実行可能な仕様記述におけるプラント制御システムの環境のモデル化

小林久浩[†] 河田恭郎[††] 前川守[††]
川崎章[††] 薮彰文[††] 小野川公也[††]

プラントの制御システムの構築のための仕様記述に対してのさまざまな研究がなされているが，このようなプラントの制御システムのモデル化を正確に行うためには，それを取り巻く環境のモデル化も重要な要素となる．特に，プラント制御システムの構築においては，その特質から，通常センサから連続的に生成されるような情報や，プラントを構成する設備間を流れるもの（例えば冷却水など）のモデル化が必要不可欠である．本研究では，プラント制御システムのモデル化をするときに特徴的に現れる以上述べたような外界の現象を，実行可能なオブジェクト指向仕様記述という枠組のなかで，どのように記述するべきかを提案するものである．実世界のオブジェクトとの対応をとりつつ，連続・離散系を問わず両系を柔軟にサポートし，かつ多様なコンテキストで適用可能な部品化の概念を用いて再利用性を提供する．

Modeling External Objects of Process Control Systems in Executable Specifications

Hisahiro Kobayashi,[†] Yasuro Kawata,[††] Mamoru Maekawa,[††]
Akira Kawasaki,[††] Akifumi Yabu[††] and Kimiya Onogawa[††]

External (i.e., belonging to the environment) objects of process control systems typically possess the following peculiarities that are foreign to the data-intensive application domain: i) some data are "continuous" in that they are "continuously" acquired by sensors; and, ii) some substances such as cooling water "flows" from one facility to another. Writing accurate specifications of control systems demands proper modeling of the above peculiarities but existing executable specification languages fall short. In this paper, we present features of our object-oriented executable specification language Eunice, which answer all the above needs; especially, Eunice allows modeling and execution of the same object in both discrete and continuous fashion. This mechanism provides a way for highly adaptive reusable software components that automatically select one appropriate model of the two, depending on the settings they are in.

1. はじめに

プロセス・コントロール・システムやプロダクション・コントロール・システムなどの，プラントの制御システムを構築するためのさまざまな方法論や技法が提案されてきた．特に，その要求仕様記述の分野では，その仕様自体が実行可能な要求仕様記述手法が現れ，ある程度その有効性に対して合意がとれつつある．本研究もこういった実行可能な仕様記述手法を用いて，プラントの制御システムを構築することを議論するものである．

何らかの情報システムを構築するための要求分析をするときに，その情報システムを取り囲む環境（以後，「外界」と呼ぶ）に関しても分析することは，より正確に目標のシステムをモデル化するために不可欠である．プラントの制御システムの構築に関してもこのことは当てはまるが，その外界の性質は事務処理システムなどとは大きく異なる．

図1の例をとって説明する．これは，タンクの水位を一定に保つというごく簡単な制御システムであるが，プラント制御システムの外界が特徴的に持つ性質を含んだ例である（以後，この論文を通して例として引用する）．

† 東京大学理学部情報科学科
Department of Ioformation Science, Faculty of Science, University of Tokyo
†† 電気通信大学大学院情報システム学研究科情報システム設計学専攻
Department of Information Systems Design, Graduate School of Information Systems, University of Electro-Communications

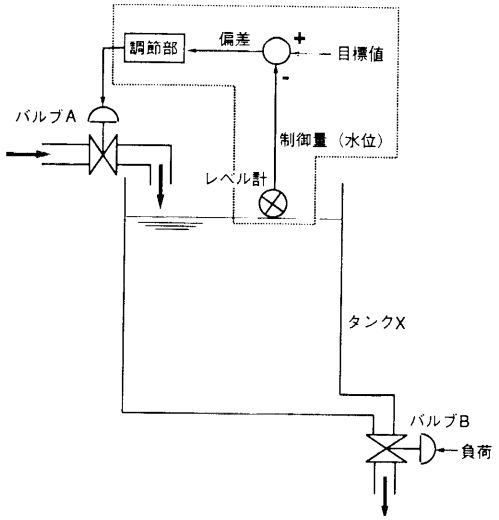


図1 液定量システム
Fig. 1 Liquid level stabilizer.

