開発プロセスとプロジェクト管理

F14 ソフトウェア工学概論

第14回

吉岡 廉太郎

前回の内容

- テスト工程の作業
  - 欠陥を見つけるのが目的
- テストの構成
  - 結合テスト手法の分類
  - システムテストの種類
- テスト計画と障害管理
  - 計画と記録が大切

今日の内容

- 開発プロセス

- プロジェクト管理
  - スコープ
  - WBS
  - 見積

プロセスの意味

- プロセス
  - 意図した出力（結果）を生み出すための、活動、制約、リソースを含む一連のステップ

- プロセスに含まれるもの
  - 目的、活動、制約、リソース
  - ツールやテクニック

プロセスの性質

- プロセスの主な活動を規定する
- プロセスでは、制限付きリソースを用いる
- プロセスでは、中間と最終成果物が生成される
- 複数のサブプロセスで構成されることもある
- 各活動には、決められた目的がある
- 各活動には、開始条件と完了条件がある
- 活動には順序があり、タイミングが明確である
- 活動、リソース、成果物には制限・制約がある

プロセスを定める利点

- 活動が構造化され、一貫性が生まれる
  - 不整合、冗長、足りないところを見つける
- 活動の理解、制御、検証、改善の指針となる
  - 目的を達成するのに必要な活動を見つけ、検証する
- 経験を蓄積し、将来に受け継ぐ手段になる

