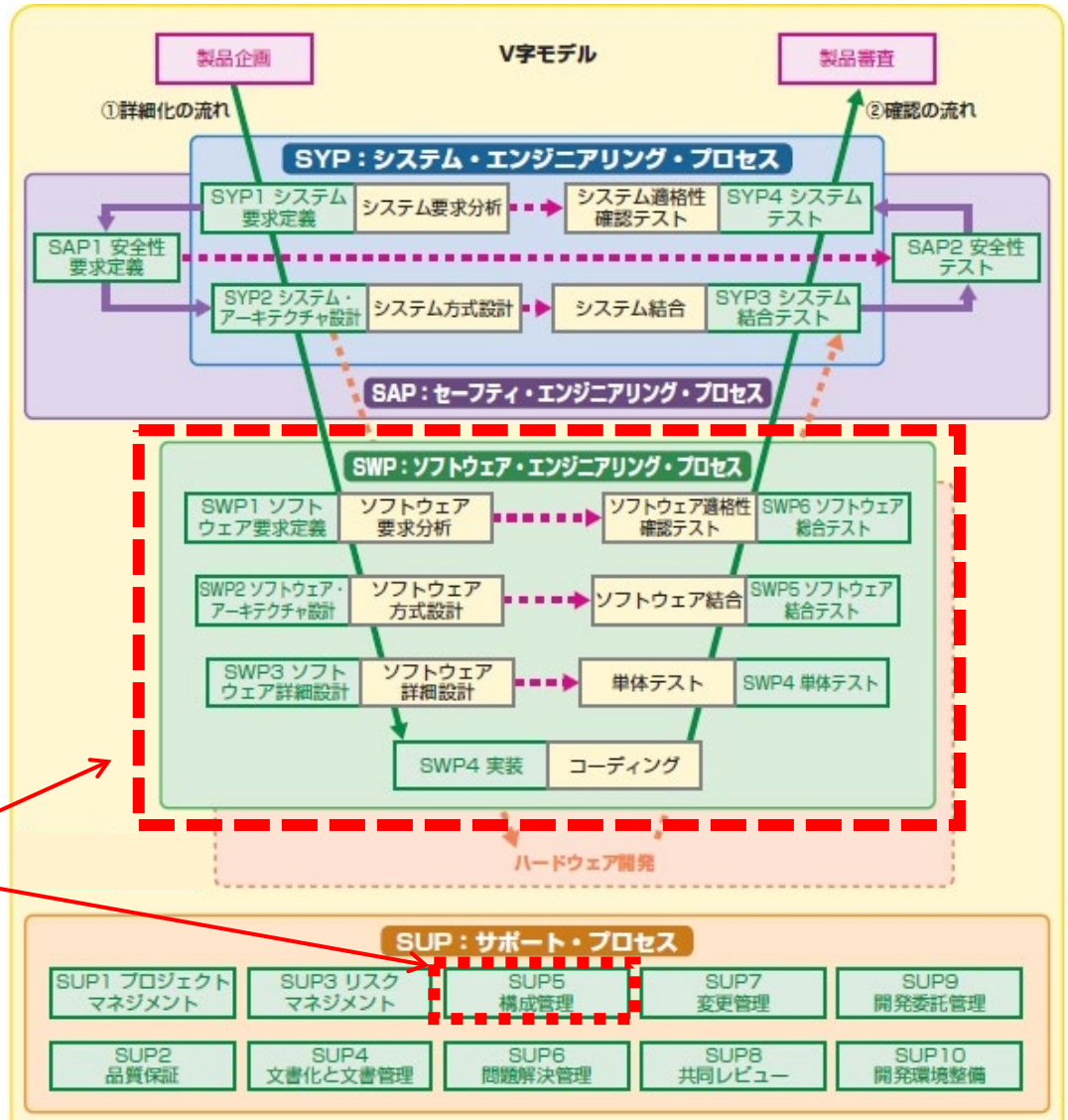
ソフトウェア開発プロセス標準

ウォータフォールモデル（V字モデル）

- ISO/IEC/IEEE 15288(JIS X0170)
  システムライフサイクルプロセス
- ISO/IEC/IEEE 12207（JIS X0160）
  ソフトウェアライフサイクルプロセス



