責務駆動設計超入門

(株)オージス総研
技術部アジャイル開発センター
藤井　拓

© OGIS-RI Co., Ltd.

アウトライン

- 責務駆動設計とは
- 会話
- CRCカード
- ロールステレオタイプ
- 適用事例
- まとめ

© OGIS-RI Co., Ltd.

責務駆動設計とは

© OGIS-RI Co., Ltd.

責務駆動設計(RDD: Responsibility-Driven Design)とは

- Rebecca Wirfs-Brockさんらが、“Designing Object-Oriented Software (1990)”で提案した設計手法
- クラス、責務、コラボレータ (CRCカード）に基づいて設計する手法
- “Object Design – Roles, Responsibilities, and Collaborations (2002)”でさらにパワーアップ！
  - UMLも取り込み、より包括的な設計手法へと進化

© OGIS-RI Co., Ltd.

責務駆動設計の良さ

- オブジェクトの発見を強力にサポート
  - 基本道具：会話、CRCカード、ロールステレオタイプ
- 大まかなモデルをすばやく作れる
  - ハイレベルな設計モデルが素早く作れる
- より良い設計への指針が満載
  - よい設計を作るためのガイドラインやヒントが数多く提供されている

© OGIS-RI Co., Ltd.

責務駆動設計の良さ-実感

- 敷居が低い
  - より多くの人がモデリングできる
- モデリングのスピードが上がる
  - 本質的でないところにあまり悩まない工夫がある

© OGIS-RI Co., Ltd.

責務駆動設計の基本道具

- 会話（形式のユースケース）
  - ソフトウェアの機能要求をどのように実現できるかという順序を追ったお話
- CRCカード
  - ユースケースの実現を満足するのに、必要なクラスの候補、責務、コラボレータを考える
  - クラス候補ですべてのユースケースの実現を満たせるか確認する

© OGIS-RI Co., Ltd.

ユースケース

ユーザとシステムの相互作用を順番に文書で記述する

- 散文
  - 機能の概要を散文で記述
- シナリオ
  - システムとユーザの相互作用を順番に記述
- 会話
  - ユーザ操作とシステムの責務で記述

会話は設計の最初のステップと捉えた方がよい！

© OGIS-RI Co., Ltd.

会話

© OGIS-RI Co., Ltd.

会話とは

会話は、発生する順序で以下の2つを記述する

- ユーザの操作
  - ユーザのソフトウェアに対する操作
- システムの責務
  - ユーザの操作に対してソフトウェアが内部的に行わなくてはならないこと

© OGIS-RI Co., Ltd.

会話の例
「図書の貸し出し」の会話

ユーザーの操作	システムの責務
図書利用者カードをバーコードスキャナーでスキャンする	■ バーコードの値を受信し、バーコードの値に対応する利用者の情報を取得する
貸し出し希望の図書をバーコードスキャナーでスキャンする	■ 「図書利用者の現在の貸し出し数 ≦ 貸し出し上限数 – 1」であることを確認する ■ バーコードの値を受信し、バーコードの値に対応する図書を取得し、貸し出し状態にする ■ 図書利用者の現在の貸し出し数を1増やす

© OGIS-RI Co., Ltd.

会話の長所

- 要求と設計のギャップを埋める
  - システムの大まかな機能を抽出しやすい
    - 大まかな機能がはっきりさせると、オブジェクトを見つけやすい
- より柔軟に相互作用が表現できる
  - 相互作用のステップが選択的に実行されることやループなどが簡単に表現可能

© OGIS-RI Co., Ltd.

CRCカード

© OGIS-RI Co., Ltd.

