ソフトウェア工学

08：UMLモデリング（Ⅲ）— システムの動的側面（オブジェクト間の相互作用）のモデリング

理工学部 経営システム工学科
庄司　裕子

今回のテーマ

UMLモデリング（Ⅲ） — システムの動的側面（オブジェクト間の相互作用）のモデリング

- オブジェクトの相互作用
- シーケンス図
- コラボレーション図
- 相互作用図の比較

2

UMLダイアグラム

- 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
- 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
  - ステートチャート図
  - アクティビティ図

3

前回の復習：システムの静的（構造的）側面のモデリング

- クラスとオブジェクト
- クラス図
- クラス間の関係
- クラスの操作
- クラス図の拡張的概念
- パッケージによるクラス図の組織化

4

システムの動的側面（オブジェクト間の相互作用）のモデリング

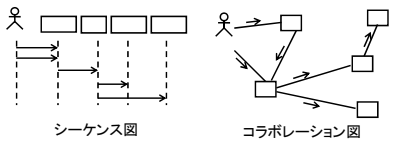
⇨オブジェクトの相互作用

- シーケンス図
- コラボレーション図
- 相互作用図の比較

5

相互作用図とは

- 相互作用図はシステムの動的な側面（オブジェクト間の相互作用）をモデリングする
- 相互作用はオブジェクト群とそれらオブジェクト間の関係（リンク／メッセージ）で構成される
- 相互作用図は2種類ある：



6

モデリング作業の流れと相互作用図の位置づけ

- ✓ ユースケースの全シナリオを定義した
- ✓ ユースケース モデルからクラスを発見した
- ✓ CRC カードを用いて、各クラスに責務を割り当て、これらのクラスを使ってユースケースのシナリオをどのように実現するかを分析(ユースケースを実現するためのオブジェクト間の相互作用を発見)した
- ⇨これらの相互作用の詳細を記述する方法が相互作用図で、以下の2種類がある
  - コラボレーション図
  - シーケンス図

7

相互作用図の説明に入る前に...

- クラスの責務
  - ☞ オブジェクトの振る舞いを規定する
- オブジェクトの相互作用
  - ☞ システムの機能(ユースケース)を実現する
- CRCカード
  - ☞ クラスの責務を識別する
  - ☞クラスの責務 →オブジェクトの振る舞い

8

クラスとクラスの責務

- クラスとは、クラスに属するオブジェクトの振る舞いを定義する責務(responsibility)の集合を具体的に表現したもの

言い換えれば

- 責務はクラスが果たすべき契約または義務である

- クラスの責務は、対応する操作と属性によって遂行される(つまり、モデルを詳細化する過程で操作と属性に変換される)
  - ☞ 責務と操作の対応は 1 対 1 とは限らない

9

オブジェクトの相互作用

- オブジェクトは、互いに協調して問題を解決できなければ意味がない
  - 各オブジェクトは振る舞いと状態を持つ
  - 単独ですべての責務を遂行できるオブジェクトは存在しない
- オブジェクト間の相互作用の方法
  - ☞ オブジェクトはメッセージを通じて相互作用する

10

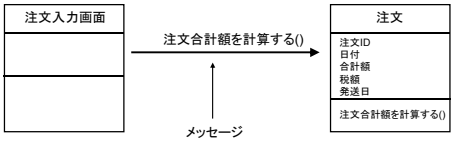
メッセージを通じたオブジェクト間の相互作用

- メッセージとは、オブジェクト間の通信の仕様であり、相手のオブジェクトに情報を伝え、その結果、アクティビティが発生することを想定したもの(Booch,「UMLユーザーガイド」より)
  - ⇨あるオブジェクトが別のオブジェクトに操作の実行を依頼すること

11

例:オブジェクト間の相互作用

- 「注文入力画面」には、注文の合計額を計算するための「注文」クラスが必要



「注文」クラスは注文合計額を計算する責務を担う

12

オブジェクト(クラス)の責務の公開

- 各操作は一意なシグニチャを持つ
  - 操作の名前、戻り値の型、および0個以上のパラメータを規定
- 操作には「可視性」という特性がある
  - 可視性は、操作が定義されているクラス以外のクラスから操作にアクセスできるかどうかを定義
    - (+)可視性が public: 他のクラスからアクセス可能
    - (-)可視性が private: 他のクラスからアクセス不可

