オブジェクト指向開発論

2020年6月4日

海谷 治彦

目次

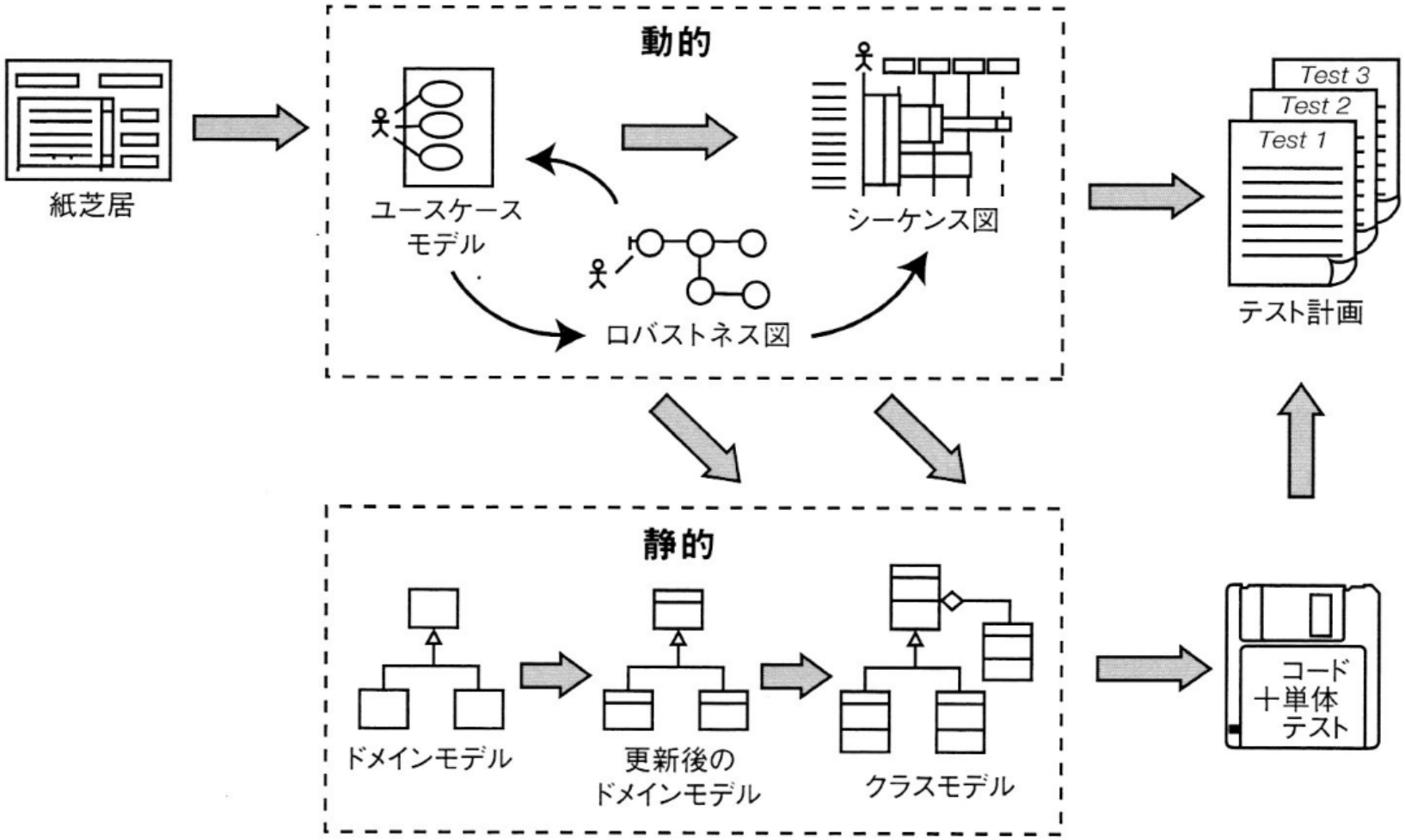
- 前回の名詞抽出法の復習します (前回スライド)

