(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4896385号
(P4896385)

(45)発行日　平成24年3月14日(2012.3.14)　(24)登録日　平成24年1月6日(2012.1.6)

(51)Int.Cl.　F I
G06F 21/00 (2006.01)　G06F 21/00　１５２
G06F 21/24 (2006.01)　G06F 21/24　１６７Ａ

請求項の数 24　(全 24 頁)

(21)出願番号　特願2004-278411 (P2004-278411)
(22)出願日　平成16年9月24日 (2004.9.24)
(65)公開番号　特開2005-129033 (P2005-129033A)
(43)公開日　平成17年5月19日 (2005.5.19)
審査請求日　平成19年9月19日 (2007.9.19)
(31)優先権主張番号　10/693,749
(32)優先日　平成15年10月24日 (2003.10.24)
(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　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　ケネス　ディー．レイ
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

最終頁に続く

(54)【発明の名称】アプリケーションのファクタリングによるアプリケーションへの高保証機能の統合

(57)【特許請求の範囲】
【請求項１】
アプリケーションのパーティショニングを管理するシステムであって、
少なくとも１つのプロセッサおよび該プロセッサに結合された少なくとも１つのメモリを有し、
前記プロセッサはベースコンポーネントを有するプログラム命令を実行するように構成され、
前記ベースコンポーネントは前記少なくとも１つのメモリに格納され、第１の環境および第２の環境のオペレーションをホストし、
前記アプリケーションは、該アプリケーションの第１のソフトウェアオブジェクトおよび該アプリケーションの第２のソフトウェアオブジェクトを有し、
前記第１のソフトウェアオブジェクトは第１のオペレーティングシステムを有する前記第１の環境において実行され、
前記第１のソフトウェアオブジェクトは複数のデータをハンドリングし、前記複数のデータの中の該第１のソフトウェアオブジェクトにより処理できない第１のデータを識別し、
前記第１のデータは改ざん阻止のためにセキュリティで保護されたデータであり、
前記第１のソフトウェアオブジェクトは前記ベースコンポーネントに前記複数のデータの第１のデータを送り、
前記ベースコンポーネントから当該複数のデータの第１のデータに対応する処理データ

を受け取り、
前記第１のソフトウェアオブジェクトは前記処理データを使用して前記複数のデータを情報処理し、
前記第２のソフトウェアオブジェクトは第２のオペレーティングシステムを有する前記第２の環境で実行され、
前記第２のソフトウェアオブジェクトは、前記複数のデータの第１のデータを前記ベースコンポーネントから、対応のラッパー付で受け取り、
前記第２のソフトウェアオブジェクトは、前記複数のデータの第１のデータが修正されていないことを前記対応のラッパーと当該第１のデータとを比較することにより検証し、
前記第２のソフトウェアオブジェクトは前記複数のデータの第１のデータの改ざんを阻止する方法で当該第１のデータを情報処理し、
前記第２のソフトウェアオブジェクトは当該情報処理された処理データを前記ベースコンポーネントに送り、
前記ベースコンポーネントは前記複数のデータの第１のデータを前記第１のソフトウェアオブジェクトから受け取り、当該複数のデータの第１のデータに対して対応のラッパーを適用する論理を有するかあるいはホストし、
前記対応のラッパーは、
前記第２のソフトウェアオブジェクトの前記第２の環境の指定と、
前記複数のデータの第１のデータが改ざんされたか否かを判定するための、および、前記複数のデータの第１のデータを前記第２の環境にルーティングするための、前記複数のデータの第１のデータをチェックすることができるシールとを有し、
その結果、前記アプリケーションの機能は前記第１のオペレーティングシステムおよび第２のオペレーティングシステムの間で分割され、
前記ベースコンポーネントは、さらに、前記第２のソフトウェアオブジェクトから処理データを受け取り、当該処理データに第２のラッパーを適用する論理を有するか、あるいはホストし、
前記第２のラッパーは前記第１のソフトウェアオブジェクトの第１の環境の指定を有すると共に、前記処理データが改ざんされたか否かを判定するための、および、前記処理データを前記第１の環境にルーティングするための、前記処理データをチェックすることができる第２のシールを有し、
その結果、前記アプリケーションの機能は前記第１のオペレーティングシステムおよび第２のオペレーティングシステムの間で分割される
ことを特徴とするシステム。
【請求項２】
前記第１のソフトウェアオブジェクトは前記複数のデータの第１のデータの代わりに別のデータを表示装置に表示させ、当該表示すべき別のデータは１つまたは複数の解読できない画像を含むことを特徴とする請求項１に記載のシステム。
【請求項３】
前記第２のソフトウェアオブジェクトによって行われる改ざん防止は、前記第２の環境が前記複数のデータの第１のデータに代わる別のデータを表示装置に関連付けられたビデオメモリに書き込み、前記表示装置の前記第１のデータを表示する場所に前記別のデータを表示するようにすることで、前記複数のデータの第１のデータの表示に対する干渉を防止することを含むことを特徴とする請求項１に記載のシステム。
【請求項４】
前記複数のデータの第１のデータがキーボードから入力されることと、前記第２のソフトウェアオブジェクトは提供する改ざん防止機能が前記複数のデータの第１のデータを前記キーボードから前記第２のソフトウェアオブジェクトの入力ストリームに転送する過程で改ざんを防止することを含むことを特徴とする請求項１に記載のシステム。
【請求項５】
第２のアプリケーションは前記複数のデータの第１のデータに署名することで、前記複

数のデータの前記第１のデータに対するそれ以降の改ざんを防止することを特徴とする請求項４に記載のシステム。
【請求項６】
前記第２の環境は、前記複数のデータの第１のデータと前記第２のアプリケーションで作成された署名に、前記複数のデータの第１のデータと前記署名が前記第２の環境で作成された証明として署名することを特徴とする請求項５に記載のシステム。
【請求項７】
前記ベースコンポーネントは第１の識別子を前記第２の環境に割り当てるコンポーネントを含むことを特徴とする請求項１に記載のシステム。
【請求項８】
前記複数のデータの第１のデータは、前記第１の識別子と前記第２のソフトウェアオブジェクトを識別する第２の識別子を含むことまたは伴うことを特徴とする請求項７に記載のシステム。
【請求項９】
前記第１の環境が前記第１の環境の挙動を説明する第１の情報に関連付けられていることと、前記第２の環境が前記第２の環境の挙動を説明する第２の情報に関連付けられていることと、前記第１の環境が前記第１の情報の示す挙動に従うことより前記第２の環境が前記第２の情報の示す挙動に従うことの方が高い保証レベルを提供することを特徴とする請求項１に記載のシステム。
【請求項１０】
前記複数のデータの第１のデータの改ざんを防止するために、前記第２のソフトウェアオブジェクトは第２の環境の挙動に依存することを特徴とする請求項９に記載のシステム。
【請求項１１】
前記ベースコンポーネントは前記第２の環境であるか、前記第２の環境に含まれることを特徴とする請求項１に記載のシステム。
【請求項１２】
アプリケーションの中の、第１のソフトウェア環境で動作する第１のソフトウェアオブジェクトの方法であって、
前記第１のソフトウェア環境はコンピュータ上で動作する第１のオペレーティングシステムを有し、およびデータをハンドリングし、該データは、要求されている機能を前記アプリケーションが正確に実行することができる信頼レベルを必要とし、前記コンピュータは、
前記データを安全に処理すべきであることを前記第１のソフトウェアオブジェクトにより、判断するステップと、
前記データを前記データを安全に処理するようにとの指示と一緒に、ベース環境に、前記第１のソフトウェアオブジェクトにより送信するステップと、
第２のソフトウェアオブジェクトが前記データを安全に処理すべきであることを前記ベース環境により判断するステップであって、前記第２のソフトウェアオブジェクトは前記第１のソフトウェアオブジェクトに対応している、判断するステップと、
前記データに対してラッパーを前記ベース環境により適用するステップであって、前記ラッパーは前記第２のソフトウェアオブジェクトの第２のソフトウェア環境を示し、該ラッパーは前記データが改ざんされたか否かを判定するための、前記データをチェックすることができるシールを有する、適用するステップと、
前記データおよび対応する前記ラッパーを前記第２のソフトウェア環境に、前記ベース環境により送るステップと、
前記第２のソフトウェア環境により改ざんを防止するステップであって、該改ざんを防止するステップは、前記データおよび前記対応するシールを使用して前記データが修正されていないことを判断するステップを有する、改ざんするステップと、
前記データを前記第２のソフトウェア環境により処理するステップと、

オリジナルの前記データを送ったソフトウェア環境に、当該データの処理結果を戻すようにとの指示と共に前記データの処理結果を前記ベース環境に前記第２のソフトウェア環境により送るステップと、
前記データの処理結果を戻すオリジナルのソフトウェア環境が前記第１のソフトウェア環境であることを前記ベースコンポーネントにより判断するステップと、
前記データの処理結果に対して第２のラッパーを前記ベース環境により適用するステップであって、前記第２のラッパーは第１のソフトウェア環境を示し、該第２のラッパーは前記データの処理結果が改ざんされたか否かを判定するための、前記データの処理結果をチェックすることができる第２のシールを有する、適用するステップと、
前記データの処理結果および前記第２のラッパーを前記第１のソフトウェア環境に前記ベース環境により送るステップと、
前記データの処理結果および前記第２のシールを使用して前記データの処理結果が修正されていないことを検証するステップと
を実行させることを特徴とする方法。
【請求項１３】
前記改ざん防止は前記データの変更防止を含むことを特徴とする請求項１２に記載の方法。
【請求項１４】
前記改ざん防止が前記画像表示装置の領域に前記データに代わる他のデータを表示することと前記領域に表示された前記表現以外の画像を消去することとを含むことを特徴とする請求項１３に記載の方法。
【請求項１５】
前記第１のソフトウェアオブジェクトは前記他のデータが画像表示装置に表示されるようにし、前記他のデータが１つまたは複数の解読できない画像を含むことを特徴とする請求項１４に記載の方法。
【請求項１６】
前記第１のソフトウェアオブジェクトまたは前記第２のソフトウェアオブジェクト、あるいは前記第１のソフトウェアオブジェクトと前記第２のソフトウェアオブジェクトの組合わせは、少なくとも１つの点で変更されて前記画像表示装置に表示されているアイテムが、前記第２のソフトウェアオブジェクトで生成された前記データの画像を表示できるようにすることを特徴とする請求項１２に記載の方法。
【請求項１７】
前記データがキーボードを使用して提供されることと、前記改ざん防止が前記データを前記キーボードから前記第２のソフトウェアオブジェクトの入力ストリームに転送する過程で変更を防止することを含むことを特徴とする請求項１２に記載の方法。
【請求項１８】
信頼レベルを必要とする前記データは前記第２のソフトウェアオブジェクトでハンドリングされることを特徴とする請求項１２に記載の方法。
【請求項１９】
前記データは、前記データを処理する場所として前記第２の環境を指定する第１のラベルを含むことまたは伴うことを特徴とする請求項１２に記載の方法。
【請求項２０】
前記データは、前記データを処理するプロセッサとして前記第２のソフトウェアオブジェクトを指定する第２のラベルを含むことまたは伴うことと、前記第２の環境は前記第２のラベルに基づいて前記第２のソフトウェアオブジェクトに前記データをルーティングすることを特徴とする請求項１９に記載の方法。
【請求項２１】
前記第２の環境が前記第２の環境の挙動を説明する第１の情報に関連付けられていることと、前記第２の環境が前記第１の情報の示す挙動に従うことを保証が規定することを特徴とする請求項１２に記載の方法。

