(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3613401号
(P3613401)
(45)発行日　平成17年1月26日(2005.1.26)　(24)登録日　平成16年11月5日(2004.11.5)

(51)Int.Cl.[7]　F I
G06F 17/21　G06F 17/21　570D
G06F 12/00　G06F 17/21　570C
　　G06F 12/00　520P

請求項の数 17　(全 24 頁)

(21)出願番号　特願平5-167139
(22)出願日　平成5年7月6日(1993.7.6)
(65)公開番号　特開平6-195339
(43)公開日　平成6年7月15日(1994.7.15)
審査請求日　平成11年11月8日(1999.11.8)
(31)優先権主張番号　07/909983
(32)優先日　平成4年7月6日(1992.7.6)
(33)優先権主張国　米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　98052-6399　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100059959
弁理士　中村　稔
(74)代理人　100067013
弁理士　大塚　文昭
(74)代理人　100065189
弁理士　宍戸　嘉一
(74)代理人　100096194
弁理士　竹内　英人
(74)代理人　100074228
弁理士　今城　俊夫

最終頁に続く

(54)【発明の名称】オブジェクトの名称を付けて結び付ける方法及びシステム

(57)【特許請求の範囲】
【請求項１】
第１の複合文書から第２の文書内のデータへリンクするための、コンピュータシステムにおける方法であって、
前記第２の文書内の前記データに結び付けるための結び付けメソッドを提供する第１のオブジェクトをインスタンス生成するステップ、
前記第１のオブジェクトの結び付けメソッドの呼出しによって、前記第２の文書内のデータへの結び付けをさせるステップと、
前記第１の文書内の前記第１のオブジェクトを記憶するステップ
を含むことを特徴とする方法。
【請求項２】
前記記憶するステップが、前記第２の文書内の前記データを特定する情報とともに前記第１のオブジェクトのクラスタイプを記憶することを含むことを特徴とする請求項１に記載の方法。
【請求項３】
前記第１のオブジェクトが、前記第２の文書内の前記データを特定する情報を記憶し、前記情報がファイル名，あるいは前記データを含むファイルの部分，を指す請求項１に記載の方法。
【請求項４】
前記第１のオブジェクトが、複数の成分オブジェクト(component object)を含む複合オブ

ジェクトであり、当該複数の成分ブジェクトの各々が、前記データを特定する情報の少なくとも一部分を記憶する、請求項１に記載の方法。
【請求項５】
前記第１のオブジェクトが更に、減少メソッドを提供する、請求項１に記載の方法であって、前記減少メソッドの呼出しによって、減少されたオブジェクトを返送するようにさせる、方法。
【請求項６】
前記第１のオブジェクトが更に、合成メソッド（compose method）を提供する請求項１に記載の方法であって、
合成メソッドの呼出しによって、複合オブジェクト（composite object）がインスタンス生成されるようにさせる、方法。
【請求項７】
第１の複合文書を第２の文書内のデータにリンクするための、コンピュータシステムにおける方法であって、
前記第２の文書内のデータに結び付けるための結び付けメソッドを提供する第１のオブジェクトをインスタンス生成するステップであって、当該第１のオブジェクトが、前記第２の文書内のデータを特定する情報を記憶するものであり、前記第１の複合文書が当該第１のオブジェクトを記憶するものであり、
前記第２の文書内のデータに結び付けるために、前記第１のオブジェクトの前記結び付けメソッドを呼び出すステップ、
を含む方法。
【請求項８】
前記結び付けメソッドが、
ランニング・オブジェクト・テーブル内で、前記第２の文書内の前記データのためのサーバ・オブジェクトを探索(locate)し、そして、
クライアント・オブジェクトに、前記サーバ・オブジェクトにポイントするポインタを返送する、
ステップを含む、請求項７に記載の方法。
【請求項９】
前記結び付けメソッドが、
前記第２の文書内の前記データのためのサーバ・オブジェクトをインスタンス生成し、そして、
クライアント・オブジェクトに、前記サーバ・オブジェクトにポイントするポインタを返送する、
ステップを含む、請求項７に記載の方法。
【請求項１０】
前記結び付けメソッドが、
前記第２の文書内の前記データのためのサーバ・オブジェクトをインスタンス生成し、
前記第２の文書内の前記データをローディングし、そして、
クライアント・オブジェクトに、前記サーバ・オブジェクトにポイントするポインタを返送する、
ステップを含む、請求項７に記載の方法。
【請求項１１】
前記第１のオブジェクトが、第１のおよび第２の成分オブジェクトを持つ複合オブジェクトであり、
前記第１のオブジェクトの前記結び付けメソッドの呼出しが、前記第１の成分オブジェクト(component object)の結び付けメソッドの呼出しを引き起こし、これが、前記第２の成分メソッドの結び付けメソッドの呼出しを引き起こす、
請求項７に記載の方法。
【請求項１２】

前記第１のオブジェクトを、前記第１の文書の一部分として永続記憶装置内に記憶するステップを更に含む、請求項７に記載の方法。
【請求項１３】
前記インスタンス生成の前に、前記第２の文書内の前記データのためにストリング名を分解するステップを更に含む、請求項７に記載の方法。
【請求項１４】
第１の、複合文書から、第２の文書内のデータにリンクするための、コンピュータシステムにおける方法であって、
減少メソッドを提供するオブジェクトをインスタンス生成するステップであって、当該オブジェクトが、前記第２の文書内の前記データを特定する情報を記憶するものであり、前記第１の、複合文書が、当該オブジェクトを記憶するものであり、および、
減少されたオブジェクトを生成するために前記減少メソッドを呼び出すステップであって、当該減少されたオブジェクトが、前記第２の文書内の前記データを特定する減少された情報を記憶するものである、
を含む方法。
【請求項１５】
前記減少オブジェクトについて、それ以上の減少が不可能となるまで前記減少を反復するステップを更に含む、請求項１４に記載の方法。
【請求項１６】
前記減少オブジェクトについて、一定の回数だけ、前記減少を反復するステップを更に含む、請求項１４に記載の方法。
【請求項１７】
第１の、複合文書から、第２の文書内のデータにリンクするためのコンピュータ・システムにおける方法であって、
合成メソッド(compose method)を提供する第１のオブジェクトをインスタンス生成するステップであって、前記第１の複合文書が第１のオブジェクトを記憶するものであり、当該第１のオブジェクトが前記第２の文書内の第１のデータを指定する第１の情報を記憶するものであり、
第２のオブジェクトをインスタンス生成するステップであって、当該第２のオブジェクトが、前記第２の文書内の第２のデータを指定する第２の情報を記憶するものであり、そして、
複合オブジェクトをインスタンス生成するために前記合成メソッドを呼出すステップであって、当該呼出しが前記第１の情報を前記第２の情報と結合するものである、
を含む方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】
本発明は一般にオブジェクトの名称を付けて結び付けるコンピュータ方法及びシステムに係り、より詳細には、他のオブジェクトに含まれるリンクされたオブジェクトの名称を付け、そのリンクされたオブジェクトのソースを探索してそれに結び付け、そしてオブジェクトを結び付けるのに必要なサーバの呼び出し回数を減少する一般化された方法及びシステムに係る。
【０００２】
【従来の技術】
現在の文書処理コンピュータシステムは、ユーザが複合文書を作成できるようにする。複合文書とは、種々のフォーマットの情報を含む文書である。例えば、複合文書は、テキストフォーマット、チャートフォーマット、数値フォーマット等のデータを含む。図１は複合文書の例である。この例において、複合文書１０１は、ある製造プロジェクトのレポートとして作成される。この複合文書１０１は、チャートフォーマットで表されたスケジュールデータ１０２と、スプレッドシートフォーマットで表された予算データ１０３と、テ

キストフォーマットで表された説明データ１０４とを含んでいる。典型的な公知システムにおいては、ユーザは、プロジェクト管理コンピュータシステムを用いてスケジュールデータ１０２を発生しそしてスプレッドシートコンピュータプログラムを用いて予算データ１０３を発生する。このデータが発生された後に、ユーザは、説明データ１０４を入力し、そしてワードプロセスコンピュータプログラムを用いてスケジュールデータ１０２と予算データ１０３を合体することにより複合文書１０１を作成する。

【０００３】

図２は、スケジュールデータ、予算データ及び説明データを複合文書にいかに合体できるかを示している。ユーザは、プロジェクト管理プログラム２０１を用いてスケジュールデータを発生し、次いで、そのデータをクリップボード２０３に記憶する。ユーザは、スプレッドシートプログラム２０４を用いて予算データを発生し、次いで、そのデータをクリップボード２０３に記憶する。クリップボード２０３は、一般にいかなるプログラムによってもアクセスできる記憶エリア（ディスク又はメモリ）である。プロジェクト管理プログラム２０１及びスプレッドシートプログラム２０４は、典型的に、データをプレゼンテーションフォーマットでクリップボードに記憶する。プレゼンテーションフォーマットとは、データを出力装置に容易に表示するフォーマットである。例えば、プレゼンテーションフォーマットは、標準ビットマップのブロック転送動作（ＢｉｔＢｌｔ）で表示できるビットマップである。データをクリップボードに記憶することを、クリップボードに「コピー」すると称する。

【０００４】

データがクリップボード２０３にコピーされた後に、ユーザはワードプロセスプログラム２０６を始動して複合文書１０１を作成する。ユーザは、説明データ１０４を入力し、そしてクリップボード２０３に存在するスケジュールデータ及び予算データをコピーすべき複合文書１０１内の位置を指定する。クリップボードから文書へデータをコピーすることを、クリップボードから「ペースト」すると称する。次いで、ワードプロセスプログラム２０６は、スケジュールデータ１０２及び予算データ１０３をクリップボード２０３から複合文書１０１の指定の位置へコピーする。クリップボードから複合文書へコピーされるデータを「埋め込み」データと称する。ワードプロセスプログラム２０６は、この埋め込みデータを簡単なビットマップとして処理し、複合文書１０１を出力装置にレンダリングするときにＢｉｔＢｌｔ動作で表示する。ある公知システムにおいては、クリップボードが一度に１つのコピーコマンドに対してデータを記憶することしかできない。このようなシステムでは、スケジュールデータをクリップボードにコピーし、次いで複合文書にペーストすることができる。その後、予算データをクリップボードにコピーし、次いで、複合文書にペーストすることができる。

【０００５】

【発明が解決しようとする課題】

ワードプロセッサは一般にテキストデータしか処理しないので、ワードプロセスプログラムのユーザは、埋め込みデータを移動又は削除することはできるが、データがテキストフォーマットにない限り埋め込みデータを変更することはできない。従って、ユーザは、例えば、複合文書１０１内にある予算データ１０３を変更したい場合には、スプレッドシートプログラム２０４を始動し、予算データ１０３をファイルからロードし、変更を行い、その変更をクリップボード２０３にコピーし、ワードプロセスプログラム２０６を始動し、複合文書１０１をロードし、そしてその修正したクリップボードデータを複合文書１０１へペーストしなければならない。

【０００６】

ある公知システムは、データを実際に埋め込むのではなく複合文書に含まれるべきデータに対するリンクを記憶する。ワードプロセスプログラムがデータをクリップボードから複合文書にペーストするときには、複合文書にリンクが記憶される。このリンクは、含まれるべきデータ（典型的にファイルに存在する）を指す。これらの公知システムは、一般に、データに対するリンクを、ワードプロセスプログラムがプレゼンテーションフォーマッ

トとして認識又は処理するフォーマットで与える。例えば、ワードプロセスプログラム２０６が、ユーザにより、埋め込みではなくリンクによってスケジュールデータ及び予算データを複合文書にペーストするように指示されたときには、そのスケジュールデータ及び予算データがプレゼンテーションフォーマットで存在するファイルの名称が文書に挿入される。多数の複合文書が同じデータに対するリンクを含むことができ、データの１つのコピーを多数の複合文書によって共有することができる。

