挙動のプロトタイピングを支援する設計環境

尾関，聡之
田浦，俊春

(Citation)
精密工学会誌,65(1):67-72

(Issue Date)
1999-01-05

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/90001545

論文

挙動のプロトタイピングを支援する設計環境*

尾関聡之** 田浦俊春***

A Study of Design Environment for Behavioral Prototyping

Satoshi OZEKI and Toshiharu TAURA

During the conceptual design stage (the early and creative stage of the design process), computer aided support of designers' trial and error is very important. In this paper, the authors define and introduce the concept of **behavioral prototyping**, which is different from conventional dynamic simulation. The authors also discuss the design environment for behavioral prototyping. First, the requirements for the behavioral-prototyping support are presented. Next, a methodology of behavioral-prototyping support, to which the impulse-based simulation technique is applied, is proposed. The authors use the system for realizing the behavioral prototyping of a product. Finally, the authors evaluate this approach to behavioral-prototyping support by considering the data related to the designer's thinking process in the design.

Key words: trial and error, design environment, behavioral prototyping, impulse-based simulation

1. はじめに

新たな人工物を設計する際，概念設計の段階において，広く様々な設計案を検討しておくことは，その後の設計プロセスをスムーズに進行させる上で，決して欠くことのできない重要な作業である．このような作業において，設計者は繰り返し試行錯誤を行う必要がある．従来は，試行錯誤の方法として，設計者は主にフリーハンドスケッチなどの方法に頼るところが多く，試行錯誤が不十分であったため，後の設計プロセスまで設計案の基本的な欠陥が認識されない場合もあった．このような理由から，概念設計における設計者の試行錯誤を，計算機を用いて支援することの重要性が指摘されてきた．

従来の試行錯誤を支援するアプローチとしては，より操作性が高く，設計者の試行錯誤に追従できるCADシステムに関する研究[1)2)]などが挙げられる．これらは，主に形状や配置のプロトタイピングを支援することで，設計者の試行錯誤を支援するものであったと言える．それに対して筆者らは，設計案の力学的挙動に関する試行錯誤の支援というものに注目している[3)]．本論文では，挙動のプロトタイピングという概念を従来の力学的シュミレーションという概念とは区別して定義し，導入する．

本研究では，挙動のプロトタイピングという概念を，「概念設計において，設計者が試行錯誤的に力学的挙動を計算機上で試作すること」と定義し，従来の力学的シミュレーションという概念と区別して用いている．もちろん，力学的挙動を視覚化するという意味では，両者は全く同一の概念といえよう．だが，一般の力学的シミュレーションは，設計案が既にほぼ確定している段階，主に実体設計・詳細設計において力学的挙動を確認するための作業ととらえられるのに対して，挙動のプロトタイピングは，概念設計において目的機能を実現するための力学的挙動を設計し，新たな設計案を作っていくための作業であるという意味で異なる概念である．

このような定義の下で，本研究では以下の3項目を目的としている．

- 挙動のプロトタイピングを支援するための要件を整理すること
- 整理した支援要件を実現する挙動のプロトタイピングの支援方法を提案すること
- 模擬的な設計により，これらの支援要件及び支援方法の妥当性を考察・検討すること

以下，2章で挙動のプロトタイピングを支援するための要件を整理し，3章で挙動のプロトタイピングの支援方法を提案する．4章では，本研究において整理・提案した支援要件及び支援方法に基づいて試作した，二次元の支援システムと，それを用いて行った模擬的な設計について述べ，その結果から本研究の有効性を評価する．そして，5章では，結論と今後の展望を述べる．

2. 挙動のプロトタイピングを支援するための要件

挙動のプロトタイピイングを支援するための要件を，シミュレーションデータの編集及び評価の2つの観点から整理する．

2.1 第一の要件（挙動データの編集を支援するための要件）

概念設計において，設計者の思考は常に変化・推移し，多様な設計案が設計者の頭の中で描かれていく．挙動のプロトタイピングを支援するためには，この設計者の思考過程に柔軟に対応しながら，設計案の力学的挙動を解析し，視覚化できなければならない．そうすることで，設計者にとって試行錯誤しやすい環境が提供されると本研究では考えている．では，試行錯誤の過程において設計者は頭の中でどのような操作を行っているのだろうか．設計者の思考が設計案Aから設計案Bへ推移する時，設計案Aの主に形状・配置・挙動情報など様々な属性情報に適当な変更操作を施すことで設計案Bに至っている，と本研究ではとらえている（図1）．このような属性情報の変更の中で，特に操作が困難なのは，設計案を構成する部品群の接触状況の変更である．