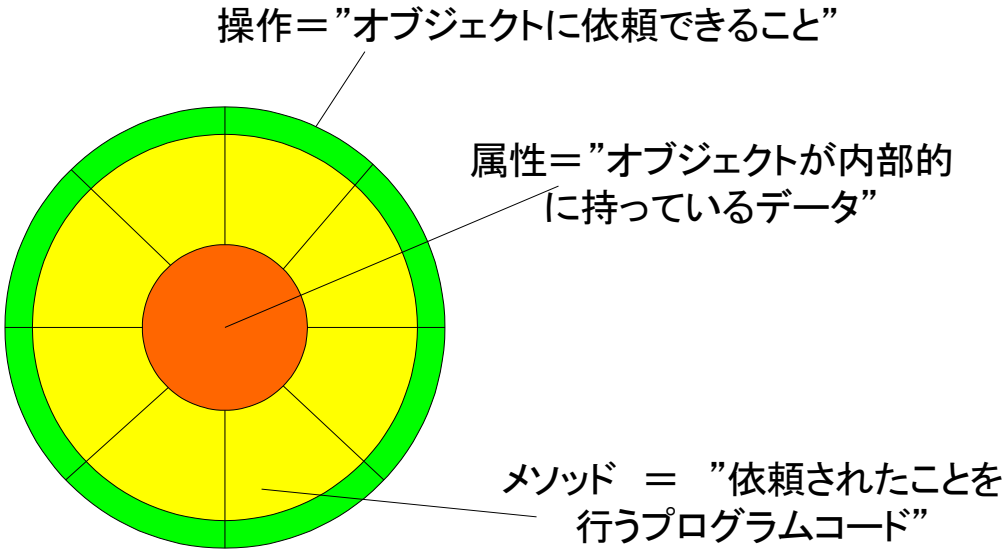
CRCモデル

- Class Responsibility Collaboratorの略
  - 日本語訳では、クラス-責務-コラボレータ
- Kent Beckらが生み出した、CRCカードを用いるモデリング手法
  - クラス毎に責務とコラボレータをカードに書き出す
  - 図を描かなくても、シナリオに即した相互作用モデルを作れる簡便さが特徴

© OGIS-RI Co., Ltd.

オブジェクトの基本構造

- オブジェクト指向は、変更に強いソフトウェアを実現するためにオブジェクトを基本構造としている

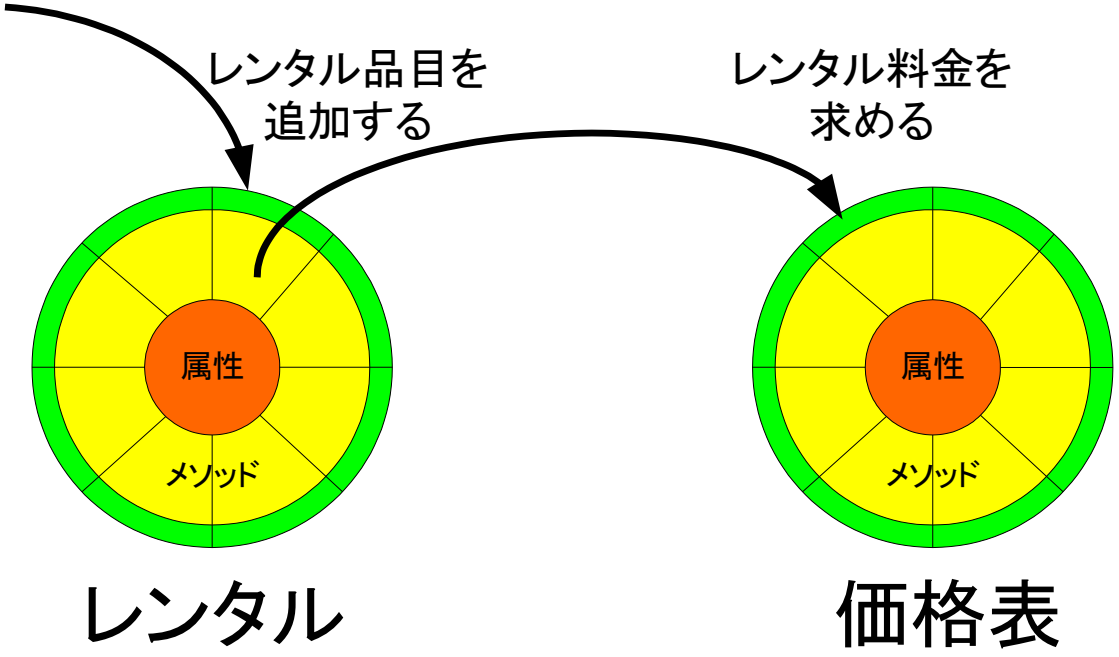


オブジェクト

© OGIS-RI Co., Ltd.

