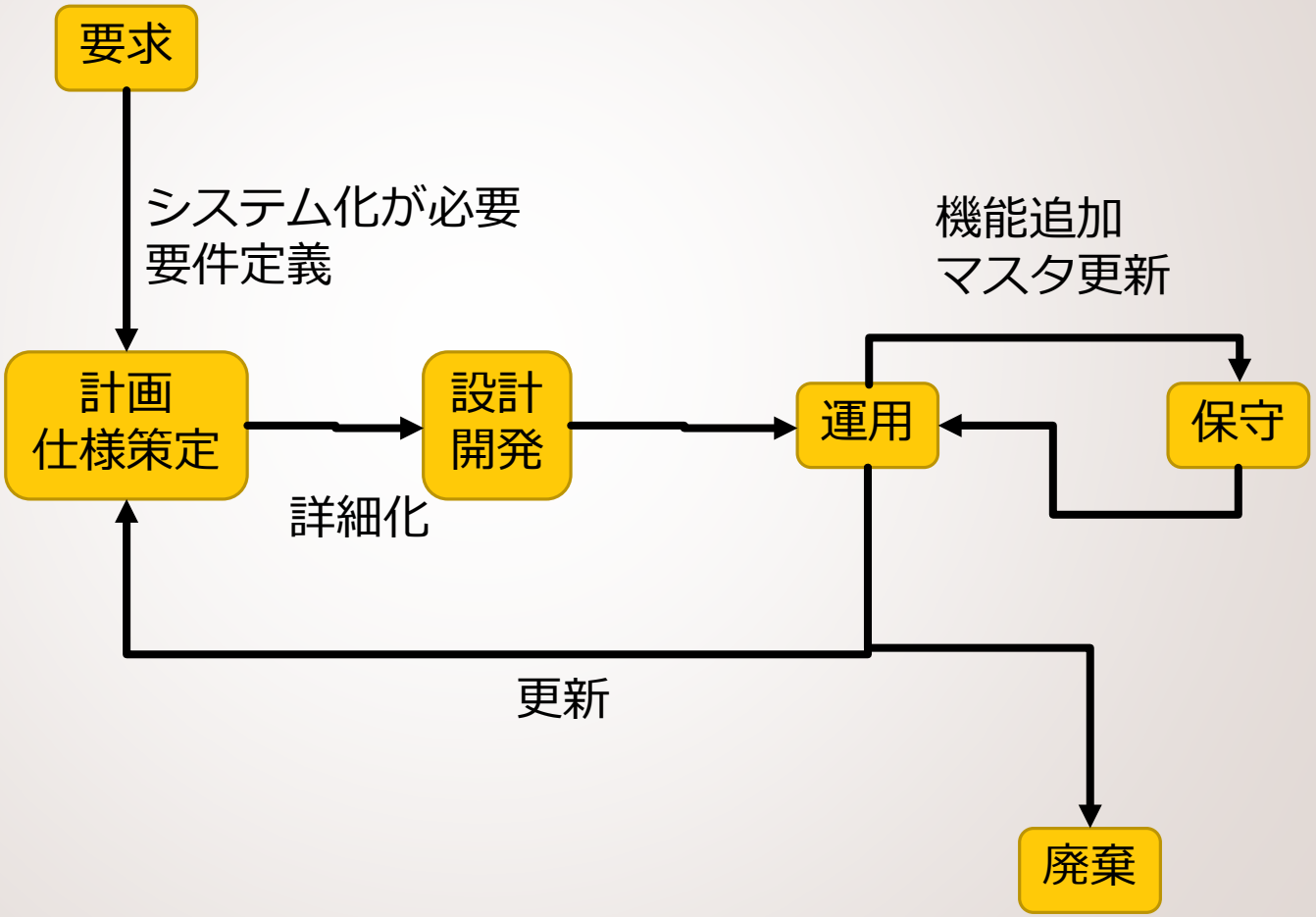
システムの設計と開発

情報科学の世界II

2020年度

只木　進一（理工学部）

システムのライフサイクル



システムの計画

- 定性的・非技術的要求を定量的・技術的要件へ
- 業務を行う部署と情報関連部署の連携が必須
- 業務改革が連動しなければ却って非効率になる危険性
- 他のシステムとの連携も意識
- 実現可能性・費用対効果・運用コストも検討

要件定義
業務を行う部署から

- システムの目的の整理
  - どの業務をシステム化するか
  - システム化で何を可能とするのか
- 業務改革は必須
  - 紙で行っていた時との違い
  - 業務見直し
  - 法律の制約

機能要件と非機能要件

- 機能要件
  - 業務に関する機能
  - 「○○ができること」
- 非機能要件
  - 性能：同時アクセス○人
  - 保守：「障害時に1時間以内に対応」
  - セキュリティ
  - 移行

詳細仕様

- 要件と技術仕様との調整
  - 業務部門とシステム開発者との調整
- 画面イメージ(mockup)
- 処理の流れ
- データベース設計

非常に重要な過程

システムの開発

- 外部設計
  - 外からの仕様：機能要件、ユーザインターフェース、他システムとの連携
- 内部設計
  - システムのモジュール化、インターフェイス詳細設計
- プログラム設計
  - データ構造、プログラムモジュール、モジュール間連携

システムの開発
モジュール化

- システムをできるだけ独立した部品に分割する
  - 業務毎の機能
  - 共通的機能
- MVC (Model-View-Control)
  - 業務のモデル
  - ユーザインターフェイス
  - 画面遷移

