UML

ゼミ担当者　：　篠原　翔，木田　直人，小畑　拓也，畠山　由貴
指導院生　：　大西　祥代，石田　裕幸
開催日　：　2007 年 5 月 24 日

1　はじめに

現在，大規模・複雑化するソフトウェア開発において，開発効率を向上させる技術としてオブジェクト指向が主流になっている．このオブジェクト指向に基づくソフトウェアをビジュアル化，仕様化するためのグラフィカル言語をモデリング言語と言う．1980 年頃から様々なモデリング言語が乱立していたが，1997 年に策定された UML（Unified Modeling Language）によりモデリング言語が標準化された．本ゼミでは UML の概要と代表的な UML のダイアグラムであるシーケンス図，クラス図について述べる．

2　UML

2.1　概要

UML(Unified Modeling Language) とは，オブジェクト指向によるソフトウェアをビジュアル化，仕様化するための統一モデリング言語である．UML は，実際には目には見えないシステムの構造や振る舞いを，目に見えるダイアグラムで表現する．UML には，目的や用途に応じた 13 種類のダイアグラムが用意されている．

2.2　必要性

現在，UML はソフトウェア開発において標準となっている．それは，以下のような場面で，複雑なシステムの構造を容易に理解できる UML が利用されることにより，システム開発を効率化できるためである．

- 考えの整理

  プログラムの規模が大きくなると，全体を捉えることが難しくなる．そこで，UML を用いて，プログラムの構造や動きを図で捉えることにより，自分の考えを整理できる．

- チーム開発におけるコミュニケーション

  大規模なシステムは，チームを組んで開発される．チームによる開発では，開発者同士の意思疎通が必要となる．そこで，UML を利用すれば，簡潔かつ視覚的にシステム設計の方針を示すことができる．

- ユーザや顧客との仕様検討

  システム開発では，ユーザや顧客の要望に沿ったシステムにする必要がある．UML ではシステムの仕様を図で表現できるため，開発者がユーザや顧客と意思疎通することが容易になる．

- システムの設計

  UML はシステム設計の設計図としても利用される．そのため，システムに要求される機能を分析する段階において，要求を分かりやすく図示し，分析を助けることができる．

2.3　各種ダイアグラム

UML の 13 種類のダイアグラムは，Fig. 1 のように，構造図と振る舞い図に分類される．構造図はシステムの静的な構造を表現したもので，それに対し，振る舞い図はシステムの動的な振る舞いを表現したものである．振る舞い図の中でも，特に時系列やオブジェクト間のメッセージのやり取りに着目し，表現したダイアグラムを相互作用図と呼ぶ．

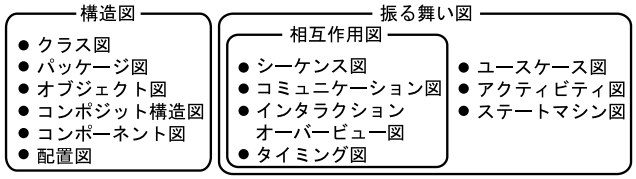

Fig. 1　各種ダイアグラム (出典：自作)

2.3.1　構造図

構造図に分類されるダイアグラムを以下に示す．

- クラス図

  クラス図は最も利用される構造図である．詳細は第 3 章で述べる．

- パッケージ図

  パッケージ図は，同じ役割を持つクラスを一つのパッケージにまとめて表現できる．そのため，モデルの管理が容易になる．Fig. 2 に，ある会社におい

て，社員用の給与システムをパッケージ図で表現した例を示す．

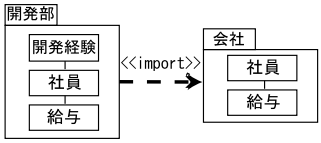

Fig. 2　パッケージ図の例 (参考文献 [1] より参照)

- オブジェクト図

　オブジェクト図は，複数のオブジェクト (インスタンス) から共通点を取り出し，クラス図を作成するためのダイアグラムとして利用される．Fig. 3 に，机とその上にある本，PC，キーボード，マウスをオブジェクト図で表現し，そこからクラス図を作成した例を示す．

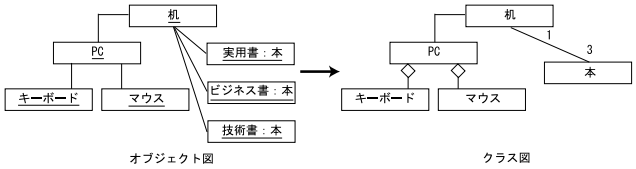

Fig. 3　オブジェクト図の例 (参考文献 [1] より参照)

- コンポジット構造図

　コンポジット構造図は，クラスの内部構造や外部との境界を表現する．クラスの枠の中にクラスを描くことにより，クラス同士の従属関係を明確にすることができる．Fig. 4 に，PC の内部構造をコンポジット構造図で表現した例を示す．

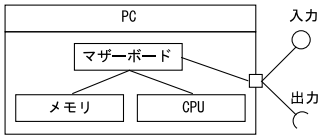

Fig. 4　コンポジット構造図の例 (参考文献 [1] より参照)

