1R-2 統合プログラミング環境（1） CASE指向ソフトウェアモデル

松本憲幸　小林茂　落合正雄
（株）東芝

1 まえがき

「統合プログラミング環境」におけるソフトウェア設計支援のプラットフォームとして Case Oriented Software Model(COS Model) について説明する。これは、コンピュータによる設計支援を想定した設計のモデルである。

ここでは CASE ツールによる思考過程の連続的なサポートのために、抽象段階から実体化段階までを、人の思考パターンに近い形式でサポート可能なモデルを見い出し、これに基づいて一貫した支援環境を構築することを考える。

2 COSモデルの基本概念

COS Model では、特に自然科学分野で人が未知の現象を理解する上で冒してきた思考上のバックトラックを極力回避可能とするような設計モデルの作成を目的として、次の概念を導入する。

(1)「原始定義」と「導出定義」
一般に、設計要素は「ある特性をもった要素の要求」を原始的な定義として生成される。この原始的な定義を、後の実体化により決定される定義（仮に導出定義と呼ぶ）と区別する。

(2)「素過程」
一般に、要素は複数の局面に出現するが、各局面で同一に振舞うとは限らない。COS Model では、各局面における要素の局所的な振舞い（素過程）の記述を基本とする。ある要素の原始定義は一般に、その要素に関する複数の素過程の合成されたものである。

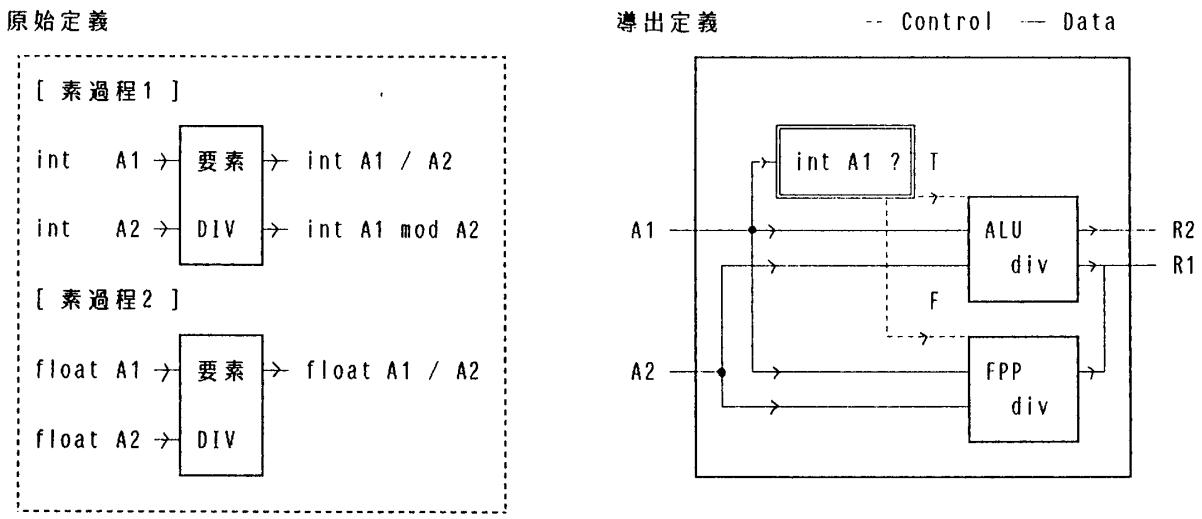
(3)「重ね合わせ」
ある要素に対する複数の素過程は、重ね合わせによりその要素の原始定義となる。この時、重ね合わせ可能な素過程の組合せは、記述言語や対象となる要素の種類によって異なる。

(4)「副作用」
すべての要素は「原始定義」を満たすが、一般にその実現形態に依存して要求以外の作用（副作用）をもち得る。この副作用は、ソフトウェアへの要求に関しては本質的ではないが、ソフトウェアの実体化に関しては本質的な問題である。要素に本来要求される作用と副作用を概念上区別する。

(5)「見かけ」と「実体」
一般に、ある要素の特性は、要素自身の特性である場合と、その要素に関係する他の要素によって与えられる見かけの性質である場合がある。「見かけの特性」と要素自身の「裸の特性」はともに現実的な意味をもっている。

設計途上にある概念である「素過程」などを積極的に導入することは、思考過程を連続的に CASE ツールでサポートする目的のほかに、最終実現形態の中に必ずしもソフトウェア設計の本質が含まれていないことへの対応、および設計の過程そのものを重要な情報として支援・管理可能とすることを目指している。

Case Oriented Software Model for Integrated Programming Environment
Noriyoshi Matsumoto, Shigeru Kobayashi, Masao Ochiai
TOSHIBA Corp.



上の図は「除算」に関する原始定義と導出定義の例である。ただし、これは概念説明のために作成した図であり、実際のソフトウェア設計図ではない。

要素 DIV に対して 2つの局面でそれぞれ「整数除算」、「実数除算」が要求され、これが DIV に関する原始定義となる。この場合の重ね合わせは、例えば要素がＬＩＳＰ言語関数であれば問題を生じないが、Ｃ言語関数であれば要注意となる。

導出定義は DIV の１つの実現形態に過ぎない。ここで示した導出定義では、 DIV は ALU と FPP の選択則を裸の特性としてもつが、除算処理は ALU, FPP に転嫁されており見かけの特性である。また、ALU, FPP が例えば他のプロセッサに対応するような場合、 DIV は副作用として各プロセッサを一時的に占有するという副作用を生じる可能性もある。

3 ＣＡＳＥツールサポート

COS Model は、簡単に言えば次の考えに基づいている。

「ある時点で目に見えるものは、確かにその実体が存在するのだが、
必ずしもその時点での認識は正確ではない。」
「ある時点で要素定義の全体を規定することは困難であるが、
各局面における要素の局所的な振舞いを記述することは可能である」

COS Model に基づいた CASE 環境では、人は「その時点で把握可能な正確な認識」を記述し、 CASE は「この分散した定義をリアルタイムで収集・検証・統合化するもの」として位置づける。例えば、ある種の要素に関する重ね合わせの規則は CASE ツールにあらかじめ知識として入力されていて、原始定義間の干渉や禁止された合成を検出する。

また、COS Model に基づいた入力は従来のイメージで「正確な設計図」ではない場合がありうるため、CASE ツールに入力された設計情報を「見かけの特性図」、「裸の特性図」など指定された条件に基づいて投影することによって設計図を得ることになる。

逆に、CASE ツールのサポートがない場合には、COS Model では設計情報の細分化・分散化を生じてしまい、その統合化に大きな労力を必要とする危険性がある。

4 あとがき

COS Model は CASE サポートの基本概念であり、実際の設計支援では、これに基づいたソフトウェア実体化のアプローチ（設計手法）が搭載されることになる。ここで導入された概念は、アプローチを規定するものではなく、既存手法への適用を考慮してかなり抽象化されている。

現在、COS Model に基づき、これをプログラム実体化まで近付けるアプローチを検討している。