「ソフトウェア工学」

(0) イントロダクション

東大理学部情報科学科講義

石川　冬樹（本務先：国立情報学研究所）
f-ishikawa@nii.ac.jp / @fyufyu
http://research.nii.ac.jp/~f-ishikawa/

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

自己紹介

IS 2000er （情報科学科2000年度進学）

- 国立情報学研究所　准教授
  - ソフトウェア工学，特にディペンダビリティ：
    形式手法，自動テスト生成，安全性論証など
- 現在の主な研究プロジェクト
  - JST MIRAI-eAI：機械学習システムのディペンダビリティ
  - JST ERATO-MMSD：自動運転システムの安全性
- 産業界向け教育・実践研究
  - トップエスイー，日科技連SQiP，電通大AISECなど
  - 機械学習工学コミュニティ（MLSE研究会，QA4AI）

eAI

ERATO MMSD

MLSE 機械学習工学研究会
MACHINE LEARNING SYSTEMS ENGINEERING

QAI
ＡＩプロダクト
品質保証
コンソーシアム

講義概要（シラバスより）

効率よく高品質なソフトウェア製品・サービスを開発・運用するための活動を対象とする学問であるソフトウェア工学について学ぶ

まず，ソフトウェア開発・運用に必要となる活動や，各活動のための技術について広く概観する

次に，様々なパラダイムや，最新の研究動向と企業事例についても議論する

これにより，変化が激しく多種多様なソフトウェアの開発・運用において，原則や既存技術を踏まえつつ適切なアプローチを定めていくための素養を身につける

講義構成（詳細は変更の可能性あり）

0. イントロ（本資料）
1. ドメイン分析・要求分析
2. システム分析・設計
3. 形式手法とプログラム解析
4. テスティング
5. 保守・マネジメント
6. アジャイルソフトウェア開発や様々なパラダイム
7. 先端トピック（企業事例や研究事例）

まずは開発プロセスの
全体像を見てみる

成績

- 講義への参加（30%）
- レポート（70%）
  - 自由課題形式・早々に出します

目次

- ソフトウェア工学概観
- モデリングとプロセス
- UML

ソフトウェア工学

- 1968年 NATO Science Committee にて用いられた用語
  - 「ソフトウェア危機」への対応
  - ようやく50年超え

[http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF]

- SWEBOK（後述）における定義

> *the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software*

難しさの原因

- 有名な言葉 “No Silver Bullet”
- 難しさとして挙げられているもの
  - Complexity: 複雑さがサイズに対して非線形に増加し，
    枝葉末節ではなく本質である
  - Conformity: 人の感覚やインタフェースなど
    都度異なるな原則に合わせることが求められる
  - Changeability: 様々な要求変化・環境変化への対応が求められる
  - Invisibility: 見えないものであり，絡み合う関係性から
    効果的な可視化も難しい

[F. P. Brooks, Jr., No Silver Bullet – Essence and Accident in Software Engineering, 1987]

考えて欲しいこと

- 「ソフトウェアの仕事」をするために
  - ※ プログラミング演習や，自分のための日曜大工ではなく
  - ※ 開発だけでなく品質保証，運用，企画や営業，発注なども
  - 他人が欲しいものを作る（作る人と使う人が違う）
    - 東大用の履修管理システムの機能はどうなっている？
  - 大きなものを複数人・複数組織で分担して作る
    - 天才1人ですべてはできない
    - 同じ人たちだけでいつまでもできるわけではない
  - お金をもらって事前に契約をして作る
    - 予算と納期，責任範囲の定義

SWEBOK

- Software Engineering Body Of Knowledge (現在 V 3.0)
  - IEEEによる知識体系

[https://www.computer.org/education/bodies-of-knowledge/software-engineering]

1. Software Requirements
2. Software Design
3. Software Construction
4. Software Testing
5. Software Maintenance
6. Software Configuration Management
7. Software Engineering Management
8. Software Engineering Process
9. Software Engineering Models and Methods
10. Software Quality
11. Software Engineering Professional Practice
12. Software Engineering Economics
13. Computing Foundations
14. Mathematical Foundations
15. Engineering Foundations

ソフトウェア工学に入るものの例

- 「システムが何をすべきか」分析する
  - 技術の例：インタビュー，エスノグラフィ
- どれだけのコストがかかるか見積もる
  - 技術の例：構成要素や複雑さの測定，コストの統計分析や予測
- 効果的な管理・連携のために情報伝達する
  - 技術の例：朝礼，カンバン

1. Software Requirements

7. Software Engineering Management

12. Software Engineering Economics

11. Software Engineering Professional Practice

ソフトウェア工学の全体：教科書の目次から (1)

■実践ソフトウェアエンジニアリング（第9版）Pressman and Maxim, オーム社，2021

- プロセス
  - プロセスモデル
  - アジャイルとプロセス
  - 推奨のプロセスモデル
  - ソフトウェアエンジニアリングの人間的側面

- モデリング
  - プラクティスの指針となる原則
  - 要求エンジニアリング
  - 要求モデリングの推奨手法
  - 設計の概念
  - アーキテクチャ設計の推奨手法
  - コンポーネント設計
  - ユーザエクスペリエンス設計
  - 移動体端末におけるソフトウェアの設計
  - パターンに基づく設計

（次頁へ続く）

ソフトウェア工学の全体：教科書の目次から (1)

■ （続き）

- 品質とセキュリティ
  - 品質の概念
  - レビューの推奨手法
  - ソフトウェア品質保証
  - ソフトウェアセキュリティエンジニアリング
  - ソフトウェアテスト～コンポーネントレベル
  - ソフトウェアテスト～統合レベル
  - ソフトウェアテスト～移動体端末と特定ドメインに対するテスト
  - ソフトウェア構成マネジメント
  - ソフトウェアメトリクスと分析

- ソフトウェアプロジェクトのマネジメント
  - プロジェクトマネジメントの概念
  - 実行可能で役立つソフトウェア計画
  - リスクマネジメント
  - ソフトウェアサポート戦略
- 先端的な話題
  - ソフトウェアプロセス改善
  - ソフトウェアエンジニアリングの新興トレンド

ソフトウェア工学の全体：教科書の目次から (2)

