要求の検証

FU14 ソフトウェア工学概論

第3回

吉岡 廉太郎

前回の内容

- 要求モデル
  - 要求を導き出すために使うモデル
  - たくさんの種類が存在する
- 要求モデルのパラダイム
  - たくさんあるモデルを分類して理解する/使い分ける
  - それぞれ要求を切り出す視点が違う
- 全てを覚える必要はない
  - 代表的な考え方、とらえ方を知る

モデル化とは？

- 共通のルール(=形式)で記述すること
  - 記述の対象：エンティティー、イベント、状態、…
  - 記述の方法：矢印、円、…

- モデル化の効果

  記述する対象や属性が決まっているため
  - 単純化できる＝一部を抜き出す
  - 漏れを防げる＝気付く
  - 共通語になる＝誰でも同じように理解できる

要求モデルのパラダイムと例

①データ･フロー図	• UMLユースケース図
②ER図	• UMLクラス図
③イベント･トレース	• UMLシーケンス図
④ステートマシン図	• UMLステートマシン図
⑤形式手法	• 真理表
⑥論理表現	• OCL

省略

ほんの一例…
他にもたくさんある

今日の内容

- 要求記述言語
- 要求工程のドキュメント
- 要求の妥当性確認と検証

どのモデルを使えばいいの？

- 目的に応じて適切なパラダイムを選択するものである
  - 表現したい対象や属性に応じて選ぶ
  - 詳細に検討したい性質や振る舞いに合わせて選ぶ
- 実際の開発では、複数のパラダイムを組み合わせて使う
  - ある程度の“経験”が不可欠である
- 基本的な組合せをメニュー化する要求記述言語の登場
  - 要求や仕様を記述するための言語体系
  - 複数のパラダイムのモデルを組み合わせたもの
  - 各モデルをどの場面で使えば良いかを規定している（＝開発者の判断によらない）
- 記述言語の例（それぞれ思想が異なる）
  - UML
  - SDL

2.2

Unified Modeling Language (UML)

複数のパラダイムの組み合わせ

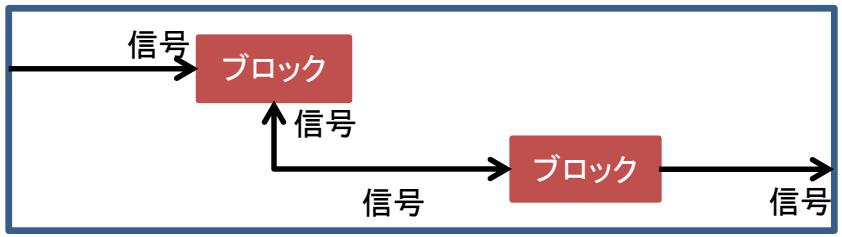
ユースケース図 （データフロー図）	• プロジェクトの初期にトップ・レベルの機能を記述。重要なエンティティーを特定することができる。
クラス図 （ER図）	• 主要なエンティティーとその関係を記述するUMLの中心的モデル。他のモデルを使って課題への理解を深め、クラス図を詳細化・厳密化してゆく。
シーケンス図 （イベントトレース）	• 複数のオブジェクトにまたがる重要なシナリオを記述。共通のサブシーケンスを見つけ、状態を明らかにする。
コラボレーション図 （イベントトレース）	• 複数のイベントトレースをクラス図上に記述。メッセージを時系列に記述できる。クラス間の関係を詳細化する。
ステートマシン図 （ステート・マシン）	• クラスごとの振る舞いを記述。詳細な情報が必要なため、課題の理解が深まってから、要求工程の最終版で作成する。
OCLプロパティー （論理言語）	• 上記のモデルに付加する。モデルの暗黙的な振る舞いを明白にしたり、振る舞いへの条件を付加する。

Specification and Description Lang.

