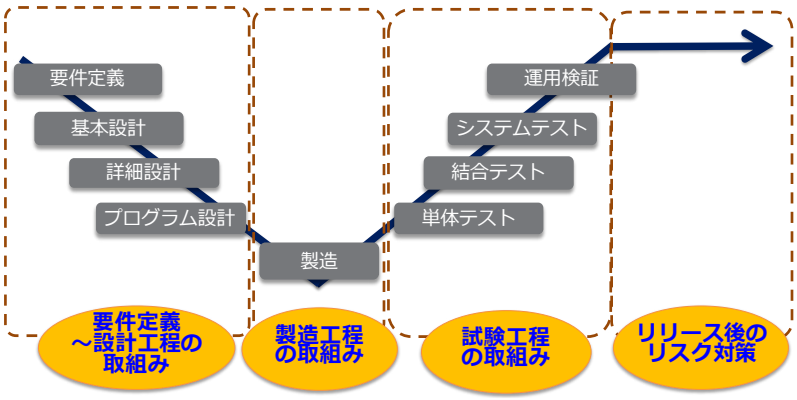


図 3.18 プロセス全体での品質保証

① 要件定義～設計工程での取組み方針を検討する

要件定義～設計工程では、「3.3 現行踏襲内容の明確化（観点 B）」で述べたように、現行の要件について現行踏襲する・しない、および踏襲する場合はその具体的な内容、踏襲しない場合は新システムでの新要件を明確化し、それに基づいて設計をしていく。

よって、本工程で再構築として特徴的な品質保証の取組みは、現行踏襲内容を漏れなく、正しく検討し、それを設計に取り込むこと、および取り込まれていることを確認すること、となる。「正しさ」は何を正解とするかをプロジェクト単位に定め、その基準をもとに判断する必要がある。

「3.3 現行踏襲内容の明確化（観点 B）」で述べた現行踏襲内容の検討、および、「(1)品質確保の内容の検討」で述べた観点に対して確認を行う。

いずれの手法でも、大きく分けて以下の 2 点の手段で、(1)の観点での品質を確認する。

- 設計書のレビューによる机上確認

- パイロット検証による実機確認

  設計内容を元にサンプルプログラムを作成することなどにより挙動を確認する。その際、現行の環境での野動作結果と比較して、設計内容が期待される挙動であるかを確認すると、より効果的である。

② 製造工程での取組み方針を検討する

再構築に特有な取組み観点はないため、説明を省略する。

③ 試験工程での取組み方針を検討する

試験工程での確認手段であるテストについて、以下 2 点を検討する。

- テストの種類
- テストの完了基準

テストの種類

どのようなテストをどの程度実施するかの方針を検討する。再構築手法ごとに必要となるテストは異なるが、アプリケーションの観点で見ると、一般的に表 3.6、表 3.7 のようになる。

表 3.6 再構築手法別のテストの種類（アプリケーション観点）

再構築手法	テストの種類				
	非互換パターンテスト（アーキテクチャ変更）	非互換パターンテスト（言語変換）	現新比較テスト	設計に基づくテスト（新規開発と同様）	サイクルテスト
ハードウェア更改	○	－	－	－	○
リホスト	○	－	○	－	○
リライト	○	○	○	△	○
リビルド	－	－	△	○	○

＜凡例＞○：必須、△：必要に応じて実施、－：基本的になし

表 3.7 再構築手法別のテストの特徴

再構築手法	アプリケーション観点でのテストの特徴
ハードウェア更改	・ 基本的にアプリケーションの変更は発生しないことから、新ハードウェアでの動作確認のためにサイクルテストを実施する。 ・ OS やミドルウェア製品のバージョン変更による製品非互換への対応でアプリケーションの修正が発生する場合、非互換パターンテストが必要となる。
リホスト	・ OS やミドルウェア製品の変更やアーキテクチャ変更に伴う非互換によるアプリケーションの修正に対して、非互換パターンテスト、現新比較テストを行う。 ・ 業務要件が満たされているかを確認するため、サイクルテストを行う。 ・ OS やミドルウェア製品の機能差異を補完するための制御アプリケーションの作り込み部分に対しては、制御アプリケーション自体のホワイトボックステストを実施した後、業務アプリケーションと結合し、業務としての動作を現新比較テストやサイクルテストの中で確認する。
リライト	・ リホストとほぼ同様のテストに加え、言語変換に伴う非互換も発生するため、その確認の非互換パターンテストが追加となる。 ・ また、言語変更により新規に作成したアプリケーションについては、設計に基づくホワイトボックステストも必要となる。(表 3.6 の「△」)
リビルド	・ 業務アプリケーション、制御アプリケーションともに、業務要件に基づき再設計するため、新規開発と同様に、設計に基づくホワイトボックステストで品質を積み上げる必要がある。 ・ それに加え、業務要件が満たされているかを確認するため、サイクルテストや必要に応じて現新比較テスト※を実施する。 ※リビルドは設計に基づくテストを行うが、実際には現行要件が再現できているかを確認するために、設計に基づくテストの一部やサイクルテスト結果について、現行との比較を行うことがある。(表 3.6 の「△」)

各テストの実施内容は表 3.8 の通り。

表 3.8 各テストの実施内容 (アプリケーション観点) (1/2)

テストの種類	実施内容概要
非互換パターンテスト	アーキテクチャ変更や言語変換に伴う非互換パターンごとに、非互換対応が期待通りに実現しているかを確認する。 ※アプリケーションの変更は、変換ツールや開発者の手修正により実施されるが、それらの変換の妥当性が担保されていることが前提となる。

表 3.8 各テストの実施内容（アプリケーション観点）(2/2)

テストの種類	実施内容概要
現新比較テスト	同一インプットで、現行システムと新システムのアウトプットが一致、または不一致かを確認する。 ただし、以下の留意点が存在するため、テスト範囲・テスト内容・検収条件を明確にすること。 ・ 現新比較テストは設計仕様に基づいた全ルートテストではないため、網羅性を保証することはできない。 ・ 新規開発の品質評価基準は使えない。
サイクルテスト	業務全体を通して要件通りに動作するかを確認する。 リビルド以外の手法で非互換部分のみを修正している場合であっても、それにより業務処理全体に影響が発生していないかを確認するために、本テストの実施は必要。
設計に基づくテスト	新規開発と同様に、設計書を正とし設計書の記載内容が実現されているかを確認する。

ポイント！

システム方式、運用方式の確認も忘れずに

上記のテストはアプリケーションの視点で示しているが、システム方式、運用方式を含む非機能面は新規開発と同様のテストにより品質確認する必要がある。
また、作り込み部分単体の品質確認だけではなく、業務処理として現行要件を満たしているかを確認する必要があり、システムテストや運用検証の確認観点とすること。例えば、ファイル転送の機能自体の確認は単体テストや結合テストで実施するが、現行システムと同じタイミングでファイル転送されるかについては、リホスト／リライトでは現新比較テストで、リビルドではシステムテスト以降での確認となる。

テストの完了基準

検討したテストの完了基準を明確にする必要がある。完了基準が曖昧なままであると、完了の判断をすることができず、テストを際限なく実施することとなり、いつまでも完了できない事態に陥る場合がある。特に現新比較テストでそのような事態になることが多い。そのため、ユーザは業務継続性を担保できる要件を定量的な基準として定めた上で、基準を満たせばテスト完了となるように、テスト実施前に完了基準を明確に決めて各ステークホルダと合意しておく必要がある。

④ リリース後のリスク対策

設定したテストでは確認していない処理や、特定のデータバリエーションでのみ発生する潜在バグが、稼働開始後に顕在化する可能性は存在する。そのような場合に備え、表 3.9 のようにあらかじめリスク対策を検討しておくことが望ましい。なお、リスク対策については、意思決定プロセスにて各ステークホルダの合意を得ておく必要がある。

表 3.9 リスク対策の例

対策	概要	注意点
並行稼働	一定期間、現行システムと新システムの両方で同一オペレーションを実施し、故障を洗い出す。	・ 利用部門の負担が増加するため、利用部門がどれほど並行稼働を許容可能か要確認。 ・ 現行システムの EOL、EOS の期限内で実施可能か要確認。
故障対応体制／マニュアル整備	故障が発生した場合に迅速に対応できる体制を事前に整備しておく。	・ 試験工程に新システム運用担当を組み込んでおく、または開発要員を運用担当として残すよう計画するなど、プロジェクト実行中の準備が必要。
段階的リリース	単一システムを複数の拠点にリリースする場合に、全拠点一括ではなくいずれかの拠点に先行リリースし、問題なく稼動することを確認した後に残りの拠点もリリースすることで、リリース直後のトラブルの影響範囲を限定的にする。	・ 段階的リリースの目的を明確にし、目的に応じたリリース時の確認や先行リリース対象を選択することが必要。 （例） ・ 拠点の規模のバリエーションを確認したい場合、大中小それぞれの拠点を対象にする。 ・ 特定の業務に注意が必要な場合、当該業務を実施する拠点を対象にする。

タスクのアウトプット

本タスクの実施により、システム化計画向けに期待されるアウトプットは以下の通り。

- 各工程の品質保証の取組み方針

見積りに反映すべき項目

本タスクで見積りに反映すべき項目は以下の通り。

- 各工程での品質保証の取組み

3.7　意思決定プロセスの策定（観点 F）

目的

再構築手法選択編にてリスクを確認し、計画策定編でリスク対応を計画するが、リスクを受容するにあたり方針の決定が必要な場合（図 3.19 の「リスク対策の計画段階の課題」）や、計画したリスク対策をプロジェクトで実施した際に対応結果が想定外となり課題化する場合（図 3.19「リスク対策の実行段階の課題」）がある。そのような事態に備え、意思決定を行うプロセスを策定しておく必要がある。

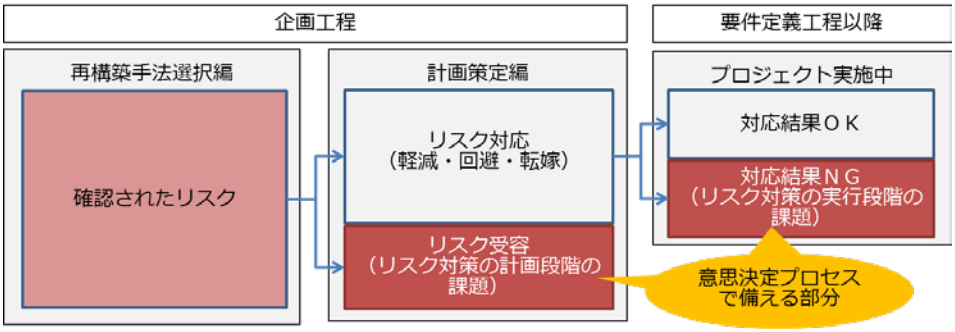


図 3.19 意思決定プロセス対象のイメージ

意思決定プロセスが存在しないと、例えば現行踏襲内容が不明な部分が判明し、ユーザ企業・ベンダ企業ともに現行仕様を把握していない場合などに、いつまでも方針が確定せず、プロジェクトの停止・遅延を招くこととなる。よって、意思決定プロセスを企画工程の時点から策定しておくことが重要となる。

タスク

下記(1)に「意思決定プロセスの策定（観点 F）」の観点におけるタスクとその該当工程を示す。

(1) 意思決定プロセスの策定

企画	要件定義	設計	製造	試験	リリース後
(1)					

タスクの内容

(1) 意思決定プロセスの策定

① 意思決定対象となる再構築特有の課題

再構築プロジェクト特有の、意思決定が必要となるような課題（表 3.10 の「リスク対策の計画段階の課題」および「リスク対策の実行段階の課題」）が存在する。主にプロジェクトの品質・コスト・納期に影響するものである。表 3.10 に例を示す。

表 3.10 意思決定対象の課題例

No	分類	課題例
1	リスク対策の計画段階の課題	現行仕様の復元が困難な部分に対しての方針決定
2	リスク対策の実行段階の課題	当初選択した再構築手法の見直し
3		計画時はテストに必要なデータ(比較元)をユーザ企業側で準備する想定だったが、実際には準備不可と判明
4		計画時はテスト実施時期に接続先システムのスケジュール確保が可能である見込みであったが、実際には確保困難であることが判明

② 意思決定プロセスの体制

意思決定対象となる課題の質によって、適切な体制の意思決定プロセスを策定する必要がある。例えば以下「③その他留意点」のように、プロジェクトの納期・コスト・品質に関わるような重要課題に対してはユーザ企業・ベンダ企業双方の、プロジェクトに関わるマネージャクラス・役員を中心とした体制を策定し、ユーザ企業側を主体として意思決定を行うものとする。

③ その他留意点

意思決定プロセスの策定にあたって、以下の点に留意すること。

- 決定内容の証跡を残す

  意思決定プロセスにおける決定内容は、ユーザ企業とベンダ企業の合意内容であり非常に重要であるため、必ず証跡を残すこと。

- 意思決定プロセスの設置ができない場合

  意思決定プロセスを事前に設定する調整がステークホルダの間で難航するケースも考えられる。その場合、意思決定プロセスがあらかじめ決められていないことによるリスクをステークホルダ間で合意し、必要な際には速やかに対応できるようにすることが望ましい。

タスクのアウトプット

本タスクの実施により、システム化計画向けに期待されるアウトプットは以下の通り。

- 意思決定プロセスについての策定結果をドキュメント化したもの。

（例）会議体体制図、意思決定対象基準

見積りに反映すべき項目

本タスクで見積りに反映すべき項目は以下の通り。

- 策定した意思決定プロセスを遂行するためのタスク

3.8　データ移行の計画（観点 G）

目的

データ移行は、アプリケーションや試験工程に影響を与えるため、計画の検討が遅いとプロジェクトの全体計画やコストにまで多大な影響を及ぼす可能性がある。検討すべき確認事項を明確にし、開発と並行して実施する計画を早期に立てることが重要である。早期に計画を立てなくてはならない理由は以下の通りである。

準備に時間がかかるため

データ移行は、移行対象やレイアウトなどユーザが事前に確認すべき観点が多く、データクレンジングも必要となるため、準備に手間や労力を要し、時間がかかる。また、作業項目が多いゆえに確認漏れによる問題が発生しやすい。

例えば、文字コードのマッピングを実施する際、レイアウトの確認を実施していないとレコードの中のカラムのレイアウトが変わっていることに気付かないまま実行し、データ移行に失敗する場合がある。これは確認漏れが原因であるが、こうした確認すべき観点は計画に盛り込むことが重要である。

また、想定外の問題も起こり得るため、問題が発生した場合の解消期間も計画に盛り込む必要がある。この作業に時間を要するため、早期に計画を立てないとプロジェクト完了の遅延やコスト増を招く。

移行したデータを試験工程で使用するため

テストでは、移行した実際の業務データを使用する場合があるため、データ移行が完了していないとテストの開始に影響が及ぶ。

テストの開始前にはリハーサルとしてクレンジングされたデータを検証環境に移行する作業や、場合によっては移行できなかった不整合データの修正、データ移行リハーサルの再実施などが発生する。

また、サイクルテストで新旧データの確認を実施する場合は、現行システムのデータを事前に用意しておく必要がある。例えば、1 年間のデータを日回しする場合、その分のデータを 1 年前から用意しておかなくてはならないため、早期の検討が必要となる。

タスク

下記(1)に「データ移行の計画（観点 G)」の観点におけるタスクとその該当工程を示す。

(1) 確認すべき観点の抽出および実施計画を早期に立てる

企画	要件定義	設計	製造	試験	リリース後
(1)					

タスクの内容

(1) 確認すべき観点の抽出および実施計画を早期に立てる

以下の表 3.11 に、データ移行で問題が起こりやすい代表的な観点を挙げる。これらを早期に計画し、確認することが重要である。

表 3.11 代表的な確認すべき観点

No	観点	確認事項
1	移行対象の整理	全部移行すると作業量が増え、時間がかかる。コスト増にもつながるので移行対象は絞った方がよい。 (例) ・ 移行元(現行)データとして送る範囲(稼動期間で 3 年以上前のデータ、トランザクションデータは捨てるなど)の方針を明確にする ・ 対象範囲のデータ量、件数を抑える
2	レイアウトの確認	移行対象のファイルや DB データのレイアウトを確認する。 ※必ず、ユーザ企業とベンダ企業双方でレイアウトを確認すること
3	データクレンジングの実施	想定外の値が含まれる恐れがあるため、データクレンジングを実施する。 ※ユーザ企業とベンダ企業双方で役割分担を明確にすること
4	文字コードのマッピング	外字や同じフォントフェイスが次期システムに存在しない場合、フォントフェイスと用語の意味、どちらを優先するか調整する。
5	移行方法の確認	一括移行／段階移行／差分移行などの移行方法を選定する。 ※データ転送方式も検討すること
6	移行実施結果の確認	チェックツールなどを用いてデータ移行の実施結果が正しいかを確認する。 ※実施結果の確認方法についてユーザ企業とベンダ企業双方で合意すること

ポイント！

- **データ移行で必要な確認観点の項目を早期に抽出すること**

  データ移行は多種多様な作業が発生するため、確認観点を洗い出しベンダ企業の協力が必要なところは合意をしておく必要がある。

- **問題が発生した際の解消期間も計画に盛り込むことが必要**

  データ移行による問題は開発の終盤で顕在化するため、解消期間も計画に盛り込むべきである。

- **試験工程までにデータ移行が完了していること**

  試験工程で現行踏襲を確認するためにはサンプルデータではなく、実際の業務データでテストを実施すべきである。そのため、試験工程に入る前に移行データが揃えられるように計画することが重要である。

タスクのアウトプット

本タスクの実施により、システム化計画向けに期待されるアウトプットは以下の通り。

- データ移行実施計画書

見積りに反映すべき項目

本タスクで見積りに反映すべき項目は以下の通り。

- データ移行実施計画書で計画されているタスク

3.9　再構築の計画と見積り（観点 H）

目的

再構築の見積りでは、計画策定編 A～G で述べた再構築で起こりうるリスクとその対策としてのタスクを盛り込むことが重要である。しかし、実際のプロジェクトの企画工程では不確定要素が多く、その結果として精度の低い見積りとなる点が課題である。「経済産業省～情報システム・モデル取引・契約書～」[6]で、多段階契約と再見積りの考え方を定義している通り、通常の新規開発と同様、再構築の見積りにおいても各工程で具体化された要件やリスクをもとに見積りを詳細化していくことを推奨する。

タスク

下記(1)～(2)に「再構築の計画と見積り （観点 H)」の観点におけるタスクとその該当工程を示す。

(1) 計画策定の観点 A～G を考慮したシステム化計画と見積り作成

(2) 段階的見積り

企画	要件定義	設計	製造	試験	リリース後
(1)	(2)				

タスクの内容

(1) 計画策定の観点 A～G を考慮したシステム化計画と見積り作成

再構築特有のリスク対策として必要な項目を見積りに反映すること。

（計画策定の観点 A～G の「見積りに反映すべき項目」を参照）

※見積りが想定より大幅に大きくなると、再構築手法選択編に戻って再構築手法の見直しが必要となる場合もある。その場合は要件定義工程までで判断すること

(2) 段階的見積り

企画工程で実施した概算見積りを工程の区切りなどで段階的に詳細化・精緻化し、見積りの精度を上げていくことが重要である。また、不確定要素を取り除く手段として事前検証の実施を推奨する。結果を見積りに反映することで、見積りの精度向上が期待できる。

タスクのアウトプット

本タスクのアウトプットは以下の通り。

- システム化計画
- 再構築の概算見積り

3.10　業務要件の変更／追加への対応

目的

現行踏襲をベースとした再構築と業務要件の変更／追加を同時に実施すると、変更／追加した要件の影響範囲の特定が難しく、作り込みの漏れや試験トラブルが発生した場合の問題の切り分けが困難になる。そのため、「現行踏襲」部分と業務要件の変更や追加した部分の整合性の確保が難しく、品質保証の難易度が非常に高くなる。

このため、本節では、本来は同時実施を回避すべきであるが、やむを得ず再構築と同時に業務要件の変更／追加を行う場合の注意点と計画段階で検討すべき観点を説明する。

業務要件の変更／追加とは

本節では、システム化されている業務に影響する要件のうち、顧客業務の内容・順序・タイミング・取扱い情報、業務の担当者（部門）・連携順序・業務実行タイミングに対して、ユーザの要求により、現行から変更または追加があるケースを対象とする。(図 3.20 参照)