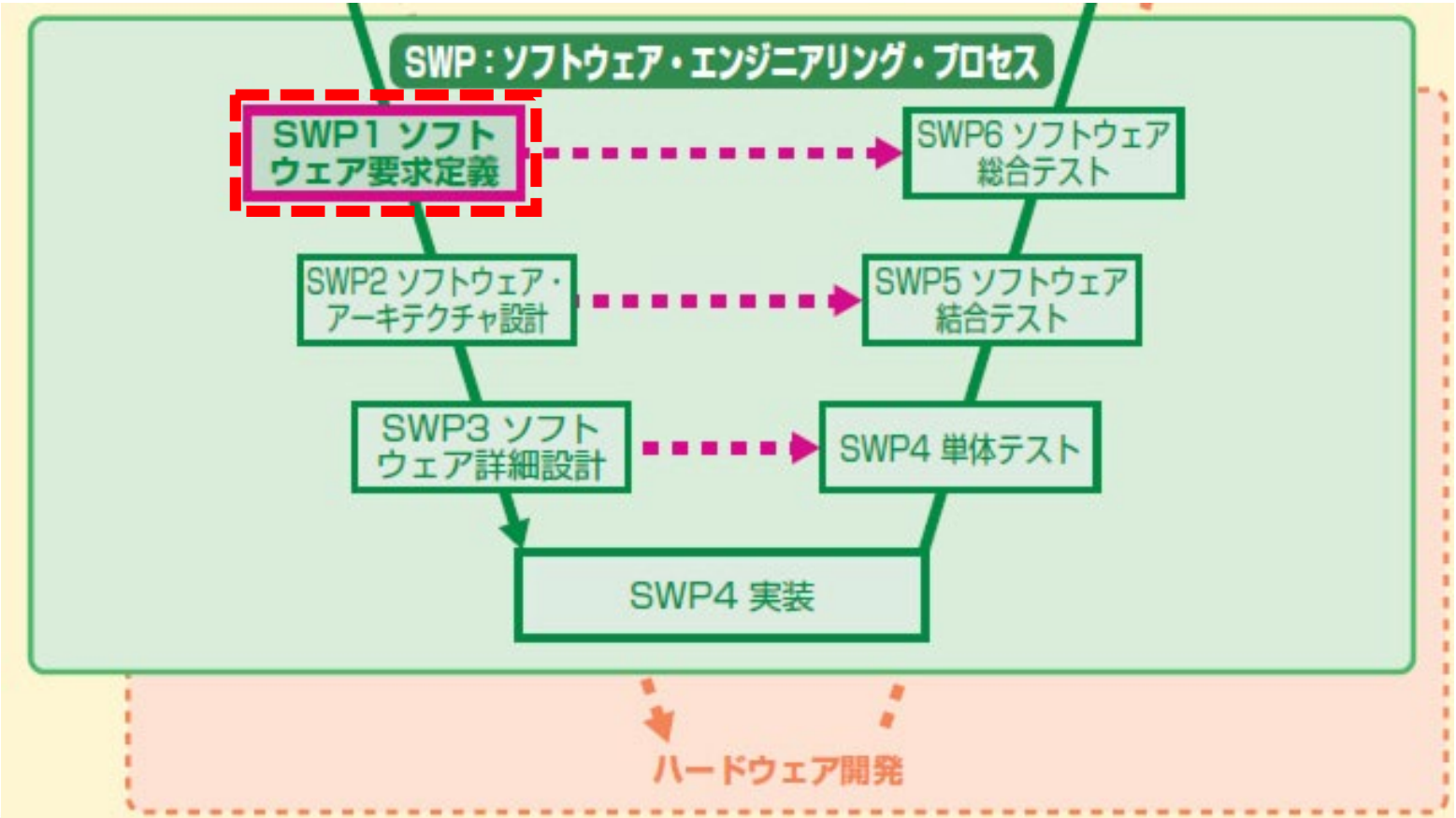
ソフトウェアシステム開発

第2回： 開発プロセス（要件定義～初期設計）

1. 要件定義（＝要求定義，要求分析）

1.1 目標

- 当該製品を実現するためにソフトウェアとして実現が必要となる要求を明確にする．



【用語】
- プロセス
- 工程またはフェーズ

1.2 入力，出力およびタスク構成

入力	タスク構成	出力
製品仕様書	A) ソフトウェア要求仕様書の作成 • 制約条件の確認 • ソフトウェア機能要求事項の明確化 • ソフトウェア非機能要求事項の明確化 • 要求の優先順位付け • ソフトウェア要求仕様書の作成 B) ソフトウェア要求仕様の確認 • ソフトウェア要求仕様書の内部確認	ソフトウェア要求仕様書， レビュー報告書（ソフトウェア要求定義）

A）ソフトウェア要求仕様書の作成

- 制約条件の確認
  - 概要

    ソフトウェアに関する要求事項を検討するにあたっての考慮すべき制約条件を明確にし，制約条件リストとしてまとめる．
  - 実施内容

    以下の①～⑥の事項を確認し，制約条件リストとして整理する．

    ①製品目標を確認する

    - 特徴的な新機能の有無
    - プロダクトライン開発の適用　など

    ②製品特性を確認する

    - 信頼性要求，安全性要求
    - 耐用年数，製品寿命

- 想定される利用状況，利用環境
- 準拠しなければならない規格・規約の有無　など

③製品のステークホルダ（利害関係者）を確認する

- サービス部門，営業，企画，ハード開発部門，製造部門など，製品に関わるステークホルダを確認する．
- 製品のエンドユーザを明確し，ユーザグループ別の特徴を確認する．
- ステークホルダ別に対応しなければならない制約条件をリストアップする．

④製品構成を確認する

- ハードウェア構成とその制約．
- 利用するOS，ミドルウェアなどを明確にし，それぞれの制約をリストアップする．
- 製品が連携動作する周辺のソフトウェア，システムやハードウェアなどとのインタフェースを明確にする．

⑤再利用ソフトウェアを確認する

- 既存ソフトウェアを再利用するか否かを検討する.
- 再利用する場合は, 再利用ソフトウェアの仕様や特徴, ならびに再利用の方針を確認する.

⑥ソフトウェアの開発環境, テスト環境, 導入環境を確認する

- 開発に利用するツール
- テスト環境, テスト用ツール, テスト方法, テストデータの利用可能性
- インストール時の制約なども明らかにしておく.

- ソフトウェア機能要求事項の明確化
  - 概要

    開発対象ソフトウェアに求められる機能要求事項を検討し，ソフトウェア機能要求リストとしてまとめる．
  - 実施内容

    ①ソフトウェア機能要求の検討の考慮点
    - 製品仕様書やシステム要求およびシステム方式設計書（あれば）を参考に，ソフトウェアとして実現する機能を明確に切り出しておく．
    - ソフトウェア機能に関連するハードウェア機能，プラットフォームなども明確にしておく．
    - ユースケース分析ではユースケース・シナリオ，ユースケース図，アクティビティ図などを作成する．
    - 要求漏れがないことを確認する．
    - 異常時を十分考慮していることを確認する．

