(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3691515号
(P3691515)
(45)発行日　平成17年9月7日(2005.9.7)　(24)登録日　平成17年6月24日(2005.6.24)

(51)Int.Cl.[7]　F I
G06F 9/54　G06F 9/46 480C

請求項の数 24 (全 29 頁)

(21)出願番号　特願平7-529840
(86)(22)出願日　平成7年5月15日(1995.5.15)
(65)公表番号　特表平10-500509
(43)公表日　平成10年1月13日(1998.1.13)
(86)国際出願番号　PCT/US1995/006074
(87)国際公開番号　WO1995/031781
(87)国際公開日　平成7年11月23日(1995.11.23)
審査請求日　平成14年5月15日(2002.5.15)
(31)優先権主張番号　08/242,204
(32)優先日　平成6年5月13日(1994.5.13)
(33)優先権主張国　米国(US)

(73)特許権者
アップル　コンピュータ，　インコーポレイテッド
アメリカ合衆国　カリフォルニア州　９５０１４　クッパチーノ，　ワン　インフィナィト　ループ（番地なし）
(74)代理人
弁理士　大塚　康徳
(72)発明者　ズィマンスキー，　スティーブン，　ジエイムズ
アメリカ合衆国　カリフォルニア州　９５０１４　クッパチーノ，ロドリーグス　アベニュー　２００６０　アパートメント　ディー　（番地なし）

最終頁に続く

(54)【発明の名称】オペレーティングシステムにおけるイベント分配装置及び方法

(57)【特許請求の範囲】
【請求項１】
コンピュータ内で発生したイベントを検出し、イベントを発生する少なくとも１つのイベント生成部と、コンピュータ内でイベントが発生したときにその通知を必要とする少なくとも１つのイベントコンシューマとを含むコンピュータにおいてイベントを分配するためのシステムであって、
前記少なくとも１つのイベントコンシューマが通知を受けるべき特定のイベントセットを格納する格納手段と、
前記イベント生成部からのイベントを受信し、受信したイベントと前記格納されたイベントセットとを比較し、適切なイベントコンシューマに適切なイベントを分配する制御手段と、
前記制御手段から前記イベントを受信し、適切なイベントコンシューマに適切なイベントを分配させるべく前記制御手段を動作させる分配手段と
を備えることを特徴とするイベント分配システム。
【請求項２】
前記分配手段はコンピュータ内で発生し得る各イベントの種類毎に分配モジュールを備えることを特徴とする請求の範囲第１項に記載のシステム。
【請求項３】
複数のイベントコンシューマが前記コンピュータ内に含まれ、該複数のコンシューマは、他のコンシューマに対して独立に、且つ、コンシューマがイベントの通知を受けた順には

無関係に動作する、互いのコンシューマ間について関連性を有しないブロードキャストコンシューマと、
それぞれが自身の処理を行なっている間は、イベントが他のコンシューマには通知されないことを要求し、他のコンシューマに関連するイベントを受信したときには影響を及ぼす能力を有する、互いに関連付けられたシーケンシャルコンシューマとを含むことを特徴とする請求の範囲第１項に記載のシステム。
【請求項４】
前記分配手段は、
ブロードキャストコンシューマがイベントを受信するまでは後続するイベントに基づいてそのイベントをモディファイする権利を前記分配手段が維持することを示す信号と共に、ブロードキャストコンシューマにそのイベントを送信するかどうかを判断する手段と、
前記判断する手段による肯定判定に応じて、前記適切なブロードキャストコンシューマに対して、イベントをモディファイする権利を前記分配手段が有することを示す信号とともに前記イベントを分配するべく前記制御手段を動作させる手段とを備えることを特徴とする請求の範囲第３項に記載のシステム。
【請求項５】
前記分配手段は、
前記分配手段が前記イベントをモディファイする権利を維持することを示す信号と共に、前記イベントがブロードキャストコンシューマに送信されるべきかどうかを判断する第１の手段と、
前記第１の手段による肯定判断に応じて、前記イベントが全てのブロードキャストコンシューマに渡されるべきものかを判断する第２の手段と、
前記第２の手段による肯定判断に応じて、前記分配手段が前記イベントをモディファイする権利を維持することを示す信号とともに、全てのブロードキャストコンシューマに該イベントを分配するべく前記制御手段を動作させる手段と、
前記第２の手段による否定判定に応じて、前記分配手段が前記イベントをモディファイする権利を維持することを示す信号とともに、選択されたブロードキャストコンシューマに該イベントを分配するべく前記制御手段を動作させる手段と、
前記第１の手段の否定判定に応じて、そのイベントが全ブロードキャストコンシューマに対して送信されるべきものかを判断する第３の手段と、
前記第３の手段による肯定判断に応じて、そのイベントを全てのブロードキャストコンシューマに分配するべく前記制御手段を動作させる手段と、
前記第３の手段による否定判断に応じて、選択されたブロードキャストコンシューマにイベントを分配するべく前記制御手段を動作させる手段とを備えることを特徴とする請求の範囲第３項に記載のシステム。
【請求項６】
前記格納手段は、
ブロードキャストコンシューマに関連付けられたイベントのサブスクリプションを格納するサブスクリプションマトリクスと、
シーケンシャルコンシューマに関連付けられたイベントのエントリを格納するシーケンシャルコンシューマデータベースとを備えることを特徴とする請求の範囲第３項に記載のシステム。
【請求項７】
前記格納手段は各ブロードキャストコンシューマに対応するイベントキューを備え、該イベントキューは、前記制御手段から分配されたイベントを受信し、対応するブロードキャストコンシューマによってイベントが消費されるまで分配されたそのイベントを格納することを特徴とする請求の範囲第３項に記載のシステム。
【請求項８】
前記制御手段は、前記シーケンシャルコンシューマデータベースにおけるエントリに従って、イベントを連続してシーケンシャルコンシューマに渡す手段を備えることを特徴とす

る請求の範囲第３項に記載のシステム。
【請求項９】
前記制御手段は、シーケンシャルコンシューマからの、イベントをハンドルしたことを示すメッセージの受信に応じて、イベントを渡すことを禁止する手段を備えることを特徴とする請求の範囲第８項に記載のシステム。
【請求項１０】
コンピュータ内でイベントが発生したことを検出し、イベントを発生し、該イベントの記述を発生するイベント生成部と、
前記コンピュータ内でイベントが発生した場合に通知を必要とするイベントコンシューマと、ここで該イベントコンシューマは第１及び第２のクラスのコンシューマを含み、
前記イベントコンシューマに通知されるべき特定のイベントセットを格納する格納手段と、
前記イベント生成部からイベントを受信し、受信したイベントと格納された前記イベントセットとを比較するイベント管理制御手段と、
前記イベント管理制御手段に応答して、イベントがイベントコンシューマに渡されるべきものかを決定する分配手段とを備え、
前記イベント管理制御手段は、
格納された前記イベントセットに基づいて、第１のクラスの適切なイベントコンシューマに前記イベントを送信する第１の手段と、
前記分配手段に応答して、第２のクラスの適切なイベントコンシューマに前記イベントを送信する第２の手段と
を備えることを特徴とするコンピュータシステム。
【請求項１１】
前記分配手段はコンピュータ内で発生し得る各イベントの種類毎に分配モジュールを備えることを特徴とする請求の範囲第１０項に記載のシステム。
【請求項１２】
前記第１のクラスのコンシューマは、互いに関連性を有するシーケンシャルコンシューマを含み、ここで、該シーケンシャルコンシューマは、自身がイベントの処理を行なっている間はそのイベントが他のコンシューマには通知されないことを要求し、他のコンシューマに関連するイベントを受信したときには影響を及ぼす能力を有し、
前記第２のクラスのコンシューマは、互いのコンシューマ間について関連性を有しないブロードキャストコンシューマを含み、ここで、該ブロードキャストコンシューマは、他のコンシューマに対して独立に、且つ、イベントを確認した順とは無関係に動作することを特徴とする請求の範囲第１０項に記載のシステム。
【請求項１３】
前記分配手段は、
ブロードキャストコンシューマがイベントを受信するまでは後続のイベントに基づいてそのイベントをモディファイする権利を前記分配手段が維持することを示す信号と共にブロードキャストコンシューマにイベントが送信されるべきかを判断する手段と、
前記判断する手段による肯定判断に応じて、前記分配手段がイベントをモディファイする権利を維持することを示す信号と共にそのイベントを適切なブロードキャストコンシューマに対して分配するべく前記イベント管理制御手段を動作させる手段とを備えることを特徴とする請求の範囲第１２項に記載のシステム。
【請求項１４】
前記格納手段は、
前記ブロードキャストコンシューマに関連付けられたイベントのサブスクリプションを格納するサブスクリプションマトリクスと、
前記シーケンシャルコンシューマが関連付けられたイベントのエントリを格納するシーケンシャルコンシューマデータベースとを備えることを特徴とする請求の範囲第１２項に記載のシステム。

【請求項１５】
前記格納手段は、
各ブロードキャストコンシューマに対応したイベントキューを備え、該イベントキューは、前記イベント管理制御手段から分配されたイベントを受信し、対応するブロードキャストコンシューマによってイベントが消費されるまで該分配されたイベントを格納することを特徴とする請求の範囲第１２項に記載のシステム。
【請求項１６】
前記制御手段は、前記シーケンシャルコンシューマデータベースにおけるエントリに従って、イベントを連続してシーケンシャルコンシューマに渡す手段を備えることを特徴とする請求の範囲第１２項に記載のシステム。
【請求項１７】
前記制御手段は、シーケンシャルコンシューマからの、イベントをハンドルしたことを示すメッセージの受信に応じて、イベントを渡すことを禁止する手段を備えることを特徴とする請求の範囲第１６項に記載のシステム。
【請求項１８】
コンピュータ内で発生したイベントを分配する方法であって、
イベントがコンピュータ内のイベント生成部によって検出されたことを判断するステップと、
イベントコンシューマに通知すべき特定のイベントセットを格納手段に格納するステップと、
イベント生成部からのイベントをイベント制御手段において受信するステップと、
受信した前記イベントと格納された前記イベントセットとを比較するステップと、
前記イベント制御手段からのイベントを分配手段において受信するステップと、
前記イベント制御手段に、適切なイベントコンシューマに適切なイベントを分配させるステップと、
前記イベント制御手段を介して、適切なイベントコンシューマに適切なイベントを分配するステップと
を備えることを特徴とする方法。
【請求項１９】
前記イベントコンシューマは、ブロードキャストコンシューマ及びシーケンシャルコンシューマを含む複数のコンシューマを備え、前記ブロードキャストコンシューマは、他のコンシューマに対して独立に、且つ、イベントを確認した順とは無関係に動作し、前記シーケンシャルコンシューマは、自身がイベントの処理を行なっている間はそのイベントが他のコンシューマには通知されないことを要求し、他のコンシューマに関連するイベントを受信したときには影響を及ぼす能力を有することを特徴とする請求の範囲第１８項に記載の方法。
【請求項２０】
前記分配するステップは、
ブロードキャストコンシューマがイベントを受信するまでは後続するイベントに基づいてそのイベントをモディファイする権利と共にそのイベントをブロードキャストコンシューマに送信するべきかを判断するステップと、
前記判断するステップによる肯定判断に応じて、モディファイする権利と共にそのイベントを適切なブロードキャストコンシューマに分配するステップとを備えることを特徴とする請求の範囲第１９項に記載の方法。
【請求項２１】
前記格納するステップは、
ブロードキャストコンシューマが関連付けられているイベントのサブスクリプションをサブスクリプションマトリクスに格納するステップと、
シーケンシャルコンシューマが関連付けられたイベントのエントリをシーケンシャルコンシューマデータベースに格納するステップとを備えることを特徴とする請求の範囲第１９

