ソフトウェア工学 64－18
（１９８９．２．３）

直接操作による情報提示システム開発環境について

大戸英隆　辻野嘉宏　都倉信樹

大阪大学基礎工学部情報工学科

マルチメディアデータをユーザからの要求，または必要に応じてリアルタイムに提示するような機能を持つソフトウェアをＭＭＰＳ（Multi-Media Presentation System）と呼ぶ．近年，ＭＭＰＳによってプレゼンテーションを行う要求が増しているが，プログラミングの知識のない者には困難である．

本報告では，ノン・プログラマによるＭＭＰＳの開発を容易にするため，直接操作インタフェースを用いて，ＭＭＰＳを作成，編集し，実行することができる環境であるＭＭＰＳ開発環境の試作について述べる．

A Multi-Media Presentation Systems Development Environment Based on Direct Manipulation Interface

Hidetaka OHTO, Yoshihiro TSUJINO and Nobuki TOKURA

Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University 1-1, Machikaneyama, Toyonaka, Osaka, 560 Japan

Recent advances in hardware technology have made it echonomically feasible to handle multi-media data such as a text, graphics, audio and video. We call a real time presentation system with multi-media data an Multi-Media Presentation System (MMPS).

In order to development an MMPS easily, we propose an MMPS development environment based on direct manipulation interface.

In this environment, even a non-programmer can easily construct, edit and execute an interactive MMPS such as CAI systems, guide systems, and so on.

１．　まえがき

最近の計算機のハードウェア技術の進歩は著しく，以前は困難であった画像，音声などのマルチメディア・データをパーソナルコンピュータで扱えるようになってきた．マルチメディア・データの有効利用を行う分野として，観光案内などの情報サービスの分野や研究発表などを行うといったプレゼンテーションの分野，ＣＡＩなどの分野などが考えられる．

これらの分野で利用されるソフトウェアに共通して要求される機能は「マルチメディア・データをユーザからの要求，または必要に応じてリアルタイムに提示すること」である．このような機能を持つソフトウェアを，ここでは，マルチメディア・プレゼンテーション・システム（ＭＭＰＳ＝Multi-Media Presentation System）と呼ぶ．

将来，パーソナルコンピュータ上でＭＭＰＳを実行することが容易になってくると，パーソナルコンピュータを使用している一般のビジネスマンが営業用の資料として用いたり，計算機以外の分野の研究者が研究発表用の資料として用いたりするなど，プログラミングの知識のないエンド・ユーザ自身が自分でＭＭＰＳを作成したいという要求がでてくるものと思われる．ところが，現在のプログラミング言語による記述を前提とする開発環境では，エンド・ユーザによるＭＭＰＳの開発は困難である．この解決手段として，ＭＭＰＳ記述用の言語の開発[1][2][3]や，プレゼンテーションの分野では，プレゼンテーション資料作成用のシステムの開発などに関する研究が行われている．

本報告では，プログラミングの知識がない者（ノン・プログラマ）でも容易にＭＭＰＳを作成できるようにするため，直接操作インタフェース[6][7]を取り入れて設計したＭＭＰＳ開発環境について述べる．

以下，２節ではＭＭＰＳの利用例を挙げてＭＭＰＳの有用性について述べ，ＭＭＰＳの実現に必要とされる機能について考察する．次に，３節ではシナリオ作成者から見た演示の流れのモデルとしてのシナリオ・モデルについて述べる．４節ではＭＭＰＳ作成のために必要と考えられる機能，枠組みについて考察し，それらを取り入れ，直接操作によりＭＭＰＳを容易に開発する開発環境の試作について述べる．

２．　ＭＭＰＳの概要

ＭＭＰＳを用いてプレゼンテーションを行う場合の例を挙げ，それらを観察して，ＭＭＰＳに必要とされる機能について考察を行う．

２．１　ＭＭＰＳの例

ＭＭＰＳを用いてプレゼンテーションを行う場合に必要とされる機能を検討するために，いくつかの応用について検討を行った．ここではその例として英会話教育にＭＭＰＳを用いたと想定した場合を考える．

