ソフトウェア仕様化／設計法のデータベースの試作

6S−3

井口和久　篠原正紀　佐伯元司

東京工業大学 工学部

1 はじめに

現在までに様々なソフトウェア仕様化／設計法が提案されているが、あらゆる分野のソフトウェアシステムに対して1つの有効な設計法はない。設計するソフトウェアの部分に応じてそれぞれに適した設計法を選択して記述したり、複数人による仕様化／設計作業で各人が異なる設計法で作業しても最終的に1つの仕様に統合することができれば、より効率的にソフトウェアの仕様化／設計作業が進められるであろう。こうした作業を支援するには、従来の支援ツール以外に設計法自身をデータとして持つデータベースが必要である。このようなデータベースを含むツールをメソッドベース [1] と呼び、本研究ではこのメソッドベースの試作を行なった。

2 メソッドベースの概略

異なる設計法による設計作業のガイドや、異なる設計法によって記述された部分的な仕様を統合するために、様々な設計法を共通の枠組で形式化して蓄積する。この枠組 (設計法の論理データモデル) をメタモデルという。メタモデルに従って記述した設計法を、その設計法のオブジェクトモデルという (図1)。設計作業はこのオブジェクトモデルに従ってガイドされる。

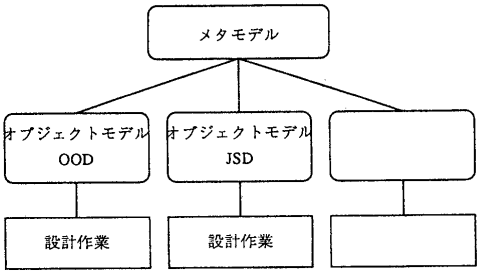

図 1: メソッドベースの論理的構造

2.1 メタモデルとオブジェクトモデル

メタモデルは、様々な設計法が共通に持っている概念を抽出し構成したもので、設計法を記述する言語の役割を果す。メタモデルは、生成されるものの構造:product の視点から構成したものと、作業手順:activity の視点から構成したものの2つからなっており、Entity-Relationship モデルで記述されている。product の部分は、例えばシステムの構造を表す Entity、機能を表わす Process や Data などの entity と、Entity 間の関係を示す EntityEntity などの relationship からなる。activity の部分は作業とその順序からなり、この activity 部分と product 部分は各作業における入出力プロダクトが何であるかを表わす relationship input, output によって結合される。

オブジェクトモデルは、その設計法の概念を、メタモデルに用意されている共通概念と対応付けることによって作られる。オブジェクトモデルにおける1つの概念が、メタモデルで複数の概念に対応することもある。

例えば、OOD[2] のオブジェクトモデルは、図2、3のようになる。図で括弧に書いてあるものがメタモデルにおける概念であり、例えば OOD における概念 Object は、メタモデルにおける共通概念 Entity と対応付けられている。

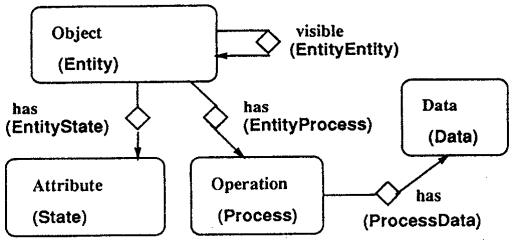

図 2: OOD のオブジェクトモデル (product)

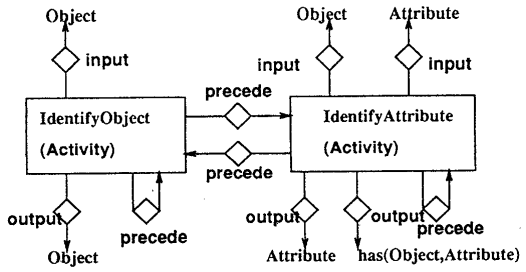

図 3: OOD のオブジェクトモデル (activity)(一部)

2.2 仕様の統合

異なる設計法で記述された (部分的な) 仕様の統合を行なうときには、メタモデル上で同じ共通概念に属するプロダクトを仲介して統合する。例えば図4で、設計法 A を用いて作られた部分仕様 a と設計法 B による部分仕様 b があるとする。

