ソフトウェア工学 71－2
（1990．2．8）

ソフトウェア仕様記述モデルの形態学

大槻 繁

（株）日立製作所 システム開発研究所

ソフトウェア開発過程の初期段階では，対象とする世界の問題仕様を記述する言語が主要な役割を果たす．近年，特に自動プログラミングのパラダイムを指向して数多くの仕様記述言語が提案されている．これ等の言語の基礎となる種々の仕様記述モデルは，問題の静的な範囲を限定化する「情報」，動的な時間上の制約を与える「ふるまい」，対象世界の世界観を構成する「個体」という3つの直交概念によって体系化することができる．この体系によって，表層的な言語表記法とは独立に仕様記述モデルを位置づけ，ひいては，対象領域の特性に応じて仕様記述モデルを選択したり，あるいは，開発の過程を分析する手がかりを与えることができる．

The Morphology of Software Specification Models

Shigeru Otsuki

Systems Development Laboratory, Hitachi, Ltd.
1099 Ohzenji Asao-ku Kawasaki-shi, 215 Japan

At the initial stage of software development process, languages for describing problems in its objective world play the important role. Many specification languages towards the automated programming paradigm are presented recently. The basic models of these languages are constructed by three orthogonal conceptual components:"information" bordering the static scope of problems, "behavior" constraining the dynamism on time or timing, and "individual existence" constructing the objective world view. By using this conceptual system, specification models are classified independent from their representational notations, criteria for selecting models from the point of the domain characteristics could be guided, and the basic framework for analyzing the development process could also be elaborated.

１．　はじめに

ソフトウェアの意義は，対象世界の中で初めて規定されるものであるから，その仕様も対象世界の中で解釈される．一面を持っていなくてはならない．すなわち，ソフトウェアの仕様には，計算機の実現とは独立に対象世界あるいはそこで解決したい意図を示す局面と，それを計算機の資源を使用して解決する意思決定を示す局面とがある．

我々は，この前者の対象世界に対応させた解釈を持つ言語のことを「対象世界記述言語」と呼んでおり，本論文では，仕様記述言語の対象世界記述言語としての側面である意味論を中心に論ずる．対象世界記述言語による仕様記述は，人間の意志伝達の手段であるから，その言語概念は，人間の対象世界の認識方法と自然な対応を成さなくてはならない．すなわち，対象世界記述言語は対象世界の像になっていなくてはならない．従って，言語の基本構成概念は，人間の対象世界の認識単位とすることを要請している．我々は，この認識単位を抽出するために既存の仕様記述言語をいくつか分析し，比較検討することによって，対象世界の問題を認識するのに必要な基本構成概念を抽出することを試みた．

２．　対象世界記述モデル

本節では，本論文の考察範囲を定めている「対象世界記述モデル」という用語について説明する．

まず，「対象世界」という用語であるが，これは，開発対象となるソフトウェアが稼働すべき実世界のことを意味している．もし，ソフトウェアの開発が，ユーザ（顧客）と開発者との間の契約によって為されるのであれば，対象世界とは，ユーザの世界である．例えば，ある図書館がその業務を支援する計算機システム（ソフトウェア）を，あるソフトハウスに発注する場合，対象世界とは，この図書館の業務世界のことを示している．無論，ユーザが同時にその開発者である場合もあり得るが，この場合でも対象世界は，ユーザの立場から見た世界ということになる．「対象世界」に対峙する用語としては，ここでは「計算機世界」という用語を用いることにする．計算機世界とは開発者の世界のことである．

人間は，何かを表現したり伝えたりしようとする時，「言語」を使用する．上述の「世界」について語る時にも言語を使う．ここで言う言語とは，アルファベットや図式要素を組み合わせた体系に限定して考える．音声による発話や，手足やものを使った記号は考察対象から排除する．我々は，対象世界について記述する言語のことを「対象世界記述言語」と呼び，計算機世界について記述する言語のことを「計算機世界記述言語」（これは通常「プログラミング言語」と呼ばれている）と呼ぶ．これ等の言語によって記述されたもののことを一般的に「仕様」と呼んでいる．従って，「ソースコード」は，ある特定の「プログラミング言語」（例えばPascal）によって記述された一種の「仕様」である．

