

## 開発プロセス

### 目次

1. はじめに	1
2. プロセスモデル	2
1 代表的なプロセスモデル	2
2 ウォーターフォールモデルとは	4
3 工程の種類	5
3. 要件定義	6
1 要件定義の概要と重要性	6
2 要件定義が定める要求の種類	6
3 要件定義書を読む	7
4 【練習】ユースケース図を描いてみよう	9
4. 基本設計書	10
1 基本設計の位置付け	10
2 基本設計書を読む	10
5. 詳細設計書	12
1 詳細設計の目的	12
2 詳細設計書を読む	12

## 1. はじめに

近年、市場にニーズに伴いシステムの大規模化、複雑化が顕著となってきています。システム開発は多くの時間や労力、専門知識が求められ、複数人のエンジニアで開発することが一般的となりました。チームを組み、品質や生産性を確保しながらシステム開発を進めるためには、各エンジニアが共通の開発の進め方を理解した上で、作業を遂行することが重要です。この開発の進め方のことを「開発プロセス（開発工程）」と呼びます。

第2章では、代表的な開発プロセスを紹介します。

また、開発プロセスに従って段階的に作業を行う際には、以下のような情報が用意されている必要があります。

- お客様がどのような機能を求めているのか。
- その機能を実現するには、どのように作成すればよいのか。

これらの情報をまとめた文書のことを設計書（仕様書）と呼びます。設計書は単純な文章と異なり、システム開発に必要な情報を独特の表現で記載されていることが多いです。そのため、予め設計書の読み方を知っておく必要があります。

第3章以降では、システム開発を行う際に使用する設計書の読み方について紹介します。今後のシステム開発に役立ててみましょう。

## 2. プロセスモデル

### 1 代表的なプロセスモデル

開発プロセスには幾つかのフェーズ(工程段階)が存在し、時系列的にどのような順序でフェーズを実施するかをモデル化したものにプロセスモデル(工程モデル)があります。代表的なプロセスモデルとしてはウォーターフォールモデル、プロトタイピングモデル、スパイラルモデル、アジャイルソフトウェア開発があります。それぞれの特徴は次の通りです。

プロセスモデル	説明
ウォーターフォールモデル	各フェーズをトップダウンで分割するプロセスモデル。プロトタイピングモデル、スパイラルモデル、アジャイルソフトウェア開発と異なり、同じフェーズを繰り返すことはせず、進捗管理がしやすいという特徴がある。(4 ページ)
プロトタイピングモデル	早期に試作品を作り、ユーザが検証した後、そのフィードバックに基づいて新たな試作品を作るという繰り返しを行うプロセスモデル。ユーザに対しより高度なシステムを提供出来るという特徴がある。(図 2-1)
スパイラルモデル	リスクが最小になるように設計とプロトタイピングを 2~3 カ月の期間で繰り返すプロセスモデル。システムの規模やスケジュールの予測がしやすいという特徴がある。(図 2-2)
アジャイルソフトウェア開発	スパイラルモデルと同じように、リスクが最小になるように設計とプロトタイピングを繰り返すプロセスモデル。スパイラルモデルよりも繰り返しを行う期間が 1~4 週間と短く、小規模開発に適しているという特徴がある。(図 2-3)

これらプロセスモデルは開発工程のひな型であり、全てのシステム開発を確実に成功へ導くような共通の開発プロセスは存在しません。プロセスモデルに縛られることなく、開発するシステムに応じた最適な工程を作ることも重要です。