【請求項２２】
前記第１のソフトウェア環境が前記第１のソフトウェア環境の挙動を説明する第２の情報に関連付けられていることと、前記第１のソフトウェア環境は前記第１のソフトウェア環境で行われる操作が正しく実行されるという第２の保証レベルを提供し、前記第２の保証レベルは前記第２のソフトウェア環境でおこなわれる操作が正しく実行される第１の保証レベルより相対的に低いことを特徴とする請求項１２に記載の方法。
【請求項２３】
前記コンピュータは、
前記第２のソフトウェア環境が前記第１のソフトウェア環境のために十分な信頼レベルを有していると前記第１のソフトウェア環境により判断して前記データの処理結果を信頼するステップと、
前記データの処理結果を使用して前記第１のソフトウェアオブジェクトを実行するステップと
をさらに実行させることを特徴とする請求項１２に記載の方法。
【請求項２４】
ユーザが第１のタイプのデータと第２のタイプのデータを操作できるようにするための符号化されたコードおよびデータを格納するコンピュータ可読記録媒体であって、前記第２のタイプのデータが前記第１のタイプのデータに比較して高レベルの改ざん防止を必要とするコンピュータ可読記録媒体において、前記コードおよびデータは、コンピュータに、
前記データを安全に処理すべきであることを前記第１のソフトウェアオブジェクトにより、判断するステップであって、前記第２のソフトウェアオブジェクトは前記第１のソフトウェアオブジェクトに対応している、判断するステップと、
前記データを前記データを安全に処理するようにとの指示と一緒に、ベース環境に、前記第１のソフトウェアオブジェクトにより送信するステップと、
第２のソフトウェアオブジェクトが前記データを安全に処理すべきであることを前記ベース環境により判断するステップと、
前記データに対してラッパーを前記ベース環境により適用するステップであって、前記ラッパーは前記第２のソフトウェアオブジェクトの第２のソフトウェア環境を示し、該ラッパーは前記データが改ざんされたか否かを判定するための、前記データをチェックすることができるシールを有する、適用するステップと、
前記データおよび対応する前記ラッパーを前記第２のソフトウェア環境に、前記ベース環境により送るステップと、
前記データおよび前記対応するシールを使用して前記データが修正されていないことを前記第２のソフトウェア環境により検証するステップと、
前記データを前記第２のソフトウェア環境により処理するステップと、
オリジナルの前記データを送ったソフトウェア環境に、当該データの処理結果を戻すようにとの指示と共に前記データの処理結果を前記ベース環境に前記第２のソフトウェア環境により送るステップと、
前記データの処理結果を戻すオリジナルのソフトウェア環境が前記第１のソフトウェア環境であることを前記ベースコンポーネントにより判断するステップと、
前記データの処理結果に対して第２のラッパーを前記ベース環境により適用するステップであって、前記第２のラッパーは第１のソフトウェア環境を示し、該第２のラッパーは前記データの処理結果が改ざんされたか否かを判定するための、前記データの処理結果をチェックすることができる第２のシールを有する、適用するステップと、
前記データの処理結果および前記第２のシールを前記第１のソフトウェア環境に前記ベース環境により送るステップと、
前記データの処理結果および前記第２のシールを使用して前記データの処理結果が修正されていないことを検証するステップと、
前記第２のソフトウェア環境が前記第１のソフトウェア環境のために十分な信頼レベル

を有していると前記第１のソフトウェア環境により判断して前記データの処理結果を信頼するステップと、
前記データの処理結果を使用して前記第１のソフトウェアオブジェクトを実行するステップと
を実行させることを特徴とするコンピュータ可読記録媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般にコンピューティングの分野に関する。具体的に言えば、本発明は、通常のセキュリティで保護されないソフトウェアに特定の保証レベルまたはセキュリティレベルを必要とする操作を統合できる方法で、アプリケーションのパーティショニング（分割）またはファクタリング（分解）をサポートするメカニズムを提供する。
【背景技術】
【０００２】
コンピューティングの分野では、一方で高度なセキュリティを提供するシステムと他方で多くの実用的な機能と高度な拡張性を提供するシステムとの間に対立関係がある。コンピューティング分野のセキュリティは、コンピュータシステムの挙動（ソフトウェアとハードウェアの両方の挙動）を確実に把握および予測できるかどうか、つまり、システムが不注意な誤用または故意の攻撃によって設計以外の動作をすることがないと保証できるかどうかにかかっている。たとえば、著作権法によって保護されたデータの複製を防止するように設計されたコンピュータシステムは、システムが実際に設計どおりに動作することを保証できる範囲でのみ保証できる。しかし、大規模なオープンアーキテクチャは扱いにくく、複雑になる傾向があり、挙動を左右し得る多くの可変要素があるので、その挙動を分析するのは困難である。現時点では、フルサービスのオペレーティングシステムやワードプロセッサのように大規模で複雑なプログラムについて、その挙動を確実に検証できる見込みはない。挙動をテストできる小規模なプログラムを記述し、さまざまな状況や攻撃のクラスについてテストすることはできるが、こうしたプログラムが提供できるのは限定的な機能群にすぎない。このように、多くの機能の提供と高いセキュリティの提供とは相反する問題である。
【発明の開示】
【発明が解決しようとする課題】
【０００３】
提示された１つのソリューションは、２つのシステム、すなわち高度な機能を備えた大規模システムと高度なセキュリティを提供する小規模システムを並列に実行することである。このように、ＷＩＮＤＯＷＳ（登録商標）ＸＰのようなフルサービスのオペレーティングシステムと小規模な高保証のオペレーティングシステムを同時に実行できる。フルサービスのオペレーティングシステムで高い信頼性を提供するように厳密に管理された形で実行する必要のある特定のイベントが発生すると、このタスクは高保証のオペレーティングシステムに渡される。
【０００４】
オペレーティングシステムは、他のプログラムを実行できる環境を提供する。しかし、２つのオペレーティングシステムが共存できるという事実だけでは、特定のアプリケーションが２つの環境をどう利用するのかという問題が残る。アプリケーションは、フル機能の環境を使用して多くの機能（高度なセキュリティを必要としない）を実行し、高保証の環境を使用して高度なセキュリティを必要とする機能を実行するのが望ましい。さらに、こうして２つの環境を利用してもユーザの使い心地の点では統合されているのが望ましい。
【０００５】
以上の観点から、先行技術の弱点を克服するシステムが必要である。
【課題を解決するための手段】

【０００６】
本発明は、アプリケーションの機能を複数の要素、すなわち特定のセキュリティレベルまたは保護レベルを必要とする操作とそうでない操作にファクタリングまたはパーティショニングするメカニズムを提供する。本発明に従って、アプリケーションを少なくとも２つのソフトウェアオブジェクトとして具体化する。つまり、１つはフル機能（ただし低保証）の環境で動作するソフトウェアオブジェクトであり、もう１つは高保証（ただし限られた機能）の環境で動作するソフトウェアオブジェクトである。フル機能の環境のオブジェクトを実行すると、特定の保護レベルを必要とするデータが検出されることがある（たとえば、データが機密性を必要とする場合、またはデータが改ざんされていないことを確認する意味で検証する必要がある場合）。フル機能の環境で動作するソフトウェアオブジェクトが保護を必要とするデータを検出すると、このソフトウェアオブジェクトによってそのデータが高保証の環境に渡される。ここで、高保証の環境で動作するソフトウェアオブジェクトがデータを操作する。このデータを処理するときにデータの入力、出力、保存が必要な場合はそのすべてを高保証の環境で実行し、高保証の環境の外部で発生するイベントによるデータの傍受や改ざんを防止する。
【０００７】
２つの環境は、両者間の通信に必要なインフラストラクチャ（または「プラミング（ｐｌｕｍｂｉｎｇ）」）を提供するベースコンポーネントによってホスティングされる。たとえば、高保証の環境で処理する必要のあるデータオブジェクトは、ベースコンポーネントで生成されたラッパーでラッピングされていてもよい。このようなラッパーによって、データオブジェクトを処理するルーティング先の環境を特定できる。ベースコンポーネントでラッピングされた後にデータオブジェクトが変更されていないという事実を検証できるシールを添付することもできる。したがって、ソフトウェアオブジェクトはデータオブジェクトをベースコンポーネントに渡すことができ、ベースコンポーネントは（１）どの環境にデータオブジェクトを渡す必要があるかを判断でき、（２）データオブジェクトが作成された後に改ざんされていないことを検証できる。前述の（２）の処理により、プラットフォーム全体にわたるオブジェクトの信頼性を確立できるインフラストラクチャが提供される。つまり、オブジェクトが第１のマシン（第１のベースコンポーネントを備える）で作成された場合、第１のベースコンポーネントはこのオブジェクトが確かにこのベースコンポーネントが認識する高保証の環境で作成されたことの証明としてオブジェクトに署名する。次に、オブジェクトを別のマシンで開く場合、もう１台のマシンがこの署名を検証し、署名元が保証できるかどうかを判断することができる（たとえば、一部のマシンおよび／またはベースコンポーネントはそのホスティング対象の環境の正確な挙動を他のマシンより適切に保証できる。つまり、各マシンは特定のデータオブジェクトが作成されたプラットフォームが保証できるかどうかを自ら判断できる）。
【発明を実施するための最良の形態】
【０００８】
本発明の他の機能について以下に説明する。
【０００９】
添付の図面を参照しながら以上の課題を解決するための手段と後述の発明を実施するための最良の形態を読むことにより、本発明をより深く理解できる。本発明を説明するために本発明の例示的な構成を図示するが、本発明は開示されたこの特定の方法および手段には限定されない。
【００１０】
本発明が提供するメカニズムを使用すると、アプリケーションをセキュリティで保護されたコンポーネントとセキュリティで保護されないコンポーネントにパーティショニングまたは「ファクタリング」でき、こうしたコンポーネントが連動することでユーザは統合されたアプリケーションとして利用できる。たとえば、ワードプロセッサプログラムはレイアウト、編集、印刷、スペルチェック、文法チェックなど、ワードプロセッサに関連する機能の多くを実行するセキュリティで保護されないコンポーネントと、何らかの方法で

