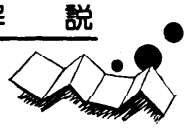
解　説



要求定義技術の動向†

野　木　兼　六††

1. はじめに

要求定義技術は，1976年に開催された第2回ソフトウェア工学国際会議において，初めて脚光を浴びて以来，現在に至るまで，この分野で最も大きな話題を集めている研究テーマと言って良いであろう．

1960年代後半に，“ソフトウェア危機”が叫ばれ始めた頃から，ソフトウェア開発における設計の重要性がいちはやく認識され，構造的プログラム作成手法[1]をはじめとする各種の設計方法論が次々と提案された．これらは，如何にして設計時の誤りを減少させるかという問題に対する解を与えてくれるものと期待されたが，1970年代の中頃になると，設計工程で発生する誤りには2種類のものがあることに，人々は気づくことになる．すなわち，設計結果がソフトウェアに対する要求を満足していないという誤りと，その要求自体が利用者の意図に反していたり，曖昧であったりするという誤りである．そして，一般に，前者の誤りより後者の誤りの方が修正も難しく，後続の工程により重大な影響を及ぼす．こうして，設計工程の初期の段階において，利用者の要求を正確に表現（仕様化）したり，それを分析して誤りを早い時期に発見するための要求定義技術の重要性が認識され，以後，要求定義の方法論や計算機によるその支援システムの研究開発が盛んに行われるようになる．

本稿では，現在までに提案されている各種の要求定義手法を整理して，より良い見通しを与えるという立場から，2. でまず設計工程における要求定義の位置づけを明確にし，3. で要求定義技術の基本的な枠組を与える．4. では，主な要求定義手法をモデルに従って分類し，それぞれの概要や特徴を説明する．最後に，今後の研究動向などについて述べる．

† Trends of Requirements Engineering by Kenroku NOGI (Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所

2. 設計工程モデル

設計工程に対する認識と用語は，人によってまちまちであり，しばしば誤解を生じる要因となる．そこで本章では，1つの設計工程モデルを与え，要求定義技術の位置づけを明確にすることを試みる．

大規模システムの設計は，通常，いくつかの段階に分割して行われる．各段階では，それぞれ，異なった概念が必要とされ，異なった言語や技術が使用される．これをいくつの段階に分割するかは，多分に運用上の問題であり，設計の性質から必然的に導かれるという訳ではない．しかし，ここでは，従来の設計技術を分類するのに都合が良いという理由から，これを3つの段階に分割する．すなわち，システム設計，ソフトウェア設計，モジュール設計である．そして，ここが重要な点であるが，これらの段階は，それぞれ，要求定義（あるいは要求分析）と設計生成から成る．要求定義は，大ざっぱな（必ずしも完全でない）要求から，それを満足するための論理構造を明確にすることによって，完全な要求仕様を作成する作業であり，一方，設計生成は，その要求仕様を実際に実現するための物理構造を決定することによって，設計仕様を作成する作業である．設計仕様は，次の段階に対する要求となる．要求定義は，「何を」するかを明確にすることであり，設計生成は，「如何に」するかを決定することであると言っても良い．従って，どちらかと言うと，要求定義よりは設計生成の方が自由度が大きく（すなわち，多くの代替案が考えられ），より創造性を必要とする作業であると言えよう．ただし，要求定義と設計生成の比重は，各段階によって変ってくる．要求定義技術の歴史的な発展過程からもわかるように，前の段階になる程，要求定義が大きな比重を占め，逆に，後の段階になる程，設計生成が大きな比重を占める．また，この比重は，対象システムによっても変ってくる．外界（あるいは利用者）とのインタフェースが複雑なシステムの場合には，要求定義の方が重要になり，逆の場合