言語には，アルファベットの並びや，図式の配置を規定している「構文（表現規則）」がある．特に，言語が人工言語の場合には，この構文は厳格に規約される．一方，言語の「意味」というのは，その言語の構文に従った「表現」の解釈の仕方を決めるものである．この解釈とは，人間の持っている共通概念への対応付けを行うことである．我々は，この意味を伴った言語体系の共通概念そのもののことを「モデル」と呼ぶことにする．従って，対象世界記述言語のモデルを「対象世界記述モデル」，計算機世界記述言語のモデルを「計算機世界記述モデル」と呼ぶ．ある言語の意味を定義するためによく行われるのは，意味が明確な他の言語への表現上の対応関係を示す方法である．意味が明確な言語として最もよく用いられるのは，数学と，厳密な使い方をした自然語（日常語）である．本論文でも，数学的な裏付けを持つ言語と，自然語とを主にモデル自身を記述する言語として用いることにする．このように「言語」や「モデル」というものをとらえると，ある一つの言語構文に対して複数のモデルが対応したり，逆に，ある一つのモデルに対して複数の言語構文が対応する場合があることに注意しなくてはならない．

本来，ソフトウェアとは，対象世界の問題を計算機によって解く仕掛であるから，ソフトウェアの仕様には，対象世界の問題そのものを記述したものと，その問題を計算機による実現解として記述したものとが含まれているはずである．対象世界と計算機世界とは，元来全く異なる世界であるから，これ等二つの世界に関する記述も異質のものであるはずである．言語は世界の像であるという立場からすれば，対象世界記述言語と，計算機世界記述言語とは，異なる言語体系がなくてはならないことになる．無論，ある一つの構文を共有し，対象世界記述モデルと計算機世界記述モデルとを別々に持つ言語も理想的な可能性としてはあるが，このような言語を一般的に設定することは，現状の技術レベルでは難しい．

しかしながら，上記のように対象世界記述言語と計算機世界記述言語とを定義すると，ある対象世界に計算機システムが含まれている場合に，その対象世界を記述するための言語は，どちらに属するものかという疑問が生ずる．例えば，図書館に文献検索システムが既に導入されている場合，この計算機システムも対象世界の一部を構成している．ここに図書の貸出管理システムを導入するための対象世界の問題仕様記述を行おうという場合，図書の登録や，貸出，返却の業務を記述すると同時に，文献検索システムの対象世界でのあり方をやはり対象世界の見方で記述できなくて

はならない．従って，一般的に対象世界記述言語の中には，計算機世界記述言語の外部仕様に関わる部分を含んでいることが要請される．

さらに，基本的に対象世界記述に対して，最終的に計算機世界記述が解として与えられる分けであるから，むやみに対象世界について何でも記述できるようにしても無意味である．この視点から言うと，既存の仕様記述言語は，概ね，計算機による解が得られることを保証している傾向がある．というより，計算機世界記述言語を対象世界の記述のために流用しているといっても過言ではない．従って，対象世界記述言語は，計算機による解を保証しつつも，対象世界本来のモデルを持つ体系でなくてはならない．

最後に，対象世界記述の範囲についての問題として，問題仕様記述との区別の問題がある．すなわち，対象世界の内，計算機を導入したり，関わりのある部分だけを記述するのが問題仕様記述である．厳密に言うと，計算機世界記述に関係するのは，この問題仕様記述の部分だけである．しかし，対象世界記述言語と問題仕様記述言語とを区別することにあまり意味はない．なぜならば，対象世界の内，どの部分が計算機に関わりがあるか，あるいは，どの部分が問題となるかということは，初期の段階で判断することは難しいし，また，ライフサイクル全体の視点から言うと問題の範囲は，対象世界の中で動的に変わって行くものであるからである．従って，本論文では，問題仕様記述モデルと対象世界記述モデルとを区別しないことにする．

3．　構成概念

3．1　情報

ここで言う「情報」とは，端的に言うならば，空間概念を示している．基礎となるモデルは抽象データ型である．「データ」，「値」，「関係」，「関数」等の諸概念はすべて情報概念によって規定される．

ソフトウェアは，情報システムであるから，その操作の対象となる情報が何であるかを値の集合，および，その値間の関係によって記述することが要請される．この意味で情報とは，仕様記述モデルの中での範囲付けを行う土台になっている．