保護する必要のあるデータオブジェクトの表示や編集を可能にするセキュリティで保護されたコンポーネントにパーティショニングされる。セキュリティで保護されないコンポーネントは、一般的な商用のオペレーティングシステムのように通常のオープンな環境で実行できる。セキュリティで保護されたコンポーネントは、特定のタイプのソフトウェアを実行でき、ソフトウェアの挙動が正しいことが確実に保証される高保証の環境で実行できる。本発明には、このようなケースに関連するさまざまな特徴がある。第１に、本発明によって２つのコンポーネントがユーザの使い心地の点では統合され、ユーザにとっては１つのアプリケーションを使用するのとほとんど変わらない。第２に、本発明によってアプリケーションがセキュリティで保護されたデータとセキュリティで保護されないデータの両方を操作でき、そのようなデータを別々のパーティションで使用できるインフラストラクチャまたは「プラミング（ｐｌｕｍｂｉｎｇ）」が提供される。

【００１１】

ユーザの使い心地の点では、アプリケーションのセキュリティで保護されない部分を使用して起動するのが一般的である。保護されない部分を使用中に発生するデータ（たとえば、ワードプロセッサ文書形式の機密のテキストアイテム、スプレッドシート形式の機密の財務諸表など）は、実際にはアプリケーションのセキュリティで保護されない部分では表示できなくても、ユーザの使い心地の点ではアプリケーションのセキュリティで保護されない部分によって何らかの形で統合されているのが好ましい。たとえば、ワードプロセッサ文書形式の機密のテキストアイテムが発生すると、ワードプロセッサのセキュリティで保護されない部分はこのアイテムを表現するボックスと何らかのグラフィック（テキストを表すくねった線など）を表示し、テキストが存在するがそれを表示できないという事実を表すことができる。１つの例では、ユーザが前述のボックスまたはグラフィックをクリックすると、アプリケーションのセキュリティで保護された部分が呼び出され、画面上でボックスが表示されていたのと同じ場所にテキストが表示される。このように、アプリケーションのセキュリティで保護された部分と保護されない部分はユーザの使い心地という点では統合されている。

【００１２】

インフラストラクチャに関しては、本発明によって１つの環境のデータオブジェクトを別の環境にルーティングできるメカニズムが提供される。一般に、機密のデータオブジェクト、つまりセキュリティで保護された部分で処理する必要のあるデータオブジェクトは暗号化されているので、セキュリティで保護されない部分では読み出すことができない。したがって、こうしたオブジェクトはデータオブジェクトの生成に関与する各コンポーネントによって一連のラッパーでラッピングされる。個々のラッパーには、ラッパーに添付されたコンポーネントの識別子とラッパー内のデータの整合性を検証できるシールが含まれるのが好ましい。外側のラッパーは、ラッパーを作成したマシン上の保証のルート、つまりベースコンポーネントによって添付されるのが好ましい。ただし、ベースコンポーネントは１台のマシン上に共存するさまざまな環境をホスティングし、よく知られている特定の権限によってその挙動が保証できることを検証されている。この外側のラッパーを使用すると、ベースコンポーネントはルータの役割を果たすことができる。このように、セキュリティで保護されない部分がこのオブジェクトを検出した場合、オブジェクトを読み出すことはできないが、それがもう１つの処理環境に送信する必要のあるオブジェクトであることは認識できる。したがって、セキュリティで保護されない部分はこのオブジェクトをベースコンポーネントに送信し、次にベースコンポーネントはラッパーで指定された環境に基づいてオブジェクトを適切な環境にルーティングする。

【００１３】

さらに、第１のマシンで作成されたオブジェクトを第２のマシンで開く場合、第２のマシンは外側のラッパーの一部である署名を調べることでオブジェクトの信頼性を確認できる。ベースコンポーネントがオブジェクトをラッピングし、シールを添付する場合は、オブジェクト作成中にベースコンポーネントが適切にその機能を実行し、オブジェクトが作成される過程で外部から改ざんされていないことを、実際にベースコンポーネントが証明

していることは言うまでもない（ベースコンポーネントは、一般に高保証の環境が自ら改ざんされないようにできるメカニズム、つまり保証できるプロセッサモジュール（ＴＰＭ：ｔｒｕｓｔｅｄ　ｐｒｏｃｅｓｓｏｒ　ｍｏｄｕｌｅ）、メモリアイソレーションメカニズムなどを提供する）。一部のベースコンポーネントはこの機能を他のベースコンポーネントより適切に実行できるので、データオブジェクトを開くマシンは実際にこのオブジェクトに適用するセキュリティ要件に従ってオブジェクトが作成されたことを保証できるかどうかを確認できる。たとえば、オブジェクトがキーボードから入力されたテキストの場合、高保証の環境ではキーボードからの入力をセキュリティで保護された形で受信できるが、この環境がキーボード入力をセキュリティで保護された形で受信できるかどうかは、ベースコンポーネントがキーボードから高保証の環境へのパスを保護できるかどうかにかかっている。ラッパーには特定のベースコンポーネントで生成されたシールまたは署名が含まれているので、ラッパーは保証モデルをサポートする。保証モデルでは、データオブジェクトがその作成元のマシンについて想定されたセキュリティに基づいていることをどの程度保証できるかを別のマシンが判断できる。

【００１４】

以下に、ファクタリングまたはパーティショニングされたアプリケーションの使用をサポートするシステム、方法、およびメカニズムについて説明する。

【００１５】

例示的なコンピューティング環境

図１は、本発明の機能を実現できる例示的なコンピューティング環境を示している。コンピューティングシステム環境１００は適切なコンピューティング環境の１つの例にすぎず、本発明の使い方または機能の範囲に関するいかなる制限を示すものでもない。また、コンピューティング環境１００は、例示的なオペレーティング環境１００に示すコンポーネントの１つまたは組合わせに関して依存性も要件もないものとする。

【００１６】

本発明は、他のさまざまな汎用または専用のコンピューティングシステム環境または構成にも適用できる。本発明の使用に適した周知のコンピューティングシステム、環境、および／または構成の例には、パーソナルコンピュータ、サーバーコンピュータ、ハンドヘルドまたはラップトップデバイス、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラム可能な家庭用電子機器、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、埋め込みのシステム、前述の任意のシステムまたはデバイスを含む分散コンピューティング環境などが含まれるが、これらに限定はされない。

【００１７】

本発明は、プログラムモジュールのようにコンピュータで実行可能な命令をコンピュータで実行する一般的なケースで説明できる。一般に、プログラムモジュールにはルーチン、プログラム、オブジェクト、コンポーネント、データ構造などがあり、特定のタスクを実行するものや、特定の抽象データ型を実装するものがある。本発明は、通信ネットワークなどのデータ送信媒体を介してリンクするリモート処理装置でタスクを実行する分散コンピューティング環境でも実施できる。分散コンピューティング環境では、メモリ記憶装置を含むローカルとリモートの両方のコンピュータ記憶媒体にプログラムモジュールとそれ以外のデータを格納できる。

【００１８】

図１を参照すると、本発明を実施する例示的なシステムに、コンピュータ１１０の形で汎用コンピューティングデバイスが配置されている。コンピュータ１１０のコンポーネントには、処理装置１２０、システムメモリ１３０、さまざまなシステムコンポーネント（システムメモリと処理装置１２０など）を接続するシステムバス１２１が含まれるが、それらに限定はされない。処理装置１２０は、マルチスレッドプロセッサでサポートする処理装置など、複数の論理処理装置を表すことができる。システムバス１２１は、さまざまなバスアーキテクチャの任意の１つを使用したメモリバスまたはメモリコントローラ、周

辺バス、ローカルバスを含む各種バス構造のいずれでもよい。こうしたアーキテクチャには、例として、ＩＳＡ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＭＣＡ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＥＩＳＡ（Ｅｎｈａｎｃｅｄ ＩＳＡ）バス、ＶＥＳＡ（Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ）ローカルバス、ＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）バス（メザニン（Ｍｅｚｚａｎｉｎｅ）バスとも呼ばれる）が含まれるが、これらに限定はされない。システムバス１２１は、ポイントツーポイント接続、スイッチングファブリック、あるいは通信装置間の同様の機能としても実現できる。

【００１９】

コンピュータ１１０には、通常はさまざまなコンピュータ可読媒体が含まれる。コンピュータ可読媒体は、コンピュータ１１０からアクセスできる任意の使用可能な媒体でよい。揮発性と不揮発性の両方、および取り外し可能と不可能の両方の媒体が含まれる。例として、コンピュータ可読媒体にはコンピュータ記憶媒体および通信媒体を含めてもよいが、これらに限定はされない。コンピュータ記憶媒体には、コンピュータ可読の命令、データ構造、プログラムモジュール、またはその他のデータなどの情報を記憶する任意の方法または技術で組み込まれた、揮発性と不揮発性の両方、および取り外し可能と不可能の両方の媒体が含まれる。コンピュータ記憶媒体には、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリなどのメモリ技術、ＣＤ ＲＯＭ、デジタル多用途ディスク（ＤＶＤ：ｄｉｇｉｔａｌ ｖｅｒｓａｔｉｌｅ ｄｉｓｋ）などの光ディスク記憶装置、磁気カセット、磁気テープ、磁気ディスクなどの磁気記憶装置、または必要な情報を格納でき、コンピュータ１１０からアクセスできる他の任意の媒体が含まれるが、これらに限定はされない。通信媒体は、搬送波やその他の搬送メカニズムのような変調データ信号に含まれるコンピュータ可読の命令、データ構造、プログラムモジュール、またはその他のデータなどを具体化したものであり、任意の情報伝達媒体を含む。「変調データ信号」という用語は、信号内で情報を符号化するように、１つまたは複数の特性を設定または変更された信号を意味する。例として、通信媒体には、有線ネットワーク、直接ワイヤ接続などの有線媒体と、音、ＲＦ、赤外線などの無線媒体が含まれるが、これらに限定はされない。上記の任意の組合わせも、コンピュータ可読媒体の範囲内に含まれるものとする。

【００２０】

システムメモリ１３０には、読み取り専用メモリ（ＲＯＭ：ｒｅａｄ ｏｎｌｙ ｍｅｍｏｒｙ）１３１やランダムアクセスメモリ（ＲＡＭ：ｒａｎｄｏｍ ａｃｃｅｓｓ ｍｅｍｏｒｙ）１３２のように揮発性および／または不揮発性メモリという形をとるコンピュータ記憶媒体が含まれる。起動時などにコンピュータ１１０内のエレメント間の情報転送を支援する基本ルーチンを含む基本入出力システム１３３（ＢＩＯＳ：ｂａｓｉｃ ｉｎｐｕｔ／ｏｕｔｐｕｔ ｓｙｓｔｅｍ）は、通常はＲＯＭ１３１に格納される。ＲＡＭ１３２には、通常は処理装置１２０から直ちにアクセスできる、および／または処理装置１２０で現在操作しているデータおよび／またはプログラムモジュールが格納される。例として、限定はしないが、図１にはオペレーティングシステム１３４、アプリケーションプログラム１３５、その他のプログラムモジュール１３６、およびプログラムデータ１３７が示されている。

