ソフトウェア工学

05: 要求分析／定義と設計(Ⅲ)
– ERモデリング、OOAD、UML

理工学部 経営システム工学科
庄司　裕子

今回のテーマ

要求分析／定義と設計(Ⅲ)
– ERモデリング、OOAD、UML
- 開発プロセスにおける位置づけと目的
- ERモデリング
- オブジェクト指向分析／設計(OOAD)
- UML

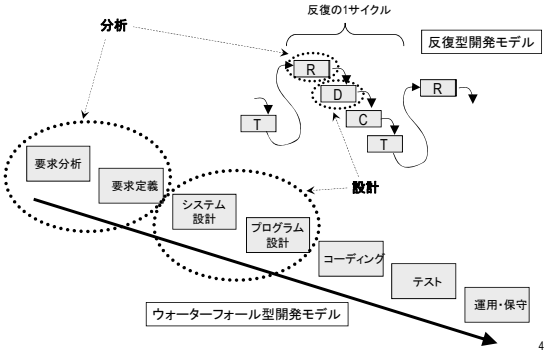
2

要求分析／定義と設計(Ⅲ)
– ERモデリング、OOAD、UML

⇨開発プロセスにおける位置づけと目的
- ERモデリング
- オブジェクト指向分析／設計(OOAD)
- UML

3

開発プロセスにおける位置づけ



4

適用範囲

- ERモデリング
  - 設計、特にデータベース設計
- OOADおよびその発展形のUML
  - 分析/設計フェーズ全般

5

プロセス中心 v.s. データ中心

☞分析／設計のフォーカスをプロセス(処理)に置くか、データに置くか
- 構造化分析／設計
  ⇨プロセス中心(Process Centered)
- ERモデリング
  ⇨データ中心(Data Centered)
- OOADおよびその発展形のUML
  - データ中心＋プロセス中心

6

要求分析／定義と設計（Ⅲ）
– ERモデリング、OOAD、UML

✓開発プロセスにおける位置づけと目的
⇨ERモデリング
- オブジェクト指向分析／設計（OOAD）
- UML

7

ERモデリングの目的

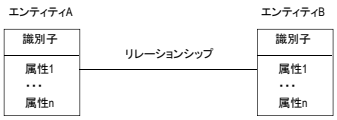
- 構造化分析／設計では
  - システムを構成するモジュールと各モジュール間のインタフェースを明確にする
  - 扱うデータはデータ ディクショナリに定義
  - ☞データ同士の関係、特に、ストアに格納されるデータの取り扱いは不明
- ERモデリングでは
  - ⇨システムに登場するデータの「静的な」関係や構造を明らかにする

8

ERモデリングとは

システムで扱うデータを
- 実体（エンティティ：Entity）
- 実体間の関係（リレーションシップ：Relationship）

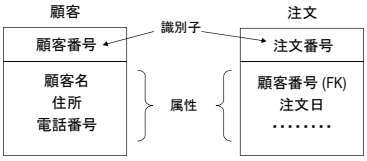
で表現する
⇨データベースの論理設計に適用される



9

エンティティとは

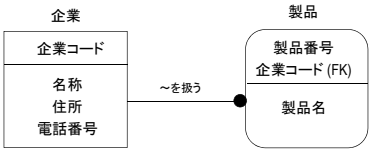
- 実世界の事物や概念の集合を表す
- 他のエンティティと区別するための一意な識別子（属性の一種）を持つ
- 属性を持つ



10

(FK) の付いた属性

- それが他のエンティティの識別子になっていることを示す
  - FK は Foreign Key（外部キー）の略
- 特に、識別子に他のエンティティの識別子を含んでいるエンティティは依存エンティティ



11

ER図（ERD）

- エンティティとエンティティ間の関係を表したダイアグラム
- 具体的な表記法は複数ある
  - IDEF1X
  - P. Chenによる表記
  - J. Martinによる表記
  - T字型ER法
  - ・・・

