

参考資料6:アーキテクチャー設計の考え方と成果物のイメージ

社会保険庁

本資料の位置づけについて

- 本資料は、「アーキテクチャ設計及びプラットフォーム性能検証等業務一式」を委託するに当たり、応札者に作業の概要をより具体的に掴んで頂くために作成した資料です。
- 成果物の案等については、当庁で想定している一例です。

目次

■ **アーキテクチャー設計の考え方**

- アーキテクチャー設計およびプラットフォーム性能検証の位置づけ P.1
- アーキテクチャー設計について P.2
- コンポーネント及びコンポーネントの共通化について P.3
- 補足資料
 - 開発工程と使用モデルの関係 P.4
 - アーキテクチャーとは P.5
 - アプリケーション・アーキテクチャーの考え方 P.6

■ **アーキテクチャー設計 成果物内容(案)**

- アーキテクチャー設計の成果物一覧 P.7
- アーキテクチャー設計における成果物の位置づけ(体系) P.8
- アーキテクチャー仕様書とは P.9
- 業務処理パターンとは P.15
- コンポーネント設計パターンとは P.19

アーキテクチャー設計の考え方

アーキテクチャ設計およびプラットフォーム性能検証の補完工程における位置づけ

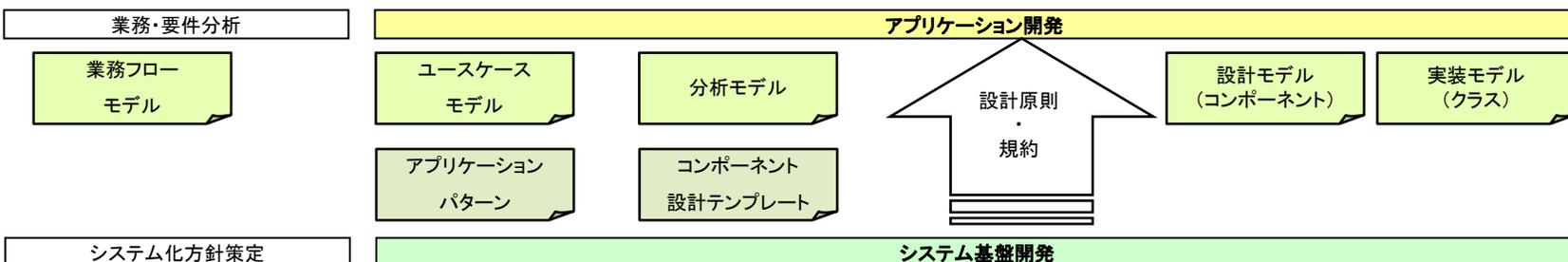
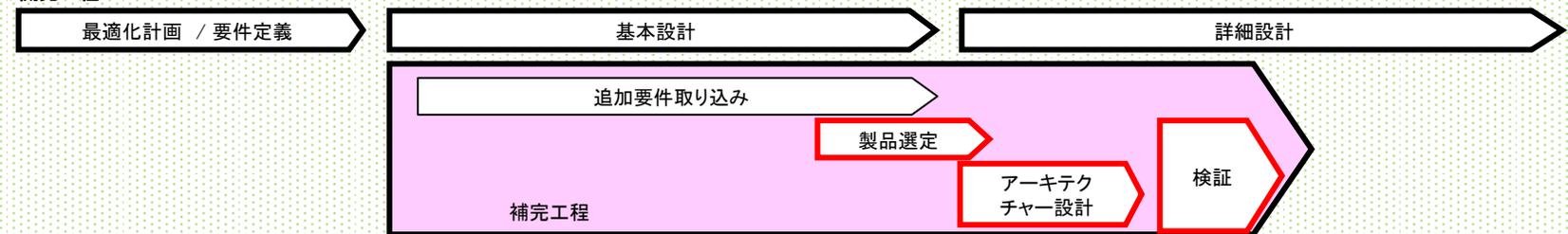
○アーキテクチャ設計

- アーキテクチャ設計は、システムの骨格・作り方を定めるものであり、アプリケーション開発を本格的に実施するまで(詳細設計開始まで)に実施すべきものである。
- アーキテクチャとして、システムを分割・構造化し、設計原則・規約を策定する。
- システム基盤開発と、アプリケーション開発が並行に実施できるよう、両者の境界を定める。

○プラットフォーム性能検証

- 本システムに求められるアーキテクチャを設計し、円滑に詳細設計を開始するためには、本システムで採用するプラットフォームや製品を決定する必要がある。
- 製品を含めた「システム全体のアーキテクチャの妥当性」を確認するとともに、「プラットフォームの性能」を検証する。

当システムの開発工程



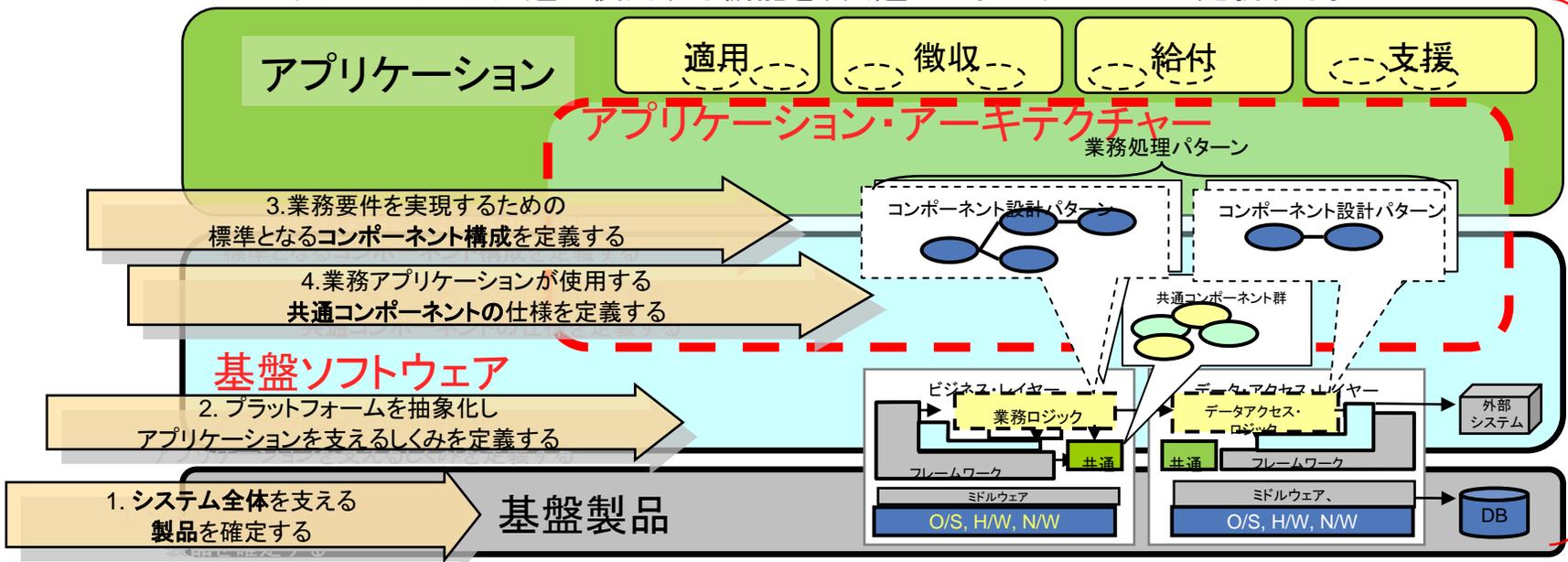
アーキテクチャ設計について

■ アーキテクチャ設計の目的

- 詳細設計において、円滑な業務の設計を行うための下地を整えるために補完工程においてアーキテクチャ設計を実施する。
- アーキテクチャ設計の結果、業務アプリケーション開発の属人性を排除する。

