要求モデル

FU14 ソフトウェア工学概論

第2回

吉岡 廉太郎

前回の内容

- ソフトウェア開発入門
  - ソフトウェアで課題を解決する
  - 品質を確保しながら
  - 課題を解決する知恵

- 要求定義工程
  - 要求定義の難しさ
  - 要求の種類と性質

今日の内容

- 要求定義に用いる要求モデルを知る
  - 役割、必要性
- 要求モデルの代表的なパラダイムを学ぶ
  - 4個のパラダイムを紹介します
- 具体例を通して要求モデルの特徴を知る
  - 新しいモデルでも理解できる素地を身につける

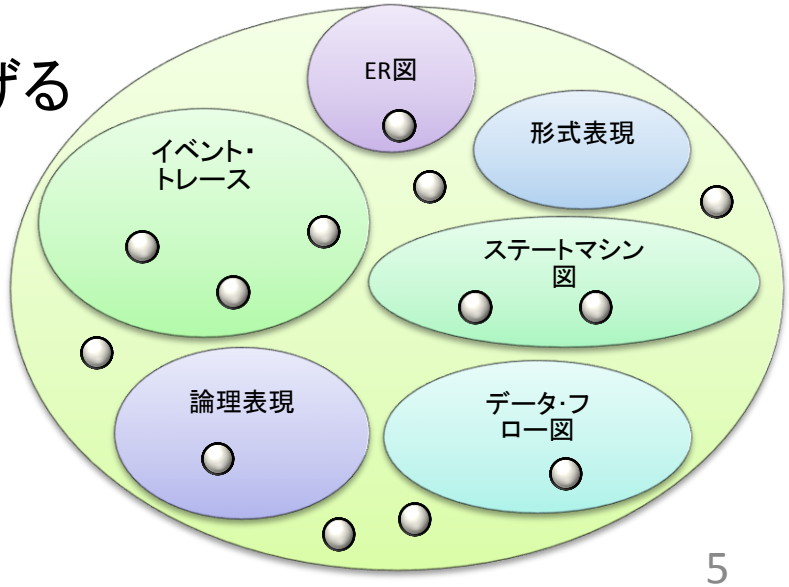
要求モデル(モデリング記法) 2.2.1

- 標準化されたモデルで要求を記述する
  - 確実な共通理解を実現する
  - モデリング、ドキュメント、意思の疎通、で役立つ

- モデル化の役割
  - 要求をまんべんなく理解する
  - モデルの過不足から、未定義や曖昧な動作に気付く
  - 出力・入力の重複や不一致から要求の矛盾に気付く
  - 顧客の要求とは全く違う方法で要求を定義し直す
    →その意味を注意深く確認する

要求モデル

- たくさんの記法や手法が存在する
  - 今後も多くのモデルが出現するだろう
  - 一つのモデルで対象の全てを記述するのは不可能
- 「パラダイム」で分類して理解する
  - 課題のコンセプト、振る舞い、性質を記述する
  - パラダイムごとに何に着目するかが異なる
- 代表的な4個のパラダイムを取り上げる
- 区別の仕方
  - どこに焦点を当てるのか
  - どの範囲で、どの細かさまで明らかになるのか



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

データ･フロー図

- 機能と機能間でのデータの流れをモデル化する
  - 活動･動作
  - データの流れ
  - データの保管場所
  - アクター（入力データを提供、出力データの受け手）を表す

