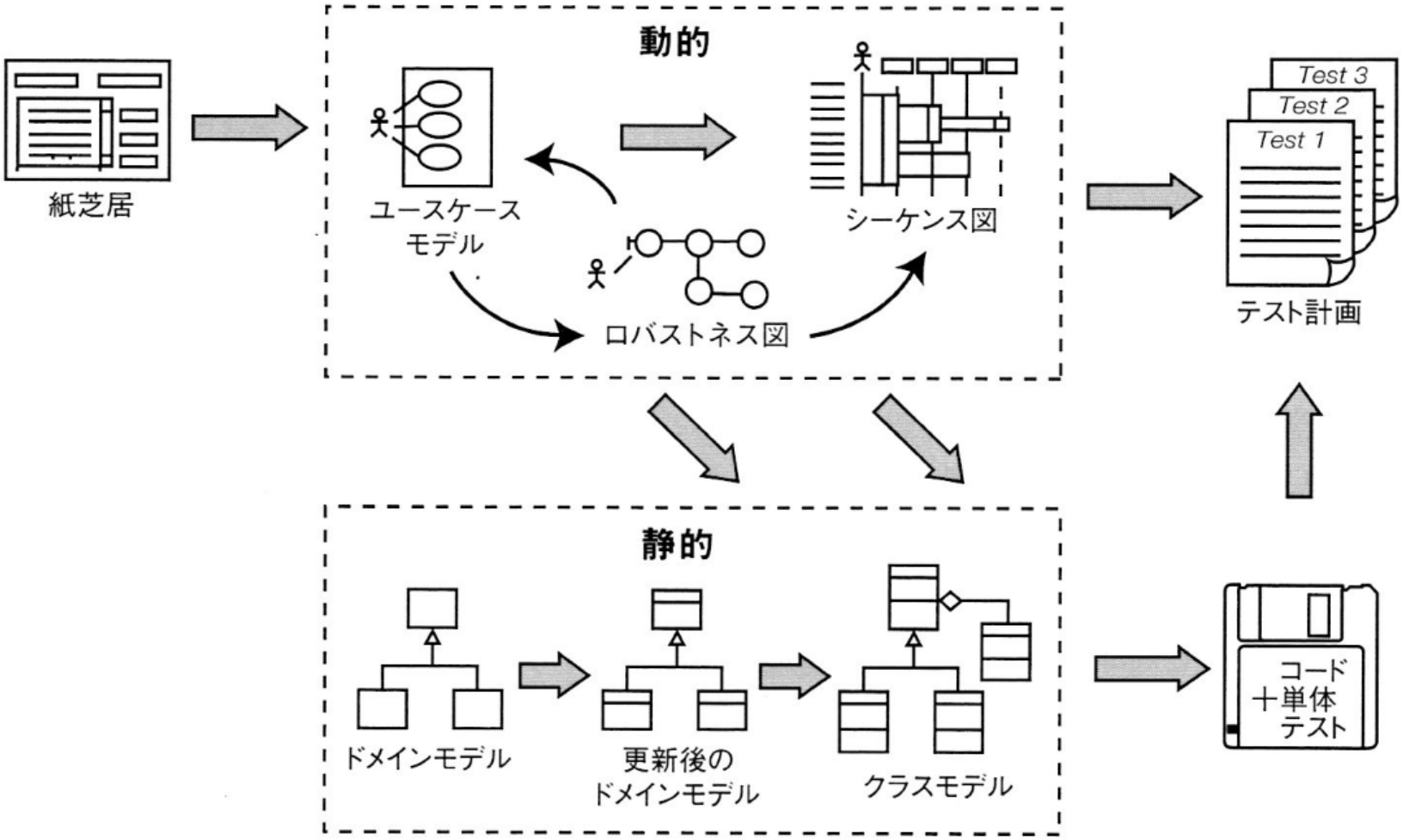
ソフトウェア工学

2022年10月24日

海谷 治彦

ICONIXの全体手順



復習
ユースケース記述

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 (略可能)
  - 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 (略可能)
  - この機能を実行する際の前提条件.
- 事後条件 (略可能)
  - 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

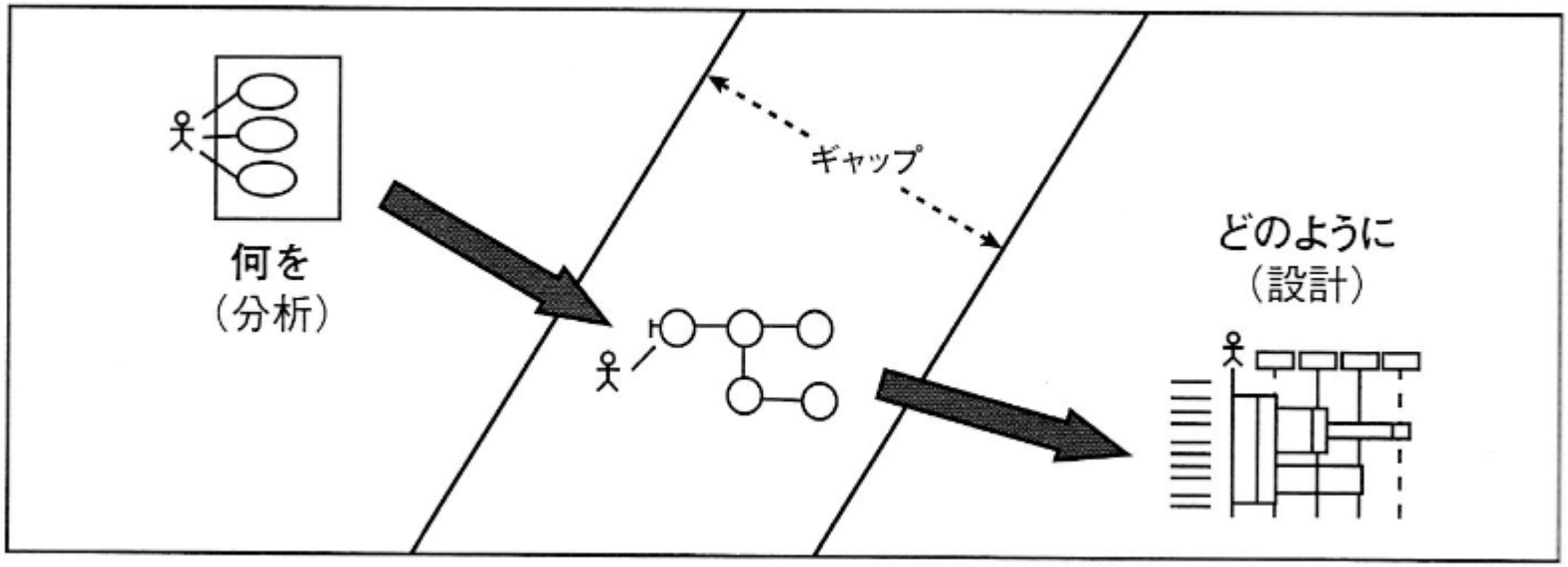
項目	内容
ユースケース	精算する
概要	購入代金の精算を行う.
アクター	顧客
事前条件	顧客はログイン状態にある
事後条件	支払いが完了している
基本系列	顧客はシステムに住所を入力する. システムは入力された住所を表示し確認ボタンとやり直しボタンを提示する. 顧客は表示内容が正しい場合，確認ボタンを押す. システムは支払い方法の選択画面を表示する. 顧客は支払い法を選択する. システムはカード決済が選択された場合，カード番号を要求する. 顧客はカード番号を入力する. システムは決済確認のためのボタンを表示する. 顧客はボタンを押し決済を承認する. システムは承認を受け付けたことを表示する.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