図1の破線で囲まれた部分が制御システムの部分であり，その外側が外界ということになる．その外界において特徴的な点は大きく次の2つが挙げられる．

- 連続的に生成される情報が存在する
  水位センサによって検知されるタンクの水位という情報があるが，この情報は任意の時間にその値を変え得るもので，常に連続的にその値が生成されていると考えてよい．
- 制御される設備の中を流れるものが存在する
  制御システムは，そのプラント内の設備を制御するわけであるが，その設備への制御は結果的にその設備の中を流れるもの（以下，「内流物」と呼ぶ）にも影響を与える．例では，制御システムは，バルブAの開閉を制御するわけだが，それによって間接的に内流物の水が制御される．

以上のような点を適確にモデル化することが，プラントの制御系の構築には必須といえる．本研究では，こういったプラントの制御システムの外界に特徴的に現れる事柄を，実行可能な仕様記述という枠組の中で，記述し実行するための記述法および機構を提案するものである．さらに，その外界に関する記述をさまざまなコンテキストで使用可能に部品化するための機構も提供する．

以下，次の章でわれわれの提案の基本的な概念を説明し，その後の章で具体的な記述法および機構を説明する．さらにその後，関連研究を挙げて我々の研究と比較する．最後に，今後の課題と現在の状況を含めて本研究のまとめとする．

2. 本研究の基本概念

前章で述べたようなプラント制御システムの外界に特徴的に現れる事柄をどのように記述するかについて述べる．大きく3つの柱がある．

2.1 オブジェクト指向的なモデル化

現在，オブジェクト指向的なモデル化の有効性は広く受け入れられつつあるが，本研究でも外界のモデル化に関して，オブジェクト指向の概念を採用する．

オブジェクト指向的なモデル化の最も根幹をなす概念の1つは，実世界のオブジェクトとモデル世界のオブジェクトがほぼ1対1に対応付けられるということであるが，本研究では次のような点においてこの概念を採り入れる．

- 実世界の設備・機器などのオブジェクトとモデル世界のオブジェクトとを1対1に対応させる．
- 実世界の設備間のインターフェースとそれらの接続を，モデル世界のオブジェクト間のインターフェースとその接続に1対1に対応させる．
- 実世界の内流物とモデル世界のそれを表すオブジェクトを直観的に理解できるように対応させる．「水」のような連続的な内流物に関しても直観的に理解しやすい記述法を与える．

2.2 連続的なモデルと離散的なモデルの混合系のサポート

プラント内の内流物に関してもう少し考察してみると，次のようなことが明らかになる．

- 内流物には2種類あり，連続的にモデル化されるのが一般的なもの（例えば水のようなもの）と離散的にモデル化されるのが一般的なもの（例えば，ベルトコンベアで運ばれる瓶のようなもの）がある．
- 通常は連続的にモデル化される内流物でも，モデルを構築する側（仕様記述者）の都合で，離散的にモデル化されることもある．例えば，「ある設備に対して水5リットルを一度に渡す」とかというように離散的に水がモデリングされる場合もあり得る．また逆に通常は離散的にモデル化される内流物が連続的にモデル化されることもあり得る．
- 1つのモデルの中に，以上述べたような連続，離散の両方が混合される場合がある．特に極端な例では，同じインターフェイスに対して離散

