

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2004-265402

(P2004-265402A)

(43) 公開日 平成16年9月24日(2004.9.24)

(51) Int.Cl.⁷

G06F 17/24

F I

G06F 17/24 554H

テーマコード (参考)

5B009

審査請求 未請求 請求項の数 41 O L 外国語出願 (全 70 頁)

(21) 出願番号 特願2004-42140 (P2004-42140)
 (22) 出願日 平成16年2月18日 (2004.2.18)
 (31) 優先権主張番号 10/377,258
 (32) 優先日 平成15年2月28日 (2003.2.28)
 (33) 優先権主張国 米国 (US)

(71) 出願人 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 9805
 2-6399 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 ブライアン マイケル ジョーンズ
 アメリカ合衆国 98052 ワシントン
 州 レッドモンド ノースイースト 18
 1 アベニュー 10630

最終頁に続く

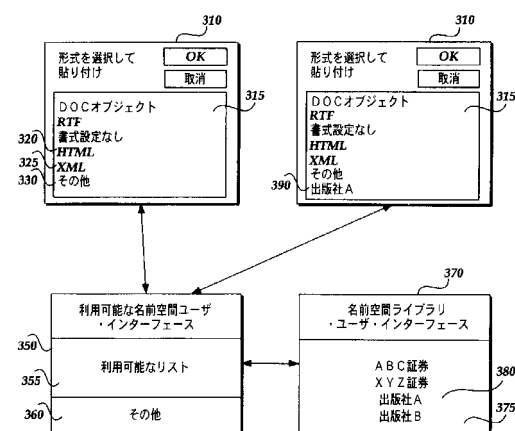
(54) 【発明の名称】 コンピュータ・ソフトウェア・アプリケーションのペースト機能を拡張するための方法およびシステム

(57) 【要約】

【課題】 コンピュータが生成した文書中にデータをペーストするためのコンピュータ・ソフトウェア・アプリケーションが利用可能なペースト機能を拡張するための、方法およびシステムを提供すること。

【解決手段】 ペーストするために文書からデータが選択される。選択データはクリップボードなどの記憶域に保存される。選択データに関連するデータ型に関する情報を消費側又はペースト・アプリケーションに提供するために、選択データに関連する情報又はデータが選択データと共に保存される。消費側アプリケーションは、ペーストされるデータのデータ構造及び書式設定がペースト前の選択データに適用されていた状態で維持されるように、選択データを第1の文書に関連付けられた第1のデータ型からその選択データを第2の文書にペーストするための第2のデータ型に変換するための、名前空間、及び拡張スタイルシート言語変換ファイルなどの関連リソースを取得する。

【選択図】 図3



【特許請求の範囲】**【請求項 1】**

コンピュータ・ソフトウェア・アプリケーションのペースト機能を拡張する方法であって、

供給側アプリケーションによって第 1 の文書を開くこと、

消費側アプリケーションによって第 2 の文書にペーストするために、前記第 1 の文書からデータを選択すること、

前記選択されたデータを記憶域に保存すること、

前記選択されたデータに関する情報を前記消費側アプリケーションに供給するために、前記選択されたデータに関連付けられたデータを前記記憶域に保存すること、

前記選択されたデータに関連付けられた前記データが、前記選択されたデータに関連付けられた 1 つまたは複数のデータ型を識別する前記選択されたデータに関連付けられた、第 1 の名前空間を識別する場合には、前記第 1 の名前空間を前記消費側アプリケーションに供給すること、

前記選択されたデータを前記第 2 の文書にペーストするために、前記 1 つまたは複数のデータ型のうちの 1 つを選択すること、および

前記消費側アプリケーションによって前記第 2 の文書にペーストするための前記データを作成するために、前記選択されたデータ型に関連付けられたリソースを取得すること
を備えることを特徴とする方法。

【請求項 2】

前記消費側アプリケーションによって前記第 2 の文書にペーストするための前記データを作成するために前記選択されたデータ型に関連付けられたリソースを取得する前記ステップに先立ち、利用可能な名前空間リストを解析して前記選択されたデータに関連付けられた前記リソースを探すこと、

前記利用可能な名前空間リストが前記選択されたデータに関連付けられた前記リソースを識別する場合には、前記リソースを前記消費側アプリケーションに供給すること
をさらに備えることを特徴とする請求項 1 に記載の方法。

【請求項 3】

前記利用可能な名前空間リストが前記選択されたデータに関連付けられた前記リソースを識別しない場合には、名前空間ライブラリを解析して前記選択されたデータに関連付けられた前記リソースを探すこと、

前記名前空間ライブラリが前記選択されたデータに関連付けられた前記リソースを識別する場合には、前記リソースを前記消費側アプリケーションに供給すること
をさらに備えることを特徴とする請求項 2 に記載の方法。

【請求項 4】

複数の名前空間が前記選択されたデータに関連付けられた前記データによって識別された場合には、前記消費側アプリケーションによって前記第 2 の文書にペーストするための前記データを作成するために、前記複数の名前空間の 1 つに関連付けられた第 2 のリソースを取得することをさらに備えることを特徴とする請求項 3 に記載の方法。

【請求項 5】

前記第 1 のリソースに従って、前記選択されたデータを前記第 2 の文書にペーストすること、および

前記第 2 にリソースに従って前記選択されたデータを前記第 2 の文書にペーストすることの選択肢をユーザに提供すること

をさらに備えることを特徴とする請求項 4 に記載の方法。

【請求項 6】

ユーザが、複数の利用可能な名前空間の 1 つから選択された 1 つまたは複数のリソースのうちの 1 つに従って、前記選択されたデータを前記第 2 の文書にペーストすることを選択できるようにすることをさらに備え、前記複数の利用可能な名前空間のそれぞれが 1 つまたは複数のリソースに関連付けられていることを特徴とする請求項 4 に記載の方法。

10

20

30

40

50

【請求項 7】

前記選択されたデータを記憶域に保存することは、前記選択されたデータを拡張マークアップ言語 (XML) 書式で保存することを含むことを特徴とする請求項 5 に記載の方法。

【請求項 8】

前記第 1 および第 2 のリソースは、前記選択されたデータを、前記第 1 の文書に関連付けられた第 1 のデータ型から、前記第 2 の文書に前記選択されたデータをペーストするための第 2 のデータ型に変換する拡張スタイルシート言語変換 (XSLT) を含むことを特徴とする請求項 7 に記載の方法。

【請求項 9】

前記消費側アプリケーションによって前記第 2 の文書にペーストするための前記データを作成するために前記選択されたデータ型に関連付けられたリソースがない場合には、拡張ペースト機能なしに前記選択されたデータを前記第 2 の文書にペーストすることをさらに備えることを特徴とする請求項 8 に記載の方法。

【請求項 10】

前記選択されたデータに関連付けられたデータを前記記憶域に保存することは、前記データを CF__XML__HEADER 書式で前記記憶域に保存することを含むことを特徴とする請求項 9 に記載の方法。

【請求項 11】

前記第 1 の名前空間が前記第 1 の文書に関連付けられていると識別された場合には、前記選択されたデータに関連付けられた前記データと共に、前記消費側アプリケーションが前記第 1 の名前空間を前記利用可能な名前空間リストまたは名前空間ライブラリ中に探し出せるようにするための情報を、供給することをさらに備えることを特徴とする請求項 10 に記載の方法。

【請求項 12】

前記選択されたデータに関連付けられた前記データと共に、前記消費側アプリケーションが前記第 1 の名前空間を前記利用可能な名前空間リストまたは名前空間ライブラリ中に探し出せるようにするための情報、を供給することは、

前記第 1 の名前空間のバージョン識別子を供給すること、

前記第 1 の名前空間のユニフォーム・リソース識別子を供給すること、および

前記第 1 の名前空間のファイル・サイズを供給すること

を含むことを特徴とする請求項 11 に記載の方法。

【請求項 13】

前記複数の名前空間が前記第 1 の文書に関連付けられていると識別された場合に、前記選択されたデータに関連付けられた前記データと共に、前記消費側アプリケーションが前記複数の名前空間を前記利用可能な名前空間リストまたは名前空間ライブラリ中に探し出せるようにするための情報、を供給することをさらに備えることを特徴とする請求項 12 に記載の方法。

【請求項 14】

前記選択されたデータに関連付けられた前記データと共に、前記消費側アプリケーションが前記複数の名前空間を前記利用可能な名前空間リストまたは名前空間ライブラリ中に探し出せるようにするための情報、を供給することは、

前記複数の名前空間それぞれのバージョン識別子を供給すること、

前記複数の名前空間それぞれのユニフォーム・リソース識別子を供給すること、および

前記複数の名前空間それぞれのファイル・サイズを供給すること

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 の名前空間が前記第 1 の文書に関連付けられていると識別された場合に、前記第 1 の名前空間に従って前記第 1 の文書に関連付けられたいずれかのデータ型を前記消費側アプリケーションに供給することをさらに備えることを特徴とする請求項 10 に記載の

10

20

30

40

50

方法。

【請求項 16】

前記第 1 の名前空間に従って前記第 1 の文書に関連付けられた前記データ型は、C F _ X M L _ V i e w 書式によって前記消費側アプリケーションに供給されることを特徴とする請求項 15 に記載の方法。

【請求項 17】

前記複数の名前空間が前記第 1 の文書に関連付けられていると識別された場合に、前記複数の名前空間に従って前記第 1 の文書に関連付けられた 1 つまたは複数のデータ型を前記消費側アプリケーションに供給することを特徴とする請求項 13 に記載の方法。

【請求項 18】

前記複数の名前空間に従って前記第 1 の文書に関連付けられた前記 1 つまたは複数のデータ型は、C F _ X M L _ D a t a 書式によって前記消費側アプリケーションに供給されることを特徴とする請求項 17 に記載の方法。

【請求項 19】

前記供給側アプリケーションはワープロ・アプリケーションであることを特徴とする請求項 1 に記載の方法。

【請求項 20】

前記供給側アプリケーションは表計算アプリケーションであることを特徴とする請求項 1 に記載の方法。

【請求項 21】

前記供給側アプリケーションはスライド・プレゼンテーション・アプリケーションであることを特徴とする請求項 1 に記載の方法。

【請求項 22】

前記消費側アプリケーションはワープロ・アプリケーションであることを特徴とする請求項 1 に記載の方法。

【請求項 23】

前記消費側アプリケーションは表計算アプリケーションであることを特徴とする請求項 1 に記載の方法。

【請求項 24】

前記消費側アプリケーションはスライド・プレゼンテーション・アプリケーションであることを特徴とする請求項 1 に記載の方法。

【請求項 25】

前記選択されたデータを記憶域に保存することは、前記選択されたデータをクリップボードに保存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 26】

コンピュータ・ソフトウェア・アプリケーションのペースト機能を拡張する方法であって、

供給側アプリケーションによって第 1 の文書を開くこと、

マークアップ言語に従って前記第 1 の文書に構造を適用すること、

第 2 の文書にペーストするために、前記第 1 の文書からデータを選択すること、

前記選択されたデータを記憶域に保存すること、

前記選択されたデータに関する情報を消費側アプリケーションに供給するために、前記選択されたデータに関連付けられたデータを前記記憶域に保存すること、

前記消費側アプリケーションによって第 2 の文書を開き、前記選択されたデータを前記第 2 の文書にペーストするために消費側アプリケーションのペースト機能を選択すること、

前記消費側アプリケーションによって、前記選択されたデータに関連付けられた前記データを読み込むこと、

前記消費側アプリケーションによって前記選択されたデータに関連付けられた前記データを読み込むことに応じて、前記選択されたデータに適用されているデータ構造に対して

10

20

30

40

50

、許容データ・コンテンツ、データ型およびデータ構造を定義する1つまたは複数の名前空間が、前記選択されたデータに、関連付けられているかどうかを決定すること、

前記選択されたデータを前記第2の文書にペーストするために、前記消費側アプリケーションによって前記1つまたは複数の名前空間の1つに関連付けられたリソースを取得すること、および

前記リソースに従って、前記選択されたデータを前記第2の文書にペーストすることを備えることを特徴とする方法。

【請求項27】

前記選択されたデータをペーストするために前記消費側アプリケーションによって前記1つまたは複数の名前空間の1つに関連付けられたリソースを取得することに先立ち、

利用可能な名前空間リストが、前記消費側アプリケーションが使用するための前記1つまたは複数の名前空間に所望の名前空間を含んでいるかどうかを決定すること、

前記利用可能な名前空間リストが、前記消費側アプリケーションが使用するための前記1つまたは複数の名前空間に所望の名前空間を含んでいる場合には、前記所望の名前空間を選択すること、および

前記選択された名前空間を前記消費側アプリケーションに供給すること
をさらに備えることを特徴とする請求項26に記載の方法。

【請求項28】

前記利用可能な名前空間リストが、前記消費側アプリケーションが使用するための前記1つまたは複数の名前空間に所望の名前空間を含んでいない場合には、名前空間ライブラリが、前記消費側アプリケーションが使用するための1つまたは複数の名前空間のうち1つについて、1つまたは複数のリソースを含んでいるかどうかを決定すること、

前記名前空間ライブラリが、前記消費側アプリケーションが使用するための1つまたは複数の名前空間について1つまたは複数の所望のリソースを含んでいる場合には、前記名前空間ライブラリから前記所望の名前空間を選択すること、および

前記選択された名前空間およびリソースを前記消費側アプリケーションに供給すること
をさらに備えることを特徴とする請求項27に記載の方法。

【請求項29】

前記選択された名前空間に従って、前記選択されたデータを前記第2の文書にペーストすることをユーザが選択できるようにすることをさらに備えることを特徴とする請求項28に記載の方法。

【請求項30】

前記マークアップ言語は拡張マークアップ言語（XML）であることを特徴とする請求項29に記載の方法。

【請求項31】

前記リソースは、前記選択されたデータを、前記第1の文書に関連付けられた第1のデータ型から、前記第2の文書に前記選択されたデータをペーストするための第2のデータ型に変換する拡張スタイルシート言語変換ファイルであることを特徴とする請求項30に記載の方法。

【請求項32】

前記選択されたデータに関する情報を前記消費側アプリケーションに供給するために前記選択されたデータに関連付けられたデータを前記記憶域に保存する前記ステップは、前記1つまたは複数の名前空間のいずれかが前記第1の文書に関連付けられているかどうかを識別することを含むことを特徴とする請求項29に記載の方法。

【請求項33】

前記1つまたは複数の名前空間のいずれかが前記第1の文書に関連付けられているかどうかを識別することは、前記利用可能な名前空間リストで、または名前空間ライブラリで前記1つまたは複数の名前空間のいずれかを探し出すための情報を、消費側アプリケーションに供給することを含むことを特徴とする請求項32に記載の方法。

【請求項34】

コンピュータ実行可能命令を記憶したコンピュータ可読媒体であって、前記コンピュータ実行可能命令が、コンピュータによって実行されると、
供給側アプリケーションによって第1の文書を開くこと、
マークアップ言語に従って前記第1の文書に構造を適用すること、
第2の文書にペーストするために、前記第1の文書からデータを選択すること、
前記選択されたデータを記憶域に保存すること、
前記選択されたデータに関する情報を消費側アプリケーションに供給するために、前記選択されたデータに関連付けられたデータを前記記憶域に保存すること、
前記消費側アプリケーションによって第2の文書を開き、前記選択されたデータを前記第2の文書にペーストするために消費側アプリケーションのペースト機能を選択すること 10
、
前記消費側アプリケーションによって、前記選択されたデータに関連付けられた前記データを読み込むこと、
前記消費側アプリケーションによって前記選択されたデータに関連付けられた前記データを読み込むことに応じて、前記選択されたデータに適用されているデータ構造に対して、許容データ・コンテンツ、データ型およびデータ構造を定義する前記選択されたデータに、1つまたは複数の名前空間が関連付けられているかどうかを決定すること、
前記選択されたデータを前記第2の文書にペーストするために、前記消費側アプリケーションによって前記1つまたは複数の名前空間の1つに関連付けられたリソースを取得すること、および 20
前記リソースに従って、前記選択されたデータを前記第2の文書にペーストすることのそれぞれのステップを実行することとを特徴とするコンピュータ可読媒体。
【請求項35】
前記選択されたデータをペーストするために、前記消費側アプリケーションによって前記1つまたは複数の名前空間に関連付けられたリソースを取得することに先立ち、
利用可能な名前空間リストが、前記消費側アプリケーションが使用するための前記1つまたは複数の名前空間に所望の名前空間を含んでいるかどうかを決定すること、
前記利用可能な名前空間リストが、前記消費側アプリケーションが使用するための前記1つまたは複数の名前空間に所望の名前空間を含んでいる場合には、前記所望の名前空間を選択すること、および 30
前記選択された名前空間を前記消費側アプリケーションに供給すること
をさらに備えることを特徴とする請求項34に記載のコンピュータ可読媒体。
【請求項36】
前記利用可能な名前空間リストが、前記消費側アプリケーションが使用するための前記1つまたは複数の名前空間に所望の名前空間を含んでいない場合には、名前空間ライブラリが、前記消費側アプリケーションが使用するための1つまたは複数の名前空間に所望の名前空間を含んでいるかどうかを決定すること、
前記名前空間ライブラリが、前記消費側アプリケーションが使用するための1つまたは複数の名前空間について1つまたは複数の所望のリソースを含んでいる場合には、前記名前空間ライブラリから前記所望の名前空間を選択すること、および 40
前記選択された名前空間およびリソースを前記消費側アプリケーションに供給すること
をさらに備えることを特徴とする請求項35に記載のコンピュータ可読媒体。
【請求項37】
前記選択された名前空間に従って、前記選択されたデータを前記第2の文書にペーストすることをユーザが選択できるようにすることをさらに備えることを特徴とする請求項36に記載のコンピュータ可読媒体。
【請求項38】
前記マークアップ言語は拡張マークアップ言語（XML）であることを特徴とする請求項37に記載のコンピュータ可読媒体。
【請求項39】

前記選択されたデータに関する情報を前記消費側アプリケーションに供給するために前記選択されたデータに関連付けられたデータを前記記憶域に保存することは、前記1つまたは複数の名前空間のいずれかが前記第1の文書に関連付けられているかどうかを識別することを含むことを特徴とする請求項38に記載のコンピュータ可読媒体。

【請求項40】

前記1つまたは複数の名前空間のいずれかが前記第1の文書に関連付けられているかどうかを識別することは、前記利用可能な名前空間リスト上で、または名前空間ライブラリで前記1つまたは複数の名前空間のいずれかを探し出すための情報を、消費側アプリケーションに供給することを含むことを特徴とする請求項39に記載のコンピュータ可読媒体。

10

【請求項41】

データをコピー・アプリケーションから消費側アプリケーションにペーストするための方法であって、

第1の拡張マークアップ言語（XML）要素を第1の文書のある領域で、前記コピー・アプリケーションからコピーし、前記消費側アプリケーションにペーストするために選択される領域の開始部分に適用すること、

第2のXML要素を前記選択された領域の終了部分に適用すること、

前記選択された領域に関連付けられたヘッダ中に、利用可能なXML名前空間の識別、および前記利用可能な名前空間のそれぞれに対応するクリップボード書式識別へのポインタを含む情報を供給すること、

