ソフトウェア工学 70－2
（1989．12．8）

代数的仕様記述のためのプロセス・モデル

白井 豊
協同システム開発（株）

別府 真　　小松 雅彦
（株）構造計画研究所

代数的仕様記述は機械的な検証が可能であること、記述途中でも実行可能であるため設計段階での強力な仕様記述法としての可能性を持っているが、現状の開発現場の技術者にとっては記述困難な側面を持っている。そこで、「代数的仕様に慣れない」技術者が用意に記述できるようにするために、仕様記述の検討過程のモデル化を行うことを目指して、記述及び思考実験を行い、その整理を行ったので報告する。実験では、代数的仕様を実際に記述しながら記述過程の思考・整理手順を記録し、他の仕様記述法との関連で考察した。

PROCESS MODEL FOR DESCRIPTION USING ALGEBRAIC SPECIFICATION LANGUAGE

Yutaka Shirai

Joint System Development Corp.
YUSEI-GOJOKAI-KOTOHIRA-BLDG, 14-1, 1-CHOME, TORANOMON, MINATO-KU, TOKYO, JAPAN 105.

Makoto Beppu　　Masahiko Komatsu

Kozo Keikaku Engineering Inc.
NIPPON-HOLSTEIN-KAIKAN, 38-13, 4-CHOME, HON-CHO, NAKANO-KU, TOKYO, JAPAN 164.

An algebraic specification method is a prospective way at design phases. The method has the advantage that written specification can be mechanically verified and executed even if not described in detail. It, however, involves certain difficulties for engineers working on developing practical systems. So, we made an experiment in order to make a model of the process of writing algebraic specifications, so that it may facilitate the task for engineers who are not used to using this method. The experiment was such that the subject, while writing a specification, recorded how he reached his ideas and arranged the given problem. The present paper will report the result of this experiment.

1．はじめに

人間の理解は「ことば」化されて初めて明示される。曖昧な理解の段階でも「～でありそうだ」、「～かも知れない」等の表現で頭に思い浮かべる。否定的な理解でさえ「なんだかおかしな気がする」等の表現を思い浮かべることによって初めてその否定的な感じ方を意識しはじめる。図表的に表現する際も「ツリー状に表現してみたら」、「この部分を絵にしてみたら」などの考えを重い巡らせながら紙の上に描いていく。

即ち、理解の最も初期段階は「ことばによる表現」である。発話が理解のスタート点である。

もっとも、人間による理解の構造が明らかにならない現段階での乱暴な定義かもしれない。しかし、言葉にならないものを研究するには形而上学的な視点が必要であり、言葉にならない段階を相手にするのは測定できないものを実験対象にするようなものである。

仕様記述の際の人間の試行錯誤において「自然言語による表現」を最初に意識するものとして選択するのは、以上のような意味で自然である。人間の理解の表現を「言語表現がもつ意味内容を計算機内部に表現すること」[1]と定義すれば、計算機内に記述された仕様記述は、人間の理解の表現を計算機内に表現したものと広い意味で同義である。

一方、自然言語表現は本質的に無限である。人間が生み出す言語表現は、時間が無限である以上増え続ける。その表現の中には曖昧なものもあり、同一の表現であっても人によって異なる場合もある。ソフトウェアの仕様記述が異なる人間同士の共通の土俵作りを行うものでなければならないとしたら、自然言語表現は極めて都合の悪いものであるといわざるをえない。

自然言語表現の無限の広がりを「文法」という形で整理し、有限な表現で押し込めようと試みた「言語学」が自然言語の理解に大きく寄与したのと同じように、ソフトウェアの仕様記述にも有限な表現形式を導入することがソフトウェアに携わる関係者間のコミュニケーションを深めることは間違いない。

しかしながら、有限の表現形式の導入は「表現の主体者」を束縛することになる。使い慣れた表現方法、例えば自然言語（話言葉を含む）ほど自由に表現できるわけではない。

我々が今回記述実験で扱った代数的仕様記述法も例外ではない。代数的仕様は、記述に慣れた技術者にとっては、平板で分かりやすく、手続き的な制御をそれほど気にする必要がないため記述しやすく、機械的な検証が可能である。一方、代数的仕様に慣れていない技術者にとって、表現方法が限られているのに加えて、数学的な厳密性が重視されることも災いして、「難しい」記述言語として考えられている記述法のひとつである。更に、既存のプログラミング言語とは一風変わった言語スタイルを持っており、ソフトウェア開発現場の技術者に受け入れられにくく普及が困難である。

