基般

オブジェクト指向設計ワンポイントアドバイス

江見圭司
京都情報大学院大学

筆者はこれまで，組込みシステム技術協会主催ETソフトウェアデザインロボットコンテスト（通称ETロボコン）でオブジェクト指向設計の教育とモデル図の審査を行ってきた．その経験と実績を踏まえて，オブジェクト指向のプログラミングの経験はあるがオブジェクト指向設計の初心者である方に向けて，間違いやすいポイントをクラス図，シーケンス図，ユースケース図などをもとに解説する．

ETロボコンとモデリング

ETロボコン[1]では，組込み技術のための「分析・設計モデリング開発」をメインの目的としている．競技内容は，本稿に深く関係するデベロッパー部門では，レゴのマインドストームを使った走行体を使い，規定の組み立てで同一バッテリを使い，ソフトウェアの違いだけによる競技を行う．指定されたコースや課題を走行体が速く正確に走行するシステムを開発する．本大会前にUML（Unified Modeling Language）などで記述されたシステムの分析・設計モデル（以下，単に「モデル図」という）を提出する．このモデル図の事前審査結果と本大会の走行競技結果で総合評価を行うのである．

なぜ，モデル化が必要なのか？

ETロボコンでは本大会の数カ月前に各地区で2日間の技術教育会を実施している．そこで使用している「ETロボコン公式トレーニング　モデリング入門」教材で述べられている事柄を中心に組込み技術業界でのモデル化の必要性について述べることにする．

❑ 要素技術主導でかつ現物合わせがなぜ起こるのか?

携帯電話，デジタル家電，自動車，複合機，産業用装置などへの組込み技術では自然現象や物理現象を相手に「人間では容易にできないことを装置によって実現する」ためのソフトウェア開発が必要である．環境制約，物理的制約，ハードウェア制約，コスト制約など現場で試行錯誤する中で構築される「要素技術」が重要となり，机上だけでは予測できない問題が多い．ビジネス系のシステム開発と異なり，考慮するべき制約が多いため，モデル化は容易ではなく，必然的に，要素技術主導でかつ現物合わせで作り込む開発スタイルが主流となる．つまり，プログラミングして動かしながら試行錯誤的に開発を行い，「最終的に動いたもの」が仕様となり，それを量産のソフトウェアとして出荷することが常態化しているのである．このような現物合わせや試行錯誤は，煩雑＆複雑なロジックを生み出す．属人的にコーディングしているため意味が分からないロジックになる傾向があり，設計図がなくソースコードだけなので，どこに何が書いてあるのか把握できなくなるのである．加えて，ハードウェア要素技術の進歩に伴う陳腐化に対応できなくなる．長期間の使用に耐える頑健さ，性能向上のために要素技術の試行錯誤などを想定する必要がある．

以上の観点から従来の開発手法つまり「モデル化しない開発」で追いつかなくなってきているのである．そこで，「属人的でないソフトウェアの開発」が求められてきた．

❑「属人的でないソフトウェア開発」を行うための大きな 2 つのポイント

大きくて複雑な問題をうまく扱いたいということから 2 つのアプローチが考えられる.

- 大きな問題を小さい問題に分割
- 複雑な問題から興味がある問題を抽出

⑴部品化＝大きな問題を小さい問題に分割

大きな問題を分割する. 手分けをした仕事を組み合わせて, 大きな仕事をこなすのである.

⑵抽象化＝複雑な問題から興味がある問題を抽出

興味のあるところを抽出する (抽象) し, 興味がない部分を削る(捨象)ことがポイントである.

部品化と抽象化をすることが「モデル化」である. これを記述するために, ET ロボコンでは前述の UML を用いている.

❑ モデル化と抽象・捨象

物理学を専攻した方なら, モデル化するということは「何かを捨てる」ことが当たり前なのであるが, ほかの分野を専攻した方は「何かを捨てる」ことが共通認識になっていないと思うので補足解説をしたい.

