学習目標

この学習モジュールを受講すると，情報システムのモデリングに用いるUMLという言語がどのようなものかということと，UMLの中のユースケース図の描き方がわかるようになります．具体的には次のようなことができるように学習しましょう．

- 「UML」とはどのようなものか，目的やメリットを説明し，UMLに含まれる「ダイアグラム」を5個以上言うことができる．
- UMLのうちの「ユースケース図」がどのようなものか，目的やメリット，及び記法を説明できる．
- 「ユースケースの記述項目」にはどのようなものがあるか説明できる．
- 与えられた情報システムの説明を適切に表現するユースケースモデルを，ユースケース図と個々のユースケースの記述によって表現できる．

モデリングとUML

モデリング

モデリングは分析対象の全体像を，理解しやすく，扱いやすい形で表現することです．例えば，自動車を開発するときに，設計図を起こす前に，粘土で自動車の形を作成します．これによって，開発者の間でデザインを視覚的に共有し，デザインについての善し悪しを議論できるようになります．

情報システム開発においては，情報システムをモデリングします．開発者とユーザの間では，開発しようとする情報システムが，どのような場面でどのように用いられるのか，どのようなデータを持ち，どのような処理を行うのかなどを共有し，議論する必要があります．また，開発者のチーム内でも，どのような構造で設計されるのかなどを共有し，議論することになります．そこで，そうした様々な視点で，情報システムを表現したモデルが必要になります．

したがって，モデルは，開発者だけでなくユーザにも理解しやすい形で表現する必要があります．また，そのモデルの利用目的に合わせて，表現方法を選び，具体化のレベルを設定することが非常に重要です．本質的な部分にフォーカスすべきであり，全てを細部まで記述する必要はありません．

UMLは，利用目的に合わせた，ビジュアルでわかりやすいモデルの表現方法を提供しています．

UMLの概要

「UML」は，**Unified Modeling Language** の略で，主に情報システムのモデルを記述するための言語です．言語と言っても，プログラミング言語ではありませんから，UMLで記述したものが，コンピュータでそのまま動作するというわけではありません．また，情報システムのモデリングを主な対象としていますが，ビジネスモデルを記述することも可能です．

統一(Unified)モデリング言語という名称は，様々なモデル表現法を統一して標準化することで制定された言語であることに由来しています．オブジェクト指向が登場して以来，様々なオブジェクト指向開発方法論が研究され，1990年代前半には，同じ利用目的のモデルを表現する方法が何種類も存在していました．特にランボー氏のOMT法，ブーチ氏のBooch法，ヤコブソン氏のOOSE法(Objectory法)がメジャーでした．しかし，多くの記述方法が存在すると，開発プロジェクトごとにモデル記述方法が違い，毎回学習が必要になったり，開発者同士で議論するための共通の記述方法がないとった問題が生じます．ランボー氏がブーチ氏の勤務していたラショナルソフトウェア社に移籍したことを契機として統一のプロジェクトが立ち上がり，UMLが誕生しました．

現在，UMLは，**OMG(Object Management Group)**という標準化団体で仕様策定が行われています．UMLのバージョン1.0が登場したのは1997年で，2005年にはバージョン2.0がOMGで承認されました．2007年9月の最新バージョンは2.11であり，今後も，仕様が変更されていくことでしょう．また，このようにモデルの表記法は統一されましたが，開発プロセスは統一されていないということも覚えておくとよいでしょう．

UMLに規定されているダイアグラム

UMLは，図(ダイアグラム)を使ってモデルを表現します．図を使うことで，人間が見て理解しやすいモデルを記述することができます．

UML 2.0 では，以下の13のダイアグラムが規定されています．しかし，これらのダイアグラムを全て使用する必要はなく，利用目的に合わせてダイアグラムを選択して記述します．

表１ **UML**に規定されているダイアグラム

ダイアグラムの名称	種別	説 明
クラス図	構造図	システム化する対象やシステムの構成要素をクラスとその関係によって表現する図
パッケージ図		パッケージの定義，パッケージ間の関係を表す図
オブジェクト図		ある時点のオブジェクト(インスタンス)間の関係を表す図
コンポジット図（複合構造図）		システム実行時の構造を表す図で，構造全体を表すコンポジットとその内部を構成するパーツから成る
配置図		システムの物理的なハードウェアの構成を表す図
コンポーネント図	システムを構成するコンポーネント，特にソフトウェアやファイルなどの構造を表す図	
ユースケース図	振る舞い図	システムが提供する機能をユーザの視点から表す図
アクティビティ図		業務やシステムの処理フローなど，アクティビティの流れを表現する図
ステートマシン図		オブジェクトの状態遷移と，それに伴うアクションの関係を表す図
シーケンス図		オブジェクト間のメッセージ交換などの協調動作を時系列で表現する図
コミュニケーション図		オブジェクト間のメッセージ交換などの協調動作をオブジェクト同士の関係の視点で表現する図
相互作業概要図		複数のシーケンス図やコミュニケーション図をつないで，各図の処理順序を表す図
タイミング図		時系列に沿ったイベントと，オブジェクトの状態遷移の対応を表す図．主にリアルタイムシステムで利用．

UMLの概要 セルフテスト

次のページに移動して，「UMLの概要」に関するセルフテストを受験して，理解を確認しましょう.