- ユースケースモデル
- ユースケース記述

- ユースケースのレビュー

- サンプル等は Teams oo04sample.zip より

ICONIXの全体手順



ユースケース

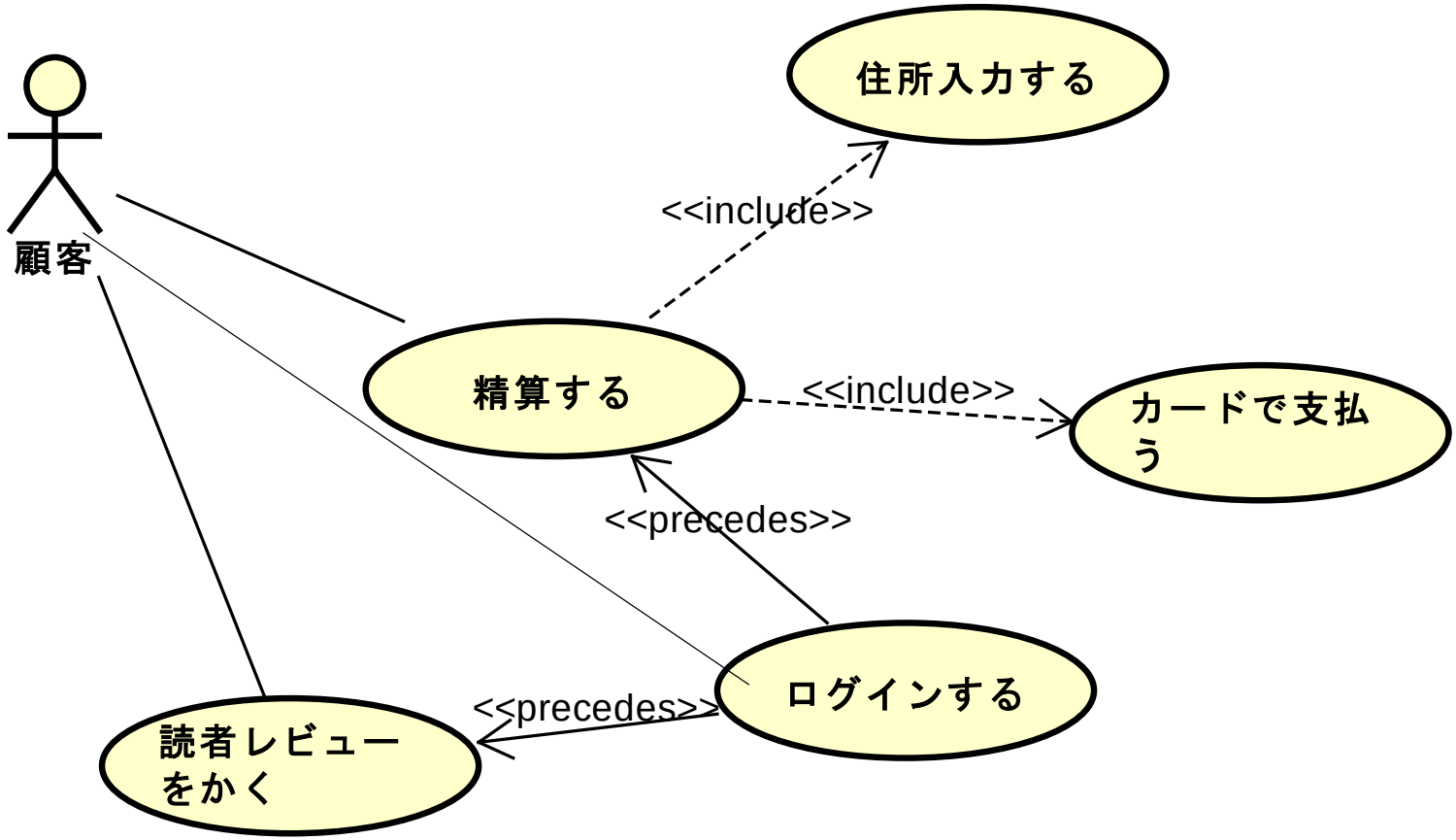
- 端的にいって，システムの機能を示す．
- その機能を遂行するために，システムとシステム外部の人や別システム(アクターと呼ぶ)が相互作用する様として仕様化する．

- ユースケースモデル
  - システム全体がどんな機能群を持ち，それぞれの機能遂行にどんなアクターがかかわるかを記述したもの．
- ユースケース記述
  - 個々のユースケースを遂行するにあたり，システムやアクターが順番に何をするかを記述したもの．

ユースケースモデル

- システムの機能を楕円でかき，
  - この機能ひとつひとつを「ユースケース」と呼ぶ
- その機能遂行に関連するアクターと線でつなぐ簡易な図.
- ユースケース間の関係はわりと沢山あるけど，本講義では，以下の二種類の利用のみ推奨する
  - includes ユースケースAがBを呼び出すこと.
    - サブルーチンや関数コールとほぼ同じ.
  - precedes AがBより先に起こること.
- ユースケースモデルではシステム内部のデータに相当するものは書いてはいけない.

ユースケースモデルの例



標準的な関連について

- invokes
  - includesにほぼ同じ.
  - 本来，こっちを使いたいが，ツールの都合，includesを使う.
- extends
  - 本授業では利用しないことを推奨
  - Javaの extends と意味が異なる.
  - Javaでの抽象クラスをexntendsするのにノリが似ていて，拡張される側が事前に拡張されることを意識した作りになっていなければならない.
- generalize
  - 本授業では利用しないことを推奨.
  - こっちは，所謂サブクラス，スーパークラスの関係.
  - どちらかといえば，overrideの側面が強調されている.

その他，例題

- サンプルはteamsからダウンロードすること

- dotcampus等のLMS LMS.asta
  - 意外に複数のアクターがかかわる機能が無い.
- Uber easts ube.asta
  - 実は使ったことない，シンプルになった.

