グラフィクスとCAD 77－9
（1995．11．9）

デザイナ向けモデラにおける履歴操作機能の開発

朝井俊光　原口雄典　二上範之
シャープ株式会社　生産技術開発推進本部

本稿では、ワイヤーフレーム、サーフェイス、ソリッドモデルを複合的に扱えるCAD／CAMシステムにおいて、3次元形状を対話的に設計する際に必要とされる履歴管理機能について提案する。今回、我々が開発した手法は、以下の特徴を有している。

(1)形状作成／変更操作に伴って生成／修正／削除された全モデルの状態遷移関係ならびに実行コマンド列を記録することで、モデルの表現形式に依存しない履歴操作が実現できること。

(2)設計過程において関係したモデルであれば、1つの関係木構造上に表示して操作できること。

(3)関係木構造へのマーキング機能により、関係木構造を直接操作しなくても履歴操作ができること。

以下、本文において各項目についての詳細な説明とその使用例について述べる。

Development of History Operations on a 3D Modeler for Designers

Toshimitsu ASAI, Yusuke HARAGUCHI, and Noriyuki FUTAGAMI
SHARP Corporation
Production Technology Development Group

In this paper we propose a method of history operations on our CAD/CAM system which can handle wireframe, surface, and solid models. It is useful for designers to manipulate histories of interactive design process, because their design process proceed through try and error.

Our new history function has following features.

(1)The history operations are managed by recording all executed commands and state information of all 3D objects relating to them. And this function can be applicable to all kinds of model representations, that is, wireframe, surface, and solid models.

(2)Designers can handle single tree including their whole design process. And using this tree, they can go back to a previous stage of the process, and regenerate an arbitrary 3D object at the stage.

(3)Quick history operations are reallized by marking mechanism which were put at arbitrary positions in the tree.

The above features and some examples are described in detail.

1　はじめに

最近、コンカレントエンジニアリングという言葉をよく耳にするが、これは製品の企画から設計、製造、販売、廃棄までの製品に係わる全サイクルにおいて、製品設計と関連プロセスとを統合化し、生産およびその支援部門において同時進行を実現するための手法である。この手法を実現するにあたっては、設計の上流にある意匠設計から技術設計、下流の金型設計までの広い範囲の製品設計におけるコンピュータ化が必要となる。この中でも設計の上位に位置する意匠設計部門のコンピュータ化が、以降の設計に大きく影響を与えるため重要となる。

現在、多くの企業では製品設計におけるコンカレントエンジニアリングを目指してＣＡＤ／ＣＡＭ／ＣＡＥシステムの導入ならびに運用を推進して来ているが、意匠設計部門のコンピュータ化が遅れているのが現状である。その理由の一つとして、現状のＣＡＤ／ＣＡＭシステムは製品設計図面を基に正確な数値情報を直接入力して形状を設計するのに適していることが挙げられる。また、現状のＣＡＤ／ＣＡＭシステムは、意匠設計部門におけるデザインプロセスに合致した構成になっていないことも挙げられる。

当社においても、前述したように意匠設計部門におけるデザインプロセスに合致したモデラの需要が高まって来たこともあり、約５年前に意匠設計部門においてＣＡＤ委員会が発足し、デザイナのためのＣＡＤ／ＣＡＭシステムの開発がスタートした。

デザイナ向けモデラ（以下デザインモデラと呼ぶ）を開発するにあたり、デザインプロセスにおける要求仕様としては様々なものが挙げられたが、基本形状を作成するプロセスにおいては、２次元的入力アプローチと３次元的入力アプローチとが存在し、それぞれのアプローチをを満足する必要があった。

２次元的入力アプローチとは、アイデアスケッチをトレースして断面形状を求め、その形状データを利用して３次元形状作成を行うようなワイヤーフレームモデルからサーフェイスモデル、ソリッドモデルへと移行するプロセスを言う。

３次元的入力アプローチとは、プリミティブを組み合わせたり、切り離したり、削ったりというソリッドモデルを基本形状に据えてワイヤーフレーム、サーフェイスモデルを有効的に活用して、ソリッドモデルを加工して行くプロセスを言う。