ユースケース

ユースケースは，何らかの目的に関するシステムの機能をシナリオのような形式で表現したものです．例えば，コース管理システムを使って小テストを受験する様子は，以下のように表すことができます．

> まず，受験する小テストのアイコンをクリックして，小テストの画面を開く．
> 表示された画面で，１問ずつ問題に解答していく．
> 解答したら解答結果を保存する．
> 間違った入力をしたり，誤って意図しない選択肢を選んだ場合は
> ，入力しなおしたり，もう一度選択しなおす．
> 全ての問題に解答し，解答を保存したことを確認したら，提出をする．

このようなシナリオを「小テストを受験する」という名を付けて，ひとつのユースケースとして記述していきます．開発しようとする情報システムが持つ機能を，ユースケースとして全てリストアップすることで，システムに要求する機能と開発する範囲を明確にすることができます．また，各ユースケースを明確に記述することによって，情報システムの開発者やユーザの間で，開発する情報システムのイメージを共有することができます．

ユースケースを表現する場合には，システムの内部構造や実装方法には言及せずに，ユーザから見た機能を，具体的な場面に即して記述することが重要です．また，ユーザが理解できるような用語を用いることとし，専門的な用語を極力用いないようにすべきです．

あるシステムに関連するユースケースのリストは，ユースケース図で記述します．ユースケース図では名前だけで表される各ユースケースは，決められた記述項目を具体的に記述します．UMLではユースケース図の記法が規定されていますが，ユースケースの記述項目は規定されていません．

ユースケース図の記法(基本編)

ユースケース図は，ユースケース，アクター，システム境界，関連から構成されます．

アクター

アクターは，システム外部に存在するもの(ユーザや外部のシステム)です．アクターは図１のように人のような形で表現し，その下にアクターの名称を書きます．

図１ アクターの例

図１は，「スケジュール管理者」というアクターと，「時間割管理システム」というアクターを表しています．アクターは人のような形で表現しますが，人間だけでなく，外部のシステムなどもアクターとなりえますので，注意が必要です．

ユースケース

ユースケースは，図２のように楕円の中にユースケース名を記述することで表現します．

図２ ユースケースの例

ユースケースの名前は，それに関連付けられるアクターの立場に立って記述します．例えば，図２のユースケースは，ユーザがスケジュールを閲覧するので，「スケジュールを閲覧する」と記述しています．これに対して「スケジュールを表示する」と書いてしまうと，システムの立場にたった表現になり，不適切です．

関 連

関連は，ユースケースとアクターを結びつける実線で表記します．そのアクターがそのユースケースで表される機能を使用することを意味します．また，ある一つのシステムが持つユースケースを，実線の四角形で囲みます．これをシステム境界と呼びます．ユースケース図の例を図３に示します．

