- 組込みソフトウェアをどのような手順で開発していくのか（開発プロセス）を説明します
- 開発プロセスの中でモデルをどのように使うかを示しながら説明します

2. ソフトウェアの開発の進め方とモデルの活用

2-1. 実際の開発現場での制約と問題

- 実際の開発では、様々な制約があります
  - 実際の開発現場では、期間・開発環境・人員が限られた中で開発する
  - 最初に仕様を確定するのは困難で、開発中に仕様変更が発生する
  - 技術的な課題の発生や人員不足で予測通りに開発が進む事は極めて少ない
  - 過去の資産の継承で、試行錯誤の結果のソースコードだけが残っている状態で開発し、ソフトウェアの設計図がない事が多い
- なりゆきで開発していくと以下のような問題が起きます
  - 期限までに機能が実現できなかったり、作業に追われるうちに機能の織り込み忘れや必要な変更の対応漏れが発生する
  - 多発する変更に対して、より短い時間での対応に迫られる状況に陥りやすいので、暫定処置的な処置を積み重ねて複雑化する
    - ⇒　現物合わせで試行錯誤したソフトウェアになる　[1-2参照]
  - 設計図がなくソフトウェア全体が把握できない状態で開発し、思わぬところで不具合が発生する

2-2. 順序立てて開発する

- 構想からソフトウェアで実現するまで段階的に具体化します
  - いくつかの工程を踏んで具体的にしていくことで、やるべき事を網羅しつつ整理しながら開発するようにする
- ソフトウェアを２段階に分けて設計します
  - まず、ソフトウェアの部品の構造と振舞いを考える
    - ユースケースや機能の仕様から必要な部品を考える
    - 開発メンバがわかり易い言葉で表現する（日本語表記など）
  - 次にソフトウェアの部品の実装の仕方を考える
    - 使用するプログラミング言語での表現に変える（命名ルールに従った英語名など）
    - プラットフォーム（OSやドライバなど）や使用するライブラリとつなぐ
    - ソフトウェアの動かし方（タスク設計など）を決める
- 段階的にテストして、ミスによる大きな変更を抑えます

2-3. 開発の工程

開発の工程と作業目的(工程でやること)

- 作る時は、全体を俯瞰して部品の構造・振舞いを考えてから具体化する
- 作ったものの動きを確認する時は、小さい単位(範囲)からチェックして、動かす範囲を広げていく

実現したいことを抽象的、ソースコードを具体的として段階的に具体化する

問題発見時の解析・修正がし易いように段階的に範囲を広げて振舞いを検査する

抽象的 → 具体的

広い → 狭い

設計し作成する工程			作成したものを検証する工程	
開発工程	やること		開発工程	やること
要求分析	実現したい事(機能や性能)を決める	検査	システムテスト	システムが期待通りに動作するか検証する
基本設計	アーキテクチャ(部品の構造・振舞い)と機能の実現方法を決める	検査	結合テスト	部品間のやり取りが正しく動作するかを検査する
詳細設計	基本設計の部品の構造・振舞いを実装できるようにする	検査	単体テスト	部品が正しく動くかどうかを検査する
実装	詳細設計の部品の構造・振舞いをコーディングする	検査	デバッグ	コンパイルエラーをチェックし修正する