【００２１】

コンピュータ１１０には、その他の取り外し可能／不可能、揮発性／不揮発性のコンピュータ記憶媒体を含めてもよい。単に例として、図１に取り外し不可能な不揮発性の磁気媒体の読み出しまたは書き込みを行うハードディスクドライブ１４０、取り外し可能な不揮発性の磁気ディスク１５２の読み出しまたは書き込みを行う磁気ディスクドライブ１５１、ＣＤ ＲＯＭや他の光媒体のような取り外し可能な不揮発性の光ディスク１５６の読み出しまたは書き込みを行う光ディスクドライブ１５５を示す。例示的なオペレーティング環境で使用できる上記以外の取り外し可能／不可能、揮発性／不揮発性のコンピュータ記憶媒体には、磁気テープカセット、フラッシュメモリカード、ＤＶＤ、デジタルビデオ

テープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭなどが含まれるが、それらに限定はされない。ハードディスクドライブ１４１は、通常はインターフェイス１４０などの取り外し不可能なメモリインターフェイスを介してシステムバス１２１に接続し、磁気ディスクドライブ１５１と光ディスクドライブ１５５は、通常はインターフェイス１５０などの取り外し可能なメモリインターフェイスを介してシステムバス１２１に接続する。

【００２２】

図１に示す前述のドライブとこれに対応するコンピュータ記憶媒体には、コンピュータ可読の命令、データ構造、プログラムモジュールなど、コンピュータ１１０のデータを格納できる。たとえば、図１を参照すると、ハードディスクドライブ１４１にオペレーティングシステム１４４、アプリケーションプログラム１４５、その他のプログラムモジュール１４６、およびプログラムデータ１４７が格納されている。ただし、こうしたコンポーネントは、オペレーティングシステム１３４、アプリケーションプログラム１３５、その他のプログラムモジュール１３６、およびプログラムデータ１３７と同じでも、異なっていてもよい。ここでは、オペレーティングシステム１４４、アプリケーションプログラム１４５、その他のプログラムモジュール１４６、およびプログラムデータ１４７には異なる番号を付けて、少なくとも別の複製であることを示している。ユーザは、キーボード１６２やポインティングデバイス１６１（一般に、マウス、トラックボール、またはタッチパッドと呼ばれる）を使用してコンピュータ２０にコマンドや情報を入力できる。他の入力装置（図示せず）には、マイクロフォン、ジョイスティック、ゲームパッド、サテライトディッシュ、スキャナなどを含めてもよい。これらの入力装置および他の入力装置は、多くの場合にシステムバスに接続されたユーザ入力インターフェイス１６０を介して処理装置１２０に接続するが、パラレルポート、ゲームポート、ＵＳＢ（ｕｎｉｖｅｒｓａｌ ｓｅｒｉａｌ ｂｕｓ）のような他のインターフェイスやバス構造によって接続してもよい。モニタ１９１または他のタイプの表示装置も、ビデオインターフェイス１９０のようなインターフェイスを介してシステムバス１２１に接続される。さらに、コンピュータには出力周辺インターフェイス１９５を介してスピーカ１９７やプリンタ１９６など、モニタ以外の出力周辺装置を接続できる。

【００２３】

コンピュータ１１０は、リモートコンピュータ１８０のような１台または複数台のリモートコンピュータへの論理接続を使用してネットワーク環境で動作できる。リモートコンピュータ１８０は、パーソナルコンピュータ、サーバー、ルータ、ネットワークＰＣ、ピアデバイス、または他の一般のネットワークノードでよい。通常は、コンピュータ１１０に関連して上で説明したエレメントの多くまたはすべてが含まれるが、図１にはメモリ記憶装置１８１のみを示す。図１に示す論理接続には、ローカルエリアネットワーク（ＬＡＮ：ｌｏｃａｌ ａｒｅａ ｎｅｔｗｏｒｋ）１７１とワイドエリアネットワーク（ＷＡＮ：ｗｉｄｅ ａｒｅａ ｎｅｔｗｏｒｋ）１７３が含まれるが、他のネットワークを含めてもよい。このようなネットワーキング環境は、職場、企業規模のコンピュータネットワーク、イントラネット、およびインターネットではごく一般的である。

【００２４】

ＬＡＮネットワーキング環境で使用した場合、コンピュータ１１０はＬＡＮ１７１にネットワークインターフェイスまたはアダプタ１７０を介して接続する。ＷＡＮネットワーキング環境で使用した場合、コンピュータ１１０は一般にインターネットなどのＷＡＮ１７３を介して通信を確立するためのモデム１７２またはその他の手段を備えている。モデム１７２（内蔵または外付け）は、ユーザ入力インターフェイス１６０または他の適切なメカニズムを介してシステムバス１２１に接続できる。ネットワーク環境では、コンピュータ１１０またはその一部に関連して記述したプログラムモジュールは、リモートメモリ記憶装置に格納できる。例として、図１にメモリデバイス１８１に格納されたリモートアプリケーションプログラム１８５を示すが、これには限定されない。図示されたネットワーク接続が例示的なものであり、コンピュータ間の通信リンクを確立する他の手段を使用してもよいことは言うまでもない。

【００２５】
パーティショニングまたは「ファクタリング」されたアプリケーション
アプリケーションプログラムなどのソフトウェアは、一般的にさまざまな機能を実行し、さまざまなデータを操作する。この意味で、アプリケーションはさまざまな機能の集まりと見なすことができる。アプリケーションをさまざまな機能に分割し、さまざまな機能を別々に実行できるようにすると便利である（たとえば、さまざまなセキュリティレベルを提供する環境に割り当てる）。図２に、アプリケーションをさまざまな機能に分割する方法を示す。
【００２６】
アプリケーション１３５は、さまざまな機能２０２（１）、２０２（２）、．．．、２０２（ｎ）、２０２（ｎ＋１）、２０２（ｎ＋２）、．．．、２０２（ｎ＋ｍ）で構成されている。たとえば、アプリケーション１３５はワードプロセッサプログラムであり、さまざまな機能は編集、表示、印刷、変更の追跡などであってもよい。図２によると、アプリケーション１３５はｎ＋ｍ個の別々の機能を備えており、何をもって独立した機能とするかを判定する何らかの基準が存在することに留意されたい。たとえば、すべての印刷操作を１つの機能と見なすことができる。あるいは、印刷は２つ以上の別々の機能で構成されると考えることもできる（たとえば、ページの印刷と文書全体の印刷）。さらに、以下に詳述するように、同じ基本操作は操作するデータのタイプによって別の機能と見なすこともできる（たとえば、文書を表示するという基本的な操作は、表示するデータがセキュリティで保護されているかいないかによって２つの別々の機能のいずれかと見なすことができる）。このように、１つは何らかのセキュリティで保護され、もう１つは保護されない２つの別々の表示機能が存在してもよい。プログラムをその構成要素となる機能に分解する作業は、実質的にはプログラムが実行するさまざまな処理（および／またはプログラムで処理するさまざまなデータ）の境界を定める作業であり、一般に、こうした境界をどう定めるかについては特定の要件がない。
【００２７】
１つの例では、機能２０２（１）～２０２（ｎ）が第１の「パーティション」２０６（１）の要素、機能２０２（ｎ＋１）～２０２（ｎ＋ｍ）が第２のパーティション２０６（２）の要素のように機能をグループ化できる。このように機能をグループ化すると、同じパーティション内の機能を同様に処理できるので便利である。たとえば、パーティション２０６（１）にはアプリケーション１３５のセキュリティで保護されない通常のデータに関する機能を含め、パーティション２０６（２）にはアプリケーション１３５のセキュリティで保護された機密のデータ（あるいは特定の保護レベルを必要とするデータ）に関する機能を含めることができる。このように、パーティション２０６（１）の機能は一般のオープンな環境（一般的な商用のオペレーティングシステムによりもたらされる環境など）で実行できるのに対して、パーティション２０６（２）の機能は高保証の環境で実行できる（高保証の環境については以下に詳述する）。
【００２８】
図３に、パーティションを使用してセキュリティで保護されたデータと保護されないデータを１つのアプリケーション１３５で操作する方法の例を示す。図３の例では、アプリケーション１３５が高レベルの保護を必要とする操作３０２（たとえば機密データに関する操作）を実行し、さらに低レベルの保護を必要とする操作または全く保護を必要としない操作３０４（たとえば機密データに関連しない操作）も実行する（この例で、データに関する「操作」にはデータの入出力やデータを使用した計算など、データに関するあらゆる処理を含めてよい）。この例では、アプリケーション１３５の機密データの操作に関連しない機能はパーティション２０６（１）で処理され、機密データ（または何らかの保護を必要とするデータ）の操作に関連する機能はパーティション２０６（２）で処理される。たとえば、アプリケーション１３５がワードプロセッサプログラムの場合は、通常の（保護されない）文書の表示はパーティション２０６（１）で実行され、機密文書の表示はパーティション２０６（２）で実行される。別の例として、アプリケーション１３５が株

取引プログラムの場合は、パーティション２０６（１）にユーザが現在の株価を参照するための機能を含め、パーティション２０６（２）にユーザ認証の後にユーザが株を売買するための機能を含めることができる（この場合、ユーザの認証情報や金融口座の詳細は機密情報に分類される）。

【００２９】

図３はアプリケーションの別々のパーティションを使用して機密のデータとそれ以外のデータを処理できる例を示しているが、図３はパーティショニングされたアプリケーションを実装または使用するための特定のメカニズムに限定されないことは言うまでもない。アプリケーションのパーティショニングをサポートし、複数のパーティションを使用しながらユーザには統合された使い心地を提供するアーキテクチャの例については、以下で図４～８に関連付けながら説明する。

【００３０】

パーティショニングされたアプリケーションの例

前述のように、アプリケーションのさまざまな機能は任意の基準で分割できる。図３の例では、アプリケーションの機能がセキュリティで保護されたデータを処理するか保護されないデータを処理するかによってパーティショニングされている。アプリケーションをパーティショニングする場合、ユーザの使い心地に関しては別々のパーティションが統合されているのが望ましい。図４に、パーティショニングされたアプリケーションのユーザインターフェイスの例を示す（「セキュリティで保護された」データとは、何らかの保護を必要とするデータを意味するが、本発明が特定のタイプの保護に限定されないことは言うまでもない。例として、データが「機密」の意味で保護を必要とする場合はデータを暗号化してもよい。別の例として、データが検証可能すなわち基準となる特定の時点以降にデータが変更されていないことを確認できるという意味で保護を必要とする場合はデータに署名してもよい）。