そこで，本研究では，多様な力学的挙動の中で特に，頻繁な接触状況の変化を伴う挙動を支援の対象としている．

すなわち，本研究では，

「設計案に対する様々な属性情報の変更が容易で，かつ頻繁な接触状況の変化に対応して挙動を提示できること」

を挙動のプロトタイピングを支援するための第一の要件としてとらえる．

* 原稿受付 平成10年3月27日
** 学生会員 東京大学大学院（現，日本電信電話(株)；東京都千代田区大手町1-5-1）
*** 正会員 東京大学人工物工学研究センター（東京都目黒区駒場4-6-1）

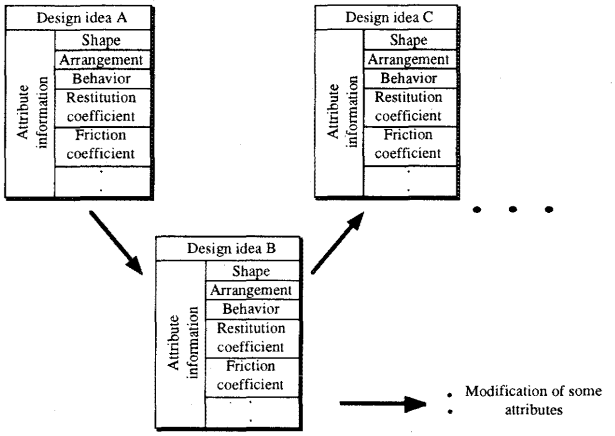

Fig.1 Changing process of design idea

ここで述べた第一の要件は，設計者の試行錯誤に柔軟に対応できる仕組が必要であるという意味で，形状や配置のプロトタイピングとも共通する支援要件である．次節では，挙動のプロトタイピング特有の支援要件について議論していきたい．

2.2 第二の要件（挙動の評価を支援するための要件）

通常，形状や配置に注目したプロトタイピングにおいて，設計案の属性情報は時間の進行に伴って変化することはなく，時間という概念が明示的に取り扱われることはない．一方，本研究で注目している挙動のプロトタイピングにおいて，設計案の属性情報は時間の流れとともに絶えず推移するものであり，挙動のプロトタイピング支援においては，この「時間の流れをどのように扱うか」が注目すべき重要な課題となる．

この課題に対処するための一指針を得るために，時間という概念に注目しながら簡単な例と共に挙動のプロトタイピングについて議論したい．ここでは例として，設計者の思考が，ある設計案Aから設計案B，設計案Cへと推移していった場合を考えてみる．そして，まず設計者は設計案Aについて時刻 $t = 0$ s から $t = 30$ s までの挙動をプロトタイピングしたとする．その後設計案B，設計案Cへと思考が推移し，それぞれについて挙動のプロトタイプを生成していくが，設計者にとって興味のある挙動の時間的な範囲は，必ずしも設計案Aと同様の30秒間であるとは限らない．むしろ，設計者の思考推移の中で，設計者にとって興味のある，挙動の時間的な範囲も流動的に変化していくと考えられる．このような観点から本研究では，挙動をプロトタイピングする時間的範囲を任意に指定可能とすることが，挙動のプロトタイピングを支援するために有効な時間の扱い方であるととらえている．そこで，

「挙動のプロトタイピングを行う時間的な範囲を，設計者が自分の興味に合わせて任意に指定可能であること」

を挙動のプロトタイピングを支援するための第二の要件としてとらえる．

3. 挙動のプロトタイピングの支援方法

3.1 接触を扱うシミュレーション技法

接触を扱う既存のシミュレーション技法の多くは，制約ベース型シミュレーション技法 (constraint-based simulation) に分類される [4)5)]．これは，物体の接触状況の変化を逐一追跡し，その変化に対応して，適用すべき接触力 (制約力) を求めていく方法であり，接触状況の変化が頻繁に起こる対象には，あまり向いていない．

これに対し，Mirtich らは力積ベース型シミュレーション技法 [6)] と呼ばれる新たなシミュレーション技法を提案し，最近注目を浴びている．この方法の最大の特徴は，物体同士のすべての接触を接触点における衝突としてモデリングしていることにある．つまり，従来のように，接触している物体に対して制約力を適用するのではなく，すべての接触タイプに対して瞬間的な力積を適用することで，物体同士のめりこみを回避するわけである．ここで，すべての接触タイプとは，瞬間的な接触（巨視的な衝突）だけでなく，転がり・滑べり・停止など連続的な接触も含んでいる．例えばテーブルとその上で停止しているコップ同士の接触においてもその例外ではない．このような連続的な接触も，非常に小さな無数の衝突が起こっているものとして，一連の力積列が適用される．このように，力積ベース型シミュレーション技法では，接触状況の変化を追跡する必要がなく，すべての接触状況を，一連の瞬間的な力積を適用する，という統一的な単一の枠組で処理できる．これは，接触状況の変化が頻繁な対象を得意とする技法であり，本研究では，第一の要件を実現する方法として注目する．