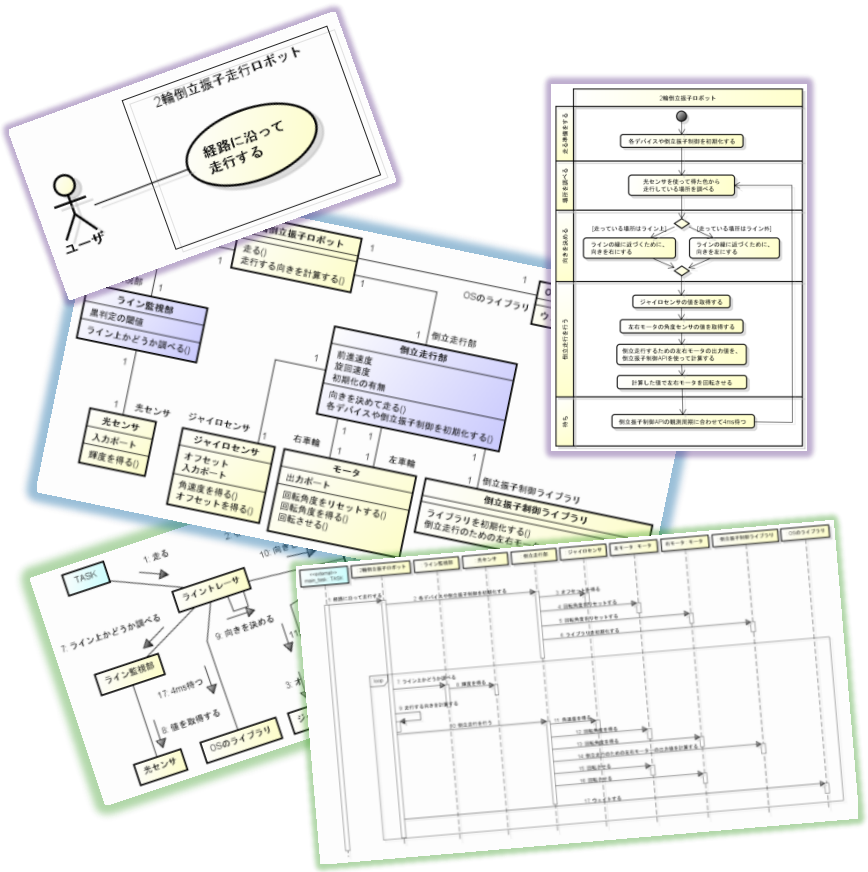
2-4. 設計内容を成果物として残し活用する

- 開発の工程ごと設計結果を残し、次の工程に受け渡します
  - 各工程ごとに検討した内容をドキュメントにする
  - 各工程の成果物が次の工程の入力として必要十分な内容が記載されるようなドキュメントを作り、受け渡す
    - ⇒ 設計内容のトレーサビリティが取れるようにする

- 開発メンバが活用できる、わかりやすい成果物を作ります
  - ソフトウェアを表すにはモデルを使うのが効果的
  - 文書だけでなく、ソフトウェアの設計図となるモデルを作る

2-5. モデルの表記（1）

- UML
  - Unified Modeling Language
  - ソフトウェア開発など利用する汎用的なモデリング言語
  - ISOやJISの規格にもなっているモデリング言語の標準的存在
  - ソフトウェアや業務を表す多くの図（記法）を提供しています



- 部品を組合わせたソフトウェアの開発に適しています
  - 部品を図の要素として表現する
  - 部品の組合わせを、図の要素のまとまりやつながりとして表現する

2-5. モデルの表記（2）

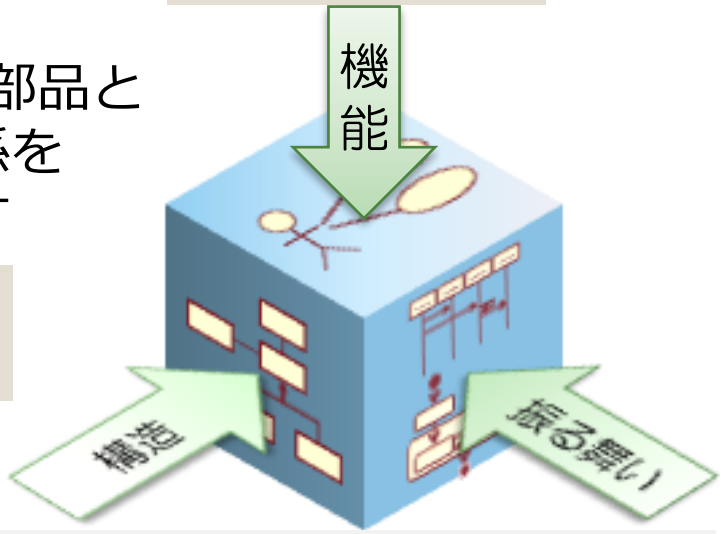
- UMLでは、3つの視点から対象をモデル化します
  - 機能・構造・振舞い
- よく使われている7種類のモデル図