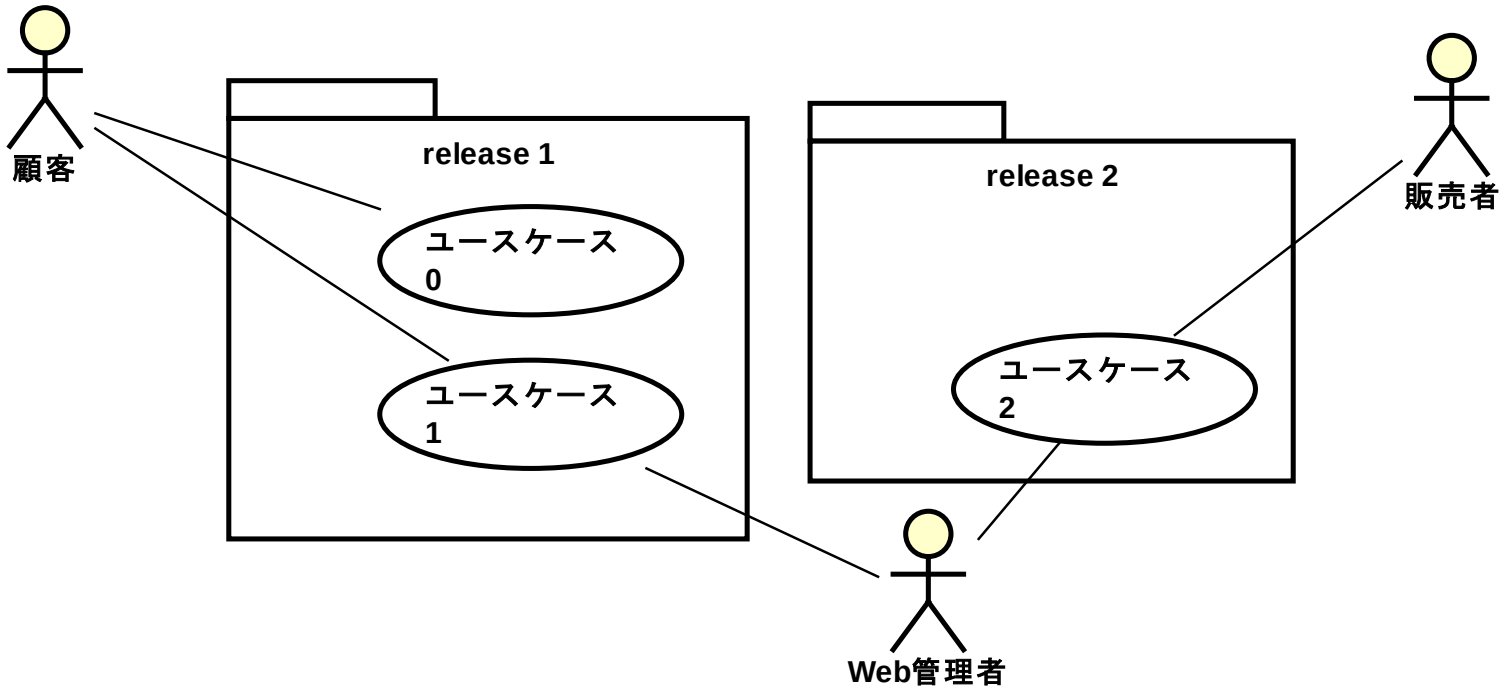
- iTrust 電子カルテシステム
  - とある大学で演習用に作られた，巨大.

ユースケースのパッケージ化

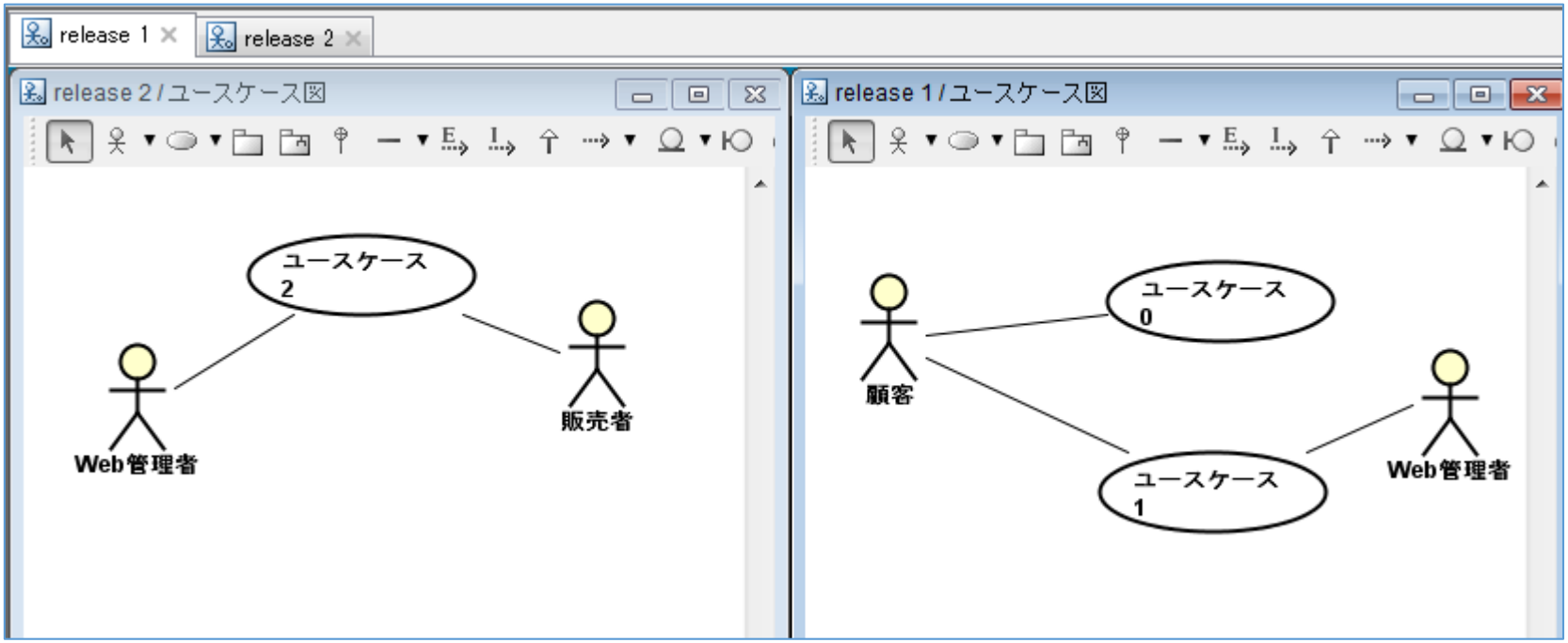
- 大きなシステムの場合，ユースケース(機能)が100を超える場合がある.
- その場合は，ユースケース図を分けてかくか，
- ユースケースをパッケージで分けるかする.
- 分ける指針としては，
  - 機能的に関連した領域
  - リリース