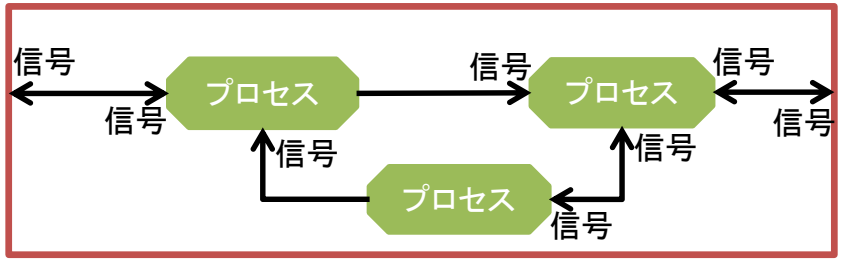
- ITU（国際電気通信連合）が標準化
- リアルタイム、並列、分散プロセスの振る舞いを定義
- 構成
  - SDLシステム図（データフロー図）
  - SDLブロック図（データフロー図）
  - SDLプロセス図（ステートマシン図）
  - SDLデータタイプ（代数的仕様）
  - メッセージシーケンス図

SDLを構成する各図の例

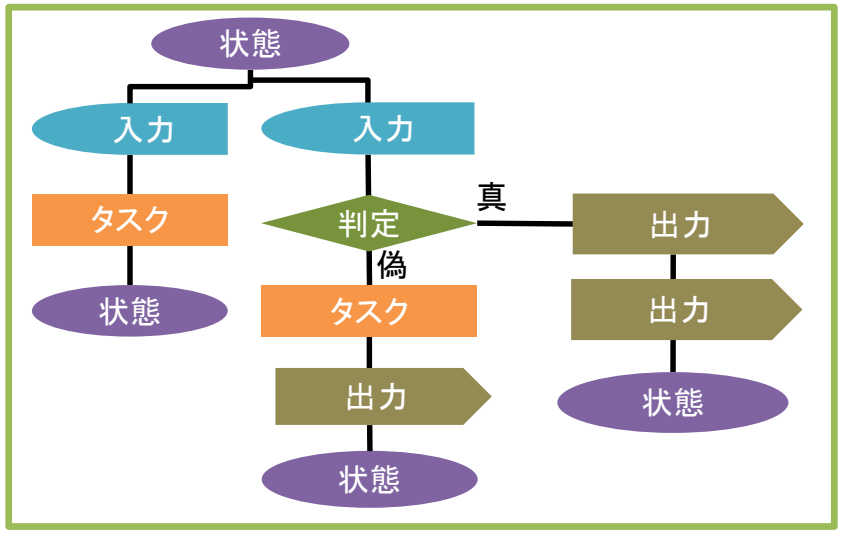
- SDLシステム図
  - トップレベルのブロックと通信路を記述する
  - 通信路には方向があり、信号の種類をラベルに記載する



- SDLブロック図
  - ブロックと通信路を詳細に記述する
  - 最も低レベルのプロセスとそれらの接続関係
  - メッセージは同期して送受信される



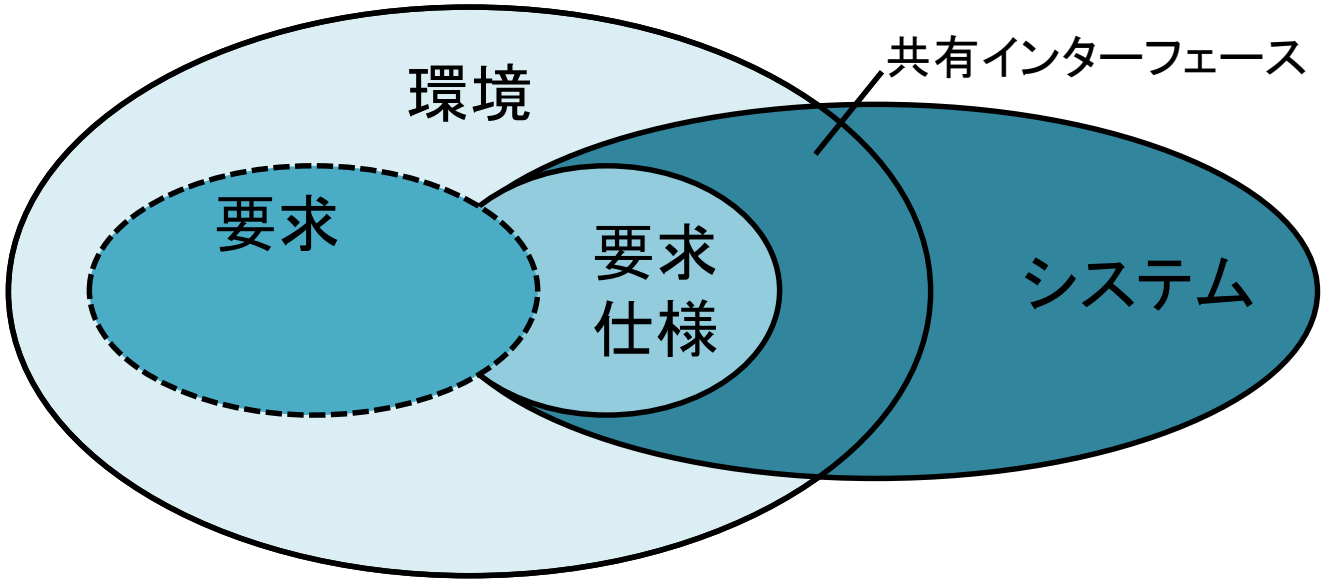
- SDLプロセス図
  - 状態の遷移を表すステートマシン
  - 遷移は一連の要素（入力、判定、タスク、出力）で表される