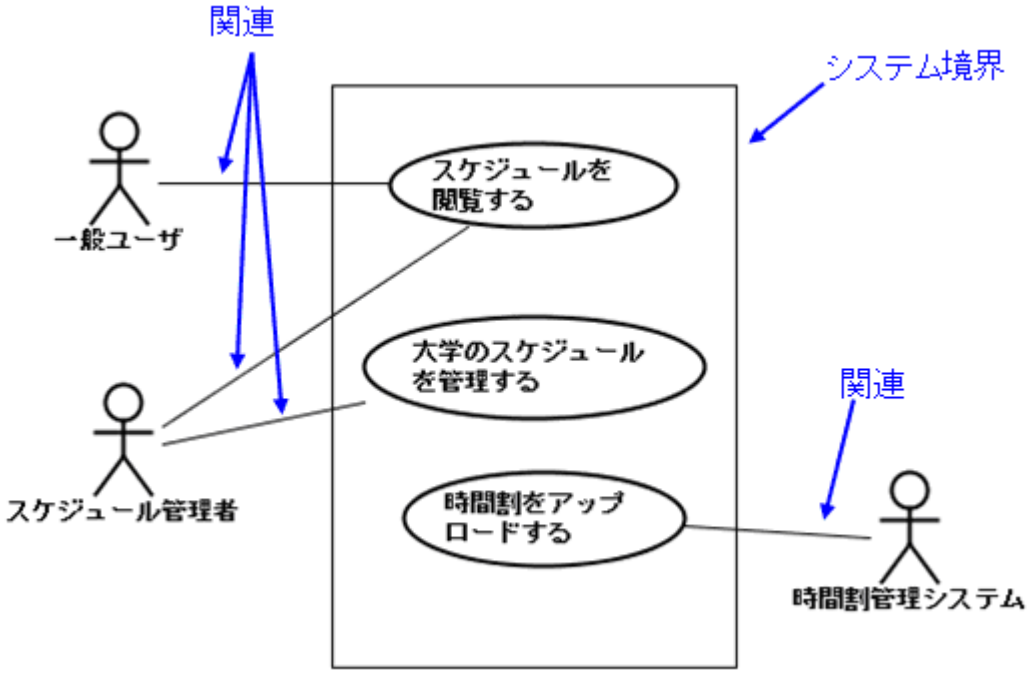


図３ スケジュール連絡システムのユースケース図の例

図３のユースケース図は，アクターとして「一般ユーザ」「スケジュール管理者」と外部システムの「時間割管理システム」があります．スケジュール連絡システムでは，外部の管理システムから，時間割データを取得してスケジュールに反映させます．このときのユースケースは，時間割管理システムの立場に立って「時間割をアップロードする」と記述されています．一般ユーザはスケジュールを閲覧することだけができ，スケジュール管理者は，スケジュールの管理とスケジュールの閲覧の両方ができます．

ユースケース図の記法(応用編)

ユースケース図の応用的記法として，「汎化」「包含」「拡張」について学びます．

汎化

先に出てきた図で示されるユースケース図では，スケジュール管理者は，一般ユーザが使える機能を使うことができ，それに加えてスケジュール管理の機能が使えます．このことは，スケジュール管理者が一般ユーザのうちで，特別な権限を持ったアクターであることを意味しています．

つまり，スケジュール管理者と一般ユーザの間に**is_a関係**(スケジュール管理者 is_a 一般ユーザ)が成り立ち，サブクラスとスーパークラスのように，スケジュール管理者を一般化したものが一般ユーザであると言えます．このような関係は，汎化で表すことができます．「汎化」は図４のように，白抜き三角の付いた実線で表現します．三角が付いている方がスーパークラスに相当するアクターになります．

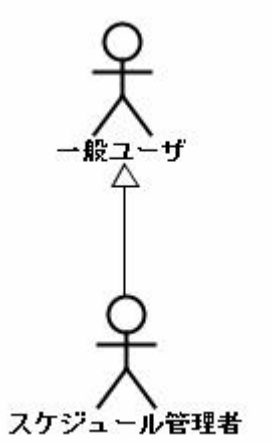


図４ 汎化の表記方法

先に出てきたユースケース図を，汎化を使って表現すると，図５のようになります．

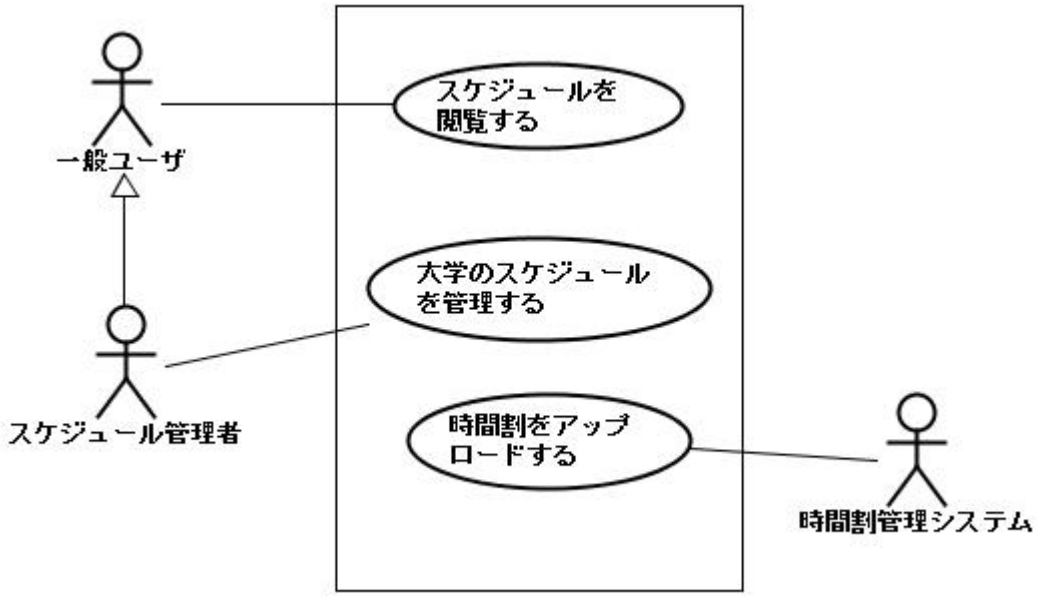


図５ 汎化を使って表したスケジュール連絡システムのユースケース図．図３と同じシステムを表現している．

上の例では，アクターの汎化関係を説明しましたが，ユースケースにも汎化の関係が存在します．例えば，スケジュール管理において，大学全体のスケジュールと研究室のスケジュールの管理があり得

るとき，汎化を使って図６のように表せます．

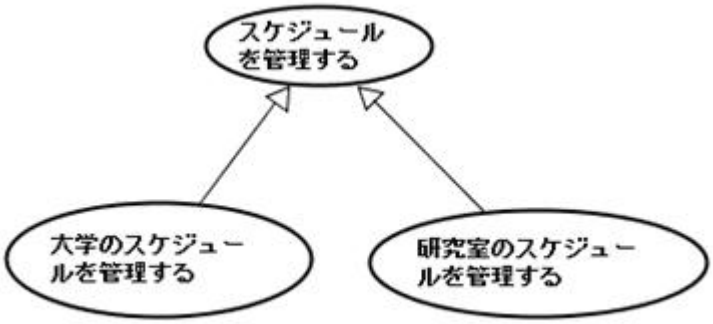


図６ ユースケースの汎化の例

包 含

あるユースケースが別のユースケースを含んでいるとき，包含の関係で表現します．包含しているユースケースから，包含されるユースケースに対して，点線の矢印を引き，<<include>>を付けて記述します．

