ソフトウェア工学 64－10
（１９８９．２．２）

リアルタイム・ソフトウェアにおけるプログラム仕様記述法に関する一考察

岡本克己，橋本正明

ＡＴＲ通信システム研究所

通信システム等のリアルタイム・ソフトウェアの仕様を記述するための非手続き的なプログラム仕様記述法について考察する．リアルタイム・ソフトウェアは，状態遷移機械モデルによって記述され，マルチプロセスで構成され，システム動作上に時間制約を持ち，システム外部との入出力を持つ．これらの特徴を，プログラムの入出力データに着目した非手続き的な仕様として記述するための方法を示す．特に，リアルタイム・システムにおいて重要となる動作時間，タイミングの取り扱いについて議論する．

A Consideration on A Program Specification Description Method for Real-Time Software

Katsumi OKAMOTO and Masaaki HASHIMOTO

ATR Communication Systems Research Laboratories

Twin 21 Bldg. MID Tower 2-1-61 Shiromi Higashi-ku Osaka 540 Japan

This article describes a nonprocedural program specification description method for real-time software. Real-time software has several characteristics such that it is usually based on finite state machine model, it consists of multi-process, it has time constraint and externalinput-output. A nonprocedural program specification description method focused on program input-output data . Especially, this paper discusses how to treat time constraint and timing which are important for real-time systems.

1．はじめに

通信システム等のリアルタイム・ソフトウェアの分野でも，高品質，高信頼なプログラムをいかに効率良く作成するかということが認識されており，この問題を解決する１つの方法としてプログラムの自動作成に関心が高まっている[1),2)]．

さて，プログラム自動作成の方法は種々あるが，本稿では，当面の実現性があり，しかも知能処理との結合により将来の発展性も予想される形式言語による自動作成の方法に着目する．この方法においては形式言語が必要であり，その言語を規定するため，まずプログラム仕様記述法を明らかにしなければならない．

そこで，プログラム仕様を容易に書き出すための記述性，プログラム仕様からプログラムを自動生成するための形式性，それにプログラム仕様を再利用するための理解性と拡張性を重視して研究中の非手続き的なプログラム仕様記述法ＰＳＤＭ（Program Specification Description Method）[3),4)]について，それをリアルタイム・ソフトウェアに適用する観点から考察を加える．

リアルタイム・ソフトウェアの特徴として状態遷移機械モデルによるシステム記述，マルチプロセス，時間の取り扱い，プロセス入出力が上げられ[2),5),6),7)]，それらの特徴を仕様として記述するための考察を加えた結果，特に時間の取り扱いについてはＰＳＤＭに新たな制約を導入するといった工夫が必要であること，その他の特徴についてはＰＳＤＭの適用方法が明らかになった．以下，第２章にリアルタイム・ソフトウェアの特徴を上げ，第３章でＰＳＤＭを概説する．それから，リアルタイム・ソフトウェアの各特徴についてＰＳＤＭの工夫点と適用方法を第４章で述べる．

2．リアルタイム・ソフトウェアの特徴

本章では，リアルタイム・ソフトウェアの各特徴についてさらに詳しく述べる．

(1)状態遷移機械モデルによるシステム記述

外部装置やプロセス等の状態が，状態遷移機械モデルで記述される．

(2)マルチプロセス

①マルチプロセス

リアルタイム・システムは多数の外部装置から構成されていることや，ソフトウェアの作り易さのために，複数のプロセスに分けて設計する．

②プロセス間通信

複数のプロセスが並行に動作すれば，プロセス間の通信が必要となる．

③排他制御と同期制御

プロセスがデータを共用すれば，データ・アクセスの排他制御や，同期制御も必要となる．

④プロセスの生成，消滅

プロセスが他のプロセスの生成や消滅を行う．

(3)時間の取り扱い

①タイミング

リアルタイム・システムは処理対象世界と同じ時間軸の上で動作しているので，制御信号やデータの入出力タイミングが処理の上で重要な意味を持つ．

②システム内時計

装置を制御するための応答が遅れると，装置が誤動作する可能性もあるので，応答時間に対して厳しい制約が加えられる．そこで，それを監視する等のためにシステム内時計がある．