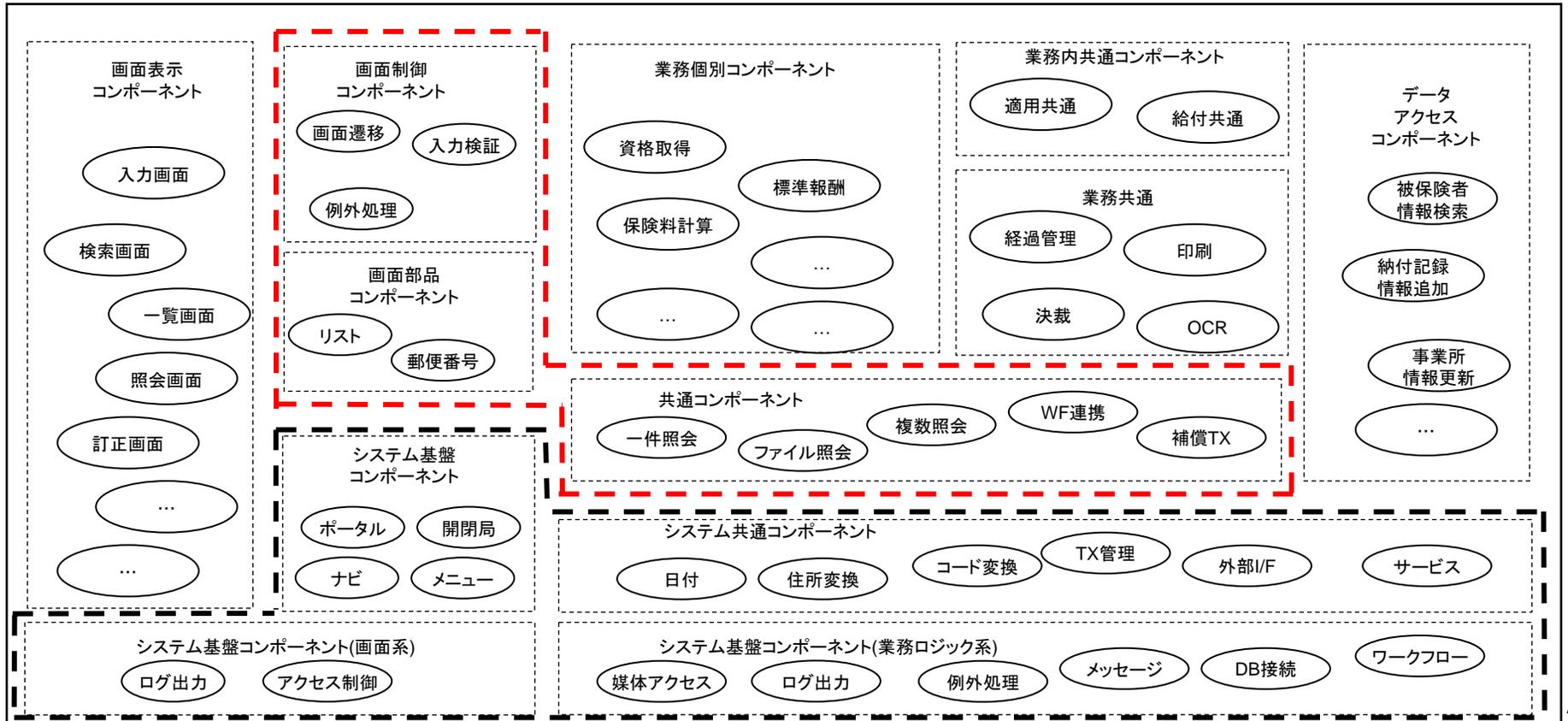
■ アーキテクチャ設計の内容

- システム全体のアーキテクチャを確定する。
- システムを支える“**基盤製品**”の確定をする。
- 基本設計に引き続き、基盤製品を隠蔽するしきみを、“**基盤ソフトウェア**”として確定する。
- “**アプリケーション・アーキテクチャ**”として、要件・非機能要件を考慮した、アプリケーション設計のためのルールを確定する。
- アプリケーションが共通に使用する機能を、**共通コンポーネント**として定義する。



コンポーネント及びコンポーネントの共通化について

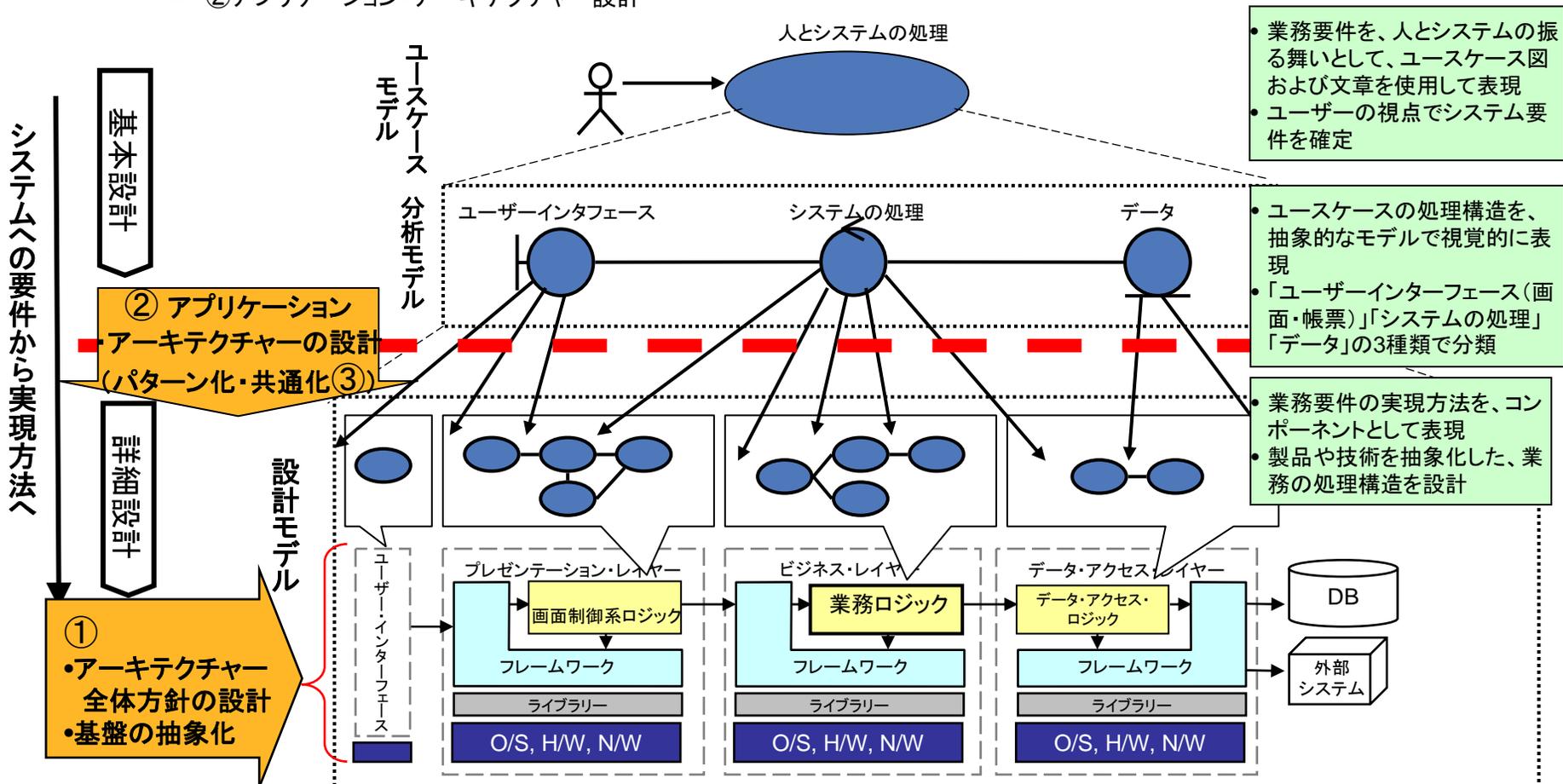
- ▷ 当システムを構成するコンポーネントの例を下に示す。黒点線の枠で示した部分は基本設計工程において、基本設計を行った範囲である。補完工程では残りのコンポーネントについて検討を実施する。
- ▷ 補完工程では業務アプリケーションを構成するコンポーネントのうち、下記赤点線で示した、共通的に用いられるコンポーネントについて設計を行う。



補足資料

開発工程と使用モデルの関係

- 基本設計では、業務要件として、ユースケースモデルを使用し、「システム化する機能(What)」を定義した
- 詳細設計では、コンポーネントを使用し、業務要件をどうシステムに適合させるかという、「実現方法(How)」を定義していく
- そのため、アーキテクチャー設計では、基本設計から詳細設計をスムーズに進めるためのルールを確定する
 - ①アーキテクチャーの全体方針の設計と、基盤機能の抽象化(基盤ソフトウェアの設計)
 - ②アプリケーション・アーキテクチャー設計



業務要件(ユースケース・モデル)は、システムを実現するための機能の固まり(コンポーネント・モデルへ)へと詳細化され、製品・実装技術上で要件を実現する。

アーキテクチャーとは

- アーキテクチャーとは、システムの動的／静的な構造を規定し、設計原則を表したものである
- アーキテクチャーは、システム全体を規定するアーキテクチャーと、段階的・部分的に適用するアーキテクチャーがある
- アーキテクチャーは、レイヤー、コンポーネント、インターフェース、サブシステム等の観点で考える
 - － システムを大きく分ける方法として、レイヤー(層)やサブシステム化
 - － コンポーネントは、取り替えることのできる、独立した機能単位
 - － インタフェースは、要素間の相互関係を定めたもの(レイヤー間、コンポーネント間、システム間など)
- アーキテクチャーを設計する目的は、
 - － システム全体の方針を定めることにより、システムを管理可能な大きさに分割することができ、また分割したものを統合する際の整合性を確保する
 - － 基盤・業務にわたるシステム全体の構造を検討することにより、システム全体として最適化を図る

