

# ソフトウェア設計論

—役に立つUMLモデリングへ向けて—

博士(情報科学) 松浦 佐江子 著

コロナ社

# まえがき

ソフトウェア開発とはなにか、考えてみたことはありますか？

コンピュータは私たちの生活のあらゆる場面に登場し、ソフトウェアを工夫することで私たちの暮らしを便利にかつ豊かにするサービスを提供しています。こうした役に立つ、使いやすいサービスを提供するためには、どのようなソフトウェアを創れば、どんな場面でどのように役立つのかを明らかにする必要があります。そして、明らかにしたことをコンピュータが実行できるように、コンピュータが理解できる指示書を皆さんがつくるわけです。私たちが役に立つサービスを考える技術や、コンピュータへそれを正しく指示できる技術をもつことが必要です。

ソフトウェアをつくるためには、プログラミング言語を知っていれば大丈夫と考えている人もいるかもしれません。でも、本当に必要なものを適切につくれなければ誰も使ってはくれません。リッチな機能をもつシステムは大規模で複雑です。人々の要求は限りないので、素早く対応しなければなりません。一人の力でつくれるものは限られます。本書では、役に立つソフトウェアをどのように考えながらつくっていくのかを、モデリング言語 UML (Unified Modeling Language) を用いて説明します。

モデルはアイデアをスケッチするのには有効だが、それ以上のものではない、と思っている人もいるでしょう。確かに、モデルは、要求を分析し、設計のアイデアを整理し、表現するものですが、そのアイデアをコンピュータへの指示書に正しく反映できなければ役に立ちません。モデリング言語の使い次第で、役に立つモデルになるかどうかが決まるのではないのでしょうか。ソフトウェア開発に携わりたいと考えている皆さんには、役に立つモデルをつくり、役に立つソフトウェアを考える力をつけてほしいと思っています。

モデリングとは、対象を表す模型（モデル）によって、対象のある側面をわかりやすくとらえる作業です。ソフトウェア開発ではなんのためにモデルをつくるのでしょうか？ ソフトウェアは、最終的にはプログラミング言語で書かれた命令列をコンピュータがその順番どおりに実行することで、人々が要求していたことを直接的あるいは間接的に実行するものです。人々の要求という、曖昧で、不正確で、実現できるかもわからないものを、その人々が「思っていることができる！」と思えることを実現する指示書としてつくり込むわけです。指示書ができ上がるまでには、要求項目を決める、要求項目を実現する方法を決める、この方法を効率よく特定の環境で実現する方法を決定する、決定に従ってプログラミング言語を使って指示書をつくる、指示書がこれらの決定に従っているかどうかをテストする、といった段階を踏みます。このような段階形式で作業を進めていくためには、そのときには考えなくてもよいことを捨てて必要な側面だけを明らかにする、という**モデリングの観点**を学習することが、役に立つソフトウェアをつくることにつながります。

観点をとらえてうまく対象の性質をモデリングできたとしても、それが適切であるかどうかを確認することも重要です。この適切であるかどうかの基準が、ソフトウェアの**品質**です。

- 課題文を読み、要求をとらえてUMLで定義する。定義したものが妥当かどうかを検討する。定義した要求はどのような品質を保証できるかを考える。
- 定義した要求をプログラミング言語 Java で定義できるようにUMLで定義する。定義されたものが要求を満たしているかを検討する。

本書では、こういったプロセスをシンプルな事例を用いて解説し、モデリングの観点と品質について一緒に考えていくことにします。

ソフトウェア関連の用語については、広義・狭義も含め、いろいろな使われ方をすることが多いと思います。ここでも、特定の概念には妥当な定義を与えて解説していきたいと思います。

本書では、例えば Web アプリケーションのフレームワークや、データベー

スとの接続などの具体的な方法には言及しません。それらを利用して、いかに  
つくりたいソフトウェアを設計していくかが本書の主な目的です。

ソフトウェア開発に携わりたいと考えている学生の皆さんが、本書を通じ  
て、人・社会・環境の中で役に立つソフトウェアをつくってみたいと思うよう  
になっていただければ幸いです。