4 ページから、これらの中でも従来から広く用いられているウォーターフォールモデルについて触れていきます。

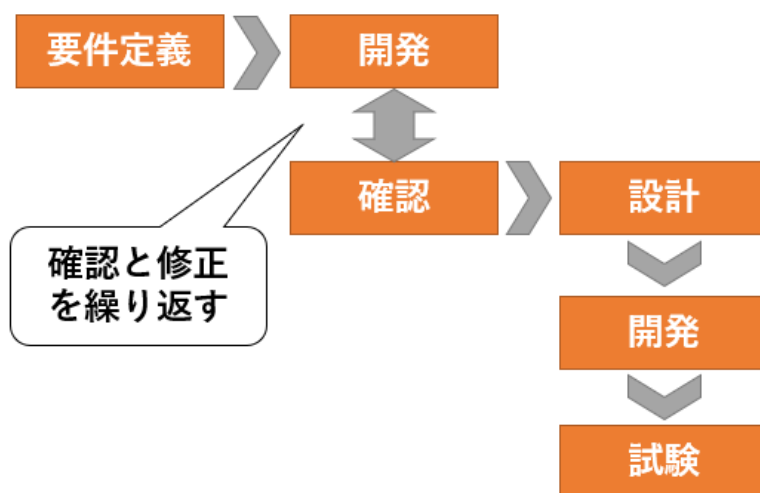


図 2-1.プロトタイピングモデル

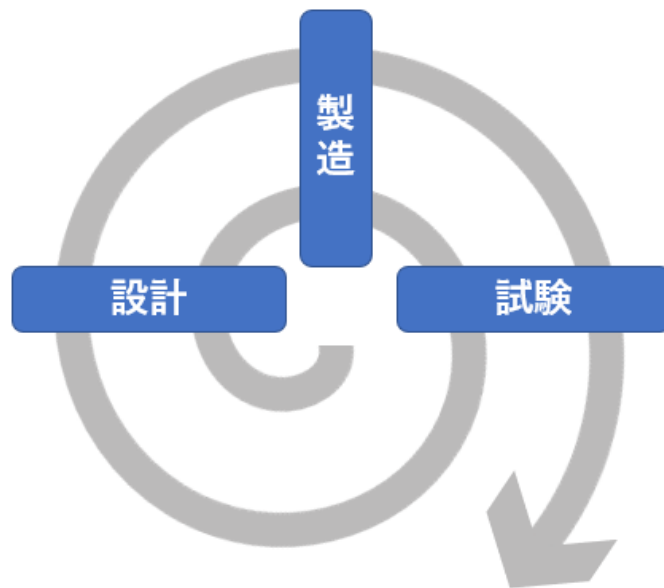


図 2-2. スパイラルモデル

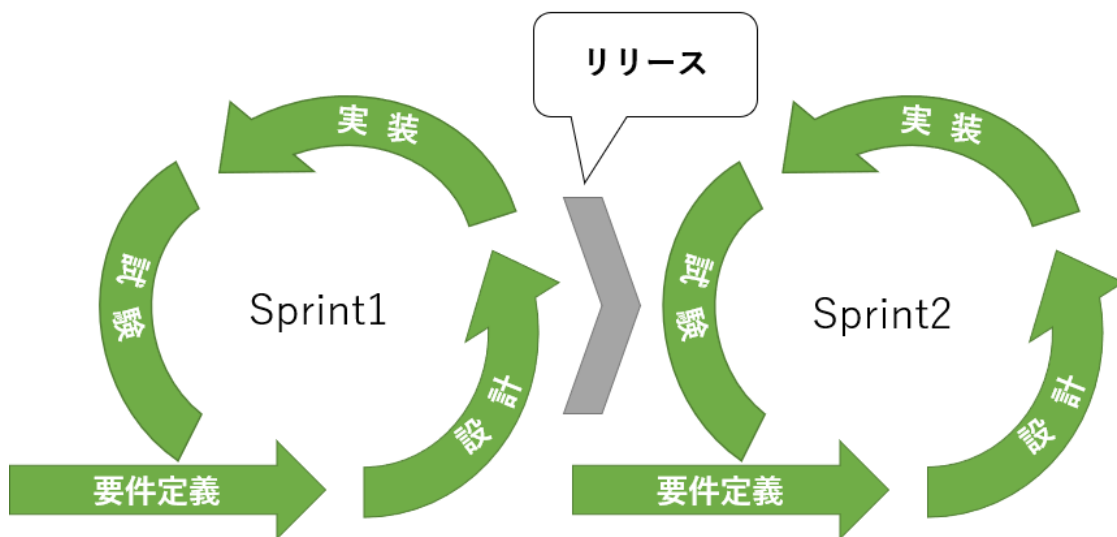


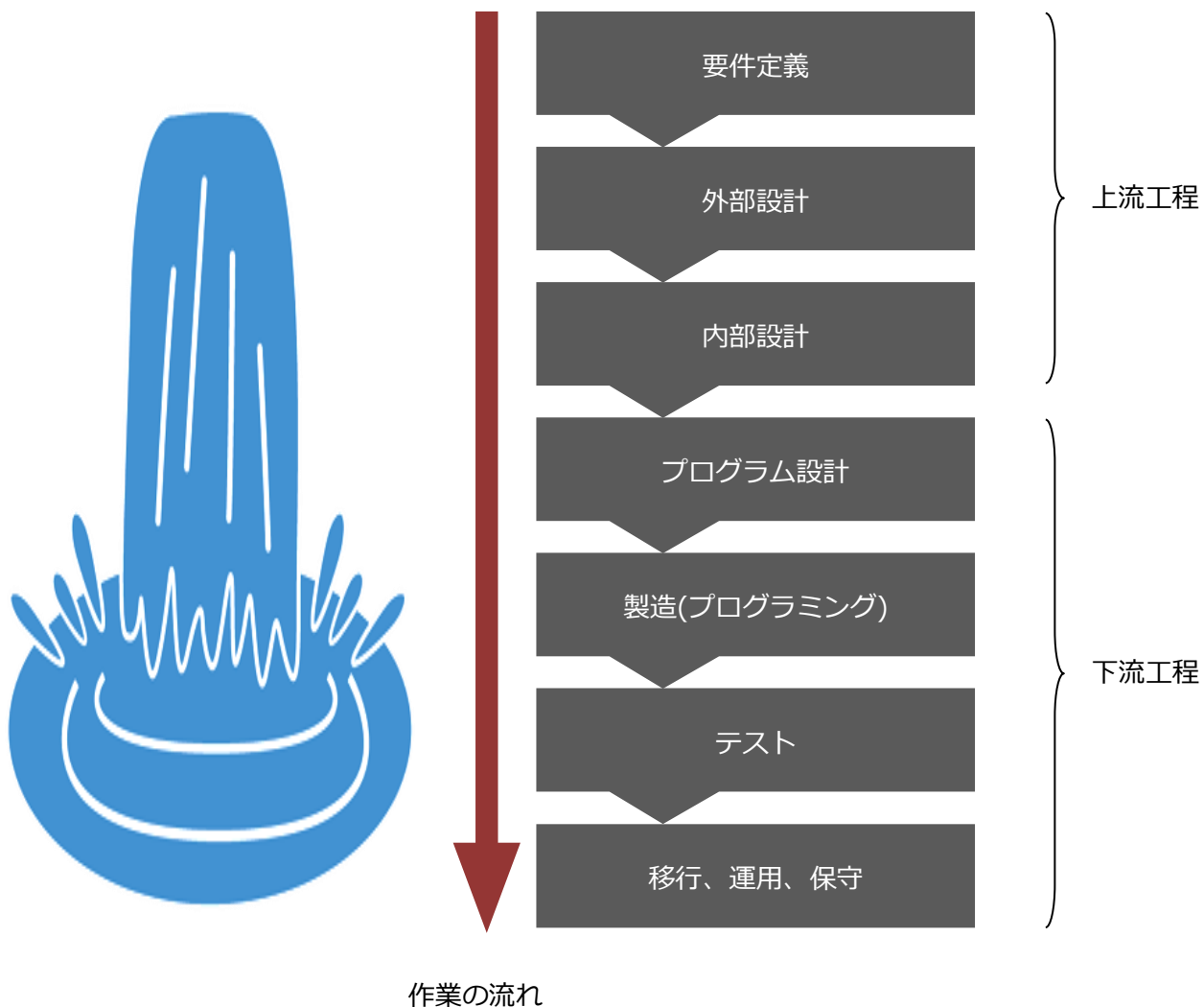
図 2-3. アジャイルソフトウェア開発

## 2 ウォーターフォールモデルとは

ウォーターフォールは滝という意味です。ウォーターフォールモデルは滝を流れる水のように、各フェーズを順々に実行していくプロセスモデルです。

各フェーズでは、設計書等の成果物が作られ、次のフェーズではその成果物を基に開発が進みます。また、各フェーズの終わりでは必ずレビューを行い、成果物を検証することでフェーズを完結させます。原則として前のフェーズに戻ることは許されません。万一不具合等で前のフェーズに戻って修正作業を行うことになった場合、工程が大幅に遅れ、多額の費用が掛かってしまう恐れがあるためです。すなわち、レビューを行うことで成果物の品質を確保し、前のフェーズへの後戻りを最小限にすることが重要となります。