(4)プロセス入出力（制御信号と外部割込）

外部装置との間で，制御信号等の入出力を行ったり，外部装置の状態変化を通知するために外部割込を起こす．

(5)ランダム・アクセス

リアルタイム・ソフトウェア特有の性質ではないが，リアルタイム・システムにおいても関係データ・ベースに代表されるランダム・アクセス・ファイルを使用することがある．

リアルタイム・ソフトウェアの記述では，上記特徴を記述できる必要がある．

3．ＰＳＤＭ

ＰＳＤＭはプログラムの入出力データの性質に着目した非手続き的なプログラム仕様記述法である．このような仕様記述法にとっては，入出力データの性質の捉え方が重要であり，ＰＳＤＭでは入出力データの性質を以下の３つの側面から捉える．

①入出力データが表す対象世界（プログラムが処理対象としている世界）に存在する事物や，その事物相互の関係を基準にして情報の枠組を定めた情報構造

②入出力データ中のデータ項目の並び方や，データ項目中の記号の並び方，さらにデータ項目と情報構造の対応を定めたデータ表現方法

③データ入出力について，ファイル等の性質や，入出力の区別等を定めたデータ・アクセス方法

そこで，上記の側面から捉えた性質を各々記述するための情報層，データ層，アクセス層に分けて，プログラム仕様を記述する．さらに各層を，プログラム仕様に現れる対象をタイプとして記述する構造と，タイプに課す条件を記述する制約に分けて記述する．

本章では，ＰＳＤＭに基づいて規定された実験的

なプログラム仕様記述言語ＰＳＤＬ（Program Specification Description Language）を用いて，ＰＳＤＭを説明する．

３．１　ＰＳＤＬ

ＡＢプロトコル（Alternating Bit Protocol）を例にしてＰＳＤＭを説明する．図１はＡＢプロトコルの送り手側の状態遷移図である．また図２は，そのプログラム仕様の情報層の構造を示したものであり，プログラム仕様全体をＰＳＤＬで記述したものが図３である．この図からわかるように，ＰＳＤＬ記述のプログラム仕様は文の集まりからできており，この文をＰＳＤＬ文と呼ぶ．

ここで，図３の01行目のprogram文はプログラム仕様記述の始まりを示すと共に，作成するプログラムのプログラム名がＡＢプロトコル（AB-protocol）であることも示している．また，39行目のend-program文はプログラム仕様記述の終りを示している．なお，表１はＰＳＤＭの記述要素である．

３．２　情報層

情報層には情報構造をプログラム仕様として記述する．

情報層の構造は，エンティティ・タイプ，リレーションシップ・タイプ，アトリビュート，及び，プライマリ・キーを用いて定める．エンティティ・タイプ，リレーションシップ・タイプはそれぞれ，エンティティ（事物の存在），リレーションシップ（事物相互の対応付け）を要素とする集合である．エンティティの性質はアトリビュートの値で表す．エンティティ・タイプの中で各々のエンティティを識別するために用いる特別なアトリビュートをプライマリ・キーという．

情報層	・エンティティ・タイプ ・アトリビュート ・プライマリ・キー ・リレーションシップ・タイプ ・リレーションシップ存在従属性制約 ・アトリビュート値従属性制約 ・エンティティ存在従属性制約
データ層	・データ・タイプ ・データの情報制約 ・繰り返し制約 ・選択制約
アクセス層	・データセット・タイプ ・入出力制約 ・データ制約

表１　ＰＳＤＭの記述要素

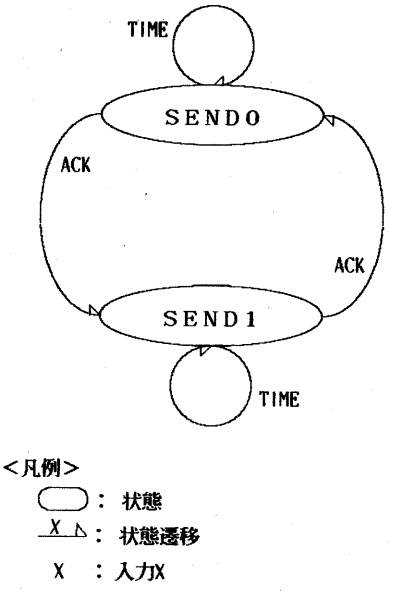

図１　状態遷移図

また情報層の制約は，与えられたエンティティ，そのアトリビュート値及びリレーションシップに基づいて，与えられたものとは異なるエンティティ，アトリビュート値またはリレーションシップを得るための計算方法を定めたものである．

