

ずっと受けたかった ソフトウェアエンジニアリングの 新人研修

B3 60161018 久木田 雄亮





文献紹介



監修： 宇治則孝
著者： 大森久美子
岡崎義勝
西原琢夫

発行人： 佐々木幹夫
発行所： 株式会社 翔泳社
初版： 2009年4月9日

ページ： P24まで



第1章 ソフトウェア開発の概要

- ソフトウェアエンジニアリングの意味と歴史
- 代表的な分析・設計手法
- 代表的な開発手法



ソフトウェアエンジニアリングの意味

- **ソフトウェアエンジニアリングとは？**

プログラム開発を工学としてとらえ、合理的な様々な方法論と開発手法を用いながら、早く、安く、正確に、誰でもプログラムを作成できることを目指す学問である。

- **ソフトウェア危機**

ソフトウェアの大規模化、複雑化に伴い、品質や生産性の低下が大きな問題となったこと



ソフトウェアエンジニアリングの歴史(1)

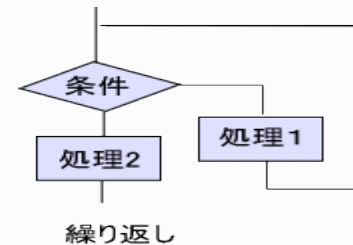
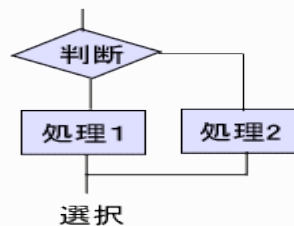
- 「ソフトウェアエンジニアリングの始まり」

1968年10月にドイツのガルミッシュで行われた
「a Working Conference on Software Engineering」
で使われたのが始まり

- 「構造化」

プログラムの制御処理を簡素化し明確にするもの
接続、分岐、反復の組み合わせで表現する(構造化定理)

構造化定理





ソフトウェアエンジニアリングの歴史(2)

- 「管理技術」

1980年代に幅を利かせていた

開発に関する人やものなどを管理する

大規模なソフトウェアの開発では適切管理することが不可欠

- 「オブジェクト指向」

1990年以降のソフトウェアエンジニアリングの重要なキーワード

主要な概念は1970年代に確立

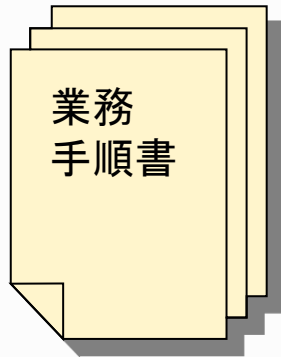
オブジェクト指向は分析・設計にも取り入れられる



代表的な分析・設計手法(1)

- 分析と設計

システム開発にはプログラミングするのに必要なシステムの設計書を用意する必要がある



分析

設計



代表的な分析・設計手法(2)

- **業務手順書**

顧客が求めるシステムの内容

例: チケット発行業務

- **システム要件**

システムに必要な機能を定めたもの

例: チケットを検索するシステム

- **設計書**

各機能をどのように作るかを定めたもの

例: 検索結果はデータベースYからとってくる



代表的な分析・設計手法(3)

