UMLによる
オブジェクト指向分析・設計入門

株式会社　豆蔵
重見　剛

趣旨

- “UMLとは?なぜUML?”を理解する
- 典型的なオブジェクト指向分析・設計の作業に沿って、UMLのダイアグラムを理解する

対象

- これから、OOで開発をはじめたい方
- UMLでモデリングしたい方
- 必要な知識
  - OOの基本概念

目次

- UMLの概要
  - UMLによる統一
  - 開発プロセス
- UMLの適用
  - オブジェクト指向分析・設計
  - 要求分析
  - システム分析
  - システム設計

UMLの概要

- UMLによる統一
- 開発プロセス

さまざまな開発方法論

- Booch法. . . 設計にフォーカス
- OMT法. . . 分析にフォーカス、3つの視点(オブジェクト、動的、機能)のモデル
- OOSE(Objectory)法. . . ユースケースの導入、オブジェクトの3カテゴリー(boundary、control、entity)
- Coad/Yordon法. . . 初めてのOOAD手法
- Shlaer/Mellor法. . . 組み込みに特化
- Drop. . . 日本発のOO開発方法論
- ・・・

さまざまな開発方法論

- 良い点
  - 開発の対象としているドメインや規模に応じた方法論を選択できる
- 問題点
  - 方法論毎に、開発の手順や適用するモデルおよびその描き方を覚えなければならない
  - 異なる方法論間では、モデルをそのまま流用できない。必ずのモデルの翻訳作業が発生する

方法論の構成要素

- 方法論は、開発プロセスと表記法で構成される
  - 開発プロセス
    - システム開発における、作業手順、成果物、管理項目や管理方法などをまとめたもの
  - 表記法
    - 開発の中で作成されるモデルの表現方法(記述法)

開発プロセス

- 開発プロセスとは、システム開発の進め方を定義したもの
  - 作業工程を定義
  - 各作業工程における、作業内容と手順を定義
  - システム構築の過程で用いるモデルを定義
  - 管理項目およびその管理方法を定義
  - 要員構成と導入時期を定義
  - ・・・

表記法

- モデルを表現する方法
  - 開発プロセスで定義されたモデルを表現するためのダイアグラムとモデリング要素およびその意味(セマンティクス)を定義したもの
- 利害関係者間でのコミュニケーションツール
  - 頭の中にあるイメージを具体化(視覚化)するための道具
  - ビジュアルなモデルを通して意思の疎通を図る

UML
(Unified Modeling Language)

- 統一モデリング言語
- 表記法の統一
- ブーチ、ランボー、ヤコブソンが中心となりまとめる
- 複数の方法論からのいいとこ取り
- ’97　11月　OMGによって標準のモデリング言語として承認される
- 業界のデファクトスタンダード

UML発展の経緯

- 1994末：開発着手
  - オブジェクト指向開発の統一方法論(Unified Method)として、ブーチ、ランボーらにより開発開始
- 1996初：UML 0.9発表
  - ヤコブソンが加わり、記述法部分のみをUMLとして発表
- 1997．11：OMG標準
- 1999．10：UML 1.3公開
- 現在：UML 1.4、UML 2.0開発中

UMLでモデル化する意義

- 開発の各工程で必要となる一連のモデル(ダイアグラム)が用意されている
- 開発プロセスの詳細から表記法を分離できる
  - ひとつの表記法だけを覚えればよい
  - プロセスが違っていても、モデルが理解できる
- 業界のデファクトスタンダード
  - モデリング言語における世界共通語

UMLのダイアグラム

- 要求をあらわす図
  - ユースケース図
    - ユーザーから見たシステムの機能を表現する

UMLのダイアグラム

- 構造をあらわす図
  - クラス図
    - クラスおよびクラス間の静的な構造を表現する
  - オブジェクト図
    - クラス図で表現された静的な構造の、ある時点におけるスナップショット(インスタンス間の関係)を表現する
  - パッケージ図
    - クラス図の特殊なもの

UMLのダイアグラム(続き)

- 振る舞いをあらわす図
  - 相互作用図：シーケンス図，コラボレーション図
    - 複数のオブジェクトによる協調動作を表現する
  - ステートチャート図
    - オブジェクトのライフサイクルにおける状態遷移を表現する
  - アクティビティ図
    - 業務や処理のフローを表現する

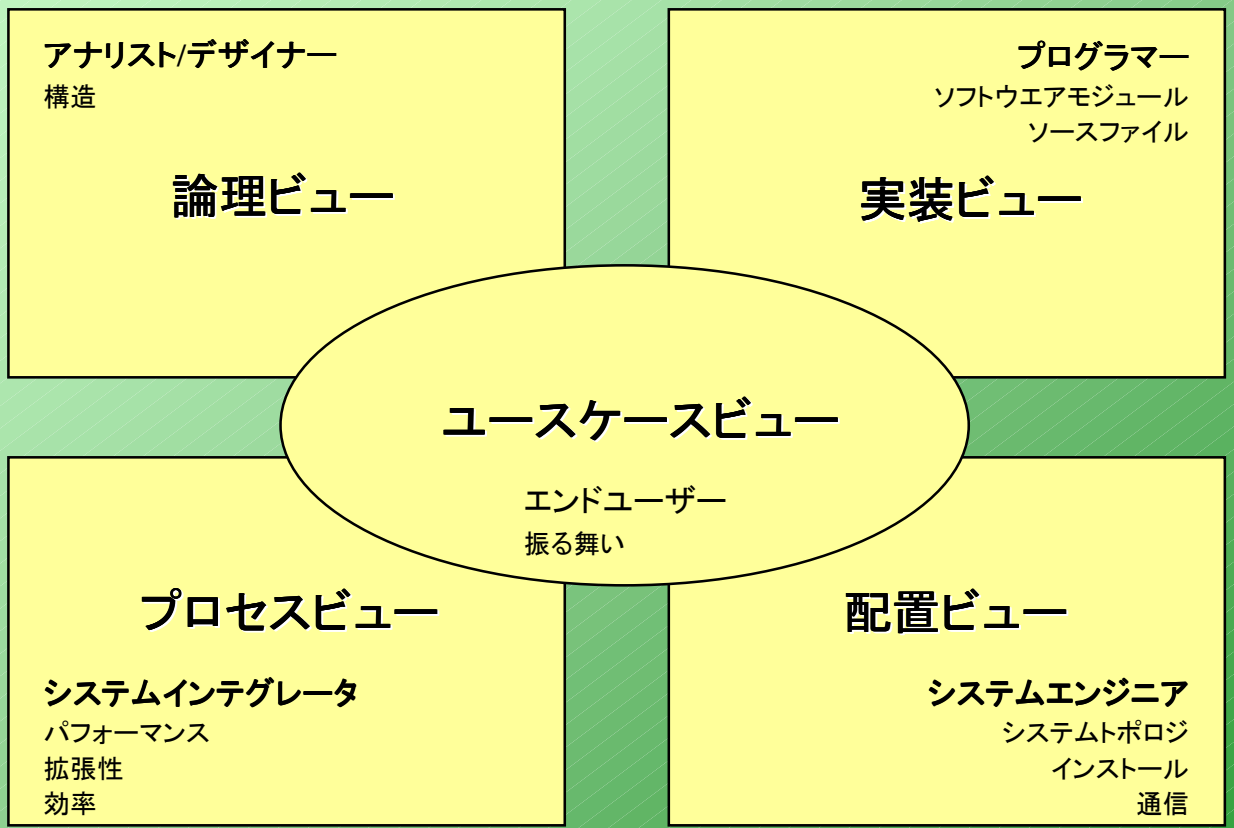
UMLのダイアグラム（続き）

- 実装をあらわす図
  - コンポーネント図
    - ソフトウエアの物理構成を表現する
    - ソフトウェアモジュール間の依存関係を表現する
  - 配置図
    - コンポーネントやオブジェクトのノード上への配置構成やノード間の接続仕様等を表現する

複数のダイアグラム

- システム開発には、様々な人々が関与し、それぞれの視点でシステムを捕らえている
- システムについて興味のあることを議論したり確認するためのモデルが不可欠である

アーキテクチャと4+1ビュー



UMLの概要

- UMLによる統一
- 開発プロセス

最新の開発プロセス

- 統一プロセス：Unified　Process
  - UMLによるモデリングを前提とした反復型開発プロセス
  - Booch法、OMT法、Objectory法の統合
- XP：eXtream　Programming
  - 開発リスクを早期に軽減することを主眼におき，反復型の開発を取り入れている
    - コーディングおよびテストに重点を置いている
    - 初期設計よりもリファクタリングによる再設計を重視している
    - UPよりもライトウェイトな開発プロセスである