アプリケーション・アーキテクチャーの考え方

- アプリケーション・アーキテクチャーは、
 - 業務の処理構造(動的／静的)の設計ルールを定めたものである
 - 業務要件を「コンポーネント」へ分解し、組み合わせるための指針をパターンとして定める
 - 共通化の方針・範囲に従い、共通のコンポーネントを定義する
 - 「基盤ソフトウェア」の提供する機能を利用し、業務開発のための標準ルールを定める
 - 基本設計で定めた「基盤ソフトウェア」の機能をもとに、アプリケーションの共通の処理構造に着目しつつ、パフォーマンスなどの非機能要件を加味した上で、「アプリケーション・アーキテクチャー」を設計する
- 基盤ソフトウェアは、
 - システム全体構造の中で、“アプリケーションとプラットフォームを分離するための層”として位置づけられ、その結果として、業務アプリケーションの移植性を上げることができる
 - 特定の製品やインフラ環境(O/S, H/W)への依存性を下げるための必要な機能を提供する
 - 実績のある製品(フレームワーク、ミドルウェア、ライブラリ)を活用し、製品に不足する機能を追加した上で、「基盤ソフトウェア」の機能として提供する
 - プラットフォームに適した処理方式を検討し、「アプリケーション・アーキテクチャー」の共通の処理構造を支える
 - 例) データベース・アクセスのコンポーネント設計指針はアプリケーション・アーキテクチャーとして検討するが、トランザクション処理の実現方式は、基盤ソフトウェアに必要な機能として検討する

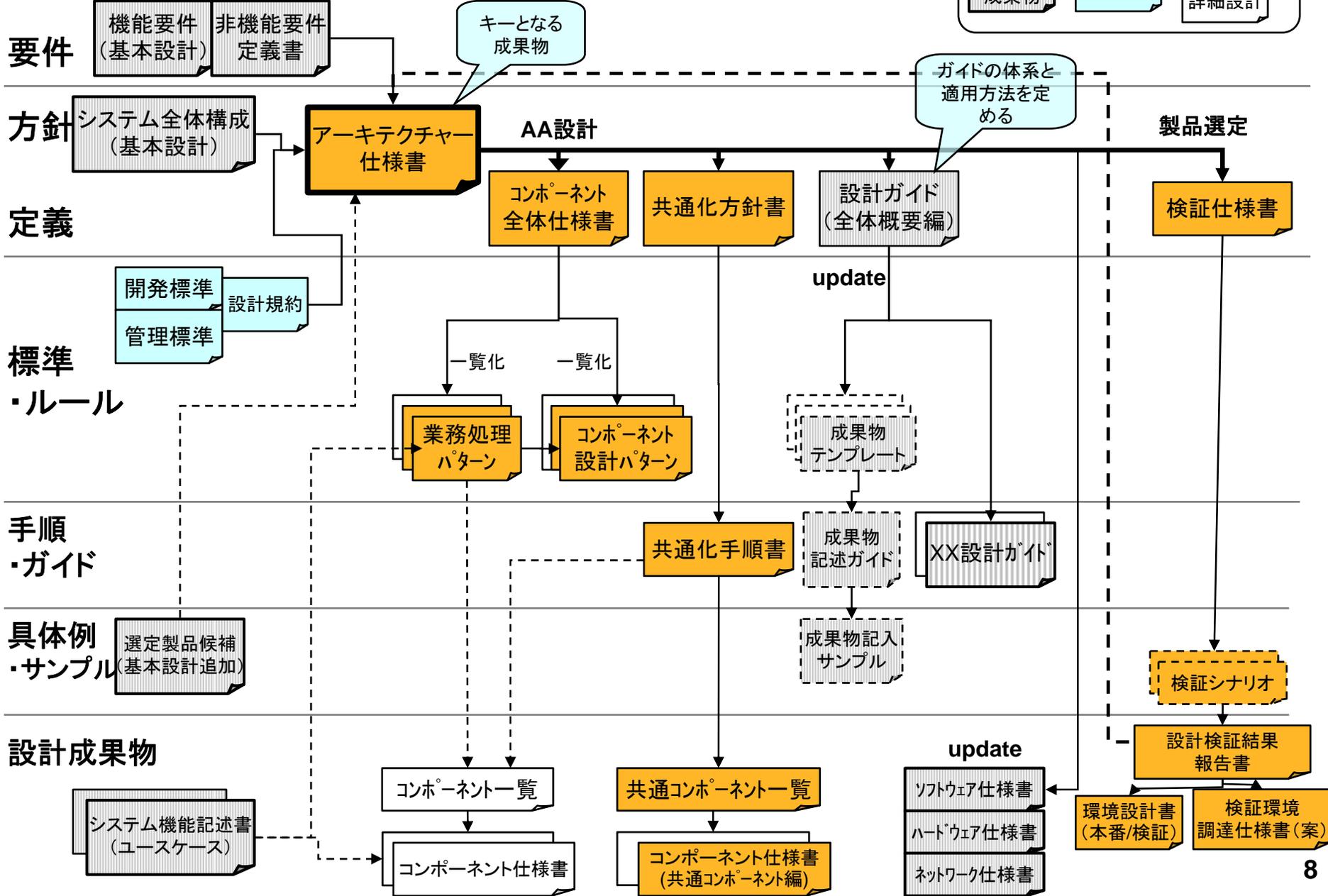
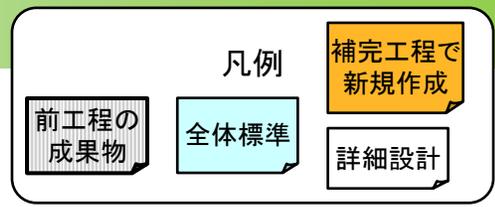
アーキテクチャー設計 成果物内容(案)

アーキテクチャー設計の成果物一覧

主要なタスク	成果物名	内 容
アーキテクチャー 全体方針設計	アーキテクチャー仕様書	システム全体のアーキテクチャーを定義した成果物
アプリケーション ・アーキテクチャー設計	業務処理パターン	業務処理パターンを定義し、パターンの名称、概要、コンポーネント構成、パターンの適用方法及び制約事項が記述されている成果物
	コンポーネント全体仕様書	コンポーネントの定義及びコンポーネントの種別の定義が記述された成果物
	コンポーネント設計パターン	業務処理パターン毎の各設計領域をパターンとして定義し、パターンの名称、概要、適用箇所（設計領域）、適用方法の説明、制約事項、対応する業務処理パターン及び関連する他コンポーネントを含めた振る舞い（静的ビュー、動的ビュー）をクラス図、シーケンス図及びパッケージ図等で表記されたもの等が記述された成果物
設計ガイド修正	各設計ガイド（修正）	詳細設計工程以降の作業を行うためのガイドを本調達の成果物に基づき更新した成果物
基盤設計の詳細化と 基本設計等の修正	ソフトウェア仕様書（修正）	基盤ソフトウェアを実現するソフトウェアの仕様を定義したもの。選定した製品に関する記述が追加されたもの。
	ハードウェア仕様書（修正）	基盤ソフトウェアを配置するハードウェアの仕様を定義したもの。選定した製品に関する記述が追加されたもの。
	ネットワーク仕様書（修正）	基盤ソフトウェアを構成するネットワークの仕様を定義したもの。選定した製品に関する記述が追加されたもの。
コンポーネントの 共通化方針設計	共通化方針書	コンポーネントレベルの共通化の範囲、観点を整理し、共通化の方針を定義した成果物
	共通化手順書	コンポーネントレベルでの共通化の手順を定義した成果物
	共通コンポーネント一覧	共通化方針書及び共通化手順書に基づいて、本システムで共通可能なコンポーネントを一覧化した成果物
	コンポーネント仕様書（共通 コンポーネント編）	共通コンポーネント一覧及びソフトウェア仕様の中から共通化がふさわしいものを選定し、複数のコンポーネントからの呼び出し方法を検討した上で仕様を確定し、仕様書として取り纏めた成果物
プラットフォーム性能検証 および アーキテクチャー妥当性検証	検証仕様書	プラットフォーム性能検証を行い、アーキテクチャー設計の妥当性を検証するための仕様を定義した成果物
	検証結果報告書	机上検証及び実機検証を通じてアーキテクチャー設計の妥当性を検証した結果を報告書として取り纏めた成果物及び検証用に作成した資材一式（検証用のテストプログラム、テストデータ等）
	環境設計書（本番環境用）	本番環境の環境仕様（HW/SW）が定義されたもの
	環境設計書（検証環境用）	検証環境の環境仕様（HW/SW）が定義されたもの

基本設計まで

アーキテクチャ設計における成果物の位置づけ(体系)