- **分析**

「何を作るか」を明らかにする作業

顧客から要望や現在の業務の流れを聞きだす

顧客が求める機能や性能をシステム要件として整理する

- **設計**

「どうやって作るか」を明らかにする作業

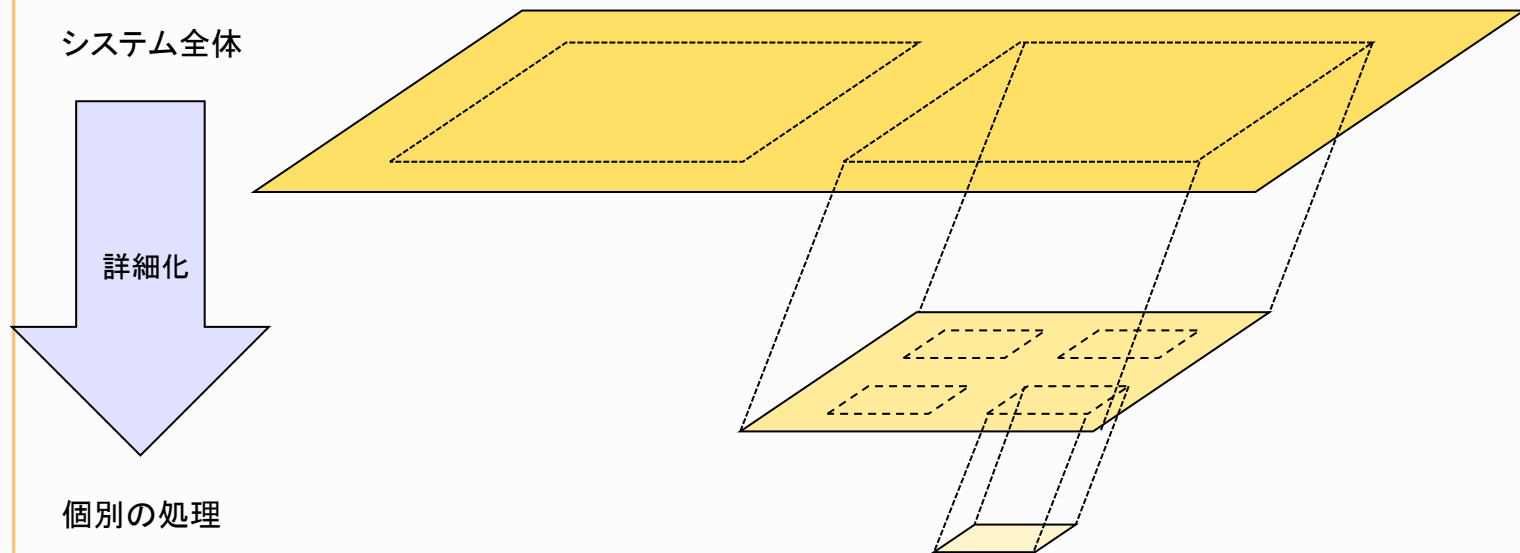
分析により明らかになった要件をもとにどのような形でプログラムしていくかを決定する



代表的な分析・設計手法(4)

- 構造化による分析と設計

すべてのシステムはそれより小さな要素の組み合わせで構成され、その要素も、またさらに小さい要素の組み合わせで構成されると考える。



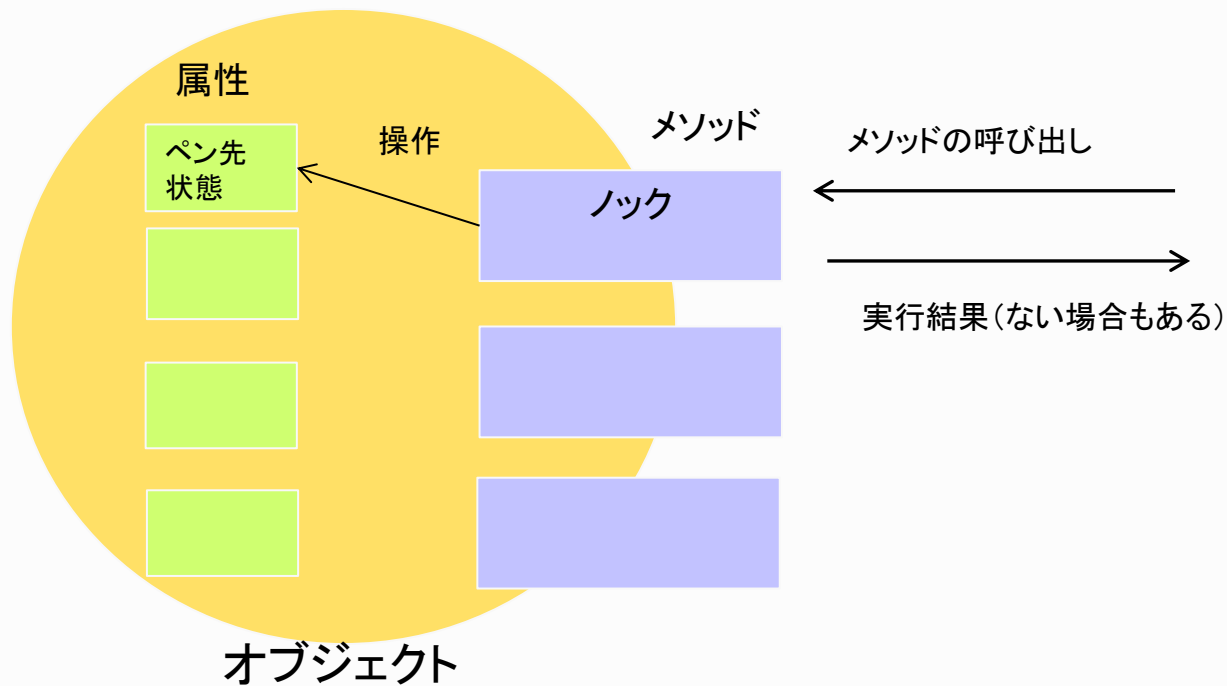


代表的な分析・設計手法(5)