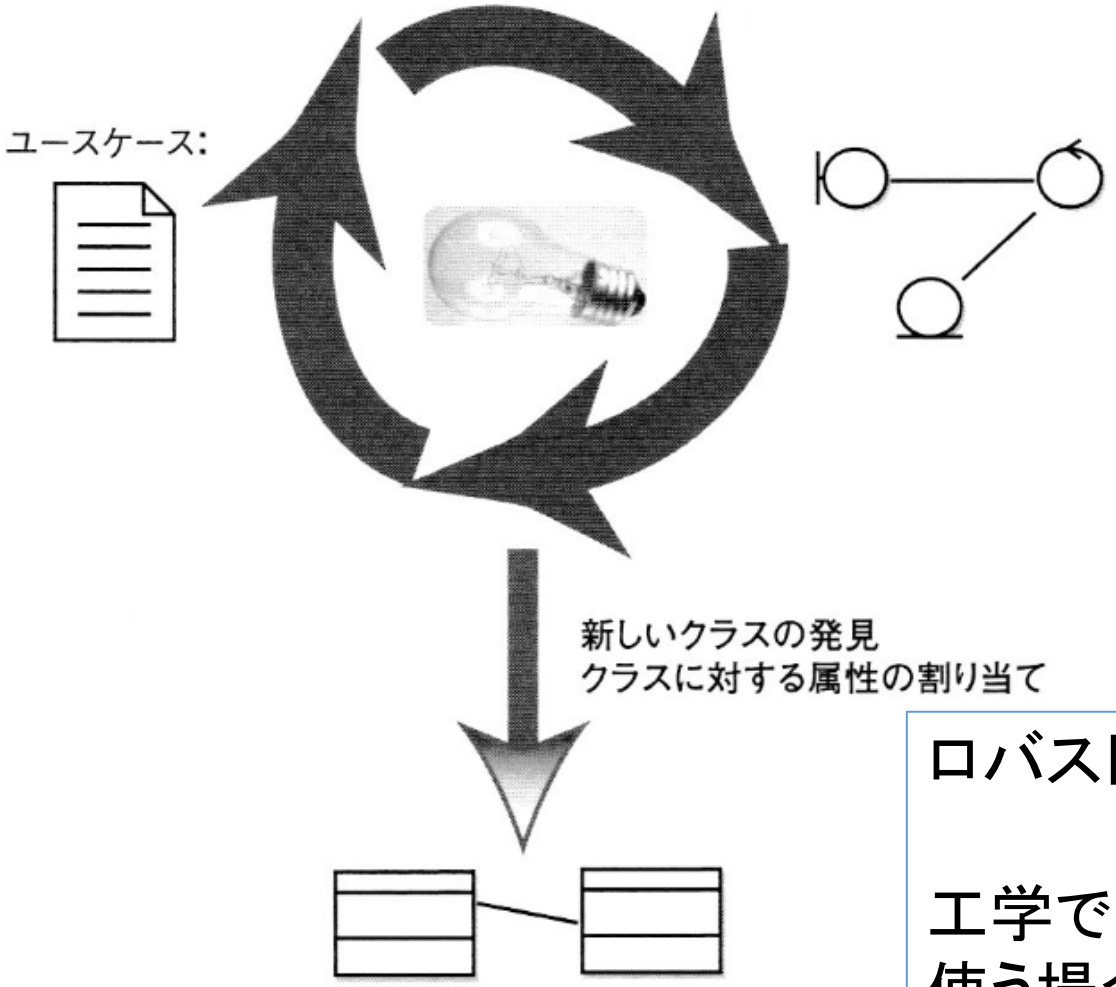
- 文の列として書く.
- 能動態の文で書く，受動態や支持的な文はダメ.
  - 「<アクター>が<何か>をする」の形式がよい.
- 各文の主語はアクターもしくはシステムでなければならない.
- 主語は省略してはならない.

ロバストネス分析

- 最終的にはクラス図，そしてコードを得なければならない．
- ユースケースを眺めていても，なかなかクラス図にはならない．
- 下記のようなギャップを埋めるために試験的な設計をするための図がロバストネス図．



チェック：
すべての代替コースをカバーしたか?
すべての操作／機能を発見したか?
すべてのデータの流れをエンティティ間に割り当てたか?

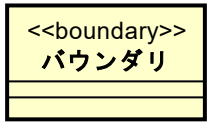


ロバスト = robust = 健全性

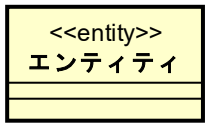
工学では「頑強性」の意味で使う場合が多いが.

