解　説

実時間システムにおける要求と仕様化について†

松　本　吉　弘††

1. まえがき

要求 (requirement) の仕様化 (specification) に関する手法の開発は20年以上も前から応用別に進んでいるが，一般化については未熟といえよう．このことは人の要求というものがいかに奔放であり，形式化しにくいものであるかということを示している．

人が他の人にものを頼むときに要求が発生する．前者を要求者，後者を実現者とよぶことにする．実現者は受けた要求を主観的に解釈し，その解釈に従って実現行為を行う．この解釈が要求者の願望と合致しているかどうかを実現以前に要求者が確認する必要がある．また，実現後にその実現結果が要求と合致しているかどうかをも確認する必要がある．この2つの確認行為が両者の間で合理的に行われるためには，要求が要求者によって客観的に形式化される必要がある．この形式化行為を仕様化とよぶ．

要求がハードウェアを対象としている場合には，要求の対象物が顕在しているために，仕様化は比較的容易であった．ソフトウェアを実現対象とする仕様化行為においては，その対象が抽象であるために可視性に乏しく，要求の定義が難しい．各企業内ではそれぞれの特殊化された分野で，その応用にとくに向いた要求の形式化がどんどん進んでいた[5]．これが表に出なかった主な理由は，それが発表に値するほど一般性をもたなかったということだろう．1970年代に入ってから部分的に一般性をもった要求の形式化やそれに対して計算機を援用したシステムが数多く発表されるようになった．必然的に発表されるものの類型化が必要になってきた．最近における比較的妥当な類型化は D. Teichroew[6], C. V. Ramamoorthy[4] らによってなされている．

本稿では筆者らが，開発している実時間システム用応用プログラムに対する要求処理システムの基本思想について述べている．その位置づけについては2で述べる．

筆者らの対象としている応用プログラムは第1要求の発生からプログラムを出荷するまで2～3年かかるという大規模のものであり，計算機が外界に深く入り込んでいるので，要求の複雑さが非常に大きく，計算機を援用した処理システムを必要としている．外界は原子力，火力などの大容量発電用システム，鉄鋼圧延プロセス，電力や水などの分配供給システムなどである．

2. 分類と位置づけ

ソフトウェアライフサイクルがいくつかの段(stage)から成ることは公知のことである．ある段で行われる活動は前段からの要求に従って行われ，この活動の結果，後段の実現者への要求が新しく生まれる．したがって要求は最終段を除いてすべての段で生れている．

図-1 はライフサイクルと仕様化の関係を示している．図の最上層に示されたものは計算機を使用する組織，団体，プラントなど (organization or data-processing system とよばれる) に関するものである．

これを設計し，製作するための活動を D_0 としている．この結果，次の層への要求が発生する．これを仕様化したものを format 0 としている．A_0 は前段で

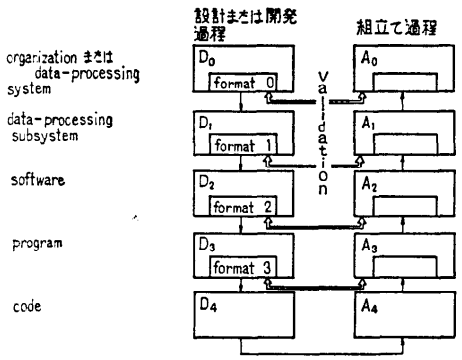

図-1　要求と仕様化との関係説明図

† Requirements and Specifications in Real-time System by Yoshi-hiro MATSUMOTO (Toshiba Corporation).

†† 東京芝浦電気株式会社

製作された製品が組み込まれて完成した「もの」である.

第2の層は最上層の一部に埋め込まれている計算機とその入出力機器より成るシステム (data-processing subsystem or DPSS とよばれる) に関係している. 更にこの下の各層はそれぞれソフトウェア, プログラム, コードである. format i (i=0,1,2,3) は D_{i+1} に対する要求仕様書である. これは D_{i+1} が正しく行われているかどうかの確認に用いられると同時に, A_i が正しく作られたかどうかの確認 (validation) にも用いられる. 各 format はこのような用途に対して十分耐えるものでなければならない.