- 目的
  - 動作や機能を明らかにする
  - 動作に関与する人や物を明らかにする

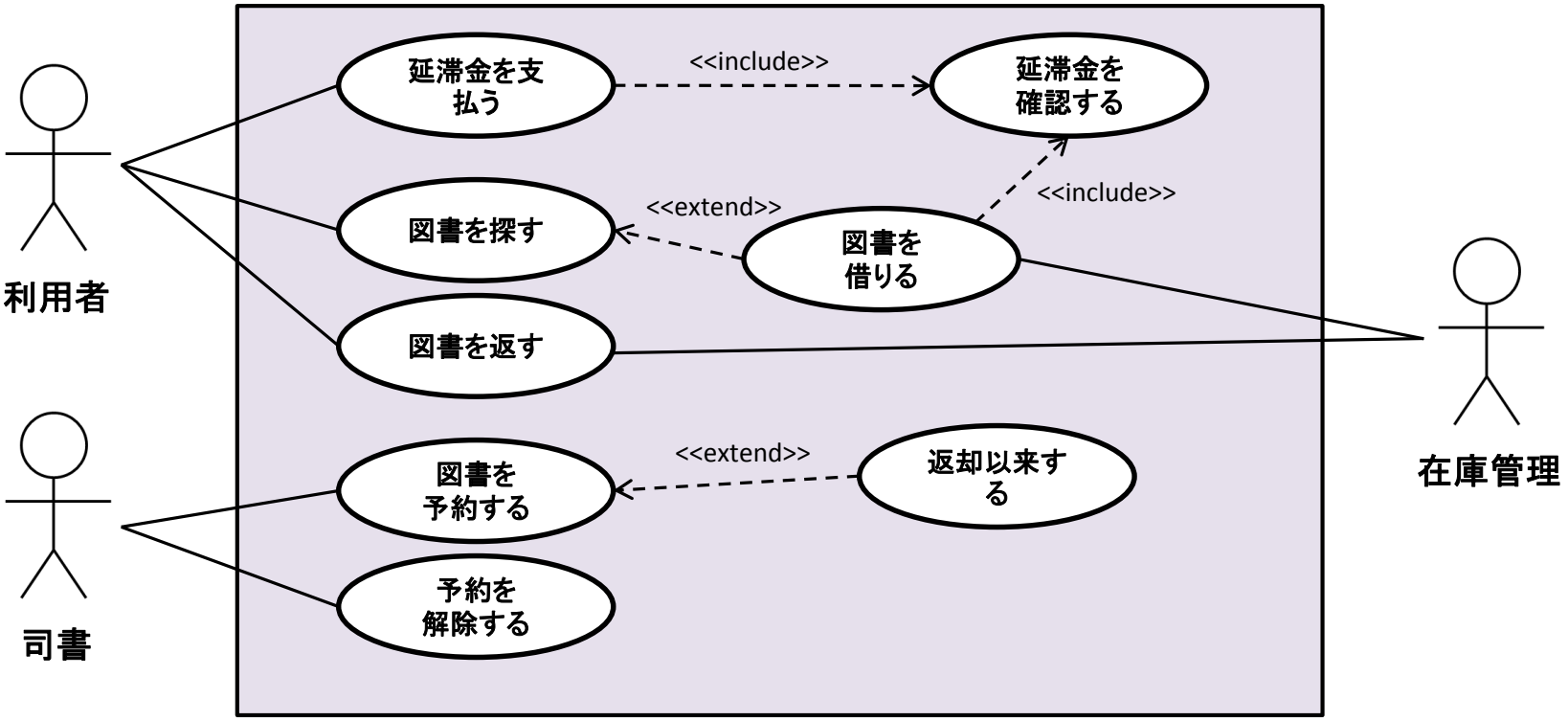
UMLユースケース図

- **U**nified **M**odeling **L**anguage（UML）
  - ソフトウェアの仕様や設計を記述するための記述言語

- 構成要素
  - 長方形：　システムの境界を表す
  - 人型：　アクター（人またはシステム）を表す
  - 楕円：　ユースケース（主な要求される機能）を表す
  - 線：　アクターのユースケースへの参加を表す

UMLユースケース図（2）

- 例：図書館のユースケース図
  - 対象範囲：貸出、返却、延滞金の支払い



データ･フロー図の特徴

- 長所
  - 直感的である
    （ER図、イベント･トレース、状態マシンは、比較的低レベルな振る舞いが対象）
  - システムの主な機能とデータの依存関係が分かる

