ソフトウェア工学 71－9
（1990．2．8）

CASE指向抽象ソフトウェアモデル

松本憲幸

(株) 東芝

プログラミング言語により具現化される以前の抽象段階において、ソフトウェアの振る舞いを統一的に記述することを想定したモデルについて説明する。

これは、ソフトウェア開発の上流過程で利用される構造化分析、構造化設計により決定される情報を代数的なモデルで融合しつつ、プログラミングに至る以前に行うべき問題の定義過程に対して1つの定型化されたモデルを与えることを目的とする。

ここで示すモデルは、設計情報のダイアグラム表現と代数表現を媒介するモデルであり、モデル側で素過程という概念の導入による情報要素の細分化により両者の対応関係を取っている。

本論文では、ソフトウェアモデルとリアルタイム構造化分析の対応関係について説明する。

Abstract Software model for CASE

Noriyoshi Matsumoto

TOSHIBA Corporation

1 Toshiba-cho, Fuchu-shi, Tokyo 183, Japan

An abstract software model is introduced to describe the specific behavior of software in Upper CASE domain where the substantiation by programming language is not yet conscioused. The model intends to unify the information of diagrams given by structured analysis and structuted design and give an algebraic form to define problem instead of graphic form. The correnpondence between both form is established by differenciating the software element into the set of elementary process of the element .

In this paper consistency and inconsistency of the software model and strucured analysis for realtime system are presented.

1 はじめに

抽象段階においてソフトウェアをモデル化することは以下の目的をもつ。

設計情報の統合化

ソフトウェアに関する特徴的な情報の記述には、Data Flow 図 (DFD), Control Flow 図 (CFD), 状態遷移図 (STD), Control Spec (CSPEC)[1],[2],[3] など目的に応じた各種の設計図が存在するが、各設計図はソフトウェア情報の断面を離散的に記述する傾向にあり、これを利用して設計を行う場合、複数の設計図により1つのソフトウェア実体を記述することになる。ここでは、これらの設計図の位置付けおよび設計図相互の相関を具現化する統合化モデルを作成する。

代数モデル化

ダイアグラム化の最大の利点は形態・配置による視覚的な情報理解にあるが、ダイアグラムが多重化されたり複雑化された場合この利点は消えてしまう。このような場合、情報の対称性や完全性の解析には情報を代数化しておいた方が、より見通しが良いと考えられる。本モデルは、ダイアグラムによって記述されたソフトウェア情報を保存しつつ代数化することを目的とする。

なお、ここで説明するモデルは、コンピュータによる設計思考過程の連続的な支援を想定したCOSモデル[4]の原始定義と呼ばれる部分を記述することを想定しているものである。

2 基本概念

通常、ソフトウェア構成要素はその名称によって識別され、これが1つの基本単位となる。これに対して、COSモデルでは、ソフトウェア構成要素が様々な局面で示す振る舞いの各々を素過程と呼び、これをモデルの構成要素の基本単位と位置付けている。このような基本単位の導入は、次の仮定に基づいている。

要求は離散的である

ある要素に対するすべての要求は、多くの場合離散的に発せられる。また、要求が一括で発せられた場合においても、人は分析の段階でこれを離散的に解決する。

名称は概念的である

抽象段階で与えられる名称は概念的である場合が多く、必ずしも最終的に唯一の物理的実体に対応するとは限らない(実際、構造化手法は、このような概念的な名称とこれに与えられた要求を、物理的な実体の名称とその各々への要求に分解していく技法と考えることができる)。

このようなケースを考えると、モデルの中に離散的な個々の要求を保存しておくことが好ましいと考えられる。本モデルではソフトウェア要素に対する離散的な要求を、各要素の離散的な振る舞いを基本単位として記述する。

このような基本単位のパラメータとして、構造化分析、リアルタイム構造化分析においてソフトウェア要素の振る舞いを特定する次のような量を選ぶことにする。

- 要素の名称
- 要素に対する入力(Data,Control)
- 要素からの出力(Data,Control)
- 要素の状態遷移

本モデルでは、これらの量の組み合わせで基本単位としての素過程を記述し、1つの名称で表されるソフトウェア要素の定義を、要素の素過程の集合で表す。

3 モデルの表現

本モデルは、図1に示すような入力情報群・内部状態量をもつソフトウェア要素・出力情報群の3項から構成される形式でソフトウェア要素の振る舞いを表現する(記述形式は量子力学におけるDiracの記法[5]を母体としている)。なお、ここで示す記述モデルは、それ自身が代数化されているというよりは、ダイアグラム表現と代数表現の対応をとる媒介的な位置にある。

