情 報 学 基 礎 20－5
（1990. 12. 18）

アプリケーションから見たオブジェクト指向
- Smalltalk-80 を中心として -

梅田靖、冨山哲男、吉川弘之
東京大学 工学部

Smalltalk-80 のアプリケーション例として知識工学的手法を用いた、故障診断システム、設計知識表現言語 IDDL、および、定性推論システムについて述べる。オブジェクト指向パラダイムに対して、プログラミング言語、および、知識表現手法という二つの側面から考察を行なう。プログラミング言語としては、ラピッドプロトタイピング、グラフカルユーザインタフェースの構築のために優れているが、情報隠蔽の原則に反したプログラミングが可能であり、また、グループでのシステム構築が必ずしも容易でないという問題点を指摘する。知識表現手法として、実体の表現は容易であるが、実体間の関係やメッセージパッシングで表現できない振舞いの記述に適さないことを述べる。

Object Oriented Programming Paradigm for Application Program Development
- A Case Study of Smalltalk-80 -

Yasushi Umeda, Tetsuo Tomiyama, and Hiroyuki Yoshikawa
Faculty of Engineering, University of Tokyo

This paper describes knowledge based application systems implemented in Smalltalk-80 for fault diagnosis, representing design knowledge, and qualitative reasoning. The object oriented programming paradigm is examined from the viewpoints of programming language and knowledge representation. Smalltalk-80 is appropriate for rapid prototyping and building graphical user interfaces. However, it has the following drawbacks and problems. Writing codes that violate the principle of information hiding is possible, and developing a system by a team is not always easy. Furthermore, message passing does not represent static relationships between objects properly.

1 はじめに

本研究室は、人間の知的作業である「設計行為」、「保全作業」を明らかにし、かつ、知的に人間を支援、自動化することを研究テーマとしている。具体的には、主に知識工学的手法を用いて設計行為、保全作業、および、設計、保全の対象である機械のモデル化と計算機上での表現、推論手法について研究を行っている。このような研究テーマの下で、手法の有効性検証のためにSmalltalk-80上でプロトタイプ・システムを構築している。これは、オブジェクト指向言語としてのSmalltalk-80[1]がラピッドプロトタイピングおよび知識の記述に適していると思われるからである。

本研究室では研究室員（学生、職員等）25名に対して、2台のSun3/60、5台のSun3/50、2台のSun SPARC1+、1台のMacintoshIIx、および、1台のMacintosh IIfx上でSmalltalk-80が利用可能である。研究室員中16名が主にSmalltalk-80を利用しており、そのほかLisp、数値計算用にC等が利用されている。

本稿では、本研究室で開発した3つのプロトタイプ・システム、すなわち、故障診断システム、設計知識表現言語IDDL、定性推論システムを簡単に紹介し、プログラミング言語、および、知識表現という二つの側面からオブジェクト指向パラダイムについて考察を行なう。

2 故障診断システムの開発

近年、従来の対象固有のヒューリスティックス（発見的仮説）に基づく故障診断エキスパートシステムの問題点が明らかになり、対象の構造、特性を明示的に表わすモデルに基づく故障診断システムの研究が盛んに行われている[2]。ここでは、従来のエキスパートシステムに欠けている知識の汎用性、推論の強靱さ、知識獲得の容易性を得ることを目的とした故障診断支援システムについて述べる[3, 4]。

まず、対象表現について述べる。診断対象の構造は部品（ギア、モータ等）と部品間の関係（機械的接続関係、電気的接続関係等）用いてネットワークを構成することにより表現する。各部品、関係はオブジェクトであり、その物理的な性質が記述されている。Smalltalk-80の階層構造を利用することにより、各部品、関係は階層的に整理して記述されている。さらに、特に機械システムの場合、故障により対象の構造が変化することが有り得る。そこで、1つ以上の部品や関係上で起こり得る「故障現象」を収集し、オブジェクトとして整理して記述している。これらの故障現象はある条件でインスタンシエイトされ、対象構造を変更する。これらを用いて診断対象を記述したものを対象モデルと呼ぶ。