【００３１】

ユーザインターフェイス４００はワードプロセッサプログラムのインターフェイスである（ワードプロセッサプログラムはアプリケーションの手頃な例であるが、決して唯一の例でないことは言うまでもない）。図４の例で、ユーザインターフェイス４００はワードプロセッサ文書を構成するさまざまなアイテム４０２（１）～４０２（５）を表示する。アイテム４０２（１）は、セキュリティで保護されない通常のテキストである。アイテム４０２（２）は、セキュリティで保護されないグラフィック３０４である。アイテム４０２（３）、４０２（４）、４０２（５）は、セキュリティで保護されたテキストアイテムである。アプリケーションのセキュリティで保護されないパーティションは、セキュリティで保護されたデータの操作に関連しないすべての処理を実行する。この例では、この処理にセキュリティで保護されないアイテム４０２（１）と４０２（２）の表示（および可能なら編集、印刷、保存など）とすべてのアイテム（セキュリティで保護されたアイテムを含む）のレイアウトが含まれる。セキュリティで保護されないパーティションがセキュリティで保護されたアイテムをレイアウトできるのは、アイテム４０２（３）、４０２（４）、４０２（５）のコンテンツは機密であるが、これらのアイテムの存在は機密でないためである。このように、セキュリティで保護されないパーティションはアイテム４０２（３）～４０２（５）を解読できないくねった線４０５として表示することで、少なくともセキュリティで保護されないパーティションでは処理できないコンテンツであることをユーザに認識させることができる。

【００３２】

ユーザがコンテンツアイテム４０２（３）、４０２（４）、４０２（５）を表示したい場合は、たとえば特定のコンテンツアイテムを表現するくねった線４０５をクリックしてもよい。この操作によって、セキュリティで保護されたパーティションを読み出すことができる。これで、セキュリティで保護されたアイテムの１つに含まれるコンテンツがセキュリティで保護されたパーティションに渡されて処理される。１つの例では、次に、セキュリティで保護されないパーティションがレイアウトしたコンテンツの位置に重なるウィ

ンドウに、セキュリティで保護されたパーティションが実際のコンテンツアイテムを表示できる。たとえば、セキュリティで保護されたパーティションがコンテンツアイテム４０２（３）を表示する場合は、このコンテンツアイテム４０２（３）を表現するくねった線４０５がすでに表示されている領域の一番上にこのコンテンツアイテムのイメージを配置する。前述のケースにはユーザとソフトウェアの２つの別々の部分（セキュリティで保護されたパーティションと保護されないパーティション）との対話が必要であるが、ユーザは１つのアプリケーションと対話しているように見えるので、２つのパーティションを使用してもユーザには一貫した使い心地が提供される。

【００３３】

図４に、パーティショニングされたアプリケーションがワードプロセッサプログラムの場合の例を示す。他にも次のような例がある。

【００３４】

スプレッドシート。ユーザはセキュリティで保護されないパーティションに式を入力し、表を作成できるが、実際に処理するデータはセキュリティで保護されたパーティションのみで利用できる。したがって、セキュリティで保護されないパーティションでは各セルに特定のプレースホルダー（たとえば“ｘｘｘｘ”）が表示される。セル内のデータの評価と表示は、アプリケーションのセキュリティで保護されたパーティションで実行される。ユーザが「計算（ｃａｌｃｕｌａｔｅ）」ボタンを押すと、ウィンドウ（セキュリティで保護されたパーティションで生成された）がポップアップし、該当する行と列のセルの値が表示される。この値は、セキュリティで保護されたパーティションが利用できる基盤となるデータに基づいて計算される。この実装では、セキュリティで保護されたパーティションが評価エンジンと（簡単な）表示ルーチンを備えているが、スプレッドシートのユーザインターフェイス、ヘルプウィンドウ、数式ビルダーなどの他の機能を組み込む必要はない。

【００３５】

証券取引アプリケーション。セキュリティで保護されたパーティションと保護されないパーティションのそれぞれに機能が割り当てられている。たとえば、取引のグラフや株式情報（公的に利用可能）の表示はセキュリティで保護されないパーティションで実行できる。一方、セキュリティで保護されたパーティションは、ユーザが株取引のシンボルや売買する株式の価格と数量を入力するために使用する。さらに、リモート取引サーバーのＩＤを検証してから、このサーバーに前述の情報を送信する。

【００３６】

画像形式の文書を出力するワードプロセッサ（たとえば、Ｐｏｒｔａｂｌｅ　Ｄｏｃｕｍｅｎｔ　Ｆｏｒｍａｔすなわち“ＰＤＦ”）。セキュリティで保護されたパーティションでは、画像の読み出しと表示が可能である。こうした文書の表示方法を文書の「複写（ｆａｃｓｉｍｉｌｅ）」と呼ぶ。基盤となるワードプロセッサ文書（すなわち、画像だけでなく書式コードとテキスト文字列を含む文書）とその複写がセキュリティで保護されないパーティションからセキュリティで保護されたパーティションに送信され、セキュリティで保護されたパーティションが複写を表示する。ユーザは、セキュリティで保護された環境でセキュリティで保護された署名エージェントを使用して、複写（または文書の要約と複写）に署名する。署名された複写（“ｆａｃｓｉｍｉｌｅ－０”）と基盤となる文書は、セキュリティで保護されないパーティションにブロブ（ｂｌｏｂ）として返される。検証する側が文書ｆａｃｓｉｍｉｌｅ－０と複写の署名を含むブロブを受け取ると、検証側はまず基盤となるワードプロセッサ文書に基づく複写の新しいコピー（ｆａｃｓｉｍｉｌｅ－１）を表示する。検証側は、ｆａｃｓｉｍｉｌｅ－０、ｆａｃｓｉｍｉｌｅ－１、Ｗｏｒｄ文書および署名をセキュリティで保護されたパーティションに送信する。これで、２つの複写（ｆａｃｓｉｍｉｌｅ－０とｆａｃｓｉｍｉｌｅ－１）が同等であり、ｆａｃｓｉｍｉｌｅ－０の署名が有効であることを検証する。このモデルでは文書の機密性は保持しないが（文書のｆａｃｓｉｍｉｌｅ－１はセキュリティで保護されないパーティションでも表示できるので）、文書の整合性、すなわち文書が初めに作成されたものであり

、変更されていないという事実を検証することはできる。

【００３７】

図５に、アプリケーションのパーティショニングをサポートするアーキテクチャを示す。図５には、ソフトウェアを実行できる２つの環境５０２（１）と５０２（２）がある。この例では、４つの別々のプロセッサ５０４（１）、５０４（２）、５０４（３）、５０４（４）がある。この場合、「プロセッサ」はマイクロプロセッサでなくアプリケーションのデータを何らかの方法で処理するソフトウェアコンポーネントを意味する。プロセッサ５０４（１）と５０４（２）は環境５０２（１）で動作し、５０４（３）と５０４（４）は環境５０２（２）で動作する。たとえば、プロセッサ５０４（１）と５０４（３）はそれぞれ１つのワードプロセッサアプリケーションのセキュリティで保護されない部分と保護された部分であってもよい。ベースコンポーネント５０８は環境５０２（１）と５０２（２）をホスティングするので、１台のマシン上に２つの環境が共存できる。本発明は、特定のタイプのベースコンポーネント５０８には限定されない。ただし、ベースコンポーネント５０８の例には、仮想マシンモニタ（ＶＭＭ：ｖｉｒｔｕａｌ　ｍａｃｈｉｎｅ　ｍｏｎｉｔｏｒ）、ｅｘｏｋｅｒｎｅｌ、ｍｉｃｒｏｋｅｒｎｅｌ、ｈｙｐｅｒｖｉｓｏｒがある。たとえば、環境５０２（１）と５０２（２）はＶＭＭまたはｈｙｐｅｒｖｉｓｏｒでホスティングされた別々のオペレーティングシステムであってもよい。別の例では、ベースコンポーネント５０８が一部のユーザ機能を提供し、さらに自らともう１つの（低保証）オペレーティングシステムとの分離を実現する高保証オペレーティングシステムのようなオペレーティングシステムであってもよい。

【００３８】

ベースコンポーネント５０８の一部として動作する１つのコンポーネントは、参照モニタ（ｒｅｆｅｒｅｎｃｅ　ｍｏｎｉｔｏｒ）５１０である。参照モニタ５１０は、特定のデータオブジェクトを適切な処理環境にルーティングする役割を果たす。たとえば、アプリケーション実行中にラベル付きデータ５０６が検出（または生成あるいは入力）されると、参照モニタ５１０によってラベル付きデータ５０６が適切な環境にルーティングされる。ラベル付きデータ５０６には識別タグが関連付けられており、参照モニタ５１０はこのタグを使用してラベル付きデータをどの環境にルーティングするかを判断できる。さらに、ラベル付きデータ５０６にはこのデータをどのプロセッサで処理するかを判断するための目的環境を可能とする別のタグを追加してもよい。ラベル付きデータ５０６の構造の例については、図７に関連付けながら以下で説明する。

【００３９】

ラベル付きデータ５０６は任意の場所で検出（または生成あるいは入力）されてよいが、１つの環境で検出されたラベル付きデータがもう１つの環境に渡されるのが最も一般的である。たとえば、ワードプロセッサアプリケーションのセキュリティで保護されない部分で、何らかの機密のテキストを含むラベル付きデータが検出されてもよい。ここで、ワードプロセッサアプリケーションはこのテキストの表現を図４に示すような方法で表示できる（たとえば、解読できないくねった線）。ユーザが実際のテキストを表示するように要求すると（たとえば、このデータが表示される領域をクリックすると）、アプリケーションのセキュリティで保護されないパーティション（たとえば、環境５０２（１）で動作するプロセッサ５０４（１））はラベル付きデータ５０６を参照モニタ５１０に渡し、さらに参照モニタ５１０がこのデータ５０６を環境５０２（２）に渡す。これで、環境５０２（２）は適切なプロセッサ（たとえば、プロセッサ５０４（３））にデータをルーティングでき、さらにこのプロセッサが図４に関連付けて上に説明した方法で画面上の適切な位置にデータを表示できる。

【００４０】

図５のモデルにおいて、アプリケーションは基本的に複数のプロセッサ（すなわち、特定のタイプのデータを操作するコンポーネント、または特定のカテゴリのタスクを実行するコンポーネントなど）で構成されており、１つのアプリケーションのさまざまなパーティションでさまざまなプロセッサを使用することは言うまでもない。たとえば、プロセッ

サ５０４（１）と５０４（３）はそれぞれ１つのアプリケーション（たとえばワードプロセッサアプリケーション）のセキュリティで保護されたプロセッサと保護されないプロセッサであってもよい。この場合、データはそれ自体がセキュリティで保護されているかどうかによってプロセッサ５０４（１）と５０４（３）のいずれかにルーティングされる。