  がある.
- 一気に全てのシステムを作るのではなく，最初は一部機能から作るならリリース毎にパッケージ化するのが良い.

パッケージの例



図を分ける


release 1
release 2
release 2 / ユースケース図
ユースケース
2
Web管理者
販売者
release 1 / ユースケース図
ユースケース
0
顧客
ユースケース
1
Web管理者

ユースケース記述について

- 各ユースケース(楕円で示した機能)が，システム外のアクターと，どんなステップをふんで機能を遂行するかの手順を書く.
- システム内の手順は書かない，書くのはシステム外部とのやり取りのみ.
- 基本，システムもしくはアクターが主語となる文の列となる.
- 文中の目的語等はドメインモデルもしくは境界クラス(後述)から選ぶ. 無ければ，ドメインモデルを更新する.
- 通常の手順に加え，代替の手順，例外の手順も書く.

ユースケース記述のフォーム構成

- 概要
  - 当該機能の概要を一～二行くらいで要約して書く.
- 事前条件
  - この機能を実行する際の前提条件.
- 事後条件
  - この機能が実行された後に成り立つ条件.
  - 要は機能の宣言的な意味.
- 基本系列
  - 機能を遂行する正攻法の手順.
  - 要は機能の手続き的な意味.
- 代替系列 (略可能)
  - 別ルートの手順. もしあれば.
- 例外系列
  - 機能遂行が失敗に終わる場合の手順.
  - システムやアクターが回復不能な状態に陥らないような手順が望ましい.

例

項目	内容
ユースケース	精算する
概要	購入代金の精算を行う.
アクター	顧客
事前条件	顧客はログイン状態にある
事後条件	支払いが完了している
基本系列	顧客はシステムに住所を入力する. システムは入力された住所を表示し確認ボタンとやり直しボタンを提示する. 顧客は表示内容が正しい場合，確認ボタンを押す. システムは支払い方法の選択画面を表示する. 顧客は支払い法を選択する. システムはカード決済が選択された場合，カード番号を要求する. 顧客はカード番号を入力する. システムは決済確認のためのボタンを表示する. 顧客はボタンを押し決済を承認する. システムは承認を受け付けたことを表示する.
代替系列	
例外系列	システムは入力されたカード番号がvalidなものでなければ，その旨を表示し精算を中断する.

記述のポイント

- 文の列として書く.
- 能動態の文で書く，受動態や支持的な文はダメ.
  - 「<アクター>が<何か>をする」の形式がよい.
- 各文の主語はアクターもしくはシステムでなければならない.
- 主語は省略してはならない.

基本，代替，例外と分ける意味

- プログラムのような形式記述なら，これらを分けて書く必要は無いかもしれない．
- しかし，原則，文のリストとしてかくため，制御構造が複雑にならないためにも，分けて書くほうがよいとされている．
- 加えて，本来，意図する動作である「基本」，バックアップとしての「代替」，障害処理的な「例外」は，それぞれ要求分析的に意味合いが違うため，文法的に合成して書けたとしても，合成すべきではない．

画面イメージ

- ユースケースでは，システムとアクターの対話を書く.
- よって，対話の接点となるユーザーインタフェース(UI)の画面イメージをメモ書きしてもよい.

境界クラスについて

- Boundary Class
- システムがアクターとのインタフェースを通して何か行う場合，
  - 「システムはWebページに注文内容を表示する」

と書く場合がある.

