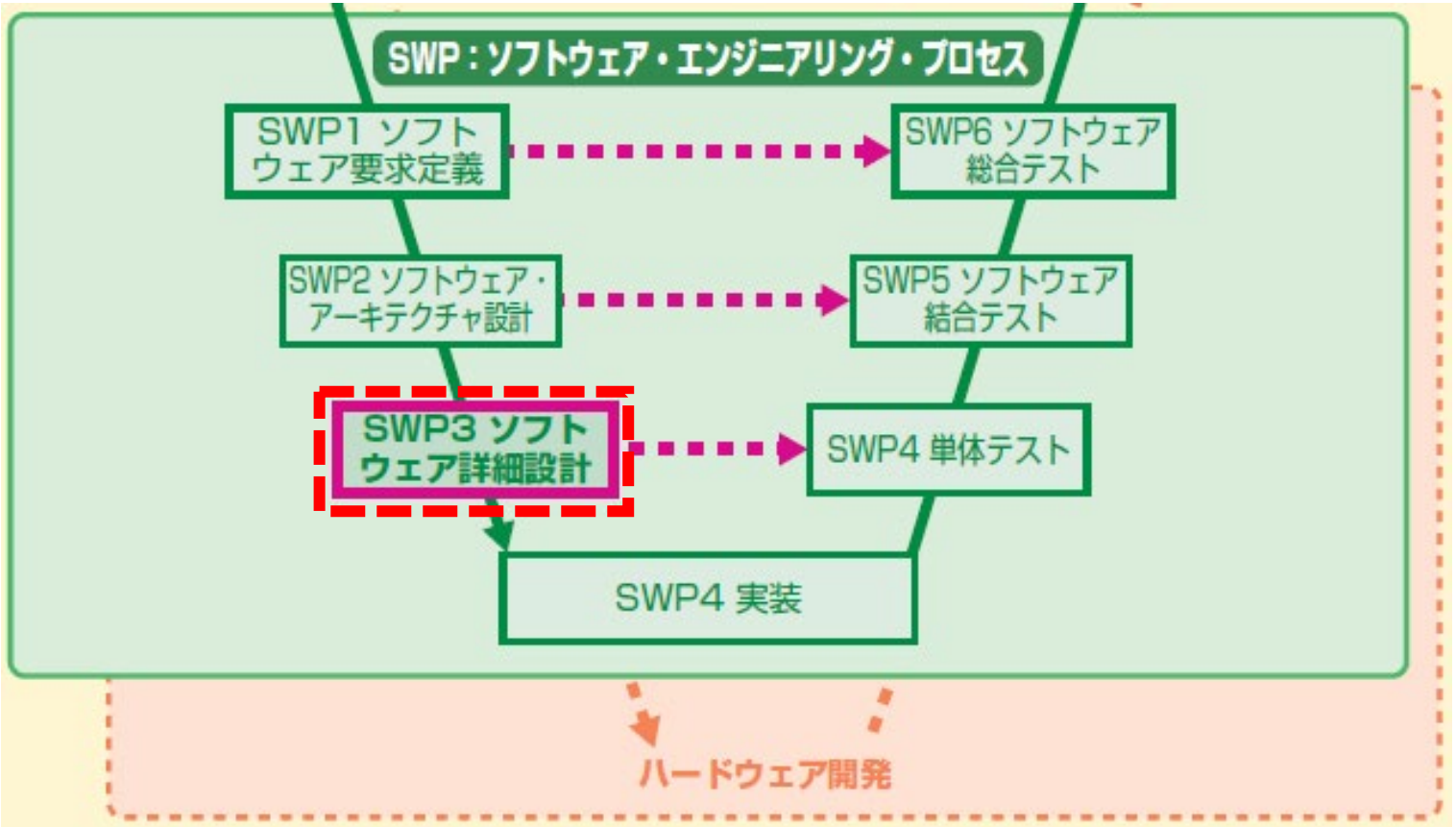
ソフトウェアシステム開発

第3回： 開発プロセス（論理設計）

3. 論理設計（＝詳細設計）

3.1 目標

・ソフトウェア・アーキテクチャ設計で定義された機能ユニットをプログラムユニットに分割し，詳細な振る舞いや論理構造などを設計する．



3.2 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 ソフトウェア方式設計書	A) 機能ユニット詳細設計書の作成 • プログラムユニット分割 • プログラムユニット設計 • インタフェースの詳細化 • ソフトウェア詳細設計書の作成 B) ソフトウェア詳細設計の確認 • ソフトウェア詳細設計書の内部確認	ソフトウェア詳細設計書 レビュー報告書（ソフトウェア詳細設計）

A）ソフトウェア詳細設計書の作成