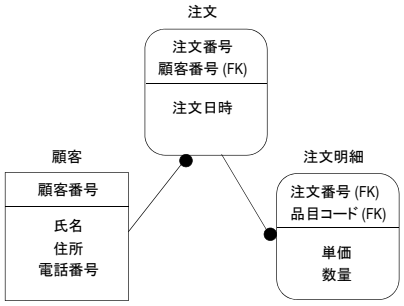
12

ERDの表記法（IDEF1Xの場合）

- 四角形
  - ⇨独立エンティティ（他のエンティティに依存しない）
- 角の丸い四角形
  - ⇨依存エンティティ（他のエンティティに依存する）
- 実線
  - ⇨依存エンティティとの依存リレーションシップ
- 破線
  - ⇨独立エンティティ同士の非依存リレーションシップ

13

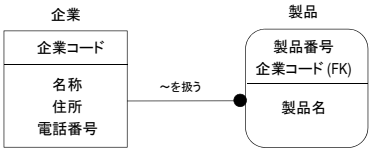
ERDの例（IDEF1Xの表記法）



14

リレーションシップの多重度

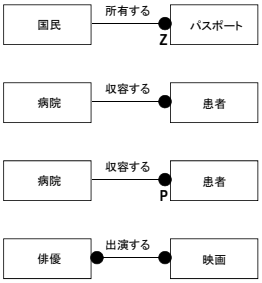
- 関係するエンティティの数の範囲を表す
  - 1対（0または1）：ー●Z
  - 1対多（0以上）：ー●
  - 1対多（1以上）：ー●P
  - 多対多：●ー●



15

多重度の表記法

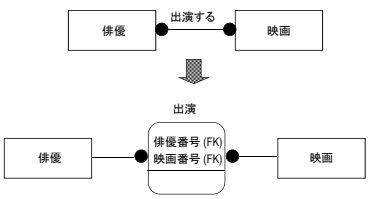
- 1対（0／1）
- 1対多（≧0）
- 1対多（≧1）
- 多対多



16

多対多の関係のモデリング

- 多対多の関係にあるエンティティの場合には、その両者を関係づけるもう1つのエンティティを登場させるのが普通



17

ERモデリングのポイント

- 識別子は必ず一意になっているか
- エンティティ間の依存関係は適切か
  - ☞他のエンティティがなければ存在できないのかどうか慎重に判断する
- リレーションシップの多重度は適切か
- 属性は適切か
  - ☞現実に存在しない属性をモデリング上の必要性から定義することは避ける
  - ☞使用状況をイメージしながら判断する

18

考えてみよう！

- 個人成績表に、学生番号、氏名、クラス名という個人情報と、試験番号、科目名、成績（得点）という成績情報が記載されているとする
- これをERモデリングして、ER図を書いてみよう

ヒント：

☞「学生」、「クラス」、「試験」という独立エンティティ（「もの」を表す）と、「成績」、「所属」という依存エンティティ（関係を表す）を用意する

19

要求分析／定義と設計（Ⅲ）
– ERモデリング、OOAD、UML

✓開発プロセスにおける位置づけと目的
✓ERモデリング
⇨オブジェクト指向分析／設計（OOAD）
- UML

20

オブジェクト指向分析／設計

- OOADと略称されることが多い
- システム分析／設計に対するデータ中心アプローチとプロセス中心アプローチを、オブジェクトという基本単位で統一的に扱う方法
- オブジェクトという情報的凝集度のモジュールに、扱うデータとそれを処理する手続きをひとまとめにし、システム全体の処理をそれらオブジェクト間のメッセージ送信でモデリングする

