

Orchestrating a brighter world



業務システム構築基盤 SystemDirector Enterprise 開発方法論 ご紹介

日本電気株式会社

\Orchestrating a brighter world

未来に向かい、人が生きる、豊かに生きるために欠かせないもの。

それは「安全」「安心」「効率」「公平」という価値が実現された社会です。

NECは、ネットワーク技術とコンピューティング技術をあわせ持つ

類のないインテグレーターとしてリーダーシップを発揮し、

卓越した技術とさまざまな知見やアイデアを融合することで、

世界の国々や地域の人々と協奏しながら、

明るく希望に満ちた暮らしと社会を実現し、未来につなげていきます。

目次

第1章 SystemDirector Enterprise開発方法論とは

- 1.1. SystemDirector Enterprise の狙い
- 1.2. SystemDirector Enterprise の体系
- 1.3. SystemDirector Enterprise の構成要素
- 1.4. SystemDirector Enterprise 開発方法論とは
- 1.5. 一般的な開発方法論との違い
- 1.6. 共通フレーム対応
- 1.7. 開発方法論ドキュメント体系

第2章 SystemDirector Enterprise開発方法論の特長と効果

- 2.1. 開発方法論の特長と効果
- 2.2. フロントローディング
- 2.3. 運用設計

第3章 SystemDirector Enterprise開発方法論の概要

- 3.1. 開発プロセス定義 概要説明
- 3.2. AP開発手順書 概要説明

第4章 サポートサービス

- 4.1. お問合せ先

第1章 SystemDirector Enterprise開発方法論とは

1.1.SystemDirector Enterprise の狙い

環境認識

技術の高度化、
新技術の台頭

短納期・
スピード重視

長期間の
保守メンテナンス

ニーズ

「信頼できるSI」

先端技術の活用
迅速な対応

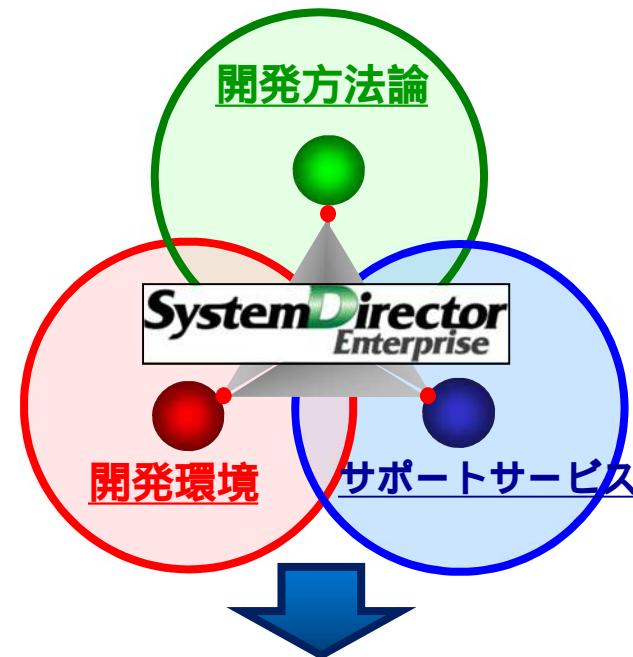
生産性の向上

十分な期間の
サポート提供

これらを実践するシステム構築環境を整備
今までのNECグループのシステム構築技術を結集
SystemDirector Enterprise の開発へ

1.2.SystemDirector Enterprise の体系

NECグループ標準の業務システム構築基盤です。開発方法論、開発環境、サポートサービスによる効率的なシステム構築を支援します。



開発方法論（誰が、いつ、何をするのか）

- 概説書、手順書、ガイド、ドキュメントサンプル集
- 各工程の作業手順をサポート

開発環境（何を作るか、どうやって作るのか）

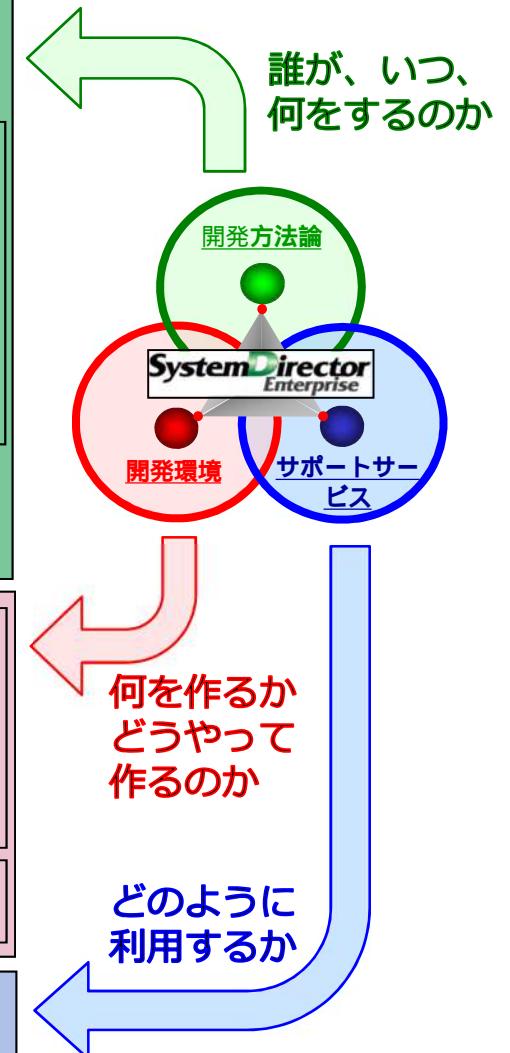
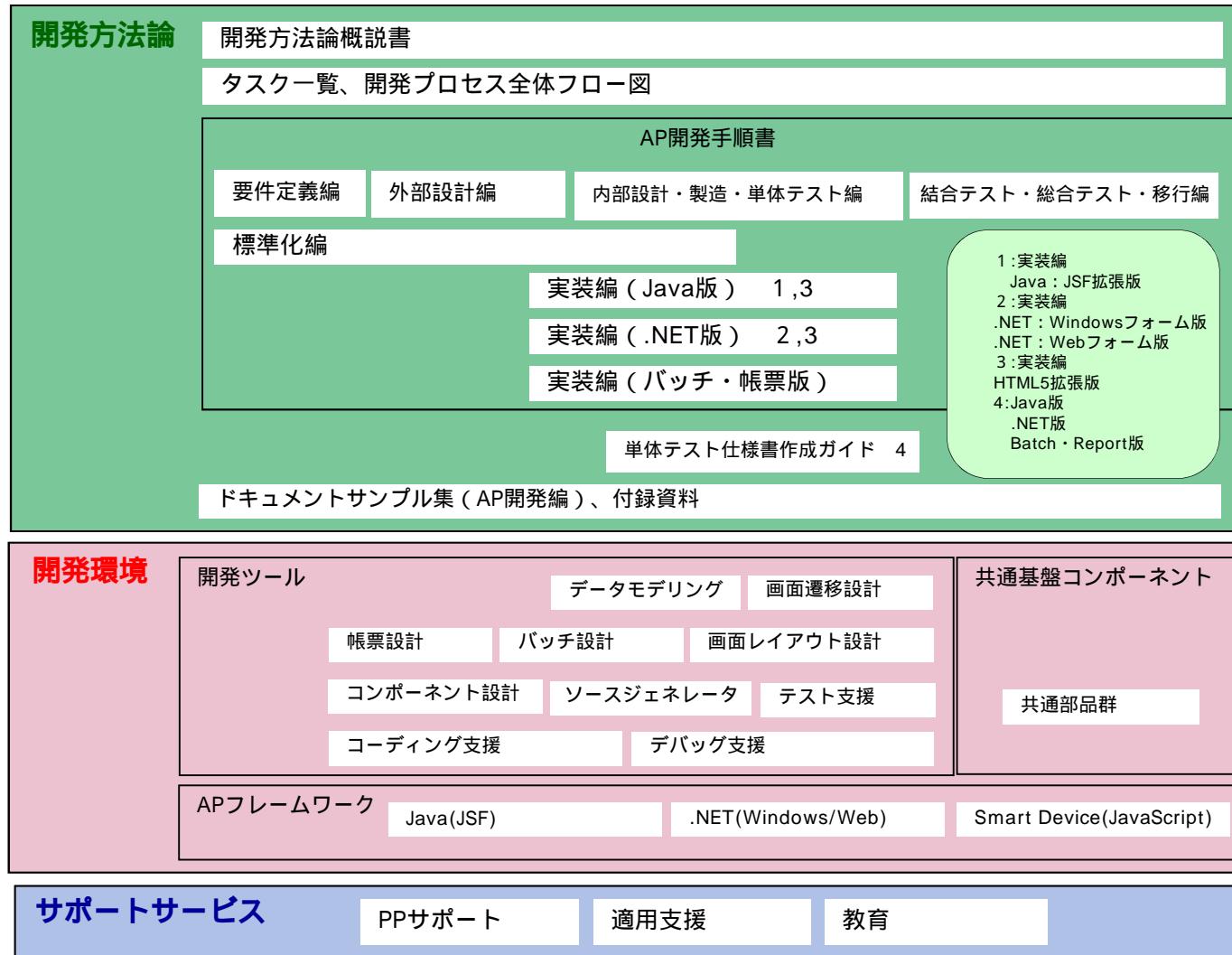
- 業界標準アーキテクチャを採用したフレームワーク
- 開発効率を向上する開発ツールの提供