さらに，値の集合とその上の操作によって規定されるデータ型に対して，外部仕様の独立性を保持するための抽象データ型や抽象型構成子の概念も情報概念に含めて考えることにする．

3．2　ふるまい

「ふるまい」は，時間概念を示している．ここでは，簡単のために，基礎となるモデルとして有限状態機械を考える．「事象」，「生起順序」等の諸概念はふるまいによって規定される．

事象とは，対象世界において、ある時点で起こる事柄のことである．事象には一般的に経過時間というものはなく，一瞬にして起こる事柄，あるいは，経過時間そのものを考慮に入れない事柄のことを言う．また，事象そのものが対象世界で観測されるか否かも問わない．

ふるまいとは，この事象の間の生起順序関係として定義される．

3．3　個体

「個体」あるいは「個体概念」とは，対象世界を認識するための世界観の構成概念である．

個体の種別としては，実体，プロセス等が考えられる．注意すべき点は，例えば，プロセスという概念を規定するために，時間を使用していないことである．我々は，プロセスという個体を時間概念とは独立に明証的に認識しているものと考える．

4．　ソフトウェア仕様記述モデル各論

以下，代表的なソフトウェア仕様記述モデルについて説明し，各モデルについて，上記の視点で個別の考察を行う．

4．1　抽象データ型

抽象化の概念は，古くはL.Dijkstra等の抽象化設計法[Dahl,72]に始まり，以来，プログラミングの技法の中心概念として発展してきた．近年この分野の理論的整備も進み，代数的仕様記述法による抽象データ型の定式化が活発に進められている．

代数的仕様記述法の理論的な基盤は，多ソート代数と等式論理であり，これを，ごく簡潔に述べると以下のようになる．

（1）データ型は，値の集合の集まりと，その上で定義されている演算の集まりとの対である．すなわち，データ型は多ソート代数である．

（2）抽象データ型は，データ型の実現や表現を隠ぺいし，その演算が何をするかということを抽象化したものである．これを演算の間に存立する項の同値関係によって記述する．（この記述によって規定される始代数をその意味する抽象データ型とみなす[Goguen,78]．）

（3）抽象データ型の実現は，別の抽象データ型の上に構成することによって与えられる．

対象世界記述において，抽象データ型は，第3節で述べた意味での情報概念を純粋な形で定義することができる．

対象世界における情報分析では，データを名前付けすることによって分類したり，さらに，これ等を構造化したりする．これは，ラベル付けを伴う値集合，及び，型構成子によるデータ型の導出を行っていることに相当する．

例えば，ジャクソン構造化プログラミング[Jackson,75]では，プログラムの入出力のデータとその構造を直積，直和，列構造を使って，図1に示すように木構造で表す．木構造の終端の箱は，基礎となる値の集合に相当し，中間（非終端）の箱は，その下位の箱の示すデータ型を直積，直和，あるいは，列の型構成子によって導出されたデータ型を表している．

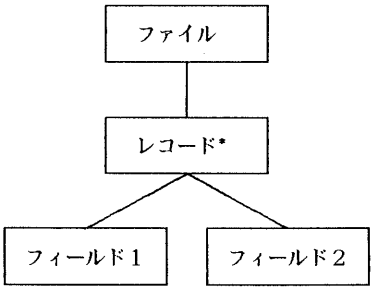


図1．　ジャクソン・データ構造図

4．2　状態機械モデル

4．2．1　有限状態機械モデル

有限状態機械の概念は，オートマタ理論[Hopcroft,70]の中で詳しく論じられている．有限状態機械の定式化には，いくつかの方法があるが，ここでは，次の方法を採用する．

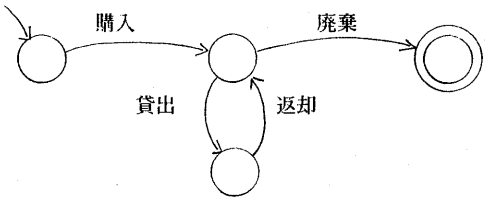
アルファベットΣ上の有限状態機械Mとは，

体系＜K，Σ，δ，q_0，F＞である．

（1）Kは，非空の状態を表す有限集合．
（2）Σは，有限の入力アルファベット．
（3）δは，K×ΣからKの中への関数（状態遷移関数）．
（4）q_0は，Kの要素であり，初期状態を表す．
（5）Fは，Kの部分集合であり，最終状態の集合を表す．