【機能の視点】
システムが利用者に提供すべき
働きやサービスをモデル化します

ユースケース図

【構造の視点】
機能の実現に必要な部品と
部品どうしの関係を
モデル化します

オブジェクト図
クラス図



モデルの視点とよく使われるUML図の種類の関係

【振舞いの視点】
機能を実現するための
部品の使い方や
部品内部の動きを
モデル化します

シーケンス図
コミュニケーション図
ステートマシン図
アクティビティ図

2-6. UMLの図の使い分け

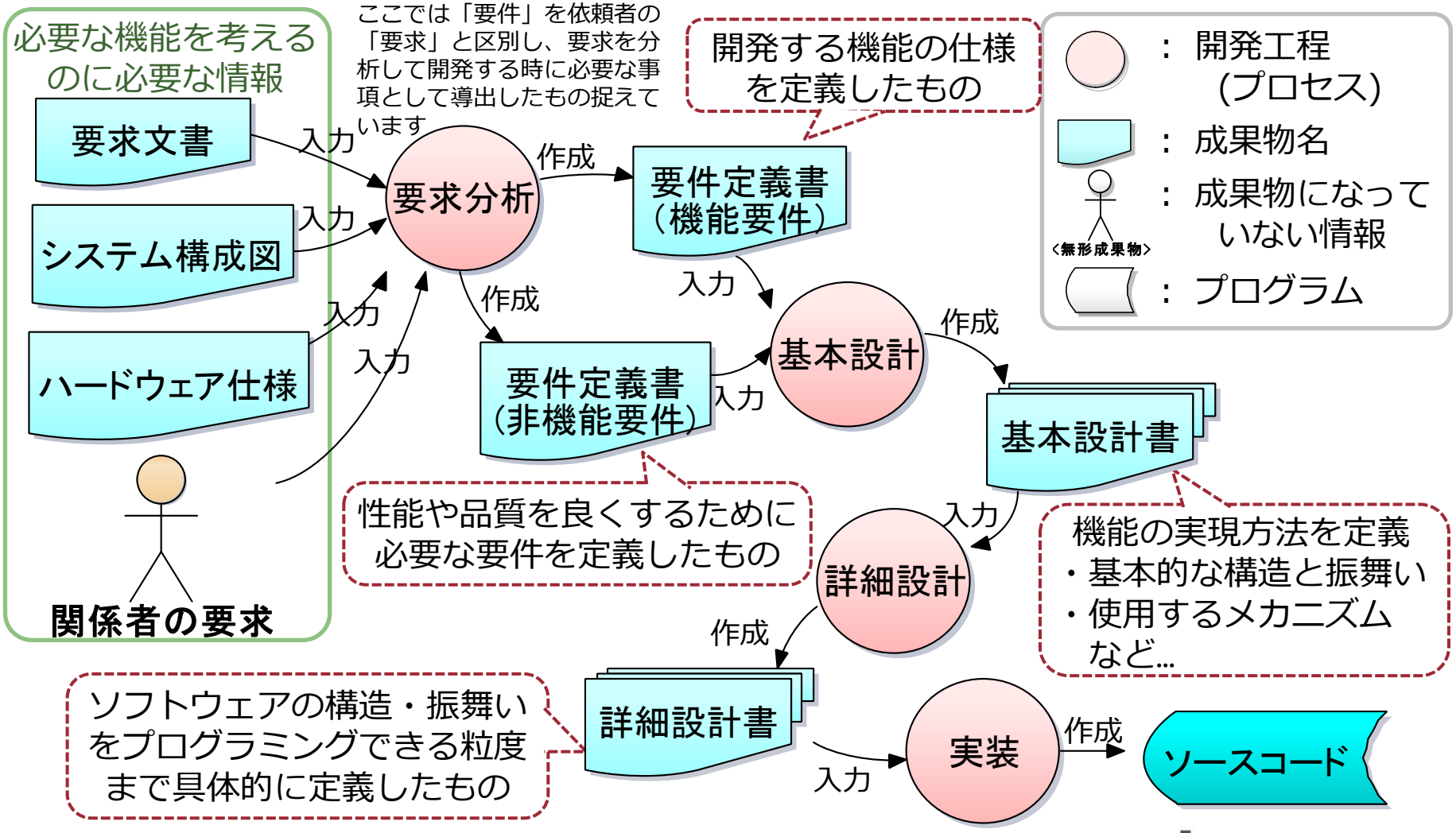
- UMLを使うモデリングでは、次の表のように図を使い分けることが多いです

順序	関心事	よく使うモデル図	モデルの役割	スコープ
1	システムの機能	ユースケース図	モデリングの対象となるシステムが提供する機能を記述します	システム（部品を意識しません）
2	システムの振舞い	アクティビティ図 ステートマシン図	機能を実現するための制御フローや、状態に応じたシステムの動作を記述します	
3	システムが動くときの構造	オブジェクト図	機能を実現するために必要な部品を定義します	システム（部品を意識します）
4	システムの構築に必要な構造	クラス図	オブジェクト図で記載された各部品の仕様を定義します	
5	システムの内部の要素間の振舞い	コミュニケーション図 シーケンス図	機能を実現するための、部品どうしの協調動作を記述します	
6	システムの内部のある要素の振舞い	アクティビティ図 ステートマシン図	部品の操作に対する制御フローや、状態に応じた部品内部の動作を記述します	部品