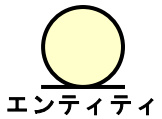
ロバストネス図の構成要素

- バウンダリ オブジェクト Boundary
  - アクターとのインタフェースに相当，名詞で表現
  - 画面，Webページ，通信回線
  - ボタンやメニュー等はダメっぽい．

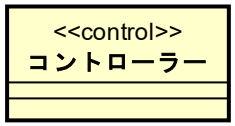


- エンティティ オブジェクト Entity
  - データや情報に相当，名詞で表現
  - ドメインモデルの要素

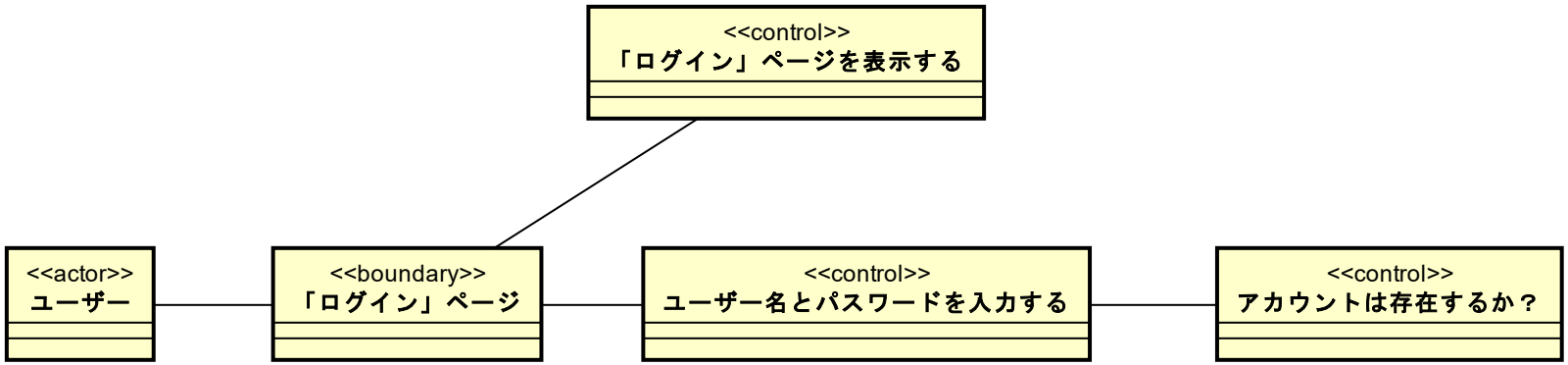




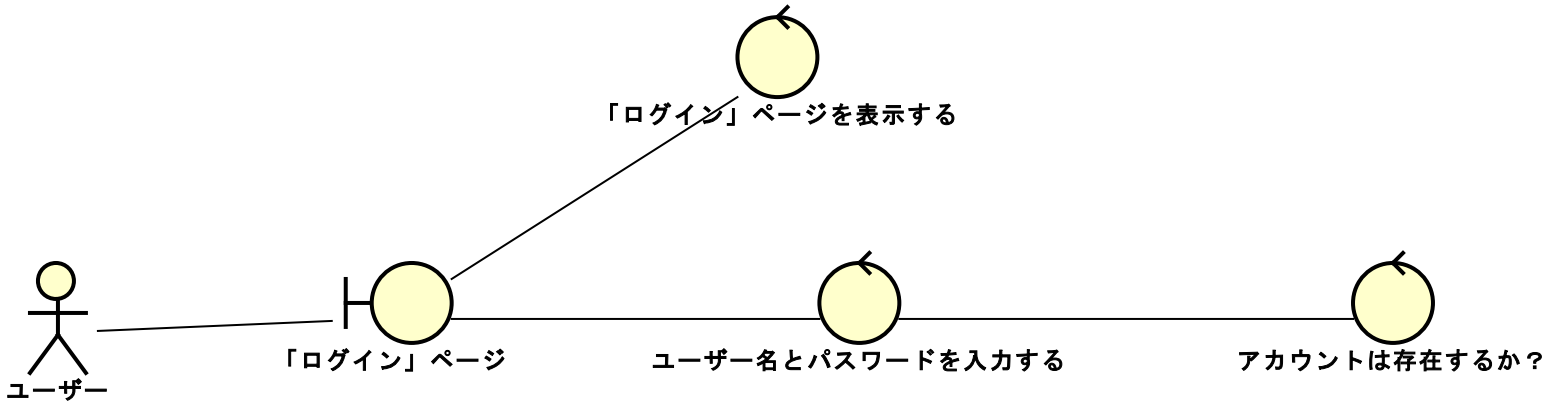
- コントローラー Controller
  - 処理や機能に相当，動詞で表現
  - 表示する，登録する，〇口をチェックする
  - CRUD+I が参考になるかも
    - Create, Read, Update, Delete, Index



実際の例


<<control>>
「ログイン」ページを表示する
<<actor>>
ユーザー
<<boundary>>
「ログイン」ページ
<<control>>
ユーザー名とパスワードを入力する
<<control>>
アカウントは存在するか？

もしくは,


「ログイン」ページを表示する
ユーザー
「ログイン」ページ
ユーザー名とパスワードを入力する
アカウントは存在するか？

astahでの書き方

- クラス図としてかく.
- クラス図のパレット中に，B, C, E のアイコンがある.
- デモ

記述上の注意

- ロバストネス図は，ユースケースモデル中の，ユースケース毎に記述する.
  - ユースケース記述をちゃんと書いてあるかをチェックするため.
- 名詞と名詞を線でつないではいけない.
  - × Entity - Entity, Boundary-Entity, Boundary-Boundary
- 本来，線には方向性を書く場合があるが，本講義では方向性は無しにする.

参考

MVCについて

- Model-View-Controller の略.
  - オブジェクト指向プログラミングで習ったかもしれない.
- アプリケーションを作る際に上記の三つに分けて設計すると良いという指針.
- Model
  - アプリで扱う業務や活動のみを扱う部分.
  - ショッピングサイトの業務なら商品，注文，顧客等がコレに相当.
  - 基本，システムとは関係ない業務依存の部分.
  - 主に普通のクラスやJavaBeans等で実現される.
- View
  - システムとしてユーザーと相互作用する部分．入出力.
  - ウエブアプリならウエブページに相当し，主にJSPが担当.
- Controller
  - ModelとViewを関連付け，業務の進行を制御する部分.
  - 主にServletが担当.

参考

MVCのメリット

- 特にModelと他を分離することで，実現方法を簡単に変更できる.
  - 例えば，ウエブアプリをやめて，アンドロイドの専用アプリを作ろうという時にも，Modelはそっくりそのまま流用できる. (Javaの場合，特に)
- Modelで表現される業務は往々にして類似したものが多いので，再利用ができる.
  - 我々が想像する以上に業務というのはワンパターン
    - アマゾンも楽天もやってることはほぼ同じ.
    - 吉野家，松屋，すき家もほぼ同じ.
  - アプリケーションフレームワーク等.

参考

CRUD + I

- Create, Read, Update, Delete の接頭語 (Acronym).
- データに対する最も一般的な処理群を表す.
- これに加えて，一覧(Index)の処理も実装する場合がある.

