デジタル・ガバメント推進標準ガイドライン

実践ガイドブック

（第３編第７章　設計・開発）

目次

事例・参考の一覧

Step. 1 設計・開発の全体の流れ

設計・開発は、専門的で内容がよくわからないものだから、専門の事業者に任せておけばよい。そう思っていませんか？

たしかに、情報システムを構築する作業は専門的で多岐にわたり、日々新しい技術が生み出される分野です。だからと言って、作業を事業者に全てお任せ（＝丸投げ）してよいわけではありません。設計・開発を円滑に進めるためには、事業者とPJMOを中心とする発注者が、二人三脚で作業を進めていく必要があります。

ここでは、PJMOが、事業者とともに設計・開発の活動を円滑にトラブルなく進めるために求められる作業や必要となる知識やノウハウについて、要点を押さえて説明します。

本ドキュメントの構成は、次のとおりです。

Step.2　設計・開発を開始するための事前準備

設計・開発を開始する間に、要件を適切に事業者に伝える必要があります。また、PJMOが求める情報システムをトラブルなく構築していくためには、仕様の調整や、できた情報システムを適切に検証することが必要となります。ここでは、これら設計・開発の前提知識や心得えを説明します。

Step.3　ブラックボックスにならない設計・開発の計画

設計・開発事業者が決まったら、最初にやることは計画を立てることです。設計・開発の活動は、PJMOにとっては、実態が見えにくい活動になりがちで、問題の発覚が遅れて大惨事になることもしばしばあります。ここでは、設計・開発の活動をブラックボックスにしないようにするための設計・開発の計画に係るポイントを説明していきます。

Step.4　設計・開発・テストの管理

設計・開発の大部分の作業は、事業者が行うことになりますが、PJMOが適切な関与を行わなければ、良い情報システムを構築することはできません。ここでは、テストを通した品質管理を中心として、設計・開発の実施を管理していくためのポイントを説明します。

Step.5　見落としがちな活動に注意

設計・開発でやらなければいけないことは、情報システムの構築だけではありません。本番で情報システムを稼働させ、サービス・業務の円滑な運営を行っていくためには様々な活動が必要になります。ここでは、設計・開発の活動の中で見落としがちな活動とそれらのポイントを説明します。

Step.6　新しい業務の運営をスムーズに行うための準備

情報システムの開発・テストが完了し、いよいよサービス・業務の開始が迫ってきました。でも、まだ心配事が残っていませんか？ここでは、情報システムを無事に稼働させ、新しいサービス・業務の運営を円滑に行っていくために必要となる最終盤の作業について説明していきます。

Step.2 設計・開発を開始するための事前準備

PJMOが、設計・開発の活動に適切に関与していくためには、最低限の知識とPJMOが果たすべき役割と責任を認識しておく必要があります。これらの内容は、要点を押さえれば、それほど難解なものではありません。

では、設計・開発の活動に必要となる前提知識や心構えを見ていきましょう。

1　設計・開発で職員が行うべき事を理解する

【標準ガイドライン関連箇所：第3編第7章第1節】

設計・開発の具体的な活動を行うのは、調達によって選定された事業者です。事業者は、調達仕様書及び附属資料である要件定義書をインプットに、設計・開発工程の活動を計画し、活動を行います。設計・開発工程の作業は、情報システムを対象とした専門的なスキル・経験が求められます。

では、職員が関与しなくても作業が順調に進むかというとそうはいきません。一般的に、職員の関与が低いほど、設計・開発の成功確率は低下します。

したがって、『専門的』でわかりづらい設計・開発工程の作業において、『職員が関与する』ことで効果がある作業とは何かを理解する必要があります。

まず、職員が作業に関与するに当たり、基本的な役割を以下に示します。

『設計・開発』を行う際の職員の基本的な役割

- 要件の内容を事業者に正しく伝える役割
- 要件どおりに情報システムができたかを確認（テスト）する役割
- プロジェクトの進捗状況を正しく把握し、スケジュールや関係者間において発生する調整を適切に行う役割

A. 『要件の内容を伝える役割』

要件は、既に要件定義書としてドキュメントに取りまとめられています。事業者は、その情報を基に『どうやって作るか』を具体化する設計作業を開始し、徐々に詳細化していきます。その作業の過程で、要件定義書だけでは読み取れない発注者側の意図や要望について、発注者側は正しく伝達することが必要となります。また、設計をする中で見えてくる課題等の対応方法を決めることも必要です。

この段階でコミュニケーションが十分にとれないと、いざ運用段階で実際に利用してみると使い勝手が悪かったり、手戻りが発生したりといった弊害を招きかねません。

以下の事例は、実際のプロジェクトで発生してしまった実例です。このような失敗をしないように留意してください。

事例：基本設計段階でのコミュニケーション失敗

事例 7-2
基本設計段階でのコミュニケーション失敗

過去の設計段階における画面設計にて、以下のような問題が発生しています。利用者が効率よく、誤解なく正確に使えるように設計段階で留意しましょう。

(1) 固定幅や固定行数で画面が設計されているため、画面の右側や各ブロックの下に空き領域が出てしまうケースがある。
※画面の一覧視認性を高める配置・構成にすべきです。
(2) 大事な情報が横スクロールしないと見られない作りになっている。
※縦スクロールする利用者画面において横スクロールの UI は避けるべきです。
(3) 直感的に使いにくいボタン配置
・「更新」ボタンが中央付近、「戻る」ボタンが最も右端にあり、操作ボタンが離れている
・ボタンレイアウトのトレンドを意識する
例：Web やスマートフォンにおける UI（ユーザインタフェース）では
「修正」「更新」
「戻る」「進む」
「キャンセル」「OK」
のように、左側のボタンが後ろ向き操作、右側のボタンが前向き操作とすることがトレンドです。
(4) 否定文を使った確認ダイアログの利用は利用者の迷いや誤操作を招きやすい
例：この操作をキャンセルしますか？「キャンセル」「OK」
(5) 開いた Web ページのサイトにおける位置を示すものがない機能構成ツリーやパンくずリストを設定しないと、利用者に全体像が伝わらない、操作が煩雑になる、あるいは迷子になることが発生します。
(6) エラーメッセージがわかりにくい
どのシステムのどのデータ、どの処理でエラーが起きたかを明示することが望まれます。
(7) インポート機能がない
複数の申請を一括処理するための機能・UI がない
※一括申請の利用者の存在と利用頻度を考慮するべきです。
(8) 異常処理系において面倒な運用操作が必要
プルダウンの選択肢を変更しようとした場合に、プルダウン項目を決めている別の項目設定をいったん空欄にしないと変更できないつくりになっていた。
※親子関係のある設定や選択肢は、子側を変更しようとした際に親側の変更もできるつくりにすべきです。

B. 『要件どおりに情報システムができたかを確認する役割』

構築された情報システムが、伝えた要件を満たすものになっているかを確認するのは職員の役割です。

もちろん、情報システムの様々な処理がきちんと動作できるのかは、職員が確認する前に、事業者がテストで十分に確認していることが前提となります。

テストにおいて職員に求められることは、テストすべき項目（テスト仕様書）をレビューし、内容の正しさをチェックすることと、事業者が実施したテストの結果を確認することです。

テストを実施する作業の中では、テスト項目に従って情報システムが正しく動作しているかどうかを確認していきます。テスト項目とは、情報システムにどう動いてほしいかの要求を詳細化したものです。これが間違えていたり、抜け漏れがあったりすると、情報システムが要求どおりに構築されたとは言えなくなります。したがって、職員は、テスト仕様書に目を通し、要求を反映したテスト項目となっているかを確認することが重要となります。

もちろん、テストには様々な種類が存在し、テストごとに職員の関わり方は異なります。詳細は「Step.3-3 テストの計画を立てる」を参照ください。

通常のテストは上記のとおりですが、受入テストは例外扱いで、職員自らがテストを実施する必要があります。

C. 『プロジェクトの進捗状況を正しく把握し適切な調整を行う役割』

この役割は、発注者側の立場によるプロジェクト管理の一つとして、PJMOに求められるものです。

設計・開発工程では、プロジェクト期間が長く、プロジェクト開始時には概要レベルの全体スケジュールを立案し、詳細な計画は、段階的に作成していくことになります。この作業を進める中で、様々な関係者が登場します。新たな情報システムを導入する際には、ほとんどのケースで業務を見直して、手順や内容の変更を行います。その過程で業務関係者間での調整がうまくいかないことは、プロジェクトの進捗を遅延させる要因となります。

また、プロジェクトには期間と予算の制約がありますので、どんなに良い方法であってもこれらの制約の関係で妥協しなければならないこともあります。

そのため、PJMOは、プロジェクトを正常に進めるための調整が重要であることを認識し、その役割を担う必要があります。

2　設計・開発全体を通して理解すべき点とは

【標準ガイドライン関連箇所：第３編第７章第１節】

PJMOが、発注者として設計・開発を適切に管理していくために、設計・開発の活動全体を俯瞰的に理解しておく必要があります。ここではその内容を説明していきます。

A. 要件を理解した職員の継続的な関与がプロジェクトを安定させる

大規模なプロジェクトの場合、設計・開発工程だけで1年以上の期間が必要となることが多くあります。

要件定義において、多くの職員からの意見を集約し、様々な情報を分析して、新しい業務を決定しましたが、その全体像を理解している職員は、要件定義に関わった職員の中でもごく一部に限定されます。

このような職員は、プロジェクトにとって非常に貴重な存在です。設計・開発工程において、事業者に要件の全体像を説明し、要件追加変更に伴う影響度の調査が行うには、全体を理解していないとスムーズにできないからです。また、要件がどのような背景で発生し決められたのかの経緯を正しく理解していないと、誤った説明や安易な変更を行うおそれがあります。

こういった問題を発生させないためにも、要件定義に関わった発注者側の職員、特に業務を理解している業務実施部門の職員をプロジェクトの体制に参画させ続けられるよう、体制の組成時に調整を行うことは効果的です。

B. 要求とコストと納期のバランスをとる

設計・開発の活動を進めていく中では、要件変更や当初計画からのかい離は、様々な要因により発生します。要因には、政権交代や世の中の情勢変化に伴う方針変更、連携する情報システムの要件変更等のプロジェクト外部で発生する要因や、設計として詳細化していく中で発覚した要件定義の誤りや受入テスト時に発見される要件の抜け漏れ等のプロジェクト内部で発生する要因があります。

このような場合に、「納期は絶対で、追加発注も不可能。でも、変更は受け入れて当初計画どおりに仕上げてください。」と指示を、建設的な議論をせずに行ったとしてもプロジェクトはうまく進みません。

事業者は、要件定義の内容、納期・スケジュールに基づいて、コストを見積り、計画を立てます。つまり、「要件（スコープ）」「コスト」「納期（スケジュール）」のいずれかが変更されると、ほかにも必ず影響が発生します。

では、「今年度の計画は変更できないので、追加や変更は全て来年度以降に先送りします」としても、そこに重要な要件が含まれている場合は、今年度出来上がった情報システムがまともに動かないリスクもあります。

これは、要件の変更だけではなく、スケジュールや納期の変更でも同様です。

では、このような場合にどう対処すればよいでしょうか？

これらを適切に管理するための仕組みが「変更管理」です。変更管理は、変更の必要性が発覚した際に、「変更に対する影響の調査」「影響度合いに応じた変更の決定の手続」等の変更に対する管理方法を定めるものです。まずは計画時に、これらを明確に定め、これに則り設計・開発工程作業中に発生する様々な追加変更を取捨選択していくことが大切です。影響度の判断によっては、当初の要件を諦め、新しく発生した要件と差し替える等、柔軟な対応が求められます。

C. 設計・開発の全体像と流れを理解する

設計・開発の活動は、設計・開発・テスト・移行等のいくつかの作業工程に区切って進めていくことが一般的です。しかし、これらの工程の組み方は多種多様で、それぞれメリット・デメリット、向き・不向きがあります。PJMOは、これらを踏まえた上で、事業者が立案した計画を理解して、プロジェクトにあった進め方となるよう、事業者に要望を伝えていきます。

ここでは、事業者の計画を正しく理解し適切な進め方を検討していくことができるよう、設計・開発全体の進め方に係る知識やノウハウを解説していきます。

◆ 設計・開発工程に含まれる全ての活動内容を俯瞰的に理解する

設計・開発では、様々な活動が行われます。事業者が作成するスケジュールを確認する際、まずは俯瞰的に工程の流れを押さえて、大きな齟齬がないかを見ていきましょう。

情報システムの設計・開発は、大まかに捉えると以下のような流れになります。

図 7-1
工程の全体像

これらの活動ごとの成果物を洗い出し、その成果物を得るために必要な作業を細分化し、作業にかかる時間や作業順を当てはめていったものがWBSに基づいたスケジュールとなります。WBSの確認については、「Step.3-2-E. WBSで作業計画を確認し進捗を把握する」を参照してください。

なお、工程の名称は、事業者によって呼び方が異なります。同じ名称でも、事業者によって捉え方が異なることがあるため、注意が必要です。工程の捉え方を間違えていると、適

切な時期に求めたレベルの成果物ができていなかった、というようなことが発生します。

表 7-1
標準ガイドラインと各社が定義する工程の比較

標準ガイドラインの定義		A社	B社	C社	D社	E社	F社
要件定義		要件定義	要件定義	RD:システム要件定義	要件定義	要件定義	要件定義
設計	基本設計	外部設計	BD:基本設計	UI:ユーザーインタフェース設計	外部設計	基本設計	基本設計
			FD:機能設計	SS:システム構造設計			
	詳細設計	内部設計	DD:詳細設計	PS:プログラム構造設計	内部設計	詳細設計	詳細設計
実装・単体テスト		コーディング／テスト	CD:コーディング	PG:プログラミング	開発	開発/単体テスト	開発/コーディング
			UT:単体テスト	PT:プログラムテスト			単体テスト
結合テスト		結合テスト	IT:結合テスト	IT:結合テスト	統合テスト	結合テスト	結合テスト
総合テスト		システムテスト	ST:総合テスト	ST:システムテスト	システムテスト	システムテスト	総合テスト
受入テスト		運用テスト	RT:受入テスト	OT:運用テスト・移行	－	運用テスト 受入テスト	運用テスト

◆ プロジェクトの特性（新規構築・更改等）によって、作業は異なる

設計・開発で行う作業の内容は、プロジェクトの特性によって異なります。例えば、情報システムを新規に構築するプロジェクトと既存情報システムの更改のプロジェクトでは、作業の内容が大きく変わります。

表 7-2
プロジェクト特性の概要

プロジェクト特性	概要	他の特性との違い
新規構築	新たに情報システムを構築する。	情報システムの構築に係る全ての作業が必要となる。
更改	既存の情報システムを再構築する。 ただし、再構築には様々な方法があり、インフラのみを変更して、新しい環境（ハードウェア・ミドルウェア等）上で既存の情報システムが稼働するようにすることや、現在の情報システムをいかしつつ機能の改修を行うこと等がある。	更改の方針に応じて、作業が異なる。 また、新規構築と比べ、テストや移行の比重が高くなる傾向にある。 事前検証を行う等、新規構築とは異なる作業が必要となる可能性がある。

また、一から情報システムを構築（＝スクラッチ開発）する場合、クラウドサービスの利用、パッケージ製品等の導入・カスタマイズでは、それぞれ活動の内容が異なってきます。例え

ば、クラウドサービス（PaaS）の利用では、機能に関する実装・テストよりも運用視点でのテストの比重が増えるかもしれません。

情報システムの更改を行うタイミングは、情報システムの機能を見直すに当たり、またとないチャンスです。現在のサービス・業務運営上の課題や既存情報システムの運用の中で寄せられた要望等を確認し、情報システムの機能改善を検討しましょう。なお、やむを得ない事情により、機能改修を伴わない更改を行うこともありますが、その際には機能の実装・単体テスト等を実施する必要がないかもしれません。このようなプロジェクトの特性に合わせて、作業内容の必要十分性を確認しましょう。

確認に当たっては、「Step.3-2 設計・開発の実施計画を立てる」で解説する設計・開発実施計画書のひな型を参考にして比較を行い、不足や不明点は事業者に確認して、事前に漏れや誤りをなくすようにしましょう。

◆ プロジェクトの進め方や開発手法により作業内容も異なる

設計・開発の進め方には、様々なものがあります。以前は、情報システムの構築といえば、ウォーターフォール型で進めることが主流でしたが、ソフトウェア開発の過去の失敗事例等を踏まえて、様々な進め方が考案されています。

標準ガイドラインで定義するプロジェクトは、大きく以下の2つに分類できます。

表 7-3
プロジェクト全体の進め方の種類

進め方の種類	概要	適用の基準例
通常の開発	定義した要件に基づいて全ての機能を構築してサービス・業務を開始する。その後、必要に応じて機能追加等を行う。	プロジェクト目標に対する具体的な実現方法が定まっており、要件の詳細な定義が可能な場合
実証実験型開発	必要最低限の機能を構築してサービスを開始し、効果をモニタリングしながら要件を明確にし、段階的に情報システムを構築していく。	プロジェクト目標に対する具体的な実現方法が定まっておらず、要件の詳細な定義が困難である場合

また、情報システムの開発手法も、複数あります。代表的な開発手法には、次のようなものがあります。

表 7-4
情報システム整備の進め方の種類とメリット・デメリット

開発手法の種類	概要
ウォーターフォール	図 7-1 で示したような工程を時系列に進め、原則として前工程の完了後に次工程を開始する情報システム構築作業の進め方である。 設計・開発の途中で変更が少ないと見込まれる場合に用いる。 工程を時系列で進めることから、計画が立てやすく、進捗の管理がしやすい。
アジャイル	開発対象となる機能（1～複数）の設計・開発・テストをイテレーション（反復）と呼ばれる短い期間に分けて進め、イテレーションが終了するごとに動く機能が出来上がる情報システム構築作業の進め方である。 設計・開発の途中で変更が多く発生すると見込まれる場合に用いる。 短期間で動く機能が出来上がるため、情報システムの利用者に確認を取りやすい。

ウォーターフォール型は従来型の開発手法ですが、その工程の中でプロトタイピングをとることも増えてきています。プロトタイピングとは、情報システムを本格的に構築する前に、試作品（プロトタイプ）を作成し、実物を見ながら要件を具体化していく手法です。ウォータ

ーフォールでの開発においても、設計工程の中でプロトタイピングを導入することで、関係者からの意見を聞きながら要件をとりまとめやすくなります。

昨今、開発手法として増えているのがアジャイルです。現時点においては、開発工程全てにわたってアジャイル開発を貫いた事例はまだ少ないですが、部分的に設計工程においてモックアップ画面等を導入して仕様確定を促進するケースは増えてきています。アジャイル開発の進め方は、サービス設計 12 か条にある「何度も繰り返す」、「一遍にやらず、一貫してやる」といった視点に沿うものですので、積極的に導入を検討してみてください。

アジャイルの考え方の規範は、「アジャイルソフトウェア開発宣言」として一般に公開されています。

注記
「アジャイルソフトウェア開発宣言」とは、従来型のソフトウェア開発のやり方とは異なる手法を実践していた開発者・研究者が 2001 年に作成した、アジャイル開発を進める上での「マインドセット」が書かれたもの