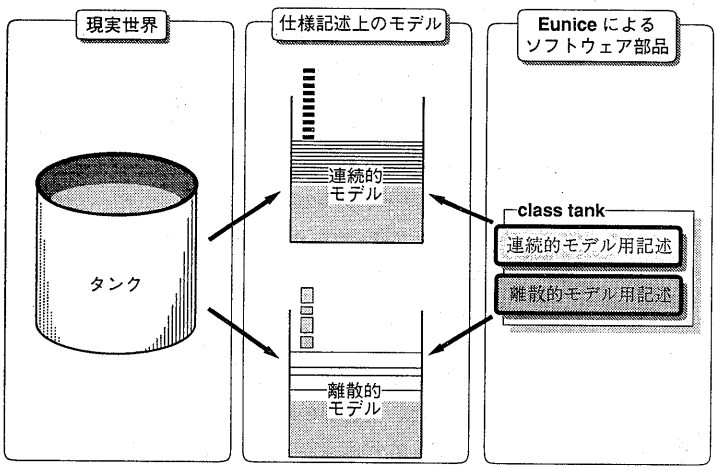


図 2 基本概念
Fig. 2 Basic concepts.

的にも，連続的にも内流物が渡されるようなモデル化がされ得る．

以上のような要請より，本研究では，連続的な記述と離散的な記述が混合されたモデル化を支援し，次節で述べるような，連続・離散どちらのコンテキストでも適用可能な部品化を提供する．

2.3 部 品 化

外界のモデル化の最終的な目的は，より正確に制御システムの要求仕様をつくることにあるので，もし外界のモデル化のコストが多大なものであるならば，本末転倒になってしまう．より低いコストで外界のモデル化ができるように，本研究では部品化およびそれによる再利用性の向上を図る．

その部品化は，以上述べた2つの概念に相反するものであってはならない．つまり，現実世界のオブジェクトの単位で部品化され，また離散的なモデリングに使われても，連続的なモデリングに使われてもきちんと機能するものでなければならない．つまり，一つの部品のなかに連続用の記述と離散用の記述を含んでおり，それを適用のコンテキストに応じて使い分ける機構が必要である．われわれは，このような部品のことを**多様コンテキスト適用可能部品**と定義し，このような部品の作成が可能な枠組を提供する．

以上，3つの基本概念のまとめを**図2**に示す．

3. 具体的な記述法および機構

ここでは，先に述べた3つの基本概念を実現するための具体的な記述法と機構を，Eunice という実行可能なオブジェクト指向仕様記述言語とともに説明する．まず，第1章で挙げた例(図1)の中のタンク X というオブジェクトを生成するクラス*の全体を挙げる（**図3**）．

このクラスは，水位計付きのタンクを部品化したもので，どのようなオブジェクトの間に組み込まれても成り立つことのみが記述されている．また，離散・連続のどちらでも使用できる部品である．つまり，離散的に使われたときのための記述と連続的に使用された場合の記述の両方が用意されている（もちろん共有部分もある）．

以下，各記述部分について順に説明していき，この部品がどういった概念や機構に支えられているかを明らかにし，その後でこの部品をどのように使うかを述べる．

3.1 連続的な情報，内流物，およびインターフェイスの表現

まず，連続的に生成される情報の表現について述べる．例の中では，**attribute** ブロックの中の *level* が **continuous** 型修飾子をもとなって宣言されているが，このことによって連続的に生成されるタンクの水位を表現することを表している．このような変数は，任意時刻に明示的な代入なしに変化し得るもので，シミュレーションの段階では，グラフや後で述べるような等式の集合による制約によってその値が与えられ，実装された後には，水位センサからの検出値で与えられるものである．以後，このような変数のことを「連続変数」と呼ぶ．

次に，内流物の表現について説明する．この例の場合，内流物は水であるが，水に関しては**図4**のように2つのクラスが定義されている．

一般的に，離散的な内流物の表現は，図4の上方の定義のように，普通のクラスから生成されるオブジェクトとして表現される．一方，連続的な内流物の表現は，ある位置におけるその内流物の流れの属性値（流量，流速，圧力など）を表す先に述べた連続変数の集合として表現される．