■ソフトウェア工学の基礎（改訂新版）玉井，岩波書店，2022

- ソフトウェアプロセス
- 要求工学
- モデル化技法とUML
- データと制御の流れモデル
- 動的振る舞いモデル
- オブジェクト指向モデル
- 形式手法
- 設計技法
- 検証技術
- ソフトウェアの保守・進化
- 開発環境とツール
- 安全・安心な社会のためのソフトウェア
- プロジェクト管理

ソフトウェア工学の全体：教科書の目次から (3)

■Googleのソフトウェアエンジニアリング，Wintersら，オライリー，2021

- 文化
  - チームでうまく仕事をするには
  - 知識共有
  - 公正のためのエンジニアリング
  - チームリーダー入門
  - スケールするリーダー
  - エンジニアリング生産性の計測

- プロセス
  - スタイルガイドとルール
  - コードレビュー
  - ドキュメンテーション
  - テスト概観
  - ユニットテスト
  - テストダブル
  - 大規模テスト
  - 廃止

（次頁へ続く）

ソフトウェア工学の全体：教科書の目次から (3)

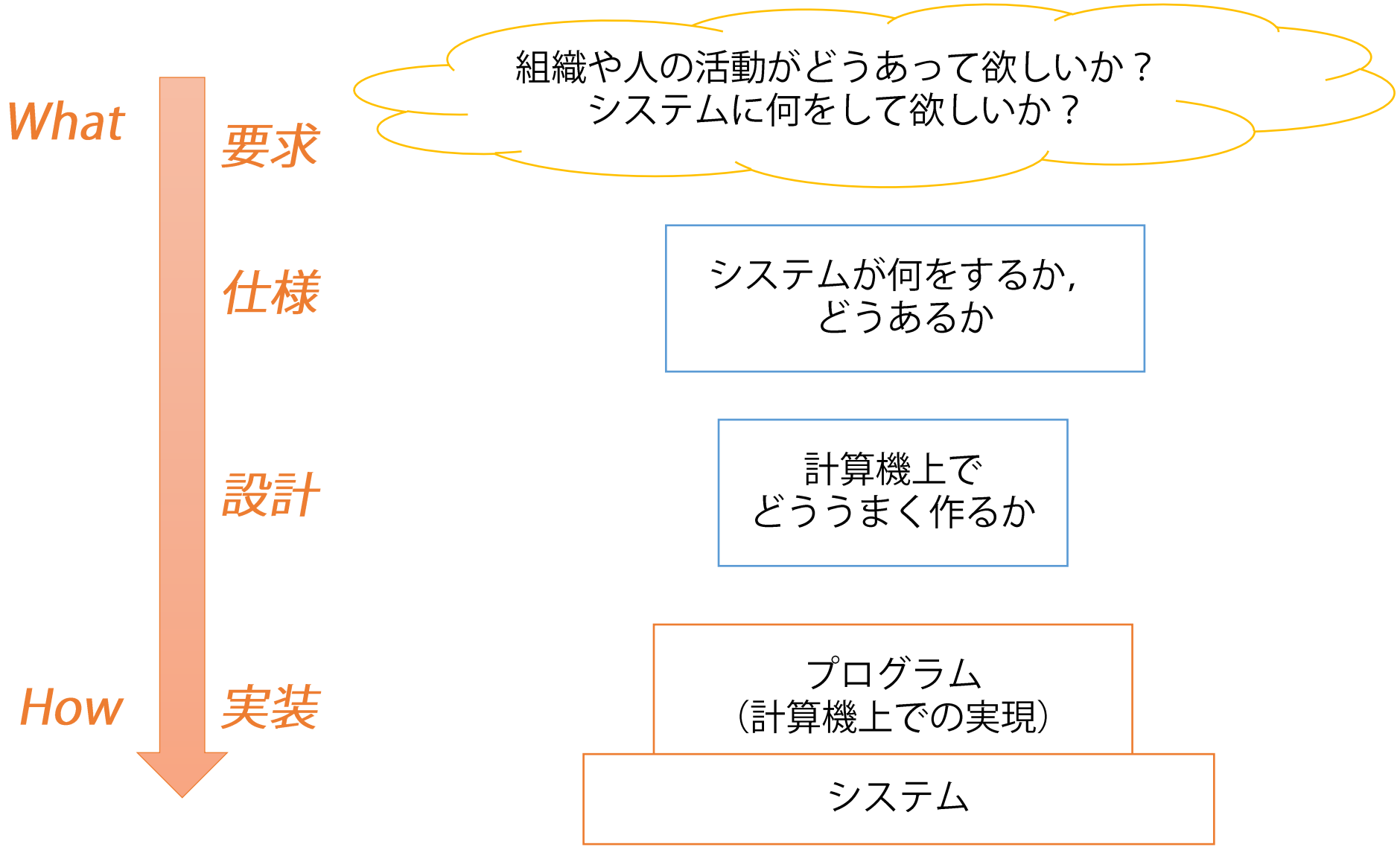
■ （続き）

- ツール
  - バージョンコントロールとブランチ管理
  - Code Search
  - ビルドシステムとビルド哲学
  - GoogleのコードレビューツールCritique
  - 静的解析
  - 依存関係管理
  - 大規模変更
  - 継続的インテグレーション
  - 継続的デリバリー
  - サービスとしてのコンピュート

目次

- ソフトウェア工学概観
- モデリングとプロセス
- UML

ソフトウェア開発における抽象度



モデル

- 「モデル」とは？

> a simple description of a system, used for explaining how something works or calculating what might happen, etc.

