解　説

3.4 制御ソフトウェアを対象としたCASEの現状と動向†

山　本　修　一††

1. はじめに

グラフィカルな表示やウィンドウ操作ができるなどの，使い勝手のよいワークステーションと，その上で動作するソフトウェア開発支援ツール（CASE ツールと銘打っている）が市場に登場してきたことから，CASE に対する関心が高まっている．現状は米国製のツールが中心であるが，日本語対応したものが出てくれば，ワークステーション価格の低下とあいまって，ソフトウェア開発者一人一人が自身のワークステーションを使って，ソフトウェア開発作業を行うのも夢ではなくなるだろう．しかし，それぞれのツールがどんな種類のソフトウェア開発を支援の主対象としているか，またどんな開発手法を支援しているかが，それらのツールが現場のソフトウェア開発で使えるかどうかを決めることになる．「CASE＝Computer Aided Software Engineering；計算機支援によるソフトウェア開発（エンジニアリング）」であることから，開発対象となるソフトウェアにより開発手順や設計のポイントは異なる．したがって，対象ソフトウェアに合った支援機能が必要であることは言うまでもない．

また，現在 CASE が注目されているのも，プログラミングの合理化，機械化だけでは，今以上の生産性向上が期待できず，要求定義やシステム（ソフトウェア）設計の上流作業支援により，より一層の生産性および保守性向上のブレークスルー技術としての期待が大きいことにある．CASE を「ソフトウェア開発の自動化」と捕え，この上流工程での仕様，設計情報からプログラムを自動生成しようとする点に大きな期待がかけられている．ここでは，制御システムのソフトウェア開発を支援する CASE について，とくに，どんな開発モデルを前提に，どんな手法を使った上流作業を推奨しているかを中心に解説する．

2. 制御システムの特徴

まず，制御システムとはどんな特徴をもったもので，何が制御システムの要求定義や設計のポイントであるかからみてみよう．

制御システムとは，センサなどから実時間（リアルタイム）で外部プラントの状態を計算機システムに入力し，その時点の各種のシステム状態（計算機内部の状態変数として保持）との関係から，必要な制御信号をプラント制御装置に出力するシステムである．

したがって，事務処理システムなどの一般のソフトウェアシステムとは違ったいくつかの特徴がある[1]．

① どんな外部発生事象にどう応答するか

一般の情報処理システムでは，どんな画面あるいは帳票が入出力されるかに関心がある．しかし，制御システムでは，制御装置から発生する信号を入力し，それに基づいてなんらかの制御を必要とする事象発生を認識し，それに対してシステムがどんな応答（制御装置への出力）をするかに関心がある．

② システムの入出力は非構造的かつ連続的に発生

一般の情報処理システムでは，入出力がいつ発生し，どんなデータ構造かはあらかじめ決められるが，制御システムの入力はセンサからであり，非構造的なデータを連続的に取り込まなければならない．また，出力も連続的で入力とオーバラップして行われる．

③ 入力と出力は必ずしも1対1に対応しない

すなわち，システムの運転モードや入力時点でのシステム状態によって，同じ入力でも出力が異なることがありうる．また，同時に複数の入力処理を必要とする．これら実時間での複数の外部入力から，システム状態を判定し出力を決定するシステムである．

† CASE Technology for Real-Time Software Development by Shuichi YAMAMOTO (TOSHIBA CORPORATION FUCHU WORKS, Engineering Administration and Information Systems Department, Software Engineering Development Group).

†† (株)東芝府中工場技術・情報システム部ソフトウェア生産技術開発担当

④ 外部システムとの関連が複雑多岐

一般の情報処理システムでは，人（画面，帳票で人出力）あるいは他計算機システムとのデータ交換がシステムの境界となるが，制御システムは外部プラントの複雑多岐な装置（電気的なものもあれば機械的なものもある）やオペレータ（システム運転範囲によって異なる操作を要求）との関連からシステム境界が決められ，そのインタフェースも複雑多岐にわたる．

⑤ 人力に対する応答精度は非常に厳しい

人を相手にするシステム（ある程度計算機システムの動作スピードに合わせてくれる）と，同時に動作している機器を制御するシステムとでは，おのずと応答時間や応答精度の基準が異なる．制御システムでは，人出力するシステムのインタフェース機器の実行スピード（計算機システムとは独立に動作）との兼合いで，計算機システムを含む全体システムの実行性能が決まるため，計算機システムへの応答要求は非常に厳しいものがある．

以上いくつかの視点から，一般の情報処理システムと制御システムに対する要求条件の違いをみたが，このことから制御システムでは本質的な並行処理が要求される．すなわち，人出力データの関係からだけでソフトウェアの実行構造を決定できず，人力データの発生頻度や外部イベントに対する応答要求などから，いつ，あるいはいつまでに出力すべきかの実行タイミングを考慮したソフトウェアの実行構造の設計が必要となる．また，このようなシステムの要求機能面からだけでなく，使用計算機の並列処理の実行性能を配慮したソフトウェア設計が必要とされる．

3. 制御システムの開発手法

米国では DoD（国防省）を中心に，システム開発の標準化が進められており，DoD-2167 A[13] の開発規程に基づいた制御システムの開発が必須条件となっている．現在市販されている米国製 CASE ツールは，ここに述べる開発モデルと設計表記，および DoD-2167 A が定めるドキュメント作成を支援している．ここでは制御システムの特徴に合わせた開発の手順および手法を解説するが，とくにシステムの要求定義および設計の，システム開発の上流工程について記述する．