対象世界記述においては，この有限状態機械を，事象の生起順序を規定するものとして解釈する．このことは，Σを，事象の集合に対応させて解釈することを意味している．すなわち，一つの有限状態機械の記述は，対象世界の純粋なふるまいの記述である．

有限状態機械モデルの表現形式として，状態遷移図式と正規表現があり，図2は，この両者によるふるまいの表現を示している．



(a)状態遷移図式

購入　（貸出　返却）*　廃棄

(b)正規表現

図2．　有限状態機械モデルの表現形式

無論，ふるまいを純粋に表現する手段として，有限状態機械モデル以外のものも考えられるが，現状の一般的な仕様記述モデルで使用されているものを分析する手段としては，これで十分である．より高度の表現能力をもったふるまいの表現手段として，ペトリネットやその拡張モデルが使われることもある．しかし，この種のモデルの用途は，現状では個別的かつ特殊なものであり，今回の考察の対象外とした．

4．2．2　一般化順序機械モデル

ソフトウェア仕様記述の分野で，上記の有限状態機械モデルが単独で使用されることは，ほとんどない．このモデルに近くて，よく使用されるものは，有限状態機械を下記のように拡張した一般化順序機械[Hopcroft,71][Hughes,79]に基づくことが多い．

一般化順序機械Gとは，

体系＜K，Σ，Δ，δ，λ，q_0＞である．

（1）K，Σ，δ，q_0は，有限状態機械に同じ．
（2）Δは，出力アルファベットの集合．
（3）λは，K×ΣからΔ*の中への関数（出力関数）．

よく知られているように，λの値域をΔに制限したものがミーリ機械[Salomaa,69]と呼ばれているものである．

対象世界記述モデルの見方からすれば，集合Σは，入力アルファベットの集合であると同時に，事象の集合である．すなわち，対象世界を認識するに当たって，事象というものを入力（を受け付ける事）のみに限っている．さらに，該当する入力を受け付けたときに行うべき動作を，関数λによって仕様化する．この時の入力と出力アルファベットは，抽象データ型の値が使われる．

４．３　データフローモデル

データフローモデルには多くの派生的なモデルが存在し，情報系のソフトウェアを対象としたシステム分析フェーズを中心として多くの生産現場で使われているものである．

４．３．１　バブルチャート

データフローモデルの中で最も単純なものは，バブルチャート[Yourdon,78]である．これは，構造化設計（複合設計[Myers,78]と区別するためにYourdon法と呼ぶこともある）と呼ばれる設計技法の中で使われているもので，モジュール設計を行う前段階として，プログラムの機能をバブルチャートで記述する．この例を図３に示す．

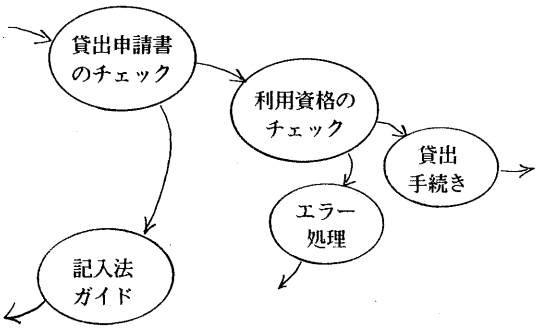


図３．　バブルチャート

バブルチャートは，図に示されているように，丸で表される処理と，矢印で表されるデータとによって構成されるグラフ（ネットワーク図式）である．ソフトウェアに対する仕様記述モデルの観点からすると，このグラフでは，データの流れのみを記し，制御や実行順序についての情報を排除しているのが特徴である．構造化設計の立場では，バブルチャートは，設計の自由度を確保するために，これ等の後者の情報を未決定にして残しておくための記述方法である．

これを，対象世界記述モデルの観点から見ると，まず，処理が認識の単位，すなわち，個体概念になっていることが分かる．このことと同時に，処理（丸）は関数を指向しており，情報の概念になっている．複数の処理の間の関係は，データを表す矢印で結合することによって示される．データもまた認識の単位ではあるが，あくまでも処理に従属している．矢印は名前付けされずに処理間の接続関係のみを示すことが多い．

