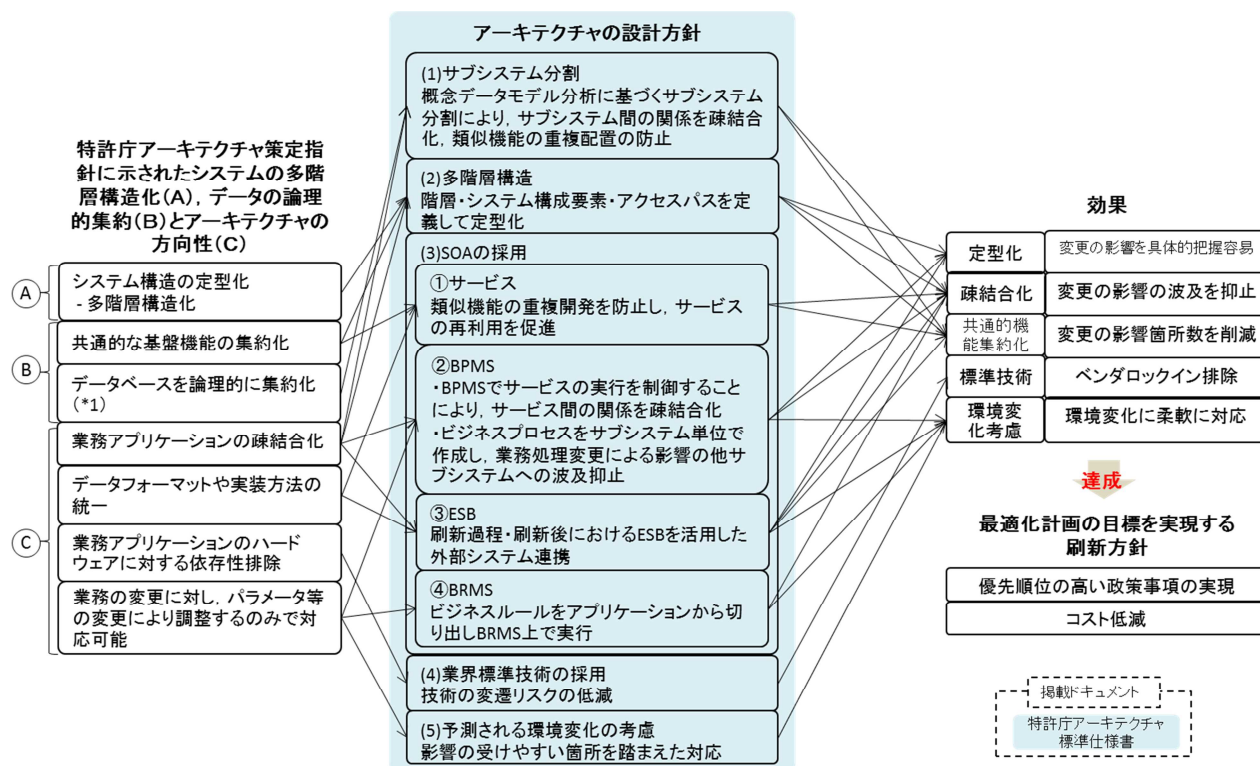


1. 設計方針

1.1 アーキテクチャの設計方針の導出

特許庁アーキテクチャ策定指針に示されたシステムの多層構造化・データの論理的な集約化による簡素化・アーキテクチャの方向性とアーキテクチャの設計方針の対応関係、アーキテクチャの設計方針から得られる効果及び最適化計画の目標を実現する刷新方針の達成について、下図に整理する。



※1 データの論理的集約化は「データ統合方針書」にて具現化する。

図 1.1-1 アーキテクチャの設計方針の導出

特許庁アーキテクチャ策定指針で示されるシステム構造の見直しの考え方、アーキテクチャの方向性を受け、システム構造の簡素化を以下に示す(1)～(3)により実現する。

- (1) サブシステム分割
- (2) 多階層構造
- (3) SOAの採用

また、その簡素化したシステム構造を長期的に利用可能なアーキテクチャとなるようにするために、以下に示す(4)及び(5)により実現する。

- (4) 業界標準的技術の採用
- (5) 予測される環境変化の考慮

このようなアーキテクチャの設計方針を採用することにより、システム構造の複雑化やデータの個別システムへの分散といった現状の問題点を解決する。

次項にてアーキテクチャの設計方針の詳細について述べる。

1.2 アーキテクチャの設計方針の詳細

以下にアーキテクチャの設計方針の詳細を述べる。

(1) サブシステム分割

特許庁システム全体を、設計・開発や管理の面における適切な単位となるよう論理的に分割し、サブシステムとして定義する。各サブシステムは、可能な限り互いに疎であることが望ましい。疎であるとは、互いの関連性が低く、連携する頻度や受け渡す情報の量が少ない状態を指す。疎の関係となるよう分割することにより、変更によるシステムの改修範囲を個々のサブシステム内にできる限り局所化することを可能とする。本設計方針においては、既存システムの枠にとらわれずに概念データモデルの観点と法律の観点から、業務を同じ目的の業務処理の集合に分割する。

また、最適化計画の方針に従い、各業務システムに分散していた重複データを論理的に共有データベースに集約して管理するにあたり、それ以外の、各業務システムにおいて個別に保有し管理すべきデータを個別データと定め（詳細については『データ統合方針書』を参照のこと。）、所定の業務システム内に隠蔽し他の業務システムからのアクセスを禁止することでデータの更新整合性³を確保しやすくする。

当該個別データを隠蔽する業務システムとして、上記業務処理単位に分割したサブシステムを採用する。

更に、将来起こりえる環境変化に対してもシステムの改修範囲が大きくなることを防ぐため、特許庁システムにおいて最も大きな環境変化の要因である法律の観点を考慮した分割とする。特許法、実用新案法等、特許庁業務の根拠法令の種類と、特許庁業務のシステムを利用した実現方法を示した特例法の双方を踏まえることで、特許庁業務を疎結合かつ適切な粒度で分割したサブシステムを採用する。

今後、新規に業務が追加された場合においても、概念データモデルのメンテナンス及び法律の観点から業務の位置づけを整理し同様のサブシステム分割を行い、どのサブシステムに業務を追加するか、あるいは新規にサブシステムを追加するか等といった検討を行うものとする。

(2) 多階層構造

分割した各サブシステムの構造を定型化するために、最適化計画の方針に従い、多階層構造を定義し、多階層の各層に配置するシステム構成要素とシステム構成要素のアクセスパスを定める。定義した多階層構造に対し、以下に示す設計方針を適用することにより、更なる定型化と疎結合化を行い、サブシステムの保守性・移植性を向上させる。

- 各層の責任分担を適切に定め層毎に機能を集約することにより、改造が必要になった場合における影響箇所の特定の容易化。
- 層と層のアクセス方法を標準化するとともに層間の依存関係を統制することにより、改造が必要になった場合における影響が他の層に拡散することを抑制。

(3) SOAの採用

SOAは2000年初頭から広まった技術であり、SOAを活用してレガシーシステムを流用しつつシステム刷新を行った実績も数多い。特許庁における最適化計画の刷新方針において、SOAの採用は以下に示す利点を有する。

- 最適化計画において掲げられたシステム構造の見直しの「業務APを疎の関係とすること」との親和性が高い点。
- 既存システムを維持しつつ段階的に個別システムを刷新することとの親和性が高い点。
- 最適化計画が10年にわたる極めて長期的な計画であることとの親和性が高い点（技術動向が比較的安定している業界標準的なアーキテクチャである点）。

³ 例えば、決裁前の処分情報を後続工程の業務システムが参照してしまうことにより、誤って決裁されたものとして後続の業務システムが処理を実行するパターン（チェックアウト中の情報参照によるデータ不整合）や、外注中の案件に対して誤って処理を実行するパターン（ロングタームトランザクション中のデータ不整合）等。共有データベースに業務として確定した最新情報を集中するとともに、共有データベースの情報又は自システム内のCRUDを制御できる個別データベースの情報のみを用いて各業務システムが処理の実行及び個別データベースの更新を実施することにより、データの整合性を担保可能とする。

したがって、サブシステム分割や多階層構造を適用したシステム構造を支える技術としてSOAを採用する。

SOAの考え方に基づき、特許庁システムのサブシステムや業務アプリケーションをサービスとして構成し、各サブシステムから利用する。特許庁システムのサービス間の連携を担う技術として、BPMSやESBを適用する。また、変更の頻度の高いビジネスルールをアプリケーションから切り出して実行するためにBRMSを使用する。具体的な方針を以下に示す。

A. サービス

SOAの考え方を適用し、特許庁システムの機能をサービスとして構成する。サービスは共通的に利用することを想定し、サービスインタフェースの仕様を管理する。これにより、サービスの提供する機能の再利用が促進され、類似機能が重複して開発されることを抑止する。

また、サービスへアクセスする方法として業界標準的な通信プロトコルを使用する。これによりサービスへの依存関係を疎結合にし、インタフェースが変わらない限り、サービスの内部構造が刷新等により変更された場合においても、その影響が利用者側に波及しないようにする。

B. BPMS

BPMSは、サブシステムや業務アプリケーションを構成するサービスの実行順序や実行条件の制御を一元管理する役割を持つ。これにより、サービス同士の直接的な連携を排除し、サービス間の関係を疎結合に保つことを可能とする。

BPMS上で実行されるビジネスプロセスのアクティビティは業務として意味のある最小の単位で記述する。また、ビジネスプロセスはBPMNで記述し、そのアクティビティはサービス実行の単位と対応付けられるものである。これにより、BPMNで業務処理の流れが可視化されることに加え、業務的に理解できる要素とシステム上対応するアプリケーションの処理が対応付けられるため、業務の変更に伴うシステムへの影響範囲の把握を容易にする。また、業務可視化資料(分析モデル)とシステム可視化資料(実行可能モデル)とを同じ記法(BPMN)で作成することにより、ワークフローの管理負担の軽減や、可読性の向上が期待できる。

また、ビジネスプロセスはサブシステムの単位で作成する。サブシステムの範囲は業務的な責務の境界であるため、業務処理を記述するビジネスプロセスやビジネスプロセスがアクセスするサービスは、同じサブシステム内に閉じて実行されるものを多く含むことが想定される。このようにビジネスプロセスの単位を定めることで、業務処理の変更の際に、他のサブシステムに対して大きな影響が波及することを防ぐ。

なお、業務の実行制御方式として、単純にアプリケーションを順次実行する処理方式と比較すると、BPMSはビジネスプロセスの進行状況を管理する仕組み上、性能面においてビジネスプロセスやアクティビティの粒度の設計に注意を要する。しかし、特許庁アーキテクチャ策定指針において高めるべき品質特性として特に保守性が掲げられているところ、上述のとおりBPMSの適用によって保守性の向上が期待され、また、他のビジネスプロセス管理技術と比較した場合、ビジネスプロセスの可視性が高く、HTTP(S)のようなWebサービスの標準規格にも対応している等の優位性があることからBPMSを採用することとする。

C. ESB

外部システムへのアクセスについては、ESBが標準化された内部システムのプロトコルでアクセス可能なサービスインタフェースを提供する。またESBには、外部システムが利用する様々な通信方式とのギャップを変換する役割を持たせる。これにより、外部システムとの連携であっても標準化された内部インタフェースのようにアクセスでき、内部システムのシステム構造の定型化を促進するほか、外部インタフェースアクセスの変換機能を集中化し、類似の機能が各サブシステムで重複して開発されることを防ぐ効果や、外部システムと疎結合を保ち、外部システムの刷新や変更による大きな影響が内部システム側に波及されることを防止する効果を期待するものである。具体的には、外部システムが刷新された際に、内部システムのアプリケーションを極力再度修正しなくてもよいようにすることを可能とする。

外部システムが提供するインタフェースと内部システムで使用するインタフェースとの変換においては、ESB製品によって実現が困難となるプロトコル変換や処理単位(単件、複数件)の変換等が想定されているため、補完的に開発する外部システム互換機能を配置し、それらが協調動作してこの役割を担うものとする。外部システム互換機能は、アプリケーション、プログラムプロダクト又はそれらの組み合わせにより実現する。

外部システムへのアクセス方式として、ESB等のプログラムプロダクトを使用せず、アプリケーションのみで実現するという方式も考えられるが、アプリケーションの作り込みを極力避け、ESB製品の設定情報の変更等の対応により保守性・移植性・開発効率性が高まる効果や、特許庁システムのToBeアーキテクチャがESBで一元化

されることによる定型化の促進や効率的なリソースの活用，期待される統制面の向上といった効果を重視する。また，他の外部システムとの連携技術と比較した場合，異種のシステム間の接続に広く一般的に使用されており，サービス間の複雑な連携や性能面の拡張性の高さ，Webサービスの標準規格にも対応している等の優位性があることからESBを採用することとする。

なお，内部システム間の連携は全て標準化された通信プロトコルにより行われるため，ESBの機能によって得られる恩恵が少ない。したがって，内部システム間の連携においては，ESBを積極的には使用しないこととするが，個別の要件によって望ましい場合においては使用する。

D. BRMS

変更の頻度の高い業務をビジネスルールとしてアプリケーションから切り出して実行するために使用する。これにより，変更頻度が高い業務をビジネスルールとしてアプリケーションから切り出し，疎結合とすることで，ビジネスルールの変更や追加の影響が局所化される。BRMSはビジネスルールをデシジョンテーブル等の形式で宣言的な表現で記述できるため，ビジネスユーザでも比較的理解しやすく，変更による影響箇所が特定しやすいビジネスルール管理技術である。

(4) 業界標準的技術の採用

特許庁システムの業務を踏まえつつも，業界で広く一般的に使用されている技術を採用するため技術参照モデル (TRM) に準拠してプログラムプロダクト毎に機能要件や仕様を標準化する。このことにより，技術の変遷リスクを低減するとともに特定ベンダの製品の機能に依存しないようにする。

なお，複数のプログラムプロダクトを統合した，いわゆるスイート製品でなければ実現できないようなアーキテクチャとはしないが，本書に示すルールに従う限り，そのような製品群を排除するものではない。例えば，多階層の各層に配置するシステム構成要素群を一つのスイート製品で実現してもよい。

(5) 予測される環境変化の考慮

過去の制度改正を踏まえて，制度改正等によって影響の受けやすい箇所を特定し，アーキテクチャ上の対策をとることにより，環境の変化によるシステムの影響範囲を極小化する。

2. ルールの考え方

本書の設計方針に従い刷新対象システムを構築するにあたり、刷新対象システムの業務特性に応じた要求仕様を考慮した設計が必要である。しかしながら、産業財産権を付与し当該権利を維持するという特許庁の事務処理業務には、出願から権利消滅に至るまでに業務的な手続き(受付、審査、登録等)の流れが存在するという点や、業務的な手続きにより出願に係る業務データの状態が変化し、業務データの状態によって実施される手続きが管理されるという点で共通の特徴があるため、本書の設計方針に基づきシステム構造を定型化することが効果的な部分も存在している。すなわち、システム構造の定型化のために遵守させる強制力の強いルールもあれば、設計の裁量を残す強制力の弱いルールも存在する。また、ルールを適用するシステムもあれば、ルールを適用しないシステムもある。

また、設計に関与するステークホルダ(設計・開発ベンダ等)の設計ノウハウを最大限に活かすことが肝心であるものの、システム構造を定型化するための何かしらの制限も必要である。

以上の点を考慮し、以下、本書の設計方針に基づいてシステム構築を行うために必要となるルールの考え方について、スコープ(ルールの適用範囲)、プロセス(ルールの適用工程)及び強制力(ルールの裁量)の3つの観点について整理する。

2.1 スコープ(ルールの適用範囲)

本書ではアプリケーションのアーキテクチャを対象としたルールを策定する。ハードウェアやネットワーク等の設備等については基本的に対象外であるが、アプリケーションのアーキテクチャを定めるために関係するものについてはその限りではない。

また、本書で定めるルールは、産業財産権の付与及び維持に関する特許庁の事務処理システムに適用する。適用されるシステムの具体例について、平成25年度時点の既存システム等に対してマッピングしたものをToBe対象システムとして『全体システム概念設計書』の1章にまとめているので参照のこと。

上記ルールは、全てのToBe対象システムに適用するものであるが、システムの特性或条件によって適用範囲が異なる。以下に詳細を述べる。

(1) 階層定型化サブシステム

特許庁の事務処理システムのうち、産業財産権法に基づく事務処理を行うシステムについては、上記のとおり事務処理内容に共通事項が多い。このようなシステムについては、階層定型化サブシステムとして、システム間のインタフェースやシステム内の構造に対し、本書にて定めたルールを適用してシステム構造を定型化する。その他の定型化による効果が見込める事務処理システムについても階層定型化サブシステムに含める。

(2) インタフェース定型化サブシステム

特許庁の他の事務処理業務と共通事項のない特殊な責務を担う事務処理システムについては、個別の要求仕様を考慮し、個別にアーキテクチャを定めるシステムとして、システム構造の定型化を部分的に留める。

具体的には、階層定型化サブシステムとのインタフェースやプログラムプロダクトの適用に関するルールを定めるに留め、責務に特化した製品や技術方式を適用するアーキテクチャの採用を許容する。

本システムの具体例を「表 2.1-1 個別にアーキテクチャを定めるシステムの例」に示す。「表 2.1-1 個別にアーキテクチャを定めるシステムの例」に示したものの以外のシステムについては、設計に関与するステークホルダ(設計・開発ベンダ等)がシステム固有の要件に基づき、特許庁と協議の上で決定する。

表 2.1-1 個別にアーキテクチャを定めるシステムの例

項番	サブシステム	システムID	理由	刷新前システム
1	外部システム連携	— (新規)	個々の外部システムとの連携でそれぞれ使用されるプロトコルに対応した製品や、その他のギャップを吸収するための仕組みを有するため。	—
2	情報活用系	SY47(*1)	特許庁におけるデータウェアハウスの機能を一般的な製品で実現可能であることが見込まれるため。	*1: データウェアハウスシステム
3	リソース管理	SY45(*2)	LDAPサーバ機能を一般的な製品で実現可能であることが見込まれるため。	*2: 共通テーブル管理システム
4	運用監視	SY58	ジョブ管理やシステム監視といった機能は一般的な製品で実現可能であることが見込まれるため。	—
5	対外連携	SY33 SY33-2 SY35-2 SY43 SY05 SY44 (*3)	連携先システムに応じた連携方法を実現するものであり、定型化することはできないため。	*3: 優先権証明書交換システム, 外国包袋参照システム, 照会システム(海外／一般ドシエ), 電子現金納付システム, DE料管理システム, 早期管理情報システム
6	ビジネスルール管理	— (新規)	ビジネスルールの実行を一元的に行うといった特殊な責務を担うシステムであるため。	—

上記のうち、外部システム連携サブシステムについては、外部要因による変更の影響を局所化するために、外部システムへのアクセス方法やアクセスの際に生じる内部インタフェースとのギャップの解消の方針について標準化の対象とする。

本書で定めるルールは、そのルールの内容に応じて、設計開発の工程のいつまでに明確化すべきかをSLCPプロセス番号で表す。SLCPプロセス番号に対し、本書でどのようなルールを規定しているのかを下表に例示する。3章に示すルールのうち、「3.1 技術方式」はSLCPプロセス番号2.4.2及び2.4.3のプロセスをカバーするルールであり、「3.2 各要素における個別ルール」は技術方式を補完又は詳細化するルールであるため、個々のルール毎にプロセスが異なり各プロセスに分散したルールとなる。

SLCPプロセス番号	アーキテクチャ標準仕様の例
2.3.2 システム要件定義プロセス	特許庁業務のうち、システムで実現する機能と手作業で行う業務に分類し、それらの関連を表現するためのBPMN記述等のルールや、非機能要求レベルの設定に関するルールを規定。
2.3.3 システム方式設計プロセス	内部システムと外部システム間で通信する際のギャップを吸収するためにサブシステムが備える機能に関するルールを規定。
2.4.2 ソフトウェア要件定義プロセス	システム構成要素レベルの技術方式の概要や、アーキテクチャに影響のある外部設計(サブシステム間サービスインタフェース仕様のプロトコルやデータ項目等の概要、画面設計、帳票設計等)のルールを規定。
2.4.3 ソフトウェア方式設計プロセス	コンポーネントレベルの技術方式の詳細や、内部設計(ビジネスプロセス、ビジネスルール等の外部設計の詳細化、処理設計)、設備設計(プログラムプロダクトに対する要求等)のルールを規定。

上記で示すSLCPプロセス番号より上流のプロセスにおいては本書が規定するルールの対象外ではあるものの、設計内容とその根拠となる上流プロセスでの要求事項との対応付けが取りやすくなるよう成果物の作成方法を工夫すること。

The V-model diagram illustrates the relationship between development and testing phases. The left side of the 'V' represents the development phases, and the right side represents the testing phases. The bottom of the 'V' represents the implementation phase.

- Left Side (Development):**
 - 基本設計 (Basic Design):** Includes system requirement definition (2.3.2), system architecture design (2.3.3), and software requirement definition (2.4.2).
 - 詳細設計 (Detailed Design):** Includes software architecture design (2.4.3).
- Right Side (Testing):**
 - 結合テスト及び総テスト (Integration and System Testing):** Corresponds to the basic design phase.
 - 結合テスト (Integration Testing):** Corresponds to the detailed design phase.
- Bottom (Implementation):**
 - 5.ソフトウェア方式設計プロセス (2.4.3):** The implementation phase.
- Annotations:**
 - 目指すアーキテクチャ (Target Architecture):** Points to the top of the right side of the 'V'.
 - 基本設計 (Basic Design):** Points to the top of the left side of the 'V'.
 - 詳細設計 (Detailed Design):** Points to the bottom of the left side of the 'V'.
 - 結合テスト (Integration Testing):** Points to the bottom of the right side of the 'V'.
 - 結合テスト及び総テスト (Integration and System Testing):** Points to the top of the right side of the 'V'.
 - 遵守 (Compliance):** Two arrows point from the development phases to the testing phases, indicating that the development phases must comply with the testing phases.

7

2.3 強制力(ルールの裁量)

ルールの強制力(裁量)を定めるにあたり、全てのルールに共通する目的は、以下のとおりである。

- 設計に関与するステークホルダ(設計・開発ベンダ等)が設計方針に従って設計を行うことにより目指すべきアーキテクチャを実現すること。

上記目的を達成するために、ルールに対して以下の(1)～(4)に示す4種類の強制力(設計指針、規約、推奨、特例)を定める。また、強制力の関係やルールの運用に係る補足を以下の(5)～(7)に示す。

(1) 設計指針

設計指針とは、ルールを適用する設計対象で選択し得る設計内容の中から何を採用するかを示す指針であって、本指針に従う限りにおいて、設計内容に裁量の余地が残されているものを意味する。裁量の余地を残す理由は以下のとおりである。

- 業務特性に応じた要求仕様を満たす設計を可能とするため。
- ライフサイクルの比較的短いものや未定のもの(例えば、プログラムプロダクトや開発言語)に対して設計の時点におけるベストプラクティス(最も適切な設計内容)を採用可能とするため。
- 本書において引用している標準・規約類(例: 技術参照モデル(TRM))において設計内容の裁量の余地を残しているため。

設計指針には、以下に示す2種類が存在する。例とともに示す。

- サブシステム全体で共通的に設計する対象についての指針を示すもの
例: BPMS製品は、次の機能を備える製品を採用すること。
- 個別の設計によって決定する設計要素についての指針を示すもの
例: ワークフローのアクティビティの粒度は、業務の意味のある最小単位とすること。

なお、裁量の余地が残されているとはいえ、設計内容がベストプラクティスであることの立証責任は設計に関与するステークホルダ(設計・開発ベンダ等)にあるため、設計指針に従いどのような根拠により設計内容としたのかを特許庁に説明すること。

(2) 推奨

推奨とは、設計指針及び当該設計指針に従い導出したベストプラクティスを例示するものの、設計指針に従い別の設計内容を採用するといった、裁量の余地を残すものを意味する。以下に例を示す。

例: フォントの種類及び文字の大きさは極力指定せず、OSやブラウザ等で設定された標準値を利用すること。

仮に、推奨されるベストプラクティスを適用しない場合は、設計に関与するステークホルダはその根拠を特許庁に説明すること。

(3) 規約

規約とは、ルールを適用する設計対象で選択し得る設計内容の中から定められた設計内容を採用するため裁量の余地はないものを意味する。業界標準技術であり、かつライフサイクルも極めて長い等の理由により、設計の裁量の余地を残す必要性がない設計対象については、本書において設計方針に従い導出したベストプラクティスに従うことをルール化する。

規約には、以下に示す2種類が存在する。例とともに示す。

- サブシステム全体で共通的に設計する対象について規約を示すもの
例: システム構成要素間のアクセスは、表に示すアクセスパスに従うこと。
- 個々の設計によって決定する設計要素について規約を示すもの
例: サービスインタフェースの電文形式はSOAPで規定される電文形式とすること。

(4) 特例

特例とは、ルールからの逸脱((6)で後述)をした設計内容のうち今後の開発においてもそのような設計が行われることが想定される場合に、妥当性が認められた条件とともにルールとして明文化したものであり、いわば判例のような情報として示すものである。特例である設計内容を採用する場合、設計に関与するステークホルダ(設計・開発ベンダ等)はその設計内容が特例の条件に合致することを特許庁に説明し、特例として認められることが妥当であるかを協議の上判断するものとする。以下に例を示す。

例： 下位層から上位層のシステム構成要素へのアクセスは原則しないものとする。ただし、別途定める特異なケースについてはそのようなアクセスを特例として許容する。

特例は、実際のシステム構築で逸脱を許容した設計内容のフィードバックを精査し、主に以下に示す性質を持つものを対象に拡充してゆくものとする。

- 技術的な問題が解決したり回避策が登場することにより、将来的に特例とする必要性が無くなることが期待される設計内容
- 事例が充実し、逸脱となる条件が十分に一般化され整理でき、将来的に設計指針や規約となることが期待される設計内容

(5) 4種類の強制力の関係

上述の4種類の強制力のうち、設計指針及び規約は、独立して存在し得る強制力である。これら強制力は、前者が一定の裁量を認める強制力であるのに対し、後者は裁量を認めない強制力であることから、排他的な関係となるため、ルールを適用する設計対象毎にいずれか一方の強制力が適用される。

上述の4種類の強制力のうち、推奨及び特例は、単独で存在できない強制力であり、設計指針又は規約に従属する強制力となる。

4種類のルールの関係を下図に示す。

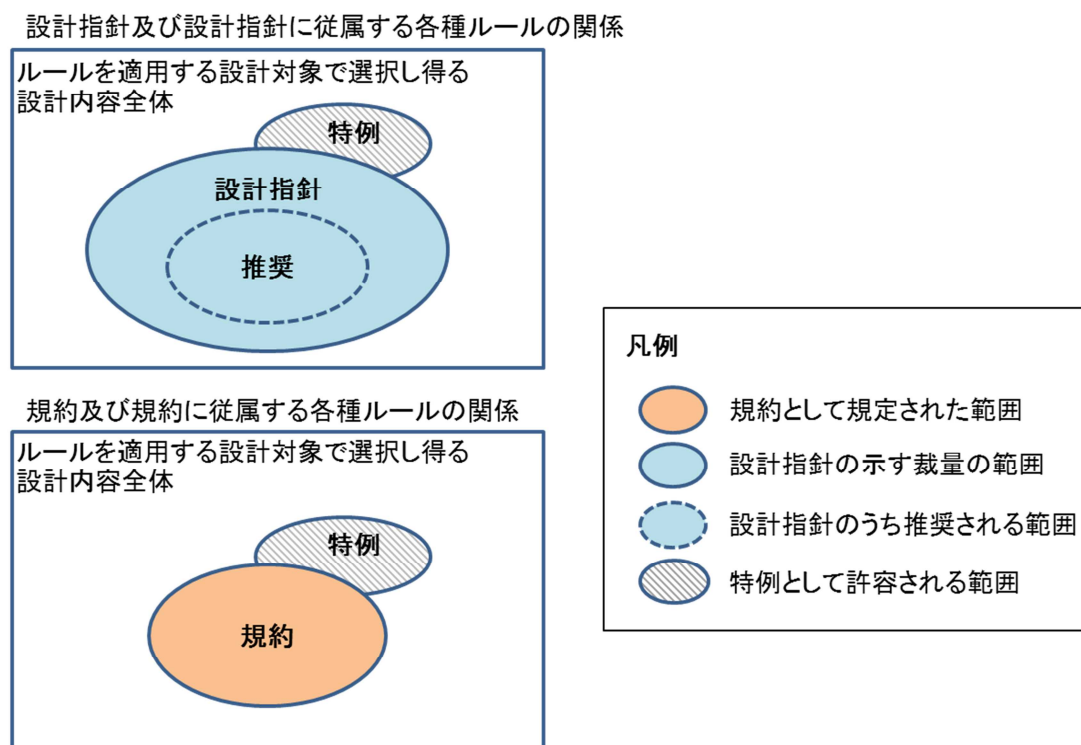
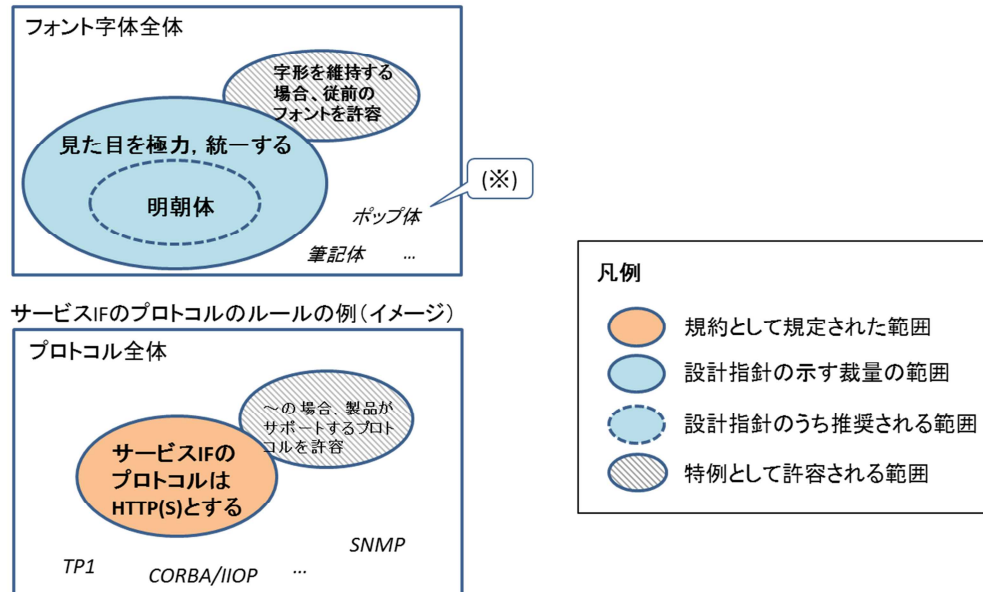


図 2.3-1 ルールの種類の関係

帳票設計のフォント字体のルール例(イメージ)



(※) POP体、筆記体が、特例(字形を維持する場合に従前のフォントとして使用する)に該当しない場合の例。
プロトコルの例も同様にTP1、CORBA/IIOP、SNMPが特例に該当しない場合の例。

図 2.3-2 ルールの関係の具体例

(6) ルールの逸脱の扱い

これまで述べた設計指針や規約等は特許庁アーキテクチャを実現するために定めたルールであり、原則として遵守することを必須とする。しかしながら、要件を満たすシステムを実現する上で、設計指針や規約をどうしても遵守できない、又は遵守することにより過度に複雑化する等の大きなデメリットが発生する場合があるため、そのような条件に該当すると認められる場合は、ルールで定めた設計内容の範囲からの逸脱を限定的に許容する。

設計指針及び規約からの逸脱は、あらゆる設計内容を許容するものではない。設計に関与するステークホルダ(設計・開発ベンダ等)がその逸脱する根拠と設計内容の妥当性を特許庁に説明し、協議の上判断するものとする。

(7) 留意点

各ルールに対する設計内容の裁量の範囲の妥当性については、実際のシステム構築からのフィードバックの蓄積により確実なものとなる。したがって、システム構築における様々なリスクを考慮し、確実に規約化できるものを除き設計指針又は推奨として規定する。そして、設計からのフィードバックに基づき各ルールの裁量の範囲を見直し、必要に応じて本書を改定することとする。

3. ルール(設計指針・規約・推奨・特例)

本章では、「2 ルールの考え方」に基づいて分類されるルールを規定する。

ルールの規定にあたり、本書の利用者に対するルールの理解を高めるため、以下に示すとおり、システムを構成する各要素⁴の横断的な関係を規定するルール(技術方式)と各要素における個別ルールとに分けて規定する。

(1) 技術方式(3.1章)

本章では、技術方式を静的な構造と動的な振る舞いの観点からモデル化し、当該モデルに基づきルール化する事項を規定する。具体的には、静的な構造として多階層構造をモデル化し、システム構成要素の責務、各階層への配置、依存関係、システム構成要素の論理構成(ソフトウェア構成)等をルールとして規定する。また、各システム構成要素の内部を構成するコンポーネントも同様に、APレイヤ・コンポーネントにおける静的な構造や動的な振る舞いをモデル化し、ルール化する事項を規定する。

(2) 各要素における個別ルール(3.2章)

本章では、システムを構成する各要素の個別ルールを規定する。各個別ルールは、以下に示すA(「3.2.1章」)、B(「3.2.2～3.2.7章」)及びC(「3.2.8～3.2.13章」)に分類される。

A. 多階層構造の層の境界において遵守すべきルール

サブシステム間や多階層構造の層の境界において遵守すべきルールを定める。

- インタフェースのルール

B. 多階層構造の層毎に遵守すべきルール

サブシステムを構成する多階層構造の層毎に遵守すべきルールを定める。

- UI層、プレゼンテーション層
- ワークフロー層
- 業務アプリケーション層
- 外部システム連携層
- 基盤機能層

C. システム開発全般で遵守すべきルール

上記A及びBに該当しないシステム開発全般で遵守すべきルールを定める。

- データ設計
- プログラムプロダクトの選定
- 文字コード
- 開発言語の選定
- 非機能要求グレード
- システム機能(システム監視やセキュリティ等)