13

CRCカード:クラスの責務を見つける方法

- CRCカードは、クラスの責務を見つけるための有効な技法(UMLの範囲外)
  - クラス(Class)、責務(Responsibility)、コラボレーション(Collaboration)をカード毎に整理する
  - CRC カードの概念は、1989 年の OOPSLA で Kent Beck と Ward Cunningham によって提唱された(http://c2.com/doc/oopsla89/paper.html)
- CRC カードは、設計者がオブジェクト中心の思考に集中するための助けとなる

14

CRC カードとは

- 4×6 インチの索引カードに次の内容を記載する
  - クラスの名前
  - クラスの責務
  - 責務に関係するコラボレータ(協調相手となるクラス)

クラス名:	コース
責務	コラボレータ
学生を追加する	学生
受講前提条件を知っている	
コースの開講時期を知っている	
コースの開講場所を知っている	

15

CRCカードによる責務の識別

- CRCカードの使用してクラスの責務を見つける手法の1つに、ロールプレイングがある
  - ただし、Kent と Ward が提唱したものではない
- 開発チームの各メンバーが1つ以上のクラスの役を演じる

16

CRC に基づくロールプレイング

1. ユースケース シナリオを選択する
   - ユースケースから 1 つのシナリオを選択する
2. 開発チームを 2 人ずつのチームに分ける
3. クラス名を記載したひとまとまりのカードを各チームに配布する
   - チームがクラスになる
4. アクターによってシナリオが開始される

17

CRC に基づくロールプレイング(続き)

5. 参加者がシナリオを実演する
   - ① あるオブジェクトから次のオブジェクトにボールが渡される
   - ② カードに責務とコラボレータを記入していく
   - ③ 次の場合、新しいカードを作成する
     - 1 つのオブジェクトの責務が多すぎるとき
     - プロセスの間に新しいクラスが見つかったとき
   - ④ シナリオが完結するまで続ける

18

CRC カードを使用すべき状況

- クラスの責務を識別するための補助が必要なとき
- プロセスのごく初期に、細部の設計に関して行き詰まったとき
- クラスの設計があまりにも混乱しているように思われるとき
- クラスが明確な定義を欠いているとき

19

CRC カードの利点

- コラボレーションのパターンを把握できる
- カードを物理的に並べ替えることによって、コラボレーションを表現できる
- カードの情報を基に、クラス間の汎化関係を特定できる
- オブジェクト指向技術に習熟していない開発チームに最も効果的である
  - 手続き的な設計に傾倒するのを防ぐ
  - 早まった汎化の定義を防ぐ
  - 「オブジェクト中心の思考」を促進する

20

システムの動的側面（オブジェクト間の相互作用）のモデリング

✓オブジェクトの相互作用

⇨シーケンス図

- コラボレーション図
- 相互作用図の比較

21

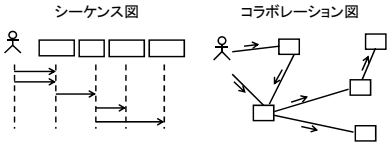
相互作用のモデリング

- このようにCRCカードによる分析を通じて、ユースケースを実現するためのオブジェクト間の相互作用が識別された
  - ユースケースを実現できることが確認された
- UMLでは、2種類の相互作用図を使って、オブジェクト間のこれらの相互作用を記述する
  - シーケンス図
  - コラボレーション図

22

相互作用図とは

- 相互作用図は相互作用を表現する
- 相互作用はオブジェクト群とそれらオブジェクト間の関係（リンク／メッセージ）で構成される
- 以下の2種類のダイアグラムがある



23

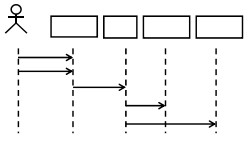
シーケンス図とは

シーケンス図はオブジェクト間の相互作用を

- 相互作用に参加するオブジェクト
- オブジェクト間でやり取りされるメッセージのシーケンス（時系列）

で表現する

⇨メッセージの発生順序を強調する



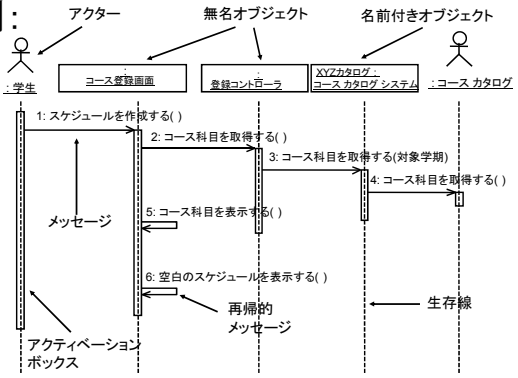
24

シーケンス図の描き方

- 縦の次元は時間軸で、上から下へ時間が進む
- 横の次元は、相互作用に参加する個々のオブジェクトを表す
  - 各オブジェクトは存在する間、「生存線（lifeline）」と呼ばれる縦の破線で表される
  - オブジェクトがアクティブである間（呼び出された操作が実行されている間）は、その生存線の上に細長い長方形（アクティベーション ボックス）を描く
- メッセージは、あるオブジェクトの生存線から別のオブジェクトの生存線へ向かう矢印で示さる
  - 矢印はダイアグラムの上から下へ時間順に並べられる