モデリングの関心事とよく使うUMLの図の役割とスコープ

2-7. 各開発工程で必要な成果物

- 実装までの各開発工程で作成する成果物と流れを示します



開発工程と成果物のつながり

2-8. 工程の成果物でのモデルの活用例

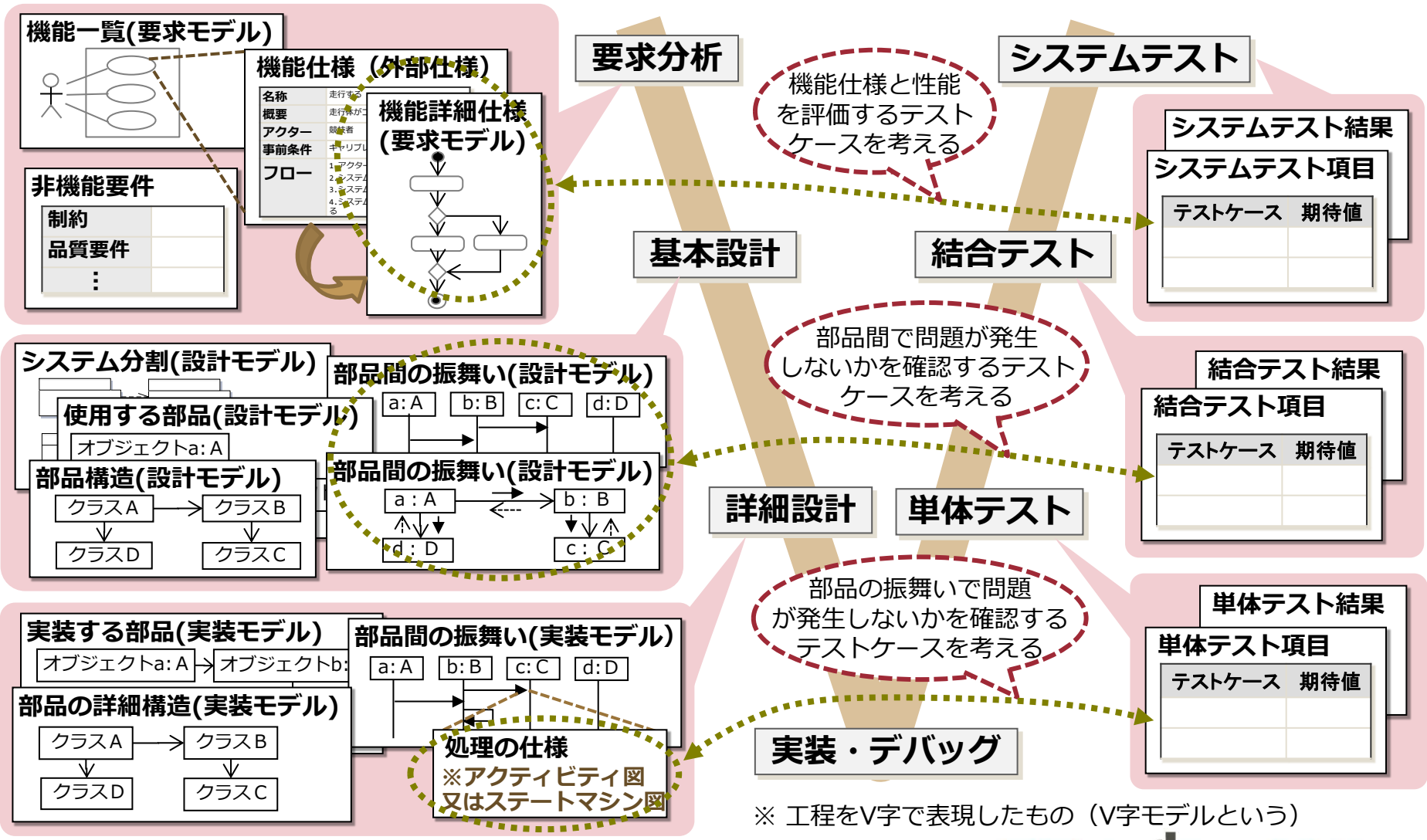
■ 各工程の検討内容に合わせて次の図の組合わせが使われることが多いです

工程	工程の成果物の要件	成果物内で使用されるUMLのモデル図の例
要求分析	【文書名： 要件定義書】 ● システムが持つ機能とその機能の仕様が明確になっている事 ● 制約や性能・品質の要件が明確になっている事	※機能図（ユースケース図） 機能一覧(要求モデル) 非機能要件 制約 品質要件 機能仕様（外部仕様） 名称 / 概要 / アクター / 事前条件 / フロー 機能詳細仕様 (要求モデル) 規模が大きい時 又はSysMLの要求図など ※フロー図 (アクティビティ図) 機能要件