20

前記ヘッダ中に、前記利用可能な名前空間それぞれのファイル・サイズに関する情報を供給すること、および

前記利用可能な名前空間のうち1つまたは複数の名前空間をクリップボード上に供給して、前記消費側アプリケーションが、前記コピー・アプリケーションから前記データをペーストするために前記1つまたは複数の名前空間から選択できるようにすること

を備えることを特徴とする方法。

【発明の詳細な説明】

【技術分野】

【0001】

30

本発明は、コンピュータによって生成した文書中にデータをペーストするためのコンピュータ・ソフトウェア・アプリケーションに利用可能なペースト機能を拡張するための方法およびシステムに関する。

【背景技術】

【0002】

コンピュータ・ソフトウェア・アプリケーションは、仕事上で、教育上で、およびレジャーにおいて、彼らを支援する様々な文書をユーザが作成することを可能にする。例えば、ワープロ・アプリケーションによって、ユーザは、手紙、記事、本、メモなどを作成することができる。表計算アプリケーションによって、ユーザは、様々な英数字データを記憶させ、操作し、印刷し、表示することができる。このようなアプリケーションは、高機能の編集、書式設定、印刷および計算を含めて、いくつかの周知の長所を備えている。

40

【0003】

多くのソフトウェア・アプリケーションのきわめて一般的かつ有益な機能として、所与の文書、スプレッドシート、スライド・プレゼンテーション、またはその他のコンピュータによって生成した文書からデータをカットまたはコピーし、次いで、そのカットまたはコピーしたデータを文書中の所望の位置にペーストする、ペースト操作を行う機能が挙げられる。ユーザは、テキスト、画像、または英数字データを含むデータを、ワープロ・アプリケーションなどの第1の、または供給側のアプリケーションからカットまたはコピーし、次いで、そのカットまたはコピーしたデータを、表計算アプリケーションなどの第2の、または消費側のアプリケーションを使ってペーストすることが多い。例えば、ユーザ

50

が、表計算アプリケーション文書から一連の数字をカットまたはコピーして、それらの数字を顧客または友人に送る手紙を作成中のワープロ・アプリケーション文書中にペーストしたいという場合がある。一般に、データがカットまたはコピーされると、そのデータは、その後のペースト操作におけるペーストに備えて、一時的に記憶域に入れられる。

【0004】

供給側アプリケーションと消費側アプリケーションが異なるソフトウェア・アプリケーションの場合に、データを供給側アプリケーションからカットまたはコピーして、それを第2の消費側アプリケーションによって別個の文書中にペーストすると、書式設定を含めて、第1のアプリケーションによって提供されていた特別の特徴(feature)を提供するのに必要な機能を、第2の、または消費側のアプリケーションは含んでいないため、全てではないとしても、第1のアプリケーションによって提供されていた特徴(feature)の多くは、書式設定を含めて、ペースト操作中に失われてしまう。例えば、ユーザが、表計算アプリケーション文書から一連のデータをコピーして、そのデータをワープロ文書中にペーストした場合、ワープロ文書中にペーストされたデータは、表計算アプリケーションにおいては存在していた書式設定が全くなしにペーストされる場合がある。すなわち、表計算アプリケーションでのデータがいくつかの列と行で並べられていた場合に、書式設定が何もない、単なる一連の数字としてワープロ・アプリケーション中にペーストされる場合がある。

【0005】

ソフトウェアのより高度な機能に対する需要について行くために、ソフトウェア開発者は、ソフトウェア・アプリケーションの通常の機能性から離れた構造で、文書あるいはその文書に関連付けた目に見える書式設定を作成する役目を負う有用な構造を文書に与えるために、ユーザがソフトウェア・アプリケーション文書に注釈を付けることを可能にする拡張マークアップ言語などのマークアップ言語を使い始めている。例えば、ユーザが、自分のワープロ・アプリケーション上で、ある出版社に送信したいと考えている記事を作成するためのテンプレート文書を作成したいという場合がある。その文書に構造を適用することによって、その文書を受け取った出版社は、その文書を処理することによりその構造を使用して、その文書構造によって定義されているデータを利用することができる。

【発明の開示】

【発明が解決しようとする課題】

【0006】

残念なことに、このような構造上の注釈を有する文書からデータをカットまたはコピーした場合、特に、異なる消費側アプリケーションによってペースト操作が実行された場合には、そのペースト操作の間に構造上の注釈が失われることが多い。消費側アプリケーションがマークアップ言語を理解し、使用するようにプログラムされている場合でも、供給側アプリケーションにおけるマークアップ言語の構造上の注釈を定義しているスキーマまたは規則は、消費側アプリケーションと比べると著しく異なる場合がある。

【0007】

本発明は、これらの、およびその他の考慮すべき課題に対処する。

【課題を解決するための手段】

【0008】

本発明は、コンピュータによって生成した文書中にデータをペーストするためのコンピュータ・ソフトウェア・アプリケーションに利用可能なペースト機能を拡張するための方法およびシステムを提供する。全般的に説明すると、第2のアプリケーション文書にペーストするために、第1のアプリケーションからデータを選択する。選択されたデータは、名前空間リストなど、そのデータに関連付けられた情報と共に記憶域に保存される。選択されたデータが第2のアプリケーションまたは消費側アプリケーション文書にペーストされると、消費側アプリケーションは、選択されたデータに関連付けられている利用可能な名前空間リストを調べて、そのデータの様々な表現のいずれが、ペースト・アプリケーションによって理解されるかどうかを調べる。ペースト・アプリケーションはまた、選択さ

れたデータを、第2または消費側アプリケーションが一層消費し易い書式に変換するための拡張スタイルシート言語(XSL)変換ファイルなどの、いずれかのリソースが利用可能かどうかを調べることができる。利用可能な名前空間で支援可能なリソースを見つけるための方法の1つは、データを消費側アプリケーションが使用できるように変換するための利用可能なリソースについて、名前空間ライブラリを調べることである。リソースが何も見つからず、また、名前空間がペースト・アプリケーションによって本来的に、未だに理解されていない場合には、そのデータは、デフォルトのペースト・リソースに従って第2のアプリケーション文書にペーストされる可能性がある。ペースト・アプリケーションはまた、名前空間に関係なくどの型のデータでも扱うためのデフォルトの方法を備えることもできる。

10

【0009】

より詳細には、コンピュータ・ソフトウェア・アプリケーションのペースト機能を拡張する方法を提供する。供給側アプリケーションによって第1の文書が開かれ、拡張マークアップ言語(XML)などのマークアップ言語に従って、第1の言語に構造が適用される。第1のアプリケーションが、既存のデータを意味のあるやり方で表示する目的を果たすだけのこともあり得る。第2のアプリケーションにペーストするために、第1のアプリケーションからデータが選択される。選択されたデータは記憶域に保存され、選択されたデータに関連付けられた情報も、選択されたデータに関する情報を消費側アプリケーションに提供するためにその記憶域に保存される。この情報は、主として名前空間リストから成る。これらの名前空間は、コピー・アプリケーションがデータを表現するために持っている、異なる方法を識別する。この一例として、財務データを備えたスプレッドシート文書が挙げられる。提供される名前空間の一方はスプレッドシートを記述するためのものであり、提供されるもう一方の名前空間は、財務データを記述するためのものである。

20

【0010】

消費側アプリケーションによって第2の文書が開かれ、選択されたデータを第2の文書にペーストするために消費側アプリケーションでペースト機能が選択される。消費側アプリケーションは、選択されたデータに関連付けられた情報を読み込む。これに応答して、消費側アプリケーションは、選択されたデータに関連付けられているデータ型を識別する1つまたは複数の名前空間を、消費側アプリケーションが本来的に理解できるかどうか、を決定する。2つ以上の名前空間がある場合には、アプリケーションは、どの名前空間および関連リソースを使用すべきかの選択肢をユーザに与え、あるいは、アプリケーションが、すでに消費側アプリケーション文書中にあるデータに最も適合する名前空間および関連リソースを選択する。提示されるこの選択肢は、どの名前空間を使用すべきかという選択肢であってもよいし、あるいは、いずれかの1つの所与の名前空間のどのリソースを使用すべきかという選択肢であってもよい。消費側アプリケーションは、名前空間をまだ理解していない場合には、選択されたデータを第2の文書にペーストするために、その名前空間に関連付けられたXSLT変換ファイルなど、1つまたは複数のリソース・ファイルを取得することができる。消費側アプリケーションが、選択されたデータを第2の文書にペーストするために1つまたは複数の名前空間のうちの1つを選択した後、その選択されたデータは、消費側アプリケーションが好む方法に従って第2の文書にペーストされる。時々、使用されることがある追加リソースとして、第2または消費側アプリケーションが消費可能なデータ型または書式に選択されたデータを変換するための拡張スタイルシート言語変換(XSLT)ファイルが挙げられる。

30

40

【0011】

本発明の一態様によれば、選択データをペーストするために1つまたは複数の名前空間のリソースを取得するのに先立ち、利用可能な名前空間リストが、消費側アプリケーションが理解する1つまたは複数の名前空間のうちの所望の名前空間を含んでいるかどうかについての決定を行う。利用可能な名前空間リストが、消費側アプリケーションが理解する1つまたは複数の名前空間のうちの所望の名前空間を含んでいる場合には、その所望の名前空間を選択して消費側アプリケーションに供給する。利用可能な名前空間リストが、消費

50

側アプリケーションが理解する1つまたは複数の名前空間のうちの所望の名前空間を含んでいない場合には、消費側アプリケーションが使用するための1つまたは複数の名前空間を消費側アプリケーションが理解するのを支援するためのリソースを、名前空間ライブラリが含んでいるかどうかに関する決定を行う。消費側アプリケーションが使用するための1つまたは複数の名前空間のリソースを名前空間ライブラリが含んでいる場合には、所望のリソースが名前空間ライブラリから選択され、消費側アプリケーションが1つまたは複数の名前空間を消費するのを支援するために、消費側アプリケーションに供給される。

【0012】

消費側アプリケーションに提供するために、選択されたデータに関連付けられた情報が記憶域に保存される。このデータは、選択されたデータに関連付けられた1つまたは複数の名前空間が存在することの指標を含む場合がある。選択されたデータに関連付けられた情報は、選択されたデータに関連付けられた各名前空間について、名前空間のバージョン識別子、名前空間を探し出すためのユニフォーム・リソース識別子、および名前空間のファイル・サイズを含む、いずれかの関連名前空間に関する情報を含むことができる。

【0013】

本発明の別の態様によれば、データをコピー・アプリケーションから消費側アプリケーションにペーストするための方法が提供される。第1の拡張マークアップ言語（XML）要素が、第1の文書のある領域の開始部分に適用される。その領域は、コピー・アプリケーションからコピーし、消費側アプリケーションにペーストするために選択される。第2のXML要素が、その選択された領域の終了部分に適用される。選択された領域に関連付けられたヘッダ中に、情報が供給される。この情報は、利用可能なXML名前空間の識別、および利用可能な名前空間のそれぞれに対応するクリップボード書式識別（clipboard format identification）へのポインタを含む。利用可能な名前空間それぞれについてのファイル・サイズに関する情報がヘッダ中に供給される。利用可能な名前空間のうち1つまたは複数の名前空間がクリップボードに供給され、コピー・アプリケーションからデータをペーストするために、消費側アプリケーションがその1つまたは複数の名前空間を選択することを可能にする。

【0014】

本発明を特徴付けるこれらおよびその他の特徴および利点は、以下の詳細な説明を読み、関連する図面を参照することにより明らかになる。上述の全般的な説明および以下の詳細な説明は例にすぎず、また説明のためのものにすぎず、特許請求の範囲に記載しているように、本発明を限定するものではない。

【発明を実施するための最良の形態】

【0015】

いくつかの図を通して、同じ番号は同様の部分または構成要素を指している添付の図面、を参照しながら、以下に、本発明の実施形態の説明を行う。本発明は、コンピュータによって生成した文書中にデータをペーストするためのコンピュータ・ソフトウェア・アプリケーションに利用可能なペースト機能を拡張するための、方法およびシステムを対象としている。ユーザは、ワープロ・アプリケーションなどの第2のアプリケーションにペーストするために、表計算アプリケーションなどの第1のアプリケーションからデータを選択する。ペースト・プロセスは、従来のカット／コピーおよびペースト操作の形態であってもよく、あるいは、選択したデータを、第1のアプリケーション文書から第2のアプリケーション文書にドラッグ・アンド・ドロップする形態であってもよい。本発明の一実施形態によれば、第1または供給側アプリケーションは、選択されたデータを拡張マークアップ言語（XML）表現でクリップボードなどの記憶域に書き込む。選択されたデータは、その選択されたデータ、およびその選択されたデータに関連付けられた1つまたは複数のデータ名前空間に関連付けられた情報とを含む書式で、記憶域に書き込まれる。選択されたデータに関連付けられた利用可能な名前空間は、その選択されたデータに関連付けることができるデータ書式設定のタイプを識別することができる。例えば、識別される利用可能な名前空間には、ハイパーテキスト・マークアップ言語（HTML）、レジューム文

書 (resume document)、またはワープロXML (word processor XML) が含まれ得る。HTML 名前空間は、データがHTMLとして構造化されている可能性があることを識別することができる。レジューム文書名前空間は、データがレジュームXMLスキーマ・ファイルに従って構造化されている可能性があることを識別することができる。ワープロXML 名前空間は、データがワープロ・バージョンのXMLに従って書式設定されている可能性があることを識別することができる。もちろん、これらの名前空間は、選択されたデータに関連付けることができる多くのタイプの名前空間の例にすぎない。

【0016】

第2のアプリケーションのペースト機能を選択することによって、または選択したデータを第2のアプリケーション文書上にドロップすることによって、選択したデータを第2のアプリケーション文書にペーストすると、第2または消費側アプリケーションは、選択したデータと共に供給された情報から、XML書式のデータがペーストされていることを検出する。選択したデータと共に供給された情報に基づいて、消費側アプリケーションは、ペーストすべきデータに関連付けられている利用可能な名前空間の全てを検出する。例えば、上述の例の場合、消費側アプリケーションは、選択されたデータがHTML、レジューム文書、およびワープロXMLのための名前空間に関連付けられていることを検出することができる。これに応じて、消費側アプリケーションは、利用可能な名前空間リストを調べて、消費側アプリケーションが選択したデータをペーストする際の援助となるリソースを探す。例えば、供給側アプリケーションが表計算アプリケーションの場合、消費側アプリケーションは、利用可能な名前空間リストを調べて、スプレッドシートXML書式のデータを消費側アプリケーションが消費および使用できる書式に変換するための拡張スタイルシート言語変換 (XSLT) ファイルを探すことができる。消費側アプリケーションがワープロの場合、消費側アプリケーションは、スプレッドシート・データをそのワープロ・アプリケーションが消費できるデータに変換するための、XSLT変換ファイルを探ることができる。例えば、データが、表計算アプリケーションで2行3列の書式で構造化されている場合、消費側アプリケーションが使用しているXSLT変換ファイルが、そのワープロ・アプリケーションが消費するためのデータを、2行3列の書式設定のまま維持されるように変換することができる。本発明のこの機能がなければ、そのデータは、消費側アプリケーションのデフォルトのペースト機能によってペーストされ、したがって、供給側アプリケーションによる書式設定なしにペーストされ得る。その結果、この例の場合でいえば、2行3列に書式設定された状態ではなく、ワープロのワークスペース上を左から右に提示された生データとして、データがワープロ・アプリケーションにペーストされる可能性がある。

【0017】

消費側アプリケーションは、複数の利用可能な名前空間が識別された場合には、所望のペースト機能を提供してくれる可能性が最も高い利用可能な名前空間を選択することができる。あるいは、消費側アプリケーションのユーザにユーザ・インターフェースが提供されて、ユーザが、利用可能な名前空間リストについての、そのアプリケーションが知っている利用可能な変換ファイルの中から選択できる場合もある。消費側アプリケーションは、利用可能な名前空間リスト中にリソースを何も見つけられない場合には、名前空間ライブラリに行って、消費側アプリケーション文書にデータをペーストするのに先立って選択されたデータを変換する際の支援となる、利用可能なリソースを探すことができる。名前空間ライブラリで複数のリソースが探し出された場合には、ユーザが利用可能なリソースの中から選択できるように、ユーザ・インターフェースが提供されることができる。利用可能な名前空間リストにも名前空間ライブラリにも利用可能なリソースが見つからなかった場合には、上述のように、消費側アプリケーションはデフォルト機能に従ってデータをペーストする。

【0018】

動作環境

図1および以下の説明は、本発明を実施することができる適切なコンピューティング環

境を、簡単かつ全般的に説明することを意図している。本発明を、パーソナル・コンピュータに関連してオペレーティング・システム上で実行しているアプリケーション・プログラムという一般的な状況で説明するが、当業者は、本発明をその他のプログラム・モジュールとの組合せで実施できることを認識されるであろう。一般に、プログラム・モジュールには、特定のタスクを実行し、または特定の抽象データ型を実装する、ルーチン、プログラム、コンポーネント、データ構造などが含まれる。さらに、当業者は、ハンドヘルド・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・ベースまたはプログラム可能な家庭用電子機器、携帯電話、ミニ・コンピュータ、メイン・フレーム・コンピュータなどを含めて、その他のコンピュータシステム構成で実施できることを理解されよう。本発明はまた、通信ネットワークを介してリンクされたりリモート処理デバイスによってタスクが実行される、分散コンピューティング環境においても実施することができる。分散コンピューティング環境においては、プログラム・モジュールをローカルおよびリモートの記憶装置の両方に配置することができる。

【0019】

図1を参照すると、本発明を実施するための例示的システムは、処理装置21、システム・メモリ22、および、システム・メモリを処理装置21に結合するシステム・バス23を含む、従来のパーソナル・コンピュータ20を含む。システム・メモリ22は、読取り専用メモリ（ROM）24およびランダム・アクセス・メモリ（RAM）25を含む。起動時などにパーソナル・コンピュータ20内の要素間における情報の転送を支援する、基本入出力システム26（BIOS）がROM24に記憶されている。パーソナル・コンピュータ20はさらに、ハードディスク・ドライブ27、例えば、リムーバブル・ディスク29からの読取りまたはそこへの書込みを行うための磁気ディスク・ドライブ28、および、例えば、CD-ROMディスク31の読取り、あるいはその他の光媒体からの読取りまたはそこへの書込みを行うための光ディスク・ドライブ30を含む。ハードディスク・ドライブ27、磁気ディスク・ドライブ28、および光ディスク・ドライブ30は、それぞれ、ハードディスク・ドライブ・インターフェース32、磁気ディスク・ドライブ・インターフェース33、および光ディスク・ドライブ・インターフェース34によってシステム・バス23に接続されている。ドライブおよびそれらの関連するコンピュータ可読媒体は、パーソナル・コンピュータ20に不揮発性記憶を提供する。上記のコンピュータ可読媒体の説明は、ハードディスク、リムーバブル磁気ディスクおよびCD-ROMディスクを対象としているが、当業者は、磁気カセット、フラッシュ・メモリ・カード、デジタル・ビデオ・ディスク、ベルヌーイ・カートリッジなど、コンピュータが読取り可能なその他のタイプの媒体もこの例示的動作環境では使用できることを理解されたい。

【0020】

オペレーティング・システム35、1つまたは複数のアプリケーション・プログラム200、220、利用可能な名前空間リスト225などのプログラム・データ、およびその他のプログラム・モジュール（図示せず）を含めて、いくつかのプログラム・モジュールをドライブおよびRAM25に記憶することができる。

