ソフトウェア工学 90－3
（１９９３．２．４）

リアクティブ データフローモデル：ソフトウェア設計ドキュメントの表示過程の記述と再現

福田 晴元　高橋 直久

NTT ソフトウェア研究所

本稿では，ソフトウェア設計ドキュメントの分析を容易にするため，ドキュメントの分析過程を記述し再現する機構を持つ，リアクティブ・データフローモデルと呼ぶドキュメント表示制御モデルを提案する．本モデルでは，データフロー計算モデルに基づき，ドキュメントの表示過程を並列プログラムとして記述し，プログラムを実行することによりドキュメントの非同期並列表示を行なう．さらに，リアクティブ機構により，ユーザとのインタラクションに応じた表示処理を行なう．ドキュメントをクラス，名前，型により定義したオブジェクトモデルにより，表示過程がツールおよびデータから独立したプログラムとして記述される．

Reactive Data Flow Model: Description and Replay of a Browzing Process for Software Design Documents

Harumoto FUKUDA　Naohisa TAKAHASHI

NTT Software Laboratories

This paper presents a document browsing model, called a reactive data flow model, to facilitate the analysis of source codes and design documents in software re-engineering. In this model, a browsing process, where a sequence of documents are displayed on windows of a terminal, can be described as a data flow program and be replayed in parallel based on a data flow computation model. It can contain adaptive behavior which is specified by a user during its replay because reactive control mechanisms are embedded in this model. A document access mechanism which uses a triple of class, name and type to operate a document makes the description of a browsing process independent of both the contents of documents and the kinds of document editors.

1 はじめに

ソフトウェアの開発では，ソースコードをはじめ，様々な技術文書，テスト結果などの生産物が，各種ツールにより作り出される．ソフトウェアの改造では，これらの生産物（本稿では「ドキュメント」と呼ぶ）の中から必要なドキュメントを捜し出し，その内容を理解することにより，既存システムの機能を理解し，変更の影響を考察する，ドキュメントの分析過程が重要である．

ドキュメントの検索・分析を容易にするため，ハイパテキスト・システムがソフトウェア開発支援環境でも利用されている[1)2)3)]．ドキュメントの開発時，あるいは，分析時に，ソースコードの断片に対して，その理解に必要なドキュメントへのリンクを付与しておけば，リンクを辿ることにより関連するドキュメントを順に読み進めることができる．しかし，リンクの辿り方は，ユーザに任されているため，開発者や分析者の意図が十分に伝えられず，ユーザは試行錯誤を繰り返してしまうという問題が生じる．一方，従来の多くのオーサリングシステムと同様に，シナリオとして表示順にドキュメントを列挙しておく方法では，ドキュメントの数が多い場合には，分析過程の把握や変更が難しくなる．

このような問題を解決するため，ハイパテキストのリンクの辿り方をペトリネットで表現し，表示順序を制御する方式[4)]が提案されている．この方式では，ペトリネットの意味規則に従い，リンクが辿られ，ドキュメントが非同期並列に表示される．また，ペトリネットの性質を用いて，リンクの到達可能性解析を行ない，表示過程の誤りの検出作業を助けることができる．しかし，この方式では，ペトリネットは並列実行動作の性質を求めるためのモデルであるので記述性が低い，ユーザとのインタラクションにより適応的に表示順序を変化させることができない，ドキュメントへのアクセス機構について考慮されていないという問題が残されている．

本稿では，上記の問題を解決するため，ハイパテキストに対して，データフロー計算モデルに基づいてリンクの辿り方に関する意味付け法を与え，構造的な並列プログラムとしてドキュメントの表示過程を表現する，リアクティブ・データフローモデルと呼ぶ，ドキュメント並行表示制御モデルを提案する．このモデルでは，設計ドキュメントの分析過程をドキュメントの表示順序制御を行なうデータフロー型並列プログラムの実行過程としてモデル化し，プログラムの再実行により分析過程を再現する機構を与える．また，リアクティブ制御機能[5)6)]により，ユーザが置かれた局面に応じて再現法を変化させる機構を与える．さらに，ドキュメントをクラス，名前，型をもつオブジェクトとしてモデル化し，オブジェクト指向プログラミングにより，ドキュメントの表示ツールやドキュメントの内容に依存しない記述を可能にする，ドキュメントアクセス機構を与える．

