



UML and project management

by SparxSystems Japan

UML とプロジェクトマネジメント



はじめに

このドキュメントは、2003年10月から11月にかけて弊社 Web サイトで限定公開されていた「UML プロジェクトマネジメント」の前後編をまとめたものである。

UMLとプロジェクトマネジメント(前編)

ソフトウェア開発の問題点

昨今のソフトウェア開発において、プロジェクトマネージャーを悩ませる問題点は数多くある。人間関係の問題・仕様変更の問題・納期の問題・費用の問題などいろいろあるが、これらの多くの問題を引き起こす問題のひとつとして、あいまいな仕様というのがあるだろう。

納期の直前になって、あるいは、納入した製品を実際にお客様が利用したときに、「こんな仕様ではない」とソフトウェアの修正・変更を要求されることもある。このような問題の原因はいろいろあり、お客様側の無理な要求という場合もあるが、お客様の要求を正確に把握できていない場合もあるだろう。

この要求の把握は、ソフトウェアを実装する前の設計の段階で、正確に把握されている必要がある。しかしながら、この段階で、例えば以下のようなあいまいさを残していることが、ソフトウェア開発の終盤になって問題を引き起こすことが多い。

[ログイン]

ユーザーは画面に ID とパスワードを入力してログインする
ID かパスワードが間違っている場合にはエラーを表示する

このような仕様の場合には、例えば、

- ID やパスワードを入力しない場合にはどうするのか？
- ID とパスワードが両方間違っている場合は？
- ID やパスワードの仕様(長さや使える文字列)は？
- 画面の構成は？
- エラーの内容は？
- エラーを表示した後はどうするのか？

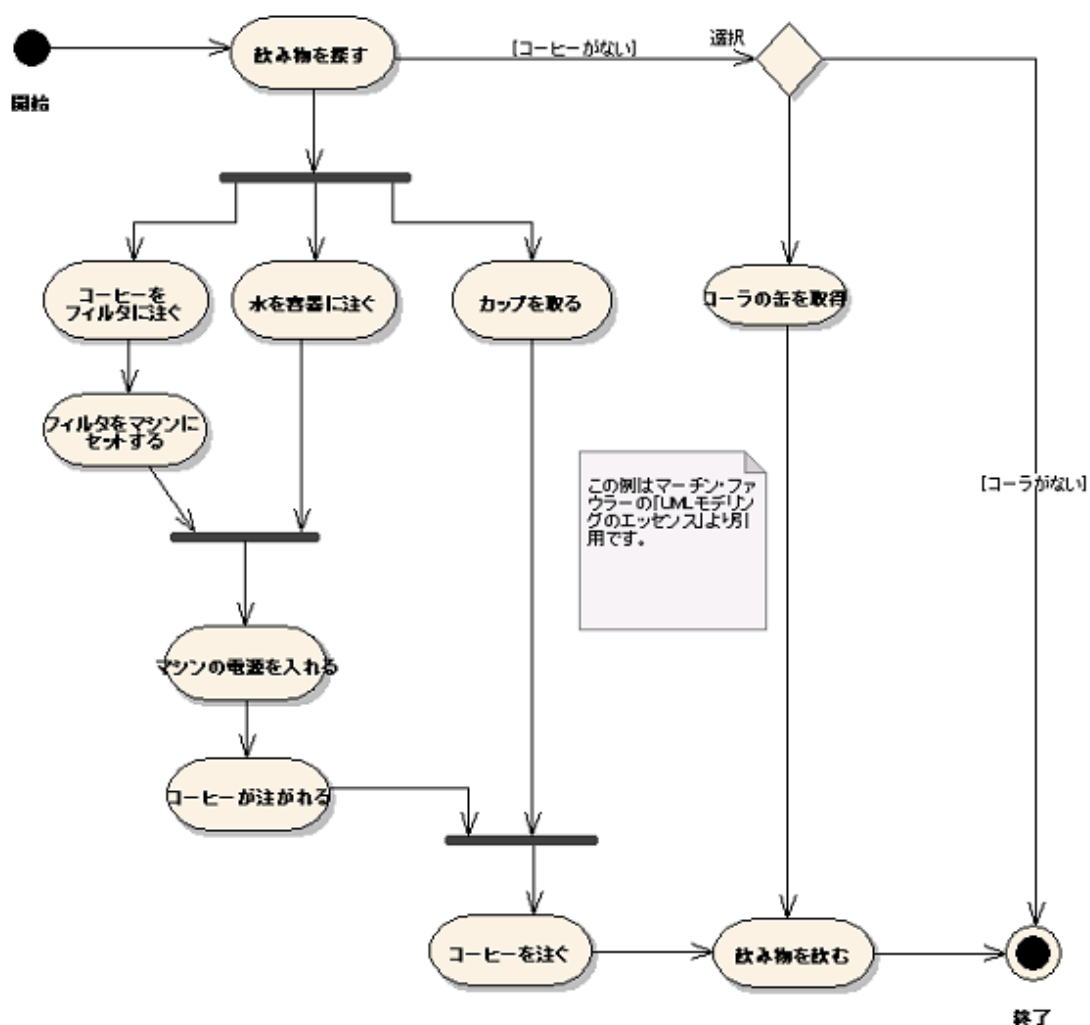
というような曖昧な点が残る。もちろん、これらのうちのいくつかは他の部分で定義されてい

