(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3856819号
(P3856819)
(45)発行日　平成18年12月13日(2006.12.13)　(24)登録日　平成18年9月22日(2006.9.22)

(51)Int.Cl.　　F I
G06F 17/21 (2006.01)　　G06F 17/21 570D
G06Q 10/00 (2006.01)　　G06F 17/21 586E
　　G06F 19/00 310H

請求項の数 1　(全 19 頁)

(21)出願番号　特願平7-507539
(86)(22)出願日　平成6年1月3日(1994.1.3)
(65)公表番号　特表平9-502037
(43)公表日　平成9年2月25日(1997.2.25)
(86)国際出願番号　PCT/US1994/000012
(87)国際公開番号　WO1995/006282
(87)国際公開日　平成7年3月2日(1995.3.2)
審査請求日　平成13年1月4日(2001.1.4)
審判番号　不服2005-16015(P2005-16015/J1)
審判請求日　平成17年8月22日(2005.8.22)
(31)優先権主張番号　08/112,821
(32)優先日　平成5年8月25日(1993.8.25)
(33)優先権主張国　米国(US)

(73)特許権者　500269956
オブジェクト　テクノロジー　ライセンシング　コーポレイション
アメリカ合衆国　９５０１４－２０８３
カリフォルニア州　クパチーノ　メールステーション　３－オーティーエル　インフィニト　ループ　１
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(74)復代理人　100115635
弁理士　窪田　郁大

最終頁に続く

(54)【発明の名称】オブジェクト指向ナビゲーション・システム

(57)【特許請求の範囲】
【請求項１】
プロセッサと、該プロセッサに接続され、その制御下に置かれているストレージと、前記プロセッサに接続され、その制御下に置かれているディスプレイとを装備したコンピュータ・システム中で、第１および第２ドキュメント・オブジェクトの間、およびフォルダ間でナビゲーションを行う方法であって、前記第１および第２ドキュメント・オブジェクトの各々はコンテナ・オブジェクトで構成され、該コンテナ・オブジェクトはその中に複数のモデル・オブジェクトを有し、その各々はモデル・オブジェクト・インタフェースを有し、前記コンピュータ・システムはファイルをまとめるためのフォルダおよび他のフォルダを有するファイル・システムを具えており、前記方法は、
（ａ）前記第１ドキュメント・オブジェクトを前記ストレージにロードし、および該第１ドキュメント・オブジェクトを前記ディスプレイに表示するステップと、
（ｂ）前記ストレージ中の第１ドキュメント・オブジェクト内の１つまたは複数のモデル・オブジェクトを示すセレクション・オブジェクトを前記ストレージ上に作成するステップと、
（ｃ）モデル・オブジェクト・インタフェースを有する参照オブジェクトを作成するステップと、
（ｄ）前記セレクション・オブジェクトに対する参照を前記参照オブジェクトに挿入するステップと、
（ｅ）前記第２ドキュメント・オブジェクトを前記ストレージにロードして、該第２ドキ

ュメント・オブジェクトを前記ディスプレイ上に表示するステップと、
（ｆ）前記参照オブジェクトのモデル・オブジェクト・インタフェースを介して、前記セレクション・オブジェクトに関連する参照オブジェクトをモデル・オブジェクトとして前記第２ドキュメント・オブジェクトに置くステップと、
（ｇ）前記第２ドキュメント・オブジェクト内の参照オブジェクトが選択されたときに、前記第１ドキュメント・オブジェクト内の、前記セレクション・オブジェクトにより示す１つまたは複数のモデル・オブジェクトを表示するステップと、
（ｈ）前記フォルダを示す別の参照オブジェクトを前記ストレージ上に作成するステップと、
（ｉ）前記別の参照オブジェクトのモデル・オブジェクト・インタフェースを介して、前記他のフォルダ内部に前記別の参照オブジェクトを置くステップと、
（ｊ）前記他のフォルダが開かれたときに、前記別の参照オブジェクトを介して、前記フォルダを開くステップと
を具えたことを特徴とする方法。
【発明の詳細な説明】
著作権所有表記
本明細書の一部には、著作権保護の対象となる内容が含まれている。著作権所有者は、米国特許商標庁の特許ファイルまたは記録に記載されている特許文書または特許開示事項を第三者がファクシミリで複製することを妨げるものではないが、その他の場合には、著作権に係る一切の権利を留保する。
関連特許出願の相互参照
本特許出願は「オブジェクト指向フレームワーク・システム」(Object Oriented Framework System)という名称のDebra L.Orton、David B.Goldsmith、Christopher P.Moeller、Andrew G.Heningerによる特許出願であって、1992年12月23日に出願され、Taligentに譲渡された特許出願の関連出願であり、その特許出願の開示事項は引用により本明細書の一部を構成するものである。
本特許出願は「動的リンキング・システム」(Dynamic Linking System)という名称のArn Schaeffer、David Anderson、Jack Palevich、Larry Rosensteinによる特許出願であって、1993年７月19日に出願され、Taligentに譲渡された特許出願の関連出願であり、その特許出願の開示事項は引用により本明細書の一部を構成するものである。
本特許出願は「コンテナ・オブジェクト・システム」(Container Object System)という名称のFrank Nguyenによる特許出願であって、1993年６月23日に出願され、Taligentに譲渡された特許出願の関連出願であり、その特許出願の開示事項は引用により本明細書の一部を構成するものである。
発明の分野
本発明は一般的には、コンピュータ・システムにおける改良に関し、より具体的には、情報のエリアとシステム全体との間をナビゲートするシステムおよび方法に関し、さらにリモート・オブジェクトがあたかもローカルの場所に置かれているものとしてリモート・オブジェクトを使用することに関する。
発明の背景
コンピュータ・システムは精油所を制御するために使用されるプロセス制御システムに見られるもののようなアプリケーションの間をリンケージで結ぶことを必要とすることがよくある。その結果、例えば、グラフィカル・ワークステーションにいるオペレータがバルブを示すグラフィカル記号を選択して、そのステータス（状況）を閉から開に変更すると、ディスプレイ上のそのイメージが変更され、これに対応して、バルブが閉じているプロセス環境が変更されるようになっている。しかし、大部分のプロセス制御システムは、特定のタスクを実施するために「ハード・コード化(hard-coded)」されている。リンケージはオペレータのコンソールを介して再定義することが容易になっていない。さらに、リンケージは柔軟性がなく、多目的にもなっていない。むしろ、各リンケージはタスクが事前に定義されているため、その変更が容易になっていなかった。