本システムの特徴はある故障が発生したときの状態を定性物理[5]の手法に基づきシミュレーションすることにある（これを故障シミュレーションと呼ぶ）。これにより、対象モデル上の接続関係、因果関係により導出された故障の候補がそれぞれ発生した状態を表わす対象の定性的なモデルが物理法則に基づき作られる。このモデルを比較することにより、その故障候補が入力された故障の症状を引き起こすかを検証することができ、さらに、故障を絞り込むために次に点検すべき箇所や同時に発生している可能性のある故障、および、その故障が発生した過程を提示することができる。

3 オブジェクト指向と論理指向の結合による設計知識表現言語IDDLの開発

計算機技術の進歩にともなって、CAD(Computer Aided Design)システムの開発が増加している。しかし、現在のCADシステムは主に設計対象の表現に注目しているが、より知的なCADを構築するためには、設計過程の表現も重要である。

IDDL(Integrated Data Description Language)[6]はSmalltalk-80の上にインプリメントされ、Smalltalk-80のオブジェクト指向の特徴を利用しながら、論理指向のプロダクション・システムを作成し、設計対象と設計過程の記述の結合を実現した言語体系である。また、IDDLはICAD(Intelligent CAD)を構築するための言語である。

IDDLにおける設計対象はオブジェクト指向のオブジェクト(object)とオブジェクト間の関係を記述する一階述語の事実(fact)で表現される。オブジェクトには内部状態を表わす属性(attribute)があり、属性は関数(function)より操作される。オブジェクトと事実の集合はワールド(world)と呼ばれ、IDDL

では複数の独立なワールドが同時に存在することが可能である。

IDDL における設計過程はシナリオ (scenario) と関数 (function) の集合で表現される。シナリオは主にプロダクション・ルールの集合からなり、宣言的な知識を表現する。関数は属性間の数値関係を記述し、手続き的な知識を表現する。IDDL 上では、設計はいくつかのシナリオと関数の実行として実現され、設計解は設計対象のオブジェクトと事実として表現される。

IDDL のプログラミング環境は IDDL Browser という（図 1）。このブラウザーの左側においては IDDL のシナリオと関数の作成（設計過程のプロトタイピング）、Smalltalk-80 コードへの変換及びシナリオの実行が可能であり、右側に実行結果のオブジェクトと事実を表示する。

IDDL 言語の拡張として、C 言語で開発された B-reps のソリッド・モデルとの結合による形状フィーチャー抽出システム、IDDL の知識を利用するためのインタフェース・システムなども開発した。

以上のように、IDDL を Smalltalk-80 を用いて構築することにより、次のような特徴が得られた。まず、IDDL 上の設計対象はオブジェクト指向パラダイムで記述される。このため、IDDL のオブジェクト、タイプ、属性、関数はそれぞれ Smalltalk-80 のインスタンス、クラス、インスタンス変数、メソッドを用いて容易に実現することができた。さらに、設計過程のシナリオと関数はカテゴリ (category) によって管理される。IDDL のカテゴリは Smalltalk-80 のクラスに相当し、シナリオと関数は Smalltalk-80 のインスタンス・メソッドとして実現されている。従って、設計知識に階層的な構造を持たせることができ、設計知識の管理、利用が容易に行える。