我々は、「代数的仕様に慣れない」技術者が「代数的仕様に慣れた」技術者と同等あるいはそれ以上の記述（量・質ともに）を可能にする道具を揃えることを目指して、手始めに仕様記述の過程のモデル化を図り（仕様記述のプロセス・モデル）、将来的にそのモデルに沿った道具を整備できる礎を築きたいと考えている。

このためには、代数的仕様による仕様記述を行う際の検討過程、検討内容を整理して慣れない技術者に提示することが必要である。更に検討内容を単に羅列するだけでなく、初心者が理解しやすい形で概念を整理し検討の際の指針作りを行うことが必要である。しかも提示過程、内容、指針を初心者に提示するにあたって、理論的な枠組みにこだわった表現ではなく、なるべく少ない常識で理解できる表現として提示する必要がある。

そこで、我々は代数的仕様を実際に記述しながら、記述過程の思考・整理手順を記録し、記述及び思考実験を行い[2]、問題の整理方法に関する考察を行ったので報告する。

記述の題材は、「仕様記述及び設計に関する第４回国際ワークショップ」(the Fourth International Workshop on Software Specification and Design)における図書館問題である。代数的仕様記述言語としては通産省ＩＰＡ関連事業の一環として協同システム開発（株）で開発中の代数的仕様記述言語 ASPELA [3] を使用した。

２．仕様記述実験の手順

本実験では、次の手順で実験及び整理を行った[2]。

(1) 図書館問題の日本語化

(2) 図書館問題を代数的仕様記述で記述。記述途上で考えていること、その際の記述内容をその場で素直に記録していく。記述途上の誤り自体もそのままに残しておき、記述の変更がいつ起こったかを明確にできるようにする。

(3) 上記(2)の記録から繰り返し作業となっている部分の拾い出し、作業項目の集約、作業項目の順序付けを行う。特に思考上のバックトラックが起きる段階、他の担当者に分割して依頼することが出来るタイミングを特に重視した。

(4) 上記(2)の内容から視点あるいは評価基準として有効な記述を拾い出し、上記(3)で順序付られた作業項目に振り分ける。

３．代数的仕様記述の手順

3.1 仕様記述手順の概要

実際の仕様記述過程を整理し、それぞれの検討段階での成果物の影響を考慮してデータ・フロー化したのが図１である（二重線枠がプロセス、単一線枠がデータを表す）。実際には筆者１名で記述実験を行ったため、並列に検討可能なものも実際には並列的に検討しているわけではなく、検討が必要な時点でシーケンスに検討を進めていることに注意されたい。

図中において、プロトタイプ的実行の準備及び式の記述の部分はもちろん代数的仕様記述に依存している部分であるが、ソート定義及び制約条件判定関数を記述する部分以外は必ずしも代数的仕様記述に依存しているわけではない。オブジェクト指向型、関数型、データ・フロー等による仕様記述でも検討しなければならないプロセスである（図１において、点線枠で囲った範囲以外の部分）。