21

モジュールの凝集度（cohesion）

- モジュール構成図に現われる全モジュールについて、モジュールとしてのまとまりの良さを8段階に分類

悪い
① 偶発的：明確な理由なく恣意的にまとめる
② 論理的：見かけ上同一だが実際には異なる機能をまとめる
③ 時間的：実行する時間が近い処理をまとめる
④ 手続き的：一連の手続きをまとめる
⑤ 通信的：同一データを入力／出力する処理をまとめる
良い
⑥ 逐次的：パイプライン的な処理をまとめる
⑦ 機能的：明確に定義できる特定の機能のみをまとめる
⑧ 情報的：特定のデータへのアクセスをひとめとめにする

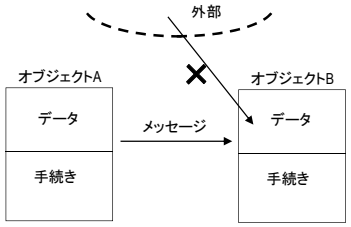
22

オブジェクト指向のキーポイント

- カプセル化：オブジェクトにデータと手続きをひとまとめにする
- 継承：オブジェクトのクラスには階層構造があり、下位のクラスは上位のクラスからデータと手続きを受け継ぐが、再定義も可能
- 情報隠蔽：オブジェクト外部からは直接そのオブジェクト内のデータにアクセスすることはできない
- メッセージ送信：オブジェクトへのメッセージ送信でオブジェクト内部のデータの操作を依頼し、それに基づいたオブジェクト間の相互作用でシステムの機能を実現する

23

オブジェクト間の相互作用



24

OOADの各種手法とUML

- OMT（Object Modeling Technique）法：James Runmbaugh
- Booch法：Grady Booch
- Coad/Yourdon法：Peter Coad & Edward Yourdon
- ・・・他多数

⇨ モデルの表記法を統一したものがUML
（方法論の統一ではない）

25

要求分析／定義と設計（Ⅲ）
― ERモデリング、OOAD、UML

✓開発プロセスにおける位置づけと目的
✓ERモデリング
✓オブジェクト指向分析／設計（OOAD）
⇨UML

26

UML（Unified Modelling Language）とは

- 1990年代に乱立していた主なOOAD方法論の概念と表記法を統一したもの
  - OMT法（James Runmbaugh）
  - Booch法（Grady Booch）
  - OOSE/Objectory法（Ivar Jacobson）
- 本当は、方法論そのものを統一したいが、それはほとんど不可能（誰しも自分の方法論に固執する）

⇨モデルの表記法が異なると、開発者がモデルの翻訳をしなければならず不都合なので、せめて表記法だけでも統一しようというのが狙い

- 現在はOMG（Object Management Group）標準であり、UML 2.xが登場している。

27

UMLダイアグラム（Ⅰ）

- システムの静的（構造的）側面
  - クラス図
  - オブジェクト図
  - パッケージ図
  - コンポーネント図
  - 配置図

28

UMLダイアグラム（Ⅱ）

- システムの動的側面
  - ユースケース図
  - シーケンス図
  - コラボレーション図
  - アクティビティ図
  - ステートチャート図

29

まとめ：ERモデリング

- システムに登場するデータの「静的な」関係や構造を明らかにする
- システムで扱うデータを、実体（エンティティ：Entity）と実体間の関係（リレーションシップ：Relationship）で表現する
- データベースの論理設計に適用される
- ER図
  - エンティティとエンティティ間の関係を表したダイアグラム
  - IDEF1X

30

まとめ:ジェクト指向分析／設計

- OOADと略称される
- システム分析／設計に対するデータ中心アプローチとプロセス中心アプローチを、オブジェクトという基本単位で統一的に扱う方法
  - 「オブジェクト」はモジュールとしての凝集度が◎
- 近年は分析設計～開発までオブジェクト指向でおこなうのが主流
- OOADの方法論多数 →UMLへ
- UMLの各種ダイアグラム（→次回以降説明）

31

参考文献

- 日経ソフトウェア（編）:「ゼロから学ぶソフトウェア設計」, 日経BP
- 鈴木正人 :「ソフトウェア工学 –プロセス・開発方法論・UML–」, サイエンス社

32