【０００７】

本発明の目的は、複合文書内に合体されるデータに対するリンクを発生する方法及びシステムを提供することである。

【０００８】

本発明の別の目的は、リンクをその基礎をなすデータに結び付ける方法及びシステムを提供することである。

【０００９】

本発明の別の目的は、これらのリンクを、その基礎をなすデータとは独立したやり方でインターフェイスする方法及びシステムを提供することである。

【００１０】

本発明の更に別の目的は、複合文書内の任意のレベルにネストされたデータにリンクする方法及びシステムを提供することである。

【００１１】

【課題を解決するための手段】

これら及び他の目的は、以下の説明から明らかとなるように、データオブジェクトの名称を付けて結び付ける方法及びシステムによって達成される。好ましい実施例では、合体されるオブジェクトに対するリンクはモニカ（あだな）として記憶される。モニカとは、合体されるデータをアクセスするのに必要な情報をカプセル化しそしてその合体されるデータに連結する方法を与えるようなオブジェクトである。

【００１２】

【実施例】

本発明は、リンクされたデータに名称を付けて結び付ける一般化された方法を提供する。好ましい実施例では、リンクされたデータを合体する複合文書は、リンクソースに対するレファレンスである「モニカ」と称する永続性のデータハンドルを記憶する。モニカは、リンクされたデータを識別するための情報を含むと共にアプリケーションプログラムをそのリンクされたデータに結び付けられるようにする方法を与えるデータオブジェクトである。

【００１３】

結び付けプロセスは、リンクされたデータをアクセスできるようにするインターフェイスの例を返送する。ある場合に、モニカは、それ自体が別の複合文書内の埋め込みデータであるデータをリンクすることができる。例えば、モニカは、ワードプロセス文書に含まれたスプレッドシートテーブル内のある範囲のセルにリンクすることができる。モニカは、複合文書内のいかなるレベルの文書にもリンクすることができる。結び付けプロセス中には、多数のアプリケーションを呼び出して、リンクされたデータを探索することができる。例えば、ワードプロセス文書内にあるスプレッドシート内のある範囲のセルにリンクするために、ワードプロセスプログラムを呼び出して、埋め込まれたスプレッドシートデータを探索できると共に、スプレッドシートプログラムを呼び出してその範囲のセルに結び付けることができる。

【００１４】

本発明は、モニカをアクセスするアブストラクトクラス（インターフェイス）を定義する。典型的に、複合文書にリンクすることのできるデータを与える各アプリケーションプログラムは、モニカインターフェースの実施をサポートする。

【００１５】

本発明の好ましい実施例では、複合文書を形成するアプリケーションプログラムは、別の

アプリケーションによって発生されたリンクされた又は埋め込まれたデータの操作を制御する。オブジェクト指向言語では、このデータをオブジェクトと称する。（１９９１年アディソン・ウェスレイ・パブリッシング社出版のバド．Ｔ著の「オブジェクト指向プログラミングの紹介（Ａｎ　Ｉｎｔｒｏｄｕｃｔｉｏｎ　ｔｏ　Ｏｂｊｅｃｔ－Ｏｒｉｅｎｔｅｄ　Ｐｒｏｇｒａｍｍｉｎｇ）」と題する参考文献は、オブジェクト指向の概念及び用語を紹介している。）複合文書にリンクされるか又は埋め込まれるオブジェクトは、文書内に「収容」される。複合文書は「収容」オブジェクトとも称し、複合文書内に収容されるオブジェクトは「被収容」オブジェクトと称する。図１及び２を参照すれば、スケジュールデータ１０２及び予算データ１０３は被収容オブジェクトであり、複合文書１０１は収容オブジェクトである。ユーザは、予算データ１０３のような被収容オブジェクトを編集したいことをワードプロセッサに指示することができる。ユーザが予算データ１０３を編集することを指示したときに、ワードプロセスプログラムは、どのアプリケーションを用いて予算データを編集すべきか（例えば、スプレッドシートプログラム）を決定し、そしてそのアプリケーションを発進（スタート）する。次いで、ユーザは、その発進したアプリケーションを用いて予算データを操作することができ、複合文書に変更が反映される。予算データが埋め込まれたオブジェクトとして記憶されるかリンクされたオブジェクトとして記憶されるかについても同じ手順が使用される。

【００１６】

図３は、複合文書のサンプルを示すブロック図である。週ごとのプロジェクトレポート３０１は、図１の同じ複合文書である。エグゼクティブサマリーレポート３０３は、週ごとのプロジェクト３０１にリンクされる予算チャート３０５を含んでいる。週ごとのプロジェクト３０１のレポートは、埋め込まれるスプレッドシート３０２を含んでいる。この埋め込まれるスプレッドシート３０２は、図２のスプレッドシートプログラム２０４によって作成されたものである。このスプレッドシートのデータであるプロジェクトの予算は、埋め込みオブジェクトであるから、週ごとのプロジェクトレポート３０１の記憶装置内に記憶される。エグゼクティブサマリー文書３０３は、ネーティブテキスト３０４と、被収容オブジェクトである予算チャート３０５とを含む複合文書である。予算チャート３０５は、複合文書３０１に埋め込まれるスプレッドシート３０２内に収容されたデータとリンクされる。

【００１７】

好ましい実施例では、アプリケーションプログラム（アプリケーション）は、オブジェクトリンク及び埋め込み構成を使用して複合文書を作成し操作するように共働する。複合文書を作成するアプリケーションをクライエントアプリケーションと称し、被収容オブジェクトを形成して操作するアプリケーションをサーバアプリケーションと称する。アプリケーションは、クライエント及びサーバの両方として働くことができる。図２を参照すれば、プロジェクト管理プログラム２０１及びスプレッドシートプログラム２０４はサーバアプリケーションであり、ワードプロセスプログラム２０６はクライエントアプリケーションである。クライエントアプリケーションは、収容オブジェクト内の種々のオブジェクトを選択しそしてその選択した被収容オブジェクトを操作するために適当なサーバアプリケーションを呼び出す役目を果たす。サーバアプリケーションは、被収容オブジェクトの内容を操作する役目を果たす。

【００１８】

好ましい実施例では、オブジェクトをリンクし埋め込む機能を果たす具現独立アプリケーションプログラミングインターフェイス（ＡＰＩ）がアプリケーションに与えられる。このＡＰＩは、クライエント及びサーバアプリケーションによって呼び出される１組の機能である。これらの機能は、他のものの中でも、クライエントアプリケーションがサーバアプリケーションとの間でメッセージ及びデータを送信及び受信するに必要な設定及び初期化を管理する。ＡＰＩは、特定の被収容オブジェクトに作用しそして被収容オブジェクトを操作するために正しいサーバアプリケーションを呼び出す機能を果たす。

【００１９】

更に、オブジェクトをリンクし埋め込むＡＰＩは、クライエントアプリケーションがそれらの収容されたオブジェクトと通信できるようにする「インターフェイス」を定める。インターフェイスとは、ある入力、出力及びふるまいルールによって遵守する１組の方法である。収容されたオブジェクトが特定のインターフェイスをサポートする場合には、クライエントアプリケーションは、そのインターフェイスがその定められたふるまいを実行するための方法を呼び出すことができる。好ましい実施例では、クライエントアプリケーションはオブジェクトデータを直接アクセスすることは許されず、サポートされたインターフェイスを用いてオブジェクトを操作しなければならない。クライエントアプリケーションは、インターフェイスに対するポインタを介して収容されたオブジェクトに結び付けられる。クライエントアプリケーションは、インターフェイスの方法を呼び出すことによりオブジェクトにアクセスする。この方法では、オブジェクトデータにアクセスするために、サーバアプリケーションにメッセージを送って指定のアクセスを要求する。好ましい実施例では、基礎をなすオペレーティングシステムによって与えられるプロセス間通信機構を用いてクライエントとサーバとの間でメッセージが送られる。

【００２０】

クライエントプロセスとサーバプロセスとの間の関係を１つの例で説明する。図１を再び説明すれば、ユーザが複合文書１０１の予算データ１０３を編集しようとする場合、次の一連の事象が生じる。先ず第１に、ユーザはワードプロセッサプログラムをスタートし、これは、オブジェクトをリンクして埋め込むＡＰＩに動的にリンクしている。第２に、ユーザは複合文書を編集のためにオープンする。第３に、ユーザは、被収容オブジェクトである予算データを選択し、その選択したオブジェクトを編集すべきことを指示する。第４に、クライエントアプリケーションがクライエントＡＰＩルーチンを呼び出し、オブジェクトに対するハンドル（選択されたオブジェクトを独特に識別する）と、動作が編集であることを示すインジケータとをルーチンに通すというオブジェクトの動作を実行する。第５に、クライエントＡＰＦルーチンは、スプレッドシートプログラムが予算データに対して動作を与えることを決定する。第６に、クライエントＡＰＩコードは、スプレッドシートプログラムがまだスタートされていなければ、これをサーバプロセスとしてスタートさせる。第７に、ワードプロセッサアプリケーションは、予算データを編集しなければならないスプレッドシートプログラムにメッセージを送る。第８に、サーバＡＰＩコードは、編集の要求を受け取り、データを編集するためのスプレッドシートプログラムのルーチンを呼び出す。編集が完了すると、スプレッドシートルーチンはサーバＡＰＩコードに復帰する。サーバＡＰＩコードはワードプロセッサアプリケーションにメッセージを送って、編集が完了したことを指示する。クライエントＡＰＩコードはメッセージを受け取ってその呼び出しから復帰する。呼び出しから復帰すると、ワードプロセッサアプリケーションは、編集が完了したことを知る。

【００２１】

クライエント及びサーバＡＰＩに加えて、本発明のオブジェクトリンク及び埋め込み構成体は、永続的なグローバル「レジストリ（登録装置）」を経てクライエント及びサーバアプリケーションに情報を与える。このレジストリは、（１）オブジェクトの各形式ごとに、そのオブジェクトの形式を具現するサーバアプリケーション、（２）各サーバアプリケーションがクライエントアプリケーションに与える動作、（３）各サーバアプリケーションの実行可能なファイルがどこに配置されているか、そして（４）各サーバアプリケーションが関連オブジェクトハンドラーを有しているかどうかといった情報のデータベースである。

【００２２】

オブジェクト指向の言語においては、インターフェイスは、「アブストラクト（抽象）クラス」であり、即ちデータ及び方法の定義は有するが、これら方法の具現はもたないクラスである。これは、オブジェクト「クラス」に対してサーバアプリケーションの役目を果たし、オブジェクトを操作するのに使用できる方法に対する実際のコードを与える。

【００２３】

図４は、オブジェクトのサンプル例を示すブロック図である。好ましい実施例において、この例のレイアウトは、「仮想機能及び仮想ベースクラスを具現化し、オブジェクト指向のプログラミング言語のためにこのポインターを設定するための方法及びシステム（Ｍｅｔｈｏｄ ａｎｄ Ｓｙｓｔｅｍ ｆｏｒ Ｉｍｐｌｅｍｅｎｔｉｎｇ Ｖｉｒｔｕａｌ Ｆｕｎｃｔｉｏｎｓ ａｎｄ Ｖｉｒｔｕａｌ Ｂａｓｅ Ｃｌａｓｓｅｓ ａｎｄ Ｓｅｔｔｉｎｇ ａ Ｔｈｉｓ ｐｏｉｎｔｅｒ ｆｏｒ ａｎ Ｏｂｊｅｃｔ－Ｏｒｉｅｎｔｅｄ Ｐｒｏｇｒａｍｍｉｎｇ Ｌａｎｇｕａｇｅ）」と題する米国特許第５，２９７，２８４号に開示されたモデルに適合する。この例は、オブジェクトデータ構造体４０１と、各々のサポートされるインターフェイス毎のインターフェイスデータ構造体４１３とを含んでいる。オブジェクトデータ構造体４０１は、インターフェイスデータ構造体４１３に対するポインタ４０２を含んでおり、そしてこの例の専用データを含むことができる。このサンプル例の専用データは、クラス識別子４０３（ＣＬＡＳＳ ＩＤ）と、オブジェクトの記憶に対するハンドル４０４と、オブジェクトの状態を追跡するデータ４０５とを含む。クラス識別子は、そのオブジェクトに対して適当なサーバアプリケーションをアクセスするのに使用される。これは、プログラミング言語に用いられるデータ構造体「タイプ」と同様である。インターフェイスはＣＬＡＳＳ ＩＤを永続的グローバルレジストリへのインデックスとして使用することにより、オブジェクトに対するサーバを決定することができる。図４に示すように、各インターフェイスデータ構造体４１３は、専用データ構造体４０６及び仮想機能テーブル４０８を含んでいる。専用データ構造体４０６は、仮想機能テーブル４０８に対するポインタ４０７を含む。仮想機能テーブル４０８は、インターフェイスの方法を具現化するコードのポインタを含む。