- このような場合，インタフェースがより具体的に決まっているならば，その具体的な名称を書いたほうが良い，それが境界クラス，たとえば，
  - 「システムは注文確定ページに注文内容を表示する」

ユースケース vs アルゴリズム

ユースケース	アルゴリズム
ユーザーとシステムの対話	不可分な計算
イベントと応答のシーケンス	ステップの列
基本/代替コース	ユースケースのーステップ
複数のオブジェクトがかかわる	クラス内部での処理
ユーザーとシステム	システムのみ

ユースケース記述を半形式的に書く

ユースケース記述を書きましたが

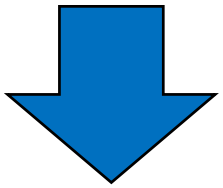
融資する / ユースケース記述

項目	内容
ユースケース	融資する
概要	銀行が融資をする支援をする機能
アクター	顧客 融資担当者 保証人
事前条件	顧客は取引実績がある. 保証人の候補は事前登
事後条件	融資が行われる

基本系列	1 顧客は融資額と保証人候補を入力する. 2 システムは金額確認を融資担当者に送る. 3 システムは保証人候補に許諾確認の連絡を行う. 4 融資担当者は金額を承認する. 5 保証人候補は保証人になることを受託する. 6 システムは顧客に融資をする旨を連絡する. 7 システムは融資担当者に融資が成立したことを連絡する.
代替系列	融資担当者が金額を承認しない場合: システムは顧客に金額再検討を依頼する. 顧客は別の金額を入れ，基本系列3にいく. 保証人候補が受託しなかった場合: システムは顧客に別の保証人候補を入れるように依頼する. 顧客は別の保証人を入れる. 基本系列5から継続
例外系列	融資担当者が金額を承認しない場合，: システムは金額の再検討を顧客に依頼する. 顧客は融資依頼をやめる旨をシステムに入力する. 保証人候補が受託したなかった場合: システムは顧客に別の保証人を入れるように依頼する. 顧客は融資を断念する.

問題点

- 文系の人には文章なので分かり易いかもだけど，理系にはまどろっこしい．
- 基本的にはアクター群とシステムとのやり取りなのに，そのキャッチボールがわかりにくい．
- 代替系列や例外系列は基本系列の派生なのに，どこで派生するかぱっと見わかりにくい．
- 文章なので，機械的処理には非常に不向き．