【００４１】

どのような環境を利用できるかに関する要件はないが、１つの環境が高保証のオペレーティングシステムであり、もう１つの環境が一般的なフルサービスのオープンなオペレーティングシステムであれば特に有効であることも言うまでもない。「高保証」オペレーティングシステムは、想定された機能を正しく実行するという点で比較的高レベルの信頼性を提供するシステムである。すべてのソフトウェア（オペレーティングシステムを含む）がそのソフトウェアの想定された機能を説明する仕様に関連付けられており、すべてのソフトウェアがそのソフトウェアの予期しない挙動の原因となる特定のレベルのバグまたは攻撃を免れないとすれば、「高保証」のオペレーティングシステムはその仕様にしたがって挙動するという比較的高レベルの信頼性を提供するシステムである（市販のほとんどのソフトウェアには、明確な仕様書が添付されているが、この場合の仕様はソフトウェアの挙動を表すものであり、書面によらない暗黙的なものでもよい）。高保証はセキュリティと同等ではないが、高保証を使用してセキュリティを提供することはできる。たとえば、機密データを処理するプロセッサが高保証のオペレーティングシステム上で動作するように設計でき、このプロセッサが機密データをどの程度保護できるかは、実行する環境の挙動が正しいこと（たとえば、システムコールの適切な実行、プロセスの適切な分離など）に基づいており、プロセッサが一般的なフルサービスのオペレーティングシステム上で実行された場合には実現できないレベルのセキュリティを提供できる（高保証の環境では、高保証の代償として機能の範囲が非常に限定される。プログラムが大きくなるほど、さまざまな状況でプログラムが正しく動作することを検証するのは困難になる。したがって、ＷＩＮＤＯＷＳ（登録商標）ＸＰのような商用のフルサービスのオペレーティングシステムは、一般に高保証とは見なされない）。

【００４２】

前述の高保証のケースでは、アプリケーションを特定のセキュリティレベルを必要としないデータを操作する低セキュリティのプロセッサと高セキュリティを必要とするデータを操作する高セキュリティのプロセッサにパーティショニングするのがしばしば有効であることは言うまでもない。高セキュリティのプロセッサは高保証の環境で実行でき、低セキュリティのプロセッサは低保証の環境で実行できる。

【００４３】

パーティショニングされたアプリケーションをサポートするアーキテクチャの例

図６に、パーティショニングされたアプリケーションを実行できるアーキテクチャの例６００を示す。アーキテクチャ６００にはベースコンポーネント５０８が配置されている。ベースコンポーネント５０８の機能は、アプリケーションの複数のパーティションが動作する別々の環境をホスティングすることである。ベースコンポーネント５０８は、前述のように、ＶＭＭ、ｅｘｏｋｅｒｎｅｌ、ｍｉｃｒｏｋｅｒｎｅｌ、ｈｙｐｅｒｖｉｓｏｒなど、さまざまな形態をとることができる。ベースコンポーネント５０８は、複数の環境（たとえば、オペレーティングシステム）のホストとして機能する以外に、こうした環境間の対話の管理（および制限）も行う。複数の環境のホスティングは、さまざまな技術を使用して実行できる。たとえば、ベースコンポーネント５０８は、複数のオペレーティングシステムのそれぞれに対して、独立した「仮想マシン」を提供してもよい。これで、各オペレーティングシステムは仮想マシンの「仮想ハードウェア」を制御し、オペレーティングシステムが仮想マシンに対して発行した命令に基づいて（オペレーティングシステムの命令と必ずしも同じではない）、ベースコンポーネント５０８が「実際の」ハードウェアに対して命令を発行する（一般的に言えば、これは従来のＶＭＭの機能である）。別の例として、ベースコンポーネント５０８はそれぞれのオペレーティングシステムに特定のデバイスとマシンの物理アドレス領域の一部分を割り当て、各オペレーティングシステ

ムに対してそのシステムに割り当てられたデバイスとそのシステムに割り当てられたアドレス領域の部分のみの制御を許可することで割り当てを実施してもよい。本発明は、ベースコンポーネントの特定の実施形態には限定されない。図６を実現するために、複数の環境が１台のマシン上に互いにある程度分離した形で共存できるように、こうした複数の環境をホスティングできるベースコンポーネントが存在することを想定しているにすぎない。

【００４４】

図６の例では、ベースコンポーネント５０８がオペレーティングシステム６０２（１）～６０２（４）をホスティングしている。オペレーティングシステム６０２（１）はＷＩＮＤＯＷＳ（登録商標）ＸＰオペレーティングシステムなどの汎用オペレーティングシステムのインスタンスであり、オペレーティングシステム６０２（２）はＬｉｎｕｘオペレーティングシステムのインスタンスであり、オペレーティングシステム６０２（３）は“ｎｅｘｕｓ”すなわち機能は限定されるがこの機能を正しく実行することが保証される高保証のオペレーティングシステムであり（前述の意味の範囲内で）、さらにオペレーティングシステム６０２（４）はＯＳ／２のような別の汎用オペレーティングシステムであってもよい。一般に、ベースコンポーネント５０８は任意の数のオペレーティングシステム（または他のタイプの環境）をホスティングでき、図６には例としてのみ４つのオペレーティングシステムを示している。

【００４５】

特定のオペレーティングシステムの下では、アプリケーションレベルのソフトウェアを実行できる。たとえば、ワードプロセッサプログラムＭＩＣＲＯＳＯＦＴ　ＷＯＲＤ（６０４）とスプレッドシートプログラムＭＩＣＲＯＳＯＦＴ　ＥＸＣＥＬ（６０６）は、ＷＩＮＤＯＷＳ（登録商標）ＸＰオペレーティングシステム６０２（１）の下で動作している。Ｌｉｎｕｘオペレーティングシステム６０２（２）、ｎｅｘｕｓ６０２（３）、ＯＳ／２オペレーティングシステム６０２（４）でも、それぞれ独自のアプリケーションプログラムを実行できる。この例では、“Ｗｏｒｄ－ｌｅｔ”（６０８）および“Ｅｘｃｅｌ－ｌｅｔ”（６１０）というプログラムがｎｅｘｕｓ６０２（３）の下で動作している。Ｗｏｒｄ－ｌｅｔ６０８は、ＷＯＲＤ６０４がセキュリティで保護されたデータを操作する必要がある場合にＷＯＲＤと連動するセキュリティで保護されたプログラムである。同様に、Ｅｘｃｅｌ－ｌｅｔ６１０はＥＸＣＥＬ６０６のセキュリティで保護されたデータを操作する。したがって、図４の例では、ＷＯＲＤ６０４はセキュリティで保護されないデータアイテムを操作し、ウィンドウ内のデータアイテムをレイアウトし、セキュリティで保護されたアイテムを表現する解読できないくねった線を表示するプログラム（または「プロセッサ」）であってもよい。Ｗｏｒｄ－ｌｅｔ６０８は、セキュリティで保護されたアイテムを開き、ＷＯＲＤ６０４で予約したウィンドウ内の領域にこのアイテムを表示するプログラであってもよい。実質的に、ＷＯＲＤ６０４とＷｏｒｄ－ｌｅｔ６０８はともに別々の環境で動作する別々のプロセッサを使用する１つのワードプロセッサアプリケーションの機能上のパーティション（または「ファクタリング」）を表している。同様に、ＥＸＣＥＬ６０６とＥｘｃｅｌ－ｌｅｔ６１０もよく似た関係にある。つまり、ＥＸＣＥＬ６０６はセキュリティで保護されたデータに関連しないスプレッドシートのほとんどの機能を操作し、Ｅｘｃｅｌ－ｌｅｔ６１０はＥＸＣＥＬに代わってセキュリティで保護されたデータを操作する。

【００４６】

前述のように、ｎｅｘｕｓ６０２（３）は保証できるアプリケーション（すなわち、機密データを開くような機密の操作に関してその挙動が十分に保証できるアプリケーション）を実行できる環境を提供する高保証のオペレーティングシステムである。ただし、多くの場合、ｎｅｘｕｓ６０２（３）が提供する機能は非常に限られている。したがって、ワードプロセッサアプリケーションをたとえばＷＯＲＤ６０４とＷｏｒｄ－ｌｅｔ６０８にパーティショニングすることで、ＷＯＲＤは汎用オペレーティングシステムで実行できるさまざまな機能を使用して広範な機能性を提供し、Ｗｏｒｄ－ｌｅｔはｎｅｘｕｓ６０２

（３）が提供する環境の高信頼性を使用して高レベルの信頼性を伴う機密性の高い（多くの場合に限定的な）機能群を実行できるのは有効である。

【００４７】

図６に、アプリケーションのパーティショニングをサポートするインフラストラクチャを提供するベースコンポーネントを示す。以下は、このインフラストラクチャが提供するいくつかの機能の説明である。

【００４８】

登録およびディレクトリサービス。ベースコンポーネントは、ベースコンポーネントでホスティングされる環境がベースコンポーネントに登録するためのインターフェイスを提供できる。ベースコンポーネントは、ベースコンポーネントでホスティングされる環境が起動する方法も提供できる（あるいは、ベースコンポーネントがホスティングされる環境の１つであってもよい）。環境の保証レベルはその環境に関連するメタ情報であり、ベースコンポーネントは体系的な方法でこの情報にアクセスし、この情報を参照するためのオプションのサービスを提供できる。

【００４９】

分離。ホスティング対象の環境には、ベースコンポーネントによってセキュリティコンテキストが割り当てられる。環境のセキュリティコンテキストには、次のようなコンポーネントを指定できる。

ａ．　ＤＡＣコンテキスト。たとえば、環境のコードＩＤ。

ｂ．　ＭＡＣコンテキスト。ＭＡＣの場合、環境のＭＡＣコンテキストは機密性のラベルとカテゴリで構成される。

【００５０】

通信。環境間の通信は、ベースコンポーネントの単方向トランスポートを使用してベースコンポーネントが制御する。

ａ．　トランスポートによって、環境間で順序正しい保証できるメッセージ配信が可能になる。同期化オブジェクトが提供され、トランスポートへのデータの到着を環境に通知する。

ｂ．　アクセス制御。トランスポートはベースコンポーネントが所有するオブジェクトであり、ホスティング対象の環境からトランスポートへの読み出しまたは書き込みの機能は、ベースコンポーネントが実施するアクセス制御モデルによって制御される。ＤＡＣの場合、これはトランスポートの「読み出し」と「書き込み」の操作に関するＡＣＬによって制御される。ＭＡＣの場合、これはトランスポートに添付されたラベルによって制御される。複数レベルのデバイスにエクスポートを提供するために、ベースコンポーネントはフロントエンドのセキュリティフィルタを実装する。フロントエンドのセキュリティフィルタ（ＦＥＳＦ：Ｆｒｏｎｔ－ｅｎｄ　ｓｅｃｕｒｉｔｙ　ｆｉｌｔｅｒ）はベースコンポーネントに登録された機能であり、トランスポートの書き込みまたは読み出しのエンドに添付できる。書き込み（読み出し）ＦＥＳＦの場合は、ＶＭがデータをトランスポートに書き込む（から読み出す）前に、ＶＭＭから書き込み（読み出し）ＦＥＳＦが呼び出される。書き込みＦＥＳＦの例はＳｅａｌ（）であり、ＭＡＣが有効な場合は、一般に高保証の環境のデータが通常の（高保証でない）環境に書き込まれる前にこのデータに適用される。ＭＡＣが有効でない場合は、特定の環境で機密と見なされるデータを別の環境に送信する前に、このデータを暗号化する必要がある。