アーキテクチャー仕様書とは

- **アーキテクチャー設計の主要な成果物**
 - システムの包括的な概要(オーバービュー)を説明する
- **刷新システムのアーキテクチャを俯瞰する、システム上の特徴や設計方針を説明したもの**
 - 基本設計書のシステム全体構成(方針および論理設計)に基づき、製品選定結果を反映し、検証されたアーキテクチャを説明するもの
 - 基盤と業務アプリケーションの関連を示し、業務アプリケーション設計者にシステムの設計方針の要点を示す
- **詳細設計以降において、システム開発に携わるメンバーがシステムの全体的な構造を理解し、円滑な設計作業を遂行できるようにするために作成する**

アーキテクチャ仕様書の記述内容(案)

アーキテクチャ仕様書<目次例>

システムの設計方針を様々な視点(全体/部分、静的/動的、機能/非機能など)から記述する。

1. 日本年金機構業務システムのアーキテクチャ全体像

1-1. アーキテクチャ設計の目的

1-2. 特徴

1-2-1. アーキテクチャ設計対象範囲

1-2-2. アーキテクチャ設計における前提及び制約事項

1-2-3. アーキテクチャ設計におけるルール

1-2-3-1. コンポーネント設計について

※コンポーネント粒度、レベルについて記述。詳細は、コンポーネント全体仕様書&手順書に記述

1-2-3-2. コンポーネント共通化について

※詳細は、コンポーネント共通化方針書&手順書に記述

1-3. 非機能要件の設計方針

※非機能に対して、システム上どのような設計としたかを記述

※方針とあるが、設計上どう対応したかの結果を記述する

1-3-1. 性能に関する考慮点と設計方針

1-3-2. 可用性に関する考慮点と設計方針

1-3-3. 拡張性に関する考慮点と設計方針

1-3-4. 可搬性に関する考慮点と設計方針

1-3-5. 保守性に関する考慮点と設計方針

1-3-6. 管理容易性に関する考慮点と設計方針 ... etc

・基盤が非機能を実現するために、提供する仕組みと業務が設計時に、考慮すべきことを記述する

1-4. 全体構造 ※システムスコープの明確化

1-5. 部分構造

※1-4の全体構造から、各処理部の設計コンセプトを記述

1-5-1. トランザクション入力部

1-5-2. トランザクション処理部

