論　文

組込み型制御システムのためのプログラムモデルとそれに基づくプロトタイプ設計支援システムの開発

手嶋　茂晴[†]　　稲森　豊[††]　　阿草　清滋[†]

EFS Program Model for Embedded Real-Time System and EFS-Based CAE Tool Schetch

Shigeharu TESHIMA[†], Yutaka INAMORI[††], and Kiyoshi AGUSA[†]

あらまし　組込み型制御システムの中でも，プログラムの応答時間が被制御系の時間的応答と同程度のシステム，リーンリアルタイムシステム（lean real-time system）の設計を対象とした「組込み型制御システム プロトタイプ設計支援システム：Schetch」を作成した．Schetch は，組込み型制御システムの動作記述体系とそのシミュレータから構成される．Schetch の動作記述においては，組込み型制御システムのソフトウェアが簡潔に表現できるように周期実行やイベント駆動実行の機能や実時間表現のためのデータ型の機能を導入した．このうち，実時間表現のためのデータ型（実時間時系列型データ）は我々が提案する新たな概念である．この支援システムを自動車用車載コントローラの制御アルゴリズムの設計に適用し，有用性を示した．

キーワード　リアルタイムシステム，機器組込みシステム，ソフトウェア工学，プロトタイピング，時系列処理

1. まえがき

我々の身の周りには，マイクロプロセッサが組み込まれ，それによって機器を制御している機器が数多く存在している．このような組込み型制御システムの設計は，制御系としての設計と計算機システムとしての設計（主には内蔵プログラムの設計）の二つの側面をもつが，近年製品の高機能化や機械部品の標準化が進んでいるため，制御システム全体の設計のうちソフトウェア設計の重要性が増加している．

ここでは，組込み型制御システムの中でも，プログラムの応答時間が被制御系の時間的応答と同程度のシステムの設計を対象とする．我々はこのような性質をもつシステムをリーンリアルタイムシステム（lean real-time system）と呼ぶ．このリーンリアルタイムシステムの性質は被制御系の時間的な要求に対して必要最低限の性能をもつCPUによって制御系を構成することを意味し，コスト制約の厳しい製品では一般的な条件である．例えば，自動車用車載コントローラ[1]，家庭用ゲーム機などが当てはまる．このような性質をもつ組込み型制御システムのソフトウェア設計は，製品設計の場では日常的であるにもかかわらず，現在，有効な設計方法論が確立しておらず，設計を支援するツールも十分でない．

これまでの制御理論はマイクロプロセッサを用いて制御システムをプログラムで実現することを仮定していないため，ソフトウェア設計には対応できていない．ソフトウェア工学の分野では構造化分析などでリアルタイム性をもつシステムへの拡張が試みられているが[2],[3]，設計工程に対しては有効な手法は知られていない．一方で，形式的なアプローチにおいては，プロセス代数やペトリネットの時間概念に関する拡張などが研究され，それに基づく記述体系や解析ツールが提案されている（例えば，LOTOS/T，High-Level ペトリネットなど[4]〜[7]）．しかし，それらの動作モデルはいずれも非決定的であるため，実行時環境での動作保証を必要とする組込み型制御システムには適していない[8]．また，これまでのリアルタイムシステムの研究では，実行に十分な計算機資源を用いることを前提としたもので，我々が対象とするようなリーンリアルタイムシステムは研究対象とされていなかった．

† 名古屋大学大学院工学研究科情報工学専攻，名古屋市
Department of Information Engineering, Nagoya University, Nagoya-shi, 464 Japan
†† (株) 豊田中央研究所，愛知県
Toyota Central R&D Labs., Inc., Aichi-gun, Aichi-ken, 480-11 Japan

そこで，我々はリーンリアルタイムシステムである組込み型制御システムの設計手法構築の前提となるプログラムモデル EFS モデルを提案し，それに基づいて組込み型制御システムの設計全工程を支援する「システム：組込み工房」を作成している．組込み工房のうち，プロトタイム設計を支援する「プロトタイム設計支援システム：Schetch」に関しては作成を完了し，設計現場での試行を開始している．

EFS モデルは，それに基づいたプログラムにおいて実時間表現のためのデータ型（実時間時系列型データ）を導入し，その型のデータへの操作によって実時間軸上の時系列を処理することを可能にしている．Schetch における動作記述では EFS モデルに基づいたソフトウェアが簡潔に記述できるように周期実行やイベント駆動実行の機能や実時間時系列型データへの操作機能をもつ．このうち，実時間時系列型データは新たに導入する概念である．実時間時系列型データは必要な信号値変化を過去にさかのぼって参照を可能にする仕組みをもつデータ型である．これによって，時系列信号の処理の際，過去の信号値系列を明示的に保存する必要がなくなり，状態を意識しないプログラムの作成が可能となり，データフローを中心とした見通しのよいプログラム構成が実現できる．

時系列を処理する機能としては時系列信号処理を対象とした DSP (Digtal Signal Processor) アプリケーション開発用にプログラミング言語の拡張がいくつか提案されている [9]〜[11]．これらは，効率的な DSP コードの生成に主眼をおくもので，設計のターゲット言語としての記述能力は高いが，リアルタイムシステムの設計には必須である時間的な制約を記述することができず，設計の上流工程での適用に適していない．また，制御システムの設計の代表的なツールである Matlab/Simulink [12]〜[14] などでは，データフローによって系を記述するが，これらデータフローの記述は定められた最小サンプリング時間を基本クロックとする同期型の記述である．そのため，時系列を処理するためには別途メモリ機能や遅延によって明示に状態を設定する必要があり，我々の提案する実時間時系列型データ型の概念はもたない．また，実時間型の制御システムの記述に適すると言われている同期型言語およびその処理系 [15]〜[19] についても同じく時系列型のデータを簡潔に処理する機能をもたない．

本論文ではまず，我々が現在作成している設計支援システムの全体構想を述べ，その中で，プロトタイプ用設計支援の位置づけについて説明する．そして，リーンリアルタイムシステム向けのプログラムモデル EFS モデルを定義し，それに基づいて作成しているプロトタイプ用設計を対象とした支援システム Schetch を紹介する．また，Schetch を自動車用車載コントローラの制御アルゴリズムの設計に適用した例を示す．