ることあるだろうが、日本語という自然言語を利用して仕様を定義する限り、あいまいさをなくすことは非常に難しい。あいまいではない詳細な文章を書くことも不可能ではないが、それには時間がかかるし、その追加した文章にまたあいまいさが残る可能性が出てくる。

つまり、このような仕様検討・設計の段階では、あいまいさを持つ言語を利用する限り、このような問題は回避するのは難しいと言えるだろう。

UML

UML という言葉が最近多くの書籍や Web サイトで見られるようになった。UML とは、Unified Modeling Language の略で、統一モデリング言語と訳されることが多い。つまり、日本語と同じような「言語」である。どのような言語かという、下の例のような「絵」を中心として設計・開発する言語である。

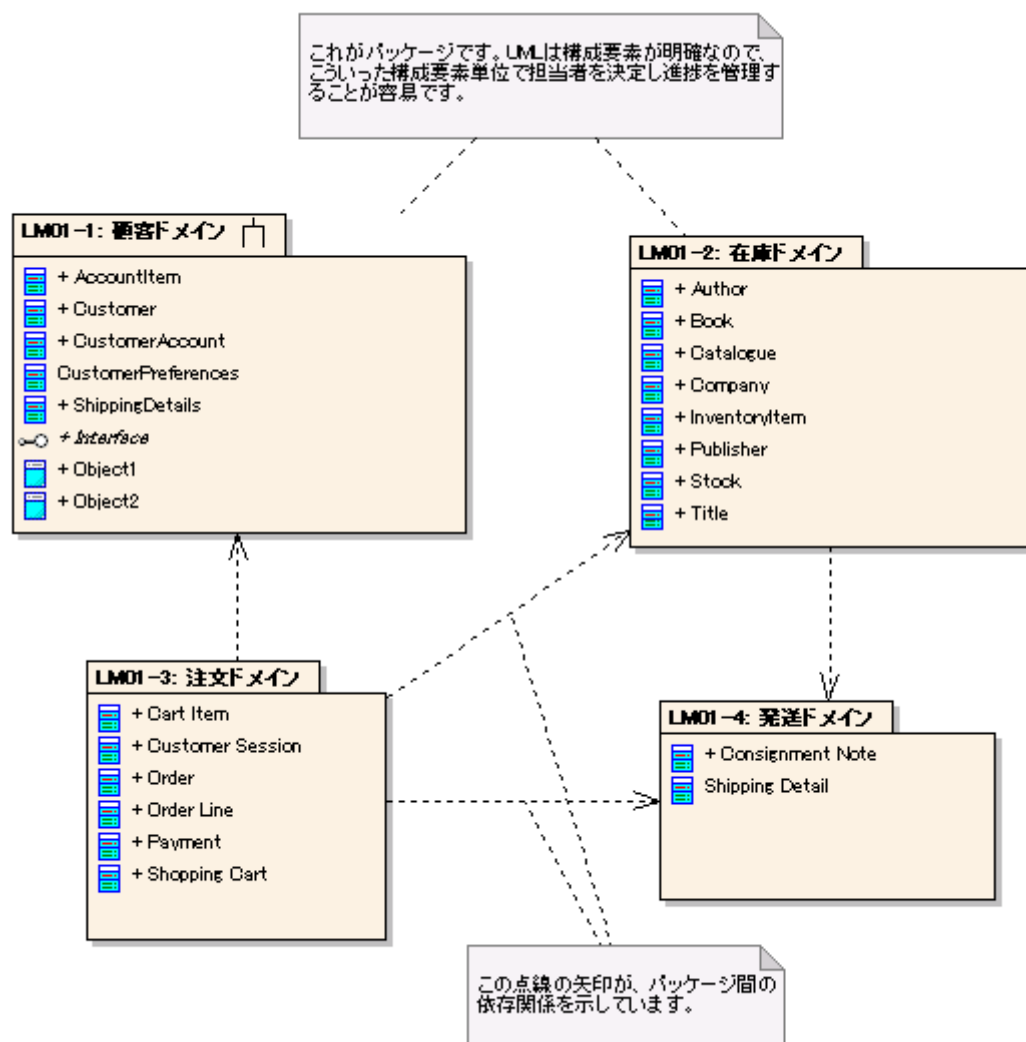


この例は、フローチャートによく似ている。つまり、UML は過去のソフトウェアの設計開発で利用されてきたいろいろなエッセンスをまとめて、厳密に定義したものである。もちろん、完全に厳密であるかという点、そうではない部分もある。しかし、少なくとも日本語のような言語で定義するよりは厳密に仕様・設計を行うことができる。

また、UML は「統一」モデリング言語の名のとおり、全世界で共通の言語を目指している。現在の多くの仕様書は、会社や部署ごとに構成や内容が異なり、記述されている内容の細かさもまちまちである。UML は、こういった現状の差異を吸収するひとつの選択肢であると言える。

プロジェクトマネジメントと UML

では、プロジェクトマネジメントにおいて UML をどのように生かすことができるか、という点について一例を簡単に紹介する。UML では、先ほどの例のような図や、パッケージと呼ばれる構成単位を利用してモデリング(設計)作業を行う。UML を利用した開発において作業分担や進捗管理を行う場合、これらの図やパッケージの単位で割り当てることができるようになるため、分担を明確にしやすいと言えるだろう。また、そういった図やパッケージの依存関係も図で示すことができるので、どの作業が遅れると誰に(どのパッケージに)影響が出るかという点についてもわかりやすい。