最近では、アップル社や他社の対話式コンピュータ・システム(interactive computer system)は、クローズしているドキュメントとコンテナ(container)またはフォルダ(folder)との間をリンケージで結んでいる。その結果、例えば、対話式コンピュータ・システムのユーザはドキュメントを示すグラフィカル記号を選択して、そのグラフィカル記号を通してドキュメントをオープンすることが可能になっている。フォルダは、フォルダを示すグラフィカル記号を使用すると、選択したドキュメントがフォルダにドロップされるようになっている。

発明の概要

本発明は、コンピュータ・システムにおいてオブジェクト間をナビゲートするシステムと方法を提供することによりさらに、コンピュータ・システムにおいてリモート・オブジェクトがあたかもローカルの場所に置かれているものとしてリモート・オブジェクトを使用することにより従来技術の問題点を解消する。接続されたストレージ（記憶装置）とディスプレイを装備したプロセッサを含んでいるオブジェクト指向コンピュータ・システムにおいてナビゲーションを行うための手法が開示されている。第１ドキュメントがストレージにロードされ、ディスプレイから表示される。そのあと、セレクション・オブジェクト(selection object)が作成され、ディスプレイ上の第１ドキュメントと関連づけられる。参照(reference)がストレージ内の第１セレクションと関連づけられる。次に、第２ドキュメントがストレージにロードされ、ディスプレイ上に表示され、参照が第２ドキュメント内の第１セレクションと関連づけられる。最後に、第２ドキュメント内の参照を通して、第１ドキュメント内の第１のセレクションへナビゲーションを行うことが可能になる。

【図面の簡単な説明】

第１図は、好適実施例におけるパーソナル・コンピュータ・システムを示すブロック図である。

第２図は、好適実施例において選択したエリアと参照オブジェクトをもつドキュメント・ディスプレイを示す図である。

第３図は、参照オブジェクトを作成する好適実施例に従って実行される処理を示すフローチャートである。

第４図は、参照オブジェクトを使用する好適実施例による詳細ロジックを示すフローチャートである。

第５図は、好適実施例においてリモート・プリンタを指す参照をもつディスプレイを示す図である。

第６図は、好適実施例においてユーザがオブジェクトのコピーを参照へドラッグしている様子を示す図である。

第７図は、好適実施例においてあるアクションがまだ完了していないとき参照オブジェクトの近くにあるオブジェクトを示す図である。

発明の好適実施例の詳細な説明

ナビゲーション・リンクの概要

参照オブジェクトは、グラフィック・ディスプレイ上で自身と他のオブジェクトとを区別する固有の特性をもっている。第一に、参照はユーザ・データを置くことができる場所ならば、どこにでも置くことができる（例えば、ドキュメントやコンテナ、プレース内に置くことも、デスクトップ上に置いておくだけでも可能である）。第二に、参照は別の参照ではなく、現実のオブジェクトを参照する。参照は、複数のオブジェクトを収めておくことができる１つセレクションであって、その１つのセレクションだけを参照する。参照は、参照の数やオブジェクトの属性（例えば、名前、アイコン）および／または包含階層(containment heirarchy)に置かれているオブジェクトの位置が変更されても、その変更とは関係なくそのオブジェクトを参照し続ける。参照をオープンすると、参照オブジェクト(referred-to object)がオープンされ、表われる。参照をコピーしても、削除しても、その参照オブジェクトはコピーも、削除もされない。参照は参照オブジェクトの存在を保証していないので、デフォルトによって参照は一方向(unidirectional)になっている。

参照の継承特性

参照は「動的リンキング・システム」という名称の特許出願（全内容は本明細書の一部を構成する）に記載されているようにモデル・アンカ(model anchor)であるので、参照は次のような特性を継承する。

・　参照は参照オブジェクトへのリンクをたどっていくことができる。
・　参照は属性（例えば、名前、アイコンなど）をキャッシュすることができる。
・　参照はアイコン・ビュー(icon view)として表示することができる。

使用例を示すシナリオ

最初の組のシナリオは、参照がシステムでどのように使用されるかを示している。

技術論文

ある著者は別の著者による論文のあるセクションへの参照を行っている。この参照はどのページとパラグラフがトピックと関係があるかを正確に規定している。読者はこの参照を使用すると、一番目の著者の論文から二番目の著者の論文中の関係情報へナビゲートすることができる。

スプレッドシート

スプレッドシートのあるセルの値を、別のセルの値を計算する式の中で参照することが望ましいことがよくある。ユーザはその参照を使用して、セルが同じスプレッドシートにあるか、別のスプレッドシートにあるかに関係なく、そのセルへナビゲートすることもできる。

フォルダ

ユーザはあるフォルダへの参照を別のフォルダ内に置いておくことできるので、その参照は別のフォルダ階層にあるかのように見える。その参照フォルダをオープンすると、ユーザは元のフォルダ・ロケーションに到達する。

アプライアンス

システム上の別のロケーションからあるアプライアンス（プリンタ、ファックスマシン、テレホンなど）へアクセスする必要がユーザにある場合、ユーザはそのアプライアンスへの参照を作成して、それを望みのロケーションに置いておくことができる。この参照をナビゲーション・エイド(navigational aid)として使用することも、その実際のロケーションへナビゲートしないでその参照を通してアプライアンスを直接に使用することも可能である。

ビジネス・カード

ある人のプリンタへの参照はエレクトロニック・ビジネス・カード(electronic business card)に置いておくことができるので、ビジネス・カードの受取人はその参照を使用して最初のパーティのプリンタへナビゲートすることができる。

以下に示すシナリオでは、ユーザと参照とのやりとりを説明するためにフォルドとドキュメントの参照を使用している。他のタイプの参照も、その働き方は大体同じである。

あるフォルダ内のドキュメントの参照を同じフォルダから行う：ユーザはドキュメントＡをフォルダＸからコマンド・ドラッグし（ドラッグしている間にコマンド・キーを押す）、それを同じフォルダの中にドロップする。これはフォルダＸ内ではドキュメントＡ（ドキュメントＡへの参照）として現われる。