デザインモデラは、上記入力アプローチを可能とするため、ワイヤーフレーム、サーフェイス、ソリッドモデルを複合的に扱える対話的な設計環境を提供する。

ところで、一般に対話的なＣＡＤ／ＣＡＭシステムにおいて、ユーザーの意図した製品形状を作成するためには、ユーザーが行った操作を取り消す機能（以下UNDO機能と呼ぶ）、取り消した操作を再度実行する機能（以下REDO機能と呼ぶ）が必要不可欠である。これらの機能の実現により、ユーザーは安心して試行錯誤を繰り返すことができ、最良の設計が可能となる。これまで、UNDO/REDO機能のような設計プロセスにおける履歴操作機能に関して、いくつかの研究事例が報告されている[(1) (2)]。しかし、これらの事例は、いずれもソリッドモデルを基本としており、デザインモデラのようにワイヤーフレームモデルから、ソリッドモデルまで複合的に扱う場合の手法については言及されていなかった。

本稿では、デザインモデラにおいて実現したモデルの表現形式に依存しない履歴操作機能の実現方法の提案、並びに対話性に富んだ履歴操作インターフェースについて報告する。

2　システム構成

デザインモデラは、ユーザーとの対話を支援する部分、設計プロセスにおける幾何計算を行う部分、設計履歴を管理する部分、形状データの状態を管理する部分から構成されている。図１にシステム構成図を示し、以下にシステムを構成する主要部分の役割について述べる。

◎GUI 部

ユーザーとの対話を司る部分であり、メニュー画面の表示や、メニューを利用したコマンドの入力、そしてユーザーに対しての適度なフィードバックを与える。

◎設計履歴管理部

GUI 部における入力操作を解釈して、コマンドの実行、設計履歴データの保管、そして、UNDO/REDO機能といった履歴操作機能の実行等デザインモデラの中枢を管理している。

◎幾何演算処理部

設計履歴管理部で解釈されたコマンドにより、サーフェイスモデル間の交線計算、ソリッドモデル間の集合演算等、デザインモデラにおける全ての幾何演算処理を行う。

◎形状データ状態管理部

デザインモデラにおける形状データ、ならびにコマンド実行前後における形状データの状態の変化を全て記録管理している。また、設計履歴管理装置からの指令を受けて、状態の変更を行う。

また、図1における各処理部間の2種類の矢印には、それぞれ異なる意味がある。⇒は、矢印の先にある処理部に働きかけて、その処理部に何らかの処理を行わせようという制御関係を意味する。例えば、ユーザーが、メニューからコマンドを選択してGUI 部を制御するという処理がそれにあたる。同様に、設計履歴管理部は幾何演算処理部が行うコマンド列の実行の制御と形状データ状態管理部が行う状態の変更の制御を行なっている。一方、→は、⇒による処理の結果としてのデータの流れを意味する。

3　設計履歴の記録

3.1　設計プロセス

デザインモデラにおける設計プロセスは、形状データの状態の変遷と考えられる。例えば、ユーザーがGUI 部から入力した形状作成／変更操作のコマンドによって、コマンド実行前と後とでは形状データの状態が変化する。この状態の変化は、設計プロセスにおける時間的変遷を意味するから、時系列に状態の変化を記憶すれば、設計プロセスの履歴情報として活用できる。この状態の記録機能を担当しているのが、前述の形状データ状態管理部である。

形状データ状態管理部では、設計プロセスにおける最終形状データ群と、各設計プロセス間の差分、すなわち1設計プロセスの前後で対象となった形状データの時間的な変化分をコマンドの実行結果として記録している。

3.2　UNDO/REDO機能

デザインモデラにおけるUNDO/REDO機能は、上述の設計プロセスにおける1設計プロセスを時間的に1ステップ戻す／進める操作として実現している。この機能の実現により、過去に存在したことのある形状データであれば、再現することが可能となる。

しかし、この方法では、過去に存在した形状データを再現した後、形状データに変更を加え、更に再度同じ設計プロセスを辿るように再実行することはできない。また、形状データ状態管理部において設計プロセスにおける形状データの差分だけを記録しているからと言って、