活動 D_i, 製品 A_i および format i を扱う要求処理システムを level-i の要求処理システムとよぶことにする. 既存の要求処理システムは文献〔4〕にも示されているように各層それぞれに対して存在する. 各システムは大体においてあるひとつの層を対象として開発されている. 〔4〕によればミシガン大学の PSL/PSA[7] は level-2 に対して格付け, SADT は level-1 に対応するものとされている. この類形化に従えば本稿で筆者が対象としているレベルは level-2 である. 単独のレベルに着目した要求処理システムと共に, 一方では複数のレベルを縦断する要求処理システムも考えられている. 単独レベルの処理システムを仮に水平形システムとよぶならば, これは垂直形システムである. すでに実用されている data-dictionary とよばれる機能は垂直形システムに適した機能である.

各過程 D_i では D_{i-1} で生まれた要求に基づく必要を分析し, モデル化する作業が行われ, format i が生れる. この分析/モデル化はできるだけ多くの観点 (viewpoint) から行われねばならない. 実時間システムにおける level-2 ではたとえばつぎのような観点をすべて, または一部, 用いることが好ましい.

- ○機能を中心とする観点 (アクティビティモデルとデータモデルなどが作られる)
- ○コントロールを中心とする観点 (イベントモデル, オートマトンモデルなどが作られる)
- ○プロセス対象 (制御される水, 油, 電力, ロケットのようなもの) を中心とする観点
- ○オペレータを中心とする観点

今までこのレベルで実時間システムを主な対象に選んで正面からとり組んでいるシステムに SREM[1] がある. それ以外に評価に耐える発表はみられていない. SREM がとっている観点は機能を中心とするそれである. これに対して本稿に述べるシステムはプロセス対象を中心とした観点から必要をモデル化するシステムである. 要求の仕様化にあたってはできるだけ多くの観点で必要をモデル化することが推奨されており, 残りの観点についても今後カバーできるようにしたい.

3. 機　能

応用プログラム一般を対象とした要求処理システムが具えるべき主な機能についてまず述べる. ここで要求処理システムというときには計算機によって支援されることを前提としている. 主な機能は4つあると考える.

(a) 使用者が要求を形式化するのに必要な規則

(b) 上の規則に従って形式化された要求をインプットする際に必要な記述用言語

(c) インプットされた内容に誤りや一貫性の欠除がないかどうか検査する機能; 実時間システムの要求の検査にはシミュレーション機能が必要.

(d) 種々の目的に従ってインプットされている要求を編集し, 文書にしてアウトプットする機能; このアウトプットは要求者と実現者との間の公式の文書としても役立つので日本語であることが好ましい.

つぎに実時間上で動くシステムに対する要求の仕様化にあたって必要な機能を述べる.

(e) organization または data-processing system 内に存在し, DPSS と関係をもつ「もの」(object または entity) が実時間で状態空間内を動く. したがって「もの」とその状態, および実時間との対応を定義せねばならない.

(f) DPSS 内の大部分の FUNCTION が organization または data-processing system 内で生起するイベント (event) によって活動を開始したり, 停止したりするので, イベントと FUNCTION およびその制御との対応を定義せねばならない.

(g) DPSS 内の FUNCTION はある一定の順序に従って連鎖的 (ひとつの FUNCTION が終結してから次の FUNCTION が活動する) に結合されるか, または並行して活動する. 連鎖的に結合される場合には, その結合に際して判定を必要とする場合がある. 並行に活動している FUNCTION 間では同期化が必要である.

これら FUNCTION と FUNCTION 間の結合関係を定義できるものでなければならない.