8.1.2

ソフトウェア・ライフ・サイクル

- ソフトウェアを構築するプロセスは、ソフトウェアのライフ・サイクルといえる

ソフトウェア・ライフ・サイクル

開発

- ✓ 要求の分析と定義
- ✓ システム（アーキテクチャ）の設計
- ✓ プログラム（詳細）の設計
- ✓ プログラムを書く
- ✓ テスト
- ✓ 納品

運用管理

- ✓ メンテナンス（運用管理）

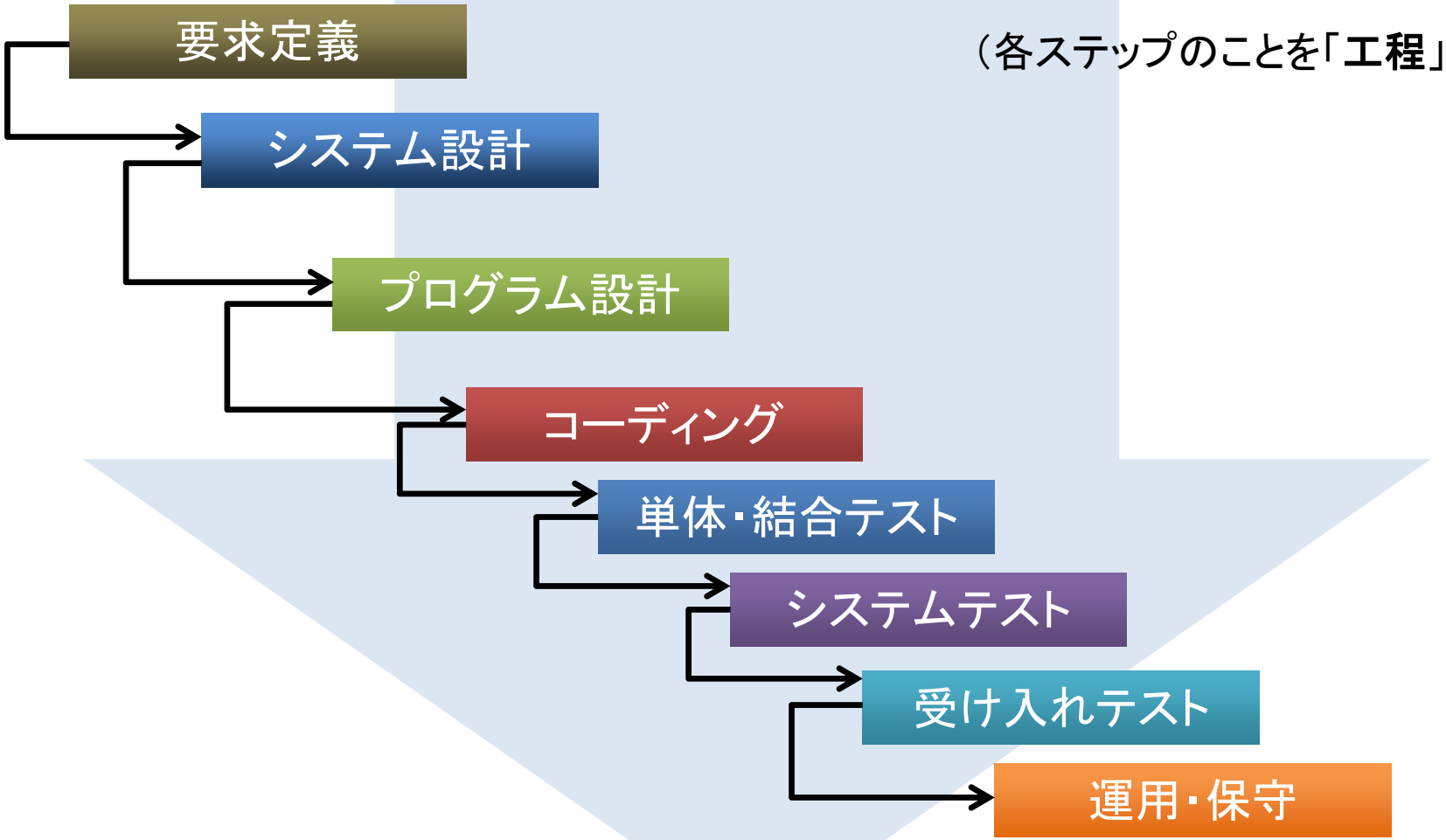
8.1.3

ソフトウェア開発のプロセス・モデル

- ウォーターフォールモデル
- V字モデル
- インクリメンタルプロセスモデル
- プロトタイピングモデル
- アジャイルモデル
- …

ソフトウェア開発における基本的なプロセスの
組み立て方を整理し類型化したもの。

ウォーターフォール・モデル



（各ステップのことを「**工程**」と呼ぶ）

8.2.1

ウォーターフォール・モデル

- 反復のない、一連のプロセス活動
  - 各フェーズがマイルストーンであり、それぞれに成果物がある
  - 単純で、お客さんも理解しやすい

- 要件の変更が起きにくい、良く理解された課題に有効
  - ソフトウェア開発を、創造（creative）プロセスではなく製造（manufacture）プロセスと見る