2. 設計支援システムの全体構想

図 1 に設計支援システム組込み工房の全体構成を示す．

組込み工房は設計フェーズに対応して大きく二つの部分システムから構成される．

図中上半分は上流設計，つまりプロトタイピングの設計段階を支援するプロトタイプ設計支援システム (Schetch) で，下半分はソースプログラムを具体的に設計する詳細設計支援システムである．

図 1 にあるようにソフトウェア設計をオブジェクト実行環境に依存しない合成と実行環境に合われた最適化の 2 段階に分ける考え方は文献 [20] で提案された方法論である．文献 [20] は組込み制御システムのソフトウェア設計を対象としたものではないが，その方法論は組込み制御システムにおいても有効である．

プロトタイプ設計支援システム Schetch において，データフロー/イベント駆動によるコントロールフローで表される機能的な動作を図的に記述する．ここでは，機能的な動作の記述であるので，具体的なスケジューリング方法やデータのデータ構造/数値精度などプロ

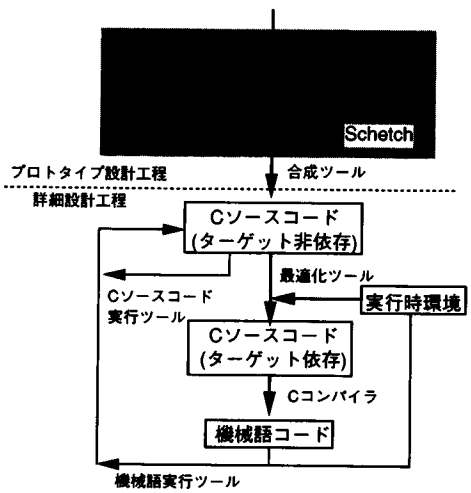

図 1 組込み工房の全体構成
Fig. 1 Total structure of NS-works (Kumikomi Kobo).

グラム実現に関する記述は行わない．Schetch の動作記述は合成ツールによってCソースコードに変換される．但し，ここでのCソースコードは，ターゲットの実行環境（タスクスケジューリングや実時間時系列型のバッファサイズ，ターゲット CPU による内蔵タイマ，入出力ポートといったターゲット CPU の差異によるハードウェア資源へのアクセス手段）などの情報をもたず，詳細設計工程でターゲット CPU や制御対象別にカスタマイズ/最適化されることを前提にしている．

一方，組込み工房では設計の各レベルで，記述をシミュレーション/検証するツールを用意している．Schetch においては，動作記述はトランスレータによって制御系シミュレータ用言語に変換され，シミュレーションが可能である．この段階においては実行処理の遅延や数値処理の誤差をなどハードウェアの制約を考慮しない実行によってプログラムの機能的な動作を確認することができる．また，Schetch の動作記述から合成されたCソースコードについては，クロス環境（UNIX および MS-Windows）で実行するツールを用意している．更に，詳細な動作確認に関してはターゲット CPU の機械語にまで落された機械語を CPU クロック時間単位でのシミュレーションによって，クロス環境での実行ツール [21] を提供する．

3. 組込み制御システム用プログラムモデル EFS モデル

3.1 プログラムの全体構造

プログラム制御が用いられるようになる以前，機器組込み制御システムはアナログ電気回路若しくは一部機械部品を用いて実現されてきた．現在でも機器組込み制御システムにおけるプログラムはこれら機械部品の置換えと考えられている．従って，制御分野で用いられるようなブロック線図は機器組込み制御システムのプログラムモデルとしても親和性が高い．

そこで，我々は機器組込み制御システムのためのプログラムモデルとしてブロック線図を基本にして，それに抽象化された実時間時系列処理機能をもつプログラムモデル EFS（Event-driven Finite State）モデルを定めた．

EFS モデルは以下のように定義される;

（1） システムは機能ブロックとその間のデータ線から構成される．

（2） 機能ブロックには，入力と出力があり，入力値の時系列に対して出力が決まる．

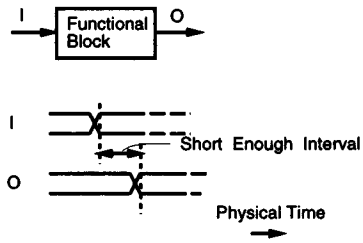

図2 機能ブロックの処理時間
Fig.2 Execution time of a functional block.

（3） 機能ブロックは，入力の変化後，十分に短い時間で出力を変化させる．但し，その時間遅れは，処理系に依存する（図 2 参照）．

（1）は，データフロー言語に共通する特徴である．（2）は陽に時系列を入力として扱うためのもので，このモデルの特徴である．（3）は制御される系が動作する物理的な時間軸（物理時間）と手続き型プログラムの実行が定義される論理的な時間軸（論理時間）との関係を規定するものである． 物理時間と論理時間という二つの時間軸の上に動作を定義する方式は，ハードウェア記述言語 [22] で一般に使われているものである．ハードウェア記述言語では，物理時間の遅れを手続きの流れとは別に宣言的に与えるので，論理時間は物理時間に対して無限小としている．ソフトウェアでは論理時間の積み重ねが，物理時間となるので，論理時間を物理時間に対して無限小とすることはできない．本モデルではプログラム各部分の時間遅れを意識しないで，全体として，実時間制約を満たしたプログラムを構成するためには（3）の性質は必要である．

3.2 実時間時系列型データ

このモデルは物理時間を時間軸とする時系列に対して，動作を定義するモデルである．つまり，データ線を流れるデータは時間によって決まる値であり，データ線はデータの時系列として形式化できる．

モデル上，物理時間を時間軸とする時系列を実時間時系列という形で定義する．

実時間時系列とは，T を 0 以上の実数集合，V を値の集合とするとき，

$< v_1, t_1 >< v_2, t_2 > \cdots < v_n, t_n >$

（但し，$t_i \in T, t_i < t_{i+1}, v_i \in V$）

なる列である．

但し，以後，本論文では実時間時系列をプログラムのデータ型という面を強調し，実時間時系列型データ

と呼ぶ．

実時間時系列型データをプログラム中で扱うとき，時間の経過に対して値を保存する/しないという性質の違いを意識する必要がある．実時間時系列型データには上記の性質の違いによって，イベント型とステイト型の二つのタイプがあると考える．従って，データ線を，イベント型データをとるデータ線であるイベント列型と，ステイト型のデータをとるデータ線であるステイト列型に分類する．