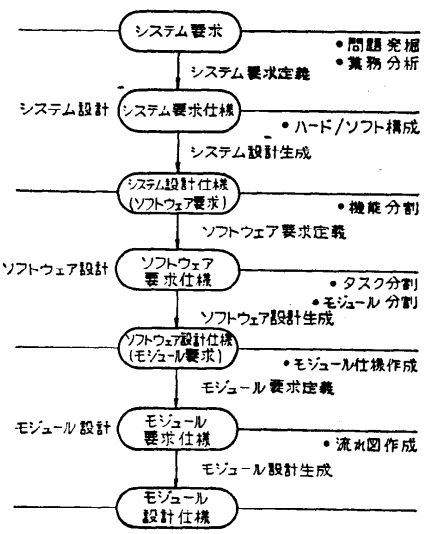


図-1 設計工程モデル

には，設計生成の方が重要になる．いずれにしても，設計工程では，要求定義と設計生成とが交互に現れるというところが大きな特徴である．この設計工程モデルを図-1 に示す．

従来，これらの概念に対して，いろいろな用語が使用されてきた．たとえば，ソフトウェア設計を方式設計，基本設計，概略設計などと呼び，モジュール設計を具体設計，詳細設計などと呼ぶ人もいる．また，要求定義のことを論理設計，外部設計，機能設計などと呼び，設計生成のことを物理設計，内部設計，処理設計などと呼ぶ人もいる．さらに，これらの用語を，異なったニュアンスで，使い分ける人も多い．要求定義という用語は，システム設計とソフトウェア設計に対してのみ使われることが多い．設計生成という用語は，筆者の造語であって，通常，使われない．

以下では，設計工程の各段階における主な作業を簡単に説明する．

（1） システム要求定義

システム要求定義では，利用者がかかえている問題を識別し，これを解決するのに必要なシステムの全体像を明確にする．このため，問題発掘，業務分析，動向調査，要求分析などの作業が行われる．問題発掘には，KJ 法[2] などのような発見的手法が利用できるであろうし，業務分析や要求分析には，SADT[15] のような機能分解手法が有効であろう．しかし，この段階では，システムの多くの側面を考慮しなければならず，1つの手法ですべてを包含するのは困難である．

（2） システム設計生成

ここでは，システム要求仕様に基づいて，具体的な実現手段を検討する．すなわち，システム要求定義で抽出された部分システムに対して，処理系（人間，機械，ハードウェア，ソフトウェアなど）を割りつけたり，情報の記録媒体（帳票，文書，カード，磁気テープなど）を決定したりして，システム性能，運用経費などの検討を行う．これに伴って，効果分析やプロジェクト計画なども行う．この段階については，従来，余り理論的な研究がなされなかったと言って良いだろう．

（3） ソフトウェア要求定義

ソフトウェア要求定義では，システムの中でソフトウェアに割りつけられた部分システムに対して，要求される機能や性能を明確にする．ソフトウェアを全くの暗箱 (black box) と考えて，その機能や性能をある程度形式的に記述するということが難しいために，通常は，何らかの意味での論理構造を与えるという方法がとられる．この作業は，従来の機能分割に対応するものと考えて良い．ISDOS[24] や SREM[26] など要求定義手法の代表的な例には，この段階を対象にしたものが多く，最も活発に研究が進められている分野と言える．

（4） ソフトウェア設計生成

ここでは，タスク分割，モジュール分割などによって，プログラムの物理構造を決定する．従来，設計手法と言えば，ほとんどがこの段階を対象にしたものであった．それだけ研究も進んでおり，構造的プログラム作成手法[1]，下降型プログラム作成手法[3]，段階的詳細化手法[4]，情報隠蔽によるモジュール化手法[5]，抽象データによるモジュール化手法[6]，複合設計手法[7] など，具体例は枚挙にいとまがない．

（5） モジュール要求定義

ここでは，モジュール分割によって得られた各モジュールに対して，その外部仕様を記述する．モジュールのように小さな単位になると，これを暗箱として扱うことが可能になり，形式的な仕様記述法[8]~[11] もいくつか提案されている．