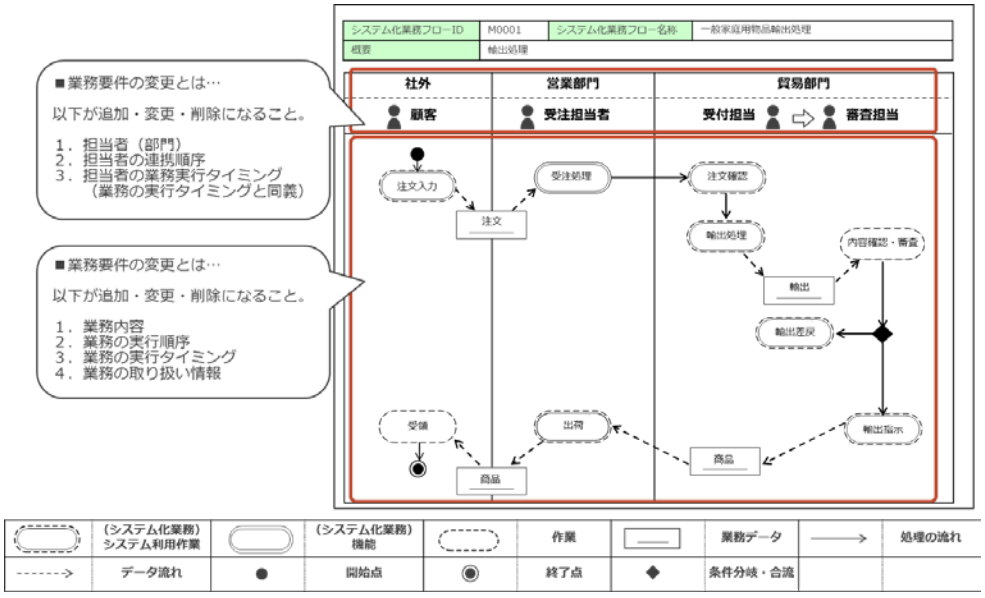


図 3.20　システム化業務フローで表現した業務要件の変更／追加

対象および対象外とするケースの例は以下の通り。

- 対象とするケース例

  ・既存の部門とは別に、品質管理部門を新設して生産から出荷までのフローに品質管理部門のチェックを行う業務を追加し、それをシステムで実現する。

  （図 3.21 は、図 3.20 に対して上記の追加を行った例である。）

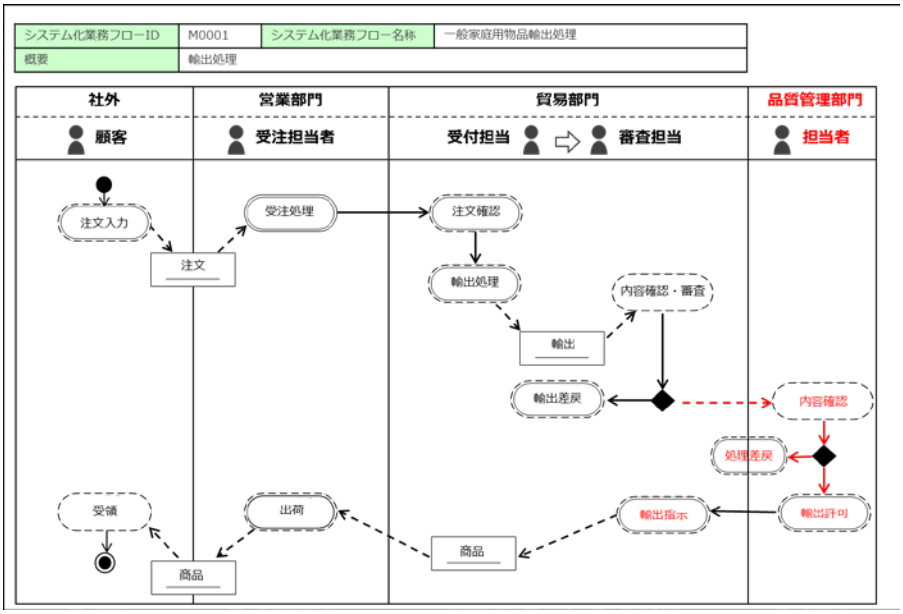


図 3.21　システム化業務フローで表現した業務要件の変更／追加（変更例）

- 対象外とするケース例

  ・現行システムに対する変更を新システムでも取り込む。

  再構築プロジェクト開始後に法令対応や業務改善などにより現行システムの変更があり、新システム側にも同様の変更を取り込む必要が生じる場合がある。この場合については、現行仕様の凍結タイミングの管理と、追いつき開発によって対応するテーマであるため、本節の対象外とする。

企画・計画工程で実施すべき内容

(1) 影響範囲の見極め

業務要件の変更／追加を実施するためには、変更／追加したい業務要件の仕様化・実装だけではなく、現行踏襲部分への影響範囲を把握し、プロジェクト期間を通じて管理ができることが必須となる。そのために、以下の事項を業務要件の変更／追加を実施する前提条件として確認する必要がある。

- 現行の業務要件や仕様を把握している、または把握ができる見込みがあること。(3.5節参照)
- 業務要件の変更／追加の影響範囲を特定し、現行踏襲部分との切り分けを行うことができること。
- 変更／追加する業務要件に関連するユーザが再構築プロジェクトに早期参画できること。

上記の確認結果をもとに、業務要件の変更／追加の実施方針を検討し、リスク対策を含めた実施計画を作成する。実施方針は以下の2通り。

①再構築と業務要件の変更／追加のフェーズを分けて実施する

②再構築と同時に業務要件の変更／追加を行う

(2) 実施方針の検討

① 再構築と業務要件の変更／追加のフェーズを分けて実施する

現行踏襲ベースの再構築を終えた後、新システム側に要件変更／追加を取り込むことが最も安全確実な方法である。現行踏襲ベースの再構築と業務要件の変更／追加のフェーズを分けて実施することが望ましい。(図3.22参照)

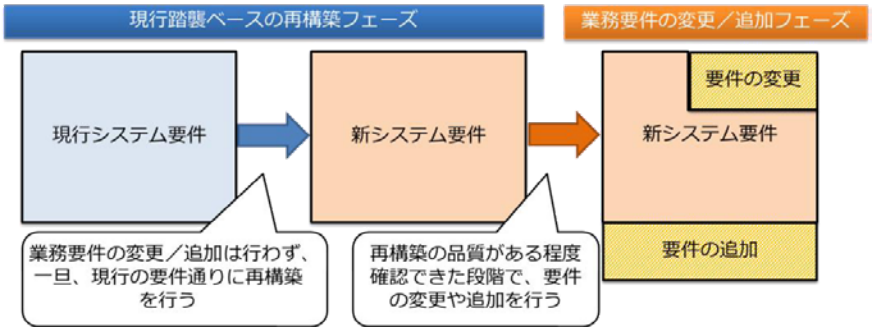


図3.22 再構築後に業務要件の変更／追加を行う流れ

この場合、要件の変更／追加の作業は、通常のシステム保守運用の中で追加開発を行う際と同様である。ただし、再構築の計画段階で、要件の変更／追加フェーズを見据えた計画を検討しておくことが必要である。

計画段階での検討ポイントは以下の通り。

(a) 要件の変更／追加フェーズの開始時期を検討する。

現行踏襲ベースの再構築の品質をある程度担保した工程後に要件の変更／追加フェーズを開始する必要がある。システムの特性や変更／追加する要件の複雑さなどから、要件の変更／追加フェーズを開始前に求められる再構築の品質や実現する工程について検討し、全体スケジュールを計画する。

（例）

- 現行踏襲ベースの再構築フェーズのシステムテスト完了後に、要件の変更／追加を開始する。

なお、例のように再構築フェーズの開発期間中に変更／追加フェーズを開始する場合、それ以降の再構築側のテストは、要件の変更／追加フェーズと並行して行うか、要件の変更／追加フェーズを経た後に行うかのどちらかになる。後者の場合、要件の変更／追加フェーズの最終段階で利用部門と認識齟齬が判明すると、大幅な手戻りになるため、利用部門との意識合わせは事前に実施することが求められる。（3.3　現行踏襲内容の明確化（観点 B）の「ポイント！・利用部門との認識齟齬を軽減」参照）

(b) 要件の変更／追加フェーズの試験方針を検討する。

- 変更／追加した要件に対する設計に基づく試験を行う。
- 想定している影響範囲外に対する確認を行う。
  （不必要なアプリケーションの修正などが行われていないか、要件の追加／変更に伴うシステムの変更の影響で関係がないと想定していた箇所の挙動が変わってしまっていないか、など）

② 再構築と同時に業務要件の変更／追加を行う

①の通り、現行踏襲ベースの再構築を終えた後に新システム側に要件変更／追加を取り込むことが最も安全確実な方法である。ただし、コスト・期間の制約から現行踏襲ベースの再構築と同時に行うケースもある。（図 3.23 参照）

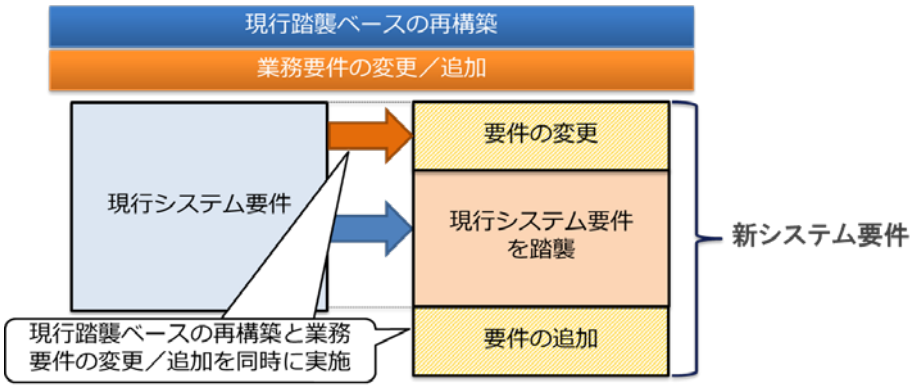


図 3.23 再構築と同時に業務要件の変更／追加を行う流れ

再構築と同時に業務要件の変更／追加を行う場合、以下のような観点に特に注意し、要件定義以降の実施計画や管理ルールの策定を行う。なお、再構築で通常求められるリスクと対策については、2 章および 3.1～3.9 を参照のこと。

(a) 現行踏襲部分と業務要件の変更／追加部分の品質管理の方法に注意する

- 品質評価・分析方針の策定

  現行踏襲ベースの再構築は、「3.6 品質保証の検討（観点 E)」で説明したような品質保証の考え方となるが、業務要件の変更／追加案件では新規開発と同様の品質保証の考え方となる。そのため、再構築と同時に業務要件の変更／追加を行う場合、両者の開発手法に応じ、品質の確認や評価・分析を検討する必要がある。

さらに試験では、再構築案件と業務要件の変更／追加案件が重複することにより、想定外の試験結果になってしまうことがある。(図 3.24 参照) そのため、試験結果の切り分け、評価方法を早期に策定するとともに、試験結果の切り分け、評価に必要な業務知識を有する要員を準備する必要がある。

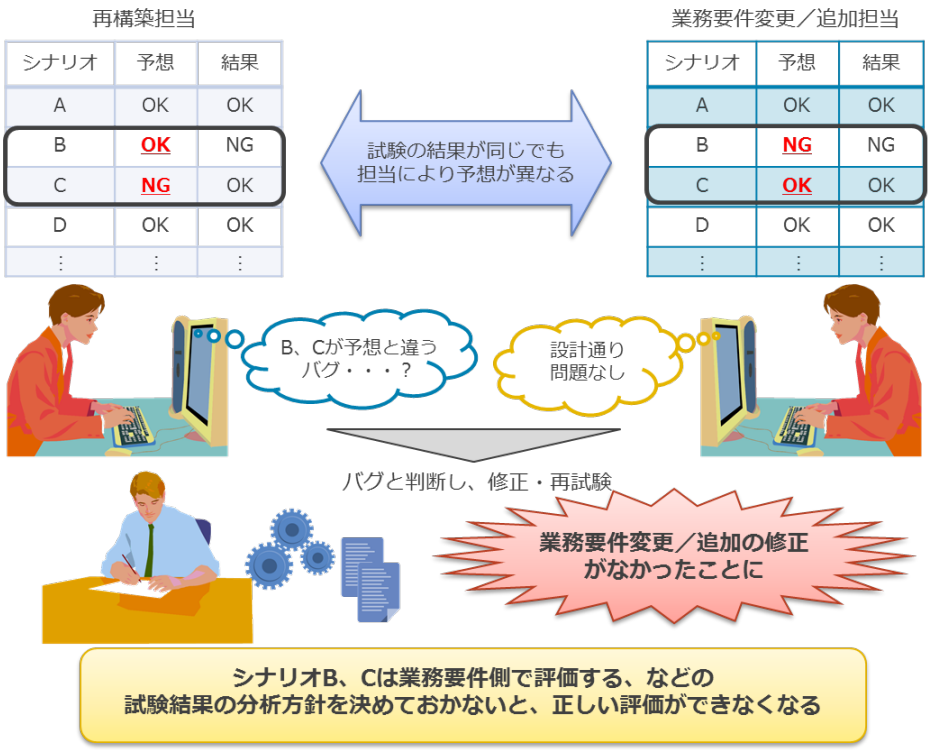


図 3.24　試験結果の分析方針の必要性

- 試験データの作成方針

  業務要件の変更／追加案件では元データ（現行のデータ）がそのまま使えず、データの新規作成・加工が必要になる可能性がある。データ設計の開発が再構築の場合より複雑化すると考えられるため、作成方針を策定しておく。

(b) 現行踏襲部分と業務要件の変更／追加部分の相互影響に注意する

業務要件の変更／追加案件の影響範囲が変わると、再構築部分のアプリケーションの変換や基盤設計等に影響を及ぼす可能性がある。そのため、各案件担当者はチーム間の連携を密に行う必要があり、対応ルールなどを企画・計画段階で検討しておくことが望ましい。

（例）

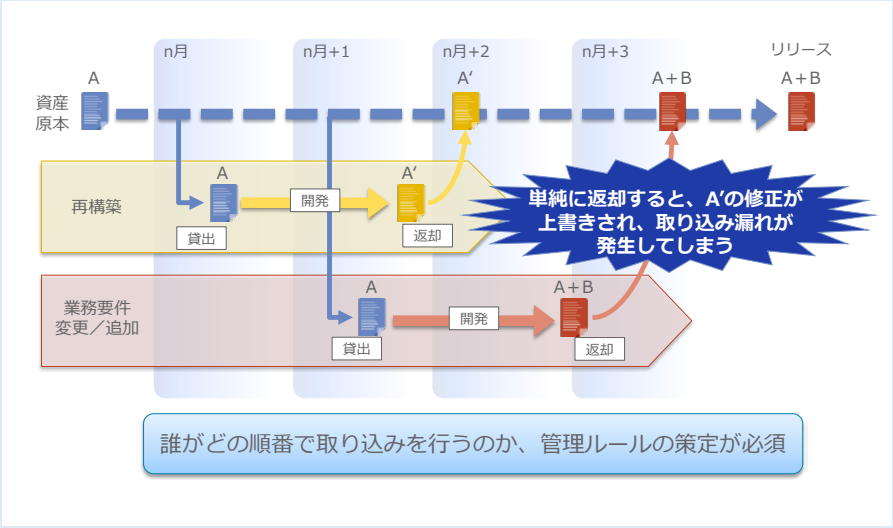
- リホスト・リライト手法でアプリケーションのコンバージョンを行う場合、変換箇所や変換設計が変わり、変換ツールの製造に影響を与える。

図 3.25 現行踏襲と業務要件の変更／追加担当者の連携

ポイント！

現行踏襲ベースの再構築と業務要件の変更／追加を同時に行う場合、再構築のみを行う場合に比べ仕様管理が複雑になるため、設計書・アプリケーション資産等の変更管理ルールの策定することが重要である。

同一資産に対し、再構築と業務要件の変更／追加にてそれぞれ変更を加える場合もあるため、取り込み漏れが発生しないよう、常に最新のリソースに修正が加えるようにリソースの配布管理を徹底するなど、適切に管理するルールの検討を計画・企画時に行う。



ケーススタディ

はじめに

3章の使い方の参考として、システム化の企画段階における例題に基づいたケーススタディを示す。

企画工程で特に重要なポイントになる「現行踏襲内容の明確化」、「品質保証の検討」について、再構築手法選択編と同じ例題を用いて、ケーススタディを示す。

本ケースを各企業の状況に置き換え、企画工程でのシステム化計画策定の参考にされたい。

【ケーススタディの概要】(再掲)

大規模な基幹システムを短期間で再構築する場合の手法選択のケーススタディで、最終的に以下の(1)～(3)の内容で開発が進められた。

(1) 「システム運用のコスト削減を目的として、基幹システムのメインフレーム保守契約満了を期に、現行システムの資産・運用・ハードウェアをオープン化」

～再構築の目的～

- 運用コストの削減
- オープン環境への開発要員のシフト
- 運用の継続性を重視した短期間で安全なオープン化
- BCP サイトの構築

(2) メインフレームから、オープン (UNIX) への移行

(3) オンラインはリビルド、バッチはリホストの2つの手法を選択

(1) 現行踏襲内容の明確化･･･本章「3.3 現行踏襲内容の明確化（観点 B)」より
① 企画工程での明確化
② 要件定義工程での明確化の計画
③ 追加コスト・時間がかかる部分への対応判断のためのプロセス

(2) 品質保証の検討･･･本章「3.6 品質保証の検討（観点 E)」より
① 業務継続性担保の具体化
上記を検証する際の品質目標（品質確保方針）の具体化
② 実施方針の検討
要件定義～設計工程での取り組み（リビルド、リホストそれぞれの検討例）
試験工程での取り組み（リビルド、リホストおよび全体での検討例）
リリース後のリスク対策

(3) システム化計画（まとめ）
全体工程計画

実施内容

(1) 現行踏襲内容の明確化
まず、企画工程では、ユーザ企業とベンダ企業の想定する「現行踏襲」の拠り所について、図 3.26 のイメージ図を参考にステークホルダ間でギャップがあることをお互いに認識し合う。

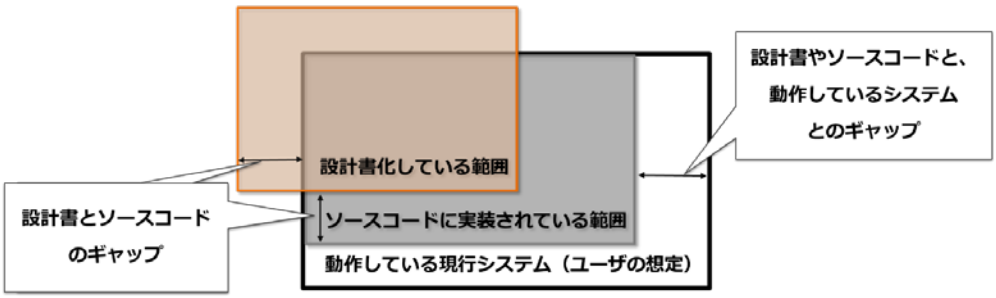


図 3.26「現行踏襲」の拠り所のイメージ図

本ケースでは、以下のような前提である。

- 現行システムの操作マニュアル、および設計ドキュメントは存在するが最新化されていない。(再構築手法選択編 ケーススタディ「(ステップ 1)現行システムの調査・分析のⅡ.業務知識」の調査結果より、図 3.26 の各ギャップあり)
- ユーザは、動作している現行システムと同じであることを想定している。ただし、なんでもかんでも現行踏襲ではなく、新しいプラットフォームや開発環境により現行と変わる場合や単純に移行できないものは、変わる内容を明確にして、必要な作り込みなどを行った。最終的な運用検証で、ユーザが運用を習熟・習得し、業務継続を確保することで、合意形成した。

① 企画工程での明確化

再構築手法決定段階では現行システムの調査・分析、新システムへの要求およびリスク対応方針の結果が存在する。さらに、図 3.26 「現行踏襲」の拠り所のイメージ図に記載されたギャップについては、ステークホルダ間で認識合わせを行う。それぞれの結果を踏まえて、企画工程での再構築方針を定める。本ケースでの方針を以下に示す。

(a) バッチ業務は、動作している現行プログラムのソースコードを活用、流用して現行仕様の再現性を高める再構築（リホスト）を行う。

(b) オンライン業務は、現行ドキュメントは活用、流用せず、動作している現行画面等を調査し、業務担当へ仕様のヒアリングを行い、設計書を作成した後にオープン開発環境にてアプリケーションを再構築（リビルド）する。これにより、オープン環境へのスキルシフトを図る。

(c) プラットフォームの変更や単純に移行できないもの（例：ISV 製品の変更によって影響を受ける運用や非機能要件など）は、次の要件定義～設計段階で変わる内容を明確にし、運用・基盤構築に必要な作り込みを行い、運用検証をすることで、ギャップを埋める。

本ケースにおける企画工程の明確化イメージを図 3.27 に示す。

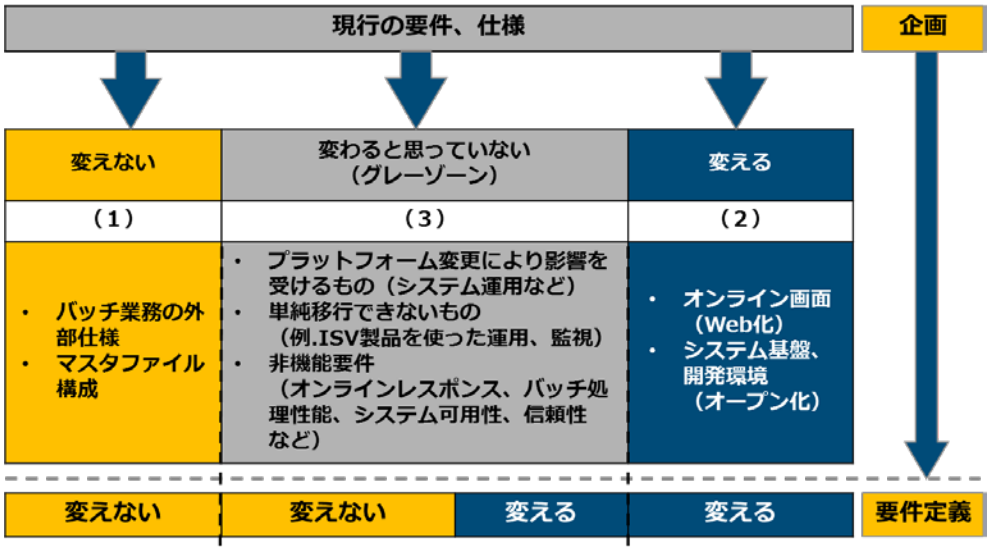


図 3.27 企画工程での明確化イメージ

② 要件定義工程での明確化の計画

再構築手法によって現行踏襲内容を明確化していくプロセスは異なるため、再構築手法ごとに、要件定義工程での明確化の計画例を示す。

(a) リホスト

本ケースでは、バッチ業務の再構築手法としてリホストを選択した。この場合、要件定義工程では、現行資産から移行先であるオープン環境との非互換調査を行う。同時にプログラム資産（定義体を含む）やマスタデータの移行方法、および ISV 製品などによるシステム固有な基盤機能の代替方法などを明確化する。そのプロセスを図 3.28 に示す。

プロセス	役割分担		実施内容	入力情報	出力情報
	ユーザ	ベンダ			
全移行対象資産提供	○		・現行資産の提供 ⇒各種プログラム資産 ⇒JCL ⇒画面定義体 ⇒帳票定義体 ⇒オーバレイ ⇒ADL	・稼動資産一覧	・現行資産
非互換調査		○	・処理方式分析(画面、帳票) ・言語非互換分析 ・ISV製品分析	・現行資産	・非互換一覧
移行対象資産調査		○	・移行対象資産の過不足調査	・現行資産	・資産一覧 ・不足資産一覧
不足資産の提供	○		・不足資産の提供	・不足資産一覧	・現行資産
移行対象資産の確定	○	○	・リホスト対象資産の確定 ・リホスト非対象資産の確定 ⇒対応方針確定	・資産一覧	・移行対象資産一覧
移行方法検討	○	○	・非互換部分の移行方法検討 ・資産内容、変換方式に関するQ/A対応、ISV製品の情報提供	・Q/A票 ・打合資料(移行方法案など)	・移行方法検討結果 ・Q/A票
移行方法設計		○	・Shell方式設計 ・COBOL方式設計 ・非互換対応方式設計	・分析結果一覧	・移行方法設計書
レビューおよび承認 判定 ドキュメント修正	○	○	・実行時非互換の事前説明 ・レビュー及び承認	・移行方針仕様書 ・移行方法設計書	・レビュー記録
バッチの移行方法確定					・移行方針仕様書 ・移行方法設計書