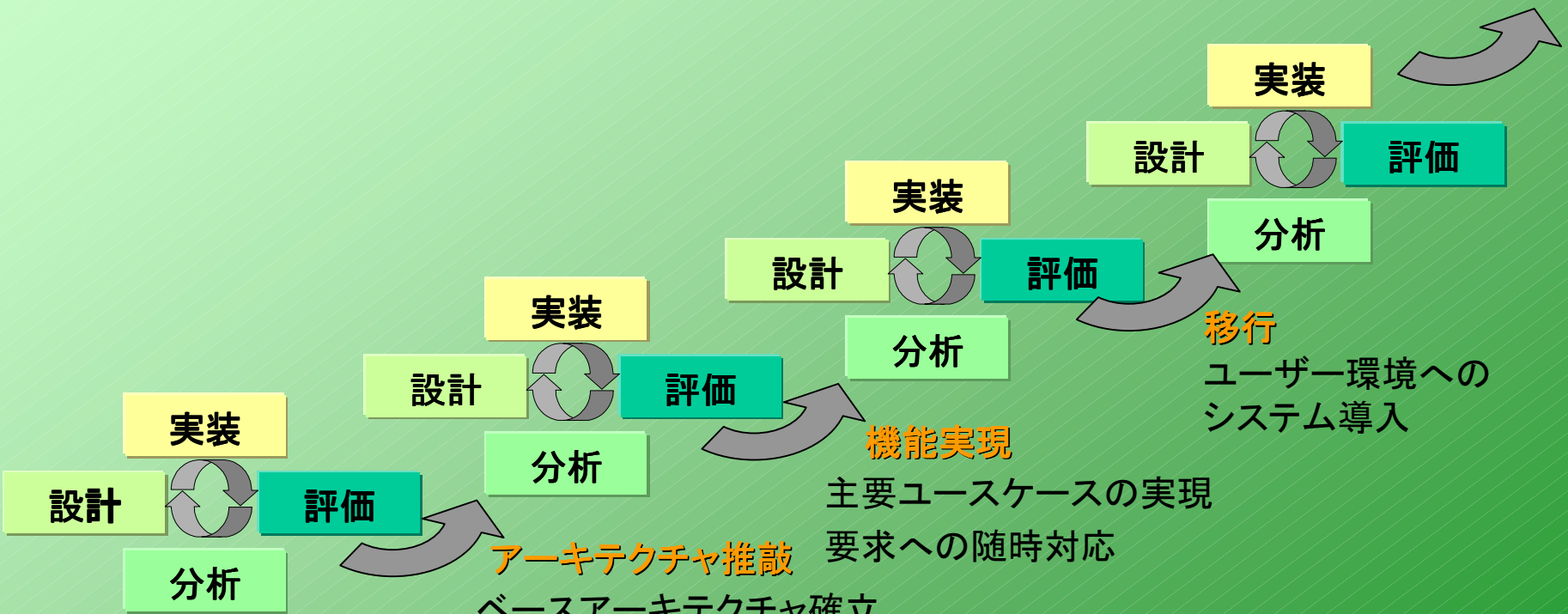
統一プロセス

- 管理された繰り返し開発
  - 要求、成果物、リスク、人、スケジュールなどを管理しながら反復開発を実施する
- ユースケースドリブン
  - 開発の単位としてユースケースを使用する
- アーキテクチャ中心
  - アーキテクチャを中心に開発を進める
- カスタマイズ可能
  - プロセス自身をカスタマイズできる

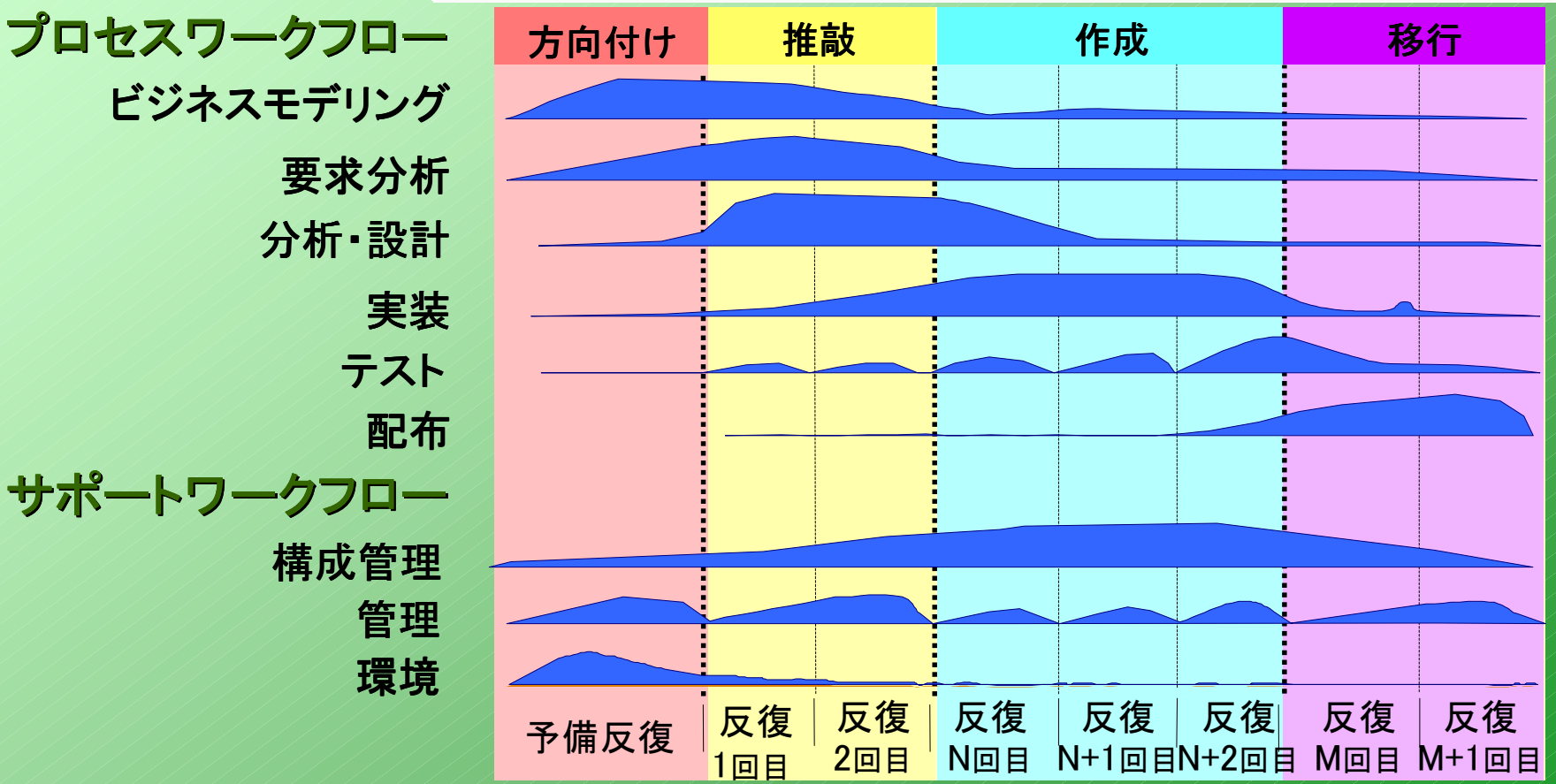
統一プロセスのフェーズ

- 方向つけ
  - システムの範囲を決める
- 推敲
  - アーキテクチャベースラインを構築する
- 作成
  - 実行可能なリリースを繰り返す
- 移行
  - ユーザーへプロダクトをリリースする