サポートサービス（どのように利用するか）

- システム構築をさまざまな形でサポート
- トータル10年間のソフトウェア製品サポートを提供

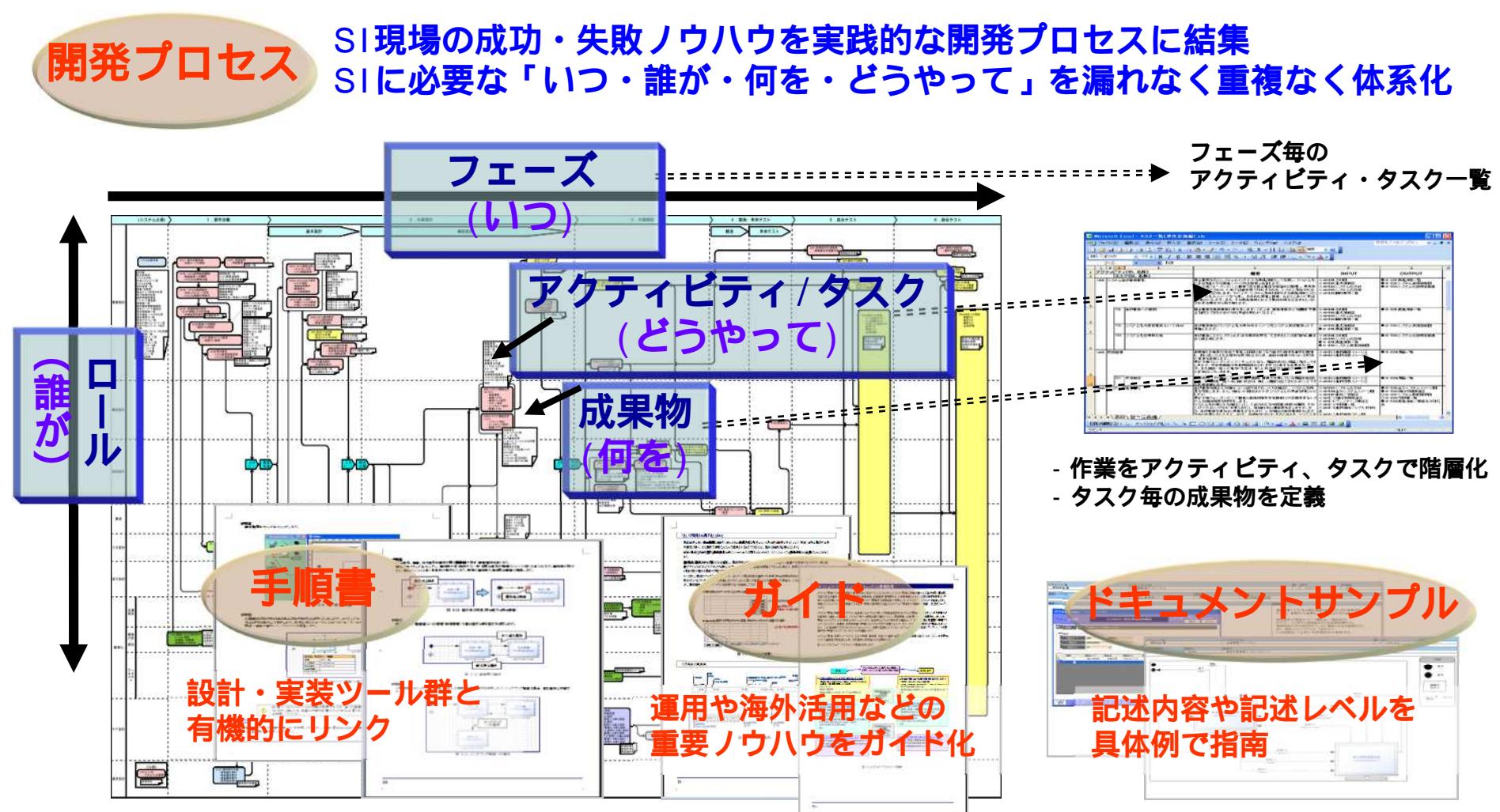
1.3.SystemDirector Enterprise の構成要素

業務システムを構築するために必要十分な支援内容を提供します。



1.4.SystemDirector Enterprise 開発方法論とは

開発プロセスに従った手順書、ドキュメントサンプルなどを提供します。



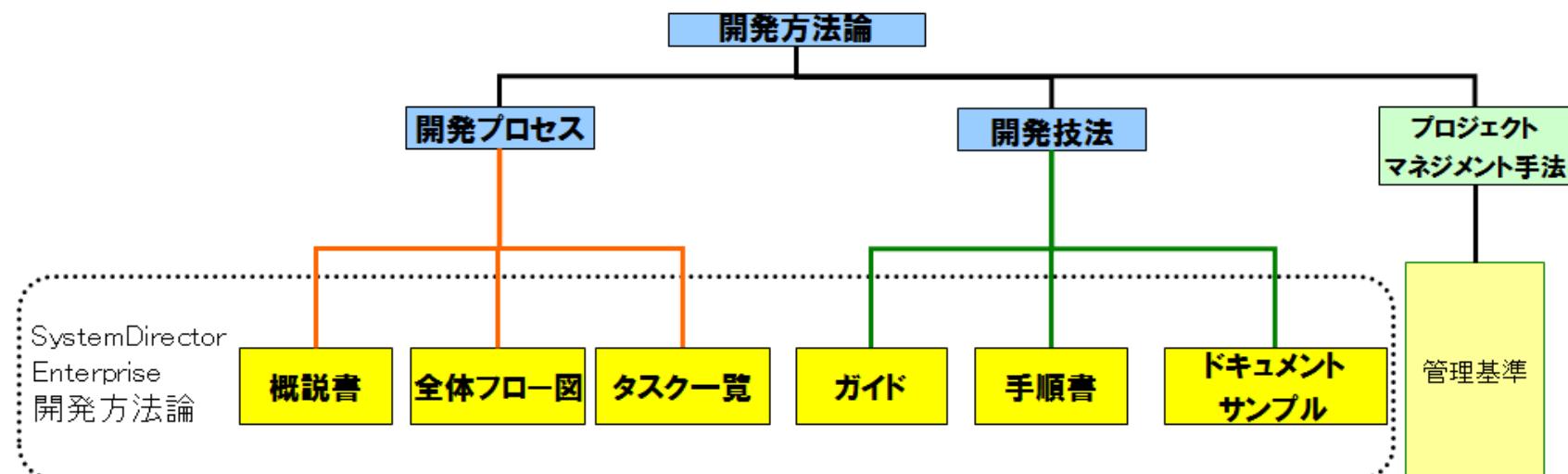
1.5.一般的な開発方法論との違い

役割を意識した、手順、成果物、規約などを体系的に定義します。

一般的に開発方法論は、以下の3つで構成されます。

- オブジェクト指向やデータ中心アプローチなどの具体的な開発手法やツール類を定めた「開発技法」
- 開発をどのような手順で進めるかを定めた「開発プロセス」
- プロジェクトマネジメント手法

SystemDirector Enterprise開発方法論は、オープン環境における業務システム開発をターゲットとして、ソフトウェアライフサイクルプロセス（SLCP）における「要件定義」「外部設計」「内部設計」「製造・単体テスト」「結合テスト」「総合テスト」「移行」の一連の開発プロセスを中心に、役割（ロール）ごとの作業手順、成果物（ドキュメント）、作業規約を体系的に定めた開発方法論です。



1.6.共通フレーム対応

業界標準に準拠した体系をサポートします。

本開発方法論で定義しているフェーズは、業界標準（共通フレーム2013：SLCP-JCF2013）に沿った体系となっています。本開発方法論は、共通フレーム2013の要件定義・開発プロセスをサポートしております。

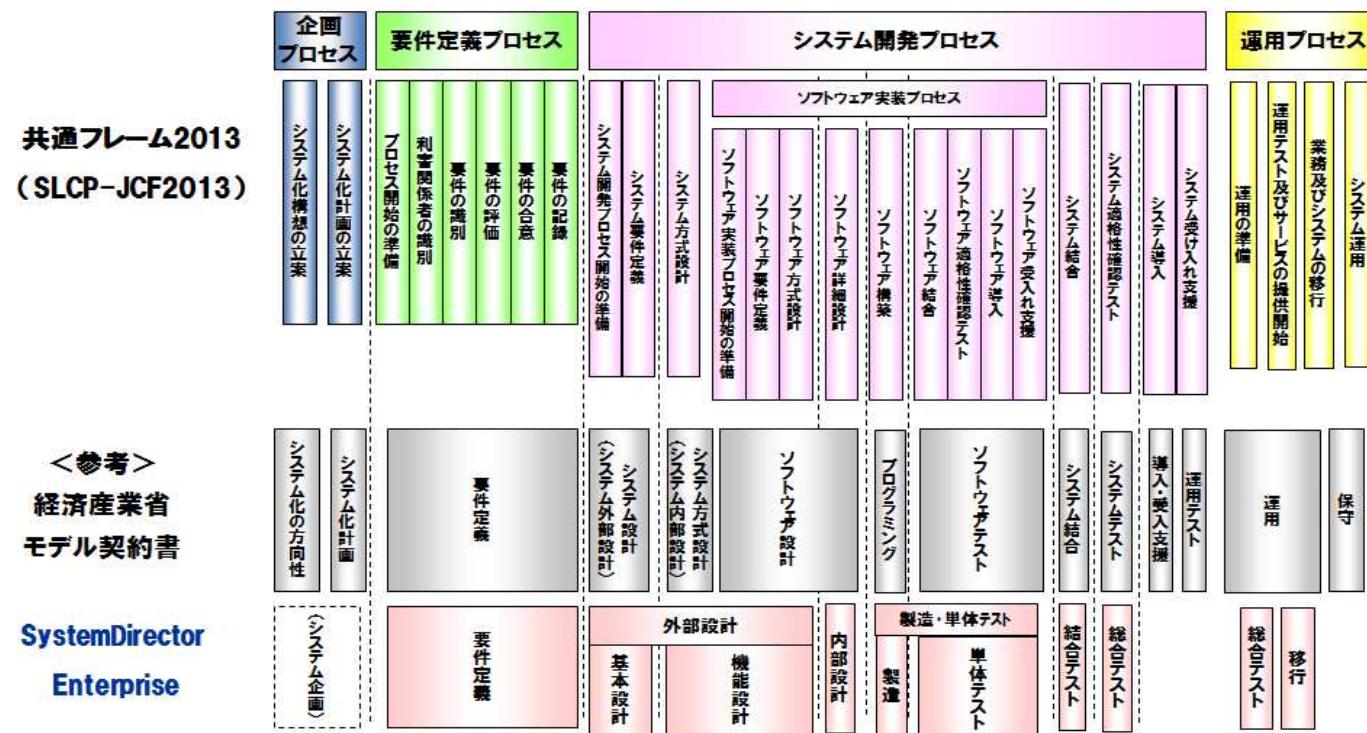
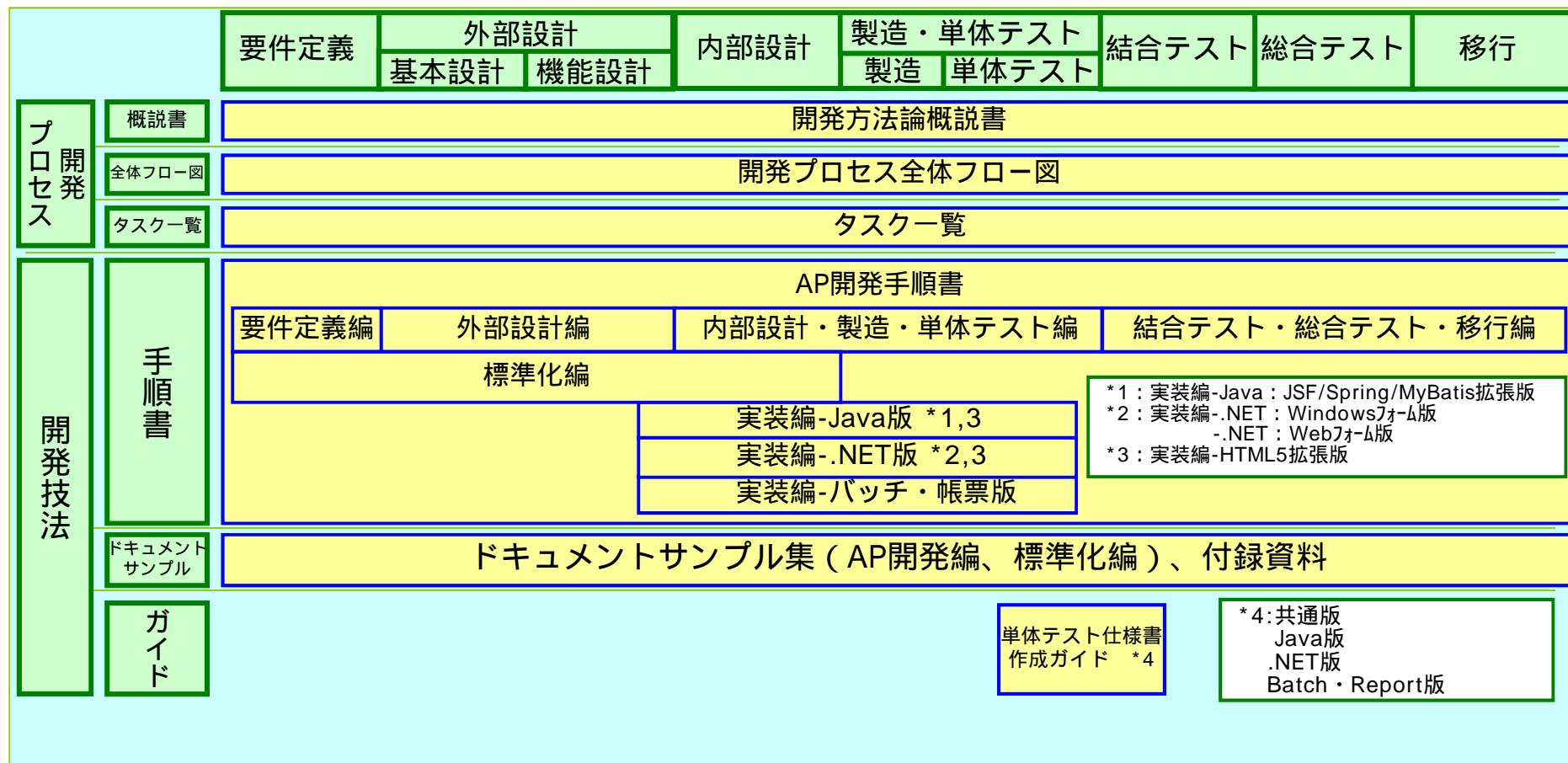


図1 プロセスレベルの対応関係

※共通フレーム2013との関係はウォータフォールモデルをベースとし、フェーズ、作業順序を考慮して整理しています。

1.7.開発方法論ドキュメント体系（1/4）

開発方法論ドキュメントは、開発プロセス（全体を説明した概説書、全体フロー図、タスク一覧）、開発技法(AP開発手順書、ある領域に特化したガイド、手順書を実践的に補完するドキュメントサンプル)で構成されています



1.7.開発方法論ドキュメント体系（2/4）

【概説書】

種別	概要説明	ロール
1 開発方法論概説書	開発方法論（AP開発）の目的、体系、基本思想、特長および開発手順の概要を説明する。	全ロール
2 補足資料	開発方法論の補足説明として、全体プロセスフロー図、各タスク一覧、用語集、成果物CRUD図、成果物フロー図、共通フレーム2007との対応関係を用意。	全ロール

【AP開発手順書】

種別	概要説明	ロール
1 要件定義編	要件定義のアクティビティやタスクフローを定義し、企画書を基に要件定義書（機能／非機能要件と業務フロー）をどのようにまとめていくかを説明する。	業務設計 DB設計 移行設計
2 外部設計編	基本設計、機能設計のアクティビティやタスクフローを定義し、要件定義書をもとにシステムフローの展開や機能仕様書へ落とし込む手順について説明する。また、フロントローディング手法の採用に対応して、テスト計画、テストシナリオの洗い出し作業手順を記述する。	業務設計 機能設計 テスト DB設計 移行設計
3 実装編-Java : JSF/Spring/MyBatis拡張版 実装編-.NET : Windowsフォーム版 -.NET : Webフォーム版 実装編-HTML5拡張版 実装編-バッチ・帳票版	オンラインアプリケーションの内部設計から製造までの作業手順とノウハウを説明する。	業務設計 機能設計 実装設計 製造
4 標準化編	要件定義フェーズから内部設計フェーズまで標準化作業の手順とノウハウを説明する。	標準化