（6） モジュール設計生成

ここでは，各モジュール内の概略処理手順を決定する．これを表現するのに，各種の流れ図 (flow chart) 記法や擬似コードなどが使われる．

3. 要求定義技術の概要

2. で述べたように，要求定義技術は，システム設計，ソフトウェア設計，モジュール設計という3つの段階に分類され，それぞれ，種々の異なった側面を持っているが，ここでは，これらに共通して見られる性質を抽出することによって，この技術の基本的な枠組を与える．

（1） 要求仕様の記述

要求定義を行うには，まず第1に，要求仕様を記述しなければならない．このために，要求仕様記述言語(図的な言語も含む)が必要になる．要求仕様記述言語は，設計工程における前段階あるいは利用者からの大ざっぱな要求を系統的に分解し，これをある程度形式的に記述することができるようなものでなければならない．記述すべき要求項目として，通常，次のものが考えられている．

（a） 機能要求
（b） 性能要求
（c） 設計上の制約条件
（d） その他の情報

機能要求では，今考えているシステム要素をとりまく環境に対して，許される入力，およびそれに対して期待される出力を明示する．この対応関係が複雑な場合には，そのシステム要素内の（論理的な）データの流れや制御の流れを記述して，理解を助けることになるかも知れない．しかし，これらは，あくまでも，機能を表現するための1手段であって，後続の段階における設計の自由度を制限するようなものであってはならない．性能要求では，各入出力に対する時間（応答時間，締切時間，周期など）とか，負荷（入出力量，頻度，分布など）によって，期待されるシステム要素の性能を明示する．設計上の制約条件では，実現時に使用されるべきハードウェア構成や操作システム (OS) の指定，推奨されるアルゴリズムなどを記述する．その他の情報としては，信頼性，保守性，拡張性などに関する要求とか，要求仕様の作成に伴って下された決定，参考資料，プロジェクト管理などに関する情報がある．要求仕様の記述法については，次章でより具体的に説明する．

（2） 妥当性の検証

要求仕様が記述されたら，その妥当性を検証しなければならない．前段階から与えられる要求には，一般に，互いに矛盾した要求とか，非現実的な要求とか，曖昧な要求とかが含まれており，これらがそのまま要求仕様にも反映される場合が多い．また，要求仕様の作成時に新たに発生する誤りも少なくない．従って，要求定義技術は，このような誤りを発見するのに有効な解析的手法を提供する必要がある．この解析的手法によって，通常，次のような検証が行われる．

（a） 一貫性
（b） 完全性
（c） 正当性
（d） 追跡可能性

一貫性とは，要求仕様の記述項目間に矛盾が無いこと，完全性とは，記述項目に抜けがないことである．また，ここでの正当性とは，データの流れや制御の流れを解析したり，シミュレーションを行うことによって，機能の正しさを確認することであり，追跡可能性とは，各記述項目からその源となった記述項目をたどることが可能であるということである．

システムに対する要求は，決して固定したものではなく，むしろ常に変化するものと考えなければならない．追跡可能性は，要求の変化に直ちに対処するための基礎を与えるものであり，その意味でも重要な概念と言える．

（3） 実現可能性の実証

要求仕様がたとえ妥当なものであっても，それが後続の段階で実際に実現できるとは限らない．このために開発の後もどりが生じる．この後もどりを防止するためには，要求仕様を記述した時点で，その実現可能性を実証しなければならない．しかし，これは技術的に非常に難しいことであり，余り一般的な方法はないようである．1つの解決策として，例となるアルゴリズムを実際に構築し，解析的なシミュレーションを行うという方法が提案されている[26]．

4. 要求定義技術の実際

本章では，現在までに提案されている主な要求定義手法を概観する．要求定義手法の分類法としては，適用段階によって分類する方法，データ中心か制御中心かによって分類する方法などが考えられる[12]が，ここでは基本となるモデルによって分類するという方法をとることにする．ただし，この分類は，各手法がいろいろな特徴を合わせ持っているという理由から，かなり主観的なものになっていることをおことわりしておく．なお，モジュール設計段階での要求定義手法（モジュール仕様記述法）については，本誌の1月号で詳

