第4章

検証を考慮して仕様/設計文書を書く

── UMLを利用する際に押さえておきたいこと

伊藤昌夫

ここではUMLによる仕様/設計文書の書きかたについて述べる．UMLを利用する場合，各図の間にはそれぞれ関係があり，その一貫性を維持しながら仕様を表現しなければならない．そのため，いくつかの配慮が必要になる．ここでは簡単なモデルを示しながら，UMLで仕様を記述する際の注意点を紹介する．また，リアルタイム・システムを表現するための拡張記述についても紹介する．（編集部）

今回，ソフトウェア設計品質ということについていろいろと考えてみました．わかっているつもり，すなわち設計については十分に経験を積んでいるつもりだったのですが，あらためて「設計品質」ということばを抜き出すと，説明がとても難しいことに気が付きました．機械の場合と違って，「設計結果図面ができたら，あとは製造の話」とはいきません．かなりの点で設計と（製造に相当する？）プログラミングが密接に結びついているからです．機械設計だと図面どおりに作るのが製造ですが，ソフトウェアの場合，そこまで図面を書こうとすると，プログラミングを行っているのとあまり変わらなくなってしまいます．したがって，「いかに抽象化するか」というのがソフトウェアの場合は大きな課題になります．

いかに抽象化するかという問題は，どのように仕様を書き，設計するかという問題でもあります．そこにはどうしても分析・設計手法が関係してきます．ここでは「オブジェクト間の協調を中心とする分析・設計」を考えて，その中で最終的に設計品質をどうとらえればよいかを説明したいと思います．道具としては，UMLの図式とその拡張を使うことにします．

● 仕様を決めることが難しい二つの理由

ソフトウェア設計において，仕様をどう記述するか，設計をどのように記述するかはつねに問題となってきました．なぜかというと，多くの場合，システムは人間のために存在するからです．人間が直接ハードウェアを操作する時代，それは単純なインターフェースしか持たず，人間はその数少ないインターフェースのみを利用していました．すなわち，人間がハードウェアに合わせていたと言えます（**図1**）．

現在では，よほどの単純なシステム以外は，例えばキーボードのようなインターフェースからなる入力機器を利用したり，ディスプレイの画面などを通じて複雑な出力を行うことになります．これにより，柔軟なユーザ・インターフェースを実現しています．

近年，ユーザの使いがってを改善するため，あるいはハードウェア自身が実現していない機能を受け持つため，ソフトウェアの規模が非常に大きくなっています．このことを仕様という側面で見たとき，以下の二つのことが言えます．

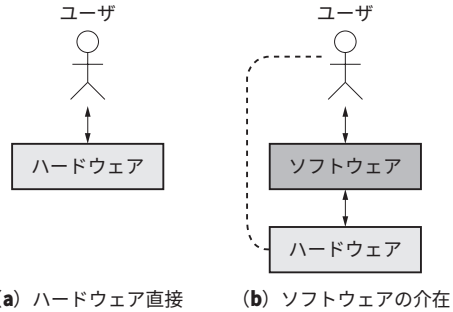

図1　人間とハードウェアの間へのソフトウェアの介在を示す模式図

ソフトウェアが媒介しないとき，(a)のようにユーザは基本的にハードウェアの提供する少数のインターフェースのみを使っていた．現在では，間にソフトウェアが介在することで，(b)のようにより柔軟なインターフェースおよびハード・ワイヤ化されないロジック（論理機能）の実現が可能となっている．図の矢印は主たるやりとりを示している．