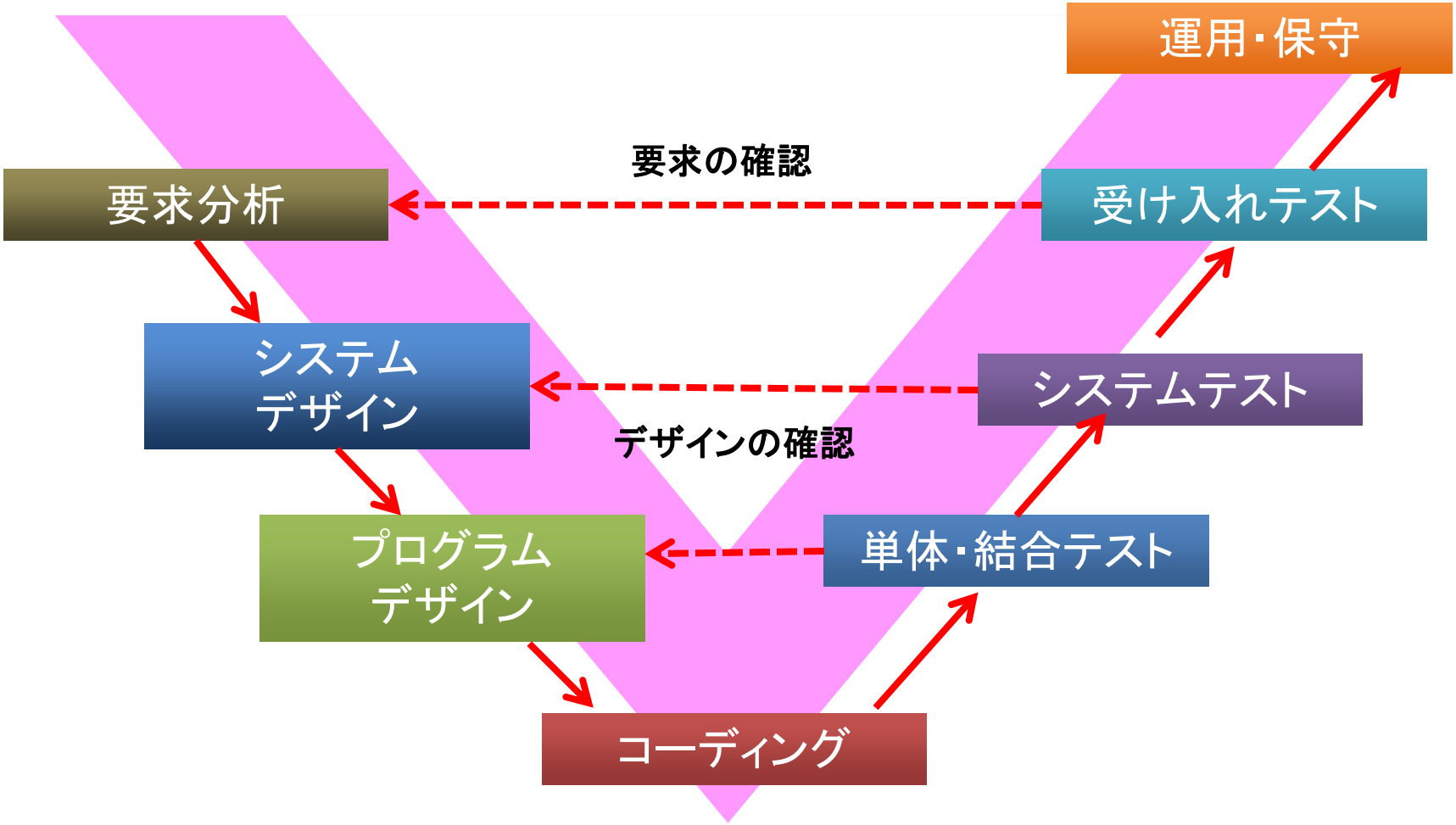
- 短所
  - 活動や成果物への変更が生じた場合への対応が決められていない
    （要求は完全に固めることができる、という前提）
  - 最終成果物を得るまでに長い待ち時間が発生する

8.2.2

V字モデル

- ウォーターフォール・モデルの変形・改良型
- 単体テストを使ってプログラム設計を評価する
- 結合テストを使ってシステム設計を評価する
- 受け入れテストを使って要求を検証する
- 評価・検証で問題が見つかった場合、V字の左側のプロセスを再実行してから、その先を継続する