４．３．２　SAモデル

構造分析技法（SA:Structured Analysis）[DeMarco,79]の中で使用されているデータフローモデルは，図４に示すように，バブルチャートに直線で表されたファイル（対象世界の意味での）を導入した点と，特別のバブルとしてシステムの外界のプロセス（四角で表す）を導入した点が際だった特徴である．

この拡張は，対象世界記述モデルの観点から見ると，認識の単位として，処理と外界プロセス，さらに，ファイルが個体概念として詳細化されている．

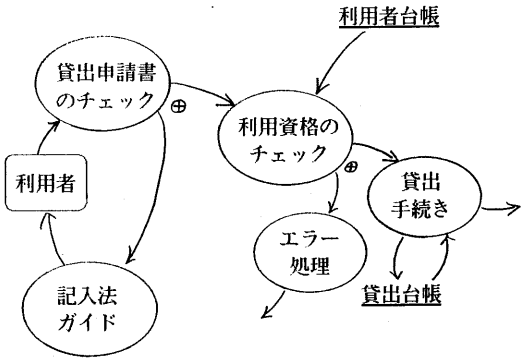


図４．　SAチャート

４．３．３　リアルタイムＳＡモデル

構造分析技法が情報システム向きの技法であるのに対し，これを，リアルタイムシステム向きに拡張したのがリアルタイムＳＡ[Hatley,84][Ward,86]である．例えば，Wardの方法では，データフロー上の各バブル間のデータ入出力，活性化，非活性化，信号等の事象間の生起順序を記述するために，仮想的なプロセスを示すバブル（点線の丸）を導入し，その中をミーリ機械によって仕様化しているのが特徴である．

この拡張は，対象世界記述モデルの観点から見ると，データフローモデルでは決定されていなかったふるまいを，一つの認識の単位（制御プロセス）として導入していると言える．

４．４　ジャクソンモデル

４．４．１　ＪＳＰ

ジャクソンプログラム構造化技法ＪＳＰ(Jackson Structured Programming)[Jackson,75]では，一つの実行単位としてのプログラムの仕様をデータ構造間の入出力の対応関係によって把握するモデルを用いている．この観点では，入出力のデータを直積，直和，列という構造によって抽象したデータ型を導入し，その間の対応関係を関数によって表した情報の規定を行っていることになる．

さらに，ＪＳＰでは，３種類の構造不一致の解決方法に関連して，中間ファイルという概念を導入している．これは，基本的にはデータフローモデルを使用しているが，図５に示すように，中間ファイルが個体概念として認識され

ている．なぜならば，これを一つの存在物として捕らえなければ，これに対して二つのデータ構造の見方を与えるということは考えられないからである．

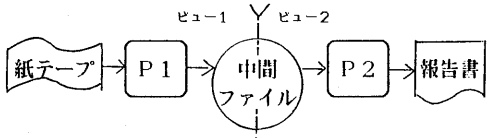


図5．　ジャクソンの中間ファイル

4．4．2　JSD

ジャクソンシステム開発技法ＪＳＤ(Jackson System Development)[Jackson,82][Cameron,86]で使用されている仕様記述モデルは，概ね，ＣＳＰ[Hoare,78]に基づいている．この観点から言うと，確かに，ＪＳＤモデルは，並列動作モデルではあるものの，しかし，その対象世界での解釈の仕方に特徴がある．すなわち，個体概念としてのプロセスに，対象世界（実世界）のモデルプロセスと，機能を規定するための機能（あるいは相互作用）プロセスとを分離し，モデルプロセスの定義は，図６に示すように対象世界のふるまいに完全に対応させている点である．

簡潔に言うならば，ＪＳＤのモデルプロセスは，プロセスという個体概念をふるまいによって記述したもの，また，機能プロセスは，ＪＳＰのデータ構造の入出力対応関係によって記述したものになる．

4．5　ステイトチャート

ステイトチャート[Harel,88]は，状態遷移モデルに，並列性と階層とを導入したモデルである．図７に示すように，状態のAND分解（点線によって区切る）によって，並列プロセスに分解される．また，状態のOR分解（状態の包含関係）によって，状態が詳細化される．いずれにせよ，ステイトチャートモデルで認識の対象となるのはふるまいであり，各プロセスは個体に対するビューを与えている．