基本設計	【文書名：基本設計書】 ● 全体の構造が明確になっている事（規模に応じて階層的に構造を示す） ● 各部品の責務と部品間の振舞いが明確になっている事	※構造図（パッケージ図） システム分割(設計モデル) ※構造図 (オブジェクト図) 使用する部品(設計モデル) オブジェクトa:A ※振舞い図（コミュニケーション図） 部品間の振舞い(設計モデル) a:A / b:B / c:C 部品構造(設計モデル) クラスA / クラスB / クラスD / クラスC 部品間の振舞い(設計モデル) a:A / b:B / c:C / d:D 処理の仕様 ※アクティビティ図 又はステートマシン図 ※構造図（クラス図） ※振舞い図（シーケンス図）
詳細設計	【文書名： 詳細設計書】 ● ファイル構成・データ定義・関数仕様(又はクラス仕様)がわかる事 ● 部品の起動方法がわかる事	※構造図(オブジェクト図) 実装する部品(実装モデル) 部品の詳細構造(実装モデル) クラスA / クラスB / クラスD / クラスC オブジェクトb:B オブジェクトc:C ※振舞い図（シーケンス図） 部品間の振舞い(実装モデル) a:A / b:B / c:C / d:D 処理の仕様 ※アクティビティ図 又はステートマシン図 ※構造図（クラス図）