オブジェクト間のコラボレーション

- オブジェクトは、依頼されたことを実現するためにメソッド内で（他の）オブジェクトの操作を呼び出す



© OGIS-RI Co., Ltd.

CRCカードの見た目（表側）

クラス名：
目的：

何をするクラスか簡潔に説明できないクラスは問題

© OGIS-RI Co., Ltd.

CRCカードの表側に記入すること

- 目的
  - そのクラスが行うことの概要を記述する
    - 他のクラスとの関わりについても、簡単に言及したほうがよい
  - 5行程度までで記述できることが望ましい
- ロールステレオタイプ（後述）
- デザインパターンのロール（上級）
  - デザインパターンのロールに対応する場合は、そのロール名を書く

© OGIS-RI Co., Ltd.

CRCカードの見た目（裏側）

クラス名：	
責務	コラボレータ

© OGIS-RI Co., Ltd.

責務とは

- そのクラスが責任を持って行うこと
  - 基本的には“操作”の別の呼び方
  - 各クラスは、まとまりのある少数の責務を担う
- 責務には、以下の3種類がある
  - ～を知っている
    - 住所を知ってる
  - ～を行う
    - 貸し出し書籍を追加する
  - ～を判断する
    - 貸出の上限数を超えないか判断する

© OGIS-RI Co., Ltd.

コラボレータとは

- コラボレーションとは、複数のクラスが自分たちの責務を組み合わせて特定の機能を実現すること
- コラボレータとは、あるクラスが責務を果たすのに必要な責務を提供する他のクラスのこと
- コラボレータは、見つけた順番にCRCカードの右の列に書き込む

© OGIS-RI Co., Ltd.

例題1:レンタルと料金表

- 「オブジェクト間のコラボレーション」というスライドで図示されていた「レンタル」と「料金表」の間のコラボレーションをCRCカードで表現してください
- CRCカードの裏側に責務とコラボレータだけ記入してください

© OGIS-RI Co., Ltd.

例題2：銀行の窓口業務

- 以下で説明されている銀行の窓口業務をCRCカードでモデリングしよう

- 窓口係
  - 顧客から通帳、ハンコ、引き出し依頼書をあづかり、預金の引き出しの依頼をうける
  - 依頼が完了したら、顧客にお金、通帳、ハンコを渡す

© OGIS-RI Co., Ltd.

例題2：銀行の窓口（続き）

- 入手金係
  - 引き出し依頼書に記入された金額を入手金機に入力したり、現金や通帳を挿入し、預け入れ/引き出しを行う
  - 預け入れ/引き出し金額を確認し、結果を窓口係に渡す
- 入出金機
  - 指定された金額または現金で現金の受け払いを行い、受け払いした現金の金額に対応して口座の残高を調整し、口座の残高を通帳に記帳する

© OGIS-RI Co., Ltd.

並行処理と順次処理

- 並行処理
  - 複数の責務を並行して処理できる（コラボレータの責務の完了を待たない）
- 順次処理
  - 1つの責務からコラボレータの責務を呼び出す場合、コラボレータの責務が完了するまで呼び出した側の責務は完了しない

ソフトウェアの動作の多くは順次処理で考えてよい

© OGIS-RI Co., Ltd.

CRCカードの利点

- 悩まずに責務を記述できること
  - コラボレーションでオブジェクトが行うことを簡潔に「文」で表現すればよい
- 関係で悩まないこと
  - クラス間の関係をどう接続するかという点や、多重度などについて悩まない