様々な場面での英会話例を動画，音声を用いて示し，次の場面をいくつかある選択枝の中から選択することにより，学習者は好きな場面を選んで英会話を学習するとする．英会話教育にＭＭＰＳを使用する際備わっていると良いと思われる機能としては，①会話の例を示す際に，音声だけでなく動画を用いて臨場感を増す，②ユーザが返答したい内容によって複数の次の状況を用意しておく，③次の場面の指定をマウスクリックなどの簡単な操作で行える，④学習者が間違った箇所があったらそこまで戻って繰り返し学習する，⑤予め時間切れを設定しておき，時間内に答えられないときは答えを教え，また，もう一度同じ所を繰り返し学習する，などがある．

２．２　ＭＭＰＳに必要とされる機能

２．１で述べたようなＭＭＰＳを考えた場合，ＭＭＰＳが共通して持つ機能として，以下のようなものが考えられる．

（１）テキスト，静止画，動画，音声などのマルチメディアデータを組み合わせて同期をとり，同時に生成，消去することにより，各場面を構成する機能．

（２）場面の切り替えなどの動作を行うきっかけとして，マウス入力，キーボード入力などのユーザ入力，時間の経過，動画，音声データが終了したなどの個々のデータの提示状態の変化を検出する機能．

（３）テキスト，静止画データのスクロール，動画の早送り，巻戻しなど個々のデータに対する操作

３．　シナリオ・モデル

ＭＭＰＳに対して行った観察とそれに対する考察に基づき，ＭＭＰＳが持つべき必要十分な機能を包括したＭＭＰＳのモデルの作成を行った．このモデルを「シナリオ・モデル」と呼ぶ．シナリオ・モデル作成の基本思想はシナリオ作成者からみたＭＭＰＳの動作をできる限り自然に表現することである．実際にＭＭＰＳを作成しようとした場合，特にノン・プログラマであれば思考の手がかりを得ることも容易ではないが，シナリオ・モデルを提示することで，ＭＭＰＳの動作を行わせる際の筋書き（シナリオ）に対する思考の枠組みを与えることができる．

３．１　シナリオ・モデルにおける階層

（１）シーン　　個々のマルチメディアデータ（静止画，動画，音声，テキスト）に対する操作を記述する際に，データを特定するため，個々のデータに対応する概念を定め，これをシーンと呼ぶ．

（２）シナリオ　ＭＭＰＳの制御構造を持つまとまった演示の単位をシナリオと呼ぶ．シナリオは１つの部品として扱うことができ，他のシナリオ中に含まれたり，他のシナリオを含んだりすることができる．このことにより，かなり大規模なシステムも構造化して作成することができる．またシナリオを部品としてブラウザに登録して，そのまま，あるいはテンプレートとして再利用する

ことができる．１つのシナリオ中に含まれているシナリオをそのサブシナリオと呼ぶ．

シナリオ・モデルにおいて１つのシナリオはいくつかのサブシナリオを内部頂点とし，いくつかのシーンを葉とする木構造をとる（図１）．

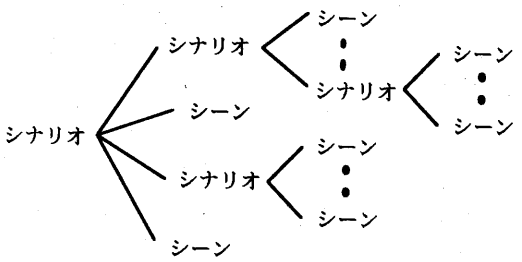

図１．シナリオの階層構造

３．２　シナリオ・モデルにおける制御構造

通常のプログラミング言語を用いてＭＭＰＳを作成する場合，ＭＭＰＳの動作は，逐次的な動作の繰り返し，分岐によって記述されるが，本来，ユーザから見たＭＭＰＳは，シーンが次々と入れ替わっていくように見えるシステムであり，その切り替えなどの動作　（アクション）はユーザのマウスクリックなど，ユーザも認識している何らかの事象（イベント）をきっかけにして行われると考えられる．このように，ＭＭＰＳはイベント駆動型の動作形態をとると考えるのが自然である．ここで，複数個のイベントの発生が起こるまでアクションの実行を待ち合わせるという同期をとる記述も，例えば，動画の提示が終わっていて，かつ，ユーザからの入力が行われたらアクションを実行する，という記述を行いたいときなどに有用である．また，１つのイベントによって複数のアクションを起こしたいという要求も考えられる．そこで，１つ以上のイベントの集合とそれをきっかけとして引き起こされる１つ以上のアクションの集合の対をイベントアクション対（図２）と呼ぶとすると，ＭＭＰＳの動作全体（シナリオ）は，イベントアクション対の集合で表現できる．