(h) 実時間システムでは DPSS の実時間上での性能を厳しく要求されることが多い. したがってある機能を「現在時刻から何秒以内に完結せよ」とか,「時刻○時○分○秒」に完結せよ」とかいうようなことを定義できなければならない. このような要求が対象としている DPSS で実現可能であるかどうかを予測し, 無理な要求を警告する機能も広義の要求処理システムでは必要である.

すでに発表された実時間システム用要求処理システムは事務処理用のそれに比して数少ない. もっとも大規模と思われるものは SREM[1] である. このシステムは (a) から (h) までのすべての機能をほぼ具えた要求処理システムと思われるが, 詳細は公開されていない. 機能 (b) としては RSL と称する言語を提供する. 機能 (g) については R-Net と称する流れグラフの記述方式を提供している.

4. 外界の定義[2),3)]

organization または data-processing system は DPSS および ENVIRONMENT または INTERFACE から構成されている. 後者は DPSS の外の世界である. 図-2 にその構成を示している. 実時間を DPSS に教えるクロックは ENVIRONMENT 内にあるものと考える.

ENVIRONMENT はいくつかの PROPERTY SPACE から構成されていると考える. たとえば原子力発電所という ENVIRONMENT には原子炉, 蒸気タービンから運転員に至るすべての構成要素のもつ特質 (property) の集合として定義される PROPERTY SPACE というのがある. たとえば発電機には発生電力, 電圧, 電流……というような PROPERTY があ

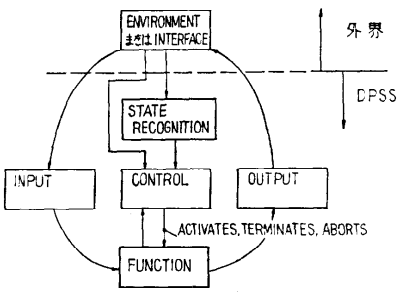

図-2 実時間システムモデル

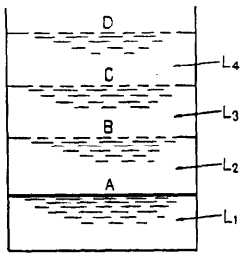

図-3(a) STATE の概念を示す例題図

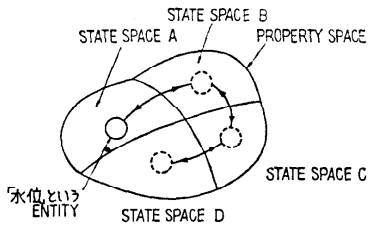

図-3(b) PROPERTY SPACE 上における ENTITY の動きの例

る. PROPERTY SPACE は ENVIRONMENT に囲まれたすべての構成要素のもつ PROPERTY の集合と考える.

ひとつひとつの PROPERTY はそれぞれ固有の「とり得る値」, または「とり得る値の範囲 (range)」をもっている. これを PROPERTY VALUE とよぶ.

PROPERTY VALUE の集合を状態 (STATE) と定義する.

以上に述べたことを例題によって示す.

ENVIRONMENT が単に 1 個の水タンクであるとする. この水タンクは図-3(a) に示すように L_1, L_2, L_3, L_4 という 4 つのレベルスイッチをもつものとする. 水位がレベルスイッチ L_i ($i=1,2,\cdots,4$) を越えるとその値は l_i となり, 水位が下ると $\bar{l}_i$ になるものとする. この例では L_i は PROPERTY であり, $l_i, \bar{l}_i$ は PROPERTY VALUE である. 今, $(l_1, \bar{l}_2, \bar{l}_3, \bar{l}_4)$ という PROPERTY VALUE の集合 (PROPERTY VALUE SET) を考えてみよう. この集合は図-3(a) で水位が A にあることを意味している. 同様に水位が B にあることは $(l_1, l_2, \bar{l}_3, \bar{l}_4)$ によって示すことができる. 水位が A, もしくは B にあるということは ENVIRONMENT の状態にほかならない. したがって PROPERTY VALUE SET により状態 (STATE) を定義することができる.