- オブジェクト指向による分析と設計(1)

メソッド: プログラムに相当するもの

属性: データに相当するものの属性



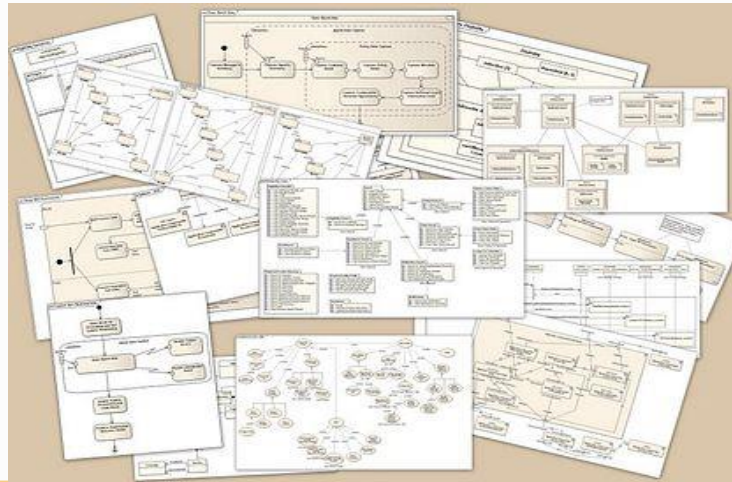


代表的な分析・設計手法(6)

- オブジェクト指向による分析と設計(2)

システムの機能をオブジェクトに分割し、オブジェクト間の呼び出し関係を定義します。

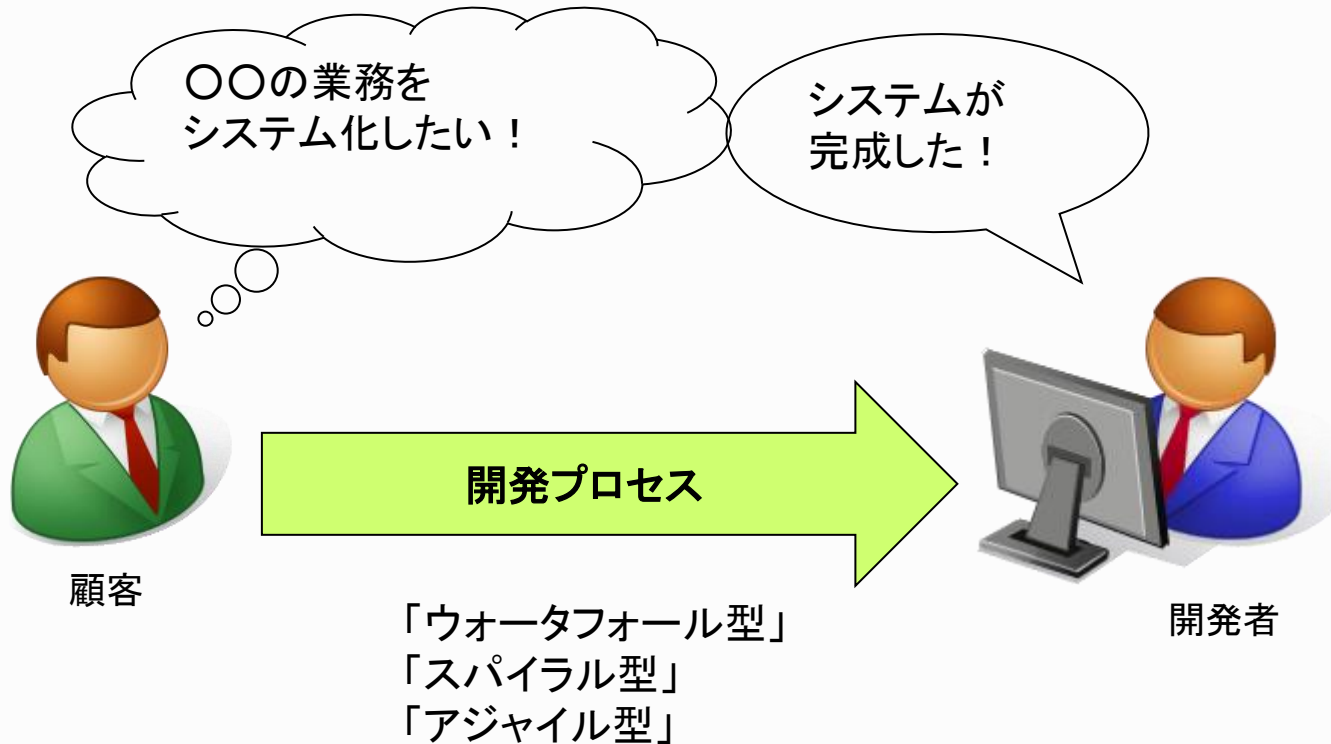
UML(Unified Modeling Language)
という専用の図法が用いられる(詳しくは第2章)