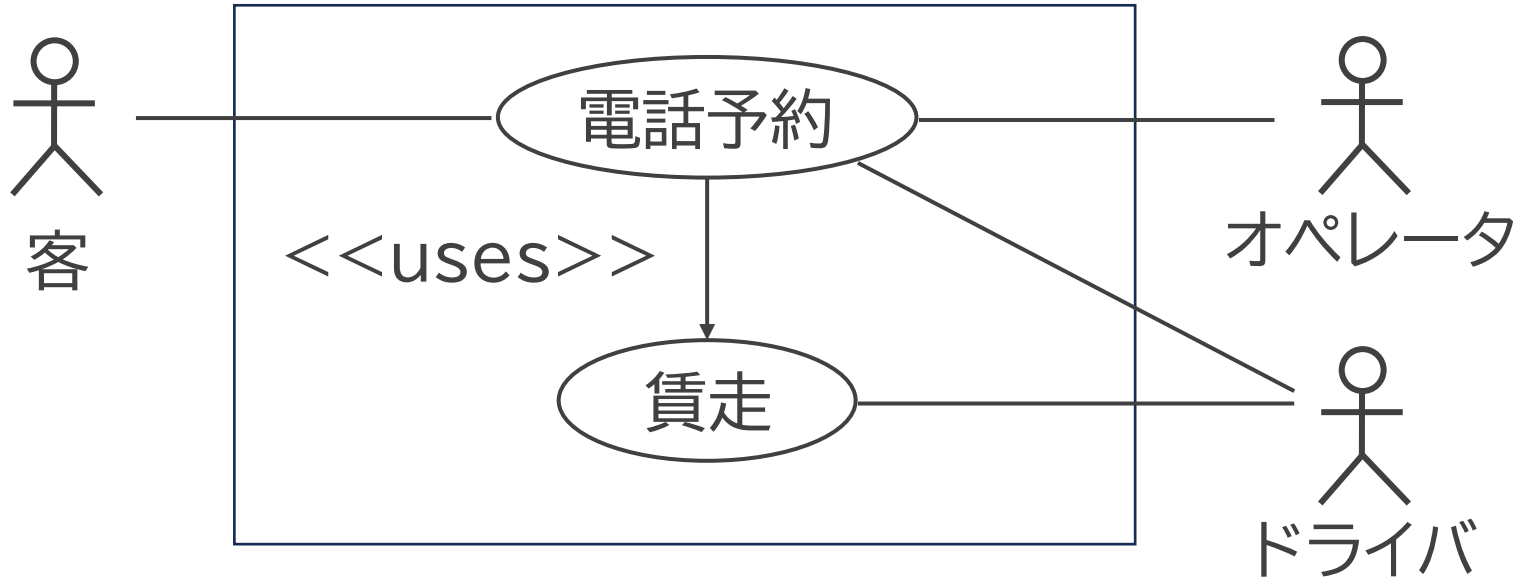
②安全性に関して要求されるソフトウェア機能の例

- フェールセーフのための機構
- 故障時におけるデータ保護

補足：ユースケース図とユースケース記述

ユースケースとは，システムが外部のユーザや外部システムに提供するサービスのことである．ユースケース図は，ユーザから見たシステムの利用の仕方を記述するための記法である．

「タクシー電話予約システム」のユースケース図の例で，客，オペレータ，ドライバがアクタ，電話予約と賃走がユースケースである．ユースケース同士にも関係があり，電話予約は賃走を利用している．



補足：ユースケース図とユースケース記述について（続き）

ユースケース記述は，ユースケース毎に，アクタが発するイベントを契機として始まるサービスを，アクタとシステムとの間のやりとりとして，自然言語で記述したものである．ユースケース記述の例は以下に示す．

「電話予約」ユースケース
客が配車センターに電話をし，タクシーを予約する．このとき，オペレータは客から名前と現在位置および目的地を聞く．オペレータは，この情報を端末に打ち込む．情報が打ち込まれると，タクシーの中から優先順位決定アルゴリズムによって決定した優先順位のタクシーから順に，予約を受け付けるかどうかを問い合わせる．問い合わせを行うと，簡易端末に客の名前，現在位置と目的地が表示される．タクシーのドライバは，その予約を受ける場合には了解ボタンを，そうでない場合には拒絶ボタンを押す．10秒以上応答がない場合には，拒絶とみなされる．了解の場合には，予約が確定する．拒絶の場合には，次の優先順位の高いタクシーを選択し，問い合わせを進める．これは，了解となるまで続けられる．
予約が確定すると，タクシーは流し中ならば直ちに客の現在位置へ向かう．そうでない場合には，先に確定している客を目的地に連れて行った後に，客の現在位置に向かう．本人であることを確認したあとで客を乗せ，以後「賃走」に移る．
例外：了解するタクシーがない場合には，その旨を客に告げ，予約を消去する．

- ソフトウェア非機能要求事項の明確化
  - 概要

    開発対象ソフトウェアに求められる非機能要求事項を検討し，ソフトウェア非機能要求リストとしてまとめる．
  - 実施内容

    ①信頼性要求

    - 特定のシステム動作条件において，ハードウェアあるいはソフトウェアの想定外の動作が必要になる状況などを検討し，システムの異常処理方式を決めておく．
    - システムにとって望ましくない状況が発生した際にも，システムとしての最低限の機能は動作するよう，ソフトウェア面での工夫も検討しおく．
    - システムの異常動作モードからの復帰手順や復帰方式を明確にしておく．

②使用性要求

システムの多くは不特定多数のエンドユーザを対象とする場合が多いことを念頭にソフトウェアで実現する部分の操作性などを検討する（システム全体としてのユーザインタフェースの統一感など）

③効率性要求

- システムの実行性能（例：処理速度，起動時間，応答時間）について考慮する．
- システムのハードウェアに起因するリソース効率（例：メモリ容量，データサイズ）に注意する．システムで扱うデータの生存期間なども考慮に入れる．

④保守性要求

フィールドトラブルが発生した場合のトラブル原因の解析を可能とするための機構の検討や動作ログ情報の記録メカニズムなど保守の方式とその実現方法も検討しておく．

⑤移植性要求

ハードウェアやOSなどの変更に伴うソフトウェアの移植しやすさなども考慮する．また既存ソフトウェアの一部を利用する場合も想定し，ソフトウェアユニットの独立性を予め考慮しておく．

⑥その他の非機能要求

- セキュリティ要求（例：データ暗号化，ユーザ認証，ウイルス対策）
- 相互運用性（例：通信プロトコル）
- 外部インタフェース要求（例：連携ソフトウェアとの関数インタフェース，通信プロトコル，ユーザインタフェース）
- データ定義

- 要求の優先順位付け
  - 概要

    ソフトウェア要求の優先順位を決定する.

  - 実施内容

    以下いくつかの視点を踏まえて, 個々の要求の開発リスクを考慮して実現可能性を評価しながら, 必須／高／低／任意といった優先順位をつけた根拠や理由も記録しておく.