ドキュメント内のテキスト・セレクションの参照を別のフォルダから行う：ユーザはドキュメントＢ内のテキスト・セレクションをコマンド・ドラッグし、それをフォルダＸの中にドロップする。これはフォルダＸ内ではドキュメントＢ（ドキュメントＡへの参照）として現われる。

フォルダ内のあるドキュメントの参照を別のドキュメントから行う：ユーザはフォルダＸ内のドキュメントＡをコマンド・ドラッグし、それをドキュメントＢの中にドロップする。これはドキュメントＢ内ではドキュメントＡ（ドキュメントＡへの参照）として現われる

参照を削除する：ユーザはドキュメントＡをドキュメントＢからドラッグしてウェストバスケット(Waste Basket-Trash Canへの参照)へ入れるか、Trash Can自体へ入れる。

参照をあるフォルダから別のフォルダへコピーする（移動する）：ユーザはドキュメント

ＡをフォルダＸからフォルダＹへオプション・ドラッグ(option-drag)する（ドラッグする）。これはフォルダＹ内ではドキュメントA′として現われる。
参照をフォルダからドキュメントへコピーする（移動する）：ユーザはドキュメントＡをフォルダＸからドキュメントＢへオプション・ドラッグ(option-drag)する（ドラッグする）。これはドキュメントＢ内ではドキュメントA′として現われる。
参照をドキュメントからフォルダへコピーする（移動する）：ユーザはドキュメントA′をドキュメントＢからフォルダＸへオプション・ドラッグする（ドラッグする）。これはドキュメントＢ内ではドキュメントA″として現われる。
クラス・アーキテクチャの詳細
ハイパリンク(hyperlink)はソース・アンカ(source anchor)からデスティネーション・アンカ(destination anchor)までの双方向リンクである。コンテナブル(containables)は、潜在的には、空のアンカ(empty anchor)からデスティネーション・アンカまでの双方向リンクである。参照は、潜在的には、空になっている可能性のあるアンカからデスティネーション・アンカまでの双方向リンクである。それぞれは、属性の表示と属性のキャッシング、およびリンクをたどってデスティネーション・アンカまでに到達するための独自のプロトコルをもっている。ハイパリンク、コンテナブル、および参照によって示される類似性を扱うための設計は次のようになっている。
・　ハイパリンク、コンテナブル、および参照に対するサポートはTContainPolicyから提供され、そのデフォルト・インプリメンテーションはTStandardContainPolicyである。これらのクラスはコンテナ・フレームワーク(Container Framework)の一部になっており、このフレームワークは、サブクラス化(subclassing)によって新しいデータ・タイプを作成するための２つの抽象モデル・クラス(abstract model classes)を与える。詳しくは後述の「モデル・クラス」セクションを参照されたい。セレクション側では、このサポートはMSelectionCanContainePolicyにより提供され、このデフォルトインプリメンテーションはTSelectionCanContain Policyである。これらのクラスもコンテナ・フレームワークの一部になっており、このフレームワークは、サブクラス化によって新しいセレクション・タイプを作成するための２つの抽象セレクション・クラスを与える。詳細は後述の「セレクション・クラス」セクションを参照されたい。
・　ハイパリンク、コンテナブル、および参照のリストはTModelAnchorポインタの単一リストである。ハイパリンクはTModelAnchorである。コンテナブルと参照は共通のクラス、つまり、TModelAnchorのサブクラスであるTContainableから派生している。詳細は後述の「ハイパリンク／コンテナブル／参照クラス」セクションを参照されたい。
ユーザは、オブジェクトまたはオブジェクトのコレクションを選択し、そのセレクションをコマンド・ドラッグしドロップすることにより参照を作成することができる。このアクションにより、ソース・モデル・セレクション（つまり、ドラッグされたオブジェクト）をパラメータとして持つMakeReferenceコマンドがソース・モデルへ送られ、ターゲット・モデル・セレクション（たとえば全体ターゲット・モデル）をパラメータとして持つペースト(Paste)・コマンドがターゲット・モデルへ送られる。ソースとターゲットは同じモデルでも、異なるモデルでも構わない。詳細は後述の「コマンド・クラス」セクションを参照されたい。
どのモデルも、またどのモデル・セレクションも(TContainPolicyをインプリメントしている場合）、別のモデルおよびモデル・セレクション(TContainPolicyをインプリメントしている場合）からの参照を表示し、操作することをサポートしている。例えば、フォルダには、複合ドキュメント(compound document)（複合ドキュメント・セレクション）内の特定のテキスト部分を指す参照を収めておくことができ、また、複合ドキュメントには、フォルダ内の特定のドキュメントを規定する参照を収めておくことができる。このような相互操作性(interoperability)が可能であるのは、参照が常にTContainableWrapperModelに包み込まれており、どのモデルも、どのモデル・セレクション(TContainPolicyがインプリメントされている場合）も、それをどのように取り入れるかを知っているからである。詳細は後述の「ラッパ・モデル・クラス」セクションを参照されたい。