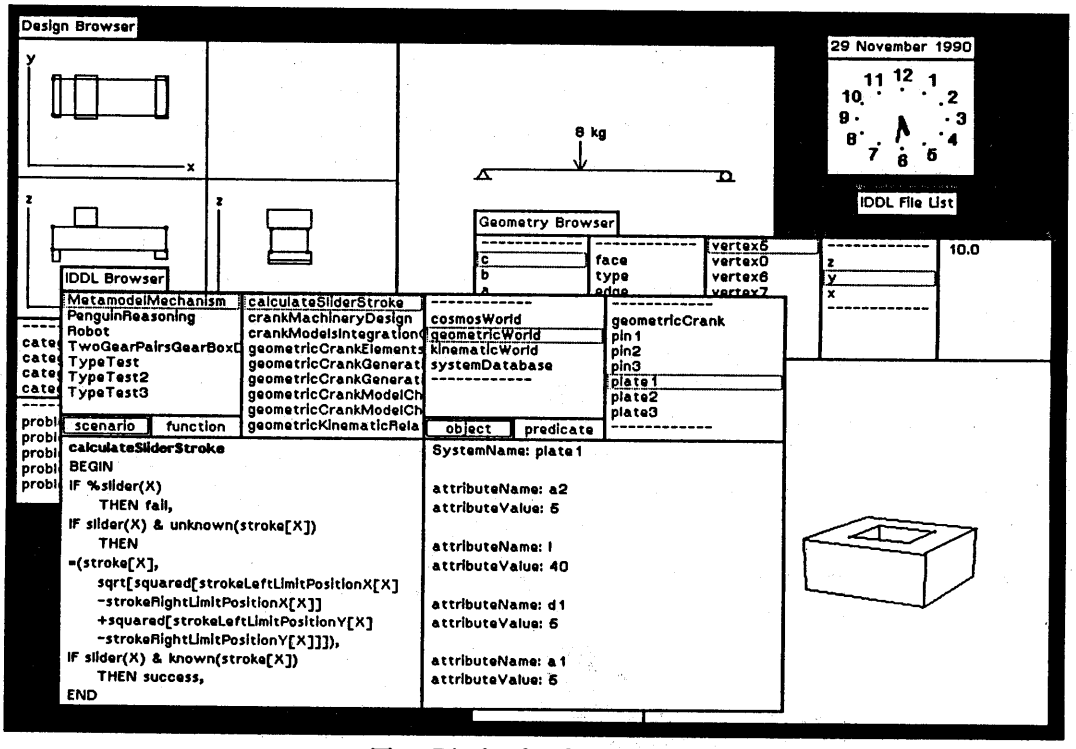

図 1: Display hardcopy of IDDL

4 定性推論システムの開発

4.1 概念設計と定性推論

機械の概念設計においては、設計対象の基本的な挙動と構造が決定される。この段階を CAD で支援するためには、設計対象の構造から予測される挙動を導出する推論が必要になる。このため我々は、定

性推論の手法に基づき設計対象の定性的な挙動を推論するシステムを開発した [7]。

このシステムでは、設計対象の構造に物理法則を適用して生起可能な物理現象を導出する。設計対象の構造は軸、ばね、コイルなどの要素と、接触、支持などの要素間の関係を記号的に表現することで与えられる。物理法則は、例えば力が働いているとき物体が加速される等の、物理現象とそれが起きるための前提条件の組み合わせで表現される。定性推論を行うと、設計対象がある初期条件から出発してどのような定性的状態を経過するかという状態変化が導出される。一般に定性推論では、与えられた初期状態から生起可能な全ての状態が導かれるため、状態の連鎖には分岐が生じることがある。この場合定性推論によって、意図する設計対象の状態変化が起こるための条件が導かれる。

4.2 定性推論システム

本システムは定性プロセス理論 [5] に基づいており、設計対象の状態の管理は ATMS (Assumption-based Truth Maintenance System) [8, 9] を使用している。定性プロセス理論は、対象とする系に存在する物体を個体、個体の持つパラメータとそれらの間の拘束関係を個体ビュウ、及び物理現象に関して生起するための前提条件と系に及ぼす影響をプロセスで表現する。系の状態変化は、個体ビュウ、プロセスの生起・消滅と、パラメータの増減で表される。