25

例：



26

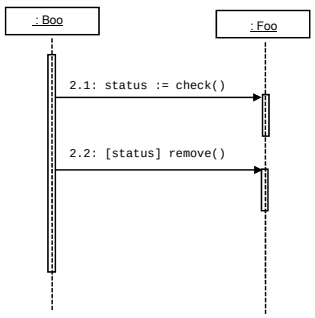
メッセージ ラベルに記載できる情報

- シーケンス番号
  - メッセージ呼び出しの順序を示す一意のドット区切り番号
  - シーケンス図では省略してもよく、その場合には矢印の物理的な位置が相対的順序を示すが、コラボレーション図では必須（他に順序を示す手段がないため）
- 制御情報 （省略可能）
  - 送信条件：[<条件句>]
  - 反復送信マーカー：*[<反復条件句>]
- <ローカル変数名>:= （省略可能）
  - メッセージの応答（戻り値）を変数に代入して、後で利用
- メッセージ名
  - 特別なケースとしてオブジェクトの生成／消滅がある
- (<引数リスト>) （省略可能）

27

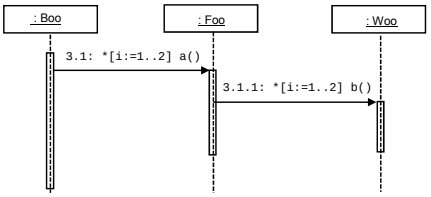
送信条件（ガード）付きメッセージ

1. メッセージの戻り値をローカル変数に代入
2. その変数値が `true`であれば、次のメッセージを送信



28

メッセージの反復送信

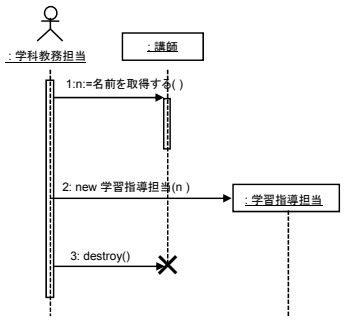


- 上記の例は、メッセージ シーケンス abbabb を表す

29

オブジェクトの動的生成／消滅、戻り値の使用

1. メッセージの戻り値をローカル変数に代入
2. 変数を使用してオブジェクトを生成
3. オブジェクトを消滅させる（生存線の最後に×印）



30

メッセージの種類

同期メッセージ	⟶	送信側は受信側の処理の間、実行をブロックされ、処理が終了すると実行を再開
非同期メッセージ	⟶	送信側は受信側の応答を待たずに、送信後も独立に動作し続ける
リターン メッセージ	<- - -	メッセージではなく、前に送信した同期メッセージからの戻りを表す。ブロックを解除
シンプル(フラット)メッセージ	⟶	制御が送信側から受信側に移り、次のメッセージはその受信側から送信される

31

シーケンス図の用途

- 主な用途は、ユースケースの実現(ユースケースの全体または一部の振る舞いを実行するためのオブジェクト間の相互作用)を示すこと
- フロー内のオブジェクトの役割を明確にし、クラスの責務およびインタフェースを決定するための基本的な情報源となるため、特に設計者にとって重要
- ユースケースに関係するオブジェクト間の相互作用は、1 つ以上のシーケンス図に図示できる
  - 一般的には、ユースケースの基本フローに対し1 つのシーケンス図を作成し、さらに、個別のサブフローごとにシーケンス図を1枚ずつ作成する

32

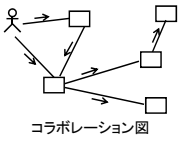
システムの動的側面(オブジェクト間の相互作用)のモデリング

- ✓オブジェクトの相互作用
- ✓シーケンス図
- ⇨コラボレーション図
- ■ 相互作用図の比較

33

コラボレーション図とは

- 相互作用に参加するオブジェクトの構成を強調する
- 以下を図示する
  - 相互作用に参加しているオブジェクト
  - オブジェクト間のリンク
  - オブジェクト間で受け渡しされるメッセージ