図 3.28 リホストの要件定義工程の計画例

(b) リビルド

本ケースでは、オンライン業務の再構築手法としてリビルドを選択した。ここでは、①企画工程での再構築方針から、現行画面の調査や業務担当への仕様のヒアリングまで、新規開発と同様のドキュメントを作成し、アプリケーションを再作成（リビルド）する計画とした。そのプロセスを図 3.29 に示す。

また、オープン環境を習得するという目的もあったため、設計工程に入る前の要件定義工程で、基礎研修や開発演習を実施する計画としている。

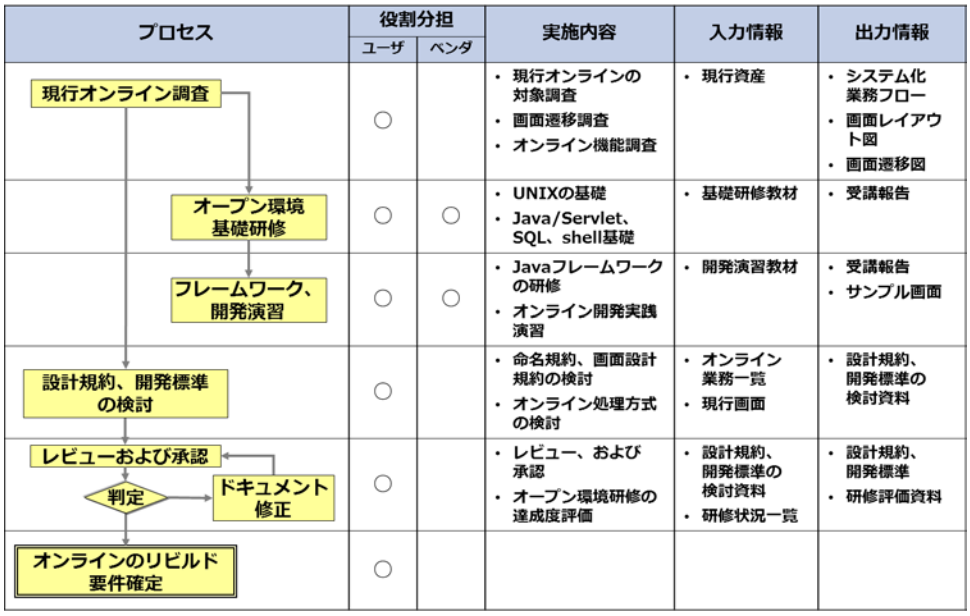


図 3.29 リビルドの要件定義段階の計画例

(c) 運用・基盤

本ケースでは、プラットフォームの変更（オープン化）および ISV 製品の変更により、現行から影響を受けるシステム運用やシステム基盤の計画も必要となる。以下の図 3.30、図 3.31 には、新システムとしての要件定義に関して、そのプロセスと役割分担、成果物で明確化する計画を定めている。

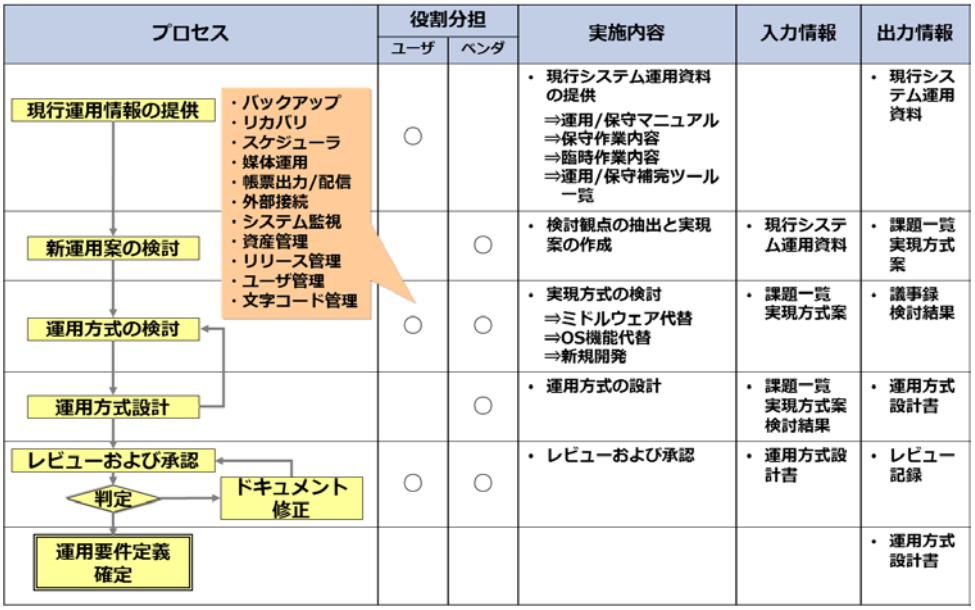


図 3.30 運用の要件定義工程における計画例

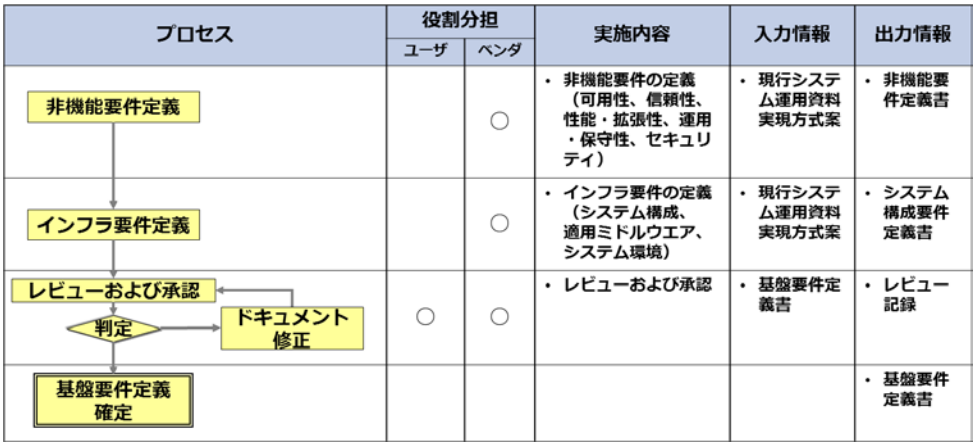


図 3.31 基盤の要件定義工程における計画例

③ 追加コスト・時間がかかる部分への対応判断のためのプロセス

リスク対応方針などから、追加コスト・時間が必要になり対応判断が求められるリスクを想定し、QCD の観点で意思決定プロセスを計画する。

本ケースで意思決定をはかった内容

(a) 安定して運用できるようにするため本稼動を延伸するか否か、本稼動時期の判断。
【D:デリバリ】

(b) 運用機能についての開発費用は、工程毎に調整する（要件定義以降の増減）。
【C:コスト】

(c) 追加コストやスケジュールに影響する、次工程に進むか否かの判断。

なお、本ケースでは、週 1 回の各グループ会で検討し、隔週のプロジェクト会議で調整のうえ、月 1 回のステアリングコミッティに上程し、意思決定をはかることとした。なお、体制を検討する上では、「3.7 意思決定プロセスの策定（観点 F)」のタスクの内容「②意思決定プロセスの体制」を参考にした。

(2) 品質保証の検討

① 業務継続性担保の具体化

(a) 品質目標の設定

再構築における、基本的な品質目標の設定例

「業務の運用が支障なくできること、支店の運用に支障がないこと。」

(b) 業務継続性を担保するための計画策定

本ケースでは、図 3.32 に示すように作り込みから検証までの各工程での品質検証の積み上げにより、業務継続性を担保していく計画を示す。最終的な業務継続性を担保する考え方は、ステークホルダ間で合意形成し、各工程での取り組み、検証計画を策定する。

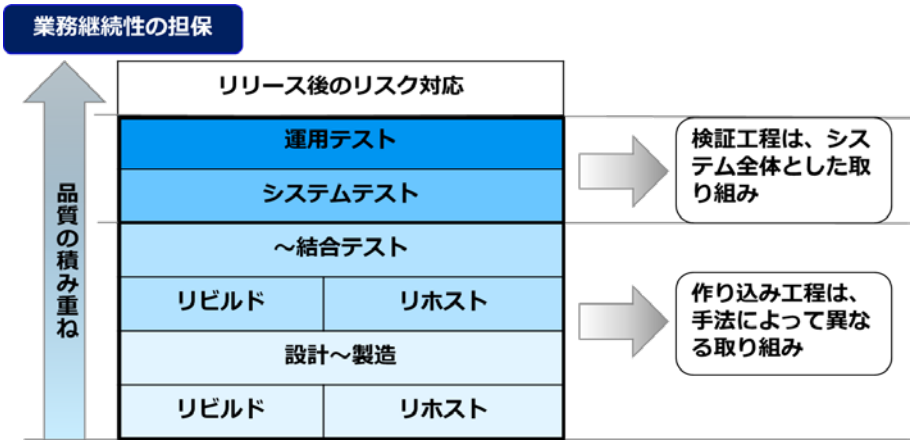


図 3.32 業務継続性担保の考え方例

次ページから、図 3.32 に示している工程と手法ごとの実施方針を示す。

② 実施方針の検討

(a) 設計～結合テストの取り組み

設計～結合テストまでは、再構築手法によって取り組み内容が異なる。そのため、手法の違いに着目して、業務継続性を担保するための品質の基礎として設計～結合テストまでの実施方針（品質確保方針）を策定する。取り組み例を表 3.12 に示す。

表 3.12　設計～結合テストまでの取り組み例

区分		設計	製造	～結合テスト
リビルド	検証観点	「新規開発と同様」（注）		• 設計書に元づいた、画面遷移や画面への出力内容およびマスタファイルの登録内容等の妥当性検証（ホワイトボックステスト）
	検証内容			• オンライン画面遷移 • マスタファイルの更新結果 • 画面メッセージの内容　など
リホスト	検証観点	• 非互換調査の充分性 • 移行方法、変換仕様、基盤部品の妥当性	• 変換後資産が、変換仕様どおりになっているかの検証	• 現行システムとの同値性 • 変換仕様パターンの網羅性
	検証内容	• 設計レビューおよび処理方式と変更仕様パターンの妥当性確認を目的としたパイロット検証	• 変換後資産のレビュー（ツール変換はサンプル、手修正は全箇所）	• 現行システムとの処理結果の比較による、新システム側の処理結果の妥当性（現新比較検証） • マスタファイルの出力結果 • 帳票の出力結果

（注）新規開発と異なる観点は、設計書に基づくだけでなく「現行システムありき」になることである。このため、利用部門では、新規開発以上にオンライン画面や操作性検証の早期参画を計画する必要がある。

(b) システムテスト以降の取り組み

システムテスト以降については、再構築手法の違いによらずシステム全体として業務継続性を検証する。そのための取り組み（検証観点、検証内容）の例を表 3.13 に示す。

表 3.13　システムテスト以降の取り組み例

区分	システムテスト1	システムテスト2 （サイクルテスト）	運用テスト （並行稼動）
検証観点	• 各業務毎のシナリオの処理結果が正当であること • 業務シナリオによる機能検証（正常系/異常系）各種運用・保守ツールの運用検証（リリース検証、障害通知検証など）	• 数日間の全主要業務が完了するまで（YY.MM.DD～YY.MM.DD＋n日） • 日付を合わせた移行データで、処理結果の数値（売上等）が現行と同値であること • 繁忙期の性能検証、トラブル時の信頼性検証	• 現行と同じ運営運用を新システムで継続して維持できること • 日々の業務運営の妥当性、正当性を検証するため、システム運用を自動実行し、他システム連携を含む、各処理間の整合性、情報収集、提供タイミングを評価する
検証内容	• 各業務イベント（シナリオ検証） • 全オンライン • 全バッチジョブ • 全帳票検証 • 全サーバ連携	• オンライン業務検証（コンペア検証） • バッチ業務検証（コンペア検証） • 数値検証（日次、月締め等でのデータ内容の妥当性とデータ間の整合性） • 主要帳票検証 • 内部サーバ連携検証 • 非機能要件（冗長化、性能、バックアップ、リカバリ、システム監視）	• 本番アクセス権での通常運営、異常時運用検証 • 外部データ連携検証 • 性能検証（レスポンス、夜間バッチ） • データ移行検証 • 現場キーパーソン教育 • マニュアル検証
実施主体	ベンダ	ユーザ	ユーザ

なお、本ケースでは、オンライン業務とバッチ業務を連動したシナリオ作成に、要件定義工程で作成した「システム化業務フロー」を活用して整備した。

また、最終的な「業務継続性」を判定する項目や完了基準のケーススタディとして、判断基準（クライテリア）の例を以下の表 3.14 に示す。

表 3.14　テスト完了の判断基準の例

＜凡例＞○:完了している、△:一部未完了（完了見込みあり、不安なし）、×:完了していない（不安がある）

判定項目	オン	バッチ	完了基準	判定根拠等
業務機能	△	○	• オンライン:全画面の検証が完了している • バッチ:全バッチ処理の検証が完了している • 日常の業務オペレーション（月末、期初イベント含む）ができる	
運用管理			• トラブル時のリカバリ対応ができる（切替えが目標時間内にできる） • 各種障害を検知し、想定とおり通知される	
性能			• オンライン:目標レスポンスで応答している • バッチ:目標時間（現行より早く）、バッチ処理が完了している	
データ移行			• オンライン:全テーブル移行できている • バッチ:過去分含めて全データ移行できている（過去分はデータ検証が完了している）	
習熟度			• 基本的な操作ができる • 現行のレベル（時間/品質）で通常対応、トラブル対応ができる	

(c) リリース後のリスク対策

本ケースではリリース後に、数週間の重点監視、バックアップ対応を実施する計画とした。重点監視、バックアップ対応中のリスク対策例を以下に示す。

- システム異常発生時の通知、連絡ルートおよび回復手順の設定
- アプリケーションアベント発生時の調査、解析手順
- 実行時非互換発生時の対応方法（リホスト）

(3) システム化計画（まとめ）

企画工程の計画策定のまとめとして、次の構築段階での工程計画（マスタスケジュール）や体制、ユーザ企業とベンダ企業の役割分担などの立案がある。

本ケースでの、計画策定のケーススタディのまとめとして、後工程の構築段階のマスタスケジュールや体制の元になるアウトプット例を図 3.33 に示す。

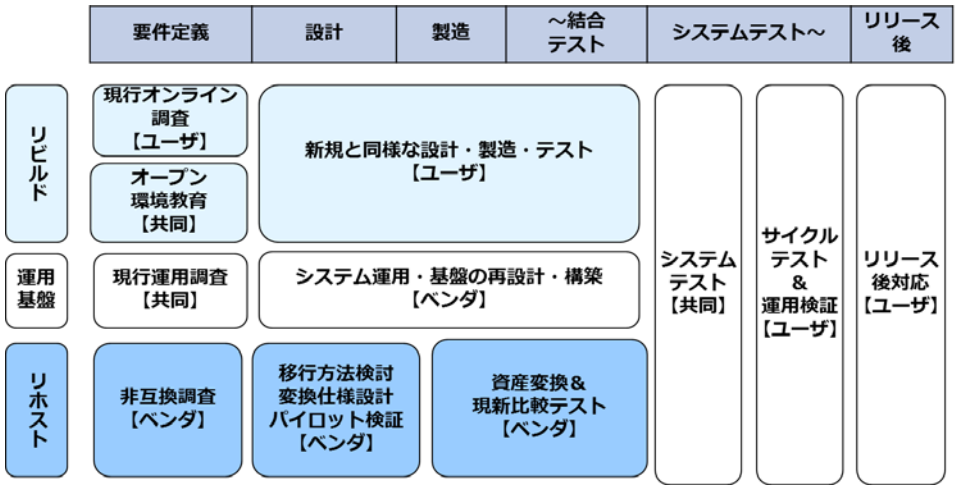


図 3.33　全体工程計画例

第４章　事例編

はじめに

事例編で紹介する各社の取り組み・サービスについて、本編との関連を以下の表に示す。

表 4.1 各社事例と本編との関連

No	本編との関連	事例タイトル
1	第 2 章 再構築手法選択編	4.1 再構築手法選択編の事例 株式会社 NTT データ
2		4.2 再構築手法選択編の事例 富士通株式会社
3		4.3 再構築手法選択編の事例 株式会社 日立製作所
4		4.4 再構築手法選択編の事例 日本電気株式会社
5	第 3 章 計画策定編	4.5 計画策定編「現行踏襲内容の明確化(観点 B)」の事例 富士通株式会社
6		4.6 計画策定編「現行資産活用方針の検討(観点 C)」の事例 日本電気株式会社
7		4.7 計画策定編「現行業務知識不足への対応(観点 D)」の事例 株式会社 ソフトロード
8		4.8 計画策定編「品質保証の検討(観点 E)」の事例 株式会社 NTT データ
9		4.9 計画策定編「意思決定プロセスの策定(観点 F)」の事例 東京海上日動システムズ株式会社
10		4.10 計画策定編「再構築の計画と見積り(観点 H)」の事例 株式会社 日立製作所
11		4.11 計画策定編「再構築の計画と見積り(観点 H)」の事例 JFE システムズ株式会社

(※) 各事例の構成は統一せず、事例ごと固有の構成にしている。また、各社の固有用語で記載していることから、本編と意味合いが異なる場合がある。

4.1　再構築手法選択編の事例
株式会社 NTT データ

取り組み背景

システム再構築プロジェクトにおいて再構築手法の正しい選択がプロジェクト成功の鍵を握っている。再構築手法の選択を誤ると、システムに求められる要件や品質を満たせず、手戻りが発生する。結果として大幅なコスト超過やプロジェクト延伸といった問題に直面することとなる。

再構築手法の選択の難しさは、「現行システムの状態、新システムに求められる要件や品質、コスト・要員の制約などによって適切な手法が変わる」ことに起因する。ゆえに各再構築手法の特徴を理解し、プロジェクトやシステムの状態を調査したうえで総合的に判断する必要がある。適切な再構築手法を選択するための重要なポイントは 2 点ある。

- 各再構築手法を選択した場合の重大なリスク（開発難易度が著しく高くなる）要素を評価する。
- プロジェクトとして重視したい要素（品質、コスト、期間など）から各再構築手法の適切さを評価する。

本事例では、ユーザ企業が再構築手法を総合的に判断するための支援として取り組んでいる「再構築手法診断サービス」について紹介する。

本編との関連

再構築手法選択編「2.2　現行システムの調査・分析（ステップ 1)」～「2.5　再構築手法の決定(ステップ 4)」と関連する。

再構築手法診断サービス

(1) 概要

「再構築手法診断サービス」とは、ユーザ企業が適切にシステム再構築手法を判断するために現行システムや要員スキルなどの評価を行うサービスである。新システムに求める要件も加味し、各再構築手法を選択した場合のリスクやコストを洗い出すことで適切な再構築手法を選択するためのサポートを行う。

(2) 取り組み内容

再構築手法診断サービスは図 4.1 の①～⑤の取り組みで実施する。

循環複雑度	コメント	調査結果 S プログラム数	S 割合	A プログラム数	A 割合	その他 プログラム数	その他 割合
0-10	非常に良い構造	50	25.4%	41	24.6%	18	26.0%
11-29	良い構造	77	40.4%	84	47.4%	27	40.5%
30-49	構造的にリスクあり	46	20.3%	37	22.4%	19	27.6%
50-74	テスト困難	29	5.7%	19	8.6%	6	9.1%
75-	いかなる変更も困難正を生む	41	19.4%	28	17.5%	9	17.4%

チェック優先度	項目	説明	評価
高	GO TO文の利用	EXITパラグラフ以外へのGO TO文、無条件GO TO、動的GO TOなどの利用度	×問題あり 5系の資材で多様にGOTO文を使用している。また、無条件GOTOや動的GOTOなど禁止している。
中	EXTERNAL句の利用	PGM内での外部アクセスなどの利用	○問題なし
中	DEPENDING ON句の利用	PGM内での可変長テーブルの利用度	○問題なし
低	フォーマットの統一性	データ項目／パラグラフ名が命名規則に沿っているか、コメント／インデントの統一性など	○問題なし

図 4.1 再構築手法診断サービスの流れ

(3) 効果

再構築手法診断サービスを適用した場合、ユーザ企業にとっての効果として以下の点が挙げられる。

- ドキュメント資産分析やプログラム資産分析によって見積りの精度を高めることが可能となる。
- 各再構築手法のリスク評価からアクションプラン提案まで支援を実施するため、その判断材料をもとにユーザ自身が更改計画を立てられる。

適用事例

(1) 概要

本事例は、数十年間稼動しているシステムを更改するにあたり、再構築手法の選択を支援したものである。以下にプロジェクト概要を示す。その他の概要については次項にて説明する。

システム区分　：　バッチシステム
言語区分　　：　オープン COBOL
システム規模　：　約 500KStep
基盤　　：　メインフレーム

(2) 実施内容

① ヒアリング

ヒアリングの結果、本プロジェクトは優先順位の高い順に以下の制約があると分かった。

(a) 開発コストが予算内に収まること
(b) 保守コストが可能な限り低くなること
(c) 10 年の運用で投資コストの 80%が回収できること

上記の要求と次工程で調査する現行システムの状態から再構築手法を選択していく。

② 現行システムの調査・分析

本事例では各再構築手法のリスクを評価するとともに、コスト重視の調査を行う必要がある。そこで開発・保守コストに影響する要素として「ドキュメント網羅性・信頼性（仕様の再定義・保守工数に影響)、プログラム複雑性（再設計・テスト工数に影響)、非互換（リホスト時の製造工数に影響)」などを調査した。

表 4.1 本事例における調査内容

No	調査	指標	説明
1	ドキュメント資産分析	網羅性	システム全体のうち仕様が明記されているレベルを評価
		信頼性	記述された仕様の信頼レベルを評価
2	プログラム資産分析	規模	母体規模を評価
		アーキテクチャ構造	アーキテクチャ構造の複雑さ･保守し易さを評価
		複雑性	業務仕様の複雑さを評価
		可読性	プログラムの読み易さを評価
		クローン率	コードクローンを RSA 率で評価
3	非互換分析	非互換	プラットフォーム変更による修正内容と修正量を評価
4	ステークホルダ分析	有識度	システム仕様を再定義可能な要員の有無と数を評価 ※特にユーザ企業と現行保守要員を評価
5	保守生産性分析	保守プロセス	保守プロセスの全体像と生産性を評価