開発スタイル



統一プロセス

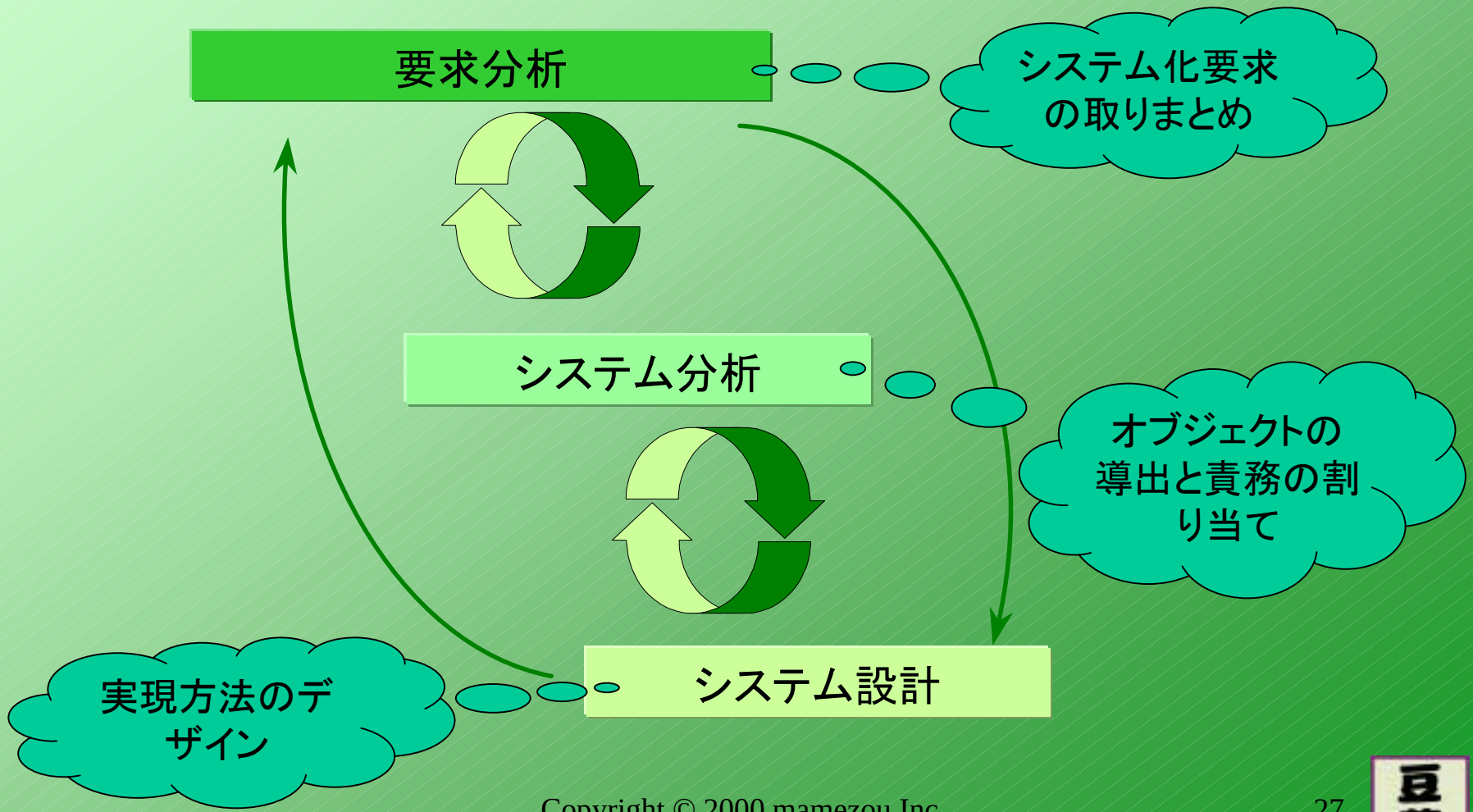


フェイズごとに各プロセスコンポーネントの重要度は変化する

UMLの適用

- オブジェクト指向分析・設計
- 要求分析
- システム分析
- システム設計

反復の中での分析・設計作業



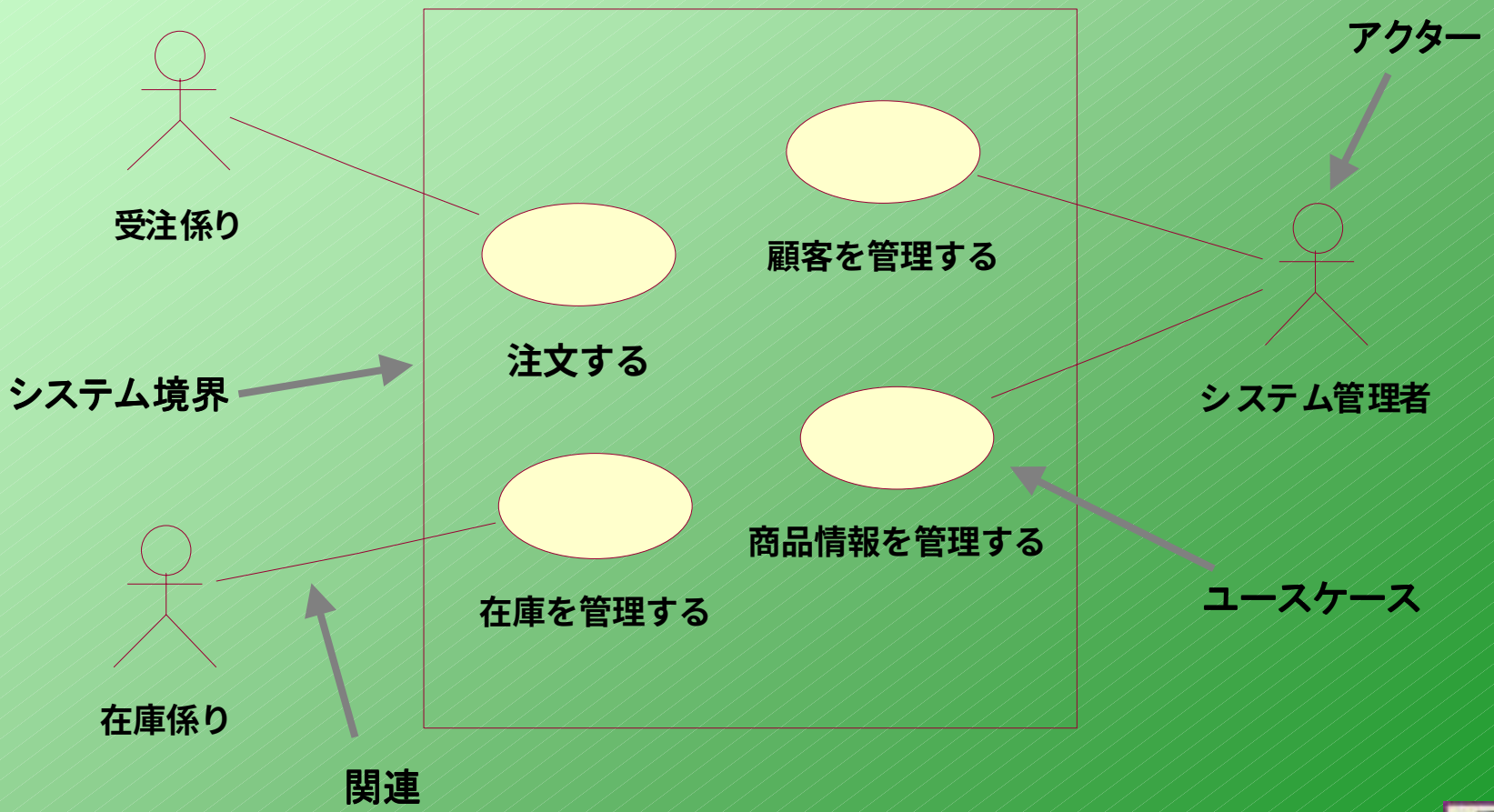
UMLの適用

- オブジェクト指向分析・設計
- 要求分析
- システム分析
- システム設計

要求分析

- 機能要求：ユーザーから見える機能
  - 注文を登録したい
  - 顧客情報を管理したい
  - 月末の集計をしたい
  - …
- 非機能要求：機能以外の要求、機能要求への制約
  - 永続化の仕組みを再利用できるようにしたい
  - 月末の集計処理は、1時間以内に完了させたい
  - 開発期間は3ヶ月としたい
  - 開発コストを、3,000万円以内に抑えたい
  - …