要求定義と要求仕様

4.1.7

- 要求定義
  - 顧客が達成したい内容をすべて記述すること
  - 要求は、システムを含み、その環境も含む
- 要求仕様
  - 顧客の要求を、提案するシステムの動作仕様として定義し直すこと
  - 仕様は、システムの範囲と一致する部分のみをカバーする



定義のレベル（詳細度）

- 定義レベルのばらつきが要求定義の難しさの一因
  - 文章の書き方による違い
  - 経験による違い
  - 異なる様式
  - 必要以上に詳述してしまう
  - 不十分で曖昧な記述

- 均一化するための指針
  - 一つの項目に要求を一つだけ書く
  - 一つの要求から他の要求への参照を避ける
  - 似たような要求をまとめる

要求定義における想定条件

- システムに関係する振る舞い
  - システムが実現する振る舞い
  - システム外の環境の振る舞い
    - 想定条件やドメイン知識という
- 想定条件として考慮すべき範囲
  - システムが正常に動作するための振る舞い
  - システムの出力に対する環境の振る舞い

4.6.1

IEEEソフトウェア要求仕様書

1. まえがき
   - 1-1. システムの目的
   - 1-2. システムの範囲
   - 1-3. 用語定義
   - 1-4. 参考文献
   - 1-5. この文書の概要

2. システムの概要
   - 2-1. 製品の概要
   - 2-2. 製品の機能
   - 2-3. 利用者の特性
   - 2-4. 制約事項
   - 2-5. 想定条件と依存関係

3. 詳細要求
   - 3-1. 外部インタフェース要件
     - 3-1-1. ユーザインタフェース
     - 3-1-2. ハードウェアインタフェース
     - 3-1-3. ソフトウェアインタフェース
     - 3-1-4. 通信インタフェース
   - 3-2. 機能要求
     - 3-2-1. クラス1
     - 3-2-2. クラス2
     - …
   - 3-3. 性能要求
   - 3-4. デザイン要件
   - 3-5. 品質要求
   - 3-6. その他の要求