本稿はつぎのような構成になっている．まず2章で，ドキュメント表示過程をモデル化し，リアクテイブ・データフローモデルの目標と概要を述べる．次に，このモデルの特長である，データフロー実行モデルによる並行表示制御，リアクティブ機構による適応型表示制御をドキュメント・オブジェクトによるドキュメントアクセス機構を3章～5章で明らかにする．さらに，6章で，このモデルに基づき，ワークステーション上に作成中のソフトウェア設計ドキュメント・ブラウザの実験システムについて述べ，簡単な実験例を示す．

2 ドキュメント表示過程のモデル化

2.1 ドキュメント表示過程の記述と再現

マルチウインドウシステムを用いてソフトウエアの分析作業を行なう場合には，関連するドキュメントを同じ画面上の複数のウインドウに同時に表示させて，それらのウインドウを交互にみながら分析を行なうことができる．また，あるウインドウを見て，その内容を理解するために別のドキュメントが必要であると気がつくと，もう1つのウインドウを開いて表示することもある．このようなドキュメントの表示過程は，複数のウインドウを非同期並列に開閉し，ドキュメントの表示・編集ツールを並列に起動するプログラムの実行過程としてモデル化される．すなわち，図1に示すように，ドキュメントの表示過程を並列プログラムとして記述しておき，そのプログラムを実行させることにより表示過程を再現する．

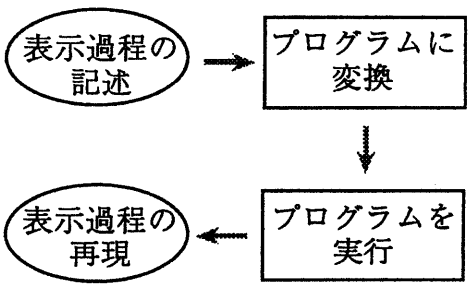

図 1: 表示過程の記述と再現

2.2 目標

本稿では，ドキュメントの表示過程を並列プログラムの実行過程としてモデル化し，次のような性質をもつ表示過程の記述と再現法を実現することを目標とする．

1. 表示過程の記述性
   表示過程の記述を容易にするため，ハイパテキストと同様にドキュメント相互の局所的な関係を記述することにより，表示過程全体を記述できるようにす

る．さらに，繰り返し構造や手続き呼び出しなどを用いて構造的に表示過程を記述できるようにする．

2. 表示過程の再現性
記述された表示過程を忠実に再現する．特に，ドキュメントの同時表示，および，順序付け表示を正確に再現し，プログラムとその仕様書など，強い関連を持つ複数ドキュメントを，同期的に表示，あるいは，閉じることができるようにする．

3. ユーザへの適応性
開発者が，あるドキュメントを見た時，その経験・知識，おかれた状況，用途（目的）に応じて，次に必要なドキュメントが異なる場合がある．このため，ユーザとのインタラクションにより適応的に再現法を変化させられるようにする．

4. ツール独立性
ソフトウェア設計ドキュメントで使用される図表類は，それぞれ専用ツールで表示編集される場合が多い．表示過程の記述と再現の際に，これらのツールの違いによる影響ができるだけ現れないようにすることにより，表示過程の記述，および，再現系の実現／機能変更を容易にする．

5. データ独立性
ソフトウェア設計ドキュメントの分析では，分析対象となるソフトウェアが異なっても，類似の分析過程を用いることができる場合が多い．たとえば，ソースコードを理解したい場合には，そのコード，機能説明，インタフェース，実行例などがあれば，参考になる．このため，データ（ドキュメント）の内容に依存せずに表示過程を記述できるようにすることにより，同じ表示過程を多くのドキュメントに対して適用できるようにする．

2.3 リアクティブ・データフローモデル

本稿で提案するリアクティブ・データフローモデルは，前節に示した目標の実現を目指して作成したドキュメント並行表示制御モデルであり，以下の特長をもつ．

- データフロー型実行モデルによる並行表示制御
ドキュメントの表示順序制御を行なうデータフロー型並列プログラムの実行過程としてドキュメントの分析過程をモデル化し，プログラムの再実行により分析過程を再現する機構を与える．