ロバストネス分析でのガイドライン 1/2

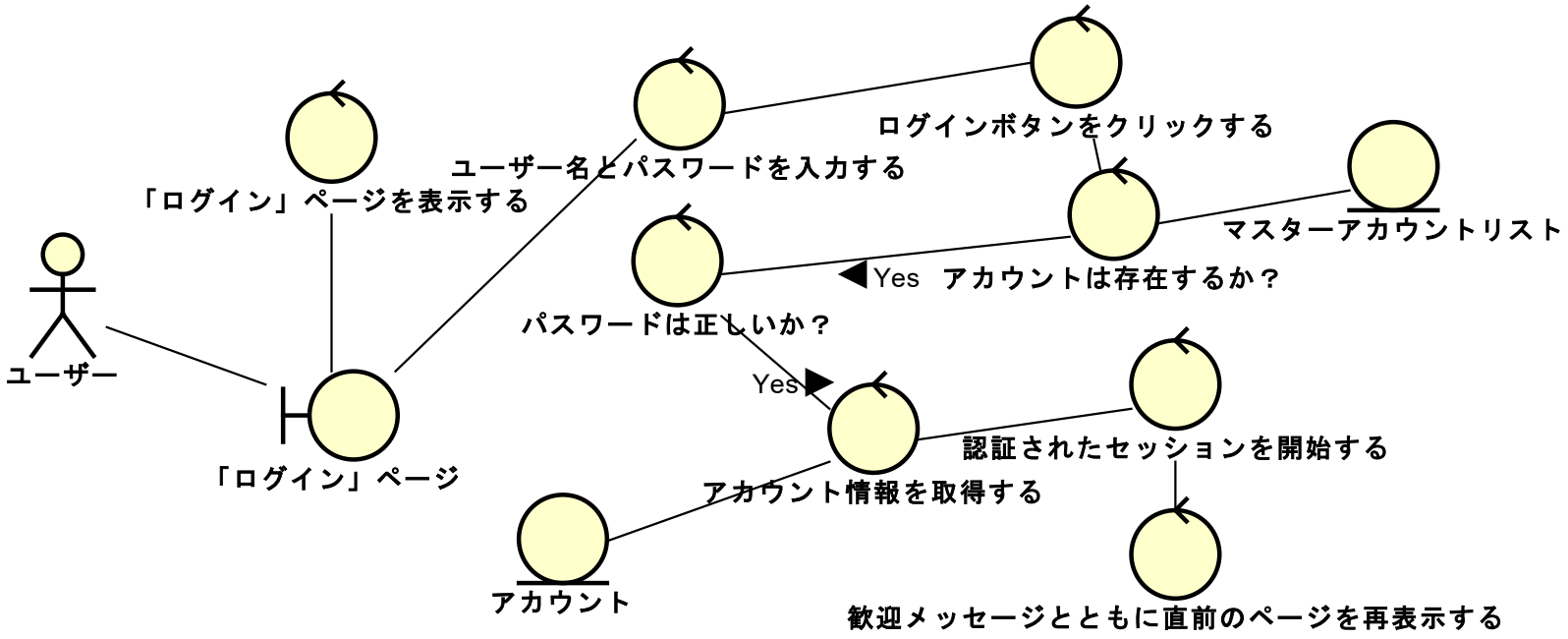
1. ユースケース記述の文言をロバストネス図に直接貼り付けよう.
2. エンティティオブジェクトはドメインモデルから取り出し，不足していれば追加しよう.
3. ロバストネス図作成中でも必要ならユースケース記述を直そう.
4. 画面単位にバウンダリオブジェクトを作成しよう.
5. コントローラは通常，ソフトウェアの論理的な機能であることを思い出そう.
   - コントロールという名前にあまり気をとられないで.

ロバストネス分析でのガイドライン 2/2

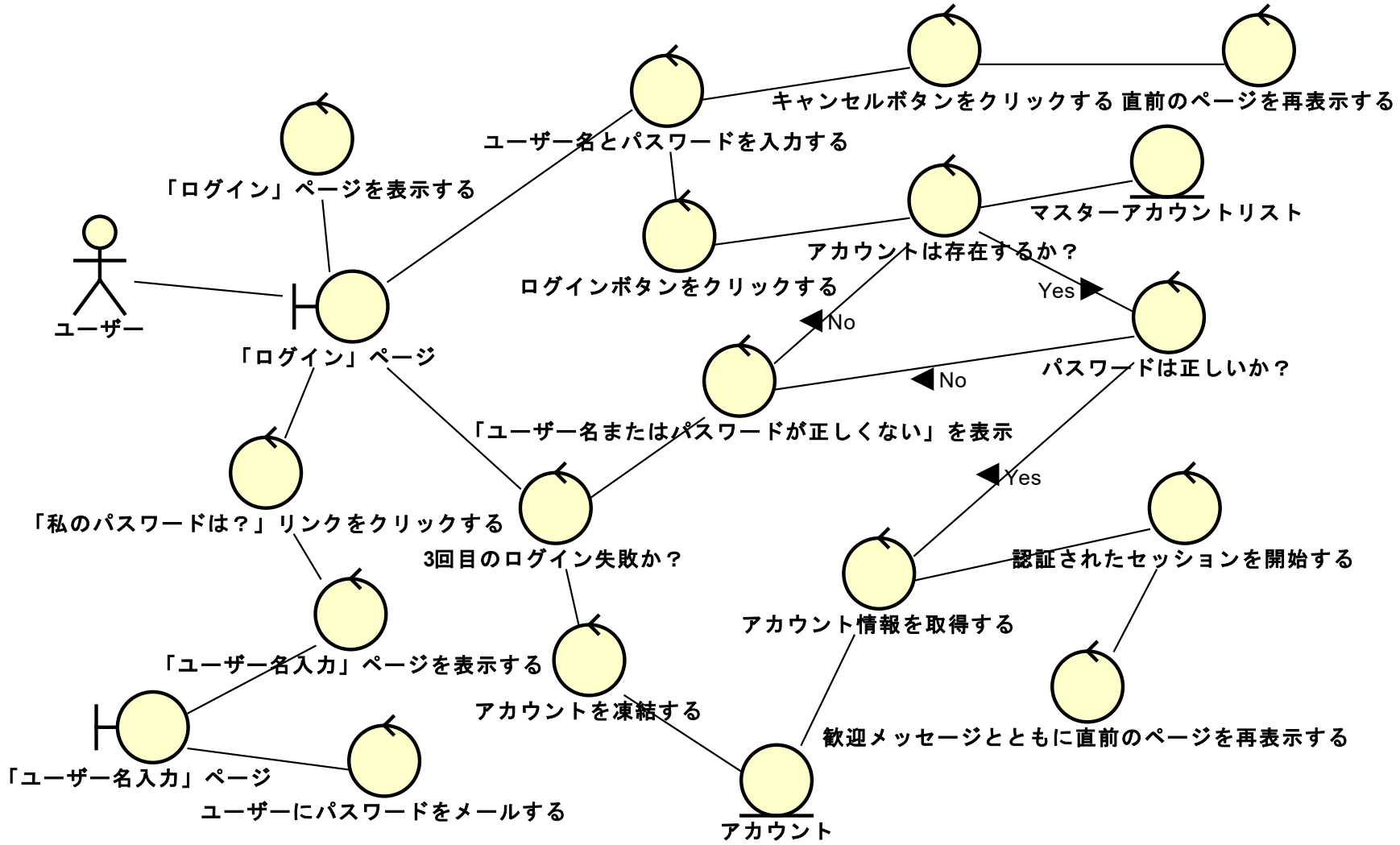
1. ロバストネス図中の線の矢印の方向は気にしないで.
   - 本講義ではそもそも方向性を書かない.
   - ロバストネス図の目的はユースケースとドメインモデルの改善であるため.
2. 呼び出し元のユースケースをロバストネス図に書いても良い.
3. ロバストネス図はユースケースに対する予備的な概念設計である.
   - 要求を理解するための，試験的な設計ということ.
4. ロバストネス図のオブジェクト群は詳細設計で姿を変える.
   - 詳細は詳細設計にて.
5. ロバストネス図はユースケースの「オブジェクトの絵」である.

