UMLの概要

- モデリングとモデル
- UMLとは
- UMLの主要モデル
  - UML1.4
  - UML2.1

モデリングとモデル

- モデリング
  - 実世界の事柄を別の物体で表現すること
  - モデルを作成すること
  - プログラミング
    - 処理をプログラム言語という手段で表現
      ↓
    - オブジェクト指向
      - データ構造をオブジェクトの属性
      - 処理を振る舞い

      としてモデリング
- モデル
  - ある視点から見たシステムの抽象的な表現
  - ダイアグラム（図）により表現
- モデリング言語
  - モデルを表現する手段

UMLとは

- UMLとは
  - Unified Modeling Language （統一モデリング言語）
    - ソフトウエア主体のシステムの成果物をビジュアル化、仕様化、構築、文書化するためのグラフィカルな言語
    - オブジェクト指向分析・設計を表現するためのモデリング言語（モデル表記法を統一したもの）
    - モデリング言語の国際標準（OMG標準、ISOでの標準化が進められている）

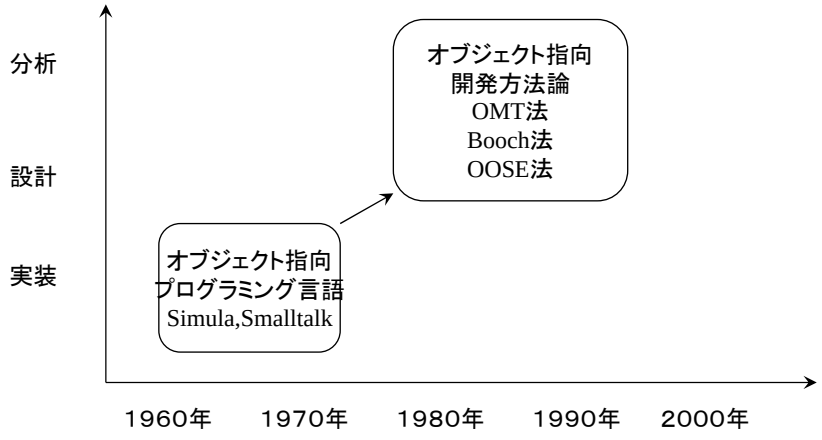
UMLとモデル

- UMLモデルは図で表現
  - 一つの図でモデルを表現できない。様々な観点（ビュー）から複数の図を記述することが必要。
- UMLは図を使って様々な要素を表現するために、それぞれの要素には図での書き方が決まっている。この書き方を表記法（Notation）という。たとえば、インターフェイスには、2つの表記法が存在する。

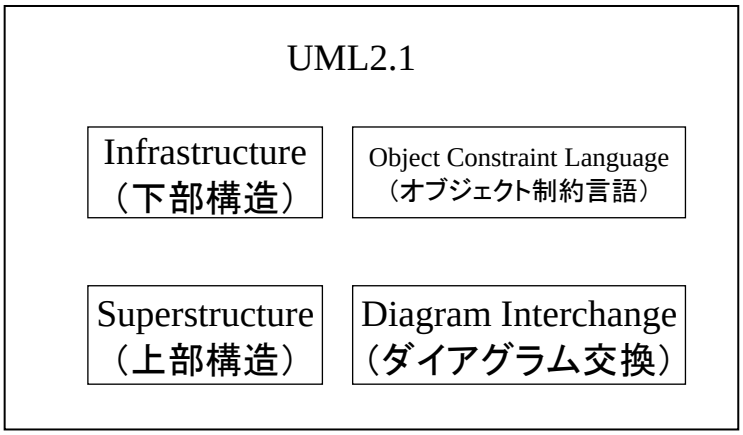
UMLと開発モデル

- UML
  - オブジェクト指向分析/設計に使用するダイアグラム（図）の描き方の文法
- 開発モデル（プロセス）
  - オブジェクト指向分析/設計をおこなうための方法論（システム開発の方法論）

オブジェクト指向開発方法論の流れ



UML2.1の構造



UML1.4のモデル（図）

- クラス図
- ユースケース図
- シーケンス図
- オブジェクト図
- コラボレーション図
- ステートチャート図
- アクティブティー図
- コンポーネント図
- 配置図