この例では水位 A, B, C, D という 4 つの状態を識

別することとしている．したがって水タンクというPROPERTY SPACE を4つのサブ空間に分けて考え，ひとつひとつの空間が状態に対応すると考える．このサブ空間のことを状態領域とよぶことにする．この例では水位という OBJECT または ENTITY が，この状態領域上を実時間の経過とともに動き回る．状態領域 a. b, c, d がそれぞれ水位 A, B, C, D に対応するものであるとすると，水位という ENTITY の軌跡は図-3（b）に示すとおりで，これ以外の軌跡はあり得ない．

以上に ENVIRONMENT を定義するためのひとつの方法を示した．大規模な organization または data-processing system では PROPERTY の数が数千個に及ぶことが珍しくない．したがって STATE を定義する場合に，PROPERTY VALUE SET を記述することが事実上，難しくなる．SET 内の要素数がきわめて大きくなるからである．したがって PROPERTY または PROPERTY VALUE の構造を考える必要がある．PROPERTY がアナログの型をもつときには，それが定められた閾値の上にあるか，下にあるかを示す PROPERTY VALUE も必要となる．

ENVIRONMENT の定義をまとめると次のようになる．

（1） PROPERTY の定義

外界の PROPERTY を定義することによって，DPSS に対するすべての入出力とその属性（信号の型，値の範囲，工学単位など），特性値（効率値や疲労度など）とその計算式などが明確になる．

（2） PROPERTY VALUE の定義

PROPERTY がもつ可能性のある値，または領域を定義する．

（3） OBJECT または ENTITY の定義

今，考えている DPSS 内の FUNCTION が着目している「もの」である．前記の例題は水位の制御を目的とし，「水位」が着目していた．したがって「水位」が ENTITY であった．

（4） STATE の定義

（3）で選択した ENTITY がとり得る状態を推定し，PROPERTY VALUE SET として STATE を定義する．

5. FUNCTION の定義

DPSS が行う「仕事」を FUNCTION と称する．これは何を（what）をなすべきかを定義する．FUNCTION は INPUT および SET 内のデータを用いて，OUTPUT 内のデータを生成するか，または SET 内のデータを更新する．SET というのは DPSS 内に管理されているデータの集合体である．

FUNCTION の活動開始（activate），終結（terminate），中断（abort）は CONTROL によって制御される．プロセスという語が計算機内のプログラムの実行における制御軌跡に対して用いられていることは公知のとおりである．この語は機能の実現方法という概念を伴うので，ここでいう「仕事」に対して用いることを避けているが，FUNCTION はプロセスとほぼ同一の概念と考えてよい．ひとつの FUNCTION はいくつかの SUB-FUNCTION の結合として詳細化される．SUB-FUNCTION はさらにいくつかの lower-level FUNCTION の結合として詳細化される．このような FUNCTION の定義法は多くのケースで採用されている．SUB-FUNCTION 間の結合に大別して2種類がある．

（1） ある一定の順序に従って連鎖的に，あるひとつの SUB-FUNCTION が終結してから次の SUB-FUNCTION が活動するような結合である．

この結合の途中に次のような機能をさしはさむことがある．

（i） 条件の判定によって結合を変更する．

（ii） タイマなどを用いて一定の時間おくれをさしはさむ．

（iii） あるイベントの発生を待つ．またはあるイベントが規定回数だけ発生するのを待つ．

（2） 複数の SUB-FUNCTION が同時に活動を開始し，並行して活動するような結合である．並行して活動している SUB-FUNCTION がその終結で同期せねばならないこともある．

FUNCTION や SUB-FUNCTION とその結合を記述するのには SADT (SofTech 社) や SREM における R-Net（チャート形式）および RSL（言語），その他の多くの方式がある．大別してチャートによる方式と言語による方式，およびその併合方式に分れる．計算機で要求を処理するシステムでは言語による記述をインプットするほうが処理上，都合がよい．