1.7.開発方法論ドキュメント体系（3/4）

【AP開発手順書】

種別	概要説明	ロール
5 内部設計・製造・単体テスト編	開発環境に依存しない下流工程（内部設計～単体テスト）の作業手順を説明する。	業務設計 機能設計 実装設計 製造 テスト
6 結合テスト・総合テスト・移行編	結合テストから移行までのテスト設計、テスト実施方法、および本番環境への移行方法を説明する。	業務設計 機能設計 実装設計 製造 テスト DB設計 移行設計

1.7. 開発方法論ドキュメント体系（4/4）

【ドキュメントサンプル集】

種別	説明	ロール
1 AP開発編	要件定義、外部設計、内部設計、製造・単体テスト、結合テスト、総合テスト、移行の成果物サンプル（記入例）、テンプレート	PJマネージャ 業務設計 機能設計 実装設計 テスト DB設計 移行設計
2 標準化編	命名規則、コーディング規約、UI標準	標準化

【ガイド】

種別	概要説明	ロール
1 単体テスト仕様書作成ガイド	単体テストフェーズにおいて、テスト者が単体テストを行う際に、テスト者にとってわかりやすいテスト仕様書を作成するために注意すべきポイントについて説明する。	テスト

第2章

SystemDirector Enterprise開発方法論の特長と効果

2.1.開発方法論の特長と効果

開発方法論では、SI生産革新を実現し、高品質なシステムをご提供するために、「フロントローディング」、「標準化への対応」、「テストの強化」を整備しています。

目的	特長と効果
▶早期に品質を確保し、後工程での後戻りを抑制する	【フロントローディング】 ▶開発初期（フロント）でやるべきことを正しく行ない（ロード）、品質を作りこむことで、開発の後半で起きる問題を最低限に抑え込む
▶業界標準に対応し、日本国内の商慣習に合わせた契約を実現	【標準化への対応】 ▶『共通フレーム2013』、および経済産業省の『モデル取引・契約書』に準拠
▶テスト計画を上流工程で立案し、実施範囲、役割分担を確立	【テストの強化】 ▶テストプロセス、テストロールを定義し、各フェーズで検証漏れを防止

2.2.フロントローディング

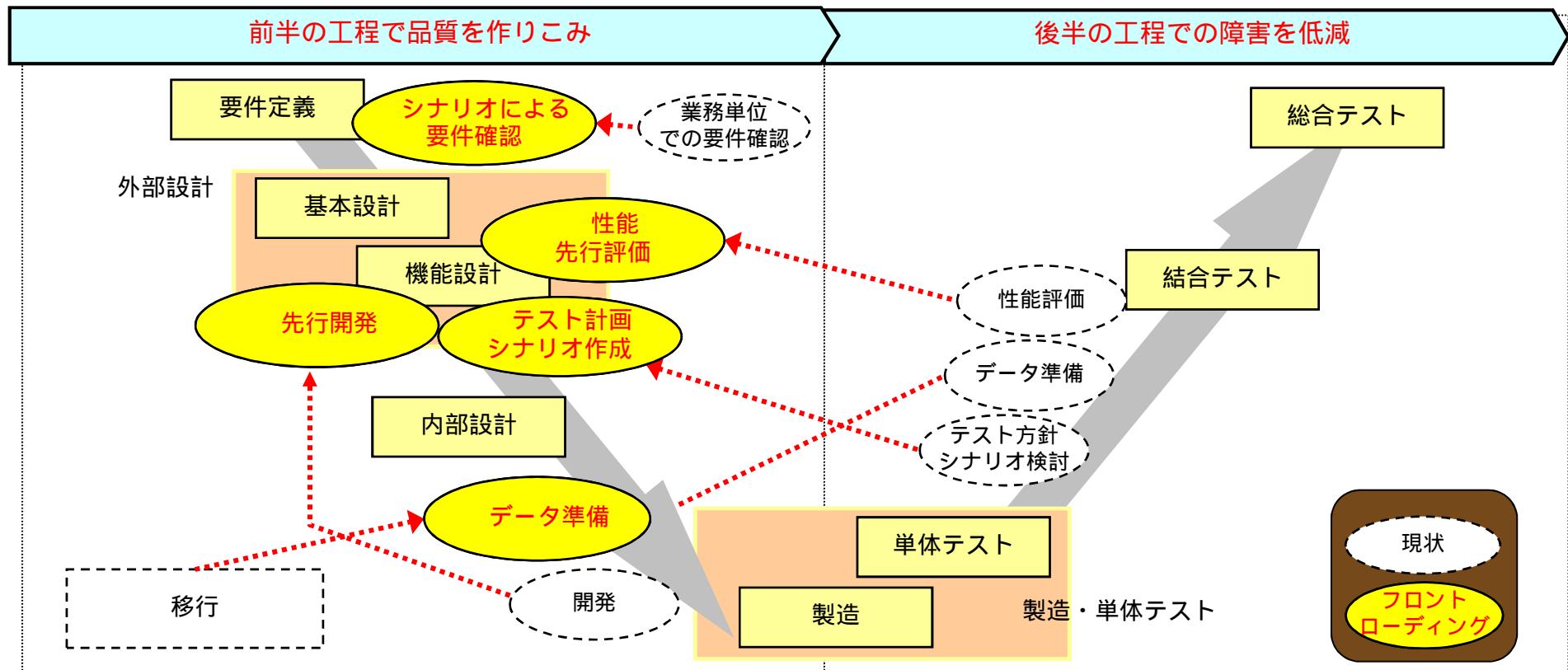
「開発」、「テスト」、「性能」の観点でフロントローディングの考え方を取り込んでいます。

「フロントローディング」とは？

開発の後半で発生する問題を未然に防ぐことを目的に、開発初期でやるべきことを正しく行い品質を作りこむ手法

フロントローディングの5つの特徴

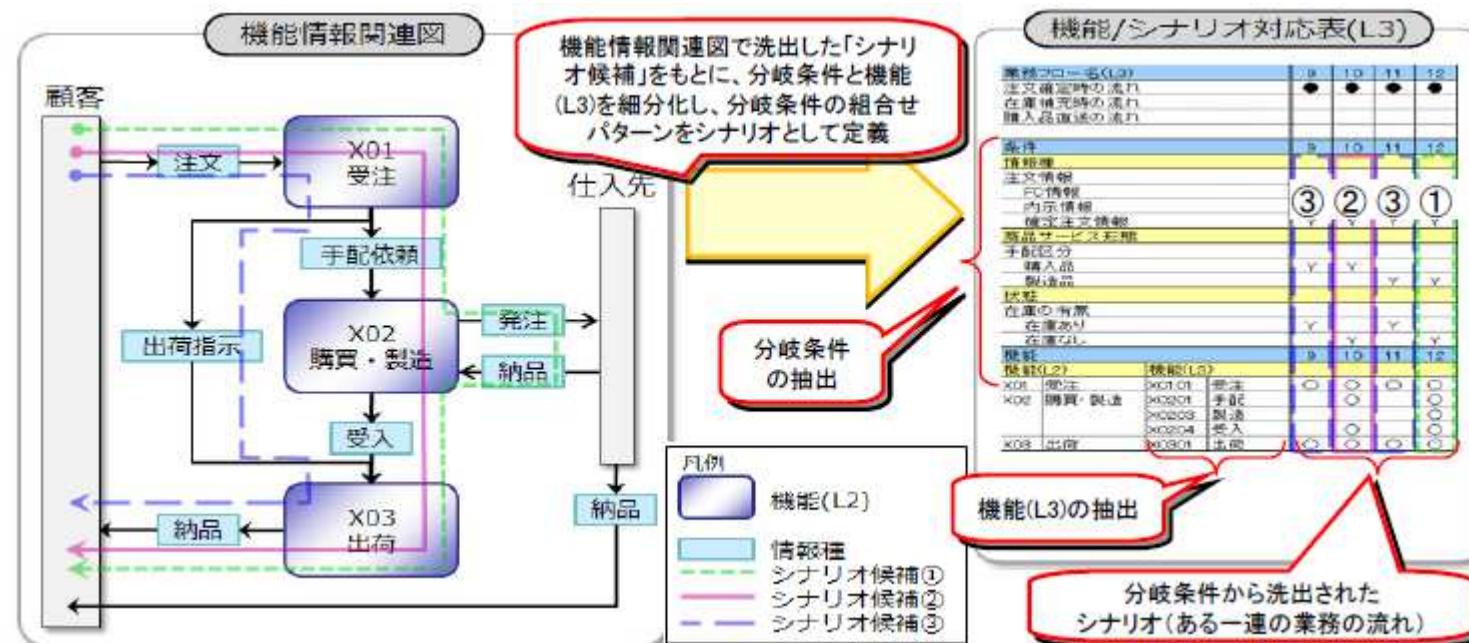
- 開発 : シナリオベースで要件を確認することで、要件の抜け漏れを防止
: 先行開発や、データ準備により品質を確保
 - テスト : 外部設計までに計画 / シナリオを作成し、後工程の品質を作り込む
 - 性能 : プロトタイプ評価やPJを通しての検証活動によって性能を担保



2.2.1. フロントローディング：[開発]

シナリオベースで要件を確認することで、要件の抜け漏れを防止します。

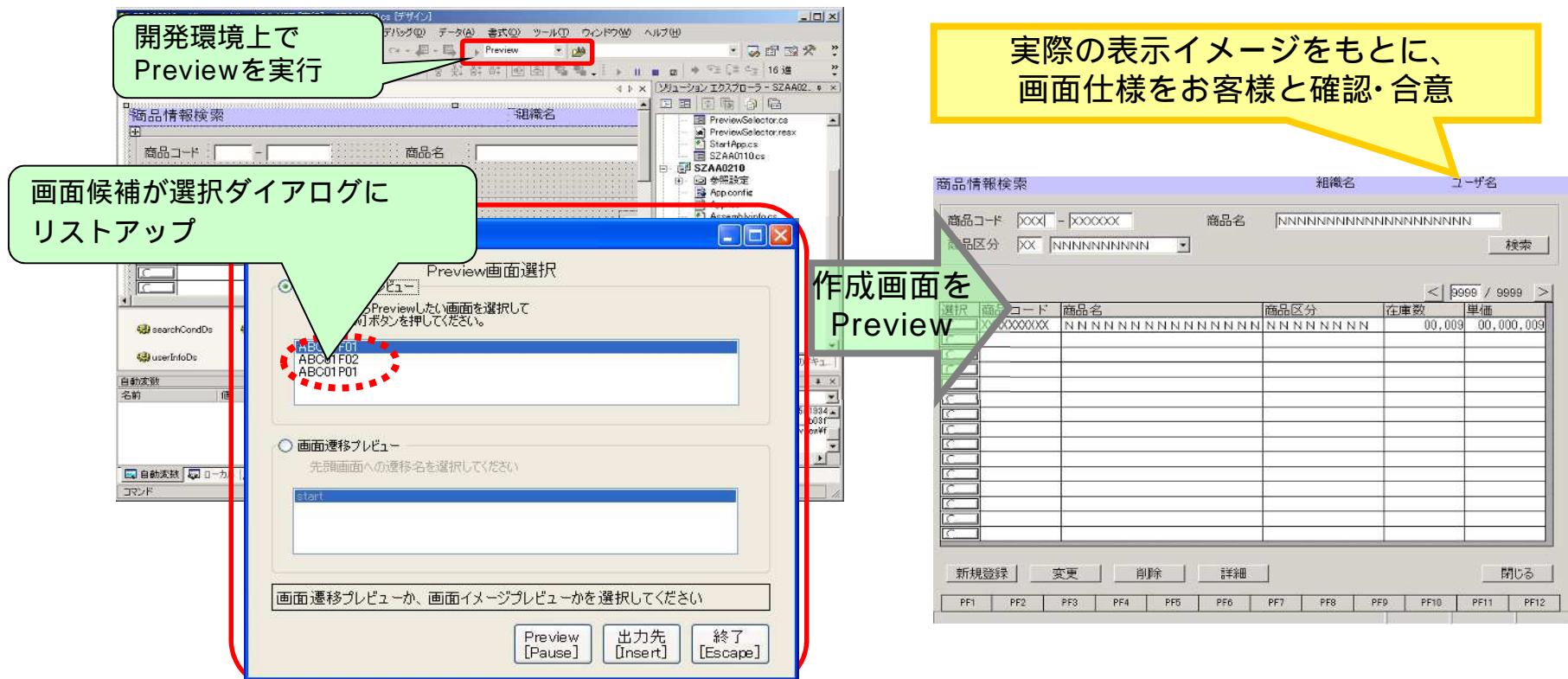
- 情報種や商品形態/販売形態等の条件を洗い出し、シナリオを作成
- シナリオと業務プロセスを鳥瞰する仕組みを導入
- シナリオを通して要件定義内容を確認することで、要件の抜け漏れを防ぐ