本章におけるルールの表記方法は、以下のとおり。

- 規約及び設計指針は連番を付与し列举する。また、設計指針はルールの表記上、「指針」と記載する。
- 推奨や特例ルールも連番を付与し、付随する規約又は設計指針のルールの直後に字下げして記載する。

⁴ システムを構成する様々な粒度の要素(例えば、システム構成要素やコンポーネント等)といった要素

以下にルールを表記例を示す。

なお、表記例における「スコープ」の詳細については「2.1 スコープ(ルールの適用範囲)」を、「プロセス」の詳細については「2.2 プロセス(ルールの適用工程)」を、「規約・指針・推奨・特例」の詳細については「2.3 強制力(ルールの裁量)」を参照のこと。

スコープ:	XXXXXXXXXX
プロセス:	XXXXXXXXXX
規約1:	XXXXXXXXXX
規約2:	XXXXXXXXXX
特例1:	XXXXXXXXXX
指針1:	XXXXXXXXXX
特例1:	XXXXXXXXXX
特例2:	XXXXXXXXXX
推奨1:	XXXXXXXXXX

3.1 技術方式

3.1.1 システムの静的な構造

本章に示す技術方式のルール(設計指針, 規約, 推奨及び特例)は, 以下の章に示す。

- 3.1.1.3 多階層構造の詳細
- 3.1.1.4 APレイヤ・コンポーネント定義

3.1.1.1 多階層構造を採用する目的

特許庁システムにおいては, システムの静的な構造としてシステム構成要素を各層に配置した多階層構造を採用し, 以下に列挙する要求を達成することを目的とする。

- 各層に配置したシステム構成要素の責務を適切に定めることにより, システム構成要素に改造が必要になった場合における影響箇所の特定容易化, 改造箇所の局所化。
- 層間のアクセスパスを標準化することにより, システム構成要素に改造が必要になった場合における影響箇所の拡散の極小化。
- 各層において利用するプログラムプロダクトの仕様を標準化することにより, プログラムプロダクトの交換容易性の向上。

3.1.1.2 多階層構造の概要

3.1.1.2.1 層の種類及び責務

層の種類及び各層に割り当てられた機能(責務)を下表に示す。

表 3.1-1 層の種類及び責務

項番	層	責務
1	UI層	ユーザに対して入出力を伴う直接的なインタフェース(GUI)
2	プレゼンテーション層	UI層とワークフロー層・業務アプリケーション層との連携
3	ワークフロー層	業務アプリケーションの連携をビジネスプロセスとして管理及び可視化
4	業務アプリケーション層	業務処理の実行
5	基盤機能層	データベース層へのアクセス
6	データベース層	特許庁システム全体で共有すべきデータの管理
7	外部システム連携層	内部システムと外部システム間で通信する際のギャップの吸収
8	ビジネスルール管理層	ビジネスルールの管理, 実行

3.1.1.2.2 層間の関係

層間には以下に列挙する関係を持たせる。

- UI層からデータベース層まで(「表 3.1-1 層の種類及び責務」の項番1から6)の間の層については階層化する。
- 外部システム連携層については, 外部システムを様々な内部システムの層としてエミュレートする特許庁システム全体でただ一つの責務を持つため階層化の対象外とする。
- ビジネスルール管理層については, ビジネスルールの管理, 実行を一元的に行う特許庁システム全体でただ一つの責務を持つため, 階層化の対象外とする。
- 層間の依存関係を統制するためアクセスパスの方向は同位層か下位層へ向かう向きとする。

3.1.1.2.3 層内のシステム構成要素の関係

層内のシステム構成要素の関係については, 各層の責務に従い以下に示す関係を持たせる。

- 特許庁業務を実現する上で, 必要最小限のアクセスパスを設ける。
- 業務の流れをビジネスプロセスとして表すのはワークフロー層の責務であるため, 業務アプリケーション層内において, 業務アプリケーションの連鎖的な実行の流れによりビジネスプロセスが構成されないようにシステム構成要素間のアクセスを制限する。

3.1.1.3 多階層構造の詳細

スコープ： 階層定型化サブシステム及びインタフェース定型化サブシステム

プロセス： 2.4.2

規約1： サブシステムの機能は、「表 3.1-2 システム構成要素の種類及び責務」に示された責務を担うシステム構成要素から実現すること。また、「3.1.1.3.1.3 サブシステムの範囲」に示される分類毎のサブシステムの機能は、それぞれ「表 3.1-5 サブシステムの分類とシステム構成要素」に掲載されたシステム構成要素から実現すること。

規約2： システム構成要素間のアクセスは、「3.1.1.3.1.2 (1)多階層構造の概要に基づくアクセスパス」に示すアクセスパスとすること。また、「3.1.1.3.1.2 (3)アクセスパスの制限事項」に示すとおり制限すること。

特例1： 「3.1.1.3.1.2 (2)アクセスパスの特例」に示すアクセスパスは、下位層のシステム構成要素から上位層のシステム構成要素へのアクセスパスを許容する。

規約3： 各システム構成要素のソフトウェア構成は、「図 3.1-12 ソフトウェア構成図」に示す構成とすること。

スコープ： 階層定型化サブシステム及びインタフェース定型化サブシステム

プロセス： 2.4.2, 2.4.3

規約1： サブシステムの処理方式は、「表 3.1-3 処理方式による分類」に示す処理方式とすること。

多階層構造における各層とシステム構成要素との関連を下図に示す。

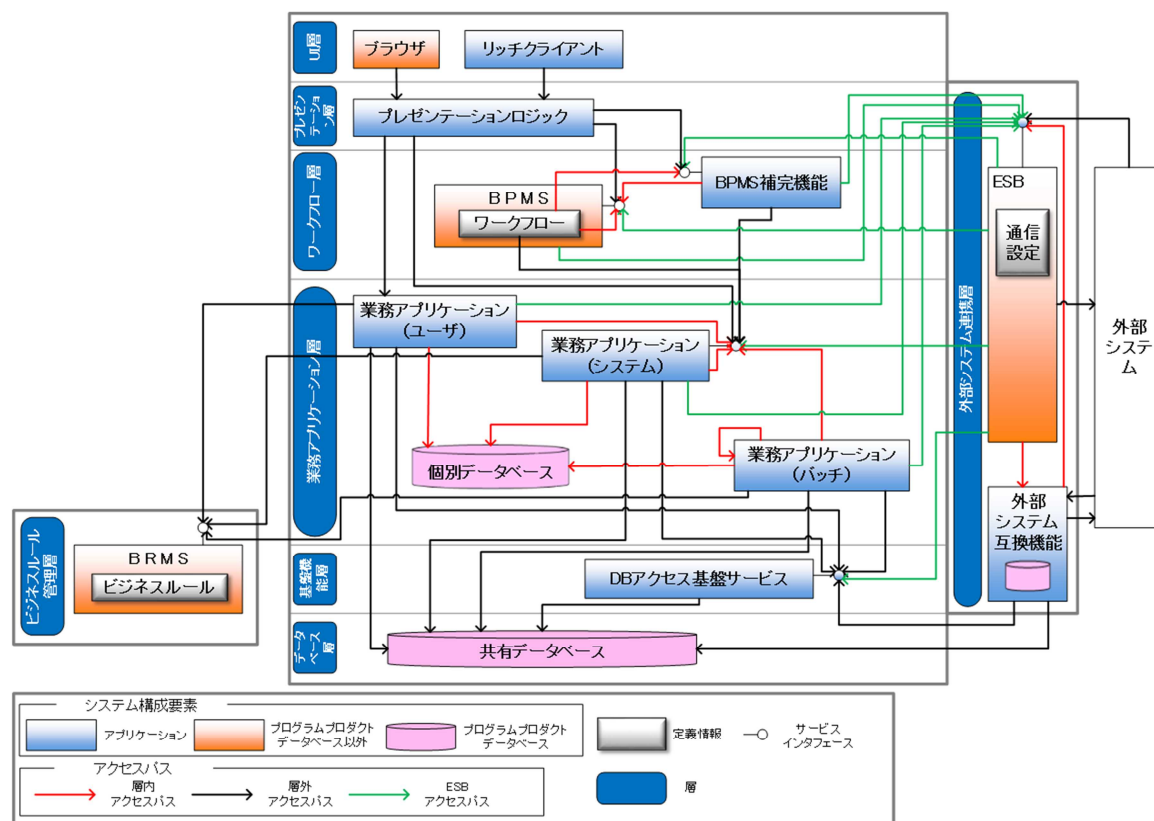


図 3.1-1 システム構成要素の関連図

3.1.1.3.1 システム構成要素の種類及び責務

特許庁システム全体として必要とする機能を抽出するとともに「3.1.1.2 多階層構造の概要」に基づきシステム構成要素を整理し、システム構成要素の種類及び各システム構成要素に割り当てられた責務として下表に示す。

表 3.1-2 システム構成要素の種類及び責務

項番	層	システム構成要素	責務
1	UI層	ブラウザ	Webアプリケーションにおいて、ユーザが操作するための画面機能の提供
2		リッチクライアント	Webサービスインタフェースによるリッチクライアントにおいて、ユーザが操作するための画面機能の提供
3	プレゼンテーション層	プレゼンテーションロジック	基本的にビジネスロジックを含まず、UI層からの要求を下位層にあるシステム構成要素で用いられるデータ構造やデータ形態を相互変換するシステム構成要素
4	ワークフロー層	BPMS	ワークフローの制御による業務アプリケーションの連携
5		BPMS補完機能	ワークフローの再開箇所の判定等、BPMSにおけるワークフロー制御の補完
6	業務アプリケーション層	業務アプリケーション(ユーザ)	ユーザからの入力情報に基づく業務処理又はユーザが必要とする情報の取得
7		業務アプリケーション(システム)	以下に示す責務のいずれか ● BPMSのサービスタスクに対応し業務として意味のある一連の処理の実行 ● 他のシステム構成要素で必要とするデータの取得又は更新
8		業務アプリケーション(バッチ)	BPMSのサービスタスクに対応せず所定のタイミング(期間、時間等)で処理の実行
9		個別データベース	サブシステム固有の業務データの管理
10	基盤機能層	DBアクセス基盤サービス	共有データベースにアクセスするためのサービスの提供
11	データベース層	共有データベース	システム全体で利用する共有データの管理
12	外部システム連携層	ESB	内部システムと外部システムの連携方式の違い(ギャップ)の吸収
13		外部システム互換機能	ギャップのうちESBで吸収しない部分の補完
14	ビジネスルール管理層	BRMS	ビジネスルールの管理, 実行

3.1.1.3.1.1 システム構成要素の処理方式による分類

システムの構造を定型化するため、代表的な処理の処理方式を分類し定める。処理方式の分類を下表に示す。

各処理方式におけるAPレイヤ・コンポーネント定義は、「3.1.1.4 APレイヤ・コンポーネント定義」を参照のこと。各処理方式における動的な振る舞いは、「3.1.2 システムの動的な振る舞い」を参照のこと。

表 3.1-3 処理方式による分類

項番	処理方式	分類	説明
1	オンライン処理方式	画面系	ユーザ操作により、画面からネットワークを介してサーバに処理要求を行い、応答を返却する方式。
2		サービス系	BPMS等からの処理要求に対し、サービスとして応答を返却する方式。
3	バッチ処理方式	複数件一括処理	多量のデータを取りまとめ、一定期間あるいはデータ量毎に一度にまとめて処理する方式。
4		単件一括処理	単件の処理を一度にまとめて処理する方式。単件処理は要件により逐次又は並行で実行する。
5	ディレードバッチ処理方式	画面系	ユーザ操作により、画面からサーバに処理要求を行い、要求受領の旨の応答を即時に返却した後、要求された処理を行う方式。
6	帳票出力方式	クライアント帳票出力	サーバで作成した帳票ファイルをクライアントにダウンロードする方式。
7		サーバ帳票出力(同期)	サーバで帳票ファイルの作成と、帳票ライブラリへの印刷要求を同期処理で行う方式。印刷は、業務要件によってフロアプリンタ又はサーバプリンタに印刷可能とする。
8		サーバ帳票出力(非同期)	サーバで帳票ファイルの作成と、帳票ライブラリへの印刷要求を非同期処理で行う方式。印刷は、業務要件によってフロアプリンタ又はサーバプリンタに印刷可能とする。
9	連携処理方式	BPMSを介したサブシステム間連携	業務フローとしてサブシステム間の連携を行う場合は、その連携ルートをBPMS上のワークフローにて記述し、BPMSを介して他サブシステムのBPMSや業務アプリケーション(システム)と連携を行う方式。
10		BPMSとジョブ管理システム間の連携	ToBeアーキテクチャでは、基本的にBPMSによって単件オンライン化されるが、一部の業務では業務アプリケーション(バッチ)により一括で処理を行うものも存在する。それらの処理を連携する処理方式。 オンライン処理方式(サービス系)とバッチ処理方式を組み合わせ、データベースやファイルでデータの授受を行い、スケジューラ起動等の契機により連携処理を行う。
11		バッチ処理方式のサブシステム間連携	サブシステム間にて、バッチ処理の連動やデータの授受を行う方式。 共有データベースやFTP(S)によるデータ授受を行い、ジョブ管理システムを介して連携を行う。

項番	処理方式	分類	説明
12		外部システム連携層を介した外部システム連携	外部システムとの連携には、それぞれの外部システムに固有のインタフェースやプロトコルが存在するため、それらのギャップを一元的に吸収する外部システム連携層を介してのみ行うものとする。その際の処理方式。 内部システムから外部システム連携を行う際、外部システム連携層が提供する内部インタフェースと同等のアクセス方法を用いて処理を実施する。

オンライン処理方式とバッチ処理方式の処理方式の分類の範囲を、下図に示す。

なお、帳票出力方式及び連携処理方式は、オンライン処理方式やバッチ処理方式と組み合わせて実行される。

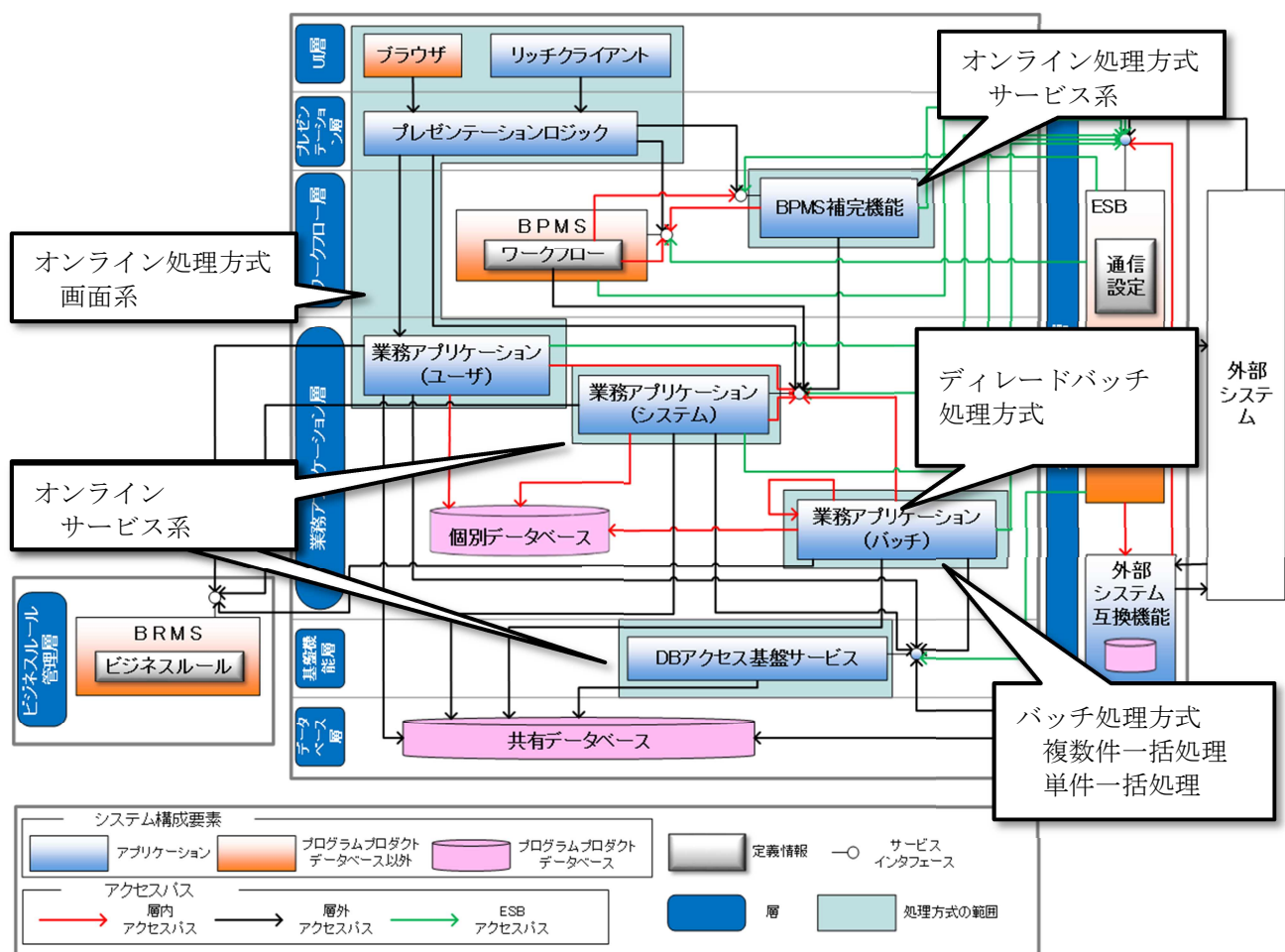


図 3.1-2 処理方式の分類の範囲

3.1.1.3.1.2 システム構成要素間のアクセスパス

(1) 多階層構造の概要に基づくアクセスパス

「3.1.1.2 多階層構造の概要」に基づきシステム構成要素間のアクセスパスを整理すると以下のとおりとなる。

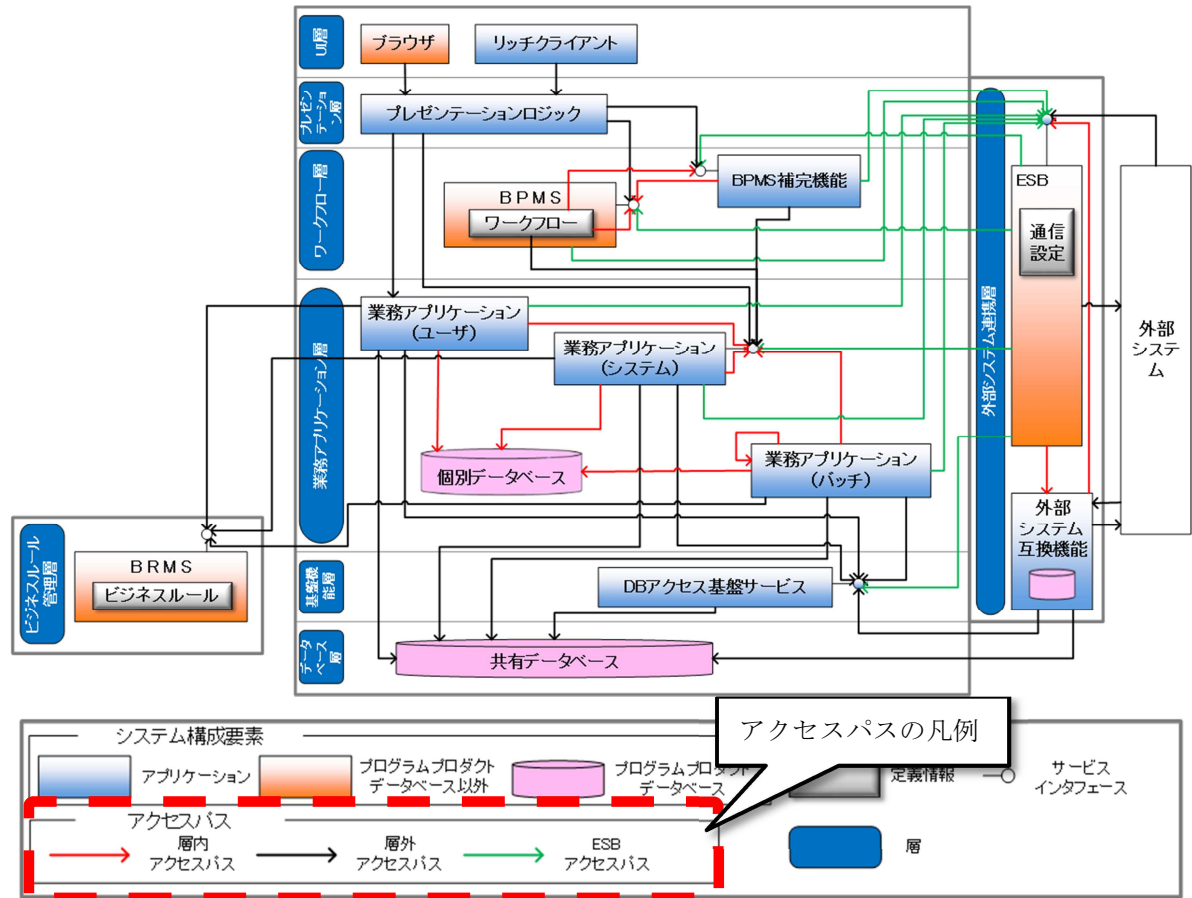


図 3.1-3 原則的アクセスパス

(2) アクセスパスの特例

下表、下図に示すアクセスパスについては、下位層のシステム構成要素から上位層のシステム構成要素へのアクセスパスとする。

表 3.1-4 アクセスパスの特例と具体例

項番	アクセス元	アクセス先	具体例
1	業務アプリケーション(システム)	BPMS BPMS補完機能 ⁵	● 業務アプリケーション(システム)のビジネスロジックで、別のビジネスプロセスインスタンスのステータス情報に基づいた処理を行うためのステータス取得を目的としたアクセスを行う場合。
2	業務アプリケーション(バッチ)	BPMS BPMS補完機能 ⁵	● 人事異動のためビジネスプロセスインスタンスに割り当てられている担当者をバッチにて一括で変更する等の目的でアクセスする場合。 ● バッチからビジネスプロセスを開始する場合。

また、下図の※で示すアクセスパス(業務アプリケーション(バッチ)から業務アプリケーション(バッチ))は、同一階層内パスではあるが、自サブシステムから他サブシステムへの連携を表す。詳細は、「3.1.2.5. 1.3 バッチ処理方式のサブシステム間連携」を参照のこと。

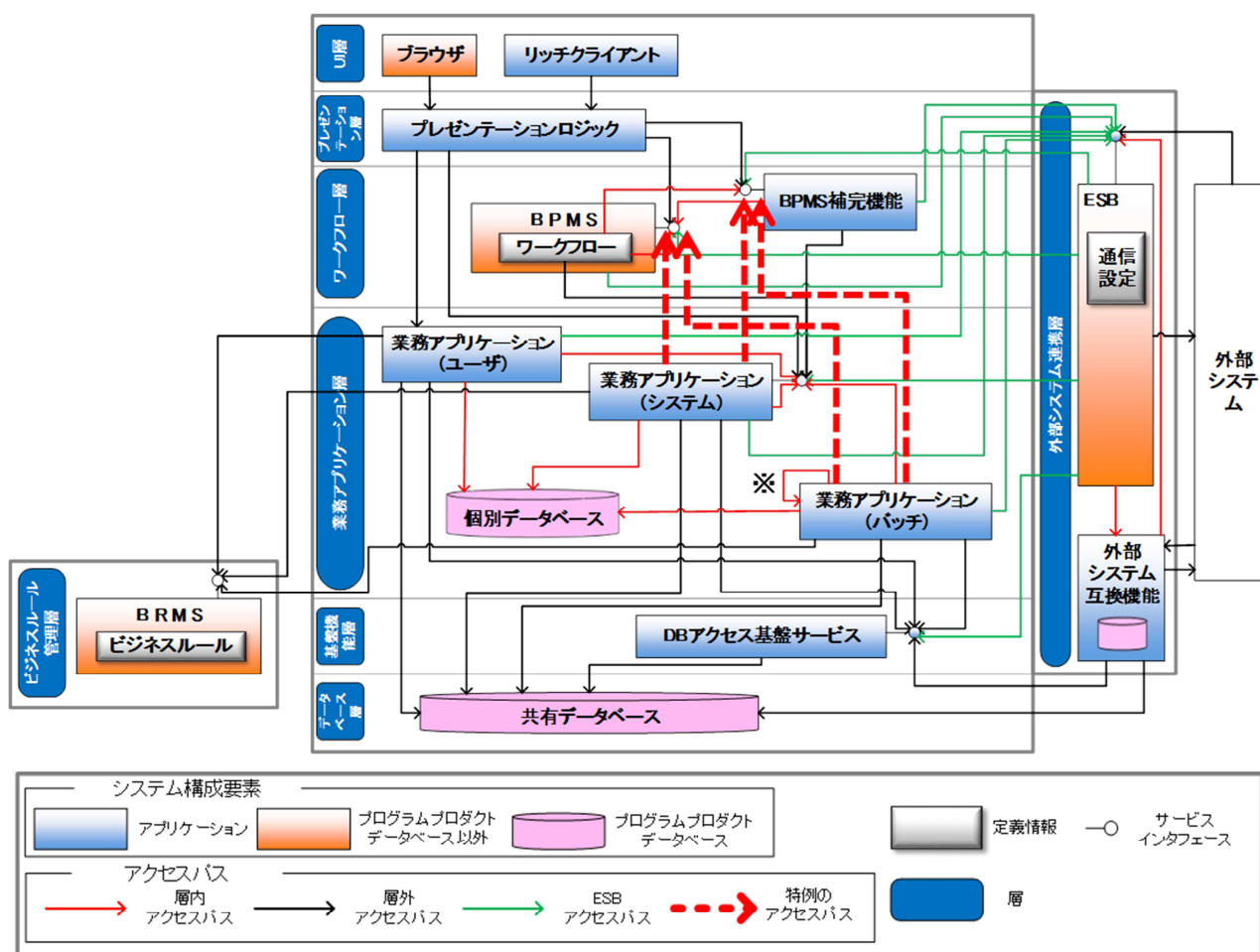


図 3.1-4 アクセスパスの特例

⁵ BPMS補完機能は、将来的にBPMS製品によって実現できることを期待して補完するものであるため、アクセス元からは区別の必要がないようアクセスパスの特例もBPMSとBPMS補完機能の双方を許容する。

(3) アクセスパスの制限事項

業務アプリケーション(システム)へのアクセス(下図の太線)は、アプリケーション同士の複雑な連携を排除しお互いを疎の関係とする目的から、提供するサービスに応じてアクセスを許容するシステム構成要素を制限する。詳細は、「3.2.4.2 サービスインタフェースの粒度 (3)アクセスパスの制限事項に関するルール」を参照のこと。

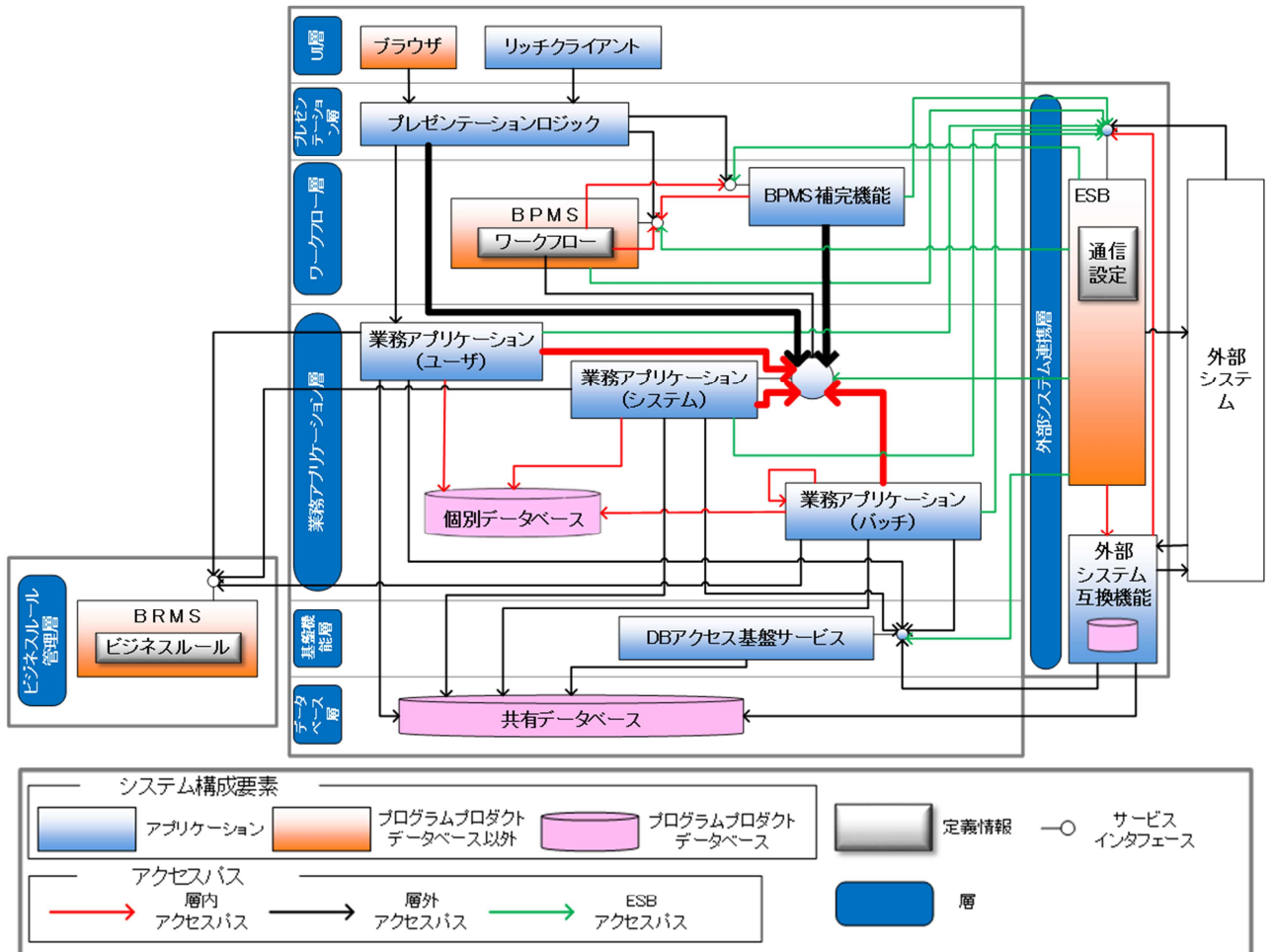


図 3.1-5 制限事項を持つアクセスパス

3.1.1.3.1.3 サブシステムの範囲

多階層構造の各層に配置されたシステム構成要素は、以下のとおり2グループに分類できる。

① 業務又は事件毎に構築されるため定型化が必要なシステム構成要素のグループ

本グループに属するシステム構成要素は「表 3.1-2」の項番2～9並びに10及び11である。当該グループに含まれるシステム構成要素は各々の依存関係が極めて強いことから、初期構築時に一体として設計を行う必要があり、1サブシステムの範囲として含める。

当該サブシステムは、階層化した複数の層から構成され、業務又は事件毎に構築される。各サブシステムのアーキテクチャは定型化されていることから、階層定型化サブシステムと呼ぶ⁶。

② 共通的に利用される単品扱いのシステム構成要素のグループ

本グループに属するシステム構成要素は「表 3.1-2」の項番12以降である。当該システム構成要素は、責務が類似し依存関係の強いものもあれば、責務が全く異なり依存関係の薄いものもある。類似する責務のシステム構成要素を各々まとめてサブシステムの範囲とする。当該サブシステムは、インタフェースを定型化するものの、それ以外については個別にアーキテクチャを定める性質を有することから、インタフェース定型化サブシステムと呼ぶ⁷。

サブシステムの分類とシステム構成要素の関係を下表に示す。なお、システム構成要素のうちブラウザについては業務用PCにインストールされるプログラムプロダクトとなるため、どのサブシステムの範囲にも属さないものとする。

表 3.1-5 サブシステムの分類とシステム構成要素

項番	サブシステムの分類	サブシステム	システム構成要素
1	階層定型化サブシステム	方式審査(出願), 分類付与, 実体審査等	リッチクライアント, プレゼンテーションロジック, BPMS, BPMS補完機能, 業務アプリケーション(ユーザ), 業務アプリケーション(システム), 業務アプリケーション(バッチ), 個別データベース, DBアクセス基盤サービス, 共有データベース
2	インタフェース定型化サブシステム	ビジネスルール管理	BRMS
		外部システム連携	ESB, 外部システム互換機能
		情報活用系, リソース管理, 運用監視, 対外連携	-

⁶ 特に、項番2～9のシステム構成要素からなるサブシステムを、業務遂行・共通リソース系サブシステムと呼び、項番10及び11のシステム構成要素からなるサブシステムを、共有データベース管理サブシステムと呼ぶ。

⁷ 特に、システム構成要素としてBRMSを含むものをビジネスルール管理サブシステムと呼び、システム構成要素としてESB及び外部システム互換機能を含むものを外部システム連携サブシステムと呼ぶ。