③ リスク評価

②で現行システムの調査・分析をもとに、各再構築手法についてリスク評価を実施した。結果は以下の表 4.2 に示す通り、本案件でのリビルドの選択は、リスクが高いため選択肢から外すことにした。残りの再構築手法については④コスト見積りで 10 年間の費用対効果を調査し、比較することとした。

表 4.2 各再構築手法のリスク評価結果

	再構築手法			
	リビルド	リライト	リホスト	ハードウェア更改
評価	×	△	△	○
備考	・業務仕様把握が困難 ・再構築が必要な仕様が多い →コスト面や開発工程手戻りのリスクが高い	・現行プログラムから処理仕様を流用できる ・特殊な開発言語なし →選択可能	・リリース条件(テストの完了条件)を設定可能 →選択可能	リスクなし →選択可能

×:リスク高／△:リスク回避可能／○:リスク低

④ コスト見積り

②で説明した開発・保守コストに影響する要素の調査結果より、開発・保守コストの確度を再構築手法の選択を誤らない（明確に差別化できる）レベルまで高めた。また、見積った開発・保守コストをもとに各再構築手法を選択した場合の 10 年間の費用対効果を算出した。

その結果、本プロジェクトでは・開発コストが安く、10 年後の費用対効果が最も高いのはリホストであることが分かった。

⑤ 報告

本案件では①で示した（a)、(b)、(c）の制約を満たしたのがリホストのみであったため、リホストを選択することとした。また、ユーザ企業の新システムの運用方針も考慮し、リホストの方法としてはオープン環境への移行が最も適すると判断した。

(3) 適用効果

再構築手法診断サービスを適用した結果、現行システムの状態の把握を経て、ユーザ企業が再構築手法を選択するための情報をご提供することが可能となった。また、選択した再構築手法のリスクが明確化され、適切な開発の工程定義も実施可能となった。

4.2　再構築手法選択編の事例
富士通株式会社

取り組み背景

長期に渡り運用・保守してきたシステムが肥大化・複雑化し続け、既存情報システムが業務の硬直化を招き、イノベーションを阻害する要因になりつつある。このため情報システムのモダナイゼーションが喫緊の課題となっている。富士通はインフラのみでなく、アプリケーションを含めたシステム全体をモダナイゼーションするサービスを提供している。

本事例では、モダナイゼーションの第一歩である現行システムの状況を分析のうえ正確に把握し、モダナイゼーションに向けた課題を抽出するアプリケーション資産分析サービス（ソフトウェア地図によるアプリケーションの見える化）の事例を紹介する。

本編との関連

再構築手法選択編「2.2　現行システムの調査・分析（ステップ 1)」と関連する。

アプリケーション資産分析サービス

(1) 概要

当社では、お客様システムの保守品質・保守コスト改善に向けたスリム化や、次期システム構築を効率的に実施することを目的として、アプリケーション資産を分析のうえ、地図形式で表現した『アプリケーションの見える化』診断を提供している。ソフトウェア地図によるアプリケーションの見える化により、お客様システムのアプリケーション資産の複雑度、肥大化度などの傾向をつかむことができる。

これにより、お客様の課題に対する施策検討など、再構築時のお客様システムの改善活動に役立てることが可能である。

(2) 取り組み内容

① ソフトウェア地図によるアプリケーションの見える化

アプリケーションの構造分析から、ソフトウェア地図と呼ぶ直感的に理解できる 3D 地図形式で表現し、アプリケーションの全体像や問題箇所を見える化する。この分析を行うことで、調査対象の資産や再構築対象を絞り込むことが出来る。

富士通が独自に開発した「ソフトウェア地図」は、一つひとつのプログラムをビルに見立て、プログラム単体の複雑度をビルの高さで表現し、2 次元平面上に配置する。(図 4.2 ソフトウェア地図のイメージ)

アプリケーションの全体構造は、ブロック（街区）を形成して、ビル間の距離で表現する。さらに、設計情報からプログラムを分類、グループ化し、サブシステム毎に色付けし見やすくする。

例えば、同一のブロックに高いビルが密集している場合、そのブロックに対応する機能を構成するプログラムのほとんどが複雑であることを示している。その複雑さを解消するためブロックを切り出し、そのブロックを優先的に刷新するといった計画が立てやすくなる。

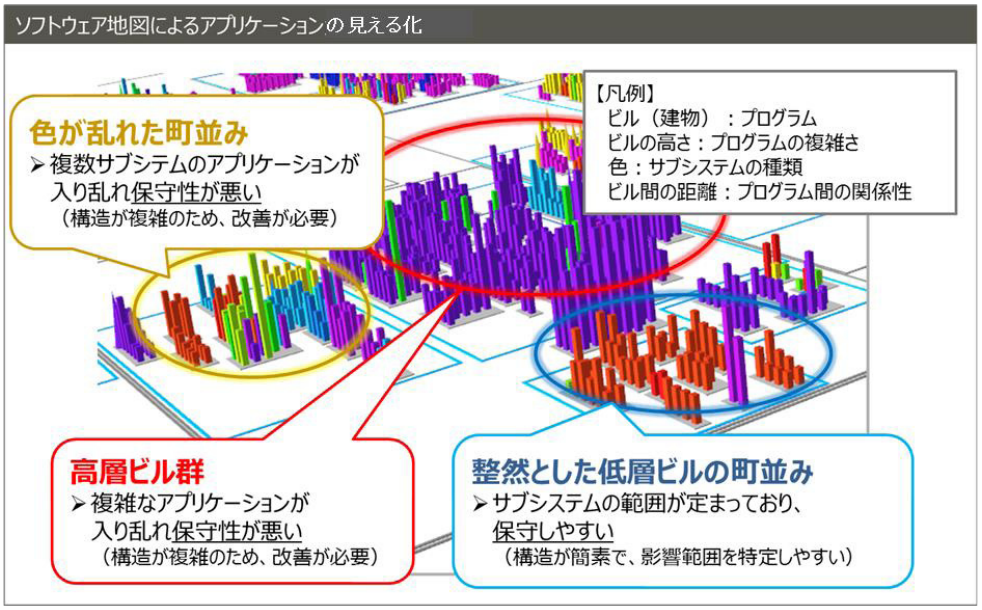


図 4.2 ソフトウェア地図のイメージ

② アプリケーション資産詳細分析

目的に応じて、より詳細な分析方法を 4 種用意している。

ソフトウェア地図による見える化により資産分析の方向性を決めてから適用することで詳細分析作業の効率化やコストセーブが図れる。

(a) 稼動資産分析：

起点情報（ジョブ情報など）より、アプリケーション資産の呼出関係や稼働ログなどの情報をもとに、稼働している資産を明確にする。

(b) 類似分析：

ソースコードを 1 対 1 で総当たり比較し、類似した資産を明確にする。

(c) 資産特性分析：

プログラム単体の複雑さ、アプリケーション構造の複雑さ、メンテナンス特性を分析し、保守性を悪化させる原因を明確にする。

プログラム単体は、条件分岐の複雑さを示す MaCabe の閉路複雑度、プログラムの構造化度などの指標により評価する。アプリケーション構造の複雑さは、入出力

データ、呼び出し関係を調査し、データを介したプログラム間のつながりを評価する富士通独自のインパクトスケールを用いて評価する。

(d) システム相関分析：

業務やサブシステムとファイル・データベースの関係（Create, Read, Update, Delete)、プログラムやジョブとファイル・データベースの関係を明確にする。

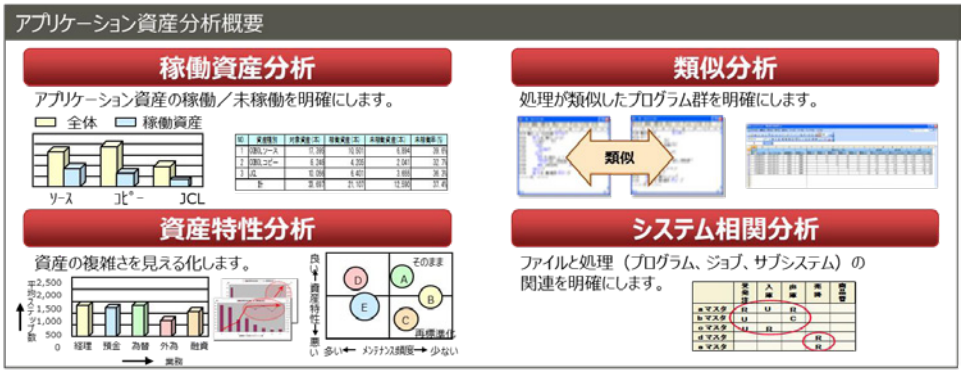


図 4.3 4 種の詳細分析

(3) 効果

ソフトウェア地図による見える化の効果として、以下の点があげられる。

- アプリケーション資産の構造を地図形式で表現し、問題点を直感的に把握できる。
- ブラックボックス化したアプリケーションの問題箇所を短期間で見える化できるため、早期の改善施策立案が可能。
- 経営者などのプログラムに詳しくない要員と、アプリケーション構造について共通認識を図ることができ、現行システムの改善施策を検討できる。

アプリケーション資産分析では、特に以下のような課題を持つ企業に効果がある。

【次期システムの企画での課題抽出】

- 次期更改時の課題把握のために、現行アプリ資産の特性を把握しておきたい。
- 次期システム再構築を検討中だが、費用見積りに当たり全体の規模感をつかみたい。
- 現状使用しているアプリ資産規模を把握し、次期システムの規模感を想定したい。

【現行システムの保守・運用での課題抽出】

- アプリの複雑度の傾向をつかみ、保守の容易性や次期システム構築の難易度の感覚を押さえたい。
- 現状のアプリの資産状況が全く見えていないので把握したい。
- アプリの保守性が悪い（保守費が大きい）ため、原因を探りたい。

適用事例

次期システム再構築に向けて、現行システムの資産調査と現行資産の活用方針策定に本サービスを適用した事例である。

＜お客様の課題＞

システム再構築にあたり、できるだけ現行資産を活用したい。そのために、現行資産の活用についての判断材料が欲しい。

＜サービス適用の狙い＞

ソフトウェア地図を用いたアプリケーション資産分析で、再構築に向けての現行資産の活用を判断する。

＜適用ポイント＞

プログラムやプログラムの管理状況を評価し、活用の判断材料とする。

- プログラムの稼働／未稼働を確認する
- 段階移行を検討する際のプログラムの切り出し単位を明確にする
- プログラムの保守性を評価し、次期システムでの活用可否を判断する
- 移行難易度を評価する

＜実施内容＞

ソフトウェア地図を作成する際の分析項目を表 4.3 に示す。

表 4.3 ソフトウェア地図の分析項目

№	分析項目	分析項目概要
1	複雑度	Thomas McCabeの閉路複雑度計量法により、セクション毎の複雑度を算出し、閉路複雑度が10を超えるセクションの数をカウントして、プログラム全体の複雑度を示す。値が大きいほど、コードが実行される際の分岐の数が多く、理解や修正が難しい。
2	構造化度	プログラムの構造化を阻害する文の使用頻度を分析し、構造化の状況を把握するための項目。
3	有効ステップ数	手続き部ステップ数 （宣言部、コメント行を除き、コピーを展開しないステップ数）
4	コメント率	プログラムソース中のコメント率（%） （コメントステップ数÷全体ステップ数×100）

（注）上記事例は、COBOL 版で Java 版も同様にあり。

＜実施結果＞

診断結果から、一部複雑度が高いプログラムはあるものの（図 4.4 の③）、多くのプログラムは、複数機能を跨る呼び出しは少なく、一部機能を切り出しやすい傾向にあること（図 4.4 の②）から、現行資産を活用することを決定した。

その後、移行難易度評価、資産移行の試行検証（POC）を行い、ホストから Windows プラットフォームへのモダナイゼーションを行った。

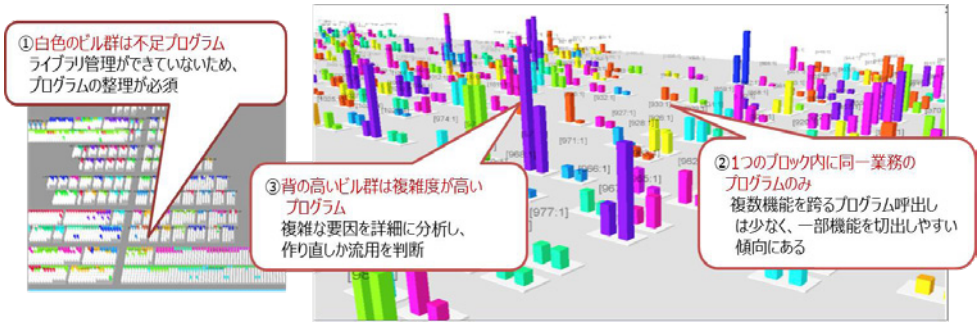


図 4.4 ソフトウェア地図の分析結果

■公開ホームページの紹介

以下の「ソフトウェア地図によるアプリケーションの見える化技術」を参照

http://www.fujitsu.com/jp/services/application-services/application-management-outsourcing/apm/services/modernization/sort/map/index.html

4.3 再構築手法選択編の事例
株式会社 日立製作所

取り組み背景

要員の入れ替えなどが原因でシステムに精通した担当者がいなくなり要件定義ができないとか、システム保守をベンダに依存しているため、現行システムの資産状況が分からないなど、システムを再構築するにあたって、様々な問題を抱えているユーザが存在する。

日立製作所では、このようなユーザに対し、システム資産や業務のホワイトボックス化によって、システム再構築時の現行資産調査をサポートするプログラム資産棚卸／プログラム仕様可視化／業務仕様可視化サービスを提供している。

これらのサービスは、アプリケーションの分析技術を用いてアプリケーションのライフサイクル全般の問題解決をサポートするアプリケーションライフサイクルマネジメントサービス［ALM（※1）］で提供しているものである。

以降では、再構築手法選択編ステップ 1 に関連する図 4.5 に示す 3 つのサービスについて紹介する。

- プログラム資産棚卸サービス（図 4.5 の②）
- プログラム仕様可視化サービス（図 4.5 の③）
- 業務仕様可視化サービス（図 4.5 の④）

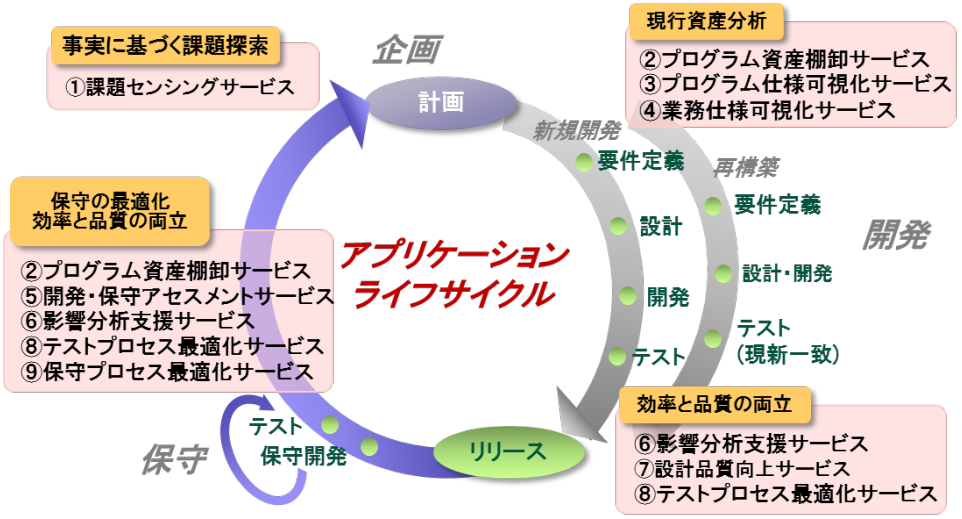


図 4.5 ALM のサービス全体図

（※1）Application Life cycle Management Service

本編との関連

再構築手法選択編「2.2　現行システムの調査・分析（ステップ 1)」と関連する。

プログラム資産棚卸サービス

(1) 概要

当該サービスは、現行システムのプログラム資産を解析して得られる資産間の関連情報と、システムのログから得られる稼動実績を元に、業務に必要なプログラム資産を洗いだすサービスである。

(2) 取り組み内容

以下の 2 点を実施する。

① 現行システムのプログラム資産を独自のツールで解析し、得られた関連情報をたどって起点に紐付いたプログラム資産を洗い出す（静的な観点）

② システムのログから得られる稼動実績と関連情報を統合し、実際に稼動しているプログラム資産を洗い出す（動的な観点）

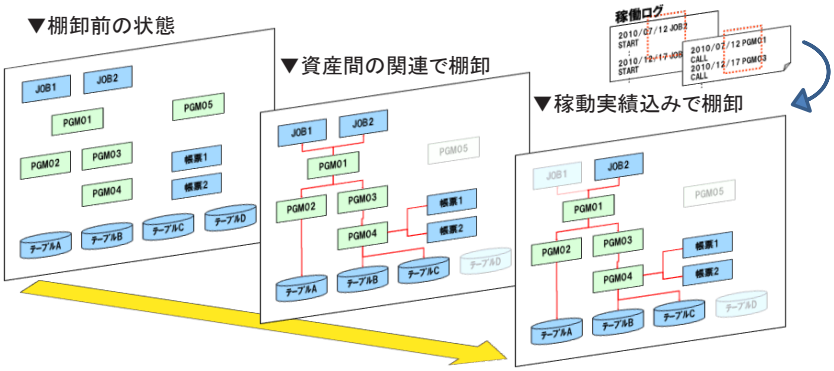


図 4.6 プログラム資産棚卸サービスのイメージ図

(3) 効果

サービスにより以下の効果が得られる。

- 再構築プロジェクトの前提となる既存システムの範囲／規模を特定することができる。
- 不要なシステム資産を特定することで再構築にあたって検討するシステムの範囲を絞り込むことができる。

プログラム仕様可視化サービス

(1) 概要

当該サービスは、現行システムのプログラム資産を解析し、プログラム構造と仕様情報を可視化するサービスである。

(2) 取り組み内容

以下の 2 点を実施する。

① COBOL を含めジョブネット／JCL／テーブル定義などのプログラム資産を解析し、仕様情報を可視化する独自ツールを提供する。(Java をはじめ、オープン系言語にも対応)

② システムの特性やお客様要件に応じて分析方式やアウトプットをカスタマイズして提供する。

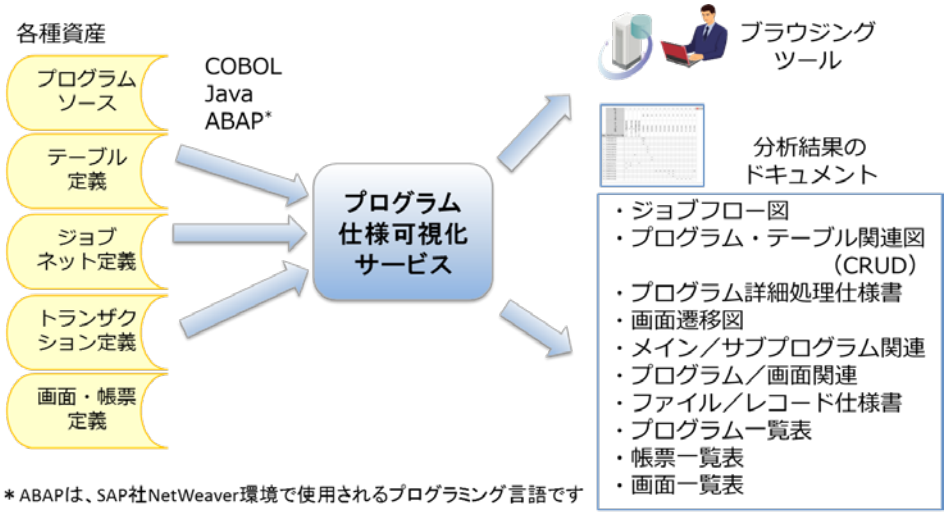


図 4.7 プログラム仕様可視化サービスのイメージ図

(3) 効果

サービスにより以下の効果が得られる。

- 再構築手法検討の前提となるシステムの全体的な構造を把握することができる。
- ツールによる調査で必要となる既存システム資産の調査を効率化できる。

業務仕様可視化サービス

(1) 概要

当該サービスは、プログラム仕様可視化結果をベースに情報収集、ドキュメント分析を実施、情報システム資産と業務の情報を関連付け、業務レベルの仕様を作成するサービスである。

(2) 取り組み内容

以下の 2 点を実施する。

① 業務参照モデルや用語辞書、業務マニュアルなどのドキュメント分析、ヒヤリングによる情報の分析を行って、システムの業務レベルのドキュメント（業務フロー図、業務機能階層図、システム機能階層図　など）を作成する。

② プログラム仕様可視化の結果を関連付けて調査することで整合性を確保すると共に、調査の抜け漏れを防ぐ。

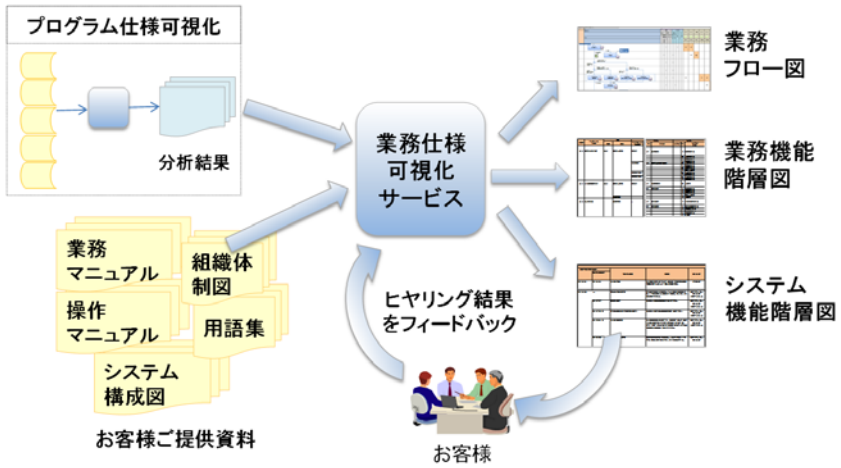


図 4.8 業務仕様可視化サービスのイメージ図

(3) 効果

サービスにより以下の効果が得られる。

- プログラム仕様可視化の結果を関連付けて調査することで、整合性を確保すると共に、調査の抜け漏れを防ぐ。
- 業務レベルのドキュメントを整理すると共に、情報システムと関連付いた情報を得ることができる。