- リアクティブ機構による適応型表示制御
データフロー計算モデルにリアクィブ機構を導入しユーザとのインタラクションにより適応的に再現法を変化させることを可能にする．

- ドキュメント・オブジェクトによるドキュメントアクセス機構
ドキュメントを構造化し，クラス，名前，型をもつオブジェクトとしてモデル化する．オブジェクトの型を用いて抽象度の高い形式で表示過程（抽象表示過程と呼ぶ）を記述し，実行時にオブジェクトの名前を抽象表示過程と結び付けて具体的な表示過程を得ることによりデータ独立性を高める．また，オブジェクト指向プログラミングを用いて，実行時にオブジェクトに対する操作をクラスに結び付けることによりツール独立性を高める．

このモデルでは，ドキュメントの表示過程を忠実に再現する機構とユーザとのインタラクションにより適応的に再現法を変化させるリアクティブ機構とを統一的で簡明な制御モデルで実現することを目指している．さらに，データフロー計算モデルにおける順序制御，非同期並列制御，手続き呼び出しなどの制御機構，および，ドキュメント・オブジェクトによるドキュメント・アクセス機構により，ドキュメントの分析過程が簡潔に表現できるようにすること，および，分析過程の表現の誤りの検出を容易にすることを目標にしている．上記項目について，それぞれ以下の各章で述べる．

3 データフロー型実行モデルによる並行表示制御

並列プログラムの実行動作をデータフロープログラムグラフとして記録し，そのグラフをデータフロー計算モデルに基づいて実行させると，プログラムの非同期並列動作を再現することができる．本章では，ドキュメントの表示過程を並列プログラムの実行過程とみなして，データフロー計算モデルに基づいて表示過程を記述し，再現する手法について述べる．

3.1 データフロー計算モデル

3.1.1 データフロー・グラフ

データフロー計算モデル[8]では，プログラムは，ノードとアークから構成されるデータフロー・グラフで記述され，手続き呼びだし，ループなどを用いて，構造的に表される．

アークは，ノード間でトークンと呼ばれるデータを受渡しする．アークで結ばれたノード間は，生成者と消費者の関係がある．すなわち，ノードで生成されたトークンが，アークを伝わって，トークンを消費するノードに渡されることを意味する．

ノードには，演算ノード，制御ノード，分配ノードがあり，それぞれ以下の機能をもつ．

演算ノード　入力アーク上のトークンに対し演算を行ない，結果を出力アーク上に出力する．

制御ノード　データを伝搬するデータアークの他に，データの伝搬を制御する信号を入力するコントロールアークを持つ．このコントロールアークをながれてきた信号に従って，トークンの流れる方向を制御する．制御ノードには，分岐ノード，関数呼び出しノードなどがある．

分配ノード　入力トークンのコピーを作成し，全ての出力アークにトークンを出力する．

3.1.2　実行制御

データフロー・グラフの開始ノードの入力アーク上にトークンを配置することにより，プログラムの実行が開始される．グラフ開始ノードの入力アーク上にトークンを置くことにより，データフロー・プログラムグラフの並列実行制御法として，以下に示すデータ駆動実行制御と，要求駆動実行制御[9]がある．

データ駆動型実行制御　データ駆動型実行制御では，データフローグラフのすべてのノードが次の処理をそれぞれ非同期並列に行なう．

1. 全ての入力アーク上にトークンが到着するまで待機する．
2. トークンが全て揃った後，ノードは与えられた機能を実行する．（これをノードの発火と呼ぶ）．この時，入力アーク上のトークンを取り除く．
3. 実行結果を出力アーク上にトークンとして出力する．

要求駆動型実行制御　要求駆動型実行制御とは，指定されたノードに対して，その値を求めるために必須な計算のみを並列に実行する制御法である．要求駆動型処理では，グラフ上の全てのノードが次の処理をそれぞれ非同期に行なう．

1-1　出力アーク上に要求信号（デマンドという）が到着するのを待つ．

1-2　出力アーク上にデマンドが到着した後，入力アークにデマンドを送出する．

1-3　デマンドを送出した入力アーク上にトークンが全て到着するまで待機する．

2　入力アーク上にトークンが揃った後，ノードを発火する．この時，入力アーク上のトークンを取り除く．

3　演算結果を出力アークに出力する

各ノードが上記のように振舞うと，全体の動作は次のようになる．まず，グラフの出力側から入力側へデマンドが伝搬する．デマンドが開始ノードまで伝わるとデマンド伝搬は終了し，開始ノードが出力アークへトークンを送出する．従って，デマンドを受けとったノードのみが選択的に実行される．