【００２４】

テーブル１ ＃はインターフェイスクラスを定める インターフェイスｉｎｔｆ｛ｐｕｂｌｉｃ： 仮想ＲＥＴＣＯＤＥ ｐｆｎｃ１ （ａｒｇ１，ａｒｇ２）＝０；仮想ＲＥＴＣＯＤＥ ｐｆｎｃ２ （ａｒｇ１，ａｒｇ２）＝０；
仮想ＲＥＴＣＯＤＥ ｐｆｎｃ３ （ ）＝０；
・・・ ｝；

【００２５】

上記のテーブル１は、オブジェクトデータ構造体４０１の第１エントリーｐｉｎｔｆ１に対するインターフェイスの定義を表している。テーブル１において、「インターフェイス」という語は、Ｃ＋＋クラスを意味するものとして定義される。この定義は、アーギュメントをもつ３つの方法を示している。各アーギュメントリストの端の「＝０」は、その方法がコードの具現をもたないことを示している。Ｃ＋＋のプログラミング言語では、これらの機能を「純仮想機能」と称する。全て純仮想機能を有するクラスをアブストラクトクラスと称する。

【００２６】

図５は、オブジェクトのパブリックビューを示すブロック図である。オブジェクトのパブリックビューとは、オブジェクトがサポートする種々のインターフェイス５０２ないし５０６である。各インターフェイスは、クライエントアプリケーションがオブジェクトをアクセスできるようにする方法を与える。各オブジェクトはＩＵｎｋｎｏｗｎインターフェイス５０２をサポートする。アプリケーションはこのＩＵｎｋｎｏｗｎインターフェイス５０２を用いて、他のどのインターフェイスがオブジェクトをサポートするか決定する。特定のオブジェクトに対してＩＵｎｋｎｏｗｎインターフェイス５０２を具現すると、他のどのインターフェイスがそれをサポートするかが分かり、そのインターフェイスに対する呼び出しアプリケーションポインタに復帰する。好ましい実施例では、この目的で、方法ＩＵｎｋｎｏｗｎ：：ＱｕｅｒｙＩｎｔｅｒｆａｃｅが使用される。インターフェイス５０３ないし５０６は、オブジェクトによってサポートすることのできる典型的なインターフェイスの例である。例えば、ＩＣｒｅａｔｅインターフェイス５０３は、オブジェクトの新たな例を形成する方法を与える。ＩＣｏｎｔａｉｎｅｒインターフェイス５０４は、オブジェクト内に含まれる被収容オブジェクトをリストする方法を与える。ＩＤａｔａ

ＣｌｉｅｎｔＳｉｔｅインターフェイス５０５は、収容オブジェクトと通信するサーバにより使用されるべき方法を与える。ＩＤａｔａＯｂｊｅｃｔインターフェイス５０６は、オブジェクトデータを操作する方法を与える。

【００２７】

図６は、ＩＭｏｎｉｋｅｒインターフェイスの典型的な例を示すブロック図である。この例のデータ構造体は、オブジェクトデータ構造体６０１と、仮想機能テーブル６０３とを含んでいる。オブジェクトデータ構造体６０１は、仮想機能テーブル６０３に対するポインタ６０２と、専用例データとを含んでいる。仮想機能テーブル６０３は、ＩＭｏｎｉｋｅｒインターフェイスの方法を具現化するコードに対するポインタを含んでいる。テーブル２は、ＩＭｏｎｉｋｅｒインターフェイスのクラスの定義である。

【００２８】

テーブル２　クラス　ＩＭｏｎｉｋｅｒ　｛　仮想　ＳＣＯＤＥ　ＢｉｎｄＴｏＯｂｊｅｃｔ（ｐｂｃ，ｉｉｄＴｏＬｅｆｔ，ｐｖＴｏＬｅｆｔ，ｐｍｋＴｏＬｅｆｔ，　ｌｉｄＲｅｓｕｌｔ，ｐｐｖｒｅｓｕｌｔ）＝０；　仮想　ＳＣＯＤＥ　ＩＩＤＢｉｎｄＴｏＯｂｊｅｃｔ（ｐｉｄｄ）＝０；　仮想　ＳＣＯＤＥ　ＢｉｎｄＴｏＳｔｏｒａｇｅ（ｐｂｃ，ｐｍｋＴｏＬｅｆｔ，ｉｉｄ，ｐｐｖＯｂｊ）＝０；　仮想　ＳＣＯＤＥ　Ｒｅｄｕｃｅ（ｐｂｃ，ｄｗＲｅｄｕｃｅＨｏｗＦａｒ，ｐｐｍｋＴｏＬｅｆｔ，ｐｐｍｋＲｅｄｕｃｅｄ）＝０；　仮想　ＳＣＯＤＥ　ＣｏｍｐｏｓｅＷｉｔｈ（ｐｍｋＲｉｇｈｔ，ｆＯｎｌｙｉｆＮｏｔＧｅｎｅｒｉｃ，ｐｐｍＣｏｍｐｏｓｅ）　仮想　ＳＣＯＤＥ　Ｅａｔｅｎ（ｆＦｏｗａｒｄ，ｐｐｃｎｍＭｏｎｉｋｅｒ）＝０；　仮想　ＳＣＯＤＥ　ＩｓＧｅｎｅｒｉｃ　Ｃｉｍｐｏｓｉｔｅ（）＝０；　仮想　ＳＣＯＤＥ　ＩｓＥｑｕａｌ（ｆｉｄｅｎｔｉｃａｌ，ｐｍｋＯｔｈｅｒＭｏｎｉｋｅｒ）＝０；　仮想　ＳＣＯＤＥ　Ｈａｓｈ（ＩＩｄｅｎｔｉｃａｌ，ｐｄｗＨａｓｈ）＝０；　仮想　ＳＣＯＤＥ　ＧｅｔＴｉｍｅＯｆＬａｓｔＣｈａｎｇｅ（ｐｂｃ，ｐｍｋＴｏＬｅｆｔ，ｐｆｉｌｅｔｉｍｅ）＝０；　仮想　ＳＣＯＤＥ　Ｉｎｖｅｒｓｅ（ｐｐｍｋ）＝０；　仮想　ＳＣＯＤＥ　ＣｏｍｍｏｎＰｒｅｆｉｘＷｉｔｈ（ｐｍｋＯｔｈｅｒ，ｐｐｍｋＰｒｅｆｉｘ）＝０；　仮想　ＳＣＯＤＥ　ＲｅｌａｔｉｖｅＰａｔｈＴｏ（ｐｍｋＯｔｈｅｒ，ｐｐｍｋＲｅｌＰａｔｈ）；　仮想　ＳＣＯＤＥ　ＧｅｔＵｓｅｒＮａｍｅ（ｐｂｃ，ｐｍｋＴｏＬｅｆｔ，ＬｐＬｐｓｚＵｓｅｒＮａｍｅ）＝０；　仮想　ＳＣＯＤＥ　ＰｕｒｓｅＵｓｅｒＮａｍｅ（ｐｂｃ，ｐｍｋＴｏＬｅｆｔ，ｌｐｓｚＵｓｅｒＮａｍｅ，ｐｃｈＥａｔｅｎ，　ｐｐｐｍｋＯｕｔ）＝０；　｝

【００２９】

これらの方法に加えて、ＩＭｏｎｉｋｅｒの具現は、典型的に、ＭｋＰａｒｓｅＵｓｅｒＮａｍｅ、ＭｋＣｒｅａｔｅＩＢｉｎｄＣｏｎｔｅｘｔ、ＭｋＣｒｅａｔｅＧｅｎｅａｉｃＣｏｍｐｏｓｉｔｅ、ＭｋＣｒｅａｔｅＦｉｌｅＭｏｎｉｋｅｒ、ＭｋＣｒｅａｔｅＩｔｅｍＭｏｎｉｋｅｒ及びＭｋＣｒｅａｔｅＡｎｔｉＭｏｎｉｋｅｒの機能をサポートし、これらについては以下に述べる。

【００３０】

図２３は、リンクされたオブジェクトを示す図である。この例では、複合文書２２０１は、複合文書１０１に記憶されたレポートのような種々のプロジェクトレポートの編集物である。複合文書２２０１は、複合文書２２０２へのリンクを含んでいる。このリンクは、ファイル「ＶＡＣ１．ＤＯＣ」に記憶された複合文書が理論的に複合文書２２０１内にあることを示している。好ましい実施例においては、リンクがモニカとして記憶される。モニカは、リンクのソースへアクセスする方法を与えるクラス（モニカオブジェクト）の例である。モニカオブジェクトは、リンクのソースを識別するデータをカプセル化し、アプリケーションプログラムがリンクのソースをアクセスできるようにする方法を与える。例えば、複合文書２２０１から複合文書２２０２へのリンクが形成されたときは、ワードプロセッサは、「ＶＡＣ１．ＤＯＣ」ファイルを指すモニカオブジェクトの例を挙げる機能を呼び出す。この機能は、新たなモニカオブジェクトに対するポインタを返送する。次いで、ワードプロセッサは、そのモニカプロジェクトを複合文書２２０１内に記憶する。ワ

ードプロセッサは、モニカオブジェクトの方法を介してリンクソースへアクセスする。例えば、モニカのクラスが「結び付け」方法をサポートする。この結び付け方法は、リンクソースがどこに配置されているかを決定し、そのソースを表すオブジェクトに対するポインタを返送する。この例では、モニカオブジェクトの結び付け方法は、ソースがファイル「ＶＡＣ１．ＤＯＣ」に記憶されていると決定し、ファイルを表すためにオブジェクトを例示し、その例示されたオブジェクトに対するポインタを返送する。次いで、ワードプロセッサは、その返送されたオブジェクトでソースデータをアクセスする方法を呼び出す。

【００３１】

図７は、図３のエグゼクティブサマリー複合文書３０３に記憶されるモニカを示すブロック図である。このモニカは、リンクされたチャートオブジェクト３０５を表している。このリンクは複合文書内に埋め込まれたデータを指すので、モニカは、複合文書を識別するのに必要なリンクと、複合文書内のデータを識別するのに必要なリンクとの合成体として示されている。従って、リンクを合成モニカと称する。合成モニカは、埋め込みハイアラーキ構成において左から右に順序付けされた成分モニカを含んでいる。合成モニカ７０１は、３つの成分モニカ７０２、７０３及び７０４を備えている。成分モニカ７０２は、複合文書３０１を参照する。成分モニカ７０２は、ＣＬＡＳＳ　ＩＤ（ＦｉｌｅＭｏｎｉｋｅｒ）と、複合文書３０１のユーザ読み取り可能な名称「Ｃ：＼ＶＡＣ１．ＤＯＣ」とを含んでいる。成分モニカ７０３は、複合文書３０１内に埋め込まれたオブジェクトを参照する。ワードプロセスアプリケーションは、成分モニカ７０３を理解しそしてこれを用いて埋め込まれたスプレッドシートを位置決めする。成分モニカ７０３は、ＣＬＡＳＳ　ＩＤ（ＷＩｔｅｍＭｏｎｉｋｅｒ）と、埋め込まれたスプレッドシートのユーザ読み取り可能な名称とを含んでいる。ＣＬＡＳＳ　ＩＤ「ＷＩｔｅｍＭｏｎｉｋｅｒ」は、ワードプロセッサがこの成分モニカを認識することを示している。成分モニカ７０４は、埋め込まれたスプレッドシートオブジェクト３０２内のデータの範囲を参照する。この成分モニカ７０４は、ＣＬＡＳＳ　ＩＤ（ＳＩｔｅｍＭｏｎｉｋｅｒ）と、その範囲に対してユーザが読み取り可能な名称とを含んでいる。ＣＬＡＳＳ　ＩＤ（ＳＩｔｅｍＭｏｎｉｋｅｒ）は、スプレッドシートプログラムがこの成分モニカを認識することを指示する。合成モニカのオブジェクトデータ構造体は、データ構造体８０１及び仮想機能テーブル８０２を含んでいる。このデータ構造体８０１は、ＣＬＡＳＳ　ＩＤ８０３と、リンクされたリスト８０５に対するポインタ８０４とを含んでいる。この例では、ＣＬＡＳＳ　ＩＤは、タイプＩＤ　ＧｅｎＣｏｍｐｏｓｉｔｅＭｏｎｉｋｅｒのモニカを指示する。