これらの定義のどちらが適用されるかは，その変数

* 厳密にはクラスと型は別の概念であるが，Eunice ではクラスと型を同一視している．

```
tank_with_a_level_meter: class
  channel
    in inlet: inflow, inblock;
    out outlet: outflow, outblock;
    out level_data_outlet: level;
  attribute
    level: continuous real;
    total_volume: continuous real;
    continuous_volume: continuous real;
    discrete_volume: real;
    inflow: continuous water;
    outflow: continuous water;
    inblock: water;
    outblock: water;
    area_of_bottom: real;
    area_of_outlet: real;
    height_of_outlet; // z coordinate;
  constraint
    always
      level = total_volume / area_of_bottom; // 第1式
      total_volume = continuous_volume + discrete_volume; // 第2式
      (1/2) * rho_water * (outflow.velocity) ** 2
      + rho_water * g * height_of_outlet + outflow.pressure
      = rho_water * g * (height_of_outlet + level) + atmospheric_pressure; // 第3式
      continuous_volume = integ(inflow.volume_per_sec)
                          -integ(outflow.volume_per_sec); // 第4式
      outflow.volume_per_sec = outflow.velocity * area_of_outlet; // 第5式
  action
    block_in is data_driven
      triggeredby inblock <-
      do
        discrete_volume := discrete_volume + inblock.volume;
      end;
    block_out is data_driven
      triggeredby outblock ->
      do
        discrete_volume := discrete_volume - outblock.volume;
      end;
  endclass;

//物理定数
rho_water: constant real is 1.0;
g: constant real is 9.8;
atmospheric_pressure: constant real is 1.0;
```

図 3 水位計付きタンクのクラス
Fig. 3 Class for tanks with a level meter.

の宣言のときに，**continuous** 型修飾子をつけるかどうかによって切り替えられる．

例えばタンクの例では *inflow*, *outflow* はそれぞれ **continuous** *water* 型として宣言されているので，図4の下の方の定義が適用され，それぞれタンクの入口と出口という位置におけるの水の流れの属性の集合を表す．一方，*inblock*, *outblock* はそれぞれ離散的に受け渡しされる水を表すものである．

次に，インターフェイスの表現については，チャネル方式を採用した．その理由としては，実世界に則したモデル化，つまり実際の設備のインターフェイスと1対1でモデル上に表現できるということ，また連続的な情報や内流物の表現が自然にモデル化できること，さらにチャネルのもつ双方向性が挙げられる．

先述の，*level*, *inflow*, *outflow*, *inblock*, *outblock* の変数は**図5**のようにチャネルと対応付けられており，そのチャネルに対してどのような型のものが渡されるかによって，その渡されたものをどの変数で受けるかが切り替えられる機構がある（**図6**）．この切り替え機構によって同じチャネルを用いて，連続・離散を問わずさまざまな物質，情報の授受が実現でき，かつ連続的なコンテキストにも離散的なコンテキストにも適用可能な部品が作成できる．

```
water: class
  attribute
    volume: real;
    mass: real;
    density: constant real is rho_water;
  endclass;

water: continuous class
  attribute
    velocity: continuous real;
    pressure: continuous real;
    volume_per_sec: continuous real;
    density: constant real is rho_water;
  endclass;
```

図 4 2つの水のクラス
Fig. 4 Two water classes.

3.2 連続的なことに関する記述

連続変数の値は，連続でない変数と同じように明示的な代入を用いて，手続き的にその値を規定するよりも，等式を用いて宣言的に記述する方が自然である．これは，連続的な変数には，常に値が暗示的に代入され続けていることに起因する．

したがって，連続変数の値や連続的な内流物の流れの属性値に関する記述は，**constraint** ブロック内に等式の集合として記述される．この **constraint** ブロック内の構文は，**図7**に示す．**always** ブロックには恒常的に有効な式が記述され，**transient** ブロックには，逐次的な実行の中でオン・オフされる **rule** とい

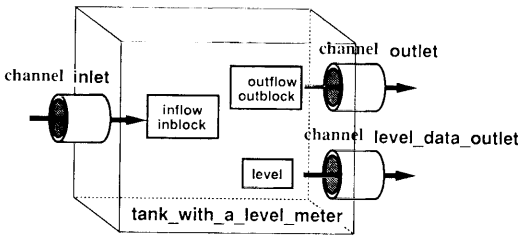


図 5 チャネルと仮引数の関係
Fig. 5 Channels and their formal parameters.