しく紹介されている[13]ので，以下では，システム設計およびソフトウェア設計段階での要求定義手法に焦点を絞って説明する．

4.1 機能階層モデルによる方法

システムやソフトウェアの機能を表現する最も自然な方法は，その入力と出力の対応関係を示すことであろう．大規模なシステムやソフトウェアの場合には，この対応関係が非常に複雑なものになるため，それが単純なものになるまで機能分解を繰り返すという方法がとられる．この方法の最も大きな特徴は，機能が概略から詳細へと段階的に示されるため，システムやソフトウェア全体の論理構造を直観的に理解することが比較的容易であるという点である．

（1） SADT

SADT (Structured Analysis and Design Technique)[14),15)] は，SofTech 社で開発された要求定義のための構造的分析手法である．この手法による要求仕様の記述は，**図-2** で示されるような階層化された図式 (diagram) と自然言語を併用することによって行われる．各図式は，箱と矢印から成り，それぞれ，名前と名札がつけられる．箱と矢印の関係は，**図-3** のように意味づけられる．ここで，入力，出力，制御はその箱のインタフェースを示し，機構はその箱を実現するための手段を示す．SADT においては，システムは "もの (things)" と "できごと (happenings)" の両面からモデル化される．このために，作業(activity)図式とデータ図式がある．作業図式では，箱で作業を示し，矢印で入力データ，出力データ，制御データおよびその作業を実行するための処理系を示す．データ図式では，逆に，箱でデータを示し，矢印で入力作業(データの生成)，出力作業(データの使用)，制御作業*およびそのデータを記憶するための装置を示す．これらの図式を記述するために，以上で述べたもののほかに，矢印の分岐や合流，省略，注釈などを示す記法が全部で 40 種類用意されている．

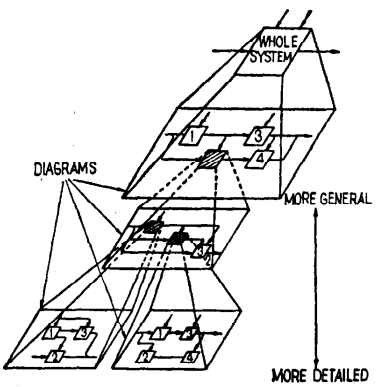


図-2 構造的分解（文献[1)]から引用）

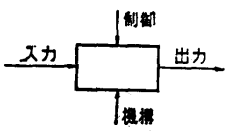


図-3 箱と矢印の関係

SADT は，既に多くのプロジェクトに適用され，かなり良い評価を得ているようである．たしかに，このような視覚的に優れた手法は，システム開発の初期段階において，問題を分析し概念を整理するのには，非常に有効であるに違いない．しかし，要求定義という観点から見れば，いくつかの問題点を指摘することができる．たとえば，SADT 図式で使われる記法の意味づけは非常に曖昧であり，ここからそのシステムがどのような場合にどのような機能を果すかということを正確に読み取ることは，きわめて困難である．また，性能要求のための記法は，全く形式化されていない．図式の改訂や管理を計算機で行うための支援システムがないことも，SADT の大きな欠点である．

（2） SAMM

SAMM (Systematic Activity Modeling Method)[16),17)]は，BCS (Boeing Computer Service Company) で開発された要求定義(と設計)のためのモデル化手法である．この手法は，SADT および同社で先に開発した DECA[18)] から大きな影響を受けている．基本的な考え方は，SADT とほぼ同じで，記法はむしろ単純化されている．すなわち，図式は作業図式のみで，しかも，そこでは入出力データの記述しか許されない．また，SADT のように，詳細な記法が用意されている訳でもない．その代りに，この手法に特有の記法として，機能のふるまいを示すための条件図 (condition chart) がある．これは，各出力データの生成に必要な入力データと条件を示すもので，SADT の曖昧さをある程度補うのに使われる．

SAMM の最大の目標は，要求の文書化や解析を計算機で支援することにある．このために，現在，SIGS (SAMM Interactive Graphics System) と呼ばれる支援システムが開発されている．