- コンポーネント図

　コンポーネント図は，コンポーネントと呼ばれるシステムの部品の構成を表現する．システム全体の構造を確認したい場合に利用される．Fig. 5 に，図書館の業務システムをコンポーネント図で表現した例を示す．

- 配置図

　配置図は，システムを構成するハードウェア上に，開発されたプログラムがどのように配置されているかを表現する．複数のマシンにプログラムが配置される場合に利用すると効果的である．Fig. 6 に，App.exe というプログラムを配置したサーバと PC を配置図で表現した例を示す．

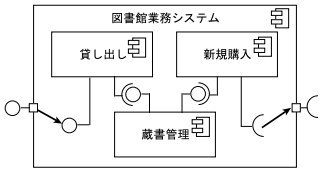

Fig. 5　コンポーネント図の例 (参考文献 [1] より参照)

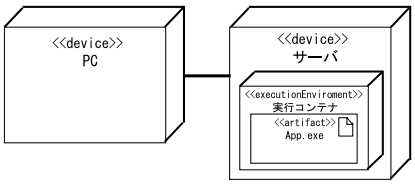

Fig. 6　配置図の例 (参考文献 [1] より参照)

2.3.2　振る舞い図

振る舞い図に分類されるダイアグラムを以下に示す．

- シーケンス図

　シーケンス図は，最も利用される振る舞い図である．詳細は第 4 章で述べる．

- コミュニケーション図

　シーケンス図が時系列的な視点で作成されるのに対し，コミュニケーション図は，オブジェクト同士の関連を重視し，結びつきの強さを明確に表現する．Fig. 7 に，旅行予約システムをコミュニケーション図で表現した例を示す．

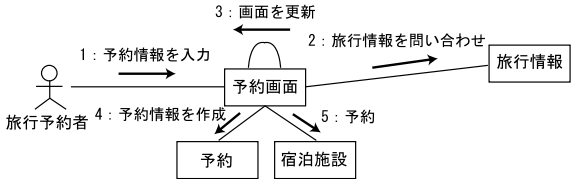

Fig. 7　コミュニケーション図の例 (参考文献 [1] より参照)

- インタラクションオーバービュー図

　インタラクションオーバービュー図は，シーケンス図やコミュニケーション図などの相互作用図同士の実行順序や，関連を表現する．シーケンス図などを分割して，大きくなりすぎることを防ぐことがで

きる．Fig. 8 に，録画予約の手順をインタラクションオーバービュー図で表現した例を示す．

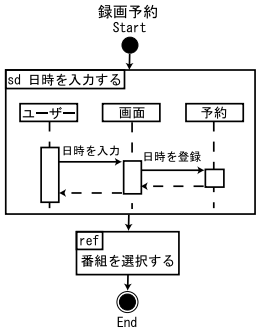

Fig. 8　インタラクションオーバービュー図の例 (参考文献 [1]) より参照)

- タイミング図

タイミング図は，オブジェクトが時間軸に沿ってどのような状態遷移を行うのかを表現する．状態遷移を細かな時間間隔で表現できるので，特に組み込み系ソフトウェア開発に効果的である．Fig. 9 に，信号機をタイミング図で表現した例を示す．

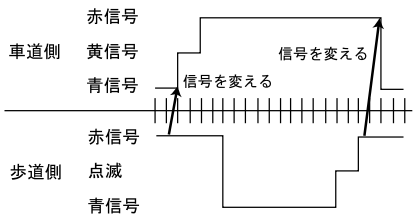

Fig. 9　タイミング図の例 (参考文献 [1]) より参照)

- ユースケース図

ユースケース図は，ユーザーの視点でシステムの機能を把握するのに役立つ．そのため，主に要求分析時に作成される．ユースケース図の作成には，システム外部の要素 (アクター) とシステム内部の要素 (ユースケース) の区分けを行うため，開発者が開発範囲を明確にできる．Fig. 10 に，銀行の ATM をユースケース図で表現した例を示す．

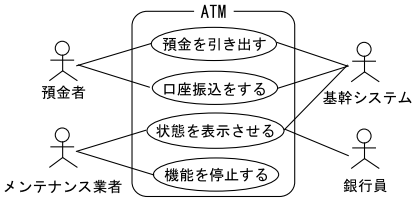

Fig. 10　ユースケース図の例 (参考文献 [1]) より参照)

- アクティビティ図

アクティビティ図は，処理の実行手順を表現する．フローチャートと近い記法になっているため，UML に詳しくないユーザに対しても，コミュニケーションを取りやすい．Fig. 11 に，お客がレストランで注文を取ってから，会計を済ませるまでをアクティビティ図で表現した例を示す．

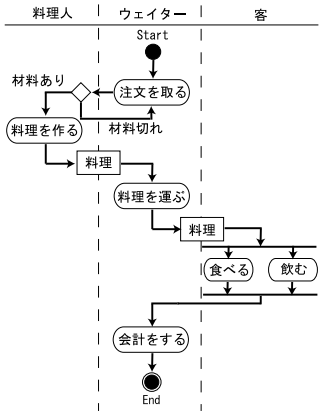

Fig. 11　アクティビティ図の例 (参考文献 [1]) より参照)