2.2.1. フロントローディング：[開発]

PR層（画面/帳票）の仕様を実APで確認することで仕様齟齬を防止します。

- 業務プログラム作成前に詳細な画面イメージや画面遷移の確認が可能
- 画面用プログラムはそのまま業務プログラム開発に引継ぎが可能
- 内部設計以降の手戻りを抑えた効率的な開発が可能



2.2.2. フロントローディング：[テスト]

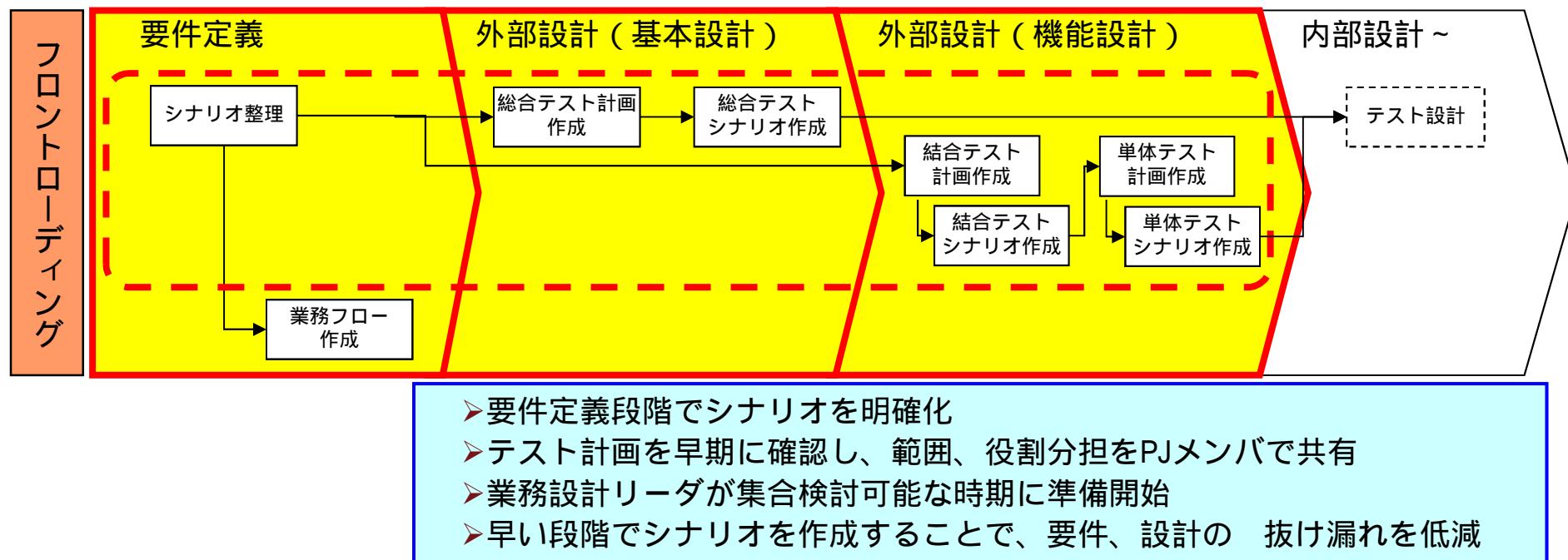
外部設計までにテスト計画 / シナリオを作成し、後工程の品質を作り込みます。

- 従来の問題点

- 従来の方法論は外部設計に「テスト方針の検討」を定義しているが、シナリオ整理のタスクがないため、実際には内部設計以降の対応となり品質面の対策が後手に回ることが多い

- System Director Enterpriseでは、シナリオ作成のアクティビティを定義

- 要件定義～外部設計で「情報種」等の条件を洗い出し、テストシナリオを作成する工程を明確にすることで、フロントローディングによる品質の作り込みを実現する



このスライドではテスト計画に必要な全てのINPUTを明記していません

2.2.3. フロントローディング：[性能]

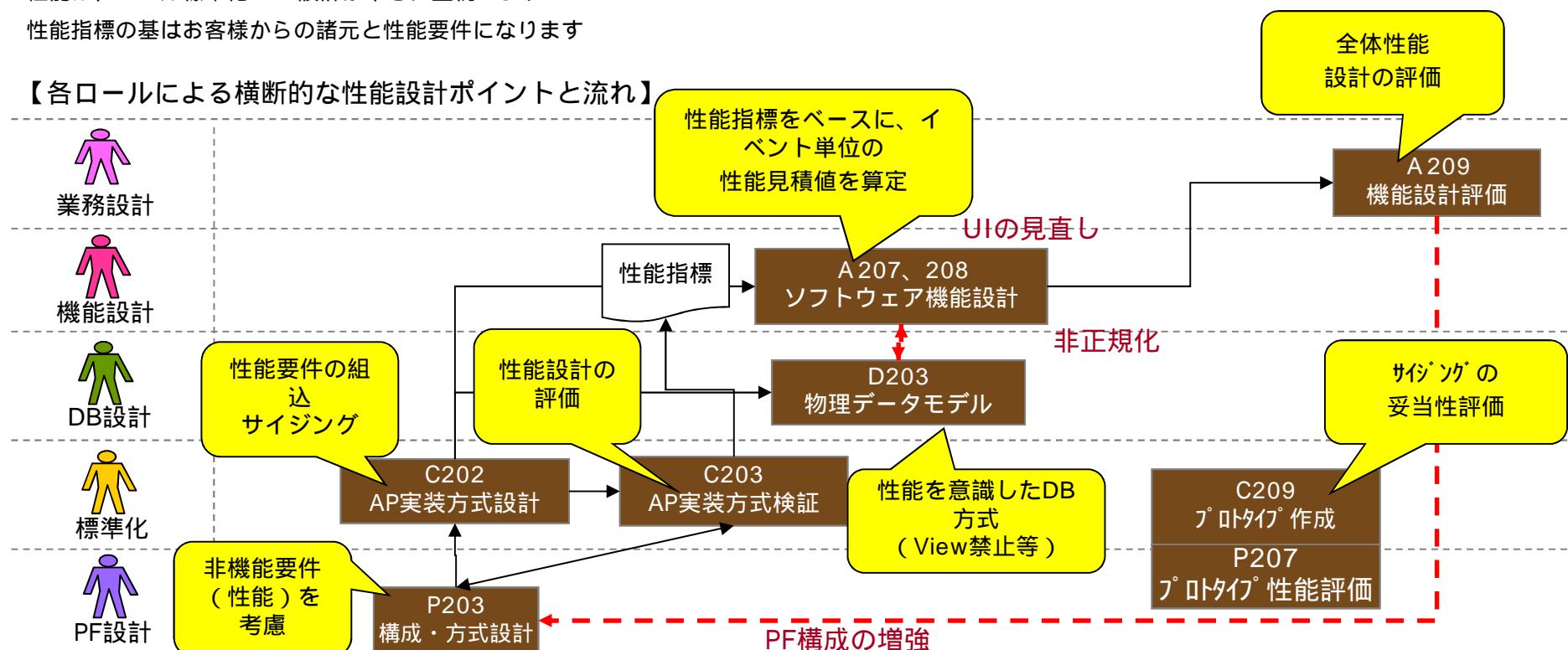
先行評価や性能を作り込む設計時点の検証活動によって性能を担保します。

- 従来の問題
 - ・性能設計のアクティビティが不明確。下流工程で性能問題が発覚する。
- SystemDirector Enterpriseでは、性能設計・検証のタスクを定義
 - ・重要な作業アクティビティにチェックポイントタスクを設定
 - ・チェックシートを活用し、作業を確認
 - ・プロトタイプ評価や実機評価を確實に実施

性能は、ロール標準化とPF設計が中心に監視します

性能指標の基はお客様からの諸元と性能要件になります

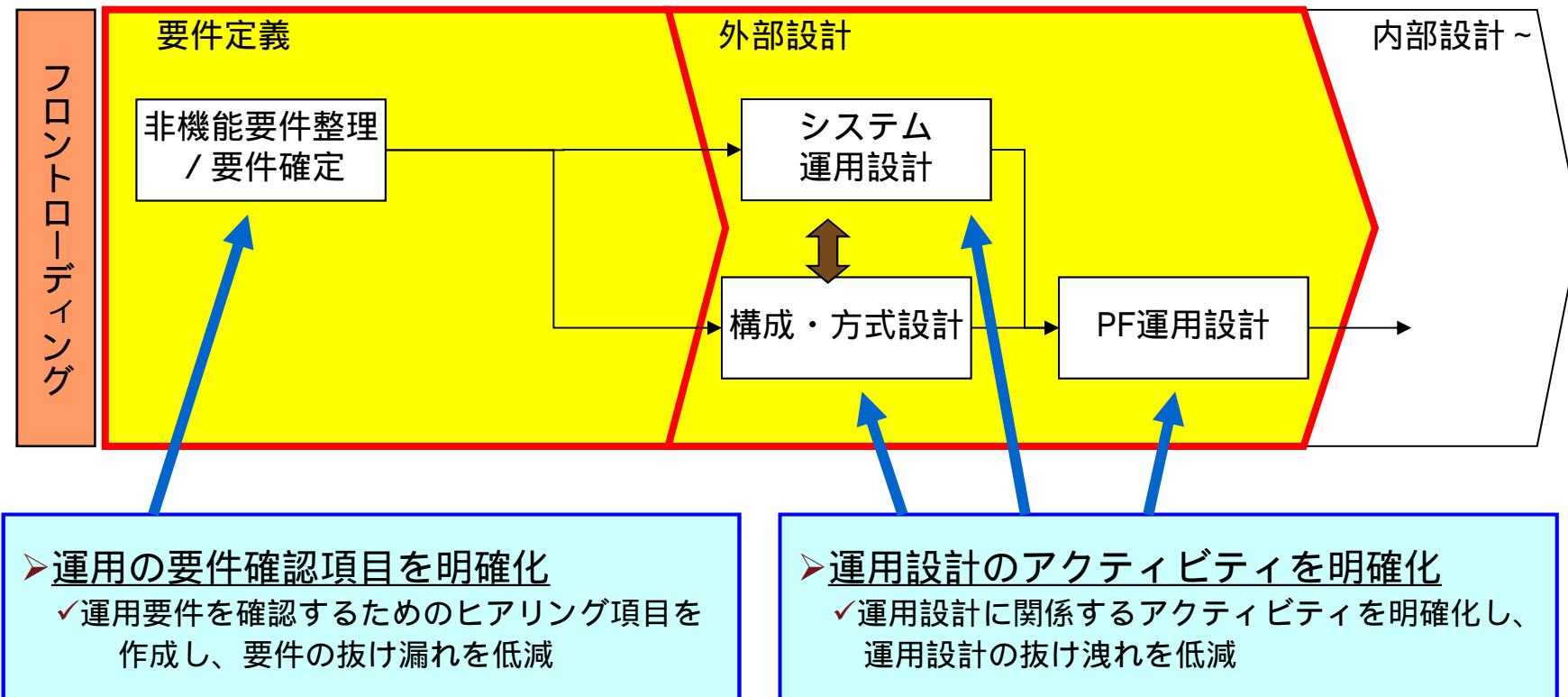
【各ロールによる横断的な性能設計ポイントと流れ】



2.3.運用設計

■ システム開発後の確実なIT運用を実現するために、SI上流での運用設計を強化します。

- ITIL準拠の運用設計を行うために、るべきWBSを標準化
- NECのアウトソーシング部隊の運用ノウハウをもとに、各作業のポイントを説明



2.3.1. 運用設計：[アクティビティの整理]