【００３２】

図８は、典型的な合成モニカオブジェクトデータ構造体を示すブロック図である。リンクされたリスト８０５は、各成分モニカごとに１つのエントリー８０６を含んでいる。各エントリーは、成分モニカ８０９ないし８１１に対するポインタを含んでいる。

【００３３】

図９は、一般的な合成モニカ方法ＣｏｍｐｏｓｅＷｉｔｈの具現化を示すフローチャートである。方法ＧｅｎＣｏｍｐｏｓｉｔｅＭＯｎｉｋｅｒ：：ＣｏｍｐｏｓｅＷｉｔｈは、指定されたモニカを左部分にそして呼称されたモニカを右部分に有する新たな合成モニカを形成する。この方法は、新たな合成モニカに対するポインタを返送する。ステップ９０１において、この方法は、呼称されたモニカのＩｓＧｅｎｅｒｉｃＣｏｍｐｏｓｉｔｅ？方法を呼び出し、それが一般的な合成モニカであるかどうかの判断をする。もしそうでなければ、この方法はステップ９０４へ続き、さもなくば、ステップ９０２へ続く。ステップ９０２において、この方法は、指定されたモニカの方法であるＦｉｎｄＦｉｒｓｔＥｌｅｍｅｎｔ方法を呼び出し、これは、第１の成分モニカに対するポインタと、残りの成分モニカに対するポインタとを返送する。ステップ９０３では、この方法は、第１成分モニカのＣＬＡＳＳ　ＩＤを抽出する。ステップ９０４では、呼称されたモニカのＣＬＡＳＳ　ＩＤが抽出される。ステップ９０３及び９０４が完了すると、この方法は、新たな合成モニカの右部分として使用するモニカのＣＬＡＳＳＩＤを得る。ステップ９０７において、この方法は、ＭｋＣｒｅａｔｅＧｅｎｅｒｉｃＣｏｍｐｏｓｉｔｅ機能を呼び出して、

左側の指定されたモニカと、右側の呼称されたモニカとで成る新たな一般的な合成モニカを形成し、次いで、その新たな合成モニカに対するポインタを返送する。

【００３４】

図１０は、機能ＭｋＣｒｅａｔｅＧｅｎｅｒｉｃＣｏｍｐｏｓｉｔｅのフローチャートである。この機能は、左右の２つのモニカを得、左右のモニカより成る新たな合成モニカを形成し、その新たなモニカに対するポインタを返送する。ステップ１００１では、この機能は、ｐｍｋＦｉｒｓｔ：：ＩｓＧｅｎｅｒｉｃＣｏｍｐｏｓｉｔｅ？方法を呼び出して、左のモニカが一般的な合成モニカであるかどうか判断する。もしそうであれば、この機能はステップ１００４へ続き、さもなくば、ステップ１００２へ続く。ステップ１００２では、新たな一般的な合成モニカを割り当て、そして左のモニカのコピーをその第１エレメントとして記憶する。更に、ステップ１００２において、必要なデータ構造体を割り当てると共に、新たな一般的な合成モニカのノードをリンクされたリストに割り当て、そしてこの値を変数ｐｒｅｖｉｏｕｓに記憶する。ステップ１００３では、通過された左側のモニカ（ｐｍｋＦｉｒｓｔ）を、第１のエレメントとして、ｐｃｍで指示された新たな合成モニカに挿入する。この方法は、次いで、ステップ１００８へと続く。ステップ１００４ないし１００７では、元の左側モニカの成分モニカを含む新たな合成モニカを形成するように機能がループする。このループは、元の左側モニカのリンクされたリストのエレメントを横切り、成分モニカのコピーを新たな合成モニカに挿入する。ステップ１００４では、左側モニカの最後のエレメントをサーチするために多数の変数が初期化される。これを行うために、変数ｐｒｅｖｉｏｕｓをリンクされたリストの第１エレメントを指すようにセットし、変数ｃｕｒｒｅｎｔをリンクされたリストの第２エレメントを指すようにセットし、そしてｐｃｍをリンクされたリストの第１エレメントによって指示された第１モニカにセットする。ステップ１００５では、変数ｃｕｒｒｅｎｔがＮＵＬＬであるかどうか判断される。もしそうであれば、変数ｐｒｅｖｉｏｕｓは左の合成モニカの最後のエレメントを指し、これは、全ての成分モニカがその新たな合成モニカに対して無効であったことを示すもので、この機能はステップ１００８へと続き、さもなくば、ステップ１００６へと続く。ステップ１００６では、変数ｐｃｍによって指示されたモニカが変数ｃｕｒｒｅｎｔによって指示されたモニカを通すＣｏｍｐｏｓｅＷｉｔｈ方法を呼び出す。ステップ１００７では、この新たな合成モニカを指すように変数ｐｃｍをセットし、リンクされたリストの次の２つのエレメントを指すように変数ｐｒｅｖｉｏｕｓ及びｃｕｒｒｅｎｔを更新し、そしてステップ１００５へとループして、変数ｐｒｅｖｉｏｕｓがリンクされたリスト終わりを指すかどうか判断する。ステップ１００８では、ｐｍｋＲｅｓｔＩｓＧｅｎｅｒｉｃＣｏｍｐｏｓｉｔｅ？方法を呼び出し、右側のモニカが一般的な合成モニカであるかどうか判断する。もしそうでなければ、この機能は、ステップ１０１０へと続き、さもなくば、ステップ１００９へと続く。ステップ１００９では、右側モニカのリンクされたリストを新たな左側モニカに添付する。この機能はステップ１０１２へと続く。ステップ１０１０及び１０１１において、右側のモニカを指す新たなエレメントが新たな左側のモニカに添付される。ステップ１０１２において、返送値ｐｐｍｋＣｏｍｐｏｓｉｔｅを新たな合成モニカであるｐｃｍにセットし、この機能は復帰する。

【００３５】

図１１は、典型的な具現化のＦｉｎｄＬａｓｔＥｌｔ方法のフローチャートである。この方法は２つのモニカを形成する。第１のモニカは、指定されたモニカの最後の成分モニカに対応する。第２のモニカは、指定されたモニカの残り部分に対応する。ステップ１１０１において、この方法は、リンクされたリストの第２エレメントを指すように変数ｃｕｒｒｅｎｔをセットし、リンクされたリストの第１エレメントを指すように変数ｐｒｅｖｉｏｕｓをセットし、そして第１の成分モニカを指すように変数ｒｅｓｔをセットする。ステップ１１０２ないし１１０５は、指定されたモニカのリンクされたリストを横切り、第２のモニカを形成する。ステップ１１０２において、変数ｃｕｒｒｅｎｔが最後のエレメントを指す場合には、この方法はステップ１１０６に続き、さもなくば、ステップ１１０３に続く。ステップ１１０６では、変数ｃｕｒｒｅｎｔによって指示されたモニカを指す

ようにパラメータＬａｓｔＥｌｔをセットし、復帰となる。

【００３６】

図１２は、典型的な具現化のＦｉｎｄＦｉｒｓｔＥｌｔ方法のフローチャートである。この方法は、ＦｉｎｄＬａｓｔＥｌｔ方法と同様であるが、第１の成分モニカに対するポインタと、残りの成分モニカより成る新たな合成モニカに対するポインタとを返送する。

【００３７】

図１３は、典型的な具現化のＢｉｎｄＴｏＯｂｊｅｃｔ方法を示すフローチャートである。このＢｉｎｄＴｏＯｂｊｅｃｔ方法は、ＢｉｎｄＴｏＯｂｊｅｃｔ（又はＢｉｎｄＴｏＳｔｏｒａｇｅ）を繰り返し呼び出し、指定されたモニカにより指示されたオブジェクトを探索して結び付けるのに使用できるオブジェクト（又は記憶）を得るものである。一般に、ＢｉｎｄＴｏＯｂｊｅｃｔ方法は、そのプレフィックス成分モニカの方法を使用して、指定されたモニカによって指示されたオブジェクト又は記憶を返送する。この方法は、最後の成分モニカのＢｉｎｄＴｏＯｂｊｅｃｔ方法を呼び出し、これは次いで右から左へそのプレフィックスモニカのＢｉｎｄＴｏＯｂｊｅｃｔ方法を呼び出す。ＢｉｎｄＴｏＯｂｊｅｃｔ方法は、次の６つのパラメータを取るものである。即ち、ｐｏｉｎｔｅｒｔｏａｂｉｎｄコンテクスト、第３パラメータのインターフェイス識別子、プレフィックスモニカに対応するオブジェクトのインターフェイスのポインタ、プレフィックスモニカ、呼び出し側が結び付けようとしているインターフェイス、及び例示されたインターフェイスに対応する出力パラメータ。ステップ１３０１において、指定されたモニカのＦｉｎｄＬａｓｔＥｌｔ方法を呼び出し、その最後の成分モニカを見つける。ステップ１３０２において、最後の成分モニカのＩｉｄＢｉｎｄＴｏＯｂｊｅｃｔ方法を呼び出す。ステップ１３０３において、この方法がＩＤ　ＮＵＬＬを返送する場合には、最後の成分モニカのＢｉｎｄＴｏＯｂｊｅｃｔ方法を呼び出して、結び付けが必要なインターフェイスを決定することが必要であり、この方法はステップ１３０４へと進み、さもなくば、ステップ１３０５へと続く。ステップ１３０４において、ＢｉｎｄＴｏＯｂｊｅｃｔ方法は最後の成分モニカのＢｉｎｄＴｏＯｂｊｅｃｔ方法を呼び出し、合成モニカの第１部分をそのプレフィックスモニカとして通して、復帰となる。ステップ１３０５において、最後の成分モニカのＩｉｄＢｉｎｄＴｏＯｂｊｅｃｔ方法によって返送されたインターフェイスがＩＤ　ＳＴＯＲＡＧＥに等しい場合には、この方法はステップ１３０６へ続き、さもなくば、ステップ１３０７へ続く。ステップ１３０６において、この方法は、プレフィックスモニカのＢｉｎｄＴｏＳｔｏｒａｇｅ方法を呼び出す（変数ｒｅｓｔ）。返送の際に、変数ｐｖは、プレフィックスモニカに対応する記憶オブジェクトの結び付けられた例を指し、ステップ１３０８へと続く。ステップ１３０７において、プレフィックスモニカに対するＢｉｎｄＴｏＯｂｊｅｃｔ方法を呼び出す。最後の成分モニカによって要求されたインターフェイスを通し、そのインターフェイスに対応する例示されたオブジェクトが変数ｐｖにおいて返送される。ステップ１３０８では、変数ｐｖによって指示された例示されたオブジェクトが、その通された結び付けコンテクストに登録される。この結び付けコンテクストは、ＢｉｎｄＴｏＯｂｊｅｃｔ方法の以前の呼び出しによって対応的に結び付けられた現在例示されているオブジェクトのリンクされたリストである。この結び付けコンテクストは、ＢｉｎｄＴｏＯｂｊｅｃｔ方法によって使用されて、その指定されたモニカにより参照されたオブジェクトが既に例示されたかどうかが決定される。ステップ１３０９においては、最後の成分モニカによって要求されたインターフェイス、モニカプレフィックスに対応する例示されたインターフェイス、及び元のＢｉｎｄＴｏＯｂｊｅｃｔ呼び出しで要求された元のインターフェイスで最後の成分モニカのためのＢｉｎｄＴｏＯｂｊｅｃｔ方法を呼び出す。次いで、この方法は復帰となる。