3.2 力積ベース型シミュレーション技法

本節では，本研究で注目した力積ベース型シミュレーション技法の基本的なアルゴリズムについて説明する (図2)．

1. 各物体ペアに関して TOI と呼ばれる値を計算する．TOI とは，*time of impact* の略で，物体ペアが衝突するまでに要する時間の下界を現時刻に加えた値である．物体ペアの最短距離は，Lin らの最接近フィーチャ探索アルゴリズム [7)] を用いる．
2. 各物体ペアを要素とするヒープを生成する．各物体ペアは TOI の値に基づいてソートされており，ルートに位置する物体ペアの TOI ($lowestTOI$) が最小となる．
3. 時刻 t=$lowestTOI$ までシミュレーションを進行させる．この時刻までは，いかなる物体ペアも決して衝突しないことが保証されている．衝突と衝突の間において，物体は自由落下運動に従う．
4. 時刻 t=$lowestTOI$ において，ヒープのルートに位置する物体ペアについてのみ衝突チェックを行う．衝突チェックとは，物体ペアの最短距離 d と，ある微小しきい値 ϵ との大小比較を行うことで，$d > \epsilon$ なら5へ，$d < \epsilon$ なら6へ．
5. ルートに位置する物体ペアに関してのみ，再度 TOI を計算し，ヒープをソートし直す．3へ．
6. 適用すべき力積 (p) を，ポアソンの仮定とクーロンの摩擦モデルの仮定のもとに，$u'(\gamma) = Mp'(\gamma)$ を積分して求める．ここで，u は物体の速度，M は衝突物体の質量・慣性モーメント及び重心に関する衝突点の相対位置に依存する 2×2 の行列，γ は衝突過程において0から始まるパラメータであり Mirtich の手法では物体の力積の法線方向成分 p_y である．
7. ヒープを構成する要素の内，衝突した物体ペアを含む要素に関して TOI を計算し，ヒープをソートし直す．3へ．

以上のような，単純な処理の繰返しにより，頻繁な接触状況の変化に対応した挙動解析が実現される．

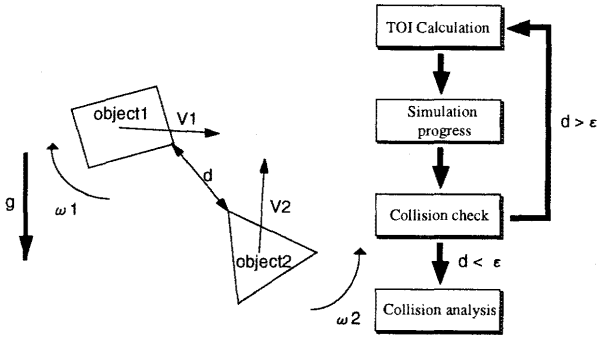

Fig.2 Fundamental algorithm

3.3 プリミティブ挙動パターンのライブラリ化

前節で述べたように，力積ベース型シミュレーション技法は，接触状況の頻繁な変化に柔軟に対応できる仕組みになっているが，これだけで本研究が挙げた第一の要件が実現されるわけではない．というのは，挙動のプロトタイピングにおいては，様々な属性情報の変更が容易でなければならず，設計者は挙動パターン情報*の変更も行うと考えられるからである．しかし，力積ベース型シミュレーション技法において，各物体は自由落下運動に従い，挙動パターンの編集を想定しているものではない．そこで本研究では，まず等速直線運動や等速回転運動などの制約運動を導入し，さらに，自由落下運動を含む幾つかのプリミティブな制約運動を制御するプログラムを，形状モデルとは独立にライブラリ化しておくことで，挙動制御プログラムと形状モデルの任意の組合せを可能とする方法を提案する．

具体的には，設計案を構成する各部品のクラス構造 (Object クラス) とその挙動を制御するクラス構造 (NumInt クラス) を以下のようにした．

- Object クラス
  Object クラスは主に，頂点座標の配列からなる形状情報，重心座標及びそのまわりの回転角からなる配置情報，重心の並進速度とそのベクトル及び重心まわりの角速度からなる挙動情報，そしてプリミティブ挙動パターン情報といった部品の属性情報とそれらに対する処理群 (メソッド群) から構成される (図 3)．形状情報 (coor) は CAD から読み込まれた後，その部品が消去されるまで変更されることはなく，部品の運動に伴って更新されていくのは配置情報と挙動情報のみとし，頂点座標の動的配列からなるポリゴン (poly) が配置情報の変更に基づいて更新され，グラフィック上の部品も更新されていく．このような構造にすることで，挙動制御クラス (NumInt クラス) は部品の形状情報に依存しないメソッド群 (ライブラリ) として記述できる．Object クラスは，ダイアログボックス (図 4) を介して設計者が任意に指定した挙動パターンに対して，NumInt クラスのメソッド群の中から適切なメソッドを選択し，配置情報及び挙動情報の更新情報を得る．設計者は挙動パターン情報だけでなく，このダイアログボックスとのインタラクションにより様々な属性情報を編集可能である．