（イベント１、イベント２,...,イベントｎ）

↓

（アクション１、アクション２,...,アクションｍ）

図２．イベントアクション対

４．ＭＭＰＳ開発環境の設計

４．１　ＭＭＰＳ開発環境の設計の基本方針

本来，ＭＭＰＳ開発環境はノン・プログラマの使用を前提としているので，良好なユーザインタフェースは不可欠である．

また，良好なインタフェースを提供するプログラミングスタイルの一手法として，

（１）複雑なシンタクスのコマンド言語でなく，実際にオブジェクトに働きかける動作やメニューの選択により入力する，

（２）画面を見ることによって何が起こっているかを理解できる，

（３）興味の対象である世界に直接働きかけ，その世界を直接眺めている感じを持つことができる，

といった利点を持つ直接操作インタフェースがある．

そこで，ＭＭＰＳ開発環境においてはノン・プログラマによるプログラミングスタイルの一手法としての直接操作インタフェースを，ＭＭＰＳの作成，編集の各段階における操作に取り入れる事を第一の方針とする．

また，ＭＭＰＳはきめ細かい演出効果を求められるシステムであり，画面レイアウトなどの実行時の様子を確かめながら作業を進める必要があるが，通常の開発環境ではプログラムの作成中に実行時の様子を確かめることができない．そこで，ＭＭＰＳ開発環境は，現在，全制御構造中のどのシナリオ上のどのシーンを実行中であるかの実行状況を動的，視覚的に把握できる機能を備える必要がある．この様な，シナリオの動的な可視化を行う機能を取り入れることがＭＭＰＳ開発環境の第二の方針である．

４．２　直接操作インタフェース

４．２．１　基本方針

４．１の第一方針であるシナリオモデルに基づく直接操作によるＭＭＰＳ開発環境の基本的なアイディアは以下の通りである（図３）．

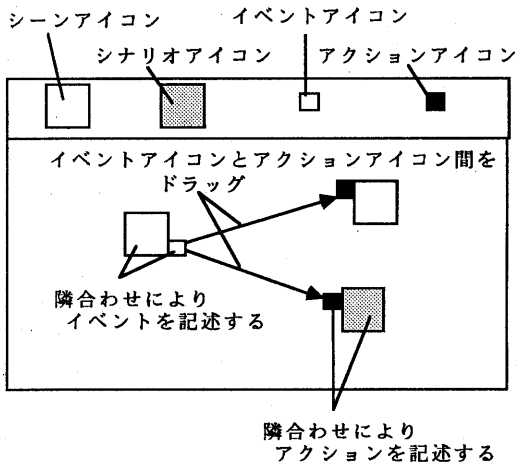

図３．ＭＭＰＳ開発環境の基本アイディア

（１）ＭＭＰＳ開発環境上でのシナリオ，シーン，イベント，アクションの表現としてシナリオアイコン，シーンアイコン，イベントアイコン，アクションアイコンを用い，属性，その他の設定の対象の指定に名前を用いず，画面上のアイコンを直接ポイントすることで指定する．
（２）シーン（シナリオ）に対するイベント（アクション）の定義はシーン（シナリオ）アイコンにイベント（アクション）アイコンを隣合わせるという動作によって行う．
（３）イベント駆動型の動作関係（イベントアクション対）の定義はイベントアイコンとアクションアイコン間をドラッグするという動作によって行う．画面上にイベントアクション対を表す有向辺がイベントアイコンとアクションアイコンの間に付け加えられる．

このような方針をとることで，直接操作の利点を活かし，編集作業はイベント，アクションの対応関係を表すグラフを作図ツールで描くような使用感で行える．また，イベント（アクション）アイコンの導入により，イベント（アクション）を各シーン（シナリオ）の属性とせず，オブジェクト（画面上に表示される，ユーザまたはシステムの操作の対象物）として扱うことにより同一の設定を持つイベント（アクション）は同一のオブジェクトとしてシステム内で一意に定まる．