【0021】

ユーザは、キーボード40、およびマウス42などのポインティング・デバイスを介して、パーソナル・コンピュータ20にコマンドおよび情報を入力することができる。その他の入力デバイス（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、スキャナなどが含まれる。これらおよびその他の入力デバイスは、システム・バスに結合されたシリアル・ポート・インターフェース46を介して、処理装置21に接続されていることが多い。しかし、ゲーム・ポートまたはユニバーサル・シリアル・バス（USB）など、その他のインターフェースによって接続されている場合もある。モニタ47またはその他のタイプのディスプレイ・デバイスも、ビデオ・アダプタ48などのインターフェースを介してシステム・バス23に接続されている。モニタに加えて、パーソナル・コンピュータは、一般に、スピーカやプリンタなどのその他の周辺出力デバイス（図示せず）を含む。

【0022】

パーソナル・コンピュータ20は、リモート・コンピュータ・サーバ49などの1つまたは複数のリモート・コンピュータへの論理接続を使って、ネットワーク化された環境において動作することができる。リモート・コンピュータ・サーバ49は、サーバ、ルータ、ピア・デバイス、またはその他の一般的なネットワークノードであってよく、一般に、パーソナル・コンピュータ20に関して説明した要素の多くまたは全てを含んでいる。ただし、図1には記憶装置50しか示していない。サーバ49および記憶装置50は、アプリケーション220などのアプリケーション・プログラム、および名前空間ライブラリ230などのデータ記憶を保持することができる。図1に示す論理接続は、ローカル・エリア・ネットワーク(LAN)51および広域ネットワーク(WAN)52を含む。このようなネットワーク化環境は、オフィス、企業規模コンピュータ・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ありふれたものである。

【0023】

LANネットワーク化環境において使用する場合、パーソナル・コンピュータ20は、ネットワーク・インターフェース53を介してLAN51に接続される。WANネットワーク化環境において使用される場合、パーソナル・コンピュータ20は、一般に、モデム54、またはインターネットなどのWAN52を介して通信を確立するためのその他の手段を含む。内蔵または外付けのモデム54は、シリアル・ポート・インターフェース46を介してシステム・バス23に接続されている。ネットワーク化環境においては、パーソナル・コンピュータ20に関して説明したプログラム・モジュール、またはその一部を、リモートの記憶装置に記憶することができる。図示のネットワーク接続は例示であり、コンピュータ間の通信リンクを確立するその他の手段も使用できることを理解されよう。

【0024】

動作

図2は、データが供給側アプリケーションを使ってカットまたはコピーされ、消費側アプリケーションを使ってペーストされる、供給側アプリケーションと消費側アプリケーションの間の相互作用を説明する、簡略化した構成図である。図2に示すように、供給側アプリケーション200を、そこからカットまたはコピーされたデータが消費側アプリケーション220の使用に備えて保存されるものとして示してある。供給側アプリケーションは、ワープロ・アプリケーション、表計算アプリケーション、スライド・プレゼンテーション・アプリケーションなどを含めて、後でペーストできるようにそこからデータをカットまたはコピーすることができる、任意のソフトウェア・アプリケーションであってよい。ユーザが供給側アプリケーション200を使ってデータを選択すると、選択されたデータを供給文書からカットすることを含み、または選択されたデータを供給文書からコピーすることを含む、カットまたはコピー操作205が行われる選択されたデータは、カットまたはコピーされると、図2に示すクリップボード210などの記憶域210に転送される。記憶域210は、後で、ユーザの選択に応じて、元の供給側アプリケーション200または消費側アプリケーション220がペーストを行えるように、そのカットまたはコピーされたデータを保持する。

【0025】

本発明の一実施形態によれば、クリップボード210は、ウィンドウ・オペレーティング・システムが維持しているメモリ・リソースを表す。クリップボードは、コピーまたはカットされたデータのコピーを記憶し、次のペースト操作によって、そのデータがクリップボード210から消費側プログラムに渡される。当業者にはよく知られているように、クリップボード210は、第2または消費側アプリケーション220が供給側アプリケーション200によって生成されたデータを読むことができる場合に、上述のように、あるアプリケーションから別のアプリケーションにデータを転送されることを可能にする。消費側アプリケーションが供給側アプリケーション200によってカットまたはコピーされたデータを確実に読み取り、理解し、利用することができるように、主に、供給側アプリケーションによってカットまたはコピーされたデータは、多くの様々な異なる消費側アプ

リケーション 220 によって消費されることができる、もっともあり得そうな書式のデータとするために、ハイパーテキスト・マークアップ言語書式などの一般的な書式でクリップボード 210 に保存される。その他のタイプの書式設定にはリッチ・テキスト書式が含まれる。あるいは、データが書式なしのデータとして保存される場合もある。これらの書式は全て、第 1 のアプリケーションがコピーしたときにクリップボード上で知らされる。その後、利用可能な書式のうちどれを要求したいかは、ペースト・アプリケーションの義務である。

【0026】

消費側アプリケーション 220 が、供給側アプリケーション 200 からカットまたはコピーしたデータのペーストを開始すると、ペースト操作 215 が選択されて、カットまたはコピーされたデータを、消費側アプリケーションが操作している文書中の所望の場所へペーストする。カット／コピー操作 205 およびペースト操作 215 を、選択されたデータが供給側アプリケーション文書からドラッグされて消費側アプリケーション文書中にドロップされる、ドラッグ・アンド・ドロップ操作の形態に結合できることを理解されたい。上述のように、カットまたはコピーされたデータを、供給側アプリケーションがいくつかの一般的な書式タイプに従って書式設定している場合、消費側アプリケーションは、所望通りにカットまたはコピーされたデータをペーストすることができる場合もあるが、しかし、特殊な書式設定を含めて、供給側アプリケーションがそのデータに与えた特徴 (feature) の多くは失われることが多い。例えば、ユーザが、ワープロの供給側アプリケーション 200 において表形式で入力されているデータをカットまたはコピーし、次に、そのデータを表計算の消費側アプリケーション 220 中にペーストした場合、元の表記式で書式設定されていたデータは、単にスプレッドシート内の 1 つのセル中に挿入される可能性があり、この場合、表計算消費側アプリケーション 220 のユーザが望むようには表計算アプリケーションがそのデータに基づいて動作できなくなる。

【0027】

拡張マークアップ言語 (XML) などのマークアップ言語の出現により、供給側アプリケーション 200 によって作成された文書は、その文書に含まれるデータを管理し、提示し、操作するために役立つ構造をその文書に与える目的で、マークアップ言語構造を使って注釈を付けるようになってきている。当業者にはよく知られているように、異なるソフトウェア・アプリケーションは、XML などのマークアップ言語を読み、理解し、利用するための異なる「ネイティブ」プログラミングを含む。すなわち、供給側アプリケーション 200 を、消費側アプリケーション 220 に比較して、それとは異なる、またはより多くの XML 構造および機能を提供するようにプログラムすることができる。したがって、異なる消費側アプリケーション 220 による消費のために、供給側アプリケーション 200 によって作成された XML 書式の文書からデータがカットまたはコピーされた場合、そのカットまたはコピーされたデータについての XML 書式設定に関連付けられた構造および機能の多くは、消費側アプリケーション 220 によって作成され操作されている文書中に消費側アプリケーション 220 がそのデータをペーストする際に失われる可能性がある。

【0028】

以下は、ユーザがワープロの供給側アプリケーション 200 で書いた記事に適用可能なサンプル XML 構造である。

【0029】

サンプル XML 構造

【0030】

10

20

30

40

【表 1】

```

<article>
  <title>EXTRA EXTRA</title>
  <body>
    <summary>Today in New York ...</summary>
    <main> Today in New York ...</main>
  </body>

  *
  *
  *

</article>

```

10

20

【0031】

このサンプルXML構造に示すように、「article」文書に注釈を付けているいくつかのXML要素を示してある。例えば、この文書を「article」として定義するために、文書の開始部分と終了部分には<article>タグが含まれている。また、「article」構造の内側には、<title>要素および<body>要素など、様々なその他の要素が含まれている。<body>要素の内側には、<body>要素の2つの子要素、<summary>および<main>がある。文書のXML注釈によって、ユーザは、文書の一部を、特定の型のデータおよびデータ構造として定義することができる。例えば、ユーザは、<title>要素の内側にある文書の一部を、「title」型のデータを含むように、また、タイトル用の所定の許容可能構造を含むように定義することができる。

30

【0032】

上記の「article」文書などの所与の文書に含むことができるデータの型および構造を管理する、1組の文法およびデータ型規則を文書に与えるために、XMLスキーマが、ユーザがそれを使って所与の文書に注釈を付けることができるXML要素およびタグのそれぞれを管理する規則を提供するために、その文書に添付されたり、または関連付けられる。例えば、「article」文書は、<article>要素、<title>要素、<body>要素などの許容可能な1組のXML要素を提供するための、「article-schema.xsd」など、添付された、または関連付けられたスキーマを備えることができる。このスキーマは、これらの要素が文書に適用される順序を管理する規則、および文書に適用される個々の要素に関連付けられた特定の規則を含む。例えば、「article」文書に添付された、または関連付けられたスキーマは、所与の要素、例えば、<date>要素（上記の説明では触れていない）に関連付けられたデータが、日付の要素、その後に月の要素、その後に年の要素を含まなければならないことを規定することができる。さらに、このスキーマは、<date>要素が<title>要素の直前になければならないことを要求することができる。

40

【0033】

当業者は理解されるように、XMLスキーマの開発者は、XML要素の名前、および、それらの要素に許容される関連データ型およびデータ構造を決定する。したがって、所与のスキーマに従ってXML構造で注釈を付けた文書のユーザは、全て、その文書の全体と

50

してのタイプおよび構造には関係なく、XML構造内に含まれるデータを利用することができる。例えば、上述の「article」文書をその文書の出版社に送信する場合、出版社が、その文書を解析し、その文書内に自社が使用するための特定の型のデータを探すソフトウェア・アプリケーションを開発することができる。出版社が、例えば、将来その記事全体を発行するために、その記事のタイトルのみを宣伝として発行したいという場合がある。XMLが識別されるこの方法は、名前空間を介して行われる。名前空間は、どのXML消費者でも、自分がどのタイプのXMLを消費しているのを知ることができるように、識別(ID)を提供する。消費側アプリケーションは、名前空間を観察することによって、XMLファイルを作成するのに何のスキーマが使用されたかを知ることになる。文書中に指定された名前空間を使って、出版社は、XML要素<title>に関連付けられたデータが、その文書を管理するスキーマに従って作成されたことを知る。したがって、出版社は、そのデータを、将来、その記事全体を発行するための宣伝として使用するために、<title>要素を探し出して、それに関連付けられたデータを抽出して、出版社独自の文書中に挿入する(何らかの特別な方法で書式設定することを含む)ためのソフトウェア・アプリケーションを、またはXSLT変換ファイルを開発することができる。

10

【0034】

この例を続けると、何社かの異なる出版社が、「article」文書に関連する規則を指示するための同じ名前空間の使用に同意することによって、各出版社は、文書の著者から「article」文書を受け取り、その記事に含まれているデータを、そのデータを構造化しているXML要素に従って使うことができる。すなわち、第1の出版社が、<summary>要素内に含まれるデータのみに関心を持ち、一方、第2の出版社が<title>要素内に含まれるデータのみを抽出することに関心がある場合がある。各出版社は、XML構造に従って所望のデータを探すための独自のソフトウェア・アプリケーションまたはXSLT変換ファイルを使うことによって、文書のその他の側面または要素には関係なく、所望するデータを抽出することができる。これは、文書の各ユーザが、文書に添付された、または関連付けられた名前空間中に規定されている、データ型およびデータ構造規則に従っていることによって可能になっている。

20

【0035】

消費側アプリケーションが、利用可能な名前空間の1つで使用するためのXSLT変換ファイルを見つけたならば、選択されたデータはそのXSLT変換ファイルによって変換され、その結果は、はペースト機能の要素として消費側アプリケーションによって受け取られる。消費側アプリケーションが変換ファイルを知っていて、利用可能な名前空間を獲得し、そのデータ自体に基づく変換を実行する場合もある。例えば、上述のように、供給側アプリケーションが表計算アプリケーションで、選択されたデータが2行3列に書式設定されている場合、例えば、ワープロ・アプリケーションによって消費されるスプレッドシート・データを変換するためのXSLT変換ファイルが、スプレッドシートで使用されている書式設定または表形式なしの生データではなく、それぞれ3つのアイテムを備えた2つのリストとして、ワープロ・アプリケーションがペーストできるように、そのデータを変換することが可能になる。第1または供給側アプリケーション文書は、カスタムXMLスキーマに従ってマーク付けできることを理解されたい。例えば、そのデータは、「株

30

40

【0036】

「株価スキーマ」を認識すると、消費側アプリケーションは、利用可能なリソース・リストを参照して、選択されたデータの各セルが企業の銘柄記号であり、各企業について、その企業に関連する記載があることを検出する。したがって、消費側アプリケーションは

50

、利用可能な名前空間リストまたは名前空間ライブラリ中にX S L T変換ファイルなどのリソースを探し出して、マークアップ・データをどのようにペーストするかを消費側アプリケーションに指示する。ユーザに提供されている消費側アプリケーションのユーザ・インターフェースは、「企業一覧 (company synopsis) をペースト」のオプションを提供することができる。ユーザがこの「企業一覧をペースト」を選択すると、そのペースト機能に関連付けられたX S L T変換ファイルは、消費側アプリケーションが、供給側アプリケーションから選択された企業の銘柄記号を、ユーザが選択したペースト機能によって、選択されたデータに関連付けられた利用可能な名前空間からインポートできる企業一覧と共に、ペーストすることを可能にする。この解決策は、スプレッドシート名前空間を選択して、そのスプレッドシートにおいて表現されていたのと同じ状態でそのデータをペーストするもう一方の解決策よりも優れている。

10

【0037】

拡張マークアップ言語に精通している人であれば理解するように、XML名前空間は、XML文書中で使用される要素および属性名を、その要素および属性名をユニフォーム・リソース識別子 (URI) の参照によって識別される名前空間に関連付けることによって、修飾するための方法を提供する。XML名前空間は、名前の集合であって、URI参照により識別され、XML文書中で要素型および属性名として使用される。1つのXML文書は、複数のソフトウェア・モジュールについて定義され、それら複数のソフトウェア・モジュールによって使用される要素および属性を含むことができる。例えば、本発明の一実施形態によれば、ワープロ文書などの1つのXML文書は、異なるソフトウェア・モジュールによって定義され、それら異なるソフトウェア・モジュールによって使用される要素および属性を含むことができる。例えば、ワープロ文書は、HTML処理モジュール、ワープロ・アプリケーションXML処理モジュールについて定義され、それによって使用される要素および属性を備えることができ、あるいは、その文書は、その文書に関連付けられた1つまたは複数のスキーマ・ファイルについて定義され、それらファイルによって使用される要素および属性、または、それらのファイルに関連付けられる要素および属性を含むことができる。例えば、要素および属性をワープロ文書に関連付けることによって、その文書を、レジューム・ドキュメント、リーガル・ドキュメントなどに関連付けられているスキーマ・ファイルに関連付けることができる。したがって、例示のワープロ文書などの個々の文書は、その文書からのデータを消費または使用できる異なるソフトウェア・モジュールのそれぞれに関連付けられた、要素型および属性名を識別する名前空間を備えることができる。上記に挙げた例を続けると、ワープロ文書は、HTML処理モジュールに関連付けられた名前空間、ワープロXML処理モジュールに関連付けられた名前空間、レジューム・ドキュメントまたはリーガル・ドキュメント・スキーマ・ファイルに関連付けられた名前空間を含むことができる。

20

30

【0038】

本発明の一実施形態によれば、第2のアプリケーション文書にペーストするためにデータが選択されると、第2または消費側アプリケーションが、選択されたデータに関連付けられた要素型および／または属性に基づいて、第2のアプリケーションのプログラミングに従って動作することができるように、その選択されたデータに関連付けられた名前空間について第2または消費側アプリケーションに知らせるために、選択されたデータに関連付けられた名前空間を識別する情報が、選択されたデータと共に供給される。例えば、第1のアプリケーションがワープロ・アプリケーションであり、第2のアプリケーションが表計算アプリケーションである場合、表計算アプリケーションは、選択されたデータに関連付けられた名前空間によって識別された情報を利用して、その与えられた名前空間の1つに関連付けられた要素型および属性に関連付けられたX S L T変換ファイルを利用することにより、選択されたデータが第2のアプリケーション文書に最適にペーストされることになると決定する。例えば、所与の名前空間の1つが、選択されたデータに含まれている操作データ (manipulating data) に関連付けられている要素型および属性を識別するが、その他の名前空間が選択されたデータの表示に関連付けられている場合、第2のアプ

40

50

リケーション、この例では表計算アプリケーションは、選択されたデータの表示に関連付けられている名前空間ではなく、選択されたデータの操作 (manipulating) に関連付けられている名前空間を選択することができる。

【0039】

消費側アプリケーションは、第1のアプリケーション文書から選択したXMLデータに関連付けられた名前空間を検出した後、本発明の一実施形態によれば、利用可能な名前空間リストまたは名前空間ライブラリを調べて、第2または消費側アプリケーションが最適に消費できるよう選択されたデータを変換するために消費側アプリケーションが使用できるリソースを見つけることができる。例えば、第2のアプリケーションが、ワープロ・アプリケーションから選択したデータをペーストしようとしている表計算アプリケーションである場合、この第2のアプリケーションは、利用可能な名前空間リストまたは名前空間ライブラリを調べて、第2のアプリケーションが使用できる、XSLT変換ファイルを取得し、表計算アプリケーションが最適に消費できるようそのワープロXMLデータをスプレッドシートXMLデータに変換することができる。当業者であれば理解されるように、拡張スタイルシート言語 (XSL) は、データの書式設定を指定するためのXMLボキャブラリを含む。XSLは、文書を1つのXML文書から別のXML文書に、またはHTML文書などの他のタイプの文書にどのように変換するかを記述するためのXSL情報ファイルを使って、XML文書のスタイルを指定する。

10

【0040】

さらに図2を参照すると、名前空間を、供給側アプリケーションが作成した文書に添付することができる。あるいは、名前空間は、その文書の近くのアクセス可能な名前空間ライブラリ230などの別個の場所に保持することができる。文書は、名前空間を探し出し、かつ／または識別するためのファイル・パス・ポイントまたは一意の名前空間識別子 (例えば、ユニフォーム・リソース識別子またはユニフォーム・リソース名) を含むことができる。名前空間ライブラリ230の詳細については、例えば、本明細書に完全に記載されているものとして参照により組み込まれている、2002年、6月27日に提出された、「System and Method for Providing Namespace Related Information」という名称の米国特許出願第10/184,190を参照のこと。供給側アプリケーション200または消費側アプリケーション220が使用するための名前空間を、名前空間ライブラリ230からダウンロードするための方法の詳細については、例えば、本明細書に完全に記載されているものとして参照により組み込まれている、2002年、6月5日に提出された、「Mechanism for Downloading Software Components from a Remote Source for Use by a Local Software Application」という名称の米国特許出願第10/164,260を参照のこと。