図３では，02行目から18行目が情報層の仕様を記述した部分であり，02行目のinformation-layer文は情報層の仕様記述の始まりを示している．以下，情報層の仕様記述のためのＰＳＤＬ文を，構造と制約に分けて説明する．

３．２．１　構造

(1)エンティティ・タイプ

エンティティ・タイプには，プライマリ・キーとなるアトリビュートを自分自身で持つレギュラ・エンティティ・タイプと，他のエンティティ・タイプのプライマリ・キーで代用するウィーク・エンティティ・タイプとがある．

レギュラ・エンティティ・タイプは，図３の03行目のようにentity-type文を用いて，

```
entity-type InQ;
```

と記述する．

一方，ウィーク・エンティティ・タイプは，entity-type文に(week)をつけて記述する．

(2)アトリビュートとプライマリ・キー

あるエンティティ・タイプのエンティティが持つアトリビュートは，そのエンティティ・タイプを記述したentity-type文に続けて，１つのdescription文でまとめて記述する．例えば，04行目のように，

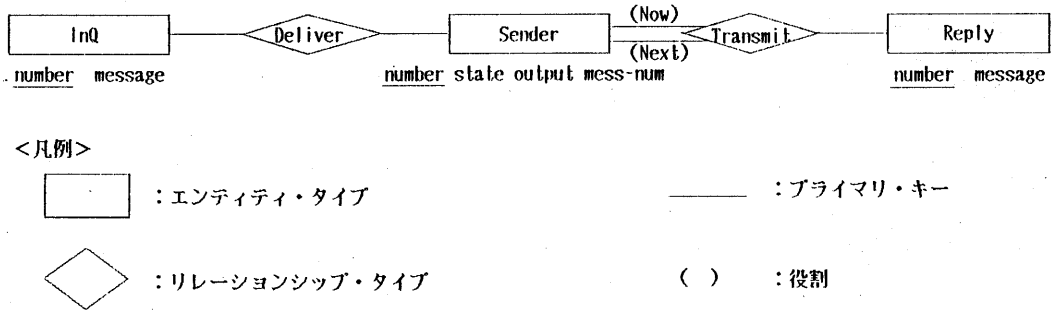

図2 情報層の構造

```
01 program AB-protocol;
02 information-layer;
03   entity-type  InQ;
04     description  number{num,id},message{num,};
05   entity-type  Sender;
06     description  number{num,id},state{num,},output{num,},mess-num{num,};
07   entity-type  Reply;
08     description  number{num,id},message{num,};
09   relship-type  Deliver;
10     collection  /InQ,/Sender;
11       rel-depn  P1:/InQ/number,/Sender/mess-num;
12       value-depn  /Sender/output ← F1:/InQ/message,/Sender/state;
13   relship-type  Transmit;
14     collection  Now/Sender,Next/Sender,/Reply;
15       rel-depn  P2:Now/Sender/number,Next/Sender/number,/Reply/number;
16       ent-depn  Next/Sender ← Now/Sender,/Reply;
17       value-depn  Next/Sender/state ← F2:Now/Sender/state,/Reply/message;
18       value-depn  Next/Sender/mess-num ← F3:Now/Sender/mess-num,/Reply/message;
19 data-layer;
20   sequence-type  InQ-R::=I-number,I-message;
21     element-type  I-number  num(8);
22     element-type  I-message  num(7);
23     ent-cons  InQ ← I-number;
24       value-cons  InQ/message[I-number] ← I-message;
25   sequence-type  Sender-R::=S-number,S-output;
26     element-type  S-number  num(8);
27     element-type  S-output  num(8);
28     ent-cons  S-number ← Sender;
29       value-cons  S-output ← Sender/output[S-number];
30   sequence-type  Reply-R::=R-number,R-message;
31     element-type  R-number  num(8);
32     element-type  R-message  num(8);
33     ent-cons  Reply ← R-number;
34       value-cons Reply/message[R-number] ← R-message;
35 access-layer;
36   dataset-type  InQ-P,input,15,InQ-R;
37   dataset-type  Sender-P,output,16,Sender-R;
38   dataset-type  Reply-P,input,16,Reply-R;
39 end-program;
```

<注>

P1:/InQ/numberと/Sender/mess-numが等しければ成り立つ条件

P2:Now/Sender/numberと/Reply/numberが等しく，かつ，Next/Sender/numberの値がNow/Sender/numberの値に1を加えたものに等しければ成り立つ条件