3.1.3　色付きトークンモデル

データフロー計算モデルでは，色付きトークン方式により多重実行を可能としている[7]．例として図 2 を考え

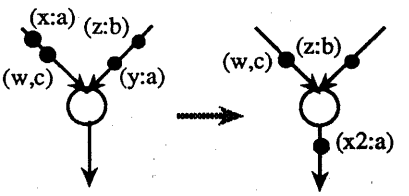

図 2: 色つきトークンモデルによる発火制御例

る．図で (x:a) は，値 x 色 a であるトークンを表す．色が同じトークンが入力アーク上に揃った場合，ノードは発火し，同じ色をもつトークンを出力アーク上に出力する．この状態では，(x:a) (y:a) は発火可能であるが，その他のトークンは色が合致しないため，発火待ちの状態となる．色付きトークンモデルにより，一つのデータフロー・グラフを，カラー別に複数の環境で使用できる．

3.2　ドキュメント表示制御モデル

3.2.1　データフローグラフによるドキュメント表示過程の記述

ドキュメントの分析過程をデータフローグラフで表現する．このグラフを 3.1.2 節に示した実行制御法に基づいて実行することにより，ドキュメントの非同期並列表示を行なう．このグラフのノードは，データフロー・グラフのノードと，ドキュメントノードからなる．アークはドキュメントの表示順序関係を表す．ドキュメント・ノードの「発火」とは，ノードに対応するドキュメントを表示し，編集作業を行なえるようにする手続きを実行することを意味する．発火後にトークンを出力アークに出力する．

3.2.2　ドキュメントの並列表示制御

ドキュメントの並列表示制御法として，データ駆動型実行制御法と，要求駆動型実行制御法を用いる．このた

めに，3.1.2節で示した実行手順の第2項を，次の手順に置き換える．

2 ノードを発火する．この時，入力アーク上のトークンを取り除き，先行ノードに対応するドキュメントを閉じる．

データ駆動型および要求駆動型の実行制御法に基づき，データフローグラフを実行させると，それぞれ次のようにドキュメントの分析過程が再現される．

データ駆動型実行　ドキュメントが先頭のノードから順に並列表示される．

要求駆動型実行　デマンド伝幡により，指定されたドキュメントに対して，到達可能なドキュメントのみが選択される．これは，指定されたドキュメントの説明に必要なドキュメントが選択された事を意味する．この結果，指定されたドキュメントに必要なドキュメントが，選択的に並行表示される．

データフロー型実行モデルに基づきドキュメントの表示制御を実現することにより，次のような利点が得られる．

- 表示過程を構造的に分かり易く表示できる
- 複数のドキュメントが同期的に表示される．
- ドキュメントの表示過程が忠実に再現される．
- 要求駆動実行により，ユーザが指定したドキュメントの説明に必要なドキュメントだけが選択的に表示される．

4　リアクティブ制御

4.1　リアクティブ・データフローグラフ

ユーザとのインタラクションにより再現法を変化させ，ユーザの置かれた局面に応じた表示を行うため，次のようなリアクティブ機構を実現する．

1. ユーザが選択したレベルに応じて表示過程を変更する機能
2. レベルとは無関係に，ユーザの要求により表示過程を変更する機能
3. ユーザの読むスピードに合わせて表示を行う機能

以下に，それぞれの機能の実現法を示す．

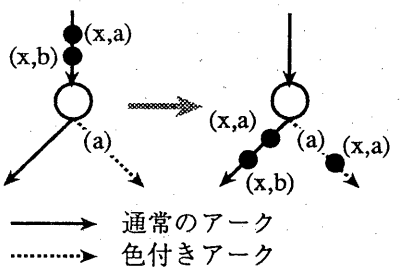

図 3: 色付きアーク

色付きアーク　色付きトークンを方式を利用して，ユーザのレベルに応じて表示過程を選択する機能を実現する．図 3に示すように，ユーザのレベルを表すカラーをトークン，および，アークに付与する．カラーを持つアーク上は，同じカラーを持つトークンのみが伝幡する．これにより，一つの表示過程を使用して，カラー別に複数の経路を選択できるようになる．トークンのカラーをユーザが選択し，カラーをトークンに与えて実行を開始させた場合，ユーザはレベルに応じた表示過程を選択することとなる．