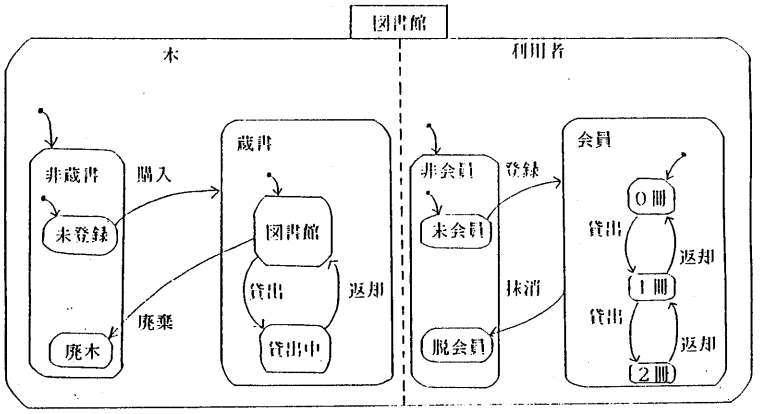


図7．　ステイトチャートのプロセスと状態

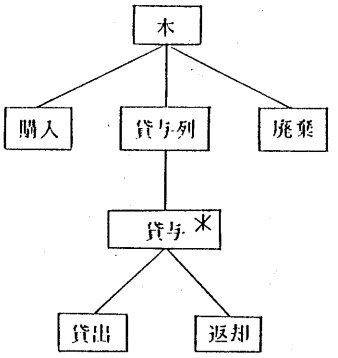


図6．　ジャクソンのモデルプロセス

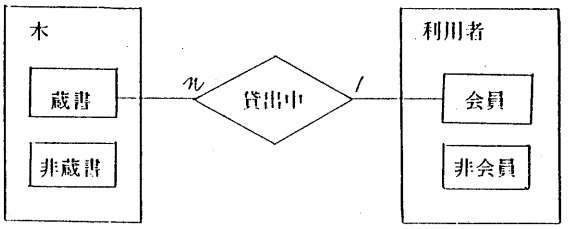


図8．　実体・関連モデル

4．6　意味データモデル

データベースの分野でスキーマ設計法として発展してきたものに意味データモデル[Hull,86]がある，これは，データ間の関係を正確にモデル化することを狙ったものである．実用的なレベルでよく使われる実体・関連モデル[Chen,76]もこの意味データモデルの一種である．

これ等のモデルは，対象世界記述の観点から見ると，図８に示すように，対象世界の存在物（抽象的なものを含む）個体概念の集合をそのままデータの値集合として捕らえていることに特徴がある．

５．　構成概念の体系

第１節の構成概念をごく簡単に体系化し，その中で種々の対象世界記述モデルを特徴付けることを試みる．

まず，各構成概念に記号を下記のように割り当てる．

・情報：　情報概念Ｉは，値集合概念Ｄ，あるいは，さらに値集合間の関係概念Ｆから構成される．
・ふるまい：ふるまい概念Ｂは，事象概念Ａ，あるいは，これを規定する状態概念Ｓから構成される．
・個体：　個体あるいは個体概念Ｏは，個体としての（抽象的）存在物の種類によって，プロセス概念Ｐ，実体概念Ｅ等から構成される．

三つ組＜α，β，γ＞によって，仕様記述モデルの表現範囲を表す．ここで，αは情報概念あるいはその構成概念を，βはふるまい概念あるいはその構成概念を，γは個体概念あるいはその種別を示している．また，対応する概念が存在しない場合これをϕで示す．また，各構成概念の対応関係を［］で囲んで示す．

上記の体系を使用すると，種々の対象世界記述モデルは，下記のように類型化することができる．

Ⅰ類：＜Ｉ，ϕ，ϕ＞

情報概念のみから構成されるモデルで，ふるまいや個体概念を持たないもの．多ソート代数に基づく抽象データ型モデル(4.1参照)が代表的である．

Ⅱ類：＜ϕ，Ｂ，ϕ＞

ふるまい概念のみから構成されるモデルで，情報や個体概念を持たないもの．有限状態機械モデル（表記方法としては正規表現や状態遷移図式等）(4.2.1参照)が代表的である．また，事象間の半順序を規定する時間論理も，これに分類される．