3.2 手法独立部分の検討プロセス

図１の中から手法独立の部分の検討プロセスについて論述する。

(1) 全体系を表現する変数の仮定

全体系を表現する変数の名前付けは、データベース検索システムにおけるデータベースそのものを、通信システムにおける系の状態の集合をなんと呼ぶかということに相当する。

実世界が変化するものであるとしたら、記述しようとする実世界をなんと呼ぶかは仕様記述でなくとも実世界が有限であるか否かに係わらず本質的に必要なことである。

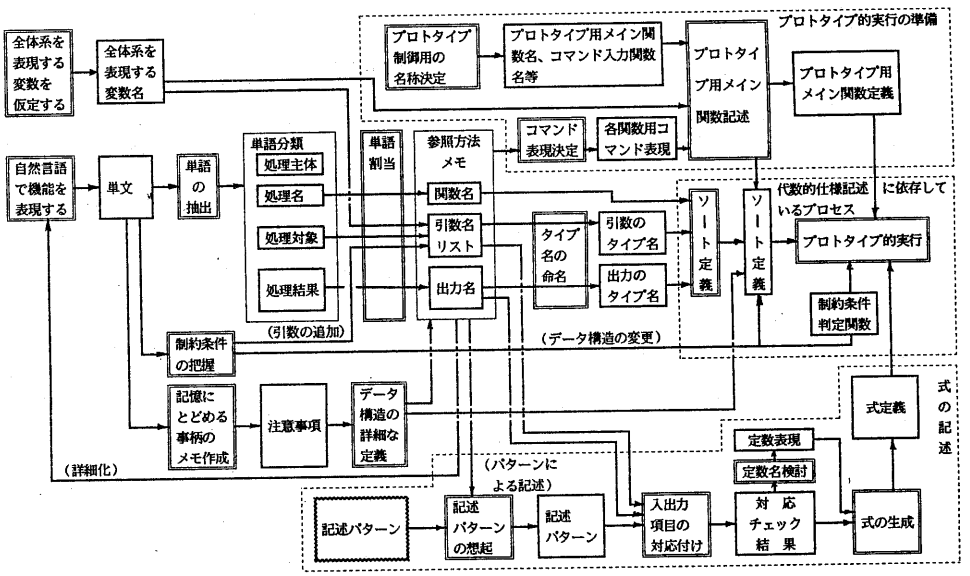

図１　代数的仕様記述による仕様記述の手順

(2) 自然言語による機能の表現

冒頭に述べたように人間の思い付きは、最初自然言語に表現されて初めて自ら意識し始めるものとなりうる。図表で表現するにしてもその初期段階は脳裏に浮かべる「どんな図表にしようか」という自然言語表現である。

自然言語でなんらかのメモ書きをしておくことが問題を整理する出発点となる。この際、各機能を箇条書で単文で表現することを筆者は薦める。というのは、単文の羅列のほうが問題をフラットに眺めることができる上に、単純な思考で問題を整理しやすいからである。

頭脳明晰な技術者は、この段階を飛び越して次の「単語の抽出」、あるいはその次「参照方法のメモ作成」に至ることができようが、後で述べる制約条件や記憶すべき事柄を並列に覚えておくことは、通常の技術者にとって検討漏れを引き起こすことにつながる。従って、この段階で少なくともメモ書きを行っておくほうが安全である。

(3) 単語の抽出

自然言語による表現中から仕様記述の要素となるべき単語を抽出する。抽出方法として直感や常識に頼る方法もあるが、なるべく直感や常識に依存する部分を少なくするほうが、経験が少ない技術者にとって有益であると考える。

なるべく直感や常識に依存しないようにするとは、すなわち形式的に処理することである。自然言語における形式的な単語の抽出において役立つのは、自然言語における文法が最も単純である。

そこで筆者は図書館問題において整理した経験から、構文に着目した単語の抽出法を提案したい。提案する単語の抽出方法とは、自然言語から主語、述語、目的語に分離し、それぞれの構文的役割に沿って、処理主体、処理名、処理対象及び処理結果に分類する方法である。自然言語における構文要素との対応は次のようになる。

(a) 主語　：処理主体

(b) 述語　：処理名

(c) 目的語またはその他の修飾句：

処理対象または処理結果

この分類において直感や常識に依存する部分は、次のとおりである。

- 抽出後の単語表現にどの程度まで修飾的な単語を付けるかという判断及び同意義語についての知識。
- 用言の活用を名詞化する際の処置の文法的な知識。
- 目的語を更に分類して処理対象と処理結果に分ける際、述語はその目的語を結果として期待しているのか、それとも目的語を処理対象として見なすべきかの判断。例えば、「Aを表示する」という表現は、Aを結果として期待しており、「Bを貸し出す」という表現は、Bを処理対象としてみなしている。この際、単語自身が持つ意味の取扱が必要である。

以上の依存度合が仕様記述の経験が少ない技術者にとってどの程度の負担があるかどうかについては、今後の検討課題であるが、少なくとも仕様記述に用いられる用語の抽出を全て直感的に行うよりは単純であると考える。

更に上記部分は、原理的には標準化、機械化が可能となる部分が多いので、今後の自然言語処理や用語辞書の拡充におおいに期待できる部分でもある。

(4) 単語の割当

上記(3)で割り当てられた単語を、記述しようとする仕様記述法の概念に振り分ける。表1にその対応関係を示す。

代数的仕様記述の場合、処理名を関数名とし、処理対象の名称を引数リストにおける変数名、処理結果の名称を出力の代表名とする他、全体系を表現する変数名を引数リストの一部に加える。代数的仕様記述の表現の中に、出力名は現れないが、メモとして記入しておくことが後々の理解に役立つ。