- イベント型：時間の経過に対して，データ線上に値が保存されない．

発生した時刻のみに意味をもつ．この型のデータを単にイベントと呼ぶ．イベント型における値は，イベントの種類を表すものである．

- ステイト型：時間の経過に対して，データ線上に値が保存される．

外部割込み信号や機能ブロックを制御する制御信号および非同期的な入出力がイベント型のデータに相当する．また，変数は次の書込みまで値が保持されるのでステイト型に相当する．

イベント列型のデータ線では，時刻 t_i で値 v_i のイベントが発生したことを意表する．また，$t_i < t < t_{i+1}$ なる時刻 t では，値は定義されない．

一方，ステイト列型のデータ線では，同じ表現の実時間時系列型データでは $t_i \leq t < t_{i+1}$ なる時刻 t で，データが値 v_i をとる．

3.3 機能ブロックの実行

システムはイベント駆動方式で制御されるものとし，機能ブロックは指定された入力データ線上にイベントが発生するたびに出力を更新するものとする．

また，機能ブロックは起動後，十分短い時間ではあるが不定時間後に実行を完了するものとする．但し，機能ブロックの内部では「機能ブロックが起動されると，そのブロック内では，物理時間の進みが停止し，完了時に時間の遅れを一気に戻して，完了時点の正確な物理時間（機能ブロックの計算時間を考慮した時間）でイベントを発生させる」と考える．これによって，プログラマは機能ブロックの内部計算によって生じる遅延とデータの時系列を定義している時間（物理時間）の関係付けを意識することなく，機能ブロックの内部を記述することが可能なる．

3.4 機能ブロックの形式的定義

機能ブロックは，$< I, S, O, F_o, F_s >$ によって定義される．

I ： 入力データ線の並び
S ： 状態変数の並び
O ： 出力データ線の並び
F_o: 出力を決める関数
F_s: 次状態を決める関数

I にあるデータ線のうち，イベント型のものを $i_{e1}, i_{e2}, \ldots$，ステイト型のものを $i_{s1}, i_{s2}, \ldots$ とし，また，S を $s_1, s_2, \ldots$，O を $o_1, o_2, \ldots$ とする．このとき，F_o, F_s は次のような関数である；

$$F_o(i_{e1}[:t], i_{e2}[:t], \ldots, i_{s1}[:t], i_{s2}[:t], \ldots) = < o_{v1}, o_{v2}, \ldots >$$

$$F_s(i_{e1}[:t], i_{e2}[:t], \ldots, i_{s1}[:t], i_{s2}[:t], \ldots) = < s_{v1}, s_{v2}, \ldots >$$

ここで，$i_{e_k}[:t]$, $i_{s_k}[:t]$ は実時間時系列 i_{e_k}, i_{s_k} の部分列であり，$i_{e1}, i_{e2}, \ldots$ のいずれかにイベントが存在する時間 t までの実時間時系列を表す．

また，$< o_{v1}, o_{v2}, \ldots >$ は時間 $t + \delta t (\delta t > 0, \delta t$ は機能ブロックの処理時間）にとる出力データ線の値の並びである．つまり，出力データ線 o_k の実時間時系列で，$< o_{v_k}, t + \delta t >$ なる要素が現れることを意味する．

同様に，$< s_{v1}, s_{v2}, \ldots >$ は時間 $t + \delta t$ にとる状態変数の値の並びである．

4. プロトタイプ用設計支援システム Schetch の実現

4.1 実現方針

組込み型制御システムの設計では，従来からプロトタイプを用いた実機実験をもとに系の物理的な挙動やプログラム入出力関係を明らかにしていきながら，要求分析およびシステム設計を行ってきた．しかし，実機を用いた実験は，最終的な動作保証のためには必須であるが，実験機器の台数や利用時間の制限，特性のバラツキなどを考えると設計段階においてはプロトタイプのシミュレーションを併用する必要がある．

そこで，我々は支援環境におけるシミュレーションエンジンおよび動作記述エディタとして，Mathworks 社の制御系の設計/シミュレーション CAE 環境である Matlab/Simulink(注1) を用いることとし，Matlab/Simulink に M 言語に我々の提案する組込み型制御システムの EFS モデルに対応するパッケージを追加した形で，Schetch を実現した．

表1　ライブラリーとして提供される機能ブロックタイプ（一部）
Table 1　Functional block types provided by Schetch (Subset).

機能ブロックタイプ名	処理内容
Table_Reference	条件から関数テーブルを引く
Conditioned_Duration_Judge	条件が設定した時間以上成立したら，1，それ以外 0
Conditioned_Duration_Timer	条件が継続している時間幅を出力
Gobal_Timer	Event によって set/reset されるタイマ．但し，複数のブロックから操作/参照可能．
Conditioned_Event_Generator	条件が成立したときにイベントを発生
Positive_Trigger	データの立上り時にイベントを発生
Negative_Trigger	データの立下り時にイベントを発生
Event_Filter	条件成立の間，Event を通過

Schetch と Matlab/Simulink のツールボックスとの違いは，Schetch は，Matlab/Simulink の動作モデルではなく，EFS プログラムモデルに従って動作が定義されることである．

Schetch の提供する組込み型制御システム・システムパッケージは，

- イベント駆動方式のシミュレーション
- 実時間時系列型データ
- 上記機能を利用した設計パターンのブロックタイプライブラリー

を提供するものである．

但し，組込み型制御システムのプログラムモデルを Matlab/Simulink 上に実現する制約から Schetch のシミュレーションの特徴として，機能ブロックの実行遅れ時間は Matlab/Simulink の数値解析の 1 ステップ分の時間(注2)となる．

Schetch では記述上の便宜を図るため，制御用のプログラムでよく用いる機能ブロックの記述パターンをあらかじめタイプ化して機能ブロックタイプライブラリーとしてシステム側で用意した．Schetch が提供するライブラリーに含まれる機能ブロックタイプで代表的なものを表 1 に示す．このうち，Table_Reference を除く機能ブロックタイプはすべてに実時間時系列型データを取り扱うもので，Matlab/Simulink では意味をなさず，Schetch の環境においてのみ有効である．