3.1 開発の手順

現在 CASE がベースとしている開発モデルはウォータフォール型で，システムの要求定義から始めて，ソフトウェア構造を設計し，プログラムの製作・試験とそれらを組み合わせてシステムとしての試験を行う．これまでのソフトウェア工学の研究から，このようなシステム開発をそれぞれの工程で中心となる視点からのモデル化と，モデル間の変換として捕える見方がある[1),2),4)]．まず，客先要求の視点からシステムに何が要求されているかの「要求モデル」を作成する．そして，そのモデルを計算機構成およびソフトウェア要素にどうマッピングするかを検討し「実現モデル」を作成，それに基づいてプログラム開発を行うというものである．すなわち，要求定義とは，どんな「要求モデル」を作成するかであり，ソフトウェア設計とは，この要求モデル要素との対応関係と実現可能性の検討から，どんな「実現モデル」を作成するかであると言える．制御システム開発でも，まず「要求モデル」を作成し，次いで「実現モデル」を作成となるが，先に述べたように一般の情報処理システムとは異なった特徴をもつことから，複数の視点からのモデル化とモデル間の対応づけを考慮した開発手順が必要となる．先に「要求モデル」の一言で片付けたものも，「システム境界の定義」を外部装置やオペレータとシステム間のデータ入出力フローやイベントフローでモデル化する．また「システムは何をするか」を，外部発生のイベント（刺激）に対するシステム応答の組でモデル化する．さらに「システム内部でどんな動作をするか」を，内部動作間のデータ入出力フローや実行制御フローでモデル化する．また「実現モデル」も並列性実現への配慮から，「プロセッサレベル」，「タスクレベル」，「モジュールレベル」でのモデル化が必要となる．

「要求モデル」では制御システムがもつ本質的な並列性の表現がポイントであるが，「実現モデル」ではこの並列性をどんな計算機構成とソフトウェア構造で実行するかの，段階的な並列性実行の設計検討が要求される（**図-1**）．

このようなモデル作成による制御システム開発手順をベースに，現状の CASE が採用しているモデル化の手法は，一般の情報処理システムでも広く使用されている構造化分析/構造化設計手法 (SA/SD : Structured Analysis/Structured Design) に，制御システムに必要な記述を追加したものである．以下，手順に従ってどんな記述でモデル化されるかを概観する．

（注） 構造化分析の記述で中心的なデータフロー図の表記法にも，Yourdon-DeMarco 法と Gane & Sarson 法があるように，制御システム向けに拡張し

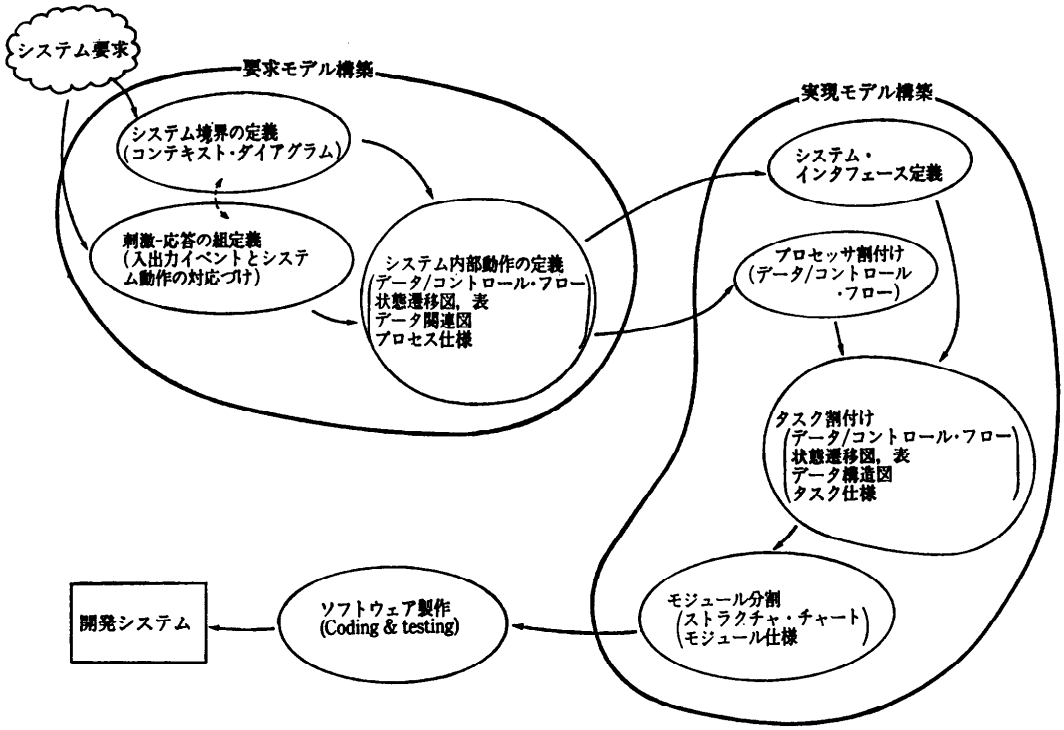


図-1 制御ソフトウェアの開発手順とモデル化

た記述方式でも，ボーイング社 Hatley 氏のものと，Ward & Mellor 氏のものとがある[14]．若干記述スタイルは異なるが，何を記述しているかについては大差ない．以下では Ward & Mellor の記述スタイルで示す．

3.2 要求モデル構築

3.2.1 システム境界定義

システム開発のまず始めは，システムの範囲がどこまでかを決めることである．「システムで何ができるか」の機能的なモデル化に先だって，まずシステムと外部との境界を明確にし，外部とどんな入出力が行われるかを決めることが必要となる．構造化分析の手法では，これを図-2 に示す「コンテキスト・ダイアグラム」で表現する．ここでは，システム自身を一つの大きな変換(円)とみなし，システムに関連する外部装置やオペレータからどんなデータ（実線アロー）や制御信号（破線アロー）を入力し，それに対してシ