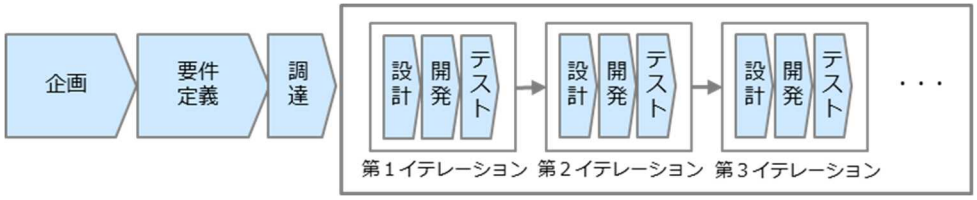


図 7-2
アジャイル開発の基本的な進め方

一方で、形式だけでアジャイル開発を採用すると失敗することになりかねません。以下のような形だけのアジャイル開発にならないように気を付けましょう。

- **投げっぱなしアジャイル**
  発注者の要件を調達以前の段階で十分に定めず、要件自体を設計工程で決めるというように後送りにする方法。調達する事業者の能力や経験によって、構築できるシステムの機能や品質が大きく変わってしまう。特に、事業者の能力が十分でなかった際に、要件定義内容を元に改善指摘をすることが難しくなってしまう。

- **優先度をつけないアジャイル**
  設計を進める中で出てきた要件について、発注者が優先度の判断をせず、全てを実現することを求めてしまう方法。限られた工数の中で効果の大きい部分を選び出していくことがアジャイルの特徴であり、優先度を判断しないのであれば後出しした要件を全て実現させるという非合理的な契約形態となってしまう。

なお、アジャイルが適さないと言われているプロジェクトもあります。一般的には、大規模なプロジェクトやミッションクリティカルなプロジェクトはアジャイルに適していないと言われています。ただし、大規模なプロジェクトであっても部分的にアジャイルを適用することは可能です。

開発手法は、必ずしもプロジェクト全体で統一しなくてはいけないわけではありません。メリット・デメリットを理解して、特定の機能群についてはアジャイルで進める、というように「良いどこ取り」していくことも大切です。

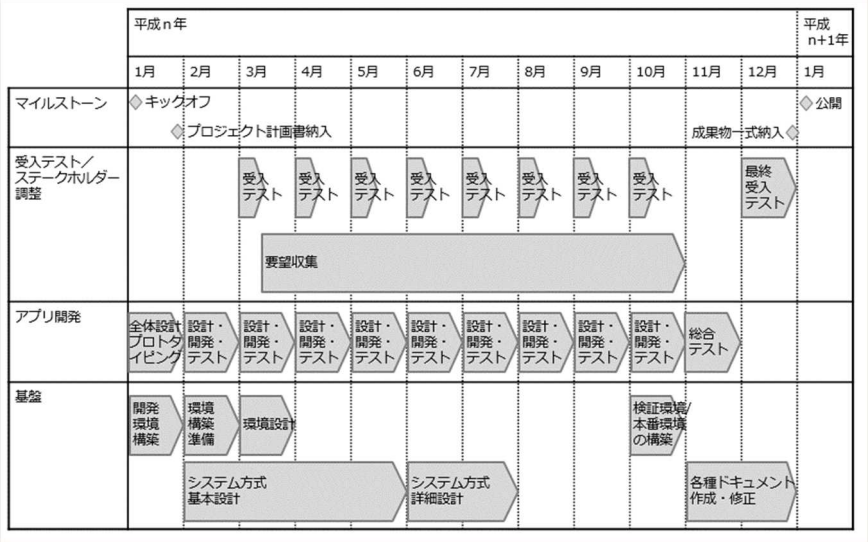
事例 7-1
ウォーターフォールとアジャイル
開発を組み合わせた開発計画

事例：ウォーターフォールとアジャイル開発を組み合わせた開発計画

ある省において、大規模な情報システムの更改を行うことになりました。既存の情報システムは長期間運用していたこともあり、現場からは多くの要望が上がっていたため、更改に合わせて情報システムを刷新することとしました。

現場からの要望は多種多様であったため、早めに実働する情報システムを用いた現場の確認を行わないと、実現イメージがすり合わずに後々変更や追加要望が多発することが予見されました。そのため、設計・開発は、アジャイルで行うことを検討しましたが、大規模な情報システムであるため、全てをアジャイルで構築しようとすると、管理コストが高くなる、要員の確保が難しい等の懸念がありました。

そこで、PJMOは、マイルストーンを明確にした上で、アプリケーションをアジャイルで、インフラをウォーターフォールで開発、構築する計画としました。アプリケーションの設計・開発は、一ヶ月単位で計8回行う計画とし、その都度受入テストを実施します。そして、出来上がったアプリケーションを確認しながら改善要望を収集し、アプリケーションの機能やユーザーインタフェース等を見直していくことにしました。



このような工夫をこらした結果、現場の要望を十分に反映した実用的な情報システムを計画どおりに構築することができました。

また、既存システムには、統計分析を行う基盤となるサブシステムがありましたが、分析の切り口を変える度に、当該サブシステムの改修や運用事業者へのデータ抽出の依頼でコストが掛かっていました。

そこで、更改にあたっては、この情報システムからデータを抽出するインタフェースを整備するとともに、職員自身がデータ分析をできるようにBIツールを導入しました。これによって、複雑なサブシステムがなくなりシステム構成がシンプルになったことで調達時の競争性が向上するとともに、改修費用も抑えることができるようになりました。

D. 通常シナリオだけでなく、緊急時の対応計画も準備する

設計・開発のスケジュールを作成する際に、各工程が順調に進むことを前提とした楽観

的なプランを1つだけ策定してしまうことが多くみられます。

実際にプロジェクトを進行する中では、予想していなかった様々な制約や問題が判明することもあり、スケジュール自体を大幅に見直すことも少なくありません。

このような際に役立つのが、緊急時対応計画（コンティンジェンシープラン）です。

緊急時対応計画の中では、予想していなかった問題が発生した際に後続工程の開始を遅らせたり、工程を分割して同時並行で進めたりするような形で、プロジェクトへの影響を最小限に抑えるための計画を準備します。このような計画を事前に準備しておけば、実際に問題が起こった際にも比較的冷静に正確な判断を行うことができます。

緊急時対応計画は、設計・開発工程だけでなく、情報システムが稼働した後の運用工程でも有効です。災害発生時等の緊急事態に際して、誰がどのように行動し何を優先して作業すべきかを定めます。

なお、緊急時対応計画と業務継続計画は類似する内容となりますが、緊急時対応計画は災害発生時の対応そのものに焦点を当てていることに対して、業務継続計画は災害発生後に必要となる業務を継続して実施することに焦点を当てているという違いがあります。

E. メンテナンス性を考慮した成果物の構成、内容を考える

プロジェクトで作成される成果物は、調達仕様書でも指定されているとおり、多岐に渡ります。ここでは、その中でも特に重要な基本設計書に関して説明していきます。

システム開発においては、事業者に基本設計以降の工程を委託することが通常の形ですが、「C. 設計・開発の全体と流れを理解する」でも説明しているとおり、事業者は設計の詳細化を行いながらシステム開発を進めていきます。そのような工程において、基本設計工程は、発注者から示される要件内容を理解し、実際のシステムの設計に落とし込む、という発注者―事業者間のインタフェースとなる工程になります。完成するシステムの良し悪しはこの工程、つまりアウトプットとしての基本設計書の内容により大きく左右される重要な工程になります。本当に要件を正しく反映したシステムができるか、使いやすいシステムができるか、あるいは保守・運用を効率良くかつ長期に維持していくためのシステムができるか、などまさに「システムの命運を握る設計図」を作成していくということになります。

また、基本設計書は要件を実現するための設計図のみならず、システムを品質良く完成させるためのテストのためのインプットドキュメントであり、保守・運用フェーズにおいては設計変更等の「要」となるドキュメントでもあります。

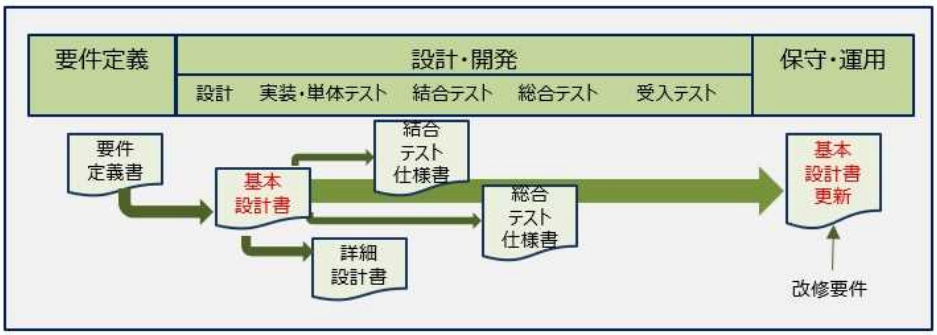


図 7-3
基本設計書の位置づけ

特に留意する事項としては、

- データと機能・処理の連携を含めた全体を俯瞰できるドキュメントになっている

か

- データに関する設計・定義事項が一元的に漏れなく記載されるようになっているか

など、要件を反映した全体像の把握とデータ利活用・連携の観点からデータに関する設計・定義状況の把握が容易なメンテナンス性の高いドキュメント構成、内容になっているかということです。

表 7-5 は基本設計書の構成例です。システムの特徴あるいは受託した事業者によって構成や内容は異なることもありますが、おおむね以下のとおりです。全体編、機能編、データ編等まずは全体構成を確認し、その上で観点や考慮点が満たされているか、あるいは分かりやすい目次構成になっているか等を十分に確認していきましょう。

表 7-5
基本設計書の構成例

項番	目次	内容	備考
1	全体編		
1-1	システム全体図	システム、業務、H／W，S／W，N／W等の各々視点から全体俯瞰する資料	
1-2	データの流れと機能構成	データ及びその流れとそれぞれの機能との関係全体を記載	
1-3	機能分割	サブシステム構成、機能分割、処理方式など基本的な設計の考え方を記載	マイクロサービス定義及び連携の設計コンセプトも記載
・・・	・・・	・・・	・・・
2	機能編		
2-1	機能・画面・帳票一覧	分割された機能を一覧として記載。また関連する機能と対応させた画面・帳票を一覧形式で記載	
2-2	画面・帳票フロー	画面、帳票と機能の流れ（展開条件、戻り条件など）を記載	バッチの場合はジョブフローとして記載
2-3	各機能別処理内容	機能毎の処理概要、入出力（データベース、ファイル、画面、帳票など）関連図、チェック／編集要領など定型標準フォームに記載	オンライン、バッチ、共通機能（API等）を処理／提供形態等により分ける。マイクロサービスごとの処理概要を含む。
・・・	・・・	・・・	・・・
3	データ編		
3-1	データモデル	要件定義で作成した概念レベルのモデルを詳細化。全てのデータ（データベース、ファイル、テーブル等）を関連づけて記載	
3-2	データ一覧、データ定義／レイアウト	データベース、ファイル、テーブルなどのデータに関する説明（マスターデータ等の分類、標準化レベル等も記載）	全てのデータ項目（コード含む）の意味定義も記載
3-3	CRUD	データと機能の処理別マトリックス。データの生成から更新、参照、消滅までのライフサイクルを記載	
・・・	・・・	・・・	・・・

Step.3 設計・開発の計画

設計・開発を担う事業者にとって、最初のインプット資料は、調達仕様書や要件定義書です。そのため、その内容には十分な質・量が求められますが、どんなに準備したとしても、要件の不備・不足は発生しますし、作業が進めば当初の計画からのかい離は起こり得ます。これらをPJMOが早期に察知して適切な関与を行っていかなければ、後々大変なことになりかねないのですが、事業者の活動は、実際の状況が見えづらくブラックボックスになりがちで、対応が遅くなってしまうことが多くあります。

ここでは、事業者の活動を正確に捉えて、設計・開発を円滑に進めて品質の良い情報システムを作り上げていくために、計画時点で理解すべき知識やノウハウを紹介していきます。

1　設計・開発の管理の要点を理解する

【標準ガイドライン関連箇所：第3編第7章第1節】

設計・開発を担う事業者の活動状況は、事業者との定例会において共有されますが、専門的な内容が多くなると、説明をただ聞くだけになりがちです。そのような状況でも、管理すべき管理の要点をしっかり押さえて確認し、不明な部分を事業者から引き出していけば、正しい方向にプロジェクトを導いていくことができます。

ここでは、設計・開発における管理の要点を見ていきましょう。

A. 定点観測こそ進捗・品質管理の要（かなめ）

もしも、あなたに情報システムの整備に係るプロジェクトの経験がなくても、まずは作業の状況を定量値で管理し、継続してその値を把握してみましょう。そうすると、現場の状況がいろいろな角度から見えてきます。これにより問題が発生する予兆を捉えることができれば、その事象を個別に分析することで、原因を捉え、必要な対策をすることにつながります。

表 7-6
工程別観測定量値と確認観点

工程	定点観測すべき定量値と確認観点
設計	WBSに対する進捗率（EVMのSPI）にて、予実のかい離と傾向を確認し、予定した作業がスケジュールどおりに完了するか、対策が必要かを確認する。
	当初想定した生産性と実生産性のかい離（EVMのCPI）にて、予実のかい離と傾向を確認し、予定した作業が完了するか、対策が必要かを確認する。
	品質状況（信頼度成長曲線等）にて、適切なレビューが安定的に行えているか、品質に問題がないかを確認する。
開発	WBSに対する進捗率（EVMのSPI）にて、予実のかい離と傾向を確認し、予定した作業がスケジュールどおりに完了するか、対策が必要かを確認する。
	当初想定した生産性と実生産性のかい離（EVMのCPI）にて、予実のかい離と傾向を確認し、予定した作業が完了するか、対策が必要かを確認する。
テスト	テストの消化数の予実、不具合数等の予実の推移（信頼度成長曲線）にて、予実のかい離と傾向を確認し、最終的に品質基準を満たすことができる見込みか、対策が必要かを確認する。
	摘出した不具合から不具合の原因の傾向を確認し、不具合の原因が満遍なく摘

注記
EVMとは、WBSにより詳細化した各作業項目に出来高計画値（PV：Planned Value）を設定し、プロジェクトの進捗を出来高実績値(EV：Earned Value)として定量化すること。EVMは Earned Value Management の略。（EVMの詳細については、Step.3-2F を参照）

注記
SPIとは、累計の出来高計画値（PV）に対する累計の出来高実績値（EV）の比率のこと。SPIは Schedule Performance Index の略。

工程	定点観測すべき定量値と確認観点
	出できているか、対策が必要かを確認する。
	プロジェクトリスク管理上の重要な対策として、テストで発見した不具合を修正して再テストするための期間をあらかじめ計画に入れておくこと。
全体	課題の発生数、解決数、解決率を確認し、滞りなく課題が解決できているかを確認し、解決が滞っている課題があれば、対策などを検討する。

注記
CPIとは、累計の出来高計画値（PV）に対する累計の出来高実績値（EV）の比率のこと。CPIは Cost Performance Index の略。

定量値の観測で大切なことは、事前に進捗率の測り方を明確にしておくことです。作業ステータスのような定性的な情報も、表 7-6 のようにプロジェクトで定めた基準に従って、定量的に管理するべきです。さらに、ステータス管理は、可能であれば作業に応じた成果物（ドキュメントやプログラミング）を捉えて、進捗を計上することをお勧めします。

このようなルールがなく担当者の恣意性に委ねると「進捗率 90%」のまま数週間経つようなことになりかねません。例えば 100 本のプログラムがあって、50 本のコーディングが完了したから「コーディング工程」の進捗率が 50%、そのうち 30 本の単体テストが完了したので「単体テスト工程」の進捗率は 30%というのではなく、100 本の機能の内、進捗率 100%が 30 機能、進捗率 50%が 20 機能というように計上します。ドキュメントについても同様です。

表 7-7
進捗計上ルールのサンプル（ドキュメント）

進捗率	ドキュメント
0～50%	事業者の担当者が実作業の進捗に合わせて計上
60%	事業者内レビューの完了
70%	レビュー指摘事項の反映完了
80%	PJMOレビューの完了
90%	レビュー指摘事項の反映完了
100%	PJMOによる最終確認

なお、コーディングと単体テストを峻別して進捗管理や品質メトリクスをマネジメントすることは、少なくてもユーザ側にとって意味がありません。

テスト駆動開発（テストを先に実装し、それにパスするようにプログラムをコーディングする開発手法）ではコーディングと単体テストが混然として実行されますし、設計されたテストケースどおりではなく単体テスト中に怪しいと感じた箇所を探るような探索的テストによって、コーディングを変更することが推奨される場合もあります。

むやみに作業を細分化して管理するのではなく、現場の行動に即して管理単位が決められているかを確認してください。

B. 判断に必要な情報を職員が理解できる説明として事業者に求める

設計・開発の内容は専門的なものであるため、どうしても事業者の資料や説明内容は、職員から見ると専門的でわかりにくいものになりがちです。そのような場合、職員は、事業者が出してきた資料を苦労して読み解こうとしてしまうことがよくあります（もちろん悪いことではありません）が、職員の役割は、問題の本質を正しく捉えて判断することです。そのための近道は、職員が内容を理解できるように、丁寧な説明や資料のまとめ直しをしてもらうことです。内容がよくわからないからといって、事業者からの説明内容をうのみにして判断したり、判断を遅らせたりしてはいけません。

ただし、説明や資料の出し直しの過度な要求には、注意が必要です。進捗の遅延を引

き起こす要因にもなりますし、事業者との関係悪化の原因にもなります。そのような状況になる前に、PMOや府省CIO補佐官に相談してみてください。設計・開発事業者と「協働して、良い情報システムを作り上げていく」という心構えが大切です。

2　設計・開発の実施計画を立てる

【標準ガイドライン関連箇所：第3編第7章第1節】

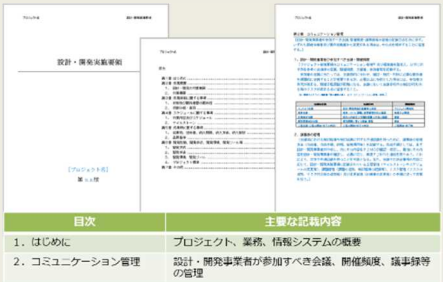
この実践ガイドブックには、別添として設計・開発の実施計画書と実施要領のひな形を示しています。

様式例 7-1
設計・開発実施計画書、設計・開発実施要領のひな形

様式例：設計・開発実施計画書、設計・開発実施要領のひな形

設計・開発実施計画書、設計・開発実施要領のひな形を本章別紙としてまとめています。

目次	主要な記載内容
1．はじめに	プロジェクト、業務、情報システムの概要
2．作業概要	設計・開発の対象範囲、作業概要
3．作業体制に関する事項	体制及び関係者間の関係性、役割分担、責務
4．スケジュールに関する事項	作業内容及びスケジュール、マイルストーン
5．成果物に関する事項	成果物、品質基準、担当者、納入期限、納入方法、納入部数、構成、内容
6．開発形態、開発手法、開発環境、開発ツール等	開発形態、開発手法、開発環境、開発ツール
7．その他	設計・開発の実施の事情に応じて必要な事項



目次	主要な記載内容
1．はじめに	プロジェクト、業務、情報システムの概要
2．コミュニケーション管理	設計・開発事業者が参加すべき会議、開催頻度、議事録等の管理
3．体制管理	作業体制の管理手法
4．工程管理	設計・開発の作業、工程の管理手法、
5．品質管理	品質基準、品質管理方法
6．リスク管理	リスクを提示する際の手順や報告様式
7．課題管理	課題を提示する際の手順や報告様式
8．システム構成管理	ハードウェアやソフトウェア製品、ネットワーク等の各資産における管理項目
9．変更管理	管理対象、変更手順、管理手法
10．情報セキュリティ管理	情報セキュリティ確保に必要な対策

あくまでこのひな形は例示です。プロジェクトの内容に応じて記載内容を個別に追加、変更してもかまいません。ひな形を見ると、何をどのようなレベルで書くべきかの参考になると思います。