ウォーターフォールモデルでは線表(ガントチャート)を使用し、各フェーズを一度で終わらせる計画を立て、進捗管理を行います。フェーズが一直線に並び、各フェーズで得られる成果物が明確であるため、どこまで作業が進んでいるかの進捗管理がしやすいというメリットがあります。



## 3 工程の種類

ウォーターフォールモデルの工程は、内部設計までの上流工程、プログラム設計以降の下流工程の2つに大別することが出来ます。

### 上流工程

フェーズ	作業内容	成果物
要件定義	<ul style="list-style-type: none"> <li>・問題点を分析し、システムを提案する</li> <li>・ユーザ要求を明確にする</li> <li>・開発体制等のプロジェクト実行計画を立てる</li> </ul>	<ul style="list-style-type: none"> <li>・システム化計画書</li> <li>・要件定義書</li> <li>・開発計画書</li> </ul>
基本設計 (外部設計)	<ul style="list-style-type: none"> <li>・システムの機能を確定する</li> <li>・ユーザ側の立場で画面設計、帳票設計、コード設計等を行う</li> </ul>	<ul style="list-style-type: none"> <li>・基本設計書</li> <li>・基本設計レビュー報告書</li> </ul>
詳細設計 (内部設計)	<ul style="list-style-type: none"> <li>・システムの機能を基に、プログラム間のデータや処理の流れを明確にする</li> </ul>	<ul style="list-style-type: none"> <li>・詳細設計書</li> <li>・詳細設計レビュー報告書</li> </ul>

### 下流工程

フェーズ	作業内容	成果物
プログラム設計	<ul style="list-style-type: none"> <li>・各プログラム内の構造設計を行う</li> <li>・プログラムのテストケースの設定をする</li> </ul>	<ul style="list-style-type: none"> <li>・プログラム設計書</li> <li>・プログラム設計レビュー報告書</li> </ul>
製造 (プログラミング)	<ul style="list-style-type: none"> <li>・ソースコードを作成する</li> <li>・ソースコードのレビューを行う</li> </ul>	<ul style="list-style-type: none"> <li>・ソースコード</li> <li>・ソースコードレビュー報告書</li> </ul>
テスト	<ul style="list-style-type: none"> <li>・単体テスト計画書を作成しテストを行う</li> <li>・複数のモジュールを結合しテストを行う</li> <li>・システム全体として動作させテストを行う</li> <li>・本番と同じ環境とデータで運用テストをする</li> </ul>	<ul style="list-style-type: none"> <li>・単体テスト設計書</li> <li>・単体テスト報告書</li> <li>・結合テスト設計書</li> <li>・結合テスト報告書</li> <li>・システムテスト設計書</li> <li>・システムテスト報告書</li> <li>・運用テスト設計書</li> <li>・運用テスト報告書</li> </ul>
移行	<ul style="list-style-type: none"> <li>・ユーザの承認を得る受け入れテストを行う</li> </ul>	<ul style="list-style-type: none"> <li>・受け入れテスト設計書</li> <li>・受け入れテスト報告書</li> </ul>
運用、保守	<ul style="list-style-type: none"> <li>・システムが正常に稼働するように修正する</li> <li>・法律の変更等によるシステムの変更を行う</li> <li>・修正、変更後はレグレッションテストを行う</li> </ul>	<ul style="list-style-type: none"> <li>・保守内容に伴い修正した設計書</li> <li>・レグレッションテスト設計書</li> <li>・レグレッションテスト報告書</li> </ul>



### 3. 要件定義書

#### 1 要件定義の概要と重要性

要件定義とは、顧客の要求を明確にし、文書として纏める工程です。それだけを聞くとさほど大変な工程には思えないかもしれませんが。例えば、作りたいシステムについて顧客に直接話を聞き、要望をメモ書きとして纏めるというような光景を想像するでしょうか。しかし、ここに最も重要なポイントが存在します。それは、開発対象となるシステムには「顧客自身が気づいていない要求が隠されている」という点です。

多くの場合、初期の顧客の要求は漠然としています。その漠然とした要望をそのまま要件定義としてしまうと、本当に必要とされるシステムは完成せず、後から「やはりあれが必要だった」などということにもなりかねません。以降の工程は全て要件定義で纏められた文書を元に進められるため、要件定義の品質はシステム開発全体の成否にも直結する重要なものなのです。