20

30

【0041】

ユーザが、作成した文書で使用するために名前空間およびその関連情報をダウンロードすると、またはその他の方法で取得すると、その名前空間は、すでに導入されているいずれかのその他の名前空間はもちろん、その名前空間およびそのリソース、から成る利用可能な名前空間リスト中にロードされることになる。このリスト225は、ユーザのコンピュータ20上に維持されているか、あるいは、分散コンピューティング・ネットワークを介して操作されるサーバ49など、ユーザがアクセス可能なリモートの記憶域により維持される。利用可能な名前空間リストは、供給側アプリケーション文書中に含まれているデータに関連付けられた名前空間を含むことができ、また名前空間は、消費側アプリケーションによる消費に選択されたデータを変換する消費側アプリケーションを支援するXSLT変換などの、リソースを探し出して取得することを可能にする情報を含むことができる。

40

【0042】

ユーザが、例えば、ワープロの供給側アプリケーション200を使って表形式で作成したデータをカットまたはコピーする場合を考える。データをカットまたはコピーするのに先立って、そのデータは、供給側アプリケーション200のネイティブXMLプログラミ

50

ングに従ってXML注釈を有して構造化される。消費側アプリケーション220が、ペースト操作215によって、カットまたはコピーしたデータをクリップボード210からペーストさせる場合、消費側アプリケーション220は、利用可能な名前空間リスト225を呼び出して、消費側アプリケーション220が、ワープロの供給側アプリケーション200からカットまたはコピーしたデータを、供給側アプリケーションアプリケーション200を使ってそのデータに適用されていたのと同じ書式設定（例えば、表形式）でペーストすることができるように、消費側アプリケーション220のネイティブXMLプログラミングを拡張するために、名前空間が、または名前空間の1つに対応するXSLT変換ファイルが利用可能かどうかを知ることができる。例えば、「wordprocessor-to-spreadsheet-schema.xsl」などのリソース・ファイル（この場合、XSLT変換）を、利用可能なワープロ名前空間から理解できるスプレッドシート名前空間に変換するために消費側アプリケーションによって使用される利用可能なリスト225中に、見つけることができる。したがって、消費側アプリケーションがさらなるXSLT変換を取得すると、消費側アプリケーション（例えば、表計算アプリケーション）は、利用可能な名前空間リストから取得した名前空間変換に従って、供給側アプリケーションから受け取ったデータをそこで使用することができるように、消費側アプリケーションのネイティブXMLプログラミングが拡張される。すなわち、利用可能な名前空間リストからXML名前空間を取得することによって、消費側アプリケーションは、供給側アプリケーションから受け取ったデータを、そのデータが供給側アプリケーションによって作成され、扱われていたのと同じように扱うために必要なXML文法およびデータ規則を受け取る。

10

20

【0043】

本発明の実施形態によれば、所与のXML名前空間に従って、供給側アプリケーションを使ってデータが構造化される場合、ユーザは、消費側アプリケーションを介して、選択したサード・パーティのXSLT変換ファイルに従って、カットまたはコピーしたデータを消費側アプリケーション中にペーストすることができる。例えば、消費側アプリケーション220のユーザが上述の「article」文書からカットしたデータを受け取ると、消費側アプリケーション220のユーザは、利用可能な名前空間リストのリソースを呼び出して、様々な出版社に関連付けられた変換ファイルが特定の出版社の要件に従って消費側アプリケーションで文書を構造化するのに利用可能かどうかを、決定することができる。消費側アプリケーションのユーザは、次いで、消費側アプリケーション220が所望の名前空間に従ってデータを利用することができるようにする変換ファイルを探すために、利用可能な名前空間リスト225のリソースを呼び出す。利用可能な名前空間リスト225中に所望の名前空間が存在しない場合には、ユーザは、ローカルまたはリモートの名前空間ライブラリ230を呼び出して、消費側アプリケーションが供給側アプリケーション200から取得したデータを使用するために利用するよう供給された名前空間を求めてさらなるリソースを探し出す。

30

【0044】

図3は、本発明の例示的实施形態による拡張ペースト機能をユーザに提供するためのユーザ・インターフェースを説明する簡略化した構成図である。消費側アプリケーション220のユーザが消費側アプリケーション220のペースト操作215を選択すると、様々なペースト・オプションがユーザに提供される。例えば、ユーザが、HTML書式などのデフォルト書式に従って、供給側アプリケーションからカットまたはコピーされたデータを、消費側アプリケーション上で使用している文書中にペーストする、単純なペースト操作を選択する場合がある。ユーザが本発明の実施形態による拡張ペースト機能を望む場合には、図3に示す、「形式を選択して貼り付け（paste special）」機能など、代替のペースト機能を選択することができる。

40

【0045】

「形式を選択して貼り付け」ユーザ・インターフェース310は、ユーザが利用可能なペースト・オプションのリストを明示する。例えば、ユーザは、データがリッチ・テキスト書式（RTF）でペーストされるように選択することができ、あるいは文書がHTML

50

書式 3 2 0 に従ってペーストされるように選択することができる。ユーザ・インターフェース 3 1 0 に示すように、ユーザはまた、データが、本発明の拡張機能なしに X M L 書式 3 2 5 に従ってペーストされるように選択することもできる。すなわち、消費側アプリケーション 2 2 0 は、カットまたはコピーされたデータを、消費側アプリケーション 2 2 0 に本来備わっている X M L プログラミングに従って、消費側アプリケーション 2 2 0 によってペーストする。ユーザは、本明細書に記載の何らかの拡張機能に従ってデータをペーストすることを望む場合は、ユーザ・インターフェース 3 1 0 の「その他」ボタン 3 3 0 を選択して、利用可能なスキーマ・ユーザ・インターフェース 3 5 0 を開始することができる。ユーザは、利用可能なスキーマ・ユーザ・インターフェース 3 5 0 を使って、利用可能なリスト 3 5 5 を選択することにより、図 2 を参照しながら上記に説明したように、
10 どの名前空間およびどの関連 X S L T ファイルが利用可能かを利用可能な名前空間リスト 2 2 5 上で調べることができる。したがって、所与の変換ファイルを選択することにより、ユーザが使用している消費側アプリケーションに、供給側アプリケーションからカットまたはコピーされたデータ上において、それらのデータが供給側アプリケーションによって使用され、書式設定されているかのように動作するための、選択された X S L T ファイルの拡張機能が提供される。

【0046】

利用可能な名前空間リストに名前空間またはその他の所望のリソースがない場合には、アプリケーションは、利用可能なデータの大きなリストのビュー（表示）を提供するために、名前空間ライブラリ機能（namespace library feature）3 7 0 を開始することもで
20 ける。この利用可能データのより大きなリストのビューは、上記に図 2 を参照しながら説明したように、ローカルまたはリモートの名前空間ライブラリ 2 3 0 から取得することができる。ユーザがユーザ・インターフェース 3 7 0 を介して名前空間ライブラリから利用可能なビューを選択した場合、選択した名前空間によって利用可能になったペースト機能を含む、更新された利用可能ペースト機能リストを備えた、「形式を選択して貼り付け」ユーザ・インターフェース 3 1 0 がユーザに提示される。あるケースにおけるビューは、名前空間とそれらの関連 X S L T 変換の任意の組合せであってよいことを理解されたい。X S L T 変換は、利用可能な名前空間を、消費側アプリケーションが本来的に理解する名前空間に変換する。異なる名前空間は、通常、異なる型のデータを表し、異なる X S L T
30 は、それら異なる型のデータの異なるビューを提示する。図 3 に示すように、名前空間ライブラリ・ユーザ・インターフェース 3 7 0 から「出版社 A」X S L T 3 8 0 を選択すると、「出版社 A」ペースト機能 3 9 0 が移植された「形式を選択して貼り付け」ユーザ・インターフェース 3 1 0 が作成される。したがって、ユーザは、次いで、ダウンロードした「出版社 A」変換 3 8 0 を適用するために「出版社 A」ペースト機能 3 9 0 を選択して、消費側アプリケーション 2 2 0 が、出版社 A によって提供された X S L T ファイルに従って、供給側アプリケーション 2 0 0 から受け取ったデータをペーストすることができるようにする。この変換は、消費側アプリケーションが理解する名前空間への「出版社 A」名前空間の変換に関連付けられたものとして識別される。当業者によって理解されるように、任意の異なる X S L T ファイル、または、特定の名前空間を変換または解釈する任意のその他の方法も、供給側アプリケーション 2 0 0 から受け取ったカットまたはコピーし
40 たデータに消費側アプリケーション 2 2 0 が拡張ペースト機能を施すために利用できるようにすることができるので、図 2 および図 3 を参照しながらここに記載した特定の X S L T ファイルは例示目的にすぎないことを、当業者は理解されたい。

【0047】

図 4 は、供給側アプリケーションからカットまたはコピーしたデータをペーストまたは消費する、ソフトウェア・アプリケーションのコンピュータ画面表示を示す。画面ショット 4 0 0 は、ユーザが文書 4 1 0 を作成している最中の例示のワープロ・アプリケーションを示す。ユーザは、本発明の実施形態に従って、表の中にスプレッドシート供給側アプリケーション 2 0 0 からコピーしたデータをペーストしたところである。文書 4 1 0 中にデータをペーストするこのペースト操作の間、「E i g h t M o n t h H i s t o r
50

y (8カ月間の履歴)」書式425、「Projected Financial Summary (予測財務要約)」書式430、および「Industry Breakdown (産業別内訳)」書式435を含む、3つの異なるデータ書式をユーザに示すダイアログ・ボックス420がユーザに提示される。ダイアログ・ボックス420は、図3に示すダイアログ・ボックス310の代わりのダイアログ・ボックスであることを理解されたい。図4に示す例示のデータによれば、ユーザは、供給側アプリケーション (例えば、スプレッドシート・アプリケーション) から受け取ったデータを、文書410で、「Eight Month History」のXML構造を指示するXSLTファイルに従って書式設定する、利用可能な名前空間リスト225中の名前空間の1つのビューである「Eight Month History」書式425を選択している。「eightmonthhistory-schema.xsl」などの変換リソースは、利用可能な名前空間リスト上において、または、図4に示す「表1」書式を提供するためのXML構造を消費側アプリケーションがサポートできるようにする、名前空間ライブラリ225中において、入手可能であることを理解されたい。ユーザが、供給側アプリケーションから「Industry Breakdown」書式に関連する情報をカットおよびコピーすることを選択した場合には、供給側アプリケーションからカットまたはコピーしたデータが、消費側アプリケーションにおいて、供給側アプリケーションによって構造化されたときの書式に従って書式設定されるように、「Industry Breakdown」書式435を選択することができる。

【0048】

本発明の一実施形態によれば、データがXML書式でカットまたはコピーされ、その後消費側アプリケーション220がペーストできるようにクリップボード210に保存されると、本発明の機能に従ってカットまたはコピーされたデータをどのように処理するかを消費側アプリケーションが決定するのを支援するために、3つまたはそれ以上のクリップボード書式を利用することができる。この3つの書式には、CF__XML__HEADER、CF__XML__VIEW、およびCF__XML__DATAが含まれる。上述のように、供給側アプリケーション200および消費側アプリケーション220を含む多くのアプリケーションは、それらのアプリケーションが作成したXMLマークアップ構造およびファイルをサポートする。しかし、上述のように、あるアプリケーションによるXMLサポートが、別のアプリケーションによるXMLサポートと互換性がないことが多く、その結果、あるアプリケーションのデータが消費側アプリケーションのファイルにコピーされるときに、書式設定およびその他のXMLに関連する特徴 (features) が失われてしまう。さらに、上述のように、様々なアプリケーションが使用できるようにサード・パーティから提供されているが、アプリケーションのネイティブXMLプログラミングがサポートしていないXMLスキーマも存在する。

【0049】

CF__XML__HEADER書式 (「HEADER」) は、コピー・アプリケーションがどの名前空間を利用可能かに関する情報を含む。全てのペースト・アプリケーションがCF__XML__HEADERを調べて、消費できる1つまたは複数の名前空間があるかどうかを決定することができる。HEADERは、ペースト・アプリケーションが消費したい名前空間または複数名前空間を決定した際に、コールバックして、所望のデータを得ることができるように、個々の名前空間をどこで見つけることができるかについても明示する。HEADERが所望の名前空間を含んでいると明示する書式にCF__XML__DATA書式がある。この情報は、消費側アプリケーションにとっては役立つ。その理由は、消費側アプリケーションがどの書式を消費したいかを決定することができるからである。このことは、コピー・アプリケーションが、特定の書式が要求されるまでデータの全てを異なる書式で供給する必要がないので、パフォーマンスという点を改善する。この説明の残りの部分では、2種類の利用可能な書式があると仮定する。

【0050】

第1のクリップボード書式は、CF__XML__VIEWであり、これは、1つまたは複

数の名前空間を備え、データ・マークアップ（「spreadsheetMLまたはdocumentML」など）というよりは、通常では、より多くの書式設定マークアップの
である。第2のクリップボード書式は、書式設定またはビューを定義しないが、代わりに
表面下でデータを定義する、1つまたは複数の名前空間を含む、CF__XML__DATA
である。これら2種類の書式の動作は、表計算プログラムを含む例を使用することによっ
て容易に見ることができる。その表計算プログラムがXML構造をサポートする場合には
、財務データを定義する名前空間をそのスプレッドシートに適用することにより、様々な
セル、およびそれらがどのように財務データに関連しているかを識別することができると
推測可能である。表計算アプリケーションがコピー操作を実行する場合、そのデータを提
示するのに2通りの方法がある。第1の方法は、スプレッドシート中に現れている通りに 10
それを提示し、この場合、スプレッドシート・マークアップ言語が使用される。スプレッ
ドシート・マークアップ言語は、書式設定、レイアウトなど、スプレッドシートに見られ
る一般的な事柄を定義する。このデータは、CF__XML__VIEWクリップボード書式
上に置かれる。この理由は、データがどのように「見える」か、をこの書式が最もよく定
義するからである。これは、内在するデータの意味よりも、見栄えやレイアウトに関心を
持つ人がデータをペーストする場合には、役立つ場合がある。

【0051】

クリップボード上のデータを提示するための第2の方法は、財務データの名前空間を使
用する方法である。この名前空間は、データの見栄えやレイアウトを含む情報は備えてい
ず、代わりに、データそれ自体が何を意味するかが記述されている。この場合、このデー 20
タは、CF__XML__DATAクリップボード書式上に置かれる。レイアウト情報が利用
不可能なため、消費側アプリケーションは、このデータを最もよく表示する方法を自分で
決定する必要がある。例えば、ワープロ・アプリケーション中にペーストする場合には、
この方法が、そのより望ましい書式である可能性がある。その理由は、財務データがスプ
レッドシートにおいて見えている状態は、それがワープロ・アプリケーションにおいて現
れる状態とは完全に異なる場合があるためである。この場合、ワープロ・アプリケーショ
ンは、財務データの名前空間を調べて、次に、そのデータを正しく表示するための支援と
なるXSLT変換など、何らかのビュー情報を見つける必要がある。

【0052】

異なる名前空間を異なるクリップボード書式に分割することによって、消費側アプリケ
ーションが供給側アプリケーションからの「VIEW」または「DATA」書式のいずれ
かまたは両方を必要としなくなる場合があるため、より効率的である。すなわち、消費側
アプリケーションは、ユーザが選択したペースト機能に基づいて、「VIEW」または「
DATA」書式のいずれかに含まれる情報が必要ではないと決定した場合、消費側アプリ
ケーションは、それ自体が使用するためにこれらの書式からデータをロードすることはない。
消費側アプリケーションが供給側アプリケーションと同じタイプのアプリケーション
（例えば、ワープロ・アプリケーション）の場合、消費側アプリケーションは、ユーザが
選択したペースト操作を実行するために追加ペースト機能を必要としない場合がある。C
F__XML__HEADER書式は、「DATA」および「VIEW」書式の両方について 30
、バージョンおよび名前空間に関する情報を含む。このヘッダ書式は、DATAおよびV
IEW書式ファイルそれぞれのサイズに関する情報も含む。以下は、サード・パーティの
「GET A Job. com」のXMLスキーマでマーク付けしたワープロ・ファイル
からコピーしたデータ・フラグメントに関連付けられたCF__XML__HEADER書式
の例である。

【0053】

【表 2】

```

<cfh:header cfh:Version="1.0" xmlns:cfh="urn:schemas-microsoft-
com:office:clipboard:header">
  <cfh:data cfh:bytes="00000001343">
    <cfh:Item cfh:name="D1" cfh:bytes="00000001343">
      <cfh:URI cfh:value="urn:schemas-getajob-com:resume"
cfh:root="yes"/>
    </cfh:Item>
  </cfh:data>
  <cfh:view cfh:bytes="00000003612">
    <cfh:Item cfh:name="V1" cfh:bytes="00000003612">
      <cfh:URI cfh:value="urn:schemas-microsoft-
com:office:word.1.0" cfh:root="yes"/>
      <cfh:URI cfh:value="urn:schemas-microsoft-
com:schemaLibrary"/>
      <cfh:URI cfh:value="urn:schemas-microsoft-com:AML"/>
    </cfh:Item>
  </cfh:view>
</cfh:header>

```

10

20

【0054】

30

CF__XML__VIEWデータ書式（「VIEW」）は、供給側アプリケーションのネイティブXMLプログラミングに関する情報を消費側アプリケーションに提供する。「VIEW」書式中に含まれているデータを読むことによって、消費側アプリケーションは、ペースト操作中に追加XSLT変換ファイルを取得する必要なく、それ自身のネイティブXMLプログラミングに従って供給側アプリケーションのネイティブXMLプログラミングを完全に利用することができるかどうかを決定することができる。以下は、供給側アプリケーションからカットまたはコピーしたデータに関連付けることができるCF__XML__VIEWヘッダ書式の例である。

```

<cb:view xmlns:cb="urn:schemas-microsoft-com:office:clipboard:view"><!--Root element for the ClipBoard-->

```

40

```

  <cb:item cb:name="V0" cb:URI="The URI of the root schema">

```

注記：一実施形態によれば、ここにデータがある場合もある。そのデータは、cb:startタグの前にあるため、消費側アプリケーションは、そのデータがコピーされたものの一部ではないことを知る。それでも、そのデータは、ソース全体の性質に関する興味ある情報を提供する場合がある。

```

  <cb:start/><!--フラグメントの開始を意味して、ラインに置かれる1つのタグ-->

```

注記：一実施形態によれば、ここにデータがある場合もある。それは、cb:startタグの後にあるため、消費側アプリケーションは、そのデータがコピーされたものの一部であることを知る。

```

  <cb:end/><!--フラグメントの終わりを意味して、ラインに置かれる1つのタグ-->

```

50

注記：一実施形態によれば、ここにデータがある場合もある。そのデータは `cb:end` タグの後にあるため、消費側アプリケーションは、そのデータがコピーされたものの一部ではないことを知る。それでも、そのデータは、ソース全体の性質に関する興味ある情報を提供する場合がある。

```
</cb:item>
```

```
<!-- ペースト・アプリケーションが決めるならば、1つ以上を置くことが可能： -->
```

```
<cb:item cb:name="V1" cb:URI="The URI of the root schema">
```

```
<cb:start/><!--フラグメントの開始を意味しているラインに置かれるシングルの  
タグ-->
```

10

```
<cb:end/> <!--フラグメントの終わりを意味して、ラインに置かれる1つのタグ-->
```

```
</cb:item>
```

```
</ cb:view >
```