F1:/InQ/messageの値に/Sender/stateの状態を表す値を付け足したものと同じ値を取る関数

F2:Now/Sender/stateの状態と/Reply/messageの値により次の状態を取る関数

F3:/Reply/messageの値がACKならばNow/Sender/mess-numに1を加えた値を取り，TIMEならばNow/Sender/mess-numの値を取る関数

図3 プログラム仕様の記述例

description number{num,id},message{num,};
と記述する．ここで，アトリビュート"number"の後ろの{･･･}内の"num"はこのアトリビュートが数値型の値を持つことを示し，"id"はこのアトリビュートがプライマリ・キーであることを示している．

(3)リレーションシップ・タイプ

リレーションシップ・タイプもエンティティ・タイプと同様に，レギュラ・リレーションシップ・タイプとウィーク・リレーションシップ・タイプに分かれる．前者は，レギュラ・エンティティ・タイプを相互に対応付けたものであり，後者は，少なくとも１つはウィークであるエンティティ・タイプを相互に対応付けたものである．

レギュラ・リレーションシップ・タイプは，09行目のようにrelship-type文を用いて，

relship-type Deliver;

と記述する．一方，ウィーク・リレーションシップ・タイプは，relship-type文に(week)をつけて記述する．

また，エンティティ・タイプ間の対応付けは，14行目のようにcollection文を用いて，

collection Now/Sender,Next/Sender,/Reply;

と記述する．

同じエンティティ・タイプを相互に対応付ける場合，それぞれのエンティティが果す役割，例えば，"Now"や"Next"を付け加え，

Now/Sender, Next/Sender

のように"役割／エンティティ・タイプ名"で記述する．またはＰＳＤＬ文でアトリビュートを参照するときは，

Now/Sender/number, Next/Sender/number

のように"役割／エンティティ・タイプ名／アトリビュート名"の形で記述する．なお，あいまいにならなければ役割は省略してよい．

３.２.２　制約

(1)識別従属性制約

３.２.１(1)で述べたように，ウィーク・エンティティ・タイプは，そのプライマリ・キーを他のエンティティ・タイプのプライマリ・キーで代用している．そこで，どのエンティティ・タイプのプライマリ・キーで代用するかを

id-depn A ← B;

のようにid-depn文で指定する．

矢印の左辺"A"にウィーク・エンティティ・タイプ名を，矢印の右辺の"B"にプライマリ・キーを代用した相手のエンティティ・タイプ名を記述する．

(2)アトリビュート値従属性制約

あるエンティティのアトリビュート値に基づいて，そのエンティティへリレーションシップで対応付けられた他のエンティティについて，そのアトリビュート値を得るための計算方法は，図３の12行目のように，

value-depn /Sender/output ← F1:/InQ/message,
/Sender/state;

と記述する．

ここで，矢印の左辺"/Sender/output"は，値を得る対象のアトリビュートである．矢印の右辺の"F1"は，"/Sender/output"の値の計算方法を定めるための関数，"/InQ/message"と"/Sender/state"は，計算のために参照されるアトリビュートであり，関数"F1"の引数として記述する．

(3)リレーションシップ存在従属性制約

いくつかのエンティティに基づいて，そのエンティティの間に存在するリレーションシップを得るための計算は，11行目のように，

rel-depn P1:/InQ/number,/Sender/mess-num;

と記述する．

ここで，"P1"は述語であり，"/InQ/number"，"/Sender/mess-num"はその引数として参照されるアトリビュートである．そして，任意の２つのエンティティ"InQ"と"Sender"に対して述語"P1"が真であれば，その２つのエンティティの間にリレーションシップ"Deliver"が存在する．

(4)エンティティ存在従属性制約

あるエンティティに基づいて，そのエンティティとリレーションシップで対応付けられる他のエンティティを得るための計算方法は，16行目のように，

ent-depn Next/Sender ← Now/Sender,/Reply;

と記述する．

ここで，２つのエンティティ"Now/Sender"と"Reply"に対応して，１つのエンティティ"Next/Sender"が存在する．そのプライマリ・キー"number"は15行目のリレーションシップ存在従属性制約を満たす値，すなわち，Now/Sender/number＋１となる．

３.３　データ層

データ層には，データ表現方法をプログラム仕様として記述する．

図３では，19行目から34行目がデータ層の仕様を記述した部分であり，19行目のdata-layer文はデータ層の仕様記述の始まりを示している．以下，データ層の仕様記述のためのＰＳＤＬ文を，構造と制約に分けて順に説明する．