【００５１】

メッセージ。トランスポートは、さまざまな環境間でメッセージを搬送する。メッセージはベースコンポーネントによって作成されるデータのブロブであり、次のような構造をとる。

ａ．　ベースコンポーネントによって割り当てられたセキュリティコンテキスト。ＤＡＣコンテキストとＭＡＣコンテキストが含まれる。ＤＡＣコンテキスには、作成する環境のサブジェクトＩＤが含まれる。ＭＡＣコンテキスには、作成する環境の機密性ラベルとカテゴリが含まれる。

ｂ．　コンテンツ。データ（作成する環境よって割り当てられたセキュリティコンテキストを含む）。

【００５２】

高レベルの抽象化。環境（またはアプリケーションのように環境内で動作するソフトウェアオブジェクト）は、ベースコンポーネントが提供する通信抽象化を使用して独自にＲＰＣインターフェイスを実装できる。これは、環境が提供するサービスでも、単にライブラリコードでもよい。もう１つの提供可能な抽象化は、環境間のソケットコールインターフェイス（ｓｏｃｋｅｔ　ｃａｌｌ　ｉｎｔｅｒｆａｃｅ）である。これは、ベースコンポーネントで実装する必要はない。

【００５３】

フォーカス管理。パーティショニングされたアプリケーションの多くでは、１つの環境から別の環境にメッセージが到着すると、セキュリティで保護された入力装置または出力装置の現在の所有元が変わる場合がある。フォーカス管理は、環境からの要求に応じてベースコンポーネントが提供する。つまり、１つの環境は別の環境からメッセージが到着した時点でフォーカスの変更を要求できる。次に、環境はセキュリティで保護された入力装置または出力装置を使用したセッションを実行してから制御を放棄する。ベースコンポーネントは、特定の環境によるＩ／Ｏセッションを強制的に終了できる。

【００５４】

パーティショニングされたアプリケーションで使用するデータオブジェクトの例

図７に、パーティショニングされたアプリケーションでデータオブジェクトを使用できるようにするデータオブジェクトの構造の例を示す。前述のように、アプリケーションがパーティショニングされている場合は、アプリケーションの第１のパーティションが、第１のパーティションでは処理できないので第２のパーティションに送信する必要のあるデータオブジェクトを検出する可能性がある。アプリケーションの第１のパーティションが「セキュリティで保護されない」部分で、アプリケーションの第２のパーティションが「セキュリティで保護された」部分の場合、データオブジェクトを処理するには他の場所に送信する必要があることを第１のパーティションが認識できるのが好ましい。第１のパーティションは、データオブジェクトについてそれ以外は一切判断できなくてもよい（たとえば、第１のパーティションはデータオブジェクトのコンテンツを読み出すことができなくてもよい）。さらに、一部のケースではデータオブジェクトが実際にセキュリティで保護された状況で作成され、それ以降に変更されていないことを検証する（たとえば、キーボードからアプリケーションのＩ／Ｏストリームに至るまでにデータのスプーフィングを防止された環境でデータが入力されたかどうかを確認できる）のが重要である。このように、データオブジェクトはその一連の操作を検証できる形で表現されるのも好ましい。

【００５５】

データオブジェクト７００にはデータアイテム７０２が含まれる。データアイテム７０２は、たとえば機密のワードプロセッサ文書（またはその一部）、スプレッドシートの機密の財務データ、財務トランザクションでユーザを認証するためのパスワード情報など、基盤となる実質的なコンテンツであり、これを搬送するためにデータオブジェクト７００が存在する。

【００５６】

データアイテム７０２は、データを操作するアーキテクチャのレイヤごとに、一連のラッパーでラッピングされている。各ラッパーの役割は、（１）アイテムをラッパーでラッピングしたエンティティ（アプリケーション、環境など）をラッパーが識別すること（後でアイテムをこのエンティティにルーティングするときに使用）と、（２）アイテムがラッピングされて以降に変更されていないことを後で検証できるように、ラッパーにアイテムのシールを添付することである。たとえば、Ｗｏｒｄ－ｌｅｔ６０８がデータアイテム７０２を作成する場合、Ｗｏｒｄ－ｌｅｔはデータアイテム７０２のデジタル署名と、後で何らかの操作が必要なときにデータアイテム７０２をルーティングする宛先のプロセッサとしてＷｏｒｄ－ｌｅｔを指定するヘッダの両方を添付できる。したがって、署名とヘ

ッダはラッパー７０４の構成要素である。ただし、ラッパーは署名とヘッダにも、いかなる特定の実施形態にも限定されないこと、さらにアイテムを識別できるようにするため、およびアイテムの整合性（すなわち変更されていないこと）を検証できるようにするためのさまざまな既知の技術が存在することに留意されたい。

【００５７】

前述のように、Ｗｏｒｄ－ｌｅｔのようなプロセッサのセキュリティは、保証できるコンポーネントの階層に基づいている。つまり、Ｗｏｒｄ－ｌｅｔが保証できるのは、Ｗｏｒｄ－ｌｅｔがｎｅｘｕｓの高信頼性に基づいているためであり、ｎｅｘｕｓが保証できるのは、ｎｅｘｕｓがベースコンポーネントの分離機能に基づいているためである。このように、データアイテム７０２を作成するアプリケーションに至る連鎖にある階層内の各コンポーネントは、それぞれ独自のラッパーでアイテムをラッピングする。データアイテム７０２はすでにＷｏｒｄ－ｌｅｔのラッパーでラッピングされているが、さらにｎｅｘｕｓのラッパー７０６でラッピングされる。この（二重にラッピングされた）アイテムは、さらにベースコンポーネントのラッパー７０８でラッピングされる。各ラッパーは、基本的にラッピングしたコンポーネントを特定し、さらにラッピングされたコンポーネントがラッピングされた後に変更されていないことを検証できる証明（ラッピングしたコンポーネントが保証できる範囲まで）を含んでいる。

【００５８】

さらに、ラッパーは階層の順に従ってネスティングされているので、ルーティングの際に各ラッパーを利用できる。アプリケーションがデータオブジェクト７００を検出すると、外側のラッパー７０８がオブジェクトを別のプロセッサにルーティングする必要があるものとして特定し、オブジェクトがベースコンポーネントに送信される。ベースコンポーネントはこのラッパーを使用してラッパー内部のコンテンツの整合性を検証し、次のレベルのラッパー７０６を使用して必要なルーティング先の環境を特定する（この場合はｎｅｘｕｓ）。次に、ｎｅｘｕｓがラッパー７０６内部のコンテンツの整合性を検証し、さらにラッパー７０４を使用してコンテンツを操作するソフトウェアオブジェクトを特定する（この場合はＷｏｒｄ－ｌｅｔ）。さらに、Ｗｏｒｄ－ｌｅｔがラッパー７０４内部のコンテンツを検証する。このコンテンツはデータアイテム７０２である。したがって、データオブジェクト７００の構造によって、データアイテム７０２を操作するプロセッサに至る連鎖の各ステップでデータアイテム７０２をルーティングでき、検証できる。

【００５９】

パーティショニングされたアプリケーションを使用するプロセスの例

図８に、パーティショニングされたアプリケーションを使用するプロセスの例を示す。図８の例では、アプリケーションがプロセッサ１およびプロセッサ２という２つのプロセッサを含むものとする。

【００６０】

まずプロセッサ１が動作しており、プロセッサ１が動作している間にプロセッサ１がデータオブジェクト（８０２）を検出する。前述のように、プロセッサ１はデータオブジェクトを読み出すことはできないが、このデータオブジェクトが他の場所に送信して処理する必要のあるものであることは認識できる。そこで、プロセッサ１はデータオブジェクトを参照モニタ（８０４）に送信する。

【００６１】

参照モニタは、データオブジェクトの一番外側のラッパーを使用してデータオブジェクトの送信先を特定する。前述のように、さらに参照モニタ（前述のベースコンポーネントの一部であるのが好ましい）は一番外側のラッパーを使用してラッパーのコンテンツの整合性を検証する。この例では、コンテンツの整合性が保持されていることと、参照モニタがデータオブジェクトをプロセッサ２（８０６）で処理する必要があると判断することが想定されている（前述のように、参照モニタはプロセッサ２の存在を直接認識はできないが、ラッパーを使用してオブジェクトを特定の環境にルーティングし、ここで受け取った環境によってオブジェクトがさらにプロセッサ２にルーティングされる。このようにネス

ティングされたラッパーの使用については図７に関連して説明したので、単に図８の例を簡素化するために、データオブジェクトがプロセッサ２に至るまでの中間のルーティングステップを無視して、データオブジェクトをプロセッサ２にルーティングできるものとする）。

【００６２】

データオブジェクトがプロセッサ２で処理されるように設計されていることが判明したら、プロセッサ２が動作しているかどうかを確認する（８０８）。たとえば、この確認はプロセッサ２が動作するように指定された環境（前述の例ではｎｅｘｕｓ）で行うことができる。プロセッサ２が動作していない場合は、プロセッサ２が起動する（８１０）。プロセッサ２が起動すると（あるいはすでに動作していることが確認されると）、プロセッサ２が入力または出力を操作する場合はプロセッサ２がフォーカスされる（つまり、入力装置と出力装置でＩ／Ｏを実行できる）。

【００６３】

ただし、以上の例は単に説明のために示されており、本発明に関する限定と理解してはならない。本発明をさまざまな実施形態に関連して説明してきたが、ここに記載した内容は説明を目的とするものであり、限定を意図するものでないことは言うまでもない。さらに、本発明を特定の手段、素材、および実施形態に関連して説明してきたが、本発明をここに開示する仕様に限定する意図はない。本発明は、前述の特許請求の範囲に示す、機能上同等のあらゆる構造、方法、使い方に拡張される。この明細書に記載する技術を利用する当業者は、これにさまざまな変更を加えることができる。こうした変更は、本発明の機能において本発明の範囲と精神を逸脱することなく実施できる。