- ステートマシン図

ステートマシン図は，1 つのオブジェクトに着目し，そのオブジェクトの生成から破棄までの状態の移り変わりを明確にする．オブジェクトの状態によって，その振る舞いに違いがある場合に利用される．Fig. 12 に，ゼミの日程決定の手順をステートマシン図で表現した例を示す．

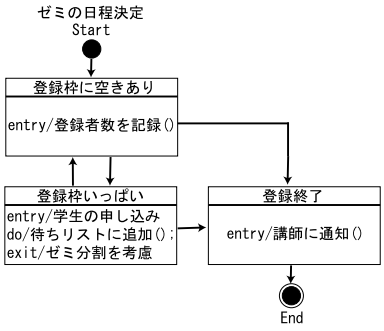

Fig. 12　ステートマシン図の例 (参考文献 [1]) より参照)

3　クラス図

クラス図とは，クラス，クラス間の関係から，システムの構造を記述する構造図である．一般的にオブジェクト指向によるシステムは，クラスを定義し，クラスを具象化して作成したオブジェクト同士が関連して働くことで動作するため，システムの構造を記述する上でクラス

に着目することは重要である．従って，クラス図は，分析，設計などシステム開発の様々な工程で利用される．

3.1　クラス

クラスは，オブジェクトを特定の基準によって分類し，共通要素を取り出して具象化したものである．クラスは属性と操作を持つ．

クラス図では，Fig. 13 に示すように，クラスを一般的に 3 つの区画に分けた長方形で表現する．各区画はクラス名，属性，操作に対応する．各区画に記述する内容を以下に示す．必要な場合，必要条件や制限など，他の区画を定義することも可能である．[1)]

- クラス名

  そのクラスの名前を記述する．

- 属性

  属性とは，そのクラスの持つ構造上の特性である．属性は，名前と型，初期値を持ち，値は数字や文字列で表される．属性は，一般的に以下の形式で記述する．

  [可視性][属性名]:[型]=[初期値]

- 操作

  操作とは，クラスの持つ動作上の特性である．操作は，操作名と可視性，パラメータと戻り値の型を持つ．操作は，一般的に以下の形式で記述する．

  [可視性][操作名]:[パラメータ]:[戻り値の型]

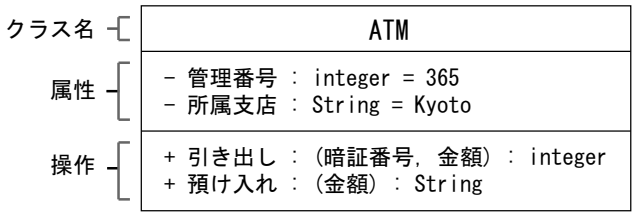

Fig. 13　クラス図におけるクラスの例 ([1)] より参照)

Fig. 13 の例では，クラス ATM の持つ属性として「管理番号」と「所属支店」，操作として「引き出し」と「預け入れ」が定義されている．

また，クラス図では可視性，static，abstract などの修飾子を表現することができる．それぞれの表現方法を以下に示す．

- 可視性

  可視性とは，該当するクラスの属性，操作に対するアクセス権限を表す．クラス図で用意されている可視性の記述を Table 1 に示す．

Table 1　クラス図の可視性表記

記述	意味
+	すべてのクラスからアクセス可能 (public)
-	自分のクラスからアクセス可能 (private)
#	自分と子クラスからアクセス可能 (protected)
～	所属パッケージ内からアクセス可能 (package)

- static

  static とは，そのクラスのすべてのオブジェクトから共有される属性や操作に付与する修飾子である．属性や操作が static である場合には，以下のように名前に下線をつけて表す．

  – 属性名
  – 操作名

- abstract

  abstract とは，抽象クラスに付与する修飾子である．クラスや操作が abstract である場合には，名前を斜体にする．

ただし，クラス図では，場合に応じて重要でない項目を省略して表記することができる．クラス名だけのクラス図や，詳細なクラス図など，システム設計の工程に応じて使い分けが可能である．

3.2　クラス間の関係

オブジェクト指向のクラス間には関連，集約，汎化，インタフェースといった関係が存在する．それぞれのクラス図での表記方法を以下に示す．

3.2.1　関連

関連とは，クラス同士の関係である．クラスを表す長方形を実線でつなぐことで関連を表す．2 つのクラス間の関連を，2 項関連という．関連は自分自身にも引くことができるが，このような関連を再帰関連という．3 つ以上の関連を表現することもできる．この場合，実線は菱形に接続される．これを N 項関連という．クラス図における関連の表記を Fig. 14 に示す．

Fig. 14 ではアイスとコーン，およびトッピングの例で関連を説明している．アイスが 2 個以上ある場合，自身と関連する，再帰的な関連となる．

関連には多重度を指定することができる．多重度とは，関連に参加するクラスのインスタンス同士の数量的な制約のことである．多重度は数字と「*」,「..」という記号で表すことができる．関連の多重度表記を Table 2 に示す．