以降では、設計・開発の実施計画を作成するときに、特に注意が必要なポイントについて説明していきます。

A. 2 種類のプロジェクト計画書の相違点を理解する

設計・開発実施計画書は、設計・開発事業者が作成する「プロジェクト計画書」と同じものでしょうか。これらは、厳密には異なるものです。

設計・開発実施計画書は、当該事業者が担当する設計・開発作業の範囲について、PJMOが作成するプロジェクト全体のプロジェクト計画（第3編第2章で示すプロジェクト計画書）を具体化・詳細化したものです。事業者との活動においても、他の関係者とのコミュニケーションや変更に対する管理は、プロジェクト計画に従って行わなければいけません。つまり、設計・開発実施計画書は、プロジェクト全体のプロジェクト計画書と整合性が取れている必要があります。

事業者が作成する「プロジェクト計画書」と必ずしも分ける必要はありません。事業者が

プロジェクト計画書を作成し、それらをPJMOがプロジェクト全体のプロジェクト計画書との整合を確認して確定するという流れでも良いです。もちろん、PJMOが設計・開発実施計画書を作成しても良いです。

大切なことは、設計・開発実施計画書、プロジェクト全体のプロジェクト計画書、事業者が作成するプロジェクト計画の整合性が取れており、全ての関係者が齟齬なく作業を行えることです。

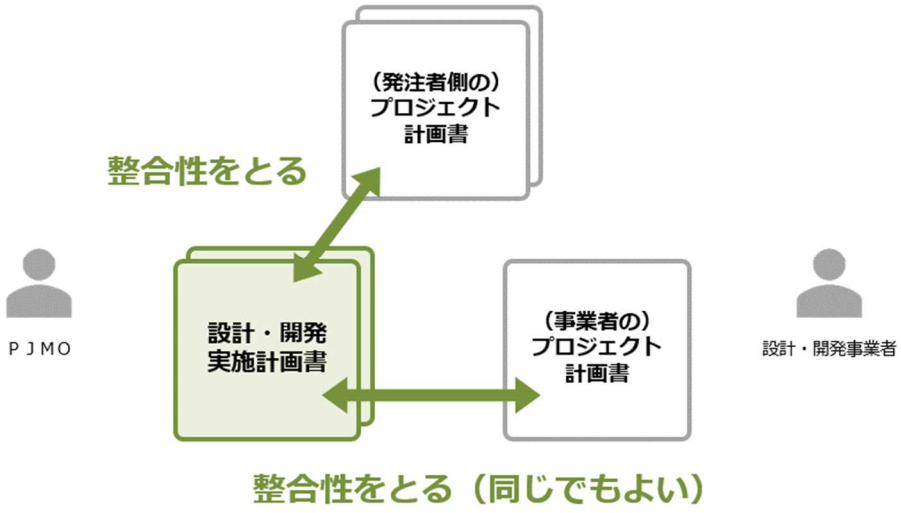


図 7-4
設計・開発実施計画書、プロジェクト計画書、事業者のプロジェクト計画書の関係性

B. 意思決定の手順を明確にする

設計・開発の実施中には、プロジェクトの内外問わず様々な課題や問題が発生します。その際に、どのように意思決定を行うかを計画時に明確にしておきます。

通常は、事業者との定例会を設置し、発生した課題や問題はその場で対応を決定していくことになります。しかし、優先度が高く解決が難しい問題を定例会の場だけでやりとりしていると、解決が遅れてプロジェクト全体の遅延につながってしまいます。

したがって、優先度が高い問題、影響が大きい問題が想定外に発生した場合の意思決定の方法をあらかじめ定めておくことが大切です。一般的には、課題や問題の影響範囲や優先度等の基準を定めておき、一定以上の基準の課題や問題が発生した際に、定例会議以外に、発生した課題を集中的に検討する会議の設置、影響範囲や影響度合いに応じての参加者等を定めておきます。

大切なことは、課題や問題が発生した場合に、それらを事業者やプロジェクトの一部の関係者だけでブラックボックスにせず、影響がある関係者に適切に共有し、共同して解決に向かうことです。

C. 当初計画からの変更は、必ず関係者で合意する

策定時に関係者全体で合意した内容が、いつの間にか当事者だけで変更されてしまうというケースがあります。特に、プロジェクトの目的に直結する重要な事項の変更は、全体関係者及び責任者の承諾なしに変更することは厳禁です。

関連する事例として、実践ガイドブック「第2章事例 2－9. 開発途中の機能追加による目標の形骸化」も参照してください。

D. 他の関係者との役割分担の境界線を定める

設計・開発の計画においては、PJMO、設計・開発の事業者、プロジェクト内の他の事業者、連携する情報システムのプロジェクト等の関係者の役割分担（誰が何を実施するか）を決定し、関係者で合意することがとても大切です。

役割分担でよく問題になるのは、他の情報システムの連携仕様を誰が具体的に定めるのか（要は、具体的なインタフェース仕様書を誰が作成するか、テストにおいて誰が主導してデータやテスト計画を作成するか）です。特に、複数情報システムで共有する連携仕様の場合は、分担が不明確になりがちなので、計画時点で明確に取り決めておきましょう。連携に関するテストを誰がどのように取り仕切るかも同様に決めておく必要があります。

E. ＷＢＳで作業計画を確認し進捗を把握する

WBSは、設計・開発のスケジュールや進捗を管理する際に一般的に使われるもので、設計・開発の活動ごとの成果物を洗い出し、その成果物を得るために必要な作業を細分化し、作業にかかる時間や作業順を当てはめていったものです。（正確には、縦軸の作業項目がWBSであり、スケジュールを含めたものはガントチャートと呼びます。）

ここでは、設計・開発の計画や進捗状況を正確に把握するためのWBSに関するノウハウを見てきます。

◆ ＷＢＳの構造を理解して計画や状況を効果的に把握する

スケジュールは、「マイルストーンを明確にする」「概略のスケジュールを立てる」「詳細な作業ごとに、作業にかかる時間と作業間の依存関係を検討し、作業順を決める」という3段階で組み立てられます。

注記
WBSとは、プロジェクトに必要な作業の全体を細かい単位に分割した上で、全体の構成を示したもの。WBSは Work Breakdown Structure の略。

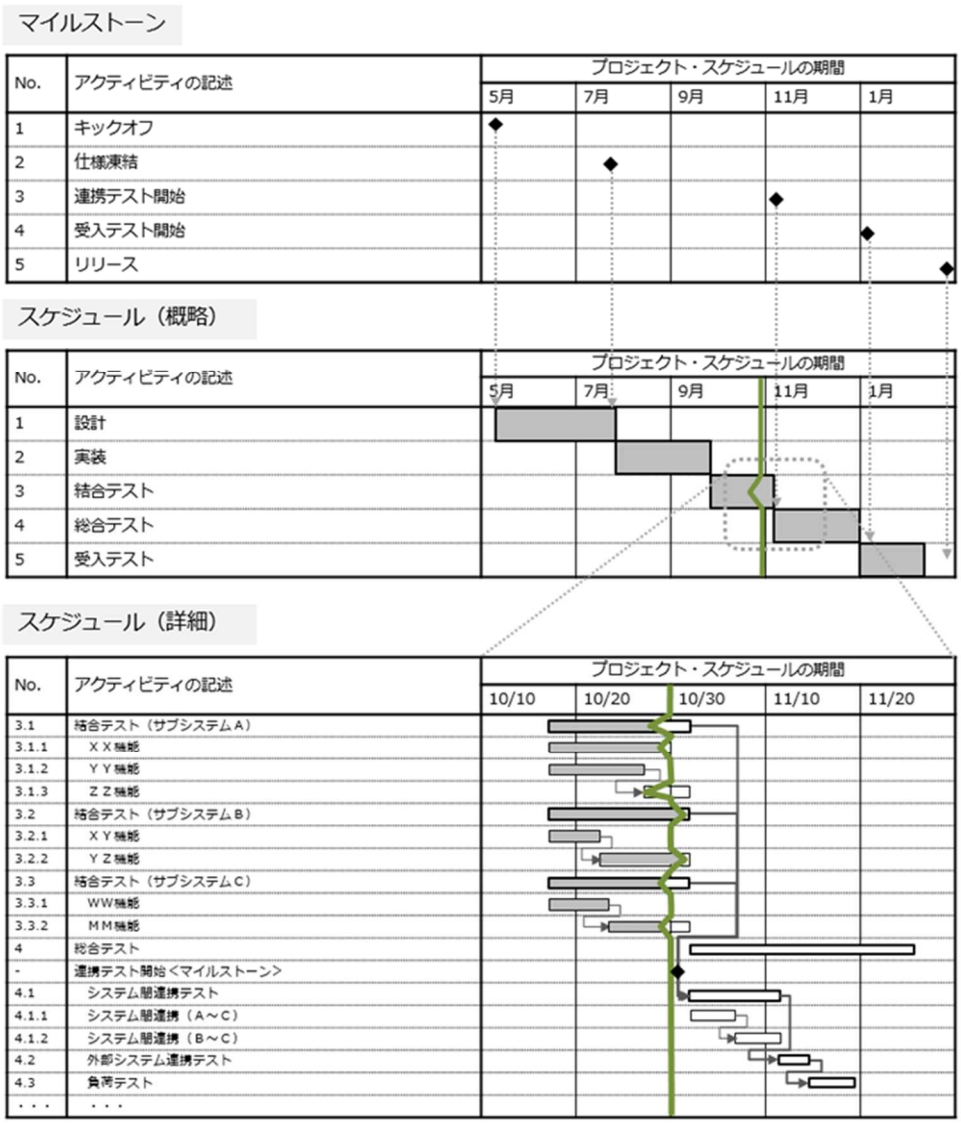


図 7-5
WBS のイメージ。
このイメージはアプリケーション開発に着目した例。運用に関しても、運用設計、運用手順書、総合テスト、運用テストという流れで同様に WBS を作成する。

WBSを確認する際には、この構造を理解し、「マイルストーンに齟齬や抜け漏れはないか？」「マイルストーンに対して、概略のスケジュールが妥当か？」「各作業の依存関係の誤りや作業の抜け漏れはないか？」「どのアクティビティの経路がクリティカル・パスとなっているのか？」のように段階的にチェックすることで、網羅的な確認をすることができます。

また、「Step3-1-A. 定点観測こそ進捗・品質管理の要（かなめ）」でも述べたように、定例会等で進捗状況を確認していると、「進捗率が 90%で止まっている作業が大量にある」というような状況がよくあります。これは、多くの場合、進捗率の定義が曖昧であることに起因しています。進捗率は、定量的に判断できるように事前に基準を定めましょう。

注記
クリティカル・パスとは、アクティビティの最長経路。この経路に必要な期間が、プロジェクトの最短所要時間となる。クリティカル・パス上のアクティビティが遅延した場合、プロジェクト期間に影響を与えるため、注意が必要。

F. ＥＶＭを用いた進捗管理手法を理解する

WBSにより詳細化した各作業項目に出来高計画値（PV：Planned Value）を設定し、プロジェクトの進捗を出来高実績値(EV：Earned Value)として定量化するEVMを用いることで、作業状況を客観的な統一尺度で可視化及び一元管理し、プロジェクトの進捗状況や進捗に係る課題・問題の把握を行い、事前に的確な対応を行うことが可能となります。

EVMを用いて進捗管理を行う場合、次に示す資料を作成します。

注記
PVとは、計画時に見積もった出来高値のこと。
PVは Planned Value の略。

注記
EVとは、進捗把握時までに完了した作業の出来高値のこと。
EVは Earned Value の略。

(1) EVM進捗管理表

WBSの各作業に対し、出来高計画値（PV）、出来高実績値（EV）、投入実績値（AC）を記載します。

本様式は各タスクの計画と実績の対比を詳細に示すものであり、進捗管理の基礎情報として用いるものであるため、必ず正確な情報を記載します。

注記
ACとは、進捗把握時までに完了した作業に投入したコストのこと。
ACは Actual Cost の略。

EVM進捗管理表

図 7-6
EVM進捗管理表

プロジェクト名：　＊＊＊システム設計・開発プロジェクト

WBS番号	作業名	実施主体	開始日		完了日		完了基準	出来高計画値(PV)	出来高実績値(EV)	投入実績値(AC)	投入実績値明細（人日）					
			計画	実績	計画	実績					20xx年xx月			20xx年		
								（人日）	（人日）	（人日）	xx	xx	xx	xx	xx	xx
1	進捗管理	PJMO														
1.1	WBSの作成	PJMO														
1.2	出来高計画値(PV)の設定	設計・開発事業者														
1.3	進捗報告	設計・開発事業者														
1.3.1	EVM進捗管理表の作成	設計・開発事業者														
1.3.2	進捗報告書の作成	設計・開発事業者														
1.3.2.1	進捗状況の把握・確認	設計・開発事業者														
1.3.2.2	報告資料の作成	設計・開発事業者														
1.3.2	進捗報告内容確認・対応策検討	PJMO														
2	要件定義の確定	設計・開発事業者														
2.1	要件定義内容の確認	設計・開発事業者														
2.2	要件定義変更案の整理	設計・開発事業者														
2.3	要件定義変更案の検討・調整	設計・開発事業者														
2.4	要件定義変更案の承認	PJMO														
2.5	要件定義確定	設計・開発事業者														
3	設計	設計・開発事業者														
3.1	次期システム実現に係る基本的事項の確認	支援事業者														
3.2	基本設計	支援事業者														
3.2.2	○○サブシステムの設計	支援事業者														
3.2.3	△△サブシステムの設計	支援事業者														

- 出来高計画値（PV）
  WBSにより詳細化した各作業項目に対し設定した出来高計画値（PV）を記載します。出来高計画値（PV）の合計は計画総コスト(BAC:Budget At Completion)と必ず一致するように設定します。
  出来高計画値（PV）の作成により、作業コストが多い作業項目や期間が可視化されるため、各期間の作業統制の難易度やリスクが顕在化した際の影響の大きさを判断することが可能となります。
  なお、出来高計画値（PV）はその精度を高めることが必要でありますが、事前に完全一致させる計画を立案することは不可能であり、また、計画外の事態は起こり得るものです。そのため、進捗管理に当たってはPVどおりに進めることのみを意識するのでなく、実績と計画の差異を常時把握し、計画どおりに戻すための対策を検討することが重要です。

- 出来高実績値（EV）
  進捗把握時までに完了した作業の出来高値を記載します。計上方法は表 7-7 に示す例を参考にあらかじめ定め、それに基づき計上します。
  出来高実績値（EV）は現状を可視化するための基礎情報であるため、リアルタイムで、かつ、正確な情報とする必要があります。

表 7-8
出来高実績値（EV）計上法

方式	説明	特徴
固定比率計上法	作業の開始時と完了時のみに進捗を計上する方法	前提として詳細な作業分界を行う必要がある。作業開始時と作業完了時に計上する比率には、0:100、30:70、50:50 等があり、事前に定める必要がある。

方式	説明	特徴
加重比率計上法	作業の達成率により進捗を計上する方法	作業期間が長い場合等、固定比率計上法では進捗を把握するのが難しい場合に有効。細かくマイルストーンを事前に定め、主観による曖昧さを排除する必要がある。

- 投入実績値（AC）
  進捗把握時までに完了した作業に対し、実際に投入したコストを記載します。
  投入実績値（AC）は現状を可視化するための基礎情報であるため、リアルタイムで、かつ、正確な情報とする必要があります。

注記
EACとは、完了時に予測される総コストのこと。
EACは Estimate At Completion の略。

注記
ETCとは、進捗把握時から完了までに予測されるコストのこと。
ETCは Estimate To Complete の略。

(2) 進捗状況表

EVM進捗管理表を集計し、月次の出来高計画値（PV）、出来高実績値（EV）、投入実績値（AC）、スケジュール効率指数（SPI）、コスト効率指数(CPI)、予測総コスト（EAC）及び残コスト（ETC）を記載します。

本様式は、月次の進捗状況の定量分析結果を示すものであり、次に示す例は、スケジュールが遅れている上にコストも超過し、体制強化や品質向上施策等の対応を至急行うべき状況を表すものです。

（様式及び記載例）

図 7-7
進捗状況表

プロジェクト名：　●●情報システムプロジェクト
サブプロジェクト名：　****プログラム設計・開発サブプロジェクト

20xx年xx月xx日

計画総コスト（BAC）（人日）	1319

	指数等	計算式等	20yy年				20xx年					
			x月	x月	x月	x月	x月	x月	x月	x月	x月	x月
1	出来高計画値（PV）（人日）	累計値(a)	6	102	202	349	549	769	1019	1169	1269	1319
		月別	6	96	100	147	200	220	250	150	100	50
2	出来高実績値（EV）（人日）	累計値(b)	6	92	182	302	452	622	822			
		月別	6	86	90	120	150	170	200			
3	投入実績値（AC）（人日）	累計値(c)	6	102	202	352	562	782	1042			
		月別	6	96	100	150	210	220	260			
4	スケジュール効率指数（SPI）	EV ÷ PV	1.00	0.90	0.90	0.87	0.82	0.81	0.81			
5	コスト効率指数（CPI）	EV ÷ AC	1.00	0.90	0.90	0.86	0.80	0.80	0.79			
6	予想総コスト（EAC）（人日）	BAC÷CPI	1319	1462.4	1463.9	1537.4	1640	1658.3	1672			
7	残コスト（ETC）（人日）	（BAC－EV）÷CPI 又は EAC－AC	1313	1360.4	1261.9	1185.4	1078	876.29	630.02			

表 7-9
進捗状況表の記載項目

記載項目	記載内容
①出来高計画値（PV）	当月及び当月までの累計の出来高計画値（PV）を記載する。
②出来高実績値（EV）	当月及び当月までの累計の出来高実績値（EV）を記載する。
③投入実績値（AC）	当月及び当月までの累計の投入実績値（AC）を記載する。

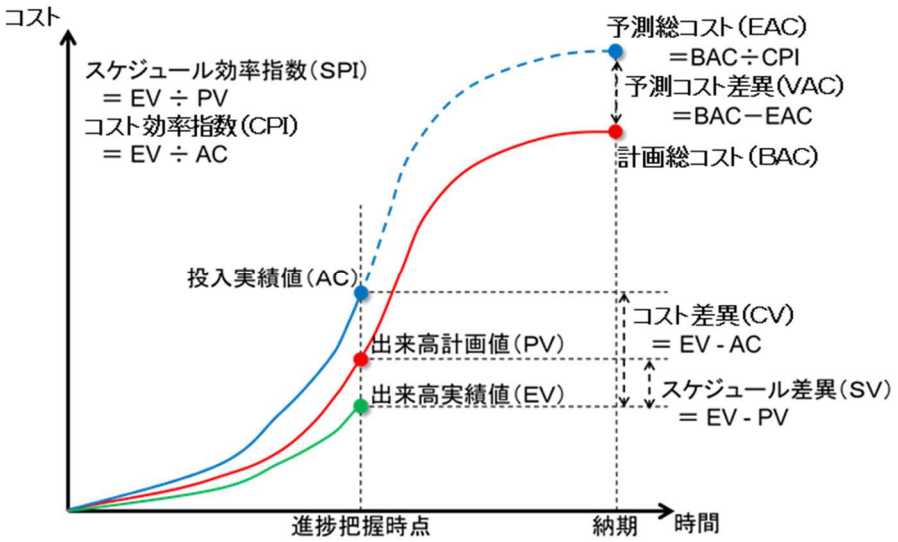
記載項目	記載内容
④スケジュール効率指数（SPI）	当月までの累計の出来高計画値（PV）に対する累計の出来高実績値（EV）の比率を記載する。 SPIが1を超えている場合は、スケジュール以上に進んでいることを示し、SPIが1を下回っている場合は、スケジュール遅延が起きていることを示す。 SPIが1を下回っている場合は、手順の見直しによるコスト効率の向上や人員の増加による投入コストの増加によりスケジュール遅延を回復することができる可能性がある。ただし、人員の増加による投入コストの増加を行った場合は、コスト効率の低下が起きることが多い。
⑤コスト効率指数（CPI）	当月までの累計の投入実績値（AC）に対する累計の出来高実績値（EV）の比率を記載する。 CPIが1を超えている場合は、作業効率が高まっていることを示し、CPIが1を下回っている場合は、作業効率が下がっていることを示す。 ある時点でCPIが1を下回っている場合は、その後の工程で開発手順や開発効率を改善させない限り、完成時において予定コスト内でプロジェクトを終了することは難しいことが多い。
⑥予測総コスト（EAC）	当該時点で予測される完了までの総コストを記載する。差異が今までと同様に推移する場合は、計画総コスト（BAC）をコスト効率指数（CPI）で除算することにより、完了時に予測される総コストについて算定することができる。 EACがBACを超える場合は、計画より多いコストで完了することが見込まれることを示し、EACがBACを下回る場合は、計画より少ないコストで完了することが見込まれることを示す。
⑦残コスト（ETC）	予測総コスト（EAC）と投入実績値（AC）の差を記載する。 ETCが完了までに投入可能なコストを超える場合は、完了遅延が見込まれることを示す。またコスト差異が今後も発生する場合には、コスト効率を考慮することも必要である。