例えば，大学のスケジュールを管理する場合には，必ず，管理者としてログインする必要があるとします．この場合，包含を使って図７のように記述できます．

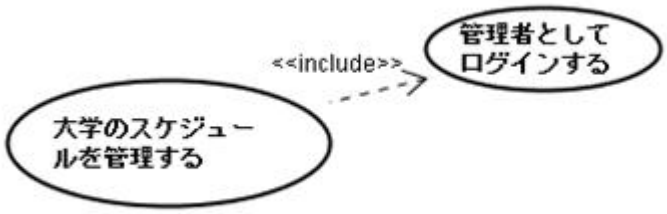


図７ ユースケースの包含の例

以下で説明する拡張との違いは，包含されるユースケースは，必ず利用されるということです．

拡 張

あるユースケースの一部を利用する場合と，利用しない場合があるとき，その部分を新たなユースケースとして作成し，もとのユースケースに対して拡張の関係で表します．拡張は <<extend>> を付けた点線の矢印で記述します．

例えば，大学のスケジュールを管理するというユースケースにおいて，スケジュールを入力する際に，キーボードから入力してもよいし，スケジュールをテキストファイルに入れておき，それをアップロードしてもよいとすると，拡張を使って，図８のように記述できます．

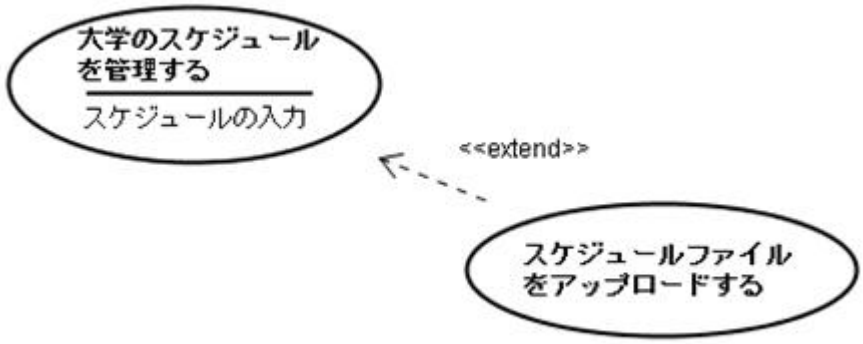


図８ ユースケースの拡張の例

図８では，拡張されるユースケース「大学のスケジュールを管理する」の下に実線を引き，「拡張点」として「スケジュールの入力」が記述されています．これは，大学のスケジュールを管理するというユースケースのスケジュールの入力時に，スケジュールファイルをアップロードするというユースケースが使われる場合があることを示しています．

包含との違いは，拡張を提供するユースケースは利用される場合とされない場合があってよいことです．

ユースケース図セルフテスト

次のページに移動して，「ユースケース図」に関するセルフテストを受験して，理解を確認しましょう．

UML記述ツール astah

この授業では，UMLモデリングツールastahを使用して演習を行います．astahには，いくつかのエディションがありますが，astah-communityは登録さえすれば，無償で利用できます．以下のサイトから入手することができます．

http://astah.change-vision.com/ja/product/astah-community.html

astahは上位互換ですので，新しいバージョンで作成したモデルは，古いバージョンでは読み込むことができません．逆に古いバージョンで作成したモデルは，新しいバージョンで読み込むことができます．常に新しいバージョンを使うようにしましょう．