３.３.１　構造

入出力データ中のデータ項目の並び方を定めるために，①それ以上分解すると意味がなくなるエレメント・データ・タイプ，②幾つかのデータ・タイプが順序を持って並んでできたシーケンス・データ・

タイプ，③同じデータ・タイプのデータが１つ以上繰り返して並んでできたイテレーション・データ・タイプ，④幾つかのデータ・タイプのうちのいずれか１つだけが現れるセレクション・データ・タイプを定義し，これらのデータ・タイプを用いて１つの入出力データを１つの木構造として表す．

例を用いて説明すると，エレメント・データ・タイプは，図３の21行目のように，

```
element-type I-number num(8);
```

と記述する．これは，"I-number"がエレメント・データ・タイプであり，それが８桁の数値であることを示す．

シーケンス・データ・タイプは，20行目のように，

```
sequence-type InQ-R::=I-number,I-message;
```

と記述する．これは，"InQ-R"が，２つのエレメント・データ・タイプ"I-number"と"I-message"の並びからできたシーケンス・データ・タイプであることを示す．

イテレーション・データ・タイプは，図３に例はないが，例えば以下のように記述する．

```
iteration-type DATA{X}::=RECORD;
```

これは，イテレーション・データ・タイプ"DATA"が，"RECORD"というデータの繰り返しであり，指標に"X"を用いることを示す．

セレクション・データ・タイプも図３に例はないが，例えば次のように記述する．

```
selection-type RECORD::=A{="a"},B{NOT="a"};
```

これは，セレクション・データ・タイプ"RECORD"として，データ・タイプ"A"または"B"が現れることを示す．なお，選択条件を示すデータの{…}については３．３．２(3)で説明する．

３．３．２　制約

データ層の制約は，データと情報層の対応，及び，繰り返し集団項目型の繰り返し終了条件や選択集団項目型の選択条件を定めるものである．

(1)データの情報制約

どのエレメント・データ・タイプやインデックスが，どのエンティティやアトリビュートの値，またはリレーションシップを表しているかを定めるため，①エンティティの表現②エンティティとリレーションシップの表現③アトリビュート値の表現に関するの３種の情報制約がある．

まず①の制約は，どのエレメント・データ・タイプやインデックスが，どのエンティティ・タイプを表しているかを示し，ent-cons文を用いて図３の23行目のように，

```
ent-cons InQ ← I-number;
```

と記述する．これは，エレメント・データ・タイプである"I-number"がエンティティ・タイプ"InQ"を表していることを示す．

また②の制約は，どのエレメント・データ・タイプやインデックスが，どのリレーションシップ・タイプを表しているかをrel-ent-cons文で記述する．

そして③の制約は，どのエレメント・データ・タイプやインデックスが，どのアトリビュートを表しているかをvalue-cons文を用いて24行目のように，

```
value-cons InQ/message[I-number] ← I-message;
```

と記述する．これは，エレメント・データ・タイプ"I-message"が，"I-number"で決るプライマリ・キー値を持つエンティティ"InQ"のアトリビュート"message"の値を表すことを示す．

(2)繰り返し制約

イテレーション・データ・タイプにおいて，データの繰り返しがどのような条件で終了するかを示す．例えば，

```
iterate-cons DATA(X){="END"};
```

の場合，エレメント・データ・タイプ"DATA"の値が"END"になれば，そこでデータの繰り返しが終了することを示す．

(3)選択制約

セレクション・データ・タイプにおいて，どのデータ・タイプが現れたかを選択するための制約であり，例えば，３．３．１項で上げたselection-type文に対しては，

```
select-cons DATA(X);
```

と記述し，エレメント・データ・タイプ"DATA"の値が"a"であれば，データ・タイプ"A"を選択し，"a"でなければ，"B"を選択する．

３．４　アクセス層

アクセス層にはデータ・アクセス方法をプログラム仕様として記述する．構造としては，ファイルやプロセス入出力を表すデータセット・タイプを設け，その名前やアクセス単位長を定める．制約としては，１つのデータセット・タイプをデータ層の１つの木構造に対応させるためのデータ制約，データセット・タイプの入力用，出力用を区別するための入出力制約を定める．例えば，図３の36行目のように

```
dataset-type InQ-P,input,15,InQ-R;
```

