

## オブジェクト指向ソフトウェアの設計教育-上流工程の過程と UML 表記-

Education for mastering object-oriented software design

-Sub-processes in the upstream process and the UML notation-

浅田 善久<sup>†</sup>堀井 健<sup>‡</sup>小谷 賢太郎<sup>‡</sup>

Yoshihisa Asada

Ken Horii

Kentaro Kotani

## 1 序論

教育現場でソフトウェア設計の経験がない学生に、オブジェクト指向によるソフトウェア設計を行わせると、次のような3つの点で問題がある。

第1に、ソフトウェアにどのような機能をもたせるのかという問題がある。実社会において、ソフトウェアにもたせる機能の決定は、ソフトウェアに関する業務知識をもつ依頼者と開発者の間でソフトウェアについて互いに同一の理解をもつことにより、その問題を解消している。一方、教育現場では、業務知識をもつ依頼者からのニーズを必ずしも得ることが出来るとはいえない。従って、設計者自身でソフトウェアにもたせる機能を考えなければいけないという問題が生じる。

第2に、オブジェクト指向によるソフトウェア設計の方法論では、モデルの記述が厳密には定義されておらず、その記述の意味も曖昧である。すなわち、記述されたモデルの一貫性が必ずしも保障されないという問題がある。教育現場でも、この問題により、モデルの入出力情報が次のモデルにどのような影響を与えるのかということを決める手段が用意されていない。

第3に、クラス図の作成が困難であるという問題がある。オブジェクト指向によるソフトウェアを開発する上で、クラス図の作成が重要な役割を果たしている。クラス図を作成するには、設計対象の十分な知識や、深い洞察力、設計対象を抽象化する能力が必要となる。ところが、オブジェクト指向による方法論では、どの段階の情報をクラスにするのかという明確な基準は設けられていない。従って、教育現場で設計経験のない学生にクラス図を作成させるためには、設計対象をクラス単位に分類できる具体的な方法が必要である。

これらのことを踏まえ、本研究では上記の問題を考慮した要求分析からクラス図の作成までの方法論を報告する。

## 2 ソフトウェア設計教育の問題点に対する解決案

本研究では3つの問題に対して、どのような解決策を提案したのかを2-1~2-3で述べる。

### 2-1 ソフトウェアに具備する機能について

今回提案する方法論では、既存のWebシステムや書籍、業務資料をもとにソフトウェアが具備すべき機能を抽出する方法を取る。それにより、ユーザニーズを得ることのできない環境であっても、設計者は設計対象とするソフトウェアにもたせる機能を考えることが可能となる。

### 2-2 モデル間の整合性の問題について

モデル間の関連に整合性を持たせるために、構造化技法を用いる。構造化技法は手続きを重視した方法論である。ゆえに、モデル間に一貫性をもたせることが出来る。そこ

で、本研究では、構造化技法をオブジェクト指向方法論に導入することにより、この問題を解決した。展開における表記法については、UMLを用いた。

### 2-3 クラス図の作成の問題について

クラスとは、共通する性質や特徴に着目して、情報を構造化したものである。オブジェクト指向の設計方法論では、設計対象とするソフトウェアの構造を把握する方法が不足している。そこで、本研究では、ソフトウェアの構造を把握する方法として、3要素スキーマ技法を用いた。3要素スキーマ技法は、要素(情報)をグループ単位でまとめ、階層図で表す一連の方法である。

階層図の下位層は上位層の情報に対して、具体的な性質や特徴を表す情報が記述されており、上位層と下位層の関係は、クラスとクラスの属性の関係に該当する。このことより、階層図の構造とクラスの構造は同質として扱える。

## 3 方法論の概要

要求分析からクラス図の作成を行うまでの手順は以下に示す6つの手順からなる。以下に各手順の概要を示す。

### [手順1 設計対象の設定]

設計対象とするソフトウェアを決定する。設計対象を設定するために以下のどちらかの方法を取る。

- ・ 学生自身が設計対象のソフトウェアを設定する
- ・ 指導者が設計対象のソフトウェアを設定する

### [手順2 機能分析・制約分析]

ソフトウェアにもたせる機能を資料から抽出する。また、その機能を達成するために必要な処理条件を資料(Webシステム、書籍、業務資料)を参考に行う。

[機能分析]

(ステップ1)

資料は表1に挙げるようにWebシステム、書籍、業務資料などを選択する。そして、それらの資料から設計対象とするソフトウェアの概要を記述する。

(ステップ2)

各資料に関して表1に示す抽出方法を適用し、機能となるものを「~を~する」という形式で列挙する。そして、列挙した各機能の概要を記述する。各機能名、概要文は表2の様式に従い機能、概要の項目に記述する。

(ステップ3)

設計対象とするソフトウェアの機能を把握する際には機能を大きく捉えることにより把握しやすい。そのため、列挙した機能で同じ目的を示しているものを1つの機能として統一する。

[制約分析]

(ステップ)

機能分析で求められた各機能を達成する際に、設定する情報の条件(処理条件)や事前に行っていないとその機能を実行することが出来ない条件(事前条件)を記述する。各

<sup>†</sup>: 関西大学大学院 工学研究科, Graduate School of Engineering, Kansai University

<sup>‡</sup>: 関西大学 工学部, Faculty of Engineering, Kansai University

資料に関して、表 3 に示す方法をもとに抽出を行う。抽出した結果は、表 2 の制約条件の項目に記述する。

表 1 機能の抽出方法

調査資料	機能候補抽出方法
Webシステム	Webシステムを利用し、Webシステムが行う処理や画面上にある用語からWebシステムの機能となる候補を列挙する
書籍・業務資料	書籍・業務資料に書かれているソフトウェアの特色や目的を機能候補として列挙する