1-5-3. データストア部

1-5-4. 外部インターフェース部

・切り口を定義する
・基盤がどんな仕組みを提供してくれるか

2. パターン別観点

※システムの振る舞いを汎化したパターン化で動きを説明する

2-1. アプリケーション・パターン

オンライン、ディレード、バッチなど

2-2. 業務処理パターン

申請ワークフロー+オンライン など

2-3. コンポーネント設計パターン

※コンポーネント設計テンプレートとの違いなど

3. アプリケーション・アーキテクチャー

※1-5よりも、より具体化した、コンポーネントレベルの設計方針を記述。設計ガイドでより詳細に定義

3-1 全体コンポーネント構成

3-2. 個別処理別

3-2-1. Web層

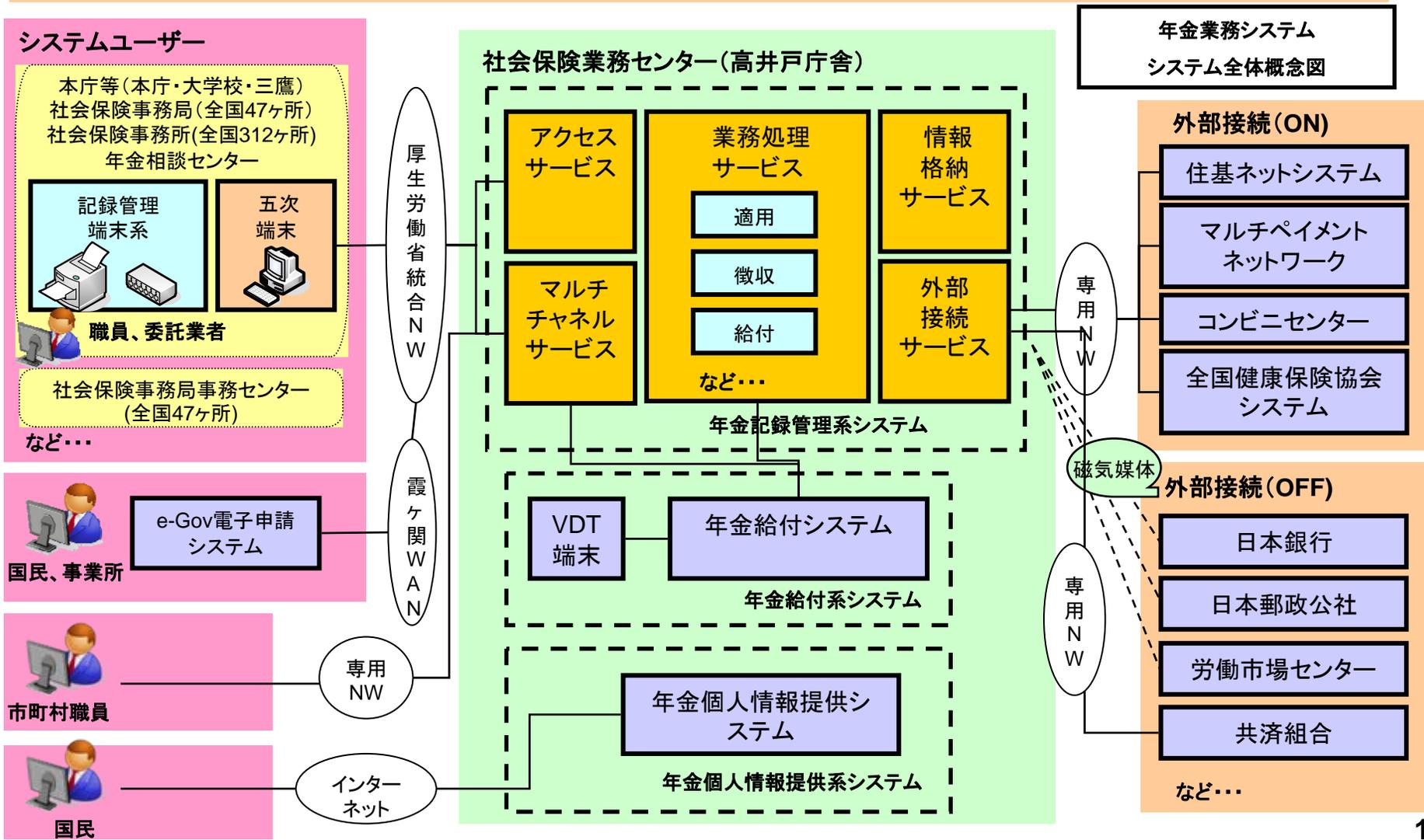
3-2-2. ビジネスロジック

3-2-3. データ・アクセス

・切り口を定義する
・設計ガイドの概要を想定

1-4. 全体構造 (Enterprise View)の例 ・静的ビュー

全体構造 (Enterprise View) ... 開発対象システムの周辺環境も含む全体構造。

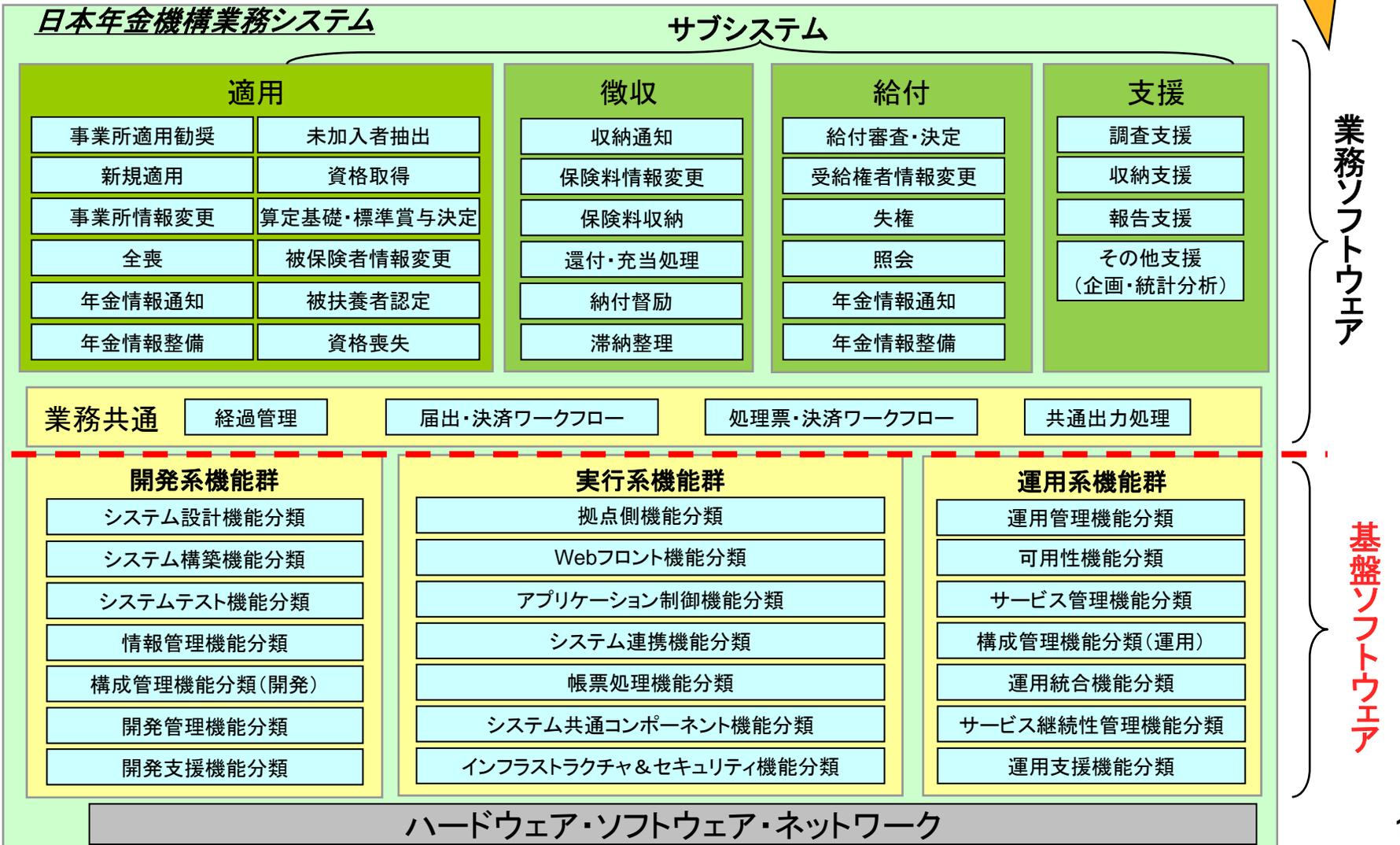


(参考)アーキテクチャー仕様書 記述内容イメージ

1-4. 全体構造(Service View)の例 ・・静的ビュー

サブシステムの機能一覧
基盤の機能一覧
基盤と業務の分離点

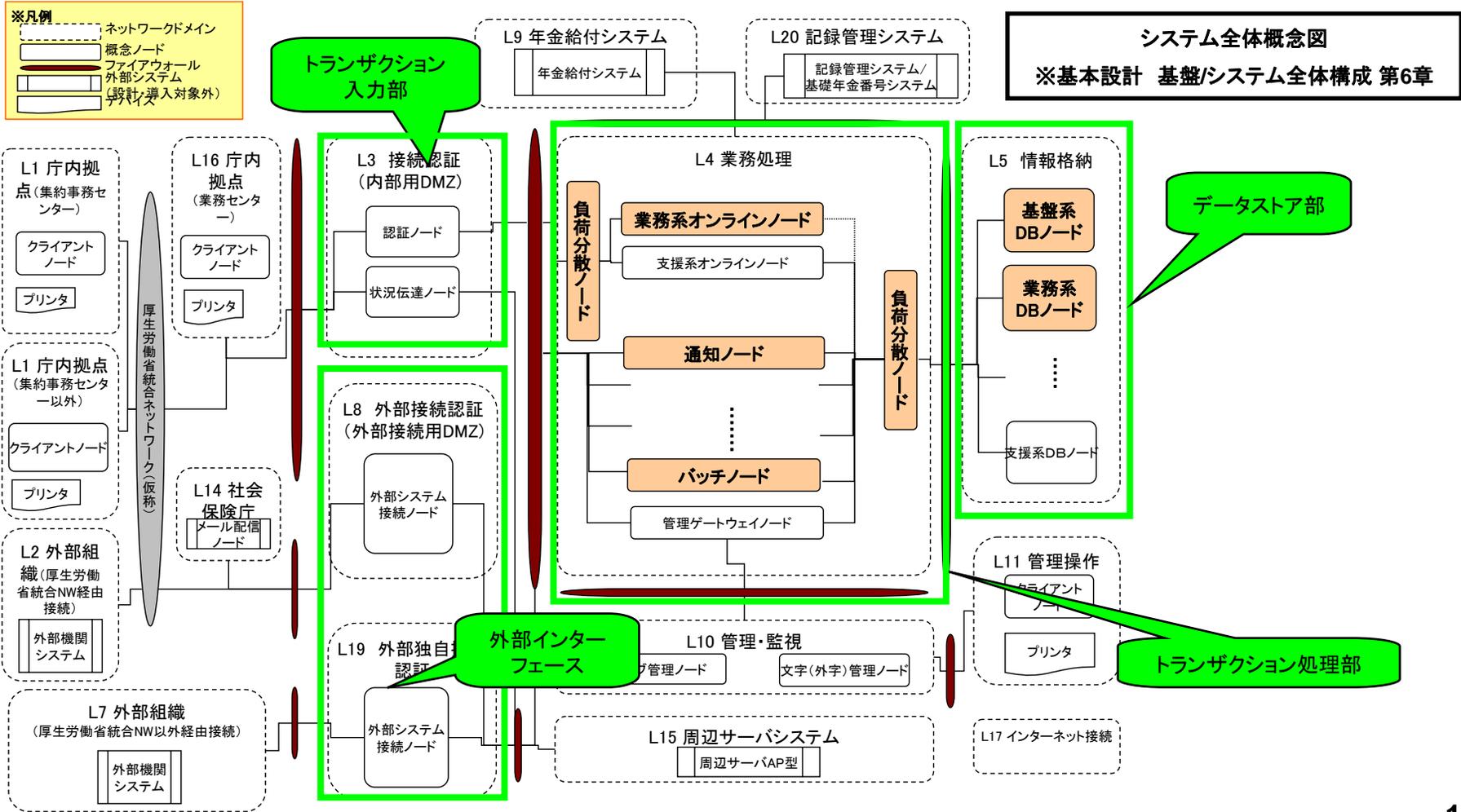
全体構造(Service View)・・・サービスの観点から開発対象システム全体構造を記述したもの。



1-4. 全体構造 (IT System View) の例 ・静的ビュー

ネットワーク構成、ノード(サーバー)など、システム基盤に注目した図

全体構造 (IT System View) ... ITシステムの観点から開発対象システム全体構造。

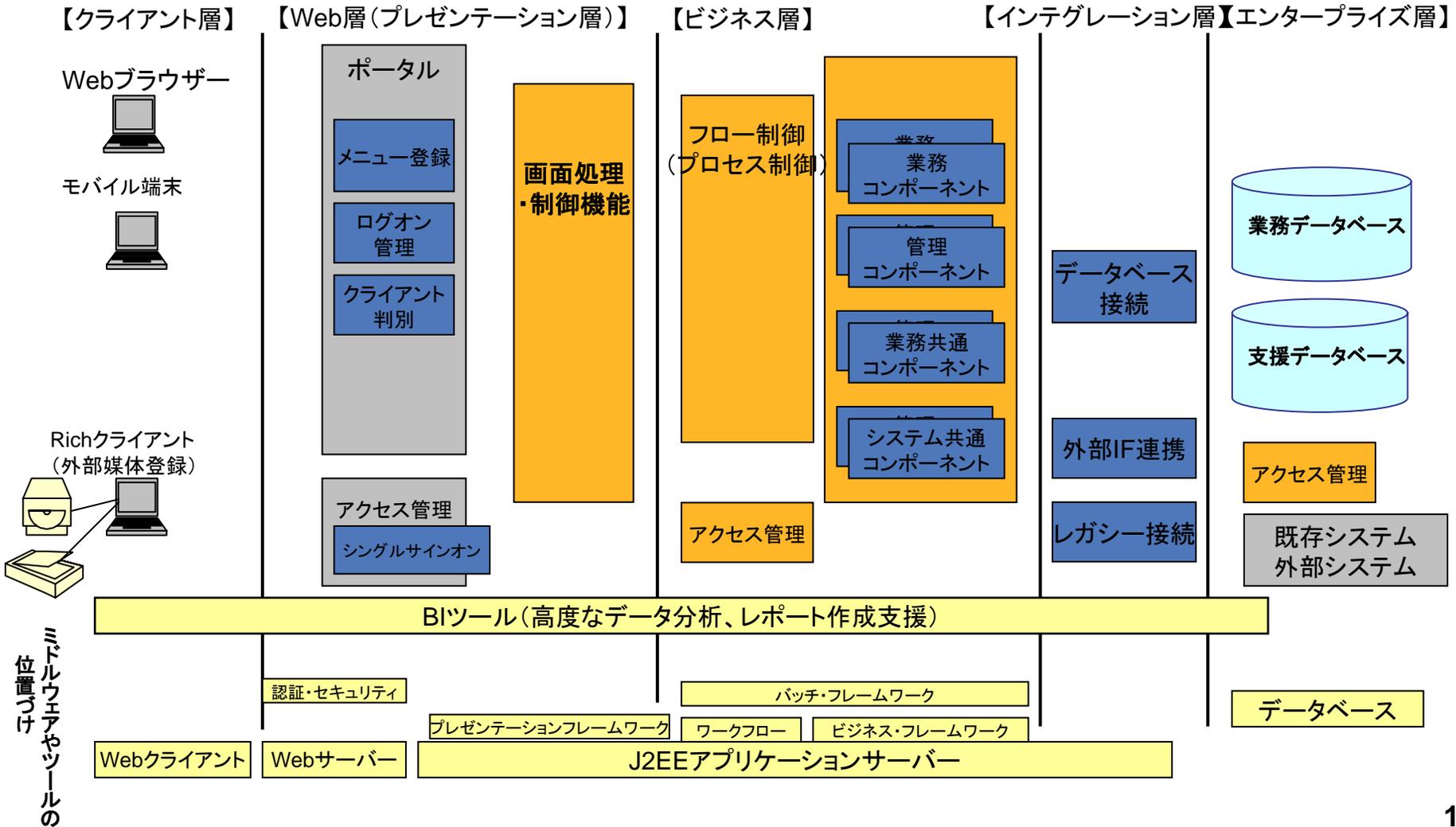


論理的に、J2EEのミドルウェアや境界、コンポーネント構成の全体方針がわかるもの。
コンポーネント設計パターンよりもハイレベル。
業務処理別にソフトウェア構成がわかるもの。