多階層構造図におけるサブシステムの範囲を下図に示す。

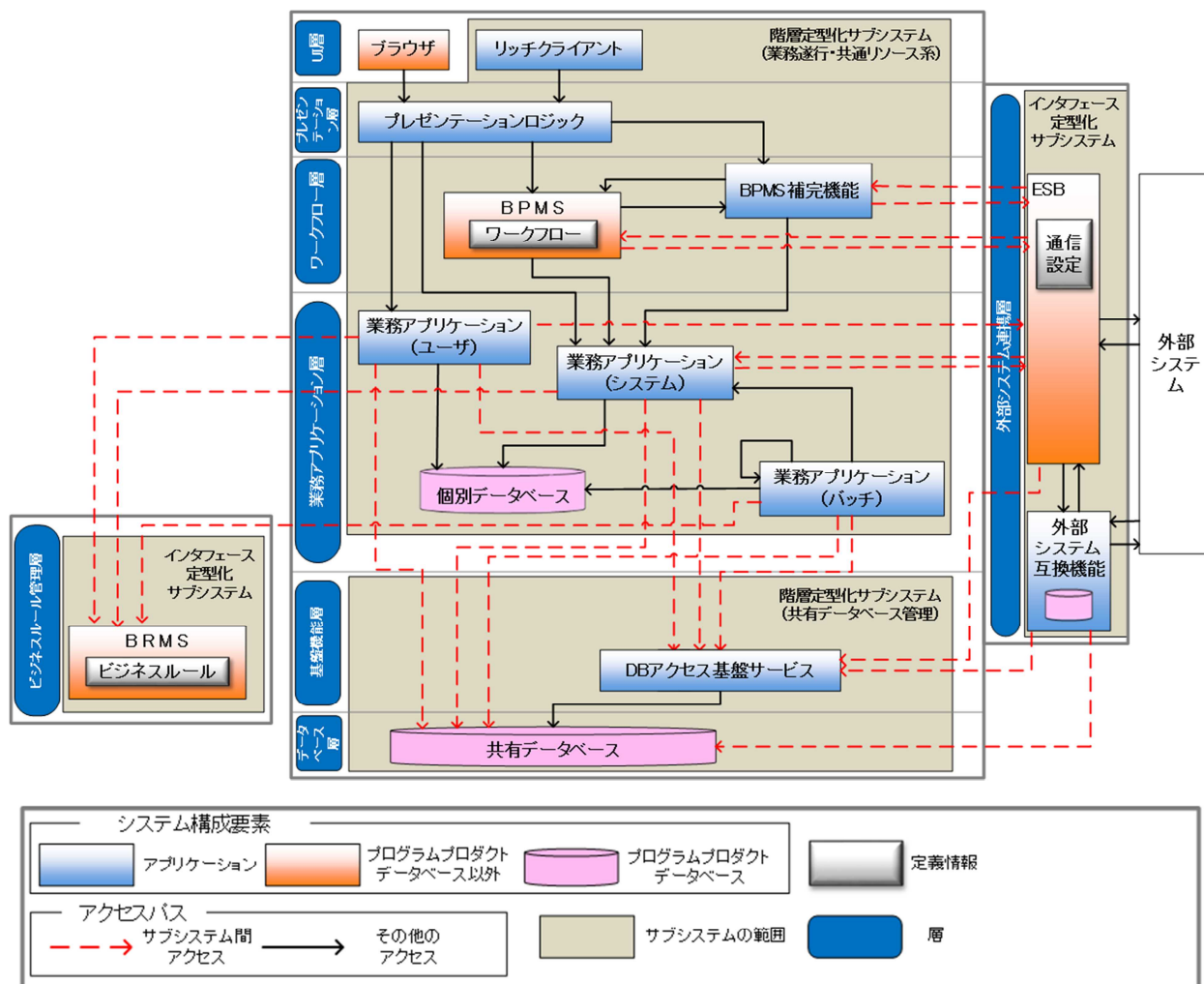


図 3.1-6 サブシステムの範囲

階層定型化サブシステム(業務遂行・共通リソース系)が複数存在している場合における, サブシステム間のアクセスを「図 3.1-7 階層定型化サブシステム(業務遂行・共通リソース系)間の連携」に示す。

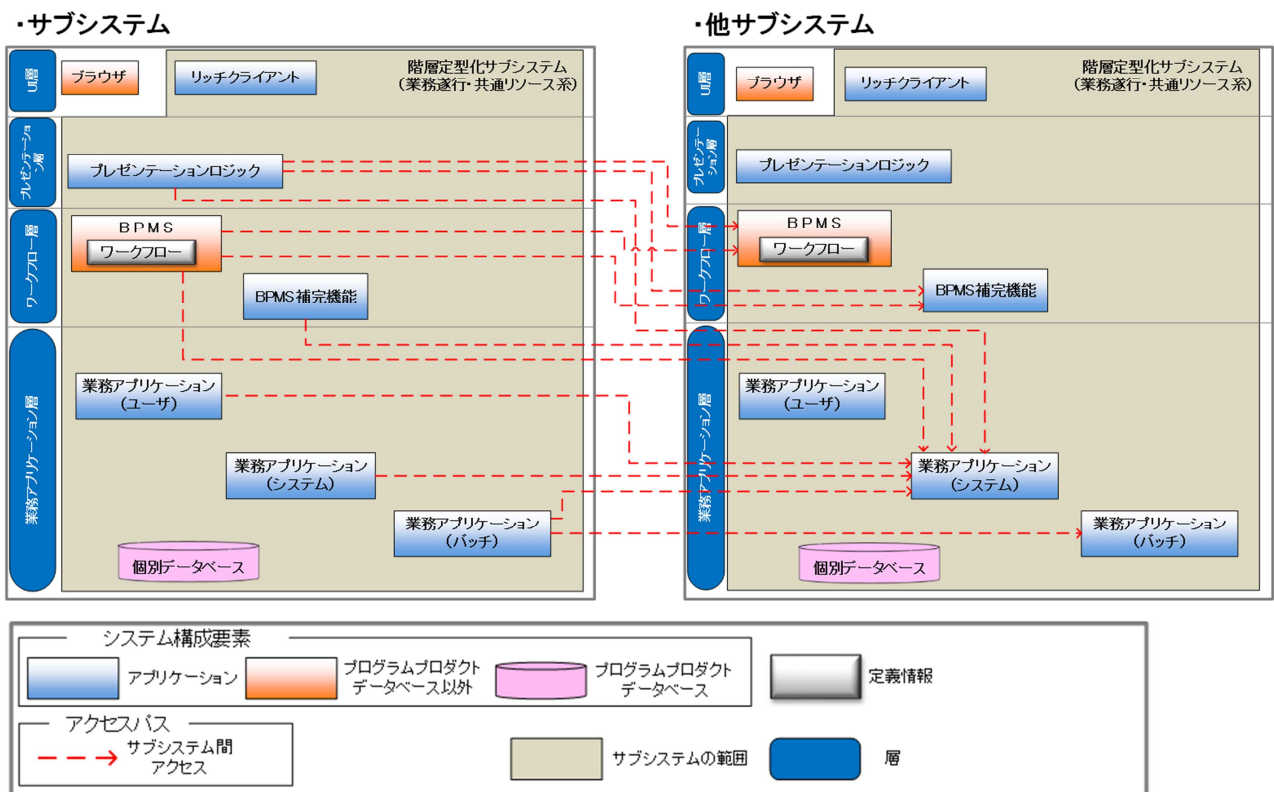
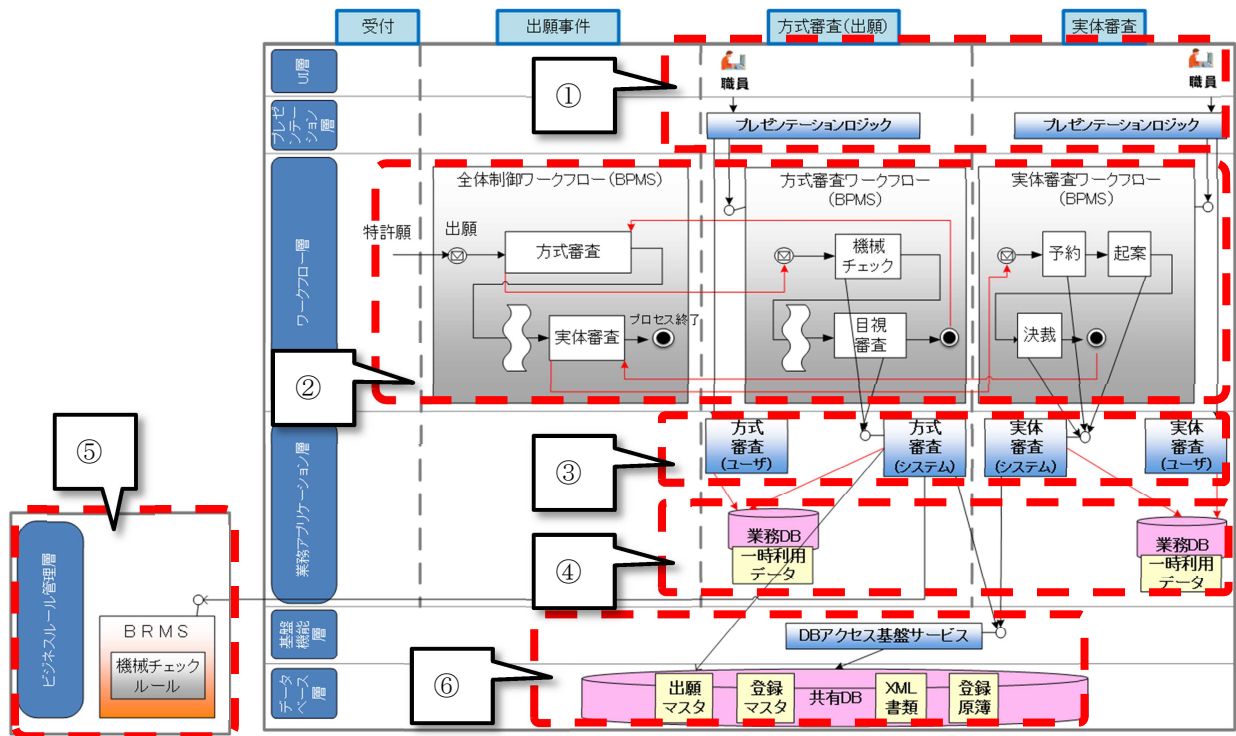


図 3.1-7 階層定型化サブシステム(業務遂行・共通リソース系)間の連携

3.1.1.3.1.4 複数サブシステムにおける連携イメージ(参考)

- (1) 多階層構造を適用した場合の複数サブシステムにおける連携イメージ
- 参考として、多階層構造を適用した場合の複数サブシステムにおける連携イメージを、方式審査及び実体審査を題材として下図に示す。
- 詳細については、『全体システム概念設計書』及び各サブシステムの『個別業務システム概念設計書』を参照のこと。



- ① 庁職員等が画面を操作し業務を行う。また、画面からワークフローの開始や状況の確認等も行う。
- ② ワークフローを制御する。
- ③ 業務アプリケーションを実行する。
- ④ サブシステム固有の業務データを管理する。
- ⑤ ビジネスルールを管理、実行する。
- ⑥ システム全体で共有するデータを管理する。

図 3.1-8 方式審査及び実体審査を題材とした連携イメージ(参考)

上図の②のワークフローを階層化して図示したものを「図 3.1-9 方式審査及び実体審査を題材とした連携イメージ(ワークフローを階層化)(参考)」に示す。

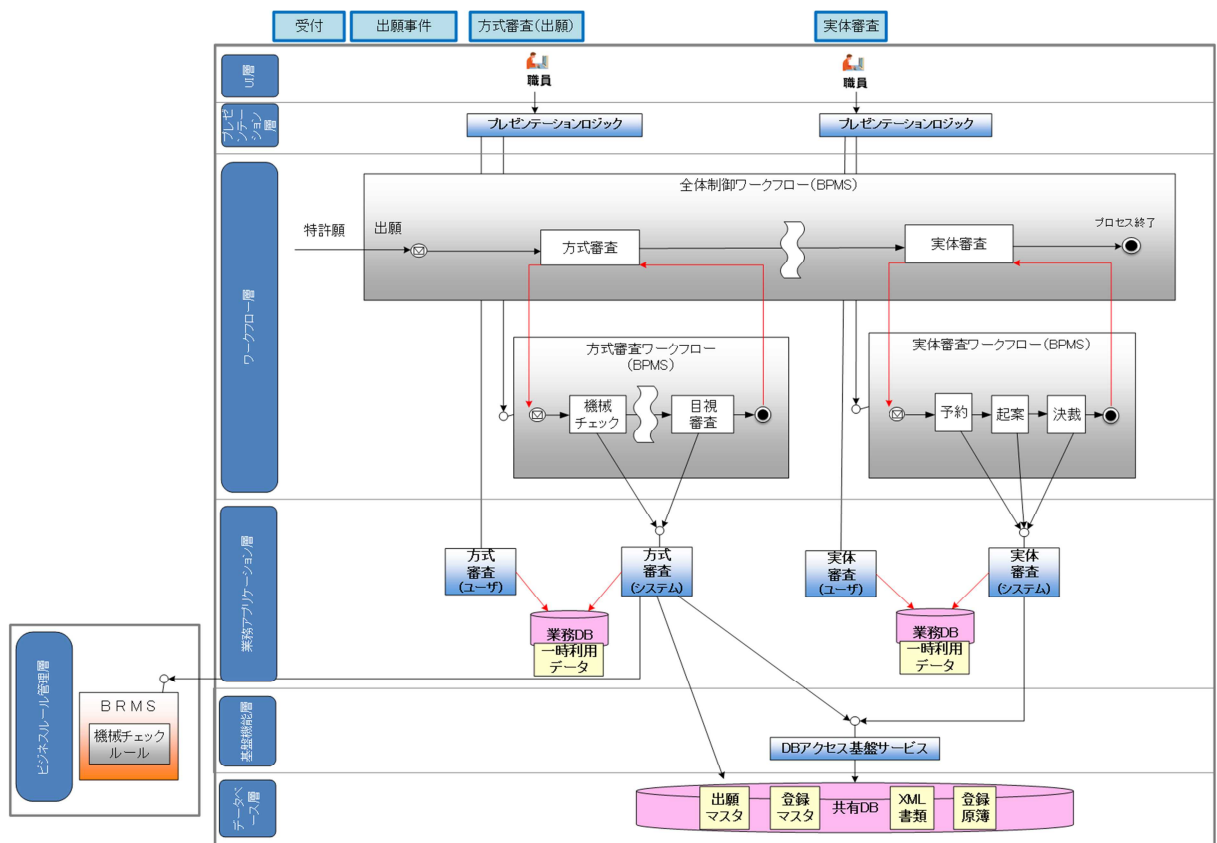


図 3.1-9 方式審査及び実体審査を題材とした連携イメージ(ワークフローを階層化) (参考)

(2) 外部システム連携層を介した内部システムと外部システムとの連携イメージ

参考として、外部システム連携層を介した内部システムと外部システムとの連携イメージを下図に示す。外部システム連携層によりギャップ吸収された結果、外部システムがあたかも刷新化されたシステム(内部システム)のようにエミュレートされる。これにより、呼び出し元から見て、呼び出し先が内部システム、外部システムのいずれであろうとも、内部システムのサービスインタフェースを呼び出すように扱える。したがって、内部システムと外部システム連携層間は、外部システムが刷新され内部サブシステムとなったときのインタフェースで連携が可能となる。

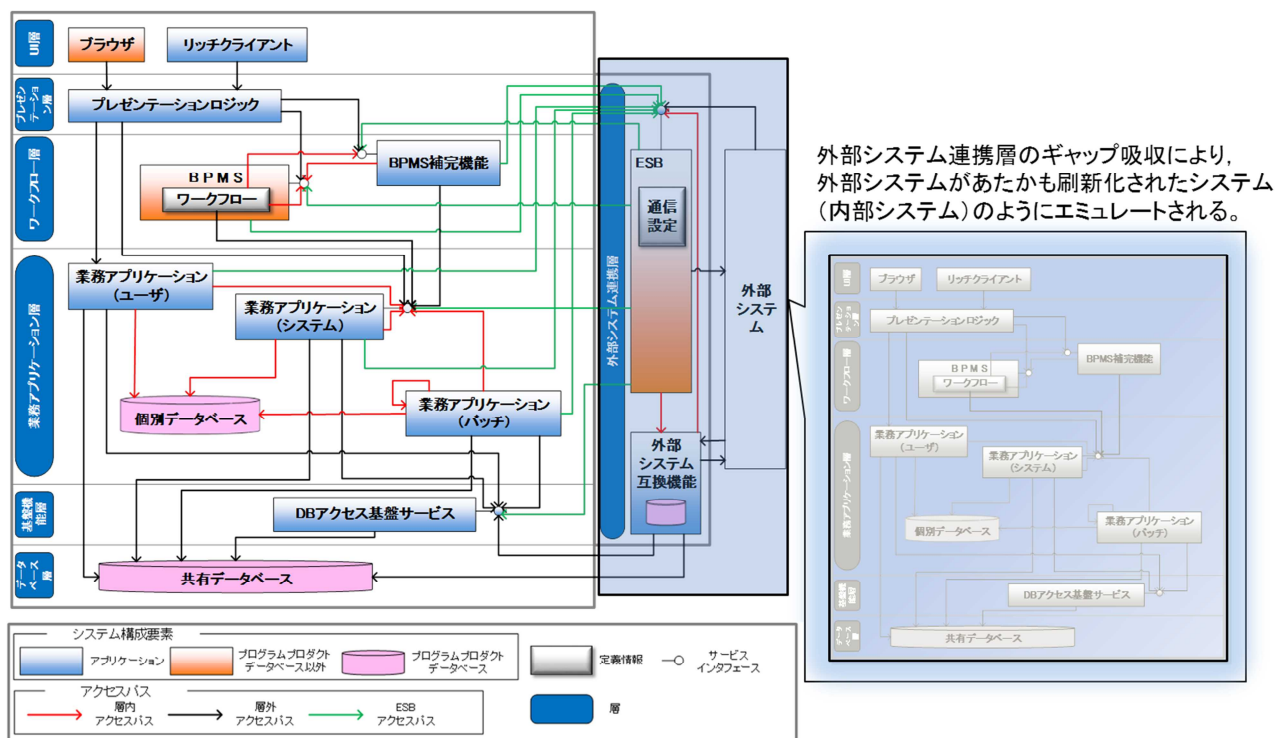


図 3.1-10 内部システムと外部システムとの連携イメージ(参考)

(3) サブシステムと多階層構造の関連のイメージ

参考として、サブシステムと多階層構造の関連のイメージを下図に示す。下図の平面図として表すものは、サブシステムと当該サブシステムが扱うデータを示す。断面図として表すものは多階層構造を示す。刷新後の特許庁システムは、サブシステム(と当該システムが扱うデータ)及び多階層構造の軸を持つキューブのような関係を持つ。

多階層構造は以下の特徴があると言える。

- UI層から業務アプリケーション層はサブシステム毎に構築する。
- BPMSは各サブシステムを横断的に処理するワークフローを持つ。
- 基盤機能層とデータベース層は、UI層から業務アプリケーション層のサブシステム間で共通的に利用される。
- 外部システム連携層は、「(2)外部システム連携層を介した内部システムと外部システムとの連携イメージ」に示したとおり、外部システムを内部システムとしてエミュレートする。

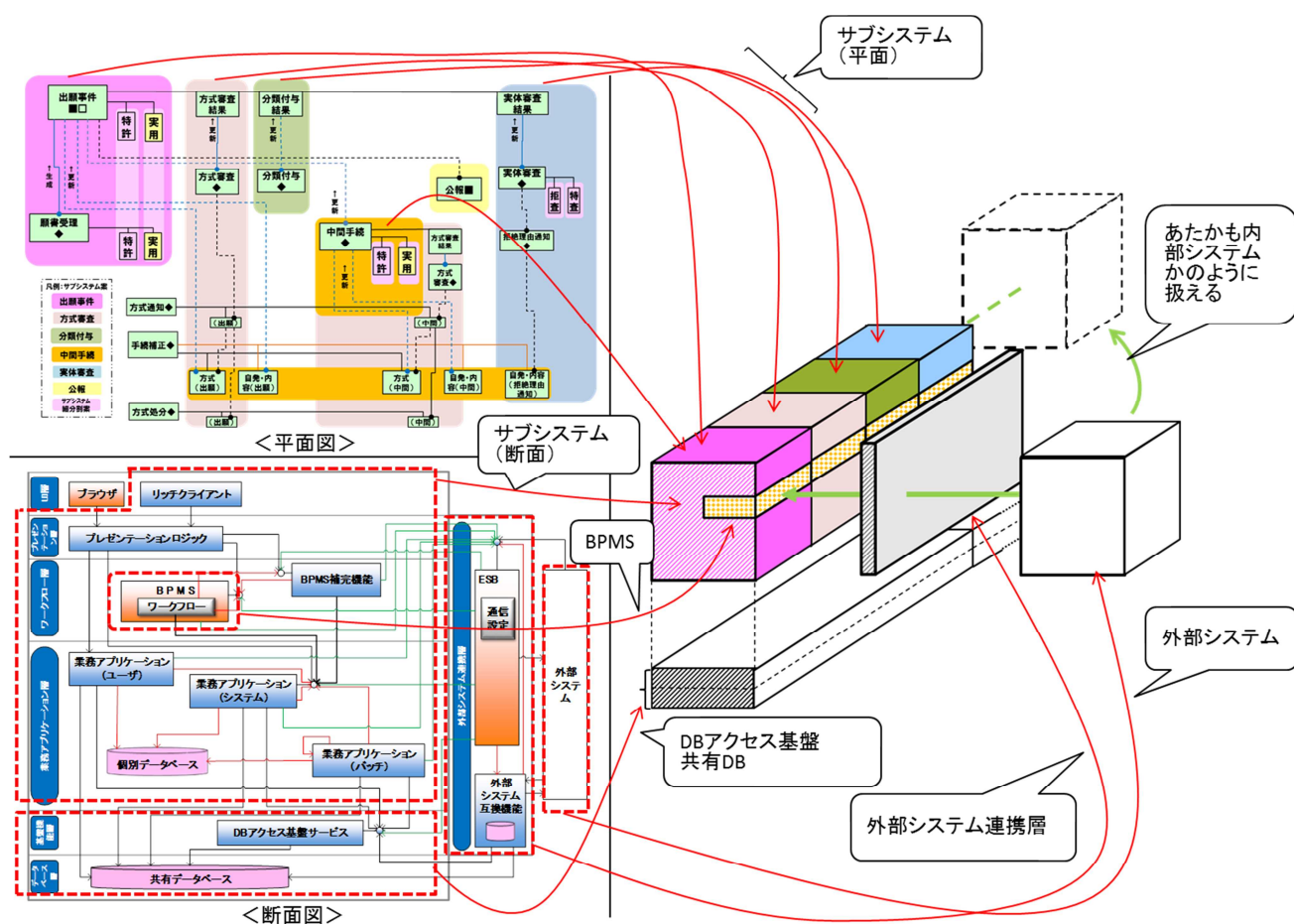


図 3.1-11 サブシステムと多階層構造の関連のイメージ(参考)

3.1.1.3.2 システム構成要素のソフトウェア構成

「3.1.1.3.1 システム構成要素の種類及び責務」の各システム構成要素を論理ノードとして定義し、その論理ノードを構成するソフトウェア構成を定義する。ソフトウェア構成として示す対象を下表に示す。

表 3.1-6 ソフトウェア構成として示す対象

項番	対象	概要説明
1	ソフトウェア	プログラムプロダクト(ミドルウェア製品、プラットフォーム製品等)だけでなく、OSや業務アプリケーションを含んだ広義のソフトウェア。
2	データベース	データベースにおけるデータ部分の要素。 基本的にDBMSとセットで扱われる。
3	論理ノード	ソフトウェア及びデータベースを組み合わせる利用する際の最小構成要素。 ※物理的に分割して配置するとソフトウェアとしての本来の利用目的を達することのできない最小の単位。 物理的な配置の際の目安とするために、スケールアウトの適性についても示す。
4	システム構成要素	論理ノードから構成される。 なお、以下の組み合わせのようにシステム構成要素間の通信がない場合、物理的に同一の論理ノードとして扱う。 ● プレゼンテーションロジックと業務アプリケーション(ユーザ) ※ただし、静的Webコンテンツ(UI層アプリケーション含む)はWebサーバ上に、動的WebコンテンツはAPサーバ上に配置されることから、それぞれを別の論理ノードとする。 ● ESBと外部システム互換機能

ソフトウェア構成は、アーキテクチャの違いが明らかに異なるサブシステム毎の定型タイプを定める。ソフトウェア構成の単位を下表に示す。

表 3.1-7 ソフトウェア構成として示す単位

項番	ソフトウェア構成の単位	ソフトウェア構成の概要	補足説明
1	階層定型化サブシステム (業務遂行・共通リソース系)	サブシステム毎の業務遂行及び共通リソース管理を行うサブシステム。	—
2	階層定型化サブシステム (共有データベース管理)	事件データ・書類データの管理を行うサブシステム。	—
3	インタフェース定型化サブシステム (外部システム連携)	ESB及び外部連携互換機能を備えるサブシステム。	インタフェース定型化サブシステムであるが、ToBeのアーキテクチャ上、新規に追加されたシステム構成要素の部分であるため、一例としてソフトウェア構成図に示す。
4	インタフェース定型化サブシステム (ビジネスルール管理)	BRMSを備えるサブシステム。	

ソフトウェア構成として示す対象及び単位を踏まえ、ソフトウェア構成図を下図に示す。なお、「ブラウザ」はサブシステム内のシステム構成要素ではないため、ソフトウェア構成図上ではシステム構成要素単体で記載している。

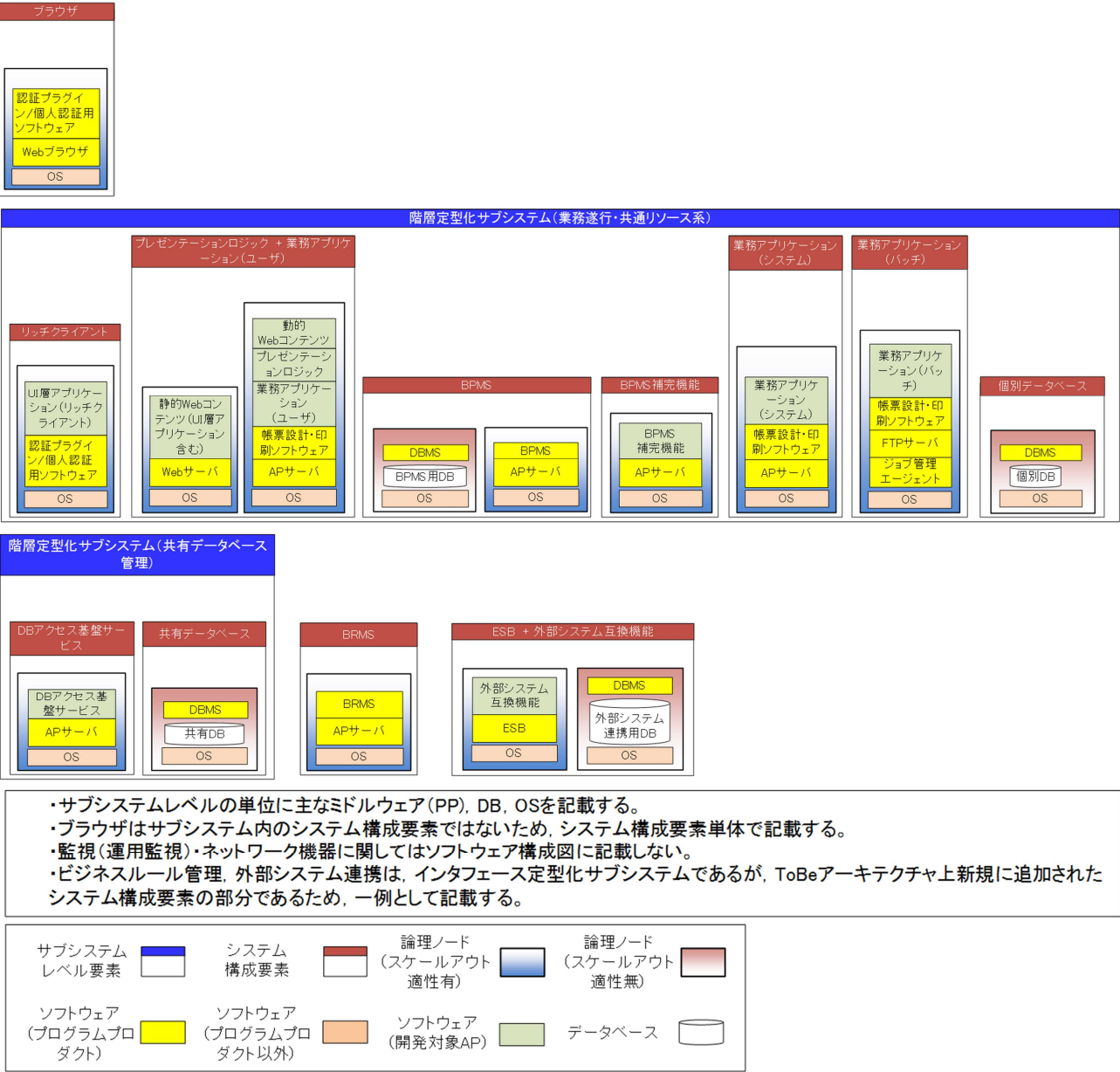


図 3.1-12 ソフトウェア構成図

ソフトウェアのうち、開発対象のアプリケーションを下表に示す。

表 3.1-8 開発対象のアプリケーション

項番	開発対象のアプリケーション	補足説明
1	UI層アプリケーション(リッチクライアント)	開発対象のアプリケーションの内容は、「3.1.1.3.1 システム構成要素の種類及び責務」の各システム構成要素の責務を参照のこと。
2	プレゼンテーションロジック	
3	業務アプリケーション(ユーザ)	
4	BPMS補完機能	
5	業務アプリケーション(システム)	
6	業務アプリケーション(バッチ)	
7	DBアクセス基盤サービス	
8	外部システム互換機能	
9	静的Webコンテンツ (UI層アプリケーション含む) ⁸	Webブラウザ上の静的な画面要素となるファイル、あるいはそこで動作するアプリケーションの実行ファイルといった、リクエストに応じてそのままレスポンスの一部として返されるリソース
10	動的Webコンテンツ ⁸	Webブラウザ上の動的な画面要素や情報を生成するためのファイルといった、リクエストに応じて動作するアプリケーションがレスポンスの一部として動的に情報を生成する際に必要とするリソース

⁸ 配置場所は、実行時に動作する場所ではなく、システム構築を行う際に配置される場所とする。

3.1.1.4 APレイヤ・コンポーネント定義

スコープ： 階層定型化サブシステム及びインタフェース定型化サブシステム

プロセス： 2.4.3

規約1： コンポーネント間のアクセスは、「3.1.1.4.1 APレイヤの定義及びコンポーネントとの関係」に示すアクセスとすること。

規約2： オンライン処理方式のコンポーネント間のアクセスは、「図 3.1-13から図 3.1-17」に示すアクセスパスとすること。

規約3： オンライン処理方式のコンポーネントは、「表 3.1-10～表 3.1-14」に定義される単位で作成し、同表に定義されている責務を全うするようコンポーネントの機能を実現すること。

規約4： バッチ処理方式のコンポーネント間のアクセスは、「図 3.1-18」に示すアクセスパスとすること。

規約5： バッチ処理方式のコンポーネントは、「表 3.1-15」に定義される単位で作成し、同表に定義されている責務を全うするようコンポーネントの機能を実現すること。

規約6： ディレードバッチ処理方式のコンポーネント間のアクセスは、「図 3.1-20」に示すアクセスパスとすること。

規約7： ディレードバッチ処理方式のコンポーネントは、「表 3.1-16」に定義される単位で作成し、同表に定義されている責務を全うするようコンポーネントの機能を実現すること。

オンライン処理方式及びバッチ処理方式のAPレイヤ・コンポーネント定義を示す。帳票出力方式及び連携処理方式のAPレイヤ・コンポーネント定義は、オンライン処理方式、バッチ処理方式及びディレードバッチ処理方式の組み合わせとなる。

3.1.1.4.1 APレイヤの定義及びコンポーネントとの関係

各階層は、責務を下表に示す3つのAPレイヤに分割し定義する。表の記述順序はAPレイヤの上下関係を示しており、上が上位レイヤ、下が下位レイヤを意味している。

コンポーネントは必ずいずれかのAPレイヤに属し、下位レイヤに属するコンポーネントから上位レイヤへ属するコンポーネントに対するアクセスは原則、禁止する(下位が上位に依存しない作りとする)。

表 3.1-9 APレイヤ

項番	APレイヤ名	責務
1	インタフェース層	サービスの機能を外部に提供するためのインタフェースを定義するレイヤ。このレイヤで、通信基盤等、アプリケーションの実装技術を隠蔽する。
2	ロジック層	サービスに求められる機能を実現するレイヤ。
3	インフラストラクチャ層	サービスに求められる機能を実現するために、SOAP等の手段により外部にアクセスする必要がある場合、アクセスするための実装技術を隠蔽するレイヤ。

3.1.1.4.2 オンライン処理方式

(1) 画面系

画面系のオンライン処理では、UI層(ブラウザ又はリッチクライアント)、プレゼンテーション層(プレゼンテーションロジック)、業務アプリケーション層(業務アプリケーション(ユーザ))の3つの階層を構築することでシステム利用者(庁職員等)に機能を提供する。

なお、プレゼンテーション層と業務アプリケーション層の業務アプリケーション(ユーザ)との連携は、レスポンス向上のためメソッド呼び出しとするため、コンポーネント構成は原則、1つにまとめる。