【0055】

C F _ X M L _ D A T A 書式（「D A T A」）は、供給側アプリケーションからカットまたはコピーしたデータに関連付けられているXML名前空間に関する情報を消費側アプリケーションに提供する。この場合、それらのXML名前空間は、供給側アプリケーションのネイティブXMLプログラミングの一部ではない。例えば、「D A T A」ヘッダは、供給側アプリケーションのユーザが作成したスキーマに関連付けられているXML名前空間に関する情報を、コピー・アプリケーションに提供することができる。あるいは、上述のように、ユーザが消費または提示するためにデータを構造化する際に使用する出版社または証券会社のスキーマなどの、サード・パーティのスキーマに関連付けられたXML名前空間に関する情報を提供することができる。以下は、C F _ X M L _ D A T A 書式の例であり、消費側アプリケーションはこの書式を受け取ることが可能であり、この書式は、供給側アプリケーションによってカットまたはコピーされたデータに関連付けられている非ネイティブXML名前空間に関する情報を消費側アプリケーションに供給する。

20

```
<cb:data xmlns:cb="urn:schemas-microsoft-com:office:clipboard:data"><!--Root e  
lement for the Clipboard-->
```

```
<cb:item cb:name="D0" cb:URI="The URI of the root schema">
```

30

注記：ここにデータがある場合もある。`cb:start` タグの前にあるため、消費側アプリケーションは、そのデータがコピーされたものの一部ではないことを知る。それでも、そのデータは、ソース全体の性質に関する興味ある情報を提供する場合がある。

```
<cb:start/><!--フラグメントの開始を意味して、ラインに置かれる1つのタグ-->
```

注記：ここにデータがある場合もある。`cb:start` タグの後にあるため、消費側アプリケーションはそのデータがコピーされたものの一部であることを知る。

```
<cb:end/> <!--フラグメントの終わりを意味して、ラインに置かれる1つのタグ-->
```

注記：ここにデータがある場合もある。`cb:end` タグの後にあるため、消費側アプリケーションは、そのデータがコピーされたものの一部ではないことを知る。それでも、そのデータは、ソース全体の性質に関する興味ある情報を提供する場合がある。

40

```
</cb:item >
```

```
<!-- ペースト・アプリケーションが決めるならば、1つ以上を置くことが可能： -->
```

```
<cb:item cb:name="D1" cb:URI="The URI of the root schema">
```

```
<cb:start/><!--フラグメントの開始を意味しているラインに置かれるシングルの  
タグ-->
```

```
<cb:end/><!--フラグメントの終わりを意味して、ラインに置かれる1つのタグ-->
```

```
</cb:item>
```

```
</cb:data>
```

50

【0056】

図5および6は、本発明の実施形態による、消費側アプリケーションのペースト機能を拡張するための方法を示す流れ図である。方法500は、開始ステップ505で開始し、ステップ510に移動して、供給側アプリケーションからデータがカットまたはコピーされ、その後の消費側アプリケーション220によるペーストに備えて、クリップボード210に保存される。ステップ515で、ユーザは、カットまたはコピーしたデータを消費側アプリケーションで使用している文書中にペーストするために、消費側アプリケーション220からペーストを選択する。ステップ520で、消費側アプリケーションは、CF__XML__HEADER書式がそのカットまたはコピーされたデータに関連付けられているかどうかを決定する。どのCF__XML__HEADER書式も利用可能ではない場合には、方法は図6のステップ575に進み、そこで、データは拡張ペースト機能なしにペーストされる。上述のように、追加XMLスキーマおよび／または構造を消費側アプリケーションが利用可能な場合には、CF__XML__HEADER書式によって情報が消費側アプリケーションに提供されることを理解されたい。

【0057】

ステップ520で「HEADER」書式が提供された場合には、方法はステップ525に進み、そこで、データがCF__XML__VIEW書式を含んでいるかどうかに関する決定がなされる。YESの場合には、方法はステップ530に進み、そこで、消費側アプリケーションは、カットまたはコピーされたデータをペーストするのに利用可能なものとしてヘッダ中に指定されている利用可能な名前空間のリストを作成する。例えば、供給側アプリケーションが表計算アプリケーションの場合には、消費側アプリケーションは、名前空間ライブラリを呼び出して、消費側アプリケーション（例えば、ワープロ）が供給側アプリケーションのXML機能に従ってカットまたはコピーされたデータをペーストするのを支援するための利用可能な変換ファイル、または、その他のリソース、を探す必要がある場合がある。名前空間ライブラリを呼び出す理由は、消費側アプリケーションが利用可能な名前空間を理解していないこと、その結果それをどのように取り扱うかに関する追加情報を見つける必要があること、である。利用可能な名前空間リスト中にXML名前空間がない場合には、方法はステップ550に進み、そこで消費側アプリケーションが名前空間ライブラリ230を調べて、供給側アプリケーションから受け取ったカットまたはコピーしたデータをペーストする際の支援となり得る追加リソースを探す。

【0058】

ステップ535で、CF__XML__DATA書式が提供された場合には、方法はステップ540に進む。そこで、消費側アプリケーションが利用可能な名前空間リストを呼び出して、上述のように、供給側アプリケーションにおいて作成された名前空間に関連付けられている、あるいは、供給側アプリケーションの外部のサード・パーティ・ソースから受け取った、変換ファイルおよび／またはその他のリソースを取得する。ステップ530および540を参照しながら説明した利用可能な名前空間リストの呼出しは、図3を参照しながら上記に説明したように、ユーザ・インターフェースを提供することにより実現され得る。

【0059】

図6のステップ545で、消費側アプリケーションが、供給側アプリケーションからカットまたはコピーされたデータについて識別された、名前空間および関連変換ファイルを理解するかどうかに関する決定がなされる。例えば、消費側アプリケーションがワープロ・アプリケーションで、供給側アプリケーションもワープロ・アプリケーションの場合には、消費側アプリケーションは、カットまたはコピーされたデータに関連付けられたXML書式設定を提供するために変換ファイルの形態で追加リソースを必要としない場合がある。その場合、方法はステップ575に進み、そこでカットまたはコピーされたデータは、消費側アプリケーションによってペースト機能拡張なしにペーストされる。

【0060】

消費側アプリケーションが選択されたデータに関連付けられた名前空間および関連リソ

ースを理解しない場合、あるいは、名前空間またはリソースが消費側アプリケーションにとって利用可能ではない場合には、方法はステップ550に進む。そこで、ユーザが供給側アプリケーションから受け取ったCFXMLDATA書式に含まれるデータに関連付けられた、追加の名前空間およびリソースを選択するのに、名前空間ライブラリ230が利用される。供給側アプリケーションから受け取った「DATA」書式が、カットまたはコピーされたデータに関連付けられた、所定の出版社のXSLTファイルなど、サード・パーティの名前空間を識別する場合には、識別された名前空間に従ってデータをペーストするために消費側アプリケーションが必要とするXSLTファイルが、消費側アプリケーションのために名前空間ライブラリ230から取得されることが可能になる。ステップ555で、消費側アプリケーションが使用するためのいずれかの利用可能な変換ファイルが、ユーザによって選択されたかどうかに関する決定がなされる。選択されていない場合には、方法はステップ560に進み、そこで、図3に示すように、ユーザ・インターフェース310のペースト機能のところにデフォルトの機能が移植される。

【0061】

利用可能な名前空間リストまたは名前空間ライブラリのいずれかにおいて、XSLT変換ファイルなどのリソースが探し出された場合には、方法はステップ565に進む。そこで、図3を参照しながら上記に説明したように、ペースト・ユーザ・インターフェース310に、ユーザが選択した追加機能が移植される。ステップ570で、ユーザがデフォルトの一般的なペースト機能を選択したか、あるいは「形式を選択して貼り付け」機能を選択したかに関する決定がなされる。「形式を選択して貼り付け」機能をユーザが望む場合には、方法はステップ580に進む。そこで、ユーザは、所望の「形式を選択して貼り付け」機能を選択する。方法はステップ575に進み、そこで、ユーザが選択したペースト機能に従ってデータがペーストされる。ステップ570で、「形式を選択して貼り付け」機能をユーザが望まない場合には、方法はステップ575に進み、そこで、データは、ユーザが選択したように、拡張ペースト機能なしにペーストされる。ステップ595で、方法は終了する。

【0062】

本明細書に記載しているように、コンピュータが生成した文書中にデータをペーストするためのコンピュータ・ソフトウェア・アプリケーションが利用可能なペースト機能を拡張するための、方法およびシステムが提供される。当業者には、本発明の範囲または趣旨を逸脱することなく、本発明に様々な修正または変更を施すことができることが明らかであろう。当業者には、本明細書を読むことにより、また、本明細書に開示している本発明を実施することによって、本発明のその他の実施形態が明らかになるだろう。

【図面の簡単な説明】

【0063】

【図1】本発明のための例示的動作環境を提供する、コンピュータ、および関連する周辺装置およびネットワーク化されたデバイスの構成図である。

【図2】データが供給側アプリケーションを使ってカットまたはコピーされ、消費側アプリケーションを使ってペーストされる、供給側アプリケーションと消費側アプリケーションの間の相互作用を説明する、簡略化した構成図である。

【図3】本発明の例示的实施形態による拡張ペースト機能をユーザに提供するためのユーザ・インターフェースを説明する、簡略化した構成図である。

【図4】供給側アプリケーションからカットまたはコピーしたデータをペーストおよび消費するソフトウェア・アプリケーションの、コンピュータ画面表示を説明する図である。

【図5】本発明の一実施形態による、消費側アプリケーションのペースト機能を拡張するための方法を説明する流れ図である。

【図6】本発明の一実施形態による、消費側アプリケーションのペースト機能を拡張するための方法を説明する流れ図である。

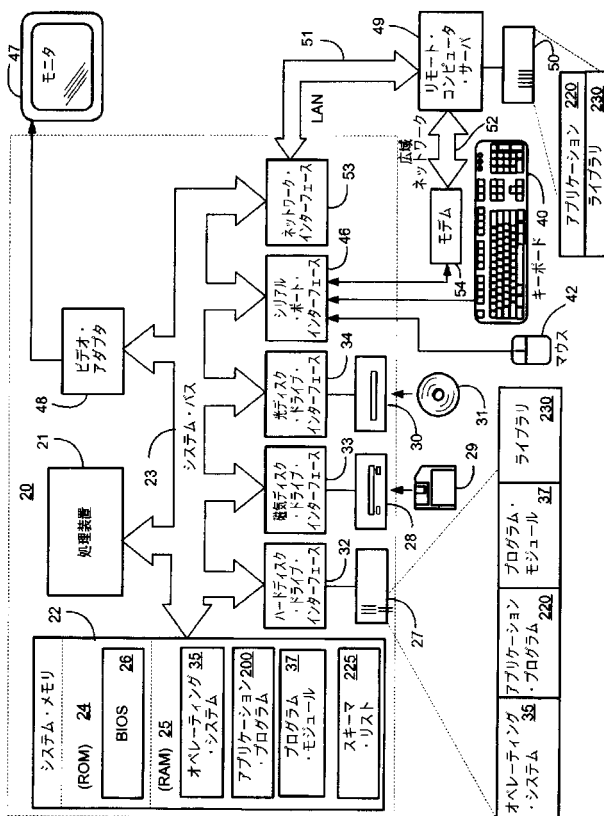
【符号の説明】

【0064】

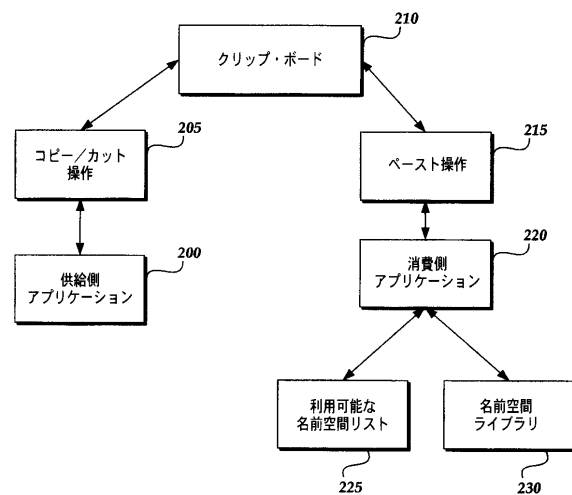
2 0	パーソナル・コンピュータ、ユーザのコンピュータ	
2 1	処理装置	
2 2	システム・メモリ	
2 3	システム・バス	
2 4	読取り専用メモリ (ROM)	
2 5	ランダム・アクセス・メモリ (RAM)	
2 6	基本入出力システム (BIOS)	
2 7	ハードディスク・ドライブ	
2 8	磁気ディスク・ドライブ	
2 9	リムーバブル・ディスク	10
3 0	光ディスク・ドライブ	
3 1	CD-ROMディスク	
3 2	ハードディスク・ドライブ・インターフェース	
3 3	磁気ディスク・ドライブ・インターフェース	
3 4	光ドライブ・インターフェース	
3 5	オペレーティング・システム	
4 0	キーボード	
4 2	マウス	
4 6	シリアル・ポート・インターフェース	
4 7	モニタ	20
4 8	ビデオ・アダプタ	
4 9	リモート・コンピュータ、サーバ	
5 0	記憶装置、メモリ	
5 1	ローカル・エリア・ネットワーク (LAN)	
5 2	広域ネットワーク (WAN)	
5 3	ネットワーク・インターフェース	
5 4	モデム	
2 0 0	アプリケーション・プログラム、供給側アプリケーション	
2 0 5	カットまたはコピー操作、カット／コピー操作	
2 1 0	記憶域、クリップボード	30
2 1 5	ペースト操作	
2 2 0	アプリケーション、アプリケーション・プログラム、消費側アプリケーション	
、表計算消費側アプリケーション		
2 2 5	利用可能な名前空間リスト、利用可能なリスト	
2 3 0	名前空間ライブラリ	
3 1 0	「形式を選択して貼り付け」ユーザ・インターフェース、ユーザ・インターフェース	
3 1 6	ダイアログ・ボックス	
3 2 0	HTML書式	
3 2 5	拡張機能なしにXML書式	40
3 3 0	「その他」ボタン	
3 5 0	利用可能なスキーマ・ユーザ・インターフェース	
3 5 5	利用可能なリスト	
3 7 0	名前空間ライブラリ機能、名前空間ライブラリ・ユーザ・インターフェース、ユーザ・インターフェース	
3 8 0	「出版社A」XSLT、「出版社A」変換	
3 9 0	「出版社A」ペースト機能	
4 0 0	画面ショット	
4 1 0	文書	
4 2 0	ダイアログ・ボックス	50

- 4 2 5 「8カ月間の履歴」書式
 4 3 0 「予測財務要約」書式
 4 3 5 「産業別内訳」書式

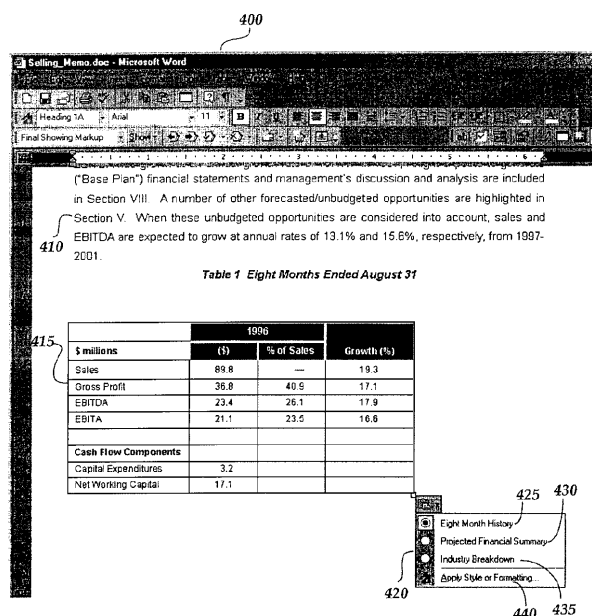
【図 1】



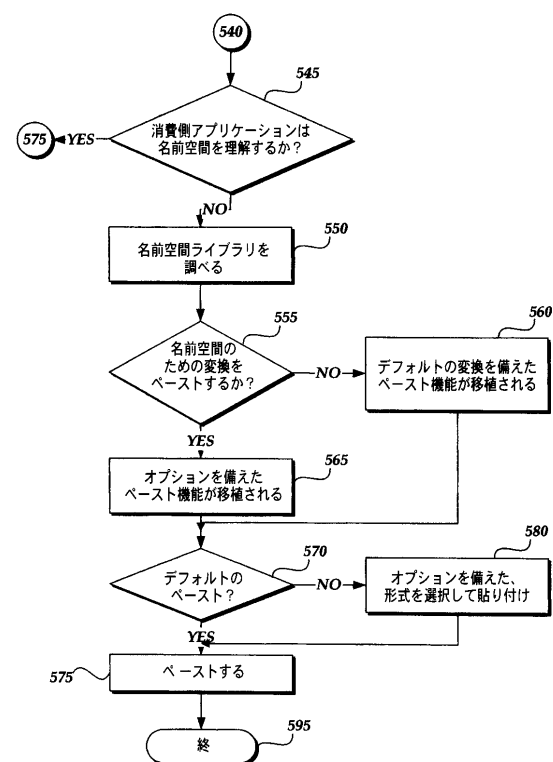
【図 2】



【图 4】



【图 6】



フロントページの続き

- (72)発明者 マーシン サウィッキ
アメリカ合衆国 98034 ワシントン州 カークランド 102 レーン 13110 ユニ
ット 6
- (72)発明者 ロバート エー. リトル
アメリカ合衆国 98053 ワシントン州 レッドモンド ノースイースト 117 ストリー
ト 21007
- (72)発明者 マイケル マレー
アメリカ合衆国 98112 ワシントン州 シアトル 16 アベニュー イースト 1505
- (72)発明者 ロバート コリー
アメリカ合衆国 98033 ワシントン州 カークランド ノースイースト 128 プレイス
11033
- (72)発明者 マイケル タッカー
アメリカ合衆国 98005 ワシントン州 ベルビュー ノースイースト 35 プレイス 1
2804

Fターム(参考) 5B009 NA03 NA07 QA06 QB12 TA11

【要約の続き】

【外国語明細書】

**METHOD AND SYSTEM FOR ENHANCING PASTE FUNCTIONALITY
OF A COMPUTER SOFTWARE APPLICATION**

Field of the Invention

This invention relates to methods and systems for enhancing the paste functionality available to a computer software application for pasting data into a computer-generated document.

Background of the Invention

Computer software applications allow users to create a variety of documents to assist them in work, education, and leisure. For example, word processing applications allow users to create letters, articles, books, memoranda, and the like. Spreadsheet applications allow users to store, manipulate, print, and display a variety of alpha-numeric data. Such applications have a number of well-known strengths, including rich editing, formatting, printing, and calculation.

A very common and useful functionality of many software applications is the ability to cut or copy data from a given document, spreadsheet, slide presentation or other computer-generated document followed by a pasting operation where the cut or copied data is pasted to a desired location in the document. Often, a user cuts or copies data including text, images, or alpha-numeric data from a first or providing application, such as a word processing application, and then pastes the cut or copied data using a second or consuming application, such as a spreadsheet application. For example, a user may desire to cut or copy an array of numbers from a spreadsheet application document in order to paste those numbers into a word processing application document where the user is preparing a letter to send to a client or associate. Typically, when data is cut or copied, the data is buffered in a memory location for subsequent pasting during the pasting operation.