直接操作インタフェースは，ノン・プログラマに対して良好なインタフェースを与える反面，ＭＭＰＳ開発などのプログラミング作業をすべて直接操作によって行うとするといくつかの問題点が生じてくる．４．２．２にそのような問題点を挙げて，さらにその解決方法について考察する．

４．２．２　問題点と解決策

直接操作インタフェースは，ノン・プログラマに対して良好なインタフェースを与える反面，以下のような問題点（あるいは疑問点）が考えられる．

（１）大規模なシステムを記述できるか．

（２）十分な記述能力を画面を煩雑にせずに提供できるか．

（３）記述方法　－　アイコンデザイン，階層化の際の各階層間のインタフェース

以下に，それぞれの問題点の解決方法について考察する．

（１）大規模なシステムを記述できるか．

直接操作によってプログラムを記述する場合，記述されるプログラムが大規模であれば（あるいはそれほど大規模でなくともそうなるかもしれないが）１画面上に納まらない場合が生じる．そのようなプログラムを画面に表示し，プログラミングを行わせる方法として以下のようなものが考えられる．

①マルチウィンドウを用いる．
②画面をスクロールする．
③拡大・縮小を行う．

シナリオ・モデルに基づくＭＭＰＳの作成，編集を支援する方式として，シナリオ（論理的なプログラムの単位）を１つのウィンドウに対応させ，そのウィンドウ内でオブジェクトを表示し編集を行う，マルチウィンドウの使用はシナリオ作成者の自然な理解にとって有効であると考えられる．このとき，１つのシナリオが１つのウィンドウ上に納まらない場合も考えられ，その場合への対処法として②，③が考えられる．②の画面スクロールはマルチウィンドウには一般的な付属機能であるが，直接操作によるプログラム開発環境においては，興味（操作）の対象とすべきオブジェクトが画面上に見えていない場合がある事はシナリオ作成者の記憶に負担を負わせるし，また，オブジェクトが現在の表示位置より大きく離れていることは操作時に煩わしい，という欠点を持つ．但し，１つのシナリオの論理的な構成が大規模で平板な場合には画面スクロールの機能も必要となる．③の，縮小して全体の構成を見て，拡大して実際にオブジェクトに対して操作を行う方式では，縮小した場合には各オブジェクトの識別が困難になる欠点がある．

ＭＭＰＳ開発環境においては，最も使用頻度の高い①のみを採用するが，①から③は互いに利点，欠点を持つ方法であるため，特定の方式に固定するよりも，シナリオ作成者のその場での用途に合わせた方式を選択できることが望ましい．

（２）十分な記述能力を画面を煩雑にせずに提供できるか．

直接操作を行う場合，操作の対象はオブジェクトとして画面上に表示しなければならない．そこで，画面を煩雑にせずに記述能力を向上させるオブジェクトを導入しなければならない．

そのような方法として以下のようなものが考えられる．

①特殊シーンの導入．
②並列な動作，同期の記述．

①特殊シーンの導入．

シナリオ作成者が理解，操作を簡単に行え，記述量を爆発させないでおくことができる表現で，ＭＭＰＳを記述する際に必要な記述能力を，システムが提供する必要がある．このための方法として，ある程度の機能をすでに包含したプリミティブをシステム側がライブラリとして提供することが挙げられる．このようなプリミティブの例としては，回数をカウントするカウンタ，時間の経過を計るタイマ，ユーザからの入力を受け付けて画面にエコーバックする入力エコーなどがある．これらのプリミティブはイベントとアクションというインタフェースを持つオブジェクトと考えることができ，特殊なシーンであると考えることで概念が単純化される．例えば，カウンタであれば「カウンタの持つ値が５を越えた」がイベントであり，「カウンタの値を３にセットする」がアクションである．