この部分は、極めて単純な作業であるが、処理対象が複数ある場合に引数リストをどう順序付けるかは人的判断である。

表1　本手法と各種仕様記述との関係

構文要素	本手法	データフロー	オブジェクト指向	代数的または関数的仕様記述
主語	主体者	プロセス	インスタンス	仕様名
述語	行動または処理名	プロセス	メソッド	関数名
目的語	客体	データ	メッセージ及び関数値	変数
	処理対象	入力データ	パラメータ	引数の変数名
	処理結果	出力データ	返却値	関数結果名

(5) タイプ名の命名

変数名をインスタンス名、タイプ名をクラス名に置換するとオブジェクト指向におけるクラス名の命名の作業に類似している。この作業は極めて直感や常識に依存しており、人的な判断を必要としている部分である。

上記(4)で割り当てられた個々の引数名や出力名で示される「もの」を総称してどう呼ぶか命名する際の基準となる。

(6) 制約条件の把握

自然言語表現の中には機能自身を表現するというよりも系自体の制限や約束ごとを表現する場合がある。このような表現を制約条件として把握し、最初に記述したデータ（引数や出力）や処理を追加する必要が出てくる。

制約条件の把握の結果を反映するのは、全ての機能表現を記述した後、全体的に見回してどう対処するかを検討するほうがよい。なぜなら、自然言語による最小限の機能表現から直接引き出した最小限のデータ項目に対してデータ項目を追加及び処理の追加を行う方向で変更するほうが単純であること、及び検討上のバックトラックが起きるのは通常制約条件の把握の部分であるためこの部分を決まりきった時期に行うことによって本来バックトラックしなければならない時点を決めつけることができるからである。

機能表現と制約条件を同時に検討すると、制約条件のためだけに必要なデータ項目と機能を実現するために本質的に必要なデータ項目との区別がつかず、既出の項目が本当に必要であるかどうか、本来単に処理を追加するだけでいいのかの判断が困難となる。

逆に言えば、機能実現に本質的に必要なデータ項目が揃った後に制約条件の把握を行うため、制約条件によるデータ項目数は単調増加となることとなる。例えば特別な引数項目の追加（当然ソート定義のタイプ名を含む）、あるいは制約条件の判定を行う関数の追加等である。

更に、この部分に関しては「如何にデータ項目を追加するか」、あるいはそれを「どう判定するか」等についての人的判断が必要である。この種の判断ではある程度の「全体を見渡す能力」も影響してくるため、多少の訓練が必要である。

(7) 記憶に留める事柄のメモの作成

自然言語表現から単語を抽出する際（上記(3)のプロセス）、注意しておくべき事柄のメモを残すことがデータ構造の詳細な定義を決定するときに役立つ。例えば図書館データベースでの自然言語での機能表現

・「特定の貸出者名によって貸し出された書籍リストを検出する」

から、

・処理名：検出

・処理対象：特定の貸出者名

・処理結果：貸し出された書籍リスト

を得た際、データベースには現在の「貸出者名」によって検索できなければならない。すなわち、

・「データベースの構成要素に貸出者名が必要

である」

ということを注意事項としてメモとして残しておくことが記憶上役にたつ。

(8) データ構造の詳細な定義

上記(7)で残されたメモからデータ構造を決定する。今回は代数的仕様記述で行ったので階層構造（part-ofの関係）のみが記述対象となったが、その他の仕様記述法ではもっと複雑となろう。この部分の仕様記述での検討過程についてはデータ構造を中心とした仕様記述法による記述実験を行う必要があろう。

しかし、代数的仕様記述法で可能な階層構造の範囲に限定して仕様記述して確認しておくことはできる。

3.3 手法依存の検討プロセス

代数的仕様記述法に依存した検討プロセスは大きく分けて次の４種類である。

・プロトタイプ的実行のための関数の準備
・式の記述
・ソート定義
・制約条件による判定関数の定義

(1) プロトタイプ的実行のための関数の準備

この作業は、代数的仕様記述を書換え規則によって実行させ、仕様が正しいかどうかを確認するために、いわゆるテスト用の主制御関数を用意するものである。この主制御関数の形式は図２に示すように定型パターンである。