4.2　イベント駆動方式のシミュレーション

Matlab のシミュレーションは，連続関数の数値解析に基づく手法なので，時間幅をもたないイベントをそのまま扱うことはできない．しかし，イベントを極めて短い時間区間に行われるデータ変化で近似的に表現し，かつ，シミュレーションの速度や効率を考慮しなければ，仮想的にイベント駆動方式を実現することができる．

イベントを <trigger, value> という 2 本の信号線上に表現する．Matlab 上の信号線は倍精度実数のデータをとるが，trigger 信号線の値はそのうち，“0”，“1” の 2 値を使い，信号線が “1” をとるとき，そのタイミングでイベントが発生したとみなす．また，そのときの value 信号線のとる値をイベントの値と考える．

上記のようにイベントを定め，以下の性質を満たすイベント駆動機能ブロックを実現した．

- イベント駆動機能ブロックは，一つまたは複数のイベント線および信号線を入力に，一つまたは複数のイベント線を出力にもつ．
- 入力のイベント線のいずれかイベントが現れたとき，機能ブロックは計算を行い出力を更新する．

Matlab/Simulink のもともとの関数はデータフローによって制御されている．従って，入力のどれか一つでも値が変化すると，その結果が出力（内部記憶をもつ場合は，内部記憶も）に反映される．しかし，我々のプログラムモデルは，イベント型データの入力にイベントが入力されたタイミングで計算を行うものなので，イベント駆動機能ブロックの内部では入力と出力それに内部記憶にそれぞれラッチを付加し，イベントによって起動されたときにだけラッチの内容を更新するようにする．図 3 にイベント駆動機能ブロックの内部構造を示す．このうち，利用者が指定する部分は外部とのデータ線と網かけされた “Function” の部分だけで，残りの部分はイベント駆動機能ブロックを定義するとシステム側で自動的に付加する．

(注 1)：Matlab は制御系のシミュレータで，tty ベースのコマンドで系の記述やシミュレーションの制御を行う．Simulink は，Matlab に対する window interface である．
Matlab/Simulink が対象とするものは，連続系の制御系で，ブロック線図を書く要領で基本ブロックを結線していくことによって系の記述を行う．シミュレーションは，基本ブロックを算術式に置き換え，オイラー法，ルンゲ—クッタ法など数値解析手法によって行う．
(注 2)：Matlab/Simulink のシミュレーションの 1 ステップ分の時間は，物理時間とは独立に十分小さくすることが可能であるので，EFS モデルの定義（3）を満たす．

4.3 実時間時系列型データの実装

データ線，内部状態（メモリ）をすべて実時間時系列型データとみなす．実時間時系列型データから必要な情報を取り出すために，State_Logger，Event_Logger の二つのライブラリーブロックを提供する．

State_Logger は，ステイト列型のデータに対するアクセスを提供するブロックで図 4(a) に示すように $(\mathrm{timeIn}_i, \mathrm{dataOut}_i)$ $(i = 1, 2 \ldots)$ の入出力の組をもつ．現在時刻との相対時間を timeIn_i に印加するとその時刻のメモリの値を $\mathrm{dataOut}_i$ に返すものである．

Event_Logger は，イベント列型のデータに対するアクセスを提供するブロックで図 4(b) に示すように last, last2, last3, ... という出力をもつ．last は最も最近，last2 は一つ前に生成されたイベントの発生時間と値を返す．

表 1 に示すような実時間時系列型データを取り扱うブロックは，この State_Logger と Event_Logger の二つブロックをもとに，記述パターンに合わせて作成したものである．

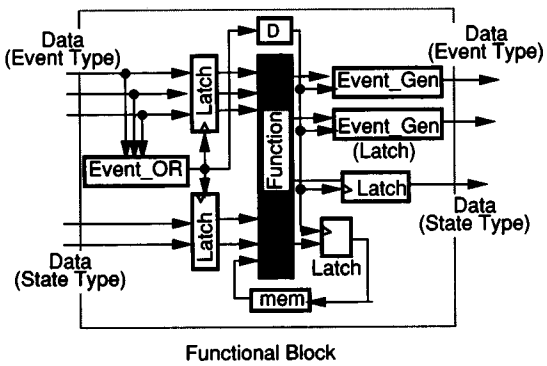

図 3 Matlab/Simulink におけるイベント駆動の実現
Fig. 3 Implementation of event-driven block in Matlab/Simulink.

4.4 記 述 例

以下に示す受信機の受信リモコン信号処理部の仕

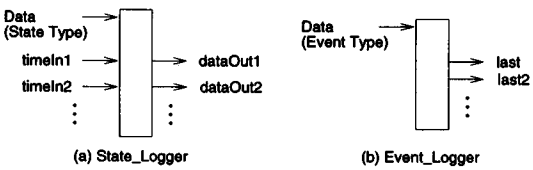

図 4 State_Logger，Event_Logger の入出力
Fig. 4 Inputs/Outputs of State_Logger and Event_Logger.