②並列な動作，同期の記述．

並列な動作や同期の記述は，アイコンの隣合わせやアイコン間のドラッグといったシナリオグラフへの直接操作によって容易に記述できる．

１つのイベントによって複数個の並列なアクションを起こす（fork）際のシナリオグラフと，複数のイベントの発生を待ち合わせる同期をとって１つのアクションを起こす（and）シナリオグラフの記述例を図４に示す．and条件の場合にはトランジションアイコンというオブジェクトを用いて，複数のイベントアイコンからトランジションアイコンにドラッグをおこなってからトランジションアイコンからアクションアイコンへドラッグを行うことで，複数のイベントを待ち合わせてから全て揃ったところでアクションを実行する動作を記述できる．トランジションアイコンから複数のアクションアイコンへドラッグを行うことで複数個のイベントを待ち合わせて複数個のアクションを並列に行う動作も記述できる．

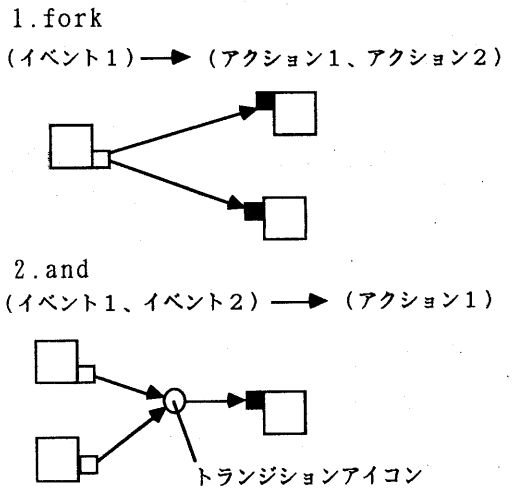

図４．同期の記述

（３）記述方法

アイコンのデザインや階層化の際の各階層間のインタフェースなどのオブジェクトの形状をどのようにしてやればよいかという問題点について考察した．

（３－１）アイコンデザイン

直接操作によるプログラムを行うことを考えれば，シナリオ作成者が作成したシナリオをシナリオ作成者が識別するために，そのシナリオ，シーン，イベント，アクションを表すアイコンを何等かの方法で生成しなければならない．その方法として，

①シナリオ作成者にデザインさせる．

②自動生成．

③システムがシナリオ作成者が使用すると考えられるアイコンを予め用意する．

がある．以下に，それぞれの方法の利害得失について述べる．

①シナリオ作成者にデザインさせる．

シナリオ作成者が，シナリオを作成する度にアイコンを作成していたのでは負担が大きすぎるが，シナリオ作成者にとって最も的確なデザインをそのアイコンに対して与えることができる点で有効である．

②自動生成．

内部の形状（シナリオであれば各シーン間のリンクの形状，シーンであればそのデータ，イベント（アクション）であれば，セットされている値）をミニチュア化してアイコンに埋め込む等の方法により各オブジェクトを識別させる．このような方式により自動的にアイコンを生成することが可能であればシナリオ作成者の負担は軽減されるが，各アイコンの形状が大変類似している場合見分けが付かない懸念がある．この判断には多くの例に当たって実際に検証することが必要である．

③システムが，使用すると考えられるアイコンを予め用意する．

シナリオ作成者の負担は軽減されるが，最適なものが得られることはない．

ＭＭＰＳ開発環境ではシナリオ作成者の負担を軽減する立場から基本的に②の方法を用い，さらにコメントをアイコンに埋め込む形で判別性をあげるとする（図５）．

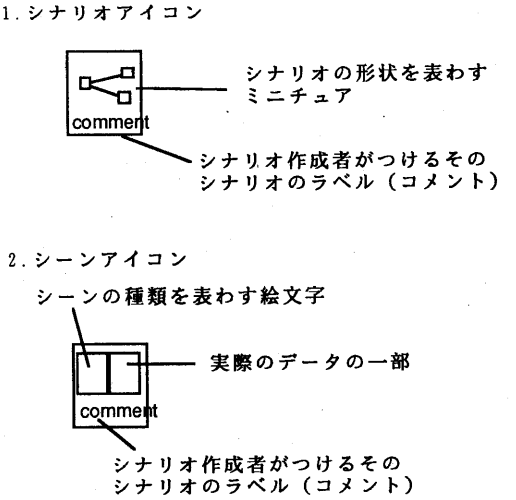

図５．アイコンの自動生成

（３－２）階層化の際の各階層間のインタフェース

シナリオ・モデルの階層関係は木構造でつくられ，各シナリオはサブシナリオを含むことができる．このとき，あるシナリオ内のサブシナリオの外から内への入力（イベント），あるいは内から外への出力（アクション）は，外の世界からみただけでは理解が困難であり，その度毎に中を覗くのでは，直接操作を行う上での障害となりう