* 制御作業は，入出力作業を制御するものとされているが，この意味づけは余り適切とは思われない．

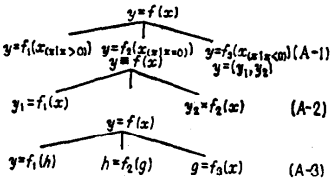


図-4 部分関数への分解規則（文献[12]から引用）

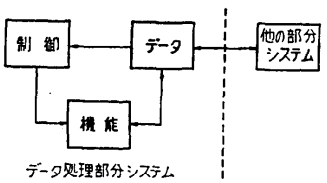


図-5 システム・モデル（文献[22]から引用）

（3） HOS

HOS (Higher Order Software)[19] は，MIT の Draper 研究所で研究中のソフトウェア定義手法である．この手法の特徴は，機能分解法に論理的な基礎を与えることにある．そこでは，各機能が数学的な関数と見なされ，図-4 に示されるような部分関数への分解規則が与えられる．そして，このような分解は，部分関数の入出力アクセス権や順序づけなどの制御に関する6つの公理を満足しなければならないものとされる．

要求定義技術の必要性だけが先走り，実体が伴っていないと言われる現状において，HOS のような形式的手法に対する期待は大きいが，この手法が実用的な水準に達するまでには，まだ多くの研究が必要であろう．

4.2 グラフ・モデルによる方法

ここで述べる手法は，いずれも，米軍の BMDATC (US Army Ballistic Missile Defence Advanced Technology Center) のソフトウェア開発支援システム SDS (Software Development System)[20] の一環として研究されているものである．これらは，戦略兵器システムなどのような大規模実時間システムにおけるデータ処理部分システムの要求定義を支援するのに使われる．このような分野では，特に，性能要求の検証が重要であり，形式的な取り扱いができるグラフ・モデルによる方法が有効となる．

（1） 検証グラフ (VG) による方法

この手法[21]は，CSC (Computer Sciences Corporation) で開発されたもので，(1)分解，(2)静的解析，(3)動的解析，という3つの段階から成る．各段階は，要求仕様の各水準に対して実行される．システム要求の分解を表わすのに分解要素 (DE) というものが使われる．DE は，入力(刺激)，機能要求，出力(応答)，性能要求などから構成される．これらの情報は，SSL (System Specification Language) という言語で定められた特定の用紙に記入されて，計算機に入力され，検証グラフ (VG) に変換される．VG は，節が機能を表わし，弧が刺激と応答の対を表わす有向グラフである．これは，システムの機能の流れを反映している．VG に対して，一貫性，追跡可能性，完全性が定義され，静的な解析が行われる．さらに，要求の動的な解析がシミュレーションによって行われる．

（2） 有限状態機械 (FSM) による方法

この手法[22]は，AFC (Aeronutronic Ford Corporation) で開発されたもので，図-5 に示すようなシステム・モデルを想定している．すなわち，システムにおいては，データ処理部分システムが中心的な役割を果し，他の部分システムを制御するものと考えられる．このモデルに従って，システムは，機能，制御，機能の流れ，データという4つの要素に分解される．システムの機能構造は，有向グラフで表現される．この有向グラフでは，節が機能を表わし，弧がデータによる影響を表わす．システムの制御機構は，有限状態機械 (FSM) として定義される．機能の流れは，FSM の出力である．すなわち，FSM がある状態に達した時に実行される一連の機能群を意味する．データは，機能の入出力や制御の入力になる．これらの4つの要素に対して，それぞれ，一貫性，完全性，到達可能性が定義され，検証される．FSM によって機能を機能の流れに類別するところが，この手法の特徴である．

（3） ペトリ・ネットによる方法

GRC (General Research Corporation) で開発されたこの手法[23]は，データ処理部分システムの要求仕様を表わすのに，ペトリ・ネットを使っている．ペトリ・ネットは，システムの動作規則を記述するのには有効であるけれども，これだけでは，要求の分解の一貫性の検証や性能要求の評価が行えない．そこで，この手法では，ペトリ・ネットを形式論理で表わすことによって，定理証明系 (theorem prover) を利用した一貫性の検証を可能にし，一方，性能要求の評価については，ペトリ・ネットの付点と推移に対して，それぞれ，“属性”と“推移の手続き”を割り付けることに