[https://www.oxfordlearnersdictionaries.com/ (access:2022/09/29)]

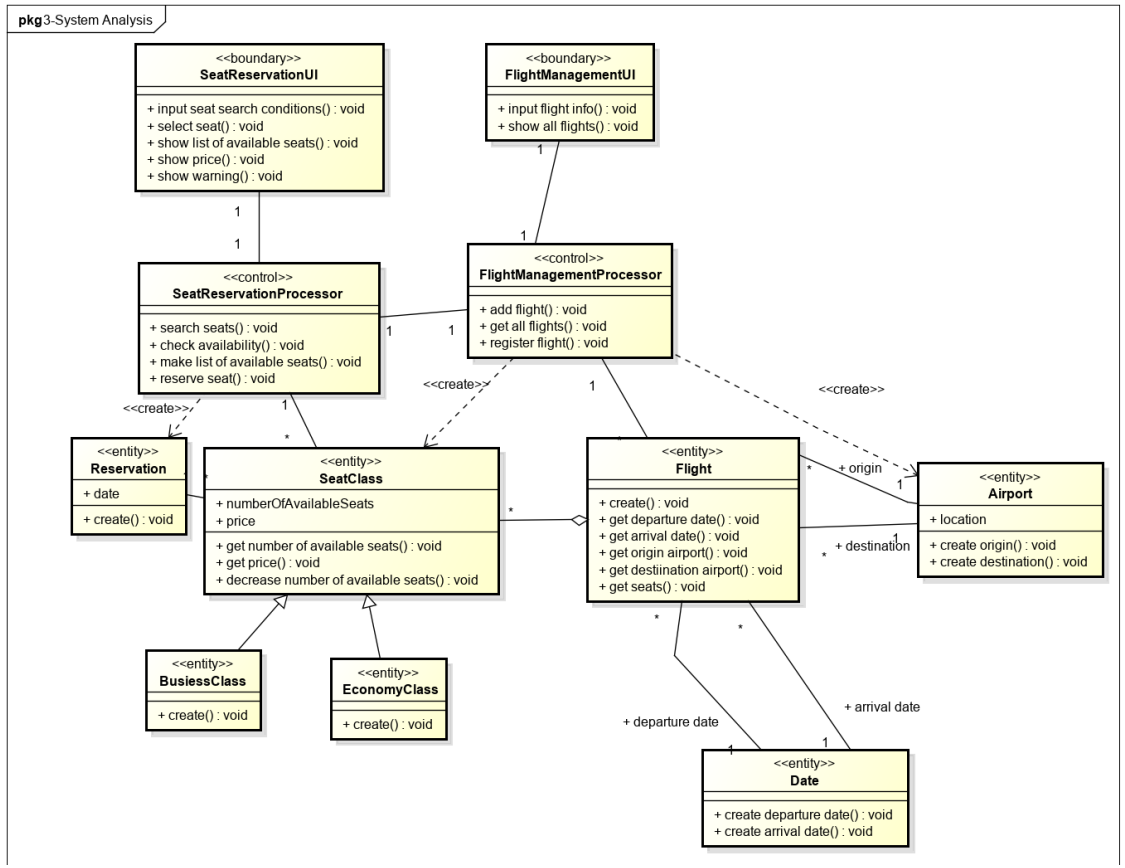
- 注目する側面に特化し抽象化・単純化（詳細を捨象）
- 効率的，効果的に分析，検証

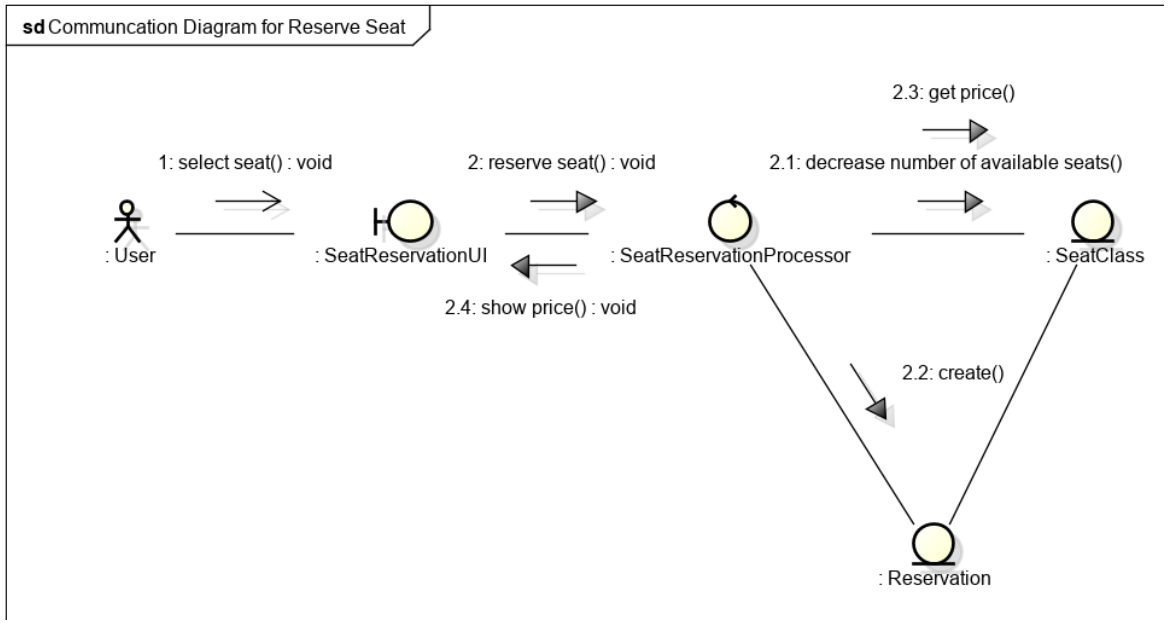
これから作ろうとするシステムや既存のシステムのモデル：
要求モデル，設計モデル，…