ユースケース図

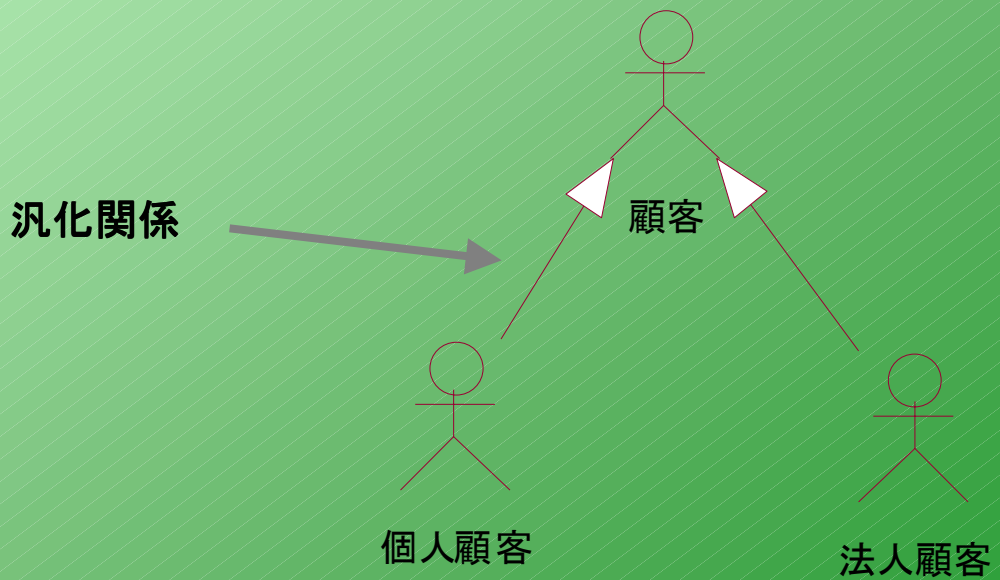


ユースケースドキュメント

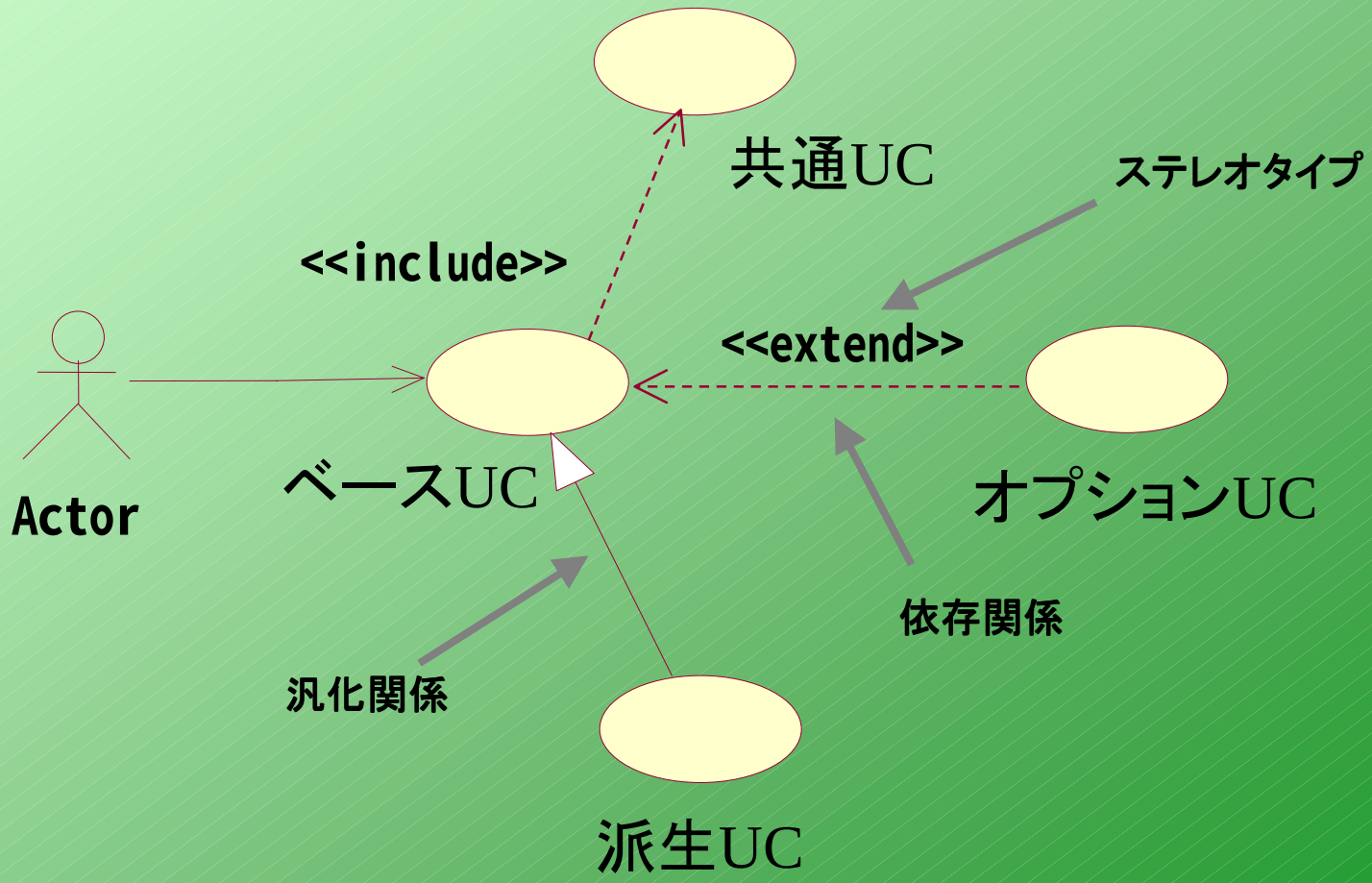
- ユースケース図だけでは、振る舞いの様子は表現できない
- アクターとシステムとの相互作用を中心にユースケースドキュメントをまとめる
  - 概要、事前条件/事後条件、イベントフロー、代替フロー、例外フロー
  - シナリオ一覧、シナリオ詳細記述

アクター間の関係

- アクター間の関係には、汎化関係が適用できる
- 抽象レベルの異なるアクター間をモデリング



ユースケース間の関係



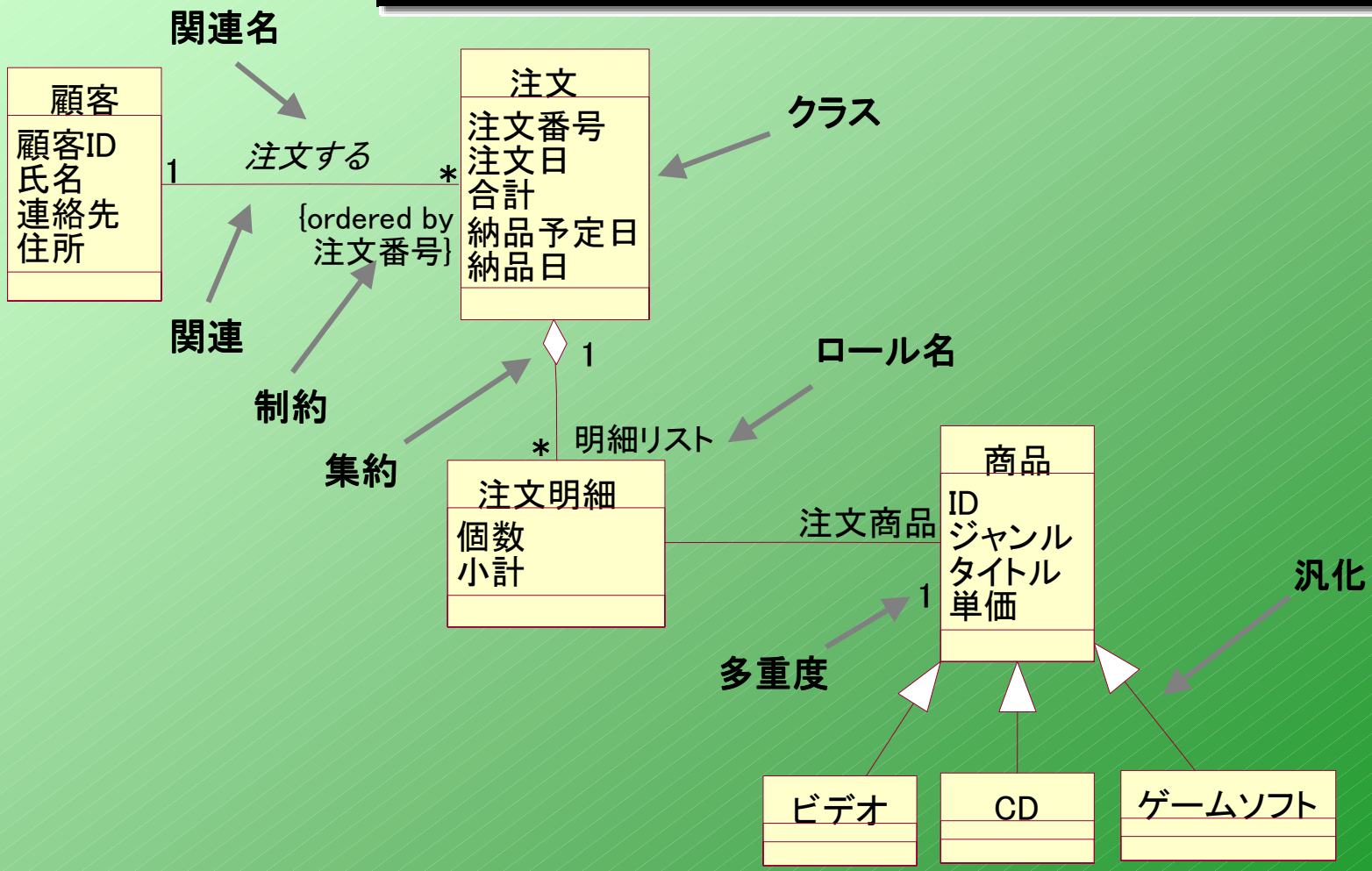
UMLの適用

- オブジェクト指向分析・設計
- 要求分析
- システム分析
- システム設計

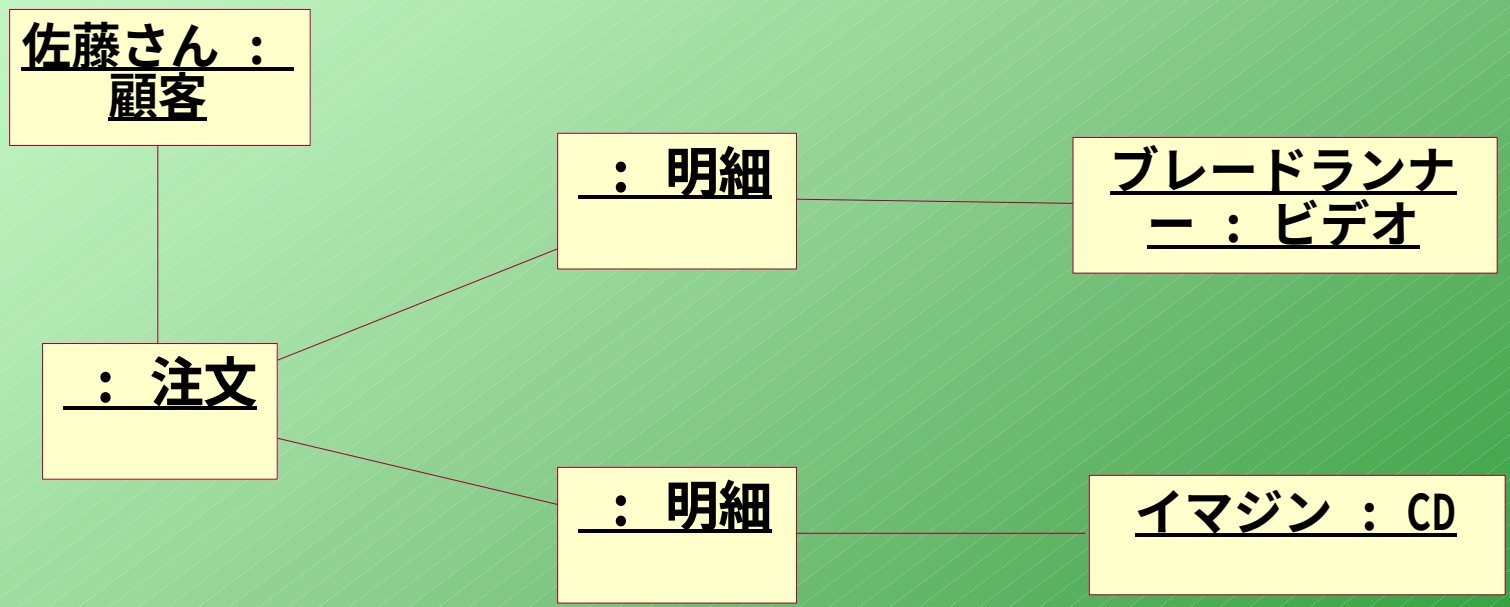
システム分析

- 要求分析の結果を入力として、機能を実現するための部品(クラス)を導出し、関係をモデリングしてシステムの静的な構造を捉える
- 機能の実行例(シナリオ)に沿って、オブジェクトの相互作用の様子を検証し、クラスに責務を与える
- システムを個々の部品(クラス)ではなく、抽象レベルを上げて見渡してみる場合もある