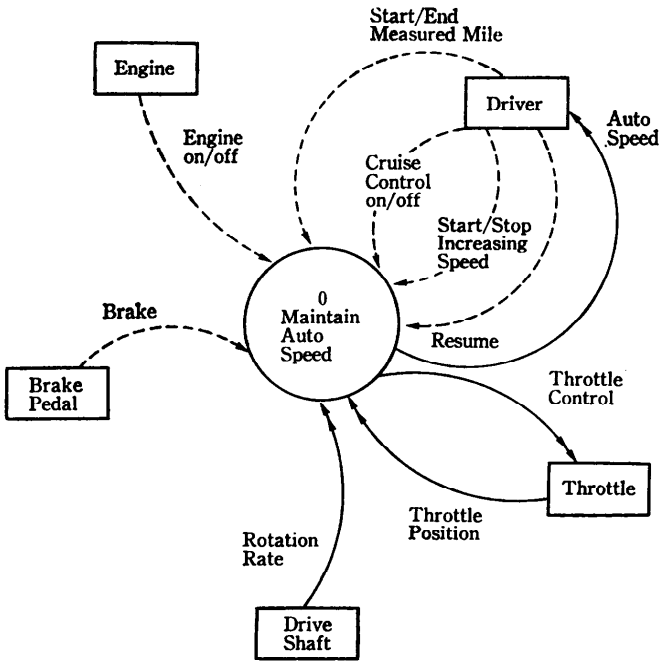


図-2 「自動車制御システム」のコンテキスト・ダイアグラム
(文献 1) より引用)

ステムはどんなデータや制御信号を出力するかの記述でシステム境界（システム処理範囲）を示す．

3.2.2 刺激-応答の組定義

制御システムの機能定義は，外部発生事象に対してどんな応答を返すかであり，図-3 に示すように，システムでの制御条件となる外部発生事象に対するシステム応答の組を定義する．すなわち，システム動作の事前条件を外部装置あるいはオペレータからのデータ入力や，制御信号（たとえば，スイッチのオン/オフ）あるいは入力値が制御条件に達した（たとえば，温度が基準値を越えた）などの制御イベントとシステム状態で定義し，システム応答を動作結果の出力データや制御イベント，システム状態で示す．また，これらの組の発生順序を図を使って定義する[4]．

3.2.3 システム内部動作定義

システム内部の動作は，コンテキスト・ダイアグラムで一つの変換として記述されたシステムを，図-4（a）に示すようないくつかのデータ変換（実線の円）

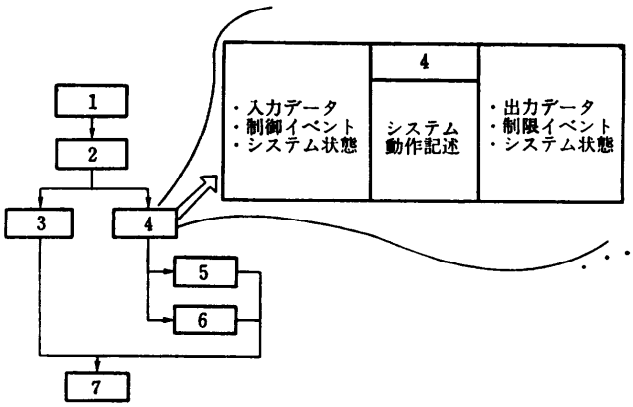
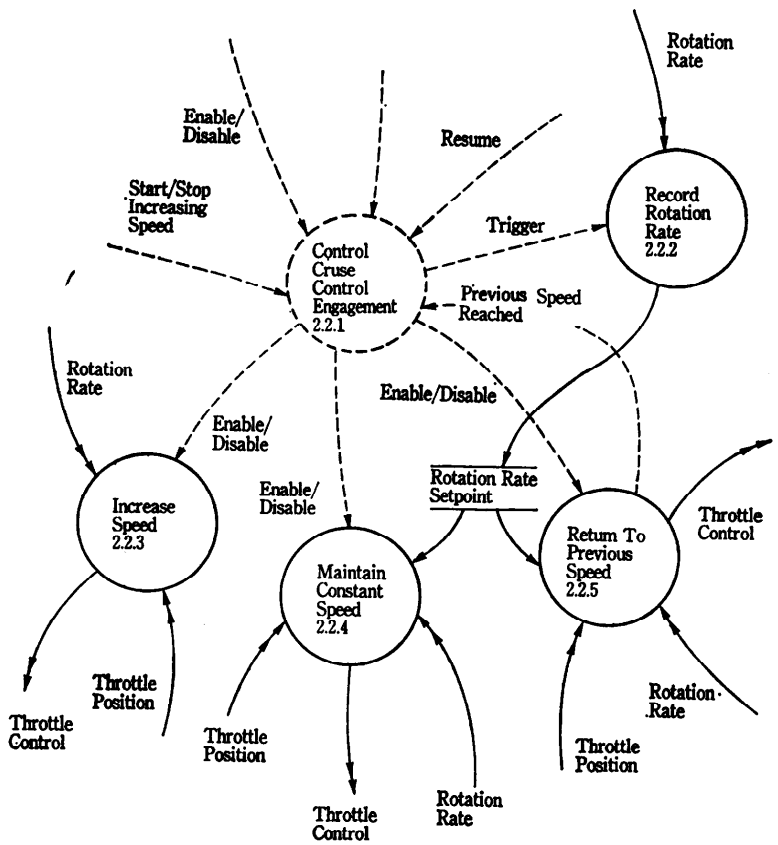
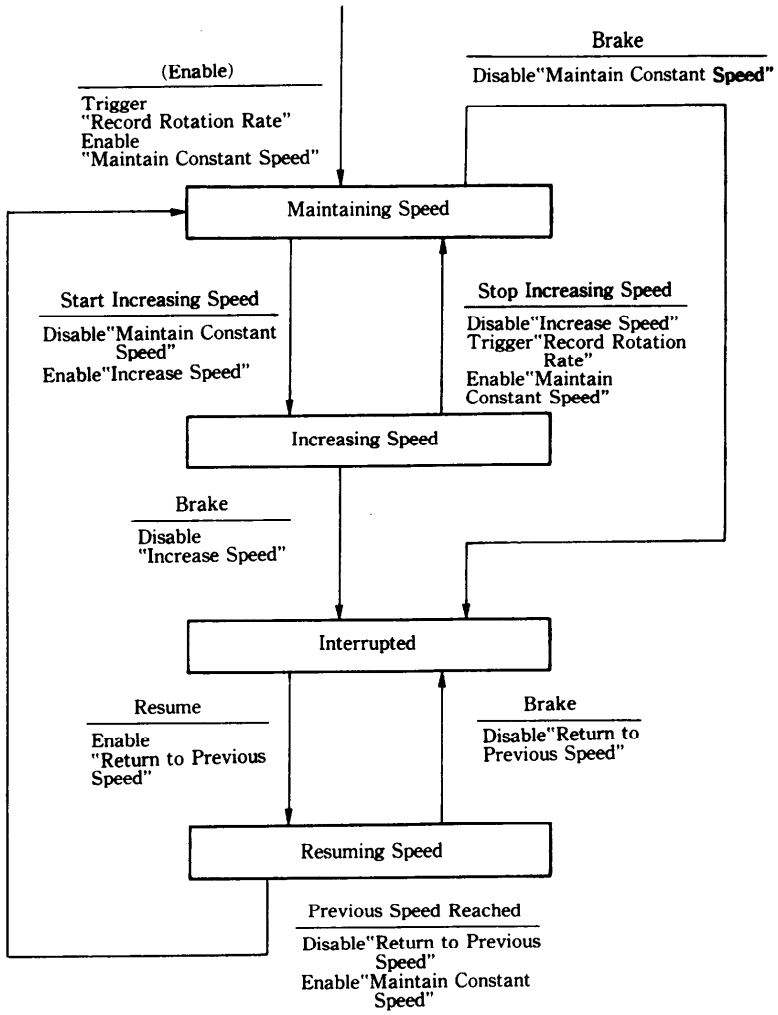