【００３８】

図１４は、ファイルモニカの典型的な具現化に対するＢｉｎｄＴｏＯｂｊｅｃｔ方法を示すフローチャートである。この方法は、指定されたファイルモニカに対応するオブジェクトがランニングオブジェクトテーブルにより指示されたように存在するかどうかを決定し、そのオブジェクトに対するポインタを返送する。ランニングオブジェクトテーブルは、

例示された全ての現在オブジェクトに対するポインタを含んでいる。さもなくば、この方法は適当なサーバを探し、ファイルに対するインターフェイスを例示する。ＦｉｌｅＭｏｎｉｋｅｒ：：ＢｉｎｄＴｏＯｂｊｅｃｔ方法は、一般的な合成モニカＢｉｎｄＴｏＯｂｊｅｃｔ方法でリストした同じ６つのパラメータを取るものである。ステップ１４０１ないし１４０４において、この方法は、対応するオブジェクトに対してランニングオブジェクトテーブルを使用して、指定されたファイルモニカが存在するかどうか判断する。ステップ１４０１において、この方法は、指定されたモニカがランニングオブジェクトテーブルにリストされているかどうかを判断する。もしそうであれば、この方法はステップ１４０２へ続き、さもなくば、ステップ１４０５へ続く。ステップ１４０２において、この方法は、要求されたインターフェイスがランニングオブジェクトテーブル内のものと一致するかどうかを判断する。もし一致すれば、ステップ１４０３において、そのインターフェイスに対するポインタを返送する。さもなくば、ステップ１４０４において、ランニングオブジェクトテーブルで見つかったオブジェクトのＱｕｅｒｙＩｎｔｅｒｆａｃｅ方法を呼び出すことによりその要求されたインターフェイスを得、そして復帰となる。ステップ１４０５において、指定されたモニカがランニングオブジェクトテーブルに見つからなかったために、この方法は、指定されたモニカに対応するサーバを見つける（ｆｉｌｅｎａｍｅ）。ステップ１４０６において、サーバを呼び出してそのクラスの例を形成する。ステップ１４０７において、その例のＱｕｅｒｙＩｎｔｅｒｆａｃｅ方法を呼び出し、結び付けのためのインターフェイスが得られる。ステップ１４０８では、結び付けインターフェイスのＬｏａｄ方法が呼び出され、復帰となる。

【００３９】

図１５は、項目モニカを典型的に具現化するＢｉｎｄＴｏＯｂｊｅｃｔ方法のフローチャートである。上記の例では、ＷＩｔｅｍＭｏｎｉｋｅｒ及びＳＩｔｅｍＭｏｎｉｋｅｒの両方が同様の具現構成である。この方法は、プレフィックスモニカのインターフェイスオブジェクトを使用し、そのＧｅｔＯｂｊｅｃｔ方法を呼び出す。このＧｅｔＯｂｊｅｃｔ方法は、ＩｔｅｍＭｏｎｉｋｅｒによって指示されたオブジェクトの例を探索してそれに結び付ける。この方法は、一般的な合成モニカのＢｉｎｄＴｏＯｂｊｅｃｔ方法に通された同じ６つのパラメータを利用する。ステップ１５０１では、結び付けのためにオブジェクトが通されたかどうか判断する。もしそうであれば、この方法はステップ１５０２へと続き、さもなくば、ステップ１５０３へと続く。ステップ１５０２では、その通されたインターフェイスのＧｅｔＯｂｊｅｃｔ方法が呼び出され、項目モニカによって指示されたオブジェクトが例示され、復帰となる。ステップ１５０３では、プレフィックスモニカがＮＵＬＬでない場合、この方法はステップ１５０４へ続き、さもなくば、復帰となる。ステップ１５０４では、レフィックスモニカのＢｉｎｄＴｏＯｂｊｅｃｔ方法が呼び出され、ＩＣｏｎｔａｉｎｅｒインターフェイスが要求される。ステップ１５０５では、その要求されたＩＣｏｎｔａｉｎｅｒインターフェイスのＧｅｔＯｂｊｅｃｔ方法が呼び出される。

【００４０】

図１６は、ＷＩｔｅｍＭｏｎｉｋｅｒを典型的に具現化するＢｉｎｄＴｏＳｔｏｒａｇｅ方法のフローチャートである。この方法は、そのプレフィックスのＢｉｎｄＴｏＳｔｏｒａｇｅ方法を呼び出し、そしてプレフィックスに対する記憶ハンドルを得たときに、指定されたモニカに関連した記憶を得る。ステップ１６０１では、その通されたプレフィックスがＮＵＬＬであるかどうか判断し、もしそうであれば、この方法は復帰となり、さもなくば、ステップ１６０２へ続く。ステップ１６０２では、プレフィックスモニカのＢｉｎｄＴｏＳｔｏｒａｇｅ方法が呼び出され、その要求されたインターフェイスを通すようにする。ステップ１６０３では、ステップ１６０２で返送された記憶のＧｅｔＳｔｏｒａｇｅ方法が呼び出され、現在ＷｉｔｅｍＭｏｎｉｋｅｒに指定されたオブジェクトに対する独特の識別子が通される。次いで、この方法は復帰となる。

【００４１】

図１７は、ＧｅｎｅｒｉｃＣｏｍｐｏｓｉｔｅＭｏｎｉｋｅｒインターフェイスの減少方

法を示す全体的なフローチャートである。このルーチンは、各成分モニカの減少方法を呼び出すことにより各成分モニカを減少しようと試み、そして成分の減少された新たなＧｅｎｅｒｉｃＣｏｍｐｏｓｉｔｅＭｏｎｉｋｅｒを構成する。この方法は、３つのパラメータ、即ち結び付けコンテクストのポインタと、プレフィックスモニカと、減少したモニカに対する出力パラメータとを利用する。ステップ１７０１において、この方法は、指定されたモニカを構成する成分モニカのリンクされたリストにおける第１エレメントを見つける。ステップ１７０２ないし１７０９において、この方法は、各成分モニカの減少方法を繰り返し呼び出しそして成分の減少された新たな合成モニカを発生する。ステップ１７０２において、リンクされたリストの全てのエレメントが処理された場合に、この方法はステップ１７０３へと続き、さもなくば、ステップ１７０４へと続く。ステップ１７０３において、出力パラメータを新たな減少された合成モニカにセットし、そして復帰となる。ステップ１７０４では、変数Ｍ　Ｅｌｅｍｅｎｔによって指示されたモニカ成分の減少方法を呼び出し、結び付けコンテクスト及びプレフィックスモニカのポインタに沿って通すようにする。プレフィックスモニカは、それまでに形成された新たな合成モニカに対応する。ステップ１７０５において、プレフィックスモニカを指す変数がＮＵＬＬであることがこの方法により決定されると、ステップ１７０６へ続き、さもなくば、ステップ１７０７へ続く。ステップ１７０６において、プレフィックスモニカを指す変数がＮＵＬＬを指す場合（ループを最初に通るときに生じる）には、これがステップ１７０４において返送された減少された成分モニカにセットされ、そしてこの方法はステップ１７０９に続く。ステップ１７０７において、プレフィックスモニカを指す変数がＮＵＬＬでない場合には、プレフィックスモニカのＣｏｍｐｏｓｅＷｉｔｈ方法を呼び出し、ステップ１７０４で返送された新たな減少されたモニカをプレフィックスと合成する。ステップ１７０８において、プレフィックスモニカを指す変数は、行われた全てのモニカ減少を含む新たな合成された結果にリセットされる。ステップ１７０９において、変数Ｍ　Ｅｌｅｍｅｎｔは、成分モニカのリンクされたリストの次のエレメントに進められ、そしてこの方法はステップ１７０２においてループの上部へ復帰する。

【００４２】

図１８は、ＦｉｌｅＭｏｎｉｋｅｒを典型的に具現化する減少方法のフローチャートである。この減少方法は、通されたプレフィックスモニカを探し、そしてそのプレフィックスモニカの最後の成分が別のファイルモニカであるかどうかを判断する。別のファイルモニカである場合は、２つのＦｉｌｅＭｏｎｉｋｅｒを１つに結合したものである新たな合成ＦｉｌｅＭｏｎｉｋｅｒを返送する。又、通されたプレフィックスから消費されたファイルモニカを除去する。この機能は３つのパラメータ、即ち結び付けコンテクストのポインタと、プレフィックスモニカのポインタと、新たに減少されたモニカが返送される場合の出力パラメータとを利用する。ステップ１８０１において、この機能は、通されたプレフィックスモニカを検査して、それがＮＵＬＬであるかどうかを調べる。もしそうであれば、この機能は復帰となり、さもなくば、ステップ１８０２へ続く。ステップ１８０２において、この機能はプレフィックスモニカを探してそれが一般的な合成モニカであるかどうかを調べる。もしそうであれば、機能はステップ１８０４へ続き、さもなくば、ステップ１８０３へ続く。ステップ１８０３において、この機能は、プレフィックスが単純なモニカであるので最後のエレメントをプレフィックスモニカにセットし、ステップ１８０５へと続く。ステップ１８０４において、レフィックスモニカのＦｉｎｄＬａｓｔＥｌｔ方法を呼び出し、プレフィックス合成モニカの最後のエレメントを得る。ステップ１８０５では、最後のエレメントのＣＬＡＳＳ　ＩＤがチェックされて、それがファイルモニカであるかどうか調べられる。もしそうでなければ、この機能はステップ１８０６へ続き、さもなくば、ステップ１８０７へ続く。ステップ１８０６では、減少が生じていないので、出力パラメータをそれ自身へセットし、復帰となる。ステップ１８０７では、このモニカに関連したファイル名を、最後のエレメントによって指示されたモニカのファイル名に添付する。次いで、ステップ１８０８において、この新たなファイル名ストリングでそれ自身の形成方法を呼び出す。ステップ１８０９において、プレフィックスを、プレフィックス

から最後のエレメントを引いたものに等しくセットするか、又はプレフィックスが合成モニカでなかった場合にはＮＵＬＬにセットする。ステップ１８１０において、新たに減少されたモニカに対応する出力パラメータを、ステップ１８０８で形成されたモニカにセットし、そしてこの機能は復帰となる。

【００４３】

図１９及び２０は、ワードプロセス項目モニカＷＩｔｅｍＭｏｎｉｋｅｒを典型的に具現化する減少方法の全体的なフローチャートである。このルーチンは、モニカの特殊な種類である記憶モニカを返送し、これは、オブジェクト結び付け方法によって使用されて、埋め込まれたオブジェクトのサーバを直接呼び出し、ワードプロセスアプリケーションの呼び出しをバイパスできるようにする。これは、ワードプロセス文書内に含まれた特殊オブジェクトテーブルを探して、埋め込まれたオブジェクトに対する適当な記憶ポインタを見つけることにより行われる。この記憶ポインタは、次いで、記憶モニカに合成され、そのオブジェクト結び付け方法が呼び出されるときに、どのサーバを呼び出すかを決めることができる。このルーチンは、３つのパラメータ、即ち結び付けコンテクストのポインタと、プレフィックスモニカと、新たに減少されたモニカを含む出力パラメータとを利用する。ステップ１９０１では、この方法は、プレフィックスがＮＵＬＬであるかどうか調べ、もしそうであれば、復帰となるが、さもなくばステップ１９０２へと続く。ステップ１９０２では、この方法は、プレフィックスが一般的な合成モニカであるかどうか調べ、もしそうであれば、ステップ１９０３へ続き、さもなくば、ステップ１９０４へ続く。ステップ１９０３では、プレフィックスモニカのＦｉｎｄＬａｓｔＥｌｔ方法を呼び出して、リンクされたリストにおける最終モニカのポインタを得る。次いで、ステップ１９０５へと続く。ステップ１９０４において、最終エレメントポインタをプレフィックスにセットする。というのは、これが合成モニカでないからである。ステップ１９０５では、最終エレメントのＣＬＡＳＳ　ＩＤを探し、それがファイルモニカであるかどうか調べる。もしそうでなければ、この機能はステップ１９０６へ続き、さもなくば、ステップ１９０７へ続く。ステップ１９０６では、単にそれ自身を出力パラメータにおいて返送する。というのは、減少が行われたと考えられないからである。ステップ１９０７では、ファイルの拡張を探すが、これは、ファイルがワードプロセスアプリケーションに属しているかどうかを判断することであると分かる。ファイルが未知の形式のものであると判断されると、この機能はステップ１９０６へ続き、さもなくば、ステップ１９０８へ続く。ステップ１９０８では、最終エレメントポインタによって指示されたモニカのＧｅｔＵｓｅｒ方法が呼び出され、プレフィックスをもつ最終エレメントのファイル名を得る。ステップ１９０９において、このようにして得たファイル名でオブジェクトテーブル内の既知の位置を探し、現在モニカに対応するエントリーを見つける。テーブル内のエレメントは、３つのタプル（ＷＩｔｅｍＭｏｎｉｋｅｒ、ＩＳｔｏｒａｇｅ、ＵｓｅｒＮａｍｅ）の形態である。このテーブルから、記憶ポインタと、形成される新たなモニカのユーザ名とを得る。次いで、ステップ１９１０において、プレフィックスの最終エレメントのファイル名のユーザ名を、得られたこの新たな名称に添付する。ステップ１９１１では、記憶ポインタ及び新たな名称をもつ新たなＩＳｔｏｒａｇｅモニカを形成する。ステップ１８１２では、プレフィックスが、プレフィックスから最終エレメントを引いた残りにセットされる。最後に、ステップ１９１３において、出力パラメータがこのように形成された新たなＩＳｔｏｒａｇｅモニカにセットされ、これで機能は復帰となる。