ユーザ依存アーク　ユーザの要求により，表示経路を選択するため，ユーザの要求を受けた時のみトークンを伝幡するユーザ依存アークを実現する．図 4に示すように，アークにユーザ依存アークであることを表すマーク

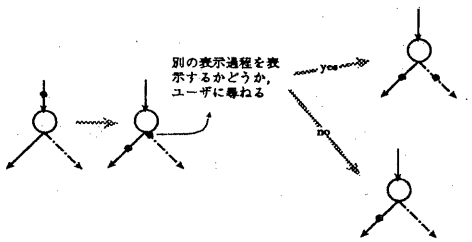

図 4: ユーザ依存アーク

を付ける．トークンがマークの付いたアーク上に到達した際に，グラフの実行機構はトークンを伝幡するかどうかをユーザに尋ねる．ユーザより要求のあった場合トークンを伝幡する．これにより，ユーザの要求があった場合，表示過程を変更できる．

実行開始契機の制御　ドキュメント表示制御モデルは，データフロー・モデルと同様に，入力アーク上にトーク

ンが揃った時に発火可能となる．データフロー・プログラムグラフでは，トークンが揃った時点で発火し，実行が進むのに対して，ドキュメント表示制御モデルでは，ユーザが発火のタイミングを与える．この制御を実現するため，3.1.2節の第2項を以下のように変更した発火規則を用い，ユーザが自分のペースで読み進むことを可能とする．

2-1 入力アークにトークンが揃った後，システムは発火が可能となった旨をユーザに知らせる．

2-2 ユーザより発火要求が来た後，ノードを発火する．この時，入力アーク上のトークンを取り除き，先行ノードに対応するドキュメントを閉じる．発火したノードが，ドキュメントノードの場合，ノードに対応するドキュメントを表示する．

5 ドキュメント・オブジェクト

3，4章では，議論を簡単にするため，表示過程の記述においてノードの実行に必要な情報がすべて与えられるとし，その記述に基づいてノードを「実行」すれば対応するドキュメントが表示されるとした．しかし，2．2節で述べたように，ソフトウェア設計ドキュメントの分析では，分析対象となるソフトウェアが異なっても，類似の分析過程を用いることができる場合が多いので，分析過程が似ている場合には同じ表示過程を使えるように表示過程の記述の際にはできるだけ詳細情報を与えないことが望ましい．また，ソフトウェア設計ドキュメントで使用される図表類は，それぞれ専用ツールで表示編集される場合が多いので，表示過程の記述と再現の際に，これらのツールの違いによる影響ができるだけ現れないようにすることが望ましい．

以上の観点から，ドキュメントを，クラス，名前，型を持つオブジェクト（ドキュメント・オブジェクトと呼ぶ）としてモデル化して，ドキュメント・オブジェクトを用いて抽象度の高い形式で表示過程（抽象表示過程と呼ぶ）を記述し，実行時に詳細情報を与えるドキュメントアクセス機構を実現する．

5.1 ドキュメントのクラス

ドキュメントのデータ構造，および，ドキュメントに対して適用可能な手続きの違いにより，ドキュメントをクラスに分類し管理する．ドキュメントと適用可能な手続き集合の対をドキュメントオブジェクトと呼び，クラスを以下のようにオブジェクト指向計算におけるクラス[10]として実現することにより，表示過程の記述を容易にするとともに，ドキュメントの管理を簡明にする．

ドキュメント・オブジェクトのクラスは階層的に定義し，ドキュメントのデータ構造およびドキュメントへ適用可能な手続き（メソッド）に対して上位クラスから下位クラスへの継承を許す．このため，クラスの定義では，上位クラスのドキュメント・オブジェクトではもたないデータ構造およびメッソドを定義すればよい．また，ドキュメントの表示／編集開始手続き，および，終了手続きをそれぞれOPEN,CLOSEという，すべてのクラスに対して同じ名前のメソッドとして定義する．このようにクラスを定義すると，表示／編集ツール，あるいは，終了時のデータ構造の変換ツールが異なるドキュメントは，それぞれ異なるクラスが割り付けられるが，その開始，終了の起動は図5に例示するように，ドキュメント・オブジェクトにOPEN,CLOSEのメソッドを実行するためのメッセージをおくればよい．