よって，シミュレーションを行っている．

4.3 関係モデルによる方法

関係モデルは，システムやソフトウェアの論理構造を表現するための，最も強力な道具であろう．このモデルは，要素，関係，属性という3つの基本概念から成る．要素は，名詞に相当するもので，システムの構成要素を表わす．各要素は，作業とか，データと言うような型によって分類される．関係は，動詞に相当するもので，要素間の関係（基本的には2項関係）を表わす．属性は，形容詞に相当するもので，要素の性質を表わす．各属性には，数とか，文字列と言うような値の集合が対応づけられる．関係モデルによる方法では，通常，各要素に対して，その要素と他の要素との関係およびその要素の属性を記述することによって，システムやソフトウェアの論理構造を表現する．この方法の特徴は，要求仕様を記述するのに各種の具体的な概念が使えること，拡張性に富んでいること，強力な支援システムが作り易いことなどである．

（1） ISDOS

ISDOS (Information System Development and Optimization System)[24),25)] は，ミシガン大学で開発された情報処理システムのための要求定義手法である．この手法では，関係モデルに基づいた言語 PSL (Problem Statement Language) によって要求仕様が記述されて，計算機に入力され，支援システム PSA (Problem Statement Analyzer) によって各種の解析および文書化が行われる．PSL によるシステムの記述は，次の9つの側面からなされる．

（a） システム入出力：システムと環境との関係．
（b） システム構造：処理などの階層的関係．
（c） データ構造：データ間の関係．
（d） データ生成：データと処理との関係．
（e） システム・アーキテクチャ：物理構造との関係．
（f） システムの寸法と量：システム負荷やデータ量．
（g） システムの動特性：システムの時間的な動き．
（h） システムの性質：キーワードや属性．
（i） プロジェクト管理：記述者や参考資料．

PSA には，計算機内に蓄積された要求仕様の変更機能や各種のポート作成機能がある．レポートには，参照レポート，要約レポート，解析レポートなどが含まれる．

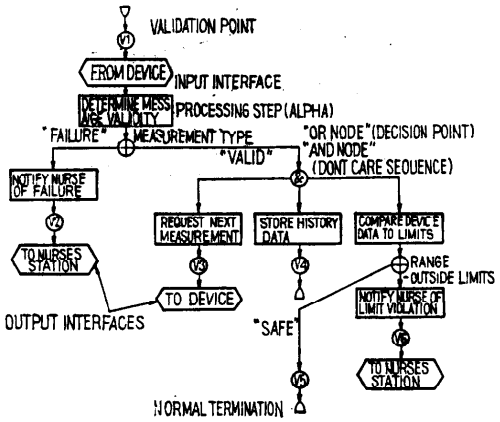


図-6 R-ネットの例（文献[24]から引用）

ISDOS は，この分野における先駆的な研究であり，他の要求定義手法に少なからぬ影響を与えた．また，これまでに多くのプロジェクトで実際に使われ，効果があったと言われている．このような使用経験を通して，現在も，言語や支援システムの改良が続けられている．この手法の問題点は，要素の局所性(有効範囲)が表現できないこと，関係の意味づけが曖昧なこと，図的なイメージが弱いこと，複雑なデータの流れが表現しにくいことなどである．

（2） SREM

SREM (Software Requirements Engineering Methodology)[26)~29)]は，TRW 社で開発された実時間システムのためのソフトウェア要求定義手法である．この手法は，要求仕様記述言語 RSL (Requirements Statement Language), 支援システム REVS (Requirements Engineering and Validation System), 要求定義手順という3つのものから構成される．RSL の最大の特徴は，関係モデルにおける要素，関係，属性に加えて，構造を4つめの基本概念としている点であろう．この構造の1つに R-ネットがある．R-ネットは，入力インタフェース対応に記述されるもので，メッセージに対する一連の処理の流れを表わしているこのようなメッセージ処理を単位として要求を分解する方法は，実時間システムにとって，きわめて自然なものであり，機能や性能の検証にも都合が良い．R-ネットの例を図-6 に示す．REVS には，R-ネットを対象とした，グラフィック端末による会話型生成編集機能，静的解析機能，シミュレーション機能などが含まれている．