さらに、構成単位が明確になっているという点について言えば、詳細設計やテストの段階で問題が発生しても、問題の影響する箇所を明確にしやすいというメリットもあるだろう。依存関係がはっきりしているので、影響する範囲も見極めることができる。

UML が標準化された共通の言語であり、いろいろなソフトウェア開発に対応できる(いろいろな開発プロセスに適用できる)汎用性をもっているために、プロジェクトマネージャーとしては管理する単位を明確にしやすいし、経験を他のプロジェクトでも比較的容易に生かすことができる。UML を使うメリットは他にもいろいろあるが、こういった明確さや問題発生時の容易な追跡可能性はひとつのメリットであると言えるだろう。また、UML はインドなどの海外ではマスターしているエンジニアも多く、海外に発注する場合の誤解や混乱を防ぐこともできるだろう。

ただし、こういったメリットを享受するためには、開発メンバー全員が UML を理解・利用することと開発のライフサイクルを通して UML を利用し続けることが必須である。プロジェクトリーダー

ーだけが利用する、あるいはクラス設計だけに利用する、というような利用方法では、プロジェクトとしてはあまり意味がない。むしろ、UML で書かれたドキュメントが更新されなかったり、実際のソースコードと乖離したりして混乱を招くだけかもしれない。

最後に、日本語にいろいろな話し方(方言・口語)があるように、UML の使い方にもさまざまな方法がある。構成単位や依存関係が不明瞭な図を書くことももちろん可能なので、開発メンバーがUML の特徴を生かした図を書くように心がけないと意味がないというのは言うまでもない。