- プログラムユニット分割
  - 概要

    機能ユニットをプログラムユニットに分割する.

  - 実施内容

    機能ユニットをプログラムユニットに分割し，プログラムユニットの構成と各々の機能を定義する

    - プログラムユニットは，以下の点を考慮して，実装・コンパイルおよび単体テストを実施する最小レベルまで分割する.
      - データフロー（データを処理する流れ）に着目する
      - 複数のプログラムユニットに共通の機能を洗い出して，共通機能として定義する
      - プログラムユニットの独立性（カプセル化）や相互の結合度などに注意する

- プログラムユニット設計
  - 概要

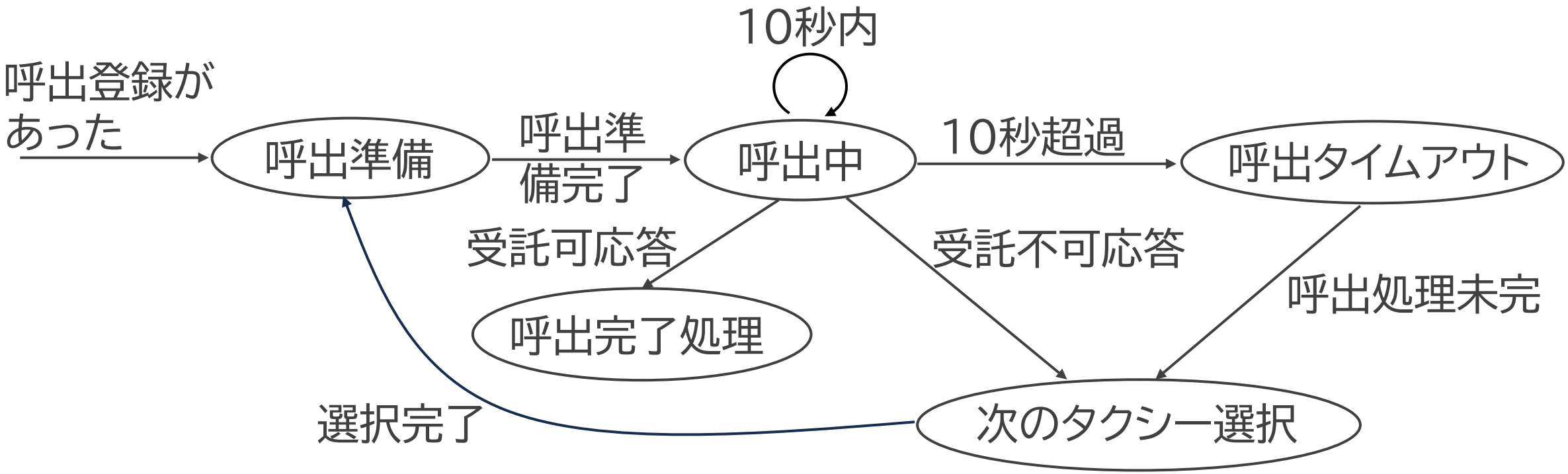
    プログラムユニットの処理内容を，実装可能なレベルまで詳細化する．
  - 実施内容