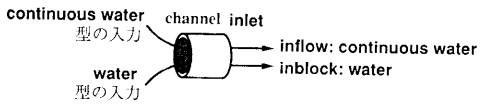


図 6 実引数の型による自動的切り替え
Fig. 6 Automatic dispatch of parameters by their types.

```
constraint
  always
    ... // 制約式の集合
  transient
    rule_identifier is rule
      ... // 制約式の集合
    end;
    rule_identifier is rule
      ... // 制約式の集合
    end;                              10
```

図 7 制約記述部分の構文
Fig. 7 Constructs for constraints.

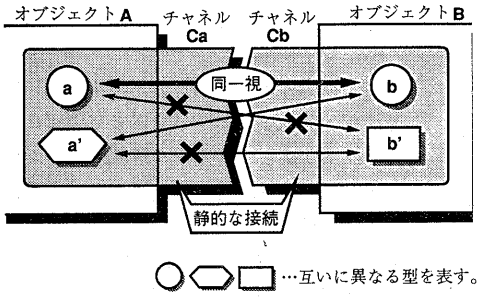


図 8 チャネル接続による変数同一視
Fig. 8 Variable association via channel connections.

う単位の制約式の集合が記述される．また，制約式は **if** 文によって評価条件を記述することもできる．つまり，一部の制約式は動的に有効，無効が切り替わる．

タンクの例では，**always** ブロックのみが存在する．以下簡単に，制約式の説明をすると，第2式では，連続的な水のやりとりに関するタンク内の全容量を表す *continuous_volume* と，離散的な水のやりとりに関するタンク内の全容量を表す *discrete_volume* との和が，総合的なタンクの総容量と等しいことを表している．*continuous_volume* は，第4式と第5式から与えられる．また，第3式はベルーヌーイの定理から与えられる式である．以上からわかるように，ここで記述されている式は，このタンク部品がどんなコンテキストで使われようと成り立つものである．

ところで，ある時刻において有効な制約式の中に現れる連続変数の値は，それが関連付けられているチャネルから暗示的に受け渡されていることになる．これは，後で説明する離散的なものの授受と大きく異なるところである．

タンクの例の **constraint** で与えられた式だけでは，*outflow* の圧力や，速度は求められない．どのように，これらの値が規定されるかを図8にそって説明する．

あるオブジェクト A 内の a という連続変数があって，それが Ca というチャネルと関連付けられており，あるオブジェクト B 内の b という連続変数があって，それが Cb というチャネルと関連付けられている場合を考える．まず，静的にあとで述べるように，Ca と Cb が接続されたとする．そして，オブジェクト A 内の制約式で，現在有効な式の中に a が出現していて，またオブジェクト B 内の制約式で，現在有効な式の中に b が出現している場合でかつ，a と b の型が同じ場合に，これらが同一視される．ここでいう同一視とは，その値の格納場所として同じメモリ空間を共有することを指す．つまり，この2つの変数は全く同じものとして扱われる．

例でいうと，タンクの *outflow* とタンクに接続されたバルブの B の *inflow* は，同一視され，そのことによってベルヌーイの定理より与えられた式が解けてタンクの *outflow* の圧力や，流速が規定されることになる．

3.3 離散的なことに関する記述

離散的な記述が，連続的な記述とくらべて大きく異なるところは，事象のタイミングが明示的に現れることである．つまり，物質あるいは情報の授受のタイミングが明示的に記述できなければならない．このことは，チャネルに対して以下のような操作を書くことで記述される（図9）．

これらの相違点は，チャネル通信の同期のとり方にある．

- 同期通信

  ある1つのチャネルに対して，送信側が **send** で送信し，受信側が **receive** で受信したときに，情報や内流物の受け渡しが完了する．**send** と **receive** が揃わない場合は，先行した方がブロックされる．