2016年8月

松浦 佐江子

# 目 次

## 1. ソフトウェアとは

1.1	ソフトウェア開発とは	1
1.1.1	ソフトウェアの役割	1
1.1.2	ソフトウェアのライフサイクル	2
1.2	ソフトウェアの品質	3
1.2.1	品質をつくり込むとは	3
1.2.2	機能要求と非機能要求	5
1.3	オブジェクト指向開発と UML	8
1.3.1	モデリングの観点とソフトウェアの構造	8
1.3.2	UML の 役 割	12

## 2. 事 例

2.1	会議室予約システム	21
2.1.1	初期要求文	21
2.1.2	課題の特徴と本書での扱い方	23
2.2	長方形エディタ	25
2.2.1	初期要求文	25
2.2.2	課題の特徴と本書での扱い方	27

## 3. 要 求 分 析

3.1	要求分析の目的とレビューポイント	30
3.1.1	ユースケース分析	30
3.1.2	ユースケース図とユースケース記述	33

3.1.3	データモデリングとクラス図	37
3.2	会議室予約システムの要求分析	40
3.2.1	システムの目標とステークホルダー	40
3.2.2	ユースケース	45
3.2.3	データモデリング	49
3.2.4	ユースケース記述	53
3.2.5	非機能要求の観点からの分析	57
3.3	長方形エディタの要求分析	63
3.3.1	ユースケース	63
3.3.2	ユースケース記述の基本フロー	66
3.3.3	データモデリング	69
3.3.4	ユースケース記述の例外フロー	73
3.4	事例の考察	82
3.4.1	会議室予約システム	82
3.4.2	長方形エディタ	84

## 4. 設 計

4.1	要求分析から設計へ	86
4.2	構造の設計	88
4.2.1	クラス図とオブジェクト図による構造のモデリング	88
4.2.2	クラスと関連	95
4.2.3	クラス図の確認	100
4.3	振舞いの設計	100
4.3.1	シーケンス図の目的	100
4.3.2	クラスへの操作の割当て	102
4.3.3	シーケンス図の確認	115
4.3.4	ステートマシン図の目的	116
4.4	モデルからプログラムへのトレーサビリティ	119

4.5	会議室予約システムの構造と振舞いの設計	128
4.6	長方形エディタの構造と振舞いの設計	135
4.7	考 察	148

## 5. 設計から実装へ

5.1	共通部分の設計	152
5.2	リファクタリングとデザインパターン	163
5.2.1	イテレータパターン	163
5.2.2	テンプレートメソッドパターン	173
5.3	モデルからのコード生成	184
5.4	考 察	199

## 6. 役に立つ UML モデリングへ向けて

6.1	モデリングの原則	201
6.2	モデル駆動開発と関連技術	204
6.3	ま と め	210

## 付 録

A.1	UML モデリングツール	212
A.2	ユースケース「認証する」のプログラム	213

## 引用・参考文献

索 引		220
-----	--	-----

# 1

## ソフトウェアとは

本章では、ソフトウェア開発の流れと、モデリングの観点について解説し、ソフトウェアをつくるときに、なにを考えればよいかについて整理する。

### 1.1 ソフトウェア開発とは

#### 1.1.1 ソフトウェアの役割

コンピュータのできることは「あらゆる計算可能な数を人より速く正確に計算する」ことであるといえる。この能力を最大限に生かすために、ソフトウェアがあり、人間の暮らしを豊かにするためのさまざまなサービスを提供している。ここでのサービスとは「コンピュータを使用して社会におけるさまざまな問題を解決する手段の提供」を意味する。

サービスはコンピュータを利用したシステムによって提供される。システムは単独のパーソナルコンピュータ（以下PC）であったり、ネットワークで連結したコンピュータであったり、特定の装置に組み込まれたコンピュータであったりする。