実際の記述例「ログイン」ユースケース

- ユースケース記述 p121ucd.xlsx を webclassからダウンロードしてください.
  - se03sample.zip にはいってます.
- とりあえず以下，基本コースのみのロバストネス図



例外も含む


ユーザー
「ログイン」ページを表示する
「ログイン」ページ
ユーザー名とパスワードを入力する
キャンセルボタンをクリックする
直前のページを再表示する
ログインボタンをクリックする
アカウントは存在するか？
マスターアカウントリスト
Yes
No
パスワードは正しいか？
No
「ユーザー名またはパスワードが正しくない」を表示
「私のパスワードは？」リンクをクリックする
3回目のログイン失敗か？
Yes
認証されたセッションを開始する
アカウント情報を取得する
「ユーザー名入力」ページを表示する
アカウントを凍結する
歓迎メッセージとともに直前のページを再表示する
「ユーザー名入力」ページ
ユーザーにパスワードをメールする
アカウント

パターン

- B - C:表示する - C:取得する - E
  - データやリストのよくある表示
- B - C:押すやクリック - B
  - ページ遷移
- C:入力する - C:・・・か？ -yes- C: -no- C:
  - 条件分岐みたいなもの

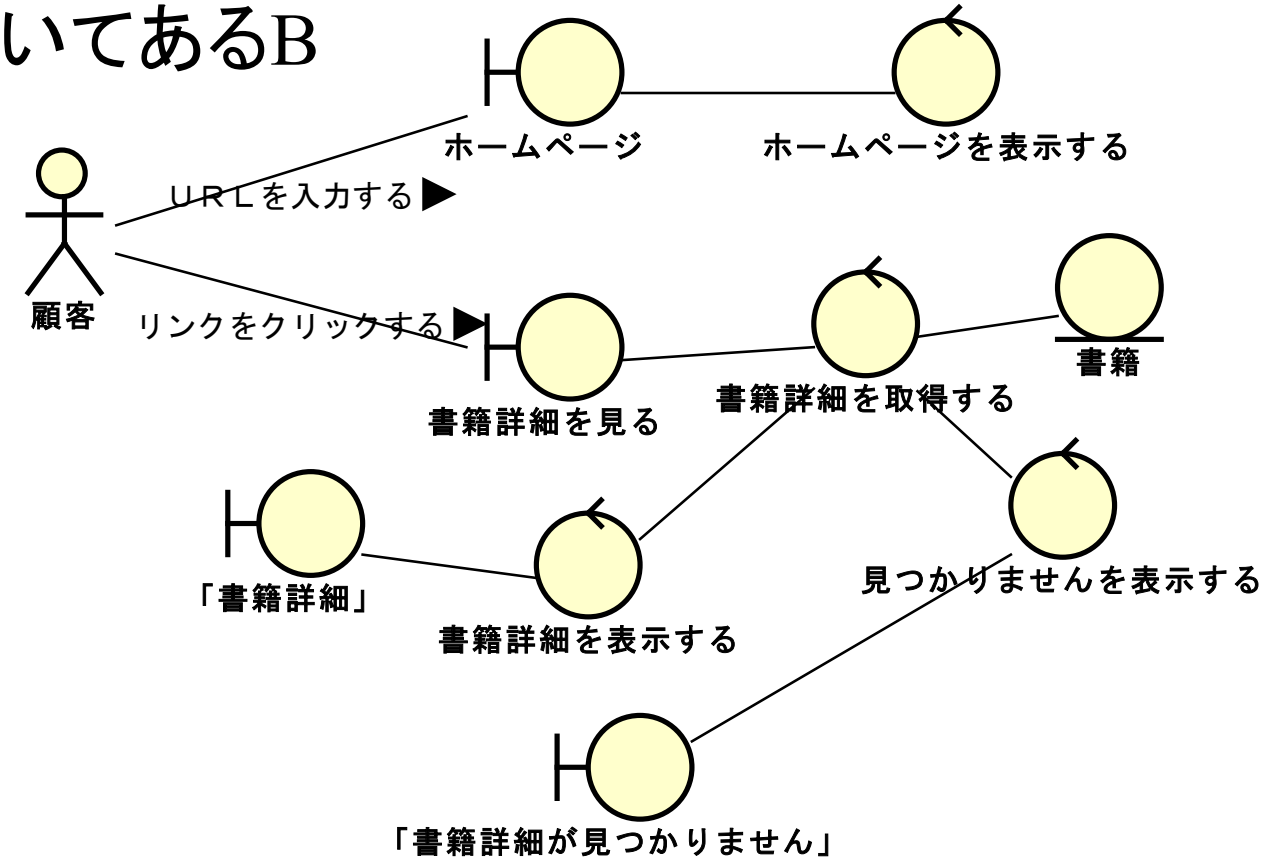
※ B=バウンダリ C=コントローラー E=エンティティ

例題1

- 書籍詳細を表示する
- ファイルは webclassのse03sample.zipから得てください
- ユースケース記述 p131ucd.xlsx
- ロバストネス図 p131rbst.asta