モデル・クラス
コンテナ・フレームワークにはTEembedderContainerModelが用意されているので、これをサブクラス化すると、組込み(embed)と収容(contain)ができる新しいデータ・タイプを作成することができる。例えば、TCompoundDocumentModelは、ハイパリンク、コンテナブル、および参照と同じように、組み込まれたモデルをサポートできるドキュメントである。同様に、コンテナ・フレームワークにはTContainerModelが用意されているので、これをサブクラス化すると、収容だけができる新しいデータ・タイプを作成することができる。例えば、TFolderModelはハイパリンク、コンテナブルおよび参照をサポートできるフォルダである。
セクション・クラス
コンテナ・フレームワークにはTEmbedderContainerModelSelectionが用意されているので、これをサブクラス化すると、組込みと収容ができる新しいセレクション・タイプを作成することができる。例えば、
TCompoundDocumentModelSelectionは、ハイパリンク、コンテナブルおよび参照と同じように、組み込まれたモデルをサポートするドキュメント・セレクションである。
同様に、コンテナ・フレームワークにはTContainerModelSelectionが用意されているので、これをサブクラス化すると、収容だけができる新しいセレクション・タイプを作成することができる。例えば、TFolderModelSelectionは、ハイパリンク、コンテナブルおよび参照をサポートできるフォルダ・セレクションである。
ハイパリンク／コンテナブル／参照クラス
TModelAnchorにより、ハイパリンク、コンテナブルおよび参照を管理するための、次のようなプロトコルが与えられる。
・　属性管理プロトコル、つまり、属性の追加、削除、リスティング、およびルックアップ
・　データ管理プロトコル、つまり、データの追加、コピー、および削除
・　リンク管理プロトコル、つまり、リンクの追加、削除、フォロー、リスティング、およびルックアップ（コンテナブルまたは参照の場合は、リンクは１つだけである）
・　ビュー作成プロトコル、つまり、ハイパリンク、コンテナブルおよび参照のアイコン・ビューの作成
ビュー作成プロトコルは、ハイパリンクのリンク・マップを作成するために複合ドキュメント・プレゼンテーションによって使用される。これは、コンテナブルと参照のアウトライン・ビュー(outline view)を作成するためにフォルダ・プレゼンテーションによって使用することもできる。
TContainableはTModelAnchorの抽象サブクラスである。これは、コンテナブル(TOwnedContainable)と参照(TReferencedContainable)に特有の新しいプロトコルを再定義する。
・　属性管理プロトコル、つまり、特殊な属性を取り出すためのアクセラレータメソッド(accelerator method)。これらの属性には、システム定義の属性（コンテナブルまたは参照のビューを作成するときのサイズ、作成者、作成日、タイプ、種々のアイコン表現）とユーザ定義の属性（名前、ユーザ定義のラベル）を含めることが可能である。TContainableはそのローカル・キャッシュ内に属性を見つけないと、モデル・ストア(model store)をアクセスして属性を取り出す。場合によっては、ユーザ・インターフェース・アーカイブ内に属性を見つけることがある。例えば、モデルにカスタム・アイコンがないと、総称的「ドキュメント」アイコンを取り出す。
・　データ管理プロトコル、つまり、コピーと削除のための再定義されたメソッド。

	TOwnedContainable	TReferencedContainable
コピー	コンテナブルとモデルをコピーする	参照のみをコピーする
削除	コンテナブルとモデルを削除する	参照のみを削除する

・　コンテナブルまたは参照をモデル内に包み込み、モデル・セレクションを使用してデータ交換（例えば、カット、コピー、ペースト）に参加できるようにするメソッド。
・　選好フレームワーク（Preferences Framework)がコンテナブル上のプロパティ・シートをオープンするためのメソッド。
TContainableの他のサブクラスは、異なるセマンティック(semantics)および／または作用(behavior)が必要なときに書くことができる。例えば、ステーショナリを「オープン」すると、新しいドキュメントが作成される。従って、TContainableのサブクラスである、TStationeryContainableは正しい作用を得るためにはそのリンク・フォロー・メソッド(link-follow method)を無効化(override)する必要がある。
コマンド・クラス
MakeReferenceコマンドは、どのモデル・セレクションとその関連モデルにも作用を及ぼす新しい標準コマンドである。MakeReferenceは次のような働きをする。
1．データをソース・モデル・セレクションから取得する。このデータはTModelAnchorポインタ型であり、ハイパリンク、コンテナブル、または参照を表している。
2．そのデータに基づき参照を作成する。
3．この参照を新しいソース・モデル・セレクションにする。
4．新しいソース・モデル・セレクションをクリップボードにコピーする。
ペースト(Paste)コマンドはどのモデル・セレクションとその関連モデルにも作用を及ぼす既存の標準コマンドである。Pasteは次のような働きをする。
1．異質モデル(foreign model)を変換できるタイプのリストをクリップボードに要求する。
2．異質モデルを変換しようするタイプと、変換した異質モデルをどのように受け入れるか（組込みか吸収か）をそのリストから選択するようにターゲット・モデル・セレクションに要求する。
3．変換した異質モデルのコピーをクリップボードに要求する。
4．変換した異質モデルを受け入れるようにターゲット・モデルに要求する。
5．このモデルを新しいターゲット・モデル・セレクションにする。
ラッパ・モデル・クラス
モデル・セレクションを使用してデータ交換（例えば、カット、コピー、ペーストなど）を行うためにはコンテナブルと参照を包み込むためのメソッドが必要である。交換カレンシ(exchange currency)はモデルである。ラッパ・モデル(wrapper model)はTContainerPolicyとTEmbedPolicyのどちらのインプリメンテーションも必要としないので、TModel抽象クラスの具現サブクラスとして作成することができる。
第３図は、参照オブジェクトを作成する好適実施例に従って実行される処理を示すフローチャートである。処理はターミナル300から開始され、即時に機能ブロック310へ移り、そこで第１と第２のオブジェクトがオープンされる。オブジェクトはテキスト・ドキュメント、グラフィック・イメージ、スプレッドシート、フォルダ、または他のコンピュータ・アプリケーションのどれであっても構わない。次に、機能ブロック320で、ディスプレイ上のエリアの第１コーナを選択し、選択したエリアを囲むように拡大する矩形アウトラインをドラッグすることにより、セレクションが第１オブジェクトで作成される。機能ブロック330で、セレクションから作成されるアンカ・オブジェクトは選択したエリアにリンクされる。このアンカは、セレクション・エリアがアンカ・オブジェクトとリンクされた

状態にある限り存続しているセレクション・エリアを管理する。機能ブロック340は参照オブジェクトの作成を示している。参照オブジェクトはアンカ・オブジェクトを取り巻くラッパであり、このラッパによりアンカ・オブジェクトを他のオブジェクトまたはコンテナの内側に置いておくことができる。これについては、相互参照欄に列挙されている特許出願に説明されている。次に、機能ブロック350で、サロゲート(surrogate)アンカが参照オブジェクトに受け入れられ(adopted)、機能ブロック360に示すように参照オブジェクトが第２のオブジェクトに受け入れられる。ここでサロゲート・オブジェクトとは、実際のアンカに対するポインタのことである。また、「受け入れる(Adoption)」とは、所有権をあるオブジェクトから別のオブジェクトへ移転するために必要な処理のことである。最後に、処理はターミナル370で完了する。この例では、参照オブジェクトが一方のドキュメントで作成されて、他方のドキュメントに受け入れられているが、この分野に精通する当業者ならば理解されるように、参照オブジェクトは第１のドキュメントの別のロケーションに置いておくことも可能である。