- 短所
  - 課題を詳細に知らないソフトウェア技術者にとっては、イライラするほど曖昧

ユースケース記述

- ユースケースをより詳しく説明する
  - ユースケースの内容を説明する文章
  - 記載する内容を特に決まっていないが、一般的なものを参考にする

- ユースケース記述に含める内容

ユースケース名	ユースケースの名称
アクター	このユースケースに参加するアクター
概要	このユースケースが何をするものかの概要
事前条件	ユースケースが行われる前に成り立っている条件
事後条件	ユースケースが行われた後に成り立っている条件
基本系列	ユースケースが正常に行われる際の基本的な手順（アクターとシステムのインタラクション）
代替系列	基本フローとは別の重要な手順
備考	補足事項

ユースケース記述：例

ユースケース名	商品販売
アクター	顧客、事業者、金融機関
目的	指定した商品を購入し、代金の決済処理を行う
事前条件	なし
事後条件	在庫の無くなった商品には売り切れ表示がなされる
基本系列	1. 顧客が商品を選択してカートに入れる 2. 顧客がレジに進み必要な情報を入力する 3. 顧客が決済を指示し、システムが金融機関に決済処理を依頼する 4. 事業者が商品の発送処理を行う
代替系列	（特典処理） … （決済失敗） …
備考	一度に購入できる商品の総額は〇〇円以内

2.2.2

ER図

Entity Relationの頭文字

- 概念モデルを記述する図
  - 対象の構造を表す
- 3つの要素で構成される
  - エンティティー
    - 共通の性質や動作を持つ実世界のオブジェクトの集まり
  - 関係
    - エンティティー同士の関係と関係の種類を表す
    - 関係の種類を表す
  - 属性
    - エンティティーに付加されるデータや性質を表す