多重度は，その関連において，一方のインスタンス 1 つに対し，もう一方のインスタンスがいくつ存在するの

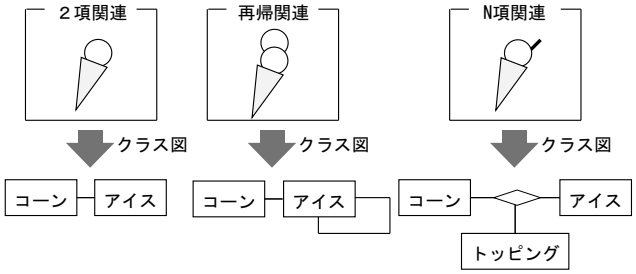

Fig. 14　クラス図における関連の表記 (出典：自作)

Table 2　関連の多重度表記

多重度	意味	該当する数
1	1 のみ	1
*	0 以上	0,1,2,3,4,5...
1..*	1 以上	1,2,3,4,5...
2..5	範囲指定 (2 から 5 まで)	2,3,4,5

かを表す．関連を表す実線の両端に，Table 2 に示す記号を記述することで多重度を表現する．Fig. 15 では，大学ひとつに対し 1 人以上の学生が存在し，学生 1 人につき 1 つの学生 ID があることが示されている．このようにクラス図で多重度を記述した場合，対応するインスタンスはその多重度分しか存在できないことを表現する．

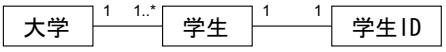

Fig. 15　クラス図における多重度の表記 ([1] より参照)

3.2.2　集約

集約は，関連の一種で，1 つのクラスが複数の別のクラスで構成されていたり，別のクラスを保持しているというような「全体と部分」の関係にある場合に用いる．集約は Fig. 16 のように白い菱形を集約する側のクラスの線の端に記述する事で表現する．

PC にはマウスやキーボードなどの外部機器を接続することができるので，Fig. 16 では，PC を「全体」，外部機器を「部分」とみなし表記している．集約を”has-a”関係と呼ぶこともある．

集約において,「集約する側のクラスが消滅すると，それに含まれている部分的なクラスが存在できない」といったように,「全体と部分」の関係がより強い場合，その関係をコンポジションという．コンポジションは Fig. 17 のように黒い菱形を集約する側のクラスの線の端に記述する事で表現する．

Fig. 17 では集約する側の会社クラスが消滅すると，そこに含まれている総務部クラスと開発部クラスはそれ以

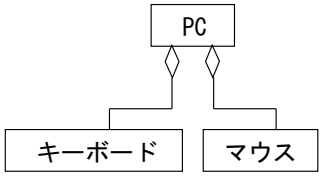

Fig. 16　クラス図における集約の表記 ([1] より参照)

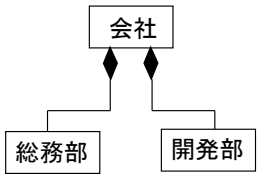

Fig. 17　クラス図におけるコンポジションの表記 ([1] より参照)

上存在できない．

3.2.3　汎化

汎化とは，あるクラスを抽象化して定義したクラスと元のクラスの関係を表す．汎化とは逆に，あるクラスを具象化して別のクラスを定義することもできる．このような関係を特化という．特化によって作成されたクラスは，一般的なクラスの属性や操作を引き継ぐことができる．特化の対象となる抽象的なクラスを「スーパークラス」（または親クラス），具象化したクラスを「サブクラス」（または子クラス）と呼ぶ．UML の表記上では，特化も汎化の表記を用いて表す．汎化の関係はスーパークラス側の線の端に白抜きの三角形を描くことで表現する．汎化，特化の関係は，継承，”is-a”の関係とも呼ばれる．

Fig. 18 では，人間クラスが学生，会社員クラスを汎化したものであることを表現している．学生クラスには学生 ID の属性しか表記されていないが，実際はスーパークラスである人間クラスの名前，性別の属性も持っている．操作に関しても同様である．

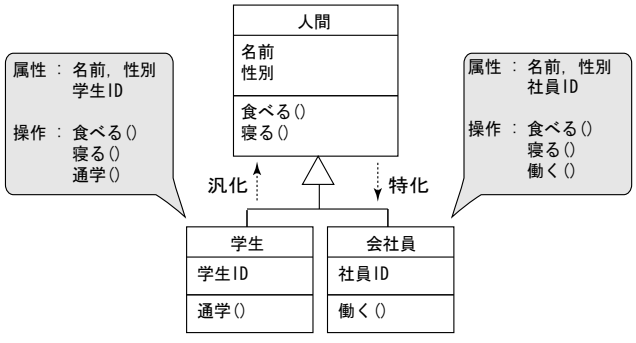

Fig. 18　クラス図における汎化の表記 ([1] より参照)

3.2.4　インタフェース

インタフェースはクラスが外部に公開する操作の型を宣言したものである．インタフェースを実装したクラスは，そのインタフェースで定義される操作を実装しなければならない．

インタフェースを表すには，Fig. 19 のように，クラスと同じように長方形の領域を区切り，名前と操作をそれぞれの区画に記述する．名前の上には《interface》と記述する．また，あるクラスがインタフェースを実装していることを表現するには，該当するクラスとインタフェースを破線で繋ぎ，インタフェース側に白抜きの三角形を付加する．インタフェースの詳細を記述する必要がない場合は，白丸に棒をつけた簡易的な表記とすることも可能である．このとき，インタフェースで定義された操作を利用する側は，ボールを受けるような鉤爪の形に棒をつけて表記する．