UMLとプロジェクトマネジメント(後編)

はじめに

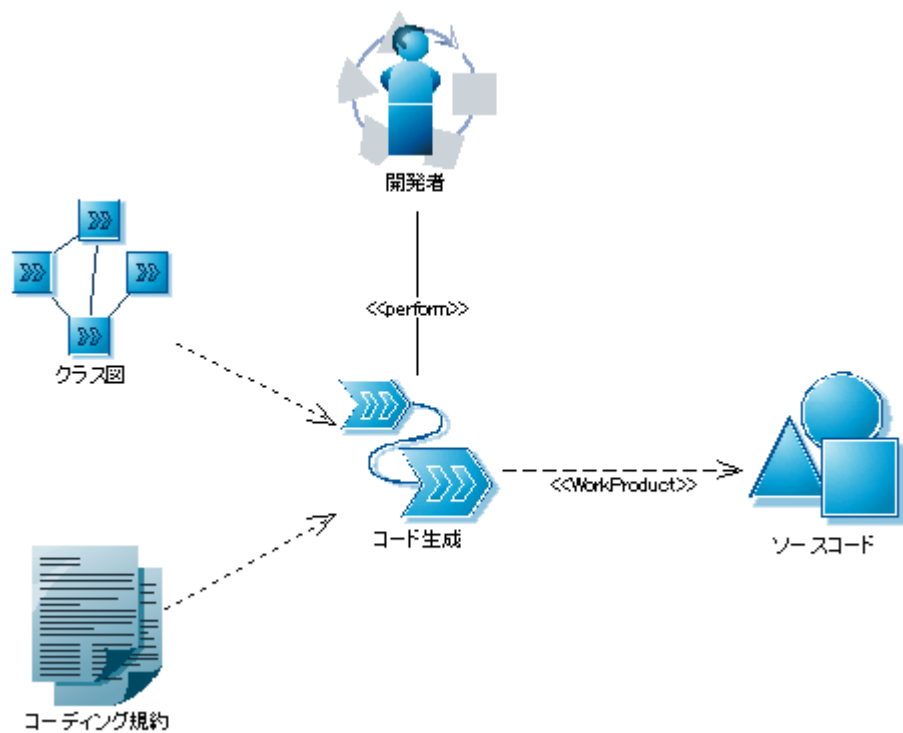
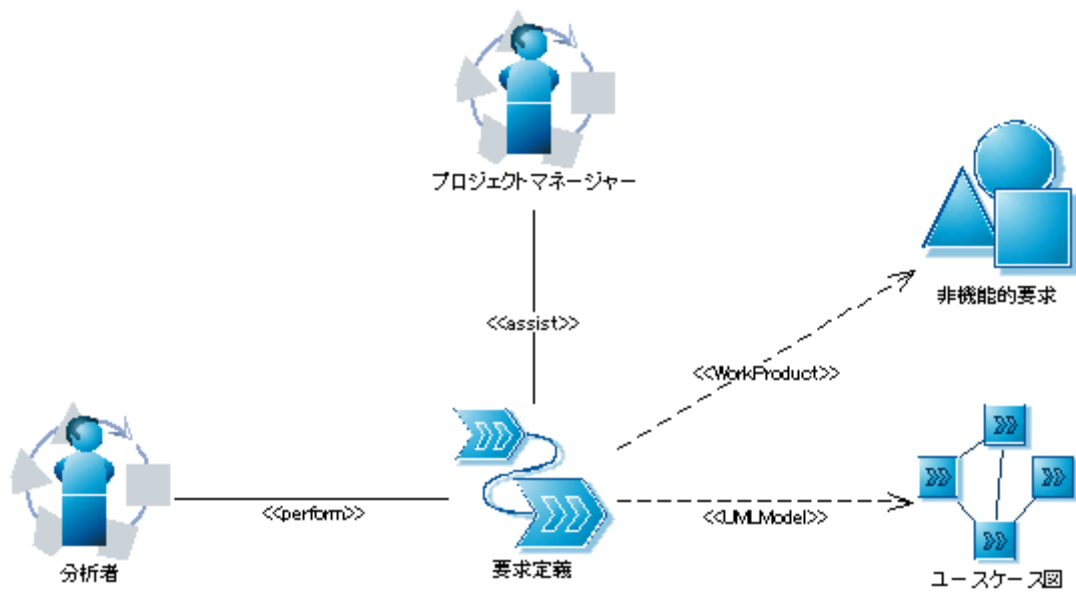
前編では、ソフトウェア開発における問題点の1つを例に、UMLを導入するとどんなメリットがあるのか、という点について紹介した。また、プロジェクトにUMLを導入する場合におけるメリットや問題点についても説明した。