【００４４】

図２１は、機能ＭｋＰａｒｓｅＵｓｅｒＮａｍｅの全体的なフローチャートである。この機能はストリング名を利用し、それに対応するモニカを返送する。この返送されるモニカは、合成モニカである。ステップ２００１において、この機能はストリングを分解して正当なファイル名を見つける。ステップ２００２において、このファイル名のファイルモニカを形成し、それを変数ｐｆｍにセーブする。ステップ２００３では、残りのストリングを、分解した最初のストリングから消費された文字を引いたものにセットする。最後に、ステップ２００４において、ｐｆｍにより指示されたファイルモニカに対するＰａｒｓｅ

ＵｓｅｒＮａｍｅ方法を呼び出し、残っているストリングと、返送モニカのアドレスとを通すようにする。ファイルモニカのＰａｒｓｅＵｓｅｒＮａｍｅ方法がいったん返送されると、ストリングの残りが分解される。次いで、この機能は復帰となる。

【００４５】

図２２は、ファイルモニカを典型的に具現化するＰａｒｓｅＵｓｅｒＮａｍｅ方法の全体的なフローチャートである。この方法は、これが含むことのできるオブジェクトの種類が分からないので、ストリングの次の部分を分解するためにファイル名によって指示されたオブジェクトに結び付ける必要があるかどうかを判断しなければならない。この方法は、４つのパラメータ、即ちＢｉｎｄＣｏｎｔｅｘｔへのポインタ、プレフィックスモニカのポインタ、分解されるべき残りのストリング名、及び得られるモニカのポインタを利用する。ステップ２１０１では、この方法は、このファイルに含まれたオブジェクトの構文を理解できるかどうか判断する。もしできれば、ステップ２１０２へ続き、さもなくば、ステップ２１０３へ続く。ステップ２１０２では、内部分解ストリングルーチンを呼び出し、分解することのできたストリングの部分に対するモニカを形成する。次いで、ステップ２１０５へと続く。ステップ２１０３において、この方法は、それ自体のＢｉｎｄＴｏＯｂｊｅｃｔを呼び出し、ＰａｕｓｅＵｓｅｒＮａｍｅ方法に通されたプレフィックスを通すようにする。次いで、ステップ２１０４において、結び付けられたオブジェクトのＰａｕｓｅＵｓｅｒＮａｍｅ方法を呼び出し、プレフィックスと、分解されるべき残りのユーザストリングとを通すようにする。最後に、ステップ２１０５において、プレフィックスのＣｏｍｐｏｓｅＷｉｔｈ方法が呼び出されて、新たに形成されたモニカを追加する。これで、この機能は復帰となる。

【００４６】

以上、本発明の好ましい実施例について説明したが、本発明はこれに限定されるものではない。当業者であれば、本発明の範囲内で種々の変更がなされ得ることが明らかであろう。従って、本発明は特許請求の範囲のみによって限定されるものとする。

【図面の簡単な説明】

【図１】複合文書の例を示す図である。

【図２】スケジュールデータ、予算データ及び説明データを複合文書にいかに合体するかを示す図である。

【図３】リンクされたオブジェクトとそのソースとの関係の一例を示す図である。

【図４】オブジェクトのサンプル例を示すブロック図である。

【図５】オブジェクトのパブリックビューを示すブロック図である。

【図６】ＩＭｏｎｉｋｅｒインターフェイスの典型例を示すブロック図である。

【図７】図３のエグゼクティブサマリー複合文書に記憶されるモニカを示すブロック図である。

【図８】典型的な合成モニカオブジェクトデータ構造体を示すブロック図である。

【図９】一般的な合成モニカ方法ＣｏｍｐｏｓｅＷｉｔｈの具現化を示すフローチャートである。

【図１０】機能ＭｋＣｒｅａｔｅＧｅｎｅｒｉｃＣｏｍｐｏｓｉｔｅを示すフローチャートである。

【図１１】２つのモニカを形成するＦｉｎｄＬａｓｔＥｌｔ方法の典型的な具現化を示すフローチャートである。

【図１２】ＦｉｎｄＦｉｒｓｔＥｌｔ方法の典型的な具現化を示したフローチャートである。

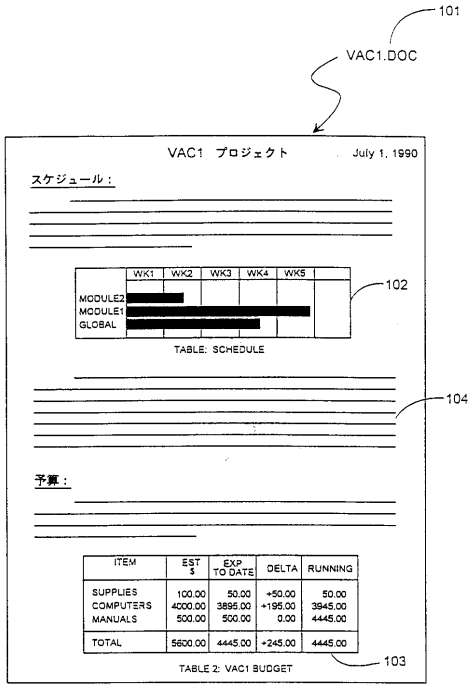
【図１３】ＢｉｎｄＴｏＯｂｊｅｃｔ方法の典型的な具現化を示したフローチャートである。

【図１４】ファイルモニカを典型的に具現化するＢｉｎｄＴｏＯｂｊｅｃｔ方法のフローチャートである。

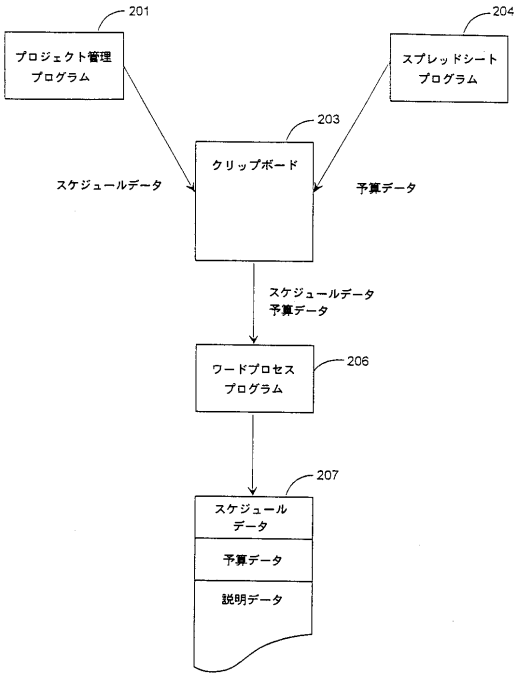
【図１５】項目モニカを典型的に具現化するＢｉｎｄＴｏＯｂｊｅｃｔ方法のフローチャートである。

【図１６】ＷＩｔｅｍＭｏｎｉｋｅｒを典型的に具現化するＢｉｎｄＴｏＳｔｏｒａｇｅ方法を示すフローチャートである。
【図１７】一般的な合成モニカインターフェイスの減少方法を示す全体的なフローチャートである。
【図１８】ファイルモニカを典型的に具現化する減少方法の全フローチャートである。
【図１９】ワードプロセス項目モニカＷＩｔｅｍＭｏｎｉｋｅｒを典型的に実施する減少方法のフローチャートである。
【図２０】ワードプロセス項目モニカＷＩｔｅｍＭｏｎｉｋｅｒを典型的に実施する減少方法の続きのフローチャートである。
【図２１】機能ＭｋＰａｒｓｅＵｓｅｒＮａｍｅの全体的なフローチャートである。
【図２２】ファイルモニカを典型的に具現化するＰａｒｓｅＵｓｅｒＮａｍｅ方法の全体的なフローチャートである。
【図２３】リンクされたオブジェクトを示す図である。
【符号の説明】
１０１　複合文書
１０２　スケジュールデータ
１０３　予算データ
１０４　説明データ
２０１　プロジェクト管理プログラム
２０３　クリップボード
２０４　スプレッドシートプログラム
２０６　ワードプロセスプログラム
３０１　週ごとのプロジェクトレポート
３０２　埋め込まれたスプレッドシート
３０３　エグゼクティブサマリー文書
３０４　ネーティブテキスト
３０５　予算チャート

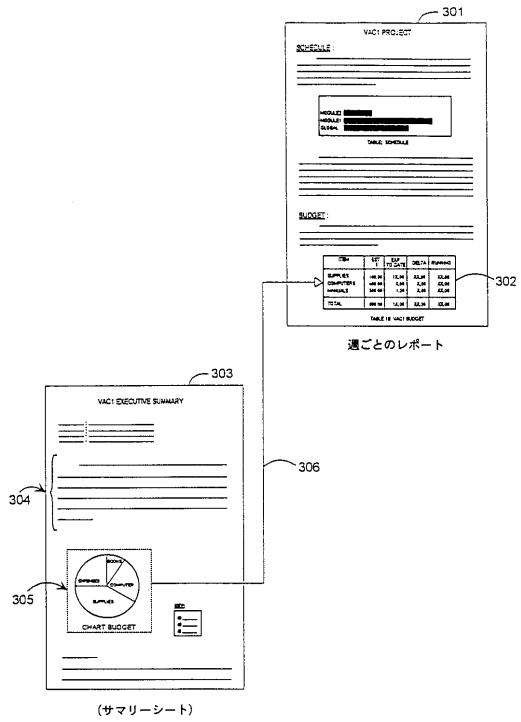
【図１】


101
VAC1.DOC
VAC1 プロジェクト
July 1, 1990
スケジュール：
WK1 WK2 WK3 WK4 WK5
MODULE2
MODULE1
GLOBAL
102
TABLE: SCHEDULE
104
予算：
ITEM
EST $
EXP TO DATE
DELTA
RUNNING
SUPPLIES 100.00 50.00 +50.00 50.00
COMPUTERS 4000.00 3895.00 +195.00 3945.00
MANUALS 500.00 500.00 0.00 4445.00
TOTAL 5600.00 4445.00 +245.00 4445.00
103
TABLE 2: VAC1 BUDGET