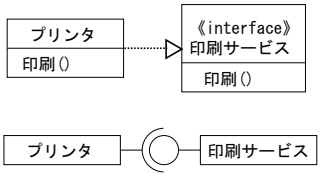

Fig. 19　クラス図におけるインタフェースの表記 ([1] より参照)

3.3　クラス図の例

例として，研究室周辺の環境を考える．研究室 1 つに対し，学生が複数名所属するため，学生クラスと研究室クラスの関係は多対 1 の関係がある．学生の中には具体的に，学部学生，修士課程学生，博士課程学生があるため，これら 3 つのクラスは学生クラスと継承関係にある．また，研究室備品クラスは研究室クラスの一部であるため，この 2 つのクラスは集約の関係にあるといえる．以上から，研究室周辺の環境は，クラス図で Fig. 20 のように示すことができる．

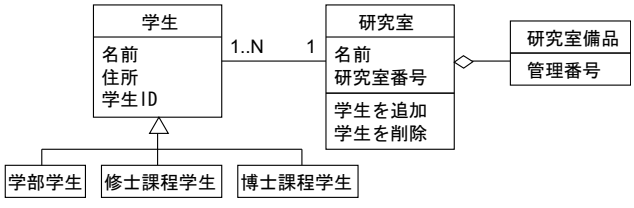

Fig. 20　クラス図の例 (出典：自作)

4　シーケンス図

シーケンス図は，アクターやオブジェクト間の通信を，時間の流れを基準として表現した振る舞い図である．シーケンス図を利用することで，オブジェクト同士の相互作用，各オブジェクトの動作を明確に表現できる．シーケンス図では，図の垂直方向が時間軸上に対応し，図の上部から下部へとたどることで時間の流れに沿った振る舞いを表現できる．また，通信の主体であるアクターやオブジェクトは横軸に配置され，これらの通信は横軸に平行な矢印で表現する．シーケンス図の例を Fig. 21 に示す．

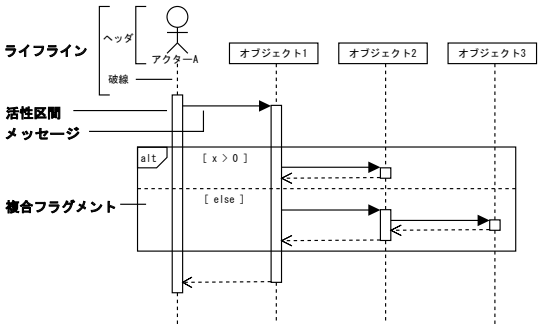

Fig. 21　シーケンス図の例 (参考文献 [1] より参照)

4.1　ライフライン

通信の主体であるアクターとオブジェクトはそれぞれ，人型の図，長方形（長方形内には「オブジェクト名:クラス名」を記述する．オブジェクト名とクラス名のいずれかは省略されることがある.）で表現され，これらをヘッダと呼ぶ．シーケンス図では，ヘッダを横軸に配置し，各ヘッダから下方に破線を引く．この破線上の上部から時系列に沿って動作を記述することにより，時間の流れを基準に振る舞いを表現できる．ヘッダと破線を併せて，ライフラインと呼ぶ．

4.2　活性区間

アクターやオブジェクトが操作を行っている区間を活性区間と呼び，ライフラインの破線上に白抜きの長方形を描いて表記する．アクターやオブジェクト間の通信はこの活性区間で行われる．

4.3　メッセージ

アクターやオブジェクト間の通信は，メッセージの送受信で行われ，横軸に平行な矢印で表現する．また，一般的にメッセージ上にラベルを記述し，メッセージの概要を表記する．メッセージの送受信は，オブジェクトが活性区間中に行われる．

4.3.1　同期と非同期

メッセージは，同期と非同期のものが存在する．これらの概要と表記法を以下に示す．

- 同期メッセージ

  同期メッセージは，そのメッセージが起動した操作から応答がない限り次の操作ができないメッセージである．同期メッセージは，実線と黒塗り三角形の矢印で表現する．同期メッセージのラベルは以下の形式で記述する．

  [操作名]（[引数 1],[引数 2],…,[引数 n]）

  また，同期メッセージによる操作が終了したことを同期メッセージの戻りといい，破線と開き矢印（→）で表記する．同期メッセージの戻りのラベルは以下の形式で記述する．

  [戻り値の型]=[操作名](_):戻り値

  同期メッセージと戻りの記述方法を Fig. 22 に示す．

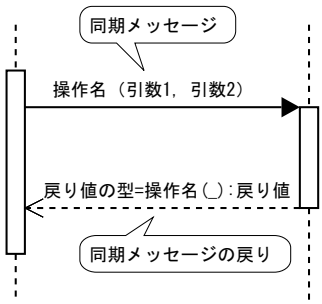

Fig. 22　同期メッセージと戻りメッセージの記述方法 (参考文献 [1] より参照)