ソフトウェアは、「サービスを実現するために、コンピュータがどのように動作すればよいかの指示を記述したコンピュータへの指示書」である。そして、コンピュータへの指示書を記述する言語がプログラミング言語であり、「プログラミング言語で書かれたコンピュータへの指示書」をプログラムと呼ぶことにする。

サービスを「コンピュータを使用して社会におけるさまざまな問題を解決す



## 2 1. ソフトウェアとは

る手段の提供」と考えると、サービスは、これらのシステムが「\*\*できる」という言葉で表現することができる。「\*\*できる」ということは、ソフトウェアの「機能」と呼ばれている。例えば、インターネットショッピングでは、いろいろな種類の商品を選ぶことができる、気に入った商品を注文することができる、注文をキャンセルすることができるなどなど、商店へ行かなくても、いつでも好きなときに、気に入った商品を購入することができる、という一つの解決策を消費者に提供している。

コンピュータを使用して社会におけるさまざまな問題を解決することにより、人、社会、環境へ有益な効果をもたらすことができる。しかし、有益な効果をもたらすには、開発者が本当に必要なことを、ソフトウェアとして正しくつくらないと意味がないことに注意してほしい。サービスが、人、社会、環境に有益な効果をもたらせるかどうかの鍵を握るという点で、ソフトウェアは大きな役割を担っている。

### 1.1.2 ソフトウェアのライフサイクル

先に述べたインターネットショッピングは、ほしい品物を手軽に手に入れたいという要求に対して、一つの解決策を与えているが、ビジネスの世界は競争が激しいので、いろいろな付加価値を付けて顧客を引き付けたり、増やしたりしなければならない。また、利用者もさらなる利便性を求める。一つのサービスも利用者の要求やハードウェアの進化で変化し続け、それに伴い、ソフトウェアも変化しなければならない。

このように、ソフトウェアは人々の要求から出発し、その解決策をプログラムとして実現した後も、その役目を終えるまで新たな要求を取り入れて進化し続ける。図1.1の外側のサイクルがこれを表している。大規模で複雑なものを考えるときの一つの手段として、観点を変えて段階的に物事を考えるという方法がある。サービスの開発では、**要求分析**、**設計**、**実装**という異なるモデリングの観点の段階を経てプログラムを作成する。各段階において、作成されるものを**プロダクト**と呼ぶ。これらは段階ごとに、仕様書、設計書、プログラムと

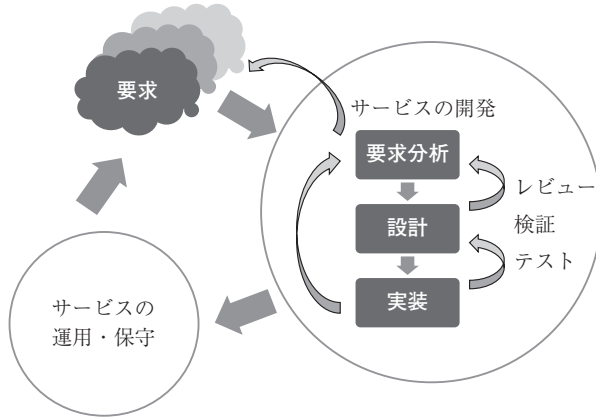


図 1.1 ソフトウェアのライフサイクル

呼ばれる。プロダクトがその前段階のプロダクトを満たしているかを確認する方法として、レビュー、検証、テストがある。図 1.1 のサービスの開発の内側のサイクルがこれを示している。要求へ向かった矢印は、要求分析のプロダクトがそもそもの要求を満たしているかを確認する作業である。これらを合わせて、ソフトウェアの**ライフサイクル**と呼ぶ。

要求の変化に伴い、プロダクトを適切に変更しなければならない。そこで、ソフトウェアが変更しやすいという性質をもっていることは重要である。このソフトウェアの性質を**保守性**と呼ぶ。この性質はソフトウェアのライフサイクルにおいて大きな意味をもち、サービスの開発の間につくり込む必要がある。

