



Office de la Propriété

Intellectuelle  
du Canada

Un organisme  
d'Industrie Canada

Canadian  
Intellectual Property  
Office

An agency of  
Industry Canada

CA 2533568 C 2013/07/16

(11)(21) **2 533 568**

(12) **BREVET CANADIEN  
CANADIAN PATENT**

(13) **C**

(22) Date de dépôt/Filing Date: 2006/01/20

(41) Mise à la disp. pub./Open to Public Insp.: 2006/08/25

(45) Date de délivrance/Issue Date: 2013/07/16

(30) Priorité/Priority: 2005/02/25 (US11/066,117)

(51) Cl.Int./Int.Cl. *G06F 17/00* (2006.01),  
*G06F 9/44* (2006.01)

(72) Inventeurs/Inventors:

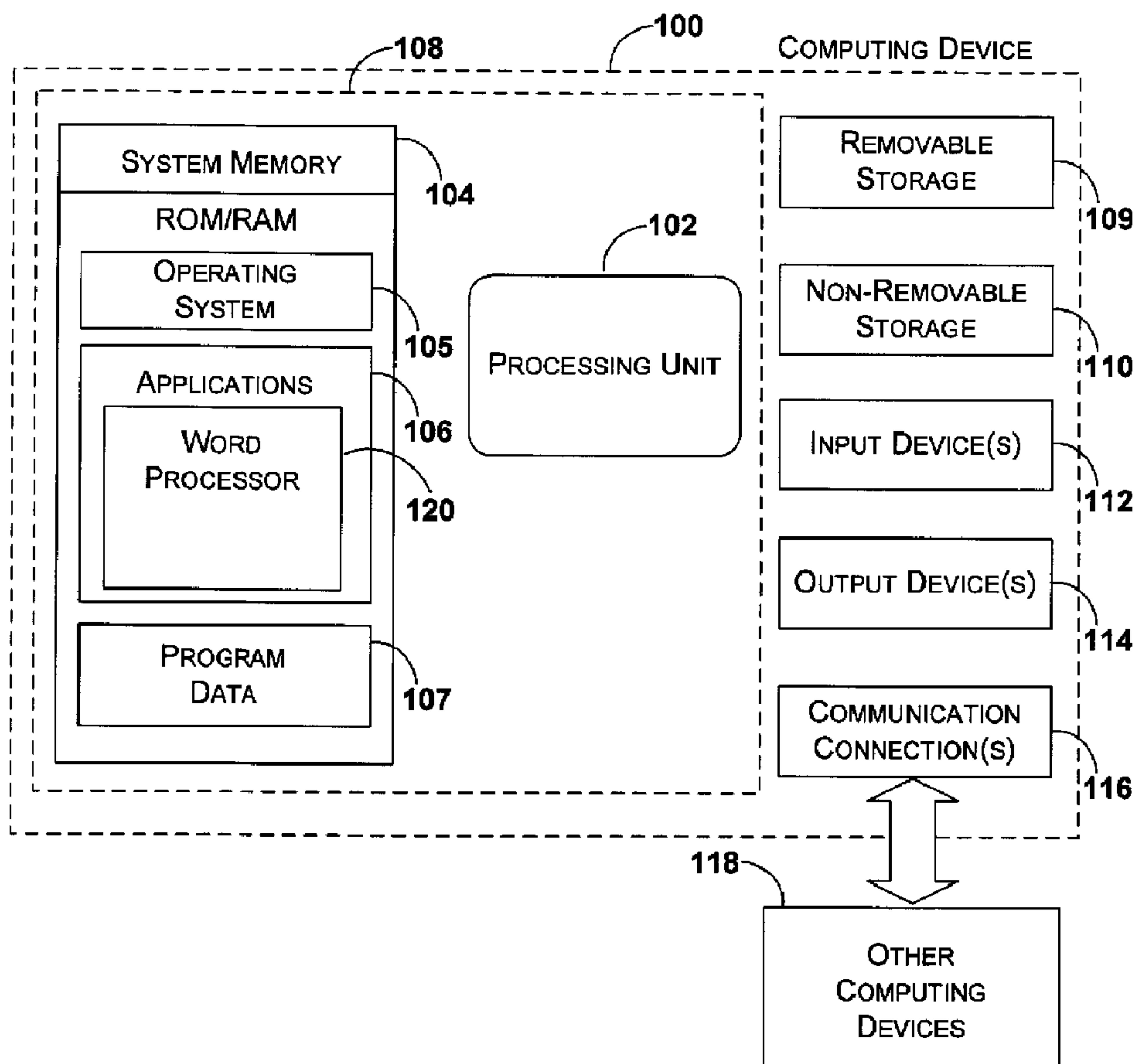
DAVIS, TRISTAN A., US;  
TALEGHANI, ALI, US;  
SAWICKI, MARCIN, US;  
LITTLE, ROBERT A., US;  
JONES, BRIAN M., US