項に記載の方法。
【請求項２２】
ブロードキャストコンシューマに対応するイベントキュー内の分配されたイベントを受信するステップと、
対応するブロードキャストコンシューマによって前記イベントが消費されるまでは、前記イベントキューに分配されたイベントを格納するステップとをさらに備えることを特徴とする請求の範囲第１９項に記載の方法。
【請求項２３】
前記分配するステップは、シーケンシャルコンシューマからの、イベントの処理が完了したことを示す継続メッセージの受信に応じて、前記シーケンシャルコンシューマに連続してイベントを渡すステップを備えることを特徴とする請求の範囲第１９項に記載の方法。
【請求項２４】
前記分配するステップは、シーケンシャルコンシューマからの、イベントをハンドルしたことを示すメッセージの受信に応じて、イベントを渡すことを禁止するステップをさらに備えることを特徴とする請求の範囲第２３項に記載の方法。
【発明の詳細な説明】
発明の背景
本発明は、コンピュータ内で発生したイベントに関する情報を分配する装置及び方法に関する。特に、コンピュータ内の適切なエンティティ（実体データや実体プログラム等）へそれらイベントを分配すべく管理するイベントマネージャに関する。
本説明において、イベントはコンピュータにおけるあらゆる発生を総称するものであり、そのコンピュータもしくは接続されたコンピュータ上で実行されるプログラムは、そのイベントを通知される必要がある可能性がある。イベントは、例えば以下のような処理の発生を含むものである。すなわち、キーストローク、マウスクリック、ディスクの挿入及び排出、ネットワークへの接続及び切断、コンピュータのスリープモードシャットダウンへの移行、ウィンドウの出現（例えば、ウインドウの内容が再表示される必要が生じた場合）、新しいファイルの生成、ディレクトリの名称変更、ファイルの内容変更、ボリュームの空きスペースの変化等である。
割り込みやエラー状態も、不定形なイベントの例として扱われる。特に、割り込みはプログラムによって処理されることが必要である。そこで、イベントマネージャは不適切な解決となる。割り込みを扱うコードは割り込みの割り込み処理に基づいてイベントを発生することができる。例えば、コンピュータは、ユーザがフロッピーディスクを挿入したときに、割り込みを発生することができる。割り込み自身は、イベントマネージャによって伝播されることはまずないが、割り込みハンドラが“ディスク挿入”イベントを発生することは道理にかなっている。エラー状態は類似している。大部分の時間において、システム上のコンピュータプログラムの一つが、エラーを扱うことは必要なことであり、それゆえ、より直接的なポイントツーポイント（point to point）機構が適切である。しかしながら、その性質上より忠告的であるエラーが存在し、これらのエラーはイベントを介して送られることが好ましい。例えば、幾つかのポータブルコンピュータは、バッテリーが低下した場合に電力消費を低減するための種々の処理が実行される。全てのソフトウエアプログラムにその状態を知らせるために、“バッテリー低下”と呼ばれるイベントを発行し、電力消費を低減可能な全てのソフトウエアに当該イベントによる処理を実行させることは好ましいことである。
最近のあらゆる周知のオペレーティングシステムは、コンピュータ内で発生したイベントを管理するための幾つかの種類のメカニズムを有する。しかしながら、これらのメカニズムは、イベントを管理するのにポイントツーポイント方式を用いている。すなわち、イベントを生成あるいは検出するエンティティが該イベントを使用するエンティティにイベントを分配する。これを達成するために、イベントを生成あるいは検出する全てのエンティティが、コンピュータ内においてイベントが発生した場合にどのエンティティにそのイベントを通知しなければならないかを把握していなければならない。この構成は、非常に負

荷が大きく、非効率的である。その上、イベントを生成或いは検出する各エンティティが、それが生成或いは検出可能な全てのイベントに関する情報を持たなければならないとともに、これらのイベントに関連する全てのエンティティに関する情報を持たなければならないので、これは資源集中的（resource intensive）である。この情報は、広範であり、かつ定常的に変化するものである。そして、このため、改造を困難なものとしている。
更に、ポイントツーポイント方式は柔軟性に欠ける。ポイントツーポイントにおいては、もし新たなイベントのコンシューマが存在した場合、その新たなコンシューマを知っている新たなバージョンのプロデューサがリリースされなければならない。或いは、新たな種類のイベントが必要となった場合、その新たな種類のイベントをどのように分配するかを知っている新たなバージョンのイベントマネージャがリリースされなければならない。
従って、オペレーティングシステムにおける全ての種類のイベントを効率的に扱えて、特定の種類のイベントに関する情報を、そのような情報を必要としているプログラムに分配するための装置を提供することが望まれている。このために、システムパフォーマンスを向上すること、そのような情報を分配するために必要とされる資源を低減することもまた望まれている。これらの目的を達成するために、各イベントプロデューサが全てのイベントコンシューマを知ることを不要とし、イベントプロデューサとイベントコンシューマとの間のコミュニケーションを容易とすることが可能な、イベントを管理するための装置を提供することが望まれている。

発明の要約

本発明によれば、上記目的及びその他の課題は、イベント管理の集中化によって達成される。特に、コンピュータ内のイベントの分配を扱うイベントマネージャを提供することによって達成される。
一つの実施形態によれば、コンピュータにおいてイベントが発生したことを検出するとともにイベントを生成するイベントプロデューサと、コンピュータにおいてイベントが発生したときにその通知を必要とするイベントコンシューマとを少なくとも１つずつ含むコンピュータに、イベントに関する情報を分配するためのシステムが導入される。このシステムは、イベントコンシューマが通知を受けるべき特定のイベントセットを格納する格納手段と、イベントプロデューサからイベントを受信し、格納されたイベントセットと比較し、適切なイベントコンシューマに適切なイベントを分配するイベント管理制御手段と、制御手段からのイベントを受信し、適切なイベントコンシューマに適切なイベントを分配するために制御手段を管理するディストリビュータとを備える。
また、別の実施形態によれば、コンピュータ内で発生したイベントを分配するためのシステムが提供される。このシステムは、イベントがコンピュータ内で発生したことを検出してイベントを生成するイベントプロデューサと、コンピュータ内においてイベントが発生したときにその旨の通知を必要とするイベントコンシューマとを備える。このイベントコンシューマは第１及び第２のコンシューマのクラスを備えている。本システムは、更に、イベントコンシューマが通知されるべき特定のイベントセットを格納する格納手段と、イベントプロデューサからイベントを受信し、格納されたイベントセットと比較し、適切なイベントコンシューマに適切なイベントを分配するイベント管理制御手段と、イベント制御手段に応答して、イベントコンシューマへイベントを渡すべきか否かを決定する分配手段とを備える。また、イベント管理制御手段は、格納されたイベントセットに基づいて第１のタイプの適切なイベントコンシューマにイベントを送る第１の手段と、上記分配手段に応答する第２のタイプの適切なイベントコンシューマに該イベントを送る第２の手段とを備える。
また、他の実施形態によれば、コンピュータにおいて発生するイベントを分配するための方法が提供される。この方法は、コンピュータ内のイベントプロデューサによってイベントが検出されたことを決定するステップと、格納手段にイベントコンシューマが通知されるべき特定のイベントセットを格納するステップと、イベント制御手段において上記イベントプロデューサからのイベントを受信するステップと、受信したイベントを格納されたイベントセットと比較するステップとを備える。本方法は、更に、分配手段において制御

手段からのイベントを受信するステップと、該制御手段に適切なイベントを適切なイベントコンシューマに分配させるステップと、制御手段を介して適切なイベントを適切なイベントコンシューマに分配するステップとを備える。
本発明の更に他の目的、特徴及び付随する利点は、本発明に基づいて構成された実施形態に関して添付の図面を参照してなされる以下の詳細な説明を読むことにより当業者に明らかとなるであろう。
【図面の簡単な説明】
以下、方法及び装置に関する一例としての好適な実施形態を参照し、また添付の図面を参照して本発明のより詳細な説明を行う。添付図面は以下の通りである。
図１は、本発明を実行可能な典型的なコンピュータのブロック図である。
図２は、本発明の一実施形態に関るイベントマネージャの構成を示すブロック図である。
図３は、本発明に係るイベントキューのデータ構造の典型的な実施形態を示す図である。
図４は、本発明に係るサブスクリプションマトリクスの典型的な実施形態を示す図である。
図５ａは、本発明に係る加入データ構造の典型的な実施形態を示す図である。
図５ｂは、本発明に係るシーケンシャルコンシューマのエントリ構造の典型的な実施形態を示す図である。
図６は、本発明に係るイベント種別ヘッダのデータ構造の典型的な実施形態を示す図である。
図７は、本発明に係るイベントネームのデータ構造の典型的な実施形態を示す図である。
図８は、本発明に係るイベントデータ構造の典型的な実施形態を示す図である。
図９Ａ、９Ｂ、９Ｃ、９Ｄは、本発明の一実施形態に係る、イベントプロデューサの観点及びイベントディストリビュータの観点からのイベントハンドリング処理を示すフローチャートである。
図１０は、本発明の一実施形態に係るシーケンシャルコンシューマの観点からのイベントハンドリング処理を示すフローチャートである。
図１１は、本発明の一実施形態に係るブロードキャストコンシューマの観点からのイベントハンドリング処理を示すフローチャートである。
発明の実施の形態
総括的に言って、本発明は、コンピュータ内で発生したイベントに関るコンピュータ内の異なるエンティティ間の効率的なコミュニケーションの必要性を認識するものである。特に、コミュニケーションにおいては、コンピュータ内のエンティティに他のエンティティで生成されたイベントについて通知することを要求される。これらのコミュニケーションを扱うために要求される方法は、関与するエンティティが他のエンティティを識別できないという理由によって複雑なものとなっている。また、この方法は、イベントを知る必要のあるエンティティの識別情報や、発生し得るイベントのリストが定常的に変化してしまうという理由で、更に複雑化されている。
本発明のイベントマネージャの一つの到達目標は、これらの種類のコミュニケーションの大部分をサポートする共通のサービスを提供することである。コミュニケーションされるべき情報はイベントとして参照される。一つの実施形態によれば、イベントは３つの部分によって記述される。イベントの種類を示すイベント識別子、イベントが発生するエンティティを特定するイベントサブジェクト、及び当該イベントがどのように発生したかを記述するインベント情報である。コンピュータ内の、その情報のソースとなるエンティティは、イベントのプロデューサとして参照される。ここで、イベントプロデューサは、イベントを生成する、もしくはコンピュータハードウエアがイベントを発生したことを検出する、コンピュータ上のあらゆるソフトウエアである。そして、イベントプロデューサは、生成もしくは検出した各イベントの記述を生成する。一方、そのような情報を受信することを必要とするコンピュータ内のエンティティはイベントコンシューマとして参照される。特に、イベントコンシューマは、イベントが発生したときに通知されることを必要とするとともに、そのイベントの記述を通知されることを必要とするあらゆるプログラムであ

る。イベントのプロデューサとコンシューマの間の接続を調整する仲介サービスは、イベントのディストリビュータとして参照される。
本発明の一実施形態によれば、２つのクラスのイベントコンシューマがある。これらは、ブロードキャストコンシューマ及びシーケンシャルコンシューマであり、イベントに関る他のコンシューマとの関係において相違する。ブロードキャストコンシューマは他のコンシューマとは無関係である。それらは、それら自身が最終的にはイベントを通知される限りにおいて、他のコンシューマが存在するか否か、どのような順序でコンシューマがイベントの通知を受けたかについて知る必要がない。一方、シーケンシャルコンシューマは、他のコンシューマとの間に非常に明確な関係を有する。これらは、これら自身がイベントを処理している間、そのイベントに関して他のいかなるコンシューマも通知を受けないことを要求する。また、それらがイベントを受信した順序によって影響を与える機能を要求する。加えて、多くのシーケンシャルコンシューマはイベント自身を改造する機能や、イベントが他のコンシューマによって受信されるの阻止する機能をも要求する。コンシューマは、失敗すること無しにイベントに反応しなければならないかどうかに基づいて、シーケンシャルかブロードキャストとして定義される。特に、シーケンシャルコンシューマは、それが関連するイベントの種類に反応しなければならない。そこで、ディストリビュータは、シーケンシャルコンシューマからそれらのイベントを引き止めるべきではない。例えば、通信プログラムは、接続のドロップを通知するイベントのシーケンシャルコンシューマとなる。通信プログラムはそのようなイベントに応答する必要があるからである。
ブロードキャストコンシューマは、典型的には、それが関与する特定のイベントセットを有し、可能な限り簡単にこのセットから次のイベントを通知されることを要求する。多くのブロードキャストコンシューマは、限定されたサブジェクトのセットのみに対して発生するイベントに関与する。そこで、本発明の一実施形態は、イベントを、それらが「何に」よって発生されたかに基づいてフィルタリングする方法を提供する。加えて、ブロードキャストコンシューマが関与するイベントセットは、関与の即時性がそうであるように、常時変化する。こうして、本発明の一実施形態は、コンシューマに、当該コンシューマに配布されるイベントセットを変更することを可能とする方法を提供する。最後に、ブロードキャストコンシューマがイベントを修正するためにイベントマネージャ制御ユニットにポーリングを行うかどうかに関して、或いはそれらが同期して通知を受けるべきか否かに関して明瞭なパターンは存在しない。それゆえ、一つの実施形態によれば、これらの両方のオプションがサポートされる。
図１は本発明に基づくソフトウエアが実行される典型的なコンピュータを示すブロック図である。コンピュータ１０は、プロセッサ（ＣＰＵ）１２、外部記憶装置１４、ディスク格納装置１６、入出力（Ｉ／Ｏ）コントローラ１８、入力デバイス２０、出力デバイス２２、メモリ２４、上記エレメントを接続する共用システムバス２６を含む。図１には、読み易さの目的のために、ただ一つの入力デバイス及び出力デバイスが示されている。しかしながら、コンピュータ１０が、１つより多くのそのようなデバイスを含み得ることは好ましいことである。プロセッサ１２、外部記憶装置１４、ディスク格納装置１６及びメモリ２４は、バス２６及びＩ／Ｏコントローラ１８を介して入力、出力デバイス２０、２２に接続される。加えて、コンピュータ１０はネットワーク２７と通信可能に適用され得る。
コンピュータ１０のメモリ２４内に格納されるのは、プロセッサ１２によって実行され得る複数のソフトウエアである。これらのソフトウエアの一つはオペレーティングシステム２８である。一つの実施形態において、オペレーティングシステム２８は、複数のアドレス空間を維持することのできるマイクロカーネルオペレーティングシステムである。イベントマネージャ３０はオペレーティングシステム内に駐在する。典型的な実施形態において、本発明のコンピュータシステム１０は、カリフォルニア州クペルチノのアップルコンピュータ社によるアップルマッキントッシュ（商標）コンピュータシステムであり、これはマイクロプロセッサと、イベントマネージャ３０を含むマイクロカーネルオペレーティングシステム２８を格納するメモリを有する。コンピュータシステムのコンポーネントは