6. CONTROL の定義

CONTROL は FUNCTION の活動開始，終結，

	INPUT	(DYNAMIC) ENTITY	STATE	EVENT	CONDITION
INPUT			SUBPARTS		MAKES
(DYNAMIC) ENTITY			IDENTIFIED BY		
STATE	PART OF	IDENTIFIES		CAUSES WHEN ENTITY___ IDENTIFIES STATE___	
EVENT			CAUSED WHEN ENTITY___ IDENTIFIES STATE___		TRIGGERS SEARCH FOR
CONDITION	MADE			TRIGGERED SEARCH BY	

表-1 実時間システムにおける要求記述用関係詞の例

中断を制御する部分である．CONTROL はイベントの発生によって始まる一連の動作である．イベントは次のようにして発生する．

（1） ENVIRONMENT 内の OBJECT または ENTITY が，ある STATE から隣接する別の STATE へ移動したときに発生する．

（2） FUNCTION の働きによって，その中での定義に従って発生する．

（3） FUNCTION によって起動されたタイマ，カウンタが指定された機能を終えたときに発生する．

（4） あらかじめ指定した実時刻が到来したときに発生する．

イベントが上のいずれかによって発生すると，CONTROL は次に示す働きのいずれかを行う．

（1） あらかじめ定められた FUNCTION の活動を直ちに開始する．

（2） あらかじめ定められた FUNCTION の活動を直ちに終結するか，または中断する．

（3） あらかじめ定められた特定の CONDITION の真偽を調べ，対応する FUNCTION の活動を開始する．

（4） カウンタを増加，または減少する．

（5） タイマの作動を開始する．

以上のような CONTROL の働きを定義するためには，イベント (EVENT) および CONDITION の定義がまず必要となる．これらは先に述べた PROPERTY VALUE SET の並び集合で定義される．両者の差異は次のとおりである．イベントは処定の値をもつ PROPERTY VALUE SET の並びが成立した瞬間の「とき」が意味をもつのに対して，CONDITION はそれが調べられるときの PROPERTY VALUE SET の値が意味をもつ．

FUNCTION はあるイベントの中に含まれるひとつの PROPERTY VALUE を書き改めて，そのイベントを cause することができる．また FUNCTION は CONDITION の中のひとつの PROPERTY VALUE を書き改めることによって，内部状態を保存することができる．

CONTROL の動作は，上に述べた INPUT, ENTITY, STATE, EVENT, CONDITION, TIMER, COUNTER, FUNCTION などのそれら自身の定義と，それらお互いの間の関係を定義することによって明白となる．関係の定義はとくに設計された関係言語による記述によっている場合が多い．この言語の例としては ISDOS の PSL や SREM の RSL がある．この言語がどのようなものであるかを示すために筆者の開発している関係言語のうち，INPUT, ENTITY, STATE, EVENT, CONDITION のみの間の関係を記述するのに用いる関係詞の一部を **表-1** に示す．この表は行の特定の項目から入って目的の列にある関係詞を用いる．たとえば STATE→EVENT の間の関係詞は CAUSES WHEN ENTITY___IDENTIFIES STATE___ である．この関係詞を STATE と EVENT との間に，STATE に続けて入れて文を作る．たとえば次のようになる．

STATE high water level CAUSES EVENT open outlet valve WHEN ENTITY water level IDENTIFIES STATE ibid.

7. 要求処理システム

上に述べたような方法で記述された定義は計算機を援用する要求処理システムへインプットされて，その定義に誤りがないかどうか検査を行う．検査は static および dynamic な誤りに対して行われる．ここで static な誤りとは次のようなものを意味している．

- 記述上の誤り（文法上の誤りや定義されていない語の使用など）
- 定義の不十分な関係
- 論理上，成立し得ないと判断される関係
- 階層構造上，不具合な関係

dynamic な誤りとは次のようなものを意味している．

- イベント，カウンタ，タイマ，state などの動的な対象相互の実時間上での動的な関係の不具合，または不十分

dynamic な誤りを検出するためにはシミュレーションが併用される．一般に行われている方法は要求処理システムを GPSS や SIMSCRIPT のような汎用シミュレータと結合する方法である．