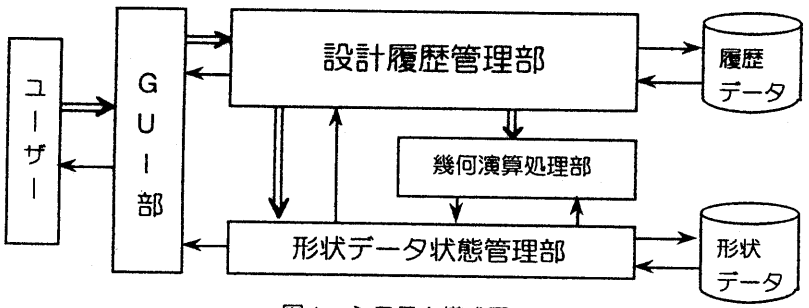

図1　システム構成図

数日間にも渡るような設計プロセスの最初から最後までの状態を記録するのには限界があるため、デザインモデラでは、システムが起動した時点を設計プロセスの最初の状態とし、システムが終了するまでの間だけ保持するように設計されている（4.1節にて、詳細に説明する）。

3.3　履歴管理の要求仕様

上述したように形状データの状態情報を保持することだけでは、設計履歴情報としては不十分である。

ユーザーに対して十分に対話的なＣＡＤ／ＣＡＭシステムの提供を行うための設計履歴の記録手法に求められる要求仕様としては、以下のものが挙げられる。

(1)システム稼働時における設計プロセスの状態の記録の他に、外部ファイルに保存が可能な設計履歴の記録ができる。
(2)形状データの全設計プロセスを記録できる。
(3)過去の設計プロセスに存在した形状データを変更して、再度同じ設計プロセスの実行ができる。
(4)ソリッドモデルだけで無く、他のモデルの設計履歴も扱える。
(5)システム稼働時においては、形状データの状態情報も設計履歴情報として利用できる。

3.4　設計履歴の記録内容

上記要求仕様を実現するために、設計履歴情報として、以下のものを記録している。

◎１設計プロセスにおける全コマンド列、ならびにコマンド列が必要とするパラメータを全て記録する。
◎１設計プロセスに係わる形状データについては、モデルの表現形式に関係無くプロセスの前後における状態遷移関係を全て記録する。
◎各設計プロセス単位に記録される情報に形状データ状態管理部との関連付けを行う。