適用事例

次期リプレース開発提案に向けて業務仕様可視化サービスを用い、現行システムに関連する業務の洗い出しを実施した事例である。

＜お客様の課題＞

お客様の課題は以下の 2 点である。

- 業務の洗い出し漏れがないか確認のために現行業務とシステムとの関係が俯瞰できる業務仕様書類が欲しい
- 他ベンダが管理している業務とのインターフェース部分を見えるようにしたい

＜実施内容＞

これらの課題に対して、システム機能階層図、業務機能階層図、業務フロー図及び外部インターフェース一覧の 4 種類の仕様書を作成し、主要業務と主要業務に関与するシステム間のインターフェースの可視化を実施した。

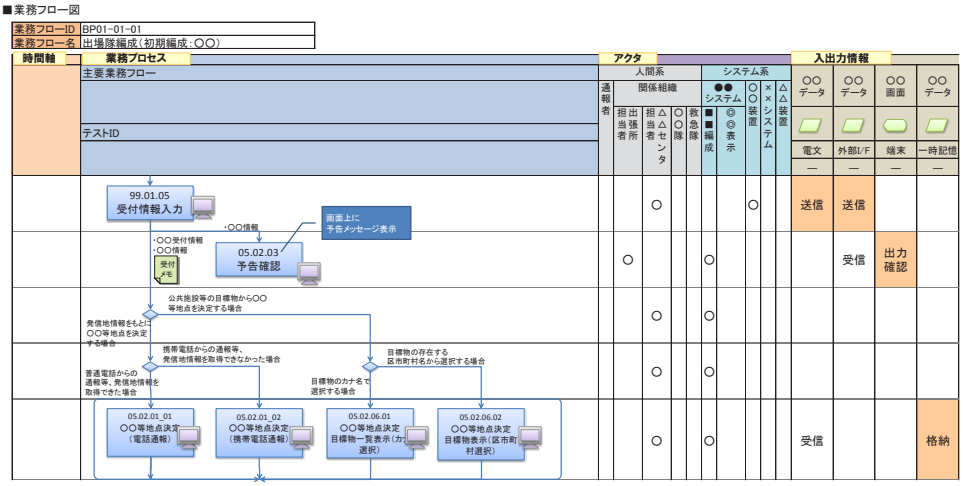
図 4.9 に成果物のイメージを示す。

＜実施結果＞

実施した結果、以下のような効果があった。

- システムと業務の関連が一目瞭然になった。
- 業務に絡むシステム間のインターフェースが見えるようになり、関連が分かりやすくなった。
- お客様が別途実施している業務の洗い出し結果の抜け漏れチェックができるようになった。

■業務フロー図



■業務機能階層図

#	Lv3 小分類		Lv5			Lv5		Lv6		
	小分類ID	小分類	実務機能ID	実務機能	実務機能概要	論理作業ID	論理作業	実務作業ID	実務作業	実務作業概要
1	BP01-01	チーム編成	BP01-01-01	出動チーム編成（初期編成：□□）	□□通報の内容を元に、□□等の発生地点の特定を行ったうえで、事前に登録されたマスタデータの内容に従って■■センター及び署所において、	99.01.01	□□通報	99.01.01	□□通報	□□等の発生をうけ、電話・携帯電話等で■■センタに対し連絡する。
2						99.01.02	通報内容聴取・発信地問い合わせ	99.01.02	通報内容聴取・発信地問い合わせ	□□等の通報を■■センタが受け付け、□□等の内容の問い合わせを行う。また受信した電話番号情報から、××システムに対し発信地情報を問合せる。
3								99.01.03	発信地情報検索	通報者の電話番号をもとに、××システムまたは、△△装置が発信地情報を特定する。
4						99.01.04	類似通報判定	05.02.07	△△表示	当日分の△△事案を表示する。

■システム機能階層図

#	Lv4		Lv5		運用条件			Lv6 画面		Lv6 要素機能	
	中機能ID	サービスコンポーネント（中機能）	小機能ID	サービスアイテム（小機能）	サイクル	時間帯	開始時刻	画面ID	画面名	サービスエレメント（要素機能）	機能概要
1	4.01	●●サブシステム	4.01.01	指定区域内町名選択	随時	24時間	ー	X1A01V	●●一覧画面	初期表示	△△装置から●●電文を受信し、受付情報から町名を表示し住所の特定を行い、指令予告処理に制御を渡す。また、●●電文に発信地情報がある場合には発信地情報の表示を行い、●●要求を送信する。
2											
3											
4											
5										送信（ENTER）	
6											
7											
8											
9										次画面要求（PF06）	
10											
11										前画面要求（PF05）	
12											
13			4.01.02	発信地自動表示	随時	24時間	ー	X1A01V	●●一覧画面	初期表示	△△装置からの発信地情報を町名一覧画面、及び端末対応の一時記憶へ出力する。また発信地情報の住所に対応する●●要求を出力する。
14											
15											
16											
17											

図 4.9 成果物イメージ

4.4　再構築手法選択編の事例
日本電気株式会社

取り組み背景

企業は、経済成長率が低迷する中、事業効率化をサポートするSoR(System of Record)の安定性や保守レベルを維持しつつ、新事業創出・事業変革を支えるSoE(System of Engagement)の開発へICT投資をシフトすることが急務となっている。しかし、長い時間をかけて作り上げられたSoRは、開発・保守要員の高齢化や退職、設計書の欠損やソースコードとの乖離などが原因で、保守コストや品質低下リスクが増えている。

既存資産の解析により資産の主要構造を見える化し、既存システムの構造理解の手助けをするとともに、資産間の関係情報より資産改修時のリスクを低減する仕組みが必要である。

本事例では、その取り組みを紹介する。

本編との関連

再構築手法選択編「2.2　現行システムの調査・分析（ステップ1)」と関連する。

業務アプリケーションの可視化／診断

(1) 概要

アプリケーション資産の全体像から、定義体のデータ項目、オンライン／バッチ資産の構造、資産間の引用・利用関係の把握と、段階的かつ、多角的な資産の把握を行っている。

(2) 取り組み内容

取り組み内容としては、以下の2点がある。

① 資産可視化

② 資産診断

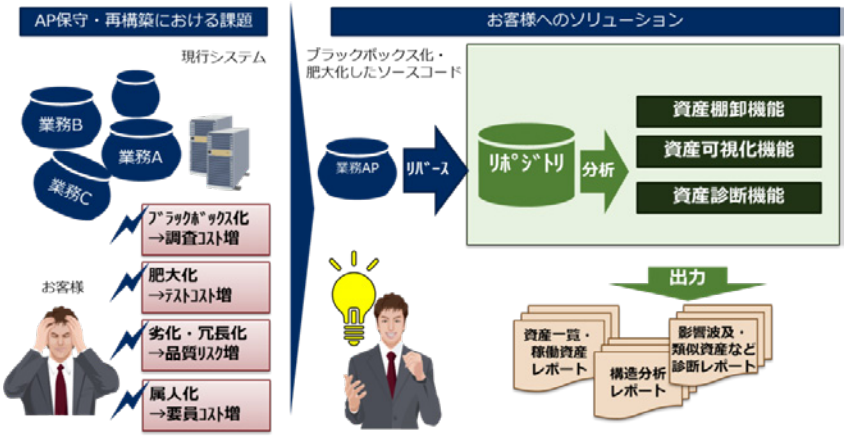


図 4.10　業務アプリケーション資産の可視化／診断の概要

(3) 資産可視化

アプリケーション資産の全体像から、定義体のデータ項目、オンライン／バッチ資産の構造、資産間の引用･利用関係の把握と、段階的かつ、多角的な資産の把握を行っている。

図 4.11　業務アプリケーション資産の可視化／診断の内容

(4) 資産診断

アプリケーションの改修を行う場合の、ソースコードの影響波及範囲の把握や、ソースコードに内在する複雑度やクローンコードの把握を行う。

① 影響波及診断

構造分析によってリポジトリに登録された資産間の関係情報を基に、ある資産を改修した場合の影響が他の資産にどのように波及するかを診断し可視化する。

② データフロー診断

業務 AP 中で業務データ（画面／DB）がどのように参照・更新されるかを診断し、業務 AP が持つデータの流れから業務処理への理解を支援する。

③ メトリクス診断

サイクロマチック複雑度、プログラム間の呼出／被呼出関係、データ結合度などメトリクス値を元に保守難易度を診断し、業務 AP 群中に存在する品質リスクを顕在化する。

④ 類似資産診断

長年の改造により資産中に発生・蓄積したクローン資産を顕在化する。ソースコードのファイル単位で比較した類似度を算出し、変数名など局所的に修正してコピーされた冗長資産を検出する。

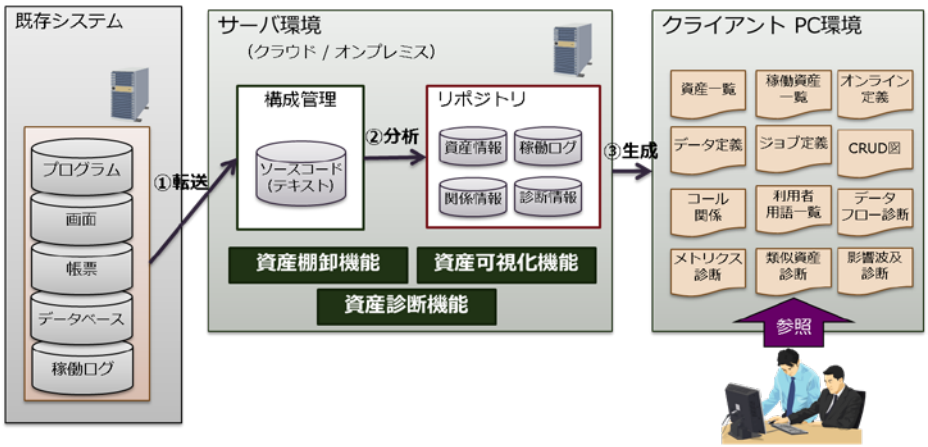


図 4.12　業務アプリケーション資産の可視化／診断の流れ

適用事例

次期システム再構築提案に向けて、現行システムの業務仕様の可視化および課題箇所の抽出を実施した事例である。

＜お客様の課題＞

- 設計書の欠損によりブラックボックス化した業務仕様を明確にしたい
- ソースコード中の冗長な部分、およびスパゲッティ化した箇所を削減したい

＜適用の効果＞

- 資産可視化により、アプリケーションの構造を俯瞰レベルから詳細レベルまで追跡可能となり、業務仕様の把握が容易となる。
- 稼働統計情報からアプリケーションの動作履歴を分析し、一定期間動作していないアプリケーションを再構築対象から除外することができる。
- 類似資産の診断により、クローンコード化したアプリケーションの集約を図ることができる。
- メトリクスの診断により、複雑化・難読化したソースコード部分の把握が容易となる。

4.5 計画策定編「現行踏襲内容の明確化（観点 B)」の事例 富士通株式会社

取り組み背景

国内企業の情報システムを取り巻く環境は、社会構造の複雑化や技術の革新により、大きく変化している。一般社団法人日本情報システム・ユーザー協会が発行した「IT 投資動向調査 2016」の調査結果によれば、IT 基盤の統合およびシステムの再構築が首位であった。また 2018 年度の予想でも首位であり、今後も IT 基盤の統合およびシステムの再構築は、企業にとって重要なテーマであることがこの調査結果からわかる。このような背景もありシステム開発では、システムの新規構築は減少し再構築が主流となっている。また現行システムは長年の維持管理および保守の改善で複雑化している。一方、現行システムには永年の業務ノウハウが蓄積されており貴重な財産となっている。しかし仕様書の最新化が滞って陳腐化が進み、システムを熟知した有識者やユーザも減少している。このような状況下で現行踏襲を前提とした再構築が実施されており、システム再構築の品質、納期、コストにおいて問題が発生している。

富士通では、この問題を解決するソリューションとして「TransMigration（トランスマイグレーション）サービス」を提供している。ユーザの将来計画を踏まえ、既存アプリケーション資産の継承と有効活用を図りながら、継続的なシステムの最適化を実現する TransMigration（トランスマイグレーション）サービスについて紹介する。

本編との関連

計画策定編「3.3　現行踏襲内容の明確化（観点 B)」と関連する。

TransMigration サービスの現行踏襲の取り組み

(1) TransMigration サービスの概要

TransMigration サービスは、既存のアプリケーション資産を分析し、その特性や使用状況などから現行踏襲する内容を明確化した上で既存アプリケーション資産を有効活用して、業務要件を変えずに異なるプラットフォームに適合するようにアプリケーションを変換するサービスである。ご提供するサービスは、企画・分析と設計・構築のフェーズに分かれている。

企画・分析フェーズでは、稼働資産分析、移行資産整理や移行方式妥当性検証（POC）のサービスを提供している。

設計・構築フェーズでは、業務要件を変えずに異なるプラットフォームに適合するようにアプリケーションを書き換えるサービスなどを提供している。概要を図 4.13 に示す。

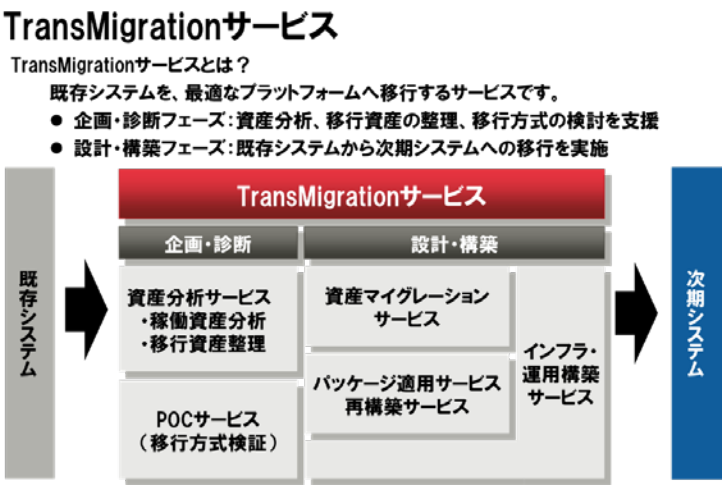


図 4.13　TransMigration サービスの全体図

(2) 取り組み内容・効果

① 現行踏襲型再構築を支援する TransMigration サービスの特長

企画・診断フェーズで提供する稼動資産分析および移行資産整理サービスを使って現行踏襲する機能および資産を明確にすることができる。

- 3 つの観点で資産を分析して現行踏襲型再構築を支援

  現行資産を「変える個所」「変えない個所」「変わってしまう個所」の 3 つの観点で分類し、業務仕様、システム仕様が変わる個所と変わらない個所をユーザとベンダで共有する。さらに変わってしまう個所の発生が予測されるケースにおいては対処策をユーザとベンダで共有する。その上で現行資産を最適なプラットフォームへ移行して再構築を実現する。3 つの観点と現行資産の関係を図 4.14 に示す。

(a) 変える個所：

新規または仕様変更したい業務機能、システム機能, ハードや OS、MW の影響を受けた変更個所（非互換）

(b) 変えない個所：

変更できない現行の業務機能（料金計算、給与計算、利率計算　など）

(c) 変わってしまう個所：

変える個所の影響を受けた変えない個所。ハードや OS、MW の変更の影響を受けた非機能要件

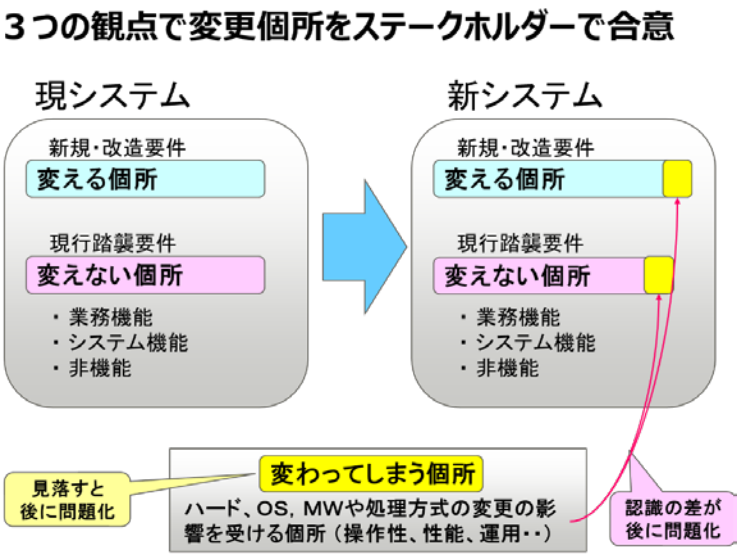


図 4.14 3 つの観点で現行資産を分析

② TransMigration サービス導入効果

- 現行システムと同等な実行環境の提供によりアプリケーションの現行踏襲が可能
  トランザクション制御など現行システムと同等で堅牢な実行環境を提供し、アプリケーションの現行踏襲を実現

- 効率的なアプリケーション移行が可能
  現行システムと共通性の高いアプリケーション実行環境を提供することにより、ホスト システムのアプリケーション資産やデータ資産の流用性が向上し、更に効率的な移行ツールを用いることで、短期間での移行を実現

- 高品質・高運用なシステム構築が可能
  富士通がこれまでのホストシステムで培った OLTP 技術をベースに、現行システム資産の流用性を高めることで、現在稼働しているアプリケーションの品質を次期システムへ引き継ぐことが可能になる。また、次期システムの運用機能を強化することで、基幹業務や大量バッチ処理にも耐えうる高い運用性を実現

③ 資産の現行踏襲を支援する実行環境

業務要件を現行踏襲し、高品質・高運用なシステムを構築して効率的なアプリケーション移行を実現するには、新システムに現行システムと同等の実行環境を構築する必要がある。

TransMigration サービスでは、現行システムと同等の実行環境を持つマイグレーションスイート製品群をご提供している。図 4.15 にマイグレーションスイートの全体図と機能概要を示す。

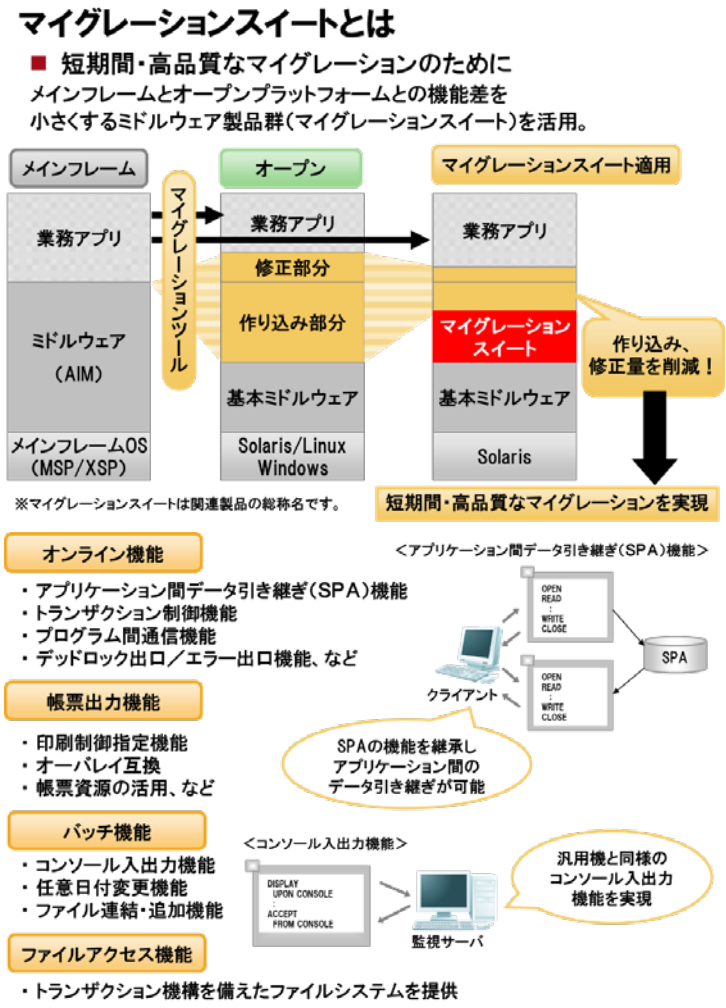


図 4.15 マイグレーションスイートの全体図と機能概要

適用事例

(1) 事例概要

本事例は、20 年間安定稼動しているメインフレームの大規模高信頼システムをオープンシステムへ移行した事例である。アプリケーション資産は 1 万 1 千本。業務を現行踏襲した。概要を図 4.16 に示す。業務の現行踏襲範囲と内容をユーザとベンダで共有して合意する作業については TransMigration サービスの計画(企画)工程と設計工程で実施する移行概要設計作業および変換仕様設計作業で行った。