#### 2 要件定義が定める要求の種類

要件定義を進める際は、顧客に対するヒアリングに加え、現行業務の調査を併せて行います。その調査結果を元に、システムに必要な要求を分析し、明確化していきます。

システムに必要な要求は「機能要求」と「非機能要求」の2種類に分けられます。今回例として配布した「コミュニティサイト要件定義書」は、機能要求を記述した文書という位置付けになります。

要求の種類	意味
機能要求	利用者から見えるシステムの機能やサービスに対する要求
非機能要求	機能要求以外の要求(性能や信頼性に対する要求など)

### 3 要件定義書を読む

それでは「コミュニティサイト要件定義書」を読み進めていきます。

#### ① プロジェクトの背景

そのプロジェクトがなぜ必要になったのかを明文化する項目です。似たようなシステムであってもその目的の違いによって作り方が変わります。大抵の場合は、顧客からのヒアリングによって導かれます。

「コミュニティサイト要件定義書」では、過疎化が進む地方都市の活性化が謳われていますので、開発するシステムの仕様がこの目的に沿うものになっている必要があります。

#### ② プロジェクトの概要

開発するシステムの形式を明らかにする項目です。必要とされるシステムが Web アプリケーションなのか、スタンドアロンアプリケーションなのか等が明記されています。今回開発する予定のシステムは地域活性化のためのコミュニティサイトなので、当然インターネットを使用しますが、社内システムの場合は社内で作られている独自のネットワークを使用する場合もあります。

また、システムが想定している利用者也明らかにしています。

#### ③ 開発環境

システム開発を進める際の開発環境を明記しています。開発言語や DB は使用するバージョンも含めて記述する必要があります。「コミュニティサイト要件定義書」ではアプリケーションサーバとして Tomcat9 を使用することになっています。

#### ④ 運用環境

開発されたシステムが実際に運用される環境について明記しています。

#### ⑤ 機能要求








システムに必要な機能を明記しています。機能一覧については表形式か箇条書きで記述されます。また、より視覚的に分かり易くするためにアクティビティ図やユースケース図などの UML が用いられることも多いです。「コミュニティサイト要件定義書」では、機能の種類が「分類」、機能の名称が「機能名」、機能の内容が「概要」、その機能を使える人が「対象権限」として纏められています。

アクティビティ図とユースケース図の読み方については次頁で解説します。



(a) アクティビティ図

アクティビティ図は処理の流れを表す図であり、考え方はフローチャートに似ています。要件定義で用いられている場合は、「業務フローのフローチャート」と考えると分かり易いでしょう。アクティビティと呼ばれる角が丸い四角の中に処理が書かれ、そこから向かう先が→で示されています。その他の記号の意味は次の表の通りです。

記号	意味
	アクティビティ(処理)
	アクティビティの順序
	分岐
	並行処理
	データ
	一連のプロセスの開始
	一連のプロセスの終了

例えば、「コミュニティサイト要件定義書」の一般会員のアクティビティ図を読むと、一般会員がログイン処理(ID、パスワードの入力)を行った後、コミュニティサイト側がその入力内容がDBに登録されている一般会員情報が確認します。また、記事詳細を見た後でその記事にコメントを書いたり、いいねを付けたりする機能は自由に選べますが、その記事を消す、あるいは記事に対して書かれているコメントを消すという機能は、その記事やコメントが自分で書いたものでない限り出来ません。(この「自分が書いたコメント」「自分が投稿した記事」などの分岐条件をガードと言います)




このようにアクティビティ図を読み進めていくことで、処理の一連の流れを視覚的に理解することが出来るようになります。

(b) ユースケース図

ユースケース図はシステムが外部に提供する機能を視覚的に記述するものです。ユースケース図を用いることによって、機能の範囲やシステムの外部の要素との関連が明確になります。

「アクター」と呼ばれるシステムの利用者と「ユースケース」と呼ばれるシステムの機能を関連付けて記述をしていきます。

次にユースケース図で使われる各記号の意味を説明します。

記号	意味
	システムの利用者
	アクターとユースケースの関連性を表す
	システムが外部に提供する機能 アクターを主語とした時に述語になるように書く

「コミュニティサイト要件定義書」のユースケース図を読むと、コミュニティサイトは一般会員と管理者に対してほぼ同等の機能を提供していますが、「会員を登録する」という機能は管理者にのみ提供されていることが分かります。