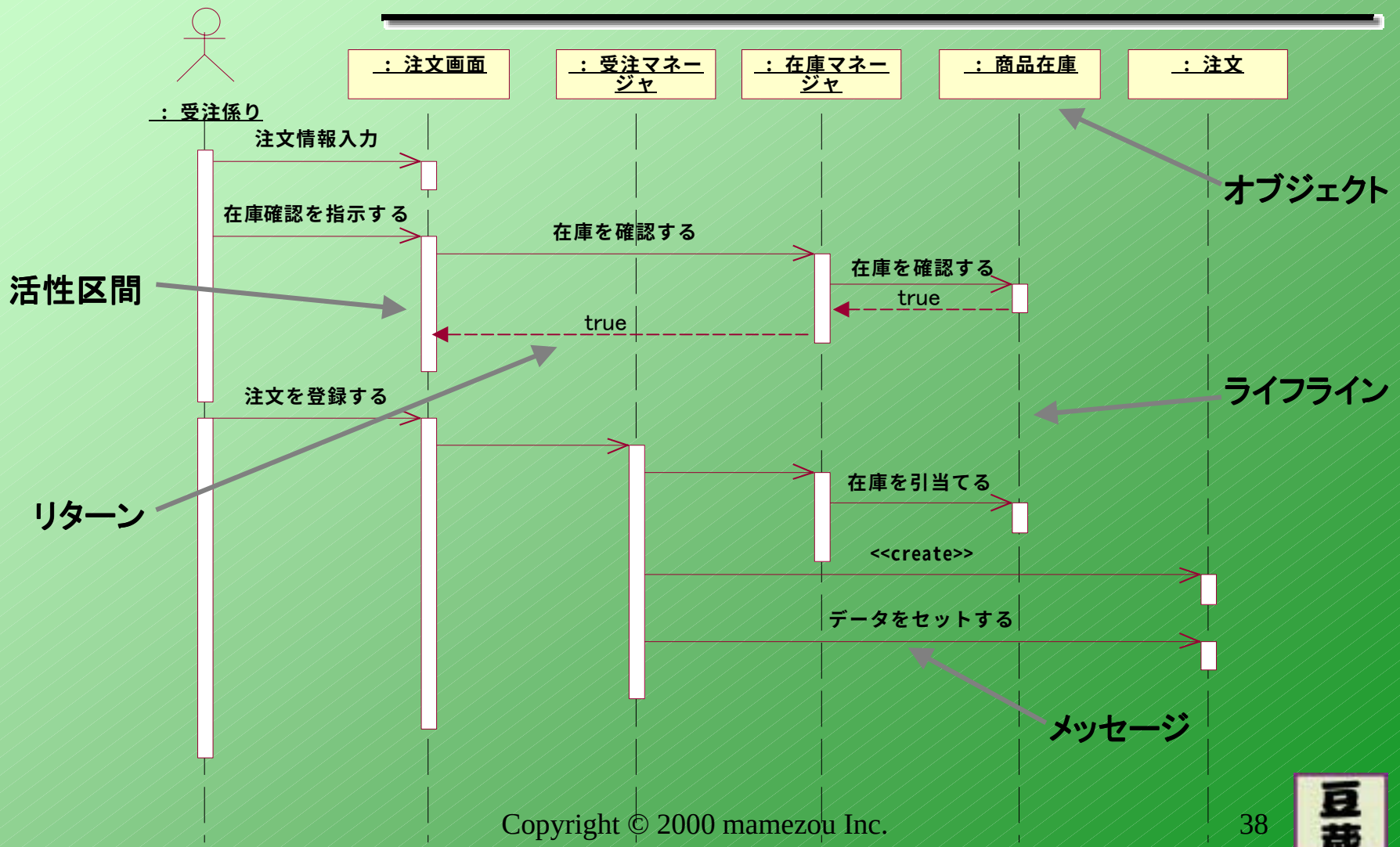
クラス図



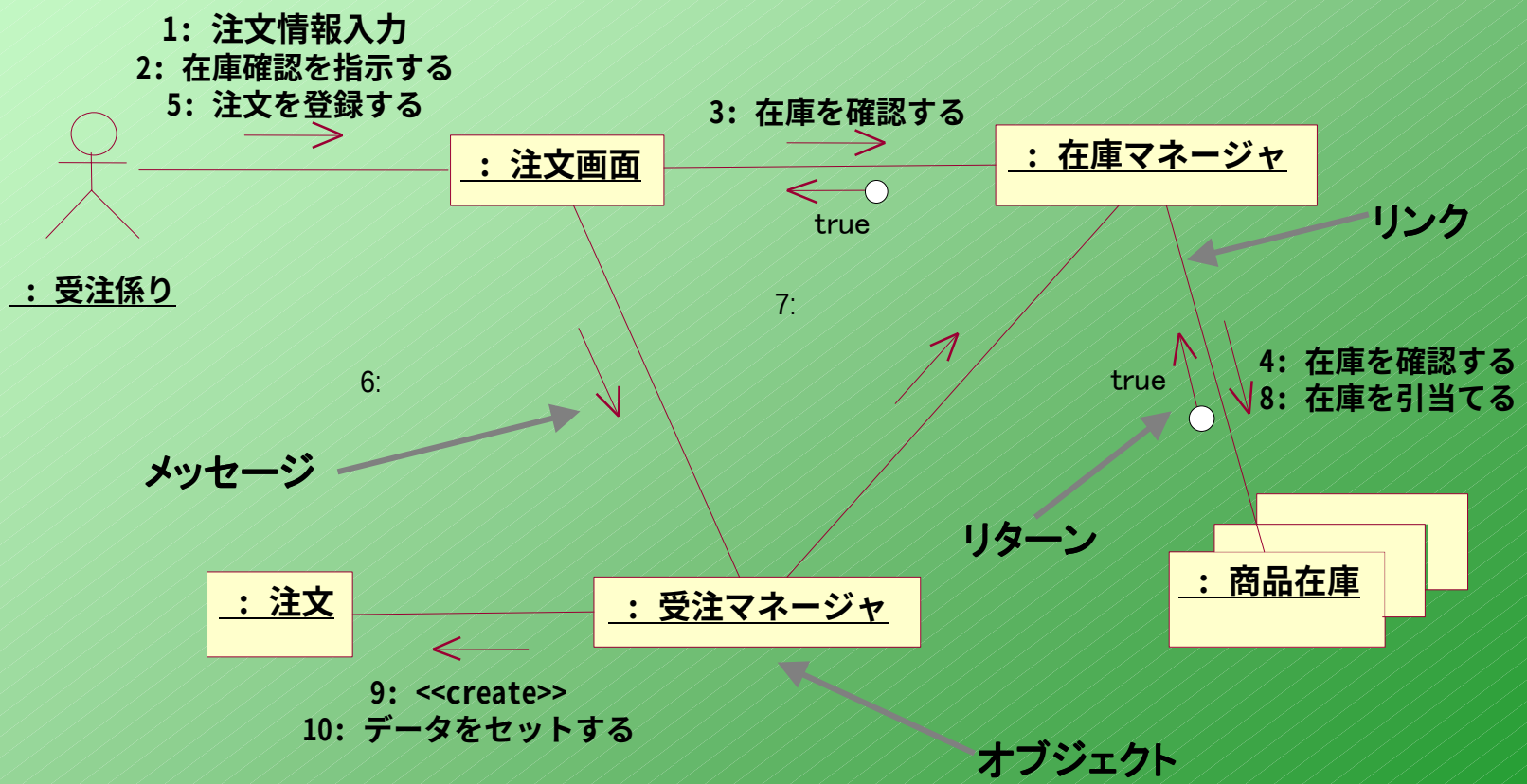
オブジェクト図



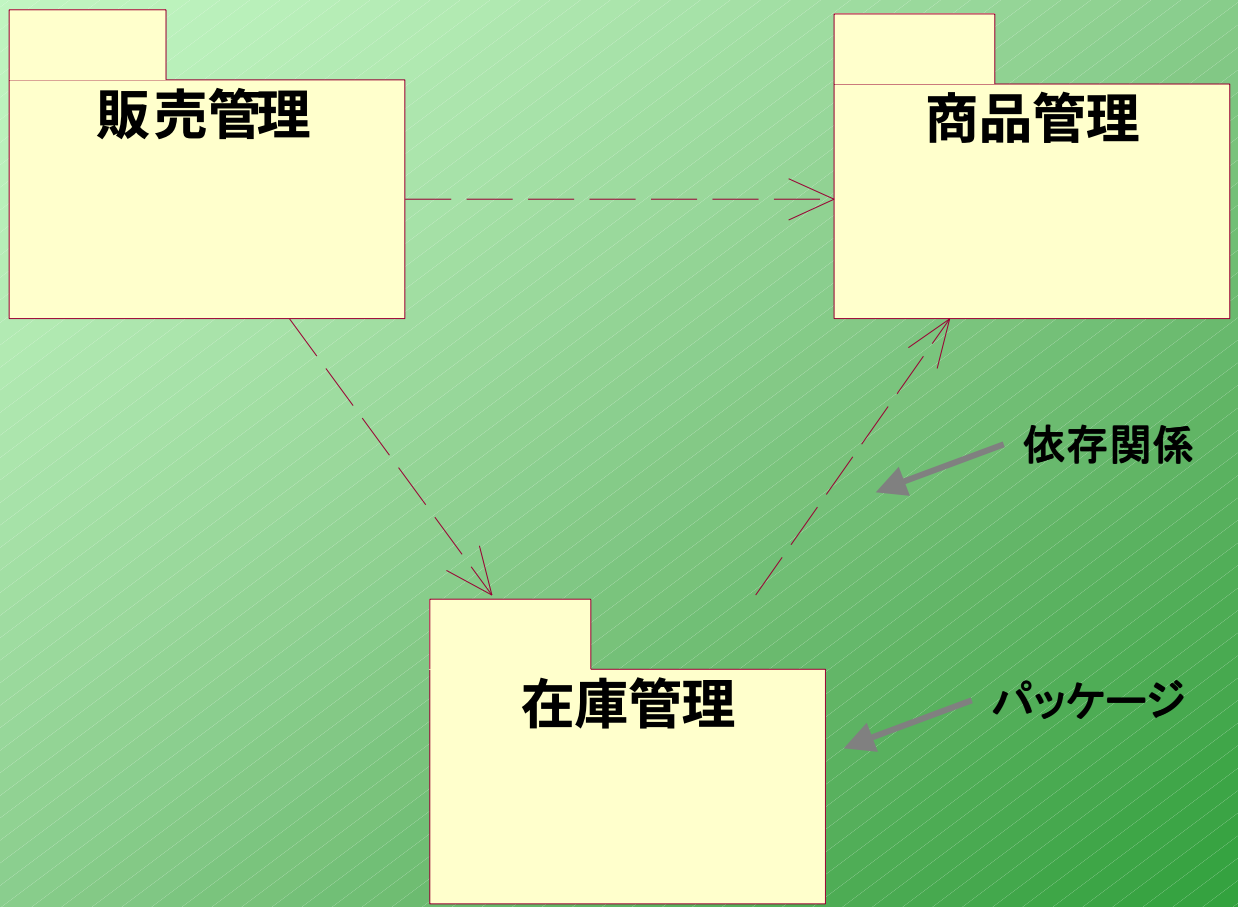
シーケンス図



コラボレーション図



パッケージ図



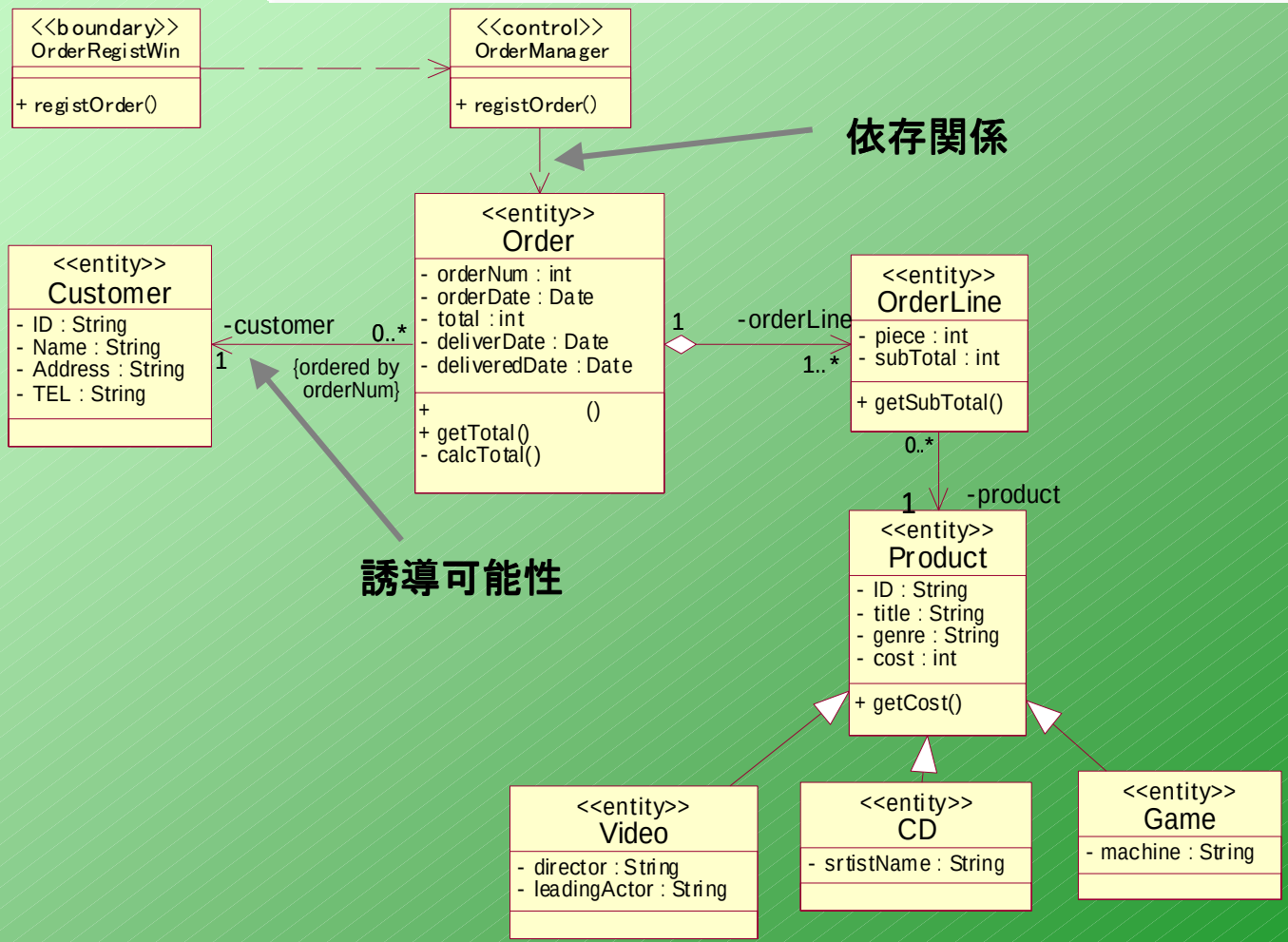
UMLの適用

- オブジェクト指向分析・設計
- 要求分析