、本開示に基づいて当業者がなしうる範囲において変化され得るものである。なお、本発明はマッキントッシュ（商標）環境において説明されるが、本発明をＤＯＳ、Ｕｎｉｘ、或いは他のコンピュータ環境において実現することは、本発明の範疇であり、また当業者の技術の範疇である。

本発明は、オペレーティングシステムにおいて発生するイベントを管理するイベントマネージャのアーキテクチャに関する。イベントマネージャのクライアントはイベントコンシューマとイベントプロデューサを含む。すなわち、アプリケーションプログラムや、ファイルマネージャのようなオペレーティングシステムの種々の部分が含まれる。

一実施形態によれば、本発明はマイクロカーネルアーキテクチャのオペレーティングシステムと協動する。ここで、カーネルはセマフォ同期メカニズムとメッセージングシステムを提供する。メッセージングシステムは、メッセージオブジェクト及び、一つもしくはそれ以上のポートオブジェクトのセットを生成、維持しする。メッセージングシステムは複数の特徴を有する。それは、各オブジェクトと協動されるべきクリエータ定義値（creator-defined value）を許可する。メッセージングシステムは複数のオブジェクトが同じポートにマップされること、メッセージがポートから受信されるかまたはそのポートに適切なタイプのメッセージが送られたときにファンクションがコールされることを許可する。更に、メッセージングシステムはメッセージの受信部に、メッセージが最初に送られる先のオブジェクトを決定し、そのオブジェクトのためのクリエータ定義値を導き出すことを可能とする。そのようなマイクロカーネルアーキテクチャの一例は、ＮｕＫＥＲＮＥＬ（商標）によって提供される。これはアップルマッキントッシュ（商標）コンピュータに用いられている。ＮｕＫＥＲＮＥＬ（商標）システムは、出願継続中の米国特許出願０８／１２８，７０６（１９９３年9月３０日出願）の「コンピュータにおける仮想メモリの分散バッキングストアのためのシステム」、米国特許出願０８／２２０，０４３（１９９４年３月３０日出願）の「オブジェクト指向メッセージパッシングシステム及び方法」、そして本願の出願に前後して、Thomas E.Saulpaugh及びSteven J.Szymanskiによって出願される「オブジェクト指向メッセージフィルタリングのシステム及び方法」と題した特許出願において説明されている。これら３つの特許出願は本願に参照として組み込まれるものである.

メッセージングシステムによれば，サービスがサーバによって提供され、そのサービスを用いることを要求するソフトウエアはそのサーバのクライアントと呼ばれる。メッセージはクライアントによって目標に送られ、サーバによって受信される。メッセージはクライアントの代わりに機能を遂行するためにサーバに向けられる。メッセージオブジェクトは、メッセージングシステムクライアントへの種々の資源を表すアブストラクトエンティティである。これらのオブジェクトは、たとえば，サーバによって管理されるデバイス、ファイルもしくはウインドウを表してもよい。クライアントはオブジェクトにメッセージを送り、これらオブジェクトはObjectIDにラベル付けされた識別子によって識別される。

メッセージポートはサービスを表すアブストラクトエンティティである。これらのポートは、たとえば，デバイスドライバ，ファイルシステム、ウインドウマネージャを表すことができる。サーバはポートからのメッセージを受信し、これらのポートはラベルPortIDによって識別される。オブジェクトはポートに割り当てられる。オブジェクトに送られたメッセージはそのオブジェクトのポートにおいて受信される。ポート及びオブジェクトはサーバの代わりにメッセージングシステムによって生成される。オブジェクトの生成は、オブジェクトに送られたメッセージがリトリーブされるポートを指定することを要求する。

一実施形態によれば、メッセージングシステムはイベントマネージャによって使用される。このイベントマネージャにおいて、イベントディストリビュータは協動するポート及びオブジェクトを有し、また、シーケンシャルコンシューマは協動するポートおよびオブジェクトを有する。そして、そのイベントマネージャのポートおよび、各ブロードキャストコンシューマについてこのポートに割り当てられたオブジェクトを維持する。イベントはイベントプロデューサによってメッセージとしてディストリビュータの協動オブジェクトに受け渡される。シーケンシャルコンシューマは関連するポートからメッセージを受信し

、ブロードキャストコンシューマはイベントを要求するために、そのフロードキャストコンシューマのために維持されるオブジェクトを介して、メッセージをイベントマネージャに送る。加えて、一つの実施形態によれば、イベントマネージャは、シーケンシャルコンシューマを実施するためにメッセージングシステムにおいて提供されるメッセージフィルタメカニズムを使用する。各シーケンシャルコンシューマのために、メッセージフィルタはディストリビュータのObjectID上に生成される。フィルタを介して受信されたコードは、整合性をチェックし（何が提供され、何によって提供されたかに関する）、シーケンシャルコンシューマに適切なメッセージを送る。そのようなメッセージフィルタリングメカニズムの一例は、上述の、Thomas E.Saulpaugh及びSteven J.Szymanskiによる特許出願に記載されている。

図２は図１に示されたイベントマネージャ３０のアーキテクチャを示すブロック図である。イベントマネージャ３０（図２において破線で示される）は複数のイベントディストリビュータ３４０、複数のイベントコンシューマ及び複数のイベントプロデューサ３００と機能上において接続され、これらと通信を行う。イベントディストリビュータの数は、システム内において可能な、異なるイベントの種類の数に対応する。

一つの実施形態によれば、複数のイベントコンシューマはブロードキャストコンシューマ３１０およびシーケンシャルコンシューマ３６０を含むことができる。与えられたシステムが一つ或いは複数のブロードキャストコンシューマ、もしくは一つ或いは複数のシーケンシャルコンシューマのいずれかを含むこと、もしくはその双方を含むことは好ましいことである。

本発明によれば、イベントは“種別”によってグループ分けされ、たとえば、全てのキーストロークは一つの種類のイベントであり、全てのマウスクリックは別の種類のイベントであり、全ての新規ファイル作成は更に他の種類のイベントである。また、イベントを上記例とは異なるようにグループ分けすることもできる。たとえば、特定キーの全ての押下は一つの種類のイベントである。これは、少なくとも部分的に好ましい実施方法ではない。数種類のイベントがすぐに管理不能となるからである。しかしながら、イベントをどのようにグループ分けするかの選定は、本開示により当業者がなし得る範囲で可変である。

イベントマネージャ３０は、イベントマネージャ制御ユニット３０５及びデータ構造を有する。データ構造はサブスクリプションマトリクス３３０、シーケンシャルコンシューマデータベース３５０、ブロードキャストコンピュータ３１０と１対１対応で提供される複数のイベントキュー３２０、イベントディストリビュータ３４０と１対１対応で提供される複数のイベント種別ヘッダ３３１を含む。イベントマネージャ制御ユニット３０５は、イベントマネージャデータ構造を管理する少なくとも一つのソフトウエアルーチンからなる。

イベントプロデューサ３００は、イベントを生成する、或いは他のエンティティがイベントを生成したことを検出する、コンピュータ上のあらゆるソフトウエアを表す。イベントプロデューサは、自身が生成もしくは検出した各イベントの記述を生成する。各種のイベントは、複数のイベントプロデューサを有することができ、与えられたイベントプロデューサは複数種類のイベントを生成もしくは検出することができる。イベントコンシューマは、ブロードキャストコンシューマ３１０とシーケンシャルコンシューマ３６０を含み、イベントが発生したときを通知されることを必要とするとともに、そのイベントの記述の通知を必要とするあらゆるプログラムを表すものである。各種のイベントは、複数のイベントコンシューマを有することができ、与えられたイベントコンシューマは、複数種類のイベントを通知されることを必要とする。

イベントコンシューマがイベントプロデューサとなること、或いはイベントプロデューサがイベントコンシューマとなることも有り得ることである。特に、ある一つのソフトウエアがある一つのイベントを処理し、その応答として別の種類のイベントを生成するというケースが多く存在する。たとえば、ファイルマネージャは“ディスク挿入”イベントを処理し、その挿入されたメディアにそのボリュームをマウントし、“新規容量”イベントを結果として生成する。与えられたプログラムが異なる種類のイベントに異なる加入を行う

ことも可能である。特に、プログラムは幾つかの種類のイベントに対してはシーケンシャルコンシューマとして加入し、他の種類のイベントに対してブロードキャストコンシューマとして加入することができる。

イベントキュー３２０はイベントマネージャ制御ユニット３０５によって保持されるイベントのリストであり、各ブロードキャストコンシューマ３１０に対し、ブロードキャストコンシューマが関与するイベントについての情報を保持する。一つの実施形態によれば、イベントキュー３２０はファーストインファーストアウトリストとして構成される。また、たとえば、ディスク挿入イベントはマウスクリックよりも常に高い優先度を有するとか、ディスクイベントプロデューサはマウスイベントプロデューサよりも高い優先度を有するというような、イベント優先度やプロデューサ優先度によって並べられるリストのような他の適したストレージ構造も使用可能である。図２において、イベントキュー＃１はブロードキャストコンシューマ＃１に関与するイベントのリストであり、イベントキュー＃２はブロードキャストコンシューマ＃２に関与するイベントのリストである。他も同様である。図２には４つのブロードキャストコンシューマ／キューのペアが示されているが、所定の時刻において何組のブロードキャストコンシューマ／キューのペアが定義されていてもよい。

一つの実施形態によれば、イベントキュー３２０は図３に示されるようなフォーマットで格納される。すなわち、先頭サブスクリプション、最終サブスクリプション、最大イベント、イベント配列である。先頭及び最終サブスクリプションのフィールドは、対応するブロードキャストコンシューマ３１０の先頭及び最終サブスクリプションへのポインタである。イベントサブスクリプションは、特定のブロードキャストコンシューマが通知を必要とする特定のイベントセットの記述である。最大イベントのフィールドは、そのコンシューマに対するキューにおいて維持されるイベントの最大数である。一つの実施形態によれば、イベント配列のサイズは、クッションを提供するために、示された最大数よりも若干大きくしてある。イベント配列は、エレメントの配列の形態で、発生したイベントのリストを格納する。ここで各エレメントは、対応するブロードキャストコンシューマが加入したイベントとなる。シングル及びダブルのリンクされたリストやダイナミックスタックのような他の適した格納構造を用いることができるが、一つの実施形態において、キューはサーキュラバッファに格納される。

サブスクリプションマトリクス３３０は全ての存在するイベントサブスクリプションについて情報を保持する構成である。特に、サブスクリプションマトリクス３３０は、ブロードキャスターコンシューマが関与するイベントのサブスクリプションのトラックを保つのに用いられる。

一方、複数のシーケンシャルコンシューマ３６０が定義され得る。図２においては、３つのシーケンシャルコンシューマ＃１、＃２、＃３が示されているが、いくつのシーケンシャルコンシューマが定義されてもよい。シーケンシャルコンシューマデータベース３５０は複数のシーケンシャルコンシューマエントリからなる。これらシーケンシャルコンシューマエントリは、各シーケンシャルコンシューマ３６０が関与するイベントをならべたものである。