る．１出（入）力であるとすると外から内が閉じているため，内を覗かなくてもよいが，外に対する分岐を持つシナリオを作ることができない．これはプログラムの記述能力を大きく低下させる．そこで，多出（入）力にして，かつ，内部と外部のどちらの視点からも容易に理解できるような仕組みが必要である．ＭＭＰＳ開発環境では内部と外部のアイコンの統一によってそれに対処している．

４．３　シナリオの動的な可視化

４．３．１基本方針

ＭＭＰＳはきめ細かい演出効果を求められるシステムであり，画面レイアウトなどの実行時の様子を確かめながら作業を進める必要があるが，通常の開発環境ではプログラムの作成中に実行時の様子を確かめることができない．そこで，ＭＭＰＳ開発環境は，現在，全制御構造中のどのシナリオ上のどのシーンを実行中であるかの実行状況を動的，視覚的に把握できる機能を備える必要がある（シナリオの動的な可視化）．この機能は，修正箇所発見（バグ，または．演出効果向上のための修正）のための強力なツールとなり，テスト，デバッグが容易になる．基本的に上述の直接操作を行った際のグラフで可視化を行えば，

①シナリオ作成者自身が作成したグラフ上で可視化が行われ，連想記憶の効果大，

②実行をトレースしていて，ストップ，すぐにそのオブジェクトを直接操作で編集できる，

等のメリットが考えられる．

４．３．２　問題点と解決策

また，シナリオの動的な可視化に関しての問題点としては，以下のようなものが考えられる．

（１）テスト実行時の実画面をシナリオ作成者に提示しながら，シナリオの動的な実行状況をどう可視化するか．

（２）実行時，異なるシナリオ階層上に複数個並列に存在する，アクティブ・シーンを編集画面上にどう表示するか．

以下に，それを解決するために考えられるいくつかの方法とその利害得失について述べる．

（１）実行時の実画面をシナリオ作成者に提示しながら，シナリオの動的な実行状況をどう可視化するか．

編集時のシナリオグラフ表示画面（シナリオ画面）と実際の演示画面（プレゼンテーション画面）をキー入力などにより画面切り替えで表示する方法ではシナリオ画面に表示されるシナリオグラフの動的な変化を見逃す．シナリオ画面上にプレゼンテーション画面をポップアップすることはプレゼンテーション画面がマルチウィンドウ表示で縮小を行ったとしてもかなり大きくなり，不可能であるか，可能であったとしても演出効果の確認作業を行う際にプレゼンテーション画面が邪魔になり望ましくない．それらの問題点の解決方法としてＭＭＰＳ開発環境では２つのディスプレイを用いてそれぞれをシナリオ画面とプレゼンテーション画面に割り当てる方法を採用する（図６）．この方法は，使用にかかるコストは大であるが，グラフ上での動的変化を実画面と密接に対応付けられる効果も大きい．また，編集時に実画面でしか編集できない，画面表示位置の編集や，マウスイベント領域の設定にも有効に利用できる．