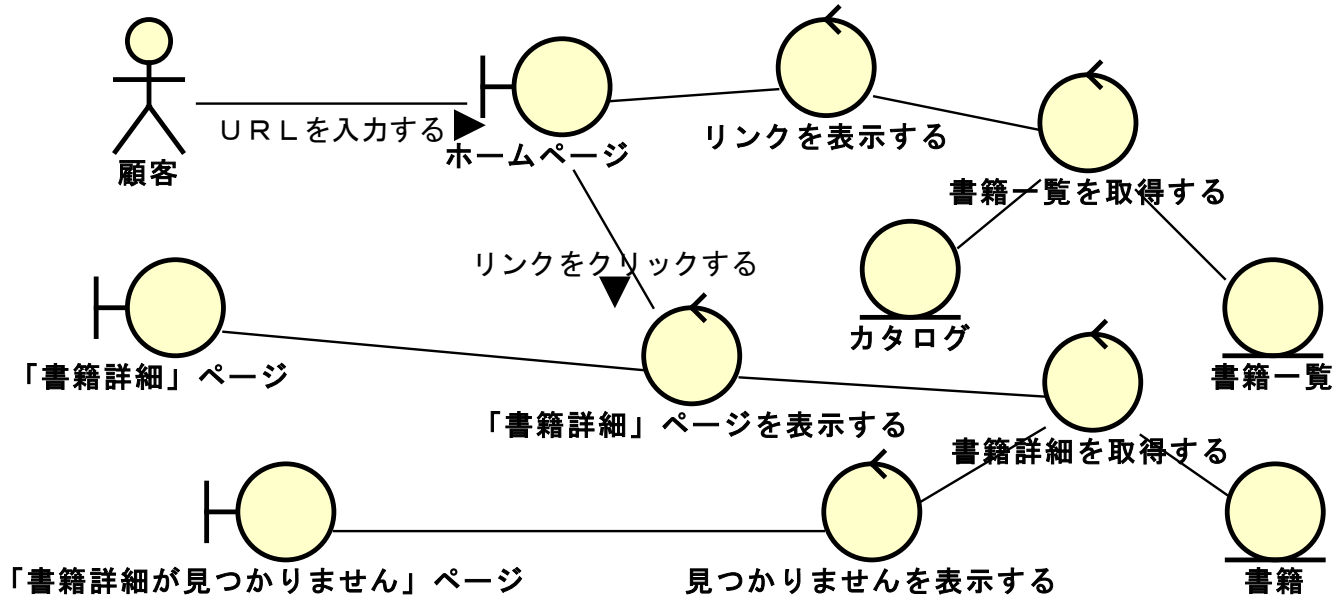
書籍詳細を表示する V.1

- B: 書籍詳細を表示する
- B: 書籍詳細を見る, B: 書籍詳細
- 動詞で書いてあるB



書籍詳細を表示する V.2

- 動詞表現のバウンダリが無くなった.
- 同じ意味っぽいバウンダリが統合された.

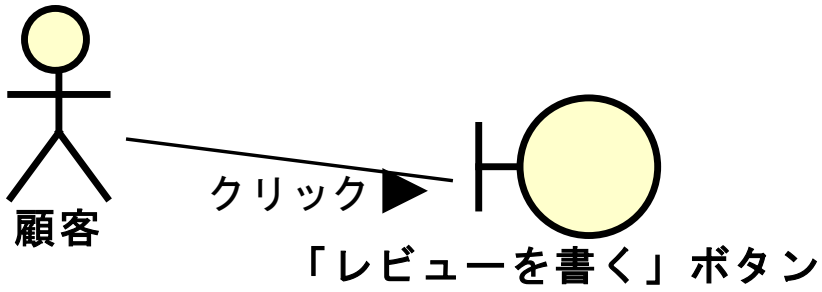


例題2

- 顧客レビューを書く
- ファイルは se03sample.zipから得てください
- ユースケース記述 p133ucd.xlsx
- ロバストネス図 p133rbst.asta

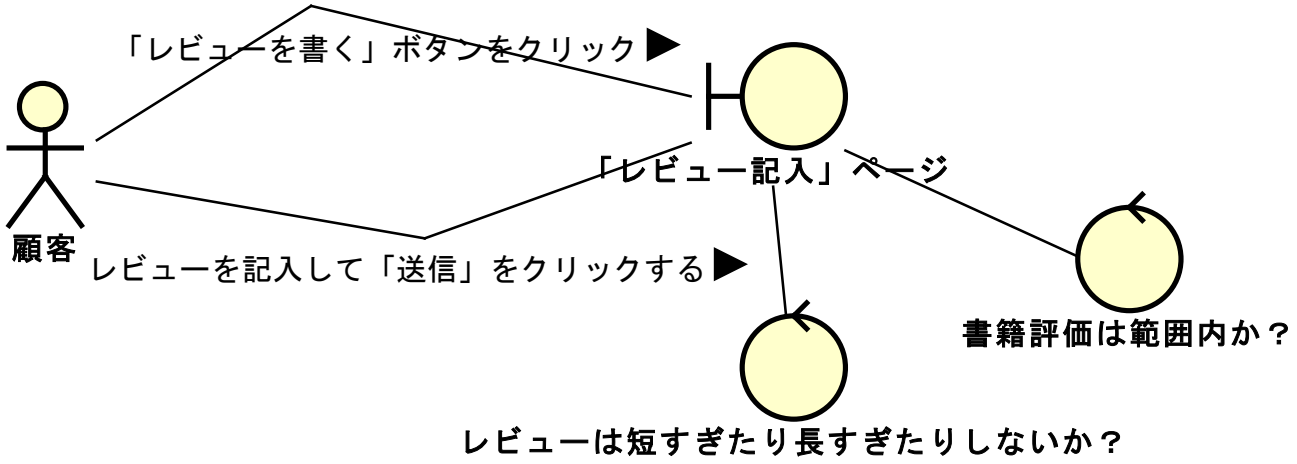
顧客レビューを書く V.1

- ボタン，ラベル，リストボックス等をバウンダリにするのは好ましくない.
  - あまりに詳細すぎる.
- なんとか画面くらいの粒度にしておくべき.