3.5　設計履歴の記録手法

図２における設計プロセスを例に設計履歴の記録手法について詳細に述べる。

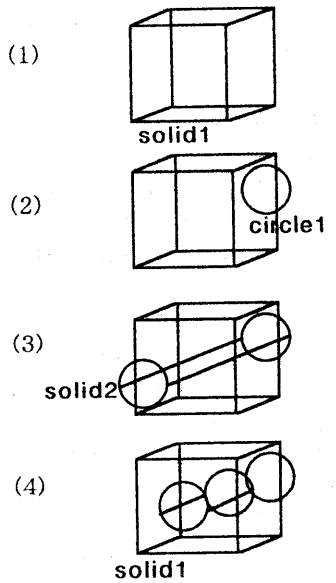

図２　設計プロセス

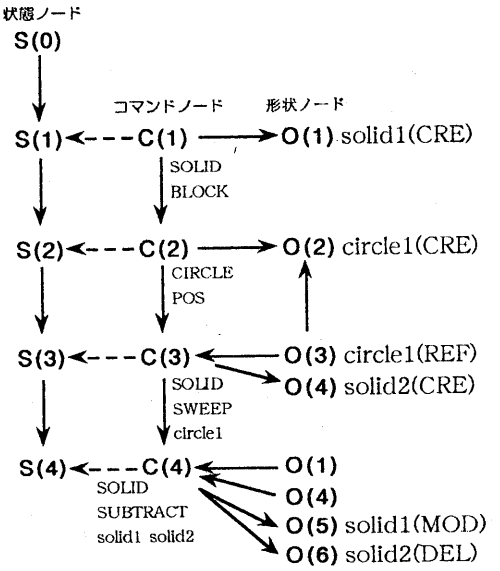

図３　設計履歴情報の関係

図2は、デザインモデラにおける実際の設計プロセスにおいて、ワイヤーフレームモデルとソリッドモデルとを扱った例である。

同図の(1)～(4)はそれぞれ、以下の設計プロセスを表している。

(1)直方体(solid1)を生成する。

(2)円(circle1)を生成する。

(3)circle1を利用して円筒(solid2)を生成する。

(4)solid1からsolid2を集合演算操作で引き算する。

図3は、上記設計プロセスにおいて、形状データ状態管理部、設計履歴管理部に記憶される設計履歴情報の関係図を示している。

同図のS(0)～S(4)は、形状データ状態管理部において管理される記憶単位を表し、状態ノードと呼ぶ。そして、C(1)～C(4)、O(1)～O(6)は、設計履歴管理部において管理される記憶の単位を表し、それぞれ、コマンドノード、形状ノードと呼ぶ。また、各ノード間の矢印は、連結状態を示している。以下、それぞれの記憶単位の役割について述べる。

◎状態ノード：S(0)～S(4)
形状データ状態管理部において管理されている記憶単位で、3.1節で述べた各設計プロセスにおける形状データの状態変化の情報を持つ。

◎コマンドノード：C(1)～C(4)
設計履歴管理部において管理されている記憶単位で、各設計プロセスにおいて実行されたコマンド列、そしてコマンド列の実行に必要な入力、及び実行結果として出力される形状ノードへの連結情報を持つ。また、コマンド列の実行により、作成された状態ノードへの連結情報も持つ。

◎形状ノード：O(1)～O(6)
設計履歴管理部において管理されている記憶単位で、各設計プロセスの前後で、コマンド列の実行により、生成／修正／削除された形状データ、ならびに、参照された形状データの情報を持つ。コマンドノード、参照された形状ノードへの連結情報を持つ。

以下、デザインモデラにおける1設計プロセスにおいて、前述の状態ノード、コマンドノード、形状ノードを記憶する手法について説明する。

step1: GUI部からコマンドが入力される。

step2: 設計履歴管理部にて、各コマンドが構文解析され、このプロセスに対応するコマンドノードC(i)が作成される。その際、コマンド列とそのパラメータが記憶される。

step3: 設計履歴管理部から来たコマンド列が幾何演算処理部にて実行され、その結果が形状データ状態管理部に管理される。その際、状態ノードS(i)が作成される。

step4: 設計履歴管理部にて、コマンド列の入力パラメータに存在する全形状データに対して、その状態が変化しているかどうか判定される。

状態が変化している場合は、コマンド実行前の状態を持つ形状ノードOin(i)をC(i)の入力として連結する。

変化していない場合は、Oin(i)を参照する形状ノード$O\mathrm{in}(i_{REF})$を作成して、C(i)の入力として連結する。

step5: 設計履歴管理部にて、コマンド列の実行により、作成／修正／削除された全ての形状データ対応する形状ノードOout(i)を作成し、C(i)の出力として連結する。

step6: 設計履歴管理部にて、S(i)をC(i)の現在の状態として連結する。

上記ステップに示されるように、実行コマンド列とその入力パラメータとなるモデルの状態を記録する方式を用いており、記憶する情報量の負荷が軽く実用的である。また、ソリッドモデルをワイヤーフレームモデルで加工する場合等、コマンド実行前後において影響を受けないモデルに対しても、参照関係を設計履歴情報として保持できるため、モデルの表現形式に依存しない設計履歴情報を過不足無く記録できる。

4　履歴操作機能の実現

4.1　UNDO/REDO機能の条件

デザインモデラでは、UNDO/REDO機能は以下の条件に基づいて設計されている。

(1)UNDO機能を実行しても設計変更として認識しない。
(2)UNDO/REDO機能はシステムが起動してから実行されたプロセスに限定する。

(1)の理由として、UNDO機能はユーザーの誤操作の取り消しの際に頻繁に使用されるからである。したがって、UNDO機能を続けて実行した直後にREDO機能をその回数分実行して元の状態に戻すことは可能だが、その間に新たなコマンドを実行して、新しいプロセスの状態を生成した場合、そのままでは設計プロセスの分岐が発生してしまうので、UNDO情報は消去される。

(2)の理由として、例えば、履歴情報を持った形状ファイルをシステムに読み込んだ直後にUNDO機能を実行する場合を考えてみると、ユーザーは過去の設計プロセスに戻ることよりもファイルを読み込んだという行為を取り消すことを望むからである。

上記の条件に基づき、UNDO/REDO機能を実行すれば、必ずターゲットとなる形状データの状態情報が存在することになり、コマンドの再実行を伴わない素早い状態の移動が可能となる。