- NumInt クラス
  NumInt クラスは，Object クラスから部品の様々な運動に伴う配置情報と挙動情報の更新情報の導出部を切り出したメソッド群からなるクラスとした．これは，任意の形状モデルに対して適用可能なメソッド群であり，再利用可能なものである．また，これらは配置情報と挙動情報の更新情報導出部のみを記述すればよく，新たな挙動パターンの追加も容易で，拡張性が高いと言える．

任意の形状モデルに任意の挙動パターンを付加するためには，必ずしもここで述べたようなクラス構造が必要となるわけではないが，新たな制御プログラムの作成が容易になると考え，このようなクラス構造にした．

現在本研究で開発した支援システムでは，自由落下運動以外に，等速直線運動，重心まわりの等速回転運動，重心固定で 1 自由度

* 挙動パターン情報とは，等速直線運動や等速回転運動といった挙動のパターンに関する情報を意味しており，これも各部品の持つ属性情報の一つである，と本研究では考えている．

```
public class Obj implements Cloneable{
  final int BALLISTIC = 0;          \\ballistic trajectories
  final int FIXED = 1;              \\fixed
  final int STRAIGHT = 2;           \\uniform straight-line motion
  final int CIRCULAR = 3;           \\uniform circular motion
  final int CENTERFIXED = 4;        \\the center of gravity fixed
  final int CONVEX = 0;             \\convex polygon
  final int CONCAVE = 1;            \\concave polygon
  private int vnum;                 \\number of vertices
  private double[][] coor;          \\array of vertex coordinates
  private double[] center;          \\coordinates of the center of gravity
  private double angle;             \\angle around the center of gravity
  private double[] transVec;        \\vector of translational velocity
  private double transSp;           \\translational velocity
  private double rotSp;             \\angular velocity
  private double mass;              \\mass
  private double I;                 \\moment of inertia
  private int behaviorType;         \\behavioral pattern
  private int polyType;             \\polygon type(convex or concave)
  private Polygon poly;             \\dynamic array of vertex coordinates
  private Property01 dl;            \\dialog box
  private NumInt integrator;        \\instance of NumInt class
  private Vector Integrator;        \\dynamic array of integrator

  public void evalBehavior(double Time) {

  if (Behavior == BALLISTIC) {
    ((NumInt)(Integrator.lastElement())).calcBallistic(Time);
                    .
                    .
                    .
  }

  \\a group of management of attribute information
}
```

Fig.3 Object class

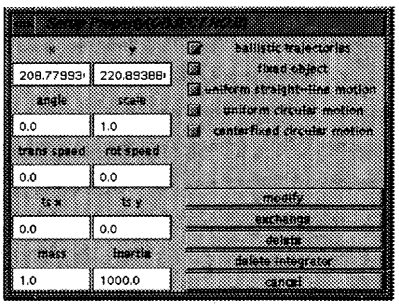

Fig.4 Dialog box for editing attribute information

(回転) のみの運動，完全な固定，の四つの制約運動を実装している．

3.4 REVERSE

本節では，挙動のプロトタイピングを支援するための第二の要件を実現するための一手法として，REVERSE 機能の実装を提案する．

以下，仮想的に時間を実世界と同様の方向に流すことを「順回転」，その逆を「逆回転」と呼ぶことにする．

一般のシミュレーションでは，時間の流れは一方向のみであり，順回転のみ扱う．もちろん，単なる挙動の確認作業を行うためには，これが必要かつ十分な機能と言える．だが，挙動のプロトタイピングにおける設計者の試行錯誤を支援するためには，不十分と考えられる．例えば，ある設計案について挙動のプロトタイピングを行っている途中で，ある部品 A が目的とする機能を満足しなかった場合，その挙動を提示された時点で，設計者の思考における部品 A は代替部品 A′ へ推移している．このような場合，REVERSE 機能は効率的な挙動のプロトタイピングを実現するための有効な手段となる．つまり，設計者は逆回転の挙動を見ながら，部品 A が全体の挙動に影響を及ぼさない時刻まで時間を戻し，そこで部品 A の属性情報を編集し部品 A′ を生成し，挙動のプロトタイピングを再開することができるわけである．REVERSE 機能は，設計者の試行錯誤をより効率的にプロトタイプへ反映し得るものと考えられる．