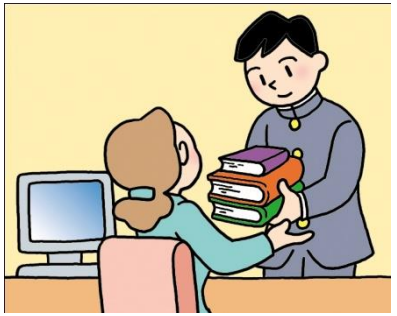
2.2.2

UMLクラス図

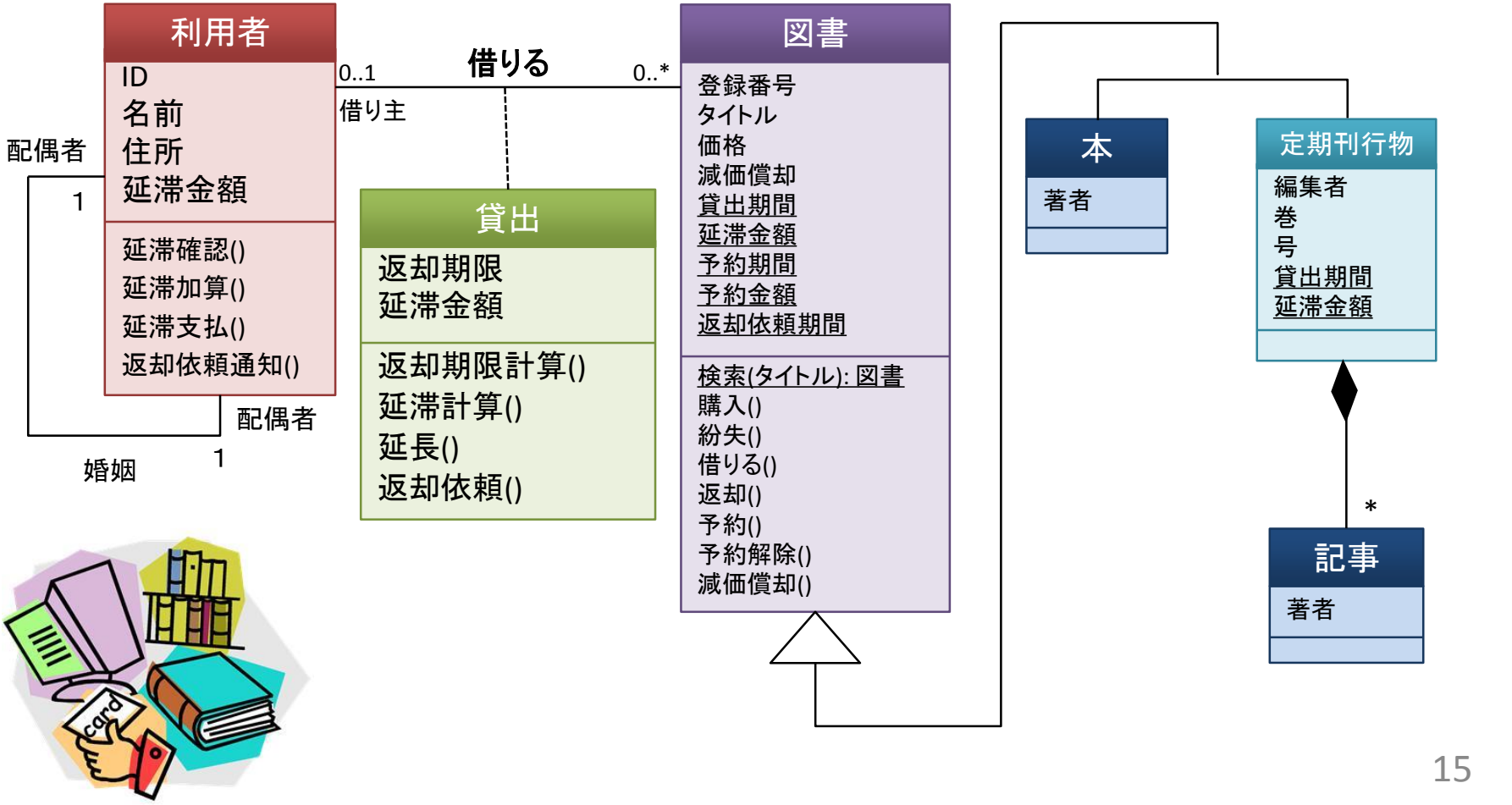
- クラス図では2つの要素でシステムを記述
  - クラス：オブジェクトを継承関係で整理したもの
  - メソッド：オブジェクトの変数に対する操作