(3) EVM推移グラフ

EVM進捗管理表を集計し、出来高計画値（PV）、出来高実績値（EV）及び投入実績値（AC）の推移についてのグラフを作成し、予測総コスト（EAC）、予測完了時点等を分析します。

本様式は、「(2) 進捗状況表」と合わせて月次の進捗状況の定量分析に用いるものです。次に示す例は、スケジュールが遅れている上にコストも超過し、体制強化や品質向上施策等の対応を至急行うべき状況を表すものです

（様式及び記載例）



← 図 7-8
EVM推移グラフ

記載項目	記載内容
①出来高計画値（PV）	各月の累計出来高計画値（PV）をグラフ化する。
②出来高実績値（EV）	各月の累計出来高実績値（EV）をグラフ化する。 出来高実績値（EV）と累計出来高計画値（PV）の差がスケジュール差異（SV）となり、グラフ作成時点の作業量前倒し（SV＞0）／遅延（SV＜0）を表す。
③投入実績値（AC）	各月の累計投入実績値（AC）をグラフ化する。 出来高実績値（EV）と投入実績値（AC）の差がコスト差異（CV）となり、グラフ作成時点の計画時点からの作業コストの増加（CV＜0）／減少（CV＞0）を表す。
④予測総コスト（EAC）	グラフ作成時点の予測総コストをグラフ化する。 予測総コスト（EAC）と計画総コスト（BAC）との差が予測総コスト差異（VAC）となり、グラフ作成時点の計画時点からの作業総コストの増加／減少を表す。 予測総コスト（EAC）と投入実績値（AC）との差が残コスト（ETC）となり、グラフ作成時点の作業完了までにかかるコスト見込みを表す。
⑤予測完了時点	グラフ作成時点の予測完了時点をグラフ化する。 予測完了時点と納期の差が納期からの前倒し／遅延の見込みを示す。

↵ 表 7-10
EVM推移グラフの記載項目

(4) 進捗状況分析図

進捗状況表を集計し、スケジュール効率指数（SPI）、コスト効率指数(CPI)の推移についてのグラフを作成し、スケジュール及びコストの効率の傾向を時系列で分析します。

本様式は、月次の進捗状況の定量分析に用いるものです。次に示す例は、品質悪化が進み、スケジュール変更等の抜本的な見直しが必要な状況を表すものです。

（様式及び記載例）

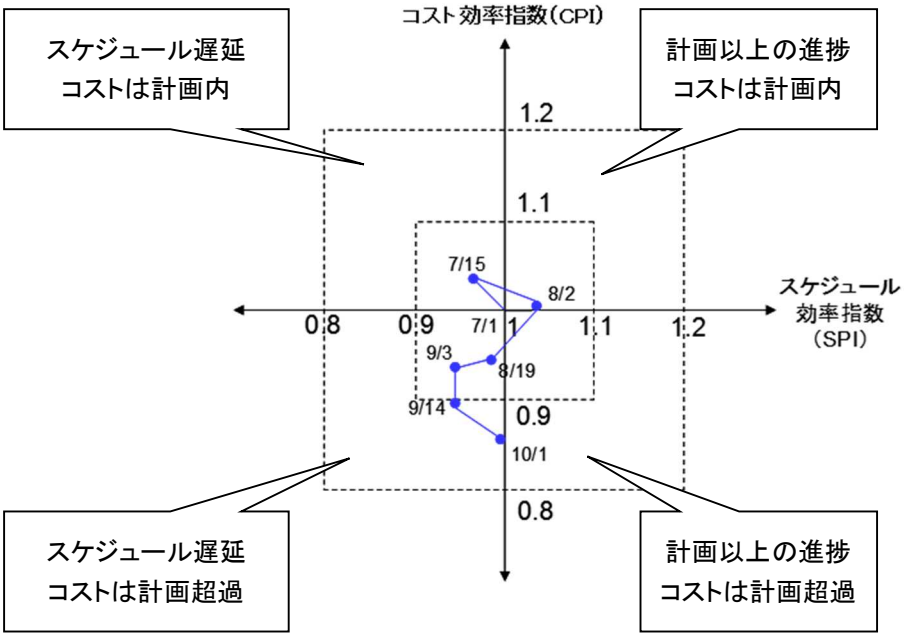


図 7-9
進捗状況分析図

- スケジュール効率指数（SPI）
  各月のスケジュール効率指数（SPI）をグラフ化する。右に行くほどスケジュール効率が高まっていることを示す。

- コスト効率指数（CPI）
  各月のコスト効率指数（CPI）をグラフ化する。上に行くほどコスト効率が高まっていることを示す。

上図の例では、7/1 から作業が始まっています。

ところが、9/3 や 9/14 頃には SPI も CPI も1を下回りました。つまり、スケジュール面では計画から遅れ、コスト面でも予定を超過（予定よりもコスト効率が低い）した状態となっています。

その後、10/1 の段階では SPI が 1 に戻りました。つまり、スケジュール面では計画どおりに追いついたということです。一方で、CPI はさらに悪化しています。当初想定以上に要員を投入したのではないかと、読み取ることができます。

(5) EVMの使い方

ここまで説明したように、EVM手法はプロジェクトの進捗度合いを管理していくための有用な手段です。端的に言えば、「進んでいるか遅れているか」をわかりやすく表現することが可能です。ただし、計画に描いたWBSそのものに抜け漏れがあったり、潜在的な課題を見過ごしたりしていれば、EVM管理上では問題がなさそうなプロジェクトにおいてもある日突然進捗が進まなくなることがあります。また、業務関係者の調整とコンセンサスが不十分であれば、開発終盤にどんでん返しのように手戻りが発生する懸念もあります。

こうしたことから、PJMOとしてはEVMの指標値を有力な手掛かりとしつつも、継続的にWBSの妥当性や関係者間の業務調整の状況にも気を配っていくことが肝要です。

３　テストの計画を立てる

【標準ガイドライン関連箇所：第3編第7章第4節 6)】

情報システムの設計・開発では、品質の管理が重要であり、そのためには十分なテストが必要だ。ということは、よく耳にすると思います。しかし、そもそも「品質の良い情報システム」とは、どういうものでしょうか？また、設計・開発では様々なテストを行いますが、何が違うのでしょうか？

ここでは、効果的なテストを行いながら品質を管理してより良い情報システムを構築していくためのテストの計画に関する基本的な考え方、知識、ノウハウをご紹介していきます。

A. Ｖ字モデルと発注者・委託先事業者の役割分担を把握する

現在、ウォーターフォール型の開発プロセスではV字モデルが一般的です。開発プロセスには各種の国際標準や国内標準もありますが、「標準ガイドライン」の工程定義にのっとると、次のように表現できます。

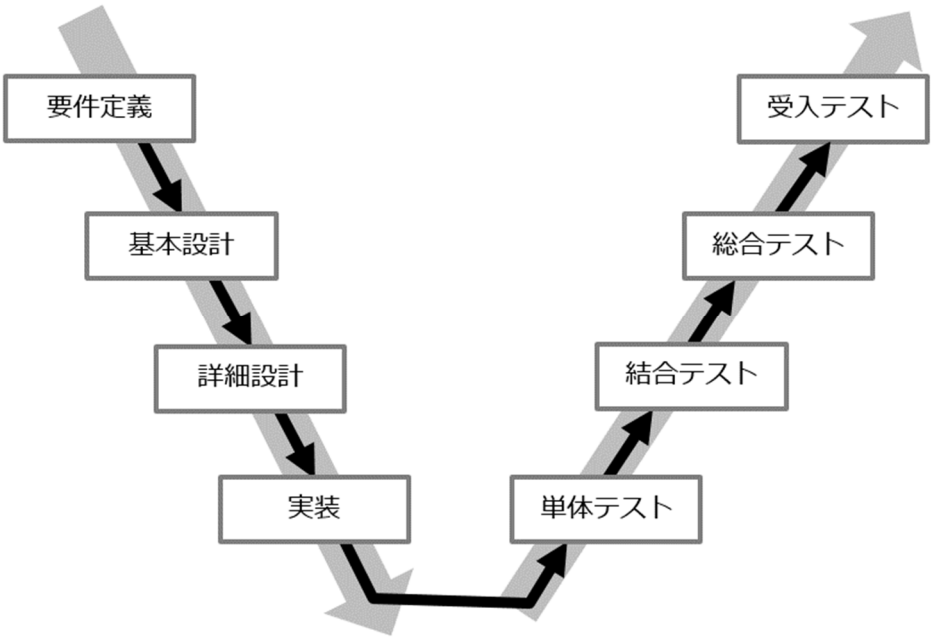


図 7-10
標準ガイドラインの定義に則ったソフトウェア開発プロセスのＶ字モデル

直線的に進んでいく工程を、あえてV字にしていることには意味があります。同じ高さにある工程が、それぞれ深く関係しています。例えば、総合テストとは基本設計で定めた要件が充足されているかを確認するテストであり、受入テストとは要件定義との充足性を確認するテストということです。

また、工程によって発注者側（PJMO）と委託先事業者の役割は変わりますし、作業の主体も変わります。このことを非常に大雑把に表すと、次のような図となります。

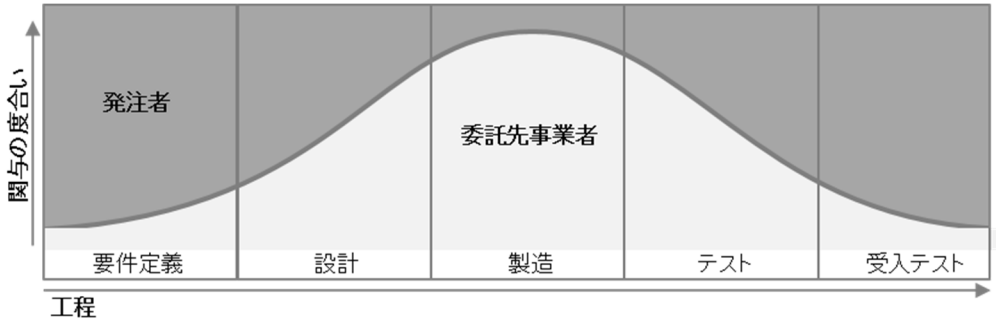


図 7-11
発注者と委託先の関わり度合の変化

多くの人は、図の左側にある要件定義や基本設計は両者で協業して作り上げていく認識を持っています。一方で、図の右側にあるテスト工程については発注者が関与するという認識を持っていないこともあります。

テスト工程において、発注者側にはテスト計画を確認し、テスト実施状況を管理し、テスト結果を評価するという重要な役割があります。特に、総合テスト以降の工程終盤になればなるほど発注者側の関与が重要であり、受入テストは発注者自身が実施するものであることを覚えておいてください。

B. テストのレベルや種類を理解する

さて、単体テスト、結合テスト、総合テスト、受入テストの4種類の工程があることを説明しましたが、これらの工程は何が違うのでしょうか。

情報システムのテストは、段階的に進めていきます。個々のプログラムが設計書どおりにできているか？プログラムをつなげて機能としてみたときに、機能の設計を満たしているか？機能同士をつなげてみたときに、要件を満たしているか？要件どおりにできたが、業務が適切に遂行できるか？このように、徐々に確認するレベルを上げていくのです。

これは、まさにV字モデルが表していることです。標準ガイドラインで定義しているテスト工程では、次のように整理しています。

表 7-11
テスト工程

テスト工程	概要	発注者の関与の仕方
単体テスト	アプリケーションを構成する最小の単位で実施するテストであり、主に機能単位で設計通りに動作するかを事業者（プログラマ）が確認する。	事業者がテストの実施主体ではあるが、発注者もテスト計画を確認した上で、実施状況の報告を求め、報告書に記載されている実施結果に不足、誤り等が発生している場合は、課題等を整理し、指摘又は指導を行う。
結合テスト	複数の機能を連携させて動作を確認するテストであり、主にユースケース単位で設計通りに動作するかをテスト担当者が確認する。	（同上）
総合テスト	システム全体が設計の通りに動作することを確認するテストであり、ユースケースを組み合わせた一連の業務が行えることを機能面や非機能面の観点からテスト担当者が確認する。	上記に加えて、テストシナリオやテスト評価方法の妥当性を確認し、過不足を指摘することで抜け漏れの無いテストの内容になるように関与する。
受入テスト	納品されるシステムが要件通りに動作することを確認するテストであり、発注	発注者が主体となりテストを実施する。実際の利用者がテストに参加すること

注記
ユースケースとは、特定の目的を達成するためにアクタ（ユーザ）が実施する手順等を定義したもの。

テスト工程	概要	発注者の関与の仕方
	者が主体となり、事業者と協力して確認する。	で、サービス・業務が円滑に実施できることを確認する。 事前に要件を十分確認できるテストシナリオかを確認し、実際にテストシナリオに基づき情報システムを操作し、テスト結果が要件どおりであることを確認する。

大きな分類は、上記のとおりですが、それぞれのテスト工程の中でも異なる種類のテストを実施します。それらの詳細は、Step.4 で触れていきます。

また、テスト工程とは別に、テスト手法の違いがあります。これらの内容は、事業者がテストの実施方法をPJMOに説明する際に使用されるため、知識として知っておくと理解しやすくなります。

表 7-12
テストの種類

テスト手法	概要
ホワイトボックステスト	ホワイトボックステストとは、プログラム（ソースコード）の内部構造、論理構造を理解した上でその構造どおりに実装できているかを確認するテストです。中身が見えている状態で行うテストなので、ホワイトボックスと呼んでいます。プログラムを「作る」人の目線でのテストともいえます。 基本的に、上述のテスト工程のうちホワイトボックステストを実施するのは単体テスト工程です。 ホワイトボックステストでは、ソースコードがテストされた割合を示す「カバレッジ（網羅率）」が重要な指標となります。しかし、カバレッジには主として3つのレベルがあるので、どのカバレッジレベルを前提としているかについて注意が必要です。 （参考）カバレッジの種類 C0　命令網羅率：プログラム内の命令文をどの程度網羅したか （単純に、通過したステップ数のカバー率を示す） C1　分岐網羅率：プログラム内の分岐をどの程度網羅したか （if 文であれば then と else の両方をカバーしたか） C2　条件網羅率：プログラム内の条件をどの程度網羅したか （if 文の条件に and や or があれば全ての組み合わせ分） なお、仕様自体の間違いや機能が備わっていないバグなどはホワイトボックステストでは検出できません。また、カバレッジは必ずしも 100%を目指す必要はありません。むしろ、100%に近づくほど等比級数的にコストが増大するので、適切にカバレッジを定める必要があります。
ブラックボックステスト	ブラックボックステストとは、プログラムの内部構造、論理構造に着目するのではなく、プログラムの入出力に着目します。つまり、プログラムの外側から見たときに仕様どおりに動作するかを確認するテストです。中身が見えない状態で行うテストなので、ブラックボックスと呼んでいます。プログラムを「使う」人の目線でのテストともいえます。 基本的に、ホワイトボックステストの完了後に、様々な粒度や観点からブラックボックステストを実施します。

C. リスクを踏まえてテストの方針を決める

テストの計画を立てる以前に、プロジェクトや情報システムの特性を見極めて、どこに深刻なリスクが存在するかを分析します。

例えば、統計システムと請求支払システムでは端数誤差に対するリスク度合いが異なる

でしょう。請求支払いシステムでは、1円の誤差でも誤請求として重大な問題になります。このように、リスクを識別することで、初めて機能、性能、セキュリティ、可用性等について当該情報システムがクリアしなければならない重要な要件をあぶり出すことができ、テストを重点的に実施すべき対象を特定できます。

このようなリスク分析は、事業者に丸投げすることはできません。情報システム特性は多様なので、様々な経験・スキル等を持った「人」によってその見え方が異なります。それはデータベースやネットワークという技術要素のことだけでなく、主に過去の障害事例やレアケースを経験しているようなことを指しています。利用者、情報システム運用者、ヘルプデスク等の関係者によって見えてくる風景が異なるので、特に改修案件の場合はこれらの人にヒアリングをして、情報システム特性やリスクを特定してください。

D. テストにおける役割分担と必要な環境を明確にする

他の情報システムと連携を行う場合（特に複数の情報システムがデータを連携する場合）は、役割分担を具体的に定めることがとても重要です。これらを事前に決めておかないと、いざテストを行う時期になってもめることになり、テストの開始が遅れて十分なテストが行えず、本番稼働を行ってから不具合が多発することにもなりかねません。

他の情報システムと連携を行う場合は、次に示す事柄を計画時点で明確にして、関係者間で合意しましょう。

他情報システム連携に関する計画の注意点

- 他情報システム連携を確認するテスト（主に総合テスト）の実施主体（誰が音頭を取るか）、各情報システムの窓口となる担当者、計画の作成者を決める。
  なお、高度な調整を要する場合は、プロジェクトの対外調整の役目を有するプロジェクト推進管理者が実施することとなる。
- どの段階で何を確認するかを明確にする。例えば、結合試験段階では、サンプルデータを授受し、事前にインタフェースの確認を行う等、段階的に品質を高めていく方法もある。また、連携テストを行う環境の疎通テスト、本番環境の疎通テストはいつ行うか等、環境や観点ごとに漏れなく計画する。
- テストデータは、誰がどの範囲のデータを作成するかを決める。基本的には出し元が作成することになるが、どのようなデータのバリエーションのテストが行いたいかは受け側と調整する必要がある点にも注意する。
- 連携テストはどの環境で行うか。環境ごとの疎通テストの実施有無、実施時期を決める。特に総合テスト以降は、様々なテストが同時並行で行われるため、環境の取決めはしっかり行うことに注意する。

E. テストツールを有効活用する

近年、情報システムの品質を向上させるためのツールは多く登場しています。これらを活用することで、設計・開発の活動を効率的に進めたり、効果的に品質を担保・向上させたりすることができます。事業者とも相談しながら、導入を検討してみてください。