When data is cut or copied from a providing application, and is then pasted into a separate document by a second consuming application where the providing application and the consuming application are different software applications, often many, if not all, of the features provided by the first application, including formatting, are lost during the pasting operation because the second or consuming application does not include the functionality necessary for providing special features, including formatting, that were provided by the first application.

For example, if the user copies an array of data from a spreadsheet application document and then pastes that data into a word processing document, the data pasted into the word processing document may be pasted without any of the formatting that was present in the spreadsheet application. That is, if the data in the spreadsheet application was set out in a number of columns and rows, the data may be pasted into the word processing application as a simple sequence of numbers without any formatting.

To keep up with demand for more advanced functionality in software applications, software developers have begun to use markup languages, such as the Extensible Markup Language to allow users to annotate a software application document to give the document a useful structure apart from the normal functionality of the software application responsible for creating the document or the visible formatting associated with the document. For example, a user may wish to create on her word processing application a template document for preparation of an article that she wishes to transmit to a publisher. By applying structure to the document, a publisher receiving the document may make use of the structure by processing the document to utilize data defined by the document structure. Unfortunately, when data is cut or copied from a document having such structural annotation, often the structural annotation is lost in the pasting operation, particularly when the pasting operation is performed by a different consuming application. Even if the consuming application is programmed to understand and use the markup language, the schema, or rules defining the structural annotation of the markup language, may be significantly different for the providing application as opposed to the consuming application.

It is with respect to these and other considerations that the present invention has been made.

Summary of the Invention

The present invention provides methods and systems for enhancing the paste functionality available to a computer software application for pasting data into a computer-generated document. Generally described, data is selected from a first application for pasting to a second application document. The selected data along with information associated with the data such as a list of namespaces is saved

to a memory location. When the selected data is pasted to the second application or consuming application document, the consuming application checks an available list of namespaces associated with the selected data to see if any of the various representations of the data are understood by the pasting application. The pasting application may also look to see if any resources such as an Extensible Stylesheet Language (XSL) transformation files for transforming the selected data to a format that is more readily consumable by the second or consuming application are available. One method for finding resources to assist with the available namespaces is for the consuming application to check a namespace library for an available resource for transforming the data for use by the consuming application. If no resources are found, and the namespaces are not already natively understood by the pasting application the data may be pasted to the second application document according to a default paste resource. The pasting application may also have a default method for dealing with any type of data regardless of the namespace.

More particularly, a method of enhancing the paste functionality of a computer software application is provided. A first document is opened via a providing application and structure is applied to the first document according to a markup language, such as Extensible Markup Language (XML). It could also be that the first application is only serving the purpose of displaying existing data in a meaningful way. Data is selected from the first application for pasting to a second application. The selected data is saved to a memory location, and information associated with the selected data is saved to the memory location for providing the consuming application information about the selected data. This information primarily consists of a list of namespaces. These namespaces identify the different ways the copying application has for representing the data. An example of this would be a spreadsheet document that has financial data. One of the namespaces provided may be for describing a spreadsheet, and the other namespace provided may be for describing financial data.

A second document is opened via a consuming application and a paste function is selected at the consuming application for pasting the selected data to the second document. The consuming application reads the information associated with the selected data. In response, the consuming application determines whether one or more namespaces that identify data types associated with the

selected data are natively understood by the consuming application. If there are more than one namespaces, the application may either give the user a choice of which namespace and associated resources to use, or the application may choose the namespace and associated resources that best fits the data already in the consuming application document. The choice presented may be a choice of which namespace to use, or it may be a choice of which resource for any one given namespace to use. The consuming application may obtain one or more resource files, such as XSLT transformation files associated with namespaces for pasting the selected data to the second document if the consuming application does not already understand that namespace. After the consuming application selects one of the one or more namespaces for pasting the selected data to the second document, the selected data is pasted to the second document according to the method the consuming application prefers. The additional resource sometimes used may be an Extensible Stylesheet Language Transformation (XSLT) file for transforming the selected data into a data type or format consumable by the second or consuming application.

According to one aspect of the invention, prior to obtaining a resource for the one or more namespaces for pasting the selecting data, a determination is made as to whether an available namespace list contains a desired namespace of the one or more namespaces understood by the consuming application. If the available namespace list contains a desired namespace of the one or more namespaces understood by the consuming application, the desired namespace is selected and provided to the consuming application. If the available namespace list does not contain a desired namespace of the one or more namespaces understood by the consuming application, a determination is made as to whether a namespace library contains a resource for assisting the consuming application with understanding the one or more namespaces for use by the consuming application. If the namespace library contains a resource for the one or more namespaces for use by the consuming application, the desired resource is selected from the namespace library and is provided to the consuming application to assist it with consuming the one or more namespaces.

The information associated with the selected data is saved to a memory location for providing to the consuming application and may include an indication as to the presence of one or more namespaces associated with the selected

data. For each namespace associated with the selected data, the information associated with the selected data may include information about any associated namespace including a version identifier for the namespace, a uniform resource identifier for locating the namespace, and a file size for the namespace.

According to another aspect of the invention, a method for pasting data from a copying application to a consuming application is provided. A first Extensible Markup Language (XML) element is applied to a beginning of a region of a first document. The region is selected for copying from the copying application and for pasting to the consuming application. A second XML element is applied to an end of the selected region. Information is provided in a header associated with the selected region. The information includes an identification of available XML namespaces and pointers to a clipboard format identification that corresponds to each of the available namespaces. Information is provided in the header on a file size for each the available namespaces. One or more namespaces of the available namespaces is provided on a clipboard to allow the consuming application to choose among the one or more namespaces for pasting the data from the copying application.

These and other features and advantages, which characterize the present invention will be apparent from a reading of the following detailed description and a review of the associated drawings. It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as claimed.

Brief Description of the Drawings

Fig. 1 is a block diagram of a computer and associated peripheral and networked devices that provide an exemplary operating environment for the present invention.

Fig. 2 is a simplified block diagram illustrating the interaction between a providing application and a consuming application where data is cut or copied using the providing application and pasted using the consuming application.

Fig. 3 is a simplified block diagram illustrating user interfaces for providing users enhanced paste functionality according to an exemplary embodiment of the present invention.

Fig. 4 illustrates a computer screen display of a software application pasting and consuming data cut or copied from a providing application.

Figs. 5 and 6 are flow charts illustrating a method for enhancing the paste functionality of a consuming application according to an embodiment of the present invention.

Detailed Description of the Preferred Embodiment

The following description of embodiments of the present invention is made with reference to the above-described drawings wherein like numerals refer to like parts or components throughout the several figures. The present invention is directed to methods and systems for enhancing the paste functionality available to a computer software application for pasting data into a computer-generated document. A user selects data from a first application, such as a spreadsheet application, for pasting to a second application, such as a word processing application. The pasting process may be in the form of a traditional cut/copy and paste operation or may be in the form of dragging and dropping selected data from a first application document to a second application document. According to an embodiment of the present invention, the first or providing application writes the selected data in an Extensible Markup Language (XML) representation to a memory location, such as a clipboard. The selected data is written to the memory location in a format that includes the selected data and information associated with one or more data namespaces associated with the selected data. Available namespaces associated with the selected data may identify the types of data formatting that may be associated with the selected data. For example, identified available namespaces may include Hypertext Markup Language (HTML), resume document, or word processor XML. The HTML namespace might identify that the data may be structured as HTML. The resume document namespace might identify that the data may be structured according to a resume XML schema file. The word processor XML namespace might identify that the data may be formatted according to a word processor version of XML. Of course, these namespaces are by way of example only of numerous types namespaces that may be associated with the selected data.

When the selected data is pasted to the second application document by selecting the second application paste functionality or by dropping the selected

data onto the second application document, the second or consuming application detects from the information provided with the selected data that XML formatted data is being pasted. Based on the information provided with the selected data, the consuming application detects all of the available namespaces associated with the data to be pasted. For example, from the foregoing example, the consuming application may detect that the selected data has associated namespaces for HTML, resume document, and word processor XML. In response, the consuming application looks to a list of available namespaces for a resource to help the consuming application paste the selected data. For example, if the providing application is a spreadsheet application, the consuming application may look to the list of available namespaces for an Extensible Stylesheet Language Transformation (XSLT) file for transforming the spreadsheet XML formatted data into a format for consumption and use by the consuming application. If the consuming application is a word processor, the consuming application may look for an XSLT transformation file for transforming the spreadsheet data into data for consumption by the word processing application. For example, if the data was structured in the spreadsheet application in a format comprised of two rows and three columns, the XSLT transformation file used by the consuming application may transform that data for consumption by the word processing application so that the data will be maintained in a format of two rows and three columns. Without the functionality of the present invention, the data may be pasted according to the default paste functionality of the consuming application and thus may be pasted without the formatting of the providing application. Consequently, the data may be pasted to the word processor application document, according to the present example, as raw data presented from left to right across the word processor work space as opposed to being formatted in two rows and three columns.

If multiple available namespaces are identified to the consuming application, the consuming application may select the available namespace that most likely provides the consuming application with the desired pasting functionality. Or, a user interface may be provided to the user of the consuming application to allow the user to select among the available transformation files that the application is aware of for the list of available namespaces. If no resources are found by the consuming application in the list of available namespaces, the consuming

application may go to a namespace library in search of an available resource for assisting the consuming application in transforming the selected data prior to pasting the data to the consuming application document. If multiple resources are located in the namespace library, a user interface may be provided for allowing the user to choose among available resources. If no available resources are located in either the list of available namespaces or the namespace library, then the consuming application pastes the data according to the default functionality, as described above.

Operating Environment

Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of an application program that runs on an operating system in conjunction with a personal computer, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, cell phones, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to Fig. 1, an exemplary system for implementing the invention includes a conventional personal computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that couples the system memory to the processing unit 21. The system memory 22 includes read-only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 further includes a hard disk drive 27, a

magnetic disk drive 28, e.g., to read from or write to a removable disk 29, and an optical disk drive 30, e.g., for reading a CD-ROM disk 31 or to read from or write to other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide non-volatile storage for the personal computer 20. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD-ROM disk, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 25, including an operating system 35, one or more application programs 200, 220, program data, such as the available namespace list 225, and other program modules (not shown).

A user may enter commands and information into the personal computer 20 through a keyboard 40 and pointing device, such as a mouse 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers or printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the personal computer 20, although only a memory storage device 50 has been illustrated in Fig. 1. The server 49 and memory 50 may hold application programs such as the application 220 and data storage such as the

namespace library 230. The logical connections depicted in Fig. 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the LAN 51 through a network interface 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the WAN 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Operation

Fig. 2 is a simplified block diagram illustrating the interaction between a providing application and a consuming application where data is cut or copied using the providing application and pasted using the consuming application. As shown in Fig. 2, a providing application 200 is illustrated from which cut or copied data is saved for use by a consuming application 220. A providing application may be any software application from which data may be cut or copied for later pasting, including a word processing application, a spreadsheet application, a slide presentation application, and the like. Once a user selects data using the providing application 200, a cut or copy operation 205 is performed which may include cutting the selected data from the providing document or may include copying the selected data from the providing document. Once the selected data is cut or copied, that data is transferred to a memory location 210, such as the clipboard 210 illustrated in Fig. 2. The memory location 210 holds the cut or copied data for subsequent pasting by the original providing application 200 or by the consuming application 220 as selected by the user.

According to an embodiment of the present invention, the clipboard 210 is illustrative of a memory resource maintained by windowing operating

systems. The clipboard stores a copy of the data that is copied or cut, and a subsequent paste operation passes the data from the clipboard 210 to the consuming program. As is well known to those skilled in the art, the clipboard 210 allows data to be transferred from one application to another, as described above, provided that the second or consuming application 220 can read the data generated by the providing application 200. In order to ensure that the consuming application can read, understand, and utilize the data cut or copied by the providing application 200, typically, the data cut or copied by the providing application is saved to the clipboard 210 in a generic format, such as Hypertext Markup Language format, so that the data is most likely in a format that may be consumed by a number and variety of different consuming applications 220. Other types of formatting include rich-text formatting, or the data may be saved as unformatted data. All of these formats are advertised on the clipboard when the first application copies. It is then up to the pasting application to decide which of the available formats it would like to request.

Once the consuming application 220 is launched for pasting the cut or copied data from the providing application 200, a paste operation 215 is selected to paste the cut or copied data a desired location into a document operated by the consuming application. As should be understood, the cut/copy operation 205 and the paste operation 215 may be combined in the form of a drag and drop operation where selected data is dragged from a providing application document and is dropped into a consuming application document. If the cut or copied data was formatted by the providing application according to a number of generic formatting types, as described above, the consuming application may be able to paste the cut or copied data, as desired, but many of the features provided to the data by the providing application, including specialized formatting, is often lost. For example, if the user cuts or copies data entered in tabular form in a word processing providing application 200 and subsequently pastes that data into a spreadsheet consuming application 220, the data originally formatted in tabular form may be simply inserted into a single cell within the spreadsheet rendering the spreadsheet application unable to operate on the data as desired by the user of the spreadsheet consuming application 220.

With the advent of markup languages, such as the Extensible Markup Language (XML) often documents prepared by the providing application 200 have been annotated with markup language structure in order to give the document useful structure for managing, presenting and manipulating data contained in the document. As is well known to those skilled in the art, different software applications include different “native” programming for reading, understanding, and utilizing a markup languages such as XML. That is, a providing application 200 may be programmed to provide different or more XML structure and functionality as compared to a consuming application 220. Accordingly, if data is cut or copied from an XML-formatted document created by a providing application 200 for consumption by a different consuming application 220, much of the structure and functionality associated with the XML formatting of the cut or copied data may be lost when the consuming application 220 pastes the data into a document created and operated by the consuming application 220.

The following is a sample XML structure that may be applied to an article written by a user with a word processing providing application 200.

Sample XML Structure

```
<article>
  <title>EXTRA EXTRA</title>
  <body>
    <summary>Today in New York . . .</summary>
    <main> Today in New York . . .</main>
  </body>
  *
  *
  *
</article>
```

As shown in the sample XML structure, a number of XML elements are illustrated annotating the “article” document. For example, the <article> tag is included at the beginning and end of the document to define the document as a “article” document, and a variety of other elements such as a <title> element, and

<body> element are included inside the “article” structure. Inside the <body> element are two child elements, <summary> and <main>, of the <body> element. The XML annotation of the document allows the user to define portions of the document for certain types of data and data structure. For example, the user may define the portion of the document inside the <title> element for including data of the type “title” and including a prescribed allowable structure for the title.

In order to provide the document with a set of grammatical and data type rules governing the types and structure of data that may be included in a given document such as the “article” document illustrated above, an XML schema is attached to or associated with the document for providing the rules governing each of the XML elements and tags with which the user may annotate the given document. For example, the “article” document may have an attached or associated schema such as “article-schema.xsd” for providing the allowable set of XML elements such as the <article> element, <title> element, <body> element, and so on. The schema includes the rules governing the order with which those elements may be applied to the document and specific rules associated with individual elements applied to the document. For example, a schema attached or associated with the “article” document may prescribe that data associated with a given element, for example a <date> element (not shown above), must include a day element, followed by a month element, followed by a year element. Additionally, the schema may require that a <date> element must immediately precede the <title> element.

As is understood by those skilled in the art, developers of XML schemas determine the names of XML elements and the associated data types and data structures allowed for those elements. Then, all users of documents annotated with XML structure according to a given schema may utilize the data contained within the XML structure without regard to the overall type and structure of the document. For example, if the “article” document, illustrated above, is transmitted to a publisher of the document, the publisher may develop software applications for parsing the document to locate specific types of data within the document for use by the publisher. The publisher may, for example only wish to publish the title of the article as an advertisement for the future publication of the whole article. The way in which XML is identified is through a namespace. The namespace provides an identification (ID) such that any consumer of the XML will know what type of XML

they are consuming. By observing the namespace, the consuming application will know what schema was used to create the XML file. Using the namespace specified in the document, the publisher will know that the data associated with the XML element <title> has been prepared according to the schema governing the document. Accordingly, the publisher may develop a software application or an XSLT transformation file for locating the <title> element and for extracting the data associated therewith for insertion into the publisher's own document (including formatting it in some special way) for using that data as an advertisement for the future publication of the whole article.

Following with this example, a number of different publishers may subscribe to the same namespace for dictating the rules associated with the "article" document so that each publisher may then receive the "article" document from the author of the document and use the data contained in the article according to the XML elements structuring the data. That is, a first publishing company may only be interested in the data contained within the <summary> element, while a second publisher may be interested in extracting only the data contained in the <title> element. Each publisher may extract the data it desires without regard to other aspects or elements of the document by using their own software applications or XSLT transformation files for locating the desired data according to the XML structure. This is made possible by the fact that each user of the document follows the data type and data structure rules prescribed in the namespace attached to or associated with the document.

If the consuming application does find an XSLT transformation file for use with one of the available namespaces then the selected data is converted by the XSLT transformation file and the results are recieved by the consuming application as part of the paste function. It may also be the case that the consuming application is aware of a transform file, and it consumes the available namespace and performs the transform on that data itself. For example, as described above, if the providing application is a spreadsheet application and the selected data was formatted in two rows and three columns, then the XSLT transformation file for transforming the spreadsheet data for consumption by a word processing application, for example, may allow the data to be transformed so that the word processing application may paste the data as two lists with three items in each as opposed to

raw data without formatting or the table form used in the spreadsheet. As should be understood, the first or providing application document may be marked up according to a custom XML schema. For example, the data may be marked up according to a "stock price schema." In that case, if the selected data is copied for pasting from a spreadsheet application into a word processing application, for example, the namespace associated with the "stock price schema" as well as the namespace associated with the visual layout of the spreadsheet will be associated with the selected data when the selected data is written to memory for pasting into the word processing application.

Upon recognizing the "stock price schema", the consuming application references the list of available resources and detects that each cell of the selected data is a company stock symbol, and that for each company, there is an associated description for the company. Accordingly, the consuming application will locate a resource such as an XSLT transformation file in the list of available namespaces or namespace library to instruct the consuming application on how to paste the marked up data. The user interface provided to the user of the consuming application may provide the option to "paste company synopsis." If the user selects the "paste company synopsis", the XSLT transformation file associated with that paste function may allow the consuming application to paste the company stock symbols selected from the providing application along with a company synopsis that may be imported via the paste function selected by the user from the available namespaces associated with the selected data. This solution is better than the alternative of choosing the spreadsheet namespace and pasting the data in the same way it was represented in the spreadsheet.

As understood by those familiar with the Extensible Markup Language, XML namespaces provide a method for qualifying elements and attribute names used in XML documents by associating those elements and attribute names with namespaces identified by uniform resources identifier (URI) references. XML namespaces are collections of names, identified by URI references, that are used in XML documents as element types and attribute names. A single XML document may contain elements and attributes that are defined for and used by multiple software modules. For example, in accordance with an embodiment of the present invention, a single XML document, such as a word processing document, may

contain elements and attributes defined and used by different software modules. For example, a word processing document may have elements and attributes defined for and used by an HTML processing module, a word processing application XML processing module, or the document may contain elements and attributes defined for and used by or associated with one or more schema files associated with the document. For example, elements and attributes may be associated with the word processing document to associate the document with a schema file associated with a resume document, a legal document, and the like. Accordingly, an individual document, such as the exemplary word processing document may have a namespace identifying the element types and attribute names associated with each of the different software modules that may consume or use data from the document. Following from the examples listed above, the word processing document may contain a namespace associated with the HTML processing module, a namespace associated with the word processing XML processing module, and a namespace associated with the resume or legal document schema files.