ATMS は事実（ノード）と、ノードが成立するための前提となる仮説ノードとの依存関係を管理する機構である。仮説ノードの集合で環境を定めると、その環境で成立するノードの集合を ATMS から取り出すことができる。本システムでは、パラメータの値とパラメータ間の大小関係を仮説ノードとすることにより、系の状態と ATMS 内の環境を対応付けている。設計対象の構造と考慮するべき物理現象の集合を定めて定性推論を行うと、可能な全ての物理現象とそれを支持する仮定の組み合わせが ATMS 内に作られる。その上で、ある定性的状態を初期状態として選ぶと、その状態から遷移しうる定性的状態をシステムが推論する。

4.3 システムの動作

本システムで、図 2の 2 極モータの機械的な挙動に関する定性推論を行ってみる。このモータは回転軸、軸に固定される一対のコイル、左右の永久磁石、2 個の整流子からなる。整流子がモータの挙動推論に組み込まれる場合、挙動記述のパラメータである接点の回転角はモータのコイルの回転角に、N や S の磁極というプロセスはコイルの磁極の条件に対応づけられる。ユーザが初期状態を与えると、システムはモータの状態の時間的な変化を図 3に示すグラフとして表示する。同図で、このモータは静止状態 1 から出発して、2 から 5 の定性的状態を順に取りながら回転することがわかる。また状態 6 はモータの死点であり、この状態からは回転しないことがわかる。

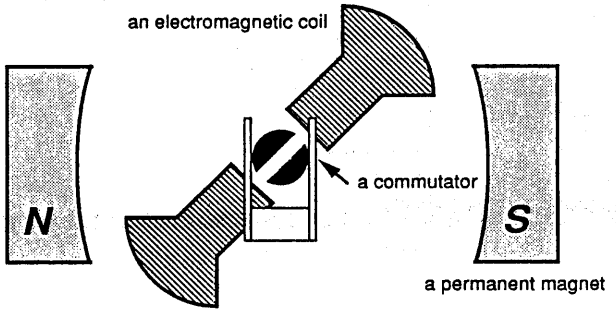

図 2: Structure of a motor

5 考察

以上、Smalltalk-80 を利用した 3 つのアプリケーションを述べた。本節では、この経験をもとに、プログラミング言語、および、知識表現手法という二つの側面からオブジェクト指向パラダイムを考察す

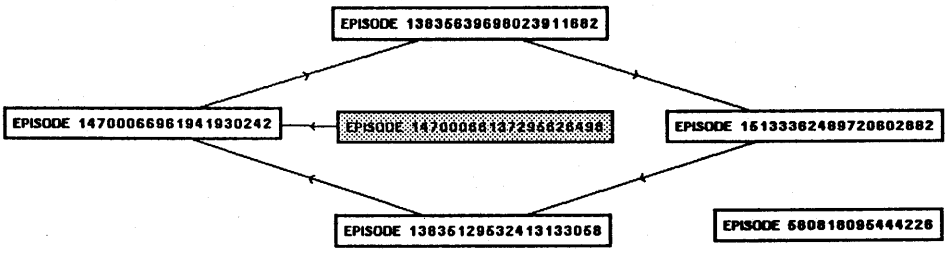

図 3: State transition of the motor

る。

5.1 オブジェクト指向言語 Smalltalk-80 の特徴

1. 長所

ラピッドプロトタイピング システムの基本構造さえ決めておけば、システムの構築は他言語に比較して容易である。これは、Smalltalk-80 のデバッガが強力であることと、差分プログラミング、各機能ごとにオブジェクトを作って情報隠蔽することができるためである。ただし、Smalltalk-80 上では場当り的にシステムを構築することも可能であるが、クラスの再利用が困難となり、また、システムが大規模になるとその内容を理解することが困難になる。そのため、あらかじめシステムの基本構成を決めておくこと、および、クラスを再利用することを常に意識しながら汎用的にクラスを作ることが重要である。