開発の活動をとらえるモデル：プロセスモデル

設計モデルの例

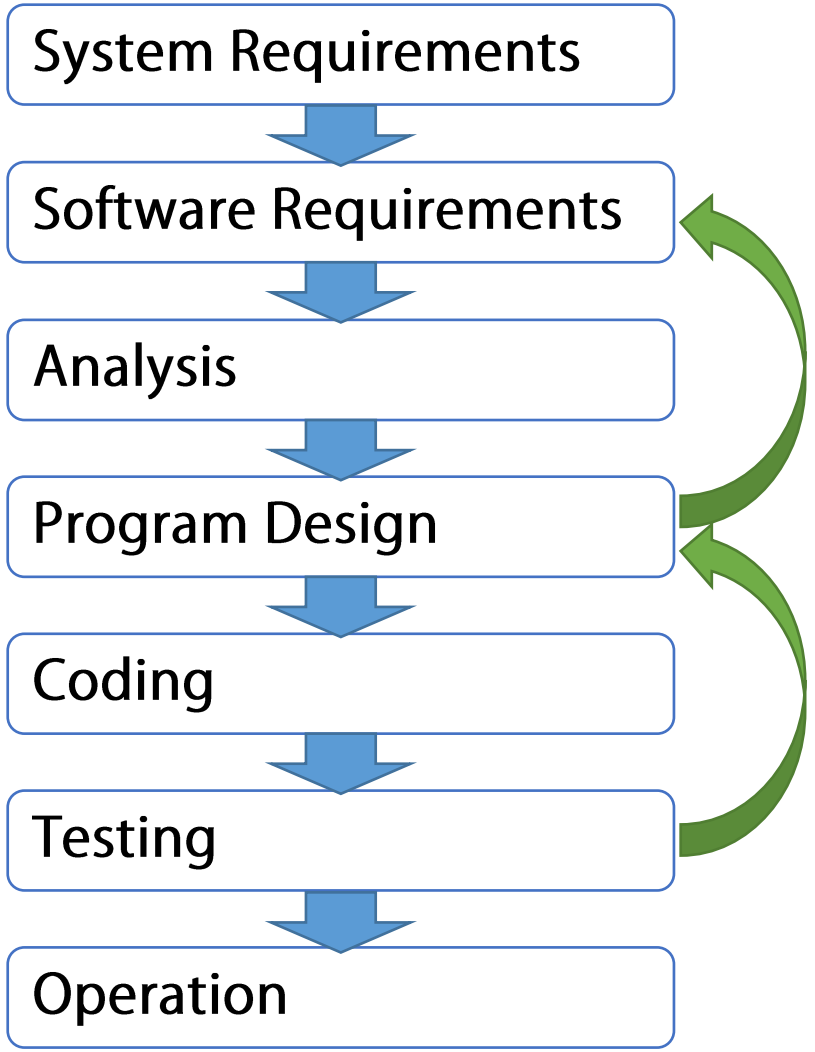
- UML (Unified Modeling Language)
  - クラス図（左）・コミュニケーション図（右）





プロセスモデルの例：ウォーターフォールモデル

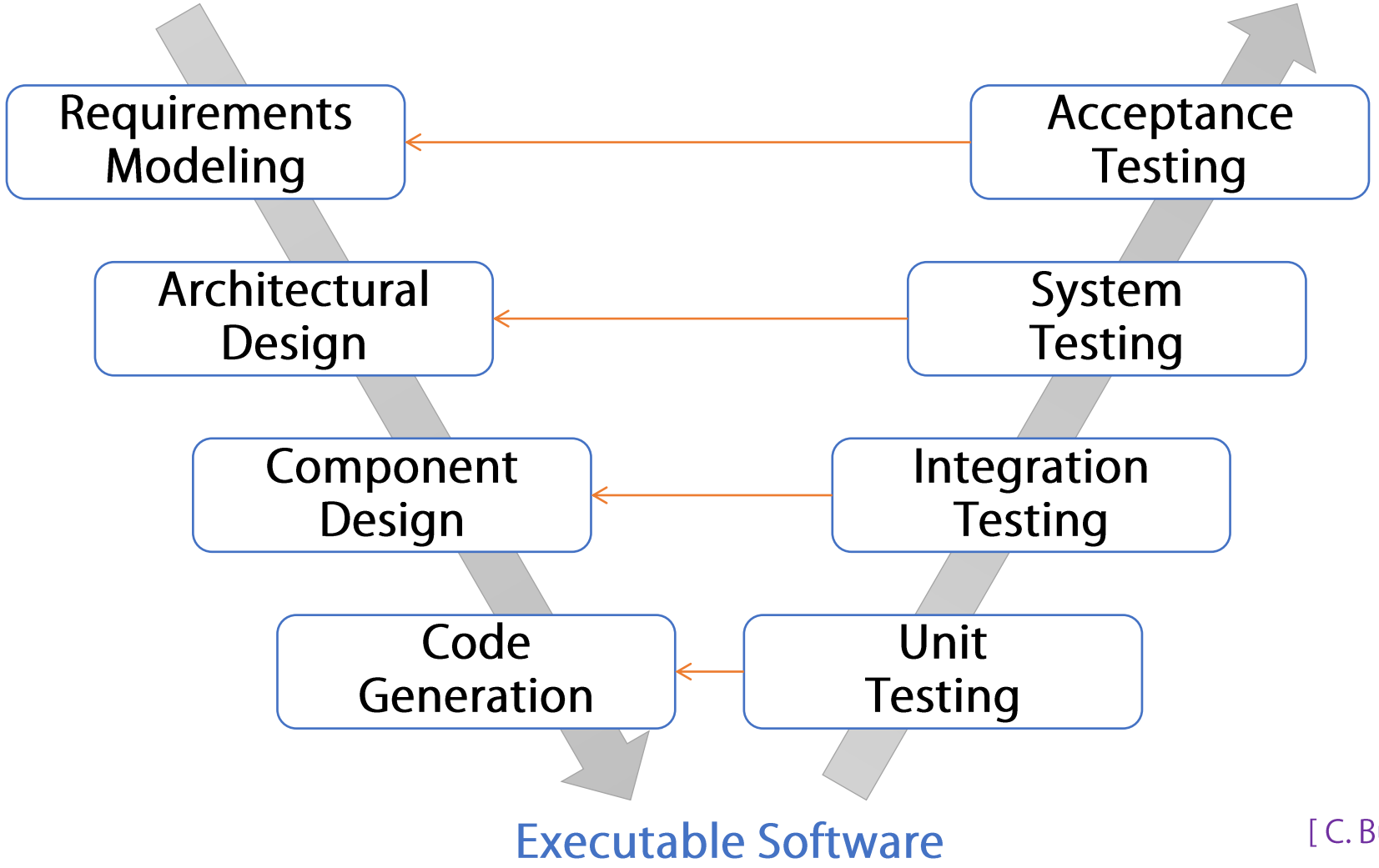
- 右図：Royceによるプロセスモデル
  - 「抽象 → 具体」の工程を定義
  - 工程の名称，後戻りの有無や位置は多数のバリエーションあり
- ウォーターフォールモデル
  - 右図で後戻りがないものが典型的
  - 「後工程での発見・変化に対応する柔軟性が欠如している進め方」の象徴として批判的に扱われることが多い