Ⅲ類：＜Ｉ，Ｂ，ϕ＞

情報とふるまい概念のみから構成され，個体概念を含まないもので，構成概念の組み合わせ方により，下記のように詳細化される．

・１型：［Ｆ＝Ａ］

情報の構成概念である関数を一つの事象として捕らえるモデルである．すなわち，ある事象が生起するということを，その時の入力に対していかなる出力が為されるかということによって仕様化する．一般化順序機械モデルやミーリ機械モデル(4.2.2参照)がこれに分類される．

・２型：［Ｆ＝Ｂ］

入力データと出力データとの間の対応関係としての関数概念とふるまい概念を同一視するモデルである．これは，ＪＳＰの入出力データ構造対応関係がこれに属する．ただし，これで，構造不一致が生じて中間ファイルを導入するとⅤ類３型になる．

Ⅳ類：＜ϕ，Ｂ，Ｐ＞

ふるまいと，個体概念としてのプロセスのみから構成され，情報概念が明示的に現れないもので，構成概念の組み合わせ方により，下記のように詳細化される．

・１型：［Ａ＝Ｐ］

一つの事象をそれぞれ存在物としてのプロセスとみなすモデルで，ペトリネットがその代表例である．

・２型：［Ｂ＝Ｐ］

一連の事象の生起順序関係であるふるまいに対して，プロセスを割り当てるモデルである．ステイトチャート(4.5参照)やＣＣＳモデル[Milner,89]は，これに分類される．これ等のモデルでは，プロセスというものが全て事象の関係という見方の中でのみモデル化されている．

Ⅴ類：＜Ｉ，ϕ，Ｏ＞

情報と個体概念のみから構成され，ふるまい概念を含まないもので，構成概念の組み合わせ方により，下記のように詳細化される．

・１型：＜Ｉ，ϕ，Ｅ＞［Ｄ＝Ｅ，Ｆ＝Ｅ］

情報の構成概念である値集合と関数（関係）とを，対象世界の個体概念（存在物）として捕らえる静的なモデルである．業務世界(enterprize schema)を対象としたデータベース技術の分野のデータモデルは，概ね，これに分類される．従って，意味データモデルや実体・関連モデル(4.6参照)はこれに属する．

・２型：＜Ｉ，ϕ，Ｐ＞［Ｆ＝Ｐ］

関数概念をそのまま個体概念として捕らえるモデルである．これには，バブルチャート(4.3.1参照)や一般の関数型モデルがある．

・３型：＜Ｉ，ϕ，Ｏ＞［Ｆ＝Ｐ，Ｄ＝Ｅ］

２型の発展形として，これに加え，値集合をそのまま個体概念として認識するものがある．これには，ファイル概念を追加したデータフローモデル(4.3.2参照)がある．また，ＪＳＰの構造不一致で現れる中間ファイルを含む仕様(4.4.1参照)もこれに属する．

Ⅵ類：＜Ｉ，Ｂ，Ｏ＞

全ての概念を含むモデルであり，下記のように詳細化される．

・１型：＜Ｉ，Ｂ，Ｐ＞［Ｂ＝Ｐ］

これは，Ⅳ類２型の情報概念の導入による拡張型である．この代表としてＬＯＴＯＳ[Brinksma,88]がある．

・２型：＜Ｉ，Ｂ，Ｏ＞
［Ｂ＝Ｐ（モデルプロセス），Ｆ＝Ｐ］

上記Ⅵ類１型とⅤ類２型とを融合化したもでる．すなわち，対象世界のプロセス概念をふるまいによって記述し，これと，関数型やバブルチャートの考え方を合わせたもの．ＪＳＤモデル(4.4.2参照)がこれに相当する．

・3型：＜I，B，O＞
［B＝P（制御プロセス），D＝E，F＝P］
V類3型にふるまい概念によって規定される制御プロセスを導入したもので，リアルタイムSAモデル(4.3.3参照)がこれに相当する．

・4型：＜I，B，O＞［F⊆P］
プロセスというものを関数の族としてとらえ，プロセスの内部状態（変数）を許すもの．CSP[Hoare,78]や，オブジェクト指向言語の多くがこれに属す．