4. 付録

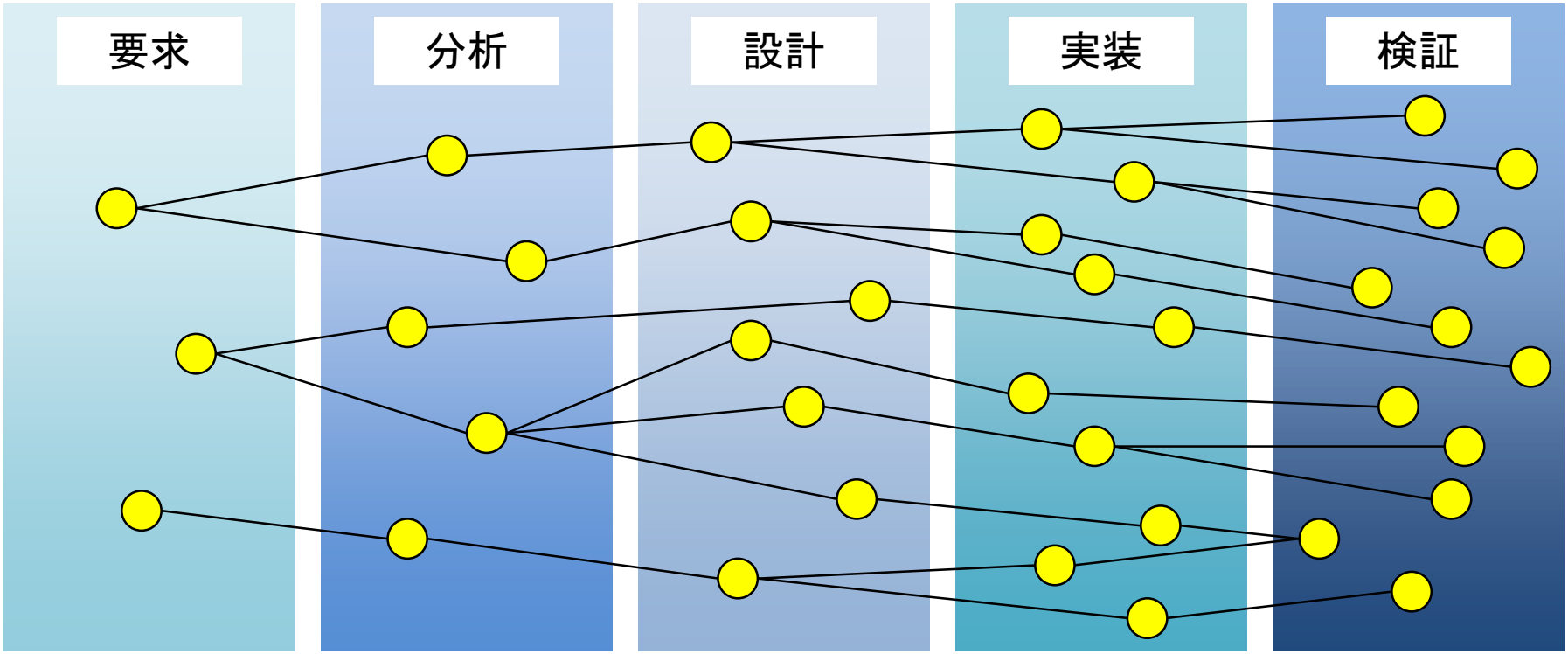
IEEE Std.830 1998年に標準化

プロセス管理と要求の追跡

- 一連の作業として開発プロセスを管理する
  - システムの動作を定義する要求
  - 要求から生み出される設計
  - 設計を実装するプログラミング
  - システムの機能を検証するテスト
  - システムを記述するドキュメント
- 個別に作られる部品をドキュメントと成果物で結ぶことでシステムとして統合する

開発の活動

- 各工程の活動はつながっている



Validation と Verification

4.7.1
7.1.1-2

- 用語の定義
  - Validation　妥当性確認
  - Verification　検証

■ 要求妥当性確認 requirements validation
  - 要求が注文主のニーズを正確に満たすか確認する

■ 要求検証 requirements verification
  - 各ドキュメントや生成物に矛盾が無いかを確認する

■ 妥当性確認
  - 求められているシステムを開発していることを確認

■ 検証
  - システムを間違いなく開発するための確認

4.6.2

要求の妥当性確認

- 要求の性質　妥当性確認の規準となる

正当性	• 要求の理解と合致する記述になっているか
無矛盾性	• 矛盾する要求や条件はないか
非曖昧性	• 要求を読んだ人たちの解釈は一致するか
完全性	• 全ての状態や条件における、全ての入力に対する振る舞いや出力が定義されているか
実現可能性	• そもそも注文主の要求に応えることができる解決策は存在するか（品質要求では注意が必要）
検証可能性	• 成果物が要求を満たすことを確認する手段があるか（受け入れテスト）
追跡可能性	• 各要求が簡単に参照できるようになっているか（番号付けなど）、要求定義書と要求仕様書の要求は一致するか

要求記述のテスト

- 解決策が要求を満たすことを検証する
  - 定量的な要求は検証方法も簡単
  - 定性的な要求は難しい

- 要求を検証しやすくする3つのテクニック
  - 形容詞と副詞それぞれを定量的に表す
  - 代名詞をシステムの構成要素の名前で置換える
  - 各名詞の定義が複数無いことを確認する

4.7.2

妥当性確認の手法

- ウォークスルー
- インスペクション
- チェックリスト
- プロトタイプ

矛盾の解決

- 関係者の立場ごとに要求がある
  - 相互に矛盾する要求もある
- 要求に優先順位をつける
- 例えば、3段階に分ける方法では
  - 必須：必ず満たすもの
  - 希望：必要だが必須ではないもの
  - オプション：あると良いがなくても良い