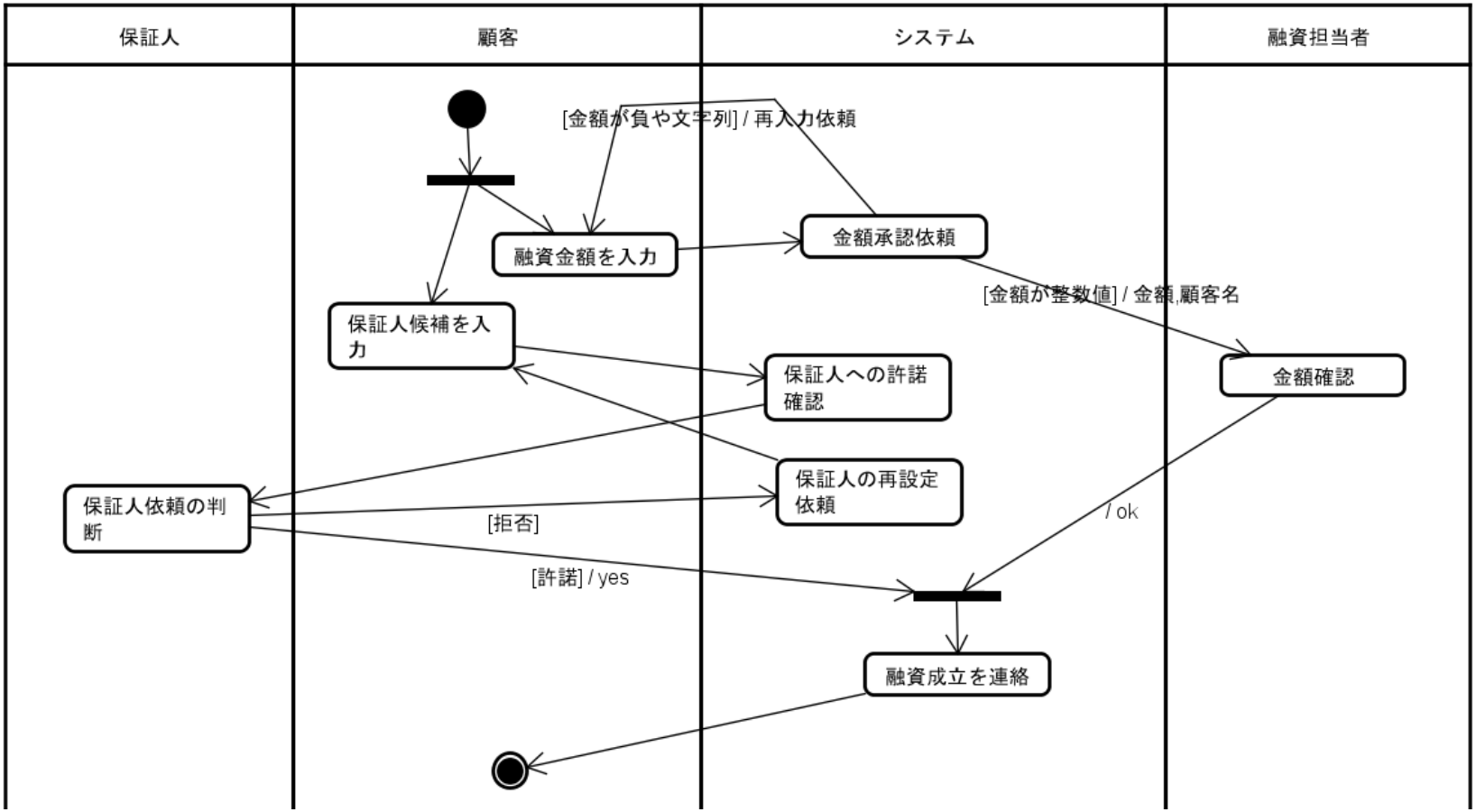


- 図式で書いたほうがいいんじゃない？

アクティビティ図

- UMLの振る舞い的な側面を書く図
  - 他にシーケンス図，ステートマシン図がある．
- 一応，インスタンスの図ではなくクラスレベルの図である．
  - 特定の事例を記述したものではない．
  - 一方，シーケンス図は基本，特定のシーケンス例を書いたインスタンスの図．
- 雰囲気はフローチャートに似ている．
- 理論的な部分としてペトリネットの考えを利用している．
  - 並列に処理してよい流れの見える化
  - 状態遷移図の雰囲気も入っている．
- アクター等の切り分けを明確にできる．
- 所謂，組織のワークフローを書くのに便利．

例 銀行の融資



以降，基本的な文法の紹介.

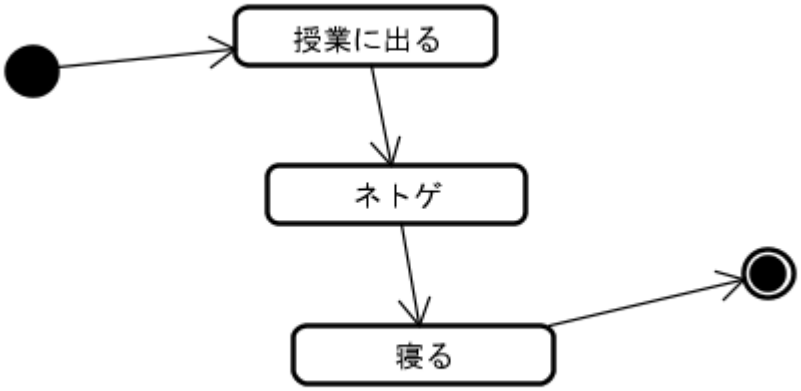
パーティション

- 前頁の外枠それぞれに相当.
- 枠がアクターもしくはシステムに相当する.
- 枠内にあるアクションを枠に対応するアクター等が遂行する.

- 誰が何をやることで，あるユースケースが実施されていくかが俯瞰しやすい.

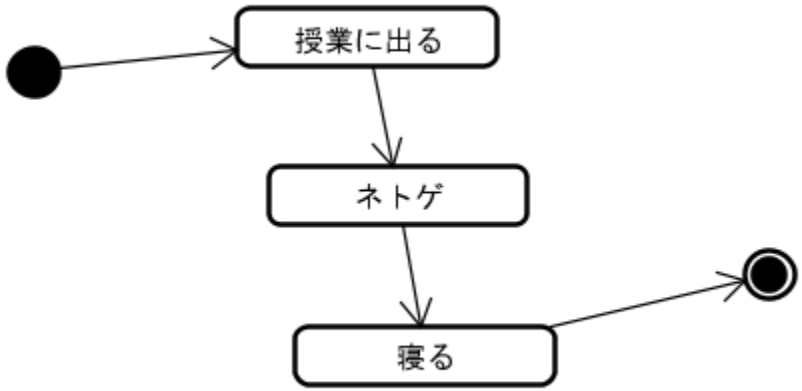
開始と終了ノード

- 一連の処理の開始と終了を示す.
- 終了は複数あってもよい.
- アイコンは状態遷移図やペトリネットに由来する.



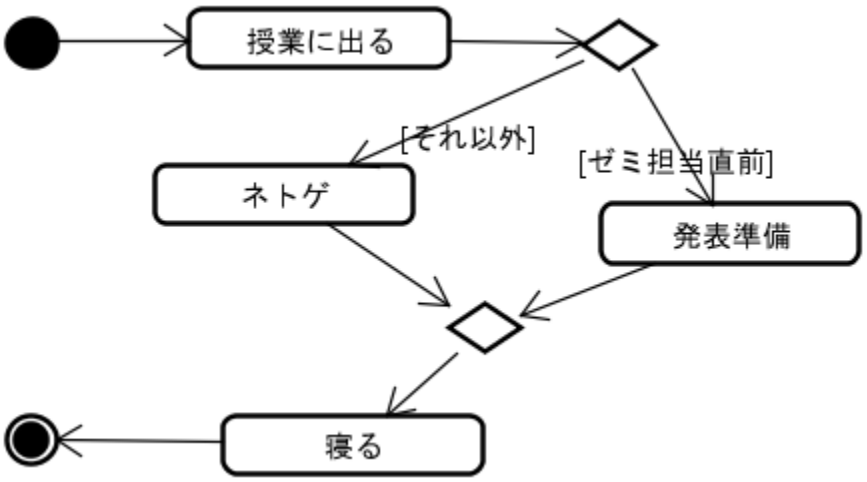
アクションとフロー

- システムもしくはアクターが行うことを示す.
- 「・・・する」に相当.
- フローはアクションの順序を示す.