上記を踏まえ、画面系のオンライン処理方式におけるAPレイヤ・コンポーネント構成を下図、下表に示す。

● ブラウザ又はリッチクライアントの場合

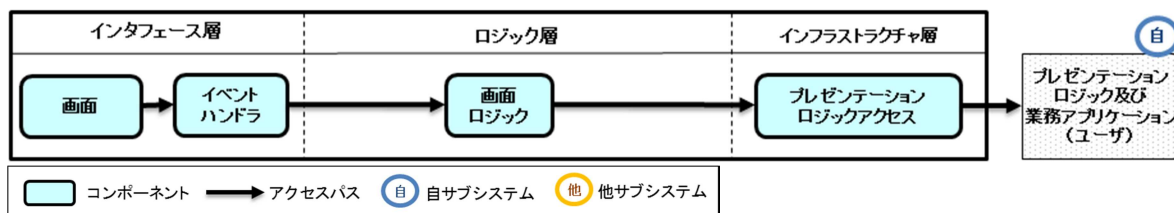


図 3.1-13 APレイヤ・コンポーネント定義(画面系・ブラウザ又はリッチクライアント)⁹

● プレゼンテーションロジック及び業務アプリケーション(ユーザ)の場合

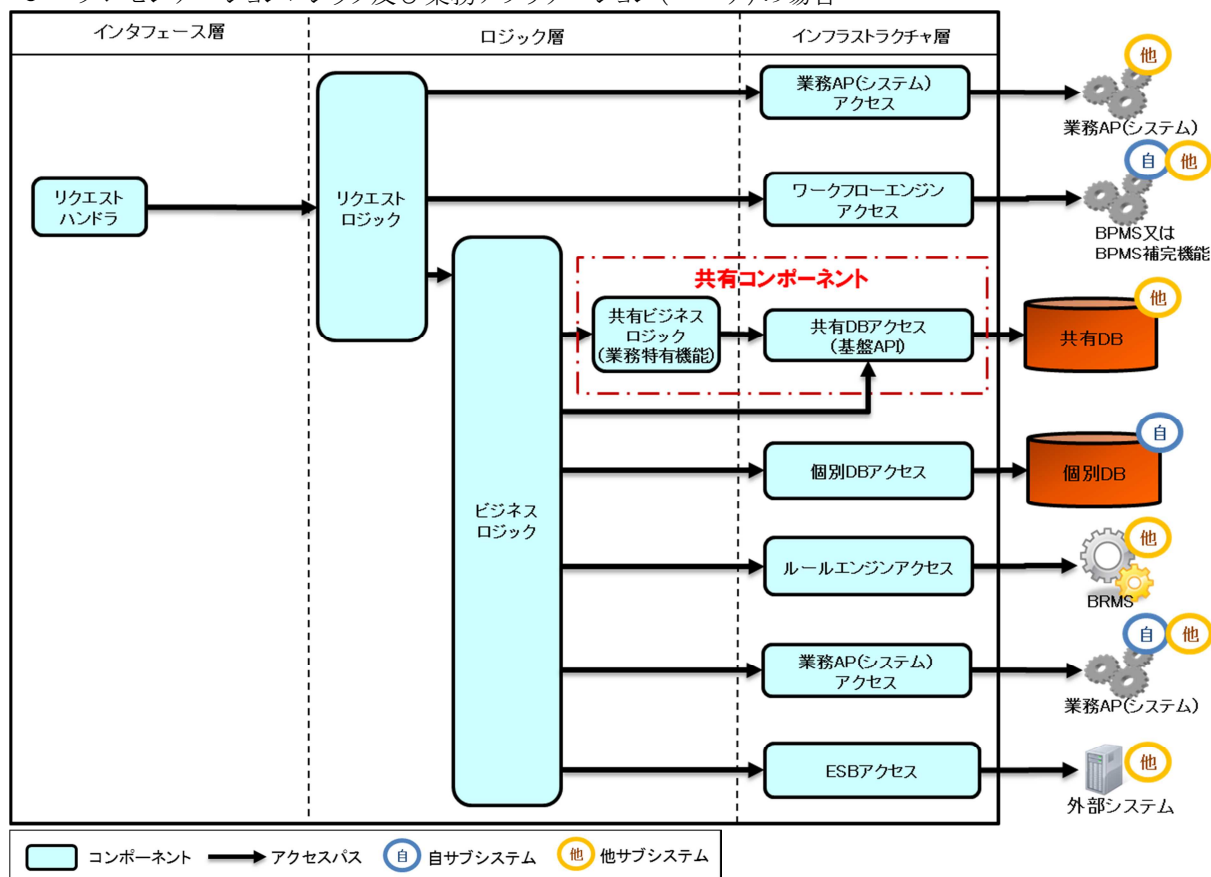


図 3.1-14 APレイヤ・コンポーネント定義(画面系・プレゼンテーションロジック及び業務アプリケーション(ユーザ))

⁹ 各サブシステムのプレゼンテーションロジックから出力されブラウザ上に配信されたアプリケーション(HTMLやJavaScript)の構成を表している。この図におけるブラウザから見た自サブシステムは、配信元のプレゼンテーションロジックが属するサブシステムのことを指す。

表 3.1-10 ブラウザ又はリッチクライアントの責務

項番	コンポーネント	配置するAPレイヤ	責務
1	画面	インタフェース層	システム利用者が操作するUIを提供する。また、入力規制(桁数制限等)等、入力補助も行う。
2	イベントハンドラ	インタフェース層	画面を操作した際に発生するアクションイベントを制御し、画面ロジックの呼び出し等を行う。
3	画面ロジック	ロジック層	クライアント側で可能な入力チェック(必須チェック、フォーマットチェック等)や画面内の制御、プレゼンテーションロジックからの表示制御等を行う。また、サーバでの処理が必要な場合は、サーバへ送信する要求データの組み立ても行う。
4	プレゼンテーションロジックアクセス	インフラストラクチャ層	プレゼンテーションロジックの呼び出しを行う。また、呼び出す際の固有の通信技術や手順を上位層から隠蔽する。

表 3.1-11 プレゼンテーションロジック及び業務AP(ユーザ)の責務

項番	コンポーネント	配置するAPレイヤ	責務
1	リクエストハンドラ	インタフェース層	プレゼンテーションロジック呼び出しのインタフェースの仕様を定め、認証チェックやインタフェースの入力チェック等(必須チェック、フォーマットチェック等)、リクエストロジックやビジネスロジックを呼び出す上での条件を満たすことを保証する。 なお、入力チェックは、クライアントでも実施しているものであってもセキュリティ上、サーバでも再度実施する。 また、クライアント構成に応じた情報の返却(HTML, SOAP電文等)を行う。
2	リクエストロジック	ロジック層	リクエストに応じて下位層のシステム構成要素である業務アプリケーション(ユーザ)や業務アプリケーション(システム)、ワークフローの呼び出しを行う。また、画面に受け渡す値の生成を行う。
3	ビジネスロジック	ロジック層	業務アプリケーション(ユーザ)として提供する主たる機能を実装するコンポーネント。 必要に応じてインフラストラクチャ層のコンポーネントを呼び出し、参照・判定・更新といった処理の組み立てを行う。 また、業務チェック(DB突合チェック等)も行う。
4	共有ビジネスロジック(業務特有機能)	ロジック層	ビジネスロジックのうち、複数の業務アプリケーションで共有し、かつ特許庁業務特有の機能(期間計算等)を提供するコンポーネント。
5	業務AP(システム)アクセス	インフラストラクチャ層	業務アプリケーション(システム)が提供するサービスの呼び出しを行う。アクセスパスの制限事項については「3.2.4.2 サービスインタフェースの粒度(3)アクセスパスの制限事項に関するルール」参照。業務アプリケーション(システム)が提供するサービスを呼び出す際の通信技術や手順を上位層から隠蔽する。
6	ワークフローエンジンアクセス	インフラストラクチャ層	BPMS又はBPMS補完機能の呼び出しを行う。 ロジック層に対して汎用的なAPIを提供するために製品固有のAPIの呼び出しを上位層から隠蔽する。

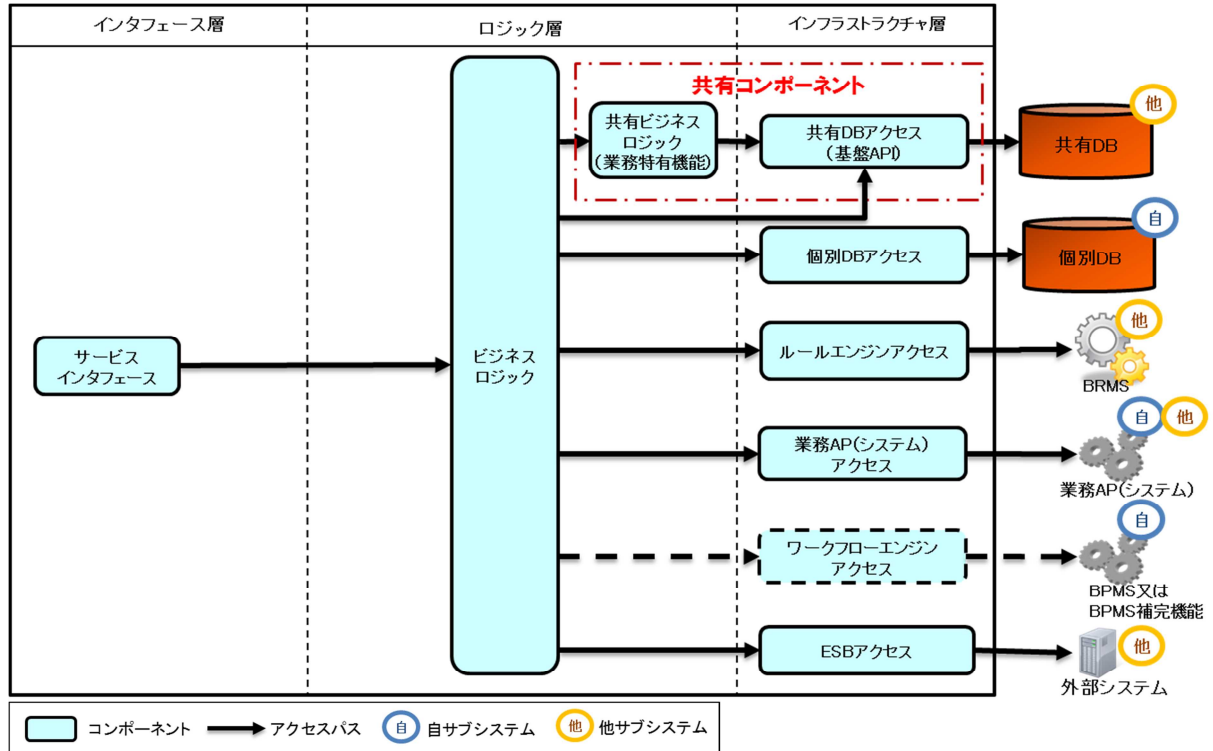
項番	コンポーネント	配置するAPレイヤ	責務
7	共有DBアクセス (基盤API)	インフラストラクチャ層	共有データベースへの参照・更新機能を提供するコンポーネント。本コンポーネントは、特実記録原本管理システムの設計開発基準に記載されている「基盤API」を指す。詳細は『DBアクセス基盤利用者向けガイドライン - 6. 基盤API』を参照のこと。
8	個別DBアクセス	インフラストラクチャ層	個別データベースへの参照・更新機能を提供するコンポーネント。ロジック層に対して汎用的なAPIを提供するために製品固有のデータアクセス技術、永続化手段を上位層から隠蔽する。 なお、横断的に利用する共通的な機能をコンポーネントとして提供し、データベースアクセス処理毎に開発が必要なSQL定義やエンティティクラス等はコンポーネントとは別に管理する。
9	ルールエンジン アクセス	インフラストラクチャ層	BRMSヘルールの実行を要求する。 原則、ビジネスルール管理サブシステムのサービス呼び出しを行うが、高度な性能要件によってはBRMSのAPIライブラリを利用したコンポーネントにする。BRMSの呼び出し方式の詳細は、「3.2.7.1.2 B RMSの呼び出し方式」を参照のこと。 ロジック層に対してBRMSの呼び出し方式を隠蔽する。
10	ESBアクセス	インフラストラクチャ層	外部システムの呼び出しを行う。外部システムを呼び出す際の通信技術や手順を上位層から隠蔽する。

(2) サービス系

サービス系のオンライン処理には、業務アプリケーション(システム)とDBアクセス基盤サービス、BPMS補完機能が提供するサービスの3つがある。

上記を踏まえ、サービス系のオンライン処理方式におけるAPレイヤ・コンポーネント構成を下図、下表に示す。

● 業務アプリケーション(システム)の場合

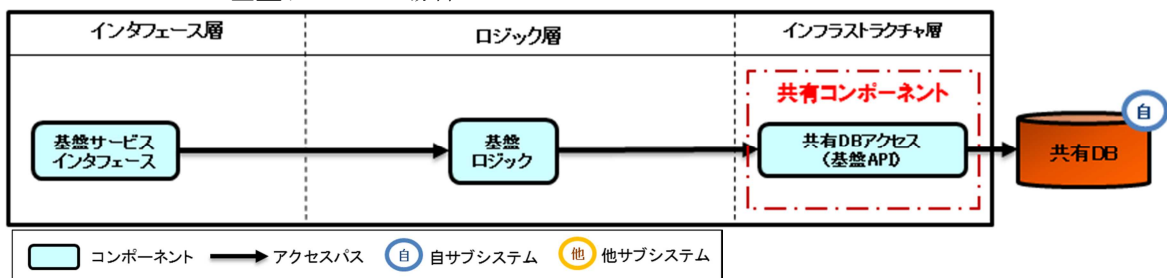


※「ワークフローエンジンアクセス」は、アクセスパスの特例において利用を許容する。

詳細は、「3.1.1.3.1.2 (2)アクセスパスの特例」を参照のこと。

図 3.1-15 APレイヤ・コンポーネント定義(サービス系・業務アプリケーション(システム))

● DBアクセス基盤サービスの場合



● BPMS補完機能の場合

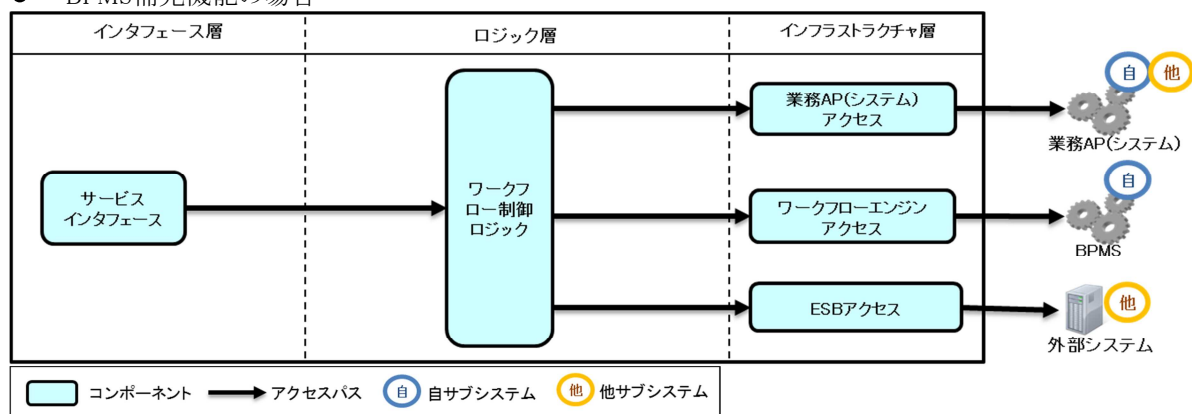


図 3.1-17 APレイヤ・コンポーネント定義(サービス系・BPMS補完機能)

表 3.1-12 業務AP(システム)の責務

項番	コンポーネント	配置するAPレイヤ	責務
1	サービス インタフェース	インタフェース層	サービス起動のインタフェースの仕様を定め、インタフェースの入力チェック等、ビジネスロジックを呼び出す上での前提条件を満たすことを保証する。また、取得したデータの形式を変換して返却する。
2	ビジネスロジック	ロジック層	業務アプリケーション(システム)として提供する主たる機能を実装するコンポーネント。 必要に応じてインフラストラクチャ層のコンポーネントを呼び出し、参照・判定・更新といった処理の組み立てを行う。 また、業務チェック(DB突合チェック等)も行う。
3	共有ビジネスロジック (業務特有機能)	ロジック層	ビジネスロジックのうち、複数の業務アプリケーションで共有し、かつ特許庁業務特有の機能(期間計算等)を提供するコンポーネント。
4	共有DBアクセス (基盤API)	インフラストラクチャ層	共有データベースへの参照・更新機能を提供するコンポーネント。本コンポーネントは、特実記録原本管理システムの設計開発基準に記載されている「基盤API」を指す。詳細は『DBアクセス基盤利用者向けガイドライン - 6. 基盤API』を参照のこと。
5	個別DBアクセス	インフラストラクチャ層	個別データベースへの参照・更新機能を提供するコンポーネント。ロジック層に対して汎用的なAPIを提供するために製品固有のデータアクセス技術、永続化手段を上位層から隠蔽する。 なお、横断的に利用する共通的な機能をコンポーネントとして提供し、データベースアクセス処理毎に開発が必要なSQL定義やエンティティクラス等はコンポーネントとは別に管理する。
6	ルールエンジン アクセス	インフラストラクチャ層	BRMSヘルールの実行を要求する。 原則、ビジネスルール管理サブシステムのサービス呼び出しを行うが、高度な性能要件によってはBRMSのAPIライブラリを利用したコンポーネントにする。BRMSの呼び出し方式の詳細は、「3.2.7.1.2 BRMSの呼び出し方式」を参照のこと。 ロジック層に対してBRMSの呼び出し方式を隠蔽する。
7	業務AP(システム) アクセス	インフラストラクチャ層	業務アプリケーション(システム)が提供するサービスの呼び出しを行う。アクセスパスの制限事項については「3.2.4.2 サービスインタフェースの粒度 (3)アクセスパスの制限事項に関するルール」参照。業務アプリケーション(システム)が提供するサービスを呼び出す際の通信技術や手順を上位層から隠蔽する。
8	ワークフローエンジン アクセス	インフラストラクチャ層	BPMS又はBPMS補完機能の呼び出しを行う。 ロジック層に対して汎用的なAPIを提供するために製品固有のAPIの呼び出しを上位層から隠蔽する。 なお、特異な条件時のアクセスとして許容される。詳細は、「3.1.1.3.1.2 (2)アクセスパスの特例」を参照のこと。
9	ESBアクセス	インフラストラクチャ層	外部システムの呼び出しを行う。外部システムを呼び出す際の通信技術や手順を上位層から隠蔽する。

表 3.1-13 DBアクセス基盤サービスの責務

項番	コンポーネント	配置するAPレイヤ	責務
1	基盤サービス インタフェース	インタフェース層	サービス起動のインタフェースの仕様を定め、インタフェースの入力チェック等、基盤ロジックを呼び出す上での条件を満たすことを保証する。 また、取得したデータの形式を変換して返却する。
2	基盤ロジック	ロジック層	共有データベースからデータを取得し、返却用のデータを組み立てる。
3	共有DBアクセス (基盤API)	インフラストラクチャ層	共有データベースへの参照・更新機能を提供するコンポーネント。本コンポーネントは、特実記録原本管理システムの設計開発基準に記載されている「基盤API」を指す。詳細は『DBアクセス基盤利用者向けガイドライン - 6. 基盤API』を参照のこと。

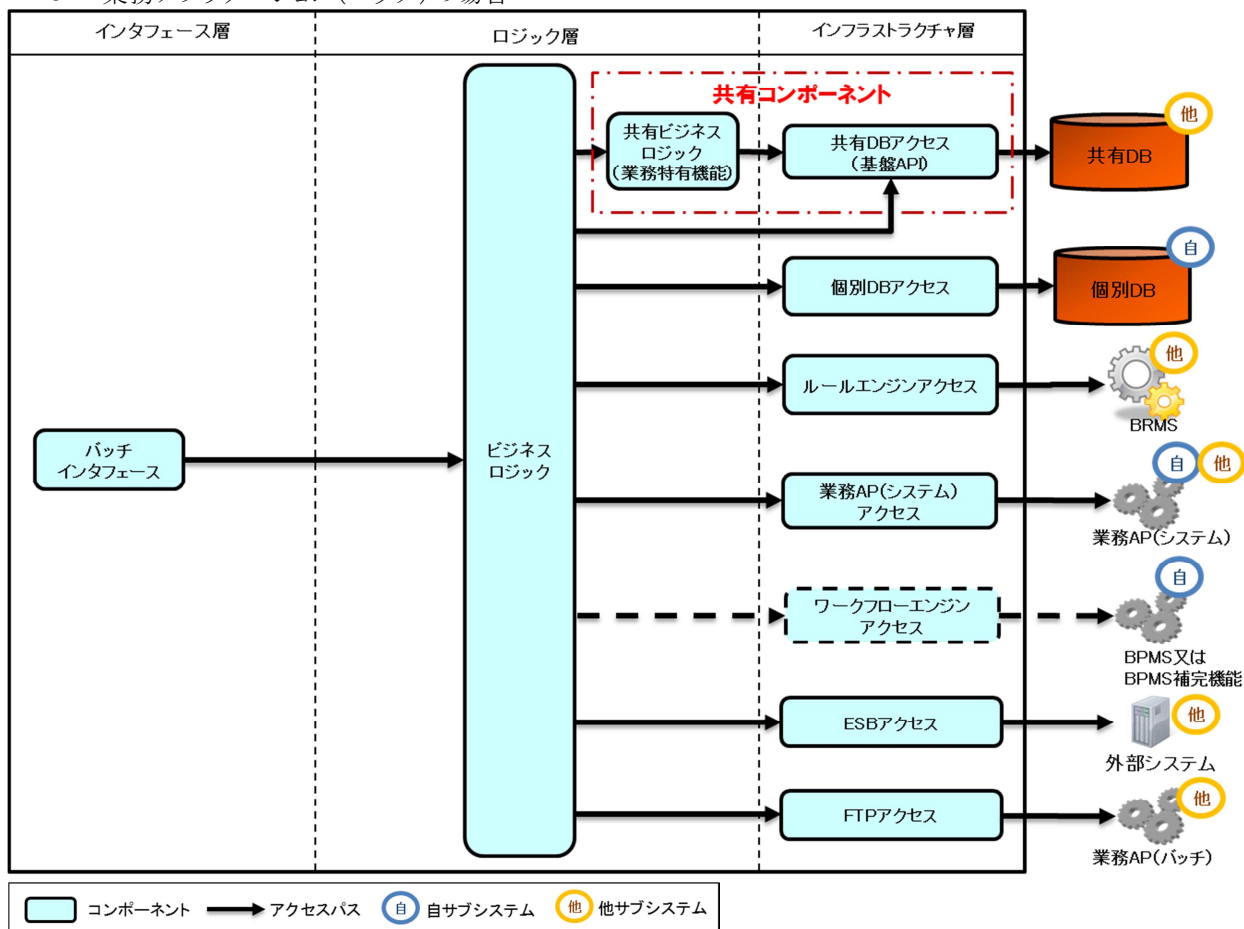
表 3.1-14 BPMS補完機能の責務

項番	コンポーネント	配置するAPレイヤ	責務
1	サービス インタフェース	インタフェース層	サービス起動のインタフェースの仕様を定め、インタフェースの入力チェック等、ビジネスロジックを呼び出す上での条件を満たすことを保証する。 また、取得したデータの形式を変換して返却する。
2	ワークフロー制御 ロジック	ロジック層	BPMS補完機能として提供する主たる機能を実装するコンポーネント。 必要に応じてインフラストラクチャ層のコンポーネントを呼び出し、参照・判定・更新といった処理の組み立てを行う。
3	ワークフローエンジン アクセス	インフラストラクチャ層	BPMSの呼び出しを行う。 ロジック層に対して汎用的なAPIを提供するために製品固有のAPIの呼び出しを上位層から隠蔽する。
4	業務AP(システム) アクセス	インフラストラクチャ層	業務アプリケーション(システム)が提供するサービスの呼び出しを行う。アクセスパスの制限事項については「3.2.4.2 サービスインタフェースの粒度 (3)アクセスパスの制限事項に関するルール」参照。業務アプリケーション(システム)が提供するサービスを呼び出す際の通信技術や手順を上位層から隠蔽する。
5	ESBアクセス	インフラストラクチャ層	外部システムの呼び出しを行う。外部システムを呼び出す際の通信技術や手順を上位層から隠蔽する。

3.1.1.4.3 バッチ処理方式

バッチ処理は、業務アプリケーション(バッチ)が機能を提供する。
バッチ処理方式におけるAPレイヤ・コンポーネント構成を下図、下表に示す。

● 業務アプリケーション(バッチ)の場合



※「ワークフローエンジンアクセス」は、アクセスパスの特例において利用を許容する。
詳細は、「3.1.1.3.1.2 (2)アクセスパスの特例」を参照のこと。

図 3.1-18 APレイヤ・コンポーネント定義(バッチ)

表 3.1-15 コンポーネントの責務(バッチ)

項番	コンポーネント	配置するAPレイヤ	責務
1	バッチインタフェース	インタフェース層	インタフェースのパラメータチェック等、ビジネスロジックを呼び出す上での条件を満たすことを保証する。 また、実行結果を返却する。
2	ビジネスロジック	ロジック層	業務アプリケーション(バッチ)として提供する主たる機能を実装するコンポーネント。 必要に応じてインフラストラクチャ層のコンポーネントを呼び出し、参照・判定・更新といった処理の組み立てを行う。
3	共有ビジネスロジック (業務特有機能)	ロジック層	ビジネスロジックのうち、複数の業務アプリケーションで共有し、かつ特許庁業務特有の機能(期間計算等)を提供するコンポーネント。
4	共有DBアクセス (基盤API)	インフラストラクチャ層	共有データベースへの参照・更新機能を提供するコンポーネント。本コンポーネントは、特実記録原本管理システムの設計開発基準に記載されている「基盤API」を指す。詳細は『DBアクセス基盤利用者向けガイドライン - 6. 基盤API』を参照のこと。
5	個別DBアクセス	インフラストラクチャ層	個別データベースへの参照・更新機能を提供するコンポーネント。ロジック層に対して汎用的なAPIを提供するために製品固有のデータアクセス技術、永続化手段を上位層から隠蔽する。 なお、横断的に利用する共通的な機能をコンポーネントとして提供し、データベースアクセス処理毎に開発が必要なSQL定義やエンティティクラス等はコンポーネントとは別に管理する。
6	ルールエンジン アクセス	インフラストラクチャ層	BRMSヘルールの実行を要求する。 原則、ビジネスルール管理サブシステムのサービス呼び出しを行うが、高度な性能要件によってはBRMSのAPIライブラリを利用したコンポーネントにする。BRMSの呼び出し方式の詳細は、「3.2.7.1.2 B RMSの呼び出し方式」を参照のこと。 ロジック層に対してBRMSの呼び出し方式を隠蔽する。
7	業務AP(システム) アクセス	インフラストラクチャ層	業務アプリケーション(システム)が提供するサービスの呼び出しを行う。アクセスパスの制限事項については「3.2.4.2 サービスインタフェースの粒度 (3)アクセスパスの制限事項に関するルール」参照。業務アプリケーション(システム)が提供するサービスを呼び出す際の通信技術や手順を上位層から隠蔽する。
8	ワークフローエンジン アクセス	インフラストラクチャ層	BPMS又はBPMS補完機能の呼び出しを行う。 ロジック層に対して汎用的なAPIを提供するために製品固有のAPIの呼び出しを上位層から隠蔽する。 なお、特異な条件時のアクセスとして許容される。詳細は、「3.1.1.3.1.2 (2)アクセスパスの特例」を参照のこと。
9	ESBアクセス	インフラストラクチャ層	外部システムの呼び出しを行う。外部システムを呼び出す際の通信技術や手順を上位層から隠蔽する。

項番	コンポーネント	配置するAPレイヤ	責務
10	FTPクライアント	インフラストラクチャ層	他サブシステムの業務アプリケーション(バッチ)に対し、ファイルでデータを連携するためにFTP通信を行う。FTP呼び出しを行う際の通信技術や手順を上位層から隠蔽する。

3.1.1.4.4 ディレードバッチ処理方式

ディレードバッチ処理は、画面からの処理要求に対して要求を受け付けた旨の応答を即時に返却した後、非同期に実際の処理を行う場合に用いる方式である。非同期処理の実現方式としては、ユーザからの処理要求を登録しユーザへ応答を返却するまでを行う部分と、ディレード処理要求管理情報を定期的に確認してバッチとして実行する部分とで構成する。前者の部分は「3.1.1.4.2 オンライン処理方式」で実現するものであり、本項にて特筆すべき点はない。ディレードバッチ処理方式としては、後者のディレード処理要求管理情報の確認から処理の実行までを行う部分を対象とする。ディレードバッチ処理は、業務アプリケーション(バッチ)が機能を提供する。

以下に概念図と処理概要を示す。

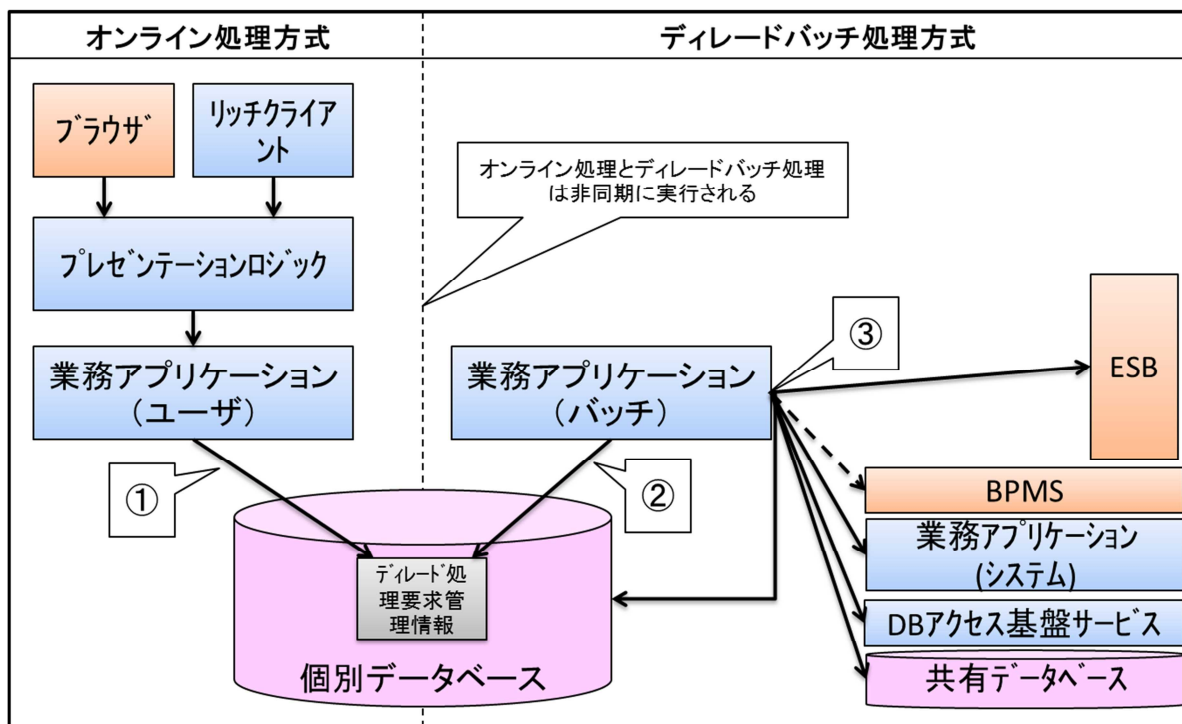


図 3.1-19 非同期処理とディレードバッチ処理方式の概念図

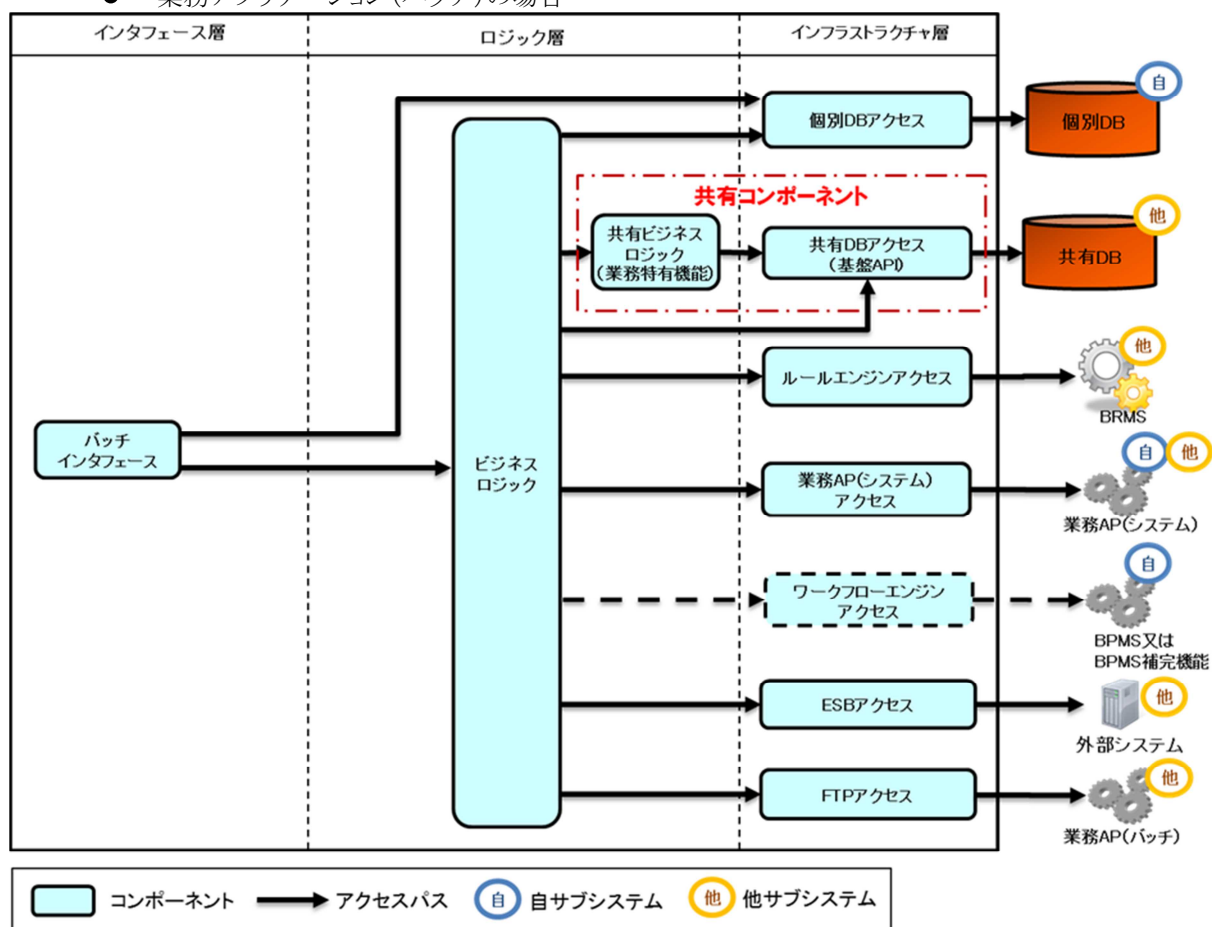
- ① 画面からの処理要求を「ディレード処理要求管理情報」として個別データベースに登録し、実際の処理の完了を待たずに要求を受け付けた旨の応答を画面へ返却するオンライン処理方式で実現する。それ以外にも、後続②以降のバッチ処理に対してインプットとなるデータを生成する等、要件に応じた処理を実行した上でディレード処理要求の登録を行ってもよい。
- ② ディレード処理の業務アプリケーション(バッチ)はあらかじめ起動済みで常駐プロセスとして動作し、定期的に「ディレード処理要求管理情報」を確認し処理要求を待ち受ける。登録された要求を確認すると要求内容に応じて③のバッチ処理を開始する。①の処理と②以降の処理は非同期に実行される。
- ③ 登録された処理要求の内容に応じたビジネスロジックを実行する。バッチ処理方式のビジネスロジック同様に、必要に応じて他のシステム構成要素にアクセスする。^{10,11}