たとえば, 鉄道の路線図を考えてみよう. 路線図のモデル化では実際の地図上の路線から, 駅間距離や方向を捨象して駅と駅とつながりだけのモデル化された路線図をつかう. 興味のある「駅と駅のつながり」を抽出し, 興味がない部分である「駅間距離, 線路の方向」を捨象する.

高等学校の理科で扱う原子模型について考えてみよう. 図 -1 では 2 つのモデルを表している. 左は量子力学の知見に基づいた現実の原子モデルであるが, 実際は太陽系モデルでも問題なく化学現象を説明することができる. 要するに, 興味のあるところを抽出する(抽象)し, 興味がない部分を削る(捨象)ことが重要である.

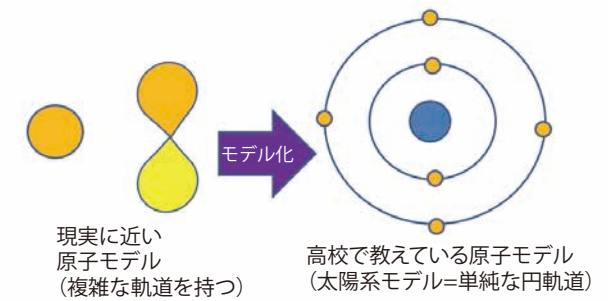

図 -1　原子モデル. 左は量子力学の知見に基づいた現実の原子モデルであるが, 実際は太陽系モデルでも問題なく初歩的な化学現象を説明することができる

あまり知られていないことではあるが, 地動説と天動説について述べたい. 実はこの違いはどちらが正しいかではなく, 基準座標の取り方に由来するもので, モデル化の違いにすぎず, どちらが抽象的かということでもない.

オブジェクト指向の解説では「樹木をモデル化しよう」という例題があるが不適切である. 興味がある部分が人によって異なる. それで, 幹と枝をモデル化する人, 幹と葉をモデル化する人, 実をつける人などさまざまなモデル化があり得る. これは興味がある部分を規定しないとモデル化できないのである. 少し話がそれて恐縮ではあるが, 心理学では「樹木画テスト」で書いた人の関心ごとを調べることが実施されているほどであり,「樹木」には客観性は存在しない. モデル化により抽象・捨象を定義しないと, 属人性の高いコーディングができあがるような現状を感じてほしい.

UML

ジェームズ・ランボー (James Rumbaugh) のオブジェクトモデル化技法 (OMT) とグラディ・ブーチ (Grady Booch) の Booch 法, そしてイヴァー・ヤコブソン (Ivar Hjalmar Jacobson) が加わり, 多様なモデリング言語から統一モデリング言語 (UML) を開発した. ET ロボコンでは, 以下のような 3 つの観点でモデル化を実現し UML で記述することを推奨している.

開発する目的や実現したいこと(要求や機能)
…主としてユースケース図
構成する要素とそのつながり(構造)
…主としてクラス図
処理の流れやできごとと処理のかかわり (振舞い)
…主としてシーケンス図

❑ クラス図

まずは，構成する要素とそのつながりを表すクラス図のアドバイスをしよう．「モデル化と抽象･捨象」で述べたように，部品化と抽象化に直結するのが型としてのクラスであり，実態としてのオブジェクト（インスタンス）である．図 -2 のような例題を使って，クラスの抽象化をした継承関係（is-a）などを説明している．ライブラリが存在するオブジェクト指向プログラミング言語では，継承関係のクラスは最初から存在するが，設計においては，継承関係は開発者が作っていくことになる．コンポジット（part-of）や関連も重要である．初心者は，他のクラスから孤立しているクラスが存在することに不安を覚えるようで，初心者はクラス間にむやみに関連を持たせる傾向があり，あとでプログラミングするときに問題になる．孤立クラスは間違いではないので注意してほしい．