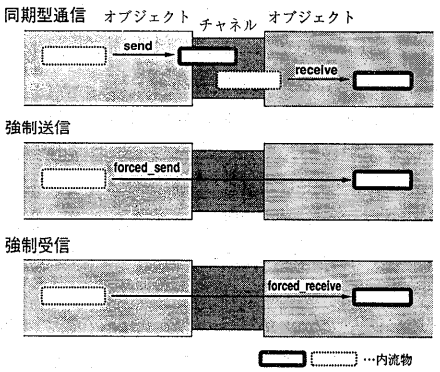


図 9 チャネルでの離散的な情報・内流物の授受
Fig. 9 Three types of discrete parameter passing via channels.

- 強制送信

  ある1つのチャネルに対して，送信側が **forced_send** で送信した場合，受信側の状態にかかわらず，通信は終了する．

- 強制受信

  ある1つのチャネルに対して，受信側が **forced_receive** で受信した場合，送信側の状態にかかわらず，通信は終了する．

タンクの例の **action** ブロックの2つのメソッドは，予約語 **data_driven** によって示されるように，データや内流物の移動にともなって駆動されるメソッドである．つまり，メソッドの駆動条件が，情報あるいは内流物の移動によって規定される．その駆動条件は，予約語 **triggeredby** の後に書かれ，変数名とその内流物の移動形式により記述される．この移動形式には2通りあり，1つはオブジェクトの外から入ってきたことを示す <- と，外へ出ていったことを示す -> である．

例に関して説明すると，最初の *block_in* メソッドは，*water* 型の内流物がチャネルに入ってきて，先に述べた切り替え機構により *inblock* で受け取られたときに駆動される．そして，*discrete_volume* に対して，その *water* 型のオブジェクトの総量を加算する．これと逆のことが，*bolok_out* メソッドで行われる．

->，<- で表されるような，情報，内流物の移動を生む操作について考える．例では，メソッド *block_in* が駆動されるのは，何らかのオブジェクトの中で，タンクのチャンネル *inlet* に対して，**forced_send** が適用されたときで，逆にメソッド *block_out* が駆動されるのは，何らかのオブジェクトの中で，タンクのチャネル *outlet* に対して，**forced_receive** が適用されたときである．

3.4 オブジェクトの生成と接続

以上で，実世界のタンクに対応をとりつつ，連続的なモデル化のときにも，離散的なモデル化のときにも使用できる部品を作成したが，今度はその部品をどうやって使用するかについて述べる．

最終的に，図1の例は，次に示す**図10**のオブジェクトで記述される．

consistof ブロックは，*water_level_stabilizer_system* の中の構成オブジェクトを示し，ここでクラスからバルブ *A* やバルブ *B* やタンク *X* がオブジェクトとして宣言されている．

connection ブロックで，オブジェクト間の接続がなされる．接続としては，*water_level_stabilizer_system* 内のオブジェクト間のチャネルとメソッドの接続と，*water_level_stabilizer_system* 内のオブジェクトのチャネルと，*water_level_stabilizer_system* オブジェクト自身のチャネルとの接続がなされている．*water_level_stabilizer_system* 内のオブジェクト間の接続で接続されなかったチャネルやメソッドは，*water_level_stabilizer_system* 自身のチャネルやメソッドとして再定義される．このことによってオブジェクトの階層性が生じ，さらに上位のオブジェクトに対して制約を記述することで，部品化しにくい複数部品にまたがる制約などの記述が可能になる．もっとも似たアプローチをとっているものに OBSERV[7] の compound objects があるが，これらのオブジェクトがそれ自身の属性やメソッドを持つことはできない．

```
water_level_stabilizer_system: object
  channel
    in inlet;
    out outlet;
  consistof
    valve_A: solenoid_valve;
    tank_X: tank_with_a_level_meter;
    valve_B: variable_state_valve;
    controller;
  connection
    valve_A.outlet = tank_X.inlet;
    tank_X.level_data_outlet = controller.level_data_inlet;
    tank_X.outlet = valve_B.inlet;
    inlet = valve_A.inlet;
    outlet = valve_B.outlet; //チャネルの静的な接続
    controller.open_event = valve_A.open;
    controller.close_event = valve_A.close;
  endobject;
```

図 10 システム全体のオブジェクト
Fig. 10 The object for the whole system.