表 7-13
テストツール

ツールの種類	概要	メリット
ソースコードの静的解析ツール	ソースコードから、機械的にコード規模（コード行、スペース行、コメント行等）、複雑度、複製度/重複度、正当性、セキュリティ観点からの好ましくない行、パターン等を機械的に抽出するツール。	静的解析ツールは、ソースコードレビュー（インスペクションとも言います）を助け、コード品質の向上、レビューワの負荷軽減、期間短縮に効果を発揮します。 コード特性を可視化することができることから、全体を俯瞰しながら個々の問題や指摘箇所について検討することができます。このため、プログラマはツール結果を見ながら自分で問題点を検討し、修正することもできます。一人では解決できない場合も、レビュー時にレビューワにツール結果を見せることにより、レビューワも問題の特定が容易となり作業負荷の軽減、時間の短縮にもつながります。
自動テストツール	ソフトウェアテストを行うための作業（テストケースの設計、テストの実行と結果の確認、テストの進捗管理、レポートの作成）又はその一部を自動化するツール。	効率良く自動テストを実行するよう、スケジューリングすることで、手動でのテスト工数を削減することが可能です。
継続的インテグレーション	コンパイル・テスト・デプロイといったソフトウェア開発のサイクルを頻繁に繰り返し実行する手法。	短期間で品質管理を行うため、問題の早期発見や開発の効率化が可能です。
タスク管理ツール	プロジェクト全体のタスクを管理することができ、進捗の見える化や共有化などにより、タスクを管理しやすくするツール。	タスクのツリー構造を定義し、整理することができます。また、タスクの順序や優先度合いを設定し、スケジュール管理をすることができます。スケジュールや進捗具合を、自動でガントチャートなどのグラフ化で表現でき、直感的に状況を把握することができます。

Step.4 設計・開発・テストの管理

設計・開発の実施計画が完成したら、いよいよ設計・開発の作業が始まります。PJMOは、事業者との定例会議で進捗状況や課題等を確認していくことになりますが、報告を聞いているだけでは、後々トラブルを招きかねない問題を早期に発見することはできません。

ここでは、問題を早期に発見・対応して品質の良い情報システムを構築していくための知識やノウハウについて説明していきます。

1　設計内容を確認・調整する

【標準ガイドライン関連箇所：第３編第７章第５節】

事業者は、設計作業において、要件定義の内容を具体化・詳細化し、設計書を作成します。設計書は、専門的な内容も多く、分量も非常に多くなります。全ての設計書を念入りに確認する時間があればよいですが、なかなかそんな時間は取れないでしょう。しかし、ポイントを押さえて設計書を確認して、必要な指摘や調整を行えば、後々のトラブルを避けて円滑にプロジェクトを運営していくことができます。

A. 基本設計の内容を確実にレビューする

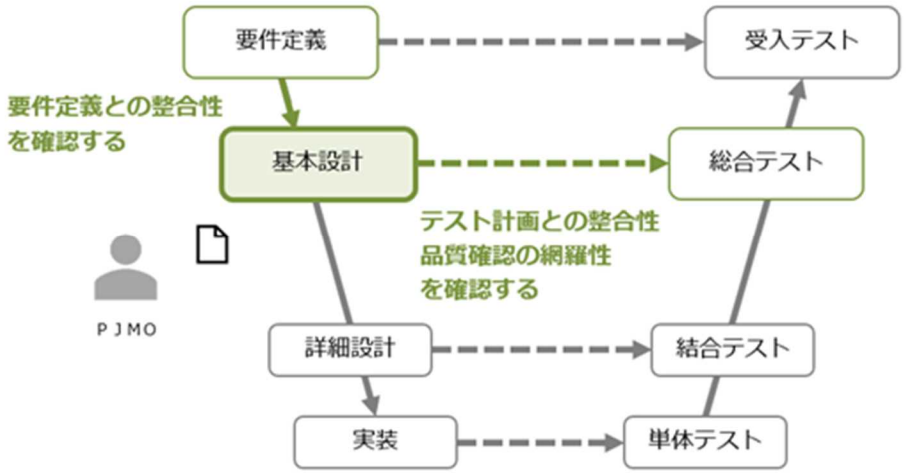
設計書に書かれていることと本来の要件との間にかい離がある場合、その事に気づくのが遅れるほど、修正に要するお金と労力が増えていきます。設計書は、要点を押さえ対象を絞る等の工夫をしながら必ず内容をレビューしましょう。

設計書のレビューは、基本的に「基本設計」で作られた成果物を対象とします。これは、発注者側で用意した要件定義書と事業者の作成する成果物の界面になるからです。基本設計以降は、基本設計に基づいて詳細設計や実装等が行われるため、それらの整合性を確認するのは基本的に事業者の責任範囲となります。レビューを行う際の観点を以下に示します。

レビューの観点

- 設計書と要件定義書との整合性、設計書間での整合性が取れているかを確認する。
- テスト計画での確認内容と設計書の内容の整合性が取れているか、要件定義で示されている内容がテスト計画で網羅されているかを確認する。

図 7-12
設計レビューの観点イメージ



コラム 7-1
忘れがちな突合作業

コラム：忘れがちな突合作業

たくさんの設計会議を経て最終的に設計書を確認するタイミングでは、今まで設計会議で詰めた内容を思い出しながらレビューすることが中心となってしまいがちですが、設計会議で議題にならなかった内容について反映を忘れてしまうことがあります。

当初の要件定義で定めた項目が漏れなく設計書に反映されているか、レビューの大前提として再確認する必要があります。これは、開発等の後工程における要件漏れによる手戻りを防止し、イレギュラーな開発コストの増大を抑えるために極めて重要です。

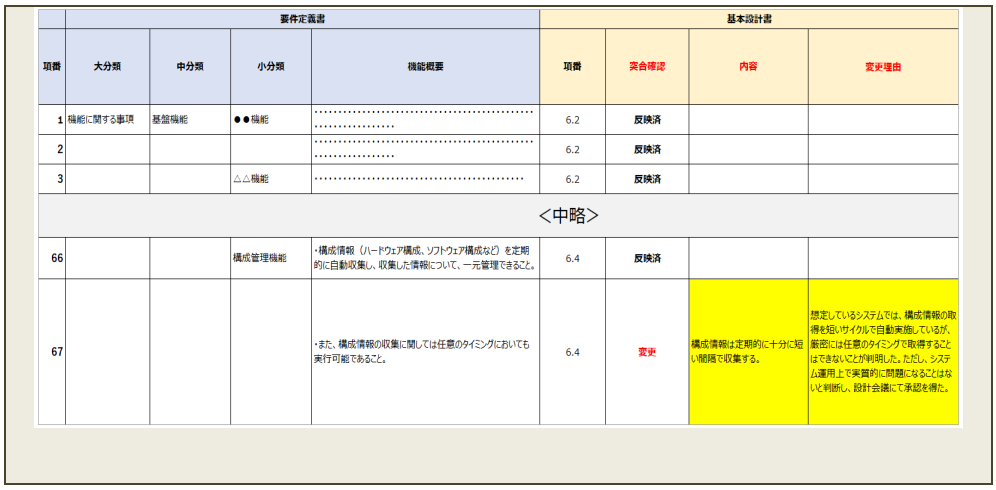
設計事業者によっては、プロジェクト管理ツール等で要件定義と設計内容の突合作業を実施しているところもあるので、PJMO としてもその内容を必ず確認するようにしましょう。

ツールでなくとも、Excel 等で突合確認することもあります。

実際の突合資料の一部を示します。

下表は RTM(Requirements Traceability Matrix)として整理した例です。下表の左側には要件定義書で記載した項目全てを要件単位で列挙しています。一方右側には作成した設計書案で要件に対応する部分があるかどうかチェックし、要件が正確に反映されていた場合は「反映済」としています。また、要件から変更があったものについては、その変更理由を明確にするとともに、変更することをプロジェクトとして承認したかについても確認します。

このような突合作業を行うと、要件に記載していた事項が設計に反映されていないという「漏れ」に気づくことができます。



	要件定義書				基本設計書			
項番	大分類	中分類	小分類	機能概要	項番	突合確認	内容	変更理由
1	機能に関する事項	基盤機能	●●機能	……	6.2	反映済		
2				……	6.2	反映済		
3			△△機能	……	6.2	反映済		
<中略>								
66			構成管理機能	・構成情報（ハードウェア構成、ソフトウェア構成など）を定期的に自動収集し、収集した情報について、一元管理できること。	6.4	反映済		
67				・また、構成情報の収集に関しては任意のタイミングにおいても実行可能であること。	6.4	変更	構成情報は定期的に十分に短い間隔で収集する。	想定しているシステムでは、構成情報の取得を短いサイクルで自動実施しているが、厳密には任意のタイミングで取得することはできないことが判明した。ただし、システム運用上で実質的に問題になることはないと判断し、設計会議にて承認を得た。

B. 他の情報システムとのデータ連携には細心の注意を払う

情報システムの多くは、他の情報システムとデータ連携を行います。そして、このデータ連携では、高い確率で様々な問題が発生します。以下に、他の情報システムとの連携における代表的な問題を示します。

データ連携における代表的な問題

- 処理タイミングにずれがあり、データが必要な時点までにデータ連携が間に合わない。
- 項目、内容、定義にずれがあり、データが連携されても双方でデータ項目に対する認識が異なる。
- 過去データの扱いに差異があり、最新情報だけが連携されて過去データが連携されない場合がある。履歴データの保持期間が双方で異なる、等。
- データ連携は可能であるが、大量データのやり取りに許容範囲を超える処理時間がかかる。

これらの問題を起こさないためには、まずは、他の情報システム側の担当者等との協力体制を築くことが第一です。それを踏まえた上で、次の点に注意して設計・開発を進めていきましょう。

データ連携での問題発生を防ぐために

- 他の情報システムの担当者や事業者と定期的なコミュニケーションが図れるように、計画段階で役割を明確化し、情報共有や調整の仕方を定めておく。
- データ連携仕様やインタフェース設計については、早期に作成し、関係者と共有し、十分な相互レビューを経て合意する。また、なおこれらの仕様を確定する時期については、重要なマイルストーンとしてスケジュールに組み込み、入念に進捗管理を行う。
- 既に他の情報システムでデータ連携仕様が決定している場合は、その情報を早い段階で受領する。
- 他の情報システムとのテストは、早いうちから段階を踏んで実施する。例えば、

データのバリエーション試験は、結合テストの段階で他の情報システムからデータを受領して行う等、工夫する。
また、テストを実施するためには、基本的に本番環境とは別のテスト環境が必要となるので、その準備や調整を早期から行う。

2　品質管理の考え方を理解する

【標準ガイドライン関連箇所：第３編第７章第５節】

PJMOは、設計が完了した後は、テストを通じて品質を確認していくことがメインになっていきます。テスト工程に応じた品質確認の知識やノウハウは、以降で示していきますが、ここでは、テスト工程全体を通じて大切な品質管理のポイントを説明していきます。

A. 見えない品質を見える状態にする

テストを実施する中では、多くの不具合が発生します。それらの不具合のことを「障害」とし、障害管理を行っていきます。

適切な品質は適切なプロセスによって作り込まれます。信頼度成長曲線は、バグの収束傾向を見極めるものですが、同時に進捗と品質も見ることで、プロセスも評価します。下図の青の実線は「テストケース残数」、青の点線は「テストケース見込残数」、オレンジの実線は「障害累計件数」、オレンジの点線は「障害見込件数」、緑の実線は「障害解決累計件数」です。

図 7-12 の青い実線を見ると、きれいな逆S字カーブでテストケースを順調に消化していたことがわかります。

オレンジの点線（障害見込件数）は、テスト実施前に正確に見積ることは難しいでしょう。一般的には、単体テストや結合テストではシステム全体での障害摘出（件数／KStep）を過去の経験等から設定し、テスト実施対象のステップ数と掛け合わせることで求めることが多いです。一方、総合テストではテストシナリオ単位で個別の障害見込件数を予測することが難しくなるので、システムの全体規模（KStep）に対して障害見込件数を計算した上で、最終的な収束見込を管理することもあります。

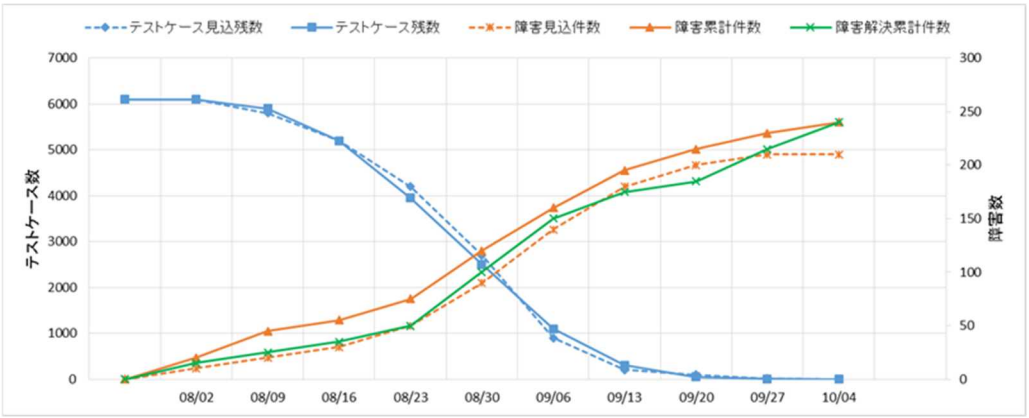


図 7-13
信頼度成長曲線のイメージ

障害累計件数の実績が、計画に対して下振れしていた場合に、考えられることは 2 つあります。1 つは、システムの品質が想定より良いという好ましい状態、もう 1 つは、テストの実施方法が甘く、十分に障害を摘出できていない状態です。このように両方の解釈が考えられるため、テストの途中段階で計画と実績のかい離に一喜一憂することにはあまり意味がありません。ただし、計画に対して実績が大きくかい離している場合は、テストの実施方法

が妥当であるかもう一度検討してみてください。

特に重要なのは、最終的にテストを全件消化したときの障害摘出件数が十分か、そして摘出した障害に対して確実に対応が出来ているかどうかです。信頼度成長曲線では、システム全体での障害件数を合計していますが、一部の機能に障害が残っていても見過ごしてしまう危険性があります。機能単位でどのような障害が発生しているかを評価する方法については、後述します。

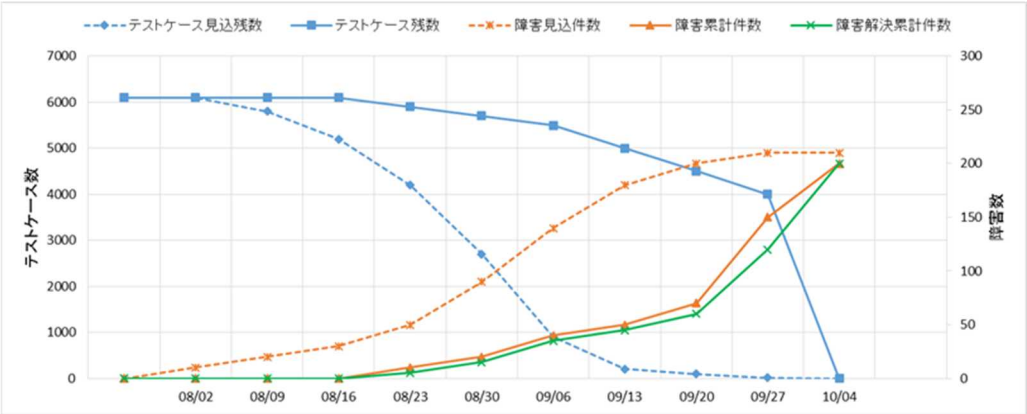


図 7-14
信頼度成長曲線（悪い例）

図 7-13 は、最後1週間でテスト消化を駆け込んだように見え、手間が掛かる複雑な条件や異常系テストが不十分な可能性があります。

こういった場合は要注意であると知っておく必要があります。

信頼度成長曲線を見るポイントは、①テストケースがムラなく消化され、②不良が滞留せずに順調に消化され、③終盤には障害累計曲線が寝ている（摘出障害がほとんど増加しない）ことです。

◆ 障害原因について、納得ゆくまで説明してもらう

発見した障害については、対応状況を確実に追いかけましょう。

対応完了として一度クローズした障害がまた再発することもあるので、障害については一回きりの報告書ではなく、図 7-14 のような継続管理できる一覧表形式とすることが良いでしょう。この一覧表では、障害発生日、環境等に続けて、障害内容、原因（真因）、対応を中心に記述しています。

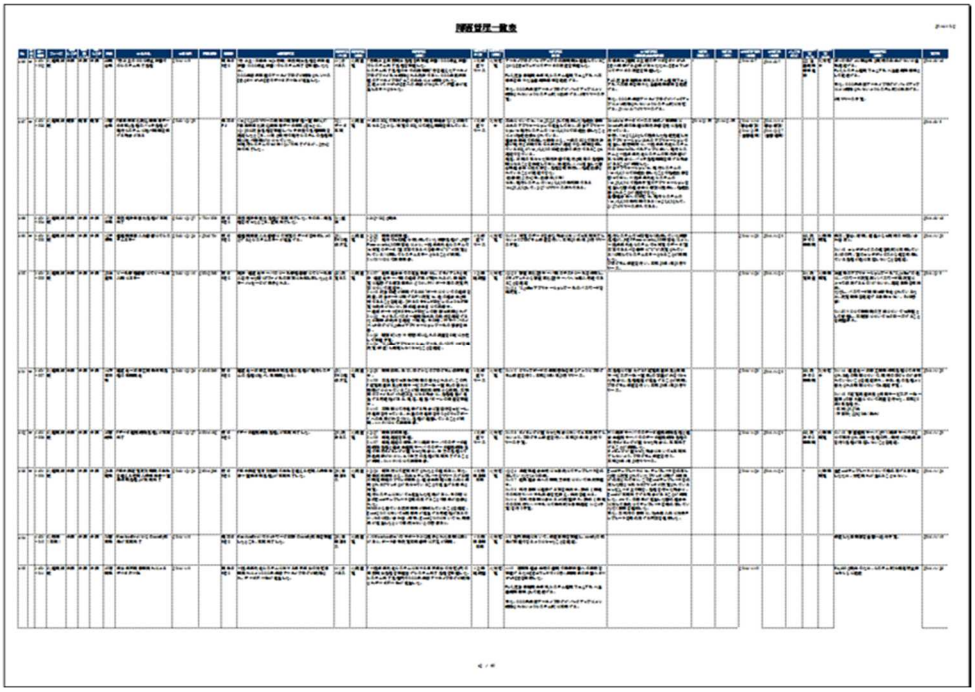


図 7-15
障害管理一覧表の例（イメージ）

「障害原因」については、表層の理由だけでなく、深層の原因（真因）にまで深掘りして調査することが重要です。深掘りしていった結果、アプリケーションの不具合だと思っていた事象が、OSやDB、ミドルウェア、あるいはハードウェア等、異なる領域へ広がっていくことはよくあることです。事業者をまたいで調査することも多く発生しますので、発注者側が対応作業をコントロールし、このような横断調査が円滑に進むように調整することも重要です。

また、必要に応じて、PJMO の職員自身がハードウェアやミドルウェアの製品供給元の会社等から直接に話を聞くことも有益です。障害等の発生原因についてより深い理解を得ることができますし、対応策についても専門的観点から助言を受けることができます。

3　単体テスト・結合テストの品質を評価する

【標準ガイドライン関連箇所：第３編第７章第５節】

A. 単体テストの留意点

単体テストは実装（＝コーディング）と一体的に行われるのが実態です。

テスト管理では、過負荷とならないようにマネジメントに留意します。単体テストは開発者が自ら試行錯誤しながら実施するので、不具合件数は過少報告されがちです。その結果、管理ポイントはむしろ進捗管理であり、品質面ではソース規模とテストケース管理です。不具合報告を正直にあげることには勇気がいるので、くれぐれもそれを非難したりすることのないようにしましょう。