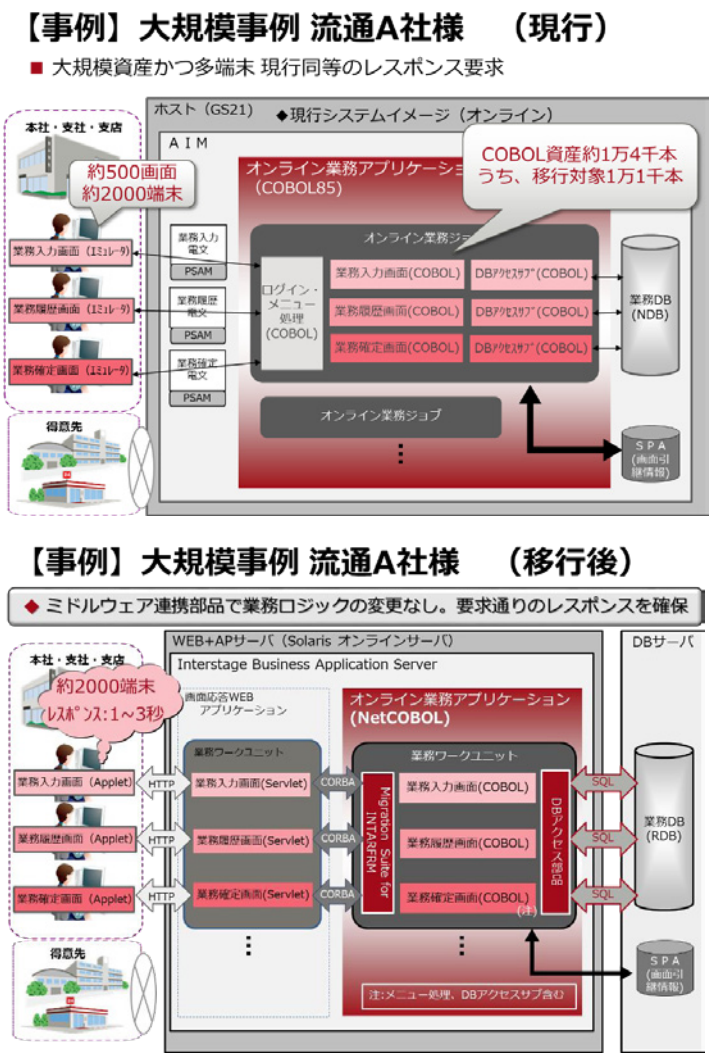


図 4.16 大規模事例　流通 A 社

(2) システム再構築方針と適用した TransMigration サービス機能範囲

本プロジェクトのシステム再構築方針および適用した TransMigration サービスの機能範囲とその他システム移行作業の関係を図 4.17 に示す。

システム移行サービスの選択に当たってはプロジェクト特性と移行要員の保有スキルに応じた移行サービスの選択が肝要である。

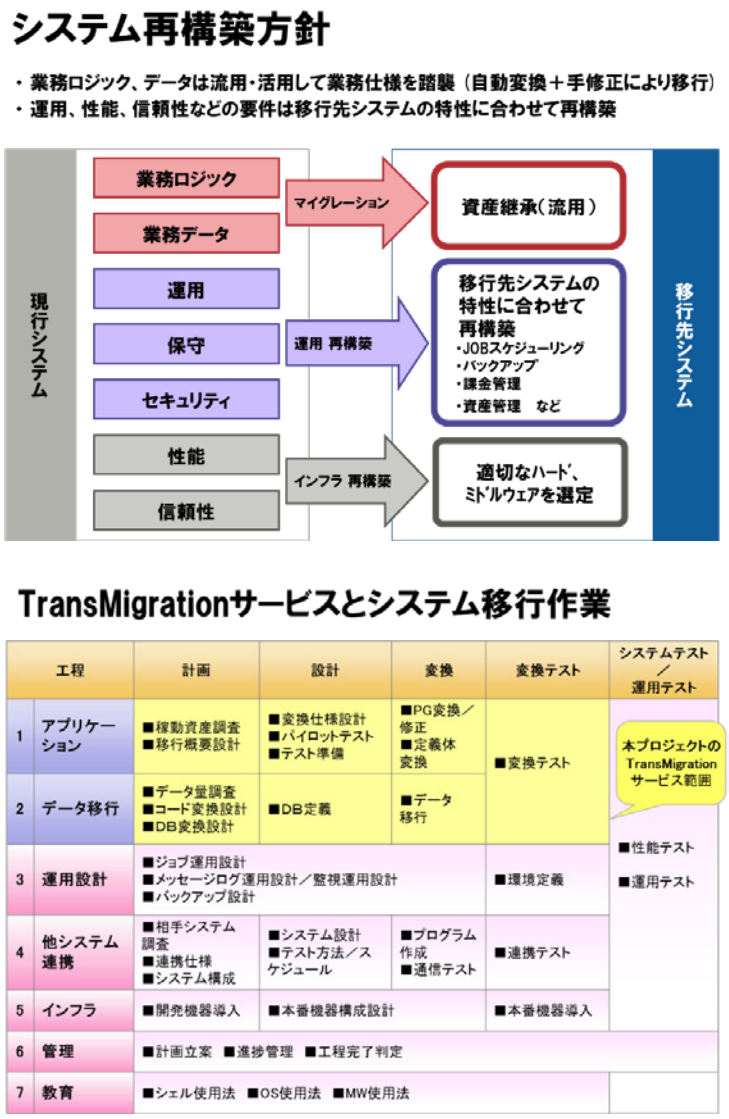


図 4.17 システム再構築方針と TransMigration サービス適用範囲

(3) 現行踏襲内容の明確化プロセス

TransMigrationサービスから現行踏襲の内容を明確化するプロセスとして移行概要設計と変換仕様設計について解説する。特徴は企画・診断フェーズで明らかにした機能や資産に対して、どの資産のどの機能を踏襲するのか。更に踏襲する方法をユーザとベンダで段階的に詳細化して合意することにある。

① 移行概要設計作業による現行踏襲範囲の確定

対象資産の過不足を調査、対象資産を確定し、プラットフォームの変更に伴う非互換、特性などを調査して、現行資産の「変える個所」「変えない個所」「変わってしまう個所」を抽出する。

② 変換仕様設計作業による現行踏襲内容と踏襲方法の確定

抽出した現行資産の「変える個所」「変わってしまう個所」について、変換仕様書に現行踏襲を実現するために何をどのように変更するのかを詳細に記述する。次にレビューにより変更内容と方法をユーザとベンダで合意する。変換仕様書の妥当性を検証する手段としてパイロット検証を実施する。

(4) ユーザ評価

① ポジティブ評価

- 現行資産を調査・分析して計画工程の移行概要設計、設計工程の変換仕様設計と段階的に現行踏襲内容を詳細化できた。現行踏襲する機能と範囲を段階的に整理することで理解が深まった。
- 20年間蓄積してきた貴重な業務ロジックを新システムに引き継ぐことができた。
- 全社ITコスト（保守、運用コスト）を半減できた。
- 企画段階でPOCを実施したことで新システムの動作が実感できた。さらに移行方式の作業課題、クライアント操作性の変更点、システム運用の変更点を事前に抽出できた。この結果をもとに現行踏襲機能と範囲の調整を行うことができた。

② ネガティブ評価

- リホストによる現行踏襲であっても業務の有識者を必要とした。特に試験手順書のレビューおよび障害の解析に業務の有識者を必要とした。
- 現行と新システムでアーキテクチャが異なる移行では各種機能の非互換が発生する。この非互換を吸収するにはOS／ミドルウェアに精通した技術者を必要とした。
- メインフレームの集中方式から分散サーバ方式への移行により運用要員が増加した。

■ 公開ホームページのご紹介

http://jp.fujitsu.com/solutions/sdas/migration

4.6　計画策定編「現行資産活用方針の検討（観点 C)」の事例
日本電気株式会社

取り組み背景

現行資産を活用するケースにおいて、活用方針の検討と活用範囲の検討を行い、システム化計画に適切に開発対象のボリュームを反映させるための事前作業の事例を紹介する。

本編との関連

計画策定編「3.4　現行資産活用方針の検討（観点 C)」と関連する。

現行資産の活用方針の検討

(1) 概要

システムを移行する際には現行資産の活用方針及び活用範囲を検討し、活用可能部分と開発対象を把握し、正しく見積ることが重要である。そのためには、技術リスクの除去、開発方式の決定、生産性の把握、といった観点で検証した上で移行方針を決定する必要がある。移行元（現行資産）のシステムで利用されている技術は構築当時の状況に依存し様々なケースが存在し、移行先のシステムで利用する技術も要件により様々な選択肢が考えられる。そのためシステム全体を自動変換するツールが存在することはほぼなく、手順を踏んで検討を進めて行くことになる。

(2) 取り組み内容

検討にあたっては、以下のステップで進めて行くものとする。

ステップ 1）移行元システムの現状把握
ステップ 2）移行先システムに求められる要件の把握
ステップ 3）移行方式・ルール・手順の決定
ステップ 4）移行作業の実行計画の策定

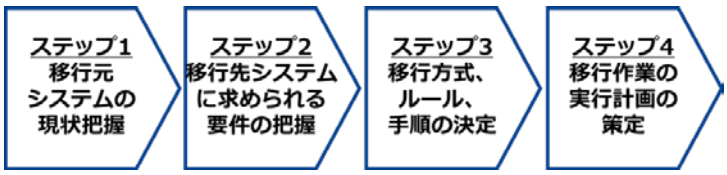


図 4.18 現行資産活用方針の検討のステップ

適用事例

業務システムをリライトした事例である。移行元のシステムはリッチインターネットアプリケーション（RIA）の技術を用いて作られており、その現行資産を活用しサーバサイド Java 技術を用いた Web システムに移行した。次のステップに従い移行作業を進めた。

ステップ 1）移行元システムの現状把握

現行資産を調査した。ドキュメント系資産として現行システムの設計であるアプリケーションアーキテクチャ、実装系資産としてプログラム、および現行資産を開発した際の開発ツールが保有している定義情報などを確認した。

特に移行先システムのアプリケーションアーキテクチャが変更になるため、現行との差異がリスクと考え、例として、GUI コンポーネントを用いて実現している画面の見映えや操作性、性能に影響を与えることになる処理方式の違い（処理がサーバサイドで行われるかクライアントサイドで行われるか）、同じく性能に影響を与える内部的な処理や格納するデータの構造や方式、などについて調査し現状把握を行った。

ステップ 2）移行先システムに求められる要件の把握

移行先システムのアプリケーションアーキテクチャの選定にあたり、ステップ 1 で挙げた移行元との差異に着目し調査を行った。これらは一例であり、他にも検討が必要なさまざまな観点が存在する。

- GUI コンポーネントを用いて実現している画面の見映えや操作性
  移行元で使用しているすべての GUI コンポーネントについて調査し、移行先に採用する技術で見映えや操作が実現可能か否か確認する。例えば、「矢印キー↑や↓でグリッド（表部品）の選択行を変更できない」、「グリッドの列幅を調節すると他の列幅に影響を与える」などがある。実現できないものがある場合は、新規開発の検討、要求仕様のお客様との調整、他の実現手段や実装技術への変更の検討、といった選択肢がある。

- 性能に影響を与える処理方式の違い
  処理方式の変更の観点から性能が要件を満たすことができるか予測を行う。例えば、入力値チェック処理の実行がクライアントサイドで行われていたものが、サーバサイドで行われることになる場合である。サーバサイドでのリソースの負荷の変化を確認し、移行前と同等の処理が可能か否かを確認し、実現できない場合は、システム構成や処理を見直す。

- 性能に影響を与える内部的な処理や格納するデータの構造や方式
  グリッドで多量のデータを表示すると画面の表示性能や操作性能が低下することがある。移行元システムでの処理は、表示する部分の画面コンポーネントのみを内部で生成していたため性能要件を満たしていたが、移行先の採用技術では「データの保有形式が理由でデータ量が多い」、「表示しない部分も含め画面コンポーネントすべてを生成する処理を行う」ため表示性能が顕著に劣化することが分かった。表示するデータをすべてではなく一部にする仕様変更をお客様と調整することや、部分的に別の技術の採用を検討した。

ステップ 3）移行方式・ルール・手順の決定

移行方式を現行資産の活用度合いにより、活用する、一部活用して自動化などで補う、新規作成、の 3 つに分類している。移行元を分析し、移行先を定め、作業対象領域ごとに、どの方式に当てはまるかを見極め、詳細を確認していく。

- 方式 A「現行資産の活用」
  移行先システムを考慮し、対象を特定し、現行の設計書やソースコードを活用する。

- 方式 B「現行資産の一部活用」
  移行先システムを考慮し、現行の設計書やソースコードを、選定もしくは開発した移行ツールを用いて移行先資産を生成する。

- 方式 C「新規作成」
  現行の設計書やソースコードを参考に、移行先資産を新規に作成する。

本事例では以下の方針とした。

表 4.4　作業対象ごとの移行方式の整理（例）

作業対象	作業内容	移行方式	参照元	作成物	補足
①プレゼンテーション層(画面)	画面、画面制御ロジックの作成	C	旧ソースコード	画面定義ファイル ソースコード *Action.java Control*.java	画面は新規作成する。旧ソースコードを解析して画面制御の部分と業務ロジックをそれぞれ新ソースに書き換える。
②プレゼンテーション層(ロジック)	設計書作成	B	旧仕様書 画面定義ファイル	機能設計書 （項目定義部分）	仕様書で足りない情報を画面定義ファイルを参照して補って、機能設計書を作成する。
	コード生成	B	機能設計書	ソースコード *Form.java *Action.java *config.xml	ツールを用いて、機能設計書からソースコード、定義ファイルを生成する。
③アプリケーション層/データベースアクセス層	アプリケーション層のソースコード移行	A	旧ソースコード Controlxxx.java xxxLogicData.java	ソースコード Control*.java *LogicData.java *Action.java	既存ソースコードを元にソースコードを作成する。
	SQL文移行	A	旧ソースコード xxxDao.Java	SQL定義ファイル *_sql_map.xml	既存ソースのSQL文を抽出、プロパティファイルに移す。 （動的SQLは書き換え要）

① プレゼンテーション層（画面）

方式C「新規作成」とした。画面、画面制御ロジックの作成を行うにあたり、画面は新規作成する。また、旧ソースコードを解析して画面制御の部分と業務ロジックをそれぞれ新ソースに書き換える。

② プレゼンテーション層（ロジック）

方式B「現行資産の一部活用」とした。移行先システムのアーキテクチャを考慮し、現行の設計書やソースコードを、選定もしくは開発した移行ツールにより移行先資産を生成する。

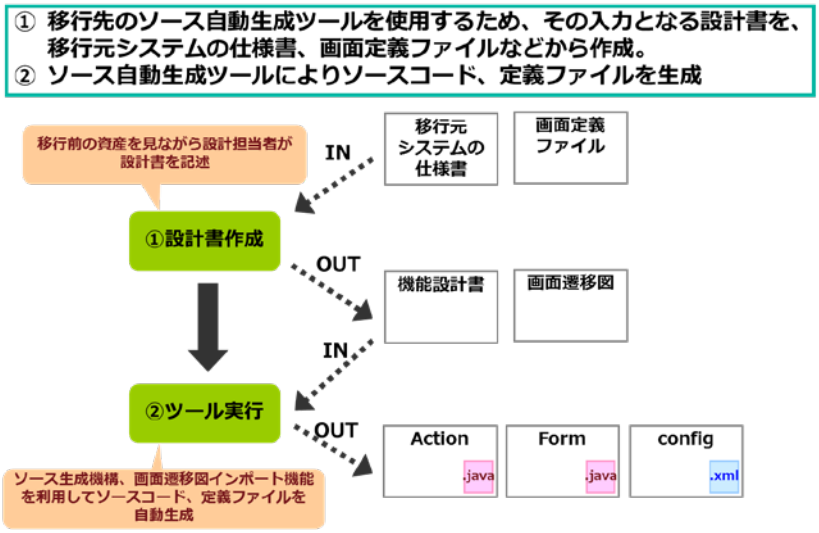


図 4.19　現行資産の一部活用の事例

③ アプリケーション層／データベースアクセス層

方式 A「現行資産の活用」とした。移行先システムのアーキテクチャを考慮し、対象を特定し、現行の設計書やソースコードを活用する。アプリケーション層のロジックに相当するソースコードとデータベースアクセス層の SQL は現行資産を活用できた。

ステップ 4）移行作業の実行計画の策定

ステップ 3 までに決定した方針に従い、現行資産の種類・規模から開発ボリュームを見積り、プロジェクトの開発標準として定め、移行作業の実行計画を策定する。

以下の図は、移行先において現行資産を活用する部分をまとめたドキュメントの一部を記載したものである。

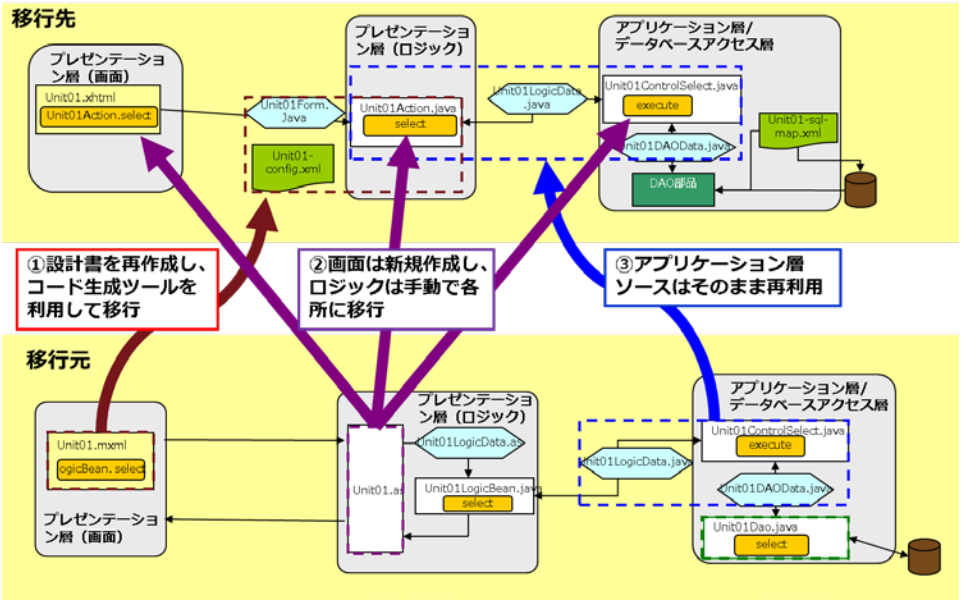


図 4.20 移行元及び移行先において現行資産を活用する部分を示した概念図

4.7　計画策定編「現行業務知識不足への対応（観点 D)」の事例
株式会社 ソフトロード

取り組み背景

システム再構築を検討する際に現行システムに関する様々な情報が必要になるが、多くの場合そのシステムが最初に構築されてから 10 年以上の長い年月が経っているので、構築当初には揃っていた設計ドキュメントなどが、その後のプログラム改修の際にドキュメント更新だけ行われず古い内容のままであるとか、ドキュメントが紛失してしまっていることも多い。当初の設計に関与した人も異動や退職などで居なくなったり、詳細な記憶が失われたりしている。

再構築の検討の際には、現行プログラム資産やドキュメント資産の棚卸しをして、現行システムに関する情報を確認することが多いが、一度失われた情報を再現するのはとても困難なことだ。

ソフトロードでは、このような「情報不足」を補完するために、「調査」「分析」「推測」「移行方式設計」の作業を行っている。現行システム情報不足への対応の具体的な工程と、幾つかの事例を紹介する。

本編との関連

計画策定編「3.5　現行業務知識不足への対応（観点 D)」と関連する。

現行システム情報不足への対応

(1) 不足する情報の種類

失われたドキュメントに記載されていた情報、もともと記述がなく担当者の記憶にしかない情報、偶然に実現されていた設計情報などは、現行システムのソースプログラムやシステム設定情報などの分析では把握できない情報であるが、モダナイゼーションの開発やシステム構築には必要な情報もある。以下は代表的な「不足する情報」である。

① 正しいソースコードの確認

プログラムのソースコードが様々なライブラリーに分散しているために、どれが最新の正しいソースコードかを判定するのに手間取ることが意外に多い。特に改修頻度の低いシステムほど、ソース・ライブラリー管理がおろそかである。また、プログラムの改修作業にともなって、それ改修前、改修中、そして改修後のバージョンが共存することで複数ライブラリーに拡散したりすることも重複資産発生の原因である。

② システム概要・業務概要・操作に関する情報

再構築のために、現行システムのソースコードや各種システム設定情報を把握するために、システム全体の概要の理解、業務全体の概要の理解が重要である。オンライン処理の主な画面操作なども確認する場合がある。

③ 処理量、処理サイクル、処理スケジュールに関する情報

オンライン処理のトランザクション量、ピーク時の増加率、バッチ処理の日次・週次・月次・四半期・年次などの処理サイクル、定常バッチの起動時刻や終了予定時刻、オンライン処理の起動時刻・終了時刻などは、新システムの要件として把握しておくことが必要である。

④ 排他制御処理に関する情報

システムによっては排他制御処理が必要なものがあるが、アプリケーションプログラムとして明示的に排他制御をするのではなく、OS などの基盤ソフトの仕組みを利用して結果的に排他制御が実現されているものがある。こうした場合、ソースコードにも開発ドキュメントにもユーザ企業担当者の記憶にも、そのことが明示されていないことがある。

通常の機能テストでは、排他制御が働くようなタイミング依存のテスト環境が作れないので再現性も困難である。

⑤ 検証データに関する情報

現行システムの実際の本番データには、プログラムの例外処理になるようなデータの多様性が含まれないことが多い。そのため、再構築後の検証データとしてプログラムのテスト網羅性（カバレッジ）が低くなってしまう傾向がある。

⑥ 使用されていない処理に関する情報

プログラムに盛り込まれた機能の中には、その後業務が変更された現在は使用してない機能がそのまま残っている場合が多い。しかし、新システムを再構築する際に使用されていないロジックを削除するにも、どの部分が使用していないか、誰もわからない場合も多い。

(2) 不足情報を補完するための方法と作業

① ヒアリング

再構築を必要としているシステムには、作られてから 30 年以上も経っていて、開発当時の関係者は一人も残っていないことが多い。しかしながら現行システムを保守している過程で現在の担当者が把握している情報の中にも、全体像を理解するため

の情報の断片が残っている場合もある。
担当者にヒアリングするのは、システム概要の情報、主要なオンライン処理の画面操作、などである。これとは別に入手しているプログラムのソース分析情報などと組み合わせて不足していた情報を推測していく。

② 別なドキュメントからの情報入手
開発ドキュメントが失われていたり、更新されずに古い内容のままだったりして、必要な情報が不足する場合に、他のドキュメントによって一部の情報が補完されることがある。
エンドユーザ向けの操作マニュアルなどは、開発ドキュメントに比べて最新内容に更新されていることが多い。

③ 操作トレーニング
画面操作などの情報は、机上のヒアリングだけでなく、実際にテスト環境での操作トレーニングによる方が返って短時間で要点が把握されることもある。

④ ソース分析、稼働ログ分析からの推測
現行システムのプログラム・ソースコードを分析することで、重複資産、不足資産などが判明する。これには、主にプログラムや JOB の呼び出し（連携）情報による分析と、稼働ログによる稼働情報による分析、さらにそれら 2 つの分析結果を総合したもの、などが使われることが多い。ただし、オープン系のシステムでは OS が自動的に稼働ログを取らないので、過去の稼働情報が無いシステムが一般的であり、この場合には連携情報のみで分析する。派生的な情報として、画面遷移・階層の情報、JOB のネットワーク情報、プログラムとデータの CRUD 情報などが抽出される場合もある。

⑤ 現新比較テストによる同一性検証
開発ドキュメントやソース分析やヒアリングでは入手できない情報もあるが、現行システムの処理結果と再構築された新システムの処理結果を「現新同一性検証」することが可能である。これによってあるべき処理結果から逆算してロジックや設定情報を推測することができる場合もある。