と記述し，これはデータ層の"InQ-R"とデータセット・タイプ"InQ-P"とが対応し，"InQ-P"は入力用プロセスであり，レコード長が１５バイトであることを示す．

４．ＰＳＤＭの工夫点と適用方法

ＰＳＤＭをリアルタイム・ソフトウェアへ適用する場合の適用法と，適用するに当っての工夫点について述べる．

4.1 状態遷移機械モデルによるシステム記述

状態遷移機械モデルで表されるシステムのプログラム仕様は，以下のように記述できる．まず，遷移前の状態を示すアトリビュートを持ったエンティティと，遷移後の状態を示すアトリビュートを持ったエンティティ，入力を示すアトリビュートを持ったエンティティの間にリレーションシップを設ける．そして，アトリビュート値従属性制約として状態遷移関数を定める．例えば，図２の状態遷移図で表されたＡＢプロトコルの送り手側の状態遷移は，図３の17行目のように，「"Now/Sender/state"と"/Reply/message"の値を用いて関数"F2"によって"Next/Sender/state"の値を定める．」と記述することができる．ここで，関数"F2"が状態遷移関数である．

4.2 マルチプロセス

(1)マルチプロセス

リアルタイム・ソフトウェアでは，１つのシステムを複数のプロセスに分けて設計することが多いので，プロセス間の関係について仕様記述法を定めるのも重要であるが，個々のプロセスの仕様をいかに表すかも重要である．そこで当面，後者に着目して研究中である．

(2)プロセス間通信

プロセス間の通信データも他の入出力データと同じなので，情報層には，通信データが表している対象世界の情報構造を記述する．またデータ層には，通信データの形式を記述し，アクセス層には，通信先名（データセット・タイプ名）や送受（入出力）の区別等を記述する．

なお前項で述べたように，個々のプロセスの仕様記述に着目しているので，通信方法についてどれを採用するかはプログラムの実行環境に委せる．

(3)排他制御と同期制御

プロセス間でデータを共用するには排他制御や同期制御が必要であるが，排他制御についてはプログラム実行環境のオペレーティング・システム（ＯＳ）の機能に委せる．

プロセス間の同期を実現するには幾つかの方法があるが，例えばセマフォを使って実現するには，セマフォの機構も含めたシステム記述を行うことによって，ＰＳＤＭで記述可能である．

(4)プロセスの生成，消滅

あるプロセスが他のプロセスを生成，消滅させるには，前者のプロセスがＯＳに対してプロセスの生成，消滅オペレーションを発する．

このため，プロセスがＯＳに対して生成，消滅オペレーションというデータを出力するようにＰＳＤＭで記述する．

4.3 時間の取り扱い

ＰＳＤＭにおいて時間を取り扱うには，①事象の生起，完了の時刻や事象の継続時間，②ある事象が他の事象よりも先に生起すること，複数の事象が同時に継続していること，ある事象が完了したら次のある事象が生起すること等を記述する必要がある．そこでＰＳＤＭの情報層では，時間に関する性質を以下のように取り扱う．

まず①については，事象を示すエンティティのアトリビュート値で表し，また②については，リレーションシップ・タイプとして表す．

そして，それらリレーションシップ・タイプは，事象の生起時刻を示すアトリビュートなどに基づくリレーションシップ存在従属性制約から求めることができる．また事象の継続時間は，事象の生起，完了時刻を示すアトリビュートに基づくアトリビュート値従属性制約から表すことができる．最後に，ある事象の発生に基づいて他の事象が発生することは，エンティティ存在従属性制約から表すことができる．

このようにＰＳＤＭでは，エンティティ・タイプ，アトリビュート，プライマリ・キー，リレーションシップ・タイプ，３種の制約を用いることによって時間に関する性質を表すことが可能である．

しかし，以下に示す特徴を記述するためには，データ層やアクセス層に特別な工夫が必要となる．

(1)タイミング

入力データに入力時刻が表されていれば，事象間の時間的前後関係は情報層におけるリレーションシップ・タイプを取ることによって容易に知ることができる．しかしリアルタイム・ソフトウェアでは，入力データに入力時刻が表されていなくても入力タイミングから事象間の時間的前後関係を知る必要が生じる場合がある．このような場合には先に述べた情報層の記述だけでは対処不能である．

この問題を解決するために，どのデータ項目間に時間的前後関係があるかをデータ層に制約として記述する．さらにアクセス層では，そのデータ項目を含むデータ集合について時間的前後関係が定義されていることを制約として記述する．