    - 開発プロジェクトの期間や投入コスト, リソースなどの視点
    - 新規の技術導入や技術習熟などの視点
    - ビジネス面からの要求

- ソフトウェア要求仕様書の作成
  - 概要

    ソフトウェアの要求事項を整理し，ソフトウェア要求仕様書としてまとめる．
  - 実施内容
    - ソフトウェア要求定義の過程で出力された資料を整理・体系化して文書化する．
    - 内部レビュー等での指摘事項が適切に反映されていることも確認する．
    - 作成された文書は構成管理および変更管理にて確実に管理する．

B）ソフトウェア要求仕様の確認

- ソフトウェア要求仕様書の内部確認
  - 概要

    ソフトウェア要求仕様がソフトウェアとして求められる事項を満たしているかを確認し，内部レビュー報告書としてまとめる．
  - 実施内容

    以下の①～⑦の視点で，ソフトウェア要求仕様書の内部確認を行う．確認結果は内部レビュー報告書として整理し，確認作業で指摘された問題およびその対応を明記した上で関係者に配布する．

    ①妥当性を評価する

    - 記述されている要求事項が製品仕様やシステム要求などと照らし合わせて妥当であるか

②実現可能性を評価する

- 記述されている要求事項は開発部門が保有する技術によって実現可能かの評価

③テスト可能性（ソフトウェア要求はテスト可能か）を評価する

- システムが想定する動作フィールドにおいてのテストや動作確認が可能な要求であるか

④運用・保守性を評価する

- ソフトウェアのバージョンアップの考え方やその方式は適切であるか
- 不具合があった場合の対処方法が適切に検討されているか

⑤追跡可能性を評価する

- 個々のソフトウェア要求事項に識別情報が付けられているか
- 個々の要求事項に関して，元になる成果物が明示されているか

⑥一貫性を評価する

- 関連する成果物の内容と矛盾がないか，また，ソフトウェア要求仕様書の内部で矛盾がないか

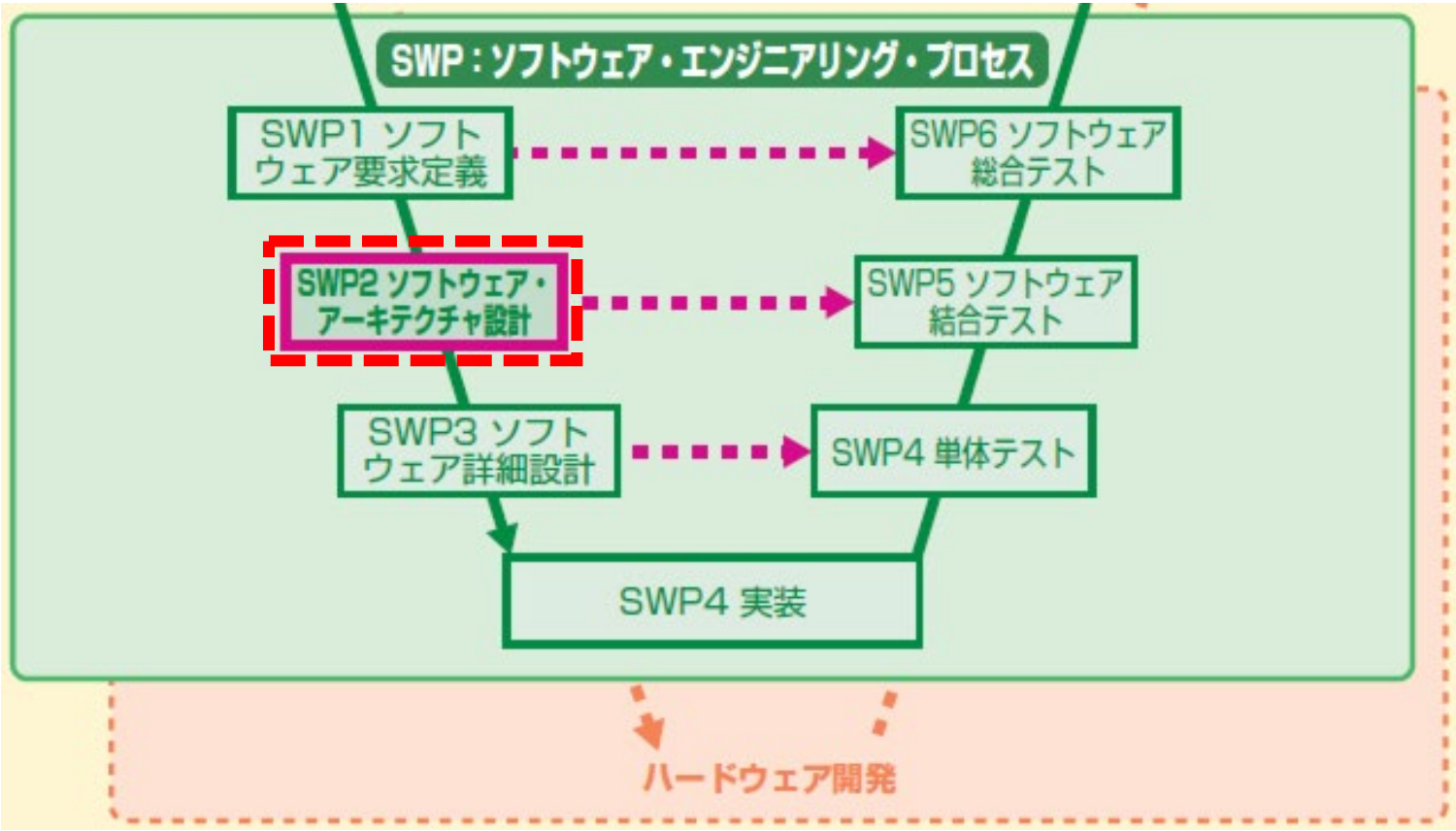
⑦完全性を評価する

- 機能，性能，設計上の制約，属性，外部インタフェースなど，必要事項がすべて記述されているか
- システムの動作において想定可能なあらゆるコンテキストに対して，その条件下でのソフトウェアの振る舞いが定義されているか
- 要求事項は一意にしか解釈できない記述になっているか

2. 初期設計（＝アーキテクチャ設計，方式設計）

2.1 目標

・ソフトウェアのアーキテクチャ（＝動作（振る舞い）と構造）を決定し，ソフトウェア方式設計書を作成する．



2.2 入力，出力およびタスク構成

入力	タスク構成	出力
ソフトウェア要求仕様書	A) ソフトウェア方式設計書の作成 • 設計条件の確認 • ソフトウェア構成の設計 • ソフトウェア全体の振る舞いの設計 • インタフェースの設計 • ソフトウェア方式設計書の作成 B) ソフトウェア方式設計の確認 • ソフトウェア方式設計書の内部確認とレビュー	ソフトウェア方式設計書， 内部レビュー報告書（ソフトウェア方式設計）

A）ソフトウェア方式設計書の作成

- 設計条件の確認
  - 概要

    ソフトウェア・アーキテクチャを設計するにあたっての要求・条件を確認する.
  - 実施内容

    ソフトウェア要求仕様書に記述された機能要求，非機能要求や制約事項などを確認しておく.