According to an embodiment of the present invention, when data is selected for pasting to a second application document, information identifying namespaces associated with the selected data is provided with the selected data for informing the second or consuming application of the namespaces associated with the selected data so that the second or consuming application may operate on the element types and/or attributes associated with the selected data according to the programming of the second application. For example, if the first application is a word processing application and the second application is a spreadsheet application, the spreadsheet application may utilize information identified by the namespaces associated with the selected data to determine that the selected data will be optimally pasted to the second application document by utilizing an XSLT transformation file associated with the element types and attributes associated with one of the given namespaces. For example, if one of the given namespaces identifies element types and attributes associated with manipulating data contained in the selected data, but other namespaces are associated with displaying the selected data, the second application, spreadsheet application according to this example, may select the namespace associated with manipulating the data of the selected data, as opposed to namespaces associated with displaying the data.

After the consuming application detects the namespaces associated with the XML data selected from the first application document, according to an embodiment of the present invention, the consuming application may look to a list of available namespaces or to a namespace library to find resources that may be used by the consuming application for transforming the selected data for optimum consumption by the second or consuming application. For example, if the second application is a spreadsheet application attempting to paste data selected from a word processing application, the second application may look to a list of available namespaces or a namespace library to obtain an XSLT transformation file that may be used by the second application for transforming the word processor XML data into spreadsheet XML data for optimum consumption by the spreadsheet application. As is understood by those skilled in the art, the Extensible Stylesheet Language (XSL) includes an XML vocabulary for specifying formatting of data. XSL specifies the styling of an XML document by using XSL information files to describe how a document is transformed from one XML document into another XML document or other type of document such as an HTML document.

Referring still to Fig. 2, namespaces may be attached to the document prepared by the providing application, or a namespace may be maintained in a separate location such as a namespace library 230 accessible by the document. The document may contain a file path pointer or unique namespace identifier (e.g., uniform resource identifier or uniform resource name) for locating and/or identifying the namespace. For a detailed description of the namespace library 230, see U.S. Patent Application entitled "System and Method for Providing Namespace Related Information", Serial No. 10/184,190, filed June 27, 2002, which is incorporated herein by reference as if fully set out herein. For a detailed description of a method for downloading a namespace from the namespace library 230 for use by a providing application 200 or consuming application 220, see U.S. Patent Application entitled "Mechanism for Downloading Software Components from a Remote Source for Use by a Local Software Application," Serial No. 10/164,260, filed June 5, 2002, which is incorporated herein by reference as if fully set out herein.

Once the user downloads or otherwise obtains a namespace and its related information for use with documents created by the user, that namespace may be loaded into an available namespace list consisting of that namespace and its

resources, as well as any other namespace that has already been installed. That list 225 maintained on the user's computer 20 or accessible to the user from a remote storage location such as a server 49 operated via a distributed computing network. The available namespace list may include namespaces associated with data contained in the providing application document and the namespaces may include information allowing the consuming application to locate and obtain the resources, such as XSLT transformation for assisting the consuming application with transforming the selected data for consumption by the consuming application.

Consider for example that a user cuts or copies data prepared in tabular form using a word processing providing application 200. Prior to cutting or copying the data, the data is structured with XML annotations according to the native XML programming of the providing application 200. If the consuming application 220 via the paste operation 215 is caused to paste the cut or copied data from the clipboard 210, the consuming application 220 may call upon the available namespace list 225 to see if a namespace or appropriate XSLT transformation file for one of the namespaces is available for enhancing the native XML programming of the consuming application 220 so that the consuming application 220 may paste the data cut or copied from the word processing providing application 200 in the same formatting (for example, tabular format) as was applied to that data using the providing application 200. For example, a resource file (XSLT transformation in this case) such as "wordprocessor-to-spreadsheet-schema.xsl" may be found in the available list 225 for use by the consuming application to transform from the available word processor namespace into the understood spreadsheet namespace. Accordingly, when the consuming application obtains the additional XSLT transformation, the consuming application's native XML programming is enhanced to allow the consuming application (for example, a spreadsheet application) to now use the data received from the providing application according to the namespace transform obtained from the available namespace list. That is, by obtaining the XML namespace from the available namespace list, the consuming application receives the XML grammatical and data rules required for treating the data received from the providing application in the same manner as that data was created and treated by the providing application.

According to embodiments of the present invention, if data is structured using the providing application according to a given XML namespace, the user via the consuming application may paste the cut or copied data into the consuming application according to a selected third party XSLT transformation file. For example, if a user of a consuming application 220 receives data cut from the "article" document described above, the user of the consuming application 220 may call upon the resources of the available namespace list to determine whether transformation files associated with a variety of publishers are available for structuring the document in the consuming application according to the requirements of a particular publisher. The user of the consuming application then may call upon the resources of the available namespace list 225 for a transformation file that will allow the consuming application 220 to utilize the data according to a desired namespace. If the desired namespace is not present in the available namespace list 225, the user may call upon the local or remote namespace library 230 to locate additional resources for the namespaces provided for use by the consuming application for using the data obtained from the providing application 200.

Fig. 3 is a simplified block diagram illustrating user interfaces for providing users enhanced paste functionality according to an exemplary embodiment of the present invention. Once the user of the consuming application 220 selects the paste operation 215 of the consuming application 220, a variety of paste options may be provided to the user. For example, the user may select a simple paste operation that will paste the data cut or copied from the providing application into the document being used on the consuming application according to a default formatting, such as HTML formatting. If the user desires enhanced paste functionality according to the embodiments of the present invention, the user can select an alternate paste function such as the paste special function illustrated in Fig. 3.

The paste special user interface 310 illustrates a list of pasting options available to the user. For example, the user may select that the data be pasted in rich-text formatting (RTF), or the user may select that the document be pasted according to an HTML format 320. As illustrated in the user interface 310, the user may also select that data be pasted according to an XML format 325 without the enhanced functionality of the present invention. That is, the cut or copied data is pasted by the consuming application 220 according to the XML programming that is

native to the consuming application 220. If the user desires to paste data according to some enhanced functionality, as described herein, the user may select the “other” button 330 of the user interface 310 to launch an available schema user interface 350. With the available schema user interface 350, the user may select the available list 355 to check to see what namespaces and associated XSLT files are available on the available namespace list 225, as described above with reference to Fig. 2. Accordingly, selection of a given transformation file provides the consuming application in use by the user the enhanced functionality of the selected XSLT file for operating on the data cut or copied from the providing application as that data would have been used or formatted by the providing application.

If the available namespace list does not have namespaces or other resources desired, the application may also launch the namespace library feature 370 in order to provide a larger list of available data views that may be obtained from the local or remote namespace library 230, as described above with reference to Fig. 2. If the user selects an available view from the namespace library 230 via the user interface 370, the paste special user interface 310 is presented to the user with an updated list of available paste functions including the paste function enabled by the selected namespace. As should be understood, a view in one case may be any combination of namespaces and their associated XSLT transformations. The XSLT transformation will transform the available namespace into a namespace natively understood by the consuming application. The different namespaces usually represent different types of data, and the different XSLTs represent different views on those different types of data. As shown in Fig. 3, selection of the “Publisher A” XSLT 380 from the namespace library user interface 370 causes the paste special user interface 310 to be populated with a “Publisher A” paste function 390. Accordingly, the user may then select the “Publisher A” paste function 390 in order to apply the downloaded “Publisher A” transformation 380 for allowing the consuming application 220 to paste the data received from the providing application 200 according to that XSLT file provided by the publisher A. The transformation is identified as being associated with converting the “Publisher A” namespace into the namespace the consuming application understands. As should be understood by those skilled in the art, the particular XSLT files described herein with reference to Figs. 2 and 3 are for purposes of example only as any different XSLT files or any

other method of converting or interpreting a particular namespace may be made available to the consuming application 220 for providing enhanced paste functionality to the cut or copied data received from the providing application 200.

Fig. 4 illustrates a computer screen display of a software application pasting and consuming data cut or copied from a providing application. The screen shot 400 shows an exemplary word processing application where a user is preparing a document 410. According to an embodiment of the present invention, the user has pasted data into a table that was copied from a spreadsheet providing application 200. During the paste operation to paste the data into the document 410, a dialog box 420 is presented to the user showing the user three different data formats including the “Eight Month History” format 425, the “Projected Financial Summary” format 430 and the “Industry Breakdown” format 435. As should be understood, the dialog box 420 is an alternative dialog box to the dialog box 310 illustrated in Fig. 3. According to the example data shown in Fig. 4, the user has selected the “Eight Month History” format 425 which is a view of one of the namespaces in the available namespace list 225 to cause the data received from the providing application (for example, spreadsheet application) to be formatted in the document 410 according to an XSLT file dictating the XML structure for an “Eight Month History.” As should be understood, a transform resource such as “eightmonthhistory-schema.xsl” may be available on the available namespace list or in the namespace library 225 that allows the consuming application to support an XML structure for providing the “Table 1” formatting, illustrated in Fig. 4. If the user had chosen to cut and copy information related to the “Industry Breakdown” format, from the providing application, the user may then select the “Industry Breakdown” format 435 so that the data cut or copied from the providing application will be formatted in the consuming application according to the format under which that data was structured by the providing application.

According to an embodiment of the present invention, when data is cut or copied in an XML format and is saved to the clipboard 210 for subsequent pasting by a consuming application 220, three or more clipboard formats can be utilized to assist the consuming application in determining how to process the cut or copied data according to the functionality of the present invention. The three formats include the CF_XML_HEADER, CF_XML_VIEW, and CF_XML_DATA.

As described above, many applications including the providing application 200 and the consuming application 220 support XML markup structure and files created by those applications. However, as described above, often the XML support of one application is incompatible with the XML support of another application, and consequently, formatting and other XML related features are lost when data from one application is copied to files of a consuming application. Additionally, as described above, XML schemas also exist as provided by third parties for use by a variety of applications but that are not supported by the native XML programming of the application.

The CF_XML_HEADER format ("HEADER") contains information on what namespace the copying application has available. The CF_XML_HEADER is where all pasting applications can look to decide if there is one or more namespaces that they are able to consume. The HEADER also identifies where the different namespaces can be found, so that the pasting application, upon deciding on the namespace or namespaces it wants to consume, can then call back to get the desired data. On place the HEADER may identify as containing the desired namespace(s) is the CF_XML_DATA format. The information is useful to the consuming application, because the consuming application may be able to determine which formats it would like to consume. This can help with performance issues since the copying application does not need to provide all the data in the different formats until those specific formats are requested. For the rest of this description, we will assume that there are two formats available.

The first clipboard format is CF_XML_VIEW that has one or more namespace(s) and is usually more of a formatting markup rather than a data markup (such as "spreadsheetML; or documentML"). The second clipboard format is CF_XML_DATA with contains one or more namespaces that do not define the formatting or view, but instead define the data underneath. These two formats in action can easily be seen by using an example involving a spreadsheet program. If that spreadsheet program supports XML structures, then it could be imagined that a namespace defining financial data could be applied to this spreadsheet to identify the different cells and how they relate to financial data. When the spreadsheet application performs a copy operation, it will have two ways of presenting the data. The first way it to present it as it appears in the spreadsheet, and that would use the

spreadsheet markup language. The spreadsheet markup language defines the general things seen in a spreadsheet such as formatting, layout, etc. This data would be placed on the CF_XML_VIEW clipboard format since it best defines how the data “looks.” That may be useful to the person pasting the data if they are more concerned with the look and the layout than they are with the underlying data’s meaning.

The second way to present the data on the clipboard is to use the financial data’s namespace. This namespace has no information involving the look or layout of the data, but instead describes what the data itself means. This data would be placed on the CF_XML_DATA clipboard format. The consuming application would need to decide for itself how to best display this data, since the layout information is not available. In the case of pasting into a word processing application, for example, this may be the more desired format. The reason for this is that the way the financial data looks while in a spreadsheet may be completely different than how it should appear in a word processor application. The word processor application would need to look at the financial data namespace, and then find some view information, such as an XSLT transformation to help it display the data in the proper way.

By splitting the different namespaces out into different clipboard formats, it is much more efficient, since the consuming application may not need either or both of the “VIEW” or “DATA” formats from the providing application. That is, if based on the paste function selected by the user, the consuming application determines that it does not need the information contained in either the “VIEW” or “DATA” formats, the consuming application will not load data from those formats for use by the consuming application. If the consuming application is the same type application (for example, word processing application) as the providing application, the consuming application may not require additional paste functionality to perform the paste operation selected by the user. The CF_XML_HEADER format contains information on the version and namespace for both the “DATA” and the “VIEW” formats. The header format also includes information on the size of each of the DATA and VIEW format files. An example of CF_XML_HEADER format associated with a data fragment copied from a word

processing file marked up with a third party “Get A Job.com” XML schema is as follows.

```
<cfh:header cfh:Version="1.0" xmlns:cfh="urn:schemas-microsoft-
com:office:clipboard:header">
  <cfh:data cfh:bytes="000000001343">
    <cfh:Item cfh:name="D1" cfh:bytes="000000001343">
      <cfh:URI cfh:value="urn:schemas-getajob-com:resume"
cfh:root="yes"/>
    </cfh:Item>
  </cfh:data>
  <cfh:view cfh:bytes="000000003612">
    <cfh:Item cfh:name="V1" cfh:bytes="000000003612">
      <cfh:URI cfh:value="urn:schemas-microsoft-
com:office:word.1.0" cfh:root="yes"/>
      <cfh:URI cfh:value="urn:schemas-microsoft-
com:schemaLibrary"/>
      <cfh:URI cfh:value="urn:schemas-microsoft-com:AML"/>
    </cfh:Item>
  </cfh:view>
</cfh:header>
```

The CF_XML_VIEW data format (“VIEW”) provides information to the consuming application as to the native XML programming of the providing application. By reading the data contained in the “VIEW” format, the consuming application may determine whether it is capable according to its own native XML programming to fully utilize the native XML programming of the providing application without the need to obtain additional XSLT transformation files during the pasting operation. The following is an example CF_XML_VIEW header format that may be associated with data cut or copied from a providing application.

```
<cb:view xmlns:cb="urn:schemas-microsoft-com:office:clipboard:view"> <!--Root
element for the ClipBoard-->
```

<cb:item cb:name="V0" cb:URI="The URI of the root schema">

Note: According to one embodiment, there may be data here. Since this data is before the cb:start tag, the consuming application knows that this data is not part of what was copied, but it still may provide interesting information on the nature of the entire source.

< cb:start /> <!--A single tag to be placed inline signifying the start of the fragment-->

Note: According to one embodiment, there may be data here.

Since it is after the cb:start tag, the consuming application knows that this data is part of what was copied.

< cb:end /> <!--A single tag to be placed inline signifying the end of the fragment-->

Note: According to one embodiment, there may be data here.

Since this data is after the cb:end tag, the consuming application knows that this data is not part of what was copied, but it still may provide interesting information on the nature of the entire source.

</cb:item >

<!-- It is possible to place more than one if the pasting app decides to: -->

<cb:item cb:name="V1" cb:URI="The URI of the root schema">

< cb:start /> <!--A single tag to be placed inline signifying the start of the fragment-->

< cb:end /> <!--A single tag to be placed inline signifying the end of the fragment-->

</cb:item >

</ cb:view >

The CF_XML_DATA format ("DATA") provides information to the consuming application about XML namespaces associated with the cut or copied data from the providing application where those XML namespaces are not a part of the native XML programming of the providing application. For example, the "DATA" header may provide information to the copying application concerning an XML namespaces associated with a schema created by the user of the providing application, or information may be provided regarding an XML namespace

associated with a third party schema, such as a publisher or stock market company schema for use in structuring data for consumption or presentation by users, as described above. The following is an example of a CF_XML_DATA format that may be received by the consuming application to provide the consuming application with information regarding non-native XML namespaces associated with data cut or copied by the providing application.

```
<cb:data xmlns:cb="urn:schemas-microsoft-com:office:clipboard:data"> <!--Root
element for the ClipBoard-->
```

```
<cb:item cb:name="D0" cb:URI="The URI of the root schema">
```

Note: there may be data here. Since this data is before the cb:start tag, the consuming application knows that this data is not part of what was copied, but it still may provide interesting information on the nature of the entire source.

```
< cb:start /> <!--A single tag to be placed inline signifying the start of
the fragment-->
```

Note: there may be data here. Since it is after the cb:start tag, the consuming application knows that this data is part of what was copied.

```
< cb:end /> <!--A single tag to be placed inline signifying the end of
the fragment-->
```

Note: there may be data here. Since this data is after the cb:end tag, the consuming application knows that this data is not part of what was copied, but it still may provide interesting information on the nature of the entire source.

```
</cb:item >
```

```
<!-- It is possible to place more than one if the pasting app decides to: -->
```

```
<cb:item cb:name="D1" cb:URI="The URI of the root schema">
```

```
< cb:start /> <!--A single tag to be placed inline signifying the start of
the fragment-->
```

```
< cb:end /> <!--A single tag to be placed inline signifying the end of
the fragment-->
```

```
</cb:item >
```

</cb:data >

Figs. 5 and 6 are flow charts illustrating a method for enhancing the paste functionality of a consuming application according to an embodiment of the present invention. The method 500 begins at start step 505 and moves to step 510 where data from a providing application is cut or copied and saved to the clipboard 210 for subsequent pasting by a consuming application 220. At step 515, a user selects paste from a consuming application 220 for pasting the cut or copied data into a document in use with the consuming application. At step 520, the consuming application determines whether a CF_XML_HEADER format is associated with the cut or copied data. If no CF_XML_HEADER format is available, the method proceeds to step 575, Fig. 6, and the data is pasted without any enhanced paste functionality. As should be understood, as described above, if additional XML schemas and/or data structures are available to the consuming application, that information is provided to the consuming application via the CF_XML_HEADER format.

If the "HEADER" format is provided at step 520, the method proceeds to step 525 and a determination is made as to whether the data includes a CF_XML_VIEW format. If yes, the method proceeds to step 530 and the consuming application creates a list of the available namespaces specified in the header as being available for pasting the cut or copied data. For example, if the providing application is a spreadsheet application, the consuming application may need to call the namespace library for an available transformation file or other resource for assisting the consuming application (e.g., word processor) in pasting the cut or copied data according to the XML functionality of the providing application. The reason for the call to the namespace library is that the consuming application does not understand the available namespace, and needs to find additional information on how to handle it. If the XML namespaces are not in the available namespace list, the method proceeds to step 550 and the consuming application checks the namespace library 230 for additional resources that may be of assistance in pasting the cut or copied data received from the providing application.

If at step 535 a CF_XML_DATA format is provided, the method proceeds to step 540 and the consuming application calls the available namespace

list to obtain transformation files and/or other resources associated with namespaces created at the providing application or received from a third party source external to the providing application, as described above. As should be understood, calling the available namespace list as described with reference to steps 530 and 540 may be accomplished by providing a user interface, as described above with reference to Fig. 3.