「継承」と「コンポジット」はどうやって決めるのかという質問があるが，これは設計者の設計思想によるものであり，客観的な解は存在しない．

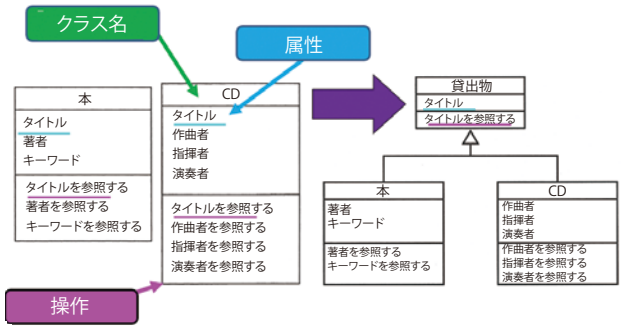

図 -2　UML 図の基本であるクラス．左側の「本」クラスと「CD」クラスの共通部分から右側のような「貸出物」クラスを生成する．情報処理技術者試験ソフトウェア開発技術者試験平成 13 年から引用

❑ シーケンス図

処理の流れやできごとと処理のかかわりのあるシーケンス図で注意するべき点は 2 つある．

1 つ目は「クラス図のクラス」と「シーケンス図のクラス」を対応させる必要がある．図 -3 はカラーの枠線で書いている．この図ではシーケンス図の「顧客」クラスがクラス図には存在しない．ここでは，本来，アクターとするべきであるが，場合によっては別のシステムのクラスを使う場合もある．モデル図を作成するにあたって，シーケンス図にだけ存在するクラスがないように最初にチェックするべきことである．シーケンス図を作成しながら，設計上どうしても必要なクラスをつくるときには，クラス図にも反映することを忘れないでほしい．

2 つ目の注意点は，図 -4 に表している．シーケンス図の書き方で，とても問題となる．この簡単な原理を守れば，UML 描画ツールで楽しく間違いなくシーケンス図を描くことができるようになる．