[W. Royce, Managing the Development of Large Software Systems, 1970]

プロセスモデルの例： V字

- 検査の範囲・対象を明確にしたプロセスモデル

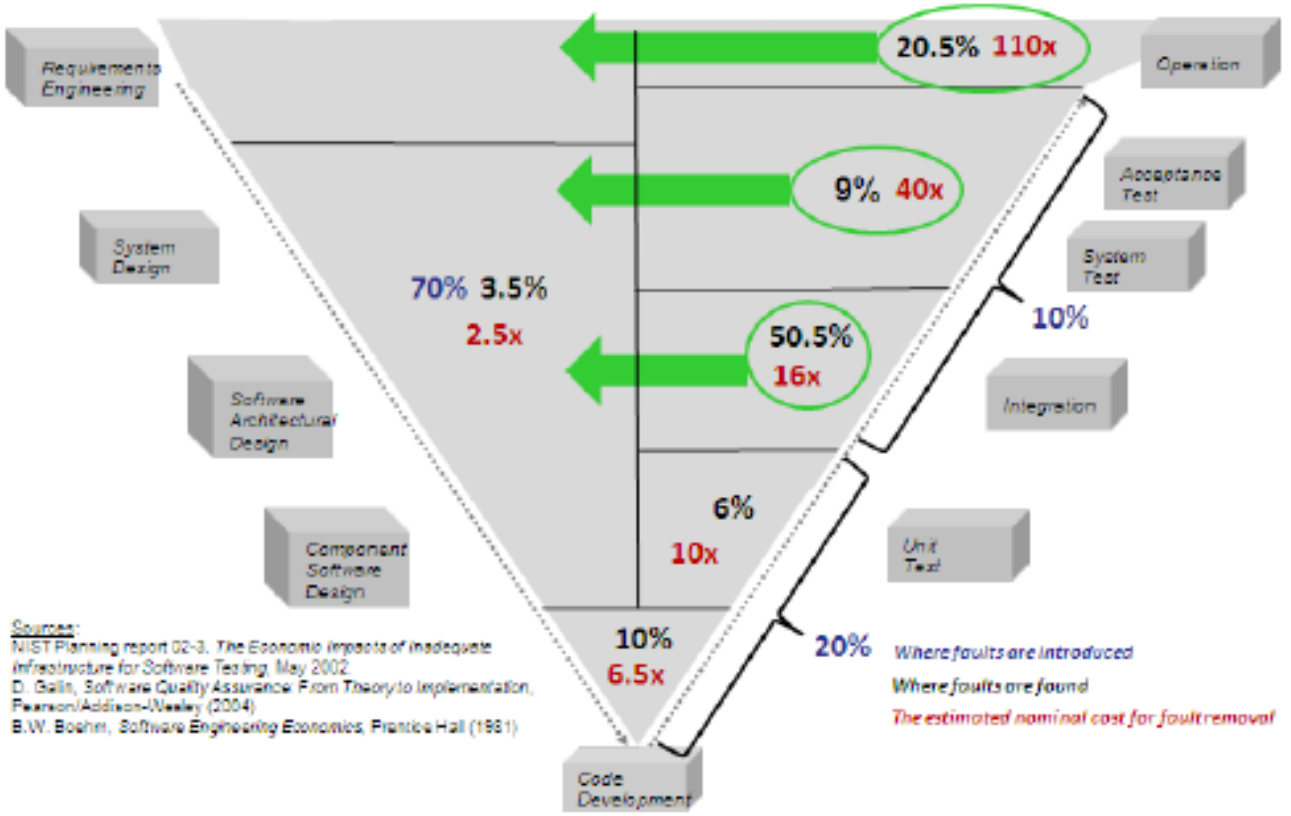


※ これも詳細な定義は
多数バリエーションあり

[C. Bucanac, The V Model, 1999]

開発プロセスにおける上流工程の重要性

- 誤りを早く発見することに関する統計の一例



青字：
不具合が埋め込まれる場所

黒字：
不具合が発見される場所

赤字：
不具合の除去におけるコストの増加率

[P. H. Feilerら, System Architecture Virtual Integration: An Industrial Case Study, 2009]

再掲：講義構成（詳細は変更の可能性あり）

0. イントロ（本資料）
1. ドメイン分析・要求分析
2. システム分析・設計
3. 形式手法とプログラム解析
4. テスティング
5. 保守・マネジメント
6. アジャイルソフトウェア開発や様々なパラダイム
7. 先端トピック（企業事例や研究事例）