V字モデル


運用・保守
要求の確認
要求分析
受け入れテスト
システム
デザイン
システムテスト
デザインの確認
プログラム
デザイン
単体・結合テスト
コーディング

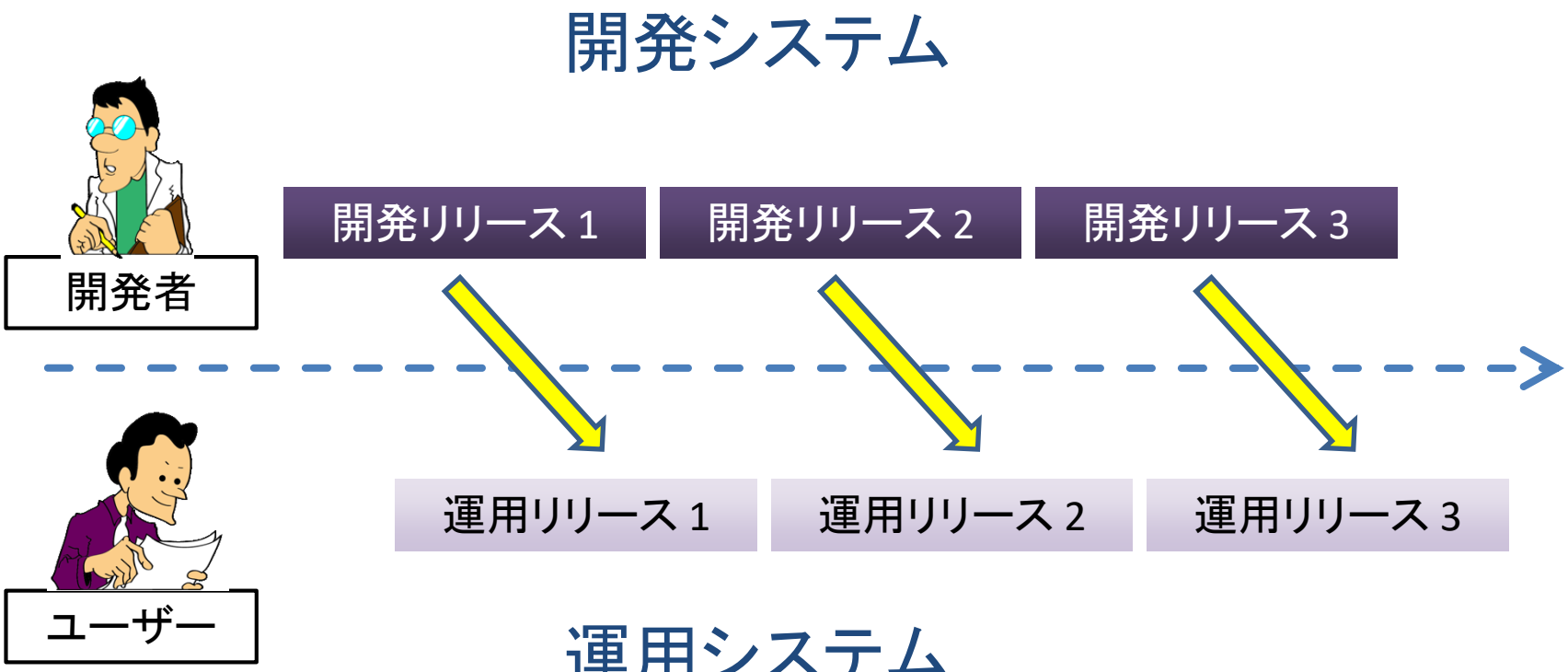
8.2.3

インクリメンタルプロセスモデル

- 短い周期で繰り返す
- システムを複数の部分に分けて納品
  - ユーザは、部分的に完成した機能を使いながら他の開発を待つことができる
- 好まれる理由
  - リリースが多いことで、開発者が予期しない問題を素早く直すことができる
  - リリースごとに特定の技術要素に集中でき、開発チームの負担が減る