開発工程の要件とよく使うモデル図

2-9. 各工程の設計内容ごとに動作を検証

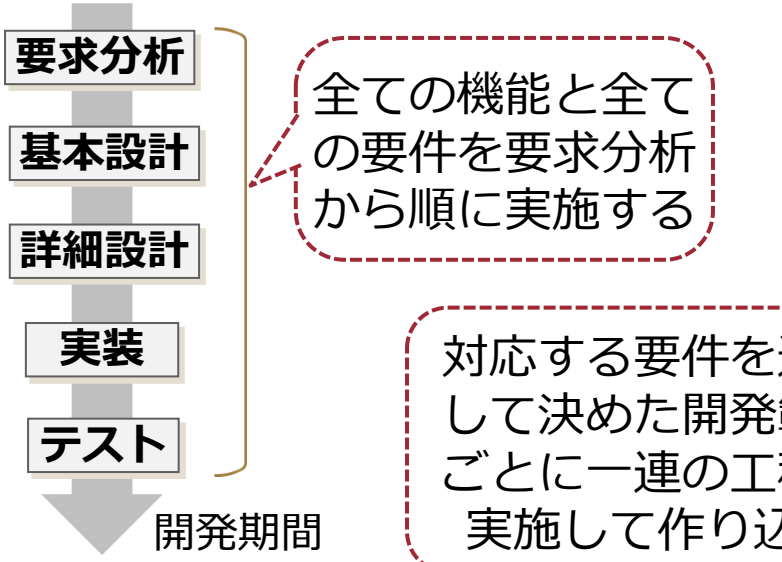
- プログラムが各工程で設計した内容通りに動作するかを検証します



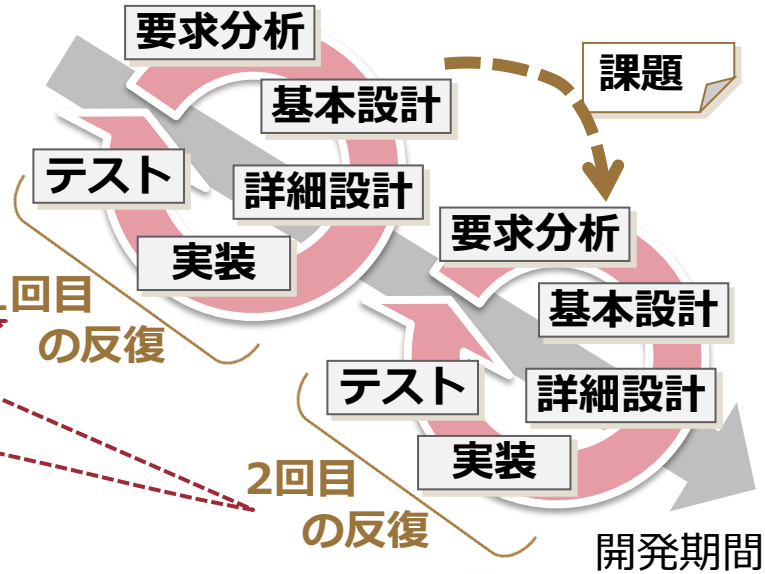
※ 工程をV字で表現したもの（V字モデルという）

2-10. 状況に合わせて開発工程をまわす

- 開発工程(プロセス)の進め方にはパターンがあります
  - 想定される技術課題・作業負荷に対して開発メンバの構成や期間などの制約を考えて開発工程の進め方を選択し、開発計画を立てる
- 開発工程の進め方のパターン(プロセスモデル)の例
  - ウォーターフォール型