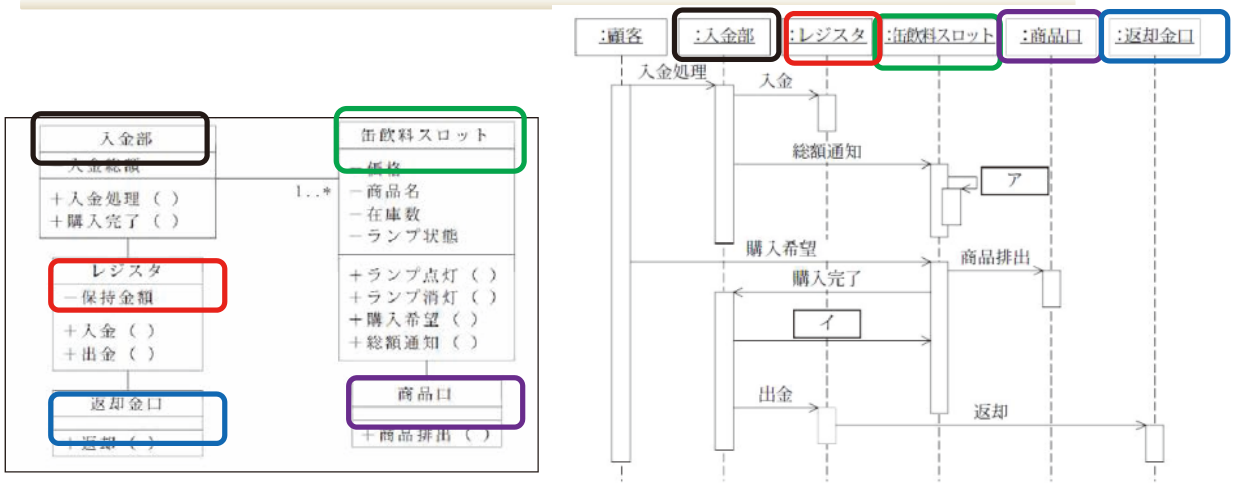

図 -3　クラス図とシーケンス図の関係．情報処理技術者試験・ソフトウェア開発技術者試験平成 16 年から引用

❑ ユースケース図

開発する目的や実現したいこと（要求や機能）を表すためにユースケース図を考えてみよう．英語の授業では，「いわゆる第 3 文型：S（主語）V（動詞）O（目的語）」を教えるが，実はオブジェクト指向でも SVO がある．O はオブジェクトすなわちインスタンス，V はメソッドである．S はコンピュータに命令する人物あるいは人工知能になる．この S がユースケース図の「アクター」なのである．

アクターに対して，ユースケースを書くのであるが，これが難しい．ET ロボコンではマインドマップを用いて，設計のためのアイディアをまとめることを推奨している，マインドマップからユースケース記述をするのである．ユースケース記述を具体化するとシナリオになる．ET ロボコンではロボットを走行させるので，むしろシナリオを先に書いてからユースケース記述などを行うこともあり得るが，これは悪い設計手法ではない（図 -5）．

UML の解説書はユースケース図から書くことを推奨しているが，素人にはユースケース図は一番難しいのである．初めのうちはユースケース図は書けなくても気にしなくてもいい．

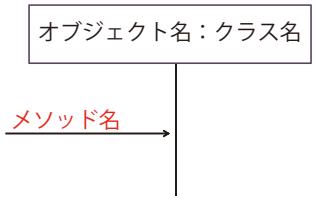

図 -4　シーケンス図のメッセージ（図で赤字）は矢印の先のクラスのメソッドであることを理解すれば作成はやさしい．また上部の四角内には「コロンのあとにクラス名」を書くことに注意すればいい

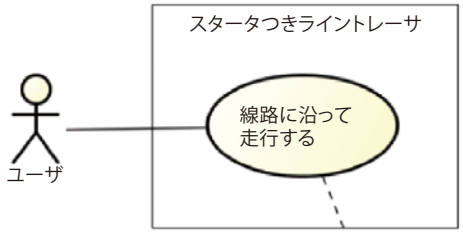

図 -5　ET ロボコンでの最も単純なユースケース図

❑ UML 図の各図の相関

最後に UML 図の各図の相関を図 -6 にまとめた．

型と実態（具体化）に分けている

機能＝ユースケース図が型で，シナリオで実態化する

構造＝クラス図が型で，オブジェクト図で実態化する

振舞い＝シーケンス図・アクティビティ図という実態が先にあり，ステートマシン図が型である．

通常の設計手順では，ユースケース図からはじめる．

さて，ユースケース図を書いた後は，ソフトウェアを使用するときのシナリオを日本語で書いてみるのがよい．このシナリオを基にシーケンス図を作ってみるべきであるが，先ほど述べたように，クラスがないのにシーケンス図を書くと矛盾が発生する．

一方，ユースケース図からクラス図を作るのはかなり難しい．ロバストネス図[2]などの手法が提案されているので参考にしてほしい．そして，クラス図とシーケンス図は相互に連携しながら設計していくことになる．本稿では詳しく述べなかったが，ステートマシン図も学んでほしい．

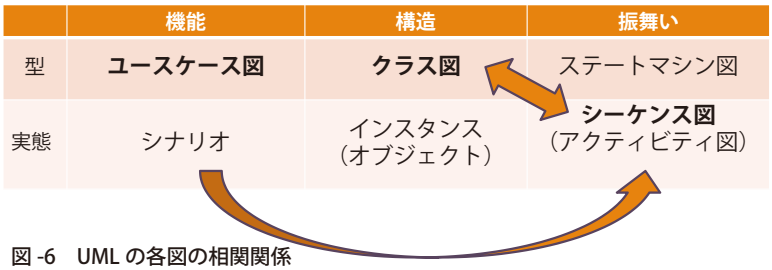

図 -6　UML の各図の相関関係

UML 図からプログラミング言語

❑ プログラミング言語への対応

UML 図からソースコードや実行コードをある程度生成できるツールもある．

UML 図から C++ や Java などのオブジェクト指向プログラミング言語への変換は，それほど問題ではない．ET ロボコンで最も問題になるのは，C（以下「C 言語」という）への変換である[3]．組込み業界では，C 言語はまだまだ現役である．ET ロボコン