【図面の簡単な説明】

【００６４】

【図１】本発明の機能を実現できる例示的なコンピューティング環境を示すブロック図である。

【図２】機能上の構成要素にファクタリングされたアプリケーションを示すブロック図である。

【図３】１つのアプリケーションを構成する別々のコンポーネントにデータをルーティングする方法を示すブロック図である。

【図４】ファクタリング可能なアプリケーションのユーザインターフェイスの例を示すブロック図である。

【図５】ファクタリングされたアプリケーションの使用をサポートするアーキテクチャの例を示すブロック図である。

【図６】ファクタリングされたアプリケーションを使用できる環境の階層の例を示すブロック図である。

【図７】ファクタリングされたアプリケーションの使用をサポートする環境で使用するデータオブジェクトの例を示すブロック図である。

【図８】ファクタリングされたアプリケーションでデータを処理できるプロセスの例を示す流れ図である。

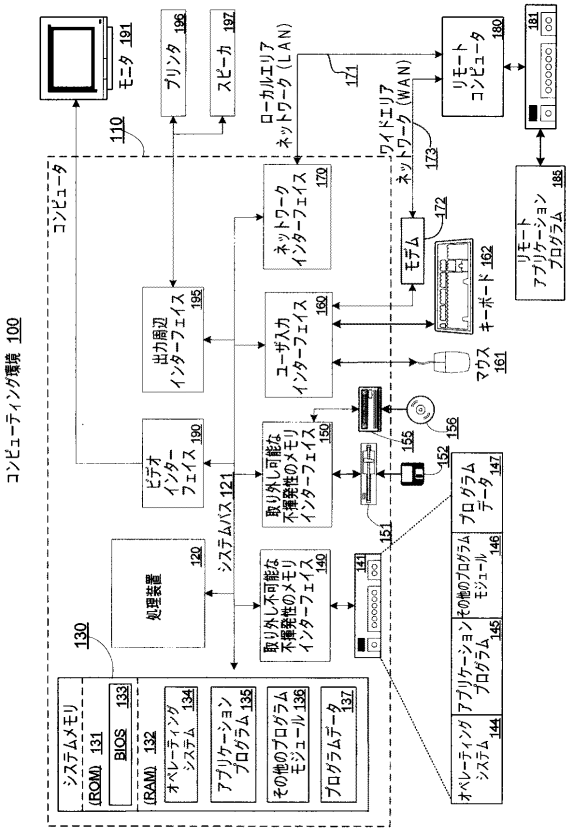
【符号の説明】

【００６５】

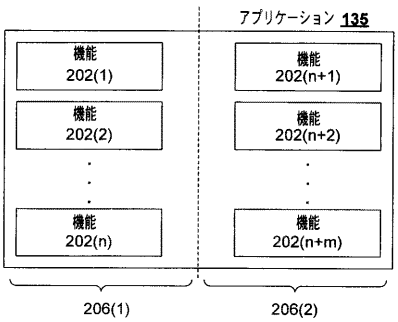
１００　コンピューティング環境
１１０　コンピュータ
１２０　処理装置
１２１　システムバス
１３０　システムメモリ
１３１　ＲＯＭ
１３２　ＲＡＭ
１３３　ＢＩＯＳ
１３４　オペレーティングシステム

１３５　アプリケーションプログラム
１３６　その他のプログラムモジュール
１３７　プログラムデータ
１４０　取り外し不可能な不揮発性のメモリインターフェイス
１４４　オペレーティングシステム
１４５　アプリケーションプログラム
１４６　その他のプログラムモジュール
１４７　プログラムデータ
１５０　取り外し可能な不揮発性のメモリインターフェイス
１６０　ユーザ入力インターフェイス
１６１　マウス
１６２　キーボード
１７０　ネットワークインターフェイス
１７１　ローカルエリアネットワーク（ＬＡＮ）
１７２　モデム
１７３　ワイドエリアネットワーク（ＷＡＮ）
１８０　リモートコンピュータ
１８５　リモートアプリケーションプログラム
１９０　ビデオインターフェイス
１９１　モニタ
１９５　出力周辺インターフェイス
１９６　プリンタ
１９７　スピーカ

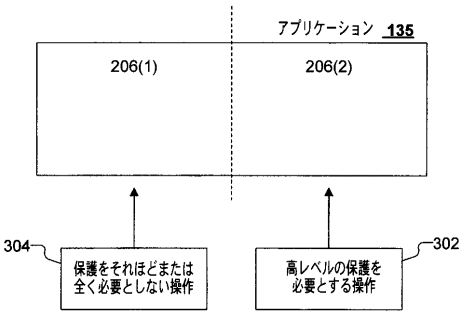
【図１】



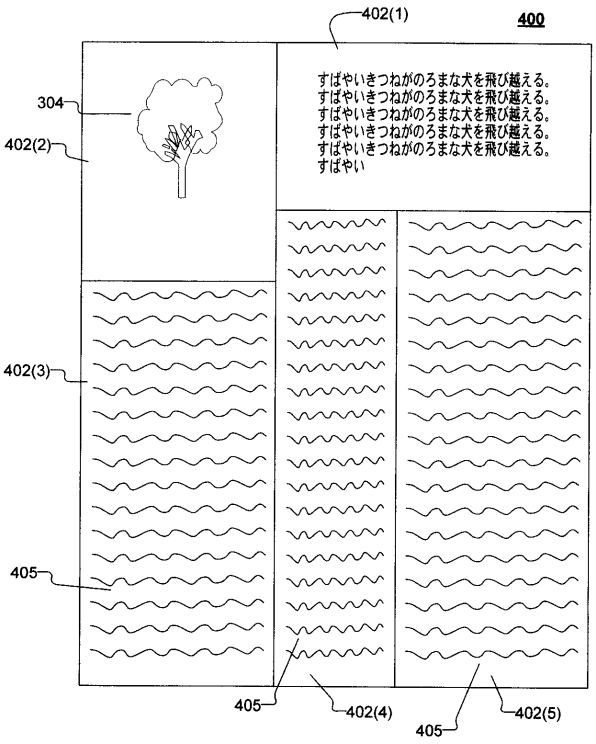
【図２】



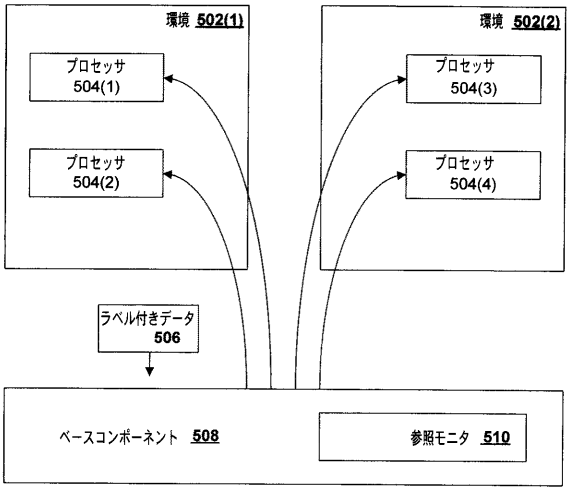
【図３】



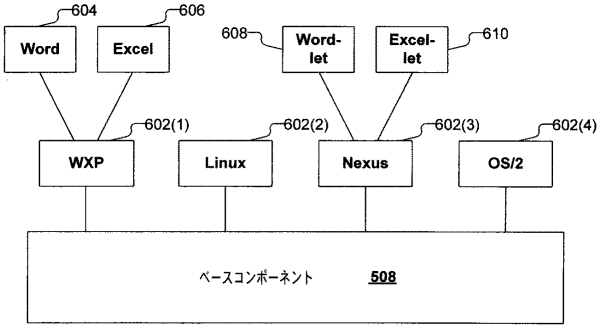
【図４】


402(1)
400
すばやいきつねがのろまな犬を飛び越える。
すばやいきつねがのろまな犬を飛び越える。
すばやいきつねがのろまな犬を飛び越える。
すばやいきつねがのろまな犬を飛び越える。
すばやいきつねがのろまな犬を飛び越える。
すばやい
304
402(2)
402(3)
405
405
402(4)
405
402(5)

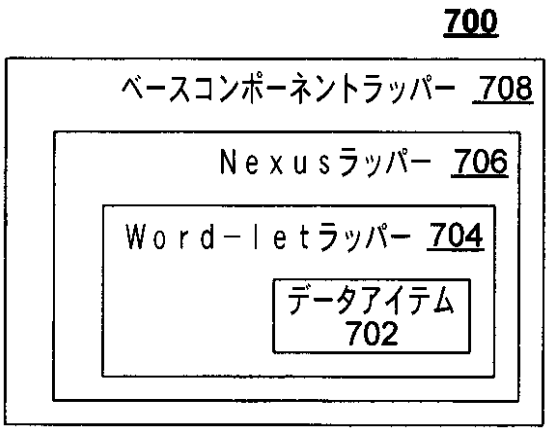
【図５】


環境 502(1)
環境 502(2)
プロセッサ 504(1)
プロセッサ 504(2)
プロセッサ 504(3)
プロセッサ 504(4)
ラベル付きデータ 506
ベースコンポーネント 508
参照モニタ 510

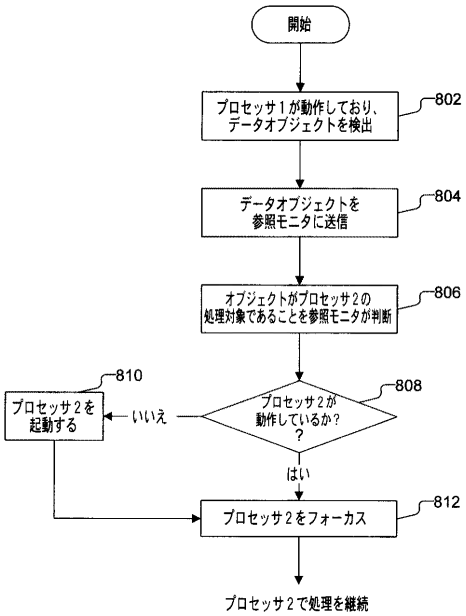
【図６】


604
606
608
610
Word
Excel
Word-let
Excel-let
602(1)
602(2)
602(3)
602(4)
WXP
Linux
Nexus
OS/2
ベースコンポーネント 508

【図７】


700
ベースコンポーネントラッパー 708
Nexusラッパー 706
Word-letラッパー 704
データアイテム 702

【図８】


開始
プロセッサ１が動作しており、データオブジェクトを検出
802
データオブジェクトを参照モニタに送信
804
オブジェクトがプロセッサ２の処理対象であることを参照モニタが判断
806
プロセッサ２が動作しているか？
808
いいえ
プロセッサ２を起動する
810
はい
プロセッサ２をフォーカス
812
プロセッサ２で処理を継続

フロントページの続き

(72)発明者 マルクス ペイナード
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション内
(72)発明者 ポール イングランド
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション内
(72)発明者 テクサラカル バルギス クリエン
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション内

審査官 市川 武宜

(56)参考文献 米国特許第０６４９０７２０（ＵＳ，Ｂ１）
米国特許第０６００６３３２（ＵＳ，Ａ）
米国特許出願公開第２００３／０１０７５８４（ＵＳ，Ａ１）
Tal Garfinkel,Ben Pfaff ,Jim Chow ,Mendel Rosenblum ,Dan Boneh, Terra:A Virtual Machine-Based Platform for Trusted Computing, Proceedings of the 19th ACM Symposium on Operating Systems Principles, 米国, ACM , ２００３年１０月１９日, Volume 37, Number 5, pages 193-206