このような REVERSE 機能の実現方法として，あたかもビデオの巻き戻しのように，部品群の過去の情報をすべて保持する，あるいは，時間が進行している時と全く同様に，時間を逆に流しな

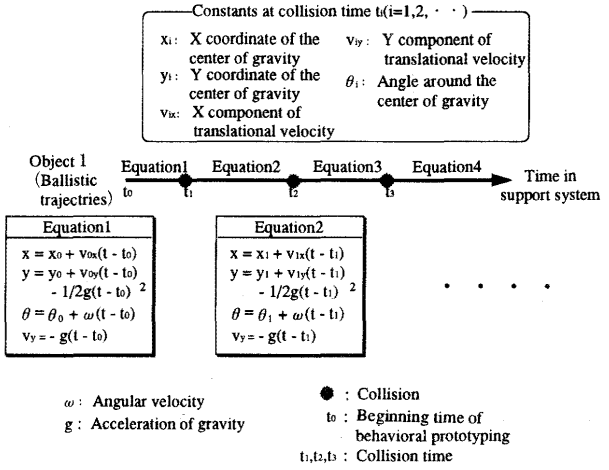

Fig.5 Concept of REVERSE function

がら，衝突チェック・衝突解析を行うという方法が考えられる．だが，前者の方法は，保持すべき情報量が膨大であるし，後者の方法は膨大な時間を要する上に実現が困難である．

そこで本研究では，以下のような REVERSE 機能の実現方法を提案する．今，ある設計案を構成する部品群 (設計対象が扱う物体を含む) に対して，設計者が時刻 $t=t_0$ において挙動のプロトタイピングを開始し，自由落下運動をする部品 1 が時刻 $t=t_1$，時刻 $t=t_2$，時刻 $t=t_3$ において，他の部品と衝突した場合を想定してみる (図 5)．順回転における時刻 $t=t_0$ から時刻 $t=t_1$ までの間，部品 1 の配置情報及び挙動情報の更新情報は，四つの微分方程式 (1) を積分した時刻 t を変数とする図5に示すような方程式 1 において，時刻 t を t_0 から微小単位でインクリメントしていくことで求められる．

$$\frac{dx}{dt} = v_x,\ \frac{dy}{dt} = v_y - gt,\ \frac{d\theta}{dt} = \omega,\ \frac{dv_y}{dt} = -g \qquad (1)$$

方程式 2，3，4 についても，定数の値こそ異なるが，全く同型の方程式として記述することができる．例えば，方程式 2 については，図 5に示したように記述できる．ここでは，自由落下運動に従う部品を例に述べたが，本研究において導入した制約運動についても全く同様で，配置情報及び挙動情報の更新情報を導出するための，衝突間の方程式は同一部品に関しては，全く同型で記述することができる．本研究では，これを利用し，過去に関する最小限の情報のみを保持することで REVERSE 機能を実装した．ここで，最小限の情報というのは，図 5に示した衝突時刻 t_i における定数群と衝突時刻 t_i のことであり，REVERSE を実現するために部品群に関する過去の情報を微小時間ごとに保持しておく必要はないということである．なぜなら，上述したように同一部品に関して方程式の型は変化しないため，衝突時刻 t_{i+1} から直前の衝突時刻 t_i までの配置情報及び挙動情報は，これらの定数を用いて時刻を微小単位でディクリメントしていくことで導出できるからである．

3.5 KEEP & RESTORE

本節では，挙動のプロトタイピングを支援するための第二の要件を実現するためのもう一つの手法として，KEEP & RESTORE 機能の実装を提案する．

今，設計者がある設計案についての挙動のプロトタイプを生成した結果，設計案の構成部品 A の挙動が目的機能の実現に至らなかった場合を考えてみる．そして，プロトタイピングの結果から，設計者は部品 A に代わる代替部品 A′ を考案したとする．この代替部品 A′ についてのみ挙動のプロトタイピングを行いたいのであれば，前