第４図は、参照オブジェクトを使用する好適実施例に従って実行される処理を示すフローチャートである。処理はターミナル400で開始され、即時に機能ブロック410へ移り、そこで第１のオブジェクトがオープンされる。オブジェクトはテキスト・ドキュメント、グラフィック・イメージ、スプレッドシート、フォルダ、他のコンピュータ・アプリケーションのどれであっても構わない。次に、機能ブロック420で、参照オブジェクトが選択される。この参照オブジェクトがアクションのターゲットを抽象的に規定しているのは、そのセレクションを規定しているアンカがそこに入っているためである。このアクションはフォローまたはルックアップ時に解決される。機能ブロック430は、start_side（開始側）でのコマンドの作成処理を示している。この作成処理は、ユーザがダブルクリックしてオブジェクトをオープンするといった、特定のアクションを行うとこれに応答する。あるアクションがとられると、そのアクションにサービスするために適切なコマンドが出される。そのため、例えば、このコマンドはフォロー、またはオープン・コマンド、プリント・コマンド、削除コマンド、またはdest_side（デスティネーション側）ドキュメント内のセレクションに関係するどのコマンドであっても構わない。この処理はドキュメントのstart_sideで指示されたコマンドを作成する。次に、機能ブロック440で、コマンドは他方のドキュメントのアンカのdest_sideへ送られる。この送信オペレーションは機能ブロック450に示すようにリンケージを照会し、必要ならばdest_sideドキュメントをオープンし、セレクションをdest_sideのアンカに一致するようにセットし、機能ブロック460に示すように、アンカの位置をビューまでスクロールする。次に、start_sideで適用されたコマンドが機能ブロック470に示すようにdest_sideで適用される。最後に、処理はターミナル480で完了する。この例では、参照オブジェクトが一方のドキュメントに置かれ、参照されたアンカが他方のドキュメントに置かれているが、この分野に精通する当業者ならば理解されるように、参照オブジェクトと参照されたアンカは同じドキュメントに置いておくことも可能である。

第５図は好適実施例によるリモート・プリンタへの参照をもつディスプレイを示す図である。ドキュメント・アイコン500は、現在、ドキュメントNew Request 510に置かれている。リモート・プリンタMyPrinter 520への参照はアイコン形式で使用可能にされる。リモート資源を操作するために使用されている参照オブエジェクトの例は第５図、第６図および第７図とこれらの図に付属する説明に示されている。

第６図は好適実施例においてユーザがオブジェクトのコピーを参照へドラッグしている様子を示している。第６図はドキュメント・アイコン600とドキュメントのアウトライン610を示している。アウトラインは、アイコン600をドラッグしている間にオプション・キーを押すと作成される。アウトラインはドキュメントのコピーを表している。

第７図は好適実施例においてあるアクションがまだ完了していないときの参照オブジェクトの近くにあるオブジェクトを示している。ドキュメント・アウトライン710はプリンタ参照720でドロップされると、ドキュメント700でプリント・コマンドが実行される。