UML1.4の主要モデル

- 利用者から見たシステムの機能を表すモデル（機能モデル）
- 問題領域やシステムの静的構造を表すモデル（静的モデル）
- 振る舞いを表す動的モデル（動的モデル）
- 構成モデル（物理的なモデル）

UML1.4の９つのダイアグラム（１）

- 利用者から見たシステムの機能を表すモデル（機能モデル）
  - ユースケース図
  - アクティビティー図
- 問題領域やシステムの静的構造を表すモデル（静的モデル）
  - クラス図
  - オブジェクト図

UML1.4の9つのダイアグラム（2）

- 振る舞いを表す動的モデル（動的モデル）
  - ステートチャート図（状態遷移図）
  - シーケンス図
  - コラボレーション図（協調図）
- 構成モデル（物理的なモデル）
  - コンポーネント図
  - デプロイメント図（配置図）

UML2.1の図

- UML2.1のSuperStructureで定義されている13種類の図
  - 構造に注目してモデリングするための構造図
  - 振る舞いに注目してモデリングするための振る舞い図

UML2.1の構造図（1）

- モデリングする対象の構造着目にしてモデリングするための図
  - **クラス図**
  - **オブジェクト図**
  - パッケージ図（非公式）
  - コンポジット構造図（UML2.1）
  - **コンポーネント図**
  - **配置図**

UML2.1の振る舞い図（2）

- モデリングする対象の振る舞いにしてモデリングするための図
  - **ユースケース図**
  - **アクティビティ図**
  - **状態マシン図**
  - **シーケンス図**
  - **コミュニケーション図**（コラボレーション図 UML1.4）
  - 相互作用概要図（UML2.1）
  - タイミング図（UML 2.1）

クラス図

- 各クラス間の関係を表現することで、システムの静的な構造を表現
- クラス図は、クラス名、属性、メソッド（操作）という3つの部分から構成
- 一般に、長方形で表され、横線で3つの部分に分けられる
  - 属性
    - 「可視性　名前:型=デフォルト値」
  - メソッド
    - 「可視性 名前（引数1, 引数2, ….）:返り値」

アクセス指定（制御）

- 属性やメソッドの可視性を指定
  - +
    - public （どこからでも可視）
  - #
    - protected（パッケージ内および派生したクラスから可視）
  - -
    - private（クラス内でのみ可視）
  - ~
    - package（パッケージ内のみで可視）

継承

- クラス間の継承関係
  - サブクラスからスーパークラスに向かう白抜きの矢印で表現
- 継承の表現
  - 分割表記
  - 共有表記

継承

- 継承
  - 既存のクラスに属性や操作を追加して、新しいクラスを定義すること
  - 共通の属性と振る舞いを抽象化して、新しいクラスを作成できる
- スーパークラスとサブクラス
  - スーパークラス（親クラス）
    - あるクラスが継承によって定義された場合、その継承元のクラス
    - スーパークラスを継承したクラスをサブクラスという
  - サブクラス（子クラス）
    - あるクラスから、継承によって定義されたクラス
    - 継承元のクラスをスーパークラスという

汎化と特殊化

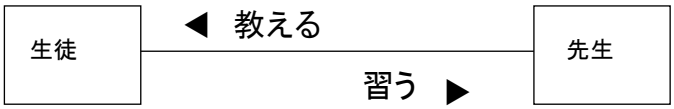
- 汎化
  - 複数のクラスの共通の特性を抽出してクラスを作ること
  - 汎化してできるクラスをスーパークラスとよぶ
  - 汎化の関係をis-a関係とよぶ
- 特殊化
  - ひとつのクラスを特性の違いに注目して分類すること
  - 分類されたクラスをもとのクラスのサブクラスという
  - 特殊化の関係をa-kind-of関係とよぶ