グラフィカルなユーザインタフェース Smalltalk-80 上でグラフィカルでインタラクティブなユーザインタフェースを構築することは、MVC と dependency を理解しなければいけないという点で間口が狭いが、慣れてしまえば簡単である。特に、表示されるオブジェクト自身に表示に関する情報を持たせることによりシステムが美しくなる。

言語としての優位性 プログラムはユーザが必要としてる動作、処理を得るためのものである。このとき、ユーザが思い浮かべている必要な動作、機能のひとかたまりを単位としてオブジェクトとして表現して行くというオブジェクト指向のプログラミングスタイルは人間の直感に適合している。

2. 問題点

hierarchy の構築 Smalltalk-80 では、オブジェクトの階層を構築するために inheritance メカニズムを採用している。これは、アプリケーション構築の概念設計段階をシステム上で行なうことの障害となっている。すなわち、inheritance においては、一般的に、superclass はいくつかの subclass の共通概念を示すものである (いわゆる is-a 関係)。一方で、実際のインプリメンテーションでは上位クラスから下位クラスへとクラスを作成していく。このため、システム構築の前にそのアプリケーションに必要な概念を整理し、かつ、階層的に整理するという作業が必要になる。これに対し、delegation[10] を用いると、概念を整理し、概念間の階層を構築する過程も、システム上で支援できる可能性がある [11]。

内部構造を仮定したインプリメンテーション Smalltalk-80 では、情報隠蔽の考え方に基づき各オブジェクトの内部構造を見せないインプリメンテーションを基本としている。しかし、実際には内部構造をオブジェクト外部から操作するプログラムも記述可能である。例えば、図 4 に示すロボットにおいて、アーム arm_1、arm_2 はクラス Arm のインスタンスであり、インスタンス変数として endPoint1、endPoint2 を持つとする (図 5)。また、クラス Arm にはインスタンス変数 endPoint1、endPoint2 の値を返すインスタンスメソッド endPoint1、end-

Point2が記述されているとする。今、アーム arm_1 に注目し、次のようなコードを記述することが可能である。

```
| point |
point ← arm1 endPoint1.
point x: 5 y:10.
```

この場合、一時変数 point には、座標を表すオブジェクト p_{21} が代入され、その point に対し、座標値の設定を行なっているので、結局 arm_1 の端点の座標が変更される、すなわち、アームを動かすことができてしまう。しかし、このコードは情報隠蔽の原則からはみ出している。情報隠蔽の原則からすると以下のようなコードを記述するのが妥当である。

```
arm1 changeEndPoint1: 5 @ 10.
```

前者のような記述を行なうとオブジェクトの独立性を低下させ、結果としてオブジェクトの再利用性、可換性を失わせてしまう。これは、プログラミングスタイルの問題ではあるが、守らなければならない問題である [12]。

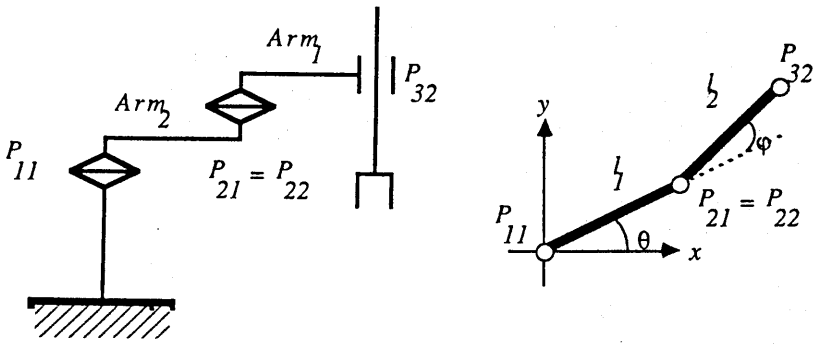

図 4: A robot

class Arm	arm1	arm2
Instance Variables		
endPoint1	P11 (0,0)	P22 (10,5)
endPoint2	P21(10,5)	P32(18,9)
length	l1	l2
relativeAngle	θ	φ