図-3 刺激-応答の組定義（文献 4）より引用）



（a）「自動車制御システム」でのデータ/コントロール・フロー図の一部（文献 1）より引用）



(b) "Control Cruise Control Engagement" の状態遷移図 (文献1) より引用)

図-4

とそれらの間を流れるデータフロー(実線アロー)と，それらのデータ変換をいつ実行するかをコントロール変換(破線の円)とコントロールフロー(破線アロー)で記述する．すなわち，一般的なデータフロー図にコントロール変換とコントロール・フローを付加して，制御システムに固有な，どんな条件のときにどのデータ変換を実行するかの実行タイミングを記述する．したがって，コントロール・フローは外部発生イベントあるいはシステムがその発生を識別してどのデータ変換を実行するかの実行制御イベントを表し，コントロール変換の詳細定義として図-4(b)に示す状態遷移図や表で，どのシステム状態のときにどのコントロール入力があったら，どのシステム状態に遷移し，どのコントロール出力をするかの実行制御の詳細を記述する．また個々のデータ変換は同様の記述でさらに詳細化される．この詳細展開は，「システムで何をするか，どう動作するか」がシステム開発者，客先の間で共通に理解できるレベルまで行う．最終的には一つ一つのデータ変換は，「プロセス仕様」と呼ばれる擬似コードを使った処理記述にまで詳細化される．(データ変換の内容が簡単に理解できるものについては，ここまで記述しない場合もある．)

このような詳細化により，システムは図-5(a)に示すような「コンテキスト・ダイアグラム」を最上位と

する仕様記述の階層構造で，システム要求モデルが作成される．

3.3 実現モデル構築

3.3.1 システム・インタフェース定義

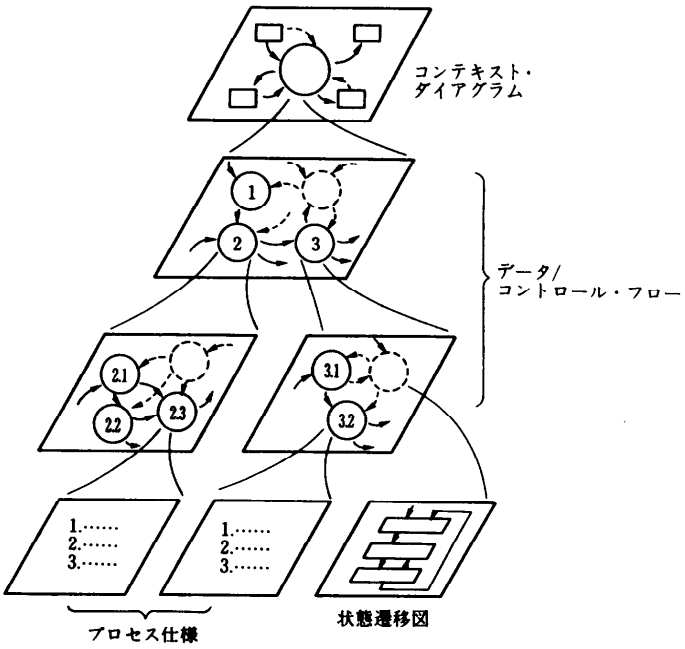
これで，システムの機能的な面とシステム動作の仕様がモデル化された．次はこのモデルをもとに，具体的なシステムを設計していくことになるが，まずシステムのインタフェース定義から始める．ちょうど一般の情報処理システムでのファイル，レコード設計や画面，帳票設計に相当する．制御システムでは先にモデル化されたシステム外部とのデータ，イベントフローの仕様を決めることになる．このインタフェースデータは，ソフトウェア技術者だけでなく，ユーザやハードウェア技術者とも共有するものであり，計算機用語ではなくエンジニアリング用語を使った定義が必要となる．たとえば，データが取る値についても，エンジニアリングの言葉での単位（kg, km/sec, ℃, …）や範囲，精度，分解能といった定義が必要だし，アナログかディジタルかパルスかの入出力の形態とビット表現，タイミングや，外部装置とどうインタフェース（RS-232C，セントロ仕様, …）し，どんな入出力命令で行うかなどの定義が必要となる．