実際には、設計履歴管理部において、GUI部からUNDO/REDOイベントを受けた後、コマンドノードが参照している状態ノードの情報から、形状データ管理部に状態の変更を指令することで、UNDO/REDO機能を実現している。

4.2　設計履歴の関係木構造

3章で述べたように、設計プロセスにおける設計履歴情報は記録されるが、その情報をユーザーに的確に伝えることが必要である。デザインモデラでは、形状データが生成されるプロセスを関係木構造で表現し、ユーザーに設計プロセスを把握し易くしている。

設計履歴管理部にて、関係木構造を生成する際に設計プロセスをトレースする方法を述べる。

step1:　形状データに相当する形状ノードを探す。
step2:　step1で見つけた形状ノードを生成したコマンドノードを探す。ただし、形状ノードが他の形状ノードを参照している場合は、参照している形状ノードを生成したコマンドノードを探す。
step3:　コマンドノードの入力に連結された全ての形状ノードに対して、step1～step3の処理を行う。コマンドノードに連結された形状ノードが無ければ、処理を終わる。

上記のステップにより、形状データを生成するプロセスがトレースされ、関係木構造が生成できる。図4は、図2におけるプロセス(4)の形状データsolid1が生成されるプロセスを関係木構造で表現した例である。ユーザーが理解できるように、各ノードが示すコマンド名、ならびに形状データ名を使用して表現される。

デザインモデラでは、上記形状データの関係木構造を作成する機能を利用して、次節より説明する履歴操作機能を実現している。

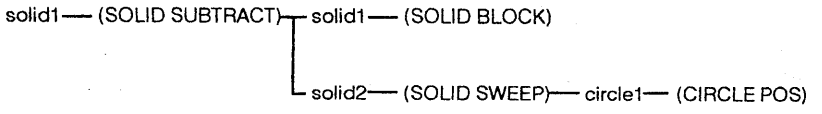

図4　関係木構造

4.3 MOVE機能

履歴情報として記録された全設計プロセスにおける任意の状態に1回の操作で戻れる機能である。後述する関係木構造インターフェース上で形状データの設計プロセスを確認した後、形状データ名を指示するだけで、その形状データが存在した状態まで戻ることができる。例えば、図4のsolid1の設計プロセスにおいてcircle1を指示した場合、設計履歴管理部は、circle1が生成された状態である図2におけるプロセス(2)へ移動する。

デザインモデラでは、この機能を単なる誤操作からの回復としてではなく、ユーザーが意図的に行った設計変更行為として認識しているので、自動的に移動前の状態から移動後の状態までの設計履歴情報を現在の設計履歴情報から切り放し、設計変更履歴として別個に管理する。その際、設計プロセスの分岐点に位置する形状ノードに対して自動的にマーク属性を付加する（以下マーキング機能と呼ぶ）。

上記機能を実現するにあたり、設計履歴管理部では、以下のステップで処理が行われる。

step1: 前節のstep1と同様。
step2: 前節のstep2と同様。
step3: step2で見つけたコマンドノードが参照している状態ノードが存在するか判定する。
状態ノードが存在する場合は、状態ノードの情報から、形状データ管理部へ、状態の変更の指令を出し、状態を変化させる。
状態ノードが存在しない場合は、その状態に存在し得る全形状データの関係木構造を作成し、各形状データを各々の生成プロセスに沿ってコマンド列を実行して再生成する。
step4: 移動前の状態から、移動後の状態までの全べてのコマンドノードと形状ノード情報を別個の履歴情報として管理する。その際、step1で見つけた形状ノードを設計プロセスの分岐点として、マーク属性を付加する。

前記step4で実施されるマーキング機能により、マークされた形状データの情報は、後述の情報一覧インターフェースにおいて、マーク一覧としてリストアップでき、関係木構造を操作することなく、リストから選択するだけで、マークされた状態まで戻ることができる。　また、MOVE機能実行時に別個に管理される設計変更履歴も同様に一覧できる。

4.4 COPY機能

履歴情報として記録された全設計プロセスにおいて存在した任意の形状データを過去の状態に戻ることなく、再現する機能である。MOVE機能と同様に、関係木構造上で形状データ名を指示するだけで、その形状データの生成されるまでのコマンド列が自動的に再実行され、形状データの再現ができる。