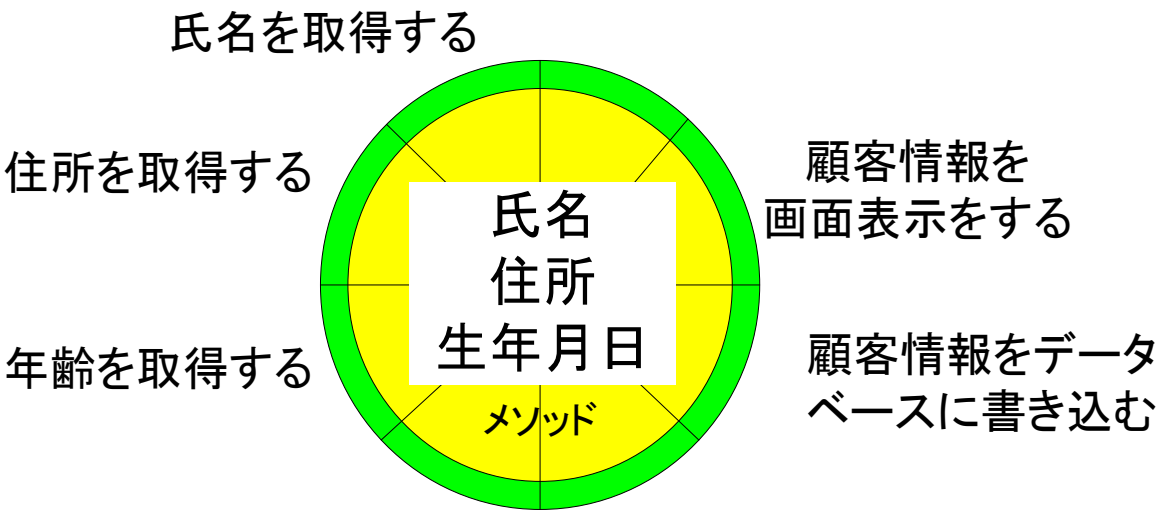
© OGIS-RI Co., Ltd.

変更に強い設計とは

- 各クラスの定義が、以下のような条件を満足していることが望ましい
  - あまり多くのことをしない
    - 20以上も操作があるクラスは品質が低い可能性あり
  - 比較的少数のクラスとコラボレーションする
  - 備えている操作に統一性がある
    - クラスの名前（目的）と各操作に整合性がある

© OGIS-RI Co., Ltd.

備えている操作に統一性がない



顧客クラス

© OGIS-RI Co., Ltd.

オブジェクトになるもの とならないもの

- 責務を持つものは、オブジェクトである
  - 何かを知っている
  - 演算や処理を行う
  - なんらかの判断をする

© OGIS-RI Co., Ltd.

ロールステレオタイプ

© OGIS-RI Co., Ltd.

ロールステレオタイプ-1

クラスを見つけるヒントとすべく、クラスが一般的に持つ役割をRebecca Wirfs-Brockが分類したもの

- 構造化役
  - 階層関係などの構造を形成する役割
- 情報保持役
  - 個人の名前、住所などの情報を保持する役割
- サービス提供役
  - 演算を実行するなどの特定のサービスを実行する役割

© OGIS-RI Co., Ltd.

ロールステレオタイプ-2

- インタフェース役
  - 通信,UI,ライブラリ呼び出しなどシステムの境界で外界とのやり取りを行う役割
- 調整役
  - 特定の依頼を複数のコラボレータに分配する役割（条件判断はあまり行わない）
- 制御役
  - 特定の依頼を果たすため、複数のコラボレータに指示を出し、判断する役割（条件判断を行う）

© OGIS-RI Co., Ltd.

ロールステレオタイプについての補足

- 構造化役
  - 集合を保持するクラス
  - 複雑な構造の骨格となるクラス
    - 例えば、「文書」
- 注意点
  - 1つのクラスが複数のロールステレオタイプを持ってもよい

© OGIS-RI Co., Ltd.

設計のステップ-1

- 要求：まず、ソフトウェアがしなければならないことや画面を決める
  - 会話（「ユーザの操作」と「システムの責務」）を定義
- 設計：要求に書かれていることを果たすために、ソフトウェアが内部で行うことを考える
  - 「システムの責務」を行うために必要なことをもう1段階分解して、文で書き出す

© OGIS-RI Co., Ltd.

例題3：住所録管理

- 個人の氏名や住所などを管理するための住所録管理用ソフトウェアをつくろうと考えた
- まず、画面から個人の氏名や住所などを入力して個人を登録する機能の設計を考えることから設計を開始しようと考えた

© OGIS-RI Co., Ltd.

例題3：住所録管理（続き）

- さらに、最初は画面を構成する細かいオブジェクトやデータベースなどのデータをずっと保存する仕組みは考えないことにした

- 住所録管理用ソフトウェアが内部的に行うことはなんだろうか？

© OGIS-RI Co., Ltd.