関連

- クラス間における関連
  - クラス間に結びつきがあることを表す
  - クラス間に直線を引いて表現
  - 関連名は線の近くに記述

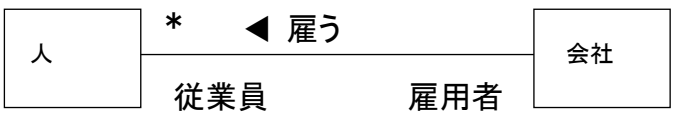
ネームディレクションアロー

- 関連名の横に塗りつぶした三角形を添えることで
  その関連の方向性を示す



ロール名

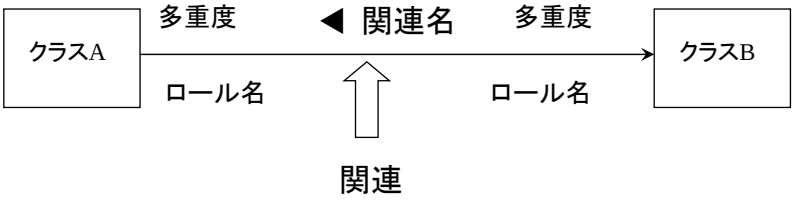
- 関連先のオブジェクトの役割を表す名前を、関連の端点に記述する。
- 多くは関連の役割の名前（属性名）などを利用する



多重度

- 関連するクラスのオブジェクト（インスタンス）数を表現
- 一方のクラスのオブジェクトとリンクすることができる
- 多重度の表現
  - ＊:　　0以上
  - 1..＊:　1以上
  - 0..1:　　0または1

関連のまとめ



集約

- クラス間の関係が「部分ー全体」であるという関連を表現
- has-aを表現
- 特殊な関連で、関連の「全体」側に白抜きの菱形を付けて表現

コンポジション

- 集約の一種
- 「全体が消滅した場合、部分も消滅する」という強い条件が該当する集約
- 「全体」を表すクラスの端に塗りつぶした菱形を付けて表現

コンポジション

- オブジェクトを他のオブジェクトで構成すること
  - コンピュータ: ビデオカード、キーボード、ディスクドライブなど
  - テレビ: チューナ、ブラウン管など
- コンポジション関係はhas-a関係とよばれる
  - オブジェクトに対して，メッセージを動かすように依頼するものをメッセージを呼ぶ

ユースケース図 （1）

- システムに必要とされる機能や要件を分析し、その結果を表すために利用
- 「システム」と「システムの利用者」の間のやり取りの様子をあらわす
- ユースケース図の構成要素
  - ユースケース
  - アクター
  - 関連

ユースケース図 （2）

- ユースケース
  - 外部から見たシステムの機能
  - システムの持つ内部的な機能ではない
  - 個々のユースケースが内部的にどのように実現されるかについては触れない
  - 内部にユースケース名を書き込んだ楕円形のシンボルとして表現
  - ユースケース群の周囲は、システム境界を表す四角形で囲んでおくことができ、このシステム境界は、ユースケース群を含むシステムを表す

ユースケース図 （3）

- アクター
  - システムの外部利用者を表す
  - 人であったり、システムと接続された別のシステムや、システムで制御されるハードウエアを表す
  - つまり、ユースケース図で表現するシステムを外部から利用するもの
  - 人型のアイコンや四角形の中にアクター名を書き、その上にステレオタイプ表記<<actor>>したもので表現
  - 役割を代表するものであり、個々の「実体」を表現するものではない

ユースケース図 （4）

- 関連
  - アクターがそのユースケースに関わっていることを示す
  - ユースケースという「機能」をそのアクターが「利用する」ことを表現
  - 関連

シーケンス図 （1）

- オブジェクト間におけるメッセージのやりとりを記述するに利用
- オブジェクト間のメッセージを時系列で表現（作業の進行表）
- どのようなオブジェクトがどのようなメッセージをどういう順序でやり取りしあうかをわかりやすくまとめることが可能
- シーケンス図の構成要素
  - オブジェクトシンボル
  - ライフライン
  - メッセージ
  - テキストシンボル

シーケンス図 （2）

- シーケンス図の構造
  - 水平と垂直の二次元の軸により表現
    - 水平軸
      - シーケンス図が表す場面に登場するオブジェクト
      - 各オブジェクトがやり取りするメッセージ
    - 垂直軸
      - 時間の経過を表す
      - 図の上方に並べられた各オブジェクトが生成されてから消滅するまでの時間の流れを表現
      - 時間は図の上から下に向かって進む