- クラス図はUMLの中心的モデル
  - 仕様に含まれるクラスの関係を詳細に記述する
  - 一般に、クラス図は段階的に詳細化する

UMLクラス図（2）

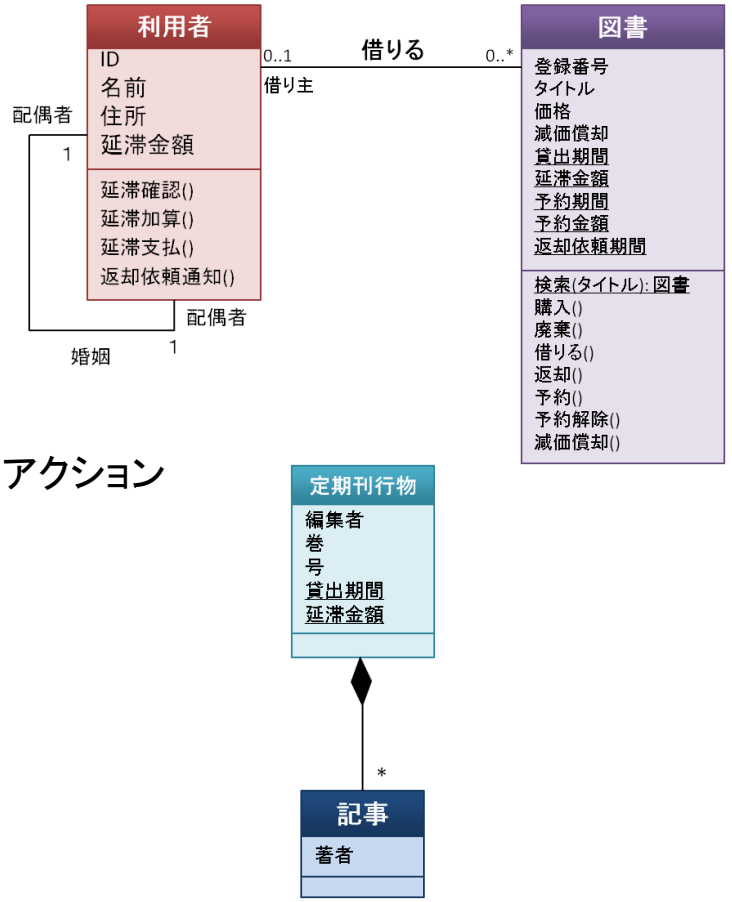


- 図書館のUMLクラス図



UMLクラス図（3）

- 属性とアクションはクラスと関係づけられる
  - 注意：クラスのインスタンスではなく
- クラス・スコープの属性
  - 属性に下線をつけることで示す
  - すべてのクラスで共有される属性
- クラス・スコープのアクション
  - アクションに下線をつけることで示す
  - 抽象クラス（クラスのインスタンスではなく）で実施されるアクション
- アソシエーション
  - クラス間の線として示す
  - クラス間の関係を表す
- 集約（aggregation）
  - 白抜きの菱形
  - 関連の一種で、持っていることを示す
- コンポジション（composition）
  - 黒塗りの菱形
  - 集約の一形態で、構成要素であることを示す



★詳しくは教科書を参照：2.2.2

ER図の特徴

- 長所
  - 課題の概要を理解しやすい
  - 要求が多少変化しても、見た目は大きく変化しない
- 短所
  - 適切なレベルでモデル化（抽象化）するのが難しい
  - エンティティーと属性の区別が難しい
- 要求工程の初期段階で利用されることが多い

イベント・トレース

- 一連のイベントのやりとりを図式的に表す
  - 各エンティティーの時間軸
  - 2つの エンティティーで取り交わされるイベント
- 全体では、たくさんの動作が起こる
  - 一つの図は、特定の対象範囲のイベントを追う（トレース）ことでシステムの動作を説明する