一つの実施形態において、サブスクリプションマトリクス３３０は図４に示されるような構成を有する。ここで、各サブスクリプションに対して格納される一つのサブスクリプションストラクチャ３３００が存在する。イベントキューやイベント種構造に関するサブスクリプションのsparse arrayやダイナミックリストのような他の適切な形態が使用されてもよい。サブスクリプションマトリクスに格納されたサブスクリプションストラクチャ３３００はイベントキュー３２０及びイベント種別ヘッダ３３１に接続される。特に、システムに知られた各イベントの種類に対して一つのイベント種別ヘッダ３３１が存在し、こうしてイベント種別ヘッダの数はイベントディストリビュータの数と等しくなる。イベント種別ヘッダ３３１は、イベントマネージャ制御ユニット３０５が、どのディストリビュータが現在処理されているその種類のイベントを扱うかを決定するのに用いる情報を格納する。レジスタディストリビュータ（後述）コールはイベント種別ヘッダを生成する。

一つの実施形態によれば、サブスクリプション３３００はサブスクリプションマトリクス３３０に図５ａに示す形態で格納され、以下のフィールドを有する。すなわち、イベントキューポインタ、イベント種別ヘッダポインタ、次サブスクリプション及び同一イベント、前サブスクリプション及び同一イベント、次サブスクリプション及び同一キュー、前サブスクリプション及び同一キュー、イベントサブジェクトである。イベントキューポインタは、サブスクリプションが属するイベントキューへのポインタである。イベント種別ヘッダポインタは、サブスクリプションのサブジェクトであるイベントの種類に対するイベント種別ヘッダへのポインタである。イベントサブジェクトはそのイベントが発生したストラクチャを識別する。残りのフィールドは他のサブスクリプションへのポインタである。

シーケンシャルコンシューマエントリの典型的なフォーマットは図５ｂに示されている。一つの実施形態によれば、エントリは以下のフィールドを含む。すなわち、イベント識別子、イベントサブジェクト、シーケンシャルコンシューマオブジェクトＩＤである。イベント識別子は、シーケンシャルコンシューマが関与するイベントの種類を識別する。イベントサブジェクトはそのシーケンシャルコンシューマが関与するイベントのサブジェクトを識別する。シーケンシャルコンシューマオブジェクトＩＤは、シーケンシャルコンシューマデータベースにおいてエントリを生成したシーケンシャルコンシューマのObjectIDを識別する。一つの実施形態によれば、シーケンシャルコンシューマデータベース３５０は、サブスクリプションマトリクス３３０の一部であり、各シーケンシャルコンシューマに対するサブスクリプションストラクチャへのポインタがある。図２に示されるもう一つの実施形態によれば、シーケンシャルコンシューマデータベース３５０は、上述のようにサブスクリプションマトリクス３３０とは別々に提供される。イベントマネージャ制御ユニット３０５はデータ構造と検出されたイベントの比較を実行し、当該イベントがそのシーケンシャルコンシューマへ送られるべきか否かを決定する。

一つの実施形態によれば、メッセージフィルタリングメカニズムが、イベントの発生時にそのイベントを適切なシーケンシャルコンシューマに提供するために要求される情報を保持するのに用いられる。そのメカニズムは、上述のSaulpaughとSzymanskiによる特許出願に記載されている。

一つの実施形態によれば、イベント種別ヘッダ３３１は図６に示されるフォーマットによって格納される。イベント種別ヘッダ３３１は以下のフィールドを有する。すなわち、次ヘッダ、ディストリビュータオブジェクトＩＤ、イベント識別子、先頭サブスクリプション、最終サブスクリプションである。次ヘッダフィールドは、イベント種別ヘッダの単一にリンクされたリストにおける次のヘッダへのポインタである。ディストリビュータオブジェクトＩＤは対応するイベントディストリビュータに対するディストリビュータのObjectIDである。イベント識別子は関連するディストリビュータが扱うイベントの種類を識別する。先頭及び最終サブスクリプションフィールドは、当該イベントの種類のために定義された先頭及び最終のサブスクリプションへのポインタである。

イベントディストリビュータ３４０は、適切なイベントを適切なブロードキャストコンシューマに分配するためにイベントサブスクリプションを解釈するプログラムである。一つの実施形態によれば、各種のイベントに対してイベントディストリビュータが存在する。特に、あらゆるイベントプロデューサによって検出された全てのキーストロークを扱うイベントディストリビュータや、全てのマウスクリックを扱うイベントディストリビュータを扱う別のイベントディストリビュータ等が存在する。

一つの実施形態によれば、どのブロードキャストコンシューマにイベントが通知されるかはディストリビュータによって決定される。ディストリビュータがどのコンシューマに実際に通知するかに関して最終的な報告を行う間、イベントマネージャ制御ユニット３０５は、サブスクリプションマトリクス３３０を用いて、イベントを要求するコンシューマのトラックをキープする。与えられたディストリビュータは常に全ての関連するコンシューマに結果を与えるかもしれないし、常に一つを選択するかもしれないし、各コンシューマをみていくつかを選択し、他を排除するかもしれない。これは非常に柔軟なアーキテクチ

ャを可能とする。例えば、全てのアプリケーションがマウスクリックに関与する場合（それゆえ全てのアプリケーションがその種のイベントに加入しており）、唯一の（最前の、或いはクリックされたウインドウに対するアプリケーションのいずれか）ものが実際にそのイベントを受け取る。マウスクリックイベントディストリビュータはどの加入者が最前のアプリケーションであるかを決定し、そのイベントをそのアプリケーションのみに送る。他の例としては、いつ新規ファイルが生成されたかを知る必要のある数本のコードがあるような場合、全てのコードが通知されるべきである。それゆえ、新規ファイルイベントディストリビュータは常に全ての加入者にイベントを送る。

図２に示されるように、各ブロードキャストコンシューマ３１０は、イベントマネージャ制御ユニット３０５を介してそれぞれのイベントキュー３２０に機能上接続されており、通信可能となっている。各ブロードキャストコンシューマ３１０はまた、イベントマネージャ制御ユニット３０５を介してサブスクリプションマトリクスにも機能上接続されており、通信可能となっている。イベントキュー３２０の各々もイベントマネージャ制御ユニット３０５を介してサブスクリプションマトリクス３３０に機能上接続されており、通信可能となっている。シーケンシャルコンシューマデータベース３５０はイベントマネージャ制御ユニット３０５と機能上接続され通信可能であり、シーケンシャルコンシューマ３６０はイベントマネージャ制御ユニット３０５を介してシーケンシャルコンシューマデータベース３５０と機能上接続され、通信可能となっている。複数のイベント種別ヘッダ３３１は複数のイベントディストリビュータ３４０と機能上接続され、これら複数のイベントディストリビュータ３４０は複数のイベントプロデューサ３００と機能上接続されている。

各ブロードキャストコンシューマ３１０はイベントマネージャ制御ユニット３０５を用いてイベントキュー３２０を生成し、イベントが報告された時点からブロードキャストコンシューマがそのイベントを消費するまでの間、関与するイベントを保持する。各ブロードキャストコンシューマ３１０はイベントマネージャ制御ユニット３０５へ通知を必要とする全てのイベントを記述した加入イベントのセットを通信する。この加入イベントのセットは、常時変更が可能である。イベントマネージャ制御ユニット３０５は、その情報を、イベントディストリビュータ３４０によって使用されるように、サブスクリプションマトリクス内に格納する。

要約すると、本発明の一実施形態によれば、イベントプロデューサ３００はイベントを検出し、イベント記述を生成する。そして、イベントマネージャ制御ユニット３０５をコールすることで、これらの記述をイベントマネージャ３０に送る。イベントマネージャ制御ユニット３０５はそのイベント記述を、シーケンシャルコンシューマデータベース３５０内のエントリに基づいて、各シーケンシャルコンシューマ３６０に順番に送る。イベントマネージャ制御ユニット３０５は、その後、イベント記述を、そのイベントの種類に応答するイベントディストリビュータ３４０に送信する。イベントディストリビュータ３４０はイベントマネージャ管理部３０５をコールし、サブスクリプションマトリクス３３０内の情報に基づいて適切と判断されたブロードキャストコンシューマ３１０にイベント記述を送る。イベントマネージャ制御ユニット３０５は、最初にこれらの記述をイベントキュー３２０に格納することにより、適切なブロードキャストコンシューマ３１０にイベント記述を与える。ブロードキャストコンシューマ３１０は、レディ状態である場合、対応するイベントキュー３２０内に格納された次のイベント記述をリトリーブするために、イベントマネージャ制御ユニット３０５をコールする。

アプリケーションプログラマインターフェース

本発明によれば、イベントコンシューマ、イベントプロデューサ及びイベントディストリビュータは、イベントマネージャの製造者以外のサードパーティによって記述されてよい。それゆえ、アプリケーションプログラマインターフェース（ＡＰＩ）は、これらサードパーティがイベントマネージャとコミュニケーションできるように仕様を提供するべく定義されている。以下は、本発明に係るイベントマネージャとのコミュニケーションを可能とするＡＰＩの一実施形態である。

イベント構造
一つの実施形態によれば、イベントは３つのパートから成る。すなわち、どのイベントが発生したか（イベント識別子）、何がイベントを発生したか（イベントサブジェクト）、どのようにしてイベントが発生したか（イベント情報）である。処理可能なイベントセットが（オペレーティングシステムの他の部分の開発者によって、或いはアプリケーションプログラマによって）拡張可能であることは望ましいので、“どの”の部分はイベントネームとして参照されるユニークな識別子を定義する。一つの実施形態において、イベントネーム識別子は、ＯＳＴｙｐｅとして知られている４キャラクタコードとして実施される。本開示に基づいて当業者によって置き換えられ得る他の適切な識別子が用いられてもよい。また、一つの実施形態によれば、識別子のペアであってもよい。特に、本開示において記述された実施の形態は２つのＯＳＴｙｐｅを用いる。図７に示されたイベントネーム識別子の構造４５は典型的なものである。イベントサービス識別子は、例えば文書処理プログラムといった、イベントを定義したサービスの“サイン”もしくは名前として機能する。そして、イベント種別識別子は、例えば新規ファイル生成といったような、イベント自身を識別するものである。こうして、全てのイベントの名前の集合が、名前空間として参照されるものであるが、サインを制御することによって管理される。
イベント構造の“何が”のフィールドは、一実施形態においてイベントサブジェクトとして定義されるものであるが、これは定義することが難しい。イベントマネージャの全ての可能な使用が予期できず、それゆえ、“何が”として記述されるかもしれないハードウエアとソフトウエアエレメントの全ての形態も予期し得なくなるからである。幸運なことに、イベントのこの部分のためにサポートされる必要のある共通のオペレーションは等価判定のためのテストである。すなわち、ここで等価とは、ブロードキャストコンシューマによって加入のために提供された“何が”に対する等価性である。そして、イベントマネージャ制御ユニット３０５は加入者の構造を知る必要はなく、未解釈のバイト配列として定義され得る。本開示に基づいて当業者に明らかな範囲で、固定長、もしくは可変長フィールドのような、“何が”のフィールドの適切な形態が用いられ得る。
構造の“詳細”部分は、図８における情報フィールドであり、全体としてオープンエンドとなっており、各イベントに関してのみならず、イベントの各瞬間において潜在的に変化する。こうして、この構成の一つの実施形態はオープンエンドのバイト配列となる。
図８は、本発明の一実施形態に係るイベント構造のブロック図である。イベント４０は、イベントネーム、イベントサブジェクト、バイトカウント、イベント情報フィールドから構成される。イベントネームフィールドは“どの”を特定し、イベントサブジェクトフィールドは“何が”を特定し、バイトカウント及びイベント情報フィールドは“どのように”を特定する。イベント情報は実データであり、バイト数はそのデータ長を表す。
システムコール
イベントプロデューサは、イベントマネージャ制御ユニット３０５にイベントを委ねるためのシングルコールとともに供給される。特に、そのシングルコールはイベントメッセージをイベントマネージャ制御ユニット３０５に送信する。本実施形態によれば、イベントプロデューサは、そのイベントに対する応答を必要としているすべてのコンシューマがそのシングルコールの処理を行う機会があることの確認を受信する。特に、そのコールが返答されると、ディストリビュータは、すべてのシーケンシャルコンシューマがそのシングルコールを確認したことを認知し、すべてのブロードキャストコンシューマに対してそのイベントがキューされる（enqueue）。他の実施形態によれば、この確認は必要とされず省略することができる。本実施形態によれば、このシングルコールは、二つのコール（ファインドディストリビュータ（FindDistributer）とプロデュースイベント（ProduceEvent））に分割し、この分割によって、その種類のイベントをどのように分配するかを知るもの識別を知ることを効率的に行える。このＡＰＩをテーブル１に示す。
テーブル１とそれに続くテーブルはＣ言語で記述された構造定義を示している。ここでＣ言語の記述例を示しているが、本発明は、当業者によって他の適合したプログラミング言語によっても実行される。加えて、テーブル１に参照されるコードは本実施形態に伴う本