要求処理システムはタイムシェアリングシステムに組み込まれて，多数ユーザが端末からインプットすることを一般に可能としている．出力の一部は端末にアウトプットできるようになっている．要求を図形を用いて定義する方式では，特定の図形をインプットできるような端末が必要となる．文書のアウトプットは要求処理システムの重要な機能であるから，良質のラインプリンタ（日本語の印字機能をもつ）が必要である．要求処理システムが日本語処理機能をもつときには，端末も日本語処理能力をもつ必要がある．

要求処理システムにおいては，副次的ではあるが，data-dictionary という機能をもつことが要求される．これは入力されているある項目のもつ関係や使われている箇所を指定された条件に従って，まとめて表示する機能で，設計過程で効果的に利用される機能である．

以上に述べた機能を要求処理システムがもつためには，優れたデータベース管理システムを計算機がもつ必要がある．

8. むすび

一般に実時間システムは大規模であり，organization または data-processing system の中に DPSS が複雑に入り込んでいるものと考えられる．前者がプラントのような場合には，プラント主機の仕様が決定するまで長期間を必要とすることが多く，DPSS の設計はこれによって著しく影響を受ける．このような実時間システムは米国では国防省 (DoD: Department of Defense) 傘下の DPSS に見られる．DoD で要求処理システムが育ってきた最大の動機はソフトウェアライフサイクルにおける各段が異なる団体によって分担されているからであるといわれる．米国では伝統的に仕事が専門化され，横割りで分担されることが多いようである．これに比して，日本は文化的に縦割り社会とまでいわれるように，仕事を縦割りで処理する慣習に親しんでいる．タテヨコの長短はさておいて，要求処理システムにおいては，日本人のものの考え方に合致した独特の性質を付与する必要があると考えている．それがどのようなものであるかを言明できるほど，われわれの開発は進んでいない．本稿に述べた思想によって設計された要求処理システムは既に使用可能（英文で）となっており，試用している．ここで仕様化された format 2 と実現 A_2 との間の正しさ確認作業はわれわれの扱う大規模なソフトウェアでは基本的には人間の知能を用いて行うこととしている．なお，それを支援するツールが必要と思われるので設計を進めている．A_2 の format 2 に対する正しさの証明という問題は，ごく小さなソフトウェアについてまだ学究的に扱う段階にある．この事情は format 3 と A_3 との間の問題，すなわちプログラムの正しさ証明の問題と似ている．

本稿の内容の 3 以降については英文[3]で D. Teichroew の査読を受けたので，謝意を表しておく．

参考文献

1) Alford, M. W.: A Requirements Engineering Methodology for Real-time Processing Requirements, Trans. IEEE, Vol. SE-3, No. 1. pp. 60-69 (Jan. 1977).
2) Matsumoto, Y.: A Method of Software Requirements Definition in Process control, Proc. of COMPSAC 77, IEEE, pp. 128-132 (Nov.1977).
3) Matsumoto, Y.: A Study of a Requirements Definition System for a Part Controlling Functions of Embedded Computer System, ISDOS

Project paper (July 29, 1978).

4) Ramamoorthy, C. V.: Software Requirements and Specifications: Status and Perspectives, (revised). Tutorial: Software Methodology, IEEE publication No. EHO 142-0/78/0000-004 30 (1978).

5) Tanaka, S., et al.: New Concept Software System for Power Generation Plant Computer Control, COPOS, Proc. of 9-th PICA Conf., pp. 267-275 (1975).

6) Teichroew, D.: A Survey of Languages for Stating Requirements for Computer-based Information Systems, Proc. of FJCC, pp. 1203-1224 (Apr. 1972).

7) Teichroew, D. et al.: PSL/PSA: A Computer-aided Technique for Structured Documentation and Analysis of Information Processing System, Trans. IEEE, Vol. SE-3 No. 1, pp. 41-48 (Jan. 1977).

(昭和54年2月19日受付)