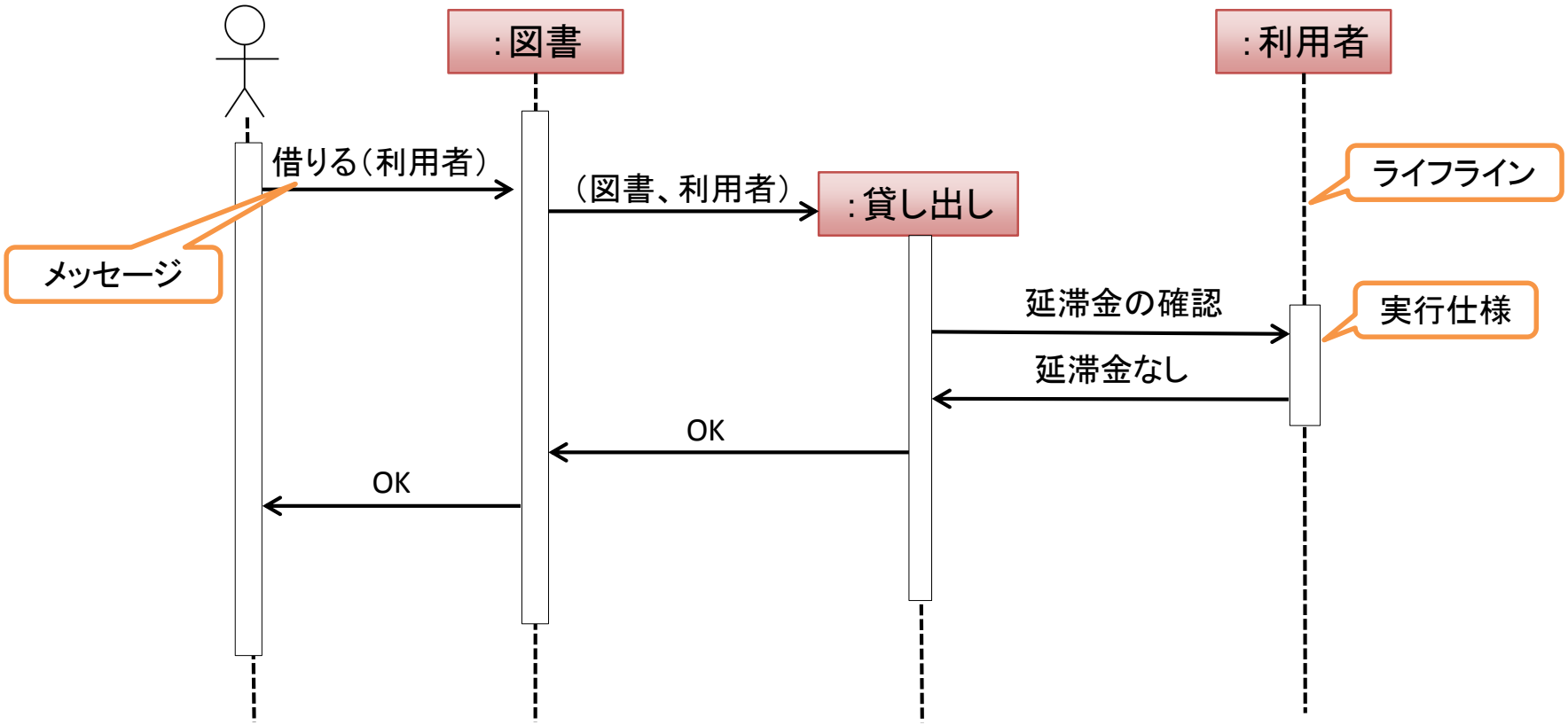
2.2.4

UMLシーケンス図

- システムの構成要素間またはそれらとシステム外部の要素との間で取り交わされるメッセージを例示する
  - ライフライン
    - やりとりに参加するオブジェクトとその時間軸
  - メッセージ
    - 送信側から受信側に向かうやりとり
  - 実行仕様
    - オブジェクトの時間軸上のアクションを表す