4.5 EXCHANGE機能

形状データの設計プロセスにおいて存在した任意の形状データを別の形状データに置き換えて、以降の設計プロセスを自動的に再実行する機能である。MOVE機能と同様に、関係木構造上で変更したい形状データ名を指示するだけで、設計プロセスを自動的に再実行できる。

5 履歴操作インターフェース

デザインモデラでは、ユーザーからの入力を以下に示すインターフェースを利用して対話的に処理する。

5.1 関係木構造操作インターフェース

図5は、形状データの関係木構造をユーザーが確認しながら操作を行う際に利用されるインターフェースの例である。ユーザーは、形状データ名の箇所をマウスでクリックすることで設計プロセスを1階層ごと確認することができる。また、コマンド名の箇所をクリックすれば、図6に示すコマンド情報ウィンドウが現れ、コマンド実行時の情報が確認できる。その際、コメント欄をクリックすれば、自由にメモ書きができるので、設計プロセスを後で確認する際に便利である。

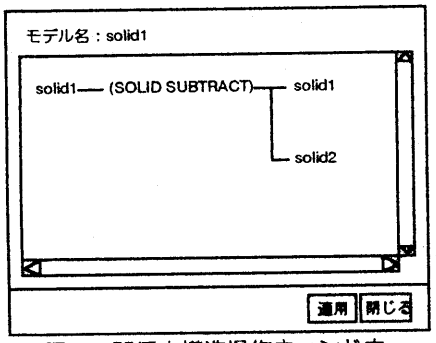

図5　関係木構造操作ウィンドウ

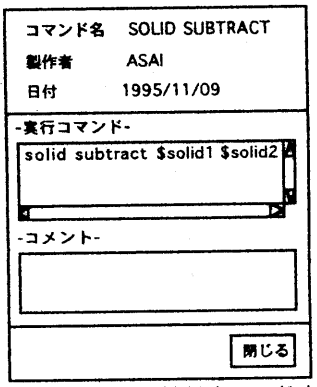

図6　コマンド情報ウィンドウ

5.2　情報一覧インターフェース

図7は、変更履歴の一覧、マーク箇所の一覧を行い、関係木構造を直接操作することなく素早い状態の移動、状態の再現を行う際に利用されるインターフェース例である。ユーザーは、ただ一覧の項目を選択し、適用ボタンを押すだけで状態の変更を実施できる。

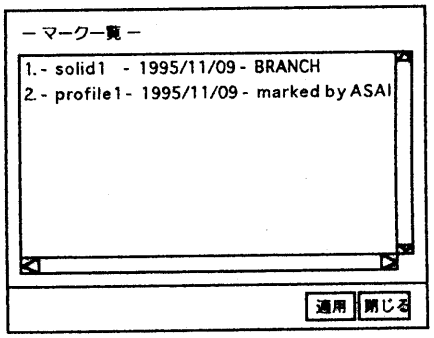

図7　情報一覧ウィンドウ

6　おわりに

本稿では、各設計プロセスにおいて実行されたコマンド列、プロセスの前後での形状データの状態遷移情報を記録することで、形状の表現形式に依存しない設計履歴の記録手法を提案した。本手法は、当社の意匠設計用ＣＡＤシステム「デザインモデラ」上において履歴操作機能として実現され、既に全事業部のデザイン部門にて試用稼働している。

今後の課題としては、ユーザーに提示するコマンド情報が文字情報だけなので、イメージ情報等を付加してユーザーに親切なインターフェースを実現して行きたい。

謝辞

最後に本開発を進めるにあたり、ご協力を頂いた各事業部デザインＣＡＤ運用グループの方々に、また終始助言、討論をして頂きましたデザインＣＡＤ開発グループの方々に深く感謝いたします。

参考文献

(1)乙井克也、二上範之、小堀研一：”ソリッドモデラに於ける履歴操作機能の開発”、情報処理学会第36回全国大会、1988

(2)H.Toriya, T.Satoh, K.Ueda, and H.Chiyokura : "UNDO and REDO Operations for Solid Modeling", IEEE Computer Graphics and Applications, 6(4):35-42, 1986