(2)システム内時計

システム動作上に時間制約があり，それを監視するための時計がシステム内にあり，それから"タイムアウト"等の信号がシステムの入力となる場合や入力データ中に時刻を示すデータがなく，データ入力時にシステム内部の時計から時刻をデータとして取り込む場合には，どのデータ集合がシステム内時計からの入力に該当するのかをアクセス層に制約として記述する．

	状態遷移機械モデルによるシステム記述	マルチプロセス	時間の取り扱い	プロセス入出力	ランダム・アクセス
情報層	・状態 ・状態遷移	——	・事象の生起，完了の時刻や継続時間 ・事象間の時間的関係	——	——
データ層	——	・プロセス間通信のデータ形式 ・プロセスの生成，消滅オペレーションを表すデータ形式	・入力データ項目間のタイミング制約	・アクセス単位のデータ形式	・アクセス単位のデータ形式
アクセス層	——	・プロセス間通信の通信先と送受の区別	・プロセス入力のタイミング制約 ・システム内時計の入力制約	・データの種類 ・アクセス方法	・アクセス方法 ・アクセス・キー

表2　リアルタイム・ソフトウェアに対する工夫点と適用方法

4.4　プロセス入出力（制御信号と外部割込）

制御信号や外部割込もデータの一種と見なせるので，情報層やデータ層の記述法には特別な工夫はいらない．但し，アクセス層においては，アクセス方法等が普通のデータと異なるので，データの種類が制御信号や外部割込であることと，そのアクセス方法を記述する必要がある．

4.5　ランダム・アクセス

従来ＰＳＤＭでは，シーケンシャル・アクセス・ファイルのみを対象としていたために，ランダム・アクセス・ファイルを扱うには以下の工夫が必要である．

まずデータ層では，シーケンシャル・アクセス・ファイルの場合，ファイル全体のデータ形式をまとめて１つの木構造で記述した．一方，ランダム・アクセス・ファイルの場合は，アクセス単位となるレコードのデータ形式を１つの木構造で記述する．

アクセス層では，アクセス方法がランダム・アクセスであるかシーケンシャル・アクセスであるかの区別を記述する．さらに，ランダム・アクセスの場合，アクセス・キーも記述する．

以上に述べたリアルタイム・ソフトウェアに対する工夫点と適用方法をまとめると表２に示すようになる．なお表中の網かけ部分（▒）は，リアルタイム・ソフトウェアにのみ使用される部分である．

5．おわりに

リアルタイム・ソフトウェアの特徴を述べ，次にリアルタイム・ソフトウェアの仕様記述にＰＳＤＭを適用する場合の工夫点と適用方法の考察を行った．

その結果，２章で述べた特徴を持つリアルタイム・ソフトウェアについては，ＰＳＤＭを用いて記述できることが明らかになった．なお，アトリビュート値従属性制約などの制約で用いられる関数や述語の記述方法については，さらに研究が必要である．

また現在，本稿の内容を反映した，さらに理解性に優れたプログラム仕様記述言語ＰＳＤＬや，この言語で記述されたプログラム仕様からプログラム生成するのに必要なプログラム構造設計の自動化，ＰＳＤＭを用いた要求分析について研究中である．

謝辞　日頃ご指導いただく，ＡＴＲ通信システム研究所の葉原会長，山下社長，門田室長に深く感謝致します．また，ご意見をいただいた通信ソフトウェア研究室の諸氏に感謝致します．

参考文献

1）山下紘一，門田充弘：講演:通信ソフトウェアの自動作成について，情報処理学会ソフトウェア工学研究会資料62-1（1988）．
2）本位田真一，内平直志，中村英夫：制御分野における自動プログラミング，情報処理，Vol.28，No.10，pp.1398-1404（1987）．
3）橋本正明：ＥＡＲモデルに基づく情報構造記述を用いたプログラム仕様記述法ＰＳＤＭ，情報処理学会論文誌，Vol.27，No.7（1986）．
4）橋本正明：データ中心のプログラム仕様記述法，P.210，井上書院，東京(1988)．
5）松本吉弘：リアルタイムシステム，P.190，昭晃堂，東京(1984)．
6）三巻達夫，桑原洋：制御用計算機におけるリアルタイム技術，P.257，コロナ社，東京(1986)．
7）Robert L.Glass：REAL-TIME SOFTWARE，P.453，PRENTICE-HALL（1983）．