検証

- 要求仕様が要求定義と対応することを確認する
  - 要求定義書を作成していない場合は、顧客の要求と付き合わせる

- 追跡可能性
  - 要求定義に記載された各要求が、要求仕様に含まれていることを確認する

検証の手法

- クロスリファレンス
- シミュレーション
- 整合性のチェック
- 完全性のチェック
- 達成不可能な状態や遷移のチェック

要求の計測

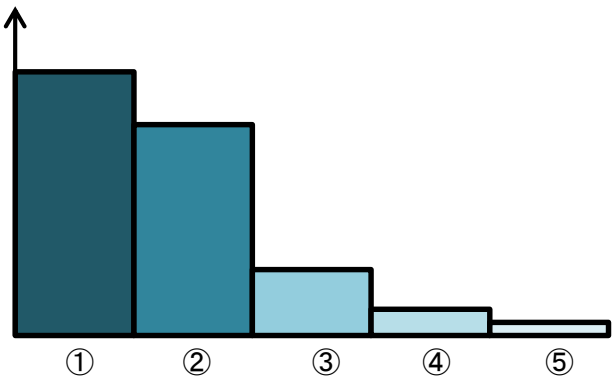
- 計測の対象
  - 成果物
  - プロセス
  - リソース
- 要求の数
  - 開発するシステムの規模を推測できる
- 要求に対する変更の数
  - システムに対する理解の成熟度の目安になる
- 要求の種類ごとに記録して比較・検討する

要求の採点と評価

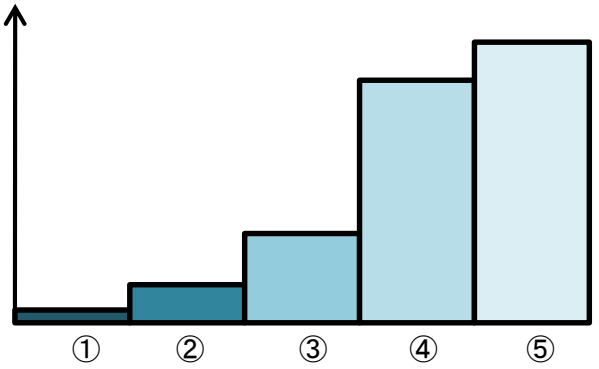
- 例）基準に沿って、設計・テスト担当者などが採点
  - ① 要求を理解できる。同様の要求を設計したことがあり、問題を感じない。
  - ② 初めての要求もあるが、過去に設計した要求と似ている。
  - ③ 設計したことがない要求だ。ただ、要求は理解できたので設計できると思う。
  - ④ 理解できないところがあり、設計できるか分からない。
  - ⑤ 全く要求を理解できず設計できない。