¹⁰ 業務アプリケーション(バッチ)からBPMSへのアクセスは、アクセスパスの特例において利用を許容する。

¹¹ ディレードバッチ処理を契機にビジネスプロセスを開始する方法として、画面からのディレード処理要求を行った後にプロセストークンを進める方法として、プレゼンテーションロジックからBPMSへアクセスする方法(方法1)と、業務アプリケーション(バッチ)からBPMSへアクセスする方法(方法2)がある。どちらの方法を採用するかは、バッチ処理とBPMSによるビジネスプロセス中のタスクの関係性(同期する必要があるか、又は非同期でよい)、及び同期する場合には方法1ではBPMN上に待機するフローを記載する必要があるが、それがビジネスプロセスの可視化の点で妥当であるかを考慮する。

ディレードバッチ処理方式におけるAPレイヤ・コンポーネント構成を下図、下表に示す。バッチインタフェースコンポーネントの責務とアクセスパス以外は、「3.1.1.4.3 バッチ処理方式」と同様である。

● 業務アプリケーション(バッチ)の場合



※「ワークフローエンジンアクセス」は、アクセスパスの特例において利用を許容する。
詳細は、「3.1.1.3.1.2 (2)アクセスパスの特例」を参照のこと。

図 3.1-20 APレイヤ・コンポーネント定義(ディレードバッチ)

表 3.1-16 コンポーネントの責務(ディレードバッチ)

項番	コンポーネント	配置するAPレイヤ	責務
1	バッチインタフェース	インタフェース層	個別データベースの「ディレード処理要求管理情報」を定期的に確認し、処理要求の登録を確認すると要求内容に応じたビジネスロジックを実行する。

※上記以外のコンポーネントについては「表 3.1-15 コンポーネントの責務(バッチ)」と同様である。

3.1.2 システムの動的な振る舞い

本章に示す技術方式のルール(設計指針, 規約, 推奨及び特例)は, 以下の章に示す。

- 3.1.2.1 オンライン処理方式
- 3.1.2.2 バッチ処理方式
- 3.1.2.3 デイレードバッチ処理方式
- 3.1.2.4 帳票処理方式
- 3.1.2.5 連携処理方式

「3.1.1.3.1.1 システム構成要素の処理方式による分類」において示した処理方式の各々について, 具体的な処理シーケンスを示す。

3.1.2.1 オンライン処理方式

スコープ： 階層定型化サブシステム及びインタフェース定型化サブシステム

プロセス： 2.4.3

指針1： オンライン処理方式の処理シーケンスは、「3.1.2.1.2 処理シーケンス」に示す例を参考に設計すること。

指針2： オンライン処理方式のデータ保護方式は、「3.1.2.1.3 データの保護方式」に従って設計すること。

指針3： オンライン処理方式のエラー処理方式は、「3.1.2.1.4 エラー処理方式」に従って設計すること。

指針4： オンライン処理方式の業務継続方式は、「3.1.2.1.5 業務継続方式」に従って設計すること。

オンライン処理方式は、クライアント等からの処理要求に対するサーバ処理が同期的に全て完遂する処理を基本とする。

ただし、Ajaxを利用し非同期にデータ授受をすることを許容する。

以下、オンライン処理方式について処理方式パターンや処理シーケンスを示す。

3.1.2.1.1 処理方式パターン

オンライン処理方式における処理方式のパターンを下表に示す。

表 3.1-17 オンライン処理方式の処理方式パターン

項番	分類	説明
1	画面系	ユーザ操作により、画面からネットワークを介してサーバに処理要求を行い、応答を返却する方式。
2	サービス系	BPMS等からの処理要求に対し、サービスとして応答を返却する方式。

3.1.2.1.2 処理シーケンス

3.1.2.1.2.1 画面系

画面系のオンライン処理方式における処理シーケンスのサンプルを下図、下表に示す。

例：システム利用者がブラウザ上で入力したデータを、チェック後、データベースに登録する場合

- ブラウザ

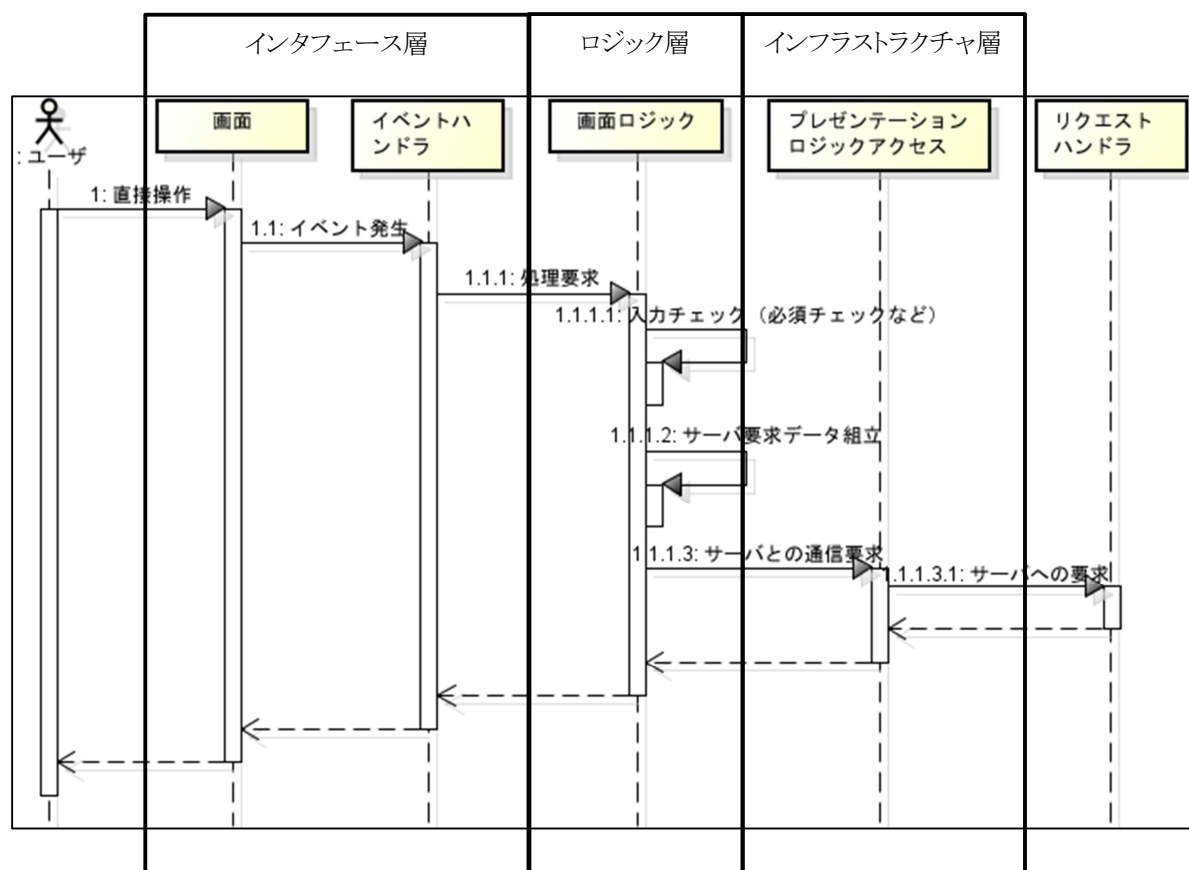


図 3.1-21 ブラウザの処理シーケンスの例

表 3.1-18 ブラウザの処理概要

項番	処理	処理概要
1	直接操作	ユーザが画面上で、入力や選択、リンクやボタンのクリック等を行う。(ブラウザのブックマークの選択やURLの直接指定も含む)
1.1	イベント発生	ユーザの操作によりイベントが発生する。
1.1.1	処理要求	入力チェックやサーバへの通信を行う。 (Ajax等によるJavaScriptからの通信を含む)
1.1.1.1	入力チェック (必須チェック等)	クライアントでチェック可能な必須チェックや関連チェック等を行う。
1.1.1.2	サーバ要求データ組立	サーバに送信する要求データを組み立てる。
1.1.1.3	サーバとの通信要求	インフラストラクチャ層にサーバとの通信を依頼する。
1.1.1.3.1	サーバへの要求	画面からの要求をサーバに送信する。

● プレゼンテーションロジック

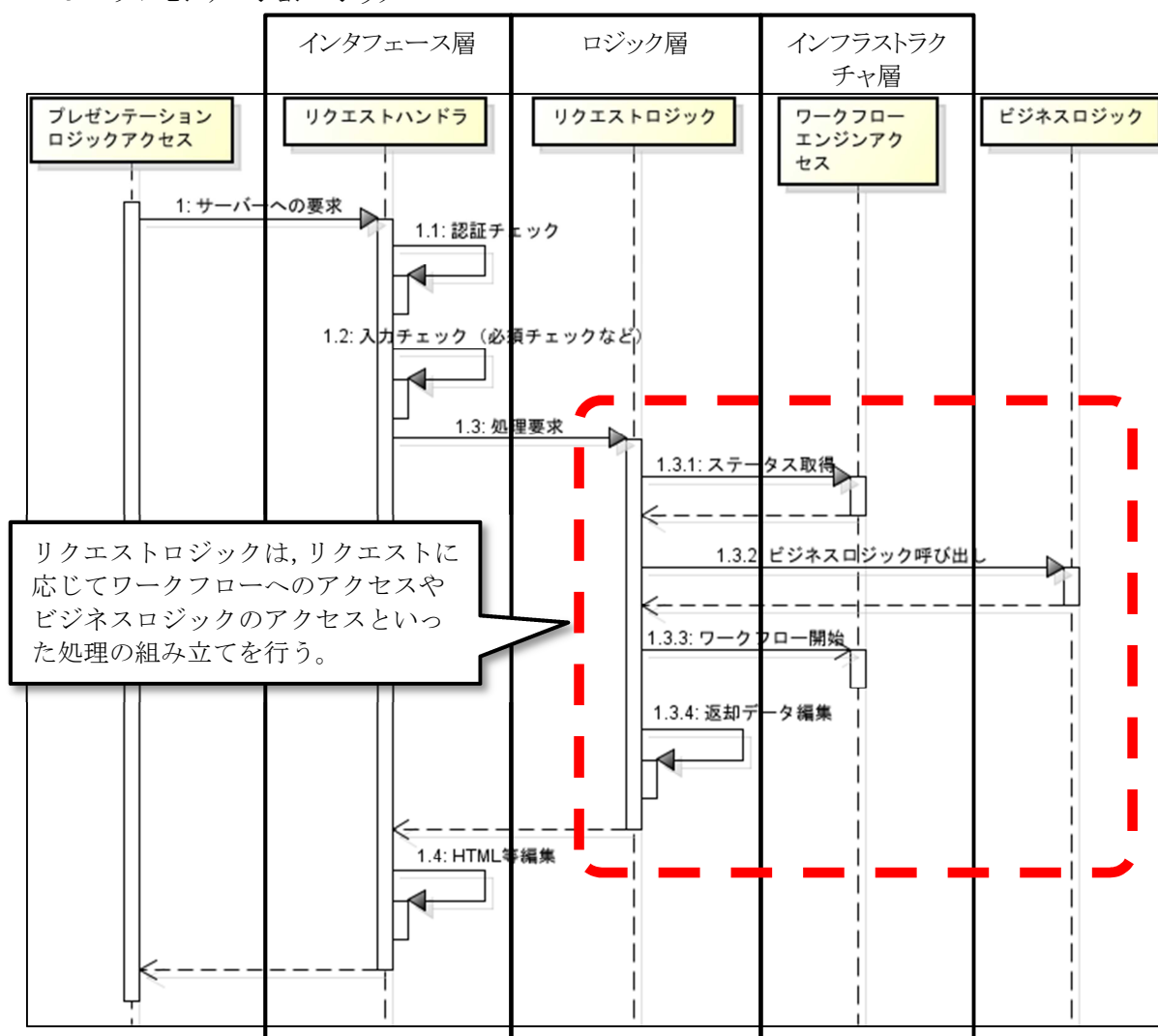


図 3.1-22 プレゼンテーションロジックの処理シーケンスの例

表 3.1-19 プレゼンテーションロジックの処理概要

項番	処理	処理概要
1	サーバへの要求	画面からの要求をサーバに送信する。
1.1	認証チェック	サーバでの処理の最初で、ユーザ情報を利用し、認証を行う。この際、取得した認可情報で必要に応じ、以降の処理の認可チェックをする。
1.2	入力チェック(必須チェック等)	入力チェック(必須チェックや関連チェック等)を行う。UI層で実施しているものであっても、セキュリティ上、サーバでも再度チェックする。
1.3	処理要求	ワークフローへのアクセスや、ビジネスロジックの実行等の処理要求をする。
1.3.1	ステータス取得	必要に応じて、ワークフローにアクセスし、ステータスの取得を行う。
1.3.2	ビジネスロジック呼び出し	必要に応じて、ビジネスロジックを呼び出し、業務処理を行う。
1.3.3	ワークフロー開始	必要に応じて、ワークフローの開始を行う。
1.3.4	返却データ編集	クライアントに返却するデータを編集する。
1.4	HTML等編集	ブラウザに返却するHTML等(Ajax等の場合はJSONやXML等)を編集する。

- 業務アプリケーション(ユーザ)
- ※プレゼンテーション層からメソッド呼び出しされるため、インタフェース層はない。

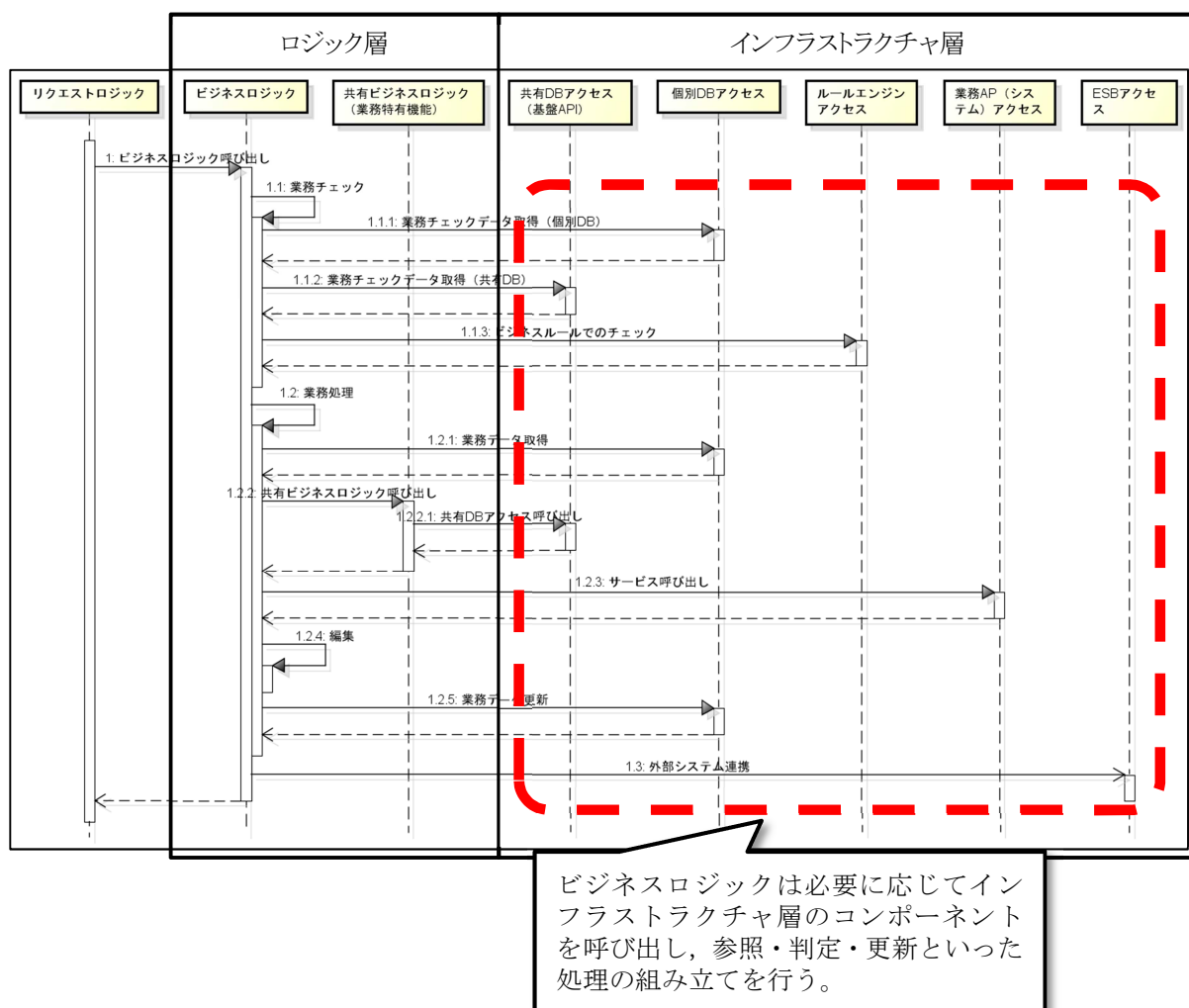


図 3.1-23 業務AP(ユーザ) 処理シーケンスの例

表 3.1-20 業務AP(ユーザ)の処理概要

項番	処理	処理概要
1	ビジネスロジック呼び出し	ビジネスロジックを呼び出し、業務処理を行う。
1.1	業務チェック	データベースやBRMSを利用し、データの存在チェックやビジネスルールを利用したチェック等、業務観点でのチェックを行う。
1.1.1	業務チェックデータ取得 (個別DB)	必要に応じて、個別データベースからチェックに必要なデータを取得する。また、存在チェック等も行う。
1.1.2	業務チェックデータ取得 (共有DB)	必要に応じて、共有データベースからチェックに必要なデータを取得する。また、存在チェック等も行う。
1.1.3	ビジネスルールでのチェック	必要に応じて、BRMSにアクセスし、ビジネスルールを実行する。
1.2	業務処理	業務処理を行う。
1.2.1	業務データ取得	必要に応じて、個別データベースから業務処理に必要なデータを取得する。
1.2.2	共有ビジネスロジック呼び出し	必要に応じて、共有ビジネスロジックを呼び出し、共通的な業務処理を行う。
1.2.2.1	共有DBアクセス呼び出し	必要に応じて、共有データベースにアクセスし、データの取得、更新等を行う。
1.2.3	サービス呼び出し	必要に応じて、業務アプリケーション(システム)が提供するサービスを呼び出す。アクセスパスの制限事項については「3.2.4.2 サービスインタフェースの粒度 (3)アクセスパスの制限事項に関するルール」参照。
1.2.4	編集	処理結果やデータを編集する。
1.2.5	業務データ更新	個別データベースにアクセスし、データの登録や更新等を行う。
1.3	外部システム連携	必要に応じて、外部システムと連携する。

3.1.2.1.2.2 サービス系

サービス系のオンライン処理方式における処理シーケンスのサンプルを下図、下表に示す。

例：システム利用者がブラウザ上で入力したデータを、チェック後、データベースに登録する場合

● 業務アプリケーション(システム)

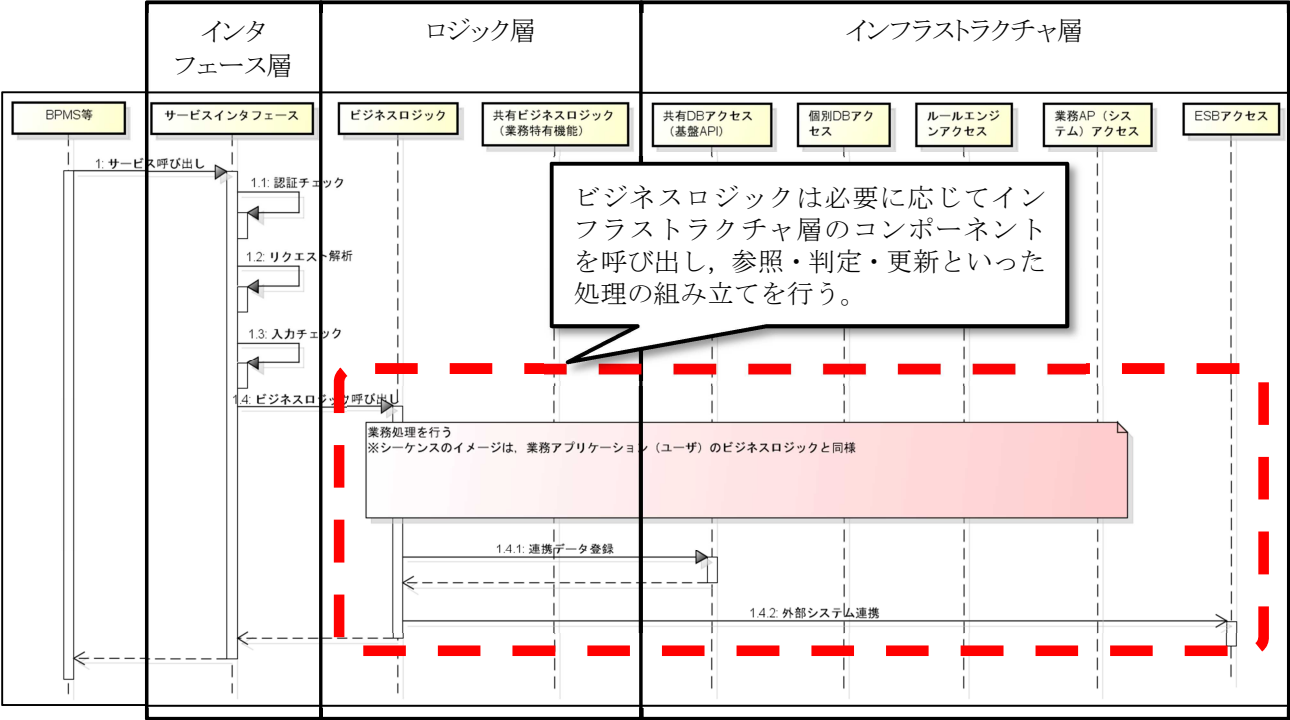


表 3.1-21 業務AP(システム)の処理概要

項番	処理	処理概要
1	サービス呼び出し	BPMS等からサービス呼び出す。
1.1	認証チェック	サーバでの処理の最初で、ユーザ情報を利用し、認証を行う。この際、取得した認可情報で必要に応じ、以降の処理の認可チェックをする。
1.2	リクエスト解析	リクエスト(SOAP電文等)を解析する。
1.3	入力チェック	パラメータの入力チェック(必須チェックや相関チェック等)を行う。
1.4	ビジネスロジック呼び出し	ビジネスロジックを呼び出し、業務処理を行う。
1.4.1	連携データ登録	他に連携するデータを共有データベースに登録する。
1.4.2	外部システム連携	外部システムと連携する。

例：DBアクセス基盤サービス経由で共有データベースにアクセスする場合

- DBアクセス基盤サービス

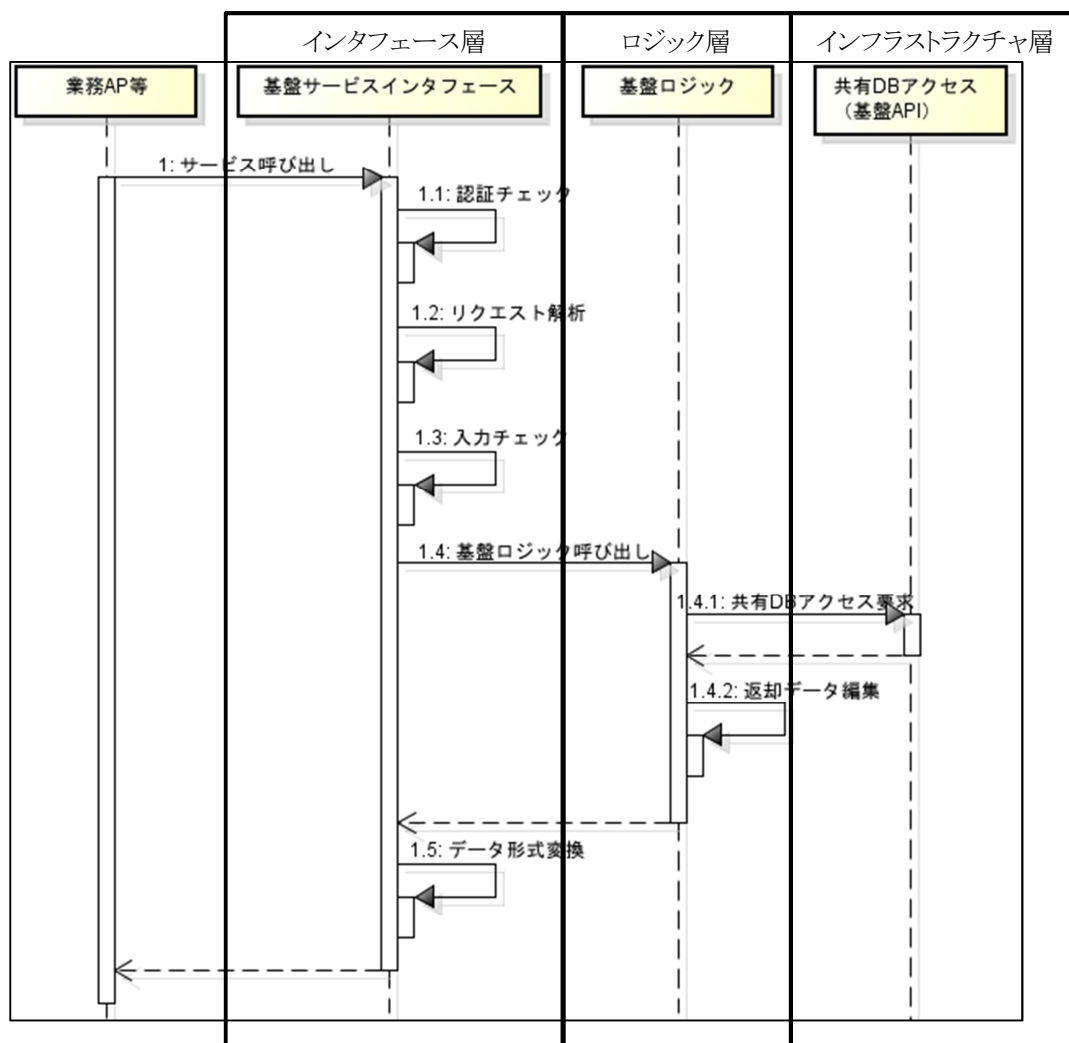


図 3.1-25 DBアクセス基盤サービスの処理シーケンスの例

表 3.1-22 DBアクセス基盤サービスの処理概要

項番	処理	処理概要
1	サービス呼び出し	BPMS等からサービスを呼び出す。
1.1	認証チェック	サーバでの処理の最初で、ユーザ情報を利用し認証を行う。
1.2	リクエスト解析	リクエスト(SOAP電文等)を解析する。
1.3	入力チェック	パラメータの入力チェック(必須チェックや関連チェック等)を行う。
1.4	ビジネスロジック呼び出し	ビジネスロジックを呼び出し、業務処理を行う。
1.4.1	共有DBアクセス要求	共有データベースへのアクセスを要求する。
1.4.2	返却データ編集	処理結果やデータを編集する。
1.5	データ形式変換	要求元に合わせて、データ形式を変換する。

例： 業務AP(システム)サービスの結果からワークフローの待機を解除する

● BPMS補完機能

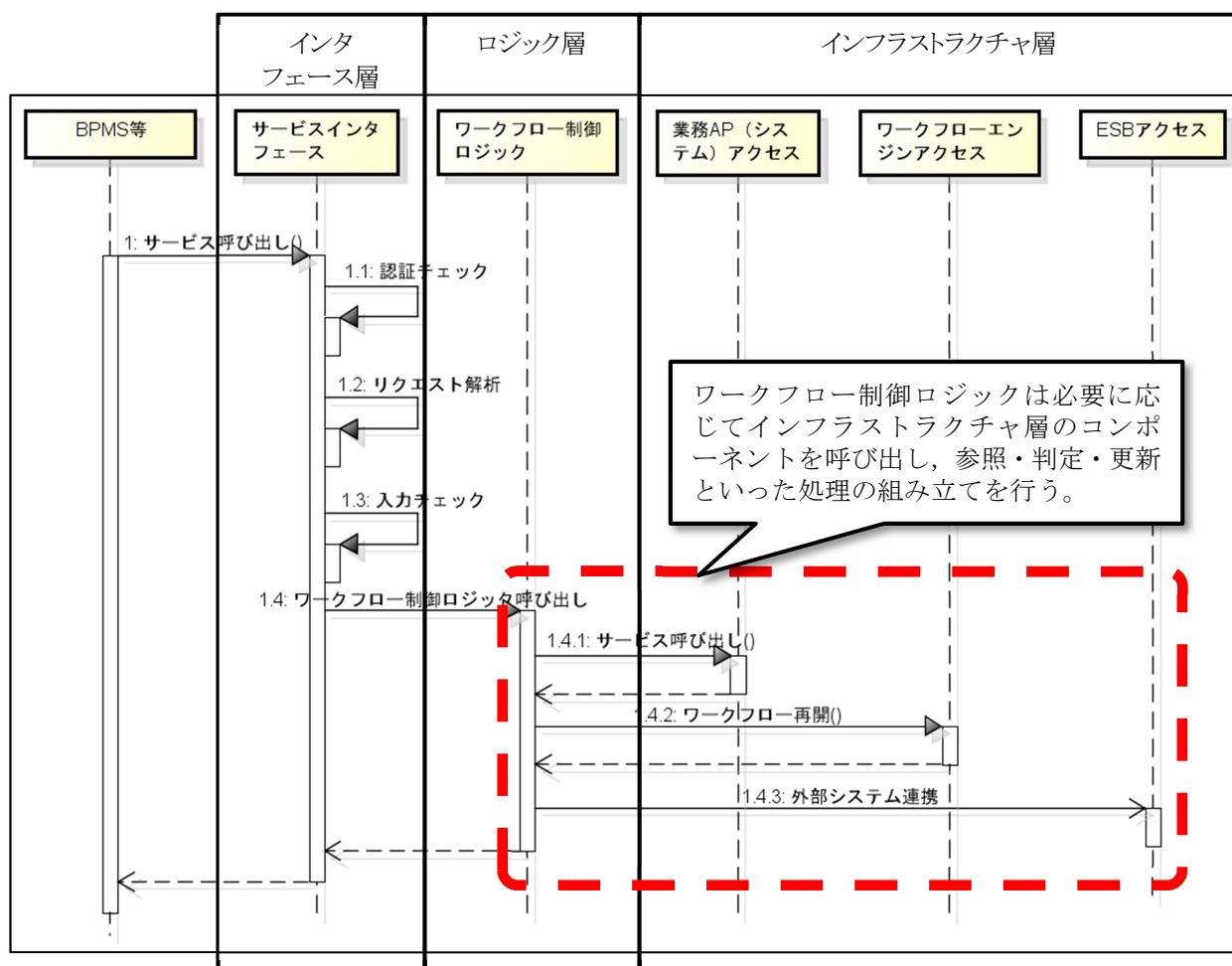


図 3.1-26 BPMS補完機能の処理シーケンスの例

表 3.1-23 BPMS補完機能の処理概要

項番	処理	処理概要
1	サービス呼び出し	BPMS等からサービスを呼び出す。
1.1	認証チェック	サーバでの処理の最初で、ユーザ情報を利用し、認証を行う。この際、取得した認可情報で必要に応じ、以降の処理の認可チェックをする。
1.2	リクエスト解析	リクエスト(SOAP電文等)を解析する。
1.3	入力チェック	パラメータの入力チェック(必須チェックや関連チェック等)を行う。
1.4	ワークフロー制御ロジック呼び出し	ワークフロー制御ロジックを呼び出し、BPMSを補完する処理を行う。
1.4.1	サービス呼び出し	業務アプリケーション(システム)が提供するサービスを呼び出し、ワークフローの再開可否を確認する。 なお、データベースへのアクセスが必要な場合、BPMS補完機能から直接アクセスするのではなく、業務アプリケーション(システム)を介してデータベースにアクセスする。アクセスパスの制限事項については「3.2.4.2 サービスインタフェースの粒度 (3)アクセスパスの制限事項に関するルール」参照。
1.4.2	ワークフロー再開	BPMSにアクセスし、ワークフローを再開する。
1.4.3	外部システム連携	必要に応じ、外部システムと連携する。

3.1.2.1.3 データの保護方式

ビジネスプロセスのアクティビティは業務として意味のある最小の単位となるため、データ整合性をアクティビティの単位で確保する必要がある。

短時間に単一データベースの中でACID特性を確保する場合は、RDBMSの「システムトランザクション」機能を使用する(ACID特性が求められる単位にトランザクションを制御し、エラー発生時はロールバックする)。

リクエストをまたがる等、長時間に同一のデータに対して同時に複数のアクセスがある場合や複数データベースをまたがるアクセスがある場合、データ整合性を確保するためにアプリケーションにおいて下記の排他制御方式を使用する。なお、複数データベースをまたがるアクセスがある場合、2フェーズコミット方式によりデータ整合性を確保する方法もあるが、ベンダ・サポートを受けるにはトランザクションに参加する製品が全て同一ベンダである必要がある等の制約があるため、原則、使用しない。