図２中の「 _系」という変数は、全体系を表現する変数として決定されたものを使用し、その他はプロトタイプ的実行用に独自に決定することが必要となる。いわゆるマクロを用意することによってこの作業は単純となる。

但し、代数的仕様記述において引数の型を一つの型にしておくほうが取扱い易い。困ったことにプロトタイプ的実行は複数の機能の出力結果であるから当然のことながらデータ構造は異なることになるのが一般的である。このような場合には結果自体をコントラクタとみなして型を表現するものを導入すると記述しやすい。今回実験では

sort 新ＤＢ:書籍ＤＢ　->結果;
sort 書籍表:書籍リスト->結果;
sort 借用者:氏名　　->結果;

という３種類のコントラクタを導入した。記述上の知恵と記憶しておきたい事柄である。

```
主制御(_系) =判定(ｺﾏﾝﾄﾞ入力表現,_系);

判定(ｺﾏﾝﾄﾞ1,_系)=処理１表現;
判定(ｺﾏﾝﾄﾞ2,_系)=処理２表現;
判定(ｺﾏﾝﾄﾞ3,_系)=処理３表現;
・
・
```

図２　プロトタイプ的実行のための関数のパターン

(2) 式の定義

式の定義では、記述経験が大きな役割をもつ。なぜなら、代数的仕様記述における個々の処理の記述は定型的な表現になることが多く、その定型的な表現をどれだけ想起できるかが効率よく記述するための鍵となる。

図１に示したパターンによる記述部分は、筆者が代数的仕様で記述する際の手順を整理したものである。客観的かどうかについては、他者の思考手順を観察する必要があるが、この手順はほとんど単純作業となる。

必要とされる支援ツールは単なるマクロ展開ではない（特に入出力項目の対応付け）が、何らかの支援が期待できる部分でもある。

(3) ソート定義

前述した「3.2 手法独立の部分の検討 (5) タイプ名の命名」が終了すれば、単純な作業である。参照方法のメモと各変数タイプとの対応表によって１対１で変換できる。

(4) 制約条件による判定関数の定義

前出の「(2) 式の定義」とほぼ同一であるが、より多くの記述パターンを用意する必要があろう。今回の記述実験で必要になったのは、貸出冊数のカウントのための処理であるが、今後の記述実験を続けながら必要なパターンを蓄積する必要がある。

3.4 記述の詳細化

個々の詳細な記述に先立ち、上位記述者による関数の仕様を自然言語で記述することによって詳細化が進められる。もちろん関数自体の処理が単純な場合は、前述した定型パターンによる式の定義を行う。

本実験では問題が小さいため、更に自然言語で記述し段階的詳細化を行う場面はなかったが、次のような点に注意して下位の記述者に記述依頼を行うべきであろう。

(1) 詳細化に先立つ自然言語での記述は分担せず、上位記述者が全て行い、下位の記述者に手渡すべきである。

(2) 上位記述者は、以下の資料を下位記述者に渡すべきである。

(a) 自然言語で記述された担当関数の仕様

(b) 上位の関係するインターフェース

・関数のソート定義

・引数のコンストラクタのソート定義

(c) プロトタイプ的実行の結果。これはインターフェースの例示として下位記述者のインターフェースの理解に役立つ。

4. バックトラック

図1に示す検討過程で記述を変更するタイミングは次の2ヶ所である。

・制約条件の把握

・プロトタイプ的実行

但し、本検討プロセスの多くは単純なものが多いため後者の多くは名前の記述ミスや文法上のミス等がほとんどである。

問題は前者の「制約条件の把握」の際のバックトラックである。この部分をまとめて処理することにより制約条件を効果的に反映することが出来ることは既に述べた。

制約条件の把握により以下の処置のいずれかが必要である。

(1) 制約条件を判定するためのデータと判定用の関数の用意

・引数の追加、または予定したソート定義の変更

・制約条件の判定関数の定義

(2) 各関数記述の際の考慮

・自然言語による機能表現に対して注意事項を追加する。

5. 知的支援ツール

仕様記述の際、どの部分を支援すれば「知的」なツールとなりうるかを結論的に以下に列挙する。

(1) 記述手順のガイダンス

慣れていない記述者に無駄な思考をやらせないような記述手順をガイドする仕組み。特に、

・本来、思考上のバックトラックが必要となる段階では、記述の見直しを記述者に促す。

・他の担当者に分割して記述を任せることができる段階では、引継のために必要な整理内容

を示す。

等のガイドは有用であると考えられる。

(2) 自然言語の理解

自然言語の理解のモデルとして数々の提案がなされているが、一般的なモデルとして提案されているため利用可能な規模・性能を持った処理系の実現は困難である。一方、代数的仕様記述の際に最低限必要な自然言語理解は、