    ①プログラムユニットの処理内容を実装可能なレベルまで詳細化する
    - 初期設定，制御方法，タイミング
    - 状態管理（状態遷移表/図等で，入力イベントに対してソフトウェアの振る舞いを決定するための状態を管理する）
    - システムコール，利用機能（汎用ライブラリ，共通機能等）の引数値
    - リソース定義（ユニット内で使用するリソース）
    - エラー処理
    - システム初期化処理

補足：状態遷移図

状態遷移図は，着目する対象がとり得る状態の集まりをノードの集まりとして列挙し，それらの状態間の移動・遷移をアークで記述し，遷移を起こす条件，入力，あるいはイベントをそのアークの上に記述するチャートである．

下図は，「タクシー電話予約システム」の「呼出と受託問合せ機能」の状態遷移図の例で，各状態と状態間の遷移を示す．



②不具合を解析する際に利用する機能もプログラムユニットとして詳細内容を検討する

- ハードウェア制御時の状態（特にエラー発生時の情報）
- ソフトウェア実行状態の確認

- 注意すべき事項
  - 共通に使用する値（テーブル，バッファ，エラー値，コンパイル条件など）を抽出し，具体的な数値を定める．
  - 非同期で動作するプログラムユニット間の排他，イベント等を考慮する．
  - ユーザインタフェースを実現するプログラムユニットについては，ユーザの操作性（ユーザの誤操作をまねくことがないよう）などを考慮する．
  - 製品シリーズ化（プロダクトライン）などを考慮し，ソフトウェアのコアの部分を構成するプログラムユニット，製品バリエーションに対応するプログラムユニットなどを明確に区分する．

- インタフェースの詳細化
  - 概要

    機能ユニット間およびプログラムユニット間のインタフェースを，実装可能なレベルまで詳細化する．
  - 実施内容

    ①機能ユニット間インタフェースを詳細化する

    ➢ アーキテクチャ設計で定義されたインタフェース仕様を実装可能なレベルまで詳細化する．

    - アーキテクチャ設計で定義されたインタフェース仕様を実装可能なレベルまで詳細化する．
    - 機能ユニットの入力
    - 機能ユニットの出力
    - 参照・設定するデータ（テーブル，バッファ等）