オープンソース・プロジェクトに多い

インクリメンタルプロセスモデル



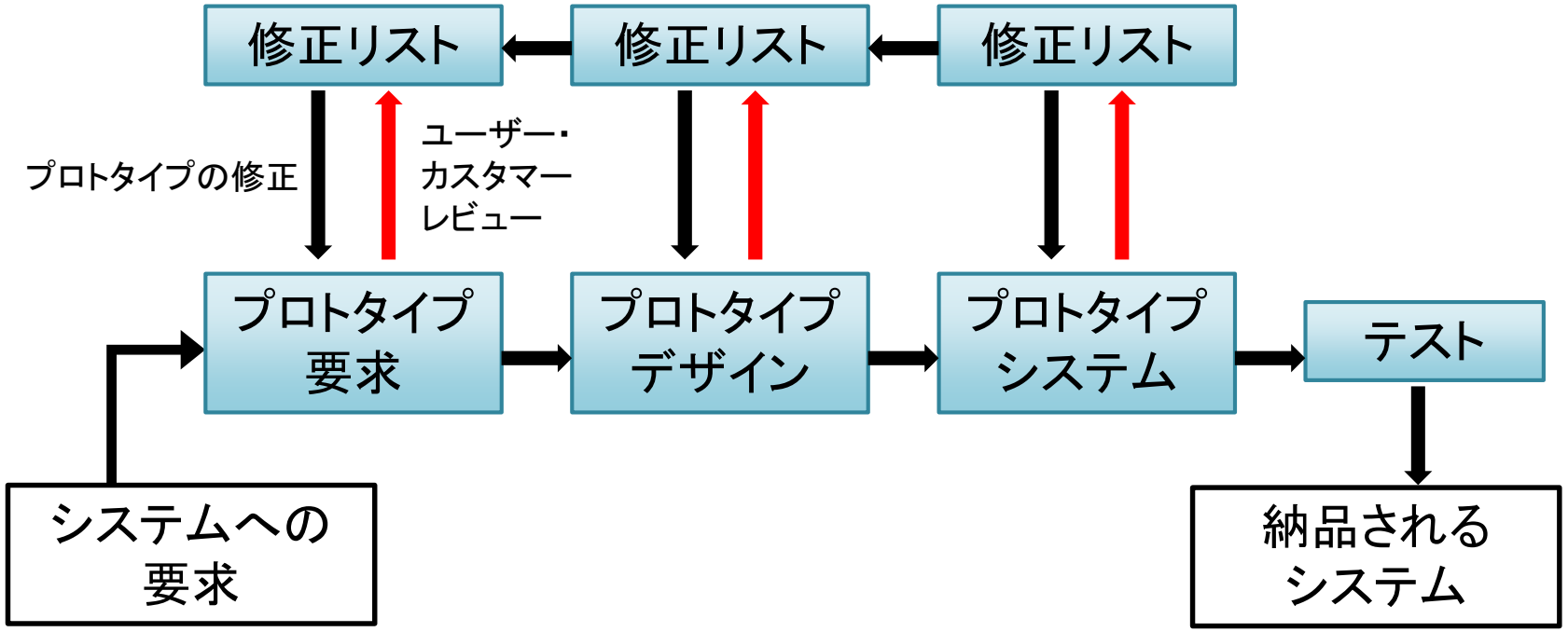
二つのシステムを同時に利用できる(オープンソース・プロジェクトに多い)

- 運用システム(リリース n):利用中
- 開発システム(リリース n+1):次のバージョン

8.2.4

プロトタイピング・モデル

- 要求や設計を、繰り返し確認できる
- 開発のリスクと不確実さを減らす効果がある



8.2.5

アジャイル開発

- ソフトウェアを速く、確実に開発するために柔軟性を強調

- アジャイルで重要視される要素
  - プロセスやツールより、人と意思の疎通を重視する
  - 質の高い文書を作ることより、動くソフトを生産することに時間を投資する
  - 契約交渉より、顧客との協業を楽しむ
  - 計画を守ることより、変化への対応に注力する

アジャイル開発：手法の例

- スクラム
  - 14～30日での反復（sprint）
  - 自己管理型チーム
  - プロダクトバックログ
  - プロダクトオーナー

- エクストリームプログラミング(XP)
  - コミュニケーション
    - 顧客と開発者の継続的な意見交換
  - シンプル
    - 最もシンプルな設計や実装を選択する
  - 勇気
    - 早め、多めに機能を届けることにコミットする
  - フィードバック
    - 開発プロセスのあらゆる活動に組み込む

プロジェクト管理

- プロジェクト
  - 独自の目標を持つ活動の単位
  - あらかじめ定められた期限を持つ

- プロジェクトには制約が3つある

スコープ	• 達成すべき目標;活動の範囲
時間	• 活動の完了するために必要な期間
コスト	• 活動に必要な資源にかかる費用

プロジェクト管理の知識

12.1.2
12.1.3

- プロジェクトの完遂には特有の知識とスキルが必要
  - メンバーの活動を統率するためのルール
  - 求められる成果にたどり着くための手順や仕組み

- PMBOK【ぴんぼっく】
  - Project Management Body of Knowledge
  - 実績のある実践的手法や考え方をまとめた体系
  - プロジェクトの実体に合わせて適用してもちいる

- PDCAサイクル
  - 計画（Plan）→実行（Do） →評価(Check) →対処(Act）を繰り返すこと
  - 活動を実体に合わせて改善していくという考え方