(73) Propriétaire/Owner:  
MICROSOFT CORPORATION, US

(74) Agent: SMART & BIGGAR

(54) Titre : MAGASIN DE DONNEES POUR DOCUMENTS D'APPLICATION LOGICIELLE

(54) Title: DATA STORE FOR SOFTWARE APPLICATION DOCUMENTS



(57) Abrégé/Abstract:

A data store is provided for storing, relating and for allowing use of data associated with a computer-generated document. Data for structuring information associated with a document, such as document metadata, is maintained in the data store where

**(57) Abrégé(suite)/Abstract(continued):**

relationships between different pieces of data are maintained in a location distinct from the surface level view of a document. The data store exposes interfaces to the various pieces of data in the data store for allowing different applications to access and operate on one or more of the data pieces. The pieces of data may be structured according to a markup language such as the Extensible Markup Language (XML), and XML schemas may be associated with each piece of data for allowing the data store to validate the structure applied to the data based on an XML schema associated with a given piece of data.

## **DATA STORE FOR SOFTWARE APPLICATION DOCUMENTS**

### **Abstract**

A data store is provided for storing, relating and for allowing use of data  
5 associated with a computer-generated document. Data for structuring information  
associated with a document, such as document metadata, is maintained in the data store  
where relationships between different pieces of data are maintained in a location distinct  
from the surface level view of a document. The data store exposes interfaces to the  
various pieces of data in the data store for allowing different applications to access and  
10 operate on one or more of the data pieces. The pieces of data may be structured  
according to a markup language such as the Extensible Markup Language (XML), and  
XML schemas may be associated with each piece of data for allowing the data store to  
validate the structure applied to the data based on an XML schema associated with a  
given piece of data.

M&G No. 14917.0082US01/MS311572.01

## **DATA STORE FOR SOFTWARE APPLICATION DOCUMENTS**

### **Field of the Invention**

5        The present invention generally relates to managing data associated with computer-generated documents. More particularly, the present invention relates a data store for storing and relating data associated with computer-generated documents in a separate location from presentation data for a document's typical presentation format.

### **Background of the Invention**

10      With the advent of the computer age, computer and software users have grown accustomed to user-friendly software applications that help them write, calculate, organize, prepare presentations, send and receive electronic mail, make music, and the like. For example, modern electronic word processing applications allow users to prepare a variety of useful documents. Modern spreadsheet applications allow users to 15 enter, manipulate, and organize data. Modern electronic slide presentation applications allow users to create a variety of slide presentations containing text, pictures, data or other useful objects.

According to prior methods and systems, documents created by such applications (e.g. word processing documents, spreadsheets, slide presentation 20 documents) have limited facility for storing/transporting the contents of arbitrary metadata required by the context of the documents. For example, a solution built on top of a word processing document may require the storage of workflow data that describes various states of the document, for example, previous workflow approval states (dates, times, names), current approval states, future workflow states before completion, name 25 and office address of document author, document changes, and the like. According to such prior methods and systems, the options for storing this information were primarily limited to the use of document variables or existing custom object linking and embedding (OLE) document properties that have several limitations. For example, such

prior methods can only store name/value pairs (no hierarchical data). Such methods are limited to 255 characters maximum. Such methods are built to contain only text. All properties for such methods are stored in a single store, for example, an OLE properties store, which means the properties have a possibility of conflicting. Further, such stored 5 properties have no data validation because they are plain text. The result of these limitations is that it is difficult for users of such applications and related documents to store arbitrary data with documents, which is a common need of many users.