- 非同期メッセージ

  非同期メッセージは，そのメッセージが起動した操作の応答がなくても次の操作ができるメッセージで，並列処理などに相当する．ラベルの形式は同期メッセージと同様である．非同期メッセージは，実線と開き矢印（→）で表記する．非同期メッセージの記述方法を Fig. 23 に示す．

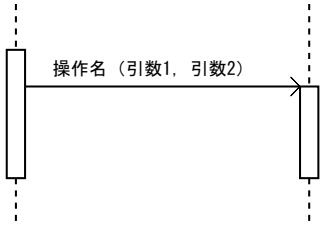

Fig. 23　非同期メッセージの記述方法 (参考文献 [1] より参照)

4.3.2　特殊なメッセージ

特殊な意味を持つメッセージを以下に示す．

- 生成メッセージ
  生成メッセージは，オブジェクトを生成する際のメッセージである．生成されるオブジェクトに向けて破線の矢印を引くことで表記する．生成メッセージには同期のものと非同期のものがある．ただし，同期の場合は黒塗り矢印，非同期の場合は開き矢印（→）を用いて記述する．ラベルの記述方法は一般的な同期・非同期メッセージと同様である．生成メッセージの記述方法を Fig. 24 に示す．

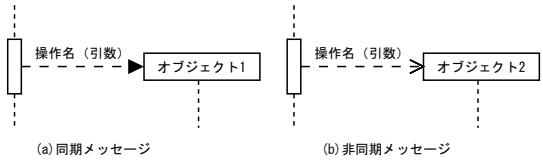

Fig. 24　生成メッセージの記述方法 (参考文献 [1] より参照)

- 再帰呼び出し
  再帰呼び出しは，自分自身の操作を呼び出すためのメッセージである．矢印を自分自身に向けて記述し，活性区間を重ねて記述する．再帰呼び出しの記述方法を Fig. 25 に示す．

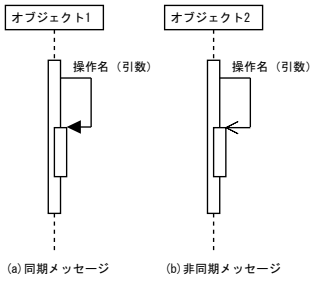

Fig. 25　再帰呼び出しの記述方法 (参考文献 [1] より参照)

4.4　複合フラグメント

シーケンス図では条件分岐や繰り返し処理を記述することができる．その際に利用するのが，複合フラグメントである．

4.4.1　条件分岐

通信が条件分岐する場合，複合フラグメントの 1 つであるオルタネイティブ (alternative) を使用する．alternative では，分岐する処理全体を枠で囲み，枠の左上の欄に「alt」と表記する．そして，その枠を破線で分割し，条件文によってどの領域の処理を行うかを判断する．条件文は分割したフレームの最上段に記述する．alternative の記述方法を Fig. 26 に示す.

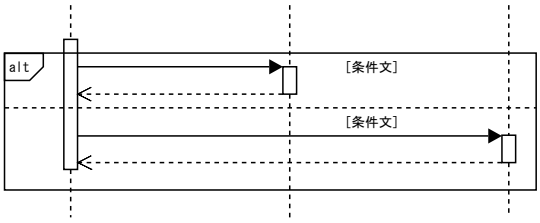

Fig. 26　alternative の記述方法 (参考文献 [1] より参照)

4.4.2　繰り返し

通信を繰り返し実行する場合，複合フラグメントの 1 つであるループ (loop) を使用する．対象となる範囲を枠で囲み，枠の左上の欄に

loop minint, maxint, [条件文]

と記述する．minint，maxint はそれぞれ最小繰り返し回数，最大繰り返し回数を示す．繰り返しが終了するケースは 2 種類ある．minint を超えて条件文が成立しなくなるケースと，繰り返し回数が maxint を超えるケースである．(条件文が成立していても，繰り返し回数が maxint を超えると終了する．)loop の記述方法を Fig. 27 に示す.

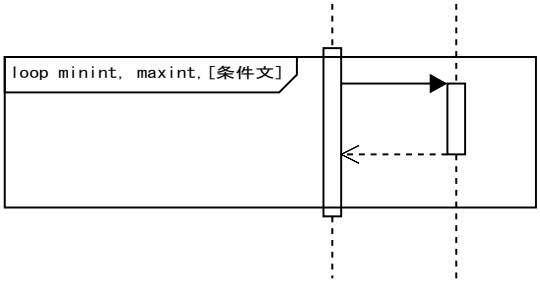

Fig. 27　loop の記述方法 (参考文献 [1] より参照)

4.5　シーケンス図による具体例

以下の手順で，預金者が ATM でお金を引き出すシステムをシーケンス図で表現したものを Fig. 28 に示す.