スコープの計画と管理

プロジェクトのスコープにまつわる難しさ

- 達成すべき目標（スコープ）をどのように定めるか
- 目標を達成するために必要な成果物は何か

作業の構造化

- 作業の細分化＝最少単位を明らかにすること
- 作業の依存関係を明らかにする

段階的詳細化が必須

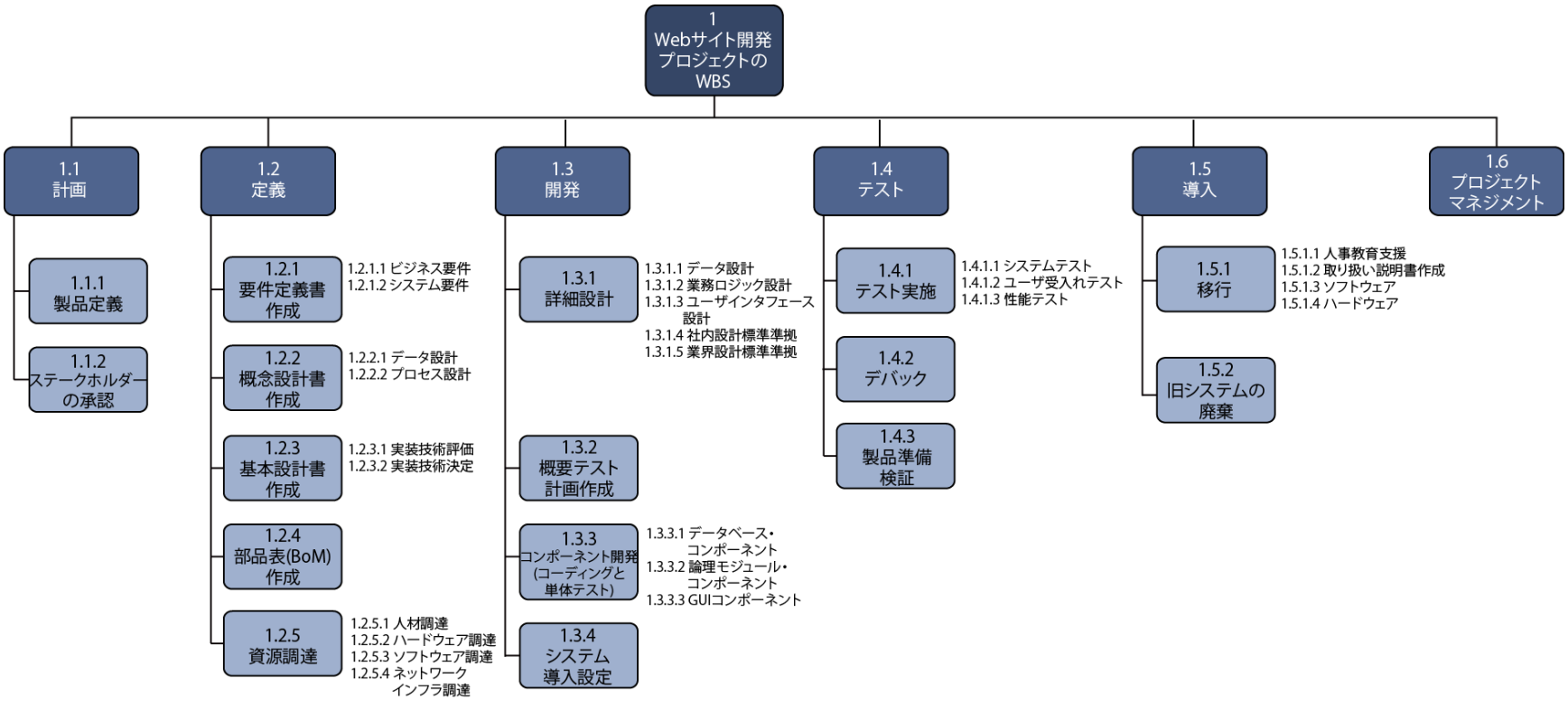
- プロジェクトを始める時点では良く分からないことがある
- 制約の多くは、最初は大雑把にしか定められない
- 理解が深まるにつれて詳細化するプロセス

WBS

- Work Breakdown Structure
  - プロジェクトで実施すべき作業を構造化したもの
  - 作業を階層的に分解する
  - 最少単位をワークパッケージという