また，オペレータとのインタフェースとして，グラフィカルなシンボルを使った，外部環境を表示する操作画面のフォーマット定義と，そこにどんな情報を入出力するかや，ボタンやポインティングデバイスによる画面操作（複数操作画面の遷移など）の定義を行う．

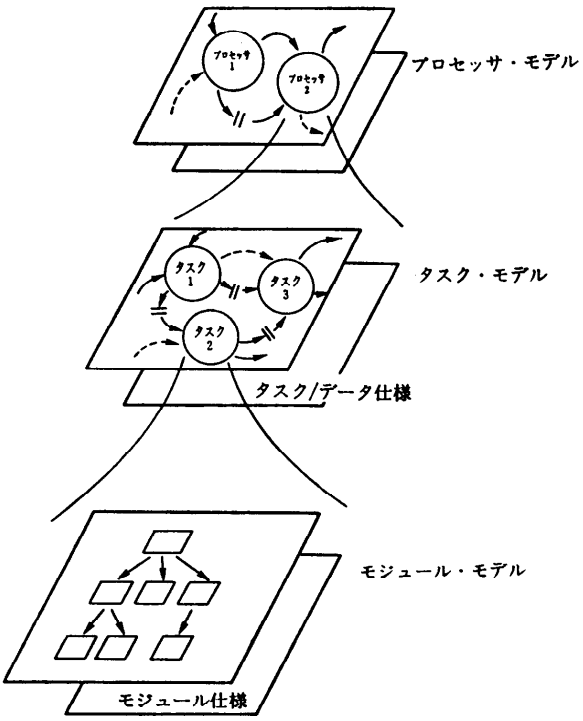
3.3.2 プロセッサ，タスク割付け

次にいよいよシステム内部処理をどんな計算機構成，ソフトウェア構成で実行するかの設計を行う．はじめに述べたように，制御システムがもつ本質的な並列性を，どんな計算機システム構成で実現するかが設計のポイントであり，まずプロセッサへのシステム動作処理の割付けであり，次に1プロセッサ内でのタスク割付けである．

これは先の「システム内部動作」記述を



(a) 要求モデルの階層構造



(b) 実現モデルの階層構造

図-5

並列性実現の観点から再整理するもので，一つのデータ変換あるいはコントロール変換として表現されていたものを複数のタスクに分割したり，逆に複数の変換とされていたものを一つのタスクにまとめたりの作業を行う．この結果，新たにプロセッサ間あるいはタスク間のインタフェースができることになり，これらをどんな手段で実現するかの設計が必要となる．ここでは，基本的な表現は「システム内部動作」の記述方法と同じだが，より具体的なインタフェース（プロセッサ間，タスク間）記述を可能とするため，キューやメイルボックス，セマフォといった記述を採用する方法もある[5),6)]．

なぜこのようなタスク構成となったかは，「要求モデル」でのシステム動作の機能的な側面（並列性）と同時に，要求される実行性能を満足させるためのものであり，タスク割付けと同時にプロセッサ内でのタスク実行の優先度の設定が必要となる．したがって，いったん設計したタスク構成が，所要実行性能を満足するかどうかが設計評価のポイントである．この性能評価についてはいろいろな研究がなされており，最近の CASE ツールにはこの実行性能を予測した（もちろん，そのツールが対象としている OS を前提としているが）レポートを出してくれるものもある．

3.3.3 モジュール分割

設計の最終段階では，このような手順で設計されたタスク個々の内部モジュール構造を設計する．これは一般的な構造化設計の手法を用いて，モジュール階層とモジュール内部の処理手順を擬似コードなどで記述する．しかし米国では，制御システムのソフトウェア記述を Ada で行うことが多くなっており，タスクあるいはモジュール設計の表記として，Ada のタスク，パッケージの概念を図式化した表記を採用する場合もある[7),8)]．したがって，制御システムを対象とした CASE ツールには，構造化設計のストラクチャ・チャートと Ada の設計表記の両方を支援するものがある．

実現モデルも，これらの記述結果は図-5(b)に示すような階層構造となる．

3.4 ドキュメンテーション

制御システムの開発では一般の情報処理システム以上のドキュメント作成が要求され，米国ではどの工程でどんなドキュメントを作成するかを DoD-2167 A で詳細に規定している．また，各ドキュメントを分かりやすく構造的に記述することはもちろんだが，客先からの要求文書中のどこが「要求モデル」のどの要素と対応しているか，「要求モデル」要素は「実現モデル」のどの設計要素と対応しているかの，トレーサビリティ（仕様の追跡可能性）の記述も要求されている．この要求は，システムに対する要求事項が具体的にはどの設計要素と対応しているか，なぜその要素と対応したかの設計判断や，要求事項に対する実現もれを保証するためである．現状の CASE ツールではまだまだ十分とは言えないが，このようなトレーサビリティを支援する機能が盛り込まれ，また DTP (Desk Top Publishing) ツールとのデータ交換機能により，ドキュメントの電子出版を支援している．