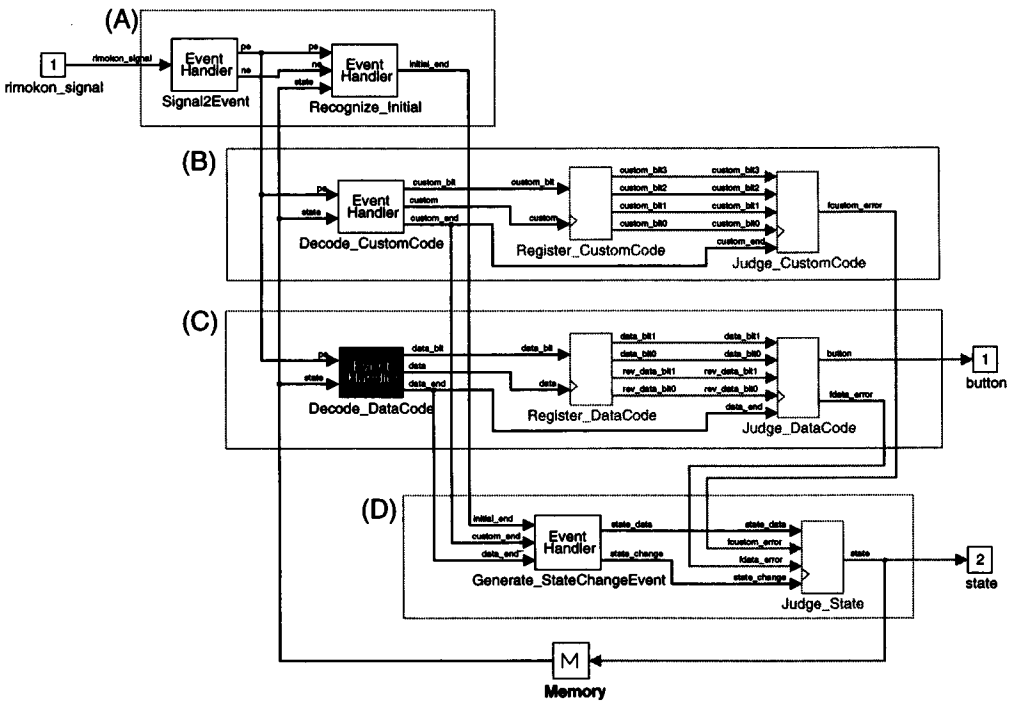

図 5 受信リモコン信号処理部の全体
Fig. 5 Signal process unit of remote controller.

様を実現する動作を記述した（詳細仕様を付録 1. に示す）.

- 受信リモコン信号は，リモコンのどのボタンが押されたかをパルスで表したものである．信号は，初期信号，カスタム信号，データ信号からなり，この順で送られてくる．
- 出力は，受信リモコン信号をデコードした数値（押されたボタンに相当）である．

図 5 は，全体の処理の流れ，データの流れを記述したものである．ここでは，信号の立上り/立下りをすべてイベントとしている．

図 5 中，(a) は初期信号を認識する部分，(b) はカスタム信号を認識する部分，(c) はデータ信 号をデコードする部分であり，(d) は，(a) (b) (c) を起動するシーケンスを決める部分である．(a) (b) (c) のいずれを起動するかを状態として保存し，イベントによって，(a) (b) (c) の部分に起動をかけている．

図 6 は図 5 中，黒く反転して示す Decode_DetaCode モジュールの処理内容を記述したものである．このモジュールは，リモコン信号のデータコード部分のパルスを一つずつ取り込み [（1）data_signal_filter]，パルス幅を測定し [（2）pulse_width_timer]，ビットデータ（0 or 1）に変換する [（4）pulse_to_bit].

Decode_DetaCode モジュールで中心的な働きをするブロックは以下に示す機能を実現する．

- 機能ブロック data_signal_filter（機能ブロックタイプ：Event_Filter）は図 5(c) の部分が起動されているときのみ，イベントを通過させる．
- 機能ブロック pulse_width_timer（機能ブロックタイプ：Global_Timer）は続いて発生した二つのイベントの間の発生時間の差を計り，パルス幅を計算する．
- 機能ブロック pulse_to_bit（機能ブロックタイプ：Table_Referece）はパルス幅が 0，1 のデータとしてデコードすべき値かをテーブルを引いて調べる．

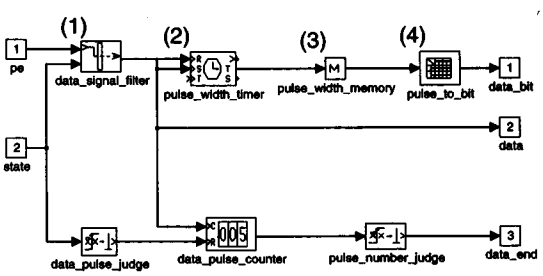

図 6　Decode_DetaCode モジュールの内部構造
Fig. 6　Structure of Decode_DetaCode module.

5. 車両制御への適用

ブレーキ油圧制御するシステムである ABS (Antilock Brake System) 制御 [23], [24] に適用した結果を示す．

5.1 ABS 制御のあらまし

ABS は，タイヤがロックしない範囲の最大圧でブレーキをかけるようにブレーキ圧を制御するシステムである．ロックしない範囲での最大ブレーキ圧を厳密に求めるには，路面摩擦係数や路面に対する車体の絶対速度など今の技術では計測不可能な物理量が必要であるので，実際の制御アルゴリズムでは，近似的にその機能を実現している．

具体的には以下のように動作する;

- 車軸回転に同期したパルス（車輪速パルス）を取り込む（実際に車載されている ABS では 4 輪分を入力するが，本論文で例示するものは簡略化のため 1 輪分を入力とする）.
- 車輪速から車体状態/路面状況を推定する．例えば，車体速，車体加速度，路面摩擦，異径タイヤ装着の有無．
- 車体状態/路面状況 に応じて，ブレーキ圧を調整する．

図 7 に ABS 制御システムの全体構成を示す．ブレーキ圧の増減は，油圧システム（Oil Pressure System）にある電磁バルブを開閉することによって行われる．二つの電磁バルブによって，ブレーキに対して増圧 (Up)/保持（Keep)/減圧（Down）の 3 状態を実現する．ブレーキ圧の増減の度合は，増圧と保持，若しく

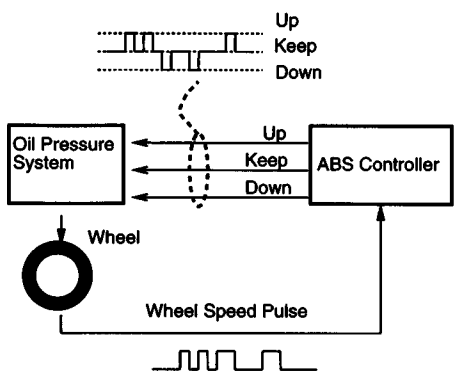

図 7　ABS 制御システムの構成
Fig. 7　Context of ABS System.