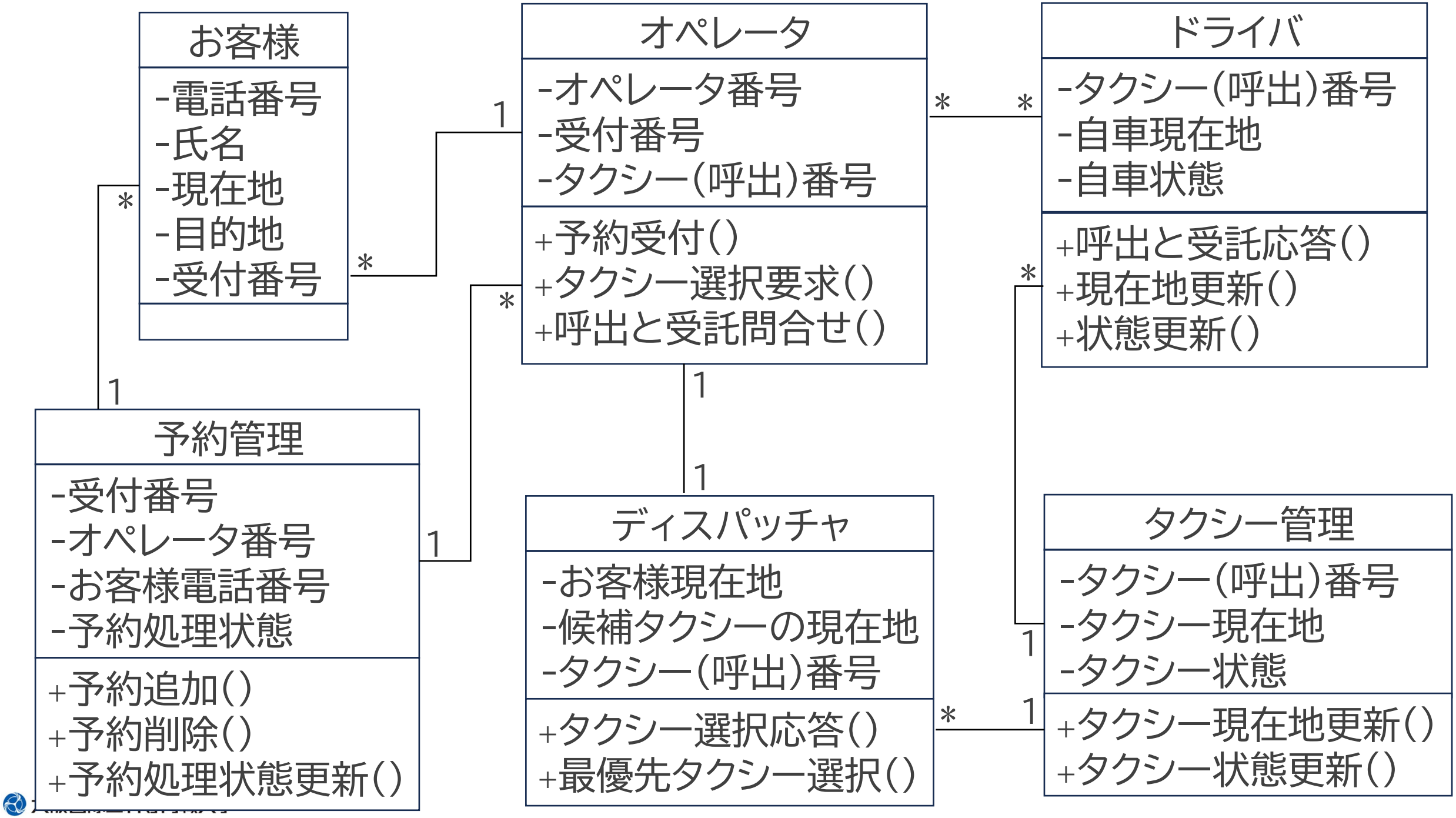
②プログラムユニット間インタフェースを設計する

- インタフェースを実装可能なレベルまで詳細化する.
  - プログラムユニットの入力
  - プログラムユニットの出力
  - 参照・設定するデータ（テーブル, バッファ等）
  - クラス図の作成

補足：クラス図

クラス図は，システムの論理的，静的な構造を表す．システム内に存在するあらゆるオブジェクトとその関係としてとらえる．クラスには属性や操作があり，クラス間には関係がある．なお，属性の型，操作の引数と戻り値の型，属性や操作の公開/非公開など，プログラミングレベルの詳細を記述することもできる．

次のページに，「タクシー電話予約システム」のクラス図の例を示す．


お客様
-電話番号
-氏名
-現在地
-目的地
-受付番号
オペレータ
-オペレータ番号
-受付番号
-タクシー(呼出)番号
+予約受付()
+タクシー選択要求()
+呼出と受託問合せ()
ドライバ
-タクシー(呼出)番号
-自車現在地
-自車状態
+呼出と受託応答()
+現在地更新()
+状態更新()
予約管理
-受付番号
-オペレータ番号
-お客様電話番号
-予約処理状態
+予約追加()
+予約削除()
+予約処理状態更新()
ディスパッチャ
-お客様現在地
-候補タクシーの現在地
-タクシー(呼出)番号
+タクシー選択応答()
+最優先タクシー選択()
タクシー管理
-タクシー(呼出)番号
-タクシー現在地
-タクシー状態
+タクシー現在地更新()
+タクシー状態更新()
1
*

- ソフトウェア詳細設計書の作成
  - 概要

    ソフトウェア詳細設計の事項を整理し，ソフトウェア詳細設計書としてまとめる．
  - 実施内容
    - ソフトウェア詳細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，ハードウェア仕様との整合性の確認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）ソフトウェア詳細設計の確認