表 2 機能分析・制約分析の記述形式

機能	制約条件	概要
を××する	事前に必要な機能内容記述	概要記述

表 3 制約条件の抽出

調査資料	制約条件の抽出方法
Webシステム	抽出した機能を実際に操作し、処理条件や事前条件をWebシステムの画面から抽出する。
書籍(業務資料)	抽出した機能に関して、書籍や業務資料の特色や目的などから処理条件や事前条件を抽出する。

### [手順 3 アクタとの関係分析]

設計対象であるソフトウェアと人、あるいは外部システムとの関係を明確に区別する。

(ステップ 1)

表 2 の機能の項目で記述した各ユースケースを使用するアクタ(外部システムやソフトウェア利用者が該当する)を表 2 から抽出、もしくは各ユースケースに該当すると思われるアクタを設計者自身で想定する。想定する場合はサービスを提供する外部システム、サービスを提供する人(運業者)、またはサービスを利用する人(カスタマー)という視点から具体的なアクタ名の記述を行う。

(ステップ 2)

ユースケース間の関係を記述する。ユースケース間の関係では「汎化」「包含」「拡張」という関係項目がある。汎化関係は、機能分析のステップ 3 で行った統一関係が該当する。包含、拡張の関係については業務資料から該当する関係を調べる。記述形式はユースケース図の記述形式に従い、作成する。

### [手順 4 コンテキストダイアグラムの作成]

ソフトウェアとアクタの入出力関係を明確にし、ソフトウェアの業務全体の構成を明確にする。

各ユースケースを実行した際に発生するソフトウェアと人との間の入出力情報をコンテキストダイアグラムの記述形式に従い、記述する。

(ステップ 1)

ユースケース図の各ユースケースを実行した時に設計者が入力する情報とソフトウェアが出力する情報を記述する。各資料に関して、表 4 に示す抽出方法を適用する。記述形式は表 5 の様式に従う。表 5 のアクタ名とユースケース名は手順 3 で作成したユースケース図から抽出する。

(ステップ 2)

ステップ 1 で抽出した入出力情報をグループにまとめ、そのグループの総称を設計者が決定し(または資料から抽出し)、グループ名を命名する。そして、そのグループ名を構成する入出力情報を記述する。入出力情報でグループ化できないものはその情報をグループ名とする。記述形式は表 6 の様式に従う。そして、ステップ 1 で作成した表 5 の入力情報、出力情報をグループ名に書き換え、表に示す。

(ステップ 3)

ステップ 1、2 で抽出した情報からコンテキストダイア

グラムを作成する。コンテキストダイアグラムの設計対象とするシステム名の項目には、設計対象とする資料のソフトウェア名を用いる。この情報と表 5 の入力情報、出力情報のグループ名に書き換えた表の結果をコンテキストダイアグラムの記述形式に従い、記述する。

表 4 入出力情報の抽出方法

調査資料	抽出方法
Webシステム	各ユースケースを実際に行い、初期画面で入力した情報を入力情報、ユースケース実行後に得た情報を出力情報とする。
書籍(業務資料)	各ユースケースを実行する際にどのような情報が必要となるのかを調べ、入力と出力に該当する情報が当てはまる。

表 5 入出力情報の記述形式

アクタ名	ユースケース名	入力情報	出力情報
アクタ名の記述	ユースケースを記述	入力情報を記述	出力情報を記述

表 6 入出力情報のグループ化

グループ名	入出力情報
グループ名を記述	グループ名を構成する入出力情報を記述する

### [手順 5 アクティビティ図の作成]

アクタとソフトウェアの処理過程を具体的に記述する。ユースケースごとに設計対象のソフトウェアとアクタとの動的振る舞い(以後、イベントフローとする)を記述する。イベントフローは一般に、入力情報からある変換によって出力情報を出力することであり、「入力、変換、出力」という情報処理に相当する。アクティビティ図の作成には「入力、変換、出力」といった情報処理に対応させて記述を行う。また、イベントフローの記述は「A の B を C する」という形式で記述を行う。

(ステップ)

「入力、変換、出力」の処理名を入力処理、問題処理、出力処理とする。入出力処理に関しては、実際にソフトウェアを操作した処理内容や出力された出力結果を記述する。問題処理については、アクタが入力した情報に対して、ソフトウェアがどのような内部処理を行うのかという視点で発想する。記述形式はアクティビティ図に従い、記述する。

### [手順 6 クラス図の作成]

アクティビティ図で記述したイベントフロー「A の B を C する」という形式で表した各処理を、3 要素スキーマ技法を用いて、クラス図の作成を行う。

## 4 まとめ

本研究では、教育現場でソフトウェア設計教育を行う時に生じる問題点についてまとめ、その問題点を解決した方法論を提案した。

今後は、この方法論を実際に学生に適用することと、ソフトウェア開発に向けた方法論とするために、UML モデルとプログラム開発の工程の関係について検討し、プログラム作成までの方法論について検討していきたいと考えている。

[文献は省略する]