は，減圧と保持を繰り返すパルスの幅や継続時間によって調整する．

5.2 Schetch 動作記述による ABS 制御の記述

Schectch が提供する機能記述体系によって，一輪の ABS 制御プログラムのプロトタイプ[25]を記述した．プロトタイプの構成は以下のようになっている（図 8）．

- 各機能ブロックは 大域変数（ステイト列型データ）を介してデータの受渡しを行う．
- 車輪速パルスをイベントとした．
- 車輪速パルスをカウントアップする機能ブロックのみを，車輪速パルス（イベント）によって起動．
- 内部タイマが発する一定周期ごとのイベントが他の機能ブロックを起動．

プロトタイプ記述は 3 階層から構成される．図 8(a) 最上位の記述で，各モジュール間のデータフロー，イベントフローを表現している．図 8(b), (c) は第 2 階層の記述で，モジュールの内部構造を Schetch が提供する機能ブロックの結線によって記述する．図 8(d) は第 3 階層の記述で，ライブラリーブロックの記述のうち，設計者がカスタマイズする部分を示す．ライブラリーブロックの内部構成のうち，設計者がカスタマイズする部分は Matlab/Simulink の基本ブロックの組合せで構成される．一方，イベントや実行周期の記述など EFS モデルを実現している部分は設計者には隠ぺいされている．Schectch が設計者のカスタマイズ部分を EFS モデルの動作モデルに従うように周辺に必要なブロックを追加している．

5.3 シミュレーション実施例

ABS システムは，車両および路面によって，制御システムにフィードバックをもつシステムであるが，ここでは，制御システム単独で開ループのシミュレーションを行った結果を示す．

入力は，車輪速パルス（1 輪分）の時系列である．図 9 に最初 0.2 sec 分の車輪速パルスを示す．

車輪速パルス（Wheel Speed Pulse）から推定され

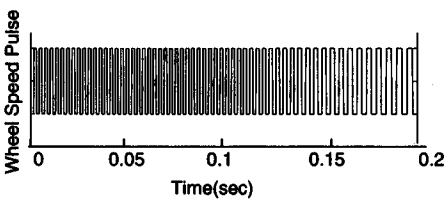

図 9 シミュレーションの入力（車輪速パルス）
Fig. 9 Input pattern for simulation (Wheel Speed Pulse).

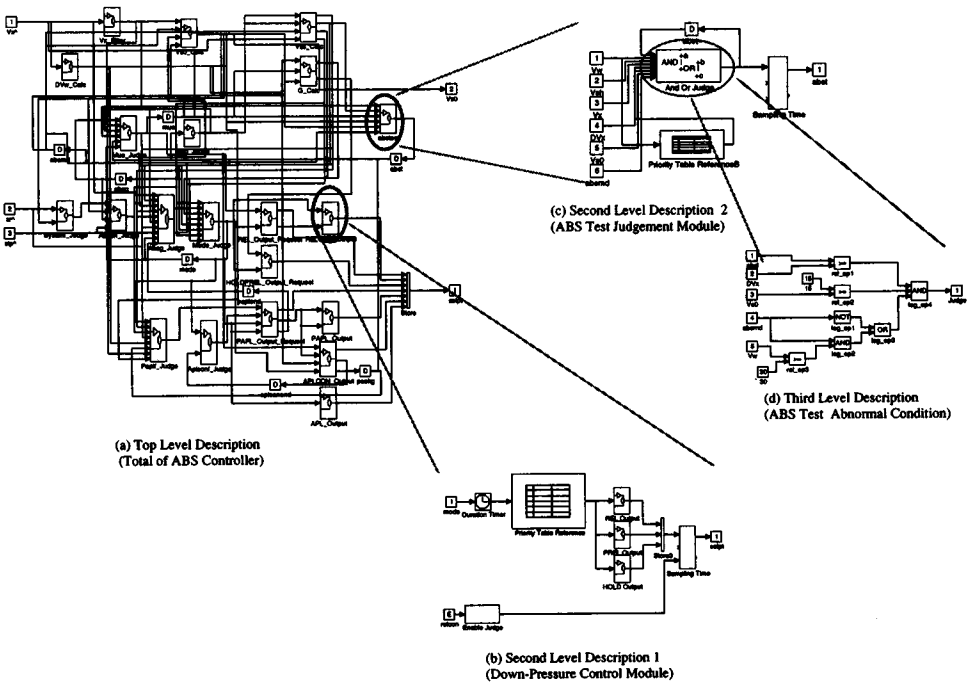

図 8 Schectch 動作記述による ABS 制御システムプロトタイプ記述
Fig. 8 Description of prototype of ABS controller in Schectch/M.

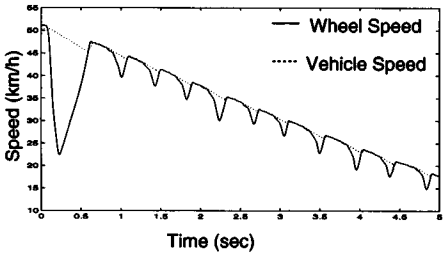

図 10　シミュレーション結果（車輪速度，車体速度）
Fig. 10　Results of simulation (Wheel Speed and Vehicle Speed).

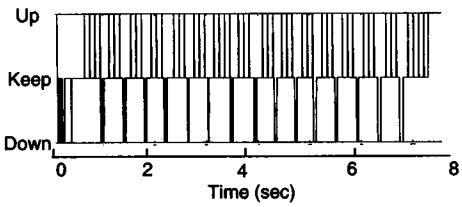

図 11　シミュレーション結果（ブレーキ油圧制御出力）
Fig. 11　Results of simulation (pulse for oil pressure system).