⑥ 現行システムの設計ドキュメントの再生
再構築の検討段階で、現行ソースコードから各種の開発ドキュメントを再生することもできる。これによってある程度の概要を視覚的に把握できる。

適用事例

(1) 案件事例：システム間連携情報の不足

外部企業とのシステム間連携が複雑に存在するようなシステムは、業務継承型の移行でも難易度が高い部類である。これは、連携のインターフェース情報だけでなく、内部のデータ条件でプログラム連携の挙動が変化するためだ。これらの情報を詳細に示す開発ドキュメントが失われて存在しなかったので、大きな工数をかけて試行錯誤での確認が必要であった。図 4.21 は、このような複雑なシステム間連携が存在したシステムの全体構成図である。

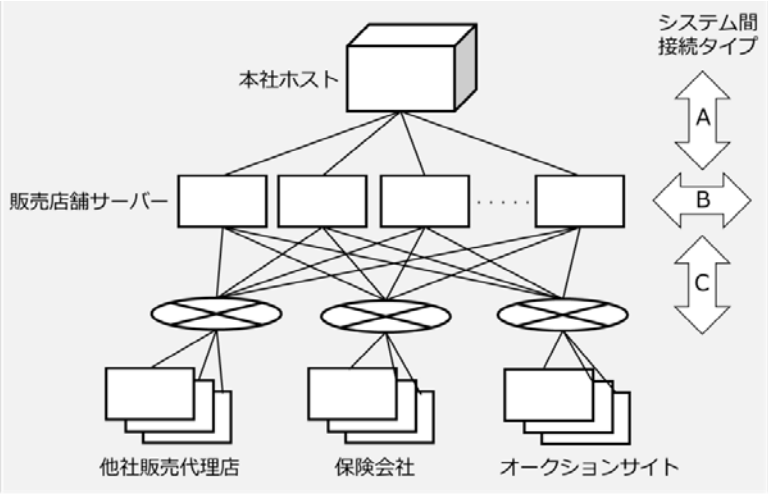


図 4.21 複雑なシステム間連携の移行事例

標準的なマイグレーションとしては不適な案件であるが、最大限このような難易度まで「情報不足」でも解決できた事例でもある。

システム間連携機能は、初期構築時は詳細な通信プロトコルの規定や受け渡されるデータ値についての条件記述が必ず存在しており、それに従ってプログラムを行い、テストを実施した。ところが長く稼働している間に開発当初には存在していた一連の設計ドキュメントが失われたり、当初の設計者は既に退職していたりするため、どんなデータが流れてくるかを設計書ではなく既存プログラムのロジックから逆算して想定して、そのシナリオパターンに従ってテストを実施した。具体的にはプログラム内の主要な分岐パスと分岐条件を解析することで主要な流れを作っているインプットデータの条件を特定することが可能になった。このようなプログラム解析作業には分岐パス分析をするソフトウェアツールを利用しながらも丹念な人手による解析作業で初めてデータ条件が見極められるものである。

(2) 案件事例：ホスト系システムのオープン化での移行対象範囲の判断ミス

ホスト系／オープン系には無関係なことだが、ユーザが指定した移行対象プログラムの範囲に大きな判断ミスが発生した。このシステム再構築では、全体を幾つかの領域に分割して、それぞれの再構築を別々なベンダが担当していた。そこで、両側にまたがる共通なサブプログラムなどが、いずれの再構築プロジェクトでも対象として含まれるべきところを誤って一方のプロジェクトに寄せてしまったため、他の再構築プロジェクトでは本来移行対象として含まれるべきところが欠落してしまい、開発工数の算出が過少見積りされてしまった。

刷新移行プロジェクトの全体は以下のような開発方針で進めることが決まった。

- 経理　：　自社開発　→　ERP
- 商品管理　：　自社開発　→　再設計スクラッチ開発
- 調達　：　自社開発　→　ERP
- 物流　：　自社開発　→　リフォームによる移行
- 物流基礎データ（勘定系）・・・70%の業務を継承、30%改修
- 物流情報（情報系）・・・・・・40%の機能を削減して、BIツール化
- 物流計画（計画系）・・・・・・80%の出力を削減して、BIツール化

多くのサブシステムにまたがった処理やデータが多数存在していたのだが、削減の判断をするのに、サブシステム横断的な分析に基づく判断が行われず、誤った削減判断をユーザ企業がしてしまった。

なおこのような間違いは、開発見積りに先立ってソース棚卸で過不足のない範囲を特定することで防止できる。

＜実施結果＞

再構築が必要になるシステムの多くは、本来あるべきシステムの設計情報が失われている。これは再構築だけでなく、日常のプログラム・メンテナンス作業にも支障をきたしていることが多い。

失われた情報は再生できないものが多いが、発注者であるユーザ企業との協力で最大限に不足情報を補完して再構築の品質を高める努力をすることで、再構築開発の効率化の犠牲になりやすい「保守性劣化」を少しでも改善することもできるので、多くの再構築プロジェクトで高い顧客満足を実現している。

4.8 計画策定編「品質保証の検討（観点 E)」の事例 株式会社 NTT データ

取り組み背景

再構築の品質保証では「業務継続性の担保」が求められる。具体的には、再構築手法ごとの特性を加味した更改計画を策定することが重要なポイントとなる。例えば、メインフレームからオープンへリホストする場合、業務アプリケーション以外の変更による影響が大きく、試験工程で業務処理エラーやデータ不一致が多発し、故障解析・対応に多大な時間と稼働がかかる。その結果、テストが大きく遅延し、納品に関する合意調整も難航し、大問題化することとなる。こうしたことが起きないように、再構築手法を考慮したテスト計画を立てて実施することが重要である。

以下にリホスト案件でテストを通じて品質を積み上げていった弊社の取り組み事例を紹介する。

本編との関連

計画策定編「3.6　品質保証の検討（観点 E)」と関連する。

品質保証の取り組み

(1) 概要

再構築手法の特性を加味したテストを実施して品質を確認した状態を、業務継続性が担保できる品質レベルであり、新システムのリリース時の品質目標として設定した。また、テストによる解消が困難なリスクに対しては、あらかじめ対応体制を準備することとした。

(2) 取り組み内容

① 再構築手法の特性を踏まえたテストの実施

再構築手法（リホスト）の特性を踏まえて図 4.22 の通り、テストを計画し、実施した。

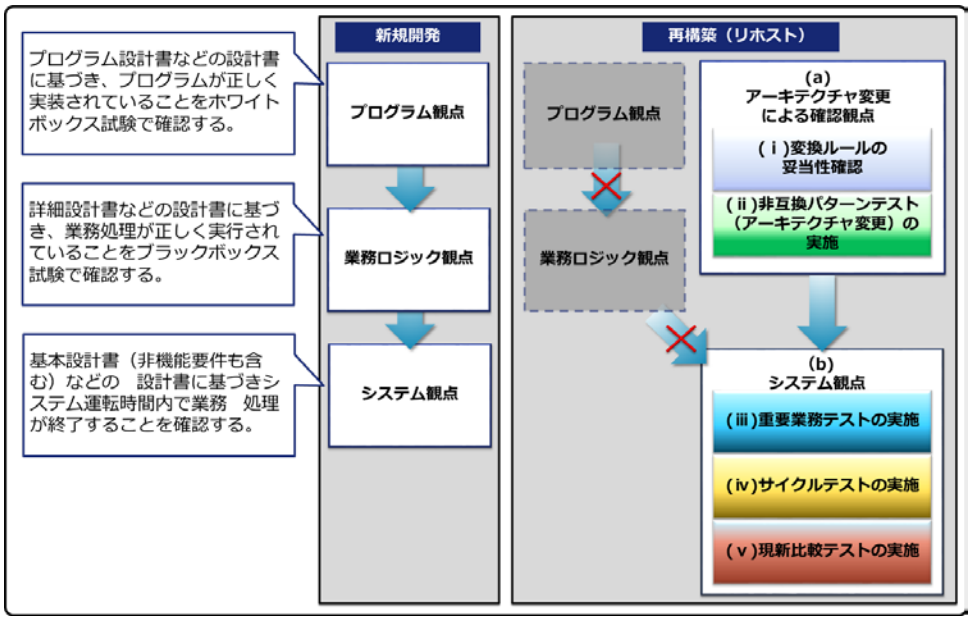


図 4.22　新規開発とリホストの業務アプリケーション開発作業イメージ

② テストでは解消しきれない残存リスクへの対応
サービス開始後においても、必要な時期に適切に要員を配置してインシデント発生に備えた。

(3) 効果
リリース時の品質向上、残存リスクに対する対応の明確化

適用事例

＜プロジェクト概要＞

本事例は、メインフレームからオープンへのリホスト案件であり、お客様の業務特性から、このシステムに障害が起きた場合の社会的影響は甚大となるため、高い品質目標が求められた。プロジェクト概要は以下の通り。

システム区分　：　バッチシステム
言語区分　　：　オープン COBOL
システム規模　：　約 500KStep
基盤　　：　メインフレーム

＜実施内容＞

① 再構築手法の特性を踏まえたテストの実施

本事例では、テストによる品質の積み上げにて現行業務継続性を担保するという品質目標を設定し、達成に向けて取り組んだ。具体的には、再構築後のシステムにおいてこれまでと同様に業務を継続実施できるレベルが何かを分解し、定量的な基準に置き換えていった。

例えば、日次の業務処理が正常に実行されることという目標に対し、ある特定の1日についてサイクルテストを行い、実行エラー0件もしくはエラー原因が特定されており解消見込みであることを完了基準とした。目標到達までに必要なテストと完了基準を策定し、それらが充足した状態をカットオーバークライテリアとして設定した。なお、テスト内容についてはリホストの特性を踏まえたものとすることで、可能な限りリホスト特有のインシデント（実行時非互換など）を排除し、品質の向上を図った。

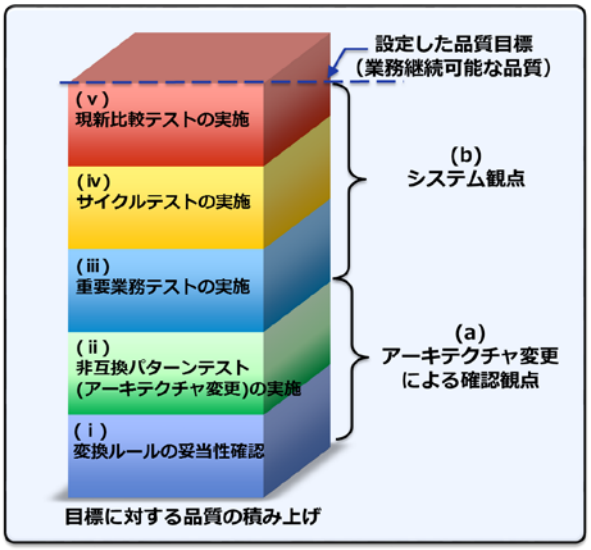


図 4.23 リホストのテスト計画

（i）変換ルールの妥当性確認

リホストの場合、基本的には業務アプリケーション（今回はCOBOL）をそのまま流用するが、COBOLはベンダ依存している記法があるため非互換対応が必要となった。対応は変換ツールにて機械的に実施することとした。そのため、ツールの変換ルールの妥当性に関する確認テストを実施した。なお、言語非互換に対する機械的変換しか実施していないため、プログラム観点、業務ロジック観点のテストについては実施不要と判断し、実施対象アクティビティから除外した。

（ii）非互換パターンテスト（アーキテクチャ変更）の実施

リホストの場合、基本的に業務アプリケーションに関するテストは実施不要であるが基盤が変更となっているため、非互換パターンテストを実施する必要がある。非互換パターンテストの例として今回の事例では以下のような取り組みを実施した。

- 方式基盤の変更の例

  現行システムはメインフレーム（z/OS）のコールドスタンバイで構成されているが、再構築後のシステムはオープン機（AIX）と PowerHA によるホットスタンバイ構成に変わる。

  【対応】

  システムテストで非機能要件の確認を実施、障害テストについても実施した。

- 文字コード変更の例

  現行システム内のファイルなどデータの移行にあたり、文字コードなどの変換が必要。

  【対応】

  移行ツール製造・移行機能・基盤テストにてコード変換結果確認を実施した。

- DB 方式変更に配慮した例

  現行のデータ定義に基づき正しく格納されていないデータが存在したため、データクレンジングが必要。

  【対応】

  実データにて調査・確認を実施した。

（iii）重要業務テストの実施

業務継続性を担保する上で、本システムの根幹業務についてはリリース後の重大トラブル発生を未然に防ぐ必要があった。そのため、根幹業務については複数バリエーションで正常に業務処理ができることを確認すると共に、業務処理エラーが設計通りに検出できることも確認するため、正常系網羅テストとエラー分岐検証テストを実施した。

（iv）サイクルテストの実施

バッチジョブが現行の運用設計と同様に設計できているか確認する必要がある。特定日のサイクルテストを行うことで、システムとして一連の処理が正しく実施されることを検証した。

（v）現新比較テストの実施

実行時非互換による処理不整合を検出するため、本番データを利用した現新比較テストを実施することで、データバリエーションの拡大を図った。現新比較テスト結果を突合するツールを作成し、自動で照合結果を確認し取り纏めることで、テストの効率化を実現するとともに、要員は原因解析作業に注力させてインシデントの早期解消を図った。

② テストでは解消しきれない残存リスクへの対応

サイクルテストや現新比較テストでも実施できなかったデータバリエーションについては、リリース後も残存リスクとして残るため、あらかじめ対応要員を準備することでインシデントの発生に備えることとした。具体的には、年末年始などの連続休暇明け稼働日や年次バッチの初回実行予定日について、保守要員の追加配置を計画として盛り込んだ。

＜実施結果＞

それぞれの実施内容に対して以下のような結果が得られた。

① あらかじめ再構築手法の特性を踏まえたテストを計画し追加実施したことで、リリース後も重大インシデントは発生することなく安定した運用を実現することができた。

② 保守要員の追加配置を計画段階から組み込むことで、リリース後の保守体制を適切に構築するとともにインシデント対応に備え有識者を確保することができた。

4.9　計画策定編「意思決定プロセスの策定（観点 F)」の事例
東京海上日動システムズ株式会社

取り組み背景

システム再構築プロジェクトに限ったことではないが、「意思決定プロセス」が不明確な状態でプロジェクトを進めることにより、プロジェクト内で判明した課題（プロジェクトそのものに大きく影響するような課題）に対して、判断の遅れはもとより、“スケジュール遅延”、“コスト超過”、“品質の劣化“などのリスクの発生確率が高まる恐れがある。

このため、「意思決定プロセス」を明確化し、課題のレベルに応じた責任者が判断できる体制を整えることで、リスク発生の軽減を図った取組みを紹介する。

本編との関連

計画策定編「3.7　意思決定プロセスの策定（観点 F)」と関連する。

意思決定プロセスの明確化

(1) 概要

本プロジェクトにおいても、他のプロジェクト同様の開発体制を構築。加えて、本プロジェクトにおけるリスクを鑑み、各組織（部署）の責任者が一堂に会する場を設けることで、更なる「意思決定プロセス」の明確化を図ることとした。

(2) 取り組み内容

取り組み内容としては、以下の 2 点がある。

- 各組織（部署）の責任者が一堂に会する場（＝ステアリングコミッティ）を設置し、体制図に明記する。
- 責任者が自らプロジェクトの進捗状況を報告する場とし、発生した課題についてもその場で議論し、対応方針ならびに役割分担を明確化する。

(3) 効果

“ステアリングコミッティ設置”の効果としては、以下の 3 点がある。

- 責任者が自らプロジェクトの進捗状況を報告することを通じて、関係者の意思統一、プロジェクトの求心力を高めると同時に、各組織の責任を明確化することが出来る。
- プロジェクトメンバー全員の強力な動機付けができる。
- 意思決定がスムーズに行われることで、プロジェクトの滞留を抑止できる。

適用事例

次期経理システム再構築プロジェクトにおいて、ステアリングコミッティを設置した事例を紹介する。

＜プロジェクトの概要＞

- 経理システムの老朽化に伴うブラックボックス化、ならびに、四半期決算や国際財務報告基準（IFRS）などの環境変化に柔軟な対応を行うことを目的に、再構築プロジェクトを立ち上げた。
- ERP パッケージ導入ならびに BPR を行わないことを宣言し、早期本番稼働を最優先課題とした。
- 要求（要件）の実現可否を分析（Fit&Gap 分析）するフィージビリティ期間を確保した。

＜実施内容＞

ユーザ部門、IT 部門、ベンダの 3 者の部門長によるステアリングコミッティを設置。（図 4.24 ステアリングコミッティ設置のイメージ図）

プロジェクト進捗報告並びに、大きな課題などをその場で論議し、方向性を確定。以下に具体的な例を挙げる。

- プロジェクトの各工程（基本設計策定、要件定義、開発およびテスト）の完了基準にステアリングコミッティの承認を条件とした。
- 一部、要件定義不足判明における対策の議論を行い、対応方針とスケジュールを合意。
- 導入パッケージの一部機能のファーストユーザリスクに関する議論を行い、本プロジェクトにおける効率化およびメンテナンシビリティにおける効果を優先し、導入を決定　など

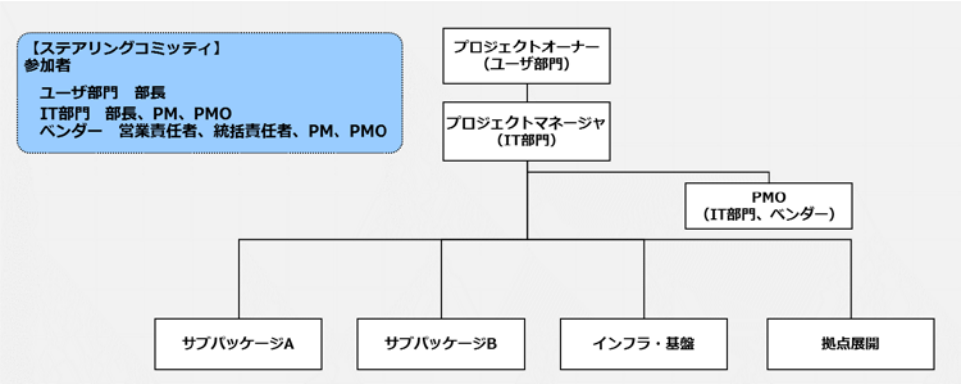


図 4.24 ステアリングコミッティ設置のイメージ図

＜実施結果＞

実施した結果、以下のような効果があった。

- 意思決定スピードが向上した。
- 組織的な対応力が求められることにより、プロジェクト推進力が強化された。
- 品質を確保しつつ、ほぼ当初計画通りの期限およびコストにてサービスインできた。

4.10　計画策定編「再構築の計画と見積り（観点 H）」の事例
株式会社 日立製作所

取り組み背景

システム再構築においては、リスクを事前に回避するため、現行システム・資産を分析し全体を把握した上で、必要となる作業を段階的に詳細化・具体化しながら推進していくこととなる。

そのため、企画段階で見積りの精度を向上させることは難しく、プロジェクトの進行に伴うフェーズ毎に、段階的に詳細化・精緻化し、見積り精度を向上させていくことが重要である。

日立製作所では、多段階契約での見積りを含め、再構築におけるフェーズ毎の作業を明確化し、すべてのプロジェクトに適用している。

本事例では、再構築作業の流れの中で見積り精度を向上させるためにも必要となる、移行性分析と事前検証（パイロット移行検証）について紹介する。

（1）再構築作業の流れ

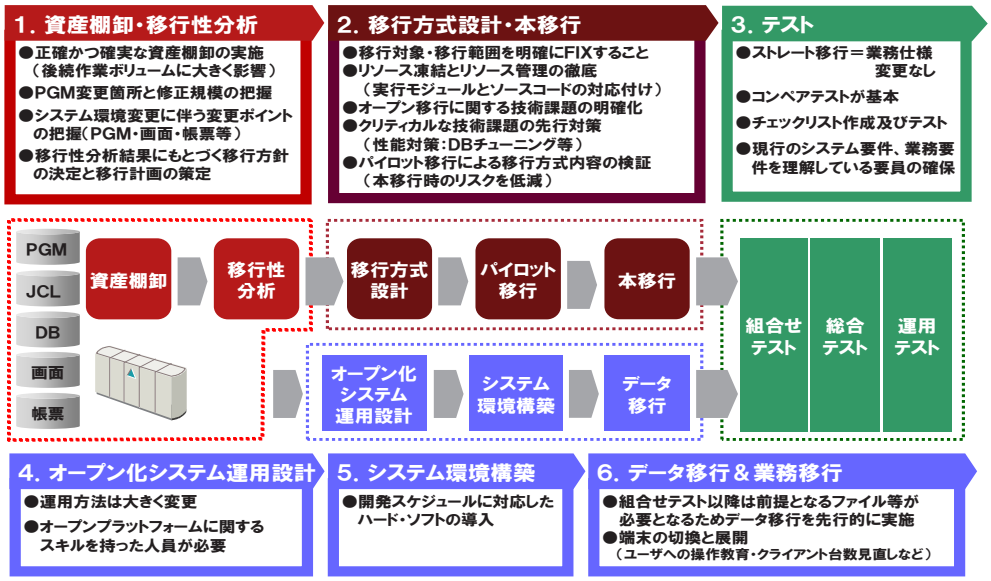


図 4.25 再構築作業の流れ

（2）再構築のスケジュール例

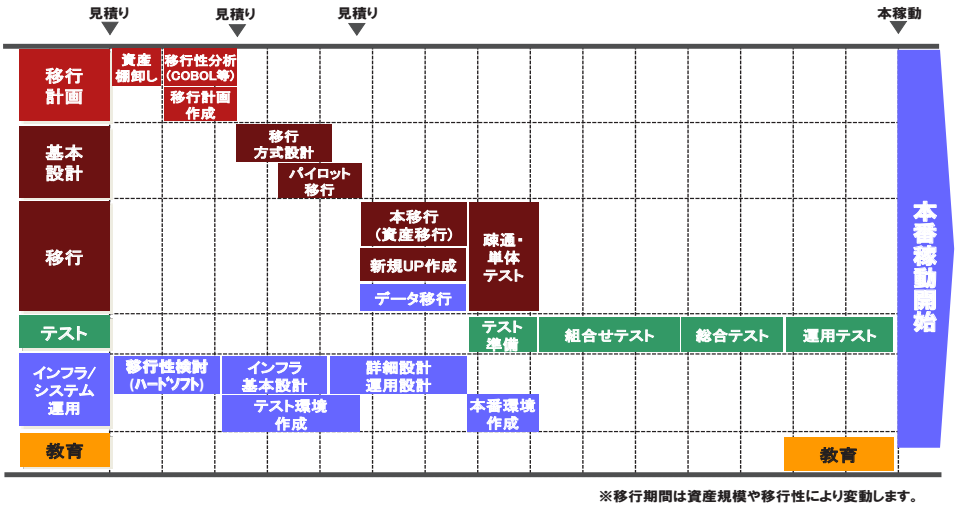


図 4.26 再構築のスケジュール例

本編との関連

計画策定編「3.9　再構築の計画と見積り（観点 H)」と関連する。

移行性分析

(1) 概要

移行性分析は資産棚卸で確定させた移行対象資産を、新システムで稼動させる際の技術的な課題・リスクを抽出し、対策案を検討した上で、移行資産すべての修正内容・変換仕様および、新規に開発が必要なプログラム機能を明らかにする。本作業によりシステム再構築のリスクの低減を図ることができる。