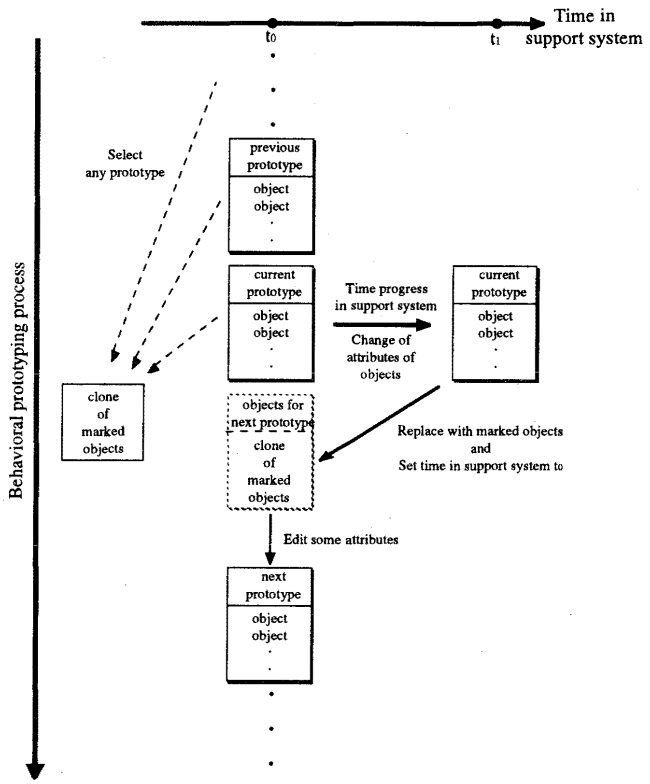

Fig.6 Concept of KEEP & RESTORE function

節で提案した REVERSE 機能を用いて，適切な時刻まで時間を逆回転させ，効率的に代替部品 A′ について挙動のプロトタイピングを行うことができる．だが，通常，ある部品の挙動が設計者の意図に合致しない場合，その後設計者は試行錯誤しながら複数の代替部品について挙動のプロトタイピングを行うことで，代替部品が目的機能を実現する可能性について繰り返し評価していく必要があるだろう．このように，挙動のプロトタイピングを繰り返し行う時間的範囲の始点があらかじめわかっているような場合，REVERSE 機能だけでは非効率的である．REVERSE 機能とは，挙動のプロトタイピングを行う時間的範囲の始点があらかじめわかっていない場合に有効となる手段と言える．だが，時間的な範囲の始点がわかっていれば，プロトタイプの状態をわかっている始点の時刻 $t=t_0$ に即座に戻せる仕組みが，第二の要件を実現するために求められる．

そこで本研究では，上述したような状況における設計者の思考過程に追従し，より効率的な挙動のプロトタイピングを可能とするため，KEEP & RESTORE 機能の実装を提案する．この機能は，設計者が，任意のプロトタイプ生成過程の任意の時刻で，マーキングしておき，その後いつでもマーキングした状態に戻ってプロトタイピングを再開できるという仕組みを提供するものである．

このような KEEP & RESTORE 機能の実現方法を，本研究では以下のように提案する．まず，設計者によりマーキングされた部品群のクローンを生成・保持する．ここで，部品群のクローンの生成とは，それぞれの部品に対して全く同一の属性情報を持つ部品を生成することを意味する．マーキングされた後，設計者が指定する任意の時刻で，部品群を保持しておいたクローンで置換し，システム内の時刻をマーキングされた時刻に戻す．そして，設計者はクローンで置換された部品群の属性情報を変更することで，次なる挙動のプロトタイピングを開始する．この一連の流れを図解したものを，図 6に示す．

ここで述べた方法は，システムで最後に生成したプロトタイプを

時刻 $t=t_0$ の状態に戻すだけでなく，マーキングさえしておけば，過去に生成した任意のプロトタイプの時刻 $t=t_0$ の状態に戻れる仕組みを提供するものである．このようにすることで，設計者は，時刻 $t=t_0$ におけるどのプロトタイプにマーキングするかで，挙動のプロトタイピングを繰り返し行う際の拠点となるプロトタイプを選択可能となる．

4. 支援システムの試作と模擬設計

4.1 支援システムの概要

本研究では，3章で述べた手法に基づいて，二次元の挙動のプロトタイピング支援システムを試作した．本支援システムは，実装言語としてJAVAを用いてUNIXワークステーション上にインプリメントされている．なお，本支援システムにおいて，衝突判定のしきい値 ϵ は $1.0e^{-2}$ cm に，シミュレーション進行のタイムステップは $5.0e^{-2}$ s に設定した．本支援システムのアーキテクチャを図7に示す．図8は，本支援システムの画面仕様を示したものである．本支援システムでは，物理定数等の情報を，容易に変更できるインタフェースを備えている．たとえば，反発係数や摩擦係数を入力したり変更する場合には，図8のEditメニューからSet Coefficientメニューを選択すると，図9に示すようなダイアログボックスが表示されるので，それを介して入力や変更の作業を行うことができる．

4.2 模擬設計

4.2.1 模擬設計の実施方法

設計者が模擬的に与えられた設計課題と設計環境において行う設計を，本研究では「模擬設計」と呼ぶことにする．模擬設計の実施目的は，ある設計環境下における設計者の試行錯誤を観察し，その設計環境を評価することにある．本研究では，以下のような設定の下で模擬設計を行い，設計者の思考過程を観察した．なお，模擬設計の実施方法には，設計実験の方法と分析に関する武田らの研究[8]を参考にした．