REFERENCE プログラムのリスティング

C++ 言語

```
//インタフェース: TReferencedContainable
class TReferencedContainable : public TContainable
{
public:
        static const TContainableKind& kKind;
        //特別メソッド
                                TReferencedContainable(const
TAbstractModel&,
                                        const TAbstractModel&);
                                TReferencedContainable();
                                TReferencedContainable(const
TReferencedContainable&);
                                ~TReferencedContainable();

        TReferencedContainable&         operator=(const
TReferencedContainable&);
        virtual     TStream&            operator>>=(TStream& towhere) const;
        virtual     TStream&            operator<<=(TStream& towhere);
```

```
        MCollectibleDeclarationsMacro(TReferencedContainable);
        //表示メソッド(Presentation methods)
        virtual     TPresentableView*       CreateIconView() const;

        //セマンティックメソッド(Semantic methods)
        virtual     TContainableKind        GetContainableKind() const;
        virtual     void                    DeleteSelf();
        virtual     TContainable*           DuplicateSelf() const;
};
//インタフェース：TMakeReferenceUserCommand
class TMakeReferenceUserCommand : public TNewAnchorCommand
{
public:
                                TMakeReferenceUserCommand(const
TText&);
                                TMakeReferenceUserCommand();
                                TMakeReferenceUserCommand(const
TMakeReferenceUserCommand&);
        virtual                 ~TMakeReferenceUserCommand();

            TMakeReferenceUserCommand& operator=(const
TMakeReferenceUserCommand&);
        virtual     TStream&            operator>>=(TStream&) const;
        virtual     TStream&            operator<<=(TStream&);
        MCollectibleDeclarationsMacro(TMakeReferenceUserCommand);
public:
        virtual     Boolean             HandleLocalDo();

protected:
        virtual     Boolean             Accepts(const TModelSelection&) const;
};
//
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++
//クラス名　：　TReferencedContainable
//説明　　：　参照のコンテナブルはドキュメントに対するエイリアスである。
//　　- CreateIconViewは参照のコンテナブルを表示するための
//　　アイコン・ビューを返す。
```

```
//          - DeleteSelfは無効化されるので、参照のコンテナブルを
//            コンテナだけから移動する処理はそれ自身を移動する。
//          - DuplicateSelf は無効化されるので、参照のコンテナブルを
//            コンテナの中にコピーする処理はそれ自身をコピーする。
//          - GetContainableKindはタイプとして
//            TReferencedContainable::kKind を返す。
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++
MCollectibleDefinitionsMacro(TReferencedContainable, 0);
const TContainableKind TReferencedContainable_kKind("TReferencedContainable");
const TContainableKind& TReferencedContainable::kKind =
TReferencedContainable_kKind;
TReferencedContainable::TReferencedContainable(const TAbstractModel&
containerModel,
        const TAbstractModel& containableModel)
        : TContainable(containerModel, containableModel)
{
}
TReferencedContainable::TReferencedContainable()
        : TContainable()
{
}
TReferencedContainable::TReferencedContainable(const TReferencedContainable&
copy)
        : TContainable(copy)
{
}
TReferencedContainable::~TReferencedContainable()
{
}
TReferencedContainable& TReferencedContainable::operator=(const
TReferencedContainable& copy)
{
        if (this != &copy)
        {
                TRY
                        TContainable::operator=(copy);
                CATCH(TStandardException)
                        RETHROW *this;
```

```
            ENDTRY
        }
        return (*this);
}
TStream&
TReferencedContainable::operator>>=(TStream& towhere) const
{
        TRY
            MAbstractModelAnchor::WriteVersion(towhere);
            TContainable::operator>>=(towhere);
        CATCH(TStandardException)
            RETHROW towhere;
        ENDTRY
        return towhere;
}
TStream&
TReferencedContainable::operator<<=(TStream& fromwhere)
{
        TRY
            VersionInfo theVersion =
MAbstractModelAnchor::ReadVersion(fromwhere);
            TContainable::operator<<=(fromwhere);
        CATCH(TStandardException)
            RETHROW fromwhere;
        ENDTRY
        return fromwhere;
}
TPresentableView*
TReferencedContainable::CreateIconView() const
{
        //参照のコンテナブル用のアイコンビューを作成する。
        TPresentableView* theView = new TContainableIconView((TContainable*)this,
/*isReference=*/ TRUE);
            return (theView);
}
TContainableKind
TReferencedContainable::GetContainableKind() const
{
```

```
	return (TReferencedContainable::kKind);
}
void
TReferencedContainable::DeleteSelf()
{
}
TContainable*
TReferencedContainable::DuplicateSelf() const
{
	return ((TContainable*)this);
}
//
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++
//クラス名　：　　TMakeReferenceUserCommand
//説明　　：　TMakeReferenceUserCommand ユーザが選択したメニュー項目
//　　によりアクチベートされる。
//　　コンテナブルにより示されるドキュメントに対する参照を作成し、
//　　コンテナ・モデルに対する参照を付加する。
//　　現在、このコマンドはTNewAnchorCommand のサングラスであり、
//　　モデル・アンカである（参照されたまたは所有された）コンテナブル
//　　の現在のインプリメンテーションに依存する。
//　　HandleReDo, HandleUndo, HandleNewAnchor を再使用する。
//　　コンテナ・モデルに対する参照を付加するためのTNewAnchorCommand
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++
MCollectibleDefinitionsMacro(TMakeReferenceUserCommand, 0);
const TStandardText kMakeReferenceMenuName("Make Reference");
TMakeReferenceUserCommand::TMakeReferenceUserCommand(const TText&
theCommandName)
	: TNewAnchorCommand()
{
	SetName(theCommandName);
}
TMakeReferenceUserCommand::TMakeReferenceUserCommand()
	: TNewAnchorCommand()
{
```

```
        SetName(kMakeReferenceMenuName);
}
TMakeReferenceUserCommand::TMakeReferenceUserCommand(const
TMakeReferenceUserCommand& theCommand)
        : TNewAnchorCommand(theCommand)
{
        SetName(kMakeReferenceMenuName);
}
TMakeReferenceUserCommand::~TMakeReferenceUserCommand()
{
}

TMakeReferenceUserCommand&
TMakeReferenceUserCommand::operator=(const TMakeReferenceUserCommand&
theCommand)
{
        if (this != &theCommand)
        {
                TRY
                        TNewAnchorCommand::operator=(theCommand);
                CATCH(TStandardException)
                        RETHROW *this;
                ENDTRY
        }
        return *this;
}
TStream&
TMakeReferenceUserCommand::operator>>=(TStream& towhere) const
{
        TRY
                WriteVersion(towhere);
                TNewAnchorCommand::operator>>=(towhere);
        CATCH(TStandardException)
                RETHROW towhere;
        ENDTRY
        return towhere;
}
TStream&
```

```
TMakeReferenceUserCommand::operator<<=(TStream& fromwhere)
{
	TRY
		VersionInfo theVersion = ReadVersion(fromwhere);
		TNewAnchorCommand::operator<<=(fromwhere);
	CATCH(TStandardException)
		RETHROW fromwhere;
	ENDTRY
	return fromwhere;
}
Boolean
TMakeReferenceUserCommand::Accepts(const TModelSelection& theSelection) const
{
	Boolean accept = TModelCommand::Accepts(theSelection);

	if (accept)
	{
		accept = !theSelection.IsEmpty(); //コマンドは、もし、セレクションが
                                          //空でない場合にイネーブルされる。
	}

	return (accept);
}
Boolean
TMakeReferenceUserCommand::HandleLocalDo()
{
	const TWorkspaceContainerSelection* theSelection = (const
TWorkspaceContainerSelection*)GetBoundSelection();
	TContainableSurrogateIterator anIterator(theSelection);
	TModelAnchorSurrogate* theSurrogate = anIterator.First();
	if (theSurrogate != NIL)
	{
		TDocumentEntry entry;
		TModel* theModel = theSelection->GetModel();
		TContainable* theContainable = (TContainable*)theModel-
>LookupAnchor(*theSurrogate);
		const TModelLink* theLink = theContainable->GetLink();
```

```
            TModelSurrogate* containerSurrogate = theLink->GetHere()-
>CopyModelSurrogate();
            TModelSurrogate* containableSurrogate = theLink->GetThere()-
>CopyModelSurrogate();
            // containerSurrogateおよび
            // containableSurrogateが参照対象のコンテナブルと
            // 等しくなる参照を作成する。
            TContainable* aReference = new
TReferencedContainable(*containerSurrogate, *containableSurrogate);
            //メンバとして、参照を持つデフォルト・セレクションを作成する。
            TModelSelection* referenceSelection = theModel-
>CreateDefaultSelection();
            referenceSelection->AddAnchor(*aReference);
            //セレクションを受ける参照を持つ。
            aReference->AdoptSelection(referenceSelection);

            //もし、参照のコンテナブルがステーショナリであれば、
            //参照をステーショナリにする。
            if (theContainable->IsStationery())
                  aReference->ToggleStationery();

            //参照を受け入れるコンテナ・モデルを持つ。
            AdoptAnchor(aReference);
      }
      return (TRUE);
}
```

以上、特定のシステム環境における好適実施例を例に挙げて本発明を説明してきたが、この分野に精通した当業者ならば理解されるように、本発明は変更を加えることにより、請求の範囲に明確化されている精神と範囲内において他の異なるハードウェアおよびソフトウェア環境で実施することも可能である。