「運用要件定義」、「標準化」、「業務運用」、「基盤運用」、「運用業務」の観点で運用設計のアクティビティを整理します。

- **運用要件定義**

要件定義工程で運用要件を整理し、設計の抜け漏れを防止

- **標準化**

メッセージ形式やドキュメントを標準化し、運用ミスを防止

- **業務運用**

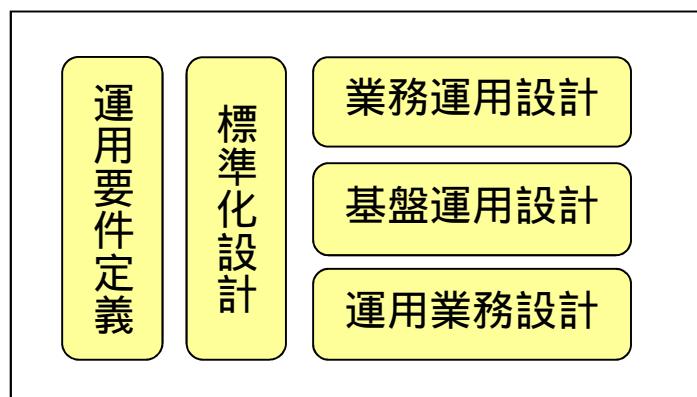
業務運用の視点でAPに運用機能を作り込み、業務運用の漏れを防止

- **基盤運用**

基盤運用の視点で環境に運用機能を作り込み、基盤運用の漏れを防止

- **運用業務**

システム全体の人間系の業務を設計し、運用の漏れを防止



➤ 運用を意識した5つの観点で設計を分類

- ✓ どの役割の人が、どの様な設計作業を行うか確認可能
- ✓ 設計者間での齟齬を防止

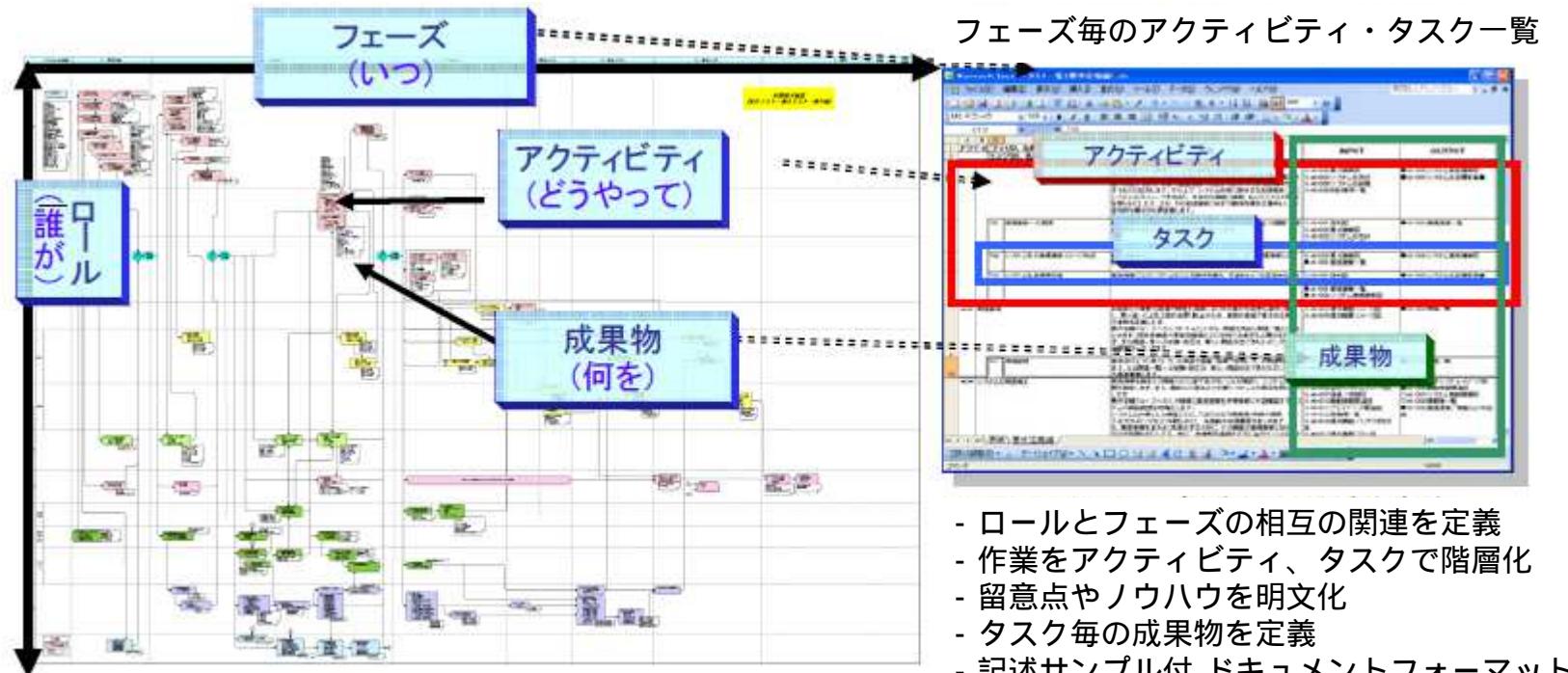
第3章

SystemDirector Enterprise開発方法論の概要

3.1.開発プロセス定義 概要説明

本方法論の開発プロセス全体フローは次のとおりです。要件定義から移行の「フェーズ」において役割「ロール」ごとにやるべき作業「アクティビティ」とアウトプット「成果物」を時系列に定義しています。

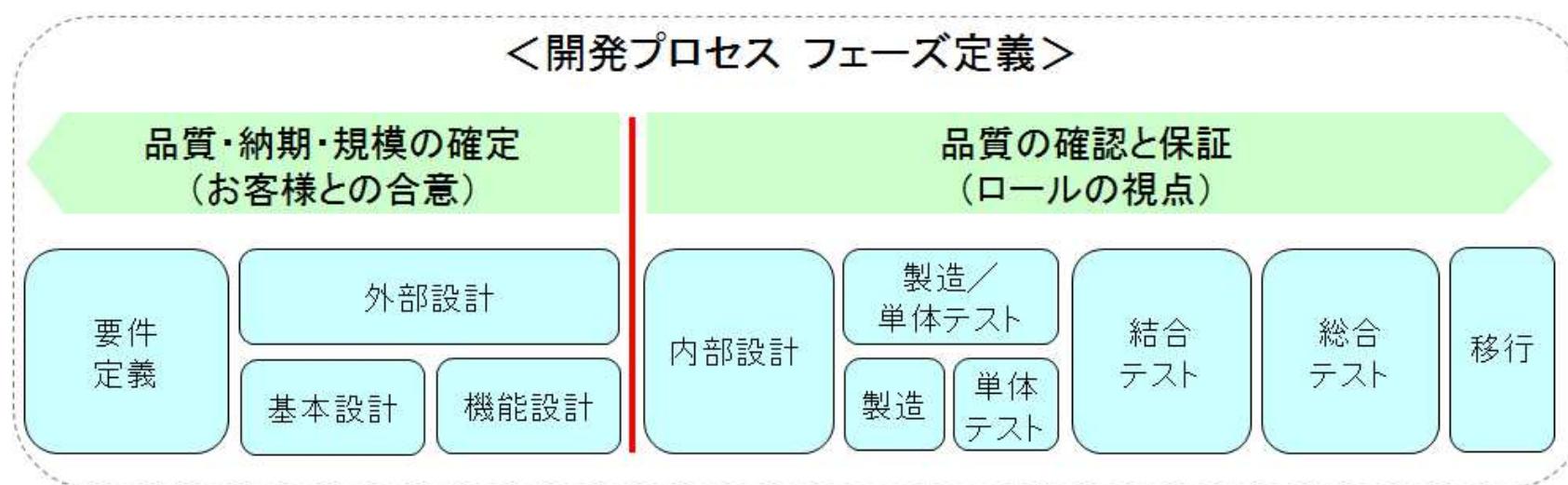
●開発プロセス全体フロー



3.1.1. フェーズ定義 (1/3)

お客様との合意の視点とその仕様を作りこむ視点を定義します。

- お客様との合意の視点でフェーズを定義する
 - ・要件定義、外部設計（基本設計、機能設計）
- 開発主体者（ロール）の切替ポイントでフェーズを定義する
 - ・内部設計、製造・単体テスト、結合テスト、総合テスト、移行



SystemDirector Enterprise では、お客様との合意が必要なフェーズと、ベンダ責任で実施するフェーズに分けています

3.1.1. フェーズ定義（2/3）

■ システム開発工程の作業項目と作業手順の階層化のレベルを次のように定義します。



- フェーズとはプロジェクトの**意思決定の単位**
 - フェーズの成果物をもとに、プロジェクトの意思決定者が以降のフェーズに進むか否かを決定する単位
-
- 相関関係が強い作業（成果物）をまとめた単位
 - アクティビティとしての**成果物をまとめる単位**
-
- アクティビティを構成する作業
 - 作業計画、作業管理を行う際の、分担あるいは管理を行う単位
 - タスクは1週間以下で終了する**作業単位**とし、進捗管理可能な単位

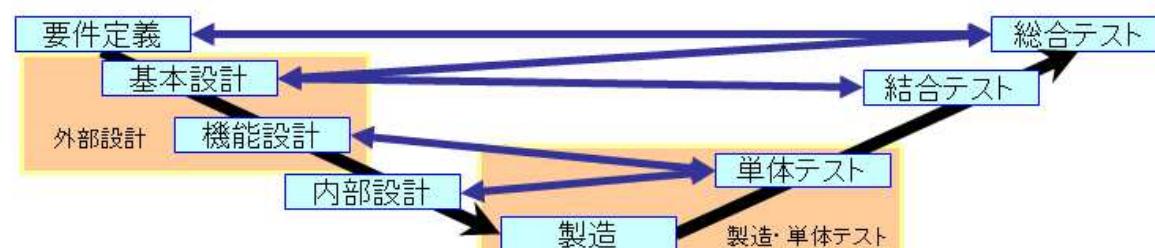
3.1.1. フェーズ定義 (3/3)

プロジェクトにおける意思決定の単位よりシステム開発工程を 7つのフェーズで構成

■本方法論では、フェーズを次のように定義します。

フェーズ	作業概要
要件定義	次期システムに対する要件(機能要件/非機能要件)を漏れなく洗い出し、システム化の範囲を確定する。
外部設計	要件定義に基づき、システムの実現方式を顧客と合意できる最小機能まで分割し、外部仕様(ユーザインタフェース、外部インターフェース、ビジネスロジック、DB仕様等)を設計する。
基本設計	要件定義フェーズの成果物をもとにシステム機能の分析を行い、必要なソフトウェアを全て洗い出す。
機能設計	内部設計へのインプット資料として必要十分なレベルまで、外部仕様を具体化する。
内部設計	外部仕様に基づき、システムの内部仕様(AP-PR層インタフェース仕様、PR層仕様、AP層仕様、バッチ仕様)を設計する。
製造・ 単体テスト	プログラミング、デバッグを行った上で、機能単位で適切に動作するか、性能要件を満たしているかを確認するため、単体テストを実施する。
製造	詳細仕様(外部仕様、内部仕様)に基づき、アプリケーションのプログラミング、デバッグを行う。
単体テスト	実装状態を機能単位(外部設計の最小単位)で、要件通りに動作することを確認する。
結合テスト	単体テスト実施済みのアプリケーションを結合し、アプリケーション間のインターフェースが適切に機能することを確認する。
総合テスト	利用者の視点で、業務運用に問題がないことを確認する。
移行	移行計画書に基づき、本番切替を実施する。

□V字モデルにおける各フェーズの関係



3.1.2.役割（ロール）定義

本方法論で定義しているロールは以下の通りです。

- 開発環境の複雑化、製造工程のオフショア活用、納期短縮化が進み、開発現場では役割別の分業が進んでいます。その実態に則して、役割（ロール）ごとにアクティビティを定義しています。

ロール名	概要
 業務設計	業務に精通し、お客様の要求仕様を分析して、要件の確定およびテスト計画作成、システム設計への橋渡しを担当します。 要件定義フェーズ、基本設計フェーズで中心的な役割を果たします。
 機能設計	業務を理解し、利用するインフラ・FWなどシステム環境を適用したソフトウェア機能設計を担当します。 下流工程との整合性を取れるレベルの実装知識も必要です。
 実装設計	内部設計フェーズで、アプリケーションの実装に関する設計を中心に担当します。 フレームワークや言語特性に合わせたアプリケーション実装設計を行います。
 製造	アプリケーションの実装を中心に担当します。 製造・単体テストのフェーズで中心的な役割を果たします。
 テスト	各テストの計画から設計、実施を担当します。 外部設計フェーズから総合テストフェーズまででテストに関する中心的な役割を果たします。
 DB設計	データベースの設計で中心的な役割を果たします。 要件定義フェーズ、外部設計フェーズで、データ分析・データモデルの最適化設計（概念/論理/物理）を実施します。
 移行設計	本番移行（システム、データ）に関する方針、計画、ツール設計、構築を担当します。
 標準化	PJで採用する開発方針・システムの処理方式の決定、フレームワーク適用などPJ全般で設計実装に関する方針決定作業を担当します。設計標準規定、AP実装方式設計・検証、共通コンポーネント設計、AP環境設計で中心的な役割を果たします。
 PF設計	主にHW、SW(OS、RDBMS、ミドルウェア)の設定などシステム基盤構築、環境・運用面を中心に担当します。
 運用設計	運用で中心的な役割を果たします。主に、インシデント／問題管理や変更／リリース管理など人間系運用業務のプロセス設計を担当します。
プロジェクトマネージャ	プロジェクトを遂行するプロジェクトオーガナイザ