また、「記事の削除」「コメントの削除」「会員情報の編集」といった機能は、一般会員に対しては、その会員のものに対してのみ提供されていることが分かります。

#### 4 【練習】ユースケース図を描いてみよう

ユースケース図を描く練習をしてみましょう。銀行のATMをテーマにユースケース図を各自で描いてみて下さい。ここでは、実際のATMの機能を簡略化し、預金の預け入れと引き出し、振込みのみとします。(フリーハンドでノートに描いて下さい)

## 4. 基本設計書

### 1 基本設計の位置付け

基本設計フェーズは、要件定義書の記述に基づき、定められた要件をどのように実現するのかを分析し、論理的に纏める工程です。具体的には、機能一覧や画面遷移、データベース定義を成果物として作成していきます。この工程で作成される成果物は、以降の工程で作成される成果物の論理的な根拠となりますので、非常に重要です。次項よりこれら基本設計フェーズの成果物一つ一つについて、「コミュニティサイト基本設計書」を例に、どのように読めば良いのかを解説していきます。

### 2 基本設計書を読む

#### ① 開発規約

開発を進めるにあたり、開発者が従わなければならない約束事が記述されています。

「コミュニティサイト基本設計書」では、プロジェクト名、パッケージ名の命名規約、画面デザインの規約、メッセージや数字の表示ルールが示されていますが、これは一例です。実際には開発プロジェクトごとに開発規約の内容は異なります。

#### ② 機能一覧

要件定義書の機能要求を、処理内容や入出力項目の内容まで、より具体的に記しているのが機能一覧です。「大分類」「小分類」という項目名で示されているのは、それぞれ「機能を大枠で纏めたもの」と「機能をトランザクション単位で分けたもの」です。例えば、「コメント一覧表示」「コメント投稿」「コメント削除」の三つの小項目は、「コメント管理」という大項目に纏められます。

「入力項目」「出力項目」は、その機能を実行する過程で必要になる入力データの種類、出力データの種類が示されている項目です。

#### ③ 画面遷移図

どの画面からどの画面に移動するのかを視覚的に理解できるように纏めた資料です。→が示す方向に画面が移動していきます。詳細設計書の画面詳細と照合し易くするために、各画面には1.1や1.3.1などの画面番号が振られています。

また、今回開発する対象となっているコミュニティサイトは、ユーザーが持っている権限によって、移動できる画面が異なりますが、その違いは→(矢印)の色分けで表現されています。例えば、「1.2 記事詳細画面」には未ログイン状態でも入れますが、「3.1 会員一覧画面」にはログインしない限り入れないことが、矢印の色から分かります。

矢印には先端が一方を向いているものと、両方を向いているものがありますが、一方を向いているものは、一方通行で元の画面に戻らない遷移を表しています。先端が双方向を向いているものは、その画面に遷移後も元の画面に戻る遷移がある場合を表しています。

#### ④ 入力仕様

入力仕様は、HTML のフォーム部品からユーザーによって入力されるデータの入力可能範囲と、使用するフォーム部品の種類を示しています。

ログイン機能を例にとると、メールアドレスの「コントロール」が「テキストボックス」とあるため、実装する部品はテキストボックスになります。また、「形式」に指定されている「メールアドレス」や、「最小文字数」の「1」、最大文字数の「256」は、入力されたデータが合っているかどうかをシステム側で判断する時に利用します。この仕様に当てはまらないデータが入力された場合は、入力エラーとし、再度入力を促す処理を実装する必要があります。

#### ⑤ データベース定義

要件定義書の機能要求を実現するために、どのようなカラムを持つどのようなテーブルが必要なのかを記した資料です。この資料で示された内容によって、プログラムの内容も変化しますので、大変重要な資料です。仮にデータベース定義に誤りがあった場合は、開発の進捗にも大きな影響を与えます。十分な分析と検討を重ねた上で作成する必要があります。

「コミュニティサイト基本設計書」では、「community\_user」のスキーマに「articles」、「comments」、「histories」「users」の4つのテーブルを作成することになっています。ここでは「articles」を例に読み方の解説を行います。