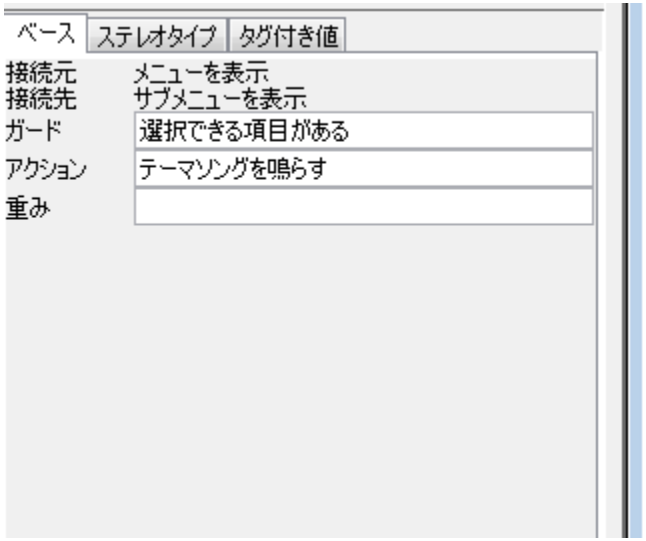
ディシションノード マージノード

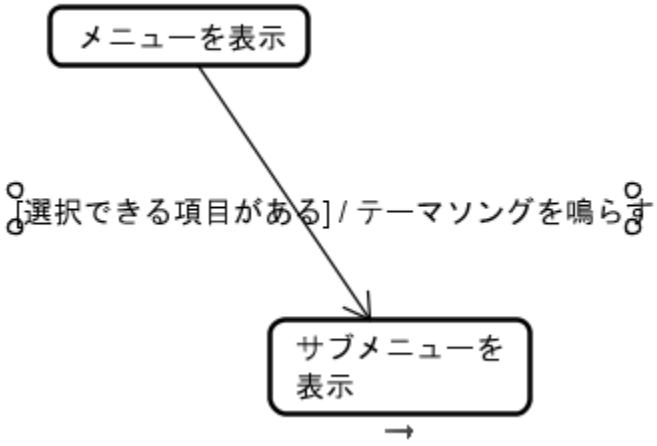
- 条件分岐みたいなのを書ける.
- 分岐は◇で分かれて，◇で合流しないといけない.



ガードとアクション

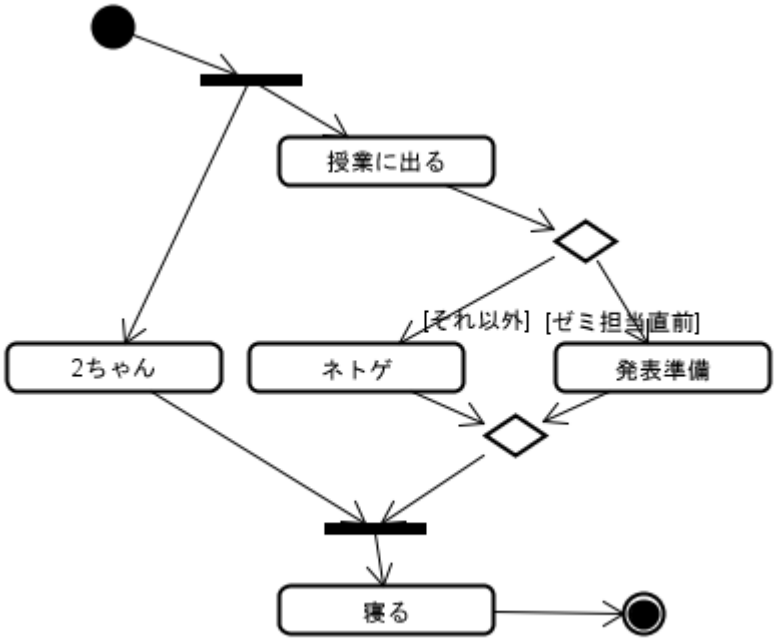
- アクションノード間の遷移において以下の二属性を書ける.
  - ガード その遷移ができる条件
  - アクション 遷移中に起こす行動
- これらの表記は状態遷移図から由来する.





forkとjoin

- 並列して行える処理を書くことができる.
- joinで並列から直列に戻ることが書ける.



書き方のポイント

- あるユースケース(機能)を遂行するのに必要な活動のフローを書く.
- それぞれのアクションを誰がやるのかを識別する.
- パーティションにアクターもしくはシステムを割り当て，それぞれが行うアクションをマッピングする.

問題点

- 複数通りの書き方ができる.
  - プログラム言語でも同様か.
  - 社内，組織内でルールを決める.
- 基本，代替，例外を全部書いたらカオスになって読めない.
  - 色で区別.
  - 予め分けてかく.
  - ツールで分割(というか投射)して読む.
- 制御フローとデータフローがごっちゃになる.
  - 今回は意図的にデータフロー部分は解説しませんでした.
- 文系人が嫌がるかも(特に老人).
  - やつらに勉強させる.