設計のステップ-2

- クラスの候補を見つける
  - 内部的に行うことを引き受けるクラスを考える
    - ロールステレオタイプを手がかりにクラスの候補を見つける
    - クラスの目的がかけないクラスの候補は捨てる
  - 最終的には、クラスの目的をつなげると、内部的に行うことすべてが表現されているべき

© OGIS-RI Co., Ltd.

例題4：住所録管理

- 例題3で説明した住所録管理ソフトウェアに必要なクラスを見つけよう
- 見つけたクラスに、そのクラスの目的を書こう
  - 目的は、「～をする（ためのクラス）」と書けばよい

© OGIS-RI Co., Ltd.

設計のステップ-3

- 責務やコラボレータを見つける
  - 画面のイベントなどのきっかけをうける責務を決める
  - 責務を果たすために必要な他のクラスをコラボレータの欄に記入する
    - 自分自身の責務を呼び出す場合は、コラボレータ欄にクラス名を記入する必要はない
  - さらに、コラボレータにおいて呼び出す責務を決める
- 最終的に、内部で行うことがすべて責務とコラボレータによりコラボレーションで表現されればよい

© OGIS-RI Co., Ltd.

例題5：住所録管理

- 例題4で考えたクラスの候補に、「個人を登録する」のに必要な「責務」と「コラボレータ」を定義しよう

- 画面に追加すべき個人の氏名、住所などが入力され、OKボタンが押されたところを始点に「責務」と「コラボレータ」を考えればよい

© OGIS-RI Co., Ltd.

責務の記述に関する注意点

- 責務は、1-2行で簡潔に記述した方がよい
- でも、簡潔すぎて行うことが類推できない責務の定義はよくない
  - 「実行する」、「管理する」などは避けた方がよい
- クラス全体で責務にまとまりがある方がよい

© OGIS-RI Co., Ltd.

適用事例

© OGIS-RI Co., Ltd.