図1で示す記述形式は次の特徴をもつ。

・要素のすべての素過程が対等に表現され、入力イベントごとの振る舞いなどを離散的に分析・記述することができる。

・ダイアグラム情報を本形式で記述することにより、自動的に各要素の素過程が切り出される。

・各種ダイアグラムにおける情報ベクトルの向きが保存される。

モデルの中で、要素に対する入力ベクトルは入力情報群 <・・|、要素からの出力ベクルは出力情報群 |・・> として表し、状態遷移の方向性については陽にベクトルを記述している。また、階層関係については、構造定義 := の左辺(上位)および右辺(下位)の関係で表現している。ただし、図1の形式からわかるように本モデルでは要素の階層は1段階ごとに分割されて記述されることになる。

本記述モデルと代表的なダイアグラム表現との対応関係を以下に示す。例1は階層構造をもった情報の流れを表し、これには図2のフロー図が対応し、階層関係は図3階層図のようになる。また、例2で記述されるような状態遷移を伴う処理過程は図4の状態遷移図に対応する。

< X1,X2, ・・・, Xn |	入力情報群 (= <X1| + <X2| + ・・・ <Xn|)
|Y1, Y2, ・・・, Yn >	出力情報群 (= |Y1> + |Y2> + ・・・ |Yn>)
< X | P | Y >	要素 **P** の状態遷移を陽に含まない1つの処理過程(**X**を入力し**Y**を出力する)
< X | P(S1->S2) | Y >	要素 **P** の内部状態遷移(**S1->S2**)を伴う過程
< X | P | Y > := < X | Q | Z > < Z | R | Y >	要素 **P** の内部における **Q,R** の直列情報結合
< X | P | Y, Z > := < X | Q | Y > ; < X | R | Z >	要素 **P** の内部における **Q,R** の並列情報結合

X1, X2, ・・・, Xn および **Y1,Y2,・・・,Yn** は **Data** または **Control** 情報
(ここでは**Control**を **a^, b^, c^** など^をつけた表現で識別することにする)

P, Q, R は処理系(プロセス)　　**S1, S2** は状態

図1 モデルの基本表現

```
< X | P | Y >
:= < X | Q | W, Z > ( < W | R | A > ; < Z | S | B > ) < A, B | T | Y > .
```

例1 階層化された情報の流れの記述

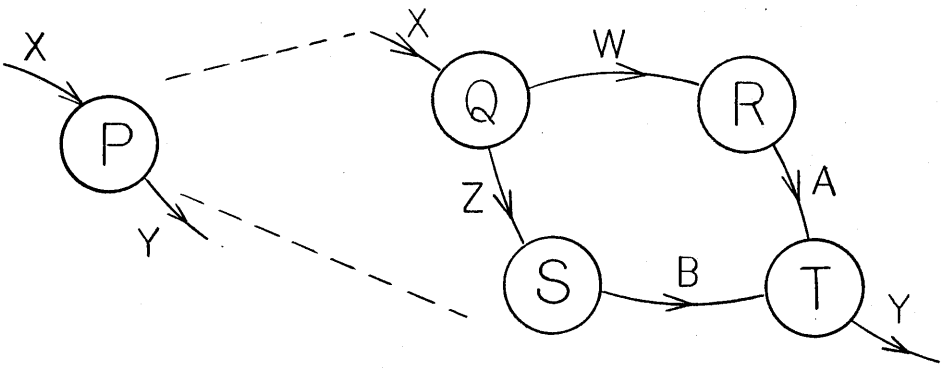

図2「例1」の階層化された情報の流れのダイアグラム表現

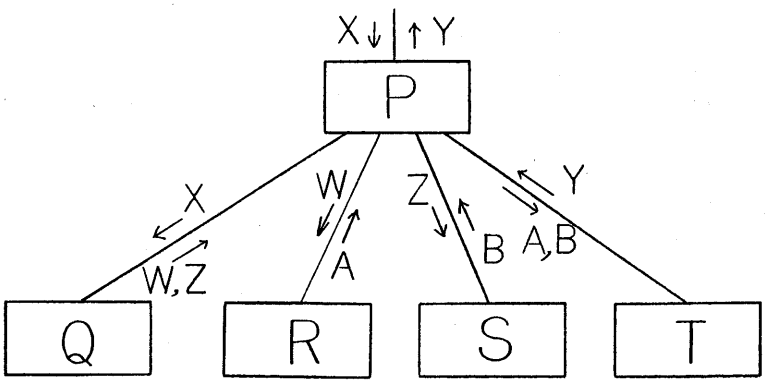

図3 「例1」の階層構造のダイアグラム表現

```
< A | P(S1->S2) | B > .
< C | P(S2->S3) | D > .
< E | P(S2->S1) | F > .
```