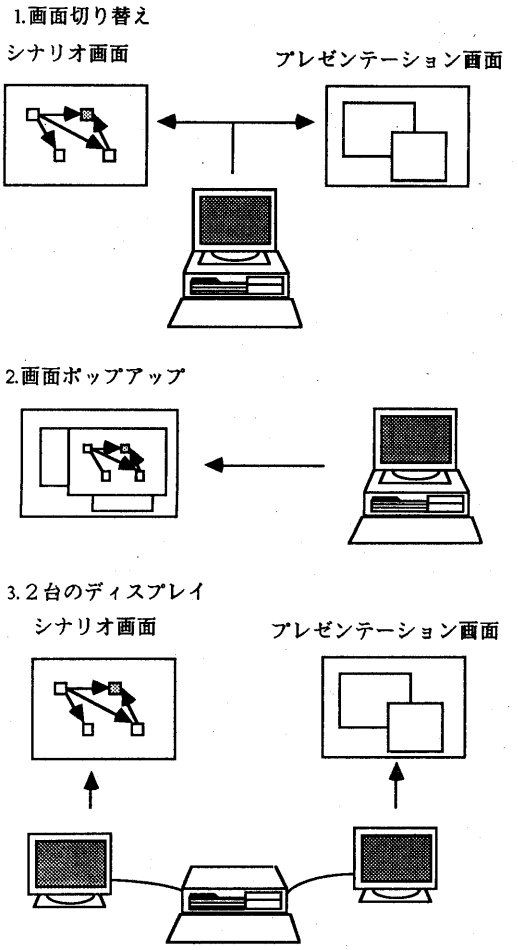

図６．シナリオ，プレゼンテーション画面の表示方法

（２）テスト実行時，異なるシナリオ階層上に複数個並列に存在する，アクティブシーンを編集画面上にどう表示するか．

１つのシナリオ階層，例えばトップレベルの階層に固定して表示する場合，深い階層で起こっている実行状況が不明になる．また，アクティブになった（アクティブシーンを持つ）カレントのシナリオ階層を表示する場合，並列な実行が考えられ，カレントが一意に決まらないし，全体の実行状況が把握できない．シナリオの動的な可視化を行うためには，イベントの発生とそれにともなうア

クションの実行を見せることが重要であるため，現在アクティブなシーンのあるシナリオ階層は原則として見せる必要があると考えられ，マルチウィンドウを用いた表示を行う．このとき，単に，アクティブになったシナリオ・ウィンドウをポップアップしていくだけでは下敷になって見えなくなるシナリオができる可能性がある．この解決方法としてＭＭＰＳ開発環境では，

①シナリオ作成者にできるだけ重ならないよう，あるいは自分の理解し易いようにレイアウトさせる．

②システムがシナリオ検索を支援する機能を提供する．をとる．

②には具体的には，

a)シナリオ階層の上下，オープン，クローズ

シナリオ作成者にとって現在の興味の対象である（でない）シナリオ・ウィンドウをオープン（クローズ）する．

b)現在アクティブなシーンアイコンの表示

グラフのオープン，クローズが可能なので，アクティブでない（アクティブシーンを包含しない）シナリオが表示されていたり，アクティブなものがクローズされている場合があり，アクティブなシナリオの一覧がみたい要求が考えられる．

以下，アクティブシーンの表示方法とその利害得失について述べる．

b-1)プルダウンメニュー

シナリオアイコンとその中に含まれているアクティブなシーンアイコンの組のリストを表示

（利点）一覧して目的のシナリオをすぐ選択することができる．

（欠点）シナリオ作成者は編集時のシナリオグラフの形状とシナリオアイコンとを合わせて記憶しているので，シナリオアイコンのみを切り放して表示した場合，識別が困難となる．

b-2)ミニチュアメニュー

ウィンドウの重なりをとり，全てのアクティブなシナリオ・ウィンドウを縮小して並べる．

（利点）ウィンドウの重なりをとることで，全てを同時に一覧することができる．

（欠点）編集時と操作対象が異なる，個々のアイコンの識別が困難となる．

b-3)シナリオ・ウィンドウの重なる順番を換えてアクティブシーンを見えるようにする．

（利点）編集時とまったく同じグラフを（レイアウトを含めて）操作対象とし，シナリオ作成者の連想記憶の効果が大．

（欠点）必ず見えるようにできるとは限らない．

b-4)ウィンドウのレイアウトを上下左右にずらしてアクティブなシーンアイコンを表示する．

（利点）同じグラフが操作対象

（欠点）必ず見えるようにできるとは限らないし，レイアウトがシナリオ作成者自身の行ったものと変えられてしまう．

現在アクティブなシーンアイコンが見えるようにすることのみを重視したアルゴリズムはシナリオ作成者の連想記憶を破壊してしまう．その解決策としては

①ずらす幅をあまり大きくしない．

②システムは重なりの検出のみを行い，シナリオ作成者に判断を委ねる．

③レイアウトを要求に応じて元に戻せるようにする．

などがある．

b-5)アクティブシーンのあるグラフの枠を透けて表示させる．

（利点）同じグラフが操作対象で，レイアウト不変．全てのアクティブなシーンアイコンをシナリオ作成者が認識できる．

（欠点）たくさん重なっている場合，表示が複雑になる場合がある．

その他，b-5の変形として

・枠のみでなく，シナリオ・ウィンドウ内のオブジェクトも薄く描いて透けてみせる．

・カットモデル．

等が考えられるが，いずれもb-5の欠点を継承している．

ＭＭＰＳ開発環境においては，必ず見えるようにできるとは限らないが，編集時とまったく同じグラフを（レイアウトを含めて）操作対象とし，シナリオ作成者の連想記憶の効果が大であるb-3の方法を基本的に採用し，見えるようにできなかったものについてはその存在をシナリオ作成者に警告する．