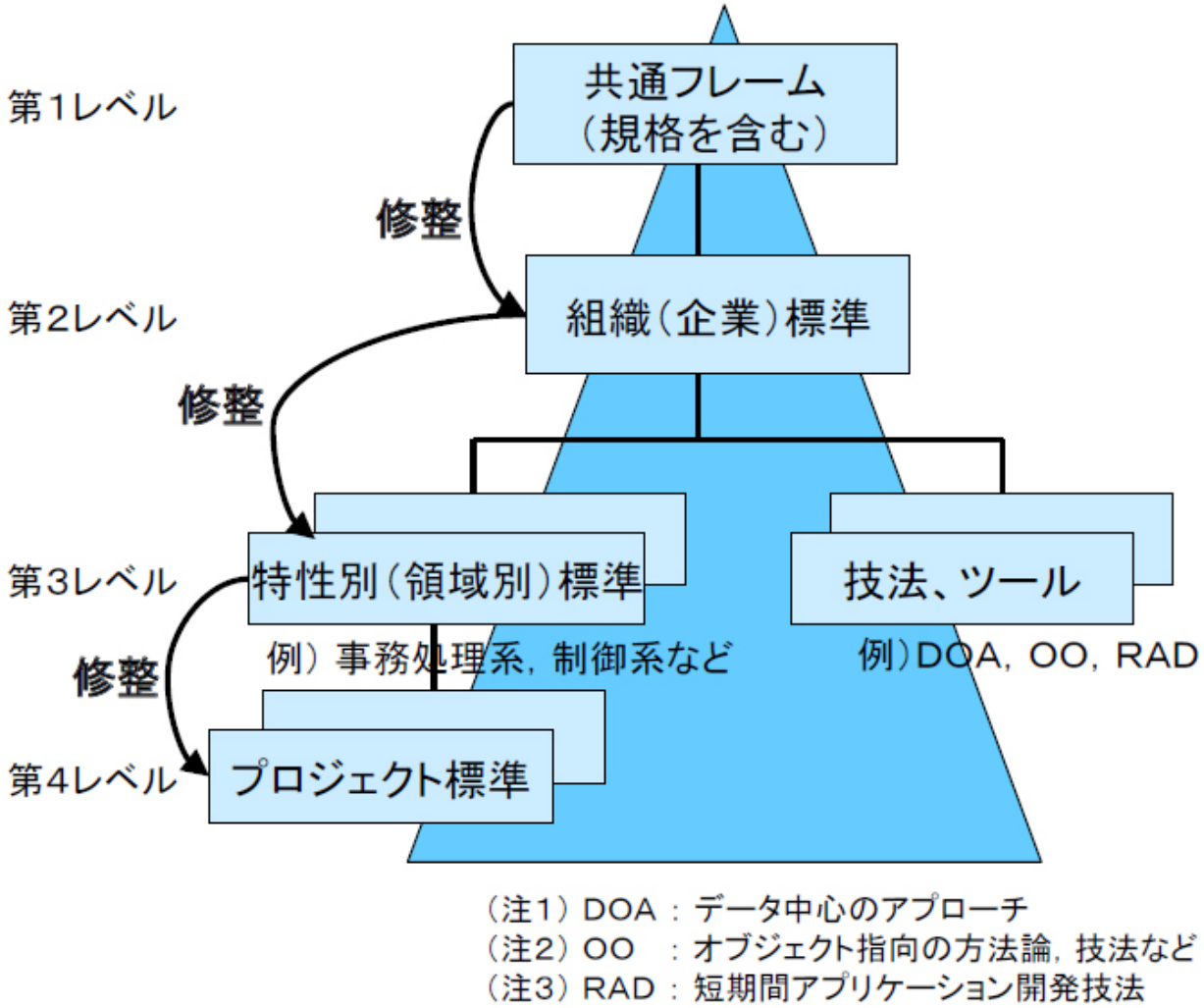
- ソフトウェア詳細設計書の内部確認
  - 概要

    ソフトウェア詳細設計がソフトウェア・アーキテクチャの設計事項を満たし，かつ，実装可能なレベルか確認し，内部レビュー報告書としてまとめる．
  - 実施内容

    ①下記の視点でソフトウェア詳細設計書の内容を確認する
    - 個々のユニットの切り分けの正しさ，および各ユニットの処理内容の明確さ
    - ユニット間のインタフェース情報の整合性
    - 個々のユニットとハードウェアとの関係
    - 各種設定値，引数値（OSシステムコール，汎用ライブラリなど）
    - 無限ループ，デッドロック（発生する条件がないか確認する）