3-1. 全体コンポーネント構成 の例

(2)アプリケーション・アーキテクチャー全体構造 (Software View, Layer Vi

※オンラインの場合※

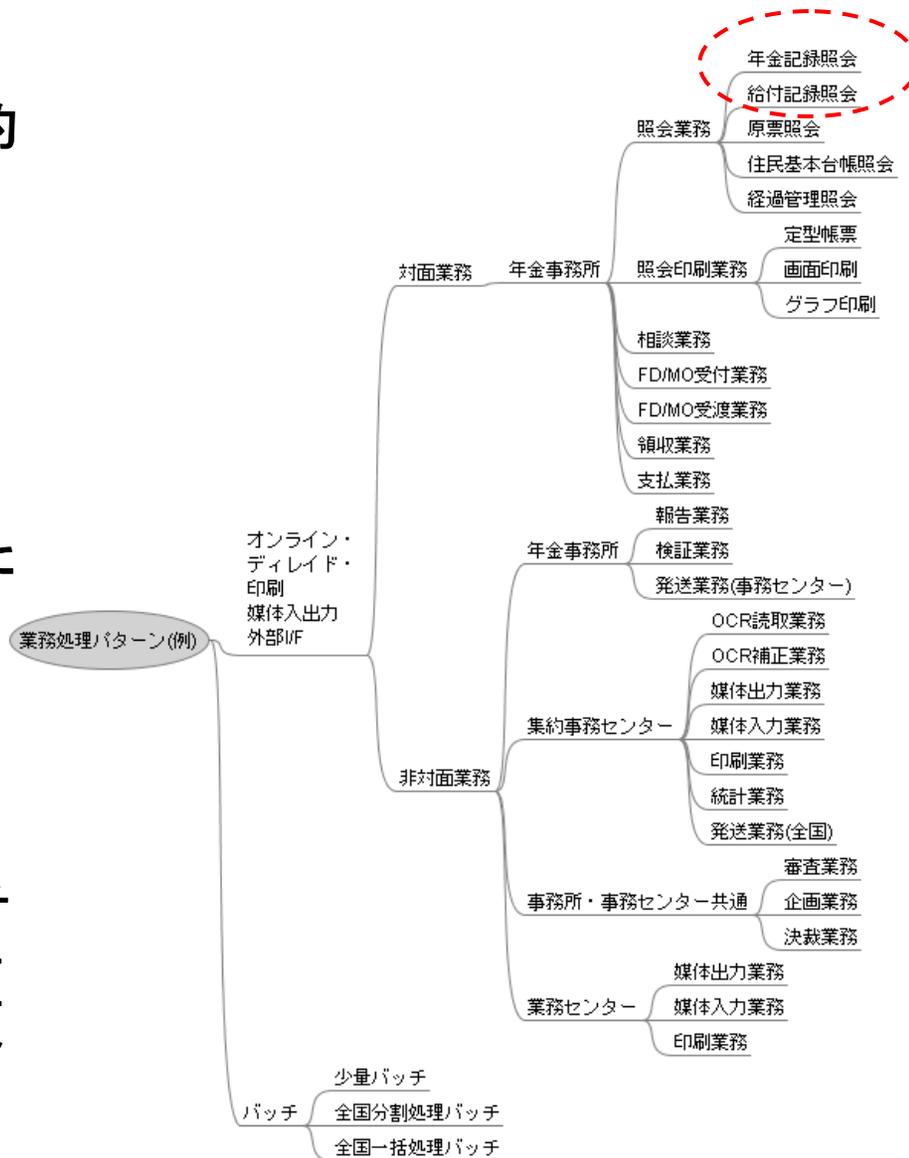


業務処理パターンとは

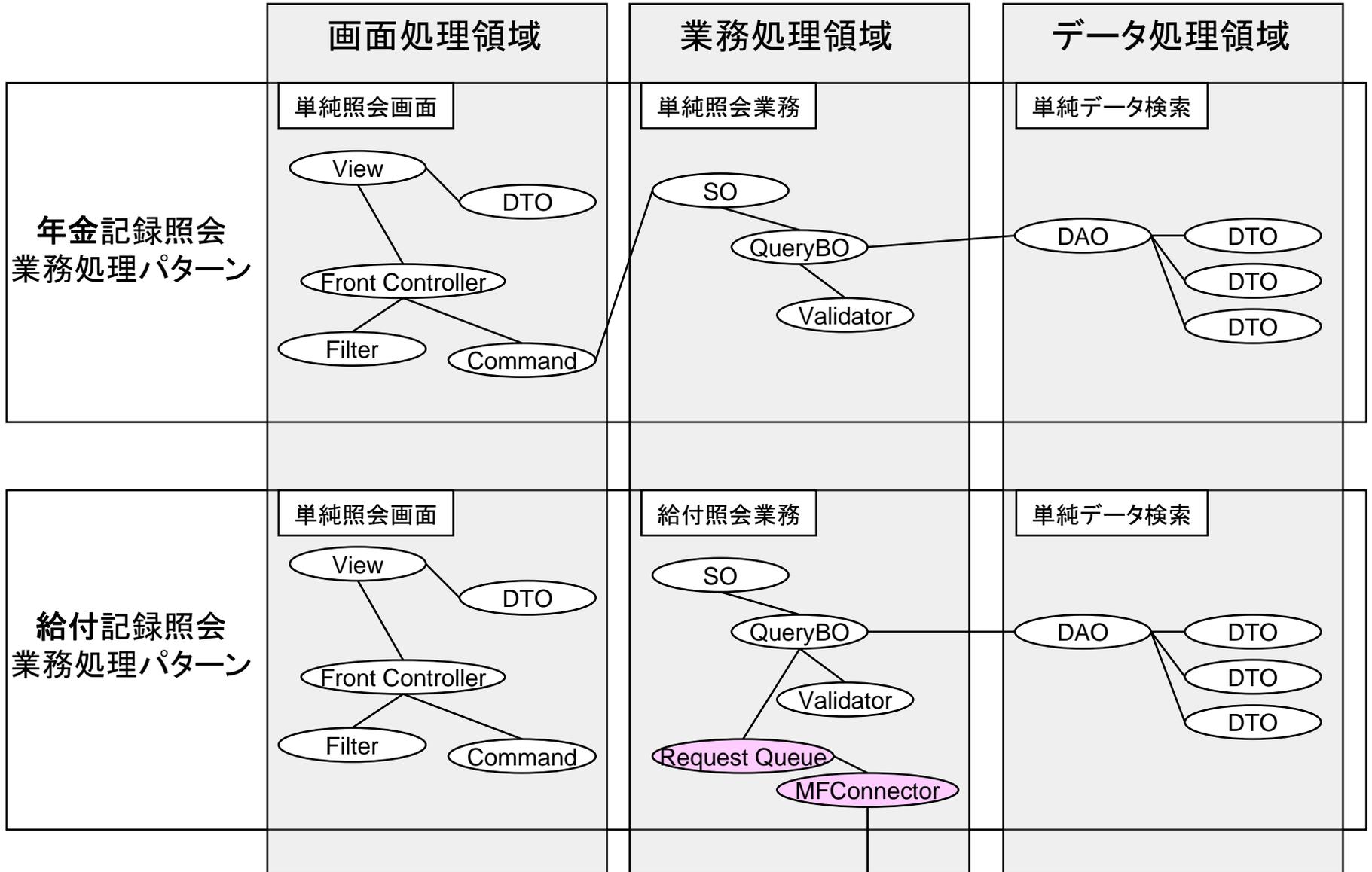
- ユースケースを実現するための、標準的なコンポーネント設計パターンの適用方法
 - ひとつのユースケースを実現するためには、複数のコンポーネント設計パターンを適用する必要がある。
 - 詳細設計において、業務アプリケーション設計者がユースケースからどうやって適切なコンポーネント設計パターン選んで適用すればよいのか、という方法が必要
 - 業務処理パターンは業務処理形態から標準的なコンポーネント設計パターンを適用するための道筋を示す。