【図１】

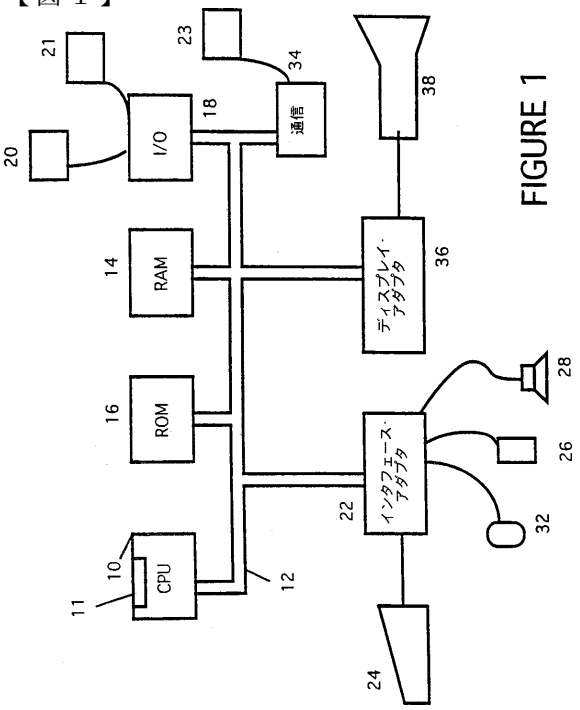

FIGURE 1

【図２】

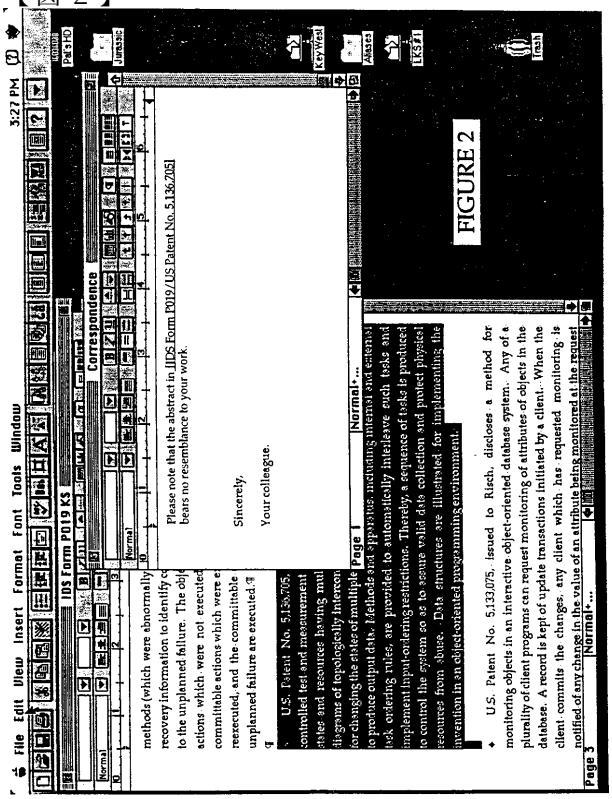

FIGURE 2

【図３】

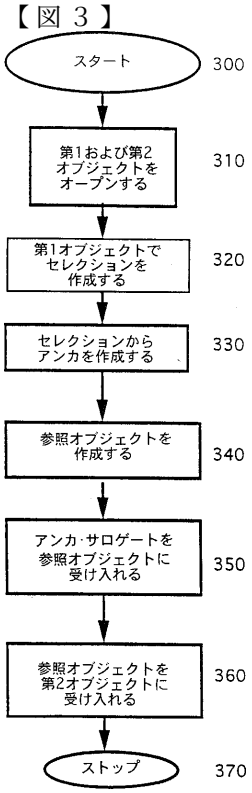

FIGURE 3

【図４】

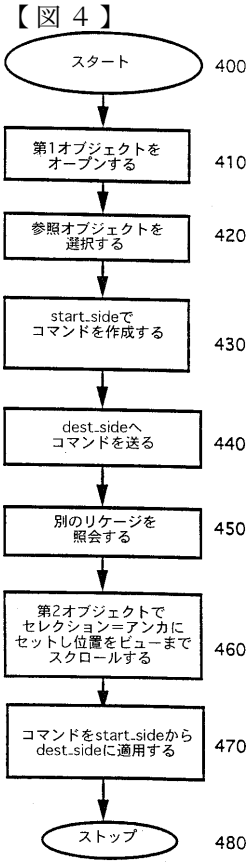

Figure 4

【図５】

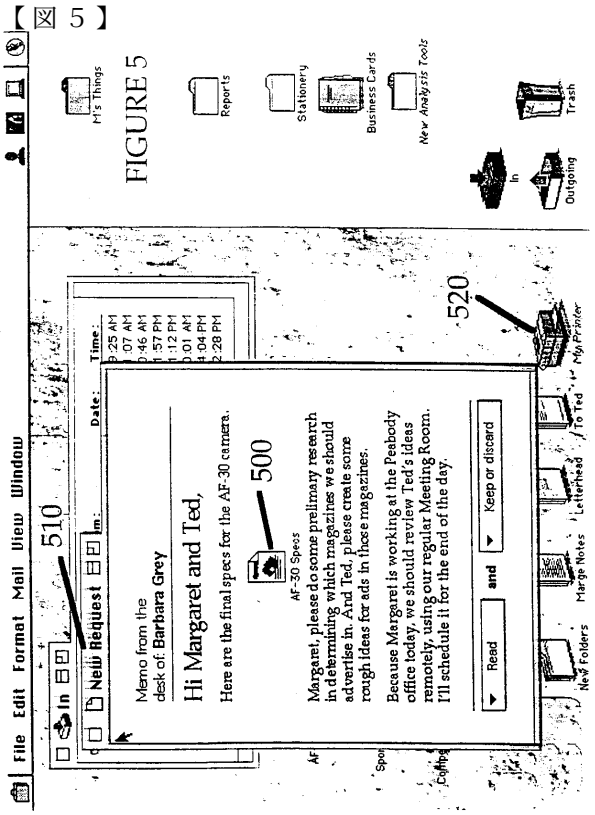
FIGURE 5
510
500
520
New Request
Memo from the desk of: Barbara Grey
Hi Margaret and Ted,
Here are the final specs for the AF-30 camera.
AF-30 Specs
Margaret, please do some prelimary research in determining which magazines we should advertise in. And Ted, please create some rough ideas for ads in those magazines.
Because Margaret is working at the Peabody office today, we should review Ted's ideas remotely, using our regular Meeting Room. I'll schedule it for the end of the day.
Read
and
Keep or discard
Date:
Time:
9:25 AM
1:07 AM
10:46 AM
1:57 PM
1:12 PM
10:01 AM
4:04 PM
2:28 PM
File Edit Format Mail View Window
In
M's Things
Reports
Stationery
Business Cards
New Analysis Tools
Trash
Outgoing
New Folders
Marge Notes
Letterhead
To Ted
My Printer