    ②確認結果は内部レビュー報告書として整理し，指摘事項およびその対応元を明記する

3.3 成果物のテーラリング（おさらい） ー 詳細設計フェーズでも



■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。

例：開発プロセス標準に規定されているソフトウェア詳細設計プロセス

入力	タスク構成	出力
ソフトウェア要求仕様書 ソフトウェア方式設計書	A) 機能ユニット詳細設計書の作成 • プログラムユニット分割 • プログラムユニット設計 • インタフェースの詳細化 • ソフトウェア詳細設計書の作成 B) ソフトウェア詳細設計の確認 • ソフトウェア詳細設計書の内部確認	プログラムユニット機能／構成設計書 プログラムユニット設計書 プログラムユニット・インタフェース設計書 ソフトウェア詳細設計書 レビュー報告書（ソフトウェア詳細設計）

この演習（プロジェクト）で採用したソフトウェア詳細設計プロセス

入力	タスク構成	出力
ソフトウェア要求仕様書 ソフトウェア方式設計書	A) 機能ユニット詳細設計書の作成 • プログラムユニット分割 • プログラムユニット設計 • インタフェースの詳細化 • ソフトウェア詳細設計書の作成 B) ソフトウェア詳細設計の確認 • ソフトウェア詳細設計書の内部確認	ソフトウェア詳細設計書 レビュー報告書（ソフトウェア詳細設計）