- 設計者
  精密機械工学を専攻し，設計について研究している大学院生を設計者とした．
- 設計課題
  設計課題は，「傾斜20°の平らな搬送路を，ランダムな間隔で流れてくる八角形の部品を，等間隔で供給し直す機構の設計」と設定した(図10)．
- 設計環境
  設計者には，本研究において開発した支援システム及びCADのみを用いて模擬設計を行ってもらった．また，設計者は二人とし共同設計を依頼した．
- データの採取方法
  設計者の思考過程に関するデータとして，主にシステムに対する設計者の操作履歴のデータをビデオにより採取した．

4.2.2 模擬設計の結果と考察

設計者は，約2時間の試行錯誤の過程において，図11に示すような第一プロトタイプから図12に示すような第七プロトタイプまでの7個のプロトタイプをシステム上で生成した．第一プロトタイプでは，等速回転運動をする正三角形を，第七プロトタイプでは，等速回転運動をする六つの歯を持つ歯車を，それぞれ設計案とした．図13は，今回行った模擬設計において設計者が各々のプロトタイプ生成過程で，システムに対して行った操作履歴を示すものである．ここで，挙動情報編集とは，並進速度ベクトルとその大きさ，及び角速度のような数値入力により指定する情報編集を意味し，挙動パタ

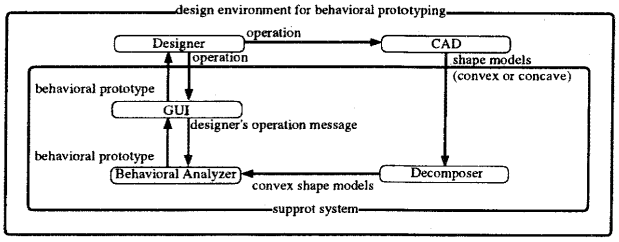

Fig.7 Architecture of our support system

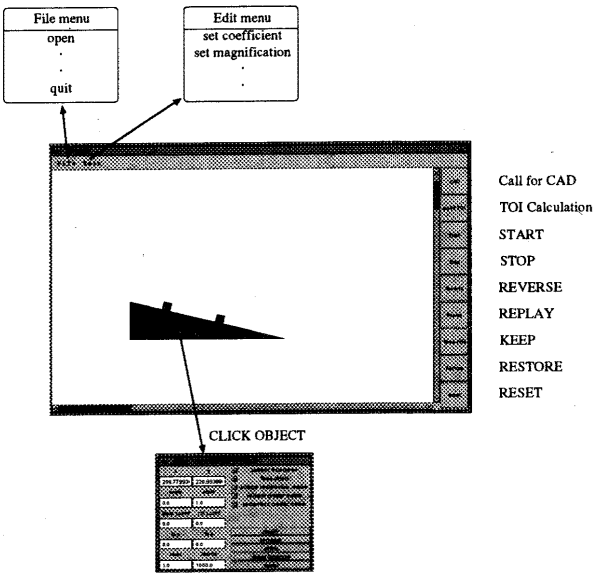

Fig.8 GUI of our support system

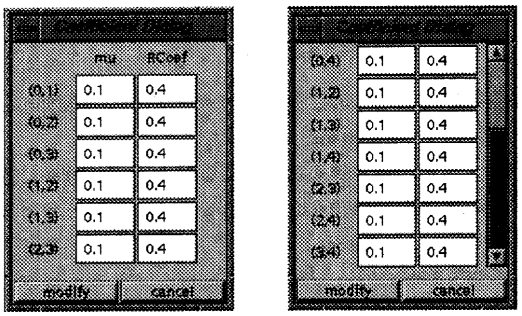

Fig.9 Dialog box for editing the restitution coefficients and the friction coefficients