た車輪速（Wheel Speed），車体速（Vehicle Speed）の変化の様子を図 10 に示す．また，制御システムの出力であるソレノイドへの出力の変化の様子を図 11 に示す．

このシミュレーションによって，車輪速と車体速の差が広がってくる，つまり，タイヤが滑べり出し始めたタイミングでブレーキ油圧を下げるように，また，車輪速と車体速の差が縮まるタイミングでブレーキ油圧を上げるように制御している様子が理解できる．制御対象の車両を含めた系全体のシミュレーションではないので，制御そのものの正確な評価はできないが，プログラムプロトタイプの挙動を解析するには十分の結果が得られている．

5.4　Schetch 導入の効果

一般に，実時間性のある制御システムの記述する場合，データフローとコントロールフローが混在し，記述を簡潔にすることが難しいとされているが，我々の提案した実時間時系列型データによる実時間処理機能によって，大部分のコントロールフローを設計者から隠ぺいすることができた．具体的には，Schetch による ABS 記述では，割込み処理以外のすべての機能ブロック間のデータ線だけで関係づけることができた．設計者にとっては，記述上，システムがデータフローに基づいて各機能ブロックが並列動作しているのと同じようにみなすことができるため，上流工程の設計において，設計者の思考形態と極めて親和性が高いことが確かめられた．

また，C ソースプログラムの設計仕様書（A4 約 200 ページ）に記載されている内容の 9 割を Schetch に置き換えることができた．残り 1 割は実験結果の数値がプロットされてたグラフによる例示や仕様の根拠を説明する文章であった．このことは，ソースプログラムの設計仕様作成においてシステムのプロトタイピングが可能となることを意味する．

これまで，ABS 制御システムのプログラムにおいては，プロトタイピングは最終的なソースコートと同様，プログラムを ROM 化して，制御プロセッサ実機による実験によって行われてきた．プロトタイプ設計支援システム Schetch を導入することによってこのような実機による動作確認を前提としたプロトタイピング工程が，シミュレーションよる仮想的な実行による動作確認を併用するようにすることが可能となり，以下の効果が明らかになった．

- 動作確認にかかる時間の短縮

実機を用いた場合，実行まで少なくとも数時間かかっていたものが，数分から数十分程度に短縮できた．

- 仕様変更およびレビューの効率化

制御仕様の変更によって，動作記述の変更を行うとき，Schetch を用いた場合，従来の印刷物としての仕様書と C ソースプログラムを用いた場合と比べ，変更箇所の検索から実際の変更までに要する時間を従来の 30%から 70%に削減することができた．

上記の定量的な効果のほか，定性的には以下の効果が確認できた．

- プロトタイプ設計と詳細設計での設計項目の明確化

理想的なハードウェアを仮定した機能的な制御アルゴリズムをプロトタイプ設計の工程で実施し，詳細設計の工程では，実機のハードウェアの制約を満たすようにソースコードを最適化することに重点をおくことができる．

設計内容をプロトタイプ設計と詳細設計とで区別することによって，各設計工程でのデバッグが可能となり，後工程に残るバグが減少する．

- 実機の“バラツキ”に影響を受けない制御論理の評価

実機試験なしで制御アルゴリズムの性能評価が可能となり，実機の性能のバラツキに左右されない評価が

できる．

- 制御システムハードウェアや被制御系の試作製造前の制御論理の評価

ハードウェア/ソフトウェア コデザインを可能にし，最適なハードウェア/ソフトウェアの機能分割を得ることができる．

- 実機実験の回数減による開発費用の低減
- 実機実験の安全性の向上

実機において制御される機械部分が制御不能に陥るような致命的なバグをプロトタイプの段階でとることができる．

6. むすび

我々が開発を進めている組込み型制御システムを対象とした設計支援システム：組込み工房を紹介し，そのサブシステムであるプロトタイプ設計支援システム Schetch について述べた．

支援システムの開発に先立ち，組込み型制御システム向けのプログラムモデルを議論し，言語機能として実時間時系列型というデータ型によって実時間処理を実現することを提案した．

Schetch の機能記述において，機能ブロック間のデータ交換を時系列型のデータを介して行うことによって，機能ブロック間の同期の問題を回避し，機能ブロックの機能的な独立性が確保できる．それによって，制御システムを見通しよく設計することが可能になることを実問題に適用することによって確認した．

今回，実問題として例示した ABS 制御をはじめとする車両制御では新規設計はまれで，ほとんどの設計は既存システムの変更という形式で行われる．従って，既存の設計仕様書からのリバースは必須の作業である．そこで，我々も実在する C ソースプログラムのための設計仕様書から，リバースして，Schetch の記述を作成した．そのため，新規に仕様を起こして記述する場合いと比較して，実時間時系列型データの考え方が適用できる部分が限定された可能があるが，プロタイピングの工程を効率化できた．

謝辞 日ごろから御議論頂く株式会社豊田中央研究所ソフトウェア研究室山崎知彦室長，佐野範佳研究員をはじめソフトウェア研究室の諸氏に感謝致します．また，名古屋大学大学院工学研究科情報工学専攻 山本晋一郎博士および濱口毅先生には数々の御助言を頂いたことを感謝致します．

文　献

[1] 宮本和俊，村松菊男，"カーエレクトロニクスとマイクロコンピュータ，" 情報処理，vol.36, no.9, pp.865–873, 1995.

[2] D.J. Hatley and I.A. Pirbhai, "Strategies for Real-Time System Specification," Dorset House, 1988.

[3] P.T. Ward and S.J. Meller, "Structured Development for Real-Time Systems," Prentice-Hall/Yourdon Press, 1985.

[4] J. Quemada, A. Ferna'ndez, and J.A. Mañas, "LOLA: Design and Verification of Protocols Using LOTOS," in IBERIAN Conference on Data Communications, Lisbon, May 1987. Also in Computer Communication Systems, ed. A. Cerveira, North-Holland, 1988.

[5] 中田明夫，東野輝夫，谷口健一，"時間制約の記述されてLOTOS 仕様からのプロトコール合成，" 情処学論，vol.37, no.5, pp.672–686, May 1996.

[6] F. Feldbrugge, "Petri Net Tool Overview 1992," LNCS vol.674, Springer Verlag, 1993.

[7] B. Grahlmann, "PEP: A Programming Environment based on Petri nets," Application and Theory of Petri Nets '95 –Tool Presentation (Torino 95).

[8] A. Benveniste and G. Berry, "Synchronous approach to reactive and real-time systems," Proc. IEEE, vol.79, no.9, pp.1270–1282, 1991.

[9] A.D. Stoyonko, "The evolution and state-of-the-art of real-time languages," The Journal of Systems and Software, pp.61–84, April 1992.

[10] S.S. Bhattacharyya, P.K. Murthy, and E.A. Lee, "Software Synthesis from Dataflow Graphs," Kluwer Academic Press, 1996.

[11] "TMS320 Family Development Support Reference Guide," Texas Instruments Inc. 1995.

[12] B.C. Kuo and D.C. Hanselman, "Matlab Tools for Control System Analysis and Design," Prentice Hall, 1994.