テスト評価では、基本的に単体テストが委託先の責任領域であるため、発注者側が単体テストに対して評価できることは限られています。それでも丸投げとならないよう、①静的解析ツール、②ソース規模測定ツール、③カバレッジ測定ツールの結果を、メソッドやモジュール単位ではなく、次の結合テストの単位である画面、バッチごとにグルーピングして評価します。また、定量評価の「ゼロ」と「100」も危険です。「バグゼロ」は高品質ではなく、テスト不十分と考えるべきです。さらにカバレッジ 100％も、通常はソースの中に起こり得ないよう

な異常処理や例外処理などが必ず含まれているので、一旦は疑ってツールの出力結果などエビデンスの提示を求めることをお勧めします。ここで必要なことは、「100％やりました」という字句をそのまま信じることなく、ツールが出力した結果をエビデンスとして確認することです。

◆ 単体テストの品質までは、発注者側はチェックできない

単体テストはそもそも、発注者側にとっては把握する必要はありません。それは事業者が掌握すべき事項であり、発注者が認識するのは結合テスト以降が妥当です。発注者側で把握する必要がない理由は、単体テストはプログラマ個人がテストし、かつクラスやメソッド単位で技術的視点に終始しているからです。

さらにアジャイル型開発でベストプラクティスとされているテスト駆動開発（テストファースト）では、図 7-15 のようにテストしながらコードを作りこむため、そもそもバグをカウントすること自体が難しくなります。

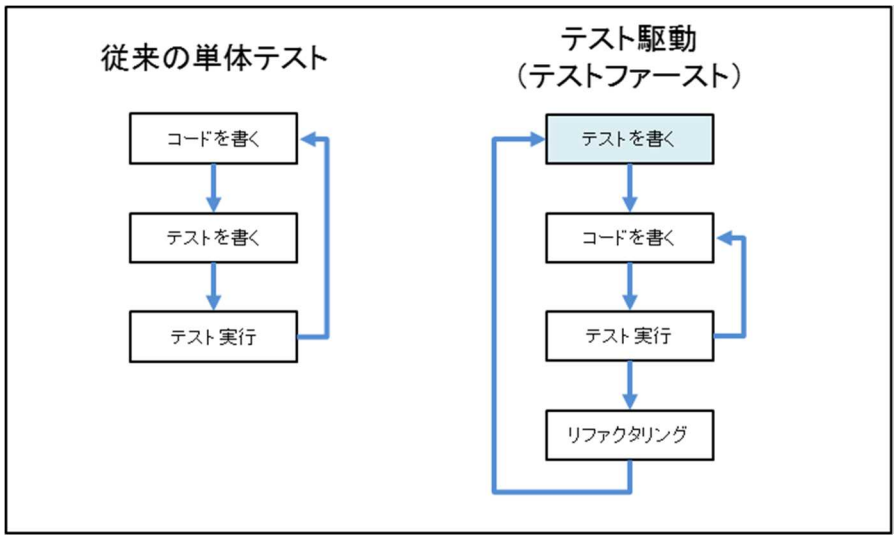


図 7-16
テスト駆動開発（テストファースト）のプロセス

◆ コードカバレッジの確認ポイント

単体テストでは、コードカバレッジの計測が重要視されます。コードカバレッジとは、命令・分岐・条件の網羅率を指す考え方です。ただし、絶対に 100％とする必要は無く、システムの特性と費用対効果を踏まえて、目標カバレッジを設定しましょう。

B. 結合テストの留意点

結合テストは事業者が主体となって実施する工程ですが、発注者もテスト計画、テスト管理状況、テスト結果等については積極的に確認しましょう。結合テストは、画面単位、ジョブ単位で機能ごとに結合させてテストします。特にシステム改修の場合は、いきなり画面やジョブに組み込んで「単体テスト」と称する場合があるので、本来やるべき単体テストを飛ばしていないか注意します。一般論として、結合テストの粒度は5～20 個ぐらいの単体モジュールの結合となります。それ以上の単体モジュールを結合テストで一気にテストしようとすると品質評価が難しくなるので、全体規模が大きい場合には次の総合テストの前に「結合テスト2」としてもう一段テスト工程を区切ると良いです。

テスト内容を設計する中では、①正常系だけでなく、②エラー・異常系、③例外的、縮退などの特殊パターンまで含めるようにします。経験上、多くのバグは②エラー・異常系の周辺に偏在しているものです。一方、③例外的、縮退などの特殊パターンはテストが面倒なので、後回しになったり抜け落ちたりしがちなので注視するようにします。また、全ての組み合わせをテストするのではなく、効率的なケース設計を推奨するようにします。これはテスト工数を削減するためではなく、テスト結果の見落としを防ぐことが目的です。結合テストでは、データベースやファイルへの入出力、ログ出力、暗号化／復号化、同時アクセス・排他制御など、業務目線だけでなくシステム目線の観点についても網羅されているかチェックします。これについてはチェックリストとして整備しておくのが望ましいです。

テスト実施では、始めに結合したモジュール疎通テストを実施後、コンポーネント間のインタフェースを念入りにテストし、その後、①正常系、②異常系、③特殊パターンの順にテストケースを消化します。

テスト管理では、進捗と品質の両面が重要となる局面なので、信頼度成長曲線で管理することが望ましいです。進捗面では、特に立ち上がり時のケース消化状況と停滞した時の原因を丁寧にヒアリングします。品質面では、「どこに」不具合が偏在しているかと、その箇所に品質強化テストを課す必要があるかを見極めます。「問題なし」というのが一番心配なので、あらかじめ「ワースト順に全体の1割に該当するモジュールは品質強化テストを課す」というようなルールを決めておくと良いでしょう。

◆ 結合テストで単体テストと同じことをしてはいけない

単体テストは結合テストの括りで整理し、両方のテスト結果を一覧で比較します。テストケース数と障害件数のそれぞれについて、単体テストから結合テストへの推移で品質を判断するのがわかりやすいです。図 7-16 で示すテスト品質メトリクス一覧表はテストの種類ごとのテストケース数を示しており、同様のフォーマットで障害件数も記録します。縦軸（サブシステム単位、機能単位）の粒度を揃えることで、横軸の単体テスト、結合テスト、総合テストと推移する過程でどこのテストケース数が甘いか、どのレベルで障害が多発しているかがわかります。またこれによってテスト項目密度、障害摘出率をサブシステム間で比較することもできます。

	サブシステム名	テストケース数															
		単体テスト				結合テスト					総合テスト						
		C0処理	異常系	境界値	…	画面系	帳票系	バッチ系	外部ＩＦ系	…	シナリオ	業務サイクル	性能	限界	可用性	セキュリティ	…
A	○○○システム																
B	□□□サブシステム																
C	△△△システム																
D	ＸＸＸサブシステム																
E	…																
F	…																
G	…																
H																	
I																	
J																	
K																	
L																	
M																	
N																	
O																	
P																	
Q																	
R																	
S																	
	○○○ツール																
	その他																
	合計																

	サブシステム名	障害件数															
		単体テスト				結合テスト					総合テスト						
		C0処理	異常系	境界値	…	画面系	帳票系	バッチ系	外部ＩＦ系	…	シナリオ	業務サイクル	性能	限界	可用性	セキュリティ	…
A	○○○システム																
B	□□□サブシステム																
C	△△△システム																
D	ＸＸＸサブシステム																
E	…																
F	…																
G	…																
H																	
I																	
J																	
K																	
L																	
M																	
N																	
O																	
P																	
Q																	
R																	
S																	
	○○○ツール																
	その他																
	合計																

図 7-17
テスト品質メトリクス一覧表（テストケース数、障害件数）

◆ テスト項目密度と不良密度の確認ポイント

テストの実施状況を機能単位等で詳細に把握するには、それぞれの機能に対するテスト項目密度と不良密度を見ます。図 7-17 のように、散布図の形でプロットすると、機能毎のテスト状況を可視化することができます。

テスト項目密度と不良密度の両方が許容範囲に収まっていれば問題ありませんが、テスト項目密度が低いもの、不良密度が高いものには注意を払います。図 7-17 の太枠の中に収まっていれば合格です。色のついていない箇所もあまり問題視しませんが、吹き出しのある箇所は念入りにチェックさせる必要があります。ただし、考え方としては、許容範囲から外れたら「不合格」というのではなく、許容範囲に収まっている分には「敢えて説明を求めない」とするのが良いです。例えば、「テスト項目密度・不良密度が下限値を下回っているのは1KLOC未満だから」とか「不良密度が高かったのでテストケースを追加した結果テスト項目密度も高くなった」とか説明ができれば問題ありません。

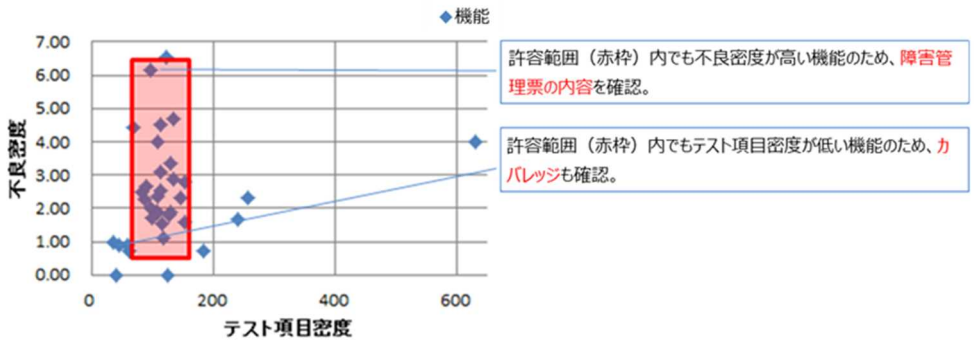


図 7-18
テスト密度と不良密度でチェックすべき箇所

なお、不良密度には真逆の解釈が成り立ちます。1つはこの値が低いことを高品質とす

る見方です。もう1つはこの値が低いのはバグが摘出しきれずに潜伏したままだとする見方です。このどちらかの立場から見るかはツールやテスト手法、要員のスキルや組織の成熟度などから総合的に判断するしかないため、上記のようなプロット図を指し示しながら、開発事業者のプロジェクトマネージャや品質管理者からヒアリングするしかありません。

事業者の品質指標は（当然ですが）許容範囲の幅を大きめにしていると考えるべきです。上限値と下限値は、中央値を設定しておおむねそれの±20～25%くらいが適当と考えられますが、全体の 25～75 パーセンタイルに収まるように調整しても問題ありません。テスト項目密度、不良密度とも、単体テストが最も大きく、テスト工程が進むにつれて指標値は小さくなります。

不良密度については、後工程になるほど小さくなります。単体テスト→結合テスト、結合テスト→総合テストと推移する中で、それぞれ4分の1から5分の1くらいになることが目安の水準といえるでしょう。

最後に重要な点を強調しますが、このような品質管理の指標自体はあくまで目安です。新規構築時と機能改修時で参考とする指標値は5～10 倍ほど変化することもあります。また、機能改修の場合の母体規模をどのように考えるかは、それぞれのシステム特性もあって一概に言えません。

一番重要なことは、このような指標を発注者自身も確認しながら、気づいた点については事業者に確認し、双方で品質を高めていくという活動を継続することです。不具合件数や不良密度といった定量評価で終わらず、具体的にどういう不具合内容で、その原因は何かを追求します。また、見つかった不具合については、同種の誤りがないか、得られた知見の横展開をどのように行ったかまで、根掘り葉掘り聞いていきます。

できれば、週1回程度のテスト状況報告会議を行い、そこで1週間分の不具合管理一覧表を提出してもらい、その内容について逐次説明を聞くようなやり方が良いでしょう。不具合の内容について発注者側が質問することによって、事業者のプロジェクトマネージャはそれを説明できるように配下のメンバーに質問することとなり、品質改善の PDCA が回り出します。発注者側は、決して「物わかりがいい」ふりをする必要はありません。分からないことは、どんどん質問していきましょう。

4　総合テストの品質を評価する

【標準ガイドライン関連箇所：第３編第７章第５節】

A. 総合テストの留意点

総合テストは、システムごとの特性で実施すべき内容も大きく変わります。

総合テストでは、業務観点からのいろいろなシナリオに基づいて機能テストを検証しますが、これに合わせてシステムの性能や信頼性等を検証する非機能テストを行います。非機能テストについては抜け漏れが発生しがちであるため、表 7-13 を参考に確認すべき非機能要件がテストから漏れていないか確認してください。

表 7-14
総合テストの観点とテスト例

観点	観点に沿ったテストの例
機能	業務を実施する手順やデータを基に様々なシナリオ・データのバリエーションを作成し、それらを組み合わせに沿って情報システムを用いて業務や機能を確認する。 シナリオ・データには、日常的によく行う業務や取り扱うデータだけではなく、月次や年次等の特定のタイミングでしか発生しないシナリオや稀にしか発生しないイレギュラデータも含めて確認する。

観点		観点に沿ったテストの例
		また、正常系だけではなく、異常系のテストも行うことで、ユーザの誤操作や予期しない現象をきっかけとしたシステム障害が起きないことを十分に確認することに留意する。 例）シナリオテスト、業務サイクルテスト等
非機能	性能・拡張性	ユーザ数、データ量、リクエスト数、レスポンス等の性能要件を情報システムが満たしているかを確認する。これらは、現在の想定だけではなく、今後の予想される増加量も含めて、確認する。 例）パフォーマンステスト、等。
	可用性	処理量や長時間稼働等のシステム限界に関する性能や拡張の要件を情報システムが満たしているかを確認する。 性能テストや可用性テストと一緒に実施されることもある。 例）負荷テスト（ラッシュテスト、ストレステスト、大容量テスト等）。
		ソフトウェア、ハードウェア、ネットワーク等の障害時の振る舞い、復旧時間、データ復旧ポイント等の可用性要件を情報システムが満たしているかを確認する。 例）可用性（障害）テスト、縮退テスト等。
	運用・保守性	可用性要件として災害対策が求められる場合、情報システム全体切り替えやそれに伴う運用の切り替えの手順や実現性を確認する。 例）災害対策テスト等
		運用監視、バックアップ、パッチリリース等の運用及び保守を実施するために必要な仕組み（ツール・環境等）の確認や、計画停止やリリース等の人手を介する運用業務の手順等を確認する。 例）運用・保守テスト
	セキュリティ	不正侵入や Web 特有の攻撃、DB サーバへの不正アクセスなどに対する対策、データの持ち出しに対する対策、マルウェア（ウィルス）対策等のセキュリティ要件を情報システムが満たしているか確認する。 例）ペネトレーションテスト、インシデントレスポンステスト、ファジング等。
	移行性	移行計画に従って、移行ツール、切り替えの仕組み、移行手順書等が作成されていることを確認する。 例）移行テスト、移行リハーサル等。
	システム環境・エコロジー	電力消費等のシステムの効率性や設置場所の耐震性等が要件を満たしているかを確認する。他の非機能観点のテストと併せて実施されることが多い。

総合テストの段階はリリースまでの残り日数が少なくなっていて、単体・結合テストと違って数日の遅延が致命的になるので、特に進捗管理には注意を払います。不具合対応による遅延はやむを得ない面もありますが、テスト環境や他システムとの調整など、マネジメントレベルの抜け漏れはリカバリ不可能なことがあります。

◆ 負荷テストには十分な時間を確保する

負荷テスト（ラッシュテスト、ストレステスト、大容量テスト等）やパフォーマンステストには大抵、テストツールを使って、端末からの大量アクセスによる負荷をサーバやネットワーク等にかけた状態を作り出します。

かける負荷は、ただ大きければよいというものではありません。現実に起きうるケースに近い形となるように負荷のかけかたを計算します。例えば、実際の運用時点で小容量データが多数集中することが想定されるケースで、大容量データ×最大数の負荷をかけることは現実的でありません。ただ、ストレステストについては、このような前提が超えられた時の挙動確認が目的なので、現実に起きうるケースに沿う必要はありません。

このように負荷を事前に計算することにも時間がかかりますが、負荷テストを通すための環境を準備することにも入念な調整が必要です。例えば、本番業務で実施しているネットワーク越しに負荷テストを実施してしまうと、他の本番業務に大きく影響が出てしまいます。

注記
パフォーマンステストとは、負荷のかかっていない通常状態で、画面等のレスポンスタイムと、バッチ処理等のスループットを計測するテストのこと。

注記
ラッシュテストとは、性能要件として想定している最大負荷（同時アクセス数等）に対して、システムの処理能力を確認するテストのこと。

注記
ストレステストとは、性能要件として想定している最大負荷を超える負荷がかかった想定外の状況に対してシステムの挙動を確認するテストのこと。

注記
大容量テストとは、バッチ処理の所要時間やネットワーク性能の十分性等を確認するために想定されている最大容量のデータの送受信等を確認するテストのこと。

注記
縮退テストとは、部分的なハードウェアの故障などに対して、冗長構成への切替えなどが想定どおりに機能することを確認するテストのこと。

注記
災害対策テストとは、大規模災害発生時の対応（マニュアル含む）が的確かを確認するテストのこと。

注記
ペネトレーションテストとは、システムに対して侵入テストを試み、適正にガードされていることを確認するテストのこと。

注記
インシデントレスポンスとは、インシデントが発生したときにどのようなメッセージがあがり、その時の対応（マニュアル含む）が的確かを確認するテストのこと。

注記
ファジングとは、検査対象のソフトウェア製品に「ファズ（英名：fuzz）」と呼ばれる問題を引き起こしそうなデータを大量に送り込み、その応答や挙動を監視することで脆弱性を検出する検査手法のこと。

とはいえ、本番環境に近い環境で試験しないと意味がありません。そのため、夜間等の影響の少ない時間帯に実施したり、本番環境とほぼ同一のテスト環境を用意したりする等の工夫を行いますが、このようにテスト環境を準備して調整することには時間がかかります。

また、これらのテストは1回で終わるとは想定しないほうが良いでしょう。テストの結果、性能が十分でないことが判明した場合は、システム上で何らかのチューニングを実施した上で、再度同じテストを実施します。場合によっては、何度もチューニングとテストを繰り返すこともあるかもしれません。テスト実施にも十分な時間を確保することが必要です。

◆ 機能テストは、データのバリエーションが重要

テスト工程での検証を有意義なものとするためには、検証に用いるデータについて様々な条件を備えたものを用意する必要があります。

まずは、現在動いているシステムが存在する場合は、その本番データにできるだけ近いデータを利用することです。テストケースを考えるときには、それまで検討を重ねてきた要件定義や設計内容がベースになりますが、そもそも要件として認識されていないものが本番データに含まれている可能性があります。例えば、制度変更時の経過措置で特別に処理したデータや、システム障害や天災に起因したやむを得ない応急措置により例外的で特別なデータが存在することがあります。現行システムで使っている本番データには、このように開発者にとって想定外のデータが存在するので、本番データを使ってテストすることには大きな意味があります。ただし、本番データに含まれる機密性の高い情報は、匿名化等の処理を行うことにも留意してください。

しかし、本番データがあれば十分かというと、実は本番データだけでは万全とはいえません。本番データを使ったテストは、時間とコストがかかるので、せいぜい3～4ヶ月分のデータでしか検証できないからです。そうすると、テストに使う本番データの中には、レアケースにより発生するデータが存在していないこともありえます。また、データ移行ミス等により、本来存在するはずのない誤データが潜んでいて、想定外のシステム障害を起こすようなこともあります。

このようなことに対処するためには、本番データを使ったテストだけで安心するのではなく、それ以外にも例外的なデータが発生することを予期した上で、ブラックボックステストにおけるテストケースをしっかり設定することが重要です。