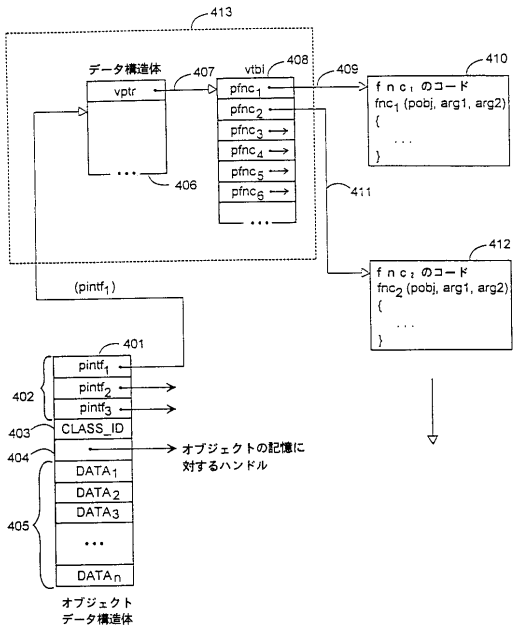
【図２】


201
プロジェクト管理プログラム
204
スプレッドシートプログラム
203
クリップボード
スケジュールデータ
予算データ
スケジュールデータ
予算データ
ワードプロセスプログラム
206
207
スケジュールデータ
予算データ
説明データ

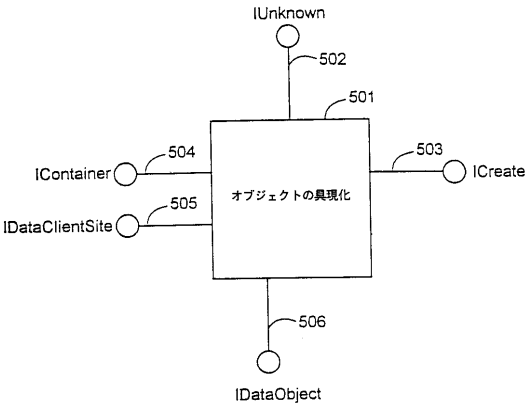
【図３】


301
VAC1 PROJECT
SCHEDULE :
BUDGET :
302
週ごとのレポート
303
VAC1 EXECUTIVE SUMMARY
304
305
306
CHART BUDGET
(サマリーシート)

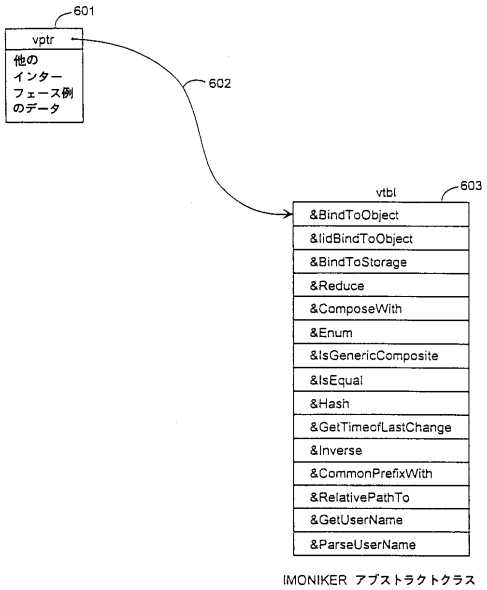
【図４】


413
データ構造体
407
vptr
vtbl
408
409
410
pfnc1
pfnc2
pfnc3
pfnc4
pfnc5
pfnc6
406
411
412
(pintf1)
401
pintf1
pintf2
pintf3
402
403
CLASS_ID
404
DATA1
DATA2
DATA3
405
DATAn
オブジェクトの記憶に対するハンドル
オブジェクトデータ構造体

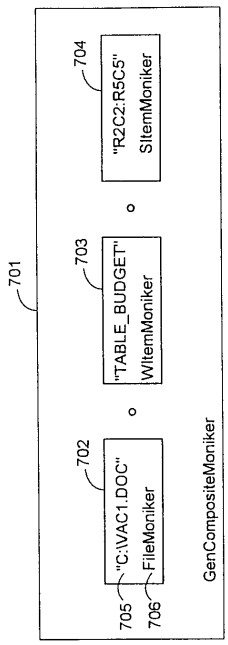
【図５】


IUnknown
502
501
504
IContainer
505
IDataClientSite
オブジェクトの具現化
503
ICreate
506
IDataObject

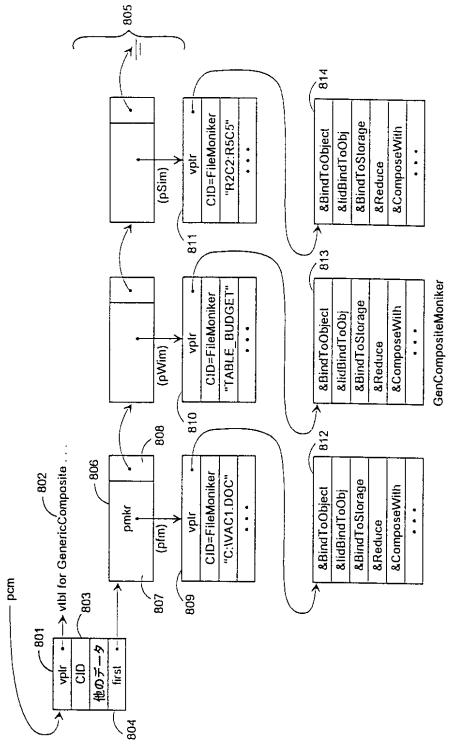
【図６】


601
vptr
他の
インター
フェース例
のデータ
602
vtbl
603
&BindToObject
&IidBindToObject
&BindToStorage
&Reduce
&ComposeWith
&Enum
&IsGenericComposite
&IsEqual
&Hash
&GetTimeofLastChange
&Inverse
&CommonPrefixWith
&RelativePathTo
&GetUserName
&ParseUserName
IMONIKER アブストラクトクラス

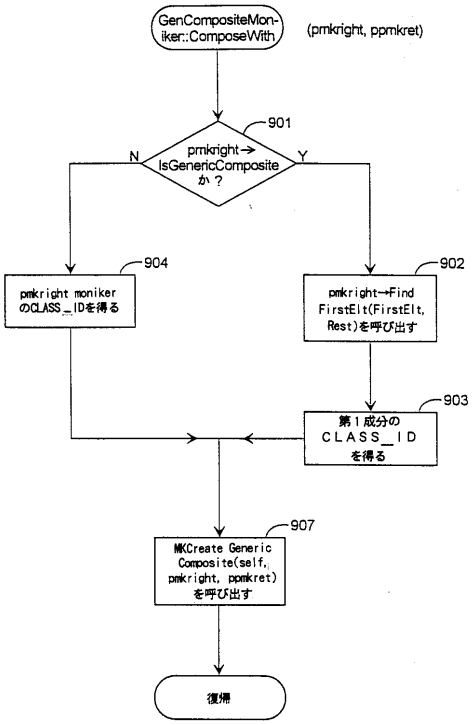
【図７】


701
704
"R2C2:R5C5"
SItemMoniker
703
"TABLE_BUDGET"
WItemMoniker
702
"C:\VAC1.DOC"
FileMoniker
705
706
GenCompositeMoniker

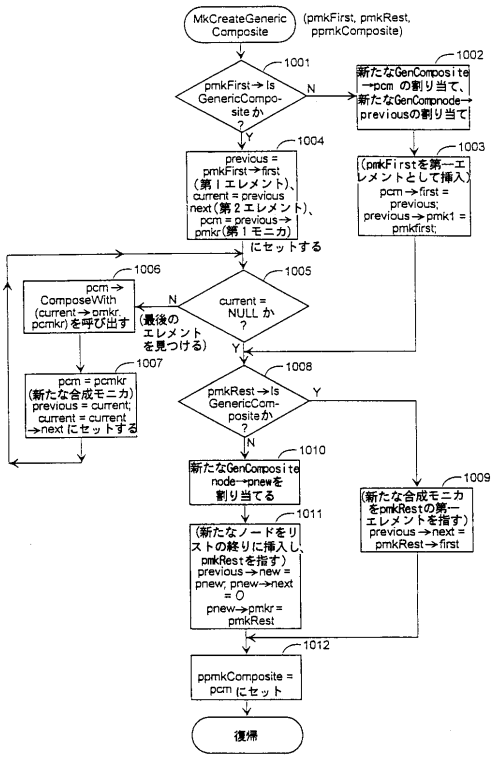
【図８】


805
(pSim)
vptr
CID=FileMoniker
"R2C2:R5C5"
811
814
&BindToObject
&IidBindToObj
&BindToStorage
&Reduce
&ComposeWith
(pWim)
"TABLE_BUDGET"
810
813
GenCompositeMoniker
806
808
802
pmkr
(plm)
"C:\VAC1.DOC"
807
809
812
pcm
vtbl for GenericComposite
803
801
CID
他のデータ
first
804