4. 代表的な CASE ツール

ここでは，制御システム向けをうたっている代表的な米国製 CASE ツールを紹介し，各ツールの特徴を解説する．ここで取り上げた製品は，すべて UNIX ベースのエンジニアリングワークステーションで動くものである．

4.1 Teamwork, StP

Teamwork と StP (Software through Pictures) は，これまで述べた仕様，設計記述（データ/コントロール・フロー図，状態遷移図および表，ストラクチャ・チャート，プロセス/モジュール仕様記述）をすべて支援する．また，データ関連図 (Entity-Relationship Diagram) も支援しており，一般の情報処理システムの開発でも使用できる標準的な CASE ツールである．制御システム固有の機能では Ada 設計表記の支援がある．図-6 に Teamwork の画面表示例を示す．

4.2 CARDtools

このツールは，制御システムのみを対象とした CASE ツールで，その名も Computer Aided Real-Time Design tools を略したものである．このツールの特徴は，システムインタフェースの定義とタスク設計でタスク間のインタフェースを，いくつかの設計要素（キュー，メッセージ，セマフォなど）ごとに別のシンボルを割り当て，明確に表現できるようになっている．また，ツール開発元が別パッケージとして提供するリアルタイム OS・VRTX のタイミングデータを使った，実行性能評価が可能である（図-7）．

4.3 AutoCode

これは前三つとは異なり，マイコンベースのダイナミック制御システムを支援対象に絞ったツールで，特徴はデータフロー表現で，ダイナミック制御記述に必要な関数ブロックを標準部品（ユーザ固有の関数ブ