例2 状態遷移を伴うプロセスの記述

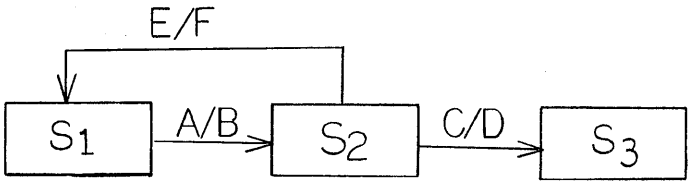

図4 「例2」から導き出される状態遷移図

4 モデルと設計図の対応

10円玉で10円のチョコレートを販売する簡単な自動販売機システムを例にとって、構造化設計図群と本記述モデルとの対応を考える。ここで考えるシステムは以下のようなものである。

- 10円投入するとランプが点灯する。
- ランプ点灯後、確認ボタンを押すとチョコが出てくる。
- ランプ点灯後、取り消しボタンを押すと10円が戻る。

構造化分析でこのようなシステムを設計すると例えば図5のような設計図群が出来上がる。図5で示された設計図群は、システムの動作を4枚の特性図の各々が部分的に記述しており、いずれか1つが欠けても意味を成さない。また、全体の動作は4枚の設計図を見比べることによって明らかになる。

次に、同じシステムを抽象モデルで記述した場合の例を例3および例4に示す。例3は自動販売機の素過程が6つの過程から成ることをを表し、例4は例3で示された(1)の過程を詳細化し、階層構造を定義したものである。