## 1.2 ソフトウェアの品質

### 1.2.1 品質をつくり込むとは

1.1.1 項で述べたように、ソフトウェアは「サービスを実現するために、コンピュータがどのように動作すればよいかの指示を記述した、コンピュータへの指示書」であり、システムが「サービスを実現するために\*\*できる」というソフトウェアの「機能」をプログラムとして定義する。1.1.2 項で述べた要

#### 4 1. ソフトウェアとは

求分析，設計，実装の段階を経て，要求を満たすプログラムをつくるわけであるが，「要求を満たす」とは「\*\*できる」というだけでよいのだろうか？

インターネットショッピングの機能には，商品を選ぶことや，商品を注文することがある。しかし，ただ選べればよい，とにかく注文できればよい，というわけではない。商品を選ぶ場合に，例えば「今日は目的の商品があり，それを探したい」とする。商品名が正確にはわからないときには，近いキーワードで探したり，カテゴリーやジャンルといった分類で探したりして，「簡単に」目的の商品にたどり着けると便利である。たくさんの商品が検索できたら，今度は「見やすい」ように商品を絞り込んだり，順番にページを「サクサクとたどってみたり」できるとさらによい。また，商品の写真を拡大してよく見ることができると「わかりやすい」だろう。お気に入りが見つかったら，「スムーズに」買い物かごに入れられるとよいし，支払いをカードとするならば，大事な情報を「安全に」相手に送れないと心配である。

このように，ソフトウェアには，価値のある，役に立つサービスとするために満足すべき性質がいくつかある。例えば，国際標準化機構 ISO (International Organization for Standardization) で策定されている規格の一つであるソフトウェア品質特性の規格では，下記のように六つの性質が定義されている。これらの性質は，ソフトウェアをつくる際に念頭に置くべき事項であり，またつくられたソフトウェアがこれらの性質を満足しているかどうかについて，つねに検討する必要がある。

**〔1〕 機 能 性** ソフトウェアは，「\*\*できる」ということを，ユーザの要求や開発システムが遵守すべき規格・法律・規則などに従って適切に処理し，期待される正しい結果をもたらすことが必要である。また，ソフトウェアやその対象となるデータに関して不当なアクセスを排除できるように，セキュリティに関する要件も満たさなければならない。

**〔2〕 使 用 性** ソフトウェアが役に立つ機能を提供していても，難しく使いこなせなければ，結局は役に立たない。わかりやすく，操作がしやすく，誰でもすぐに使えるようになることが大切である。

〔3〕 **信頼性** 交通管理や銀行などのシステムのように、故障による社会的影響の大きいシステムの場合には、ソフトウェアがその機能をきちんと維持できることが大切である。また、障害が発生してもその影響を最小限に食い止め、短時間に容易に復旧できるようにするための工夫が必要になる。

〔4〕 **効率性** ソフトウェアは限られたコンピュータの資源、例えばメモリなどを利用するため、実行時には、効率よくその資源を利用しなければならない。さらに、大量のデータに対しても、処理時間や応答時間を短くする工夫が必要である。

〔5〕 **保守性** コンピュータを利用したサービスが増加・拡大しているということは、サービスを実現する膨大な量のソフトウェアを開発しなければならないということである。そのために、すべてを新規に開発するのではなく、これまでのソフトウェア資産を再利用することによって、開発が行われる必要がある。すなわち、ソフトウェアは一度つくられたら絶対に変わらないのではなく、つねに変更されるという意識が不可欠である。そこでソフトウェアは、新たな要求に対して変更しやすいように、その内容がわかりやすく、変更によって予期せぬ問題を引き起こさないことを保証するような性質をもつ必要がある。

〔6〕 **移植性** ソフトウェアはさまざまな異なる種類のコンピュータの上で動作する。したがって、同じサービスを提供するソフトウェアが、さまざまな異なる種類のコンピュータでも容易に動作するようにできることが必要である。