    ①ソフトウェア機能要求を確認する

    ➢ どのような機能をソフトウェアとして実現することが求められているか確認する.

    ②ソフトウェア非機能要求を確認する

    ➢ ソフトウェア要求仕様書に記載されたソフトウェア非機能要求を確認する.

③制約条件を確認する

- ソフトウェア要求仕様書に記載されたソフトウェア機能を再考し，それらの機能を実現するためのソフトウェア・アーキテクチャの実現という視点から，下記のような制約条件も併せて確認する．

  - ソフトウェア作成条件（使用OS，言語など）
  - ハードウェア条件
  - 性能条件，稼働環境など
  - 安全性要求

- ソフトウェア構成の設計
  - 概要

    ソフトウェアの構成および機能ユニットの機能を設計する.
  - 実施内容

    ソフトウェア要求仕様を実現するために必要となる, ソフトウェアとしての個別要素（機能ユニット）を検討し, 整理する.

    ①機能ユニットを抽出する

    - ソフトウェアとして実現する機能を明確にし, 共通する処理など局所化, 共通化を図りながら機能ユニットを抽出する.
    - 信頼性（リカバリ／データ保証など）
    - 保守性（不具合追跡など）

    ②機能ユニットを詳細化する

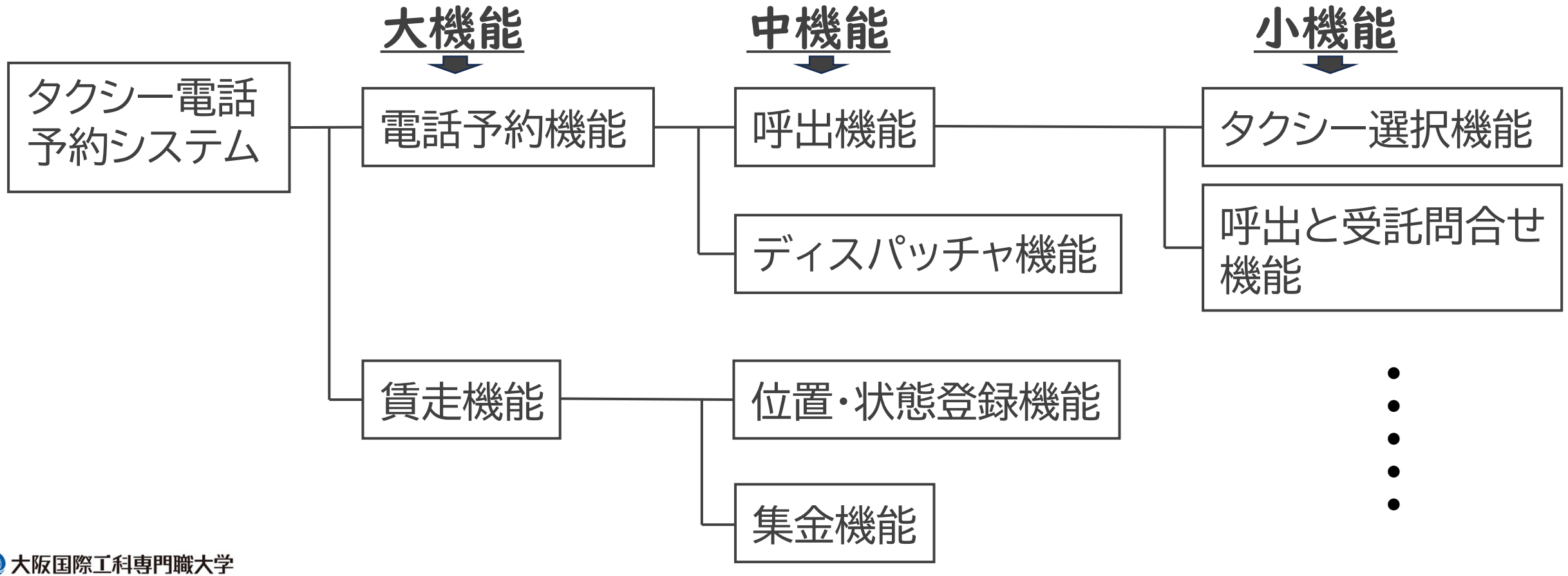
    - 個々の機能ユニットは, ソフトウェア詳細設計ができるレベル（粒度）まで構成を詳細化する.

- 設計の進め方
  - 各ソフトウェア詳細設計担当者に割り振れるレベルまで機能ユニットを詳細化する
  - 振る舞い，インタフェース設計と連携して実施する
  - 機能ユニットの位置／関係が分かる構成図（機能ユニット関連図）を作成する
  - 保守（デバッグ，テスト）のための動作ログ／取り出し機構を具備する
  - フェールセーフ機構を具備する

補足：機能ユニット関連図（機能展開図）

ソフトウェアが提供する機能を，大機能，中機能，小機能という形で階層的に分析・整理したものである．

タクシー電話予約システムの機能ユニット関連図の例



- ソフトウェア全体の振る舞いの設計
  - 概要

    ソフトウェア全体の振る舞い（動作）を設計する．
  - 実施内容

    ハードウェアを含めたシステムがどのような動的振る舞いをするかを考え，整理する．

    - システムの振る舞いのなかで，ソフトウェアが担う動作を明確にする．
    - ソフトウェア要求仕様に記載された動作シナリオや動作シーケンスを検討する．特に具体的な動作に伴うシステムとしての動作シーケンスの操作性も考慮する．
      - タスク・プライオリティと並行処理／タスク遷移など
      - データ処理の流れ：入力から出力までのデータ処理過程
      - 異常・例外発生：過負荷，ハード障害など