SREM は，初期の ISDOS から大きな影響を受け

ており，基本的な特徴には類似の点が多いが，R-ネットという新しい要素を導入したことにより，改善された点も少なくない．たとえば，図的なイメージが強化されたこと，機能の表現が形式化され自然言語で記述しなければならない部分が減ったこと，シミュレーションが可能になったことなどである．一方，R-ネットの問題点は，データの流れを陽に示すことができないことである．このため，処理間の関係や R-ネット間の関係を理解することは，支援システムを利用しない限り，きわめて困難である．また，これは関係モデルに共通した問題点であるが，データの階層構造が単なる関係によって表わされているため，下位のデータに対する入出力を上位のデータに関連づけることが非常に難しい．このことも，事態を悪化させている要因になっている．しかし，全体として見れば，SREM は，かなり優れた要求定義手法と言って良いであろう．

5. おわりに

本稿では，設計工程における要求定義の位置づけ，要求定義技術の基本的な枠組，個々の手法の概要や特徴などについて述べた．ここでは，要求定義技術の体系化に重点を置いたため，個々の手法については，説明不足の点が多かったかも知れない．詳細については，それぞれの文献を参照されたい．

本稿で余り具体的に説明しなかったものに，要求定義支援システムがある．この中で特に面白いのは，関係モデルに基づいた言語の拡張機能と汎用抽出機能である．利用者は，言語の拡張機能によって，新しい要素の型，関係，属性を自由に定義することができる．これらの情報は，データベースに格納され，汎用抽出機能によって，新しい定義に基づいた利用者からの検索要求を処理するのに使用される．このようにして，利用者は，それぞれの応用分野に適した概念を使って要求定義を行うことができるようになる．言語の拡張機能と汎用抽出機能は，SREM の REVS ならびに ISDOS の META システムと汎用解析系 (Generalized Analyzer) によって，実際に支援されている．

要求定義技術は，まだ非常に未熟な段階にあり，今後解決しなければならない技術的課題も多い．その中で最も大きな問題は，形式性 (formality) の欠如であろう．もっと多くの部分を形式化しない限り，機能を正確に伝達することもできないし，もちろん，正当性の検証も不可能である．今後，より強力なモデルの検討，形式論理の導入などが必要になろう．もう1つの大きな問題は，要求定義と設計生成の接続に関するものである．すなわち，要求仕様がうまく作成できたとして，それでは，次の段階で，これを満足するような設計仕様をどのようにして作成すれば良いかということである．これには，要求仕様から設計仕様への追跡可能性の保証と最適化の問題が含まれる．設計自動化システムやプログラム自動作成システムなどの研究開発が，今後ますます重要になろう．