参加者から筆者が直接聞いた話であるが，エンジニアがオブジェクト指向設計に対応できないというよりは，C++ コンパイラは C コンパイラよりも遥かにライセンス料が高く導入できないという事情もあるようである．

❑ プログラミング言語中心の UML 図

図 -7 の右側は各プログラミング言語とオブジェクト指向設計（UML 図，OO 設計）の対応を書いている．左側は初心者が陥る典型的なクラスの例である．クラスだけが存在して，属性も操作も存在しないもので，これは設計ではなく，明確な間違いである．次によくあるのが，クラスに操作が 1 つだけ存在する場合である．これはあまり推奨はされないが，初心者の場合は，こういうクラスを乱発したモデル図を書いてくる．モデル図を審査する側から見れば，「C 言語の関数をクラスとして記述した」ことが一目瞭然である．実際に，モデル図を提出した方に聞き取り調査を行うとほぼ全員が「C 言語のソースコードを UML 図にしました」と答えている．ソースコードから設計図を後追いで作成するというのは本末転倒であるが，手続き型言語である C 言語で開発しているのに，オブジェクト指向設計手法で設計することに無理があるともいえる．ネット記事 [3] にあるように UML 図から C 言語への変換にはある程度のノウハウがあるので，本稿の読者も参考にしてほしい．

最後に，たとえば UML 図でデリゲートを表現できても，たとえオブジェクト指向言語の Java や C++ では実装できなかったりするので，必ずしも完全ではない．このようにオブジェクト指向設計はや UML 図はプログラミングするという観点では完璧ではない．

UML 図を書いてフィードバックをもらいたい方は，ぜひとも ET ロボコンに参加してみることをお勧めする．オブジェクト指向設計が普及することを願って，本稿を終えたい．

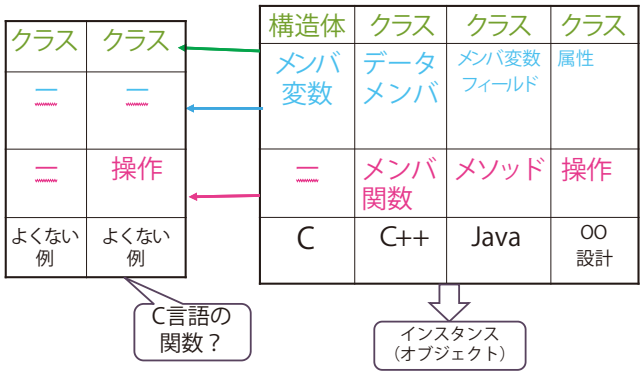

図 -7　オブジェクト指向設計（OO 設計）と Java，C++，C 言語との名称の対応．左側は，ET ロボコン参加者が作成するよろしくないクラス図の例

参考文献

1) ET ロボコン概要【ET ロボコン 2017 公式サイト】，http://www.etrobo.jp/2017/gaiyou/intro.php（2018 年 1 月 1 日閲覧）
2) 古川正寿：独習オブジェクト指向開発　第 2 版，翔泳社 (2010).
3) ET ロボコン 2006 へと続く道（3），UML モデルをどうやって C 言語に落とし込むか，http://monoist.atmarkit.co.jp/mn/articles/0606/30/news128.html（2018 年 1 月 1 日閲覧）

（2018 年 1 月 9 日受付）

江見圭司（正会員）　k_emi@kcg.ac.jp

2001 年金沢工業大学工学部情報工学科講師，2006 年京都情報大学院大学准教授（現職），2008 年 ET ロボコン関西地区実行委員長．