NO.	作業内容	成果物	担当	
			貴社	弊社
1-3	新業務要件定義		―	―
1-3-1	新システムの入出力フォームの検討		○	―
1-3-1-1	現行入力シートの取りまとめ、提示	現行シート	○	―
1-3-1-2	ヒアリングシート作成	ヒアリングシート	―	○
1-3-1-3	各部門ヒアリング	議事録	○	―
1-3-1-4	ヒアリング結果まとめ	議事録	○	―
1-3-2	新システムの計算処理検討	議事録	―	○

WBS 例：Webシステム



12.2.2

WBSの役割

作業の網羅	•プロジェクトで実施する作業を網羅する •WBSの「100%ルール」 上位の作業は下位の成果物で構成される
計画値の整合性	•スケジュール、体制、品質を管理する元資料
担当の明確化	•プロジェクトで実施する作業の担当者を漏れなく記入する
管理指標	•達成状況を評価するための指標になる

時間の管理

12.3.3

- 開発活動に必要な時間をどのように計画するか
  - 客観的、定量的な見積りは難しい
- 工数：活動にかかる時間の単位（一般的には“時間”）
- 三点見積り
  - 過去プロジェクト、類似作業の実績などを参考に予測
  - 典型的な所要時間（Mostly）
  - 楽観的な所要時間（Optimistic）
  - 悲観的な所要時間（Pessimistic）

予測所要時間=(O+4M+P)/6

コストの管理

12.4.1

- 人月単価
  - ソフトウェア技術者一人にかかる一月分のコスト
  - 職種によってことなる：プログラマ、SE、上級SE など
  - 工数を表す単位としても使われる
  - 例）10人月：2人なら5ヶ月、5人なら2ヶ月と同じ

- 見積り手法の分類

比較（類推）見積り	• 過去の類似の活動のコストを基準に見積もる
係数（パラメトリック）見積り	• 過去の統計情報から求めた数式を用いて見積りする
ボトムアップ見積り	• WBSで分解した活動ごとに仕事量（工数）を集計する

12.5.3

ソフトウェアの規模を測る・見積る

- SLOC（Source Lines of Code）法
  - ソースコードの行数（LOC、SLOC）を測定する
  - 測りやすく客観性が高い
  - プログラミング言語に依存する

- FP（Function Point）法
  - ソフトウェアの機能要件をルールに沿って定量化する
  - IFPUGで標準化されていて誰でもできる
  - プログラミング言語に依存しない

- ユースケースポイント法
  - アクターとユースケースに重み付けをした個数に技術要因と環境要因を加えて定量化する
  - 要求分析段階から見積りができる

本日のまとめ

- 開発プロセス
  - 確実に開発するための活動の規定
- プロジェクト管理
  - スコープはWBSで段階的に詳細化する
  - WBSで明らかにした活動に必要な時間を見積もる
  - 規模の見積る

今後の学習

- この授業の範囲
  - “ソフトウェア工学”という分野を知ること
  - ソフトウェア開発の工程を理解し、各工程で行う作業の基本を知る

- 次のステップ
  - 「知る」→「理解」→「応用」

次ステップの授業科目

- IE04 ソフトウェア総合演習（3年）
  - 本科目の演習と同じ開発手順を踏んで開発手法を身に付ける
- SE4 ソフトウェア工学Ⅱ（3年）
  - システム開発とチーム作業の実践的技術を学ぶ
- L09 IT技術者の基礎（4年）
  - 企業での実業務を意識したシステム開発とプロジェクト管理を学ぶ
- SE5 ソフトウェアスタジオ（4年）
  - 実際のお客さんを相手にした開発を実践する中でソフトウェア工学を応用する

期末試験

- 日時
  - 2018年8月9日（木） 3,4限　10:50-12:30
- 場所
  - M1 : [CS]+[IT-SPR]+[TGU]
  - M2 : [SY]+[SE-DE]
  - M3 : [IT-CMV]+[CN]　※席も指定します
- 範囲・内容
  - 全14回の授業と演習のすべて
- 注意
  - 持ち込みはできません
  - 再試験はありません
- アドバイス
  - 授業資料のスライドを自分の言葉で補って説明できるようにする
  - 演習課題で行った作業を出来るようにしておく
  - http://borealis.u-aizu.ac.jp/classes/se1/
    ※質問に対する回答も掲載しています☺