シーケンス図 （3）

- オブジェクトシンボル
  - シーケンス図に登場するオブジェクト群を上方に横一列に並べる
  - オブジェクト名とベースとなったクラス名を書き込む
- ライフライン（オブジェクト生存線）
  - 個々のオブジェクトから下方に向けて引かれた破線
  - オブジェクトのライフスパン(生存線）を表す
  - 破線が続いていることは、オブジェクトがシステム内に生存していることを表す
  - オブジェクトの消滅は、ライフラインの下端にターミネーションアイコン（×）を付けることで表現

シーケンス図 （4）

- メッセージ
  - あるライフラインから別のライフラインに向かって、ライフラインに垂直に引かれている矢印
    - 同期メッセージ
      - 閉じ実線矢印
    - 非同期メッセージ
      - 開き実線矢印
    - 同期メッセージへのリターン
      - 開き破線矢印
- テキストシンボル
  - メッセージの説明をおこなう
- 活性区間
  - ライフライン上でオブジェクトの活性区間を表現
  - オブジェクトが実際に活動している区間を表現
  - 矩形で表現

オブジェクト図 （1）

- クラス図
  - クラスの構造やクラス間の関連をあらわすためのダイアグラム
- オブジェクト図
  - クラス図に描かれたクラス群の実体のある時点における様子を表現したダイアグラム
  - クラスシンボルのインスタンス形であるオブジェクトシンボルを用いて表現

オブジェクト図 （2）

- オブジェクト
  - クラスの実体であり、インスタンスとも呼ばれる
- オブジェクトの表現
  - オブジェクトは長方形で表現される
  - 長方形の中には、オブジェクト名：クラス名の形式で記述される
  - クラス名やオブジェクト名は省略可能
  - クラス名だけを記述する場合には、“:”から記述する
  - オブジェクトの長方形を2つの区画に分けて、属性を表現できる
  - 属性は、属性名：型＝値の形式で表す

オブジェクト図 （3）

- リンク
  - クラスに対して実体（インスタンス）があるように、関連の実体をリンクという
  - 関連と同様に、実線で表す

コラボレーション図（コミュニケーション図）（1）

- システムにおけるオブジェクトの動的な振る舞いとオブジェクト間の関係を同時に表す
- シーケンス図
  - メッセージの時系列の順序を強調
- コラボレーション図
  - メッセージを送受信するオブジェクトの空間的な構造を強調
- 複雑な繰り返しや分岐、複数の並行制御フローの表現には、コラボレーション図のほうが適している

コラボレーション図（コミュニケーション図）（2）

- オブジェクト
  - クラスのインスタンスを表し、シーケンス図と同様に、長方形で表現する
- リンク
  - オブジェクト同士を意味的に関連付け、実線で表す
- メッセージ
  - オブジェクトからオブジェクトけの通信を表す
  - 同期メッセージは塗りつぶした三角形の矢印で表す
  - 非同期メッセージは通常の実線で表す
  - メッセージの情報は、メッセージラベルとして記述されるが、必ずシーケンス番号を記述する必要がある

ステートチャート図

- オブジェクトの生存期間中の状態の遷移と、その状態の遷移を引き起こすイベントや、状態遷移が発生した際のオブジェクトのアクションを表す
- つまり、オブジェクトがとりうる振る舞いを表現
- 状態
  - オブジェクトのある時点における状況
  - 初期状態、終了状態
- 状態遷移
  - オブジェクトがある状態から別の状態に変化すること
  - イベント
    - 変化を起こす出来事をイベントという
  - アクション
    - オブジェクトの遷移がおこる前に実行される

アクティビティー図

- さまざまな処理の流れを記述
- フローチャートによく似ている
- フローチャートとの違いは、並行処理が表現できる点にある

コンポーネント図

- システムにおける物理的なソフトウエア要素（コンポーネント）の関係をあらわす
- コンポーネントが公開しているインターフェイスや，各コンポーネント間の依存関係を表現
- コンポーネント間の依存関係は、破線の矢印で表す

配置図

- ソフトウエアが実行する際のハードウエア環境と、そこで活動をおこなうコンポーネントの割り当てを表現
- ハードウエアを総称して、ノードとよぶ
- ノードはシステム実行に依存する物理的要素で、コンポーネントはノード上に配置される