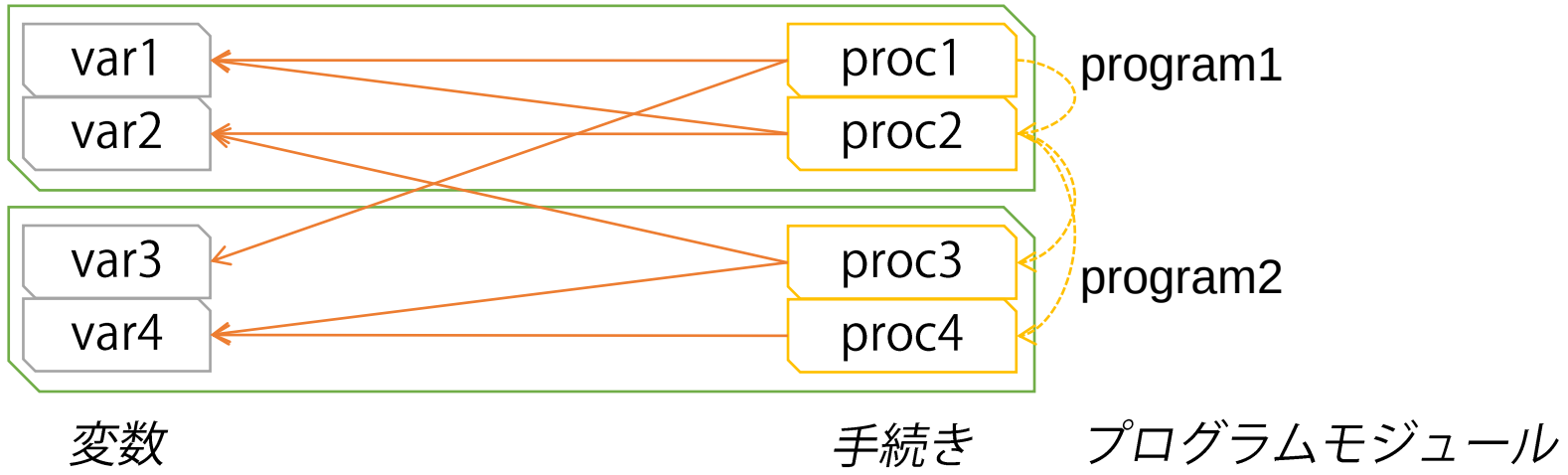
まずは開発プロセスの
全体像を見てみる

目次

- ソフトウェア工学概観
- モデリングとプロセス
- UML

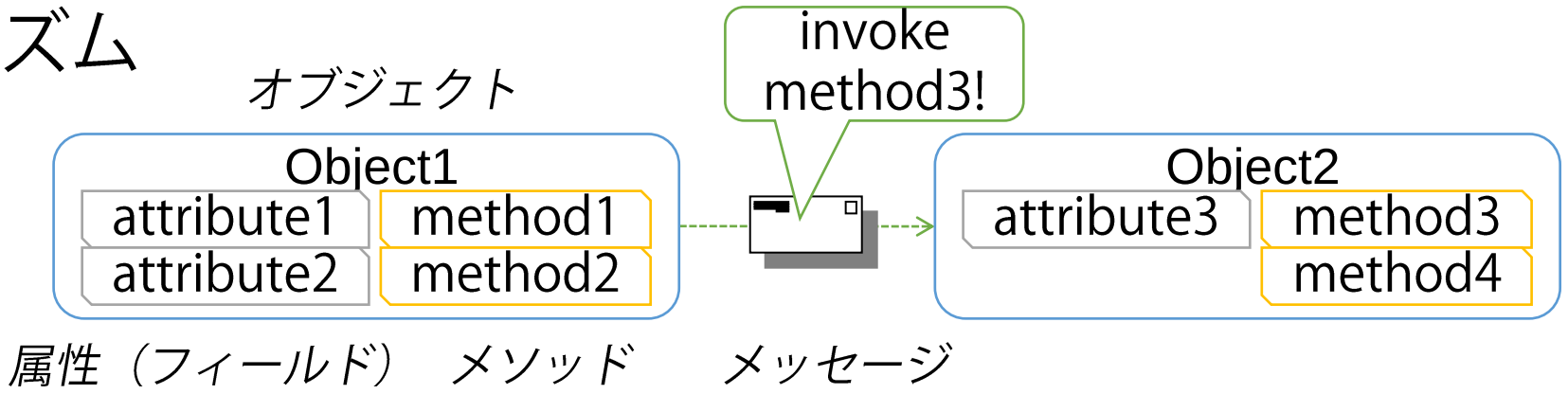
オブジェクト指向プログラミング：背景（簡略版）

- 手続き型プログラミングにおける問題
  - データと機能の暗黙的な依存関係
  - アプリケーションを分割して
    産まれた再利用性の低い手続き



オブジェクト指向プログラミング：定義（簡略版）

- オブジェクト指向プログラミング
  - データとそのデータに対する機能をオブジェクトとして部品化
  - メッセージングによるカプセル化された機能の利用
  - 継承・ポリモフィズム



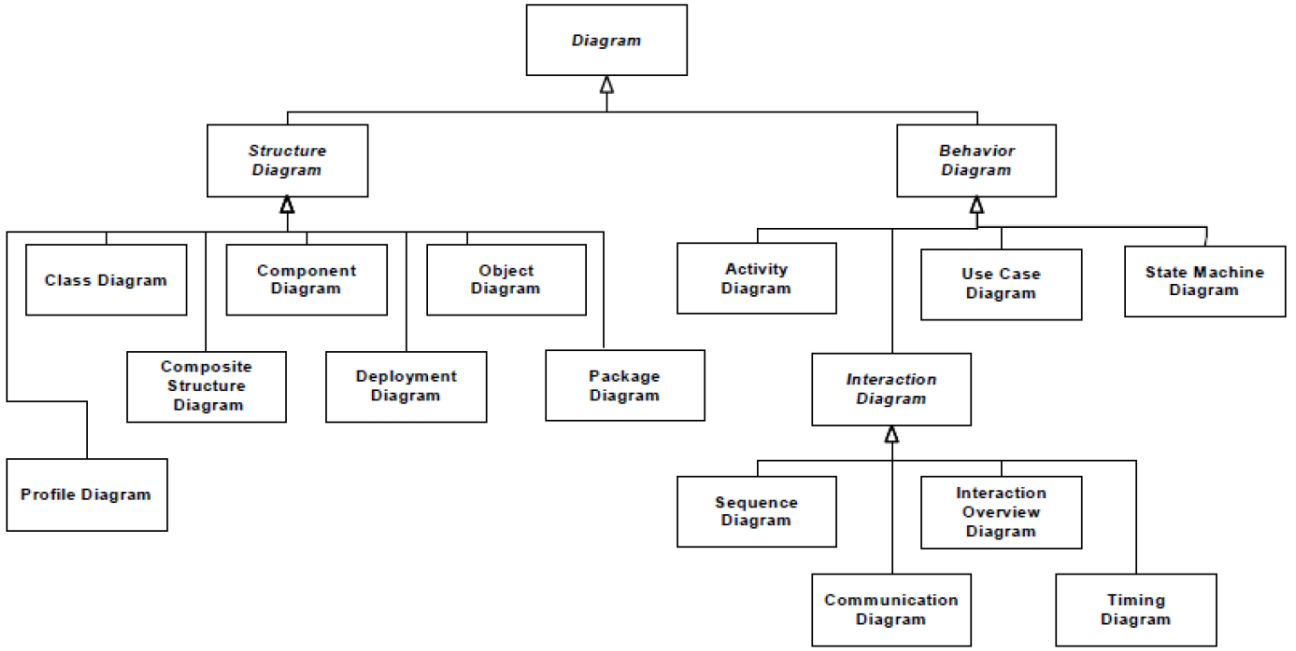
- 考え方は抽象度が高い段階でも活用可能！
  （オブジェクト指向分析，オブジェクト指向○○）