[13] "MATLAB Reference Guide," The Math Works, 1992.

[14] "SIMULINK User's Guide," The Math Works, 1992.

[15] G. Berry and G. Gonthier, "The esterel synchronous programming language: Design, semantics, implementation," Science of Computer Programming, vol.9, no.2, pp.87–152, 1992.

[16] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous dataflow programming language LUSTRE," Proc. IEEE, vol.79, no.9, pp.1305–1320, Sept. 1991.

[17] D. Harel, H. Lachover, A. Naamad, A. Pnveli, M. Politi, R. Sherman, A. Shtull-Trauring, and M. Trakhtenbrot, "StateMate: A Working Environment for the Development of Complex Reactive Systems," IEEE Trans. on Software Engineering, vol.16, no.4, April 1990.

[18] D. Harel, "Statecharts: A visual formalism for complex systems," Science of Computer Programming, vol.8, no.3, pp.231–274, June 1987.

[19] A. Benveniste and P. Le Guernic, "Hybrid dynamical systems theory and the SIGNAL language," IEEE Trans. on Automatic Control, vol.35, no.5, pp.525–546,

May 1990.
[20] R. Balzer, T.E. Cheatham, Jr, and C. Green, "Software technology in the 1990's: Using a new paradigm," Computer, vol.16, no.11, pp.39–45, 1983.
[21] 和田錦一，寺嶌立太，"制御プログラムの評価を可能にする高速な ECU シミュレータ，" 自動車技術会春季大会，pp.187–190, 1996.
[22] R. Piloty, M. Barbacci, D. Borrione, D. Dietmeyer, F. Hill, and P. Skelly, "CONLAN Report," LNCS151, Springer-Verlag, 1983.
[23] 松田俊郎，"ABS の最新実用知識<その 1>，" 自動車工学，pp.31–54, Sept. 1990.
[24] 松下晃久，近藤孝一，安田武史，渡辺秀夫，"ABS コントロールユニット，" 富士通テン技報，vol.10, no.2, 1992.
[25] 稲森　豊，手嶋茂晴，脇田敏裕，"組込み型リアルタイムシステムのための連続時間系/離散時間系統合シミュレーション環境の構築，" 日本ソフトウェア科学会 FOSE '95, pp.219–224, 1995.

付　　録

1. リモコン信号仕様

1.1 入　　力

入力は，リモコン信号（rimokon_signal）のみである．リモコン信号は，リモコンのどのボタンが押されたかをパルスで表したものである．信号は，初期信号，カスタム信号，データ信号から構成され，この順で送信される．

初期信号：初期信号は，これから送られてくる信号がリモコン信号であること表す．初期信号は，ON が 9〜11 ms 続く信号である（図 A·1）．

カスタム信号：カスタム信号は，リモコンを識別するための信号である．カスタムコードは，識別コードの 4 ビットを，0 ならばパルス間隔を 0.5〜1.5 ms，1 ならば 1.5〜2.5 ms のパルスに変換したものである．従って，カスタム信号のパルス数は 5 である（図 A·2）．

データ信号：データ信号は，リモコンのどのボタンが押されたかを識別するためのコードである．データ信号は，2 ビットのボタン識別コードと，その各ビットを反転させたエラー検出コードの，計 4 ビットをカスタム信号と同様の方法でパルスに変換したものである．但し，最初のビットは，カスタム信号の最後のパルスとデータ信号の最初のパルスとの間隔で表す．各ビットは，カスタム信号と同じ仕様のパルス実現される．データ信号のパルス数は 4 である（図 A·3）．

ボタン識別コードは，それぞれ次のボタンを表す．

- 00:「開ける」ボタン
- 01:「止める」ボタン

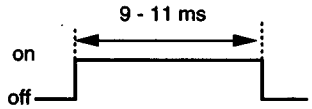

図 A·1　初期信号のタイミングチャート
Fig. A·1　Timing chart of initial signal.

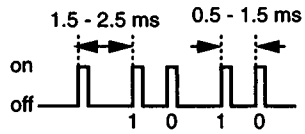

図 A·2　カスタム信号のタイミングチャート
Fig. A·2　Timing chart of custom signal.

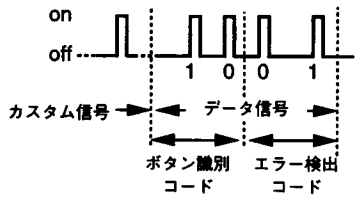

図 A·3　データ信号のタイミングチャート
Fig. A·3　Timing chart of data signal.

- 10:「閉める」ボタン

1.2 出　　力

出力は，ボタンデータ（button）のみである．

ボタンデータは，リモコンのどのボタンが押されたかをデータで表したものである．それぞれのボタンが押されたときに，次のデータを出力する．

- 「開ける」ボタンが押されたとき：0
- 「止める」ボタンが押されたとき：1
- 「閉める」ボタンが押されたとき：2

また，カスタム信号またはデータ信号に誤りがあった場合には，次のデータを出力する．

- エラーを検出したとき：3

（平成 8 年 9 月 2 日受付，9 年 1 月 20 日再受付）

手嶋　茂晴　（学生員）

1986 京都大大学院工学研究科情報工学専攻修士課程了．同年より（株）豊田中央研究所に勤務．1989〜1991 愛知教育大学非常勤講師．1993〜1994 スタンフォード大学客員研究員．1995 より名古屋大大学院工学研究科情報工学専攻博士課程在学中．

稲森　豊

1994 京都大大学院工学研究科情報工学専攻修士課程了．同年より（株）豊田中央研究所に勤務．リアルタイムシステムの仕様記述に関する研究に従事．

阿草　清滋　（正員）

1970 京大・工・電気第二学科卒．同大大学院に進学．1974 より京都大学工学部情報工学科に勤務．同助教授を経て，1989 より，名古屋大学工学部電気第二学科教授．現在は同大学院工学研究科情報工学専攻教授．1986〜1987 カリフォルニア大学客員研究員．工博．ソフトウェア工学に関する研究に従事．特に要求分析，ソフトウェア仕様 化技法，ソフトウェア開発モデル，再利用技術などに興味をもつ．ソフトウェ ア科学会評議員，京都高度技術研究所副所長．