なお、テスト工程の短期間だけでは検出しにくい障害もあります。例えば、業務運用を継続する中でデータベースのテーブル容量が肥大化していき、そのテーブルを読み取る処理のレスポンスが悪化し続けるというケースが実際にありました。このようなケースを回避するためには、テスト条件を設定する際に今後のデータ量増加等を見込む等の工夫を行ってください。

B. 発見できた障害は最大限活用する

総合テストでの障害の原因は、基本設計のような上流工程で混入したものが多いはずです。そのため、障害の内容によっては、全モジュールの総点検が必要な事態もあり得ます。障害のきっかけとなった事象がレアケースだからという理由で、パッチワークのような一時しのぎの回避策をとることは危険です。安易に蓋をしてしまったがために、本稼働後に同じ事象をきっかけとする障害が発生するというケースも残念ながら存在します。障害対応を完了としてクローズする際は、慎重に検討するようにしましょう。

回り道に思えるかもしれませんが、たまたま見つかった欠陥を氷山の一角と捉えて積極的に総点検を指示することが、結果的には近道です。

事業者に対しては、類似バグがないかという観点から横展開をしているかを確認しましょう。この横展開が形式的なものになってしまわないように、事業者がどの範囲に対してどの

ように対象を抽出したのかを追検証できるエビデンスを必ず残してもらいましょう。横展開は非常に手間の掛かる作業ですが、たまたま発見された障害を横展開して類似バグを撲滅することは、全てをテストするよりもはるかに効率的で現実的です。

障害管理には、横展開と深掘りの2つのアプローチがあります。総合テスト段階で障害残数が収束していない状況では既知の障害に基づいた「横展開」を優先し、障害残数がコントロール可能な数にまで落ち着いてきてから1件１件の事象を「深掘り」した方が良いです。

このように総合テスト段階で障害が発生すると、とにかく面倒くさいが、1つも疎かにせず障害管理表と向き合って、納得いくまで事業者の説明を求めるべきです。

5　受入テストを実施する

【標準ガイドライン関連箇所：第3編第7章第7節】

受入テストは、他のテストと異なり、職員が主体となって行う最終段階のテストです。ここでは、受入テストに関する知識やノウハウを見ていきます。

A. 受入テストと他のテストとの違いを理解する

「サービス・業務企画や要件定義で想定したとおりに情報システムができているか？」「構築された情報システムを用いて実際のサービス・業務を正しく実施できるか？」という観点での確認は、事業者ではできません。確認できるのは、職員だけです。これらの業務視点での確認を行うのが、受入テストです。

受入テストに当たっては、次に示す点に注意してください。

受入テストに関する注意点

- 受入テストを通過すると、基本的に本番リリースに向けた準備が完了したとみなされてしまいます。実際の業務では、正常な処理だけでなく、異常な処理（エラー）も発生しますので、その時になって対処に困らないように、正常系のテストだけでなく、異常系のテストもしっかり実施しましょう。その際、事業者が受入テストの案作成を支援することもありますが、その内容をうのみにするのではなく、業務担当者の目線で内容のチェックをしてください。
- 受入テストは、職員が主体的にテストを実施する必要があります。その職員とは、PJMOの職員だけではありません。システムを実施に利用する業務実施部門の現場担当者こそ、実業務を一番よく知っています。このような現場担当者を受入テストの実施者として組み込んで、協力を得ることが効果的です。
- 受入テストに使用したデータが不用意に本番環境に継続されると、システム障害を引き起こすことがあります。転ばぬ先の杖のつもりで本当に転ばないように気を付けましょう。

受入テストは、総合テストと同様に上流工程の成果を確認するテストです。単にテスト実施者が事業者から発注者に替わるだけで、内容面では似たようなテストとなってしまいがちです。例えば、総合テストで実施したテストケースから「主要な業務」を抜き出し、発注者がその再確認をするだけで終わらせるようなこともあります。あるいは、職員が「モンキーテストとしてその場で思いついた操作をしてみることもありますが、それで何か意味がある（つまり不具合が見つかる）ということはめったにありません。

受入テストは本番運用直前として、できるだけ「①本番データ」、「②本番環境」、「③実

際の利用者」、「④実際の運用者」でテストするべきですが、そのためには相当に周到な準備が必要となります。①本番データを使う理由は、開発事業者が想定できていないような「きれいでない」データで試すためです。②本番環境の使用は機器更改のタイミング等でないと現実的ではありませんが、検証環境でテストするよりは、クラウド環境やコンテナ技術が利用できるならそのほうがよいです。③実際の利用者と④実際の運用者でテストすることで、開発事業者が思い付きもしなかったようなケースが出てくるので、異常ではないちょっと「イレギュラーな」実際のケースを思い出しつつテストしてもらうのがよいです。

受入テストでは、プロダクトとしての品質の確からしさより、リリース後の本番準備が十分かを確認しておきます。そこでは、システムマニュアルや業務マニュアルに即してテストする「マニュアルベースドテスト」が有効です。特に、正常時だけでなくエラー・障害やインシデントが発生した時、マニュアルの記載どおりで問題ないかは確認しておきます。ここでも、「③実際の利用者」にマニュアル片手でテストしてもらい、ITの専門用語や日本語がおかしくて伝わらないことのないように確認しておきます。

受入テストについては、以下のような工夫をすることで、効率的・効果的なテストを行い、業務視点での品質を高めている事例もあります。

事例 7-2
テストの目的と業務シナリオを明確にする

事例：テストの目的と業務シナリオを明確にする

ある省において、既存の情報システムを刷新するプロジェクトがありました。既存の情報システムを構築したプロジェクトでは、PJMOの体制が十分でなかったため、受入テストの大部分をプロジェクト管理支援事業者に任せた結果、受入テストの内容が実際の業務シナリオを網羅したものになっておらず、サービス開始後に現場から使いづらさ等の不満が噴出しました。

刷新プロジェクトでは、過去の反省を踏まえて、受入テストにて業務シナリオが現場に即した内容となっているかを網羅的に確認するため、テスト仕様書の内容を次のように分解し、明確に区別して記述するようにしました。

「目的」
テストの目的（＝そのテストで何を確認するのか？）を記述する。

「シナリオ」
業務の実施手順に基づいたシナリオを記述する。基本的な業務のシナリオだけでなく、例外的なシナリオも含めて網羅的に洗い出す。

「シナリオとテスト項目の組合せ」
シナリオとテスト項目を組合せ、どのシナリオをどのテスト項目で確認するかを明確に記述する。

「テスト項目」
シナリオを細分化し、実施する作業内容、実施条件、確認項目等を記述する。

また、「目的」「シナリオ」はPJMOが作成し、「シナリオとテスト項目の組合せ」「テスト項目」は、プロジェクト管理支援事業者と分担して作成するルールとすることで、効率的かつ効果的にテスト仕様書を作成できるようにしました。

これらの活動の結果、受入テストにて業務シナリオを網羅的に確認することができ、サービス開始後には、現場で混乱することなく業務を提供することができました。

B. 受入テストのテスト計画書を作成する

この実践ガイドブックには、別添として受入テスト計画書のひな形を示しています。

様式例 7-2
受入テスト計画書のひな形

様式例：受入テスト計画書のひな形

受入テスト計画書のひな形を本章別紙としてまとめています。

目次	主要な記載内容
1．はじめに	プロジェクト、業務、情報システムの概要
2．テスト体制	体制、役割、責任範囲
3．テスト環境	実施場所、環境、ツール、前提条件
4．作業内容	テスト対象、実施手順、確認・検証事項
5．作業スケジュール	全体スケジュール、各工程の作業スケジュール
6．テストシナリオ	確認・検証事項、テスト結果の予測
7．合否判定基準	品質基準、合否判定基準

あくまでこのひな形は例示です。移行の内容に応じて記載内容を個別に追加、変更してください。ひな形を見ると、何をどのようなレベルで書くべきかの参考になると思います。

Step.5 見落としがちな活動に注意

設計・開発の活動は、情報システムを構築する作業だけではありません。本番の環境で情報システムを稼働するためには、データの設定、既存サービス・業務や情報システムからの切り替え等や運用・保守の作業を行わなければいけません。これらは情報システム自体の設計・開発と同時に検討していくことがとても大切ですが、見落とされがちになってしまうと、後々大変な思いをすることになります。

ここでは、情報システム自体の設計・開発以外に必要な活動についての知識やノウハウについて、紹介していきます。

1　どのプロジェクトでも必ず移行を計画する

【標準ガイドライン関連箇所：第3編第7章第8節】

情報システムの移行は、どのようなプロジェクトでも必ず発生します。既存のサービス・業務や情報システムが存在しない場合でも、本番の情報システムの構築、データの設定、切り替え、新規業務の開始に関わる業務の変更等は必ず必要です。

ここでは、これらの移行に関するポイントを見ていきます。

A. 移行の種類を理解する

「移行」と聞くと、データの移行や情報システムの切り替えは思い付くかもしれませんが、移行はそれだけではありません。移行は、大きく分類すると「システム移行」「データ移行」「業務移行」の3種類があります。

表 7-15
移行の種類

移行の種類	概要
システム移行	受入テストが終わったものを本番環境にリリースすることを指す。以下のような観点について検討し、検討結果から発生した必要なプログラムやツールに関して、設計・実装・テストを行う。 検討例 • ネットワークやDNSサーバの切り替え方式 • 外部情報システムと連携部分の切り替え方式 • 移行対象となっている設備（H/W等）の移行方式 • 新旧のマスターデータの同期の仕組み • 移行失敗時の切り戻し方式 • 端末又は端末上のソフトウェアの入れ替え方式
データ移行	既存情報システムから新情報システムへデータを適切な形で渡すことを指す。以下の観点について検討し、検討結果から発生した必要なプログラムやツールに関して、設計・実装・テストを行う。 －移行元データ 新規データ作成になるのか、又は現行に元データがあるか？ 元データがある場合、どのような形式（DB、紙、テキスト等）で、どの部分が移行対象となるか？ －実施方法 手作業か又はツール（自動）による方法かどうか？ツールの場合、スクラッチで開発するのか又は既存の製品を利用するのか？ －データ移行処理（マッピング、変換処理、クレンジング）

移行の種類	概要
	データ移行を行う際の処理について、以下の観点で検討を行う。 • テーブル定義（論理）の項目と移行元データの項目の対応（マッピング） • データ変換やクレンジングなどのデータ処理のロジック
業務移行	各種移行方式を検討した結果を踏まえながら、（所管の）業務や利用者において「移行時に発生する業務」「段階移行/平行稼働中の特殊な作業」について、洗い出しや検討を行う。 検討例： • 平行稼働中の業務データの手動保存/移行 • アクセスURLの変更 • 端末の入れ替え • 利用者のログインID/パスワードの変更　など

特に、業務移行は、PJMOが主体となって業務実施部門と調整しながら、進めていく必要があります。また、業務移行は、事業者が検討するシステム移行やデータ移行の検討結果を踏まえて検討する必要があるため、検討が遅くなりがちですが、サービス・業務に関する新たな制約が後から発覚して、移行作業に大きな影響を与えることも多々あります。

したがって、設計・開発の早い段階から、業務移行も含めて移行の検討を行ってください。

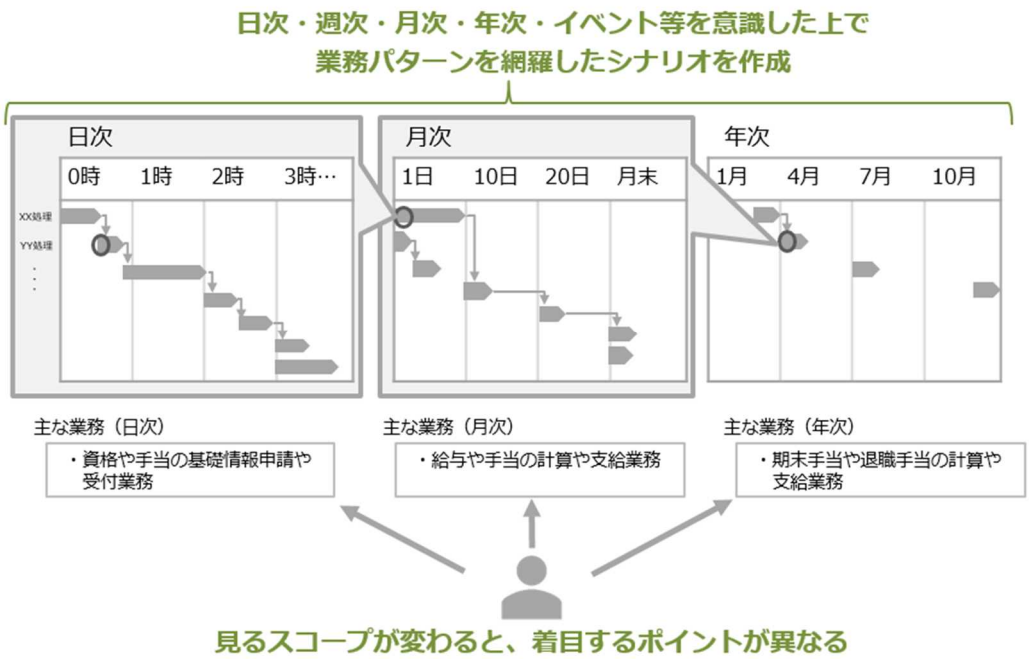
B. リハーサルも考慮した移行計画書を立てる

本番移行は、限られた時間の中での完了が求められ、その結果には正確性が求められます。そのため、手順書を作成して実施に臨みますが、それでも不測の事態が発生することがしばしばあります。このような事態を極力減らし、問題なく本番稼働を迎えるためには、移行手順に則ったリハーサルの実施が必要不可欠です。以下では、リハーサルを行うに当たっての留意点を示します。

リハーサルの留意点

- シナリオは本番業務に係る全てを想定して作成する（以下の「シナリオを作る際の注意点」を参照）。
- 職員が積極的に関与する。
- できる限り本番と同等の環境・本番データを使用する。
- 時間の測定は必ず行う。
- リハーサルの結果から手順等に修正や改善を行う場合、その重要度に応じて、リハーサルで再度検証を行う。
- リスクの影響度合い等を踏まえて、切り戻し等の不測の事態に対する対応（＝コンティンジェンシー・プラン）のシナリオもリハーサルで確認する。

図 7-19
シナリオを作る際の注意点



シナリオを作る際の注意点

- 日々の業務だけでなく、月次、年次、例外時の業務パターンを網羅し、問題なく、かつ迷わず業務が進められるか。
- 繁忙期の業務量でも想定する作業時間内で実施できるか。
- 複数の情報システムと連携する場合、連携を含めて想定どおりに業務を進められるか。

2　次の運用・保守は開発と平行して検討する

【標準ガイドライン関連箇所：第３編第７章第８節】

運用・保守の内容は、運用・保守事業者の調達に間に合うように検討を始めれば大丈夫だろう。そう思っていませんか？

実はそうではありません。運用・保守を考慮せずに情報システムを構築することで、運用・保守に膨大な費用を要する情報システムやサービス・業務の指標値の取得すら困難な情報システムとなってしまうことは少なくありません。

ここでは、そのような状況を引き起こさないためのポイントを見ていきましょう。

A. 指標値を運用作業で取得できるように検討する

情報システムの運用を開始した後には、プロジェクトの目標、KGI、KPIをモニタリングし、これらを達成できるように継続的な改善を行っていかなければなりません。しかし、情報

システムの設計時には、この観点が抜け落ちてしまうことがよくあります。

継続的な改善を行い、プロジェクト目標を確実に達成するためには、指標値の評価を容易に行えるようにして定期的に確認していくことが必要不可欠です。また、指標値の取得だけでなく、分析のためのデータ取得・集計も必要になることも多くあります。設計・開発では、指標値や関連するデータをどのように取得し集計するか？作業にどれくらいの工数がかかるか？を確認し、必要な機能や運用作業を検討してください。

事例 7-3
運用作業での指標値の取得に工数がかかってしまう

事例：運用作業での指標値の取得に工数がかかってしまう

ある省で行われている業務において、個々の業務の処理時間が長い事が大きな問題となっていました。そこで、その業務で利用する情報システムを更改するプロジェクトでは、処理時間の短縮を目標として、業務ごとの処理時間の指標値を設けました。

プロジェクトでは、対策を検討して業務の見直しを行い、情報システムを刷新することとしました。しかし、設計・開発では、業務機能の構築に重点が置かれ、指標の取得方法や運用作業の詳細な検討は不十分な状態でした。

新しい情報システムの運用が始まり、いざ運用作業で指標値の評価のために業務の処理時間のデータ取得・集計を行ってみると、評価のたびに非常に多くの運用作業工数がかかってしまうことがわかりました。

結局、定期的な指標値評価が必要なことから、さらに新たな予算措置を講じ、データの取得・集計を容易に行うための新しい機能を追加で開発することになってしまいました。

サービス・業務の運営が始まり、モニタリングを行う際に困ることのないよう、設計の段階で以下の内容を確認しておきましょう。

指標に関する確認ポイント

- 業務要件定義で定めた全ての「管理すべき指標」について、「誰が」「どのように」「どれくらいの工数で」取得できるように設計しているかを事業者に確認する。
- 取得するデータの形式が定められているか、加工・集計の要否、加工・集計が必要な場合は、誰がやるのか、データはどのように保管するルールとするかを確認する。
- 運用・保守の作業内容に、各種指標に係るデータの取得が漏れなく含まれているかを確認する。

◆ 運用・保守の検討は、実務経験のある担当者と一緒に行う

意外に思うかもしれませんが、情報システムの設計・開発と情報システムの運用では、必要とする技術や知識が大きく異なります。また、設計・開発と運用の両方の経験を十分に持つ技術者は、そう多くはありません。こういった背景もあり、設計・開発を行う事業者が必ずしも良い運用を設計できるとは限りません。

このため、運用・保守計画の案を作成する際は、情報システムの運用・保守管理経験がある職員に参加してもらえるように調整しましょう。運用・保守の管理経験のある担当者から

は、「情報システムを構築するときに、こういう点に注意しておけば良かった」「こういう運用・保守作業を最初から見込んでおけば良かった」という経験やノウハウを聞けるはずです。

もしも、運用・保守経験のある担当者の調整が難しい場合は、PMOや府省CIO補佐官に相談してみてください。

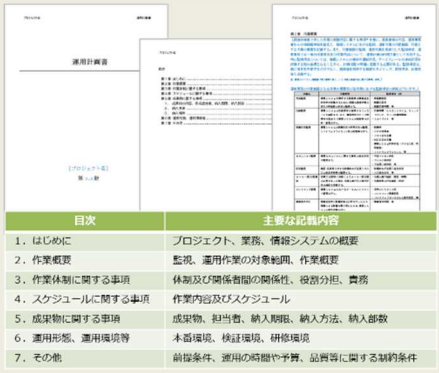
B. 運用・保守の計画を立てる

この実践ガイドブックには、別添として運用・保守に係る計画書のひな形を示しています。

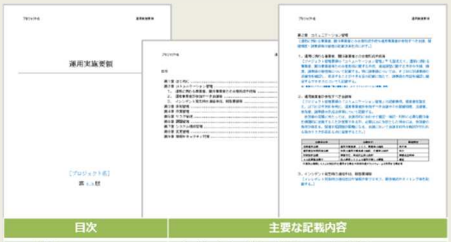
様式例 7-3
運用計画書、運用実施要領、保守計画書、保守実施要領のひな形

様式例：運用計画書、運用実施要領、保守計画書、保守実施要領のひな形

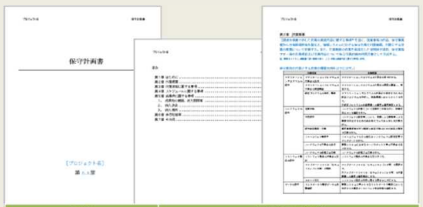
運用計画書、運用実施要領、保守計画書、保守実施要領のひな形を第9章別紙としてまとめています。