Another problem with prior methods and systems is that structured markup language data, for example Extensible Markup Language (XML) data may not be 10 concurrently edited by multiple clients (for example, multiple add-ins each independently running in the context of a word processing document.) However, in the context of many documents, there is a higher likelihood that the scenarios involving this metadata will require concurrent editing by one or more sources.

Accordingly, there is a need for a data store for storing and relating data 15 associated with a computer-generated document and for allowing use and manipulation of such data by one or more software applications. It is with respect to these and other considerations that the present invention has been made.

### **Summary of the Invention**

Embodiments of the invention solve the above and other problems by providing 20 a data store within the document, yet separate in location (and possibly format) from the primary presentation storage location for storing, relating and for allowing use of data associated with a computer-generated document.

According to one aspect of the invention, data for structuring information associated with a document, such as document metadata, is maintained in a data store 25 where relationships between different pieces of data are maintained. The data store exposes interfaces to the various pieces of data in the data store for allowing different applications to access and operate on one or more of the data pieces.

According to another aspect of the invention, the pieces of data are structured according to a markup language such as the Extensible Markup Language (XML).

XML schemas may be associated with each piece of data, and the data store may validate the XML structure applied to the data based on an XML schema associated with a given piece of data. According to this aspect of the invention, documents may contain any number of arbitrary data items, for example metadata, structured according  
5 to the Extensible Markup Language (XML). Accordingly, document solution providers may store arbitrary metadata as XML with a given document and have that information automatically processed by a given solution having access to the data when the document is opened/edited/saved by a user.

According to another aspect of the invention programmatic access is provided to  
10 the data in its XML form while the document is being edited. Thus, a standard mechanism is provided that is familiar to solution developers via which the data may be accessed and modified programmatically while the document is open. This programmatic access mimics standard XML interfaces. Programmatic access to the data is provided via application programming interfaces to one or more editing client  
15 applications (for example, document editing or creating applications and/or third party application add-in solutions, and the like). According to this aspect, multiple client applications may access and edit the same piece of document data, and any conflicting changes to a given piece of data are resolved. “Side effects” to any given change may be made (for example, in response to setting a company name to “Microsoft,” changing  
20 a stock symbol to “MSFT”). In addition, changes to data and any associated side effects may be “bundled” by the data store so that undoing one or more changes reverses all related changes. This removes the burden of development from the solution itself to ensure that it has reversed all changes when the user initiates an undo of the original change from the document surface, for example, by pressing an Undo  
25 command.

According to another aspect of the invention, standard XML schemas (XSDs) may be used to define the contents of any of the pieces of custom XML data associated with document metadata in order to ensure that XML data applied to the document data are valid. These schemas may be attached to any instance of XML data stored in the  
30 document, and the data store will disallow any change to the XML data that would

51028-90

result in the XML structure (that is, the XML tags as opposed to their contents) of that data from becoming invalid. This ensures that the solution developer can attach a specific piece of XML metadata to a document and ensure that the XML data will continue to be structurally “correct” according to the associated schema, regardless of which processes (for example, 5 add-ins) are used to modify that data.

According to another aspect of the invention, there is provided a computer-implemented method for generating an accessible metadata store in a source document, the method comprising: generating the source document on a source application, wherein the source document includes surface view data that is exposed on a presentation layer when the 10 source document is rendered and metadata that is unexposed on the presentation layer when the source document is rendered; structuring the metadata of the source document into a mark-up language (ML) document; storing the ML document that includes the metadata of the source document in the metadata store separate from the surface view data of the source document, wherein the metadata store and the surface view data are stored in the source 15 document; attaching an ML schema file to the ML document that includes the metadata of the source document, wherein the ML schema file is stored in the metadata store of the source document; exposing application programming interfaces (APIs) to the metadata store, wherein the APIs provide other applications having application types different than the source application access to the ML document that includes the metadata of the source document via 20 object models of the other applications, wherein the exposed APIs for the metadata store permit the other applications to: load the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, monitor modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, modify the metadata, stored in the metadata store that is stored in the 25 source document, independent of the surface view data, when a modification of the metadata is received at the metadata store, receive a change notification from the metadata store of the modification to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, and undo modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface 30 view data.