astahは以前はJudeという名称のツールでした．Judeで作成したファイルも拡張子の.judeを.astaに変更すると，astahで読み込むことができます．

例題実習 astahを使ってユースケース図を描く

次のユースケースと同じものを，それぞれastahで描いてみましょう．

例題１ 基本的な記法のみのユースケース図

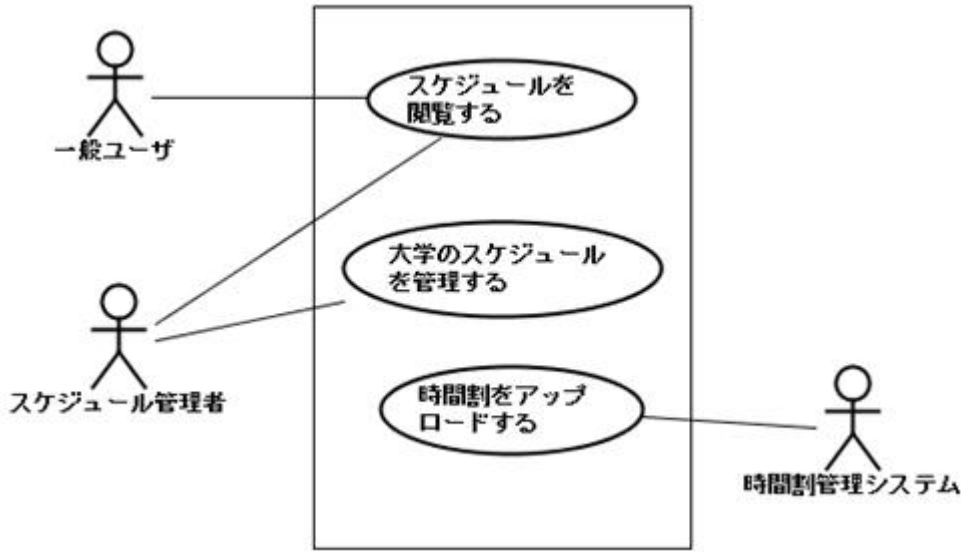


図９ 例題１

例題２ 複雑なユースケース図

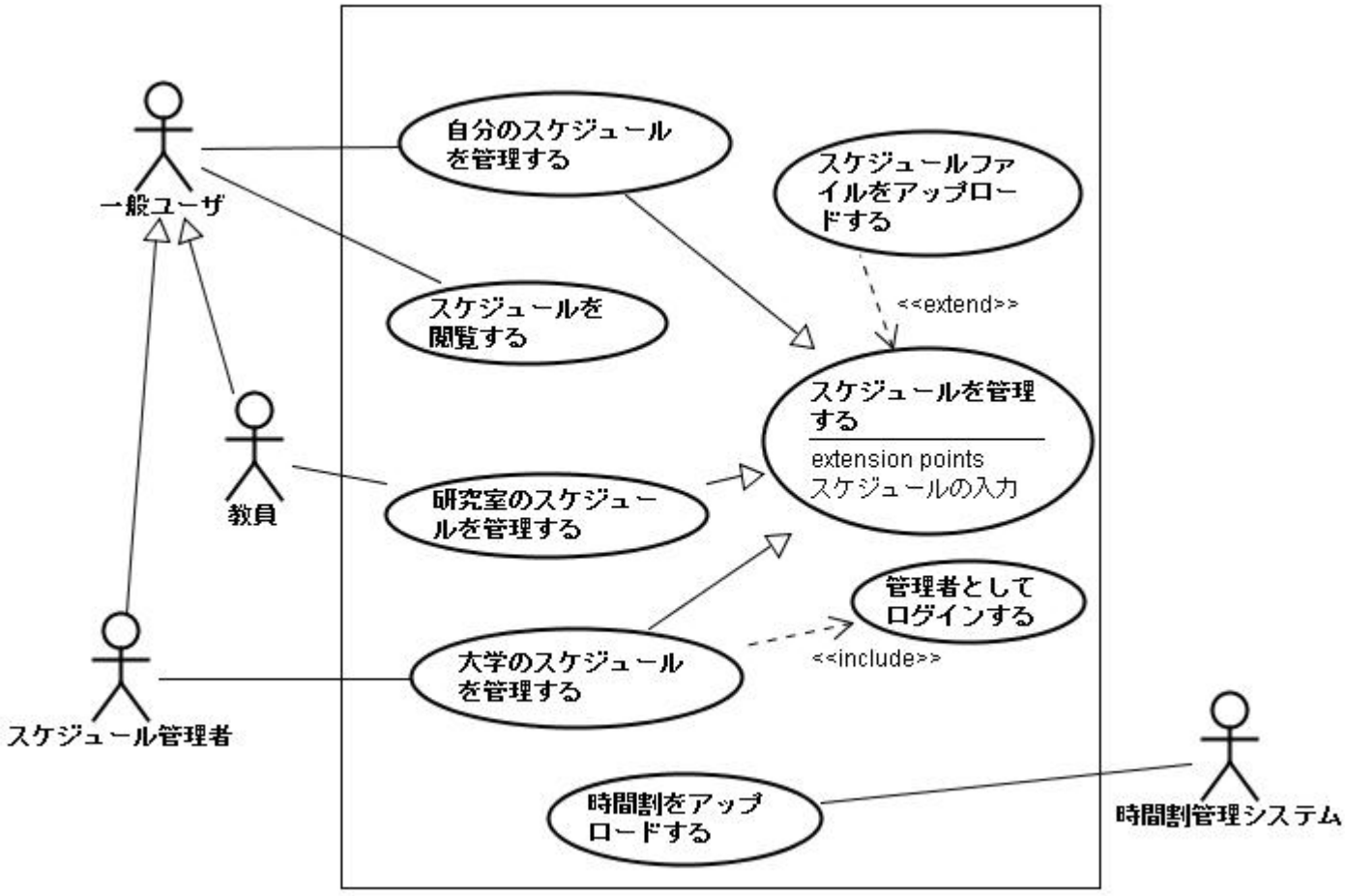


図１０ 例題２

それぞれのユースケース図の意味を確認しておきましょう．

上の図はastahがJudeというツールのときに描いた図なので， 全く同じにならなくてもよいです．

ユースケース図を描く際のポイント

ユースケース図は，実際に開発しようとする情報システムの概観を明らかにする際に記述します．ユースケース図を描くとき，まず，アクターとユースケースを洗い出し，それらに適切な名前を付けて関連付けます．ここで，名前の付け方がとても大切です．名前の付け方によって，意味の伝わり方や読みやすさが大きく異なります．