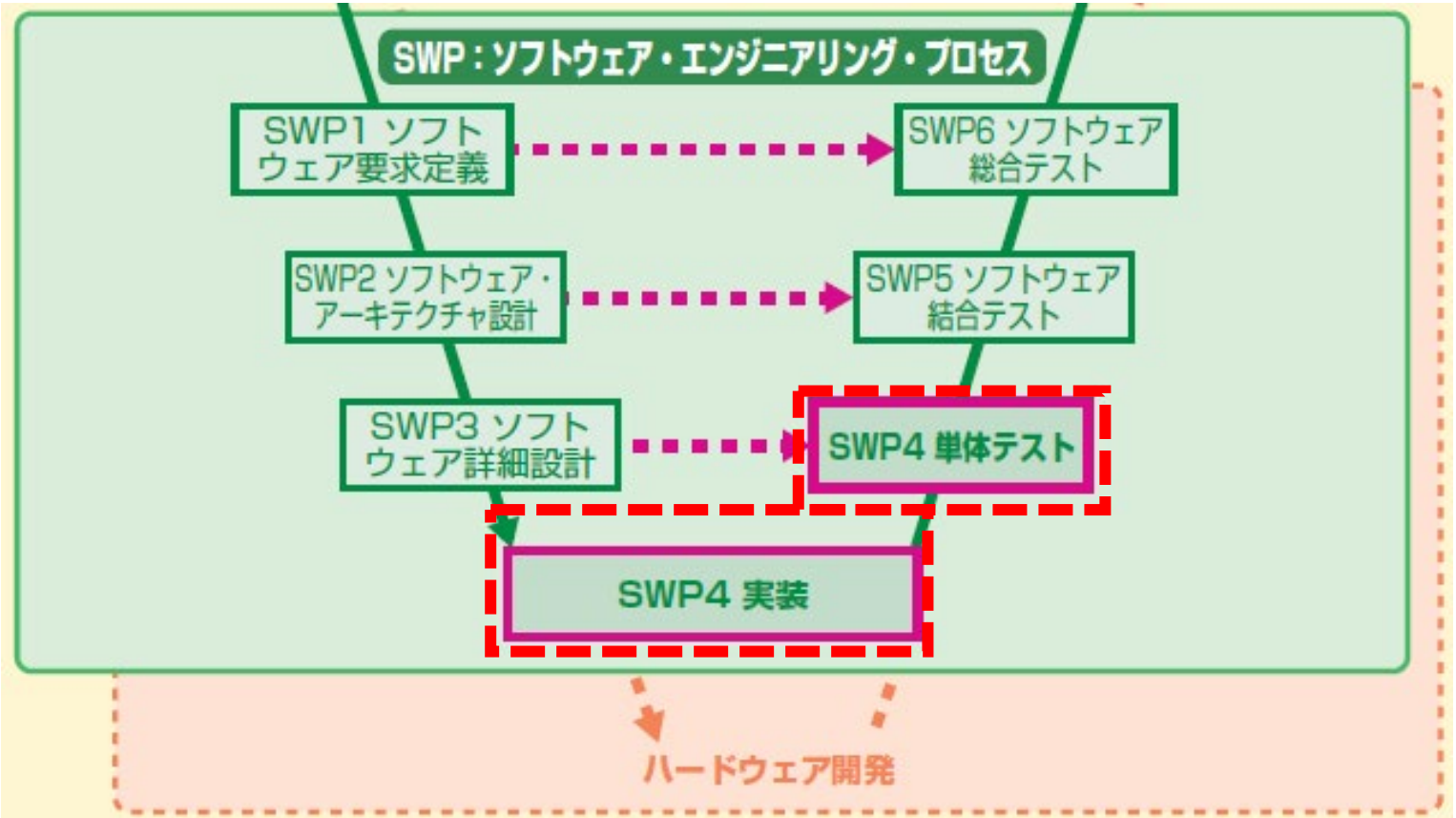
ソフトウェアシステム開発

第4回： 開発プロセス（実装）

4. 実装（プログラミング～単体テスト）

4.1 目標

・ソフトウェアを構成する個々のプログラムユニットの実装と単体レベルでの動作確認を行う.



4.2 入力，出力およびタスク構成

入力	タスク構成	出力
ソフトウェア詳細設計書	A) 実装および単体テストの準備 • 実装の準備 • 単体テストの準備 B) 実装および単体テストの実施 • プログラムユニットの実装 • 単体テストの実施 • 単体テスト結果の確認 C) 実装および単体テスト結果の確認 • ソースコードの確認 • 単体テスト結果の内部確認	プログラムユニット 単体テスト仕様書 単体テスト報告書 レビュー報告書（実装・単体テスト）

A) 実装および単体テストの準備

- 実装の準備
  - 概要

    開発環境や再利用するプログラムなどを準備し，プログラムユニットの実装が実施可能な状態にする．
  - 実施内容

    ①開発環境を準備する

    ➢ プログラムユニットを実装するための開発環境を準備する．

    ②再利用するプログラムユニットを準備する

    ➢ 開発済みのプログラムユニットを利用する場合は，該当するプログラムユニットを用意し，利用可能状態にあるか確認する．

- 単体テストの準備
  - 概要

    単体テスト仕様書を作成し，単体テストが実施可能な状態にする．
  - 実施内容

    ①単体テスト項目を準備する

    ➢ ソフトウェア詳細設計書に定義された処理が，正しく実現されているか否かを確認するために以下のようなテスト項目を準備し，テスト仕様としてまとめる．
    - 機能テスト
    - 条件網羅テスト／データ境界値テスト