コラボレーション図

34

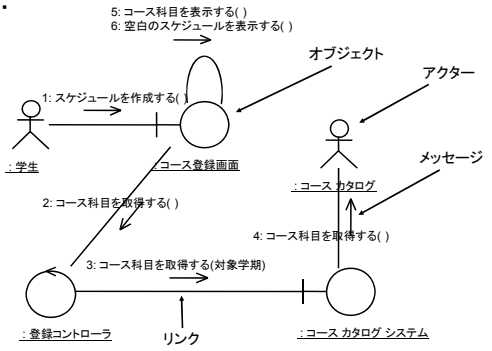
コラボレーション図の描き方

- 時間軸がない
- オブジェクト間の関連(すなわち、リンク)を実線で示し、それに沿ってメッセージ フローを矢印で示す

⇨オブジェクト図にメッセージ フローを追加した形

35

例:



36

コラボレーション図の要素：リンク

- リンクはオブジェクト間の関係で、メッセージを受け渡しするための経路
  - コラボレーション図では、リンクは 2 つのオブジェクト間の実線で表される
  - オブジェクトは他のオブジェクトとのリンクを通じて、リンク先のオブジェクトと相互作用したり、そのオブジェクトを参照したりする
  - リンクは関連のインスタンスとなることができる
  - リンクにはメッセージ フローが付加される

37

コラボレーション図の要素：メッセージ

- メッセージはオブジェクト間の情報伝達のことで、その結果、アクティビティが発生するものと期待される
  - メッセージはラベル付きの矢印で表され、リンクに沿って配置される（リンクを通じてターゲット オブジェクトに転送される）
  - メッセージを表す矢印は、ターゲット オブジェクト（メッセージ受信側オブジェクト）の方向を指す
  - メッセージを表す矢印には、シーケンス図と同じラベルと同じ種類の矢印を使用できる

38

システムの動的側面（オブジェクト間の相互作用）のモデリング

- ✓オブジェクトの相互作用
- ✓シーケンス図
- ✓コラボレーション図
- ⇨相互作用図の比較

39

シーケンス図とコラボレーション図の類似点

どちらも、

- 表現するセマンティクスは同じ
  - 情報を失うことなく、一方のダイアグラムをもう一方のダイアグラムに変換できる
  - 両者の相互変換機能を備えたUMLモデリング ツールも多い
- システムの動的な側面をモデリングする
- ユースケースのシナリオをモデリングする

40

シーケンス図とコラボレーション図の相違点

- コラボレーション図
  - 相互作用と共に関係（関連）を示す
  - コラボレーションのパターンを視覚化するのに向く
  - 特定のオブジェクトに及ぶすべての影響を視覚化するのに向く
  - ブレーンストーミング セッションでの利用が容易
- シーケンス図
  - メッセージの明示的なシーケンス（時系列）を示す
  - フロー全般を視覚化するのに向く
  - リアルタイム仕様や複雑なシナリオに適する

41

シーケンス図 v.s. コラボレーション図

- 用途に応じた使い分けが重要
- 相互変換機能を備えたモデリング ツールを使用すれば、両者の切り替えは容易

42

まとめ：UMLモデリング（動的側面）

- ■オブジェクトの相互作用
  - システムの機能（＝ユースケース）を実現するために、一連のオブジェクトが連携して働く（＝オブジェクトの相互作用）
  - 各オブジェクトの振る舞いはクラスの責務によって規定（それを見つける方法としてCRCカードがある）
  - オブジェクト間のメッセージ送信によって相互作用を実現
- ■シーケンス図
  - オブジェクト間の相互作用を時系列で表示
- ■コラボレーション図
  - オブジェクト同士がどのように相互作用するかを図示
  - 時間軸がない
- ■相互作用図の比較

43

参考文献

- Perdita Stevens with Rob Pooley, “Using UML: Software Engineering with Objects and Components, Updated Edition”, Pearson Education
  - （邦訳：「オブジェクト指向とコンポーネントによるソフトウェア工学 — UMLを使って —」ピアソン・エデュケーション）
- Martin Fowler, Kendall Scott,“UML Distilled: A Brief Guide to the Standard Object Modeling Language, 2nd Edition”, Addison-Wesley Pub Co.,
  - （邦訳：「UMLモデリングのエッセンス—標準オブジェクトモデリング言語入門 第2版」, 翔泳社）
- 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

44