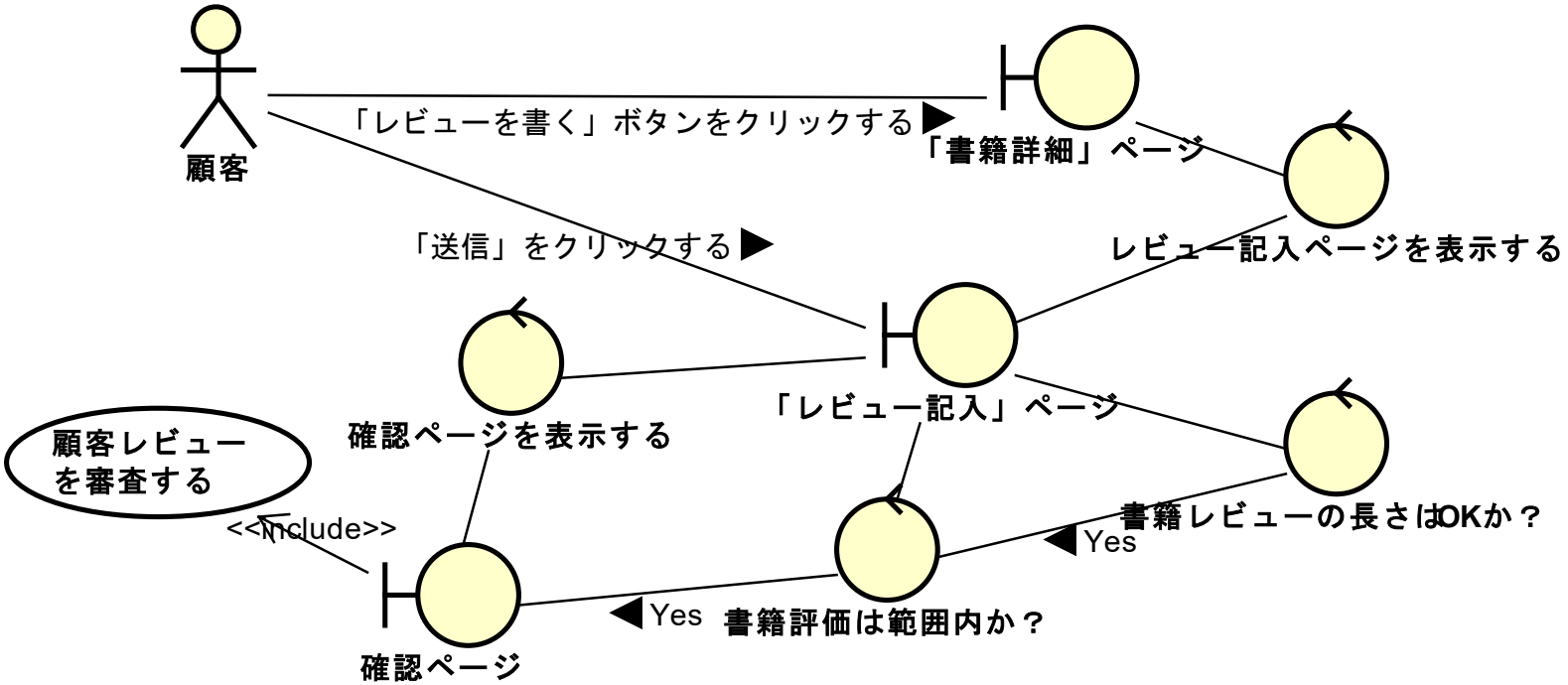
顧客レビューを書くV.2

- レビューと「レビュー記入」ページ間の関連(メッセージ)がおかしい.
  - 「レビューを書く」ボタンは1個前の画面でクリックじゃない？
- レビューは短すぎたり長すぎたりしないか？はちと長い.