最後に，日頃御指導頂いている中田育男博士に感謝の意を表する．

参考文献

1) Dijkstra, E. W.: Notes on Structured Programming, Academic Press (1972).
2) 川喜田二郎：発想法，中公新書 136，中央公論社 (1976).
3) Mills, H. D.: Top-down Programming in Large Systems, Debugging Techniques in Large Systems, Prentice Hall, pp. 41-55 (1971).
4) Wirth, N.: Program Development by Stepwise Refinement, CACM, Vol. 14, No. 4, pp. 221-227 (1971).
5) Parnas, D. L.: Information Distribution Aspects of Design Methodology, IFIP-71, pp. 339-344 (1971).
6) Liskov, B. H. and Zilles, S. N.: Programming with Abstract Data Types, SIGPLAN Notices, Vol. 9, No. 4, pp. 50-59 (1974).
7) Myers, G. J.: Reliable Software through Composite Design, Mason/Charter Pub. (1975).
8) Zilles, S. N.: Algebraic Specification of Data Types, Project MAC Progress Report, MIT, pp. 52-58 (1974).
9) Wulf, W. A., London, R. L., and Shaw, M.: Abstraction and Verification in Alphard, Introduction to Language and Methodology, CMU-CS Report (1976).
10) Parnas, D. L.: A Technique for Software Module Specification with Examples, CACM, Vol. 15, No. 5, pp. 330-336 (1972).
11) Robinson, L. and Roubine, O.: SPECIAL—A Specification and Assertion Language, Technical Report CSL-46, Stanford Research Institute (1977).
12) Ramamoorthy, C. V. and So, H. H.: Software Requirements and Specification: Status and Perspectives, bit 8月号臨時増刊，pp. 63-119 (1978).
13) 鳥居宏次，二木厚吉，真野芳久：プログラミング方法論の展望，情報処理，Vol. 20, No. 1, pp.

22-43 (1979).
14) Ross, D. T.: Structured Analysis (SA): A Language for Communicating Ideas, IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, pp. 16-34 (1977).
15) Ross, D. T. and Schoman Jr., K. E.: Structured Analysis for Requirements Definition, IEEE Transaction on Software Engineering, Vol. SE-3, No. 1, pp. 6-15 (1977).
16) Stephens, S. A. and Tripp, L. L.: Requirements Expression and Verification Aid, Proceedings of the 3rd International Conference on Software Engineering, pp. 101-108 (1978).
17) Lamb, S. S., Leck, V. G., Peters, L. J., and Smith, G. L.: SAMM: A Modeling Tool for Requirements and Design Specification, Compsac 78, pp. 48-53 (1978).
18) Carpenter, L. C. and Tripp, L. L.: Software Design Validation Tool, Proceedings of International Conference on Reliable Software, SIGPLAN Notices, Vol. 10, No. 6, pp. 395-400 (1975).
19) Hamilton, M. and Zeldin, S.: Higher Order Software—A Methodology for Defining Software, IEEE Transaction on Software Engineering, Vol. SE-2, No. 1, pp. 9-32 (1976).
20) Davis, C. G. and Vick, C. R.: The Software Development System, IEEE Transaction on Software Engineering, Vol. SE-3, No. 1, pp. 69-84 (1977).
21) Belford, P. C., Bond, A. F., Henderson, D. G., and Sellers, L. S.: Specifications: A Key to Effective Software Development, Proceedings of the 2nd International Conference on Software Engineering, pp. 71-79 (1976).
22) Salter, K.: A Methodology for Decomposing System Requirements into Data Processing Requirements, Proceedings of the 2nd International Conference on Software Engineering, pp. 91-101 (1976).
23) Balkovich, E. E. and Engelberg, G. P.: Research towards a Technology to Support the Specification of Data Processing System Performance Requirements, Proceedings of the 2nd International Conference on Software Engineering, pp. 110-115 (1976).
24) Teichroew, D. and Hershey, E. A. III: PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems, IEEE Transaction on Software Engineering, Vol. SE-3, No. 1, pp. 41-48 (1977).
25) Teichroew, D. and Bastarache, M. J.: PSL User's Manual, ISDOS Working Paper No. 98 (1975).
26) Alford, M. W.: A Requirements Engineering Methodology for Real-Time Processing Requirements, IEEE Transaction on Software Engineering, Vol. SE-3, No. 1, pp. 60-69 (1977).
27) Bell, T. E., Bixler, D. C., and Dyer, M. E.: An Extendable Approach to Computer-Aided Software Requirements Engineering, IEEE Transaction on Software Engineering, Vol. SE-3, No. 1, pp. 49-60 (1977).
28) Alford, M. W.: Software Requirements Engineering Methodology (SREM) at the Age of Two, Compsac 78, pp. 332-339 (1978).
29) Alford, M. W., et al.: Software Requirements Engineering Methodology, SREP Final Report, Vol. I, II, III (1977).

(昭和54年3月2日受付)