4. 関連研究

大きく，実行可能な仕様記述とシミュレーション言語の2つの分野の関連研究を挙げ，本研究と比較・対照する．

4.1 実行可能な仕様記述

現在，組み込みシステム構築のための実行可能な仕様記述言語としては，PAISLey[8]，Statecharts[5] をはじめとしてその他多くのものが知られているが，概して外界のモデル化は大まかなものであり，本研究のように外界の振舞いに関して連続的なモデル化までを考慮したものはない．例えば温度のような本質的には連続に変化するものも，「望む温度範囲に入った/範囲から外れた」，という単純な離散事象に置き換えられた上でとらえられる．つまり事象が起こった後のシステムの振舞いのみをモデル化し，どのようにしてその事

象が起こるかということのモデル化は不十分である．

4.2 シミュレーション言語

シミュレーションとは，着目した系の振舞いを計算機内部で疑似的に作りだすことであり，その点では実行可能仕様記述の実行はシミュレーションといえる．

しかし逆に現存のシミュレーション用言語での記述は必ずしも仕様とはいえない．なぜならシミュレーション言語での記述の最大の目的は結果的にシミュレーションが行えることであり，仕様記述としては必須のⅰ）モジュール性，ⅱ）それによる階層構造，などは二の次であったからである．

こういった問題も考慮して設計されたシミュレーション言語としては Dymola[3] と COSMOS[6] が知られている．しかし，それぞれ以下の問題がある：

- Dymola は連続系しか扱えない．文献2)では Dymola はより大きな総合的な多階層のシミュレーション枠組の1つのツールとして位置付けられている．この枠組では，離散系も考慮に入れているが，離散系と連続系は実質は全く別個に扱われるため，両系のシミュレーションはできない．
- COSMOS は離散系も扱え，混合系シミュレーションをサポートする．しかし，1つのオブジェクトが部品として，離散的記述および連続的記述のどちらにも用いられてもいいような機構は用意していない．

5. まとめ

プラントの制御システムの構築のためには，その制御システムを取り巻く環境もモデル化することが重要であるが，特にプラントの制御システムの外界には次のような特徴的な要素がある．

- 連続的に生成される情報
- プラントを構成する設備内を流れる内流物

こういった要素をモデル化するために，本研究ではオブジェクト指向的なモデル化，連続・離散の混合系のサポート，および多様なコンテキストで適用可能な部品の作成による再利用という3つの基本概念のもとに，その記述法と機構を提案した．

また，実行可能な仕様記述とシミュレーション言語という大きく2つの観点から，関連研究を挙げ，それらと本研究を比較・対照した．

今後の課題としては，効率の良い実行・シミュレーションの環境の実装が第一に挙げられるが，それは数式処理や数値解析技術も含む統合的な環境となる．現在，そういった環境の中核をなす，Eunice コンパイラ，およびグラフィカルなフロントエンド，グラフィカルなシミュレーション環境のプロトタイプを実装中である．

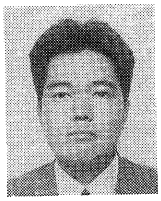
謝辞 研究およびその他様々な点において御指導，御助力いただいた東京大学理学部情報科学科の坂村健助教授に感謝いたします．また，有意義な議論の機会を与えて下さった電気通信大学大学院情報システム学研究科前川研究室の皆様に感謝いたします．最後に，流体力学についての資料を提供し，本論文中の方程式の検証をして下さった東京大学工学部機械情報工学科の藤原仁志氏に感謝します．