表 3.1-24 排他制御方式

項番	排他制御方式	処理概要	選定指針
1	悲観的排他制御	あるデータ更新作業の開始～終了時点まで、他からの更新を防ぐ排他制御	先に処理を開始した方の更新を優先したい場合
2	楽観的排他制御	データ更新時に競合をチェックし、更新してよいかを判断する排他制御	処理の開始が先か後かに関わらず、更新が早い方を優先したい場合

排他制御を行っても、複数データベースを更新する場合において、システムエラーによるデータ不整合が発生する可能性がある。

データ不整合が発生した場合は、ログを利用してデータの整合性を取るが、データの整合性を担保するには、アプリケーションが出力するログだけでなく、SQLログやBPMS等、プログラムプロダクトのログも必要である。

よって、プログラムプロダクトからもデータ整合性を担保するために必要なログを出力するよう設定する。

また、処理シーケンスにおける「ワークフローエンジンアクセス」(BPMSのステータス管理機能によるアクティビティのステータスの更新)、「共有DBアクセス(基盤API)」(共有データベースのデータの更新)、「個別DBアクセス」(個別データベースのデータの更新)の整合性を確保するための対策を考慮すること。前述のように、2フェーズコミットを使用しない方針により、これらの整合性についてもアプリケーションで取る必要がある。BPMSのアクティビティからサービスを呼び出す処理方式パターンにおける例を以下に示す。

- サービス内での「共有DBアクセス(基盤API)」と「個別DBアクセス」の整合性を取るために、更新の順序を考慮し、個々のデータベースに対してコミットを行うようにする。エラーが発生したときは、後続のアクティビティを進ませないようにエラーを返すようにする。
- 「ワークフローエンジンアクセス」とサービス内での「共有DBアクセス(基盤API)」又は「個別DBアクセス」の整合性を取るために、BPMSでのアクティビティから同期的にサービスを呼び出し、サービスの結果応答を得るまではプロセストークンを次のノードに進ませないようにする。これら一連の処理はBPMSの機能によって行われるが、ここでの不整合発生の可能性はBPMSのシステム障害が由来のものでありサービス内の処理によって回復することは出来ないため、不整合が発生した場合はBPMSのステータスを変更する等の運用対処を行う必要がある。

3.1.2.1.4 エラー処理方式

オンラインで発生するエラーの種類を下表に示す。

表 3.1-25 エラーの種類

項番	エラーの種類	定義	例
1	業務エラー	業務設計(仕様)の中で想定しているエラー	● 入力チェックエラー ● 認証・認可エラー 等
2	システムエラー	業務設計(仕様)の中で想定していないエラー	● ハードウェアの故障に起因するエラー ● ネットワークの故障に起因するエラー 等

上記のエラーが発生した場合、以下の対応をするものとする。

表 3.1-26 エラー発生時の対応

項番	対応	目的
1	エラーの詳細をログ等に出力する。	データの保護や、障害解析性を向上する。
2	データを整合性のある状態に戻す。	データの整合性を確保する。 詳細は、「3.1.2.1.3 データの保護方式」を参照のこと。
3	エラーの発生内容をクライアントに応答する。	システム利用者や運用者にエラーを知らせる。

なお、システムエラーといった、業務設計(仕様)の中で想定されていないエラーを扱う処理は、通常の業務処理のプログラム中に混在して記述すると処理の流れの見通しが悪くなり、保守性が低下する。このような問題を回避するため、エラー処理を集約的に定義・実行できる仕組み(集約エラーハンドラ)を用いることとする。

また、各モジュールの責任範囲外の入力値にデータ形式の違い等があった場合、例外を発生させ、集約エラーハンドラで集約的にエラー処理を行うものとする。

3.1.2.1.5 業務継続方式

オンライン処理方式において業務を継続するための方法は、基本的に再実行である。詳細を下表に示す。

表 3.1-27 業務継続方式

項番	分類	エラーの種類	業務継続方式
1	画面系	業務エラー	画面に表示されるメッセージを参考に入力値を変更する等の方法で問題を解決し、再実行する。
2		システムエラー	
3	サービス系	業務エラー	① 業務アプリケーション等が出力するログを利用し、原因を調査、対処する。 ② データ不整合が発生している場合は、データを整合性のある状態に戻す。 ③ 再実行する。 ● 画面系の場合、画面からシステム利用者が再度操作することで再実行する。 ● サービス系の場合、サービスの呼び出し元 (BPMS等) からサービスを再実行する。
4		システムエラー	

3.1.2.2 バッチ処理方式

スコープ： 階層定型化サブシステム及びインタフェース定型化サブシステム

プロセス： 2.4.3

指針1： バッチ処理方式の処理シーケンスは、「3.1.2.2.2 処理シーケンス」に示す例を参考に設計すること。

指針2： バッチ処理方式のデータ保護方式は、「3.1.2.2.3 データの保護方式」に従って設計すること。

指針3： バッチ処理方式のエラー処理方式は、「3.1.2.2.4 エラー処理方式」に従って設計すること。

指針4： バッチ処理方式の業務継続方式は、「3.1.2.2.5 業務継続方式」に従って設計すること。

以下、バッチ処理方式について処理方式パターンや処理シーケンスを示す。

3.1.2.2.1 処理方式パターン

バッチ処理方式における処理方式のパターンを下表に示す。

表 3.1-28 バッチ処理方式の処理方式パターン

項番	分類	説明
1	複数件一括処理	多量のデータを取りまとめ、一定期間あるいはデータ量毎に一度にまとめて処理する方式。
2	単件一括処理	単件の処理を一度にまとめて処理する方式。 単件処理は要件により逐次又は並行で実行する。

3.1.2.2.2 処理シーケンス

3.1.2.2.2.1 複数件一括処理

複数件一括処理の処理シーケンスのサンプルを下図、下表に示す。

例：単件処理で登録された業務データを一括で処理して外部システムに連携する場合

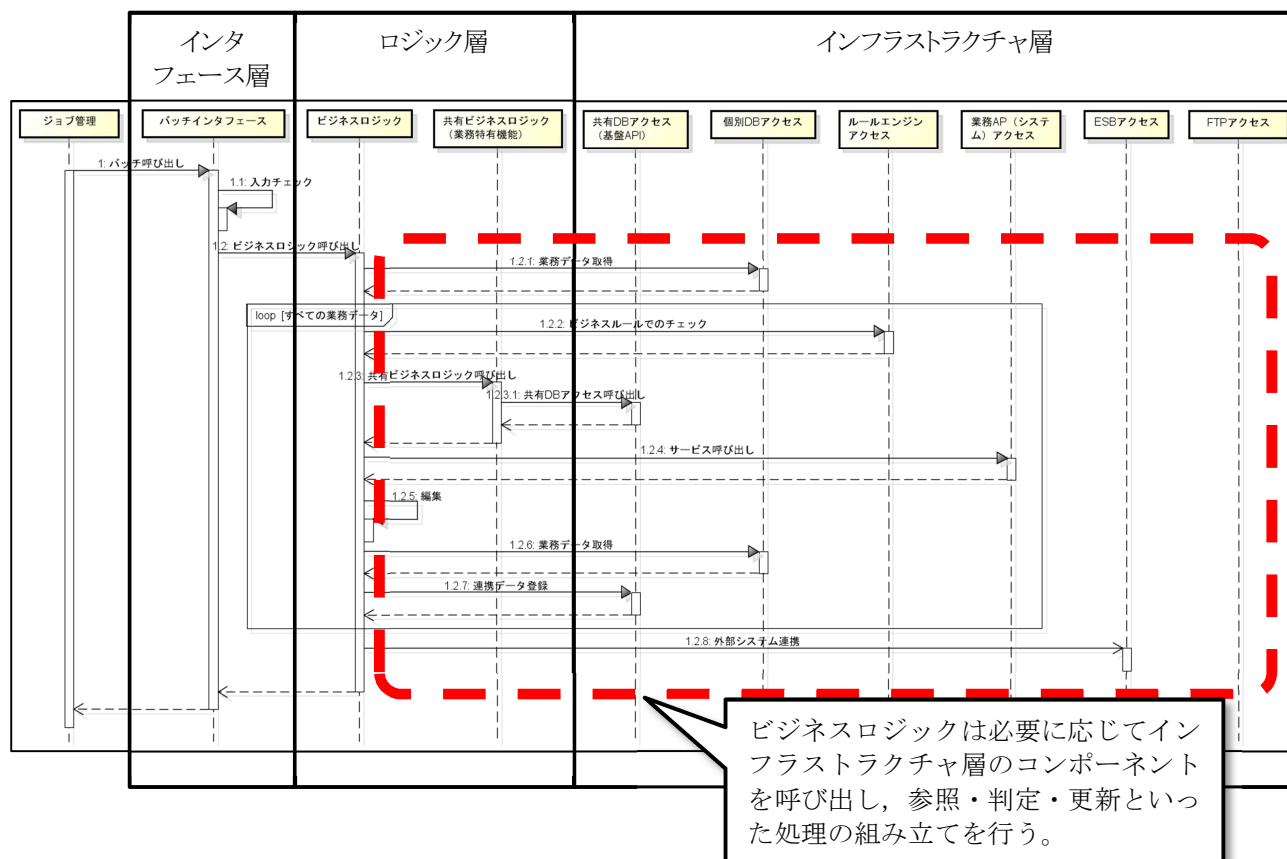


図 3.1-27 複数件一括処理の処理シーケンスの例

表 3.1-29 複数件一括処理の処理概要

項番	処理	処理概要
1	バッチ呼び出し	ジョブ管理からバッチを呼び出す。
1.1	入力チェック	パラメータの入力チェック(必須チェックや関連チェック等)を行う。
1.2	ビジネスロジック呼び出し	ビジネスロジックを呼び出し、業務処理を行う。
1.2.1	業務データ取得	個別データベースから単件処理で登録されたデータを取得する。
1.2.2	ビジネスルールでのチェック	必要に応じて、BRMSにアクセスし、ビジネスルールを実行する。
1.2.3	共有ビジネスロジック呼び出し	必要に応じて、共有ビジネスロジックを呼び出し、共通的な業務処理を行う。
1.2.3.1	共有DBアクセス呼び出し	必要に応じて、共有データベースにアクセスし、データの取得、更新等を行う。
1.2.4	サービス呼び出し	必要に応じて、業務アプリケーション(システム)が提供するサービスを呼び出す。アクセスパスの制限事項については「3.2.4.2 サービスインタフェースの粒度 (3)アクセスパスの制限事項に関するルール」参照。
1.2.5	編集	処理結果やデータを編集する。
1.2.6	業務データ更新	個別データベースにアクセスし、データの登録や更新等を行う。
1.2.7	連携データ登録	他に連携するデータを共有データベースに登録する。
1.2.8	外部システム連携	外部システムと連携する。

3.1.2.2.2 単件一括処理

単件一括処理の処理シーケンスのサンプルを下図、下表に示す。

例： 多量の単件処理を多重処理で出力する場合

- メインとなるバッチ

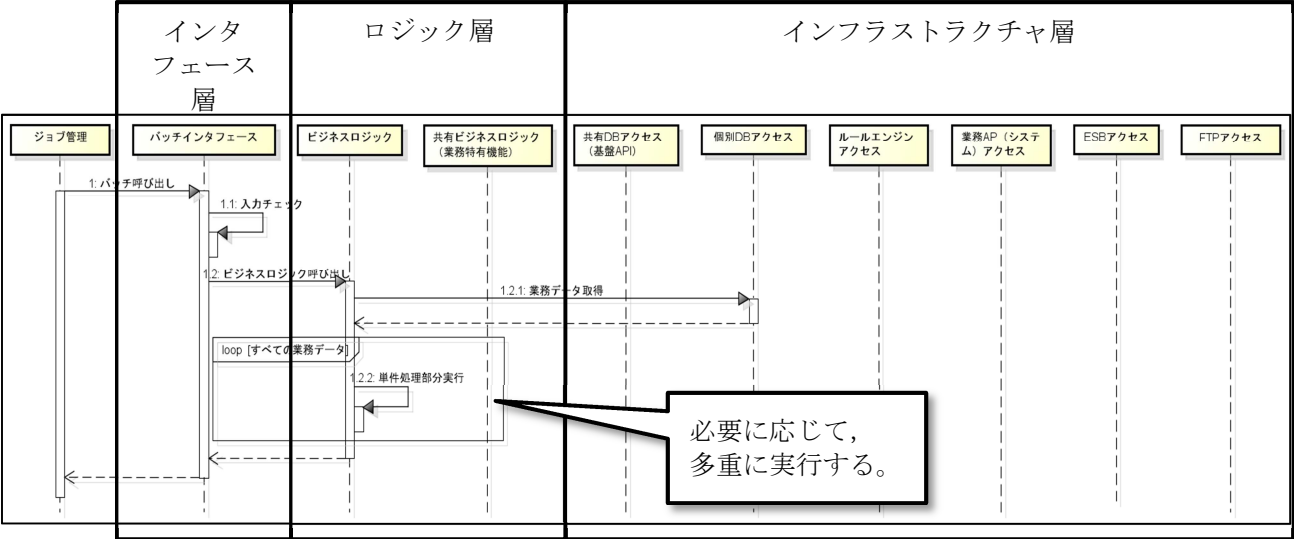


図 3.1-28 単件一括処理(メイン)の処理シーケンスの例

表 3.1-30 単件一括処理(メイン)の処理概要

項番	処理	処理概要
1	バッチ呼び出し	ジョブ管理からバッチを呼び出す。
1.1	入力チェック	パラメータの入力チェック(必須チェックや関連チェック等)を行う。
1.2	ビジネスロジック呼び出し	ビジネスロジックを呼び出し、業務処理を行う。
1.2.1	業務データ取得	個別データベースから出力する帳票の一覧を取得する。
1.2.2	単件処理部分実行	単件処理部分を実行する。

● 単件処理部分

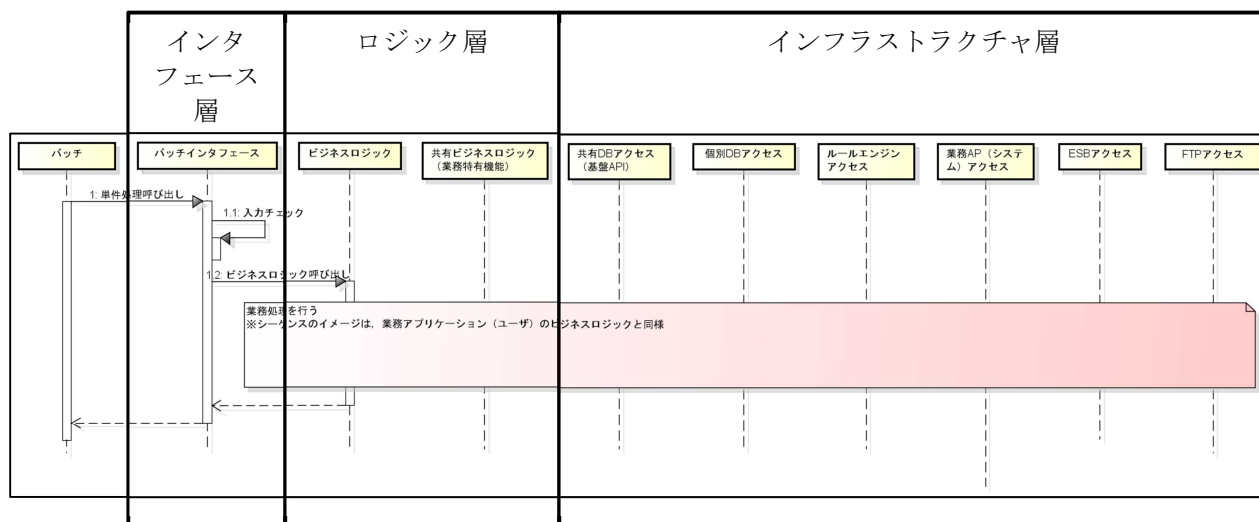


図 3.1-29 単件一括処理(単件処理部分)の処理シーケンスの例

表 3.1-31 単件一括処理(単件処理部分)の処理概要

項番	処理	処理概要
1	単件処理呼び出し	バッチから単件処理部分呼び出す。
1.1	入力チェック	パラメータの入力チェック(必須チェックや関連チェック等)を行う。
1.2	ビジネスロジック呼び出し	ビジネスロジック呼び出し, 業務処理を行う。

3.1.2.2.3 データの保護方式

バッチ処理方式のデータの保護方式は、オンライン処理方式のデータの保護方式と同様である。
詳細は「3.1.2.1.3 データの保護方式」を参照のこと。

3.1.2.2.4 エラー処理方式

バッチ処理方式で発生するエラーの種類を下表に示す。

表 3.1-32 エラーの種類

項番	エラーの種類	定義	例
1	業務エラー	業務設計(仕様)の中で想定しているエラー	● 入力ファイルの形式エラー ● データなしエラー 等
2	システムエラー	業務設計(仕様)の中で想定していないエラー	● ハードウェアの故障に起因するエラー ● ネットワークの故障に起因するエラー 等

上記のエラーが発生した場合、以下の対応をするものとする。

表 3.1-33 エラー発生時の対応

項番	対応	目的
1	エラーの詳細をログ等に出力する	データの保護や、障害解析性を向上する。
2	データを整合性のある状態に戻す	データの整合性を確保する。 詳細は、「3.1.2.2.3 データの保護方式」を参照のこと。
3	エラーの発生内容をクライアントに応答する	システム利用者や運用者にエラーを知らせる。

なお、システムエラーといった、業務設計(仕様)の中で想定されていないエラーを扱う処理は、通常の業務処理のプログラム中に混在して記述すると処理の流れの見通しが悪くなり、保守性が低下する。このような問題を回避するため、エラー処理を集約的に定義・実行できる仕組み(集約エラーハンドラ)を用いることとする。

また、各モジュールの責任範囲外の入力値にデータ形式の違い等があった場合、例外が発生させ、集約エラーハンドラで集約的にエラー処理を行うものとする。

3.1.2.2.5 業務継続方式

業務継続方式として、エラー発生時の再実行を、業務アプリケーション(バッチ)の中で行う場合と、業務アプリケーション(バッチ)の処理を一旦終了したのちジョブ管理システム等によって行う場合がある。

それぞれの業務継続方式と選定の指針を下表に示す。

表 3.1-34 業務継続方式

項番	再実行の契機	業務継続方式	処理概要	選定指針
1	ジョブ管理システム等	リラン	エラー発生時には、全てのトランザクションをロールバックする。リカバリ時は最初から処理を再実行する。	● 処理対象データを一括でコミットしないと業務に影響が出る場合 ● バッチ処理時間が短い場合(エラー発生時、最初から再実行をするだけの運用時間が取れる場合)
2		リスタート	エラー発生時には、コミット済みのリスタートポイントまでトランザクションをロールバックする。リカバリ時はリスタートポイントから処理を再実行する。	● 処理対象データを部分的にコミットしても業務に影響が出ない場合 ● バッチ処理時間が長い場合(エラー発生時、最初から再実行をするだけの運用時間が取れない場合) ● 処理を中断／再開する機能が必要な場合
3	業務アプリケーション(バッチ)	リトライ	エラー(例えばタイムアウト等)発生時、エラーとなった処理を自動的に再実行する。	● 再実行すれば処理が成功する可能性が十分にあり、かつリトライの時間がバッチの処理時間として許容される場合

3.1.2.3 ディレードバッチ処理方式

スコープ： 階層定型化サブシステム及びびインタフェース定型化サブシステム

プロセス： 2.4.3

指針1： ディレードバッチ処理方式の処理シーケンスは、「3.1.2.3.2 処理シーケンス」に示す例を参考に設計すること。

指針2： ディレードバッチ処理方式のデータ保護方式は、「3.1.2.3.3 データの保護方式」に従って設計すること。

指針3： ディレードバッチ処理方式のエラー処理方式は、「3.1.2.3.4 エラー処理方式」に従って設計すること。

指針4： ディレードバッチ処理方式の業務継続方式は、「3.1.2.3.5 業務継続方式」に従って設計すること。

以下、ディレードバッチ処理方式について処理方式パターンや処理シーケンスを示す。

3.1.2.3.1 処理方式パターン

ディレードバッチ処理方式における処理方式のパターンを下表に示す。

表 3.1-35 ディレードバッチ処理方式の処理方式パターン

項番	分類	説明
1	画面系	ユーザ操作により、画面からサーバに処理要求を行い、要求受領の旨の応答を即時に返却した後、要求された処理を行う方式。

3.1.2.3.2 処理シーケンス

3.1.2.3.2.1 画面系

ディレードバッチの処理シーケンスのサンプルを下図、下表に示す。

例： ディレード要求制御情報に登録された処理要求を取得し、ビジネスロジックを実行する場合

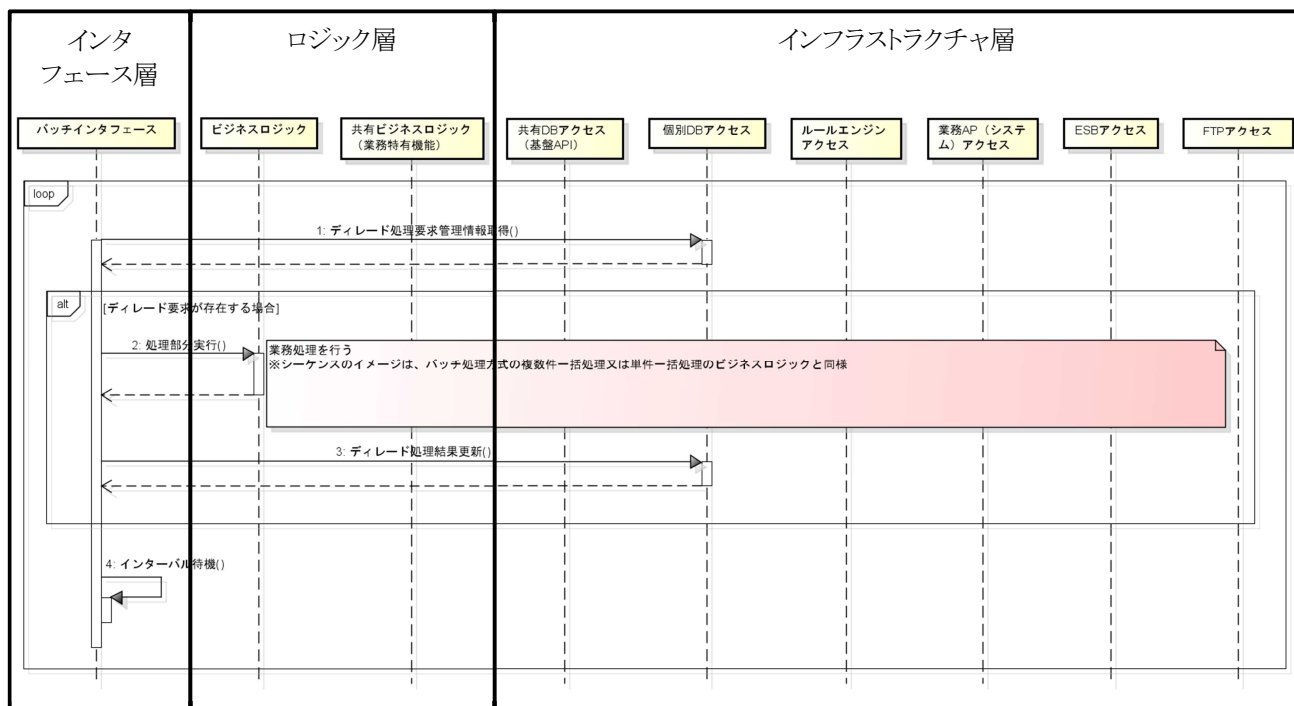


図 3.1-30 ディレードバッチの処理シーケンスの例

表 3.1-36 デイレードバッチの処理概要

項番	処理	処理概要
1	ディレード処理要求管理情報取得	個別データベースに格納されている「ディレード処理要求管理情報」を取得し、ディレード処理要求の有無の確認を行う。
2	処理部分実行	ディレード処理要求の種類に応じたビジネスロジックを実行する。要件により、バッチ処理方式パターンの複数件一括処理又は単件一括処理と同様のシーケンス処理を行う。
3	ディレード処理結果更新	ビジネスロジックの処理結果を「ディレード処理要求管理情報」に反映させる。
4	インターバル待機	「ディレード処理要求管理情報」を確認する次のタイミングまで一定時間待機する。

3.1.2.3.3 データの保護方式

ディレイドバッチ処理方式のデータの保護方式は、オンライン処理方式のデータの保護方式と同様である。

詳細は「3.1.2.1.3 データの保護方式」を参照のこと。

3.1.2.3.4 エラー処理方式

ディレードバッチ処理で発生するエラーの種類を下表に示す。

表 3.1-37 エラーの種類

項番	エラーの種類	定義	例
1	業務エラー	業務設計(仕様)の中で想定しているエラー	● 入力ファイルの形式エラー ● データなしエラー 等
2	システムエラー	業務設計(仕様)の中で想定していないエラー	● ハードウェアの故障に起因するエラー ● ネットワークの故障に起因するエラー 等

上記のエラーが発生した場合、以下の対応をするものとする。

表 3.1-38 エラー発生時の対応

項番	対応	目的
1	エラーの詳細をログ等に出力する	データの保護や、障害解析性を向上する。
2	データを整合性のある状態に戻す	データの整合性を確保する。 詳細は、「3.1.2.2.3 データの保護方式」を参照のこと。
3	エラーの発生を「ディレード処理要求管理情報」のディレードバッチ処理結果に反映する	システム利用者や運用者にエラーを知らせる。

なお、システムエラーといった、業務設計(仕様)の中で想定されていないエラーを扱う処理は、通常の業務処理のプログラム中に混在して記述すると処理の流れの見通しが悪くなり、保守性が低下する。このような問題を回避するため、エラー処理を集約的に定義・実行できる仕組み(集約エラーハンドラ)を用いることとする。

また、各モジュールの責任範囲外の入力値にデータ形式の違い等があった場合、例外が発生させ、集約エラーハンドラで集約的にエラー処理を行うものとする。

3.1.2.3.5 業務継続方式

ディレードバッチの業務継続方式として、エラー発生箇所がビジネスロジックの場合とバッチインタフェース(ディレード処理要求の制御)の場合で異なる。

それぞれの業務継続方式と選定の指針を下表に示す。

表 3.1-39 業務継続方式

項番	エラー発生箇所	業務継続方式	処理概要	選定指針
1	ビジネスロジック	リラン	エラー発生時には、全てのトランザクションをロールバックする。リカバリ時は最初から処理を再実行する。	● 処理対象データを一括でコミットしないと業務に影響が出る場合 ● バッチ処理時間が短い場合(エラー発生時、最初から再実行をするだけの運用時間が取れる場合)
2		リスタート	エラー発生時には、コミット済みのリスタートポイントまでトランザクションをロールバックする。リカバリ時はリスタートポイントから処理を再実行する。	● 処理対象データを部分的にコミットしても業務に影響が出ない場合 ● バッチ処理時間が長い場合(エラー発生時、最初から再実行をするだけの運用時間が取れない場合) ● 処理を中断／再開する機能が必要な場合
3		リトライ	エラー(例えばタイムアウト等)発生時、エラーとなった処理を自動的に再実行する。	● 再実行すれば処理が成功する可能性が十分にあり、かつリトライの時間がバッチの処理時間として許容される場合
4	バッチインタフェース(ディレード処理要求の制御)	再起動	エラー発生時には、実行中のトランザクションをロールバックする。リカバリ時は最初から処理を再実行する。	—

ビジネスロジックでエラーが発生¹²した場合は、エラー情報を「ディレード処理要求管理情報」のディレードバッチ処理結果として更新する。リラン及びリスタートの契機は「ディレード処理要求管理情報」の処理結果や処理状況といった情報を更新することにより、ディレード処理要求の制御部分に再実行対象として認識させることで実現する。なお、「ディレード処理要求管理情報」の閲覧や再実行のための更新といった機能は、業務要件や運用要件によりユーザ向けの画面として実装したり運用ツールとして実装することを想定している。

「ディレード処理要求管理情報」に対し、上記のビジネスロジックのエラー情報更新や、制御のための参照・更新がエラーとなった場合は、項番4の対応となる。

ディレードバッチ処理の業務継続方式を実現するために「ディレード処理要求管理情報」には以下に示す管理情報に相当する項目を保持する。

表 3.1-40 ディレード処理要求管理情報のイメージ

項番	管理項目	説明
1	識別情報	ディレード処理要求を識別するための情報を管理する。
2	ディレード処理種別	実行するビジネスロジックとの対応付けのための情報を管理する。
3	入力情報	ディレード処理のビジネスロジックにインプットとして渡すデータやファイルパス等の情報を管理する。
4	ディレード処理結果	ディレード処理の処理結果を管理する。
5	ディレード処理状況	ディレード処理の状況(未実行, 実行中, 完了 等)を管理する。
6	上記以外の管理項目	処理開始／終了時刻等, 必要に応じ管理する。

¹² リトライによる業務継続を試みた処理については、リトライによってエラーを復旧できなかった場合

3.1.2.4 帳票出力方式

スコープ： 階層定型化サブシステム

プロセス： 2.4.3

指針1： 帳票出力方式の処理シーケンスは、「3.1.2.4.2 処理シーケンス」に示す例を参考に設計すること。

指針2： 帳票出力方式の業務継続方式は、「3.1.2.4.3 業務継続方式」に従って設計すること。

以下、帳票出力方式について処理方式パターンや処理シーケンスを示す。

3.1.2.4.1 処理方式パターン

帳票出力方式における処理方式のパターンを下表に示す。

表 3.1-41 帳票出力方式の処理方式パターン

項番	分類	説明
1	クライアント帳票出力	サーバで作成した帳票ファイルをクライアントにダウンロードする方式。 処理イメージは「図 3.1-31 クライアント帳票出力の処理イメージ」を参照のこと。
2	サーバ帳票出力(同期)	サーバで帳票ファイルの作成と、帳票ライブラリへの印刷要求を同期処理で行う方式。 印刷は、業務要件によってフロアプリンタ又はサーバプリンタに印刷可能とする。 処理イメージは「図 3.1-32 サーバ帳票出力(同期)の処理イメージ」を参照のこと。
3	サーバ帳票出力(非同期)	サーバで帳票ファイルの作成と、帳票ライブラリへの印刷要求を非同期処理で行う方式。 印刷は、業務要件によってフロアプリンタ又はサーバプリンタに印刷可能とする。 処理イメージは「図 3.1-33 サーバ帳票出力(非同期)の処理イメージ」を参照のこと。