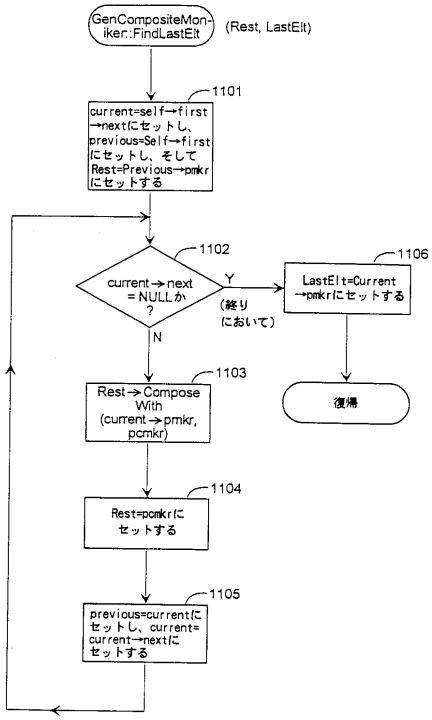
【図９】



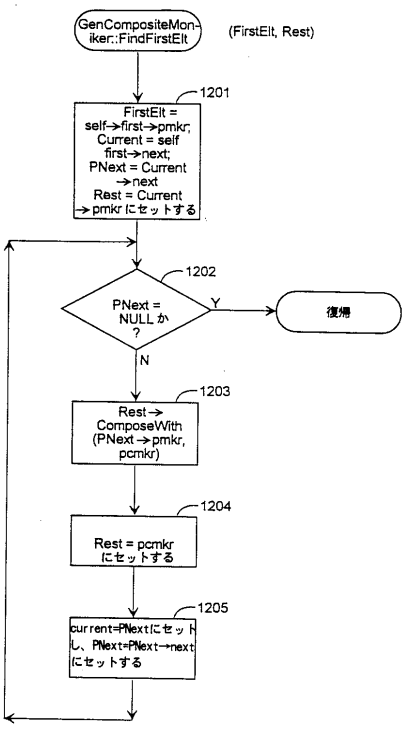
【図１０】



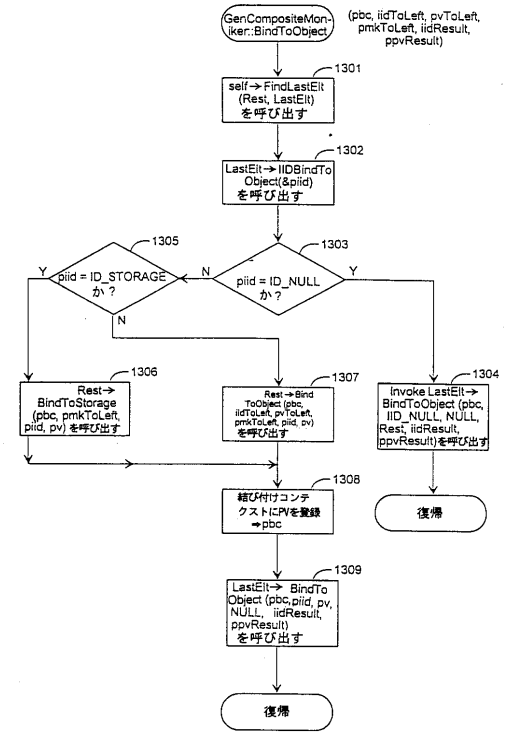
【図１１】



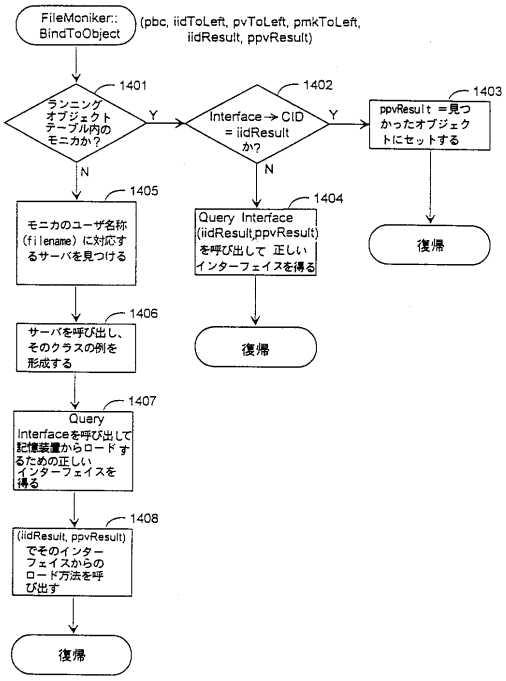
【図１２】



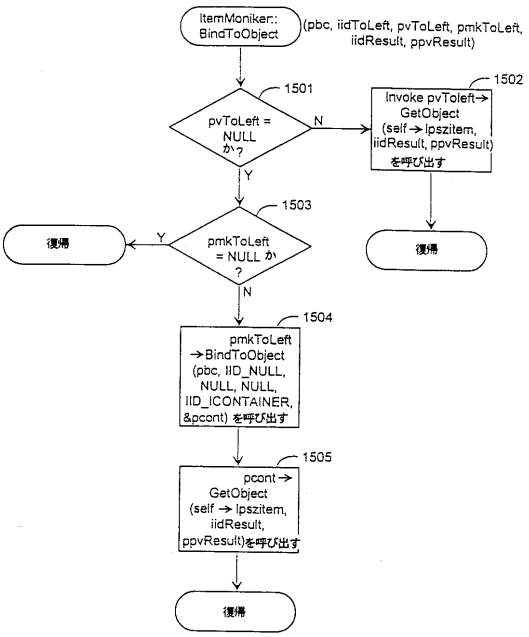
【図１３】



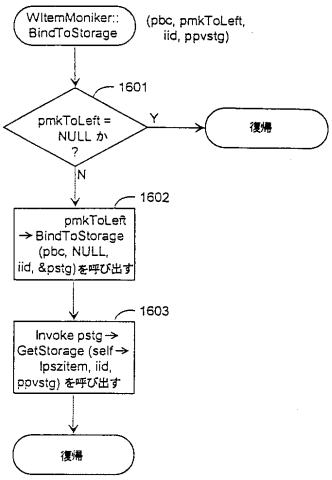
【図１４】



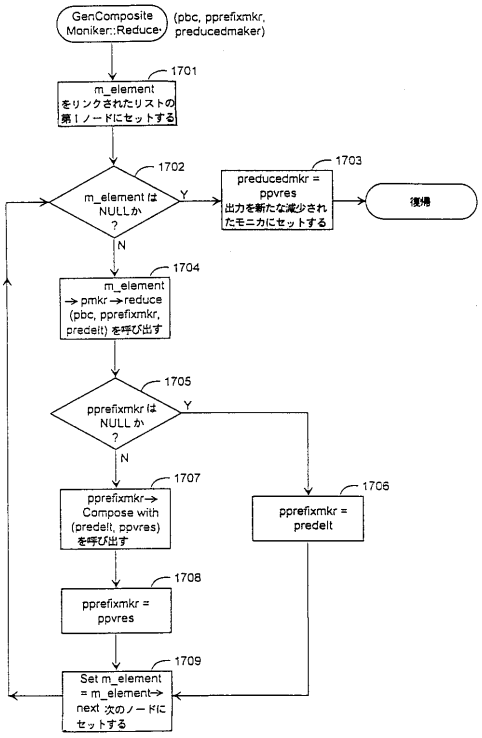
【図１５】



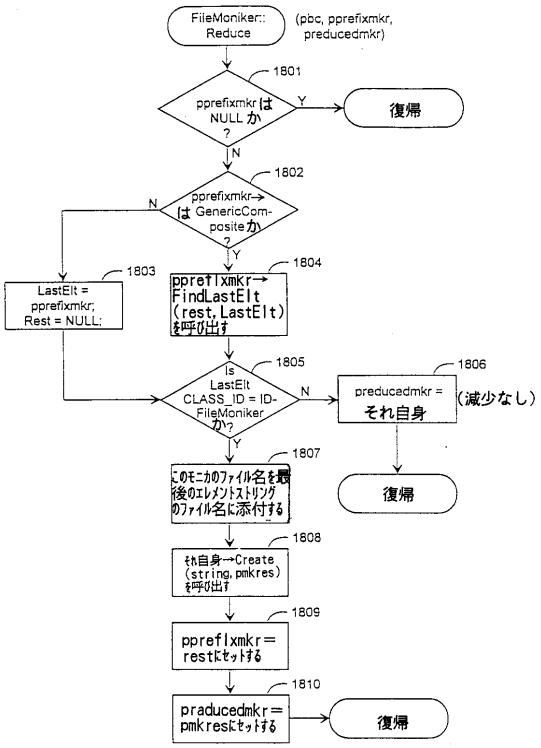
【図１６】



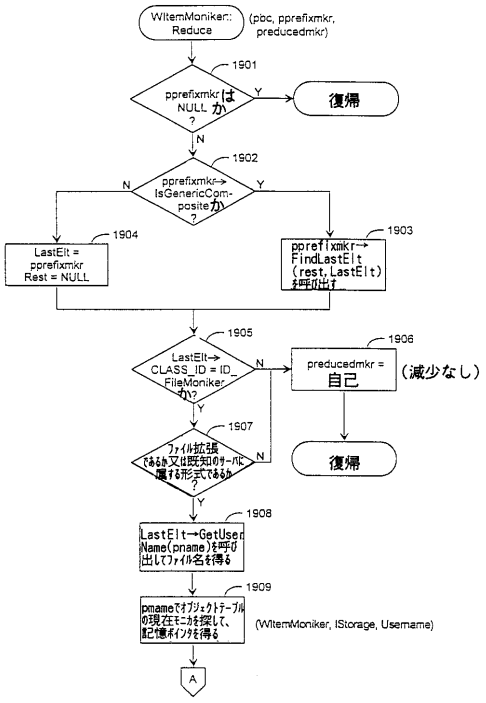
【図１７】



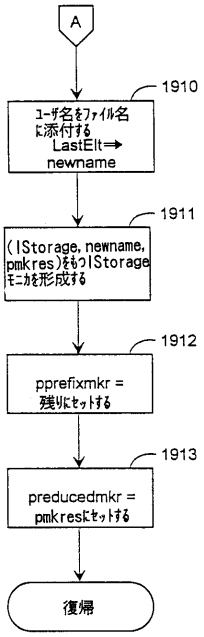
【図１８】



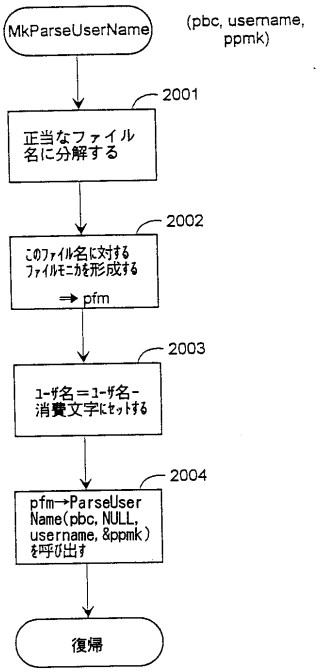
【図１９】



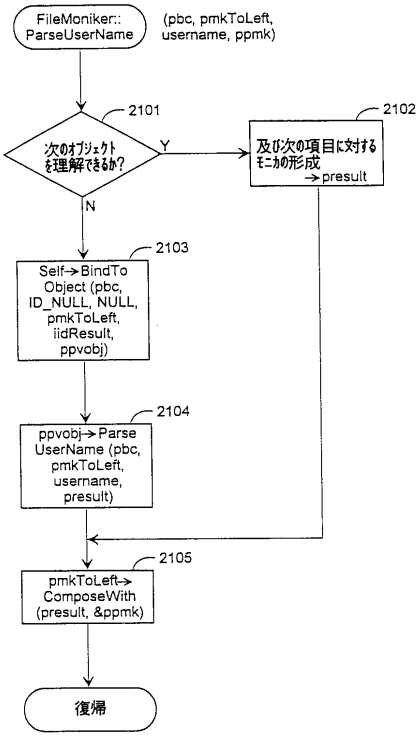
【図２０】



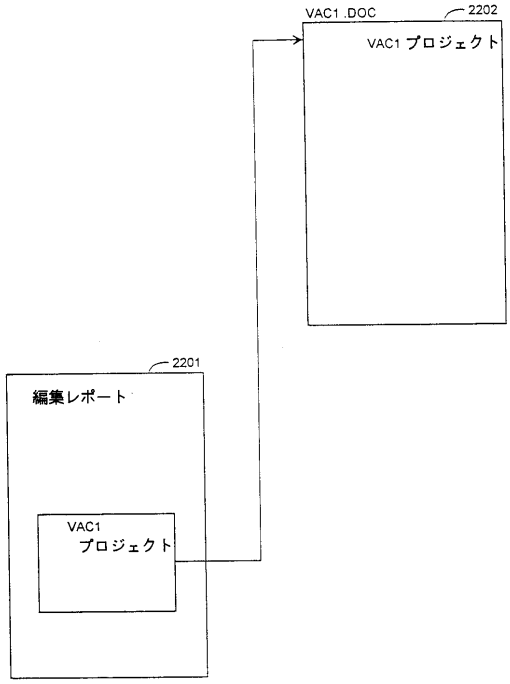
【図２１】


MkParseUserName
(pbc, username, ppmk)
2001
正当なファイル名に分解する
2002
このファイル名に対するファイルモニカを形成する
⇒ pfm
2003
ユーザ名＝ユーザ名-消費文字にセットする
2004
pfm→ParseUserName(pbc, NULL, username, &ppmk)を呼び出す
復帰

【図２２】


FileMoniker::ParseUserName
(pbc, pmkToLeft, username, ppmk)
2101
次のオブジェクトを理解できるか?
Y
N
2102
及び次の項目に対するモニカの形成
→presult
2103
Self→BindTo Object (pbc, ID_NULL, NULL, pmkToLeft, iidResult, ppvobj)
2104
ppvobj→Parse UserName (pbc, pmkToLeft, username, presult)
2105
pmkToLeft→ComposeWith (presult, &ppmk)
復帰

【図２３】


VAC1 .DOC
2202
VAC1 プロジェクト
2201
編集レポート
VAC1
プロジェクト

フロントページの続き

(74)代理人 100084009
弁理士 小川 信夫
(74)代理人 100082821
弁理士 村社 厚夫
(72)発明者 ロバート ジー アトキンソン
アメリカ合衆国 ワシントン州 ９８０７２ ウッディンヴィル ノースイースト ワンハンドレッドアンドナインティシックススストリート １７９２６
(72)発明者 アントニー エス ウィリアムス
アメリカ合衆国 ワシントン州 ９８０５３ レッドモンド ノースイースト フォーティシックスス ストリート ２２５４２
(72)発明者 エドワード ケイ ユング
アメリカ合衆国 ワシントン州 ９８０５２ レッドモンド ワンハンドレッドアンドフィフティシックスス アベニュー ノースイースト ４３０６ アパートメント エイエイ－１０１

審査官 石川 正二

(56)参考文献 特開昭６１－１５６２８９（ＪＰ，Ａ）
特開平０２－０７７８７２（ＪＰ，Ａ）
特開平０３－１９１４２９（ＪＰ，Ａ）
大用昌之、秋山秀樹，Ｗｉｎｄｏｗｓアプリケーション間連携機構ＯＬＥ，日経バイト，日本，日経ＢＰ社，１９９２年 2月 1日，第９６号，ｐ．２３９－２５２

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 12/00
G06F 17/21