名前を付ける際に，以下の点に注意しましょう．

(1) 適切な抽象度にすること

あまりに抽象的な名前では，何を表しているのか適切に理解できません．アクターに「人」とか，「システム」といった名称を付けると役割がわかりませんし，ユースケースに「データをアップロードする」などという名前を付けると，何をアップロードするのかわかりません．一方，具体的過ぎても，理解しにくくなります．例えば，アクターに「渡辺さん」という担当の名前をつけても役割はわかりません．ユースケース名に「曜日・時限・科目名・履修者リストをアップロードする」というのも具体的過ぎます．

(2) 正確な名前にすること

例えば，「閲覧」や「管理」といったユースケース名では，何を閲覧するのか，何を管理するのかわかりませんから，ユースケース図の作成者の意図がきちんと伝わりません．ユースケース名は，「～を ～する」といった名称にするとよいと言われています．

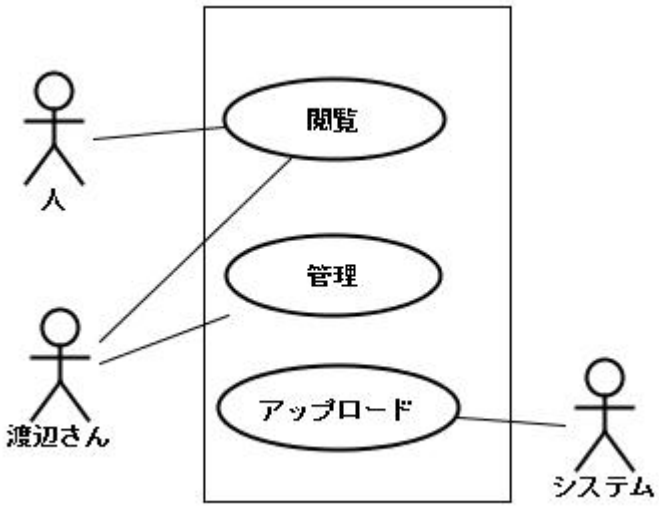


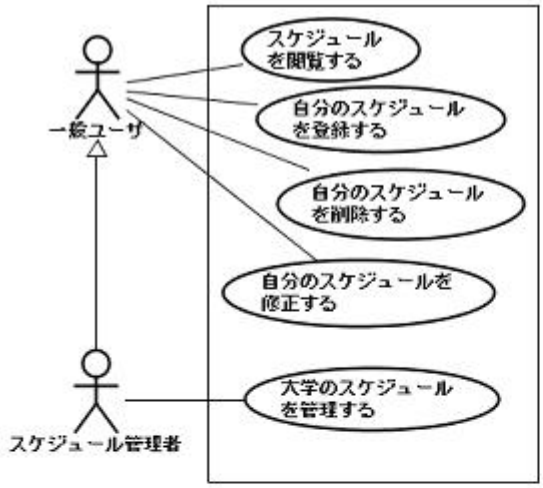
図１１ 名前の付け方が良くないユースケース図の例

(3) 表現を統一すること

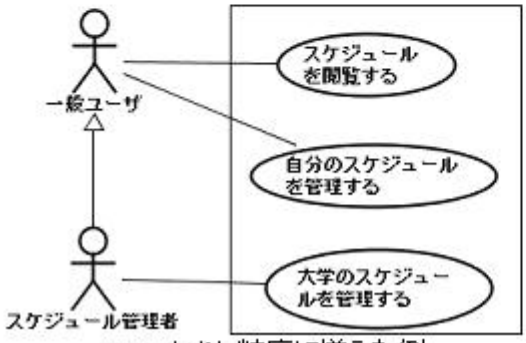
同じ意味に対して，複数の用語を使うと理解しにくくなります．例えば，「スケジュール」と「予定表」という用語で，同じものを指すのは避けなければなりません．同じ意味で使われるなら，どちらかに統一すべきです．

ユースケースを洗い出す場合，一つのユースケース図において粒度を揃えることも重要です．例えば，「スケジュールを管理する」というユースケースは，スケジュールの登録，削除，修正を含むため，大きな粒度のユースケースです．それに対して「スケジュールを登録する」や「スケジュールを削除する」は小さな粒度のユースケースになります．図１２(a)のように粒度が揃っていないのは，悪