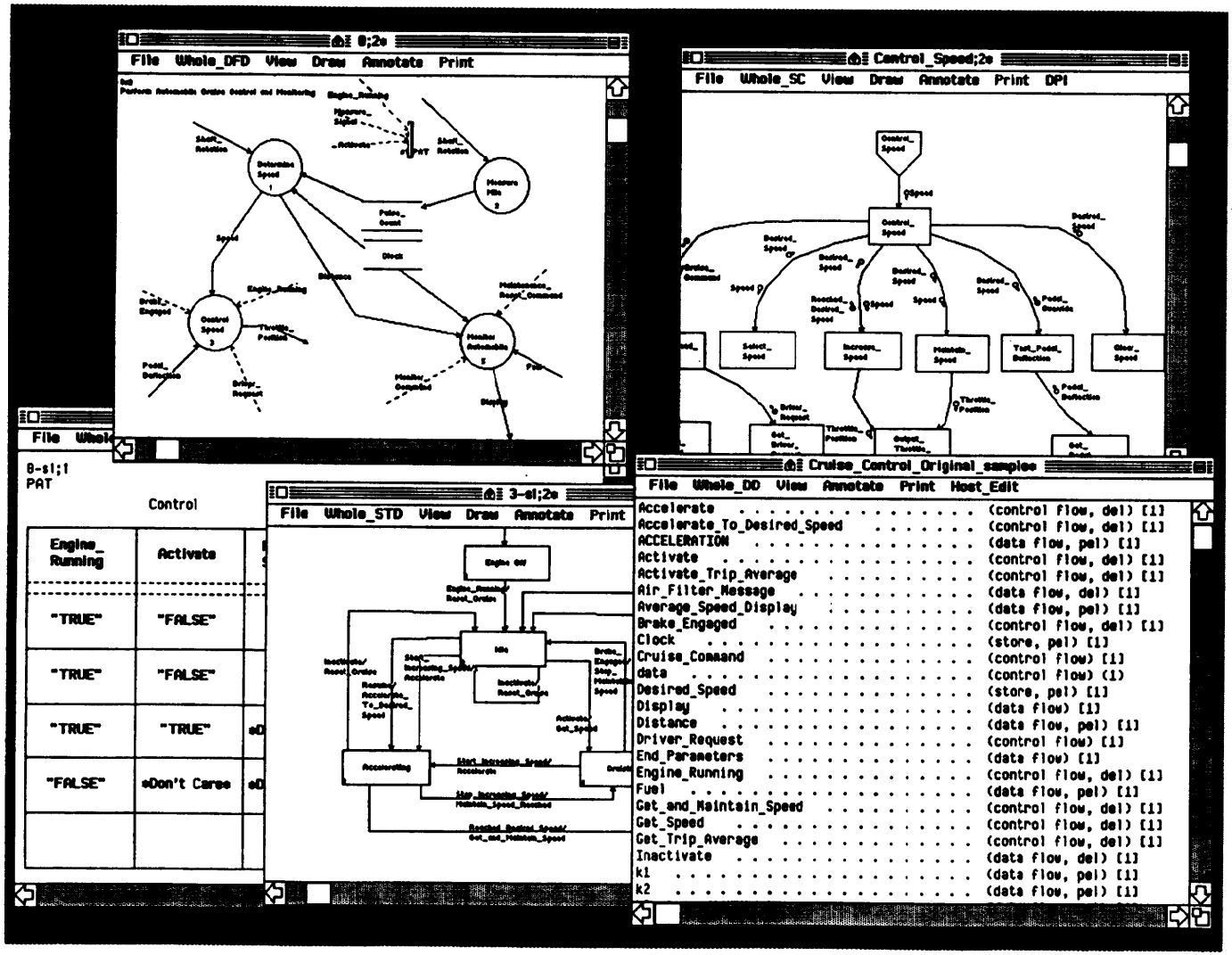


図-6 Teamwork：データフロー，状態遷移，ストラクチャ・チャート，データ辞書表示例

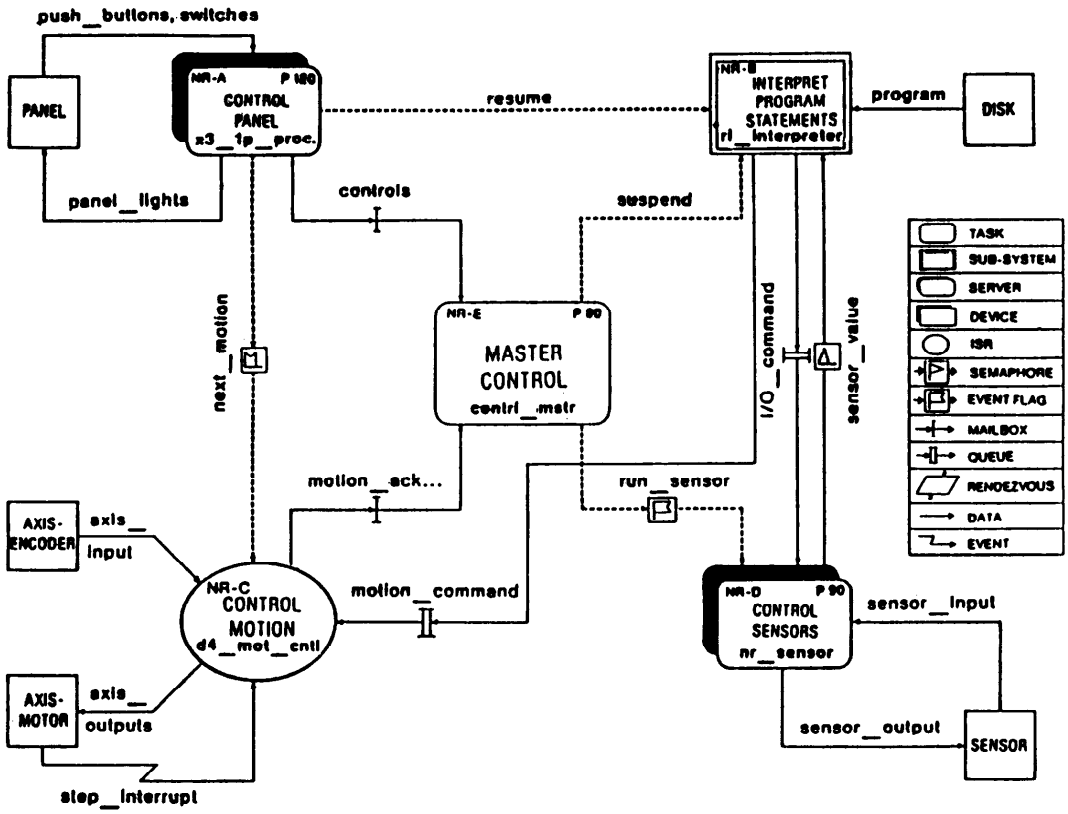
Numeric Value Specification Form

1) Section 2.2.3
2) Number 1.2
3) Input Data Item: Sensor_input
   4) Acronym : SNSRIN
   5) Hardware : Angular Sensor
   6) Description : Optical robot arm sensor
7) Characteristics of Value
   A. Units : degrees
   B. Range : −180 to +180
   C. Accuracy : 0.2 degrees
   D. Resolution : 0.1 degrees
   E. Maximum Derivative : 18 degrees per second
8) Data Representation: 10 Bits
   Scale : 1.0
   Offset : 0.0

MS															LS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0										

9) Timing Requirements: Sensor A/D converter register must be checked every 15 msecs
10) Instruction Sequence : Query sensor by raising bit 1 at address 17774. Fetch sensor values for address 17621 and set vector 320 to 0
11) Comments :

(a) CARDtools: システム・インタフェース定義の例



(b) CARDtools: タスク・マップ・ビルダ (TMB) 表示例

TIMING REPORT

COMPONENTS SET: Critical Path 1

DESCRIPTION: Time from switching on to sensor output

PATH CONSTRAINT 3,000 microseconds **FREQUENCY** 15/second

CPU No.	TYPE	CLOCK (Mhz)	OS	VERSION	RAM	ROM
1	68020	8.0	VRTX32	1	2	0

TASK NAME	TYPE	PROCESSOR	PRIORITY	TIME (Microseconds) CONSTRAINT	ESTIMATED	CALCULATED
CONTROL PANEL	SERVER	1	120	500	480	510
CONTROLS	MBOX	1		47	47	47
MASTER CONTROL	TASK	1	80	2000	1500	1500
RUN_SENSORS	EVENT FLAG	1		65	65	65
CONTROL SENSORS	SENSOR	1	90	1,000	1,000	1,000
TOTAL:				3,612	3,092	3,122
CPU UTILIZATION:				5.42%	4.64%	4.68%