UMLシーケンス図

- 例：図書館のシーケンス図



ステートマシン図

- システムと環境のやりとりを記述した図
  - イベント間の定常的な状態
  - イベント発生による振る舞いや条件の変化（遷移）

- パス
  - 初期状態から、遷移によってたどる一連の状態
  - その環境の中での観測可能なイベントを表す

- 決定論的状態マシン
  - 全ての状態とイベントに唯一つの応答がある場合

2.2.5

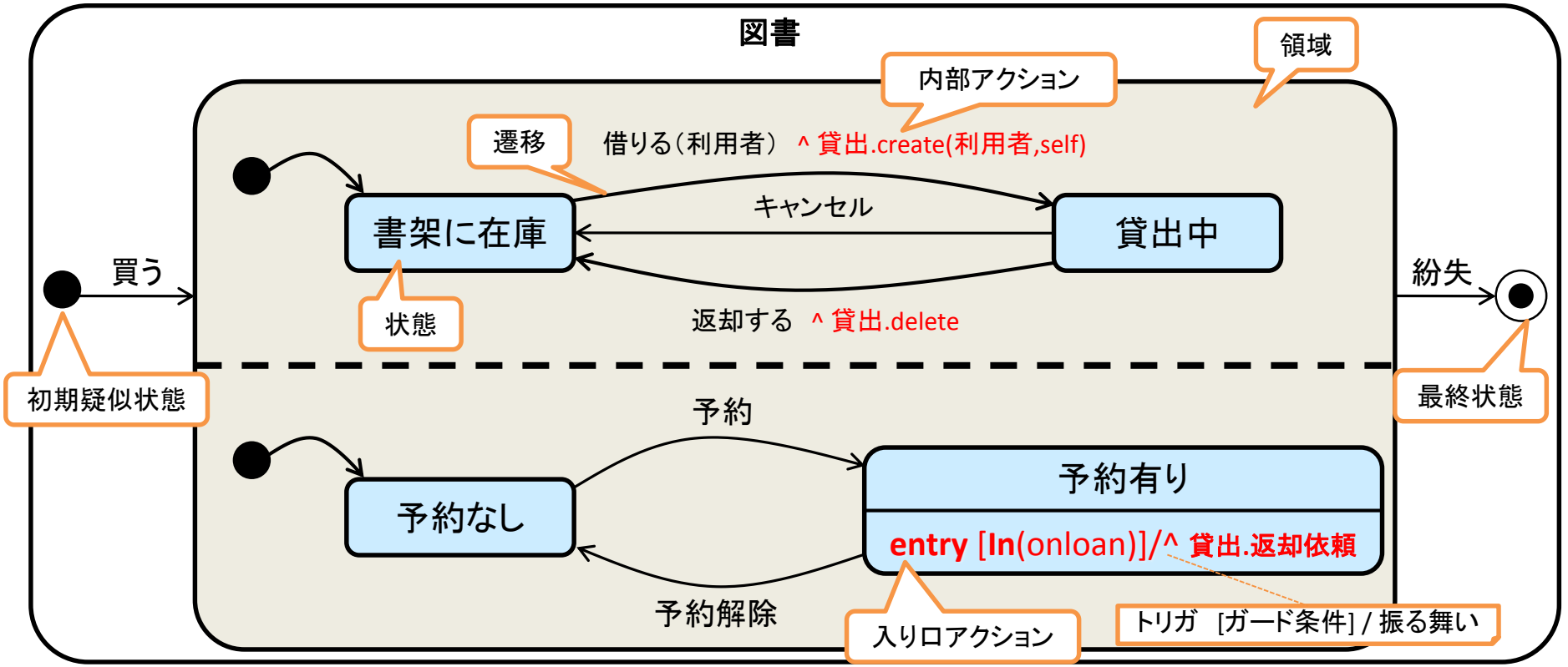
UMLステートマシン図

- UMLクラス図のオブジェクトについて、その動的な振る舞いを表す図
  - UMLクラス図には、エンティティーの振る舞いや、振る舞いの変化についての情報がない

- 全てのオブジェクトを考えると、多数の状態図が同時に実行されている

UMLステートマシン図（2）

- 例：図書館のUMLステートマシン図
  - 図書クラス



ステートマシン図の特徴

- 利点
  - 動的な振る舞いを記述できる
  - 一連のイベントに対する振る舞いの変化を記述できる
  - システムが認識しなければならない一連の入力と出力イベントを定義できる

ステートマシン図の特徴

- 状態の選択が重要
  - オブジェクトの振る舞いのうち適切な状態を抜き出す

- 状態の選択方法
  - 一連のイベントにより繰り返し出現する状態
  - 一定の時間間隔で出現する状態
  - オブジェクトが処理を行う場面や入力待ちの状態
  - オブジェクトの振る舞いを左右する状態

本日のまとめ

- 要求定義工程で用いる要求モデル
  - 要求を正確・確実に理解する
  - 要求を完全に導き出す
- 代表的なパラダイムを4個紹介した
  - 各パラダイムが課題のどの性質に焦点を当てるか
  - 各要求モデルの違い・特徴を知る
  - 場面と状況に応じて適切な要求モデルを選ぶ
- 次週は、要求定義の最後
  - 仕様記述言語、検証方法、等

次回講義の事前学習：
4.1.7, 4.6.1, 4.7.1, 4.7.2, 7.1.1, 7.1.2