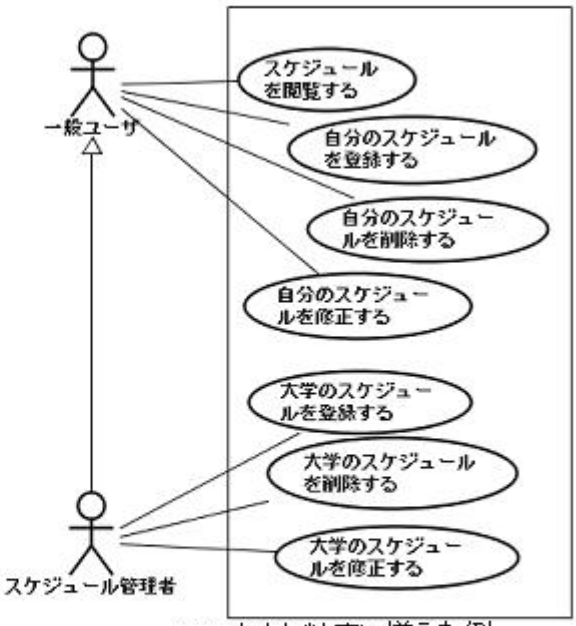
い例になります．



(a) 粒度が揃っていない例



(b) 大きな粒度に揃えた例



(c) 小さな粒度に揃えた例

粒度が小さすぎるのも問題です．「スケジュールを削除する」は，「スケジュールの一覧を表示する」，「削除するスケジュールを選択する」，「削除するスケジュールを確認する」，「スケジュールの削除を実行する」などと，さらに小さな粒度に分けることもできますが，これでは粒度が小さすぎます．

ユースケース記述の項目と例

ユースケースにはどのような項目を記述するか，また，その具体例を示します．

ユースケースの記述項目

情報システムのユースケースを表す場合に，ユースケース図だけでは，各ユースケースがどのようなものか，具体的にはわかりません．そこでユースケース記述を併用することが重要です．ユースケース記述には標準規格はありませんが，以下のような項目を記述します．

表 2 ユースケース記述の項目

ユースケース名		ユースケースについている名前．ユースケース図と揃える．
目的		ユースケースを実行する目的．
アクター		
このユースケースに関係するアクター．ユースケース図でユースケース図と関連で結ばれているアクターが該当する		
前提条件		このユースケースが開始される前提として整っていなければならない条件．
開始条件(起動トリガ)		このユースケースが開始されるきっかけ．
事後条件		このユースケースを実行することで満たされる条件．
イベントフロー	メインフロー	このユースケースが正常に進んだ場合に，アクターのやることシステムのやることを箇条書きで時間順に書く．
	代替フロー	エラーが起きた場合や，メインフローと違った処理を行わなければならない特別なケースのイベントフロー．代替フローを実行しても，事後条件は満たされる．
	例外フロー	ユースケースの実行を断念しなければならないようなケースのイベントフロー．例外フローを実行した場合，事後条件は満たされない．
備考		説明の補足．後工程作業者への申し送り事項など．
シナリオリスト		イベントフローの具体例（シナリオ）としてよくあるもののリスト
シナリオ記述		シナリオリストのシナリオを具体的に記述する．アクターやデータには具体的なインスタンスを用いる．