業務処理パターン（作成例）

- まず、システム化対象の業務を体系的に分類する。
- 各末端の業務処理形態ごとに適切なコンポーネント設計パターンを選択する。（ここまでを補完工程で実施する）
- 詳細設計において、業務アプリケーション受託者は業務処理パターンをヒントにして、ユースケースを実現するための適切なコンポーネント設計パターンを選択する。
- 注) 業務処理パターンは業務処理形態だけでなく、**アーキテクチャ設計の違い**に注目して作成する。例えば、同じ照会業務でも、年金記録照会と給付記録照会は異なるコンポーネント設計パターンが適用されるので、別の分類とする。

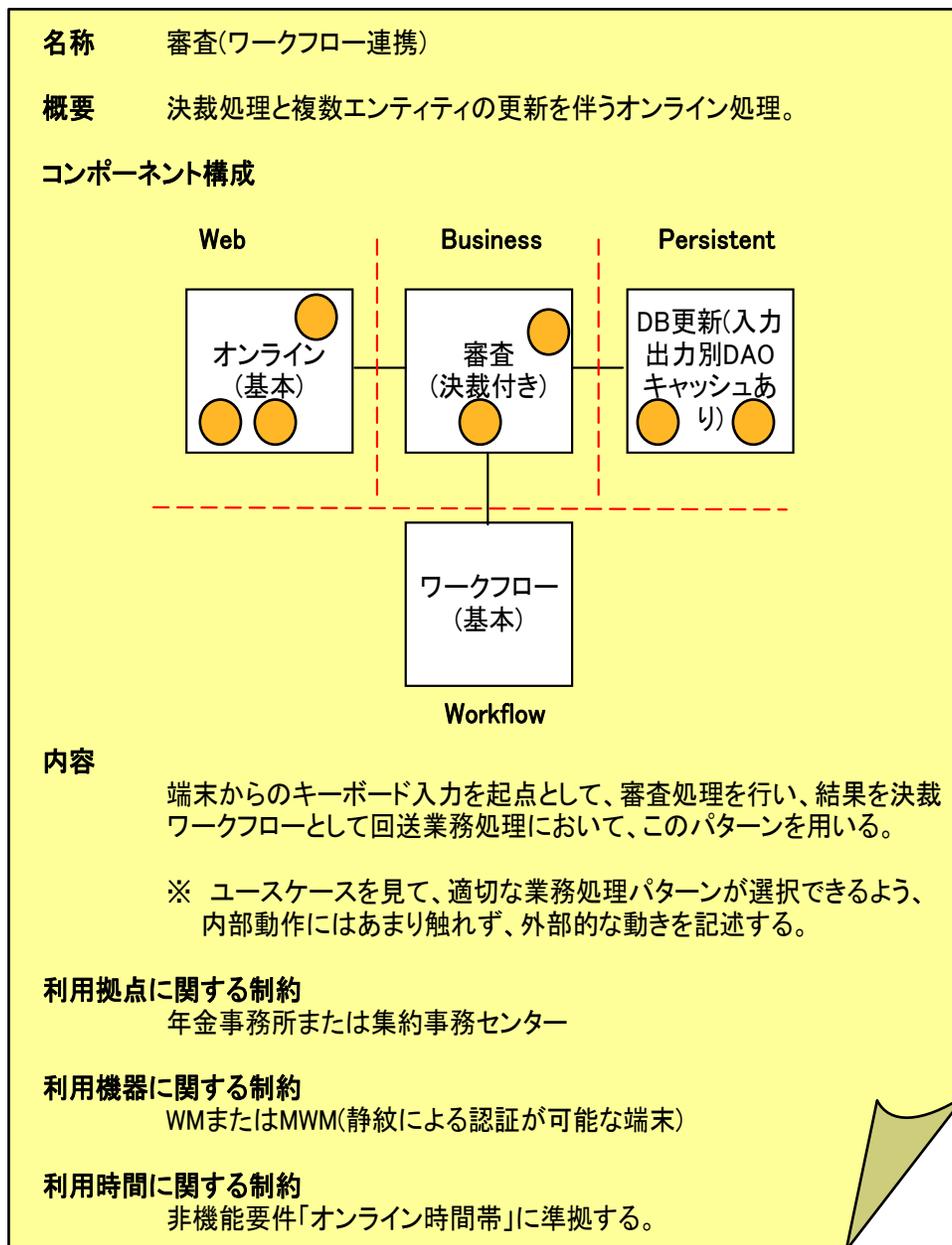


業務処理パターン (作成例2)



「業務処理パターン」の説明として、記述すべき内容(内容例)

- 業務処理パターンの数分、作成する
- 業務処理パターンが決まると、コンポーネント設計パターンの組み合わせが決まり、ひいてはコンポーネントの構成が決まることとなる

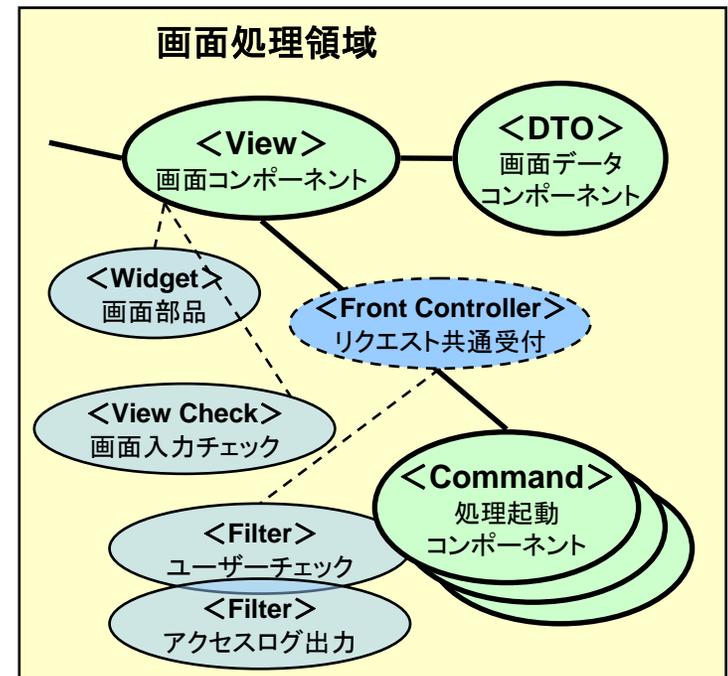


コンポーネント設計パターンとは

- 機能を実現するための標準的なコンポーネント構成を定めたもの。
 - 一つの画面機能を実現するためには、例えば以下のようなコンポーネント構成をとる。
 - コンポーネント構成をパターン化することにより、コンポーネント分割における属人性を排除する。

注:コンポーネント設計パターンは「画面処理領域」「業務処理領域」「データアクセス領域」などの『**設計領域**』毎に作成する。

すなわち、一つの業務処理を完結するためには、複数のコンポーネント設計パターンを組み合わせ適用する。



「コンポーネント全体仕様書」に記述すべき内容(案)

- アーキテクチャ仕様書を元に、コンポーネントの定義や、他のパターンとの関連などを記述するための資料

- 想定内容
 - － コンポーネントとは
 - コンポーネントの種別(ステレオタイプ)の定義
 - コンポーネントの粒度定義
 - － ユースケースとパターンとの関係
 - ユースケースと業務処理パターン
 - 業務処理パターンとコンポーネント設計パターン