まず列名に「列名(論理名)」「列名(物理名)」と2種類ありますが、論理名というのはその列に格納されるデータの意味を表現した名前のことです。それに対して、物理名というのは、実際にテーブルを作る際に(CREATE TABLE を実行する際に)列名として指定する名前のことです。articles の最初の列を例に採りますと、articles には「id」という名前で作られていますが、そこに入るデータは「記事 ID」を表しているということになります。

また、「データ型」に「NUMBER」、「桁数」に「6,0」が指定されていますので、「id」列は小数点以下が入らない6桁の整数値が入ります。「制約」に「PRIMARY KEY」とありますから、作成時に「id」列に対して主キー制約を施す必要があります。以下同様に読み進めて下さい。

#### ⑥ ER 図

ER 図はシステムで利用されるデータ同士の関連性を、エンティティ(実体)とリレーションシップ(関連)で表したもので、データモデルの最もメジャーな表記法です。色々な描き方がありますが、ここでは→(矢印)の始端が1を表し、終端が多を表しています。例えば、articles と comments では、articles 側から矢印が伸びていますので、articles のインスタンス一つと、comments の複数のインスタンスが結びついていることが分かります。

## 5. 詳細設計書

### 1 詳細設計の目的

基本設計フェーズは要件定義を論理的に実装可能な状態まで落とし込む工程でしたが、詳細設計フェーズは、さらにその基本設計を物理的に実装可能な状態まで落とし込む工程です。具体的には、使用するプログラミング言語、フレームワークの特性、Web ブラウザの制約などを考慮して設計を進め、文書を作成していきます。

この工程で求められる成果物は、画面詳細、メッセージ仕様、クラス図などがあります。以降は「コミュニティサイト詳細設計書」を使って、これらの成果物の読み方を解説します。

### 2 詳細設計書を読む

#### ① 画面仕様

システムで使用する各画面が、具体的にどのような構成になっているのかを記述する仕様書が画面仕様です。画面仕様によってボタンや入力フォーム、出力画像、出力メッセージの配置が分かります。特に Web アプリケーションにおいては、ユーザーの使用感に直接影響する設計になりますので、設計全体の中でも重要な位置を占めています。

「コミュニティサイト詳細設計書」では、基本的なレイアウトを示している「基本構成」に始まり、「3.3.1 会員登録完了画面」までの全てのシートが画面仕様になっています。

例として「1.2 記事詳細画面」を見てみましょう。記事に対する「いいねボタン」「戻るボタン」「削除ボタン」が、記事の下部(投稿者の名前、投稿日時の下)に配置されていることが分かります。プログラミングの担当者は、こういった仕様に基づいてページの製造を進めていくことになります。

また、同じページのコメントを一覧表示している箇所を見ますと、「・表示件数の上限は無い」という一文がありますので、コメントに関してはページングを行わず、コメントの数だけ画面が下に伸びていく仕様であることが分かります。この仕様はコメント表示の機能の一環として、基本設計フェーズで既に決められている仕様ですが、プログラミングの担当者は詳細設計書を参考に作成することが多いので、画面仕様で改めて記述し直す必要があります。

#### ② メッセージ一覧

システムで使用するメッセージを一箇所で纏めて指定する資料が、メッセージ一覧(または、メッセージ定義書)です。今回開発対象としているコミュニティサイトにつきましては、各機能固有の通常時に表示するメッセージは、各機能の製造者に任されていますが(※基本設計書の開発規約参照)、入力エラー時に出力するメッセージはこの資料で指定されています。メッセージコードが「errors.login」のように「単語.単語」の形になっているのは、このシステムがフレームワークでの実装を前提にしているからです。実際にはコードの書き方には様々なものがあります。

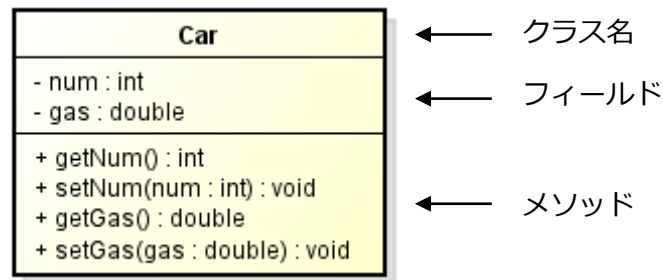
また、「{0}」は「{1}文字以上、{2}文字以下で入力して下さい。」などに見られる{数字}は、変数になっています。実行される機能によって当てはまる数字は異なります。(パスワードは8文字以上、16文字以下で入力して下さい等)