上記の体系によって，概ね，仕様記述モデルを分類することができる．

＜φ，φ，O＞という分類が存在しない理由は，個体という存在概念を直接表現するモデルがないからである．我々は，これを認識はしているが，そのことを表すために，情報やふるまいのモデルを使用してこれを行っているものと考えられる．

6．おわりに

ここで述べた概念体系は，粗い体系であるが，これによって代表的な仕様記述モデルの位置づけを行うことができ，技法選択の見通しを得られるものと思う．さらに，直交概念の組み合わせにより導出される概念は，各ソフトウェア開発技法の手順の中で出現する基礎的な認識単位になっており，開発過程そのものを定式化する[Komiya,89][Saeki,90][Otsuki,89]補助的な枠組みを与えることができるものと考えられる．

最後に，本研究の機会を与えていただいた（株）日立製作所システム開発研究所の堂免信義所長，ご討議いただいた同研究所の西尾高典氏，金藤栄孝氏，および，情報処理振興事業協会のプロトタイピングの調査研究会の委員の方々に感謝いたします．

【参考文献】

[Brinksma,88] Brinksma,E.,"Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique based upon the Temporal Ordering of Observational Behavior", International Standard ISO8807,1988

[Cameron,86] Cameron,J.R."An Overview of JSD",IEEE Transaction on Software Engineering,Vol.SE-12,No.2, pp.222-238,1986

[Chen,76] Chen,P.P.,"The Entity-Relationship Model Toward a Unified View of Data",ACM Transactions on Database System,Vol.1,No.1,pp.9-36,Mar.,1976

[Dahl,72] Dahl,O.-J.,Dijkstra,E.W., and Hoare,C.A.R., "Structured Programming",Academic press,1972

[DeMarco,79] DeMarco,T.,"Structured Analysis and System Specification",Prentice-Hall,Inc., 1979

[Goguen,78] Goguen,J.A.,Thatcher,J.W. and Wagner,E.G., "Current Trends in Programming Methodology IV: Data Structuring", Prentice Hall,1978

[Hoare,78] Hoare,C.A.R.,"Communicating sequential processes",Communications of the ACM,Dec.1978

[Harel,88] Harel,D.,"On Visual Formalisms", Communications of the ACM,Vol.31,No.5,pp.514-530, May 1988

[Hatley,84] Hatley,D.,"The Use of Structured Methods in the Development of Large Software-based Avionics Systems",Proc. of IEEE 6th Digital Avionics Systems Conference,pp.6-15,1984

[Hopcroft 71] Hopcroft,J.E., and Ullman,J.D., "Formal Languages and their Relation to Automata", Addison Wesley,1970

[Hughes,79] Hughes,J.W.,"A Formalization and Explication of the Michael Jackson Method of Program Design",Software Practice and Experience, Vol.9,pp191-202,1979

[Hull 86] Hull,R., and King,R.,"Semantic Database Modeling: Surgey, Applications, and Research Issues",ACM Computing Surveys,Vol.19,No.3, pp.201-260

[Jackson,75] Jackson,M.,"Structured Programming", Prentice Hall,1975

[Jackson,82] Jackson,M.,"System Development", Prentice-Hall,1982

[Komiya(古宮) 89] Komiya,S.,"仕様記述過程モデル化のための実験と分析", 情報処理学会 ソフトウェア工学研究会69,1989 11月

[Milner,89] Milner,R.,"Communication and Concurency", Prentice-Hall,1989

[Otsuki,89] Otsuki,S.,"ソフトウェア仕様化技法における事象と状態概念の抽出過程に関する一考察", 日本ソフトウェア科学会第6回大会論文集, pp.365-368,1989 10月

[Saeki(佐伯),90] Saeki,M.,"仕様化・設計の方法論形式化のための一考察",ソフトウェアプロセスワークショップ（伊東），1990 2月

[Salomaa,69] Salomaa,A.,"Theory of Automata", Pergamon Press,1969

[Ward,86] Ward,P.,"The Transformation Schema: An extension of the Data Flow Diagram to Represent Control and Timing",IEEE Trans. Software engineering, Vol.12, No.2,pp.198-210,Feb.1986

[Yourdon,78] Yourdon,E., and Constantine,L.L., "Structured Design", Yourdon Press,1978