代表的な開発手法(1)

- 開発プロセス

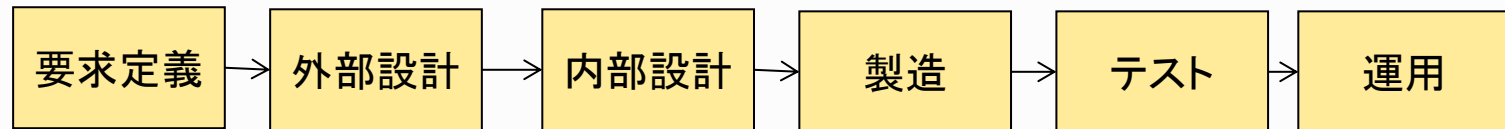




代表的な開発手法(2)

- **ウォーターフォール型**

- 基本的に後戻りせず時系列に沿って進める



要求定義: 顧客が求める機能や性能を定義する

外部設計: システムが持つべき機能を定める

内部設計: 機能の具体的な実現方法を定める

製造: プログラムを製造する

テスト: 設計通りに動作するか確認する

運用: システムを稼働させ、不具合修正や簡単な機能の追加を行う

- **問題点**

不具合が最後の工程まで見つからないことが多い

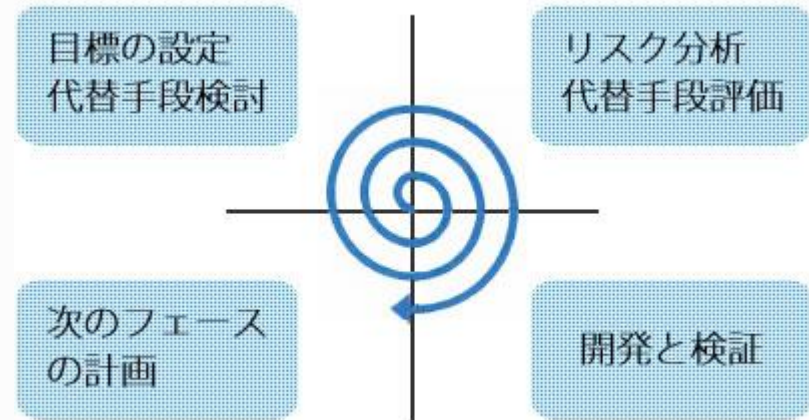
「ムービングターゲット」に柔軟に対応できない



代表的な開発手法(3)

- **スパイラル型**

ウォーターフォール型の開発工程にフェーズを設けてそれを繰り返しながら開発する



開発の前半にプロトタイピングを行う
プロトタイピングにより動作を具体的にイメージできる
分析や設計の段階での不具合を取り除ける



代表的な開発手法(4)

- **アジャイル型**

設計時に最適だったシステムが完成した時には用をなさない
開発期間を短縮し、開発費を安くすることが求められる

アジャイル宣言 (Agile Alliance, 2001年)

- プロセスやツールより、個人そのものや個人間の交流を重視せよ
- 広範に渡る大量の文書作成より、きっちり働くソフトウェアの作成に注力せよ
- 契約に関わる交渉より、顧客と協調することに重点をおけ
- 無理に計画に従うより、目の前の変化へ柔軟に対応せよ



代表的な開発手法(5)

- **XP(1)**

代表的なアジャイル型の開発手法

小規模で効率よくプログラミングするという視点で考えられた

キーワード

「コミュニケーション」、「シンプル」、「フィードバック」、「勇気」

小さい単位で開発を行い、常にテストを行っているので不具合が見つかれば直ちに修正される



代表的な開発手法(6)

• XP(2)