発明の一つの実行例である。本開示に基づいて、当業者によって他の実行方法も可能であることが理解されるであろう。
テーブル１において、OSStatusは、通常、シングルコールが完了したか否かを示すために用いられるリータン値として用いられる様式である。OptionBitsは、通常、シングルコールがどのように処理されるかにおける微小変化を特定するためにユーザに許可するコールの入力パラメータとして用いられる。

テーブル１

```
typedef ObjectID                EventDistributorID;
typedef OptionBits              PrducEvntOptions;
enum
    {
    kProducerEventsynchronously          =0x00000001
    // don't return until all Sequential Consumers have completed
    }
OSStatus FindDistributor (  Eventname           EvntName,
                            EventDistributorID  *inputID);
OSStatus PriducerEvnet (    EventInformationPtr event,
                            EventDistributorID  distributor
                            ProduceEventOptions Options);
```

コンシューマ
本発明の実施形態に従う上述の説明によれば、ブロードキャストコンシューマとシーケンシャルコンシューマは、それらの関係において他のイベントのコンシューマと異なる。ブロードキャストコンシューマは、自身がイベントの通知を受ける限りにおいて、他のコンシューマが存在するか否かを知る必要がないばかりか、どの順番でコンシューマがイベントを通知されるかを知る必要がない。シーケンシャルコンシューマは、自身がイベントを処理する間は、イベントについて通知される他のコンシューマが一つもないことを必要とし、一連のシーケンスの中でイベントを受信する場合の影響に対する能力を必要とする。加えて、多くのシーケンシャルコンシューマは、他のコンシューマによってイベントが受信されることを阻止したり、イベント自身を変更する能力を必要とする。
本実施形態によれば、ＡＰＩは“何が”に基づいてイベントをフィルタリングする方法を提供する。これは、イベントを通知されるサブジェクトの設定を制限するブロードキャストコンシューマの許可を誰が必要とするかに基づく方法である。加えて、ＡＰＩは、ブロードキャストコンシューマが、常時、関与するイベントの設定を変更することを可能とする方法を供給する。このように、ＡＰＩは、ブロードキャストコンシューマがイベントを収集するためにイベントマネージャ制御ユニット３０５をポーリングすることをサポートし、あるいは非同期通知機能を供給するメッセージングシステムの使用によってブロードキャストコンシューマが非同期に通知され得ることをサポートする。
本実施形態に従うインタフェースにおいて、コンシューマは、関連するイベントのリストを表現するコンシューマ構造を生成するためにイベントマネージャ制御ユニット３０５をコールする。これらのコンシューマ構造は、呼び出し側にとっては不明瞭である（即ち、その詳細はイベントマネージャ制御ユニット３０５だけが知らされるものである）。
本実施形態に従えば、このコンシューマ構造はイベントキュー３２０である。ＡＰＩは、ブロードキャストコンシューマがイベントキューを生成するイベントマネージャ制御ユニ

ット３０５をコールすることを許可する。特に、イベントキュー３２０は、CreateBroadcastConsumerのコールの結果としてイベントマネージャ制御ユニット３０５によって生成され、DisposeBroadcastConsumerのコールの結果として消去される。本実施形態のキューの詳細は図３に示される。イベントキューは、コンシューマ名（特定のコンシューマを識別するためにディストリビュータによって使用される）、コンシューマのためにイベントキューに格納することによってイベントマネージャ制御ユニット３０５がバッファする最大イベント数を含んでいる。コンシューマは、無限量のバッファリングをイベントマネージャ制御ユニット３０５には委ねられないので、この後者の値を供給することに応答する。この値が越えるべきとされた場合は、付加的なイベントが廃棄される。そして、コンシューマが獲得する次のイベントは、イベントマネージャ制御ユニット３０５自身からのOverrunイベントである。テーブル２のコード例においてこの特徴を示す。

テーブル２

```
typedef ObjectID EventConsumerID;
typedef FilterName EventConsumerName;
OSStatus CreateBroadcastConsumer(
        EventConsumerID     *consumer,
        EventConsumerName       name,
        unit32                  maxPending);
OSStatus DisposeBroadcastConsumer(
        EventConsumerID     consumer);
```

ブロードキャストコンシューマがEventConsumerIDを持つと、関連するイベントの種類に加入することになる。加入処理の一部において、ブロードキャストコンシューマは、イベントの通知をコンシューマが必要とする場合を決定するために照合されるべきイベントネーム（EventName）とイベントサブジェクト（EventSubject）の両方を特定できる。イベントディストリビュータ３４０によるイベントサブジェクト（EventSubject）の照合の方法は、処理されるイベントの種類に依存する。本実施形態によれば、この照合は、バイト毎の比較によって実行することができる。また、他の適応可能な実行方法を用いてもよい、例えば、３つの種類が異なる加入目的が同一なイベントがある場合は、即ち、同一のコンシューマにそれらのイベントが通知される場合は、それゆえ、これらのイベントは、たとえビットが実際に異なっていても同一のものとして比較される。EventConsumerIDに関連するイベントのリストは、加入ブロードキャストコンシューマ関数（SubscribeBroadcastConsumer）の使用によっていつでも拡張することができる。また、イベントの種別は、非加入ブロードキャストコンシューマ関数（UnsubscribeBroadcastConsumer）によってリストからいつでも取り除くことができる。特定のイベントの種類が非加入である場合、既にそのコンシューマに収集された種類のイベントの手続は破棄される。これをテーブル３に示す。

テーブル３

```
OSStatus SubscribeBroadcastConsumer(
        EventConsumerID     consumer,
        EventName           eventName,
        EventSubjectPtr     subject);
OSStaus UnsubscribeBroadcastConsumer(
        EventConsumerID     consumer,
        EventName           eventName,
        EventSubjectPtr     subject);
```

イベント処理

ホールドイベント（HoldEvents）／アンホールドイベント（UnholdEvents）コールは、イベント処理において、ある特殊なケースを平易にするために供給される。イベントの種類が保持されると、ブロードキャストコンシューマのためにこれらのイベントの依頼が収集される。しかしながら、それらのイベントが保持されなくなるまで、それらのイベントはConsumeコールによるユーザへの返答はなされない。この方法においては、イベントマネージャが特定の種類のイベントがまだブロードキャストコンシューマと関連する状況を取り扱うが、これらのイベントの処理は一時的に不可能となるか、あるいは優先されない。他の実施形態によれば、同様の機能が２つに分割されたブロードキャストコンシューマＩＤを生成することによって提供され得る。これらの一方は非保持のイベントであり他方は保持可能なイベントである。ブロードキャストコンシューマは、第２のEventConsumerIDから消費する場合を選択することができる。しかしながら、本実施形態では完全には実用的ではない、なぜなら、何がどのカテゴリに属するかを常に予想することはできないからであり、また、保持のための決定処理が、しばしばイベントの処理サイクルから完全に取り除かれてしまうからである。本実施形態に従うホールドイベント（HoldEvents）／アンホールドイベント（UnholdEvents）コールは、その決定処理を分割するための平易なインタフェースを供給する。これらのコールをテーブル４に示す。

テーブル４

```
OSStatus HoldEvents(EventConsumerID     consumer,
        EventName           eventName,
        EventSubjectPtr     subject);
OSStaus UnholdEvents (EventConsumerID   consumer,
        EventName           eventName,
        EventSubjectPtr     subject);
```

最終的に、イベントに関係するリストが構築されると、ブロードキャストコンシューマは同期コールと非同期コールの二つのコールの一つを用いて収集されたイベントを消費する。両方のコールは、キューを特定するためのコンシューマのobjectIDを特定し、イベントがコピーされるバッファを提供する。更に、Asyncコールが、新しいイベントでバッファが満たされた場合に実行される動作を記述するEventNotifucation構造を要求する。一つの実施形態によれば、EventNotifucation構造とEventInformationPtrはオペレーティングシステムのカーネルによって定義される型である。供給されたバッファがイベント記述を保持するのに十分な大きさでない場合、記述の主要な部分がバッファへとコピーされる。

そして、残りの記述は捨てられる。これらのコールをテーブル５に示す。

テーブル５

```
OSStatus ConsumeEvent(
        EventConsumerID     consumer,
        unit32              maxEventSize,
        EventInformationPtr event);
OSStaus ConsumeEventASync(
        EventConsumerID     consumer,
        EventNotification   *copletion,
        unit32              maxEventSize,
        EventInformationPtr event);
```

FlushEventsは、ブロードキャストコンシューマのために収集されたイベントで、まだ消費されていないイベントを取り除くために用いられる。ブロードキャストコンシューマは、イベントネーム（EventName）（ワイルドカードを用いることができる）とイベントサブジェクト（EventSubject）を特定する。イベントマネージャ制御ユニット３０５は、記述が照合されたイベントと未処理の収集されたイベントを処分するが、処理はしない。このコールをテーブル６に示す。

テーブル６

```
OSStatus FlushEvents(
        EventConsumerID     consumer,
        EventName           eventName,
        EventSubjectPtr     subject);
```

シーケンシャルコンシューマはＡＰＩに対してかなり異なる要求を有している。それらはイベントの伝達のボトルネックとなるように機能するので、それらのＡＰＩは、最大スループットを得るように設計されるべきであり、イベントマネージャ制御ユニット３０５が次のシーケンシャルコンシューマあるいはブロードキャストコンシューマにイベントを渡すことができるときを提示するために、各イベントの応答を要求するように構成すべきである。これに基づき、本発明の一実施形態のＡＰＩでは、イベントは適切なコミュニケーション機能を用いて、メッセージとしてシーケンシャルコンシュームに渡されるものとし、これは、例えば、上述で説明したマイクロカーネルオペレーティングシステムによって供給されるメッセージングシステムのようなものである。本実施形態によれば、シーケンシャルコンシューマは、次のイベントの要求をイベントマネージャ制御ユニット３０５に送信し、応答のメッセージでイベント自身が返送される。本開示に基づいて、当業者が採用し得る他の実行方法でも実現可能である。一実施形態によれば、シーケンシャルコンシューマは、迅速な応答時間を保証するために受信動作とプログラムの実行が重複することを許容する受任機能を介してあるいは非同期受信によってメッセージを受信する。そのようなメッセージングシステムの一例は、ＮｕＫＥＲＮＥＫ（登録商標）によって提供される。

一実施形態によれば、メッセージングシステムがファンネリングメカニズム（funneling mechanism）を提供し、シングルポートと協動する複数のオブジェクトを有しているので、加入コールがシーケンシャルコンシューマに対しは必要とされない。その代わり、シー

ケンシャルコンシューマは、関連する各イベントの種類に対するオブジェクトを生成することが要求され、また、これらのオブジェクトはコンシューマが所望するあらゆる方法でポートと関連付けられ得る。これらのオブジェクトは、後述するＡＰＩを用いてイベントの種類に関連付けられる。上記構成の他の適切な実施形態も本開示に基づいて当業者が適用し得る範囲である。

一実施形態に従えば、コンシューマの実際のシーケンスは、メッセージフィルタリングメカニズムの使用によって、リストメッセージ、スクリーンメッセージ、変更メッセージ、或いは再発送メッセージに明瞭に提供される。メッセージフィルタは、他の処理に優先してあるフィルターの使用を許可するために所定の優先順位で順序付けすることができる、合成されたメッセージインターセプションデバイス上のソフトウェアである。このメカニズムの詳細は、上記のSaulpaughとSzymanskiの名で出願された出願に記載されている。

InstallSequentialConsumerコールは、与えられた名前とサブジェクトのイベントが与えられたオブジェクトに渡されるべきであることをイベントマネージャ制御ユニット３０５に報告するために用いられる。名前、順序、オプションパラメータは、これらのイベントと関係のあるシーケンシャルコンシューマにイベントを与える実行の順番を決定するために用いられる。RemoveSequentialConsumerコールはインストールされたシーケンシャルコンシューマを取り除くために用いられる。インスタレーションコールとリムーバルコールをテーブル７に示す。