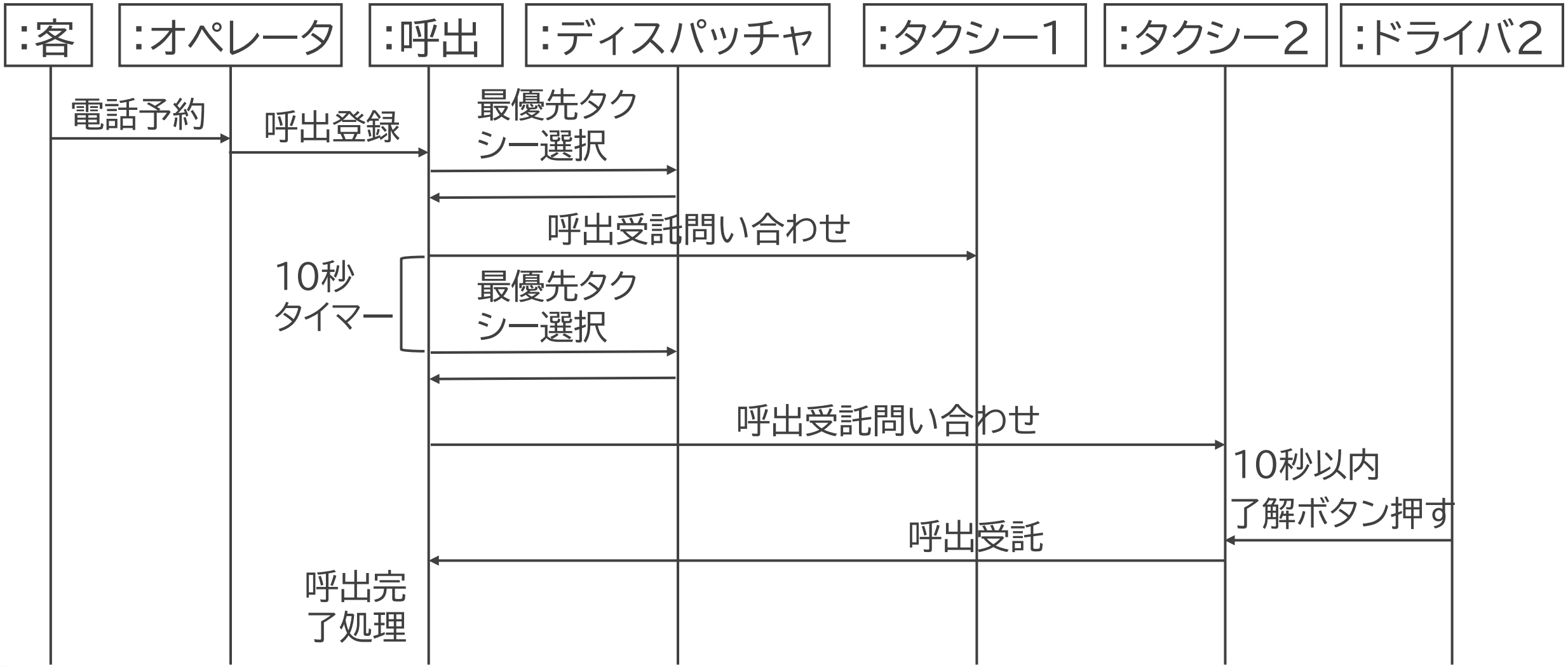
補足：シーケンス図

シーケンス図は，1つのユースケースに関する1つのシナリオを，オブジェクト間のメッセージのやり取りによって表現する図法である．

シーケンス図は，ユースケース記述を基にして作成することができる．ユースケース記述と同様に，アクタからの動作/イベントを契機として，開始してからシステムの応答が一通り完了するまでを記述する．シーケンス図では，オブジェクト間のやり取りを時間軸に沿って記述することで，システムの動的にな側面を表現でき，システムが実行される流れを細かく記述することができる．

補足：シーケンス図（続）

タクシー電話予約システムのシーケンス図の例



- インタフェースの設計
  - 概要

    機能ユニット間のインタフェースを設計する.
  - 実施内容

    ソフトウェアを構成する機能ユニット間のインタフェースを設計する.

    ➢ 呼び出し手順, パラメータ, 復帰情報等を設計する.
    ➢ 変更に対して修正負荷の高い情報は, 一元化し, かつ論理値化しておく.
      - テーブル・オフセット
      - 共通定数など

- ソフトウェア方式設計書の作成
  - 概要

    ソフトウェア・アーキテクチャ設計の事項を整理し，ソフトウェア方式設計書としてまとめる．
  - 実施内容

    前記各ステップの成果を取りまとめてソフトウェア方式設計書を作成する．

    - ソフトウェア・アーキテクチャ設計の過程で出力された資料を整理・体系化して文書化する．
    - 内部確認，レビュー等での指摘事項が適切に反映されていることも確認する．
    - 作成された文書は構成管理および変更管理にて確実に管理する．

B）ソフトウェア・アーキテクチャ設計の確認

- ソフトウェア方式設計書の内部確認
  - 概要

    ソフトウェア・アーキテクチャ設計がソフトウェアの要求事項を満たし，かつ，機器／システムとしての動作が製品仕様を満たしているか確認し，内部レビュー報告書としてまとめる．

  - 実施内容

    ①ソフトウェア方式設計書の内容が適切であるかどうかを確認する

    - 構成要素である機能ユニットの集合が，製品仕様，ソフトウェア要求を正しく実現できるかを確認する．
      - 機能ユニットの機能の明確性・妥当性
      - 機能ユニットの集合として振る舞いの明確性・妥当性
      - 機能ユニット間のインタフェースの明確性・妥当性