- システム分析
- システム設計

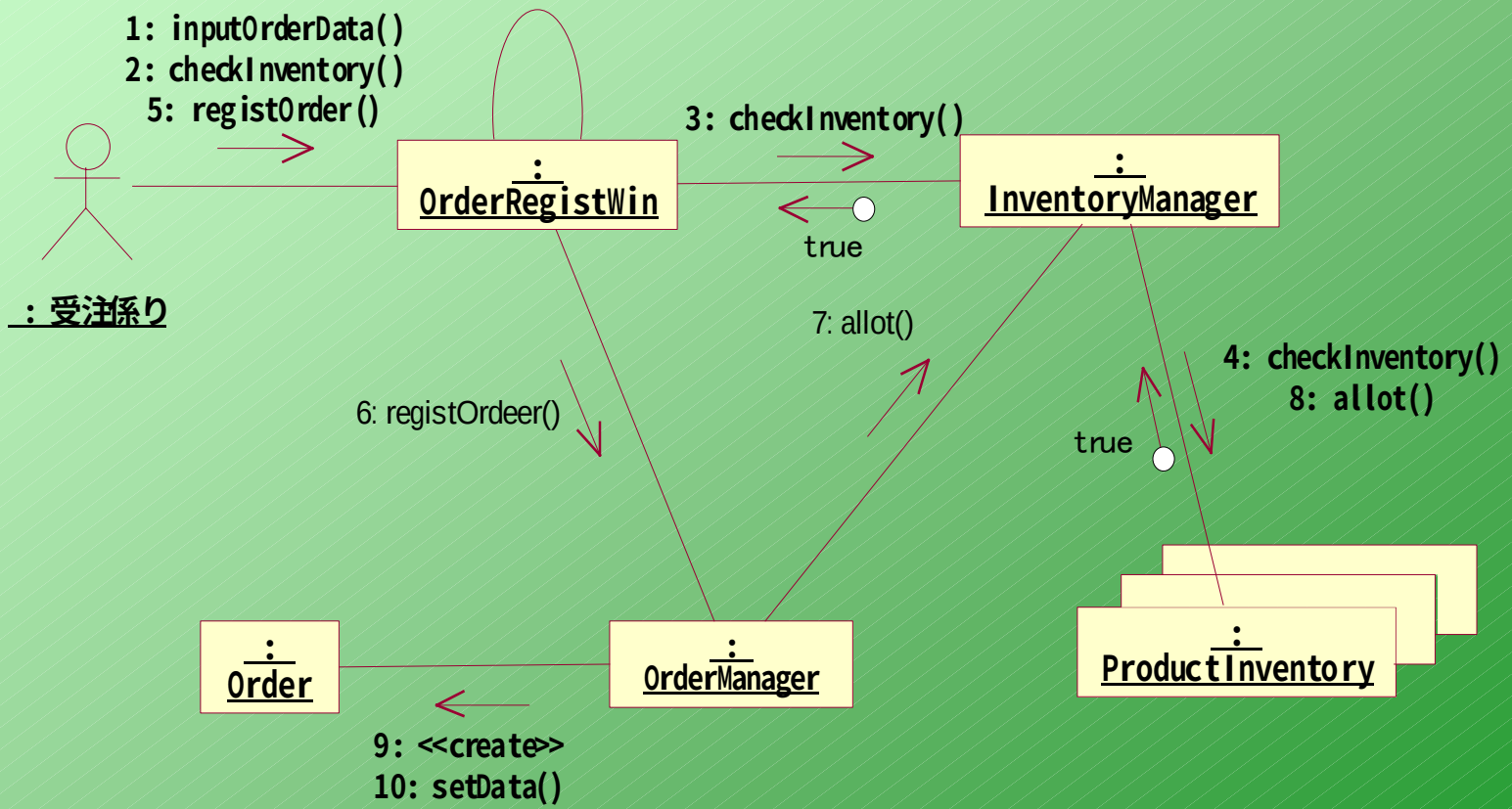
システム設計

- 分析結果を入力として、実装環境を考慮して、静的な構造に振る舞いを割り当てる
  - 分析クラスの役割の明確化
    - <<boundary>>、<<control>>、<<entity>>、…
  - 実装環境を考慮し、設計判断を加えて分析レベルから設計レベルへ移行
    - クラスの追加、関係の設計、…
  - 設計されたクラスによる、相互作用の検証