テーブル７

```
typedef FilterID SequentialConsumerID;
typedef FilterName SequentialConsumerName;
OSStatus InstallSequentialConsumer(
        SequentialConsumerID        *consumer,
        SequentialConsumerName      consumersName,
        ObjectID                    consumersObject,
        FilterOptions               consumerOptions,
        FilterObjectPair            consumerOrdering,
        EventName                   eventName,
        EventSubjectPtr             eventSubject);
OSStaus RemoveSequentialConsumer(
        SequentialConsumerID        Consumer);
```

一度シーケンシャルコンシューマがインストールされ、イベントを受信し或いは消費すると、イベントマネージャ制御ユニット３０５が次のシーケンシャルコンシューマ（シーケンシャルコンシューマがそれ以上ない場合にはブロードキャストコンシューマ）にイベントを渡すことができるように、シーケンシャルコンシューマは、そのイベントの処理を終えたときを提示することが必要である。特に、シーケンシャルコンシューマは、イベント処理を完了したことを示すNextConsumerコールを、イベントマネージャ制御ユニット３０５に対して作成する。イベントマネージャ制御ユニット３０５は、次のシーケンシャルコンシューマ、あるいは適切なブロードキャストコンシューマにイベントを渡す。そのイベントを渡す代わりに、シーケンシャルコンシューマは、あらゆる様式のあらゆるコンシューマに対し分配されないようにイベントのハンドリングを宣言する能力を有する。特に、シーケンシャルコンシューマは、上記のイベントの分配を禁止するためのEventHandledコールを生成することができる。これらの二つのコールをテーブル８に示す。NextConsumer

は、メッセージングシステムコールであるContinueMessageを用いる。このContinueMessageは、メッセージングシステムにおいてフィルタチェーンにおける次のオブジェクトに継続する自動フォワードコマンドとして実行される。また、EventHnadledは、メッセージングシステムコールであるReplyToMessageを用いるこれは、チェーンにおけるすべてのオブジェクトがそのメッセージを処理したことを特定する。このメカニズムの詳細は、上記のSaulpaughとSzymanskiの名で出願された出願に記載されている。

テーブル８

```
OSStatus NextConsumer(MessageID        eventMessageID);
OSStaus  EventHandled(MessageID        eventMessageID);
```

ディストリビュータ

上述したように、各ディストリビュータは異なる種類のイベントを処理するものである。各ディストリビュータはそれゆえ、処理すべきイベントを知っている必要がある。

一つの実施形態によれば、大部分のイベントが適切なシーケンシャルコンシューマを介して発送され、このイベントに関連することを提示した全てのブロードキャストコンシューマにそのコピーが与えられる。デフォルト時の実施によれば、イベントディストリビュータによるあらゆるイベントサブジェクトの整合チェックが全ての値に渡る直接的なバイト比較によってなされ、どのコンシューマがそのイベントを通知されなければないかを決定する。本開示に基づいて当業者に明らかな範囲で、他の適切な実施の形態が採用され得る。

一つの実施形態によれば、デフォルトディストリビュータは、ある種のイベントを全てのブロードキャストコンシューマに渡すべく提供される。例えば、バッテリー低下イベントのようなイベントは、常に全ての関与するコンシューマに分配される。デフォルトディストリビュータによって分配され得る各イベントはテーブル９において示されるRegisterEventWithDefaultDistributorコールによって、デフォルトディストリビュータに登録されなければならない。このコールは、システムの稼働中に、イベントの種別毎に１回ずつ行なわれ、それらのイベントを担当するデフォルトディストリビュータに通知される。

テーブル９

```
OSStatus RegisterEventWithDefaultDistributer(
                        EventName       eventName);
```

カスタムディストリビュータは、イベントマネージャ制御ユニット３０５に登録され、デフォルトディストリビュータで扱われない種類のイベントを扱う。一つの実施形態によれば、カスタムディストリビュータをイベントマネージャ制御ユニット３０５に登録することは、イベントをオブジェクトに連携させるコールによって遂行される。ここで、そのコールは、関連するイベントが発生した場合にイベントマネージャ制御ユニット３０５がどのディストリビュータでそのイベントの分配を扱うかを知ることができるように、イベントをオブジェクトに連携させるものである。カスタムディストリビュータは独立的にロードされたサービスであり、プロデューサとディストリビュータ間の全てのコミュニケーションがメッセージの送出及び受信によって達成される。処理及び分配を行うべくイベントを受信するために、各ディストリビュータはObjectIDを提供しなければならない。イベントは、そのObjectIDに対してプロデューサから送られることになる。カスタムディストリビュータの登録及び抹消のためのコールをテーブル１０に示す。

テーブル１０

```
OSstatus RegisterDistributor(
                    EventName       eventname,
                    ObjectID        distributor);
OSStatus UnregisterDistributor(
                    EventName       eventName,
                    ObjectID        distributor);
```

どのブロードキャストコンシューマがプロデューサによって報告されたイベントの通知を受け取るべきかを決定する処理には、ディストリビュータがより複雑な試行錯誤を適用する必要のある幾つかのケースがある。サブスクリプションマトリクスにおける情報は、ディストリビュータがどのブロードキャストコンシューマに通知するかを決定するのを補佐するものであり、ディストリビュータにとって有効である。一実施形態によれば、ディストリビュータは、それを要求していないブロードキャストコンシューマにイベントを与えることはできない。そして、ディストリビュータはサブスクリプションマトリクスを調べ、そのイベントを要求しているコンシューマを見つける。しかしながら、ディストリビュータは、それを要求したブロードキャストコンシューマのサブセットのみにイベントを与えることができる。ディストリビュータがイベントの分配に影響する必要があるであろうこの種の方法は以下のものを含む。

１．　イベントサブジェクトのマッチングをとる方法は、直接的なバイト比較よりも複雑である。考慮される必要のあるサブジェクト識別におけるサブストラクチャがあるか、もしくは、扱われる必要のあるワイルドカード処理の幾つかのフォームがある。

２．　特定のイベントは、おそらく一つであろうが、関連を提示しているブロードキャストコンシューマのサブセットに送られることを必要としてよい。例えば、マウスイベントは、それに関与するアプリケーションのうちの一つの適切なアプリケーションに送られる必要がある。

３．　関連付けられたイベント結合されることを必要としてもよい。例えば、同じウインドウに対する複数の領域更新イベントは、アプリケーションがそのイベントを実際に処理するまで結合されるべきである。

なお、これらの可能な形態は、シーケンシャルコンシューマには何の影響も及ぼさない。シーケンシャルコンシューマが必要とされる理由の一つが分配のプロセスに影響を与えるものだからである。言い換えれば、シーケンシャルコンシューマは、それらが関与するイベントの種類に対して反応しなければならないので、そのようにラベル付けされる。こうして、ディストリビュータが特定のシーケンシャルコンシューマにイベントを送らないように選択するという状況はなくなる。それゆえ、イベントマネージャ制御ユニット３０５は全ての生成されたイベントを、ディストリビュータに手渡す前に、シーケンシャルコンシューマのガントレット（gauntlet）を介して送る。

図９Ａ、９Ｂ、９Ｃは、本発明の実施形態にかかるイベントプロデューサの観点、及びイベントディストリビュータの観点からのイベントハンドリング処理を示すフローチャートである。

図９Ａに示されるように、イベントプロデューサは最初にイベントを検出、もしくは生成する（ステップ６００）。イベントプロデューサはイベント記述を生成する（ステップＳ６０２）。そして、ブロードキャスト及びシーケンシャルコンシューマの双方の、適切なイベントコンシューマにイベントを送るために、イベントマネージャ制御ユニット３０５をコールする（ステップ６０４）。

図９Ｂは、本発明の一実施形態にかかるイベントマネージャ制御ユニット３０５によるイベントの処理を示す。以下では、コンピュータのエレメント間にイベントを送信するため

にメッセージングシステムを使用した場合を説明するが、本開示によって当業者には明らかとなる他の適切な形態によって実施されてもよい。
ステップ６０６において、イベントプロデューサが、そのイベントがハンドリングされた時点を知ることを必要としているか否かを判断する。すなわち、その要求を、同期或いは非同期のいずれで処理するかを判断する。もし、イベントプロデューサがいつそのイベントがハンドリングされたかを知る必要がなければ（そのコールの入力パラメータによって示される）、プロデューサに戻され（ステップ６０８）、そして、この図における残りのステップを非同期にて処理する（ステップＳ６１０）。イベントプロデューサが知ることを必要とする場合は、同期による処理が行なわれる（ステップ６１２）。
テストは、ステップ６０６で送られたメッセージを非同期メッセージとすることによって実施される。さもなければ、それは同期メッセージである。特に、イベントプロデューサは、検出されたイベントに関るメッセージを送るためにイベントマネージャ制御ユニット３０５をコールする。プロデューサは応答を待つか否かを示す値を有する入力パラメータを提供する。このケースにおいて、メッセージは同期して送られる。さもなければ、メッセージは非同期で送られる。取り決めに従って、メッセージが同期として送られると、イベントマネージャ制御ユニット３０５は、そのメッセージが全てのシーケンシャルコンシューマによって処理されるまで、コールから戻らない。もし、メッセージが非同期で送られるなら、メッセージが送られるや否や、イベントプロデューサがメッセージが処理されたかどうかを知り得ないうちに、イベントマネージャ制御ユニット３０５は、イベントプロデューサに戻る。
ステップ６１４において、イベントマネージャ制御ユニット３０５は、シーケンシャルコンシューマデータベース３５０内のイベント記述にマッチするエントリを有する各シーケンシャルコンシューマにイベントを送る。これは一度に一回行なわれ、各シーケンシャルコンシューマは、イベントマネージャ制御ユニット３０５をコールすることにより、次のシーケンシャルコンシューマへ送られるメッセージを有することができるか、あるいはイベントが処理されたことを宣言することができる。こうして、シーケンシャルコンシューマがそのイベントの処理を完了した後、シーケンシャルコンシューマはそのイベントの処理を継続するべきかどうか、もしくはそのイベントが処理されたのでそのイベント処理をストップするかどうかを示す（ステップ６１６）。そして、シーケンシャルコンシューマがイベントをハンドリングしているかどうかの判断がイベントマネージャ制御ユニット３０５によってなされる（ステップ６１８）。もしハンドリングされているならば、他のコンシューマへのイベントの分配が禁止され、処理はプロデューサをコールする処理へ戻る。シーケンシャルコンシューマがイベントのハンドリングを行っていないことを示すと、シーケンシャルコンシューマが最後のものかどうかを判断する（ステップ６２２）。もし最後でなければ、ディストリビュータはステップ６１６へ戻り、次のシーケンシャルコンシューマへイベントを分配する。
次に図９Ｃを参照する。全てのシーケンシャルコンシューマへイベントが分配されると（ステップ６２２）、イベントマネージャ制御ユニット３０５は、ブロードキャストコンシューマに分配するために、そのイベントの種類に応答するイベントディストリビュータにイベントを送る（ステップ６２４）。イベントはディストリビュータへメッセージとして送られる。全てのフィルタがそれを処理し、何れもが応答しなくなった場合に（それによって、イベントがハンドルされたことを宣言する）、メッセージングシステムはそのイベントをディストリビュータへ自動的に通過させる。例えば、この時点でイベントマネージャ制御ユニット３０５にイベントディストリビュータをコールさせるというような、本開示に基づいて当業者が想到し得る他の適切な実施が可能である。
プロセス６２４を図９Ｄに示す。ディストリビュータはこのプロセスにおいてどのようにイベントを分配するかを決定する。一つの実施形態によれば、これは、加入したコンシューマのどれがイベントを得るべきであるかを見極めるために、その種類のイベントに対する全加入者を順番にならべるために、イベントマネージャ制御ユニット３０５をコールすることによってなされる。一つの状態は、ディストリビュータが全てのもしくはいくつか

のコンシューマにイベントを送る必要がある場合であり、しかしながら、コンシューマが受信し処理するまでに、イベントデータ自身を修正することを可能とする必要がある場合である（ステップ６２４０）。例えば、ウインドウ環境において、ウインドウが露出した場合（例えば、一番上のウインドウが終了した場合）、ウインドウ露出イベントが生成される。露出されたウインドウの別の部分がさらに露出され得るかもしれないので、ウインドウ露出イベントディストリビュータは、そのイベントが処理されてしまうまでに、このイベントを修正することを可能とすることを要求する。この場合、ディストリビュータは、実際のメッセージのためにプレイスホルダ構造を提出し、これを適切なコンシューマに与える。