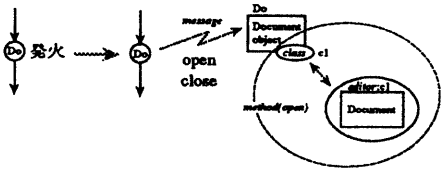

図5: ドキュメント・オブジェクトに対するアクセス制御の例

5.2 ドキュメントの型

ソフトウェアのプロセス・プログラミングでは，プロセスで作られるプロダクトを型づけすることは，プロダクト間での関係を記述する上で重要であると指摘されている[11]．型を導入することにより，プロセスプログラムでは，ソフトウエアツールをある型のプロダクトを別の型のプロダクトに変換する操作とみなされる．

本稿では，ドキュメントの表示過程の抽象度を高めて，一つの表示過程をできるだけ多くのドキュメントに対して適用できるように，ドキュメント・オブジェクトに対して，その記述の種類に応じた型を導入する．さらに，ドキュメント・オブジェクトに対して，その記述対象となっているソフトウェアを表す名前を付与する．この結果，同じソフトウェアに関係のあるドキュメントはすべて同じ名前をもち，同じ用途で作成されたドキュメントは同じ型をもつ．

ドキュメントの分析過程の記述では，図 6(a) に例示するように，ドキュメント・オブジェクトの型を用いて抽象表示過程を記述する．分析過程の再現時に，図 6(b),(c) に例示するように，ドキュメント・オブジェクトの名前を抽象表示過程と結び付けることにより，具体的な表示過程を得る．図で，t_1 ～ t_4 は型を表し，soft1，soft2は，記述対象ソフトウェアの名前を表す．たとえば，t_1 を機能概要，t_2 をソースコード，t_3 を機能の詳細記述，t_4 を実行例とすると，図 6(a) の抽象表

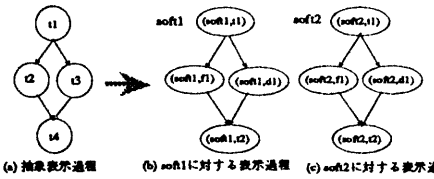

図 6: ドキュメントの型を用いた表示過程の表現と再利用の例

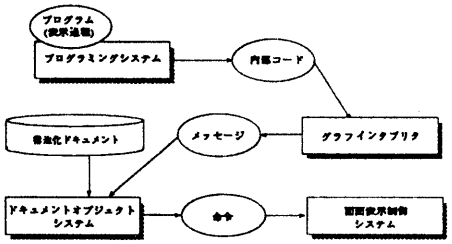

図 7: ブラウザの構成

示過程は，機能概要を表示してから，ソースコードと機能の詳細記述を同時に表示し，最後に実行例を表示することを意味する．また，(b) の表示過程は，ソフトウェア soft1 に対して上記の一連の表示を行なうことを意味する．

6 ソフトウェア設計ドキュメント・ブラウザ

前章までに述べたリアクィブ・データフローモデルに基づき，ワークステーション上にソフトウェア設計ドキュメント・ブラウザの実験システムの作成を進めている．このシステムは，図 7に示すように，プログラミング・システム，グラフインタプリタ，ドキュメントオブジェクトシステム，画面表示制御システムからなる．以下に，各機能の概要を示す．

プログラミングシステム　ユーザに表示過程を作成する環境を提供し，作成された表示過程を内部コードに変換する．

グラフインタプリタ　リアクティブ データフローモデルの実行を行う．ドキュメントを表示するため，表示開始・終了，名前，型のメッセージをドキュメントオブジェクトシステムに送る．

ドキュメントオブジェクトシステム　名前と型で決まるドキュメントを，そのドキュメントの属するクラスのメソッドにより表示開始・終了を行うように，画面表示制御システムに命令を送る．