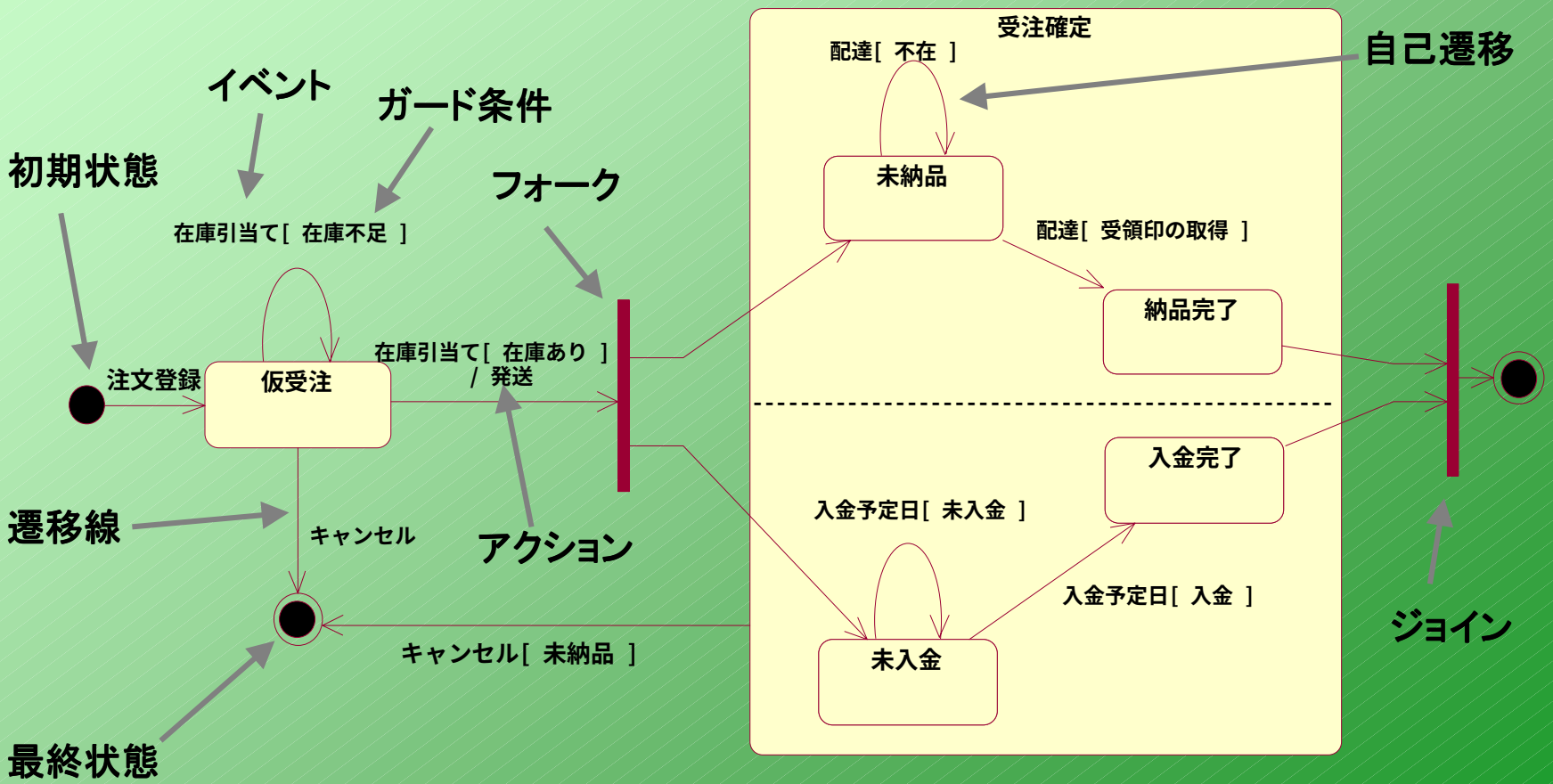
設計レベルのクラス図



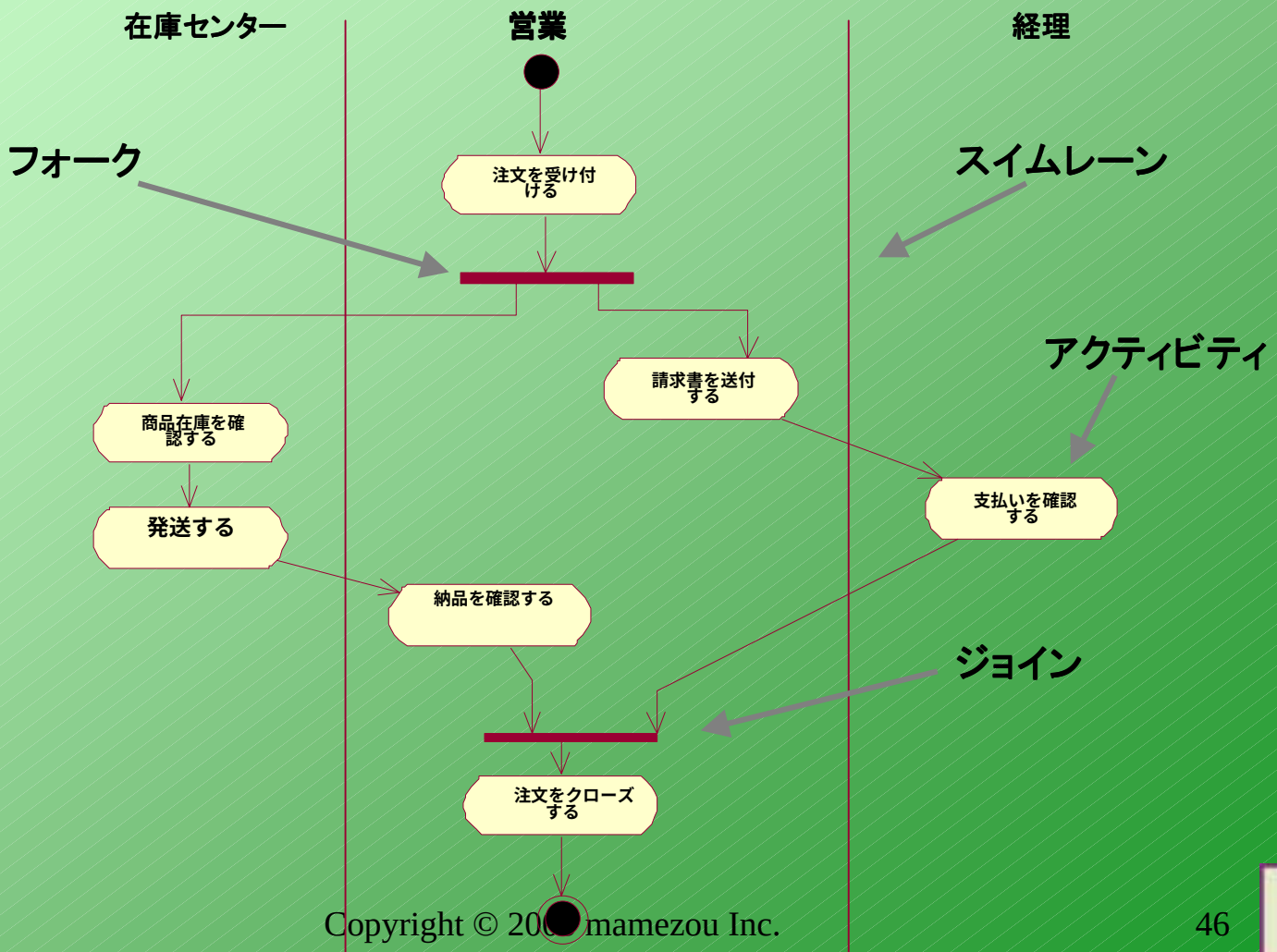
コラボレーション図



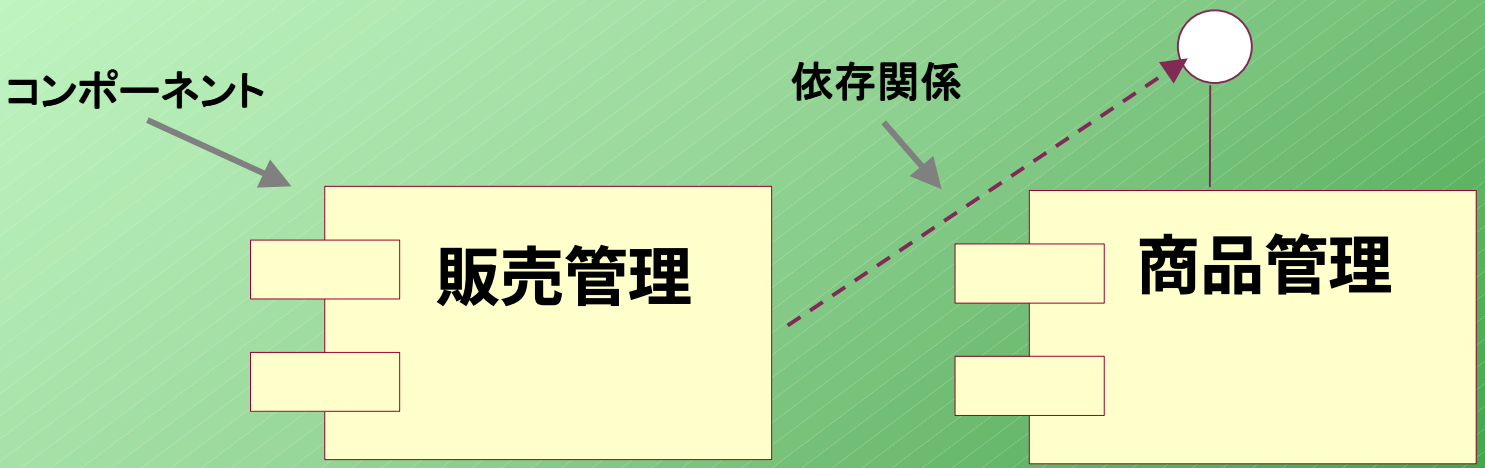
ステートチャート図