ロールDB設計、移行設計、標準化は、ロール業務設計、機能設計、実装設計と連携を密に取りながら作業します

ロールとフェーズの関係については、『開発プロセス全体フロー図』を参照してください。この関係を意識してPJの体制を組む必要があります

3.1.3. 成果物定義 (1/2)

本方法論の、成果物は次のとおりです。

成果物とは

プロジェクトにおいて、各アクティビティ、各タスクで発生する具体的で検証可能な作業結果であり、アクティビティおよびタスク間を連携させる情報である。

成果物には、以下のものがある。

- ドキュメント
- プログラム(ソース、モジュールなど)

ドキュメントには、ドキュメントとワークドキュメントがある。

以下に、ドキュメントの内容を示す。

ドキュメントとは

各アクティビティで作成する設計書や仕様書の構成要素である。

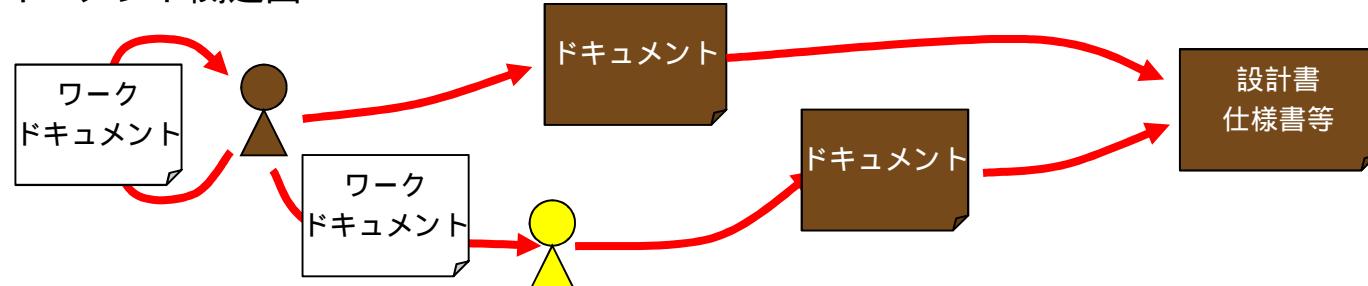
ドキュメントには、～一覧、～対応表、～図、～定義などがある。

ワークドキュメントとは

ドキュメントを作成する為の確認情報や、情報伝達の為に作成する一時的な文書である。

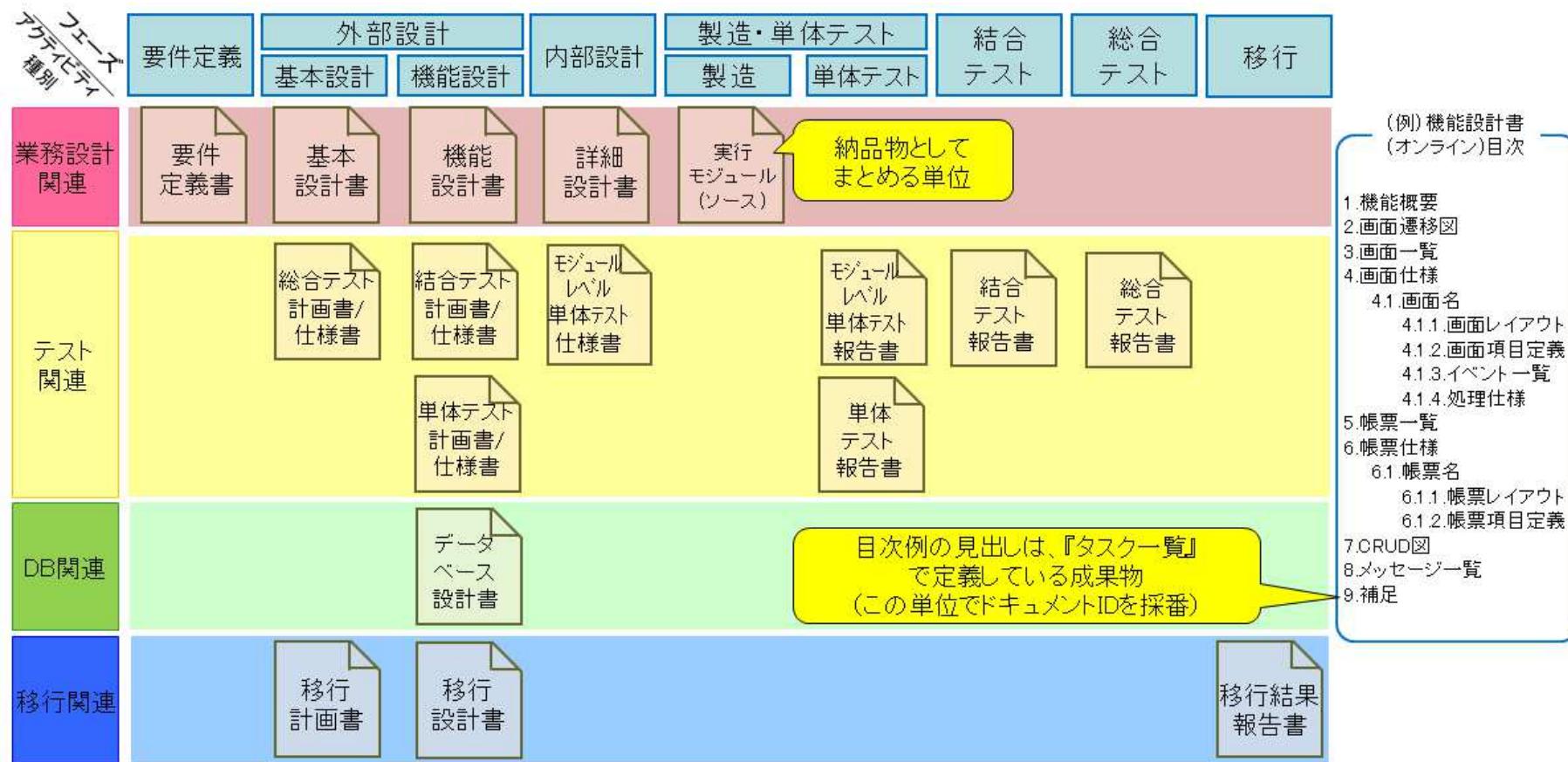
ワークドキュメントには、～ワークシート、～候補一覧などがある。

成果物ドキュメント関連図



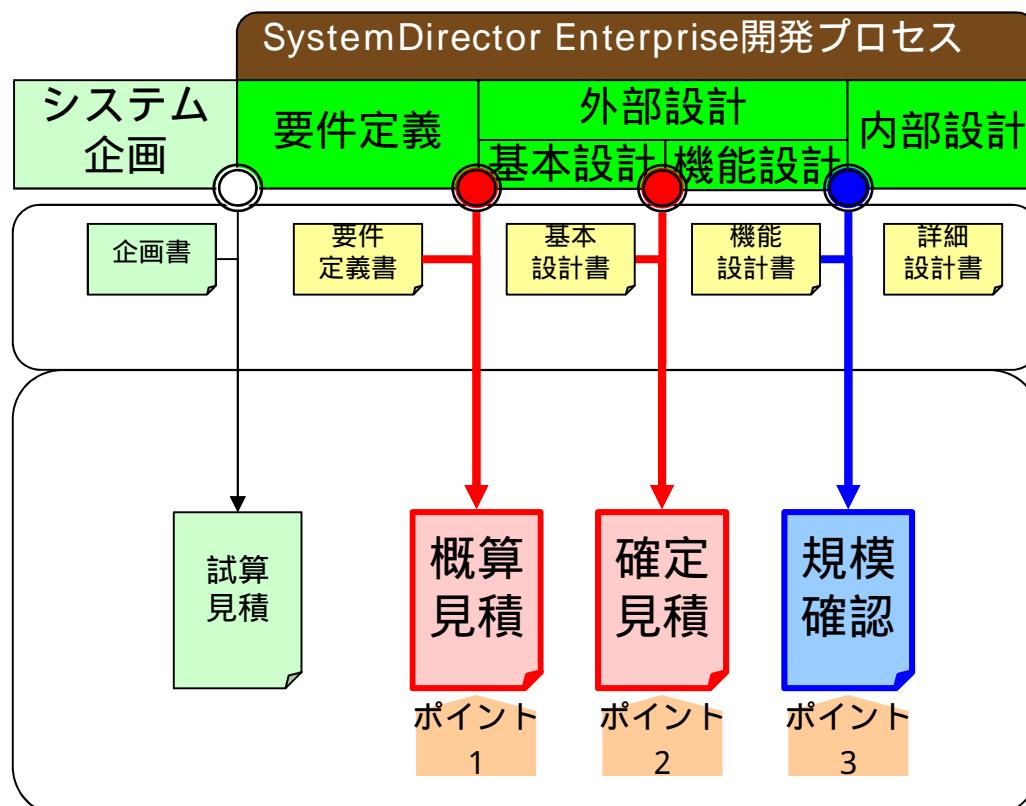
3.1.3.成果物定義 (2/2)

以下は、各フェーズのドキュメントを纏めたものです。



3.1.4. 見積ポイント

本方法論では、見積品質を高めるために開発プロセスにおける見積ポイントを3つ定義しています。(SEC BOOKSから提供されているガイドを参考に定義)



納期遅延や開発規模の増加等を予防するために、提案時の規模見積（試算見積）との差異を確認しながらプロジェクトを遂行していくことが重要です。試算見積との差異がある場合には、原因を明らかにした上で、お客様と規模の見直し（機能仕様や作業分担の変更など）を行う必要があります。本開発方法論では、以下の3つの見積ポイントを設定しています。

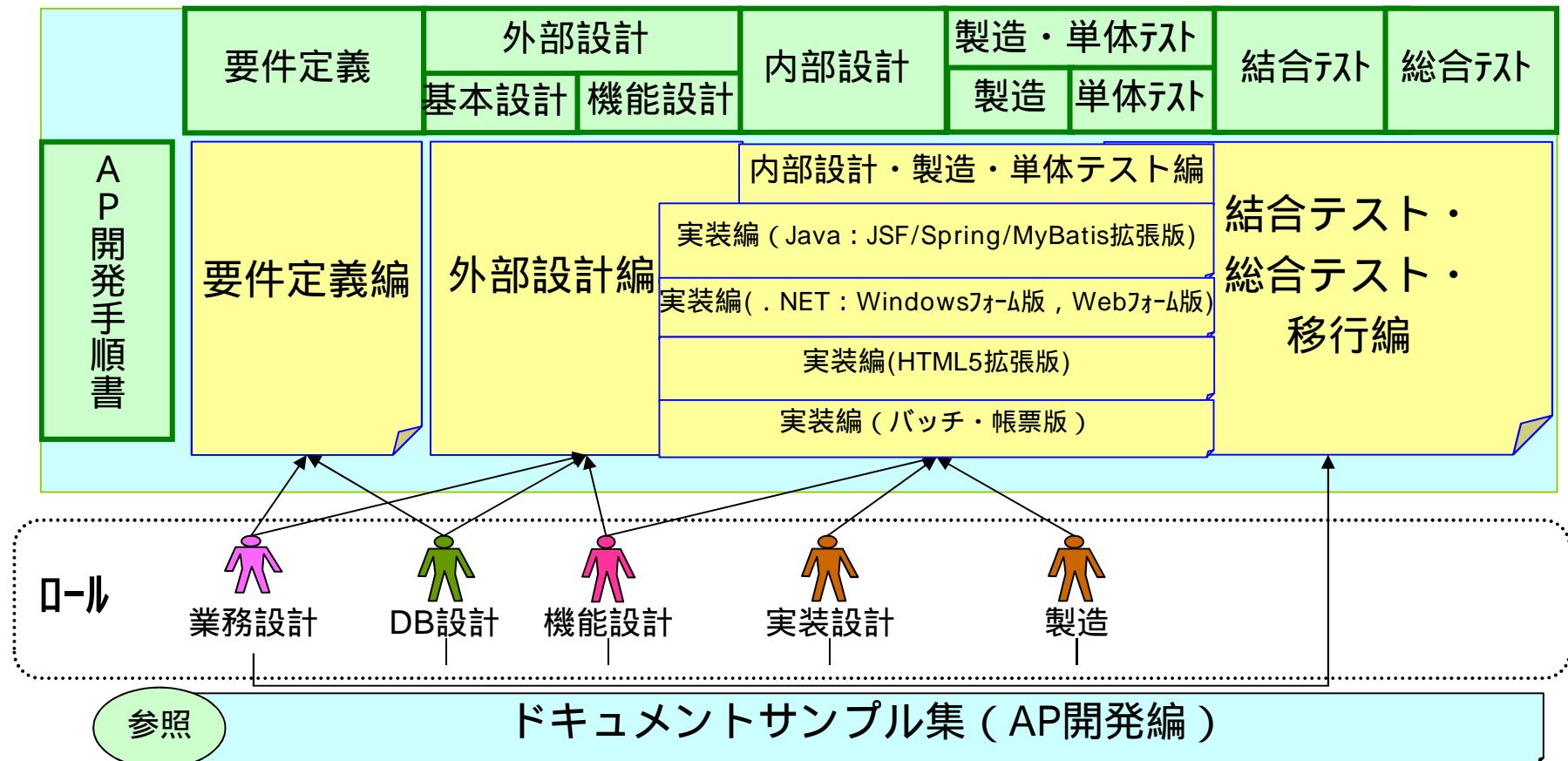
- **ポイント1：概算見積**
業務軸から洗い出したシステム候補の機能数をもとに概算見積を行います。
- **ポイント2：確定見積**
基本設計のシステム機能数をもとに確定見積を行います。
本ポイント以降、変更管理の対象となります。
- **ポイント3：規模確認**
各システム機能の深さ・重みを考慮した規模確認を行います。
お客様と最終的な開発規模の合意を行います。