c)シナリオ作成者がマーク付けしたシナリオ・ウィンドウをアクティブか否かに関わらず表示．

d)ヒストリ

イベントアクション対の実行を１ステップと考えてそのときのアクティブシーンを記憶しておき，要求に応じて，過去のステップの起きたアクティブなシナリオ・ウィンドウをシナリオ画面上に表示する．

d-1)ボタンをクリックして，過去の状態を１ステップずつ遡って，そのときイベントアクション対の実行されたシナリオ・ウィンドウをシナリオ画面上に表示する．

（利点）時間の遡りをシナリオ作成者に認識させられる．

（欠点）シナリオ作成者が表示したいイベントアクション対の実行されたシナリオ・ウィンドウにすぐにたどり着けない．

d-2)スライドボタン

（利点）時間の遡りをシナリオ作成者が認識でき，シナリオ作成者が表示したいイベントアクション対の実行されたシナリオ・ウィンドウへもすぐたどり着ける．

その他，b)と同様のプルダウンメニュー，ミニチュアメニュー等の方式も考えられる．但しこの方式をヒストリに取り入れる場合には時間の概念の表現を番号づけなどの方法で行う必要がある．

ＭＭＰＳ開発環境においては時間の遡りをシナリオ作成者が認識でき，シナリオ作成者が表示したいイベントアクション対の実行されたシナリオ・ウィンドウへもすぐたどり着けるd-2の方法を採用する．

e)プレゼンテーション画面上に表示されている静止画，動画などを直接指し示すことによって指示を行い，それに対応するシーンアイコンをシナリオ画面上に表示する．

５．あとがき

ＭＭＰＳ開発環境を，ノンプログラマにも容易に使用できる環境とするための方法として，

（１） ＭＭＰＳの動作をユーザから見たときのイメージに基づいたモデル（シナリオ・モデル）を提案し，このモデルに基づいた直接操作でＭＭＰＳの作成，編集を行う，

（２）ＭＭＰＳのシナリオの実行時の様子を動的に可視化し，その様子を確認しながら，ＭＭＰＳの編集と実行をインタラクティブに行うことができる，

などの特徴を持っており，シナリオモデルに基づいたＭＭＰＳの作成，編集，実行を支援する．

今後の課題としては，

（１）プロトタイプの使用感の評価，検討を行った上で，さらに細かな使い勝手の向上を目指す，

（２）音声，動画データなどを扱えるシステムにする，

などがある．

文献

［１］北風，宮下，笠原：”プレゼンテーションシステムＡＣＴＯＲⅡ － 人間の思考過程を考慮したユーザ・インタフェース －”，情報処理学会３２回全国大会 Ⅳ－５ (1986)．

［２］丸山，北風，宮下，笠原：”プレゼンテーションシステムＡＣＴＯＲⅡ － シーン作成方式 －”，情報処理学会３２回全国大会 Ⅳ－６ (1986)．

［３］小山博生：”マルチメディア演示のためのシナリオ記述言語に関する研究”，大阪大学大学院基礎工学研究科修士学位論文(Feb 1985)．

［４］田中，辻野，都倉：”マルチメディア表示システム記述用言語の設計とその処理系の実現”，ソフトウエア工学研究会資料86-47，(May 1986)．

［５］大戸，辻野，都倉：”イベント駆動に基づく情報提示支援系について”，ソフトウェア工学研究会資料88-59，(May 1988)．

［６］Norman, D.A., Draper, S.W: ”USER CENTERED SYSTEM DESIGN”, LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS (1986).

［７］淵，古川，溝口：”インタフェースの科学”，共立出版，(1987)．