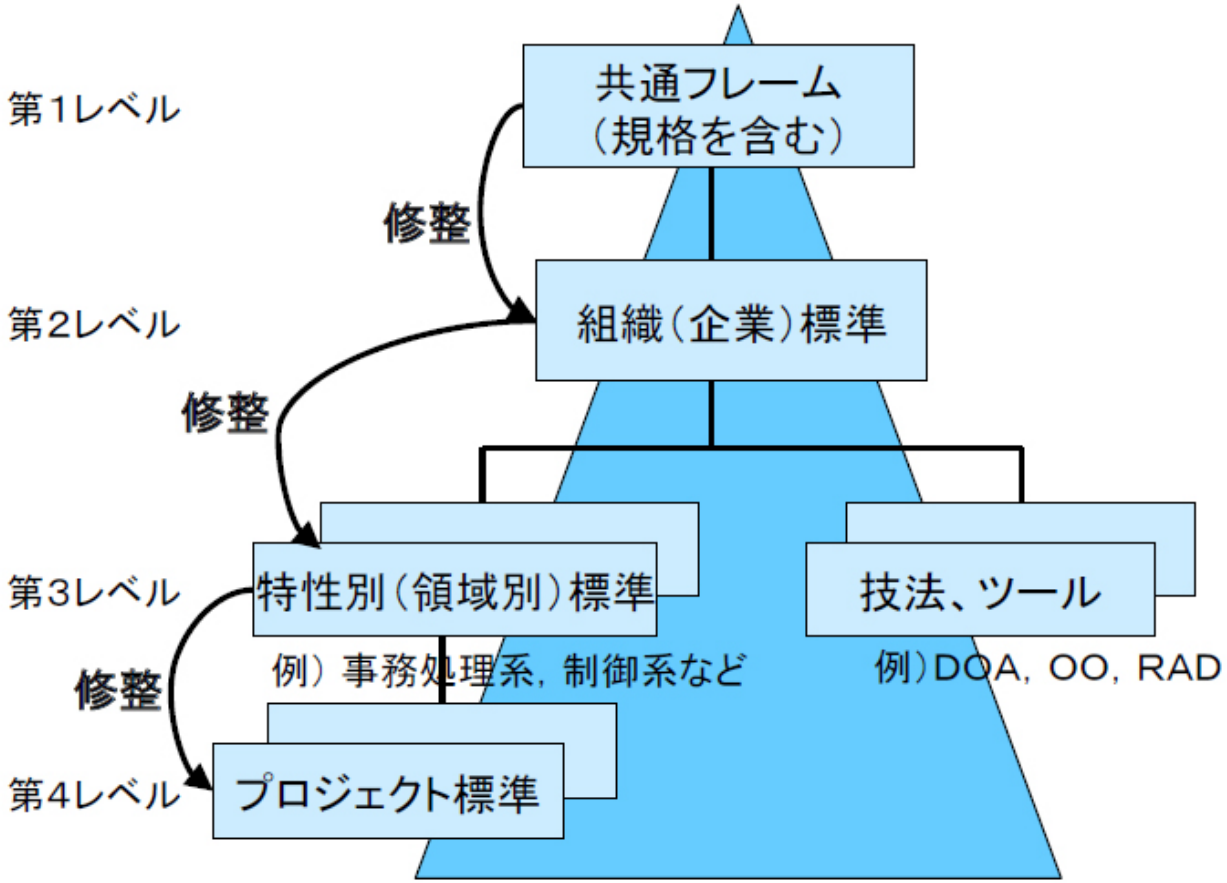
- ソフトウェア非機能要求を十分に反映し，運用／テスト／保守の実現が可能かを評価する．
  - 使用者のレベルを考慮した分かりやすさと安全性が考慮されているか
  - 稼働状態（性能，負荷ピーク，長時間稼働）の考慮がされているか
  - テスト／保守性の容易性が考慮されているか

②機能ユニットの詳細化設計を確認する

- 下記の基準を考慮して，個別のソフトウェア・アーキテクチャ設計を評価する．
  - 機能，インタフェース，振る舞いの妥当性
  - 具体化レベルと詳細設計の実現可能性
  - 設計標準への準拠（設計手法／表記法／用語／名称の体系化）
  - 既存ソフトウェア／市販ソフトウェア／オープンソフトウェアを利用する場合の十分な調査
  - 他プラットフォームへの移植性，機能拡張性の容易性