本見積のポイントは見積実施可能タイミングを示すものです。
お客様に見積提示をするタイミングを示しているものではありません。

システム企画フェーズ、および試算見積はSystemDirector Enterprise開発プロセスの対象外

3.2.手順書 概要説明

手順書はフェーズごと、および開発環境ごとに分冊化した11冊で構成します。
併せてドキュメントサンプル集を提供します。



3.2.1.AP開発手順書の概要：要件定義編

要件定義の概要

要件定義フェーズでは、前工程であるシステム企画フェーズで明らかになった重点施策の内容を元に、トップダウン方式にて次期システムに対する要件（機能要件 / 非機能要件）を明らかにします。

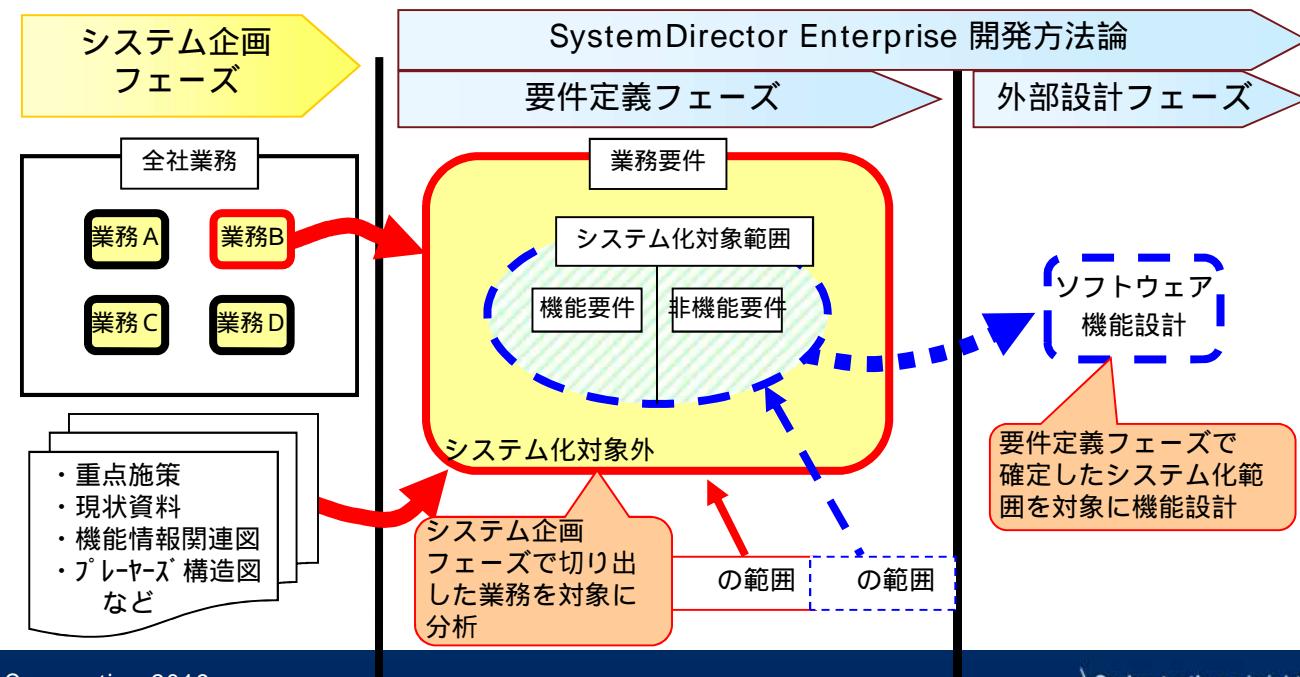
業務要件の明確化

システム利用部門の業務の流れを、重点施策の内容を考慮しながら業務要件として明らかにします。

業務要件を明らかにする上では、人が実施する業務の手順に着目し、商品やサービス、取引先による業務の違いを、抜け漏れなく明らかにすることが重要です。

システム化対象範囲の明確化

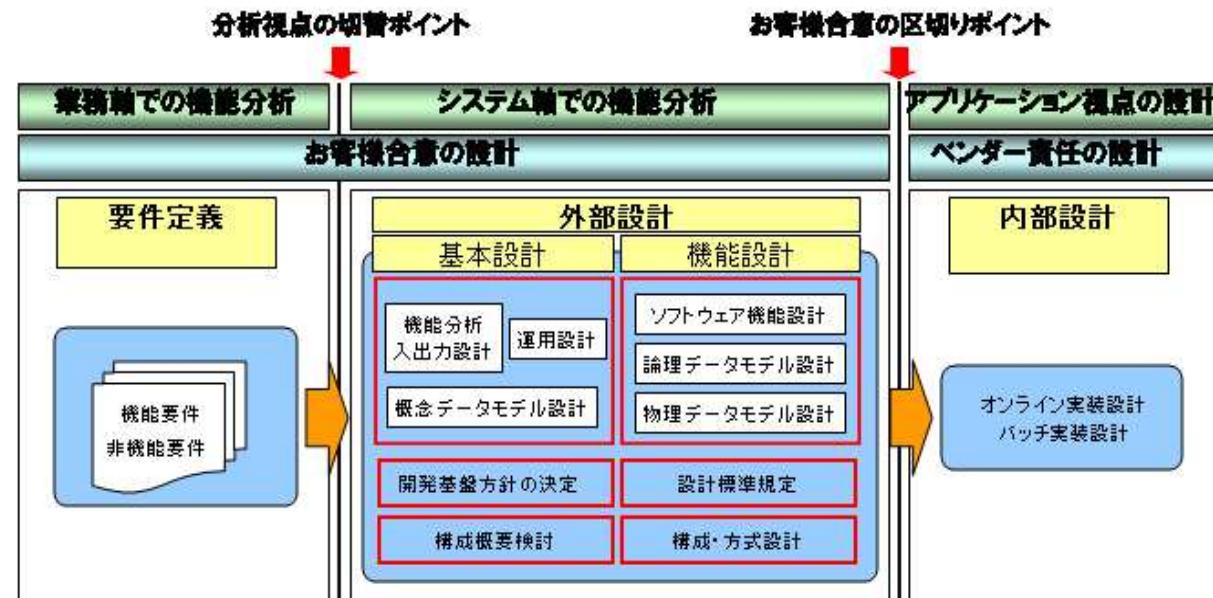
業務要件で明らかになった機能をシステム化する機能とシステム化しない機能に分け、今回のシステム化の範囲を明らかにします。システム化する機能については、機能概要、画面 / 帳票イメージを確認し、システム化に必要な機能要件を定義します。また、性能や信頼性、拡張性、セキュリティなどを非機能要件を定義します。



3.2.2.AP開発手順書の概要：外部設計編

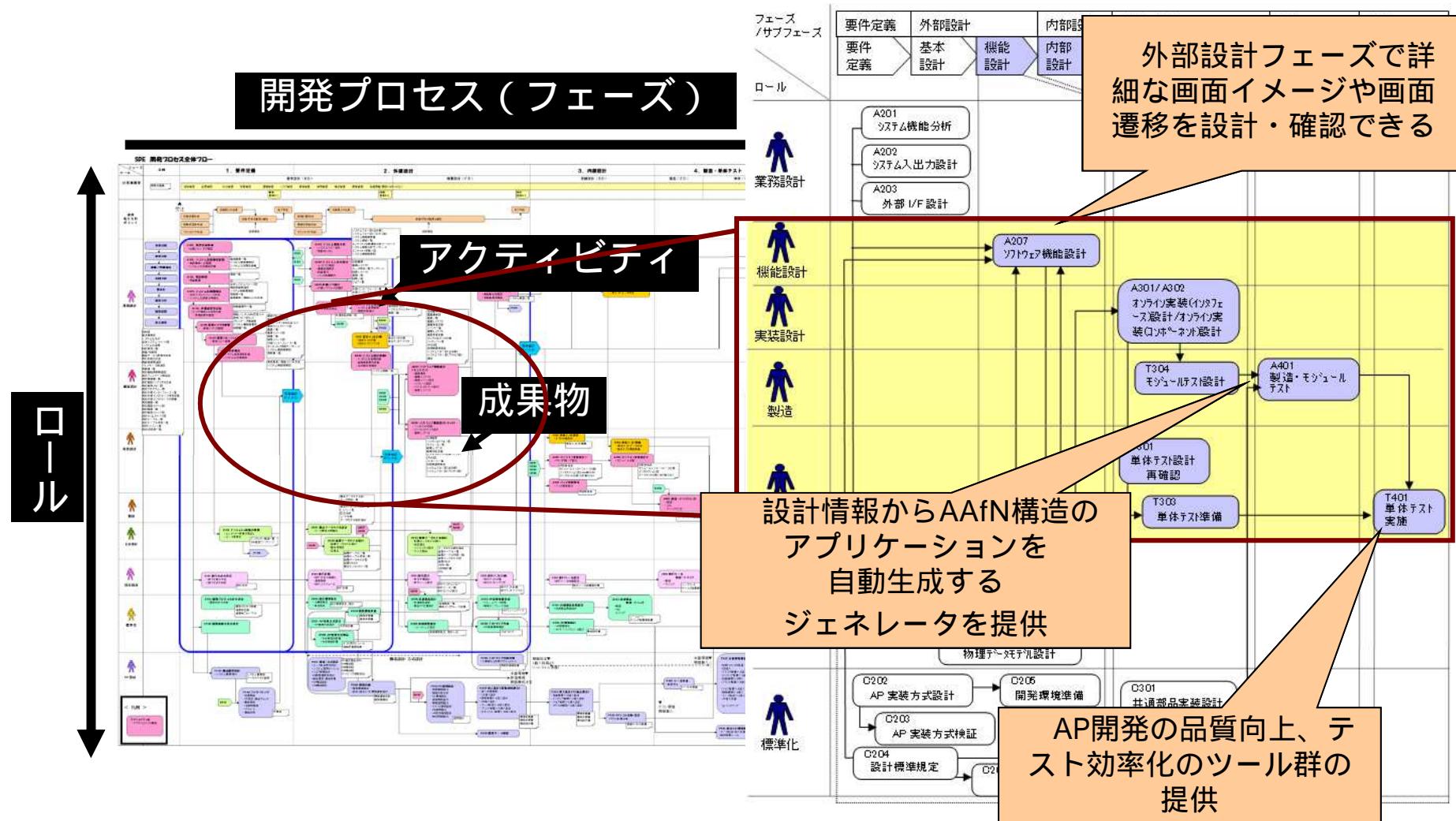
外部設計フェーズとは

- 外部設計フェーズは、要件定義フェーズで明らかにした要件にもとづき、開発するシステムが外部仕様としてシステム利用者や外部システムにどのようなインターフェースを提供するのか、どのようなビジネスロジックが必要かを設計する工程です。
- 外部設計フェーズには、基本設計フェーズと機能設計フェーズがあります。
 - 基本設計フェーズでは、システム機能の分析を行い、必要なソフトウェアを洗い出します。基本設計の結果を基本設計書としてまとめ、システム機能数をもとに確定見積を行い、機能設計以降のフェーズに入る意思決定を行います。
 - 機能設計フェーズでは、内部設計フェーズへのインプット資料として必要十分なレベルまで、外部仕様を具体化します。機能設計の結果を機能設計書としてまとめ、各システム機能の深さ・重みを考慮した規模確認を行い、内部設計以降のフェーズに入る意思決定を行います。