ユースケース名，代替フロー，例外フロー，シナリオ記述などには，必要に応じて，IDを付けておくと，対応付けが容易になります．

ユースケース記述の例

図１３のユースケース図のユースケース「大学のスケジュールを登録する」について，ユースケースの例を示します．

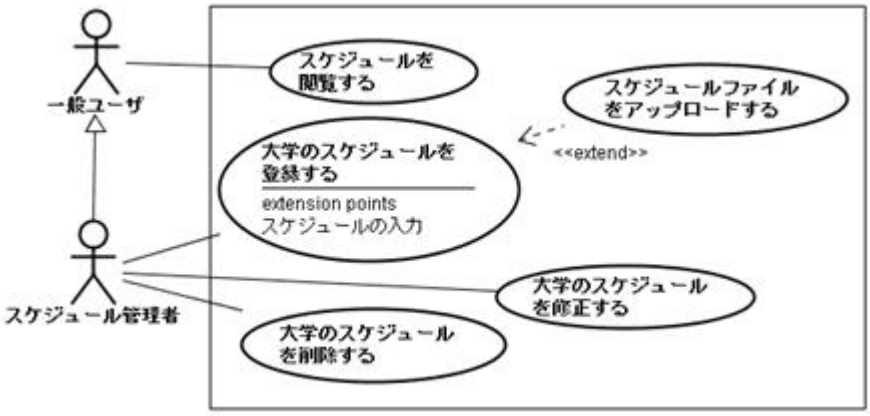


図１３ シンプルなスケジュール連絡システム ユースケース図

表３ ユースケース記述の例

ユースケース名		大学のスケジュールを登録する(UC002)
目的		大学のスケジュールの一覧に新しいスケジュールを追加すること
アクター		スケジュール管理者
前提条件		・システムが待機中であること
開始条件(起動トリガ)		・スケジュール管理者がスケジュール登録要求を出す
事後条件		・スケジュール一覧に，スケジュール管理者が追加を希望したスケジュールの内容が登録されていること
イベントフロー	メインフロー	1.アクターがシステムにスケジュール登録を要求する．
		2.システムはアクターにスケジュール情報の入力を求める．
		3.アクターは新しいスケジュール情報を入力する．(Alt-01)
		4.システムは入力されたスケジュール情報が有効なものか確認する．(Alt-02)
		5.システムはアクターに入力されたスケジュール情報を提示する．
		6.アクターは提示されたスケジュール情報を確認する．(Alt-03)
		7.システムは入力情報をスケジュール一覧に保存する．
	代替フロー	Alt-01: アクターがファイルからの入力を指定した場合
		1.アクターはファイルから入力したい場合，ファイルのアップロードをシステムに要求する．
		2.システムは「スケジュールファイルをアップロードする(UC005)を実行する．
		3.メインフロー3に戻る．
		Alt-02: アクターが入力したスケジュール情報が無効だった場合
		1.システムは入力されたスケジュール情報が無効な理由を提示する．
		2.メインフロー3に戻る．
		Alt-03: スケジュール情報の入力ミスをした場合
		3.システムは入力されたスケジュール情報の修正を要求する．

		4.アクターはスケジュール情報を修正する.
		5.メインフロー6に戻る.
	例外フロー	Ext-01: メインフロー6までにアクターが登録作業のキャンセルを要求した場合
		1.アクターは処理のキャンセル要求をする.
		2.システムは途中の情報を破棄する.
		3.システムはアクターにキャンセルが終了した旨を通知する.
		4.システムは待機状態に戻る
		事後条件：
		・登録時に入力途中の情報は破棄されていること.
		・アクターにキャンセルが終了した旨が通知されていること.
		・システムが待機状態に戻っていること
備考		特になし
シナリオリスト		UC002-01 大学のスケジュールを登録する場合(正常処理)
		UC002-02 ファイルからスケジュールを登録する場合
		UC002-03 入力したスケジュール情報に誤りがあった場合
		UC002-04 途中で入力作業をキャンセルした場合
シナリオ記述		UC002-01大学のスケジュールを登録する場合(正常処理)
		事前条件：スケジュール管理者(Aさん),
		登録するスケジュール情報(6月10日カレッジインターンシップ)
		1.Aさんが，システムにスケジュールの登録を要求する.
		2.システムはAさんにスケジュール情報の入力を求める.
		3.Aさんは6月10日カレッジインターンシップの情報を入力する.
		4.システムは入力されたスケジュール情報が有効なものか確認する.
		5.システムはAさんに入6月10日カレッジインターンシップの情報を提示する.
		6.Aさんは提示された情報を確認して，OKだったので承認する
		7.システムは6月10日カレッジインターンシップの情報をスケジュール一覧に保存する.

ユースケース記述の注意点

ユースケース記述を書く場合には，以下のようなことに注意しましょう．

厳密に書くこと．ユースケース記述はユーザ，開発者でイメージを共有するものですから，読んだ人がどのようにでも解釈できてしまう記述では困ります．ただし，厳密にといっても，細かくしすぎないことが重要です．特に開発に携わる人がユースケース記述を書くと，知らず知らずのうちに，システム内部の振る舞いを書いてしまうことがあるので注意が必要です．例えば，「アクターは入力欄に日付，内容を入力し，登録ボタンを押す」というのは細かすぎです．「アクターはスケジュール情報を入力する」程度でよいでしょう．また，「システムはスケジュール一覧テーブルから検索したレコードを表示する」というのも内部の処理内容を含んだ細かすぎる表現です．「システムはスケジュールの一覧を提示する」程度でよいでしょう．

漏れがないように書くこと．ユースケース記述に書かれていることを足し合わせたものがシステムになるので，漏れがないように書く必要があります．

空欄を作らないこと．ユースケース記述の項目のうち，まだ書く内容が決まっていない場合など T.B.D (To Be Determined；これから決まるという意味)などと 書いておいたり，特に書くべき事がなければ「なし」と書いたりして空欄がないようにします．

いつまでも書き続けないこと．厳密に書こうとするあまり，すべてを書かないと先に進めないといったことがないようにします．「ユースケース記述の終わりはいつまで立ってもやってこない」とも言われています．ある程度のユースケース記述をまとめた段階で，次のステップに進むことも重要です．

ユースケース図モデリング セルフテスト

次のページに移動して，「ユースケース図モデリング」に関するセルフテストを受験して，理解を確認しましょう.

まとめ

このモジュールでは，情報システム開発プロセスとオブジェクト指向を復習した上で，以下のことを学びました.

- 「UML(Unified Modeling Language)」は，主に情報システムのモデルを記述するための言語で，13のダイアグラムが規定されている.
- ユースケース図は，ユーザから見たシステムの機能を記述するもので，これによって開発するシステムに要求する機能を明確にすることができる.
- ユースケース図は，基本的には，ユースケース，アクター，関連，システム境界で構成される.
- 詳細なユースケース図を描くためには，汎化，包含，拡張を利用する.
- ユースケース図を描く際に，名前の付け方に注意し，表現を統一すること，粒度を揃えること，小さな粒度にしすぎないことなどに注意する.
- ユースケース図内の各ユースケースについて，名前，目的，前提条件と事後条件，イベントフロー，シナリオなどからなるユースケース記述を具体的に記述しておく.