プレイスホルダ構造は、イベントを識別するためにディストリビュータによって用いられるレファレンスコンスタントフィールド、RefConと、ディストリビュータから実データを得るために用いられるObjectIDを含む。ディストリビュータは、プレイスホルダを生成する際にRefConフィールドの値を提供する。そして、その値を取得し、実際のイベント記述を再度提供することを可能とする。こうして、コンシューマが実際にプレイスホルダを消費しようとするとき、イベントマネージャ制御ユニット３０５はメッセージをRefConを含む与えられたObjectIDに送る。ディストリビュータは、バッファに、現在の実際のイベント記述を入れ、コンシューマのバッファはConsumeへ渡される。この機能は選択された或いは全てのコンシューマの双方に提供されるので、テーブル１１に示すように２つのコールが提供される。全てのコンシューマがプレイスホルダを用いてイベントを通知されるならば（ステップ６２４８）、GivePlaxeholderToAllConsumersコールが使用される（ステップ６２５０）。もし、選択されたコンシューマのみがプレイスホルダを用いてイベントの通知を受けるのであれば（ステップ６２４８）、GivePlaxeholderToSelectConsumersコールが使用される（ステップ６２５２）。

テーブル１１

```
Osstatus GivePlaceholderToAllConsumers(
        EventInformationPtr         event,
        void*                       PlaceholderRefCon,
        ObjectID                    fullfilemntID);
Osstatus GivePlaceholderToSelectConsumers(
        EventInformationPtr         event,
        ConsumerFilterFunction      filterFunc,
        void*                       PlaceholderRefCon,
        ObjectID                    fullfilemntID);
```

もしディストリビュータがイベントデータを修正する必要がないなら、ステップ６２４０における判断の結果は「否」となる。そして、ステップ６２４２において、全てのブロードキャストコンシューマがイベントを受信すべきかどうかを判断する。なお、このような場合が常であるなら、自動的に全てのコンシューマに通知するデフォルトディストリビュータを用いることは好ましいことである。他方、特殊な処理がイベントのサブセットのみに要求され、そのサブセット内に存在しないあらゆるイベントに要求されると、分配メカニズムを起動するためにGiveEventToAllConsumersコールが使用され得る（ステップ６２４４）。このコールをテーブル１２に示す。

テーブル１２

```
OSStatus GiveEventToAllConsumers (EventInformationPtr      event);
```

メッセージを受信するためにディストリビュータがコンシューマのサブセットを選択する

ことを必要とする状況は次のいずれかの理由で発生する。すなわち、ディストリビュータが整合するいかなるサブジェクトをも処理する必要があるため、あるいは、イベントの性質がステップ６２４０におけるテストへの応答がないことによって検出された分配を制限することを要求するために発生する。特に、サブジェクトフィールドのバイト毎の比較が要求されない場合、サブジェクトが整合するかどうかをディストリビュータが判断することになる。これを達成するために、GiveEventToSelectConsumersコールが用いられる（ステップ６２４６）。そして、フィルタ関数、ConsumerFilterFunction、が、その種のイベントに加入している各コンシューマについてコールされる（それらが記述するサブジェクトにかかわらず）。もし、イベントがそのコンシューマに与えられるべきものであれば、その関数は「真」を返す。そう出なければ、「偽」を返す。このコールをテーブル１３に示す。

テーブル１３

```
typedef Boolean (*ConsumerFilterFunction)(
                    EventInformationPtr     event,
                    EventConsumerName       consumer,
                    EventSubjectPtr         subscriptnSubj);
OSStatus GiveEventToSelectConsumers(
                    EventInformationPtr     event,
                    ConsumerFilterFunction  filterFunc);
```

ステップ６２４４、６２４６、６２５０、６２５２における各コールの後に、ルーチンはイベントマネージャ制御ユニット３０５に戻る（ステップ６２５４）。

再び、図９Ｃを参照する。ステップ６２５において、イベントマネージャ制御ユニット３０５は図９Ｄにおいてどのコールが使用されたかに従ってイベントをエンキューする（enqueue）。一つの実施形態によれば、ファーストイン、ファーストアウト（ＦＩＦＯ）の順序でイベントはエンキュー、デキューされる。イベントコンシューマがイベントに関して動作レディ状態となると、イベントキューにおける最上位のイベントをデキューして、そのイベントの要求に従った処理がなされる。

もし、図９Ｂの各ステップが同期して処理される（ステップ６２６）と、ルーチンはステップ６２８にてプロデューサに戻る。さもなければ、ルーチンはプロデューサのメッセージに応答する（ステップ６３０）。

図１０は、本発明の一実施形態にかかるシーケンシャルコンシューマの観点によるイベントハンドリングプロセスを示すフローチャートである。まず、シーケンシャルコンシューマはイベントマネージャ制御ユニット３０５をコールし、インストールする（ステップ７００）。本発明の一つの実施形態によれば、コンシューマはObjectIDを提供する。ObjectIDは、イベントがメッセージとして送信される先を示す。例えば、イベント記述へのポインタとともにコールされるファンクションポインタをコンシューマが提供可能とするというような、他の適切な形態を採用することも可能である。また、このような適切な実施は、本開示に基づいて当業者がなし得る範囲のものである。

シーケンシャルコンシューマは、イベントが発生するのを待つ間は、独自に処理を行う（ステップ７０２）。一つの実施形態によれば、イベントがメッセージとして送られるので、与えられたオブジェクトを通してメッセージを受理、受信するためにシーケンシャルコンシューマがメッセージングシステムをコールする。プロセスが同期かあるいは非同期かによって、別のことがシーケンシャルコンシューマによってなされてもよい。

イベントが受信されたとき（ステップ７０４）、シーケンシャルコンシューマは適切な方法でイベントに関して動作する（ステップ７０６）。そして、シーケンシャルコンシューマは、それが他のコンシューマ（シーケンシャルもしくはブロードキャスト）にこのイベ

ントを報告するかどうかを、或いはそのイベントを十分にハンドルしたので他のコンシューマはそれを見るべきではないのかどうかを判断する（ステップ７０７）。もしそのイベントが完了しており、他のものがそれを見るべきでないと判断されたなら、イベントマネージャ制御ユニット３０５をコールすることによって、イベントハンドラメッセージが送られる（ステップ７１１）。さもなければ、シーケンシャルコンシューマはイベントマネージャ制御ユニット３０５をコールし、イベントが次のコンシューマに受け渡されるべきであることを通知する（ステップ７０９）。シーケンシャルコンシューマは、ステップ７０２の待ち状態へ戻り、ループを形成する。

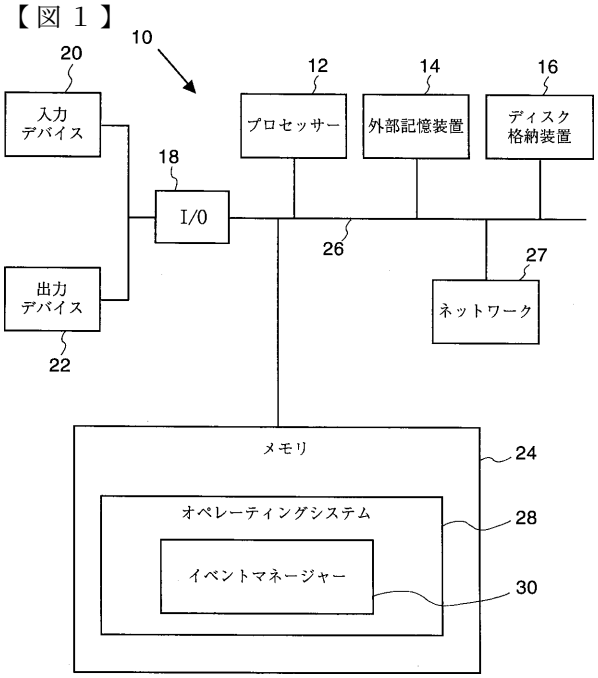
図１１は、本発明の一実施形態に係るブロードキャストコンシューマの観点からのイベントハンドリングプロセスを示すフローチャートである。ブロードキャストコンシューマはまず、イベントマネージャ制御ユニット３０５をコールし、CreateConsumerコールを用いてそれ自身を登録する。登録が行われると、ディストリビュータは、与えられたObjectIDによって参照されるオブジェクトを介してメッセージの受信を開始する。これらのメッセージの内容は、それらイベントを検出する或いは生成するプロデューサによって生成された記述である。ディストリビュータはイベントマネージャ制御ユニット３０５に、現在存在するコンシューマのどれがイベントを受信すべきかを提示しなければならない。一つの実施形態において、メッセージングシステムはイベントをプロデューサより送り出し、シーケンシャルコンシューマを経てディストリビュータに届ける。ディストリビュータはイベントマネージャ制御ユニット３０５を用いて適切なキューにイベントを置く。

イベントマネージャ制御ユニット３０５はコンシューマのためにイベントキューを生成し、戻る（ステップ８０２）。そして、ブロードキャストコンシューマは、イベントマネージャ制御ユニット３０５をコールし、それが関連する特定のイベントの種類に加入する（ステップ８０４）。イベントマネージャ制御ユニット３０５は、コンシューマのトラックをキープするためにサブスクリプションマトリクスにエントリを追加する（ステップ８０６）。

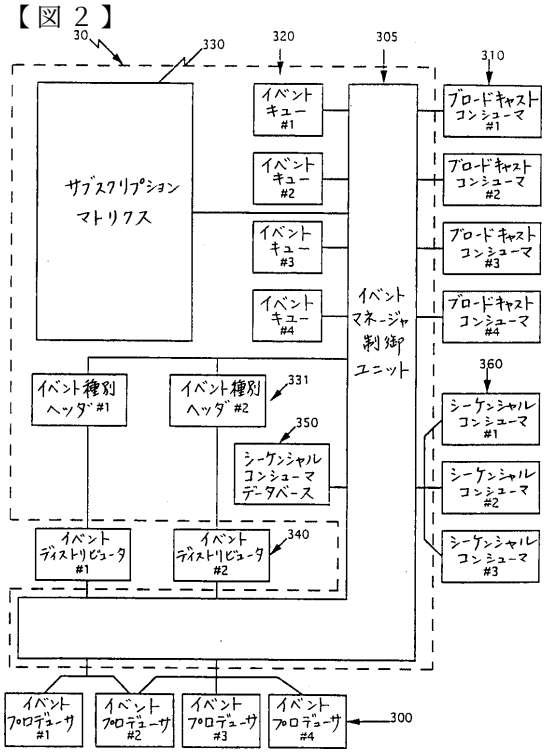
コンシューマはイベントマネージャ制御ユニット３０５をコールし、次のイベントを獲得する（ステップ８０８）。一実施形態によれば、これは、イベントマネージャ制御ユニット３０５へのメッセージの送信をコンシューマに行わせることによって実施される。このメッセージに対する応答は、そのコンシューマに対する次のイベントとなる。コンシューマは、これを同期もしくは非同期でおこなうオプションを有する。そして、コンシューマはメッセージを同期或いは非同期のいずれかで送る。なお、本開示に基づいて当業者が行える範囲で他の適切な形態を適用してもよい。コンシューマは、イベントに対応する動作を行い（ステップ８１０）、処理はステップ８０８に戻りループを形成する。

前述の詳細な実施形態は本発明の全体の性質を十分に開示するので、当業者は、現在の知識を適用することにより、総括的な概念から逸脱することなく、直ちに修正したり、種々のアプリケーションに適応させたりすることができる。それゆえ、そのような適応化、改造は、開示された実施形態の手段及び等価の範囲に含まれるべきものである。用語の表現は説明の目的で採用されたものであり、本発明を限定するものではないことは理解されるべきである。

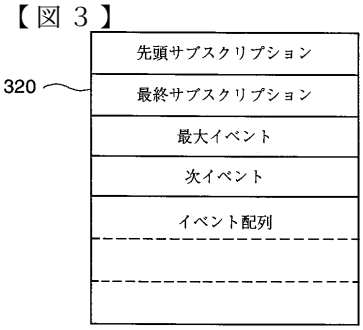
【図１】


10
20
入力デバイス
12
プロセッサー
14
外部記憶装置
16
ディスク格納装置
18
I/O
26
27
ネットワーク
出力デバイス
22
メモリ
24
オペレーティングシステム
28
イベントマネージャー
30