後編では、オブジェクト指向・UMLを取り入れた開発において、どのようにプロジェクトマネジメントを行うかについて、3つの例を紹介する。UMLは汎用的な言語ゆえに、プロジェクトマネージャーにとっても便利なツールのひとつになる可能性を秘めている。特に、例のうち、最初の2つは、プロジェクトマネージャーにだけUMLを導入しても、それなりの成果をあげる可能性もあるので、ぜひ参考にしていきたい。

プロジェクト体制とプロセスフローのモデリング

開発プロジェクトが立ち上げられてから、最初にプロジェクトマネージャーが決定しなければならない点の一つとして、プロジェクトの体制の確立がある。どういったメンバーが必要で、どのような役割を負うのかということを明確に定義しておかないと、メンバーが最大の成果を出すのは難しい。また、無駄のある配置であればプロジェクトチーム全体の成果としては望めないし、逆に穴があってもいけない。

そういったプロジェクト体制の検討・策定にUMLを利用することもできる。ここで利用するのは、SPEM(Software Process Engineering Metamodel)と呼ばれるUMLのプロファイル(拡張のための定義)である。このSPEMを利用することで、UMLを利用してプロジェクトの体制やプロセスの流れなどを定義することができる。非常に簡単な例を2つ紹介する。



最初の例は、要求分析の流れの断片である。ここでは、分析者が責任を持って要求定義を行い、プロジェクトマネージャーは補佐する立場であることが定義されている。そして、成果物としてユースケース図と、ユースケース図では表現できない非機能的要求の文書があることが

定義されている。2 番目の例では、クラス図とコーディング規約から開発者がコード生成を行い、ソースコードを生成するという例である。

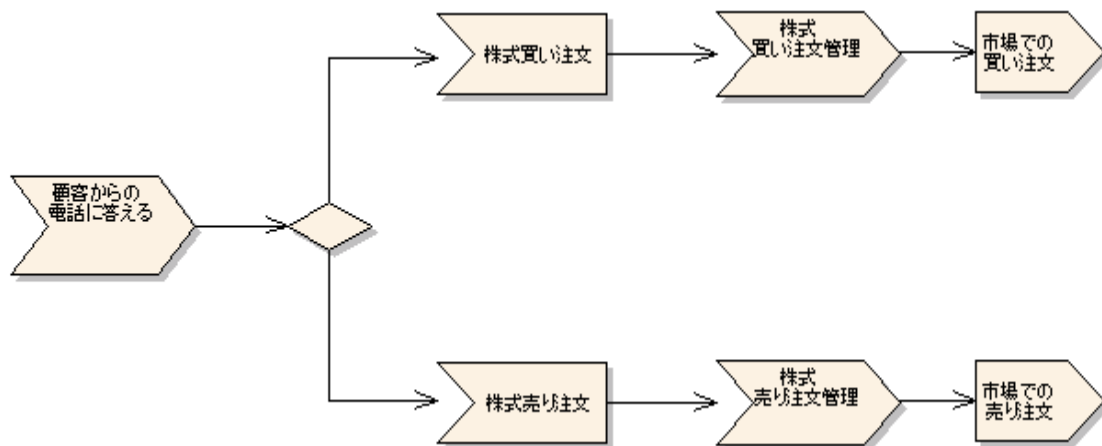
いずれの例も、単純な例ではあるが、おそらく説明をしなくても多くの人が図を見ただけで流れを理解できるのではないかと思う。この、全員が理解できる綿密な体制・プロセスの策定を日本語で行うのはなかなか難しい。しかし、UML を利用すれば、視覚で理解することができるようになる。また、成果物や作業が増えた場合でも、UML ツールを利用していれば対象の要素を追加して、矢印で結べば作業完了である。プロジェクトマネージャーの工数を無駄に割く必要はない。