役に立つサービスを開発するには、このようなさまざまな環境の中で、そのサービスに必要な性質を満たすようにソフトウェアをつくり込む必要がある。

### 1.2.2 機能要求と非機能要求

サービスは「社会におけるなんらかの問題を解決する手段の提供」であるので、ソフトウェアはその実現の目的や目標をもつ。目標を達成するために「\* \*できる」という要求は**機能要求**と呼ばれる。その他の要求をまとめて**非機能**

# 索引

<b>【あ】</b>	オブジェクト指向	11	——への振舞いの割当て	88
アクション	オブジェクト図	12, 88	クラス図	12, 30, 51, 69
アクションのクラスの操作	オブジェクトノード	14, 18	——の確認方法	100
への割当て	<b>【か】</b>		<b>【け】</b>	
アクションノード	開発プロセス	9	継承	154, 161
アクションの役割	拡張	64	厳密化・形式化	202
アクター	活性区間	101	<b>【こ】</b>	
——の役割	ガード	18, 117	構造	9
アクティビティ	関心の分離	202	——の設計	149
アクティビティ図	関連	38, 39	構造図	12
	——の名前	39	効率性	5
	——の向き	39	コミュニケーション図	13
アスペクト指向	関連クラス	90	コンストラクタ	124
<b>【い】</b>	<b>【き】</b>		コントロールの役割	103
移植性	機能	3	コンポーネント図	12
イテレータ	機能性	4	<b>【さ】</b>	
イテレータパターン	機能要求	5	再利用	184
イベント	基本型	205	再利用しやすい構造	152
インスタンス	基本型と集合	39	サービス	1
インスベクション	基本フロー	8, 31, 67	サブアクティビティ	64
インタフェース	キュー	168	サブクラス	161
インタラクションの役割	<b>【く】</b>		差分	155
<b>【え】</b>	クラス	11	<b>【し】</b>	
永続化したデータの取得	——の共通部分のまとめ方	152	シグネチャ	107, 121, 156
	——の構成要素	153	——を決定する	188
エンティティクラス	——の責務	100, 119, 133, 150, 152	シーケンス図	13, 100
エンティティの役割	——の操作に割り当てる	187	——の分析の確認方法	115
<b>【お】</b>	——へ操作を割り当てる	102	——へのマッピング	103
同意図			事後条件	31, 67
オーバーライド			システム	1
オーバーロード				
オブジェクト				

—の目標 42, 57  
 —の目標の見直し 61  
 —の役割 103  
 事前条件 31, 67, 112, 114  
 事前条件・事後条件 105  
 実装 2  
 シナリオ 43  
 集合を表すコレクション型 205  
 出力 7  
 ジョインノード 69  
 使用性 4, 28  
 状態 117  
 状態遷移 117  
 処理手順 7  
 信頼性 5

**【す】**

スケルトンコード 126, 200  
 ステークホルダー 33  
 ステートマシン図 13, 116  
 ステレオタイプ 16  
 スーパークラス 161

**【せ】**

制御構造をもつシーケンス  
 図 102  
 セキュリティ機能方針 208  
 セキュリティの規則 208  
 セキュリティ要求 185, 207  
 設計 2  
 設計時のモデリングの観点 149  
 漸増的開発 202

**【そ】**

相互作用概要図 13  
 相互作用図 13  
 操作 18  
 —のクラスへの割当て 131  
 —のシグネチャ 107, 140  
 双方向関連 93

属性 18  
 —の制約 94  
 ソフトウェア 1  
 ソフトウェア品質特性 4

**【た】**

代替フロー 31  
 タイミング図 13  
 多重度 39  
 多対多の関係 52  
 段階的詳細化 8  
 単方向関連 93

**【ち】**

抽象化 202  
 抽象クラス 161, 181  
 抽象メソッド 161, 181

**【て】**

定義本体の差分 157  
 ディレクトリの探索 163  
 デザインパターン 163  
 デシジョンノード 18  
 データ構造 13  
 データの CRUD 46  
 データモデリング 95  
 テンプレートメソッド  
 パターン 163, 173