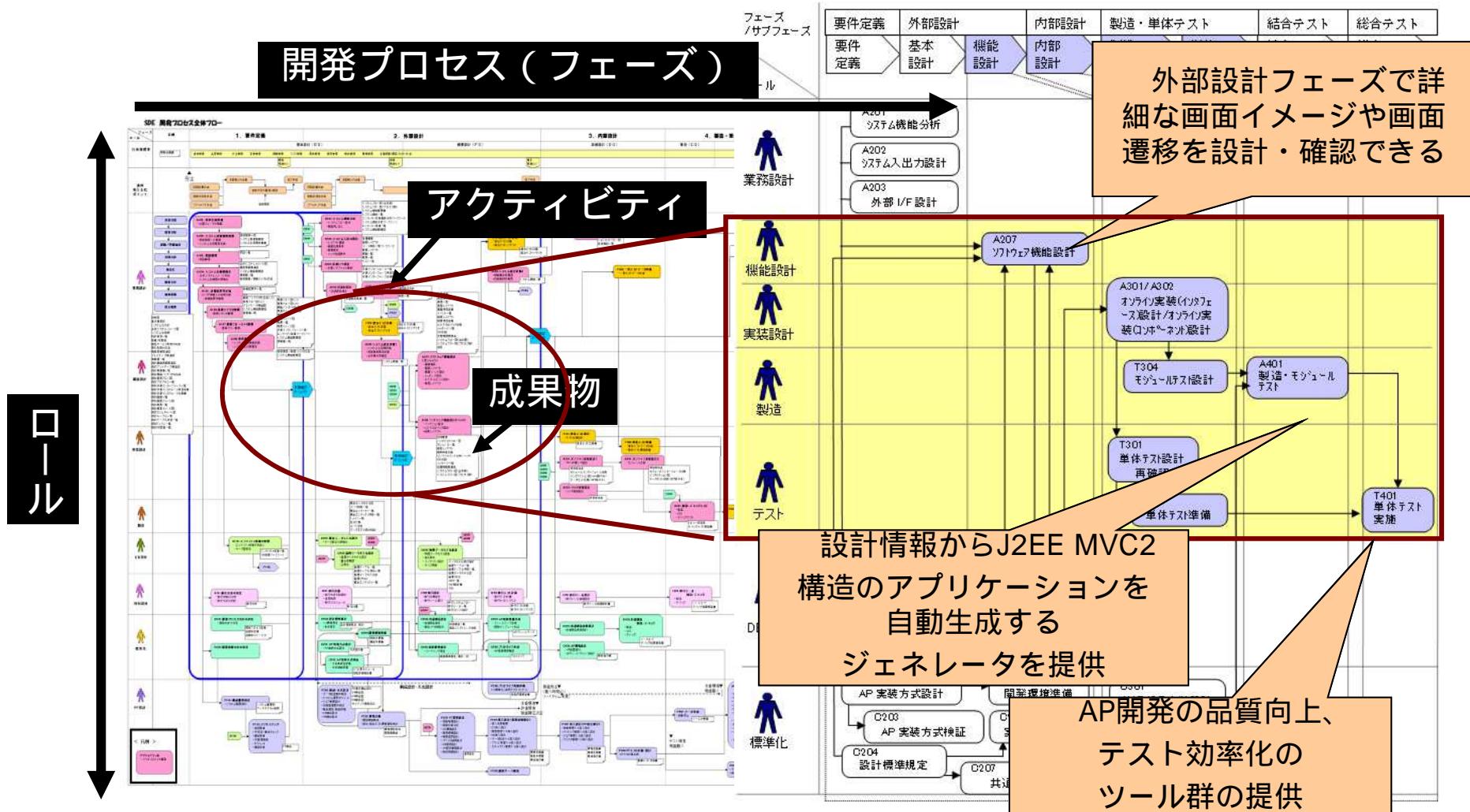
3.2.3.AP開発手順書の概要：実装編- .NET版

外部設計後半から単体テスト完了までをカバー
生産性向上、品質向上の開発ツール群を提供



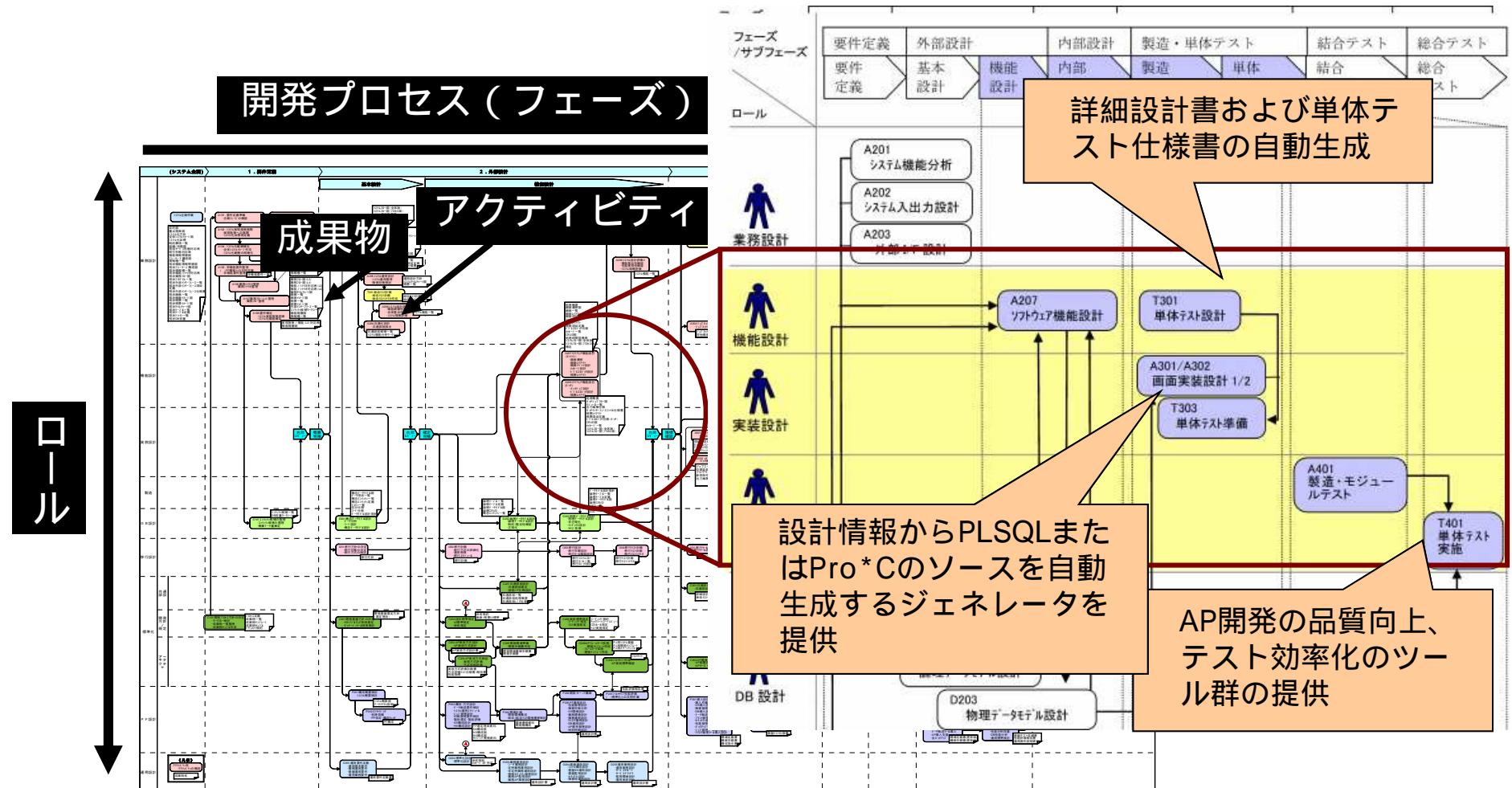
3.2.4.AP開発手順書の概要：実装編-Java版

外部設計後半から単体テスト完了までをカバー
生産性向上、品質向上の開発ツール群を提供



3.2.5.AP開発手順書の概要：実装編- バッチ・帳票版

機能設計から単体テスト完了までをカバー
生産性向上、品質向上の開発ツール群を提供



3.2.6.AP開発手順書の概要：各テスト編

要件に応じた実施すべきテストの種類をフェーズと検証視点別に定義

- 定義した**テストの種類**をベースにする事で**テスト計画を立てやすくなる**
- テストを計画的に実施することで**検証漏れ**による**後戻りを抑制できる**

(凡例)
■ : 業務系のテスト
■ : PF系のテスト

フェーズ		設計	製造	単体テスト	結合テスト	総合テスト
視点	機能要件	業務系	PF系	業務系	PF系	業務系
機能要件	業務系			コードレビュー JOBSTEPテスト JOB機能テスト 画面機能テスト	チーム内結合テスト チーム間結合テスト 外部システム結合テスト 排他テスト	業務テスト 異常系業務テスト
非機能要件	業務系	システム制約確認 プロトタイプ (特定製品動作テスト) 適合性確認		APレスポンステスト AP性能評価(処理能力確認、リソース余裕率確認) バックアップテスト データ移行 移行リハーサル	業務運用(通常系) 業務運用負荷テスト(通常系) 業務運用負荷テスト(ピーク時) システム運用(通常系) システム運用(異常系)	
非機能要件	PF系			単体テスト (運用性、可用性、性能、 拡張性、連携性)	基盤結合テスト (運用性、可用性、性能、 拡張性、連携性)	運用テスト (通常、障害、セキュリティ、 性能、拡張、連携)

3.2.7.AP開発手順書の概要：移行編

要件定義段階から移行方針を検討し、確実な本番切替を実施

- 要件定義フェーズ: 移行方針の決定

- 非機能要件整理(A105)で整理した要件をもとに移行方針を決定

- 基本設計サブフェーズ: 移行計画の立案

- 移行方針の決定内容を詳細化し、移行計画を立案

- 機能設計サブフェーズ: 移行設計

- 移行計画に基づき、移行の具体的な作業項目、順序を明確化
- 移行ツールの機能設計

- 内部設計・製造・単体テストフェーズ: 移行ツールの作成

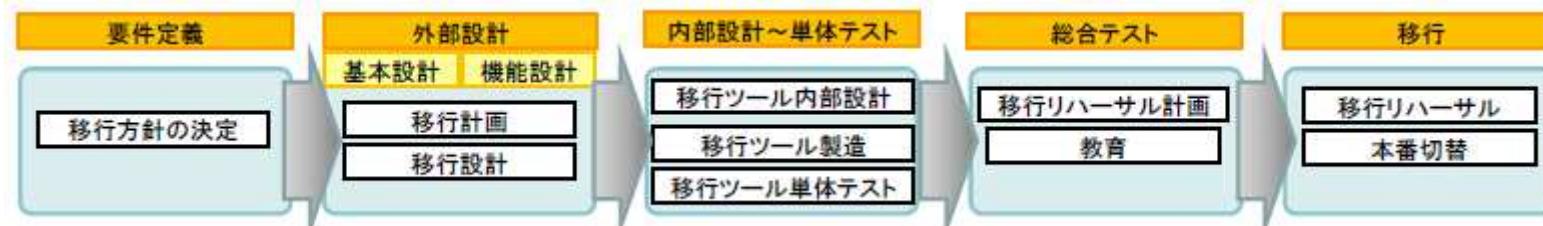
- 移行ツールの詳細設計・製造・テスト(業務アプリケーションと同様の作業)

- 総合テストフェーズ: 移行準備

- 移行リハーサル計画の立案
- 利用者教育の実施

- 移行フェーズ: 本番移行

- 移行リハーサル、本番切替を実施



内部設計～単体テストでは移行ツールの設計や移行ツール作成、移行ツールの単体テストを行います

第4章 サポートサービス

4.1.お問い合わせ先

●ご購入前のお問い合わせ

NEC SystemDirector Enterprise お問い合わせ窓口

- Web

<http://jpn.nec.com/SystemDirectorEnterprise/contact.html>

記載されている会社名、製品名は、各社の登録商標または商標です。

- SystemDirector Enterprise, InfoFrame は日本電気株式会社の登録商標です。
 - SVF, SVFX-Designerは、ウイングアーク 1 s t 株式会社の登録商標です。
 - Windows, Office, Excel, Visual Studio, .NET Framework はMicrosoft Corporationの米国およびその他の国における登録商標です。 また、Windows の正式名称は Microsoft Windows Operating System です。
 - JavaはOracle Corporation の米国およびその他の国における登録商標です。
 - Amazon Web Services, ‘Powered by Amazon Web Services’ロゴ、その他のAWS商標はAmazon.com, Inc.の米国およびその他の国における登録商標です。
 - Log4jはThe Apache Software Foundationの登録商標です。
 - Eclipseは Eclipse Foundation, Inc. の米国およびその他の国における登録商標です。
 - AndroidはGoogle Inc. の米国およびその他の国における登録商標です。
 - iOS はCisco Systems G.K.の米国およびその他の国における登録商標です。
-
- その他、記載されている会社名および製品名は、各社の商標または登録商標です。

\Orchestrating a brighter world

NEC