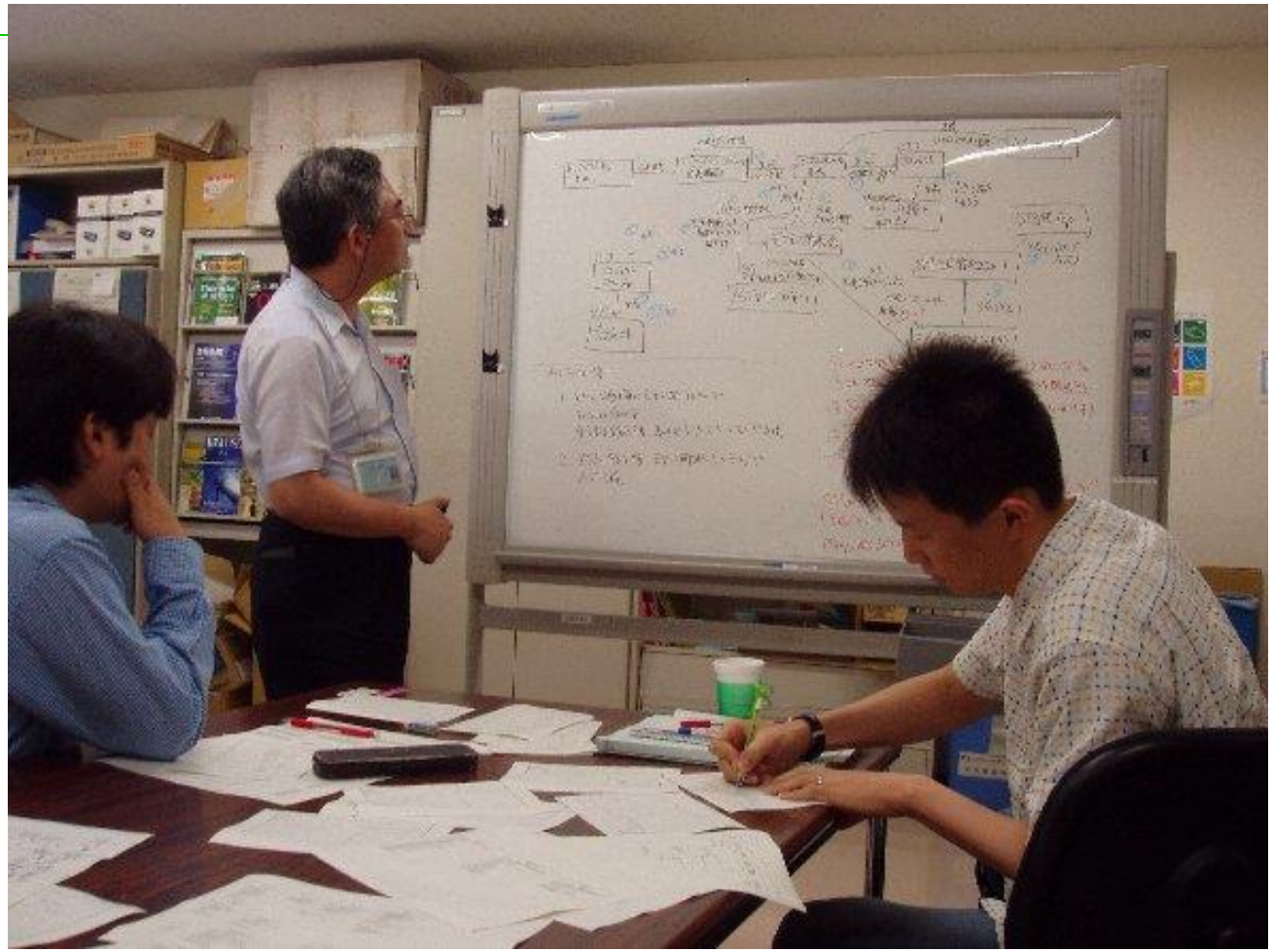
事例：Aプロジェクトの概要

開発内容：RMIやXMLを使った、ややロジック複雑度の高いC/Sシステム

- 実装言語：Java
- 開発メンバー
  - リーダ
    - Java実装経験：6ヶ月、モデリング実践経験：なし
  - メンバー2名
    - Java実装経験：2ヶ月、モデリング実践経験：2ヶ月

© OGIS-RI Co., Ltd.

詳細設計モデリングセッションの実践風景



© OGIS-RI Co., Ltd.

事例：Bプロジェクトの概要

- 開発内容：特定の分野の複数のアプリケーション群を効率的再開発するためのフレームワーク作り
  - 実装言語：Java
  - モデリングメンバー
    - UMLのモデリングに精通した開発者2名

© OGIS-RI Co., Ltd.

プロジェクトBの課題と解決策

- 課題
  - 共通機能をコンポーネントに集約しても、アプリ固有のコードは劇的に減らない
  - アプリごとの設計作業量を減らしたい
- 解決策
  - アプリケーションのロジックの共通のコラボレーションパターンを抽出する

© OGIS-RI Co., Ltd.

プロジェクトB:
解決策の実行方法

- ユースケースを構成する基本機能を整理する
  - 「システムの責務」に必要な機能を追加する
- 各ユースケースを実現するためのクラスを抽出する
  - CRCカードの表面のレベルで、大きな役割分担を決める(ロールステレオタイプを意識)
- 基本機能を実現するコラボレーションを考える
  - CRCカードの裏面で、責務とコラボレータを決める

アプリを構成する基本機能が25パターンに整理できた！

© OGIS-RI Co., Ltd.

プロジェクトB:
モデリング結果と考察

- 前のスライドで説明した責務駆動設計風のアプローチを採用することで、2週間程度でパターンの抽出が完了した

- 考察
  - クラス、責務、コラボレータような大まかなモデルの方がパターンのような共通性を見つけやすい
  - モデリングスピードが速いので、ユースケース全体を早く網羅できる

© OGIS-RI Co., Ltd.

まとめ

© OGIS-RI Co., Ltd.

まとめ

- 責務駆動設計は、クラスの役割、責務、コラボレーションを注目した設計手法である
- 責務駆動設計の基本道具は、会話、CRCカード、ロールステレオタイプである
- ２つの適用事例を通じて、責務駆動設計の有効性が実感できた

© OGIS-RI Co., Ltd.

参考資料

- レベッカ・ワーフスブラック他：“オブジェクトデザイン－ロール、責務、コラボレーションによる設計手法“, 翔泳社, 2007

- Rebecca Wirfs-Brock, Brian Wilkerson, Lauren Wiener, Designing Object-Oriented Software, Prentice Hall, 1990

© OGIS-RI Co., Ltd.