ユースケース・コンポーネント対応表(案)の作成例

- ユースケース → 業務処理パターン → コンポーネント設計パターン → コンポーネントへのトレース

基本設計で設計済み
各ユースケースに業務処理パターンを
割り当てる作業は、詳細設計工程で実施する。

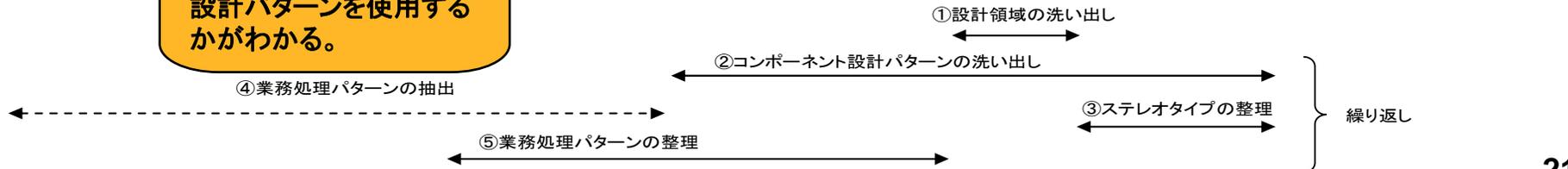
基本設計補完工程で設計

詳細設計工程で設計

ユースケースID	ユースケースステップ	業務処理パターン	コンポーネント設計パターン	設計領域	ステレオタイプ	連番	コンポーネントID	共通
T-2-2-2-5-1 新規適用届を審査する	(1) ~ (5) 1a, 2a, 3aを含む	審査(ワークフロー連携)	オンライン(基本)	Presentation	«DTO»	1	P25344101	
					«DTO»	2	P25344102	
					«Command»		P25341101	
			審査(決裁付き)	Business	«Action»		P25342101	
					«SO»		B25342101	
					«Validator»		B25343101	Y
					«BusinessRule»		B25347101	
	ワークフロー(基本)	Workflow	«AccessControl»		B25348101			
			«WorkflowAdapter»		B25346101			
	DB更新(入力出力別DAO キャッシュあり)			Persistent	«WorkItem»		B25346201	Y
					«DAO»	1	D99341101	Y
					«DAO»	2	D25341102	
					«DTO»	1	D99421323	Y
					«DTO»	2	D99421147	Y
«DTO»					3	D25341101	Y	
«DTO»					4	D25341102		
(6) ~ (9) 7aを含む	照会	オンライン(基本)	Presentation	«Cache»		D99180101	Y	
				«DTO»		P25344201		
				«Command»		P25341201		
		単純検索	Busi	«Action»		P25342201		
				«SO»		B25342201		
		DB検索(コード変換あり)	Persistent	«SO»		B25342201		
«DTO»	2			D99421147	Y			
		«DTO»	3	D25341201				
		«DTO»	4	D25341202				

業務処理パターンが決まると、どのコンポーネント設計パターンを使用するかがわかる。

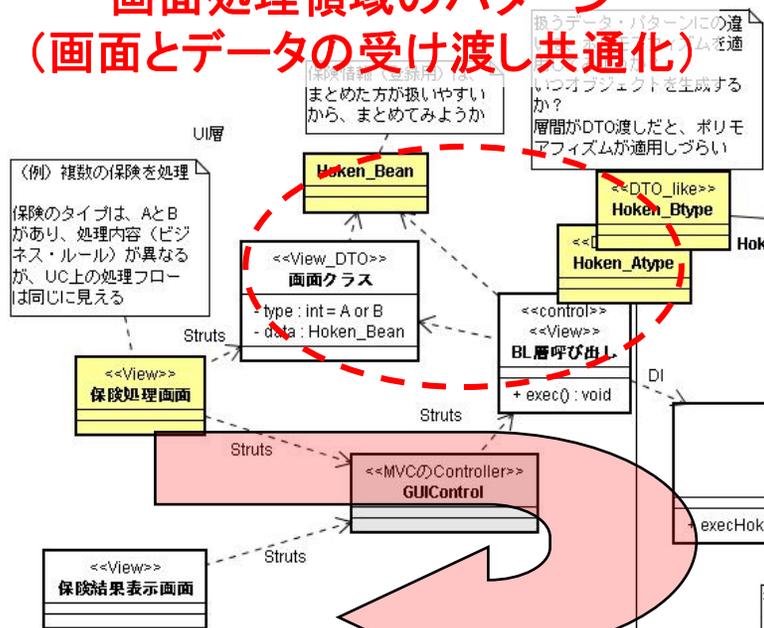
多くの関連付けを効率的に管理するために、ツールの利用を検討する



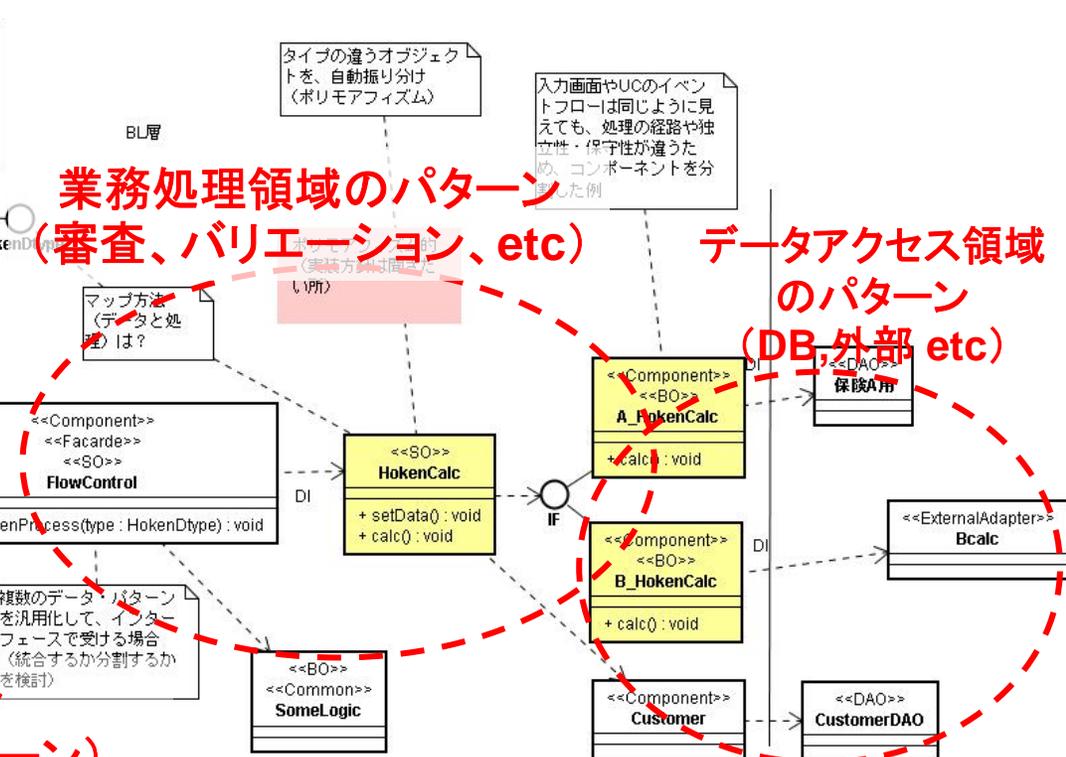
業務処理パターン、コンポーネント設計パターンの適用イメージ

- あるユースケースに対し、業務処理パターンを識別すると、適切なコンポーネント設計パターンを導くことが可能になる
- あるユースケースに対し、開発者は、設計領域ごとに、複数のパターンを適用することになる

画面処理領域のパターン (画面とデータの受け渡し共通化)



業務処理領域のパターン (審査、バリエーション、etc)



データアクセス領域のパターン (DB、外部 etc)

オンラインの基本的なパターン (サービス呼び出し & 画面遷移パターン)



「コンポーネント仕様書」の成果物イメージ(例)

- 基本的に、コンポーネント毎に作成
- 詳細設計工程で、業務コンポーネントの仕様書を作成する
- 補完設計工程で、共通コンポーネントの仕様書を作成する

- **コンポーネント責務の定義**
 - 何をするためのコンポーネントか(説明)
 - ステレオタイプ(汎用的な型)
- **コンポーネント・インターフェースの定義**
 - メソッド定義
 - 呼び出し名
 - 呼び出しの引数(型)
- **コンポーネント構造(クラス図にて視覚化)**
 - インタフェースとコンポーネントの関連
 - 引数の関連
- **コンポーネントの使用条件**
 - 事前条件
 - コンポーネントを呼び出す前の必須条件
 - 入力チェックの要否に関わる
 - 事後条件
 - 呼出し後に、システムの状態が変化しているか・していないか
 - ※テストケースの作成に必要
- **制約内容や実装の方針等**
 - 非機能要件
 - ライフサイクル(生成・破棄方法等)
 - コンポーネント内部設計の設計方針 (Option)

システム名	サブシステム名	コンポーネント仕様 (概要・属性)	作成者
日本年金機構業務システム	適用業務		
コンポーネントID	S00001		
コンポーネント名称	新規適用届の審査	XxxxService	
コンポーネント機能概要	仮原簿に登録されている新規適用届を審査するための一連の処理について、指示を...		
コンポーネント種別	ServiceObject		
パッケージ名	XXXX		
親コンポーネント			
実装インターフェース			
バージョン (永続)			
トランザクション属性 (EJB)			
対応する分析オブジェクト	Control: 「新規適用届を審査する」		
ライフサイクル	処理が呼び出されてから、審査フローが完了し、処理結果を返すまで。		
コンポーネント内部設計の設計方針 (任意入力)	端末からの入力に由来するもの、電子申請に由来するもの、媒体に由来するもの、の		

インターフェース

	インターフェース名称	戻り値	引数	責務	事前条件	事後条件
+	新規適用届審査処理の指示	処理結果	ファイルの配置場所を特定する情報	仮原簿からファイルを読み込み審査フローにかける	仮原簿に新規適用届情報ファイルが登録されていること	新規適用届の決裁が完了していること

プロパティ

名称	プロパティ名称	データ型	概要	Getter	Setter

コンポーネントを外から見た仕様

非機能要件

種類	概要
性能	
制約	