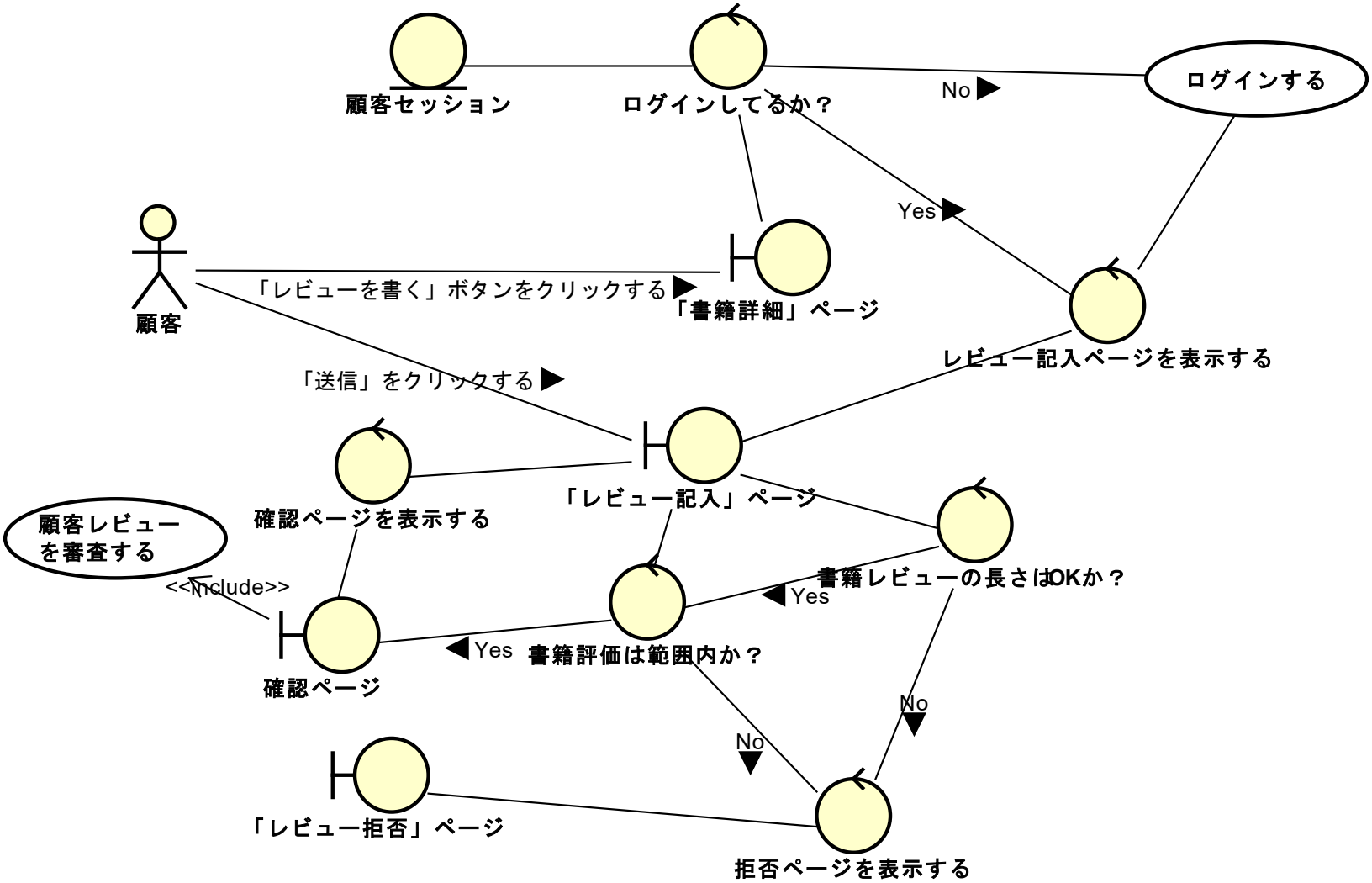


顧客レビュー V.3

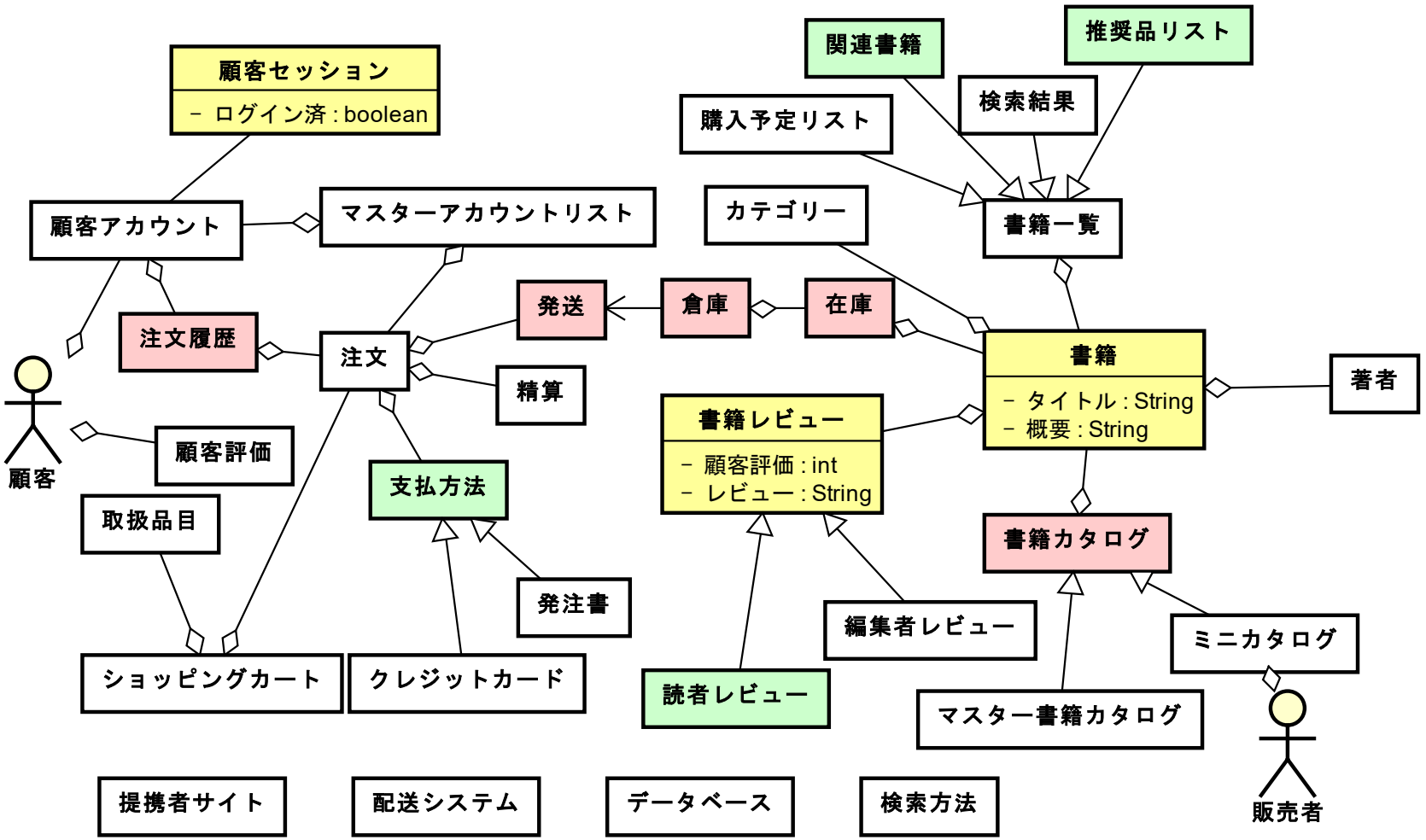
- 例外コースをフォロウしてない.



顧客レビュー V.4


顧客セッション
ログインしてるか？
No
ログインする
Yes
顧客
「レビューを書く」ボタンをクリックする
「書籍詳細」ページ
レビュー記入ページを表示する
「送信」をクリックする
確認ページを表示する
「レビュー記入」ページ
顧客レビュー
を審査する
書籍レビューの長さはOKか？
<<Include>>
Yes
Yes
書籍評価は範囲内か？
確認ページ
No
No
「レビュー拒否」ページ
拒否ページを表示する

ドメインモデルの拡充


顧客セッション
- ログイン済 : boolean
関連書籍
推奨品リスト
検索結果
購入予定リスト
顧客アカウント
マスターアカウントリスト
カテゴリー
書籍一覧
発送
倉庫
在庫
注文履歴
注文
書籍
- タイトル : String
- 概要 : String
著者
精算
顧客
顧客評価
書籍レビュー
- 顧客評価 : int
- レビュー : String
支払方法
取扱品目
書籍カタログ
発注書
編集者レビュー
ミニカタログ
ショッピングカート
クレジットカード
読者レビュー
マスター書籍カタログ
販売者
提携者サイト
配送システム
データベース
検索方法

演習1

- webclass をみてください.
- 11/7(月) 正午〆切でお願いします.
- ファイルが複数になるので, zipで1個にして提出してください.
  - webclassでは複数ファイルがアップできない模様.

本日は以上