ユースケースのレビュー

ユースケースの見直し (review)

- ドメインモデルを横目にユースケース図，ユースケース記述を書いた(はず).
- しかし，文章なので正しく書けているかは，読み直したほうが良い.
- まず，ここでは，ユースケースの見直しTIPSについて語る.
  - ユースケース図については，特にふれず.

レビューのガイドライン 1/2

1. 8割が一般利用者にも理解できるような言葉でドメインモデルに書いてあるかをチェックせよ.
2. ドメインモデルが is-a, has-a の関係で整理されているかをチェックせよ.
3. ユースケース記述が能動態(「Aが・・・をする」)で書いてあるかを確認せよ.
4. 要求仕様書にある「～しなければならない」的な文がユースケース記述に含まれないことを確認せよ.
5. ユースケースがパッケージ化されているか確認せよ. (リリース毎等)

レビューのガイドライン 2/2

6. ユースケース記述がドメインモデルの語句で記述されているか確認せよ.
7. ユースケース記述ではユーザーインタフェースの部品の名前を用いよ.
   - ボタン等(・・・ボタンと命名)の利用者が作用できるもの
   - ラベルやリスト等，利用者が見られるもの
8. ユースケース記述には画面のイメージ(GUI紙芝居)を付記せよ. 手書きOK
9. ユースケース，ドメインモデル，画面イメージを利用者等のステークホルダに確認してもらえ.
10. 後述のレビューのための8ステップに沿ってレビューせよ.

レビューのための8ステップ

1. スコープ外のものを取り除く.
2. 受動的，宣言的な文を能動的な文にする.
3. ユースケース記述が抽象的過ぎないかを確認する.
4. GUIを正確にイメージする.
5. 関係するドメインオブジェクト(ドメインモデルの要素)に名前をつける.
6. 全ての代替，例外コースがあるかを確認する.
7. ユースケースと要求仕様書が対応付くか確認する.
8. 利用者の望むことが，個々のユースケースに書いてあるかを確認する.

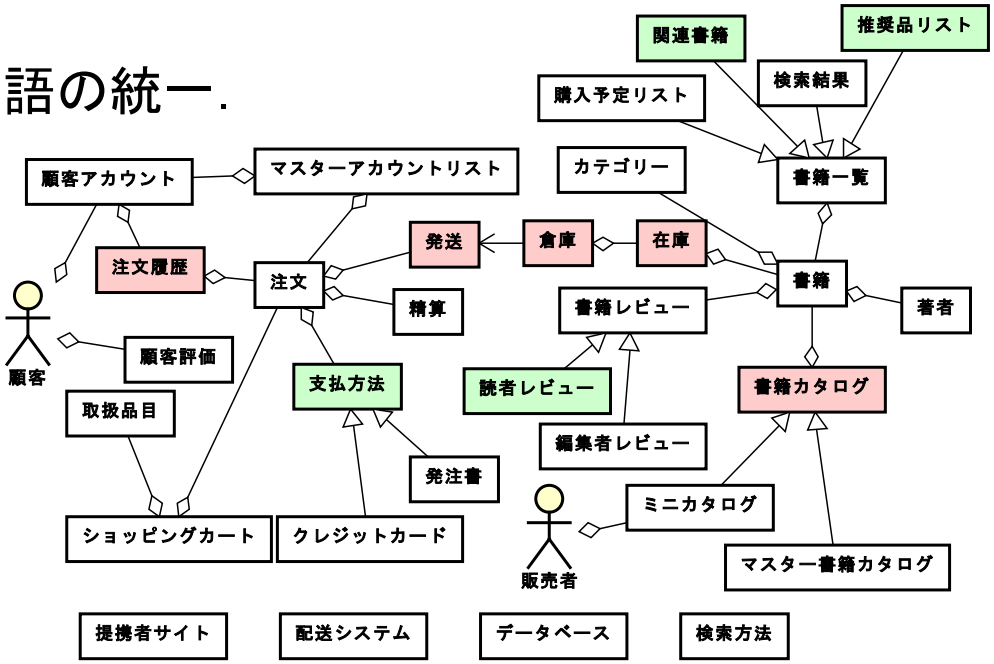
レビューの適用例

- 別紙 p101ucds.xlsx oo03dom.asta をteamsからダウンロードしながら話を聞いてください.
  - ユースケース「書籍詳細を表示する」を6回，改訂してます.
- 同時に第3回で配布した reqs.docx も見てください.

各バージョン改訂の概要

- Version 1 から 2
  - 役に立たない項目(事前，事後条件やレベル)を削除
- Version 2 から 3
  - ログインと注文はスコープ外なので削除
- Version 3 から 4
  - ドメインモデルを見ながら用語の統一.
- Version 4 から 5
  - 例外コースを追加
- Version 5 から 6
  - 抽象的な部分を具体化
- Version 6 から 7
  - さらなる具体化
  - 能動態の文に直す
  - 「容易に」等の性能的な話はカット
  - ドメインモデルで is-a でまとめてある部分はカット

oo03dom.asta より



本日は以上