ソフトウェアにおけるモデル化の意味

玉 井 哲 雄†

ソフトウェア工学全般で，モデルという言葉は偏愛されている．ここではとくに要求分析過程で頻繁に使われるモデルという概念を改めて考え直すとともに，モデルの記述形式として共通に使われるグラフ構造という観点からモデル化技法を整理し，おびただしい数のモデル化技法の存在が起しがちな混乱について議論する．

What do we mean by Modeling in the world of software engineering?

Tetsuo Tamai†

1. はじめに

ソフトウェア工学の分野では，モデルという言葉が大変好まれる．鳥居等による ICSE の過去の論文タイトルに現れる用語の頻度調査では，model は上から数えて 4 番目だという [2]．しかも他の用語は 20 年の間にはやりすたりがあるのに対し，モデルは一貫してよく使われてきている．

その理由の一つとして，モデルという言葉は，単独で使われるより他の用語との組み合わせで使われることが多いことが挙げられよう．たとえば，ライフサイクルモデル，システムモデル，設計モデル，分析モデル，プロセスモデル，領域モデル，オブジェクトモデル，計算モデル，データモデル，データフローモデル，実体関連モデル，状態遷移モデル，というように．組み合わせの相手がすたれていっても，モデルはパートナーを代えて生き続けるわけである．

それだけに，濫用されているというきらいもないではない．ここではとくに要求分析の場面で使われる「モデル」の概念について，改めて若干の考察を行うとともに，モデル記述に一般的によく用いられるグラフ構造との関連から，数多く提案されているモデル記法について考えてみたい．

† 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, University of Tokyo

2. M. Jackson によるモデルの概念

M. Jackson がその著書 [1] で取り上げている「モデル」についての議論は，やはり示唆に富む．まず Jackson は数学や論理学でいうモデルと，ソフトウェア工学でいうモデルとでは，現実世界と抽象世界との関係がまったく逆になっていることを指摘する．確かに，数学や論理システムでは理論に対しその実現となっているものをモデルという．逆にソフトウェアの世界では，現実世界の問題領域を抽象化しなんらかの記述体系で表したものをモデルというのが普通である．この指摘は重要である．しかもソフトウェア工学の主要な一角を占めつつある形式手法は形式論理学と重なるから，混乱を引き起こす可能性は高い．

ソフトウェア技術者は論理学で使うような意味でモデルという語を用いてはいけないと Jackson は主張するが，さらに単なる抽象的記述をモデルと呼ぶのも疑問としている．抽象的な記述は，単に記述と呼べばいいのではないかというのである．Jackson の立場は，それ自身が意味のある物理的存在で，対象となる現実世界の一部に対し類似の動作をするものをモデルと呼びたいというものである．たとえば，管のネットワークを流れる流体の動作を電気回路で模したとすれば，その電気回路は物理的な実体であり，かつ流体という領域と類似するモデルになっている．計算機の中に作られるモデルは，物理的に実在するものであり，同時に実世界に模した動作をするので，まさに Jackson の意味するモデルになっているという．

表 1 種々のモデルのグラフ構造

	頂点	辺
データフロー	プロセス	データの流れ
ＥＲ	実体	関連
状態遷移	状態	遷移
オブジェクト	オブジェクト	関連
ＪＳＤ	プロセス	データストリーム 状態ベクトル
フローチャート	命令単位	制御の流れ
ペトリネット	プレース トランジション	発火とトークンの流れ

3. 抽象モデルの記述法

Jackson の言うのももっともであるが，ソフトウェア工学では抽象的な記述をずっとモデルと呼んできた．データフローモデルも，ＥＲモデルも，状態遷移モデルも，みな抽象的な記述である．ここでは従来のソフトウェア工学の習慣に従いこれらの抽象的な記述をモデルと呼んだ上で，そこで一般的に使われる図式記法について考えてみる．

多くのモデル記述ではなんらかの図式表現を用いている．それもグラフ構造を使うものがほとんどである．その理由としては，

(1) 図を用いて視覚効果に訴えた方が，モデルらしい（やはり Jackson の言うように，何らかの実在性を感じさせた方がモデルらしいだろう）．

(2) 基本的な構成要素とそれらの間の関係でモデルを記述することは，もっとも一般的な方法である．それを構成要素を頂点とし，関係を辺とするグラフとして図示することは自然である．

ただ，視覚的効果というものは，意外に限界がある．実際，図が少し大きくなると，とたんに訳が分からなくなる．それに絵をきれいに描いただけで立派なモデルができた気になるが，有効なモデルになっているかどうかはまた別の話であることにも注意がいる．

同じグラフ表現も，その意味は当然のことながらモデル化技法によって異なる．しかし，グラフ構造は頂点と辺という簡単な要素で形作られるだけに，意味論の混同が起こりがちである．具体的には表 1 のように，頂点と辺にさまざまな意味づけが与えられる．

グラフには有向・無向グラフ，木・DAG・一般グラフなどの種類がある．また頂点や辺の集合をいくつかの種類に分割することもある．グラフを使うことにより，次のような処理ができる．

- 連結性の判定，連結成分の同定
- 推移律の適用，経路の意味づけ
- 探索

しかし，たとえば辺の推移的性質が因果律を表すのか，時間的順序を表すのか，単なる関係を表すかなどについて，とくに混乱を起こしやすい．

4. モデル化技法をいかに選ぶか

以上，概観したように同じようなグラフ記法をとりながら異なる意味をもつモデル化技法の種類は数多い．これらの中から何を選ぶべきだろうか．

まず，主要なあらゆるモデル化技法に習熟する必要はないだろう．一方で，統一モデル化技法があればよいが，それはどだい無理な要求というものだろう．結局，筋のいいものいくつかをマスターし，実際問題にどんどん使っていくことだろう．その際注意すべきことは，モデル＝開発手法ではないということのように思われる．

参 考 文 献

1) M.A. Jackson, *Software Requirements & Specifications: a liexicon of proctice, principles and prejudice*, Addison-Wesley, 1995. 邦訳．玉井哲雄，酒匂寛 訳: ソフトウェア博物誌 — 世界と機械の記述，トッパン，1997.

2) K. Torii, Analysis in Software Engineering, *Proc. 1st APSEC*, Tokyo, 1994.