ビジネスモデリング

開発プロジェクトの初期の作業として、多くの場合には顧客の要求をヒアリングして、作成するソフトウェアの仕様を定義する必要があるだろう。顧客の側では既にも実働しているビジネスモデルや、あるいは理想とするビジネスモデルがある場合が多く、このモデルに合致するようなソフトウェアを作成する必要がある。この段階で顧客のモデルや要求を正確に把握しておかないと、最悪の場合には納入の段階になって修正作業を強いられることもあるだろう。あるいは、実際に導入してみたら日々の作業にマッチせず、無駄なソフトウェアとなる場合もあるだろう。このモデルや要求の定義をあいまいな解釈が可能である日本語で定義している場合には、前編でも述べたような問題が発生する余地がある。この段階の定義に UML を利用すれば、そういった問題を避けることも可能になるだろう。

もちろん、顧客側が UML を知らないという場合も多いだろう。しかし、UML のダイアグラムは直感的でわかりやすいので、ある程度の補足を加えればすぐに理解できるようになる場合が多いだろう。将来的には、UML がより広まっていけば、この段階での UML の利用は加速されると想定される。

このビジネスモデルの定義にも、もちろん UML が利用できる。この分野の内容を扱った書籍として、「UML によるビジネスモデリング」がある。ここでは、この書籍から引用した図(p91 図 3.8)を紹介する。



この図は、株式の取引の流れを定義した図である。顧客からの要求に応じて、買い注文あるいは売り注文を出すということがわかる。この図で明らかになることは、買い注文と同時に売り注文、のような場合や、定義されていないそれ以外の要求がないということである。日本語の文章で、「顧客からの電話で、買い注文の場合には市場に買い注文を発行し、売り注文の場合には市場に売り注文を発行します」というような定義をすると、こういった書かれていない範囲については明確にならない。もちろん、日本語の文章で「これ以外の処理はない」というような補足を追加することも可能であるが、すべての場合にそれを行うのは難しく、あるいは手間がかかるだろう。

また、既存のビジネスの改善・変更を行うような場合でも、UML による分析は有効である。図化することで、重複や似ている箇所の発見が容易になるなどのメリットがある。

パッケージ単位の作業割り当て・進捗管理

最後に、前編で説明したような、パッケージ単位での作業の割り当てである。もちろん、作業の割り当てはどのプロジェクトでも行われていることであるが、その作業(パッケージ)の依存関係なども UML で定義できるので、UML で一元化して定義・管理することですべてのメンバーが把握できるようになるだろう。ここで、UML ツールによってはパッケージや要素単位で担当者を割り振ることができたり、進捗情報を入力・参照したりできるので、個々のメンバーが進捗情報も把握できるようになる。これは、最近話題のアジャイル(俊敏)なソフトウェア開発においては、大きな助けになるだろう。

特に、終盤の詳細設計から実装にかけては、クラス図のクラスとソースコードのファイルが 1 対 1 で対応することが多い。こういった場合に、UML のクラス図のクラスに進捗状況を記入しておくことで、全体の進捗度合いがクラス図の中だけで完結できるようになる。また、テスト項目も、

単体テストはクラス図のクラスと、機能テストはユースケース図のユースケースと対応させておく
と良いだろう。

最後に

UML を利用したソフトウェア開発プロジェクト、あるいはプロジェクトマネジメントにおける
UML の使い方の例を紹介した。これらの例はあくまでも例であり、UML の使われ方は他にもい
ろいろとあるだろう。これは、UML が「言語」であり、その言語を利用して何を表現するかという
のは、プロジェクトマネージャーの考え次第だからである。

今後、UML がソフトウェア開発全体に徐々に浸透していくと想定されるので、今のうちにい
ろいろな利用方法を検討しておくのが得策だろう。

付録:関連リンク

UMLの関連書籍

http://www.sparxsystems.jp/uml_books.htm

Enterprise ArchitectのSPEMプロファイルについて

http://www.sparxsystems.jp/spem_profile.htm