開発の後半

- プログラミング
- テスト
  - 単体テスト：各部品のテスト
  - 結合テスト：部品を連結した後のテスト
- 検収
  - 機能性能要件を満たしているか

品質保証

- ソフトウェア全般の質を保証
- コーディング規約
  - 誰が書いても同じ品質
- テスト体制
  - 開発者とは違う人がテスト
- 独立した品質保証部門

システムの運用・保守
初期

- 実際に業務へ投入し利用する
  - データの整備
    - 旧システムがある場合は、移行作業
- 利用者教育
- 不具合、要求要件との齟齬の調整
  - 要求要件との差が大きい場合には大問題

システムの運用・保守
中期

- 日常的なデータ更新・バックアップ
- OS・ミドルウェアのアップデート
  - 特に脆弱性対策
- 不具合対応
- ユーザ要求等への対応
  - 終期：更新への準備

システムの運用・保守
終期

- 更新への準備
  - 次期システムの計画
  - 何をどのように改善するのか
  - 単に「古くなったから」ではダメ
- データ移行準備
  - データクレンジング：不良データ対応

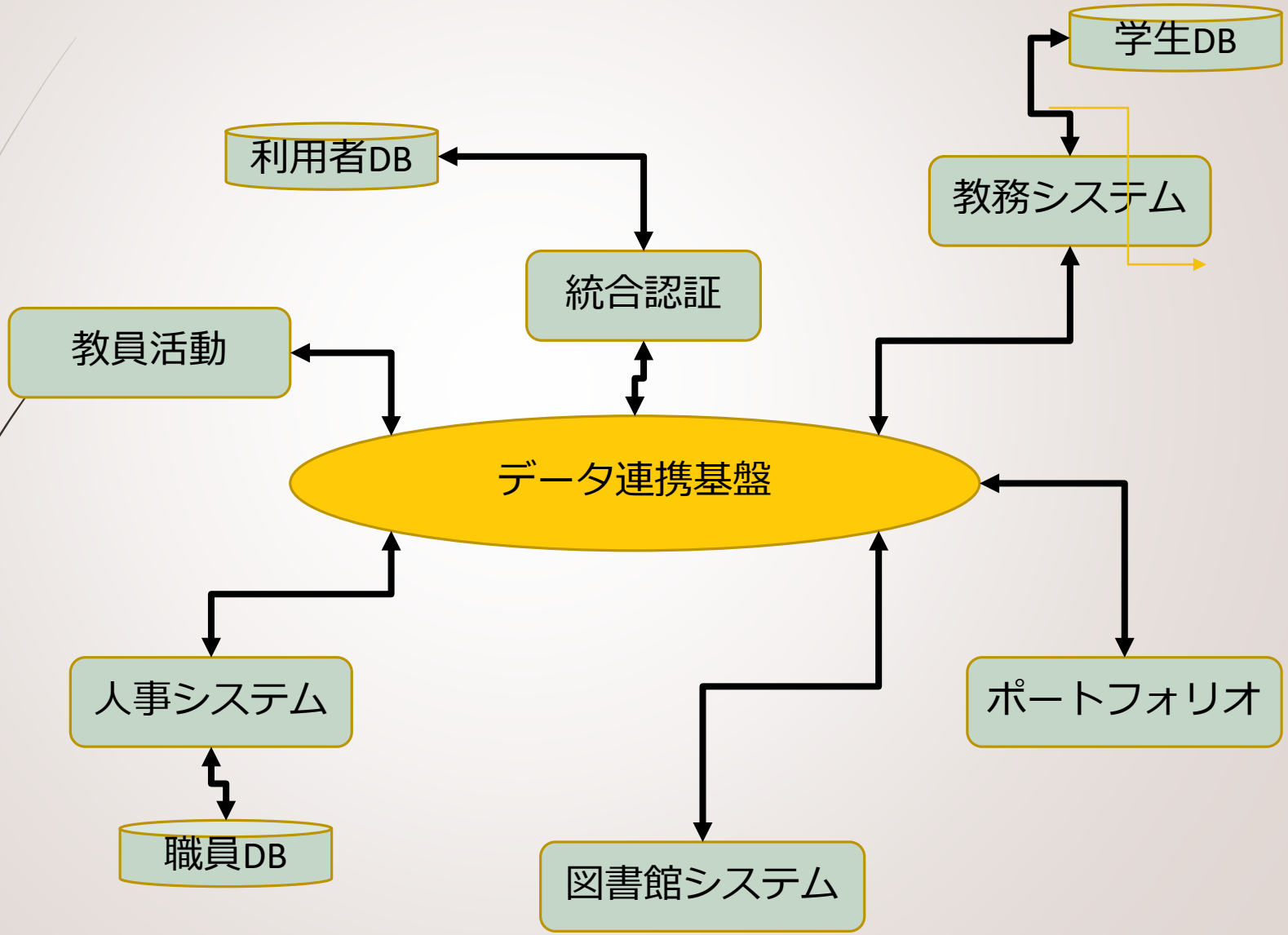
システムの開発工程

- Water Fall Model
  - 設計、開発、テストと直線的な進捗をイメージ
  - 不具合があった場合には、大きく戻る
- Agile Software Development
  - 一部の機能を持ったものを順次開発

他システムとの連携

- 他システムとのデータ連携は必須
  - 重複排除→効率化・迅速な更新
- 各システムはデータインターフェイス・データ更新タイミングが異なる
  - 柔軟な連携を可能にする基盤が有利
- 連結するためのキーが必要

データ連携イメージ


学生DB
利用者DB
教務システム
統合認証
教員活動
データ連携基盤
人事システム
ポートフォリオ
職員DB
図書館システム