    ②テストデータを準備する

    ➢ 前記テスト項目を実行するために必要となるテストデータ（具体的な入力データなど）を作成する．

③スタブ／テストドライバ（擬似ソフトウェア）を作成する

- テスト対象の機能ユニットを動作させるため，必要に応じてスタブ／テストドライバを作成する．

④テスト結果の判定基準や，テスト全体の評価基準や完了基準なども用意しておく

⑤修正確認テスト項目を準備する（修正確認の場合）

- テストで不具合などが検出され，その部分を修正した場合，不具合が解消されたこと，また修正により別の不具合が発生していないことを確認するためのテスト項目を準備する．
  - 不具合の内容および修正内容などを考慮し，テスト範囲を決定しテスト項目を選択する

B）実装および単体テストの実施

- プログラムユニットの実装
  - 概要

    プログラムユニットを実装する．
  - 実施内容

    ①プログラムユニットを実装する

    ➢ ソフトウェア詳細設計書に基づいてプログラムユニットを実装する．

    - コーディング規約等を守る．
    - 読みやすさやテストのしやすさに留意する．
    - コメントを記入して処理内容が分かるようにする．

    ➢ コンパイルを実行して，文法上の誤りがあれば修正する．

②利用プログラムユニットを確認する

- 既存のソフトウェア資産や商用のソフトウェアなどを利用する場合には，それらのソースコードなどを確認し，利用する範囲を決める．

③不具合を修正する

- ソースコードの確認やテストなどで不具合が検出された場合には，不具合管理票を確認してプログラムユニットの不具合を修正する．
  - 不具合修正の場合は，修正履歴，修正内容が分かるようにする．
  - 「実装」以降については，ソースコードの版管理（チェックイン・アウト）／構成管理に留意する．

- 単体テストの実施
  - 概要

    単体テスト仕様書に従って，単体テストを実施する．
  - 実施内容

    ①単体テストを実施する
    - 単体テスト仕様書をもとに，単体テストを実施し，出力結果を得る．
    - 不具合検出時，継続して残りのテストを実施するか，不具合が修正されるまでペンディングするかの判断をする．
    - 出力結果を収集する（各種ログやスクリーンショットなど）．

    ②修正確認テストを実施する
    - 単体テストなどで検出された不具合を修正した場合には，不具合が解消されたかテストを実施する．
    - 不具合の修正により，別の不具合が発生していないかテストを実施する．

- 単体テスト結果の確認
  - 概要

    テスト結果を確認し，テストの合否を判定する．
  - 実施内容

    ①単体テスト結果を確認する
    - テスト結果を確認して，実施したテスト項目の合否を判定する．
    - 修正確認テストの結果，不具合が解消された場合は，修正確認が済んだことを不具合管理票に記録する．

    ②不具合記録を作成する
    - 不具合を発見した場合，不具合管理票に不具合内容を記録する．

C）実装および単体テスト結果の確認

- ソースコードの確認
  - 概要

    ソフトウェア詳細設計書に記載されている機能を実現する上で正しく実装（コーディング）されているか確認する.
  - 実施内容

    個々のプログラムユニットを実現するソースコードが正しく実装されているかどうかを確認する

    - ソフトウェア詳細設計書に記載されている機能が実現できるか確認する.
    - 採用しているコーディング作法やコーディング規約などとつき合わせて以下のようなものを確認する.
      - コーディング規約等を守っているか
      - 読みやすい，テストしやすいコーディングになっているか
      - コメントを記入して処理内容が分かるようにしているか

- 単体テスト結果の内部確認
  - 概要

    単体テストで未解決の問題や未実施のテスト項目がないか確認し，内部レビュー報告書としてまとめる．
  - 実施内容

    ①下記の視点で単体テストの結果を確認する
    - 未解決となっている問題がないか
    - 未実施となっているテスト項目がないか

    ②確認結果は内部レビュー報告書として整理する（確認作業で指摘された問題およびその対応元を明記）

演習

1. 飲食店の注文・会計管理システムの状態遷移図を作成しなさい.

2. 飲食店の注文・会計管理システムのクラス図を作成しなさい.

3. 飲食店の注文・会計管理システムのソフトウェア詳細設計書を完成しなさい.（授業後）
（配布したテンプレートファイルをベースとする. なお, 作成した状態遷移図とクラス図を取り込む）