1. アクター「預金者」は,「画面に触れる ()」という同期メッセージを，オブジェクト「ATM」に送る
2. アクター「預金者」は，オブジェクト「ATM」からの「String=画面に触れる (_):メニュー」という戻りメッセージを受け取る
3. アクター「預金者」は，金額を引数とした「預金を引き出す (金額)」という同期メッセージをオブジェクト「ATM」に送る
4. オブジェクト「ATM」は,「残高を得る ()」という同期メッセージを，オブジェクト「口座」に送る
5. オブジェクト「ATM」は，オブジェクト「口座」からの「int=残高を得る (_):残高」という戻りメッセージを受け取る
6. アクター「預金者」は，オブジェクト「ATM」からの「int=預金を引き出す (_):金額」という戻りメッセージを受け取る
7. アクター「預金者」は，オブジェクト「ATM」からの「String=預金を引き出す (_):エラー」という戻りメッセージを受け取る

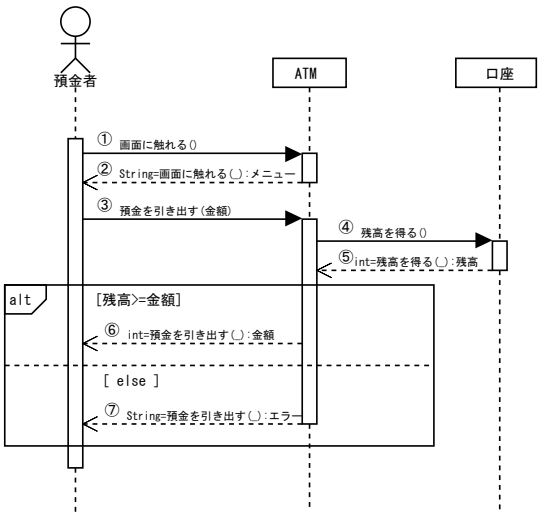

Fig. 28　預金の流れ (参考文献 [1] より参照)

付録

A　EclipseUML

UML を設計書として作成する場合は，ツールで記述するのが一般的である．代表的なツールに Rational Rose, Microsoft Visio for Visual Studio.NET などが挙げられる．これらはモデリングツールと呼ばれ，UML の記述のみでなく，記述された UML からのソースの生成機能，IDE(統合開発環境) との連携機能を備えている．ここでは，統合開発環境として広く使われている Eclipse 上のプラグインとして有名な EclipseUML を例に，使い方を説明する．

EclipseUML は，Omondo 社が開発している UML の記述，ソース生成を実行するためのツールである．EclipseUML はインストーラつきの JAR ファイルで提供されている．機能が制限された Free 版と，有料版（約 $2435.398-）の Studio がある．Free 版は，クラス図については生成機能を含めてフルサポートされるが，シーケンス図などの生成はサポートされていない．ただし，単に記述するだけであれば Free 版で作成できる．モデリングや UML の練習のみなら Free 版でも十分であるが，Free 版は他にも UML2.0 の機能が使えない，他の人とプロジェクトの共有ができない，CVS が使えない等の機能制約が大きい．以下に EclipseUML のインストール手順を示す.

1. 既に旧版の EclipseUML をインストール済みの場合は，まずアンインストーラ（/eclipse/Uninstaller/eclipseuml-uninstaller.jar）でアンインストールを行う.

2. ダウンロードページ (http://www.eclipseuml.com/download/index.html) から EclipseUML 2.0.0.20050411 Studio をダウンロードする.

3. eclipseUML_E302_studioEdition_2.0.0.20050411.jar をダブルクリックすると，インストーラ起動画面 (Fig. 29) が表示される.

Fig. 29　Omondo インストーラ起動画面 ([2] より参照)

4. インストーラが起動するので，指示に従ってインストール項目を選択する.

   (a) ライセンス利用許諾に同意

   (b) インストールパスを選択

   (c) インストールパッケージを選択

5. インストール開始

   インストール開始画面 (Fig. 30) が表示され，インストールが開始される.

6. インストール完了

これで，EclipseUML を利用する準備が整った．Eclipse を再起動すると，ツールバーに，UML Diagram のアイコン (Fig. 31) が追加される.

Fig. 30　インストール開始画面 ([2] より参照)

Fig. 31　UML Diagram のアイコン ([2] より参照)

B　クラス図作成手順

クラス図作成の手順を Fig. 32〜Fig. 37 に示す．Fig. 32 に示すように，TaxCalcApp.java ソースコード中の「TaxCalcApp」を選択し，右クリックする.「Open UML」→「Class diagram editor」を選択する.

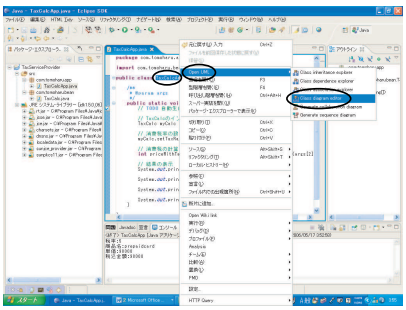

Fig. 32　クラス図作成開始 (出典：自作)

Fig. 33 に示すように,「Dependancy」にチェックし,「OK」ボタンを押す.

Fig. 34 に示すように，TaxServiceProvidor の左横にチェックを入れて「終了」ボタンを押す.

Fig. 35 に示すように,「OK」ボタンを押す.

Fig. 36 に示すように，TaxCalcApp クラスと TaxCalc クラスの内容と関係（クラス図）が表示される．クラスの配置が見にくい場合，移動させたいクラスをドラッグし，見やすい位置に移動させる.