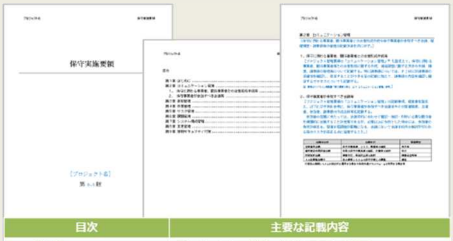
目次	主要な記載内容
1．はじめに	プロジェクト、業務、情報システムの概要
2．作業概要	監視、運用作業の対象範囲、作業概要
3．作業体制に関する事項	体制及び関係者間の関係性、役割分担、責務
4．スケジュールに関する事項	作業内容及びスケジュール
5．成果物に関する事項	成果物、担当者、納入期限、納入方法、納入部数
6．運用形態、運用環境等	本番環境、検証環境、研修環境
7．その他	前提条件、運用の時間や予算、品質等に関する制約条件



目次	主要な記載内容
1．はじめに	プロジェクト、業務、情報システムの概要
2．コミュニケーション管理	運用事業者が参加すべき会議、開催頻度、議事録等の管理
3．体制管理	作業体制の管理手法
4．作業管理	運用作業実績、情報システムの構成、運転状況、サービスレベルの達成状況
5．リスク管理	リスクを提示する際の手順や報告様式
6．課題管理	課題を提示する際の手順や報告様式
7．システム構成管理	ハードウェアやソフトウェア製品、ネットワーク等の各資産における管理項目
8．変更管理	管理対象、変更手順、管理手法
9．情報セキュリティ対策	情報セキュリティ確保に必要な対策



目次	主要な記載内容
1．はじめに	プロジェクト、業務、情報システムの概要
2．作業概要	保守作業の対象範囲、作業概要
3．作業体制に関する事項	体制及び関係者間の関係性、役割分担、責務
4．スケジュールに関する事項	作業内容及びスケジュール
5．成果物に関する事項	成果物、担当者、納入期限、納入方法、納入部数
6．保守形態、保守環境等	保守形態、テスト等を行う検証環境
7．その他	前提条件、保守の時間や予算、品質等に関する制約条件



目次	主要な記載内容
1．はじめに	プロジェクト、業務、情報システムの概要
2．コミュニケーション管理	保守事業者が参加すべき会議、開催頻度、議事録等の管理
3．体制管理	作業体制の管理手法
4．作業管理	保守作業実績、サービスレベルの達成状況
5．リスク管理	リスクを提示する際の手順や報告様式
6．課題管理	課題を提示する際の手順や報告様式
7．システム構成管理	ハードウェアやソフトウェア製品、ネットワーク等の各資産における管理項目
8．変更管理	管理対象、変更手順、管理手法
9．情報セキュリティ対策	情報セキュリティ確保に必要な対策

あくまでこのひな形は例示です。移行の内容に応じて記載内容を個別に追加、変更してかまいません。ひな形を見ると、何をどのようなレベルで書くべきかの参考になると思います。

なお、ここで作成した運用・保守の計画の案は、運用・保守事業者の調達仕様書の付属資料になり、運用・保守事業者の調達後に確定されることとなります。

参考 7-1
設定と設計は異なる

参考：設定と設計は異なる

情報システムの構築では、ハードウェア、ミドルウェア、アプリケーションのフレームワーク等に対して様々な設定が必要となります。これらの設定には、設定値の決定に十分に検討や調査が必要なものから、他の設計により自明で決まるようなものまで様々な難易度のものがあります。

事業者に見積りを依頼すると、大量の設定に関する設計書が他の設計書の同じような工数で見積られていることがあります。このような場合は要注意です。たしかに設定に関する設計書は必要ですが、全ての設定に対して機能の設計等と同じだけ工数がかかることは稀です。

このため、見積り精査の際には、設定と設計を区別して、工数を確認するようにしましょう。設定に対する工数割合が大きい場合は、事業者に理由を確認するようにしてください。

3　種類を理解し揃えるマニュアルを厳選する

【標準ガイドライン関連箇所：第3編第7章第8節】

サービス・業務を開始するに当たっては、業務実施担当者や利用者向けのマニュアル等の教育資料が必要になります。実は、これらのマニュアルは、利用率、利用者の満足度、業務の浸透度等に大きな影響を与えます。

ここでは、情報システムに係るマニュアルに関する知識やノウハウについて、紹介します。

A. マニュアルの種類を理解する

マニュアルと聞くと、情報システムの操作マニュアルはすぐに思いつくかもしれませんが、ほかにも様々な種類のものがあります。以下に、マニュアルの例を示します。

表 7-16
マニュアルの種類

マニュアル名称	概要
業務マニュアル 業務手順書	業務実施部門の担当者が業務を遂行する上で活用する統一的な手順について詳細に記載されたものであり、人事異動で配属された職員等がこれを読んで理解すれば業務を遂行できるような内容になっている。したがって、記載内容は情報システムの具体的な操作内容ではなく、また、業務のシーンごとに記載されていることが望ましい。
システム操作マニュアル（利用者向け）	情報システムにおいて情報入力や帳票出力等、業務遂行時における一連の操作手順が記載されたものである。役割別に分かれる可能性がある。
FAQ	マニュアルを読んでもわからずに利用者からよく来る質問とそれに対する回答を、後から一覧等にまとめて公開している情報 Web サイト。
システム操作マニュアル（管理者向け）	情報システムを管理するための機能について、機能及び一連の操作手順が記載されたものである。
（虎の巻レベルのもの）	情報システムが絡まない場合、統一的な手順が詳細に書かれたものは特になく、部分的な業務遂行ノウハウを記載している。
（ベンダが作るシステ	情報システムの画面単位のマニュアル。業務遂行の観点では書かれて

マニュアル名称	概要
ムマニュアル）	いない。
（ローカルルール）	支所ごとに業務ルールカスタマイズされたもの。

ここで注意してほしいことは、「業務マニュアルとシステム操作マニュアルは異なる」ということです。業務マニュアルは、業務を遂行するために必要となる一連の作業（情報システムで行う業務以外も含む）の手順やルールを示すのに対して、システム操作マニュアルは、情報システムの操作に特化した手順やルールを示します。

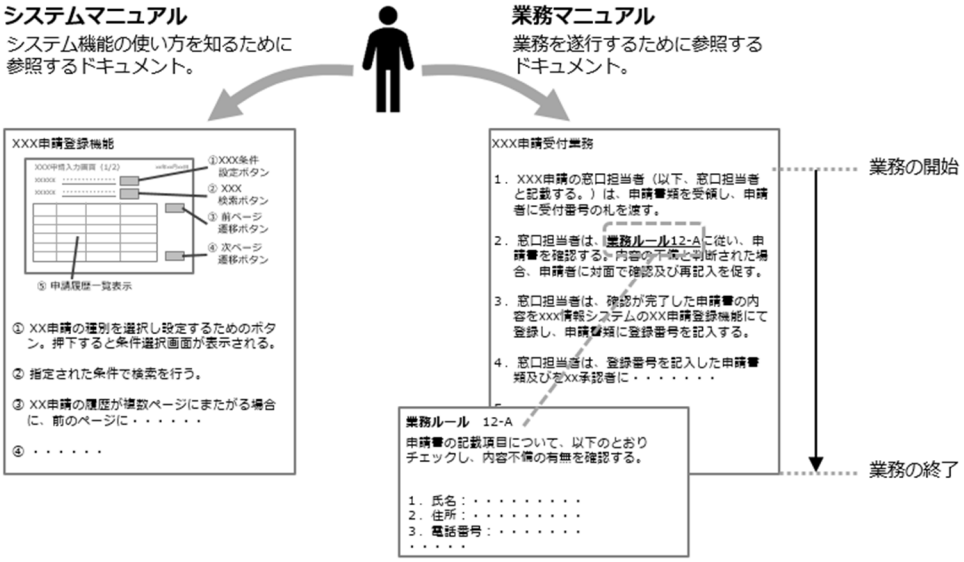


図 7-20
システムマニュアルと業務マニュアルの違い

業務マニュアルは、一般的に職員が作成するものです。業務マニュアルの作成については、実践ガイドブック「第8章 Step.2-2-A.業務マニュアルと他のマニュアルとの違いを理解する」を参照してください。

◆ 役に立つシステム操作マニュアルを作成するには

システム操作マニュアルの作成を事業者に依頼する場合も注意が必要です。事業者に対して、特に詳細を指定せずに「システム操作マニュアルを作成してください」と指示をした結果、現場になじまないマニュアルが出てきた、ということは少なくありません。事業者にマニュアルの作成を依頼する場合は、サンプル等も活用しながら、作成内容を詳細に指定する必要があります。

また、マニュアルの作成タイミングも気を付けなければいけません。実際にマニュアルを使用する部門が内容を確認しなければ、役に立たないマニュアルになってしまいます。システム操作マニュアルは、受入テストまでに作成し、それらを受入テスト内で一緒に確認します。

利用者視点のマニュアルを作成するためには、職員側がしっかりチェック（又は作成・修正）する必要があります。マニュアルに係る計画を立てる場合は、これらの点に注意してください。

マニュアル作成に関する検討事項

- 現場は、どのようなマニュアルを求めているかを、実際に使われているマニュ

アルを踏まえて確認する。使われるマニュアルを作る。

- マニュアル作成に関する職員の作業、事業者の作業を明確にする。
- マニュアルの作成タイミングに気を付ける。マニュアルはいつ、誰に確認（テスト）されるのか？それまでに、マニュアル作成が完了するように計画する。

Step.6 新業務の運営を円滑に行うための準備

情報システムのテストが一通り完了し、あとは本番移行を行いサービス・業務を開始するところまで来ました。でも、ちょっと待ってください。本当にそのままサービス・業務を開始して大丈夫でしょうか？

ここでは、サービス・業務を無事に開始し、円滑に運営していくために必要となる設計・開発の活動の締めくくりの作業についてのポイントを見ていきます。

1　本番移行と本番稼働の開始を承認する

【標準ガイドライン関連箇所：第3編第7章第8節】

サービス・業務の開始の承認には、本番移行の開始を承認する「移行判定」とサービス・業務の開始（＝新しい情報システムへの切り替え）を承認する「稼働判定」の2段階で行います。

ここでは、移行判定と稼働判定のポイントについて紹介します。

A. 移行判定と稼働判定の違いを理解する

移行判定と稼働判定は、ともに設計・開発の活動の最終盤で行うサービス・業務の開始に係る判定行為ですが、そこで判定する内容や条件等は異なります。次に示す違いの例を参考に、それぞれの判定において、「いつ」「誰が」「何を判定するために」「どのような条件で」「どのように」判定するかを事前に定めて、関係者と合意しておきましょう。

表 7-17
移行判定と稼働判定の違いの例

判定の種類	承認内容の例	タイミングの例	承認条件の例
移行判定	本番移行の開始	受入テスト、移行リハーサル完了後、本番移行前	• 受入テストの完了が承認されている • 第三次レビューで妥当と判断される • 移行計画書及びリハーサルの結果が適正である
稼働判定	本番稼働の開始（新しいサービス・業務の開始）	本番移行の完了後又は完了が確実に見込まれるとき	• 本番環境への移行の結果が適正である

◆ その稼働判定でリリースを止められますか

リリース判定と稼働判定には様々なパターンがあります。リリース判定＝稼働判定が最も多いですが、情報システム刷新の場合はリリース判定で次期環境へデプロイ（情報システムの展開・配置）、その後に現行環境を止めて次期環境へデータ移行し、稼働判定会議の結果をもって現行の情報システムから次期情報システムに切替る場合もあります。同時にデータ移行する場合や先にデータ同期をとっている場合も前者のパターンとなるので、案件によってリリース判定、稼働判定を分けて検討しておきます。

リリース判定、稼働判定はとかくシャンシャン会議になりがちです。特に、制度改正や機器更改でリリース日が延伸できない場合の判定会議はそのような場となります。そもそもなぜ判定会議が必要かと言えば（担当者から管理者への最終判断伺い及び承認行為の意味合いもありますが）、最後の最後で誤った判断をしないよう発注者側・事業者側双方のプロジェクトマネージャが自ら振り返るためです。発注者側は、状況を正しく認識できるまで事業者に噛み砕いた説明を求めて理解するか、情報システムのことがわかるPMO・PJMOの判断を仰ぐべきです。

そのためには、どういう場合には「ダメ」を出すか、あらかじめ明文化して関係者で共有しておくことが必須です。グループシンク（集団浅慮）に陥らないようにするには、リリース、稼働判定基準は厳密に定めておく必要があります。例えば「重要な障害が未解決でないこと」というのは、厳密な判定基準ではありません。その場合、判定会議直前に残障害の重要度を「軽」にしようとする交渉が巻き起こります。むしろリリース判定、稼働判定の「否認条件」を列挙し、1つでも該当すれば否認することを宣言しておくべきです。

共有する関係者には当然、事業者も含まれています。判定基準をあらかじめ提示し、「ダメ」出しするラインを知らせておくことで、事業者が当然それをクリアした上で判定会議に参加してくることを期待することができます。

◆ 情報システムの正しさだけでは不十分

稼働判定は、①開発した情報システムの正しさ、②移行・切り替え作業の確からしさ、③稼働後の運用準備状況、の3つの観点が必要となります。ここまでにテストしてきているので①開発システムの正しいのはもちろんですが、それに比べて②移行・切り替え作業、③運用準備は軽んじられる傾向があります。

開発した情報システムの正しさとは「計画した全てのテストケースを消化し、摘出された全ての障害が除去されていること」です。テスト項目密度や不良密度のような品質メトリクスは「計画した全てのテストケース」の目安の1つにしか過ぎず、それだけで判断してはいけません。「摘出された全ての障害」はまさに字句どおり全て除去すべきで「ワークアラウンド（応急処置）は確立できているので次回リリースまで…」とか「致命的な障害の手当はできているので軽微な障害はこのまま…」というのは許容しないことが必要です。逆からいうと、操作性等の改善事項であって次回リリースに回すことが妥当なものは、障害管理ではなく課題管理に含めるべきです。

移行・切り替え作業の確からしさでは「何も問題ありませんでした」ではなく、きちんとエビデンスの提供を求めることが必要です。データ移行についてはレコード件数やハッシュ比較のような機械チェックで移行結果を検証（特に移行作業が複数拠点の場合）し、その上で移行直後に「本番環境」での疎通テスト程度の確認は完了しておきます。ただし多くの場合、移行・切り替え時には何らかのデータベースのテーブルレイアウト変更が入るので、ツールで現新比較するにはフィルタリングや正規表現等の工夫が必要となります。

運用準備では運用手順書、報告フォーマット、保守連絡先のような運用・保守マニュアル類の他、移行直後の切り戻し手順、セキュリティインシデント対応手順、BCP等のコンティンジェンシープランの準備を確認しておきます。また、これらのドキュメントに即してテストする「マニュアルベースドテスト」も有効です。正常時の操作・運用だけでなく、エラー・障害・インシデントが発生した際の手順についても確認しておくようにしましょう。さらに案件によっては、エンドユーザあるいはヘルプデスクへの教育・訓練も必要となってきます。特に大規模システム更改などでは、本稼働「直後」に深刻なシステム障害が発生した場合の切り戻し手順が必要でないかを検討しておくことも大切です。

2　正しく引き継ぎを行い、トラブルを減らす

【標準ガイドライン関連箇所：第3編第7章第9節】

設計・開発に携わった事業者の交代（又は撤退）は、設計・開発の終了時点でやってくるとは限りません。一般的には、設計・開発事業者は情報システムの整備後も運用・保守等で関わることが多いですが、運用・保守の途中で事業者を交代することもあります。このような場合に、設計・開発に関する情報が欠落してしまい、トラブルになることも少なくありません。また、設計・開発の終了や運用・保守が安定したタイミング等で、事業者側の体制の縮小によりキーマンが移動することでも、同じようなことが起こり得ます。

このため、標準ガイドラインでは、設計・開発事業者から運用事業者及び保守事業者への引継ぎを行うように定めています。

引継ぎの際には、成果物についての引継ぎが中心に行われがちですが、絶対に忘れてはいけないのは「課題」の引継ぎです。残存している課題については、保守等を通して今後の対応を検討していかなければいけませんし、既に対処済みの課題であっても、運用作業の制約になっているかもしれません。

課題の引継ぎを求めると、多くの場合は課題一覧が提出されますが、これだけでは不十分な場合があります。課題の一覧には、詳細な経緯、決定事項の理由、課題に係る調査結果等が完全には含まれていないことが多く、これらを欠落してしまうことで、誤った対応や検討漏れ等を招くこともあります。

したがって、引継ぎに当たっては、課題の一覧を読み合せ等で必ず確認し、経緯の説明や不足する情報を求めるようにしてください。

事例 7-4
異なる事業者間で引継ぎをスムーズに行う工夫

事例：異なる事業者間で引継ぎをスムーズに行う工夫

設計・開発事業者が作成した運用手順書を用いて運用事業者が運用作業を行う場合、手順どおりに行ったとしてもうまくいかないことがあります。これは、設計・開発事業者が作成した運用手順書は、設計・開発時に運用作業を想定して作成されたものであり、実際に稼働している本番環境での運用作業とは異なっていることなどが原因です。

また、運用事業者が運用手順書を作成して運用作業を行っても、やはりうまくいかないことがあります。これは、設計思想などの資料に表れにくい情報があること、設計・開発を行っていない運用事業者が設計の詳細を把握できていないことなどが原因です。

運用段階で、このような問題が生じないよう、スムーズに引継ぎを行うための工夫をご紹介します。

・　設計・開発だけでなく運用作業も調達範囲とする

ある省庁では設計・開発事業者が自ら作成した運用手順書を用いて1年間の運用作業を行うところまでを調達範囲としています。設計・開発時の想定だけでなく、実際の運用を経て運用手順書を作成することができるため、設計・開発に携わった事業者が次年度に変更になり引き継いだあとも、作業が円滑に進められるようになります。

・　引継ぎ期間を長く設定する

引継ぎ期間は 1 か月程度を設定するのが一般的ですが、十分ではないケースが多くみられます。

ある省庁では、引継ぎ期間を3か月とり、設計・開発事業者が作成した運用手順書

（案）を、設計・開発事業者と運用事業者が協力してブラッシュアップし、運用手順書を完成させるという旨をそれぞれの調達仕様書に明記しています。そうすることで実際の運用作業中に発見された手順の不備を解消しつつ、設計・開発に関する情報を運用事業者に引き継ぐことができています。

このように、設計・開発事業者を運用の初期段階の実務にも関与させることで、品質の高い手順書が作成され、スムーズな引継ぎができることが期待されます。ただし、事業者間の調整を綿密に行うためには発注者側の負担も相応に増加するので、十分に計画することが重要です。

・　事業者間の協力関係を築きやすくする

障害などの問題が発生したときにばかり設計・開発事業者と運用事業者を交えた打合せを行うと、責任の押し付け合いになったり、同席を嫌がったりして円滑に運営できなくなるおそれがあります。

ある省庁では、共通のコミュニケーションツールを導入することで、問題発生時だけでなく、平時からコミュニケーションを取りやすい環境を作っています。また、情報システムの円滑な運用を目的とした、設計・開発事業者と運用事業者の双方が出席する会議体を設置するとともに、それを協力して運営することを調達仕様書で求めています。

複数の事業者の協力を得て情報システムを円滑に運用するためには、このように事業者間の協力関係を築く工夫も必要です。