図 5: Description of arms

オブジェクトの機能の暗黙性 Smalltalk-80 におけるクラス名、メッセージ名は、各クラス、メソッドの機能を直感的に理解する助けになる。しかし、メソッドの中身を注意深く読むとそれらに対する作者の意図とも言うべき裏に隠された限定（もしくは、その逆に融通性）が浮かび上がることがある。この直感的な理解と裏に隠された限定の食い違いがバグを引き起こすことが多々ある。例えば、Association の "=" の意味は key の " =" でしかないことに気づくことは容易ではない。また、変数に代入されるべきものは型の定義が無いために、その変数に何が入るべきであるのかが解りにくい。例えば、オブジェクトのコレクションを代入する場合、Set でなければいけないのかそれとも OrderedCollection でも構わないか等判断に迷うことが多々ある。このため、他人が作成したオブジェクトを利用すると思わぬバグを引き起こしてしまう。この問題は、各クラス、メソッドの機能や振舞いを各ユーザがそのコードやオブジェクトの構造から推察しているために誤解が生じることに起因している。これを避けるためには、各オブジェクト、メソッドの機能を何らかの方法で明示的に記述する必要がある。

グループでのシステム構築の問題点 本研究室では1つのテーマを数人のグループで研究しているので、似たような内容で少しづつ異なったシステムを構築することが多い。この場合、Smalltalk-80の差分プログラムの特徴を利用してオブジェクトを互いに交換することで生産性が高まるはずであるが、現実的にはそれほど上手くいかない。ここには3つの原因が考えられる。第一に、メッセージパッシングにより実現されるプログラムはアルゴリズミックなプログラムよりもプログラムの流れを掴みにくいこと。第二に、前項で挙げたオブジェクトの機能を明示していないこと。そして、第三に、各オブジェクトがいつ、誰が、どのアプリケーションのために変更したかという情報を管理する機能がほとんど無いことである。

5.2 知識表現手法としてのオブジェクト指向

ここでは、オブジェクト指向によって知識を表現する場合に、適当であるもの、あまり適当では無いと考えられるもの、および、明らかに適しないものが何であるかということ考察する。

1. 表現容易な知識
   CADや故障診断システムでは、対象システムを表すモデルを計算機上に表現する必要がある。このとき、実体や実体の性質を表現することにオブジェクト指向は適している。メッセージパッシングによるモデル化が適当な実体の振舞いを表現することにも適している。

2. オブジェクト指向による表現が最適とは言えない知識
   静的な実体間の関係、例えば、図4において arm_1 と arm_2 が接続していることを表現するためには、抽象的な概念を表すオブジェクトconnectを作り、両者を結び付けるという表現を取らなければならない。このような表現方法にはオブジェクト指向による特徴が充分生かされているとは言えず、述語論理を用いて実体間の関係を表現する方がより優れていると考えられる。この他、プロダクションルールを用いて表されるようなヒューリスティックな知識や3で述べたIDDLにおける設計過程を記述する知識もオブジェクト指向による表現が必ずしも最適であるとは言えない[12]。

3. パラダイムとして表現が不適当な知識
   例えば、機械の動作をシミュレーションする場合、各部品を表すオブジェクト間のメッセージ・パッシングにより状態の時間遷移をシミュレートする事も可能である。しかし、回路の発振をシミュレートすると無限ループに陥る。このとき、「同じ動作を繰り返す振舞いは発振である」という知識を用いて、この回路は発振するというシミュレーション結果を得たい。しかし、このようにオブジェクトとメッセージパッシングというレベルに対してメタなレベルの知識を表現するということは既にオブジェクト指向パラダイムからはみ出すことになる。その他、例えば、図6に示すように3個の剛体球が並んでおり、両側から等しい力で押された場合力が釣り合い、3個の球は動かないということをシミュレーションするとする。この場合も、3個の球を合わせた系と外力との関係を解かなければならないので、同様の問題が生じる[13]。