帳票出力方式における処理方式のイメージを下図に示す。

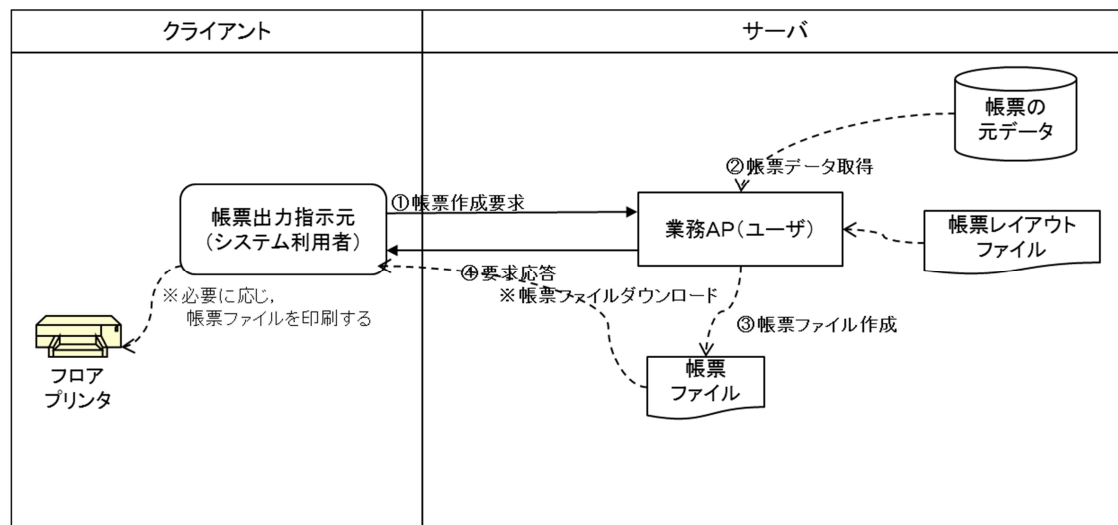


図 3.1-31 クライアント帳票出力の処理イメージ

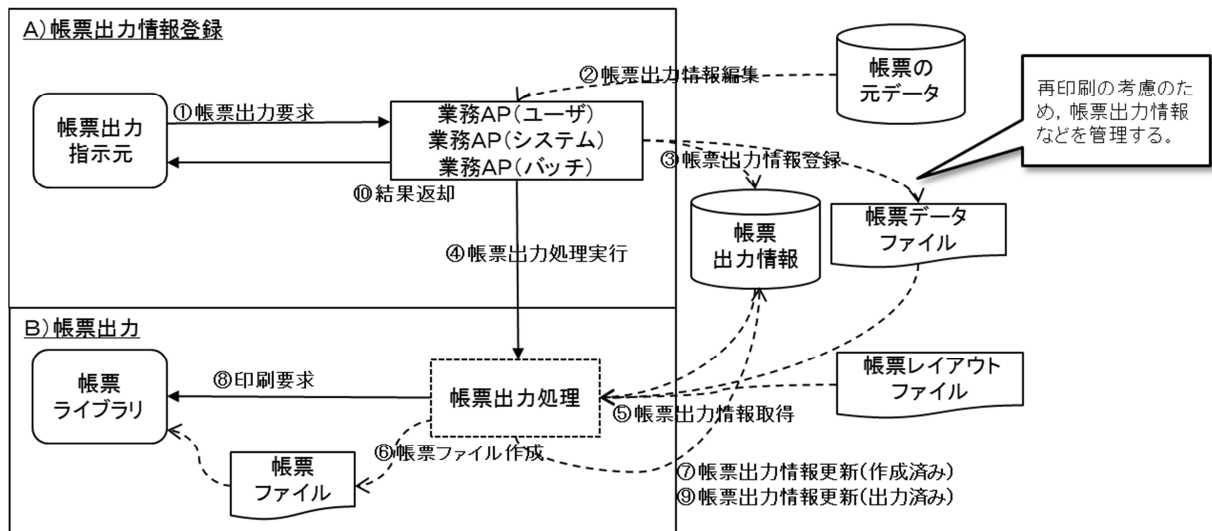


図 3.1-32 サーバ帳票出力(同期)の処理イメージ

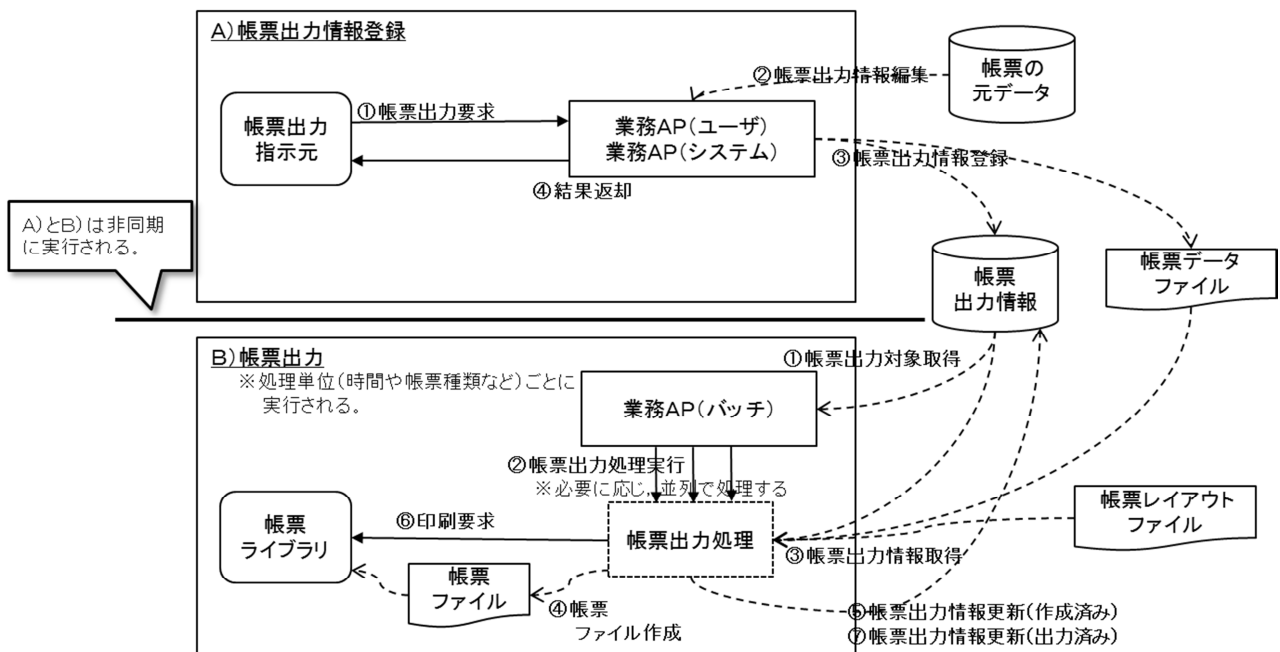


図 3.1-33 サーバ帳票出力(非同期)の処理イメージ

※ サーバ帳票出力(非同期)として説明する非同期処理の例は、非同期実行の対象として帳票ファイル作成や印刷要求を行う場合である。つまり、帳票データファイルを作成して帳票出力処理要求を登録する処理までを「3.1.1.4.2 オンライン処理方式」で実現し、処理要求に応じて帳票ファイル作成や印刷要求等を行う処理を「3.1.1.4.4 ディレードバッチ処理方式」で実現した場合の処理イメージである。別の実現方法としては、処理要求の受付のみをオンライン処理方式で実現し、帳票データファイル作成から印刷要求までの一連の処理をディレードバッチ処理方式で実現する場合がある。何の処理を非同期実行の対象とするかについては、業務要件や運用要件から再実行時に行われる処理内容を考慮し、設計するものとする。例えば、再実行のタイミングで最新データを取得して再出力するのではなく、帳票出力の指示が発生した時点のデータの内容で再出力することが要件である場合は前者の方式を採用し、そのような要件がない場合は画面のレスポンスタイムの観点から後者の方式を採用する、といった考慮をすること。

3.1.2.4.2 処理シーケンス

3.1.2.4.2.1 クライアント帳票出力

クライアント帳票出力における処理シーケンスのサンプルを下図、下表に示す。

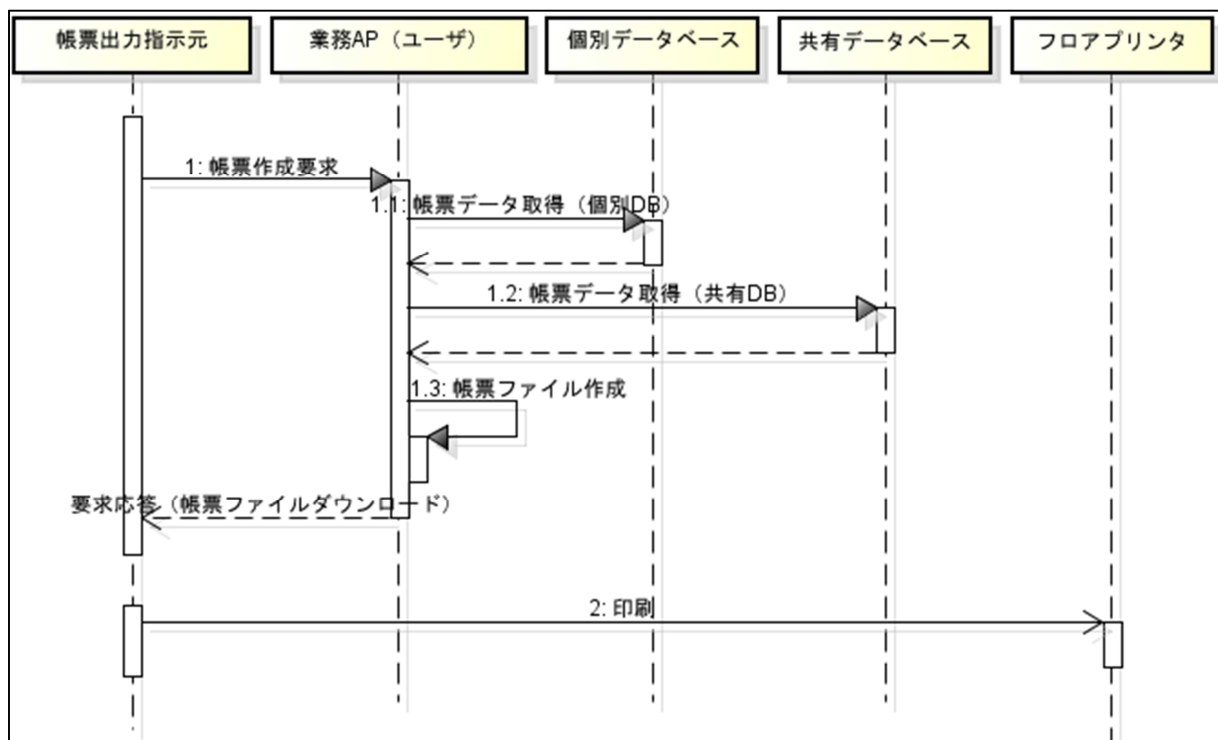


図 3.1-34 クライアント帳票出力の処理シーケンスの例

表 3.1-42 クライアント帳票出力の処理概要

項番	処理	処理概要
1	帳票作成要求	帳票ファイルの作成をサーバに要求する。
1.1	帳票データ取得 (個別DB)	個別データベースから帳票作成に必要なデータを取得する。
1.2	帳票データ取得 (共有DB)	必要に応じて、共有データベースから帳票作成に必要なデータを取得する。
1.3	帳票ファイル作成	帳票ファイルを作成し、クライアントにダウンロードする。
2	印刷	必要に応じ、ブラウザやOS機能を利用し帳票ファイルを印刷する。

3.1.2.4.2.2 サーバ帳票出力(同期)

サーバ帳票出力(同期)における処理シーケンスのサンプルを下図、下表に示す。

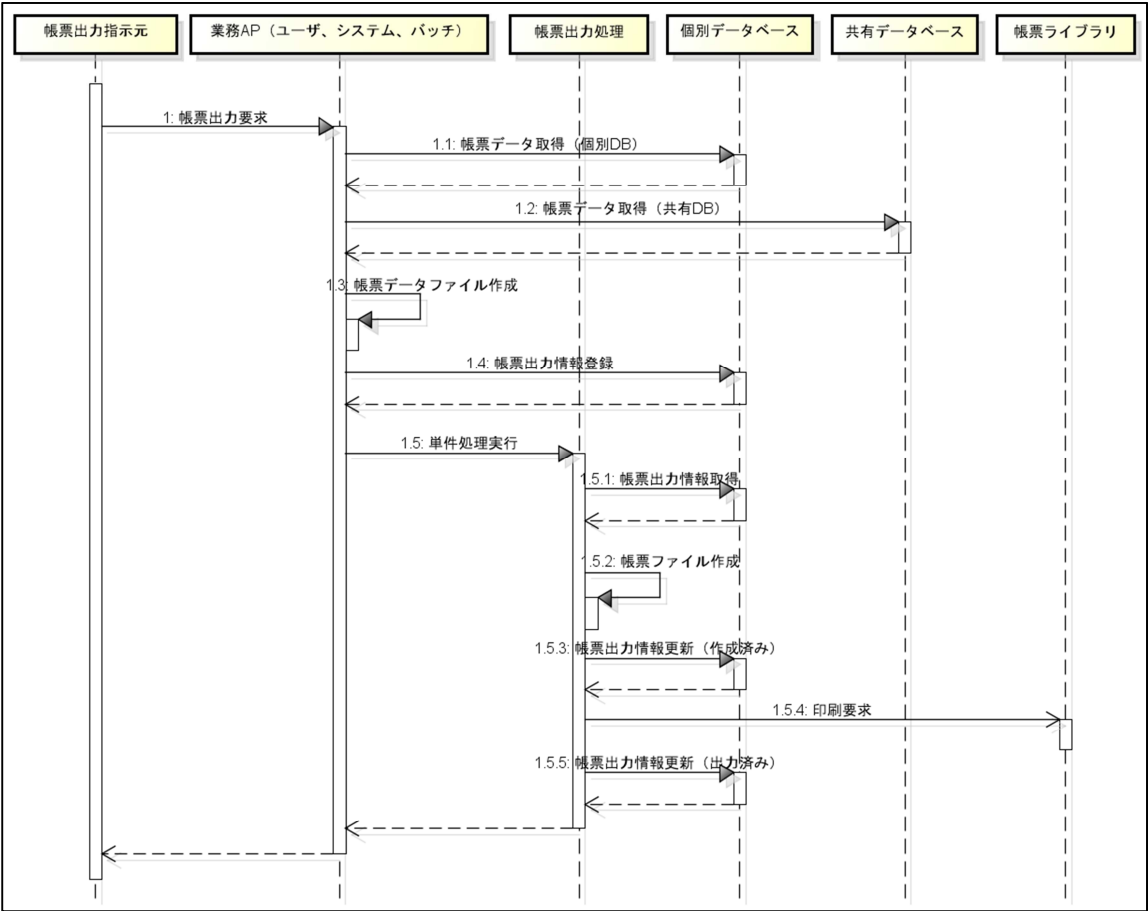


図 3.1-35 サーバ帳票出力(同期)の処理シーケンスの例

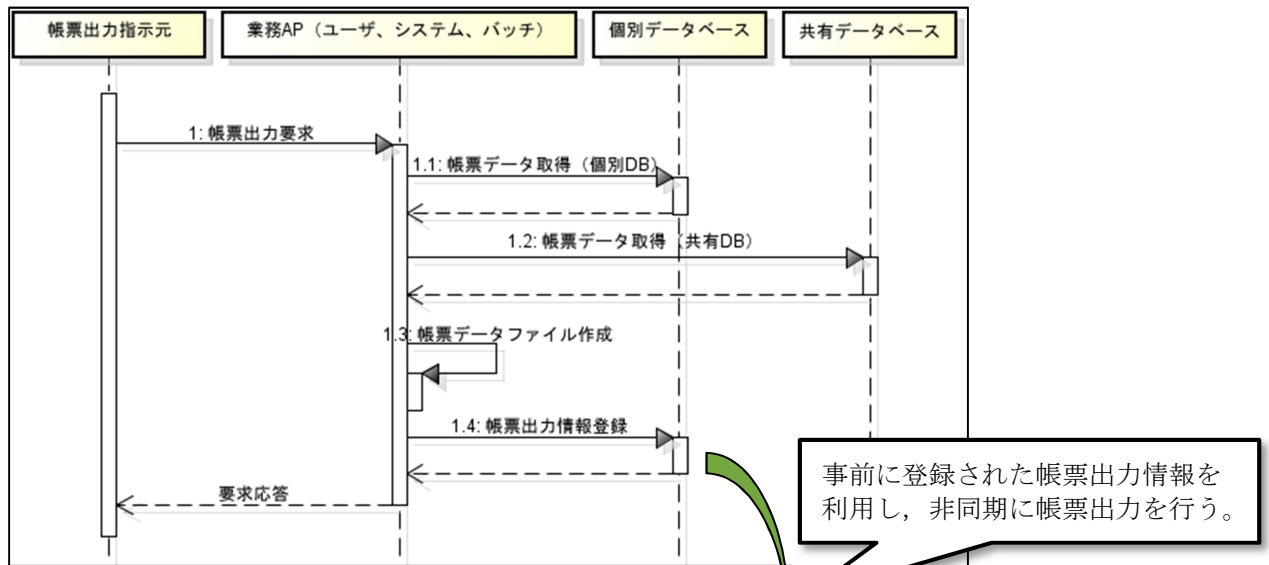
表 3.1-43 サーバ帳票出力(同期)の処理概要

項番	処理	処理概要
1	帳票出力要求	帳票出力をサーバに要求する。
1.1	帳票データ取得(個別DB)	個別データベースから帳票作成に必要なデータを取得する。
1.2	帳票データ取得(共有DB)	必要に応じて、共有データベースから帳票作成に必要なデータを取得する。
1.3	帳票データファイル作成	帳票データファイルを作成する。
1.4	帳票出力情報登録	個別データベースに帳票出力情報を登録する。
1.5	単件処理実行	帳票出力処理を呼び出す。この際、帳票出力情報の識別情報を引数として渡す。
1.5.1	帳票出力情報取得	引数の識別情報を条件に、個別データベースから帳票出力情報を取得する。
1.5.2	帳票ファイル作成	帳票出力情報の帳票データファイルや帳票レイアウトファイルのパス等を利用し、帳票ファイルを作成する。
1.5.3	帳票出力情報更新(作成済み)	帳票出力情報の帳票出力状況を作成済みに更新する。
1.5.4	印刷要求	帳票ライブラリに印刷を要求する。
1.5.5	帳票出力情報更新(出力済み)	帳票出力情報の帳票出力状況を出力済みに更新する。 ※1.5.4で帳票ライブラリへの要求が正常に完了した場合、出力済みに更新する。 その後、プリンタで紙詰まり等が発生した場合は業務継続方式に従い、再印刷を行う。

3.1.2.4.2.3 サーバ帳票出力(非同期)

サーバ帳票出力(非同期)における処理シーケンスのサンプルを下図、下表に示す。

帳票出力情報登録



帳票出力 ※非同期に実行される。

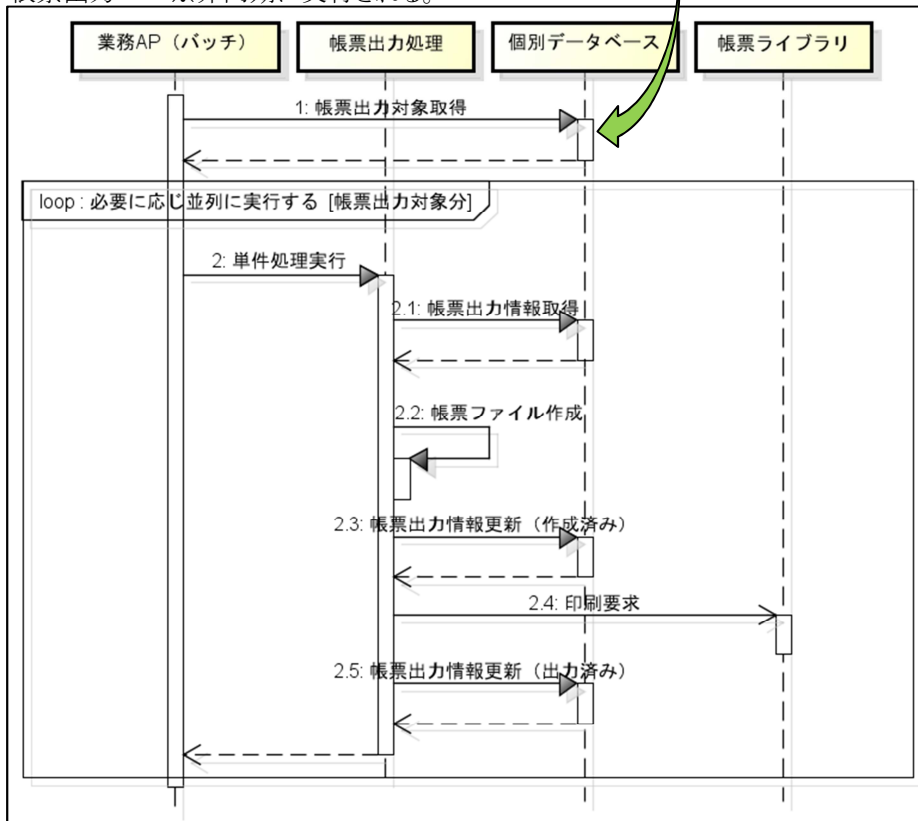


図 3.1-36 サーバ帳票出力(非同期)の処理シーケンスの例

表 3.1-44 サーバ帳票出力(非同期)(帳票出力情報登録)の処理概要

項番	処理	処理概要
1	帳票出力要求	帳票出力をサーバに要求する。
1.1	帳票データ取得(個別DB)	個別データベースから帳票作成に必要なデータを取得する。
1.2	帳票データ取得(共有DB)	必要に応じて、共有データベースから帳票作成に必要なデータを取得する。
1.3	帳票データファイル作成	帳票データファイルを作成する。
1.4	帳票出力情報登録	個別データベースに帳票出力情報を登録する。

表 3.1-45 サーバ帳票出力(非同期)(帳票出力)の処理概要

項番	処理	処理概要
1	帳票出力対象取得	個別データベースから出力対象の帳票出力情報の一覧を取得する。 出力対象か否かは、帳票出力情報の処理対象で判断する。 以降の処理は、取得した一覧全てに対し行う。
2	単件処理実行	帳票出力処理を呼び出す。この際、帳票出力情報の識別情報を引数として渡す。
2.1	帳票出力情報取得	引数の識別情報を条件に、個別データベースから帳票出力情報を取得する。
2.2	帳票ファイル作成	帳票出力情報の帳票データファイルや帳票レイアウトファイルのパス等を利用し、帳票ファイルを作成する。
2.3	帳票出力情報更新(作成済み)	帳票出力情報の帳票出力状況を作成済みに更新する。
2.4	印刷要求	帳票出力情報の帳票出力先や帳票ファイルを指定し、帳票ライブラリに印刷を要求する。
2.5	帳票出力情報更新(出力済み)	帳票出力情報の帳票出力状況を出力済みに更新する。 ※2.4で帳票ライブラリへの要求が正常に完了した場合、出力済みに更新する。 その後、プリンタで紙詰まり等が発生した場合は業務継続方式に従い、再印刷を行う。

3.1.2.4.3 業務継続方式

帳票処理方式における業務継続方式を以下に示す。

- 帳票出力前(帳票出力までの処理の途中)でエラーが発生する場合
帳票を出力する処理自体がまだ完了していないため、帳票出力を行うオンライン処理方式、バッチ処理方式、ディレイドバッチ処理方式のそれぞれの業務継続方式を実施する。
詳細は「3.1.2.1.5 業務継続方式」、「3.1.2.2.5 業務継続方式」、「3.1.2.3.5 業務継続方式」を参照のこと。
- 帳票出力後、プリンタの故障や紙詰まりが発生する場合
以下の方法で再印刷を行う。

表 3.1-46 再印刷の方法

項番	分類	説明
1	クライアント帳票出力	クライアントにダウンロードされている帳票ファイルを再印刷する。
2	サーバ帳票出力(同期)	データベースに管理されている帳票出力情報の帳票出力状況を印刷前の状態に戻し、帳票出力する機能を実行することで再印刷を可能とする。
3	サーバ帳票出力(非同期)	

なお、サーバ帳票出力(同期／非同期)では帳票出力要求時点のデータを再印刷することを可能とするため、以下の機能を持つ。

- 帳票内に出力するデータ(帳票データ)を管理する機能
- 帳票の出力状況や帳票データの在り処等のデータを管理する機能
- 上記データを利用して帳票を出力する機能

また、サーバ帳票出力(同期)が登録した帳票出力情報をサーバ帳票出力(非同期)が出力しないよう処理対象か否か判断するための情報や帳票の出力先を変更できるよう帳票出力情報に管理する。

表 3.1-47 帳票出力情報のイメージ

項番	管理項目	説明
1	識別情報	帳票出力情報を識別するための情報。
2	帳票出力状況	帳票の状況(待ち／作成済み／出力済み等)を管理する。
3	ファイル情報	以下のファイルのパス等を管理する。 ● 帳票データファイル¹³ ● 帳票レイアウトファイル¹⁴
4	処理対象	サーバ帳票出力(非同期)の処理対象か否かを管理する。 ※サーバ帳票出力(同期)とサーバ帳票出力(非同期)の帳票情報を同一テーブルで管理する場合、必要となる。
5	帳票出力先	帳票を出力するプリンタを管理する。
6	上記以外の管理項目	帳票出力日時等、必要に応じ管理する。

¹³ 帳票に出力するデータを管理するファイルを指す。

¹⁴ 帳票のレイアウトを管理するファイルを指す。

サーバ帳票出力の業務継続方式の処理イメージを下図に示す。

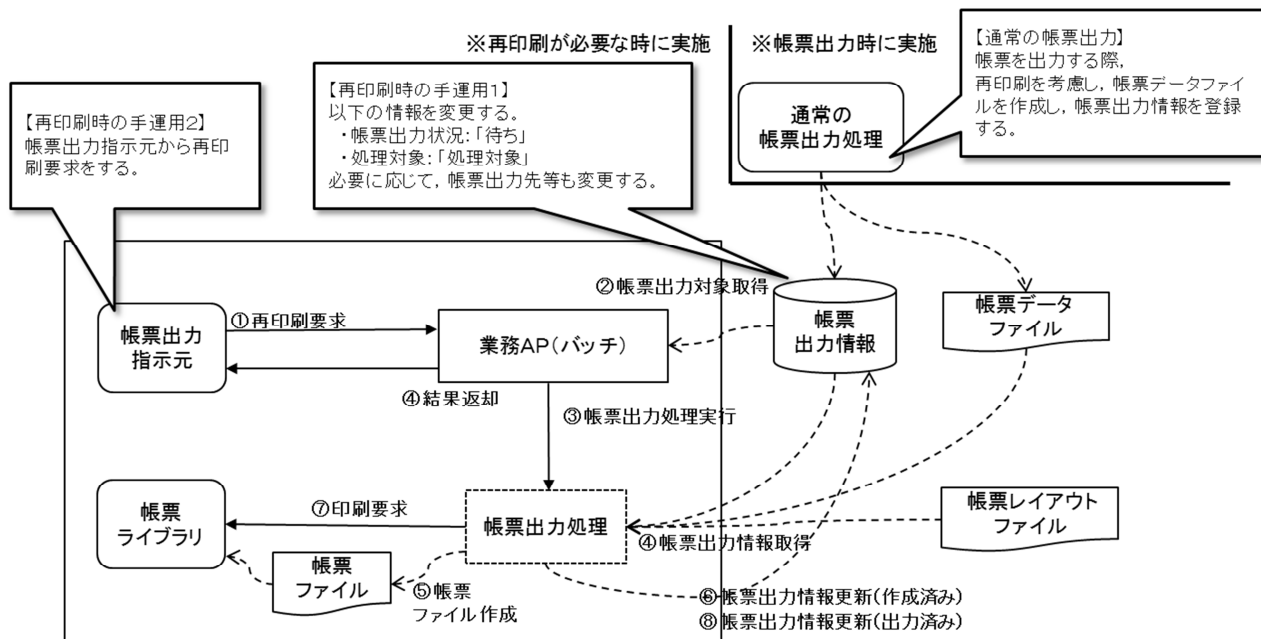


図 3.1-37 業務継続方式の処理イメージ

3.1.2.5 連携処理方式

スコープ:	階層定型化サブシステム及びインタフェース定型化サブシステム
プロセス:	2.4.2
指針1:	BPMSを介したサブシステム間連携方式は、「3.1.2.5.1.1 (1)連携方式」、「3.1.2.5.1.1 (2)設計・開発時の考慮点」に従って設計すること。
指針2:	BPMSとジョブ管理システム間の連携方式は、「3.1.2.5.1.2 (1)連携方式」、「3.1.2.5.1.2 (2)設計・開発時の考慮点」に従って設計すること。
指針3:	バッチ処理方式のサブシステム間連携方式は、「3.1.2.5.1.3 (1)連携方式」、「3.1.2.5.1.3 (2)設計・開発時の考慮点」に従って設計すること。
指針4:	外部システム連携層を介した外部システム連携方式は、「3.1.2.5.1.4 (1)連携方式」、「3.1.2.5.1.4 (2)設計・開発時の考慮点」に従って設計すること。

スコープ:	階層定型化サブシステム及びインタフェース定型化サブシステム
プロセス:	2.4.3
指針1:	連携処理方式の処理シーケンスは、「3.1.2.5.2 処理シーケンス」に示す例を参考に設計すること。
指針2:	連携処理方式のデータ保護方式は、「3.1.2.5.3 データの保護方式」に従って設計すること。
指針3:	連携処理方式の業務継続方式は、「3.1.2.5.4 業務継続方式」に従って設計すること。

連携処理方式とは、「3.1.1.3.1.1 システム構成要素の処理方式による分類」において示した処理方式同士の連携や外部システム連携層を介した内部システムと外部システムとの連携に関する処理方式を意味する。以下、連携処理方式について処理方式パターンや処理シーケンスを示す。

3.1.2.5.1 処理方式パターン

連携処理方式における処理方式のパターンを下表に示す。

表 3.1-48 連携処理方式の処理方式パターン

項番	組み合わせ	分類	説明
1	オンライン処理方式とオンライン処理方式の連携 ※サービス系の連携	BPMSを介したサブシステム間連携	業務フローとしてサブシステム間の連携を行う場合は、その連携ルートをBPMS上のワークフローにて記述し、BPMSを介して他サブシステムのBPMSや業務アプリケーション(システム)と連携を行う方式。 詳細は「3.1.2.5.1.1 BPMSを介したサブシステム間連携」を参照のこと。
2	オンライン処理方式とバッチ処理方式の連携	BPMSとジョブ管理システム間の連携	ToBeアーキテクチャでは、基本的にBPMSによって単件オンライン化されるが、一部の業務では業務アプリケーション(バッチ)により一括で処理を行うものも存在する。それらの処理を連携する処理方式。 オンライン処理方式(サービス系)とバッチ処理方式を組み合わせ、データベースやファイルでデータの授受を行い、スケジューラ起動等の契機により連携処理を行う。 詳細は「3.1.2.5.1.2 BPMSとジョブ管理システム間の連携」を参照のこと。
3	バッチ処理方式とバッチ処理方式の連携	バッチ処理方式のサブシステム間連携	サブシステム間にて、バッチ処理の連動やデータの授受を行う方式。 共有データベースやFTP(S)によるデータ授受を行い、ジョブ管理システムを介して連携を行う。 詳細は「3.1.2.5.1.3 バッチ処理方式のサブシステム間連携」を参照のこと。

項番	組み合わせ	分類	説明
4	内部システムと外部システムの連携	外部システム連携層を介した外部システム連携	外部システムとの連携には、それぞれの外部システムに固有のインタフェースやプロトコルが存在するため、それらのギャップを一元的に吸収する外部システム連携層を介してのみ行うものとする。その際の処理方式。 内部システムから外部システム連携を行う際、外部システム連携層が提供する内部インタフェースと同等のアクセス方法を用いて処理を実施する。 詳細は「3.1.2.5.1.4 外部システム連携層を介した外部システム連携」を参照のこと。

3.1.2.5.1.1 BPMSを介したサブシステム間連携

(1) 連携方式

BPMSを介したサブシステム間連携の連携方式を下表に示す。

表 3.1-49 BPMSを介したサブシステム間連携の連携方式

項番	連携方式	概要
1	ワークフロー連携	ワークフローから他サブシステムのワークフローに直接アクセスし、開始又は待機中のワークフローを再開する。(ワークフロー呼出) ただし、待機可能な箇所が複数あるワークフローを再開する場合、現在どこで待機しているか判断する必要があるため、BPMS補完機能を介してワークフローを再開する。(BPMS補完機能呼出)
2	サービス連携	ワークフローから他サブシステムのサービスにアクセスし、連携データの受け渡し等を行う。 なお、処理方式は、「オンライン処理方式」の「サービス系」と同様である。 詳細は、「3.1.1.4.2 オンライン処理方式」及び「3.1.2.1 オンライン処理方式」を参照のこと。

(2) 設計・開発時の考慮点

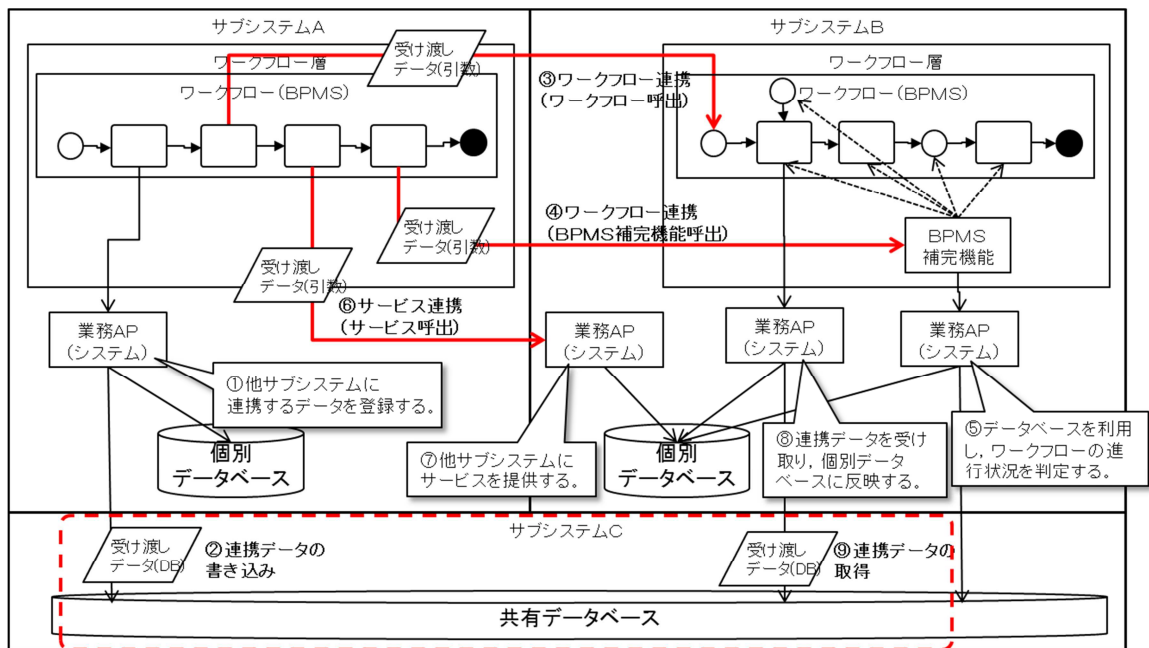
設計・開発時に考慮すべき点を以下に示す。

- データの受け渡しは、共有データベースを用いて行うこと。
- ただし、共有データベースで管理すべきと定義されていないデータは、ワークフローのプロセスデータを経由してデータの受け渡しを行うこと。
- 共有データベースの特定サブシステム間共有データ¹⁵に一時的に保持される受け渡しデータは、データを受領するサブシステム側で使用後に削除すること。

(3) 処理方式のイメージ

BPMSを介したサブシステム間連携における処理方式のイメージを下図に示す。

¹⁵ 詳細は、『データ統合方針書』の「2.3 データ配置の考え方」を参照のこと。



- 連携先のサブシステムとのデータの受け渡しは共有データベースを使用する(①, ②)。
- ワークフロー (BPMS) を介したサブシステムの連携を行う。連携の際、共有データベースに保存すべきでないデータは、連携時に引数で受け渡す。
 - ワークフロー連携 (ワークフロー呼出)
ワークフローから連携先サブシステムのワークフローにアクセスする(③)。
 - ワークフロー連携 (BPMS 補完機能呼出)
待機可能な箇所が複数ある連携先サブシステムのワークフローを再開する場合、BPMS 補完機能呼び出し、BPMS 補完機能がワークフロー再開の判断を行う。ワークフロー再開の判断に必要な情報の取得を業務 AP (システム) に依頼する(④, ⑤)。
 - サービス連携 (サービス呼出)
ワークフローから連携先サブシステムのサービス (業務 AP (システム)) にアクセスする(⑥, ⑦)。
- 連携先のサブシステムでは、共有データベース又は引数で受け取ったデータに基づき処理を行う(⑧, ⑨)。