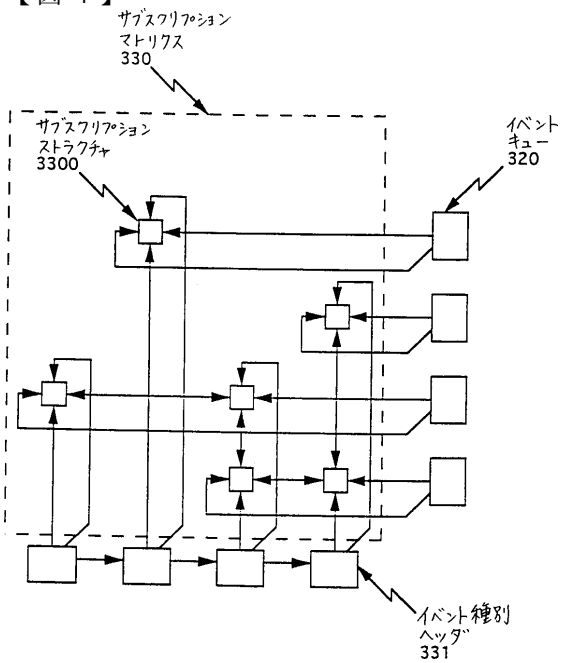
【図２】


30
330
320
305
310
サブスクリプションマトリクス
イベントキュー #1
イベントキュー #2
イベントキュー #3
イベントキュー #4
イベントマネージャ制御ユニット
ブロードキャストコンシューマ #1
ブロードキャストコンシューマ #2
ブロードキャストコンシューマ #3
ブロードキャストコンシューマ #4
360
シーケンシャルコンシューマ #1
シーケンシャルコンシューマ #2
シーケンシャルコンシューマ #3
イベント種別ヘッダ #1
イベント種別ヘッダ #2
331
350
シーケンシャルコンシューマデータベース
イベントディストリビュータ #1
イベントディストリビュータ #2
340
イベントプロデューサ #1
イベントプロデューサ #2
イベントプロデューサ #3
イベントプロデューサ #4
300

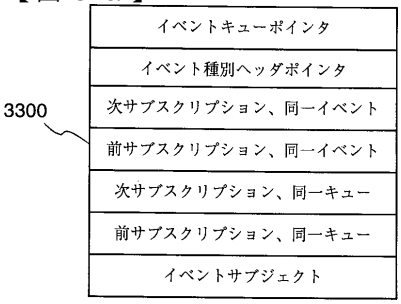
【図３】


320
先頭サブスクリプション
最終サブスクリプション
最大イベント
次イベント
イベント配列

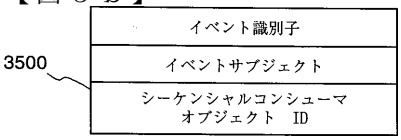
【図４】


サブスクリプションマトリクス
330
サブスクリプションストラクチャ
3300
イベントキュー
320
イベント種別ヘッダ
331

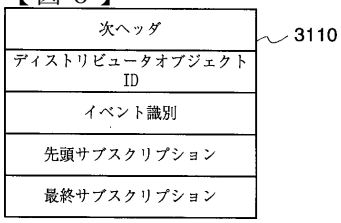
【図５ａ】



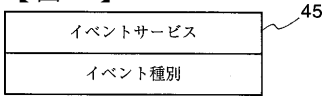
【図５ｂ】



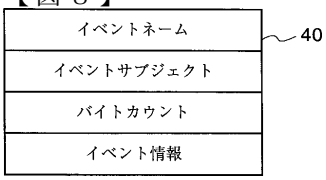
【図６】



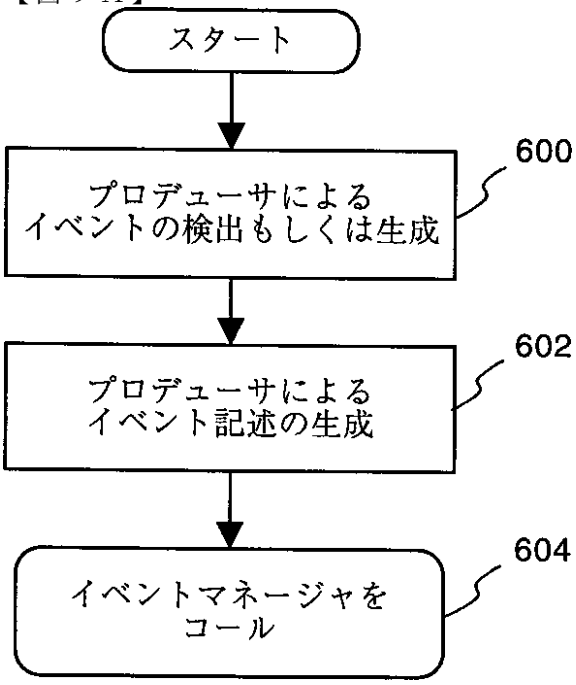
【図７】



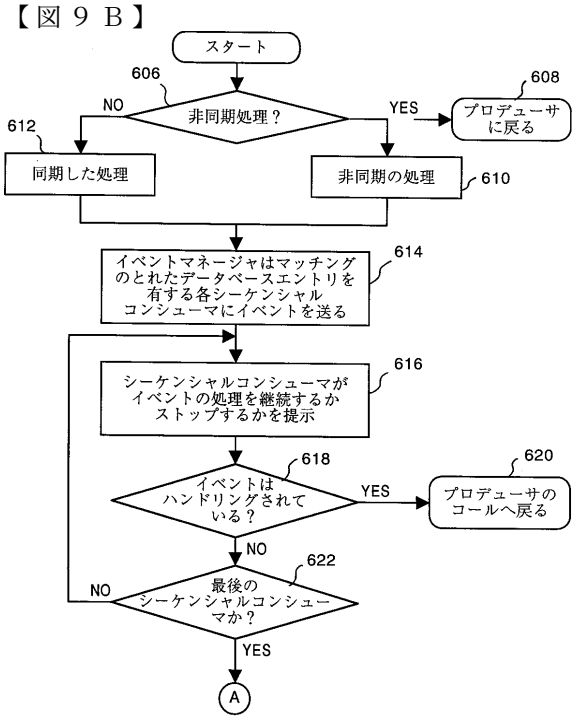
【図８】



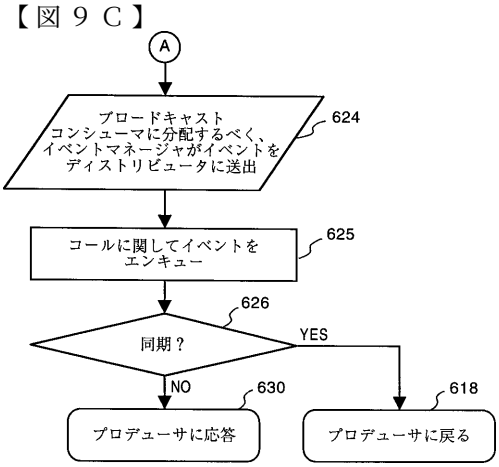
【図９Ａ】



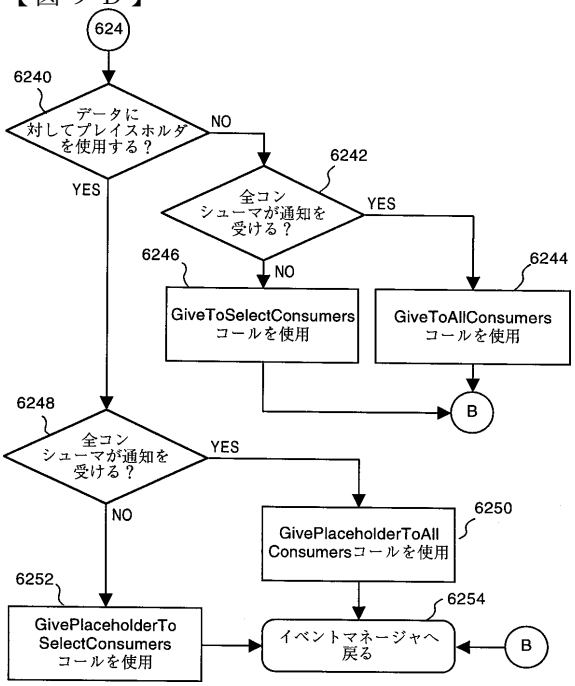
【図９Ｂ】



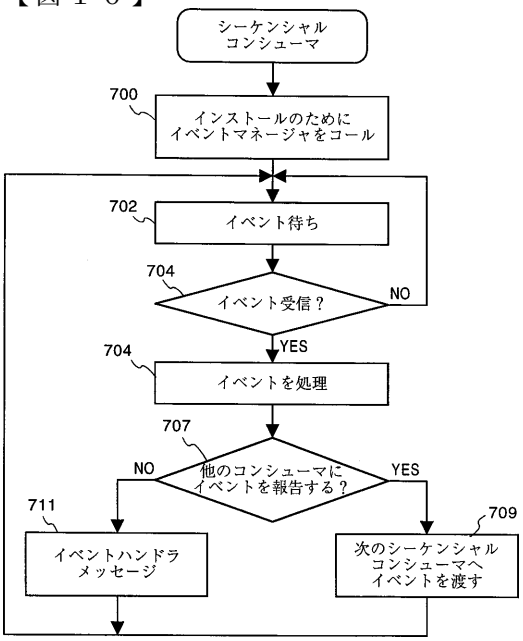
【図９Ｃ】



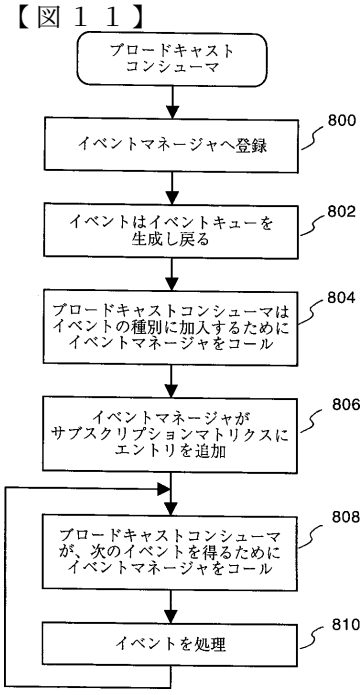
【図９Ｄ】


624
6240
データに対してプレイスホルダを使用する？
NO
YES
6242
全コンシューマが通知を受ける？
YES
NO
6244
6246
GiveToSelectConsumers コールを使用
GiveToAllConsumers コールを使用
B
6248
全コンシューマが通知を受ける？
YES
NO
6250
GivePlaceholderToAllConsumersコールを使用
6252
GivePlaceholderToSelectConsumers コールを使用
6254
イベントマネージャへ戻る
B

【図１０】


シーケンシャルコンシューマ
700
インストールのためにイベントマネージャをコール
702
イベント待ち
704
イベント受信？
NO
YES
704
イベントを処理
707
他のコンシューマにイベントを報告する？
NO
YES
711
イベントハンドラメッセージ
709
次のシーケンシャルコンシューマへイベントを渡す

【図１１】


ブロードキャストコンシューマ
イベントマネージャへ登録
800
イベントはイベントキューを生成し戻る
802
ブロードキャストコンシューマはイベントの種別に加入するためにイベントマネージャをコール
804
イベントマネージャがサブスクリプションマトリクスにエントリを追加
806
ブロードキャストコンシューマが、次のイベントを得るためにイベントマネージャをコール
808
イベントを処理
810

フロントページの続き

(72)発明者 ソルポー, トーマス, ユージーン
アメリカ合衆国 カリフォルニア州 ９５１２０ サン ホセ, ブレト ハルト ドライブ ６９３８
(72)発明者 キーナン, ウィリアム, ジョン
アメリカ合衆国 カリフォルニア州 ９４０６１ レッドウッド シティ ウエストウッド ストリート １２８７

審査官 殿川 雅也

(56)参考文献 特開平０７－１６０５１８（ＪＰ，Ａ）
欧州特許出願公開第００６５１３２８（ＥＰ，Ａ１）
特開平０５－１９７５６８（ＪＰ，Ａ）
国際公開第９５／０３１７８１（WO，Ａ１）
欧州特許出願公開第００５２８２２２（ＥＰ，Ａ１）
特開平０２－０８６２４３（ＪＰ，Ａ）
大森豊子、外3名，マルチメディア分散在席会議システム：ＭＥＲＭＡＩＤ一分散制御によるアプリケーション共有ー，テレビジョン学会技術報告，日本，社団法人テレビジョン学会，１９９１年 9月１９日，第１５巻、第４８号，第１７頁及至第２４頁
OS/2 2.0 Presentation Maneger Programming Guide，米国，QUE，１９９２年 3月，第31-5頁最終段落及至第31-6頁第3段落

(58)調査した分野(Int.Cl.^{7}，ＤＢ名)
G06F 9/46 - 9/54