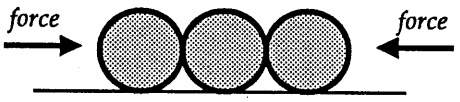

図 6: Three ball system

知識表現の問題点として、最後にもう一点以下の問題点を指摘することができる。すなわち、概念設計段階においては、一つの実体に対し様々な見方をする。例えば、一本のロボットアームを形状的側面から見ると「四角柱」と扱い、材料力学的な側面からは「弾性体」として扱い、さらに、運動学的立場からは「剛体」として扱う。この場合、「弾性体」のアームと扱う時は「剛体」その他の見方は必ずしも必要でない。このように、一つの実体に対していくつかの見方を設定し、それぞれの見方によってその

性質が異なるように扱いたい場合、知識の整理方法としてinheritance的な階層化よりも、delegationを用いた動的な変更を許した階層化の方が適している[14]。

6 おわりに

本稿では、オブジェクト指向言語Smalltalk-80上で構築したアプリケーションを3件紹介し、アプリケーション構築の経験に基づきプログラミング言語、および、知識表現手法の二つの側面からオブジェクト指向パラダイムを考察した。

なお、本稿をまとめるにあたり、桐山孝司、薛徳意、小池雄一、中田秀基各氏の協力を得たことを記し、ここに謝意を表す。

参考文献

[1] A. Goldberg and D. Robson. *Smalltalk-80:The language and its implementation.* Addison-Wesley, 1983.

[2] C. Price et al. Applications of deep knowledge. *Artificial Intelligence in Engineering*, Vol. 3, No. 1, pp. 12–17, 1988.

[3] 梅田他. 対象モデルに基づく定性物理を用いた故障診断. 1989年度人工知能学会全国大会（第3回）論文集, pp. 263–266, 1989.

[4] 小池他. 故障発生メカニズムの定性物理による表現. 1990年度精密工学会春季大会学術講演会講演論文集, pp. 679–680, 1990.

[5] K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, Vol. 24, No. 1-3, pp. 85–168, 1984.

[6] T. Tomiyama et al. An experience of developing a design knowledge representation language. In P. J. W. ten Hagen and P. J. Veerkamp, editors, *Intelligent CAD System III: Practical Experience and Evaluation.* Springer-Verlag, 1989.

[7] 中田他. 概念設計のための物理推論（第3報）－定性シミュレータの開発－. 1990年度精密工学会春季大会講演論文集, pp. 291–292, 1990.

[8] J. de Kleer. Problem solving with the atms. *Artificial Intelligence*, Vol. 28, No. 2, pp. 197–224, 1986.

[9] K.D. Forbus. Qpe: Using assumption-based truth maintenance for qualitative simulation. *Artificial Intelligence in Engineering*, Vol. 3, No. 4, pp. 200–215, 1988.

[10] H. Lieberman. Using prototypical objects to implement shared behavior in object oriented systems. In *Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 214–223, 1986.

[11] L. A. Stein. Delegation is inheritance. In *Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 138–146, 1987.

[12] T. Tomiyama. Object oriented programming paradigm for intelligent cad systems. In P. J. W. ten Hagen and P. J. Veerkamp V. Akman, editor, *Intelligent CAD System II: Implementational Issues*, pp. 3–16. Springer-Verlag, 1989.

[13] Per Gunnare et al. A graphical approach to naive physics and qualitative reasoning. 人工知能学会研究会資料, number SIG-KBS-8904-2, pp. 11–19, 1989.

[14] T. Tomiyama et al. Metamodel: A key to intelligent cad systems. *Research in Engineering Design*, Vol. 1, No. 1, pp. 19–34, 1989.