(c) CARDtools: リアルタイム・パフォーマンス・ベリフィケーション (RTPV) 計算例

図-7

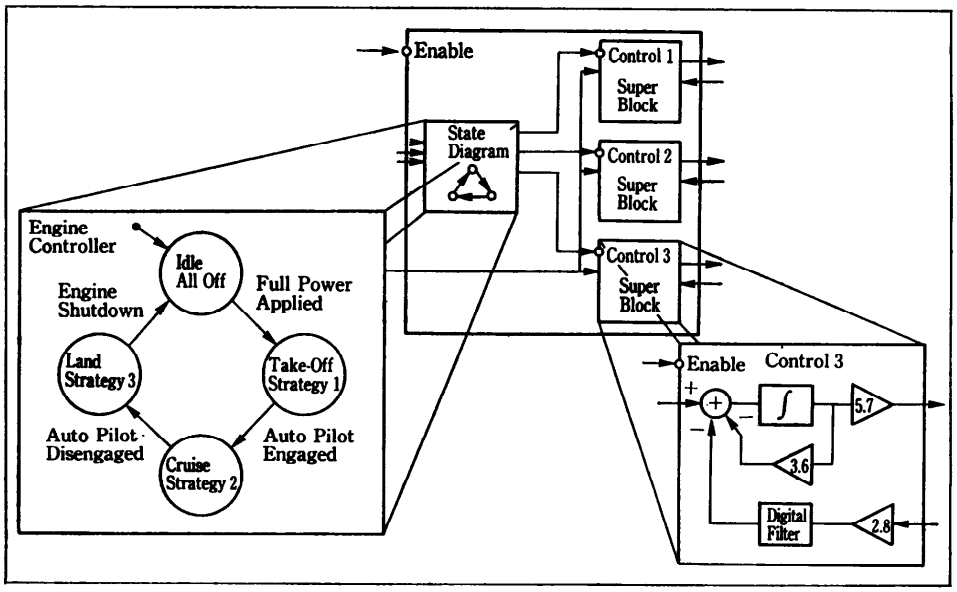


図-8 AutoCode: 状態遷移図と関数ブロック図を階層的に記述

ロック定義機能もある) とし, このデータ入出力の組が定義されたブロックをつなぎあわせて記述する. (電気 CAD の手法)

これと, 制御条件を記述する状態ダイアグラムとを組み合わせて, タスクの実行制御部分も含んだ完全なソースプログラム (Ada, C, Fortran) を生成する (図-8).

5. 今後の方向

以上で制御システムを対象とした CASE の現状解説を終える. 次に, この CASE が今後どんな方向に進んでいくかを考えてみる. キーワードはやはり「オブジェクト指向」であり,「エキスパート・システム」であろう. 今後の動向をうかがわせるものとして, いくつかの研究内容が紹介されている[9)~11)]. 設計手順のガイドや定義もれのチェック, 仕様レベルでの実行シ

ミュレーションなどがあげられる．また，すでにリアルタイム・エキスパート・システム構築ツールも市場に現れており，これらを使ったシステム設計でのプロトタイピングが考えられるようになってきた．一般の情報処理システムでプロトタイピングと言うと，画面や帳票のフォーマット定義や画面遷移を思うが，制御システムでは，外部環境のシステム構成要素を組み合わせ，それへの制御ルール記述によるシステム動作のプロトタイピングが要求される．これにより，システムインタフェースの定義や制御方式などの定義が，よりユーザの意向を反映したものになりうる．したがって，システム内部動作の仕様定義の前段階で，このようなAI技術と組み合わせた開発手順が今後の CASE として考えられる．

6. おわりに

制御システム開発を対象とした CASE の機能的な面を中心に解説した．CASE がここまで話題になってきたのも，ワークステーションの進歩とその上で動く CASE ツールが市場に現れてきたことによる．しかし，CASE が支援する開発手法は，70年代から始められたシステム仕様や設計をどんな視点でどう表現するかの研究成果である構造化分析/設計手法である．これらの手法の普及がまず先決であり，そのための教育に CASE ツールが使用されていくといったことが現在の状況であると考える．今後，この技術が発展するかどうかは，いつに手法普及にかかっており，従来の制御システム開発の方法を根本的に見直していく姿勢が必要となる．分野を限定すれば，すでに完全なプログラムを生成できるツールも現れており，手法普及と同時に，これらの CASE ツールを使って，仕様，設計情報とプログラムとの関連を整備した，全体的な再利用データベースの構築を進めていきたいものである[12]．

参考文献

1) Ward, P. T. and Mellor, S. J.: Structured Development for Real-Time Systems, Vol. 1, 2, 3, Yourdon Press (1985).
2) Hatley, D. J. and Pirbhai I. A.: Strategies for Real-Time System Specification (1987, 88): 立田種宏(訳): リアルタイム・システムの構造化分析, 日経 BP 社 (1989).
3) Ward, P. T.: The Transformation Schema: An Extension of the Data Flow Diagram to Represent Control and Timing, IEEE Trans, on SE, Vol. 12, No. 2, pp. 198-210 (Feb. 1986).
4) Deutsch, M. S.: Focusing Real-Time Systems Analysis on User Operations, IEEE Software, pp. 39-50 (Sep. 1988).
5) Gomma, H.: SOFTWARE DEVELOPMENT OF REAL-TIME SYSTEMS, ACM, Vol. 7, No. 7, pp. 657-668 (July 1986).
6) Gomma, H.: A SOFTWARE DESIGN METHOD FOR REAL-TIME SYSTEMS, ACM, Vol. 27, No. 9, pp. 938-949 (Sep. 1984).
7) Booch, G.: Object-Oriented Development, IEEE Trans. on SE, Vol. 12, No. 2, pp. 211-221 (Feb. 1986).
8) Ward, P. T.: How to Integrate Object Orientation with Structured Analysis and Design, IEEE Software, pp. 74-82 (Mar. 1989).
9) Luqi, V. B.: Rapidly Prototyping Real-Time Systems, IEEE Software, pp. 25-36 (Sep. 1988).
10) Blackburn, M. R.: Using Expert Systems to Construct Formal Specifications, IEEE EXPERT, pp. 62-74 (Spr. 1989).
11) Estrin, G. et al.: SARA (System ARchitects Apprentice): Modeling, Analysis, and Simulation Support for Design of Concurrent Systems, IEEE Trans. on SE, Vol. 12, No. 2, pp. 293-311 (Feb. 1986).
12) Biggerstaff, T. J.: Design Recovery for Maintenance and Reuse, IEEE COMPUTER, pp. 36-49 (July 1989).
13) DOD-STD-2167 A, 29 Feb. 1988, AMSCNO. N 4327.
14) McCLure, C.: CASE is Software Automation, Prentice Hall (1989).

(注) 紹介ツールの詳細問合わせ先
Teamwork: 東陽テクニカ, StP: SRA
Cardtools: セイコー電子工業
AutoCode: 住商エレクトロニクス

(平成2年2月22日受付)