この移行性分析についての具体的な取り組みは以下の通りである。

(2) 取り組み内容

取り組み内容としては、以下の 2 点がある。

① 移行観点情報を元に、移行修正箇所・規模を分析調査

② 移行性分析結果より、早い段階でのリスク回避、品質確保が可能

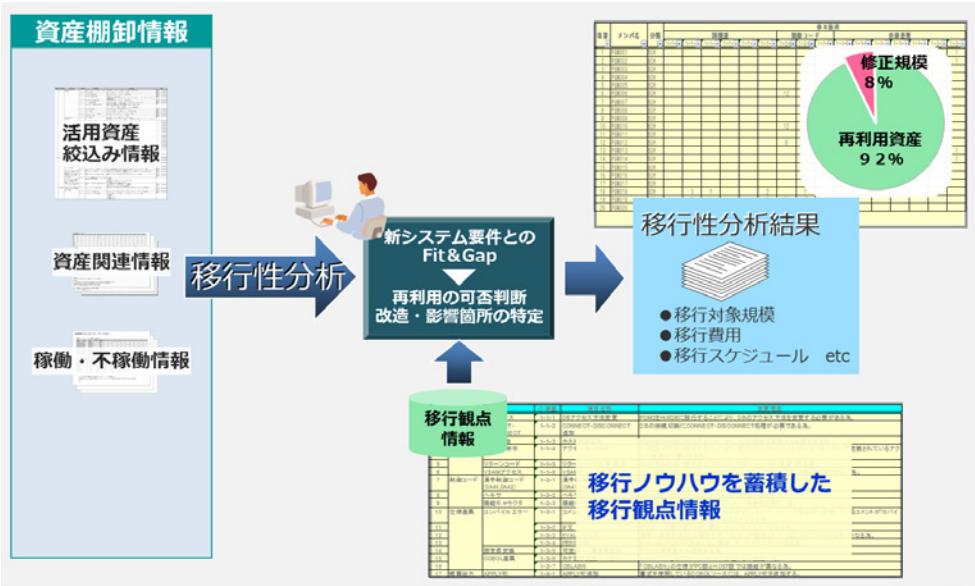


図 4.27 移行性分析のイメージ図

(3) 効果

移行性分析の適用によって得られる効果としては、以下の 2 点がある。

① 本番移行作業を実施する前に、移行対象資産全体に対するオープン化の移行性を分析

② 移行性の可否判断により、後続プロジェクトのリスクを事前に回避

事前検証（パイロット移行検証）

(1) 概要

移行性分析結果を元に、実機でのパイロット移行を実施し、移行結果の評価を行う。その結果を移行方式設計書および移行ツールなどに反映することで、次工程である本移行作業において、リスクの低減を図ることができ、より効率的でスムーズなシステム移行が実現できる。

また、本事前検証の結果を見積りに反映することで、見積り精度の向上を図ることが可能である。

この事前検証についての具体的な取り組みは以下の通りである。

(2) 取り組み内容

取り組み内容としては、以下の 2 点である。

① 移行分析の結果を元に移行観点を網羅する資産を選出
特に、出来るだけ多くのテストケースを実施することが重要。

② パイロット資産の移行と現行との比較検証を実施し、移行設計書と移行ツールを必要に応じて修正する

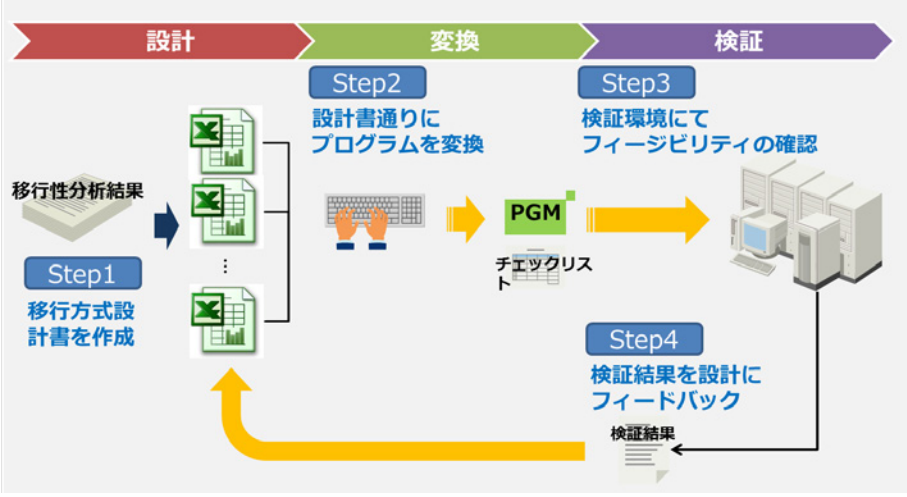


図 4.28 パイロット検証の作業イメージ

(3) 効果

事前検証の適用によって得られる効果としては、以下の二点がある。

① 本移行作業を実施する前に、移行方式設計内容の確認、移行ツールの妥当性を事前に検証

② 後続プロジェクトのリスクを事前に回避

適用事例

資産調査、移行性分析を重点的に実施した事例である。

＜お客様の課題＞

お客様の課題は以下の 3 点である。

- 既存メインフレームシステムのランニングコストが高い
  （システム改修の際のメンテナンスに手間がかかる）
- 既存システムリソースはノウハウの固まりなので捨てたくない
- インフラ基盤、システム運用の標準化

＜実施内容＞

これらの課題に対して、現行資産調査・移行性分析によりツール適用率及び手修正比率を事前に把握するとともに、パイロット移行による移行方式設計内容の確認および問題点の早期抽出を実施した。

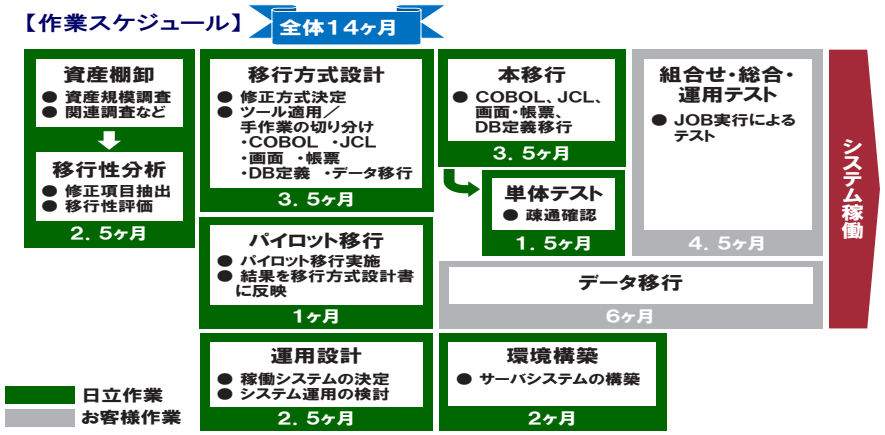


図 4.29 作業スケジュール

＜実施結果＞

実施した結果、以下のような効果があった。

- よりリスクを低減した形でオープン環境への移行を実現
- 早期の新システム稼働により、ランニングコスト削減が早期に実現
- 顧客インフラ基盤の標準化を実現
- システム運用の標準化による、システムメンテナンスの効率化を実現

4.11　計画策定編「再構築の計画と見積り（観点 H)」の事例
JFE システムズ株式会社

取り組み背景

システム再構築において、プロジェクト実施計画の策定はプロジェクトの開発工程を着実に遂行するためにも重要な作業プロセスの一つである。再構築の計画策定は、通常の開発推進方法に加え再構築特有のリスクを取込んだもので計画そのものについては大きな変化はなく愚直に開発工程を推進していくことが重要と考えている。

今回紹介する当社の取り組みは、2012 年に刷新した開発工程標準の定着活動とプロジェクトマネージャー（以降は PM）や開発メンバ向けの支援活動について記載する。

この取り組みの背景には、ビジネス環境における変化のスピードが想定以上に早くなり、お客様の経営課題をより正確にかつ早く、しかもビジネス環境に応じて機敏に「変化する IT を設計・構築・運営」していかなければ生き残れないと考えたからである。

本編との関連

計画策定編「3.9　再構築の計画と見積り（観点 H)」と関連する。

開発工程標準の定着と支援システム

(1) 概要

当社は、2012 年にこれまであった開発標準を刷新し新開発工程標準を策定した。この開発工程標準は、情報システムの開発に携わる全てのライフサイクルを定義している。一般的な開発プロセスだけでなく現状を評価するプロセスや、業務のあるべき姿を要求として定義するプロセスも標準として加えた点が特徴にあげられる。開発工程標準は「開発工程」、「実施要領」、「作業ガイド」で構成され規定されている。

この開発工程標準は、当社において既に適用を開始している。社内では標準を有効に活用するために社内教育体系へ開発工程標準の教育を盛り込んだ。また、利用局面を想定し開発者を支援するナビゲーションシステムを構築し特定部門に公開している。以降は、この取り組みについての具体的な内容を以下に記す。

(2) 取り組み内容

取り組み内容としては、以下の 3 点がある。

① 開発工程標準、プロジェクト管理知識教育の教育体系への取込み、定着化
② 開発プロセスナビゲーションツール
③ 適用評価

① 開発工程標準、プロジェクト管理知識教育の教育体系への取込み、定着化

開発工程標準は、普及・定着させるために社内教育体系にカリキュラムとして整備されている。

コースは、若年層向けの基礎編。上流工程作業者向けの上流編のそれぞれが準備されており、それぞれ年２回開催されている。この教育により標準の目的、コンセプトから実際の各プロセスの知識を深め層別に定着を図っている。この他にも、一般的なプロジェクトマネージメント体系教育も準備されており、講座の中で過去の開発プロジェクトにおいて対処した事例や課題なども織り込みながら開発工程標準と一般知識・知見を融合した教育プログラムをもとに定着を促進させている。

② 開発プロセスナビゲーションツール

整備された標準も利用されなければ無用の長物になってしまう。プロセス毎の工程標準・実施要領・ガイド、成果物をナビゲートし支援するシステムが「開発プロセスナビゲーションツール」である。当システムは、実際のシステムを構築しトップ画面に開発工程を時系列で表示している。プロセスや工程を参照しながらプロジェクトで必要な標準やガイド、成果物を確認することができるためプロセスに沿った開発の推進において開発者を支援が可能となる。

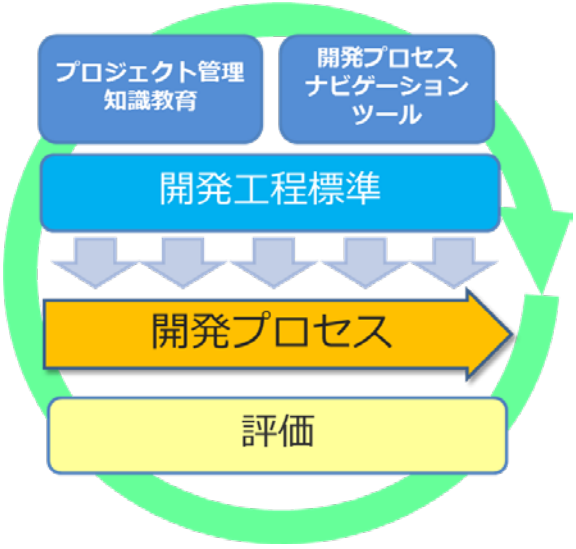


図 4.30 開発プロセス支援イメージ図

③ 適用評価

利用部署のプロジェクトマネージャーに対して１回／年、難易度、有用性、お客様との要件合意状況などをアンケートし定着度、理解度、合意性、有効性の評価

を実施中である。ここでの評価は、開発工程標準や開発プロセスナビゲーションツールなどの各種改善に繋げさらなる発展を推進中である。

(3) 効果

今回の開発工程標準の定着（特に業務要件整理）よって得られる効果として以下の 2 点がある。

- 要件定義プロセスで要求要件を一覧に整理することでお客様および当社メンバそれぞれに「気付き」が得られ要件提示漏れがなくなり、見積り漏れや下工程での課題発覚などが減少する。また、プロジェクト推進中でのスコープ調整の合意形成が図りやすくなる。
- 開発着手時に開発プロセス（開発手順）、プロセス別の成果物がテンプレートとして作成されており、プロジェクト経験の浅い SE でも迷うことなく開発が推進でき、プロジェクト品質レベルの均一化が図れた。

適用事例

当社の開発工程標準を利用したシステム再構築した事例を紹介する。

＜課題＞

特にシステム再構築においては、お客様から提示された業務要件やドキュメントやソースコードが全てを網羅出来ているわけではない。特に「現行踏襲内容」は、お客様の認識とシステム会社の認識が不明瞭なまま工程を推進することが多く、後工程でお客様の認識と構築したシステムのギャップが大きくプロジェクトの納期遅延や予算超過が発生していた。

＜実施内容＞

開発工程標準の業務要件プロセスにおいて、業務要件とシステム要件の結びつきを可視化することでプロジェクト全体の問題発生リスクをヘッジする。以下が実施内容である。

- 開発標準の重要成果物である「業務要件・要求一覧表」の作成時に、現行踏襲する業務要件とシステム要件を明確化する。現行踏襲は、ドキュメントとソースコードから類推する。
- 要件は、非機能要件も現状を調査し記載する。
- 業務要求に対して要求の理由、解決方法、要求する機能を検討・記述し、紐付く形でシステム要求事項を記載し要求事項とシステム対応を可視化する。(要求とシステムの結びつき)

5.1.2	案件名（又は、業務名）		作成日		頁
業務要求・要件一覧			作成者		
			更新日		
			更新者		

業務層								システム層	
★業務要求	★要求理由	★解決方向	No.	区分	★業務要件	★分類	★業務機能	★システム要求	★システム要件
①管理レベル向上のための機能構築									
1_案件明細のお客様への開示	お客様が発注時に、発注情報と明細情報を比較し注文書を生成するために、最新の案件明細情報を取得したい。	1.お客様毎に保有する項目が相違するため、画面で参照できる全情報を参照可能とする。 2.お客様単位に一括で取得できるダウンロード機能を準備する。			・発注情報の一覧取得機能の新設		検索機能の構築	①お客様ごとに取得項目を自由選択可 ②一覧のダウンロード	検索/ダウンロード機能の構築
2_品目管理の改善	注文情報に対して品目情報での集計を行いたい。	注文情報に品目情報の付与を行う。			注文情報に対して品目情報での集約を可能とする。		集計＆ダウンロード機能の構築	受注実績情報に品目項目を追加	テーブルに項目追加
[★]…業務要件定義プロセスにおいて必須整理事項。									

図 4.31 業務要件・要求一覧表のイメージ

＜実施結果＞

一覧表で業務要件とシステム要件が結びついており、ステークホルダ全体で要件の共通認識がはかれたことで特に以下 3 点について有効であった。

- プロジェクトスコープの共有化ができた。
- 後工程での認識のブレがなくなり、プロジェクトが円滑に推進できたため、納期遅延や予算超過などの発生のリスクを抑止できた。
- 要件定義で、現行踏襲すべき内容が明確化できたため上流工程でシステムテストの現行保証テストや負荷テストなどの検証や移行計画策定時のデータの扱いや保証手段などを検討できた。

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

用語集

＜A to Z＞

AP (Application)

アプリケーションの略語。

DB (Database)

データベースの略語。

EOL

End of Life の略語。

EOS

End of Service の略語。

HW (Hardware)

ハードウェアの略語。

ISV 製品

Independent Software Vendor の略語。コンピュータメーカ系列ではない、パッケージソフトウェアの開発・販売会社の総称。

MW (Middleware)

ミドルウェアの略語。

PF (Platform)

プラットフォームの略語。

SW (Software) など

ソフトウェアの略語。

WS (Workstation)

ワークステーションの略語。

＜か行＞

業務仕様

業務要件を実現する具体的な方法を定義したもの。

（例）設計書、業務マニュアル等

業務要件

対象の業務を遂行するうえで課題対応や問題点への対策を起因とする業務追加・改善のために必要な事項をまとめたもの。

（例）業務プロセスの定義、業務手順の変更

現行資産

現行システムのソフトウェアやドキュメント類のこと。

（例）ソースコードやその実行モジュール、設計書等

現行踏襲

現行システムの要件や仕様を新システムに引き継ぐこと。

（注）現行踏襲の対象は曖昧になりやすいため、対象を具体的に明確化することが必要である。

現新比較テスト

現行システムと新システムに対して、同一条件でインプットとアウトプットの比較を行う試験。（類語：新旧比較テスト）

＜さ行＞

資産凍結

新システムに移行する資産を一定の期限で確定すること。

制御機能

業務を実行するための補完機能であり、ハードウェア、OS 以外のもの。

＜た行＞

テーラリング

一般的な標準を基に、個々の利用シーンに合わせた標準を策定すること。

＜は行＞

ハードウェア更改

保守切れ対策を契機にハードウェア製品を入れ替えること。アプリケーションは基本的に変更しないが、バージョンアップに伴う非互換の吸収を行う。

（例）Linux のバージョンが上がった事により、Linux 上で動いているミドルウェア製品のバージョンが上がる。

復元

仕様書などのドキュメントの復元を指しており、現行システムのプログラムの解析や、関連する既存のドキュメント、有識者からの情報を整理して不足しているドキュメントを書き起こす作業のこと。

部門

- **利用部門**

  システムを直接使用して業務を行うエンドユーザ。画面を操作しながら業務を遂行するオペレータなどのこと。

- **業務部門**

  経営層のビジネス戦略を踏まえ、事業戦略の立案を行う部門。機能要件の責任所管であり、機能要件を承認する立場である。

- **システム部門**

  システムを構築する部門。非機能要件の責任所管であり、非機能要件を承認する立場である。

- **運用部門**

  現行システムの運用を行う部門。業務運用とシステム運用の両方を含む。

＜ま行＞

モダナイゼーション

稼働中のソフトウェアに対して、現行資産を活かしながら最新の製品や設計で置き換えること。特に稼動後数十年経っているようなメインフレームを中心とする基幹システムの刷新を行うことを意味する。本ガイドではリビルド、リライト、リホスト、ハードウェア更改等の再構築手法を総称してモダナイゼーションと呼ぶ。

＜ら行＞

リビルド

業務要件を変更せずに、新システムの特性に合わせて業務仕様を変更する。業務仕様に基づき新システムで効率よく動作するように業務アプリケーションを作り変える。

リホスト

業務仕様は変更を加えず、プラットフォームである OS やミドルウェアやハードウェア及びアプリケーションを移行すること。現行システムと次期システムで同一の言語を使用する。

（例）Unix から Linux、メインフレームからオープンシステム

リライト

業務仕様は変更を加えず、プラットフォームであるミドルウェアやハードウェアを移行すること。業務アプリケーションについては移行要件に従って現行システムとは別の言語で書き換える。

参考文献

[1] 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

[2] PM 学会発表　2015 年度「レガシーモダナイゼーションプロジェクトにおける提供品質確保の取り込み」, 2015

[3] PM 学会発表　2016 年度「新旧比較照合テスト自動化による,マイグレーション開発の品質保証」, 2016

[4] PM 学会発表　2016 年度　キーノートスピーチ「デジタルビジネス時代に求められるインテグレーション」, 2016

[5] 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

[6] 経済産業省・情報システム・モデル取引・契約書（2007・4）, 2007, 13 頁
http://www.meti.go.jp/policy/it_policy/keiyaku/model_keiyakusyo.pdf

[7] 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

[8] Project Management Institute, Inc.　2013 年度「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　日本語版」, 2013

[9] 雑誌 FUJITSU　2012 年 3 月号 VOL.63 NO.2 「新たな時代の SI 技術」

[10] IPA/SEC,「機能要件の合意形成ガイド」, 2010,
https://www.ipa.go.jp/sec/softwareengineering/std/ent03-a.html

[11] IPA/SEC,「非機能要求グレード」,2014,
https://www.ipa.go.jp/sec/softwareengineering/std/ent03-b.html

執筆（敬称略）

システム構築上流工程強化部会
「モダナイゼーション WG」

主査：	大山　宏	株式会社エヌ・ティ・ティ・データ
委員：	御魚谷　かおる	富士通株式会社
	小林　茂憲	日本電気株式会社
	小林　豊	三菱ケミカルシステム株式会社
	崎本　壮	株式会社日立製作所
	千田　正一	富士通株式会社
	内藤　克郎	東京海上日動システムズ株式会社
	渡邉　崇	JFE システムズ株式会社
オブザーバ：	新子　剛弘	株式会社エヌ・ティ・ティ・データ
	高橋　宏	富士通株式会社
	手島　さくら	株式会社エヌ・ティ・ティ・データ
	鈴木　良尚	日本電気株式会社
	斎藤　洸一	株式会社エヌ・ティ・ティ・データ
	林　慎一郎	東京海上日動システムズ株式会社
	山本　一也	株式会社三菱東京 UFJ 銀行
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
（事例協力）	遠藤　玄声	株式会社ソフトロード

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：	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
	小浜　耕己	スミセイ情報システム株式会社
	加藤　一郎	日本電気株式会社
	崎本　壮	株式会社日立製作所
	長山　一	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
	横山　隆介	株式会社日本取引所グループ

SEC BOOKS
システム再構築を成功に導くユーザガイド　第2版
～ユーザとベンダで共有する再構築のリスクと対策～

平成29年3月10日　1版1刷発行
平成30年2月23日　2版1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 978-4-905318-57-6　Printed in Japan

ISBN978-4-905318-57-6
C3055 ¥1759E

定価:本体1,759円+税

IPA 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-TN17-005