At step 545, Fig. 6, a determination is made as to whether the consuming application understands the namespaces and associated transformation files identified for the data cut or copied from the providing application. For example, if the consuming application is a word processing application and the providing application is a word processing application, the consuming application may not require any additional resources in the form of transformation files in order to provide the XML formatting associated with the cut or copied data. If so, the method proceeds to step 575, and the cut or copied data is pasted by the consuming application without paste functionality enhancement.

If the consuming application does not understand the namespaces and associated resources associated with the selected data, or if no namespaces or resources are available to the consuming application, the method proceeds to step 550 and the namespace library 230 is made available to the user to select additional namespaces and resources associated with the data contained in the CF_XML_DATA format received from the providing application. If the "DATA" format received from the providing application identifies a third party namespace associated with the cut or copied data, such as the XSLT file of a given publisher, an XSLT file required by the consuming application for pasting the data according to the identified namespace may be obtained for the consuming application from the namespace library 230. At step 555, a determination is made as to whether any available transformation files are selected by the user for use by the consuming application. If not, the method proceeds to step 560 and the paste functions of the user interface 310 are populated with the default functions, as illustrated in Fig. 3.

If resources, such as XSLT transformation files, are located in either the available namespace list or the namespace library, the method proceeds to step 565 and the paste user interface 310 is populated with the additional functionality selected by the user, as described above with reference to Fig. 3. At step 570, a

determination is made as to whether the user selects a default generic paste function or whether the user selects a paste special function. If the user desires a paste special function, the method proceeds to step 580, and the user selects the desired paste special function. The method proceeds to step 575 where the data is pasted according to the paste function selected by the user. If at step 570, if the user does not desire to select a paste special function, the method proceeds to step 575 and the data is pasted without enhanced paste functionality as selected by the user. The method ends at step 595.

As described herein, methods and systems are provided for enhancing the paste functionality available to a computer software application for pasting data into a computer-generated document. It will be apparent to those skilled in the art that various modifications or variations may be made in the present invention without departing from the scope or spirit of the invention. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein.

1. A method of enhancing the paste functionality of a computer software application, comprising:

- opening a first document via a providing application;
- selecting data from the first document for pasting to a second document via a consuming application;
- saving the selected data to a memory location;
- saving data associated with the selected data to the memory location for providing the consuming application information about the selected data;
- if the data associated with the selected data identifies a first namespace associated with the selected data identifying one or more data types associated with the selected data, providing the first namespace to the consuming application;
- selecting one of the one or more data types for pasting the selected data to the second document; and
- obtaining a resource associated with the selected data type for preparing the data for pasting to the second document by the consuming application.

2. The method of Claim 1, prior to the step of obtaining a resource associated with the selected data type for preparing the data for pasting to the second document by the consuming application, parsing an available namespace list for the resources associated with the selected data; and if the available namespace list identifies the resource associated with the selected data, providing the resource to the consuming application.

3. The method of Claim 2, whereby if the available namespace list does not identify the resource associated with the selected data, parsing a namespace library for the resource associated with the selected data; and

if the namespace library identifies the resource associated with the selected data, providing the resource to the consuming application.

4. The method of Claim 3, whereby if a plurality of namespaces are identified by the data associated with the selected data, obtaining a second resource associated with one of the plurality of namespaces for preparing the data for pasting to the second document by the consuming application.

5. The method of Claim 4, further comprising providing a user selectable choice between pasting the selected data to the second document according to the first resource and pasting the selected data to the second document according to the second resource.

6. The method of Claim 4, further comprising providing a user selectable choice for pasting the selected data to the second document according one of one or more resources selected from one of a plurality of available namespaces where each of the plurality of available namespaces is associated with one or more resources.

7. The method of Claim 5, whereby saving the selected data to a memory location includes saving the selected data in an Extensible Markup Language (XML) format.

8. The method of Claim 7, whereby the first and second resources include Extensible Stylesheet Language Transformations (XSLT) for transforming the selected data from a first data type associated with the first document to a second data type for pasting the selected data to the second document.

9. The method of Claim 8, whereby if no resource is associated with the selected data type for preparing the data for pasting to the second document by the consuming application, pasting the selected data to the second document without enhanced paste functionality.

10. The method of Claim 9, whereby saving data associated with the selected data to the memory location includes saving the data to the memory location in a CF_XML_HEADER format.

11. The method of Claim 10, whereby if the first namespace is identified as associated with the first document, then providing with the data associated with the selected data information for allowing the consuming application to locate the first namespace in the available namespace list or namespace library.

12. The method of Claim 11, whereby providing with the data associated with the selected data information for allowing the consuming application to locate the first namespace in the available namespace list or namespace library includes providing a version identifier for the first namespace, providing a uniform resource identifier for the first namespace, and providing a file size for the first namespace.

13. The method of Claim 12, whereby if the plurality of namespaces is identified as associated with the first document, then providing with the data associated with the selected data information for allowing the consuming application to locate the plurality of namespaces in the available namespace list or namespace library.

14. The method of Claim 13, whereby providing with the data associated with the selected data information for allowing the consuming application to locate the plurality of namespaces in the available namespace list or namespace library includes providing a version identifier for each of the plurality of namespaces, providing a uniform resource identifier for each of the plurality of namespaces, and providing a file size for each of the plurality of namespaces.

15. The method of Claim 10, whereby if the first namespace is identified as associated with the first document, providing the consuming application any data types associated with the first document according to the first namespace.

16. The method of Claim 15, whereby the data types associated with the first document according to the first namespace are provided to the consuming application via a CF_XML_View format.

17. The method of Claim 13, whereby if the plurality of namespaces is identified as associated with the first document, providing the consuming application one or more data types associated with the first document according to the plurality of namespaces.

18. The method of Claim 17, whereby the one or more data types associated with the first document according to the plurality of namespaces are provided to the consuming application via a CF_XML_Data format.

19. The method of Claim 1, whereby the providing application is a word processing application.

20. The method of Claim 1, whereby the providing application is a spreadsheet application.

21. The method of Claim 1, whereby the providing application is a slide presentation application.

22. The method of Claim 1, whereby the consuming application is a word processing application.

23. The method of Claim 1, whereby the consuming application is a spreadsheet application.

24. The method of Claim 1, whereby the consuming application is a slide presentation application.

25. The method of Claim 1, whereby saving the selected data to a memory location includes saving the selected data to a clipboard.

26. A method of enhancing the paste functionality of a computer software application, comprising:

opening a first document via a providing application;

applying structure to the first document according to a markup language;

selecting data from the first document for pasting to a second document;

saving the selected data to a memory location;

saving data associated with the selected data to the memory location for providing a consuming application information about the selected data;

opening a second document via the consuming application and selecting a consuming application paste function for pasting the selected data to the second document;

reading by the consuming application the data associated with the selected data;

in response to reading by the consuming application the data associated with the selected data, determining whether one or more namespaces are associated with the selected data defining permissible data content, data type and data structure for structure applied to the selected data;

obtaining by the consuming application a resource associated with one of the one or more namespaces for pasting the selected data to the second document; and

pasting the selected data to the second document according to the resource.

27. The method of Claim 26, prior to obtaining by the consuming application a resource associated with one of the one or more namespaces for pasting the selected data,

determining whether an available namespace list contains a desired namespace of the one or more namespaces for use by the consuming application;

if the available namespace list contains a desired namespace of the one or more namespaces for use by the consuming application, selecting the desired namespace; and

providing the selected namespace to the consuming application.

28. The method of Claim 27, whereby if the available namespace list does not contain a desired namespace of the one or more namespaces for use by the consuming application, determining whether a namespace library contains one or more resources for one of the one or more namespaces for use by the consuming application;

if the namespace library contains one or more desired resources for the one or more namespaces for use by the consuming application, selecting the desired resource from the namespace library; and

providing the selected namespace and resource to the consuming application.

29. The method of Claim 28, further comprising providing a user selectable choice for pasting the selected data to the second document according to the selected namespace.

30. The method of Claim 29, whereby the markup language is the Extensible Markup Language (XML).

31. The method of Claim 30, whereby the resource is an Extensible Stylesheet Language Transformation file for transforming the selected data from a first data type associated with the first document to a second data type for pasting the selected data to the second document.

32. The method of Claim 29, whereby the step of saving data associated with the selected data to the memory location for providing the consuming application information about the selected data includes identifying whether any of the one or more namespaces is associated with the first document.

33. The method of Claim 32, whereby identifying whether any of the one or more namespaces is associated with the first document includes providing the consuming application information for locating any of the one or more namespaces on the available namespace list or at the namespace library.

34. A computer readable medium having stored thereon computer-executable instructions, which when performed by a computer, perform the steps of:

- opening a first document via a providing application;
- applying structure to the first document according to a markup language;
- selecting data from the first document for pasting to a second document;
- saving the selected data to a memory location;
- saving data associated with the selected data to the memory location for providing a consuming application information about the selected data;
- opening a second document via the consuming application and selecting a consuming application paste function for pasting the selected data to the second document;
- reading by the consuming application the data associated with the selected data;
- in response to reading by the consuming application the data associated with the selected data, determining whether one or more namespaces are associated with the selected data defining permissible data content, data type and data structure for structure applied to the selected data;
- obtaining by the consuming application a resource associated with one of the one or more namespaces for pasting the selected data to the second document; and
- pasting the selected data to the second document according to the resource.

35. The computer readable medium of Claim 34, prior to obtaining by the consuming application a resource associated with the one or more namespaces for pasting the selected data,

- determining whether an available namespace list contains a desired namespace of the one or more namespaces for use by the consuming application;

- if the available namespace list contains a desired namespace of the one or more namespaces for use by the consuming application, selecting the desired namespace; and

providing the selected namespace to the consuming application.

36. The computer readable medium of Claim 35, whereby if the available namespace list does not contain a desired namespace of the one or more namespaces for use by the consuming application, determining whether a namespace library contains a desired namespace of the one or more namespaces for use by the consuming application;

if the namespace library contains one or more desired resources for the one or more namespaces for use by the consuming application, selecting the desired resource from the namespace library; and

providing the selected namespace and resource to the consuming application.

37. The computer readable medium of Claim 36, further comprising providing a user selectable choice for pasting the selected data to the second document according to the selected namespace.

38. The computer readable medium of Claim 37, whereby the markup language is the Extensible Markup Language (XML).

39. The computer readable medium of Claim 38, whereby the step of saving data associated with the selected data to the memory location for providing the consuming application information about the selected data includes identifying whether any of the one or more namespaces is associated with the first document.

40. The computer readable medium of Claim 39, whereby identifying whether any of the one or more namespaces is associated with the first document includes providing the consuming application information for locating any of the one or more namespaces on the available namespace list or at the namespace library.

41. A method for pasting data from a copying application to a consuming application, comprising:

applying a first Extensible Markup Language (XML) element to a beginning of a region of a first document, where the region is selected for copying from the copying application and for pasting to the consuming application;

applying a second XML element to an end of the selected region;

providing information in a header associated with the selected region, where the information includes an identification of available XML namespaces and pointers to a clipboard format identification that corresponds to each of the available namespaces;

providing information in the header on a file size for each the available namespaces; and

providing one or more namespaces of the available namespaces on a clipboard to allow the consuming application to choose among the one or more namespaces for pasting the data from the copying application.

1. Abstract

Methods and systems are provided for enhancing the paste functionality available to a computer software application for pasting data into a computer-generated document. Data is selected from a document to be pasted to a second document. The selected data is saved to a memory location, such a clipboard. Along with the selected data, information or data related to the selected data is saved for providing a consuming or pasting software application information about any data types associated the selected data. In response, the consuming application may obtain a namespace and an associated resource such as an Extensible Stylesheet Language Transformation file for transforming the selected data from a first data type associated with the first document to a second data type for pasting the selected data to the second document in order to maintain data structure and formatting in the pasted data as was applied to the selected data prior to pasting.

2. Representative Drawing

FIG. 3

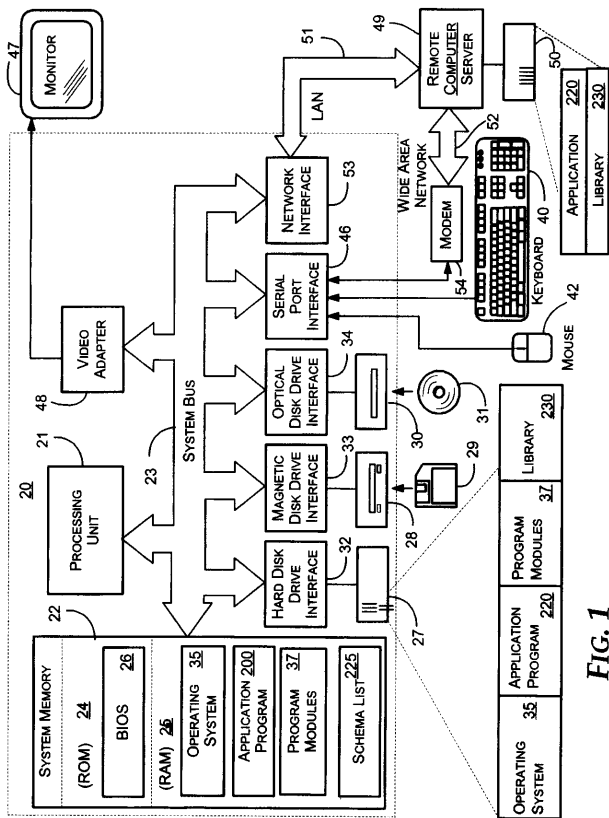


FIG. 1

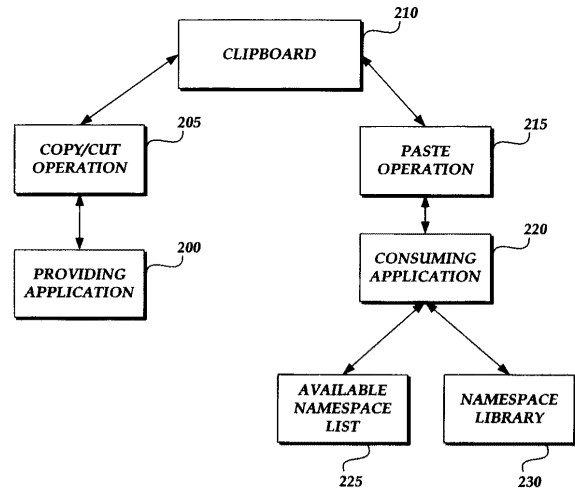


Fig. 2

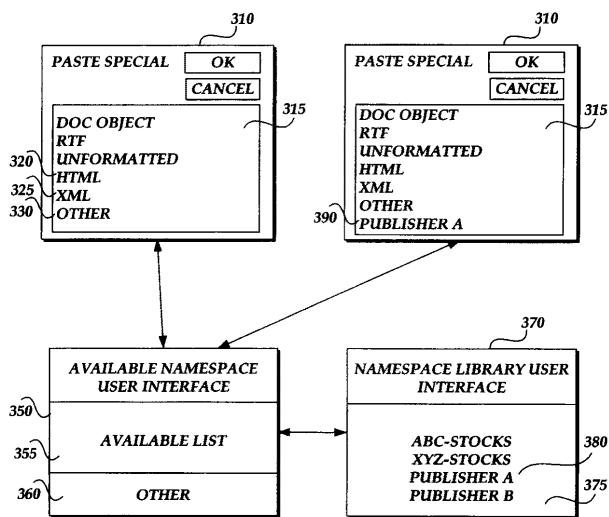


Fig. 3

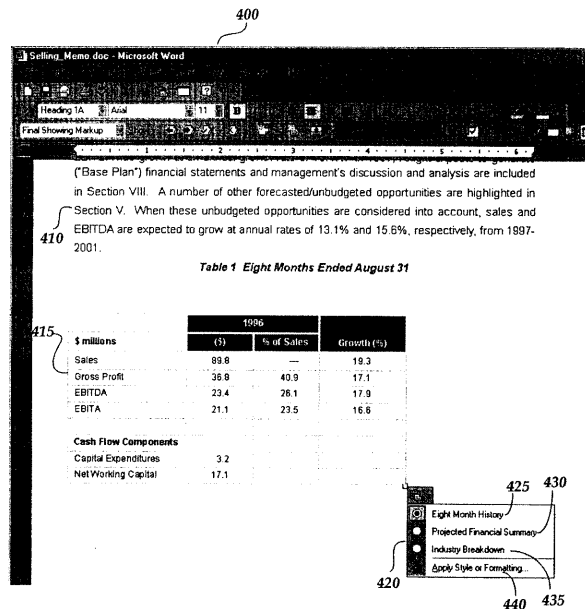


Fig. 4

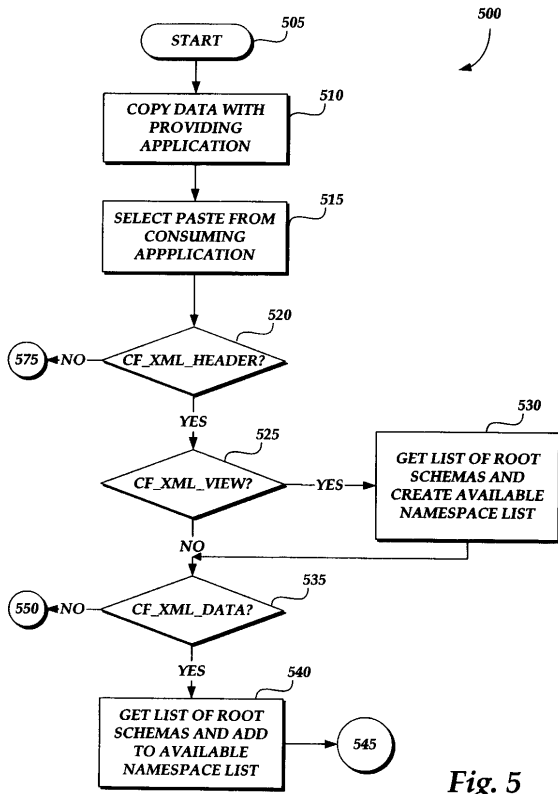


Fig. 5

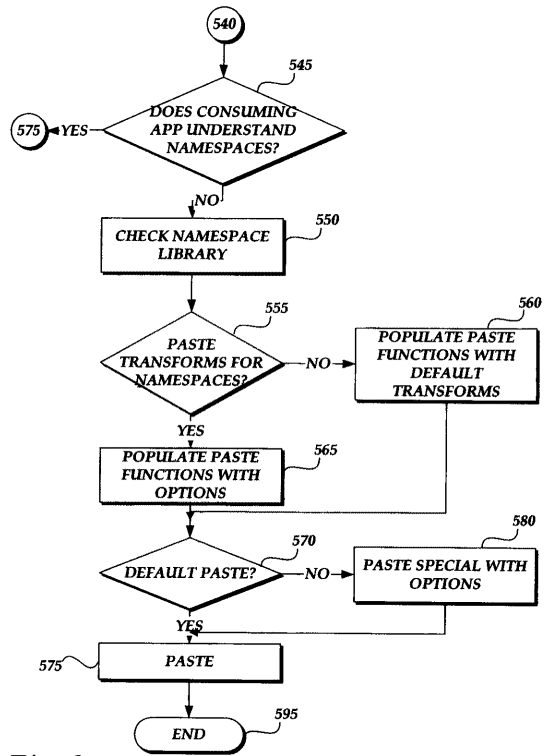


Fig. 6