参考文献

1) Agresti, W. W. (ed.): *Tutorial: New Paradigms for Software Development*, IEEE Computer Society Press, Washington, D. C. (1986).
2) Cellier, F. E., Zeigler, B. P. and Cutler, A. H.: Object-Oriented Modeling: Tools and Techniques for Capturing Properties of Physical Systems in Computer Code, in Baker, H. A. (ed.), *Computer Aided Design in Control Systems*, IFAC Workshop Series international Federation of Automatic Control, Pergamon Press, Oxford/New York (1992); Selected Papers from the IFAC Symposium, Swansea, UK, 15-17 (1991).
3) Cellier, F. E. E. and Elmqvist, H.: Automated Formula Manipulation Supports Object-Oriented Continuous-System Modeling, *IEEE Control Systems*, Vol. 13, No. 2, pp. 28-38 (1993); Special Issue on Computer-Aided Control Systems Design.
4) Gehani, N. and McGettrick, A. D. (eds.): *Software Specification Techniques, International Computer Science Series*, Addison-Wesley, Massachusetts (1986).
5) Harel, D.: Statecharts: A Visual Formalism for Complex Systems, *Sci. Comput. Programming*, No. 8, pp. 231-274 (1987).
6) Kettenis, D. L.: COSMOS: A Simulation Language for Continuous, Discrete and Combined Models, *Simulation*, Vol. 58, No. 1, pp. 32-41 (1992).
7) Tyszberowicz, S. and Yehudai, A.: OBSERV—A Prototyping Language and Environment, *ACM Transactions on Software Engineering and Methodology*, Vol. 1, No. 3, pp. 269-309 (1992).
8) Zave, P.: An Operational Approach to Re-

quirements Specification for Embedded Systems, *IEEE Trans. Softw. Eng.*, Vol. SE-8, No. 3, pp. 250-269 (1982) ; Reprinted in Ref. 1) and in 4).

(平成5年9月20日受付)
(平成6年4月21日採録)



小林 久浩

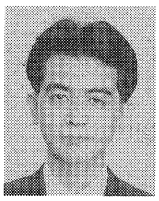
1969年12月17日生．1994年3月東京大学大学院理学系研究科情報科学専攻修士課程修了．同年NTTデータ通信(株)に入社．修士課程では主にオブジェクト指向モデリング，実行可能な仕様記述，ソフトウェア開発における詳細化に関しての研究に従事した．修士（理学）．

河田 恭郎

1965年8月5日生．1992年3月東京大学大学院理学系研究科情報科学専攻修士課程修了．以後電気通信大学大学院情報システム学研究科情報システム設計学専攻助手．実行可能な仕様記述を中心とした統合ソフトウェア開発環境の研究に従事．修士（理学）．

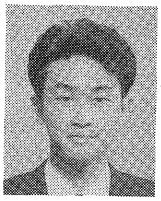
前川 守（正会員）

1942年3月8日生．1965年3月京都大学工学部数理工学科卒業．同年東京芝浦電気(株)入社．東京大学理学部情報科学科助教授等を経て，現在電気通信大学大学院情報システム学研究科教授．主としてコンピュータソフトウェア，ソフトウェア開発環境，分散システム，OS等の研究に従事．Ph. D.



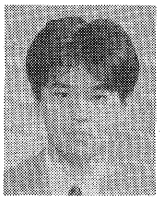
川崎 章

1969年10月29日生．1994年3月電気通信大学大学院情報システム学研究科情報システム設計学専攻博士前期課程修了．同年松下電器産業(株)に入社．修士課程では主に実行可能な仕様記述言語とその視覚的環境に関しての研究に従事した．修士（工学）．



藪 彰文

1969年11月11日生．現在電気通信大学大学院情報システム学研究科情報システム設計学専攻博士前期課程に在学中．実行可能な仕様記述言語，エレベータシステム，主に制御系システムにおける離散・連続混合系，および部品化に関しての研究に従事．



小野川公也

1970年6月13日生．現在電気通信大学大学院情報システム学研究科情報システム設計学専攻博士前期課程に在学中．実行可能な仕様記述の研究に従事．