・処理される対象物を表現する単語（通常、目的語）

・処理を表現する単語（通常、述語）

・処理結果を表現する単語（通常、目的語）

・制約条件の判定条件（通常、但し書）

の分離で充分である。代数的仕様記述に必要な範囲に限った自然言語解析の仕組みはより簡便で、実用に耐えるシステムの実現性は高い。

(3) 関数の詳細化の際の自然言語記述パターンの候補選択

処理を表現する単語（述語）に対応する詳細な自然言語記述の過去の例を並べてより的確な文章パターンを見つけるための仕組みも有用である。述語をキーとした検索システム程度でも充分利用効果が期待できるが、更に現在表現しようとしている問題の分野名、他の述語との関連で候補が絞れるような仕組みもより有用であろう。

(4) 記述パターンによる代数的仕様記述の生成

既存の言語のアナロジーからはマクロのような機構である。但し、表１の13、14、15 の段階での結果欄に示すように、記述パターンはある程度あいまいな表現としておき、記述しようとしている述語に対応する入力項目と出力項目の対応関係から生成できることが望ましい。

６．終わりに

本来仕様記述法とはある視点から限られた範囲を要領よく表現するための「言語」であり、従って万能ではない。従って、複数の仕様記述法を用いて複数の視点から問題を整理することが仕様記述に必要になってくる。

一方、多くの仕様記述法では同一の言葉を異なる意味に用いてたり、その言葉が示す概念の範囲が異なっている場合がある。更に同一の問題を複数の仕様記述法で記述する際、仕様記述間の用語の意味が異なっていることが、それぞれの仕様記述法を現場に導入する際の障害となるであろう。これを避けるためには、複数の仕様記述法で異なっている言葉を整理し、標準化する必要がある。

代数的仕様における検討過程を整理してきたが、検討過程は必ずしも代数的仕様記述に特有の部分だけでなく、同じ様な整理が他の仕様記述法にも必要である可能性があるとの確信を得た。そこで表１に示す対応表を作成したわけであるが、より多くの仕様記述法での共通の検討プロセスを収集する必要があると考える。この際も言葉の標準化の役割は大きい。言葉の標準かは、なるべく少ない常識に沿って仕様記述法での記述過程を説明するための基本となる。

協同システム開発（株）でＩＰＡ関連事業の一つとして進められている「ソフトウェア環境統合化技術開発計画」での支援ツールのひとつである ASPELA の言語仕様で記述実験を行った。実験作業は情報処理振興事業協会技術センターの「ソフトウェア・プロトタイピング技術の調査研究」プロジェクトの一環として行ったものである。両プロジェクトへ支援を頂いている通産省、情報処理振興事業協会、及び技術センター・プロジェクトで数々の議論をさせて頂いているワーキング委員会の委員、オブザーバの方々に感謝する。

[参考資料]

1)野村浩郷：自然言語理解の構造－理解の表現，情報処理，Vol.30，No.10，pp.1161-1168（1989）

2)白井豊、別府真、小松雅彦：仕様記述のためのプロセス・モデル，人工知能学会第１回知的ソフトウェア開発ワークショップ，1989.

3)稲垣康善、直井徹：抽象データタイプの代数的仕様記述法の基礎，情報処理，Vol.25，pp.47-53,491-501,708-716,971-986(1984)

4)別府真、小松雅彦：代数的仕様記述言語 ASPELA の概要，ソフトウェア環境統合化技術開発計画テクニカル・レポートＮｏ．２，pp5-11（平成元年１月）

5)Pott,Colin,et al： Structured Common Sence , A Requiements Elictation and Formalization Methd for Modal Action Logic. Alvey FOREST Deliverable Report 2. Chelmsford : General Electric Co. Research Laboratories, 1986.

6)松浦佐江子、大林正晴：概念による設計法を基礎とした設計支援ツールの基本概念，ソフトウェア環境統合化技術開発計画テクニ カル・レポートＮｏ．２，pp23-28（平成元年１月）