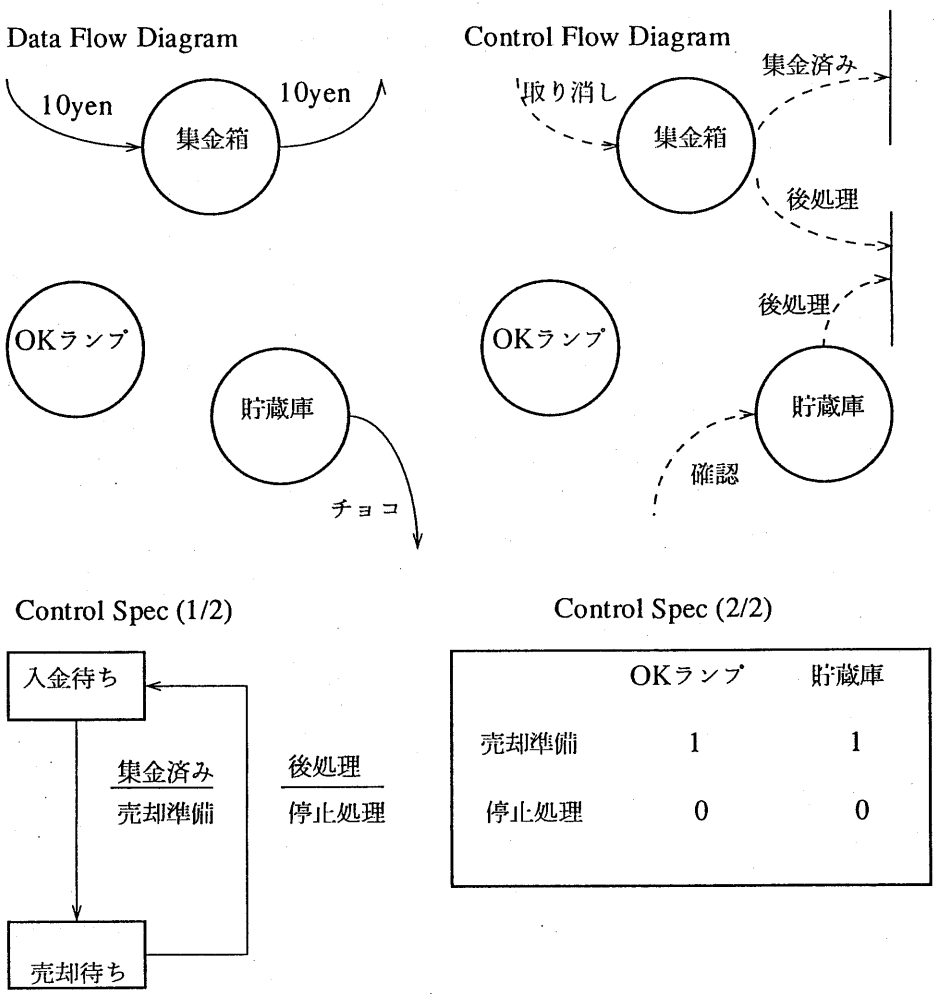

図5　10円でチョコレートを販売する自動販売機の設計図

```
<  10yen   |販売機(OKランプ : off->on )|>.                    ・・・(1)
<  10yen   |販売機(OKランプ : on->on ) | 10ye n>.             ・・・(2)

<   確認^   |販売機(OKランプ : on->off) |チョコ>.              ・・・(3)
<   確認^   |販売機(OKランプ : off->off)|>.                   ・・・(4)

< 取り消し^|販売機(OKランプ : on->off)| 10yen >.             ・・・(5)
< 取り消し^|販売機(OKランプ : off->off)|>.                   ・・・(6)
```

例3　10円でチョコを販売する自動販売機の素過程

```
<10yen|販売機(OKランプ : off->on)|>
   := <10yen|集金箱|集金済み^>
      <集金済み^|状態管理部(入金待ち->売却待ち)|売却準備^>
    (<売却準備^|OKランプ(off->on)|>;<売却準備^|貯蔵庫(off->on)|>).
```

例4　例3 (1) の素過程の構造化された記述

実際に、抽象モデルを利用してシステムを分析する場合の思考の流れも例3->例4のように進んでいく。すなわち、まず例3で示されるように販売機に関して考えるべき問題を列挙する。この段階で、例3の形式は問題の対称性・非対称性や完全性を見通しやすくしている。(例3は入力情報に着目して素過程を整理しているが、状態遷移の動的な完全性を評価するような場合には内部状態遷移に着目して整理するなどの再配置が必要である。このような再配置やダイアグラムへの変換投影にコンピュータサポートを想定している)。

次に、例3で定義された個々の素過程に対して、必要に応じて詳細な構造を定義していく。本形式は例4からわかるように、階層構造の感覚的な理解には向いていないが、階層化の段階で出現する各要素(例4の右辺に現れる要素)に対して、自動的に素過程を定義して行くことができる。

ここで示したように、抽象モデルで記述するする情報は、ほぼそのまま構造化分析情報と対応付けることが出来るが、両者の間で次のような相違点が見いだされる。

・構造化分析の設計図群が構成要素の全体的な仕様を記述するのに対して、抽象モデルは各要素の局面ごとの仕様を記述するため、完全な構造化設計図を求める場合には、素過程の和を取らなければならない。

・本形式は、Data, Control の流れや状態遷移を、統合化した1つの形式内で表現するため、状態を具現化する実体(例4の右辺に現れる状態管理部)を陽に記述する必要がある。

5 おわりに

ここでは、ソフトウェアの手続き的な部分についてモデル化して説明したが、構造をもったデータやデータ集団の記述などは現在のモデルに陽には含まれていない。また、最終的な素過程の統合化の問題や、各過程の干渉性の問題に関しても検討する必要がある。

モデルの表現は最終的にはコンピュータによ

る静的・動的な解析を想定しているが、現実的なコンピュータサポートを考えるには図1の漠然とした定義に対して、さらに厳密な定義と解釈を与えることが必要である。

[参考文献]

[1] Tom DeMarco, Structured Analysis and System Specification, YORDON, Inc., 1979.
[2] Derek J. Hatley, Strategies for Real-Time System Specification, Dorset House Publishing Co., Inc., 1986.
[3] Edward Yordon, Managing the Structured Techniques, Prentice-Hall, Inc., 1986.
[4] 松本 他, CASE指向ソフトウェアモデル, 情報処理学会第39回全国大会予稿集, p1379-1380, 1989.
[5] P.A.M.. Dirac, The Principles of Quantum Mechanics, Oxford At The Clarendon Press, 1958.