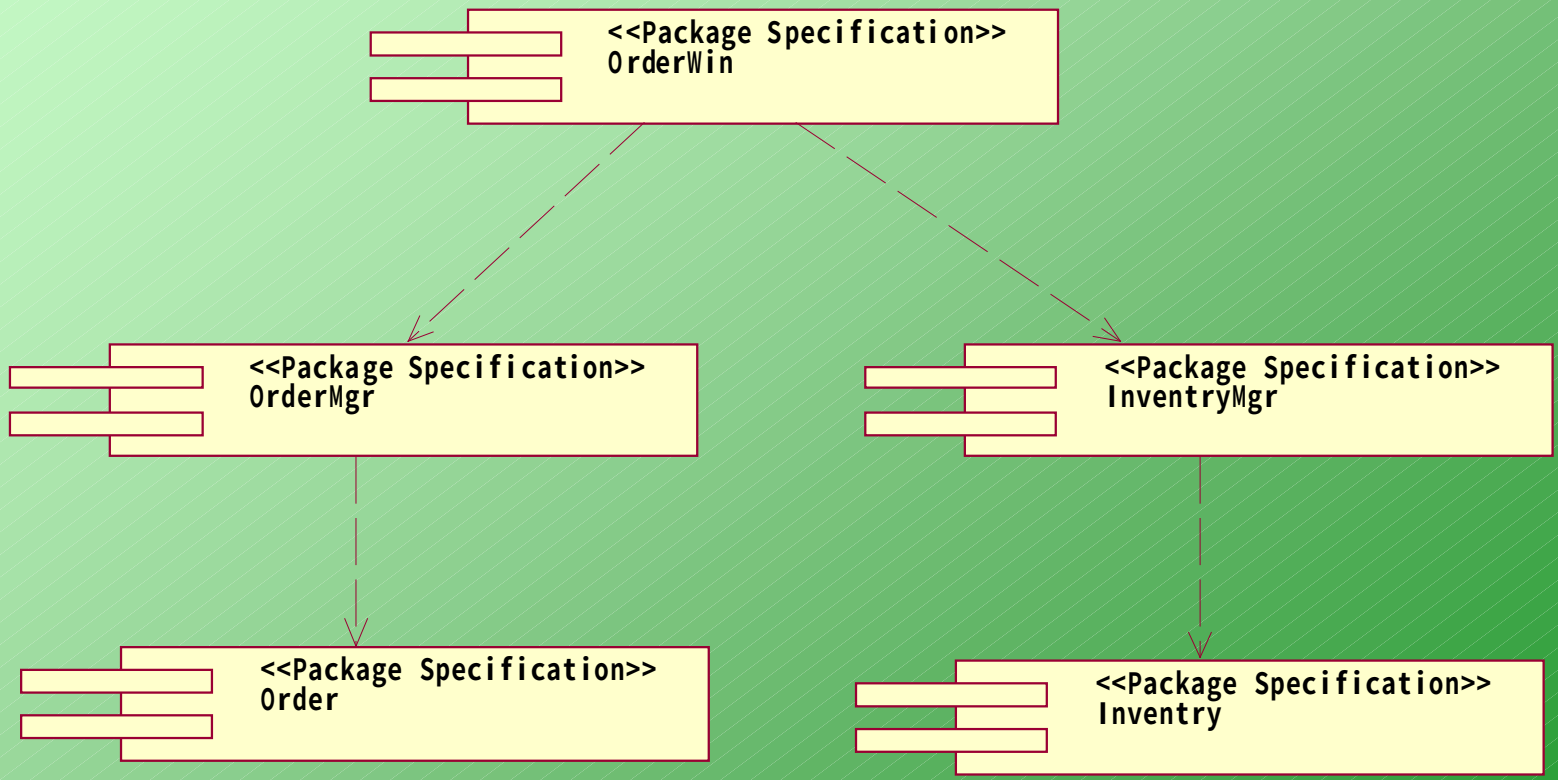
アクティビティ図



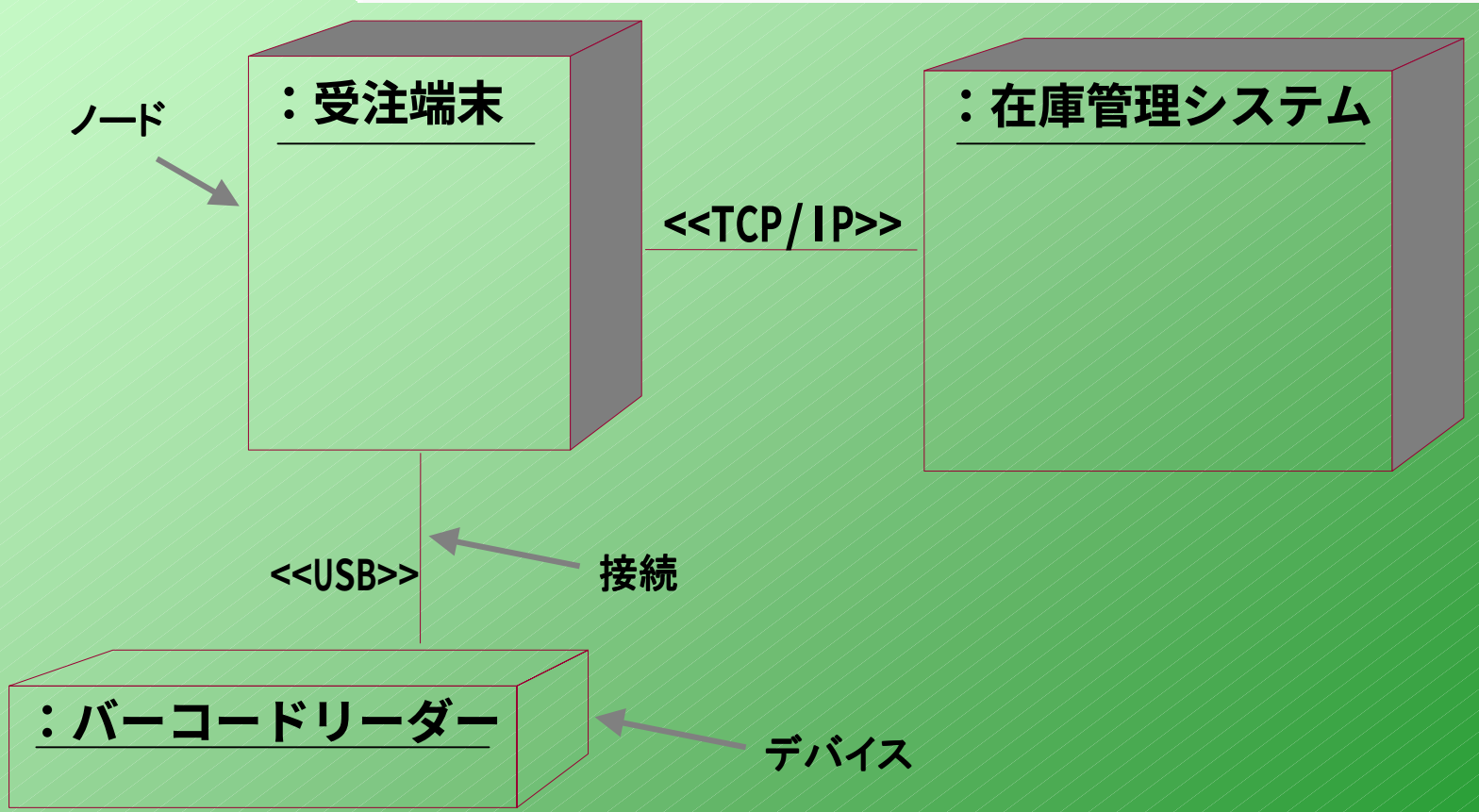
コンポーネント図



コンポーネント図



配置図



拡張メカニズム

- ステレオタイプ
  - 分類を拡張
- タグ付き値
  - プロパティの拡張
- 制約
  - セマンティクスを拡張(条件を定義)