A Prototyping of a Database of Software Specification / Design Methods

Kazuhisa IGUCHI, Masaki SHINOHARA, Motoshi SAEKI

Faculty of Engennering, Tokyo Institute of Technology

この２つの部分仕様に、同じ名前を持つプロダクト c があり、設計法 A では概念 α に属し設計法 B では概念 β に属するとする。この α と β が、メタモデルで同じ共通概念 F に属するとき、２つの部分仕様はプロダクト c で統合される。統合された仕様を読む時には、プロダクト c を境界として a は設計法 A の観点から、b は B の観点から参照・操作することになる。設計法の形式化にメタモデルという共通概念を使うことにより、設計法のプロダクト間の対応関係の記述が容易に行える。

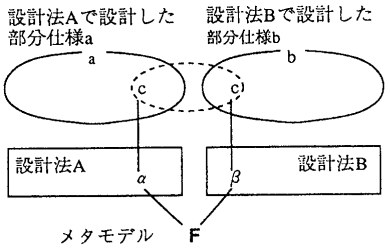

図 4: 仕様の統合

3 メソッドベースのプロトタイプ

上記のようなメソッドベースの基本機能を実現するプロトタイプを UNIX-WS 上に試作した。メタモデルはプログラム中にデータとして埋めこまれている。メソッドベースはオブジェクトモデルの activity 部分のうち precede、input、output を解釈し、作業のガイドを行う。ある作業での input データを人間に示し、output データを人間から受け取る。１つの作業が終了すると、relationship precede に従って次に行うべき作業を提示する。つまり precede で指定された順に従って作業が進められる。作業の input, output データを基にした仕様各部の依存関係も保持されるため、仕様変更時の波及範囲も解る。

図 5は前記の OOD のオブジェクトモデルをスクリプトにしてメソッドベースに読み込み、作業者のガイドを行なっている例である。図 3と同じ部分の作業であり、太字の部分が人間の入力 (作業の出力) である。

4 実用的なメソッドベース

実用的なメソッドベースは、サーバーとクライアントに分れて構築される (図 6)。サーバー部は、オブジェクトモデルの管理と設計作業中に得られたプロダクトや作業履歴の管理及び設計作業のガイドを行なう。クライアント部は主にユーザーインターフェースを実現する。サーバー側はメタモデルに従って記述されたオブジェクトモデルのみを持っているので設計法特有の図表現などが失なわれてしまっている。例えば DFD などは２項関係の集合として扱われることになる。設計法それぞれにクライアントを用意し、図表現とオブジェクトモデルとの変換を行うことで各々の設計法固有の表現法を用いた設計作業を進めることができる。また複数のクライアントを同時に使うことで、複数人による仕様化／設計作業にも使用できる。

activity:IdentifyObject	作業名
input: Object={}	入力 (最初なので無い)
Object: **lift**	出力
select next activity	次に行うべき作業
0: IdentifyObject	
1: IdentifyAttribute	
—:**1**	作業 1 を選択
activity:IdentifyAttribute	作業名
input: Object={lift}	作業の結果作られたもの
input: Attribute={}	(次の作業の入力の候補)
output 0: Attribute	この作業での出力の候補
output 1: has(Object,Attribute)	
select:**0**	出力 0 を選択
Attribute:**currentFloor**	出力
output 0: Attribute	
output 1: has(Object,Attribute)	
select:**1**	出力 1 を選択
has(Object,Attribute)	
has:**has**	
Object(id):**0**	
Attribute(id):**1**	

図 5: OOD による作業例 (一部)

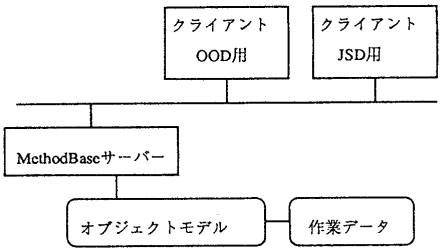

図 6: メソッドベース

5 おわりに

本研究では、ソフトウェア仕様化／設計作業の効率化を計るために、仕様化／設計法のデータベースである、メソッドベースの構造の考案とそのプロトタイプを試作した。今後はより多くの設計法の形式化を試み、メタモデルの洗練を行ない、またサーバークライアント型のメソッドベースを製作する。

参考文献

[1] 郭文音, 佐伯元司. ソフトウェア仕様化／設計法のデータベース化について. 情報処理学会ソフトウェア工学研究会, Vol. 92, No. 85, pp. 39–46, 1992.

[2] Grady Booch. *Software Components with Ada.* The Benjamin/Cummings Publishing Company, 1986.

[3] Sjaak Brinkkemper. *Formalisation of Information Systems Modelling.* Thesis Publishers, 1990.