③ソフトウェア要求との対応（トレーサビリティ）が取れているか確認する

2.3 成果物のテーラリング



（注1）DOA ： データ中心のアプローチ
（注2）OO ： オブジェクト指向の方法論，技法など
（注3）RAD ： 短期間アプリケーション開発技法

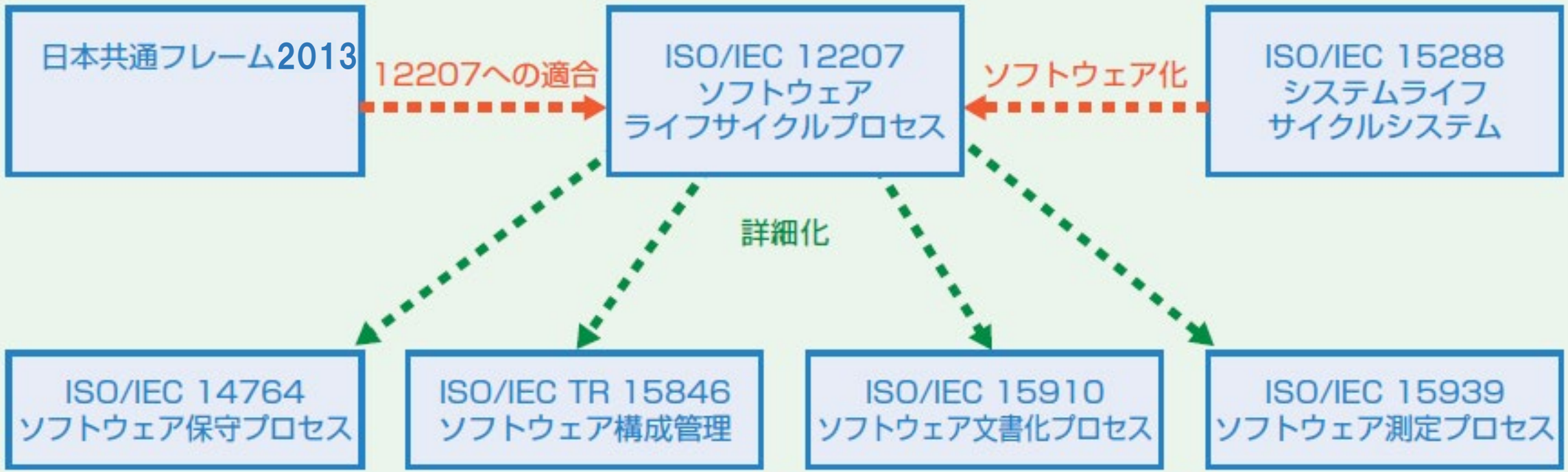
■修整（テーラリング）とは、

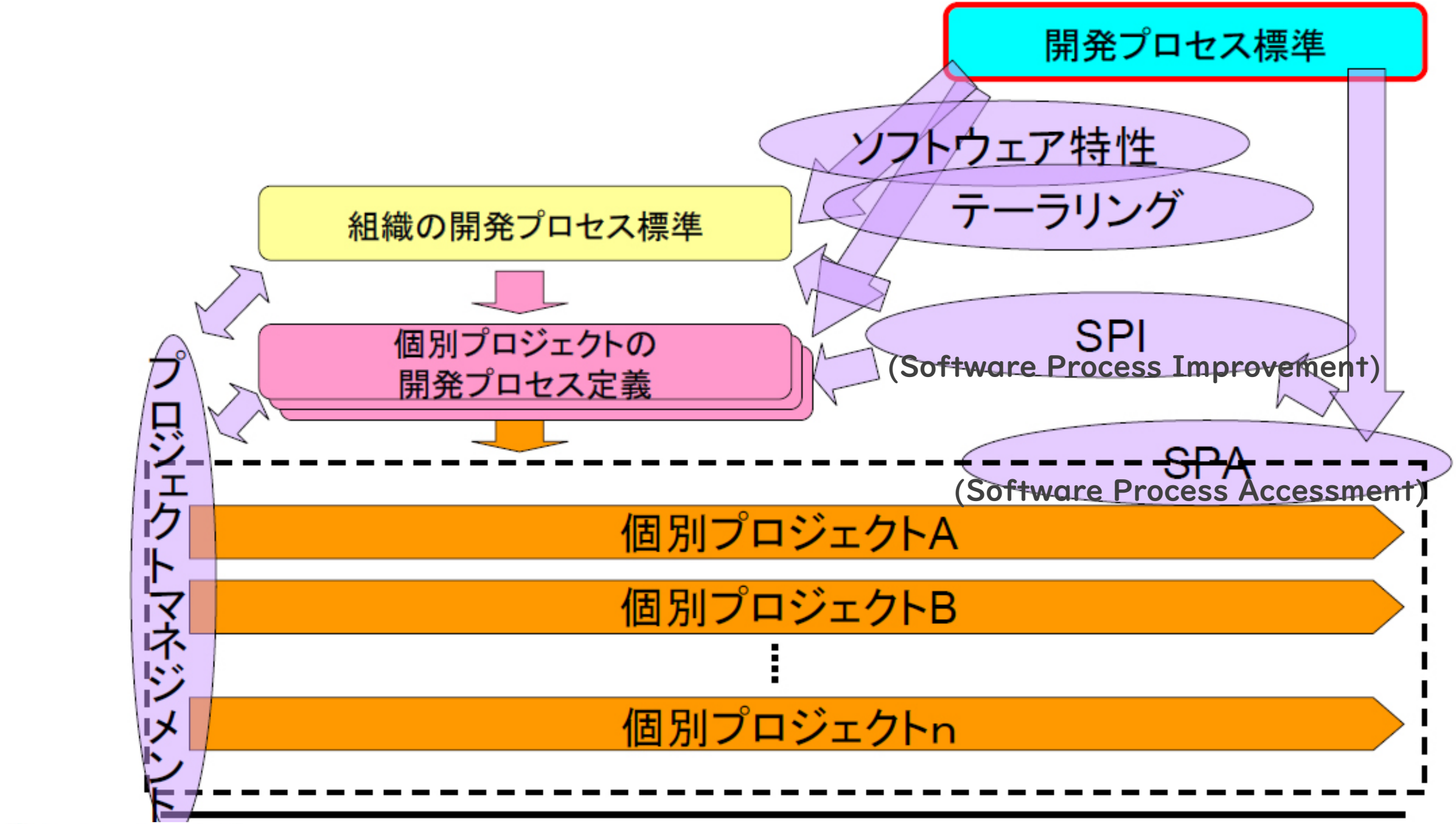
共通フレームをそのまま適用するのではなく、組織（企業）やプロジェクトの特性（例えば開発モデル）に合わせて、共通フレームで規定されているプロセス／アクティビティ／タスクを取捨選択したり、繰り返し実行できるように、又は複数を一つに括って実行できるように組み替えたりする作業をいう。

この修整（テーラリング）プロセスが、共通フレームに含まれていることによって、共通フレームの適用が、非常に柔軟なものとなる。

Related Standards

開発プロセス関連規格




開発プロセス標準
ソフトウェア特性
テーラリング
組織の開発プロセス標準
個別プロジェクトの
開発プロセス定義
SPI
(Software Process Improvement)
SPA
(Software Process Accessment)
プロジェクトマネジメント
個別プロジェクトA
個別プロジェクトB
個別プロジェクトn

例：開発プロセス標準に規定されているソフトウェア・アーキテクチャ設計

入力	タスク構成	出力
ソフトウェア要求仕様書 システム方式設計書	A) ソフトウェア方式設計書の作成 • 設計条件の確認 • ソフトウェア構成の設計 • ソフトウェア全体の振る舞いの設計 • インタフェースの設計 • ソフトウェア方式設計書の作成 B) ソフトウェア方式設計の確認 • ソフトウェア方式設計書の内部確認とレビュー • ソフトウェア方式設計書の外部レビュー	ソフトウェア構成設計書， 機能ユニット設計書， ソフトウェア動作設計書， ソフトウェア・インタフェース設計書 ソフトウェア方式設計書， 内部レビュー報告書と外部レビュー報告書（ソフトウェア方式設計）

この演習（プロジェクト）で採用したソフトウェア・アーキテクチャ設計プロセス

入力	タスク構成	出力
ソフトウェア要求仕様書	A) ソフトウェア方式設計書の作成 • 設計条件の確認 • ソフトウェア構成の設計 • ソフトウェア全体の振る舞いの設計 • インタフェースの設計 • ソフトウェア方式設計書の作成 B) ソフトウェア方式設計の確認 • ソフトウェア方式設計書の内部確認とレビュー	ソフトウェア方式設計書， 内部レビュー報告書（ソフトウェア方式設計）

演習

1. 飲食店の注文・会計管理システムのユースケース図を作成しなさい.
2. 飲食店の注文・会計管理システムの機能ユニット関連図を作成しなさい.
3. 飲食店の注文・会計管理システムのシーケンス図を作成しなさい.

4. 飲食店の注文・会計管理システムのソフトウェア要求仕様書とソフトウェア方式設計書を完成しなさい.（授業外の時間で）
   （配布したテンプレートファイルをベースとする. なお, 作成したユースケース図を要求仕様書に, 機能ユニット関連図, シーケンス図を方式設計書にそれぞれ取り込む）