- 人間系については容易に仕様が定まらない
- 機能が複雑化・大規模化しており，仕様として表現することが難しくなっている

● 仕様，設計，検証の関係

SLCP（Software Life Cycle Process）によれば，ソフトウェア開発の上流工程では，次のような作業が行われることになります．

- **ソフトウェア要求分析**
- **ソフトウェア・アーキテクチャ設計**
- **ソフトウェア詳細設計**

組織によって，あるいはプロジェクトによって，活動はより詳細化されますが，ここではおおむねこの分けかたを採用します．

今，ソフトウェア要求分析の結果としてソフトウェア要求仕様書（Software Requirement Specification，略してSRSと呼ぶこともある）があるものとします．また，ソフトウェア・アーキテクチャ設計の結果としてソフトウェア概要設計書が，ソフトウェア詳細設計の結果としてソフトウェア詳細設計書が作られるものとします．このとき，それぞれの文書の間には**図2**のような関係があると考えられます．この詳細化が正しいかどうかを調べる行為が「検証」と呼ばれています．例えば，ソフトウェア要求仕様書に書かれていることが，誤りなく，過不足なく，かつ一貫性を維持しながらソフトウェア概要設計書に記述されているかどうかを検査することになります．したがって，次のように言い換えることも可能です．「詳細化は開発として前進する方向で行われる行為である．それに対して，検証は逆向きに行われる行為である」．

ここまではわかりやすい話なのですが，実際にはそれほど明確に仕様と設計を分割することはできません．ソフトウェア要求仕様書を考えるとき，ある程度設計のことも考えなくてはなりません．そうしないと，実現しようのない要求仕様になるかもしれないからです．逆に，あらゆることを規定しようとすると，プログラム仕様書のようになってしまいます（例えば，エラー・メッセージの文面を決めるなど）．

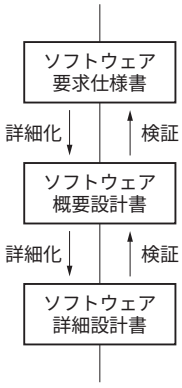

図2
詳細化の過程
下位の文書は上位の文書を「詳細化する」という言いかたをする．例えば，ATMで現金を引き出すという要求仕様書上の機能に対して，概要設計書を読むことによって，どのような構成部品の関係でそれが実現可能かを知ることができる．あるいは，キャッシュ・カードを入出力する部分や，パスワードを確認する部分といった構成部品がわかる．詳細設計書を読むと，それぞれの構成部品の内部がどのように作られているかがわかる．

ここでは，いくつかのキーワードに関してSyntropy手法[2]に従って定義しておきます．

- **「分析」とは，ソフトウェア開発プロジェクトにとって選択の余地のない（すなわち実現することを明示している）側面を見つけ，記述することである**
- **「設計」とは，ソフトウェア開発プロジェクトにとって選択可能な（すなわちまだ何も述べていない）側面を作り出し，記述することである**
- **「仕様」とは，ソフトウェア・システムの観測可能なふるまいの抽象的な記述である**

分析の結果が仕様であるとすると，この分析の定義にあるように，あくまでも（観測可能なシステムについて）実現すると宣言しているものが仕様の候補となります．したがって，現実的には，顧客によって仕様のレベルというのは随分と異なることになります．ただし，ここでは上記の“抽象的”というキーワードを強調しておきたいと思います．

● 良いソフトウェア要求仕様書とは？

仕様については，もう一つ参考になる記述があります．IEEE標準であるソフトウェア要求仕様に関する推奨です[3]．ここには，良いSRSの特性について次のような定義があり，重要な要素が明確に述べられています．

- **正確である**――正確であるとは，そこで述べられている要求が実現するべきもの（通常は，“システム”要求仕様書のような上位文書）と適合していること．
- **明白である**――ソフトウェア要求仕様書が明白であるというのは，そこで述べられている要求がたった一つの解釈しか持たないこと（用語が複数の意味を持つ場合は，用語集を通して意味を限定する必要がある．あるいは形式的な記述を用いてユニークな意味を与える）．
- **完全である**――完全であるとは次の三つの項目が成立するときである．第1に，すべての重要な要求（機能性や性能，設計制約，属性，外部インターフェース）が記述されていること．なお，システム要求仕様で定義されて