Fig. 37 に示すように，クラス図の余白を右クリックし,「Export」→「JPEG」を選択すると，クラス図を JPEG 形式の画像ファイルとして保存できる.

C　シーケンス図作成手順

Fig. 38 に示すように，ソースコード中の「main」を選択し，右クリックする.「Open UML」→「Generatesequence diagram」を選択する.

Fig. 39 の画面の,「Sequence Diagram Preferences」ボタンを押す.

Fig. 40 の画面の「Auto generate return arrow」をチェックし,「OK」ボタンを押す.

Fig. 41 で「OK」ボタンを押す.

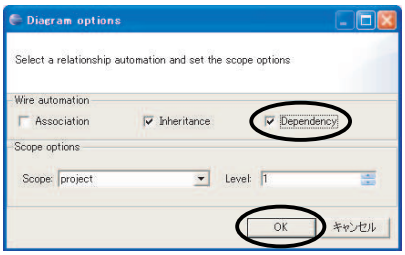

Fig. 33　ダイアグラムオプション画面 (出典：自作)

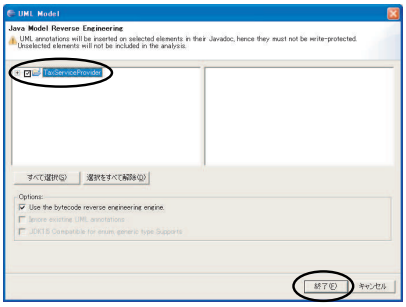

Fig. 34　モデル選択画面 (出典：自作)

Fig. 35　パッケージ選択画面 (出典：自作)

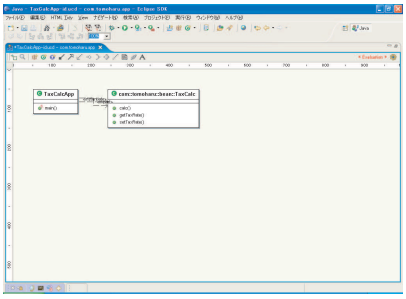

Fig. 36　クラス図表示画面 (出典：自作)

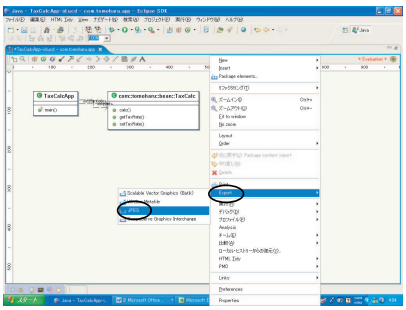

Fig. 37　クラス図の画像ファイル保存 (出典：自作)

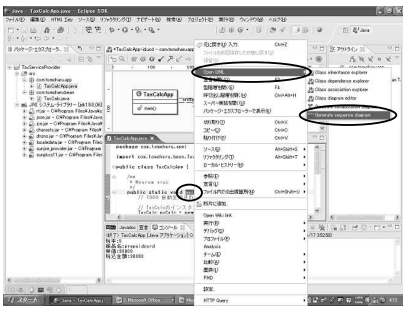

Fig. 38　シーケンス図作成開始 (出典:自作)

Fig. 39　(出典:自作)

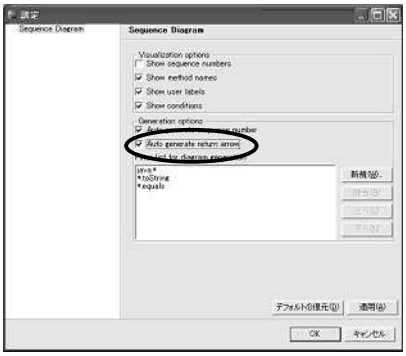

Fig. 40　(出典:自作)

Fig. 41　(出典:自作)

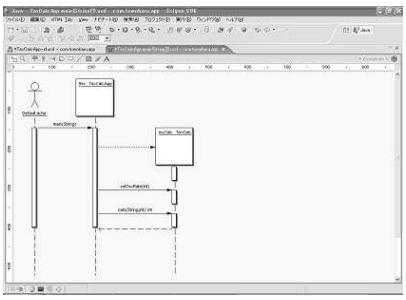

Fig. 42　(出典:自作)

Fig. 42 のように，シーケンス図が生成され，表示される．

シーケンス図は JPEG 形式の画像ファイルとして保存が可能である．Fig. 43 のように，シーケンス図の余白部分を右クリックし,「Export」→「JPEG」を選択すると，JPEG ファイルが作成できる．

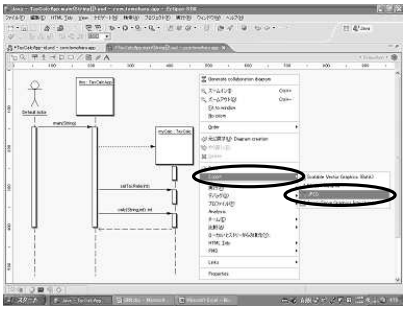

Fig. 43　(出典:自作)

参考文献

1) 独習 UML 第 3 版

株式会社 テクノロジックアート著，長瀬 嘉秀・橋本 大輔 監修

2) 浅煎り珈琲,

http://www.nextindex.net/java/UML/EclipseUML.html