### ③ クラス図

クラス図とは、クラス間の静的な構造を表現するための図です。クラス内にはクラス名、属性、オペレーションを記述し、クラスとクラスの間は線で繋ぎ関連を表現します。基本設計フェーズで利用される分析クラス図と、詳細設計フェーズで利用される設計クラス図に分かれますが、ここでは「設計クラス図」の読み方について解説します。

#### (a) クラスの描き方(Car クラスを例にします)



フィールドの欄は、「変数名 : データ型」の書式になっています。つまり、「num : int」という記述は、int num というフィールドを意味しています。

メソッドの欄は、「メソッド名(引数名 : データ型) : 戻り値のデータ型」という書式になっています。「setNum(num : int) void」という記述は、void setNum(int num)というメソッドを意味しています。

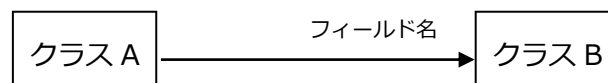
また、アクセス修飾子は、フィールド、メソッドの両方に共通となっています。各記号の意味は次の通りです。

記号	アクセス修飾子
+	public
#	protected
~	指定なし
-	private

Car クラスの場合、フィールドは private、メソッドは public になっています。

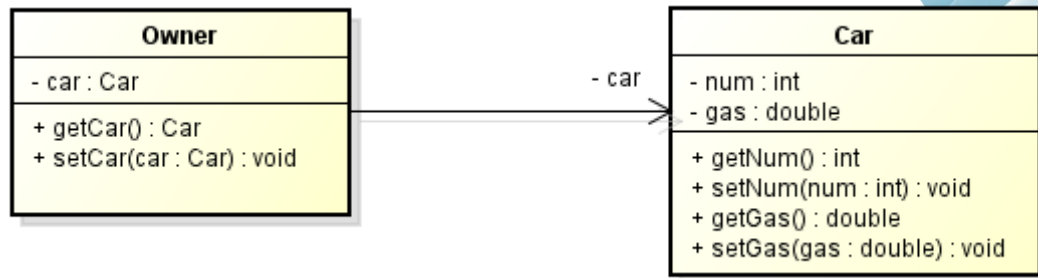
#### (b) 関連の描き方

クラス A がクラス B への参照を持つ場合(クラス A 内にクラス B 型のフィールドがある場合)、両クラスの関連は次のような→(矢印)で表されます。



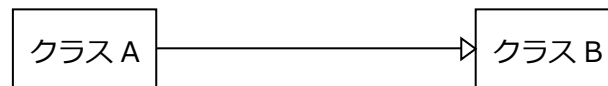


Car クラスと Car クラスを所持する Owner クラスの関連は次のように表されます。

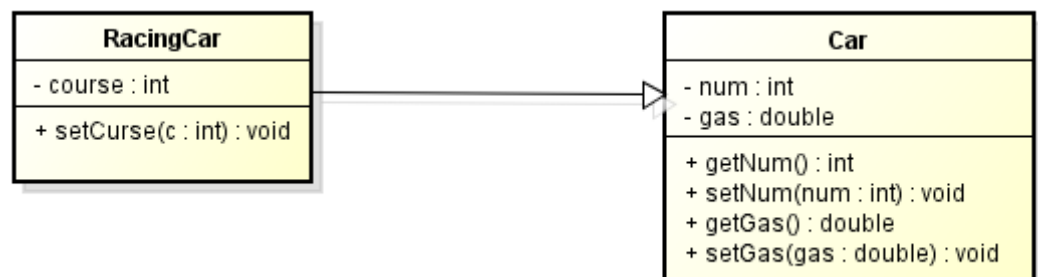


(c) 継承の描き方

クラス A がクラス B を継承している場合、次のような図で表します。

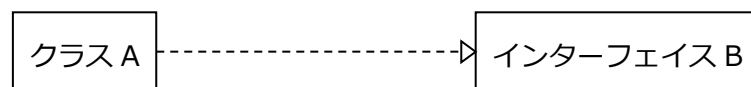


Car クラスが RacingCar クラスに継承されている場合は、次のように描きます。



(d) インターフェイスの実装の描き方

クラス A がインターフェイス B を実装している場合、次のような図で表します。



Car クラスが iVehicle インターフェイスを実装している場合は、次のように描きます。