UML

UML: Unified Modeling Language

- 図形式によるオブジェクト指向モデリング言語
- 役割の異なる14個の図を定義
- OMG (Object Management Group) による標準
  - 1996年に Ver. 1.0
  - 2017年12月 Ver. 2.5.1

[https://www.omg.org/spec/UML/]

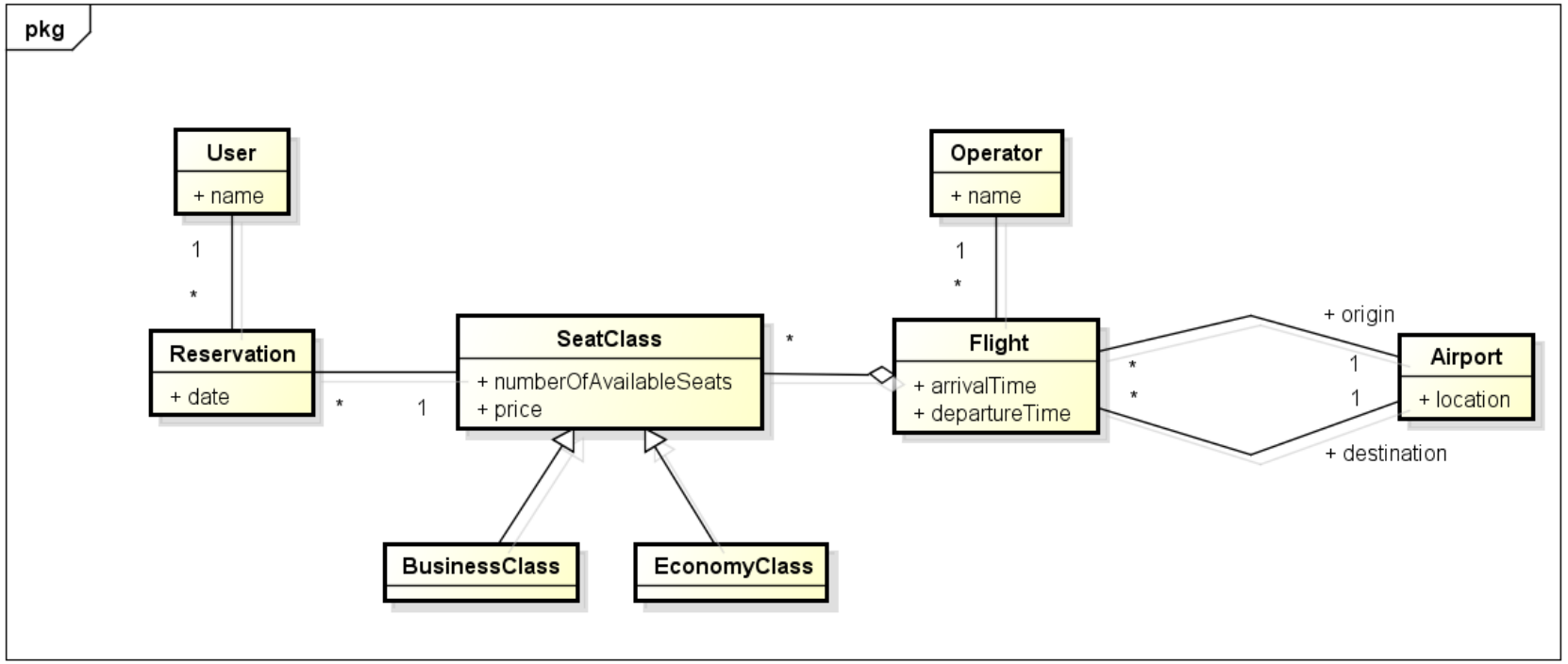


UMLの記法により
「UMLで定義された図」
という概念を表記

矢印は「汎化」（is-a関係）
（上が下の一般的な概念）

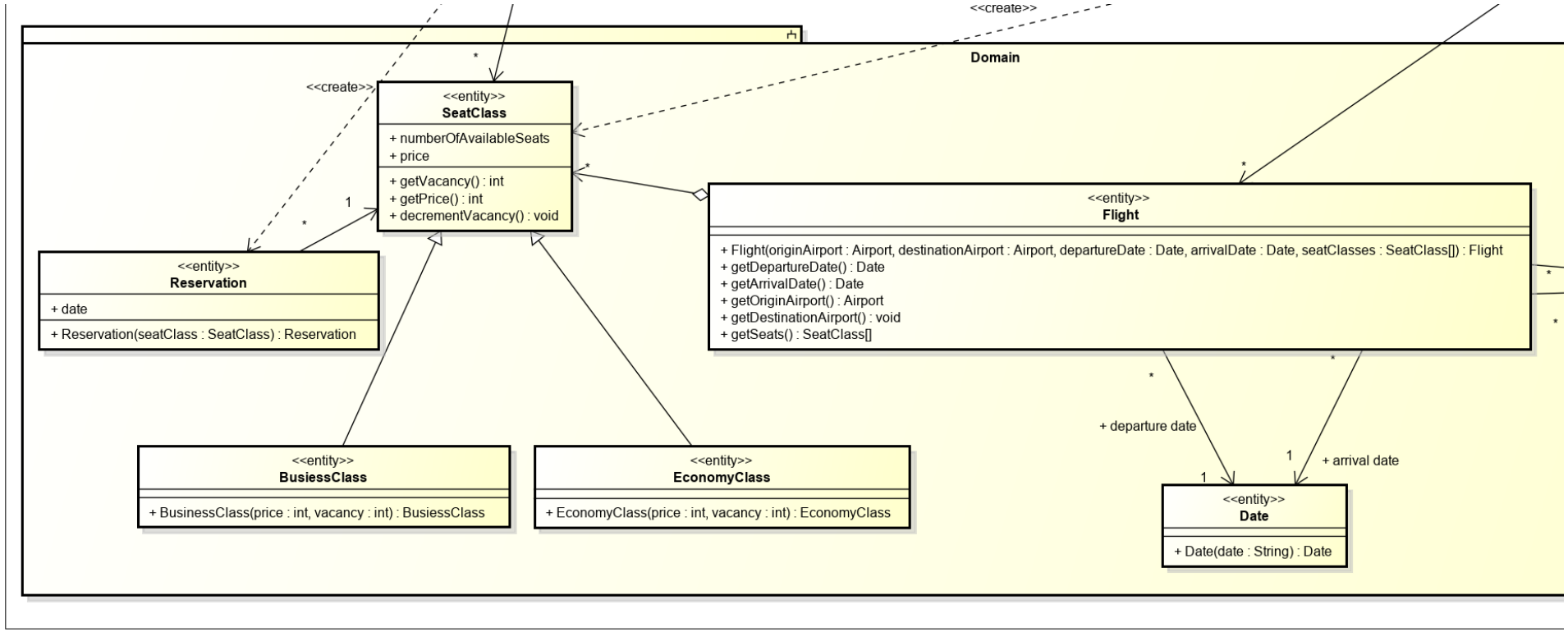
例：クラス図（概念定義）

- 航空券予約業務に関する概念



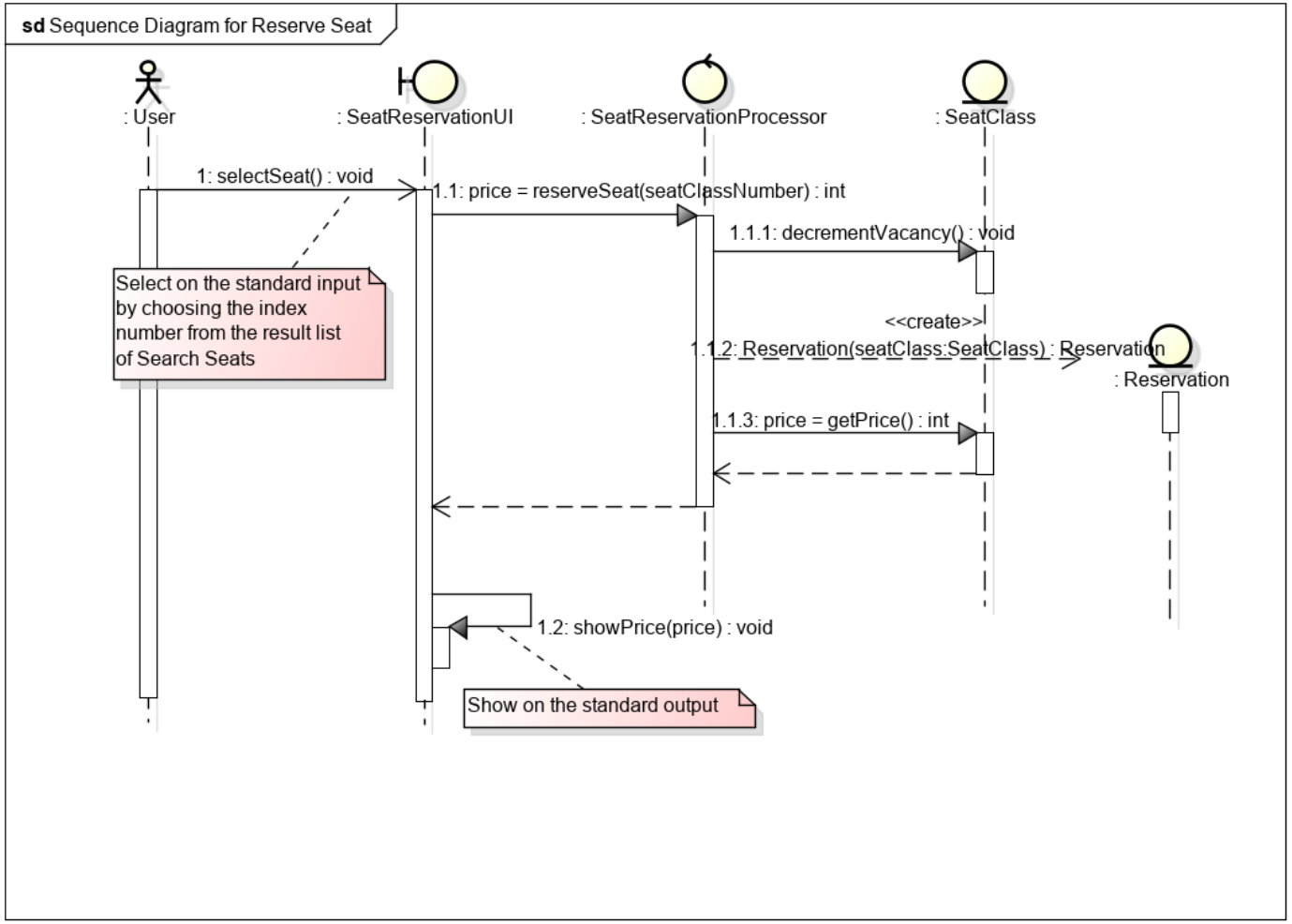
例：クラス図（プログラムの設計）

- 航空券予約システムの設計



例： シーケンス図

- 航空券予約に関する相互作用



今回の参考文献

- SWEBOK [https://www.computer.org/education/bodies-of-knowledge/software-engineering]
  - 正確な用語定義を振り返りたいときなど
- 目次を紹介した書籍3つ
  - 実践ソフトウェアエンジニアリング（第9版）Pressman and Maxim, オーム社，2021
  - ソフトウェア工学の基礎（改訂新版）玉井，岩波書店，2022
  - Googleのソフトウェアエンジニアリング，Wintersら，オライリー，2021

まとめ

- ソフトウェアの重要さは言わずもがな
- ソフトウェア工学
  - ソフトウェアの開発・運用・保守に関することがらを扱う
  - 抽象的な「解きたい問題」 「やりたいこと」から
    具体的なソフトウェアによる解法に至る過程を支援
  - 抽象的な途中成果物や複雑なシステムの本質，あるいは開発の活動をとらえ，共有，分析や検証するためのモデリングが重要