画面表示制御システム　ウィンドウの状態を管理する．送られた命令に従って，ドキュメントをその専用ツールにより開きウィンドウに表示する．

Interleaf社の DTP(Interleaf5) のカスタマイズ用の言語 Interleaf Lisp[12]を用いてブラウザを作成している．現在，グラフインタプリタについては，データ駆動型実行により表示過程を忠実に再現する機構が稼働している．また，ドキュメントオブジェクトのメソッド，クラス管理と画面表示制御は Interleaf Lisp の機能を用いて実現している．図 8に，簡単なデータフローグラフをデータ駆動型実行させた時に現れる一連の表示出力の一部を示す．

7 おわりに

本稿では，以下の特長を持つリアクティブ・データフローモデルと呼ぶドキュメント表示モデルについて述べた．

- ハイパテキストに対して，データフロー計算モデルに基づいてリンクの辿り方に関する意味付け法を与え，ドキュメントの表示過程を構造的な並列プログラムとして表現する．プログラムの再実行により，表示過程を再現する．
- データフロー計算モデルにリアクティブ機構を導入し，ユーザが置かれた局面に応じて再現法を変化させる．
- ドキュメントを，クラス，名前，型を持つオブジェクトとしてモデル化して，ドキュメント・オブジェクトを用いて抽象度の高い形式で「抽象表示過程」を記述し，実行時に抽象表示過程に対して，詳細情報を与えるドキュメントアクセス機構をもつ．

今後はドキュメントブラウザ実験システムの作成を進め，ブラウザの機能評価を行う予定である．

謝辞

リアクティブ機構に関して，有益なご意見を頂いた NTT 基礎研究所 島健一氏に感謝します．日頃ご討論頂く伊藤正樹リーダはじめ，鈴木英明，直井邦彰，八木哲の各氏，ならびに，日頃ご支援いただくグループの皆様に深謝します．

図 8: 実行例

参 考 文 献

1) 高田広章, ハイパテキストとそのプログラミング環境への応用, 情報処理, Apr 1989, Vol.30 No.4, pp406-413

2) Daniel S. Jordan, Daniel M. Russell, Anne-Marie S. Jensen, Russel A. Rogers, *Facilitating the Development of Representation in Hypertext with IDE*, Hypertext '89 Proceedings, Nov 89, pp93-103

3) Pankaj K. Garg, Walt Scacchi, *A Hypertext System to Manage Software Life-Cycle Documents*, IEEE Software, May 1990, pp90-97

4) P.DAVID STOTTS and RICHARD FURUTA University of Maryland, *Petri-Net-Based Hypertext:Document Structure with Browsing Semantics*, ACM Transaction on Information Systems Vol.7 No.1 Jan 1989, pp3-29

5) D.Harel and A.Pnueli, *On the Dovelopment of Reactive System*, Logics and Models of Concurrent Systems, NATO ASI Series, Vol.F13 1985, pp3-29

6) Frederic Boussinot, *Reactive C:An Extension of C to Program Reactive System*, SOFTWARE-PRACTICE AND EXPERIENCE, Apr 1991 Vol.21(4), pp401-428

7) 雨宮 真人, データフロー・アーキテクチャについて, コンピュータソフトウェア, Apr 1984 Vol.1 No.1, pp42-63

8) R.E. フィルマン／D.P. フリードマン共著 雨宮 真人／尾内 理紀夫／高橋 直久共訳, 協調型計算システム 分散型ソフトウェアの技法と道具立て, マグロウヒルブック

9) 高橋 直久, データ共有型並列プログラムの要求駆動型再演システムの実現と評価, 並列処理シンポジウム JSPP'90, pp361-368

10) 米澤 明憲, オブジェクト指向計算の現状と展望, 情報処理, Apr 1988 Vol.29 No.4, pp290-294

11) Leon Osterweil, *Software Process Interpretation and Software Environments*, CU-CS-324-86, Apr 1986 Draft 14

12) Paul M. English, Ethan S. Jacobson, Robert A. Morris, Kimbo B. Mundy, Stephen D. Pelletier, Thomas A. Polucci, and H. David Scarbro, *An Extensible, Object-Oriented System for Active Documents*, Proceedings of the International Conference on Electric Publishing, Document Manipulation & Typography Sep 1990, pp263-276