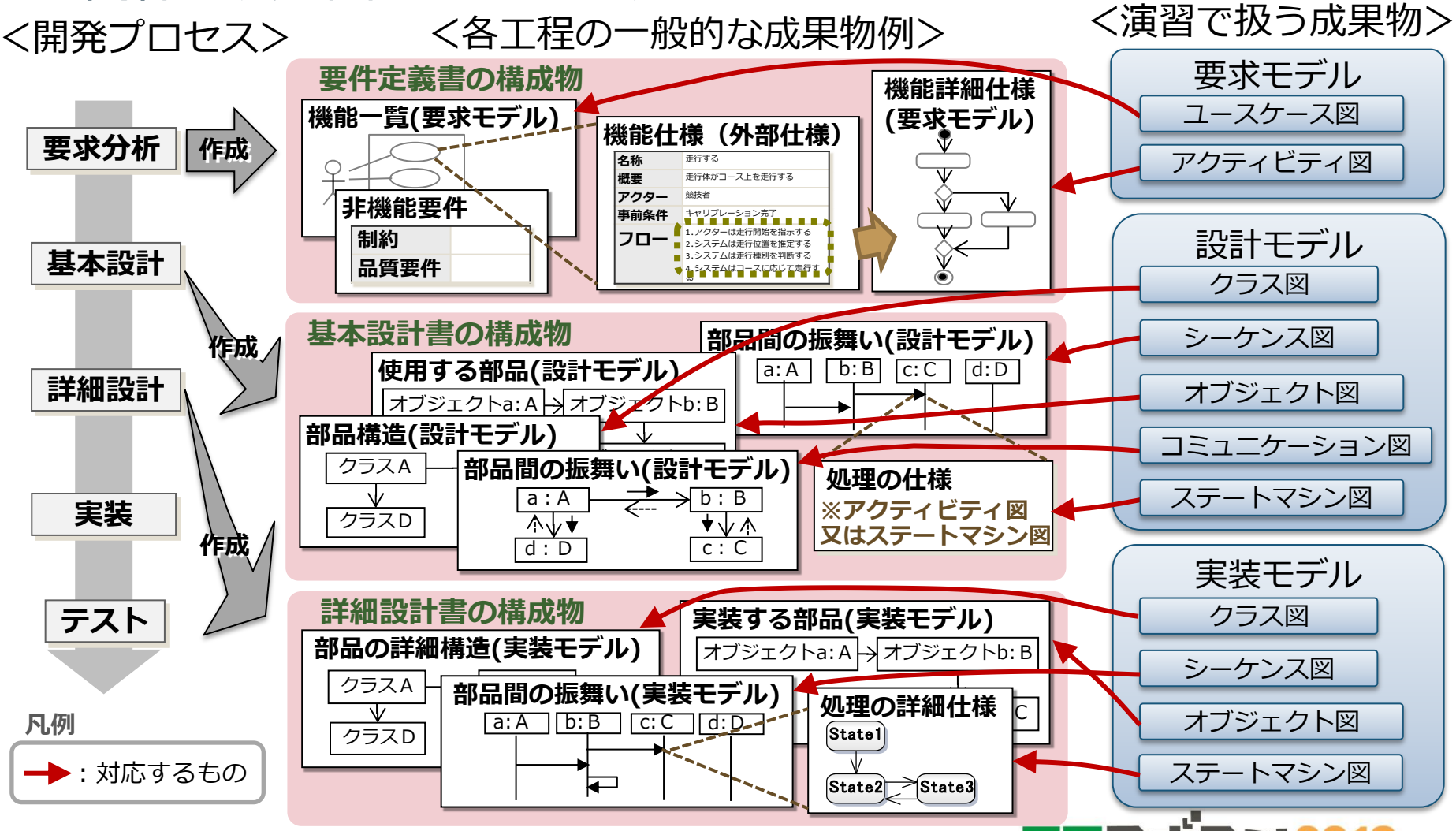
  - 反復型
    - 1回の反復で開発する範囲を絞り、開発を繰り返して段階的に作り込む



注）ここでは、単体テスト・結合テスト・システムテストをまとめて「テスト」と表記している

2-11. 演習と開発工程の対応について

- この章で紹介した開発工程と、次章以降の演習の成果物の関係は次の図のようになります



理解度チェック（２）

- ソフトウェアの開発の進め方とモデルの活用

1. 手順を踏まずになりゆきで開発するとどのような問題を起こすでしょうか？
   a. なぜ「段階的」に開発することが重要なのでしょうか？
2. ソフトウェア作成までの開発工程を４つに分けるとすると、それぞれなんという工程の名前になりますか？
   a. 上記４つの工程への入力と出力される成果物は通常どのようなものがありますか？
   b. それぞれの成果物にはどのようなUMLの図（モデル）を使いますか？
   c. それぞれの工程で行った活動の「正しさ」を確認、検証する工程はそれぞれ何と呼ばれますか？
3. UMLとは何ですか？
   a. UMLでモデル化する時の「３つの視点」とは何でしょうか？
4. モデル作成のスコープとは何でしたか？