**【と】**

トレーサビリティ 119, 184, 204

**【に】**

入出力 147  
 —の方法 144  
 入力 7

**【は】**

配置図 12  
 バウンダリクラス 87, 142, 196  
 バウンダリの役割 103

派生属性 99  
 パッケージ 87  
 パッケージ図 12  
 パーティション 17, 67

**【ひ】**

非機能要求 5

**【ふ】**

フィールド 126  
 フォークノード 69  
 複合構造図 12  
 不変条件 73, 78  
 振舞い 9  
 —の設計 149  
 振舞い系列 9, 13  
 振舞い図 12  
 プログラム 1  
 —のクラス図の変化 183

プロダクト 2  
 分割統治 8

**【へ】**

変化しないもの 170  
 変化するもの 170  
 変化の予測 202

**【ほ】**

包含 64  
 保守性 3, 5, 29, 149, 152

**【ま】**

マージノード 18  
 マッピング 128, 136

**【め】**

メソッド 121  
 メッセージ 101

**【も】**

モジュール 9  
 モジュール化 202

モデリングツール	204	ユースケース記述	16, 30		
モデリングの核になる構造		ユースケース図		<b>【り】</b>	
	86		12, 13, 16, 30, 48, 63	リスク	59
モデリングの観点	9	ユースケース「認証」	184	リバース	127
モデリングの観点と作業	83			リファクタリング	163
モデル	9, 119	<b>【よ】</b>		リンク	89
問題の一般化	202	要求仕様書	6	<b>【れ】</b>	
<b>【ゆ】</b>		要求分析	2	例外フロー	8, 31, 75
誘導可能性	91, 93	要求分析から実装までの		レビュー	33
ユーザインタフェース	150	モデリングの核になる		<b>【ろ】</b>	
ユースケース	13, 31	構造	11	ロール	39
——の拡張	64	<b>【ら】</b>			
——の抽出	45	ライフサイクル	3		
——の候補	47	ライフライン	101		

<b>【c】</b>		<b>【M】</b>		<b>【T】</b>	
CRUD	31	MDA	205	try catch 構文	109, 165
CUI	142, 196	<b>【o】</b>		<b>【U】</b>	
CUI と GUI	85, 150	OCL	205	UML	12
<b>【G】</b>		——の型	205	<b>【w】</b>	
GUI	144	<b>【s】</b>		Web アプリケーションの	
<b>【I】</b>		SFP	208	基本的な構造	134
is-a 関係	154				

— 著者略歴 —

1979年 津田塾大学学芸学部数学科卒業  
1982年 津田塾大学大学院修士課程修了（数学専攻）  
1985年 津田塾大学大学院博士課程単位取得退学（数学専攻）  
1985年 株式会社 管理工学研究所研究員  
～2002年  
2001年 博士（情報科学）（早稲田大学）  
2002年 芝浦工業大学助教授  
2006年 芝浦工業大学教授  
現在に至る

ソフトウェア設計論 —役に立つUMLモデリングへ向けて—

Software Design Methodology — Towards Useful UML Modeling —

© Saeko Matsuura 2016

2016年10月13日 初版第1刷発行



検印省略

著者 まつ とうら き え こ  
松 浦 佐 江 子  
発行者 株式会社 コロナ社  
代表者 牛来真也  
印刷所 萩原印刷株式会社

112-0011 東京都文京区千石 4-46-10

発行所 株式会社 コロナ社

CORONA PUBLISHING CO., LTD.

Tokyo Japan

振替 00140-8-14844・電話 (03) 3941-3131 (代)

ホームページ <http://www.coronasha.co.jp>

ISBN 978-4-339-02681-8

(金)

(製本：愛千製本所)

Printed in Japan



本書のコピー、スキャン、デジタル化等の無断複製・転載は著作権法上での例外を除き禁じられております。購入者以外の第三者による本書の電子データ化及び電子書籍化は、いかなる場合も認めておりません。

落丁・乱丁本はお取替えいたします