【図６】

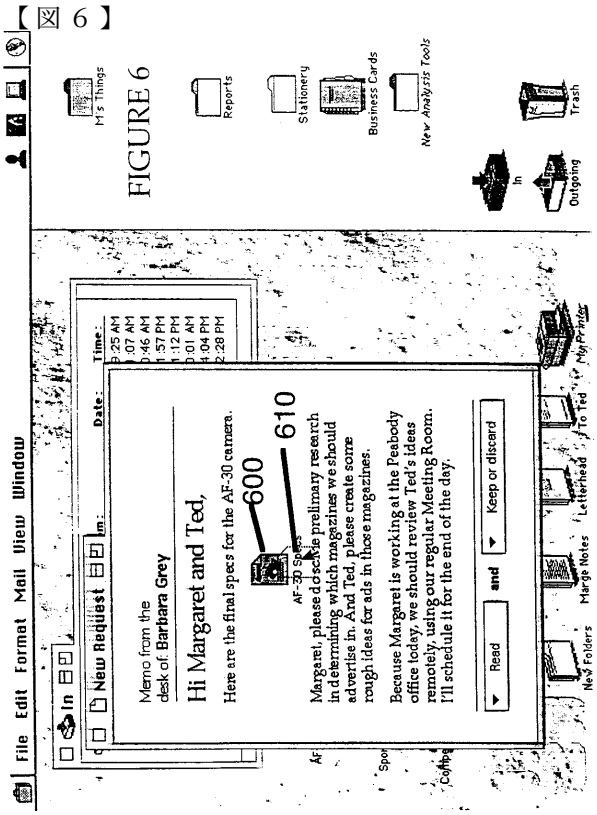
FIGURE 6
600
610
New Request
Memo from the desk of: Barbara Grey
Hi Margaret and Ted,
Here are the final specs for the AF-30 camera.
AF-30 Specs
Margaret, please do some prelimary research in determining which magazines we should advertise in. And Ted, please create some rough ideas for ads in those magazines.
Because Margaret is working at the Peabody office today, we should review Ted's ideas remotely, using our regular Meeting Room. I'll schedule it for the end of the day.
Read
and
Keep or discard
Date:
Time:
9:25 AM
1:07 AM
10:46 AM
1:57 PM
1:12 PM
10:01 AM
4:04 PM
2:28 PM
File Edit Format Mail View Window
In
M's Things
Reports
Stationery
Business Cards
New Analysis Tools
Trash
Outgoing
New Folders
Marge Notes
Letterhead
To Ted
My Printer

【図７】

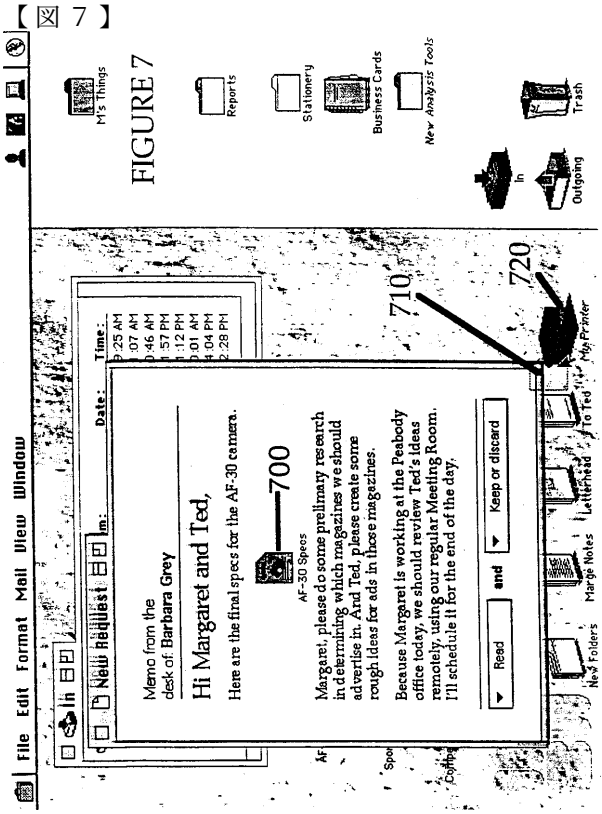
FIGURE 7
710
700
720
New Request
Memo from the desk of: Barbara Grey
Hi Margaret and Ted,
Here are the final specs for the AF-30 camera.
AF-30 Specs
Margaret, please do some prelimary research in determining which magazines we should advertise in. And Ted, please create some rough ideas for ads in those magazines.
Because Margaret is working at the Peabody office today, we should review Ted's ideas remotely, using our regular Meeting Room. I'll schedule it for the end of the day.
Read
and
Keep or discard
Date:
Time:
9:25 AM
1:07 AM
10:46 AM
1:57 PM
1:12 PM
10:01 AM
4:04 PM
2:28 PM
File Edit Format Mail View Window
In
M's Things
Reports
Stationery
Business Cards
New Analysis Tools
Trash
Outgoing
New Folders
Marge Notes
Letterhead
To Ted
My Printer

フロントページの続き

(72)発明者 グエン，フランク，ティー.
アメリカ合衆国 ９５００８ カリフォルニア州 キャンベル ケネス ストリート ４１５
(72)発明者 アンダーソン，デイヴィッド，アール.
アメリカ合衆国 ９５０１４ カリフォルニア州 クパチーノ ウエスト エステイツ ドライブ １０７６１
(72)発明者 キャトリン，ティモシィー，ジェイ．，オー.
アメリカ合衆国 ９４００２ カリフォルニア州 ベルモント スカイモント コート ７

合議体
審判長 吉岡 浩
審判官 成瀬 博之
審判官 長島 孝志

(56)参考文献 特開平４－２７４５３６（ＪＰ，Ａ）
特開平４－３０８９２２（ＪＰ，Ａ）
特開平４－１０２１６７（ＪＰ，Ａ）
坂村健，「ＢＴＲＯＮにおける統一的操作モデルの提案」，情報処理，第２６巻，第１１号，第１３２１－１３２８頁，社団法人情報処理学会，１９８５年１１月１５日

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F17/21,G06F19/00