図 3.1-38 BPMSを介したサブシステム間連携の処理イメージ

3.1.2.5.1.2 BPMSとジョブ管理システム間の連携

(1) 連携方式

BPMSとジョブ管理システム間の連携方式を下表に示す。

表 3.1-50 BPMSとジョブ管理システム間の連携方式

項番	連携方式	概要
1	BPMSからジョブ管理システムに連携する。	BPMS(単件処理)からジョブ管理システム(一括処理)に連携して処理を行う場合、以下のように処理を行う。 ① BPMSは、サービスを介して、単件処理毎に受け渡しデータを共有データベースに格納する。 ② ジョブ管理システムは、時間起動等のタイミングで、業務アプリケーション(バッチ)を実行する。 ③ 業務アプリケーション(バッチ)は、共有データベースから複数の受け渡しデータを抽出し、一括で処理を行う。
2	ジョブ管理システムからBPMSに連携する。	ジョブ管理システム(一括処理)からBPMS(単件処理)に連携して処理を行う場合、以下のように処理を行う。 ① ジョブ管理システムは、時間起動等のタイミングで、業務アプリケーション(バッチ)を実行する。 ② 連携先サブシステムの業務アプリケーション(バッチ)に連携する。 ③ 連携先サブシステムの業務アプリケーション(バッチ)は、共有データベースから複数の受け渡しデータを抽出し、単件処理に分割してBPMSを呼び出す。この際、BPMSに受け渡しデータのキー情報のみを受け渡す。 ④ BPMSは、サービスを介して、共有データベースからキー情報を条件に受け渡しデータを抽出し、単件処理を行う。

(2) 設計・開発時の考慮点

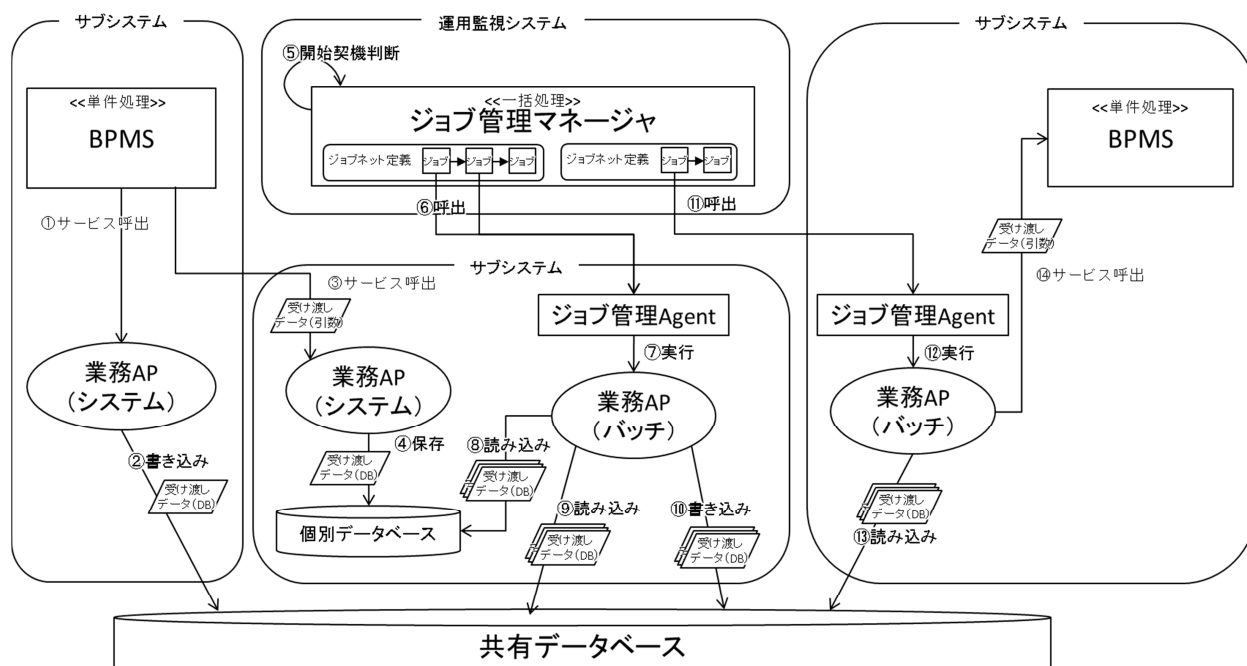
設計・開発時に考慮すべき点を下表に示す。

表 3.1-51 設計・開発時の考慮点

項番	考慮点	説明
1	BPMSからジョブ管理システムに連携する場合	データの受け渡し 共有データベースを用いてデータの受け渡しを行うこと。 共有データベースで管理すべきと定義されていないデータは、BPMSから運用監視システムが実行する業務アプリケーション(バッチ)を持つサブシステムへ渡すこと。(BPMSから渡された情報をバッチ処理実行時まで溜めておく。また、バッチ処理後に不要となったデータはデータを受領するサブシステムが削除すること。)
2	ジョブ管理システム(一括処理)の開始契機	BPMS終了時点の振る舞い 運用監視システム側が時間起動(日次等)であれば、単件処理の終了時点で運用監視システムと連携する必要はなく、共有データベースに連携用データを格納するだけでよい。ただし、何らかのデータを渡す必要がある場合は、時間起動であっても運用監視システムが実行する業務AP(バッチ)を持つサブシステムへアクセスすること。
3		ジョブ管理システム(一括処理)の開始契機 時間起動であればジョブ管理システムが自ら契機を判断する。 上流システムからの契機によって起動する場合は、BPMSから運用監視システムが実行する業務AP(バッチ)を持つサブシステムへのアクセスにより契機を判断すること。
4	ジョブ管理システムからBPMSに連携する場合	データの受け渡し 共有データベースを用いてデータの受け渡しを行うこと。 共有データベースに格納すべきでないデータは、FTP等により連携すること。また、処理後、不要となったファイルは、データを受領するサブシステムが使用後に削除すること。
5	BPMS(単件処理)の開始契機	ジョブ管理システム(一括処理)の終了時点の振る舞い 連携元のバッチ処理の終了を契機に、連携先のバッチ処理を実行する。BPMSは単件単位で実行する必要があるため、連携先の業務AP(バッチ)にて共有データベースに登録されている複数件のデータを単件単位に分解しBPMSを開始する。
6		BPMS(単件処理)の開始契機 「ジョブ管理システム(一括処理)の終了時点の振る舞い」に記載のタイミングでBPMSを開始する。

(3) 処理方式のイメージ

BPMSとジョブ管理システム間における処理方式のイメージを下图に示す。



- 共有データベースに保存すべきでないデータは、BPMSから業務AP(バッチ)と同一のサブシステムである業務AP(システム)が受け取り、引数で渡されたデータを保存する(③～④)。ジョブの実行時に、上記で保存したデータと共有データベースに格納されたデータを読み込み、一括処理を行う(⑤～⑨)。
- 連携元のジョブネットが終了し、連携先のジョブネットが開始される(⑪～⑫)。ジョブネット間の連携は、時刻起動やFTP等によるファイル到着契機がある。連携先の業務AP(バッチ)は、データベースから複数件のデータを取得し、単件単位にBPMSを呼び出す(⑬～⑭)。

図 3.1-39 BPMSとジョブ管理システム間連携の処理イメージ

3.1.2.5.1.3 バッチ処理方式のサブシステム間連携

(1) 連携方式

バッチ処理方式の連携は、『運用管理エージェント登録ガイドライン』に準じて、ジョブ管理システムで管理されるジョブネットを利用する。ジョブネットは、サブシステム毎に構成するため、他サブシステムのバッチ処理と連携を行う場合には、以下に示す方法で連携を行う。

- 他サブシステムのジョブネットとの連携は、時刻起動あるいはFTP(S)等によるファイル到着契機で実行する。

(2) 設計・開発時の考慮点

設計・開発時に考慮すべき点を以下に示す。

- データの受け渡しは、共有データベースを用いて行うこと。
- ただし、共有データベースで管理すべきと定義されていないデータは、FTP(S)を使用してファイルで受け渡しをすること。
- 共有データベースの特定サブシステム間共有データ¹⁶に一時的に保持される受け渡しデータやFTPで受領したファイルは、データを受領するサブシステム側で使用後に削除すること。

¹⁶ 詳細は、『データ統合方針書』の「2.3 データ配置の考え方」を参照のこと。

3.1.2.5.1.4 外部システム連携層を介した外部システム連携

(1) 連携方式

外部システム連携層は、インタフェース定型化サブシステムのため、外部システム連携層に持たせる機能等は、外部システム連携サブシステムの設計に關与するステークホルダ(設計・開発ベンダ等)が個別にアーキテクチャを定めるものとする。

外部システム連携層に持たせる機能は、「3.2.5.1.2 外部システム連携層に持たせる機能」を参照のこと。また、内部システムとの通信におけるプロトコルは、「3.2.1.1.2 内部インタフェースのプロトコル」を参照のこと。

(2) 設計・開発時の考慮点

外部システム連携層を介した外部システム連携のうち、内部システムとESBとの通信において考慮すべき点を以下に示す。

- データの受け渡しはキー情報のみとし、その他のデータは共有データベースを介して受け渡しすること。
- 以下に示す「FTP(S)の使用条件」に適合する場合は、FTP(S)を使用しファイルにてデータの受け渡すこと。

FTP(S)の使用条件:

- 大容量ファイルを効率よく転送したい場合や一括処理のため複数件数のレコードデータを1つのファイルにまとめてやり取りしたい場合等、データベース経由では、効率が極めて悪い場合。
- 共有データベースで管理すべきと定義されていないデータを受け渡す場合。
- 共有データベースの特定サブシステム間共有データ¹⁷に一時的に保持される受け渡しデータやFTPで受領したファイルは、データを受領するサブシステム側で使用後に削除すること。
- サブシステムと外部システムとの連携において、内部システムで扱うデータは、基本的にリアルタイム連携による業務単件データが中心であるため、外部システムに対し、業務単件データをまとめて連携する場合、連携タイミングにギャップが発生する。外部システム連携層でこのような連携タイミング等のギャップを吸収するために、サブシステムからESBへの通信は同期で連携し、その先の外部システムへの通信は外部システム互換機能を含めた形で、業務要件に応じて同期で連携するか非同期で連携するかを定めること。逆の方向、つまり外部システムからESB又は外部システム互換機能への通信、その先のサブシステムへの通信は、業務要件に応じて同期／非同期を定めること。

¹⁷ 詳細は、『データ統合方針書』の「2.3 データ配置の考え方」を参照のこと。

(3) 処理方式のイメージ

外部システム連携層を介した外部システム連携における処理方式のイメージを下図に示す。

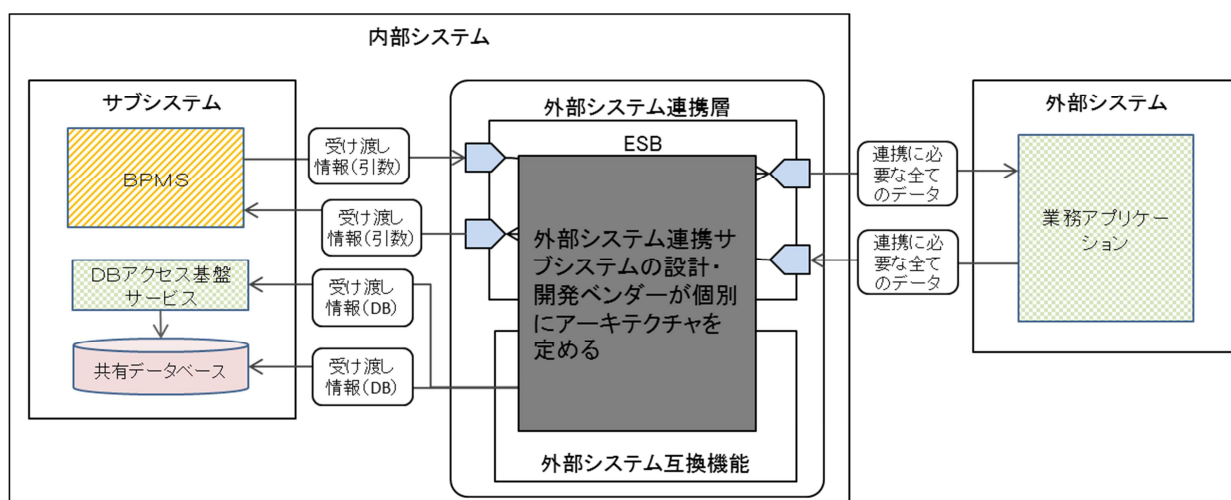


図 3.1-40 外部システム連携層を介した外部システム連携の処理イメージ

3.1.2.5.2 処理シーケンス

3.1.2.5.2.1 BPMSを介したサブシステム間連携

BPMSを介したサブシステム間連携の処理シーケンスのサンプルを下図、下表に示す。

例： ワークフロー呼び出しで、データを連携する場合

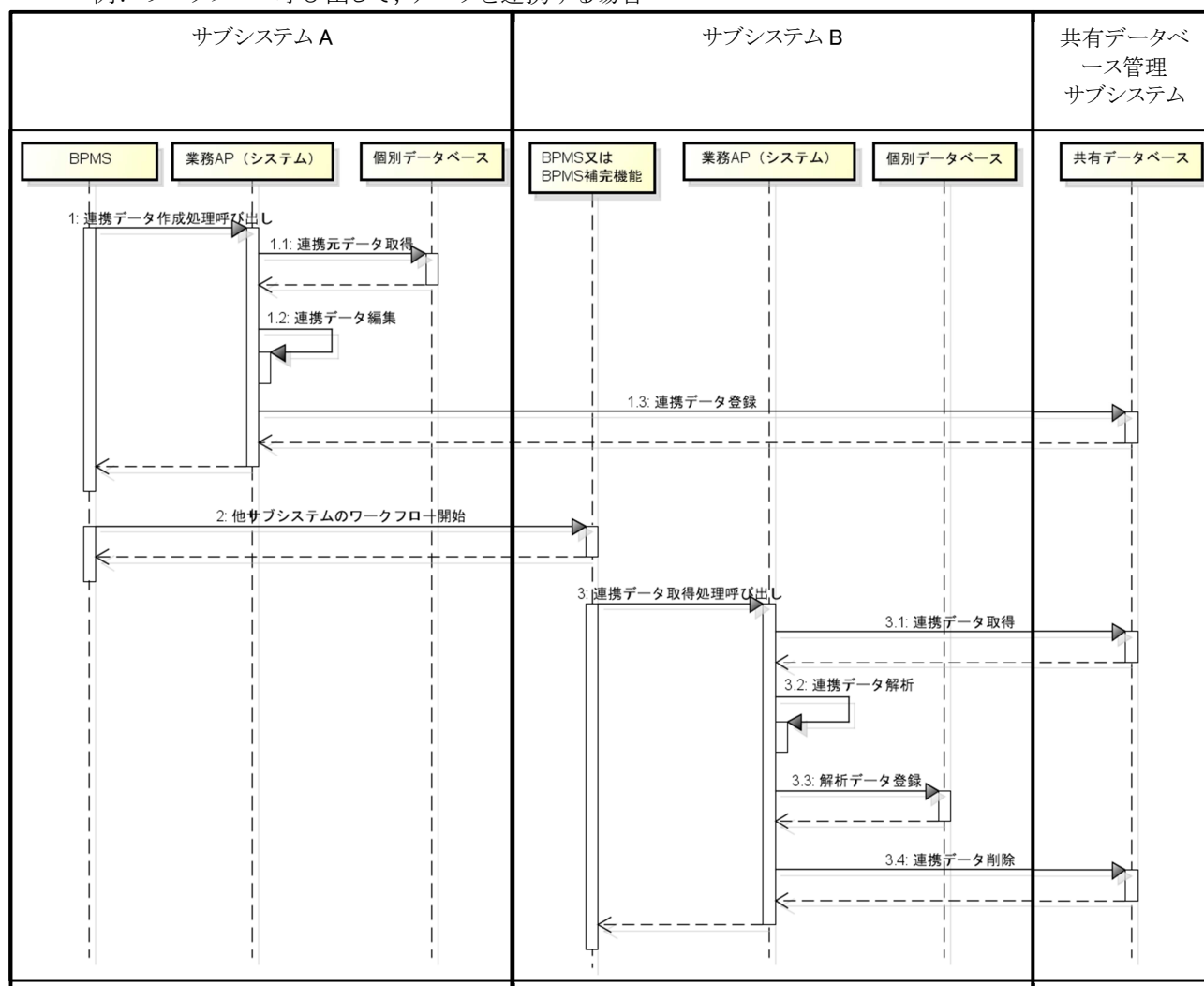


図 3.1-41 BPMSを介したサブシステム間連携の処理シーケンスの例

表 3.1-52 BPMSを介したサブシステム間連携の処理概要

項番	処理	処理概要
＜サブシステムA＞		
1	連携データ作成処理呼び出し	連携データを作成するためのサービスを呼び出す。
1.1	連携元データ取得	個別データベースから連携データの元データを取得する。
1.2	連携データ編集	他サブシステムに連携するデータを作成する。
1.3	連携データ登録	共有データベースに連携データを登録する。
2	他サブシステムのワークフロー開始	他サブシステムのワークフローを開始する。
＜サブシステムB＞		
3	連携データ取得処理呼び出し	連携データを取得するためのサービスを呼び出す。
3.1	連携データ取得	共有データベースから連携データを取得する。
3.2	連携データ解析	連携データを解析し、業務処理を行う。
3.3	解析データ登録	個別データベースに解析データを登録する。
3.4	連携データ削除	「1.3 連携データ登録」で登録したデータを削除する。

3.1.2.5.2.2 BPMSとジョブ管理システム間の連携

BPMSとジョブ管理システム間の連携の処理シーケンスのサンプルを下図、下表に示す。

例：BPMSで単件処理で登録したデータをジョブ管理システムから一括で処理する場合

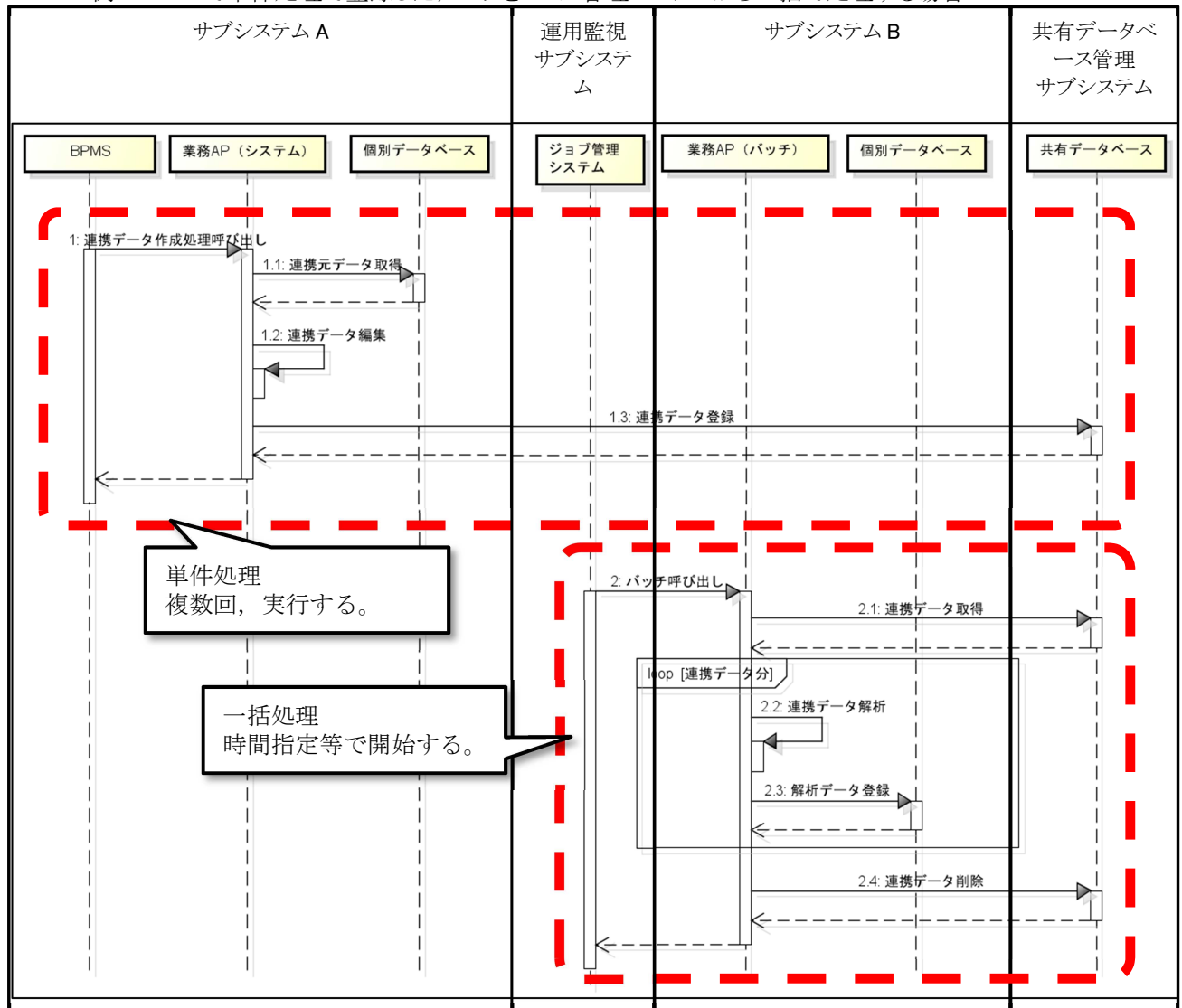


図 3.1-42 BPMSとジョブ管理システム間の連携の処理シーケンスの例

表 3.1-53 BPMSとジョブ管理システム間の連携の処理概要

項番	処理	処理概要
<サブシステムA>		
※単件処理が複数回、実行され、データを蓄積する。		
1	連携データ作成処理呼び出し	連携データを作成するためのサービス呼び出す。
1.1	連携元データ取得	個別データベースから連携データの元データを取得する。
1.2	連携データ編集	他サブシステムに連携するデータを作成する。
1.3	連携データ登録	共有データベースに連携データを登録する。
<サブシステムB>		
※データファイルの到着契機起動や時間指定等により開始され、一括で処理する。		
2	バッチ呼び出し	ジョブ管理システムからバッチを呼び出す。
2.1	連携データ取得	「1.3.連携データ登録」で登録された連携データを、共有データベースから取得する。 以降の処理は、取得したデータ全てに対し繰り返し実行する。
2.2	連携データ解析	連携データを解析し、業務処理を行う。
2.3	解析データ登録	個別データベースに解析データを登録する。
2.4	連携データ削除	「1.3 連携データ登録」で登録したデータを削除する。

3.1.2.5.2.1 バッチ処理方式のサブシステム間連携

バッチ処理方式のサブシステム間連携の処理シーケンスのサンプルを下図、下表に示す。

例： バッチ処理方式で連携ファイルを用いて他サブシステムへ連携する場合

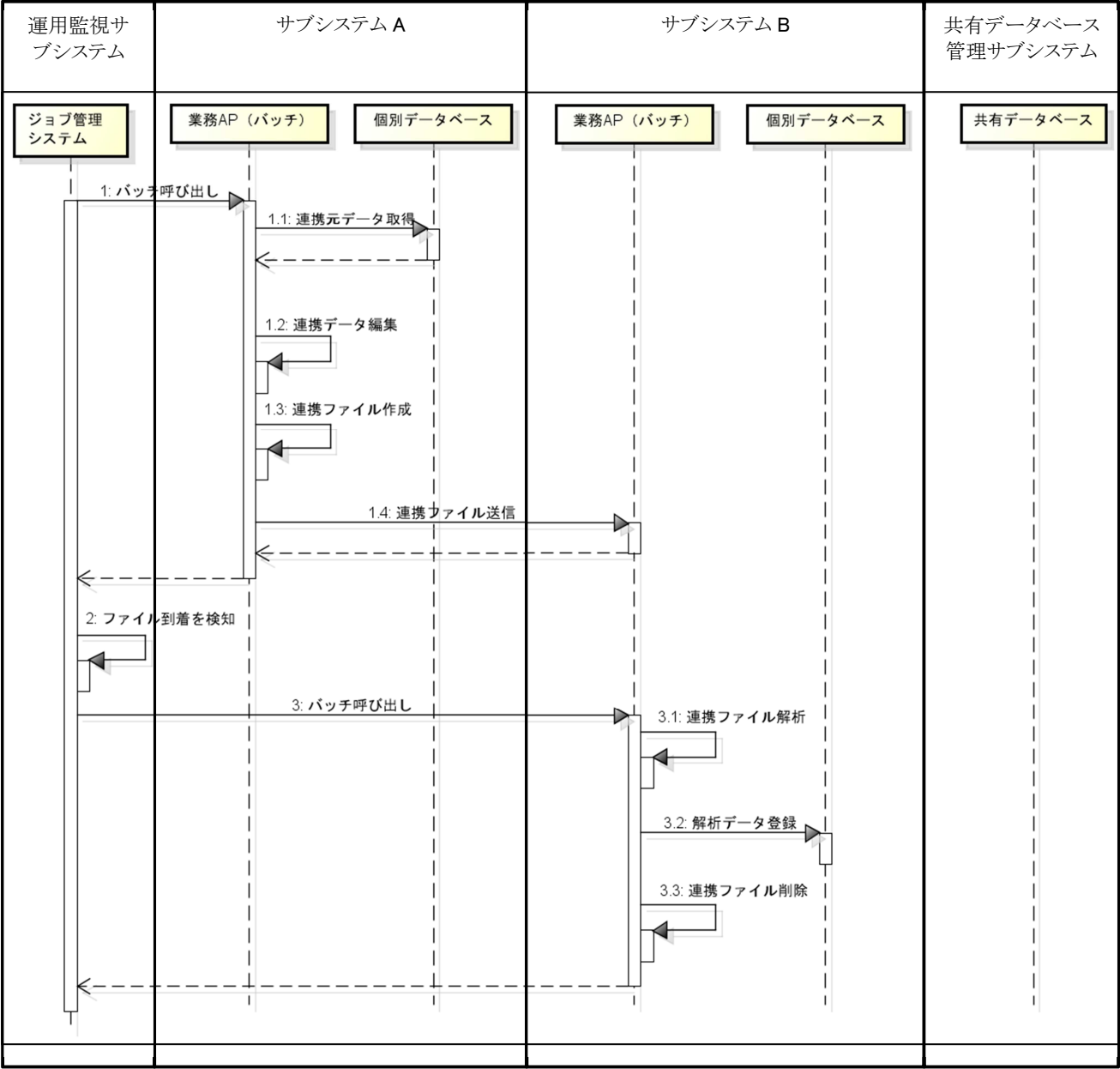


図 3.1-43 バッチ処理方式のサブシステム間連携の処理シーケンスの例

表 3.1-54 バッチ処理方式のサブシステム間連携の処理概要

項番	処理	処理概要
＜サブシステムA＞		
※時間指定等で開始され、一括で処理する。		
1	バッチ呼び出し	ジョブ管理システムからバッチを呼び出す。
1.1	連携元データ取得	個別データベースから連携元データを取得する。
1.2	連携データ編集	他サブシステムに連携するデータを作成する。
1.3	連携ファイル作成	連携ファイルに連携データを出力する。
1.4	連携ファイル送信	連携ファイルをサブシステムへ送信する。
＜サブシステムB＞		
※データファイルの到着契機起動で開始され、一括で処理する。		
2	ファイル到着を検知	ジョブ管理システムが連携ファイルが到着したことを検知する。
3	バッチ呼び出し	ジョブ管理システムからバッチを呼び出す。
3.1	連携ファイル解析	連携ファイルのデータを解析し、業務処理を行う。
3.2	解析データ登録	個別データベースに解析データを登録する。
3.3	連携ファイル削除	連携ファイルを削除する。

3.1.2.5.2.2 外部システム連携層を介した外部システム連携

外部システム連携層を介した外部システム連携の処理シーケンスのサンプルを下図, 下表に示す。

例：単件処理したデータをBPMSから一括で外部システムに連携する場合

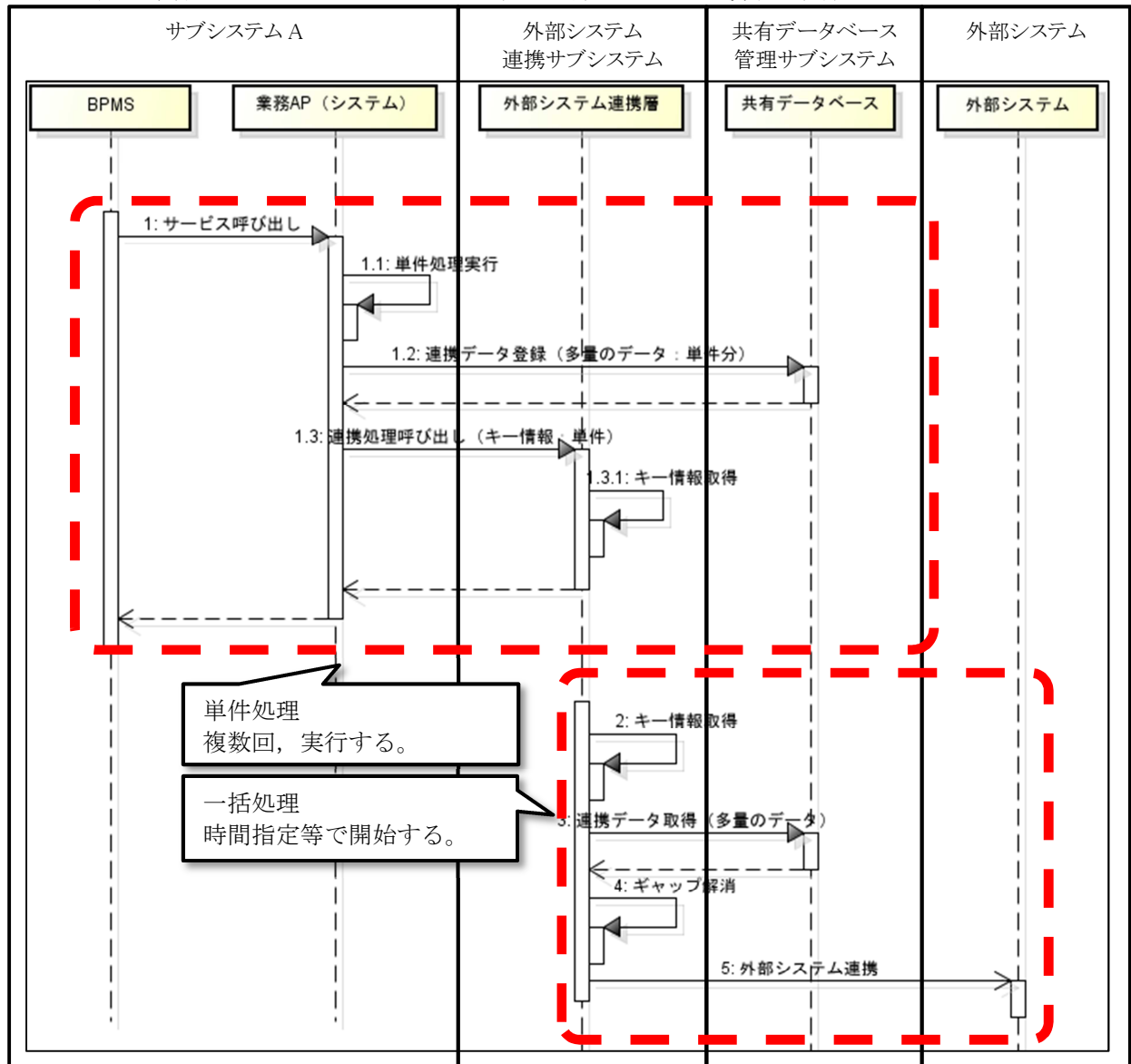


図 3.1-44 外部システム連携層を介した外部システム連携の処理シーケンスの例

表 3.1-55 外部システム連携層を介した外部システム連携の処理概要

項番	処理	処理概要
＜サブシステムA＞		
※単件処理が複数回実行され、データを蓄積する。		
1	サービス呼び出し	単件処理を実行し、外部システムと連携準備をする処理を呼び出す。 ※このサービスが完了した時点では、外部システムに連携されない。
1.1	単件処理実行	単件単位に業務処理を行う。
1.2	連携データ登録 (多量のデータ:単件分)	外部システムに多量のデータを連携する場合は、共有データベースにデータを一時的に格納する。
1.3	連携処理呼び出し (キー情報:単件分)	外部システムと連携準備をするための処理を呼び出す。 外部システムに連携するデータ(共有データベースにアクセスするためのキー情報)を引数で渡す。
1.3.1	キー情報蓄積	外部システム連携層内に、キー情報を蓄積する。
＜外部システム連携サブシステム＞		
※データファイルの到着契機起動や時間指定等により開始され、一括で処理する。		
2	キー情報取得	外部システム連携層に蓄積されているキー情報を取得する。
3	連携データ取得 (多量のデータ)	共有データベースから連携データ(多量のデータ)を取得する。
4	ギャップ解消	内部システムと外部システムのギャップを解消する。
5	外部システム連携	外部システムと連携を行う。

3.1.2.5.3 データの保護方式

連携処理方式のデータの保護方式は、オンライン処理方式のデータの保護方式と同様とする。
詳細は「3.1.2.1.3 データの保護方式」を参照のこと。

ただし、BPMSを介したデータ授受を行う場合は以下も考慮すること。

- 他のビジネスプロセスインスタンスによってデータベースの値が更新され、それを自ビジネスプロセスインスタンスが知る必要がある場合、基本的にはBPMSが通知することで、データ不整合が起きないように制御すること。
- 僅かなタイムラグで不整合が起きる可能性があるため、データベース更新時は楽観排他又は悲観排他を行うこと。

3.1.2.5.4 業務継続方式

連携処理方式の業務継続方式は、オンライン処理方式及びバッチ処理方式の業務継続方式と同様とする。

詳細は「3.1.2.1.5 業務継続方式」、「3.1.2.2.5 業務継続方式」を参照のこと。

なお、BPMSやESB等のプログラムプロダクトについては、導入する製品によって業務継続方式が異なるため、再実行の方法等を個別に設計するものとする。