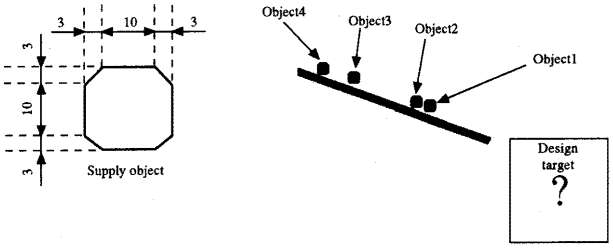

Fig.10 Design problem

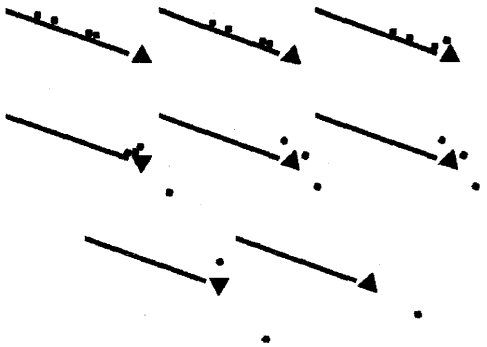

Fig.11 1st prototype

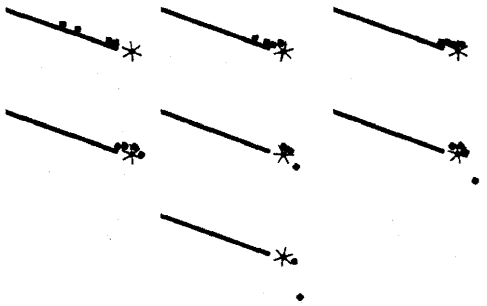

Fig.12 7th prototype

	Edit attribute information					Specify a span of time	
Prototype	Shape	Arrangement	Behavior	Behavioral pattern	Restitution coefficient	Reverse	Keep & Restore
1	○	○	○	○			
1→2			○				
2→3					○		
3→4			○				
4→5			○				
5→6	○					○	
6→7		○					○

Fig.13 List of operations during the design

ーン情報編集とは，自由落下運動や他の制約的な運動といった挙動のパターンを選択する編集を意味する．

図13から，設計者は試行錯誤の過程において，確かに設計対象の様々な属性情報を変更しながら次々と挙動のプロトタイプを生成していっていることが読みとれる．そして，このような様々な属性情報の編集を容易に行える環境を本支援システムが提供したことで，設計者は新たな設計案に対する挙動のプロトタイピングを容易に行えており，今回の模擬設計から得られた結果は，本研究で挙げた第一の要件が挙動のプロトタイピングを支援するための重要な要件であることを示す一つのデータであると言える．

次に，挙動のプロトタイピング支援の第二の要件とその実現方法の妥当性について考察する．本模擬設計において，設計者は本研究が提案する時間的範囲を指定する機能を，後半2回用いることで，新たな設計案に対して効率的に挙動のプロトタイピングを行っており，本模擬設計の結果は第二の要件の妥当性を示す一つのデータと言える．しかしながら，第二の要件に対するこれらの機能の必要十分性については，評価を下すに至るデータは得られていない．とくに，本模擬設計後，設計者は

「時間的な範囲を指定できること，それ自体はプロトタイピングを効率的に行うには有効だと思う．だが，今回の設計課題では，時間的な範囲を指定しないで，初めからプロトタイピングし直しても効率はあまり変わらない．」

との感想を述べており，今後更なる模擬設計観察の必要がある．

5. 結論と展望

本論文では，従来の力学的シミュレーションとは区別して，「挙動のプロトタイピング」という概念を定義・導入した上で，以下のことを行った．

(1) 「属性情報の編集操作の容易性」及び「挙動のプロトタイピングを行う時間的な範囲指定の容易性」に注目し，挙動のプロトタイピングの支援要件を整理し，支援方法を提案した．
(2) 試作した支援システムを用いた模擬設計により，支援要件及び支援方法の妥当性を検討した．
(3) 本研究のアプローチが，挙動のプロトタイピングの支援方法として有効である可能性を見い出した．

今後の課題としては，以下のような項目が挙げられ，さらに研究を進めていく必要がある．

(1) 多様な制約的挙動パターンのライブラリ化
(2) 多様な設計者と設計課題による設計者の思考過程・試行錯誤の観察・分析
(3) 支援システムの三次元化

参考文献

1) J.Liang and M.Green: Quick Prototyping for Engineering Design, Computer Applications and Design Abstraction, 49, (1993) 157.
2) A.J.Cartwright: Interactive Prototyping - A Challenge for Computer Based Design, Res. Eng. Des., 7, (1997) 10.
3) 田浦俊春: メディアとしてのシミュレーション，シミュレーション，15, 3 (1996) 13.
4) D.Baraff: Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies, Computer Graphics, 23, 3 (1989) 223.
5) M.Moore and J.Wilhelms: Collision Detection and Response for Computer Animation, Computer Graphics, 22, 4 (1988) 290.
6) B.Mirtich and J.Canny: Impulse-based Simulation of Rigid Bodies, Proc.1995 Sympo. Interactive 3D Graphics, (1995) 181.
7) M.C.Lin and J.F.Canny: A Fast Algorithm for Incremental Distance Calculation, Int. Conf. Robotics and Automation, IEEE, (1991) 1008.
8) 武田英明，濱田 進，冨山哲男，吉川弘之: 設計実験における実験方法の検討と作図過程の分析，精密工学会誌，58, 11 (1992) 1849.