全ての要求について集計した結果

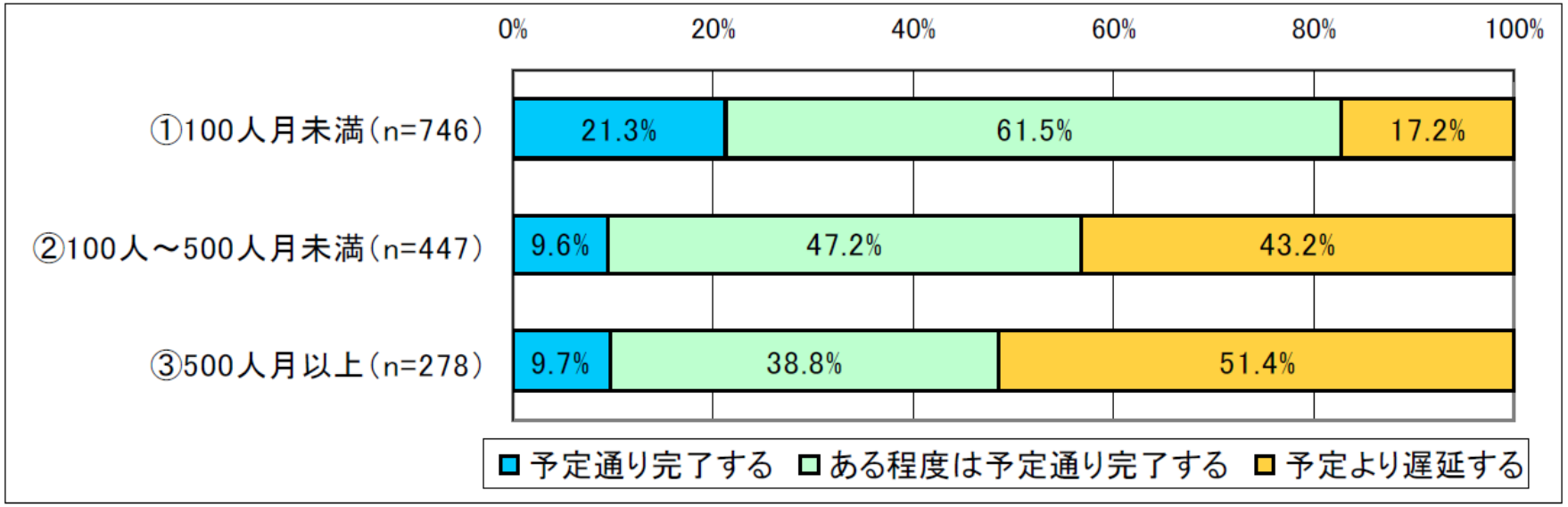
①と②が多い場合＝要求は大丈夫



④と⑤が多い場合＝要求を見直す必要あり



要求定義の失敗要因

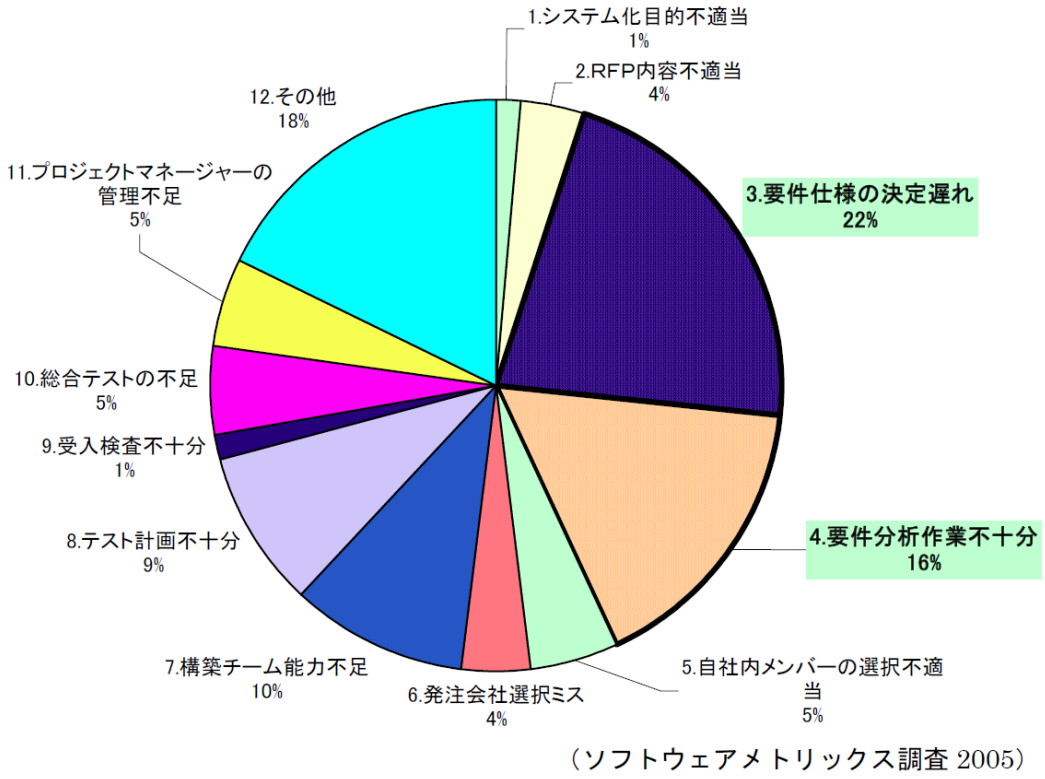


（IT 動向調査 2005）

システムリファレンスマニュアル、社団法人日本情報システム・ユーザー協会（JUAS）2005年

要求定義の失敗要因

ソフトウェアメトリックス調査報告書 2005 年版によると、要求仕様書の問題で約４０％のプロジェクトが遅延している。（他の調査データも同様な結果になっている）



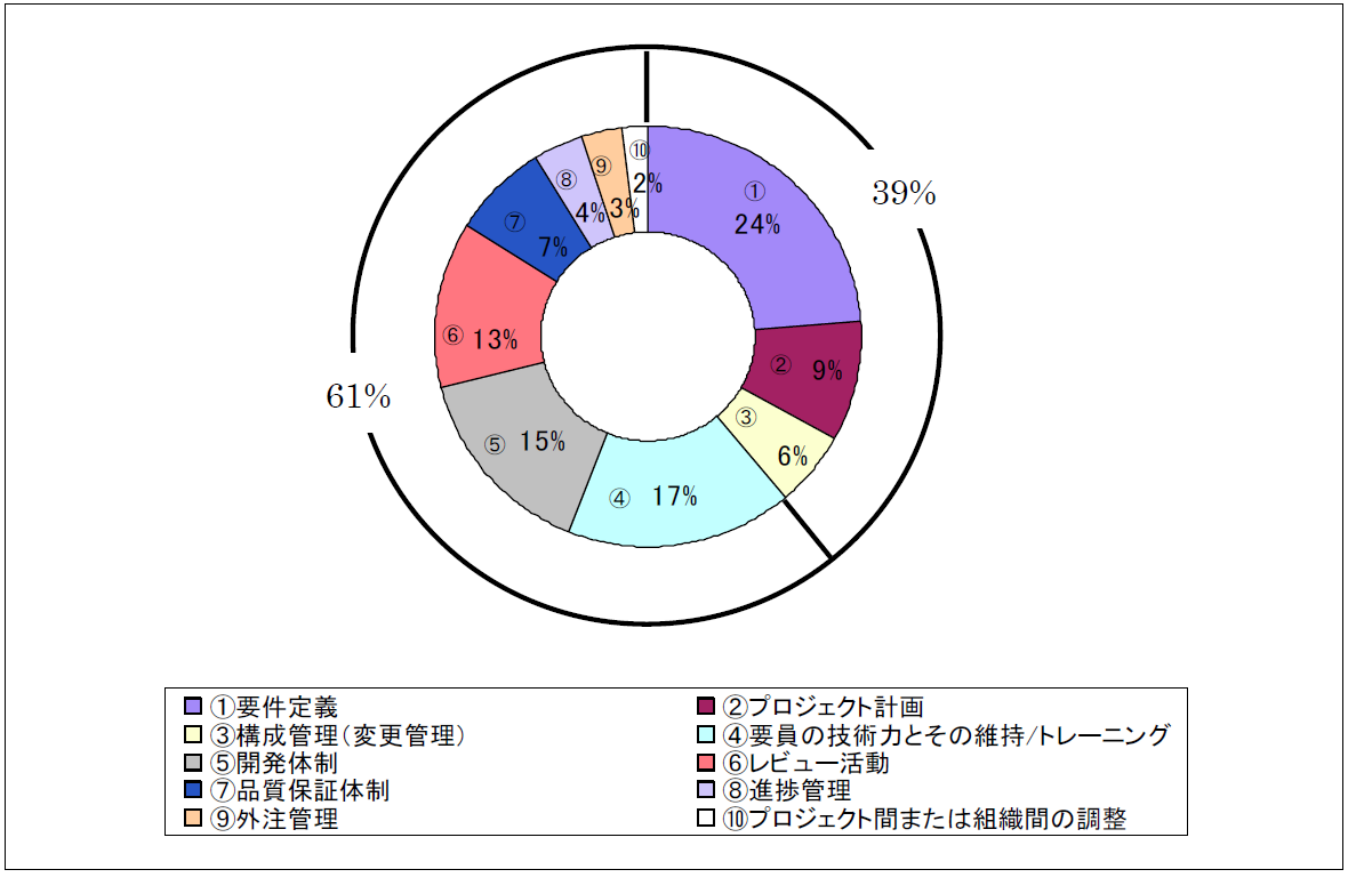
（ソフトウェアメトリックス調査 2005）

図表 3-1-4　工期遅延理由

システムリファレンスマニュアル、社団法人日本情報システム・ユーザー協会（JUAS）2005年

要求定義の失敗要因

＜品質・コスト・納期の成否に及ぼす要因＞



JISA（社団法人情報サービス産業協会）

システムリファレンスマニュアル、社団法人日本情報システム・ユーザー協会（JUAS）2005年

本日のまとめ

- 要求記述言語
  - 要求の性質・種類に応じて、どのモデルを使えば良いかの、統一的な指針である
  - 取り組む課題と開発チームの経験によって使い分ける
- ドキュメント
  - 要求定義：お客さんの言葉で表す
  - 要求仕様：開発者の言葉で表す
- 確認は2段階
  - 妥当性確認：正しいシステムを作っているのか
  - 検証：システムを正しく作れているか

次回講義の事前学習：
5.1.1, 5.1.2, 5.1.4