51028-90

According to another aspect of the present invention, there is provided a computer-readable storage medium having stored thereon computer executable instructions for generating an accessible metadata store in a source document, the instructions comprising: generating the source document on a source application, wherein the source document

- 5 includes surface view data that is exposed on a presentation layer when the source document is rendered and metadata that is unexposed on the presentation layer when the source document is rendered; structuring the metadata of the source document into a mark-up language (ML) document; storing the ML document that includes the metadata of the source document in the metadata store separate from the surface view data of the source document,
- 10 wherein the metadata store and the surface view data are stored in the source document; attaching an ML schema file to the ML document that includes the metadata of the source document, wherein the ML schema file is stored in the metadata store of the source document; exposing application programming interfaces (APIs) to the metadata store, wherein the APIs provide other applications having application types different than the source application
- 15 access to the ML document that includes the metadata of the source document via object models of the other applications, wherein the exposed APIs for the metadata store permit the other applications to: load the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, monitor modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, modify the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, when a modification of the metadata is received at the metadata store, receive a change notification from the metadata store of the modification to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, and undo modifications to the metadata,
- 20 stored in the metadata store that is stored in the source document, independent of the surface view data, modify the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, when a modification of the metadata is received at the metadata store, receive a change notification from the metadata store of the modification to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, and undo modifications to the metadata,
- 25 stored in the metadata store that is stored in the source document, independent of the surface view data.

According to another aspect of the present invention, there is provided a system for generating an accessible metadata store in a source document, the system comprising: a processor; and a memory having computer-executable instructions stored thereon, wherein the computer executable instructions are configured for: generating the

51028-90

source document on a source application, wherein the source document includes surface view data that is exposed on a presentation layer when the source document is rendered and metadata that is unexposed on the presentation layer when the source document is rendered; structuring the metadata of the source document into a mark-up language (ML) document;

- 5     storing the ML document that includes the metadata of the source document in the metadata store separate from the surface view data of the source document, wherein the metadata store and the surface view data are stored in the source document; attaching an ML schema file to the ML document that includes the metadata of the source document, wherein the ML schema file is stored in the metadata store of the source document; exposing application programming
- 10    interfaces (APIs) to the metadata store, wherein the APIs provide other applications having application types different than the source application access to the ML document that includes the metadata of the source document via object models of the other applications, wherein the exposed APIs for the metadata store permit the other applications to: load the metadata, stored in the metadata store that is stored in the source document, independent of
- 15    the surface view data, monitor modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, modify the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, when a modification of the metadata is received at the metadata store, receive a change notification from the metadata store of the modification to the metadata, stored in the
- 20    metadata store that is stored in the source document, independent of the surface view data, and undo modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data.

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

25    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.

51028-90

Other embodiments of the invention provide computer readable media having computer executable instructions stored thereon for execution by one or more computers, that when executed implement a method as summarized above or as detailed below.

#### Brief Description of the Drawings

5 FIGURE 1 illustrates an exemplary computing device that may be used in one exemplary embodiment of the present invention.

FIGURE 2 is a block diagram illustrating a relationship between one or more client applications and a data store and the contents of the data store according to embodiments of the present invention.

10

#### Detailed Description

As briefly described above, embodiments of the present invention are directed to methods and systems for storing and relating data associated with a computer-generated document and for efficiently allowing use and manipulation of data associated with a computer-generated document by one or more software applications. These embodiments 15 may be combined, other embodiments may be utilized, and structural changes may be made without departing from the spirit or scope of the present invention. The following detailed description is therefore not to be taken in a limiting sense and the scope of the present invention is defined by the appended claims and their equivalents.

51028-90

With reference to FIGURE 1, one exemplary system for implementing the invention includes a computing device, such as computing device 100. In a very basic configuration, computing device 100 typically includes at least one processing unit 102 and system memory 104. Depending on the exact configuration and type of computing device, system memory 104 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. System memory 104 typically includes an operating system 105, one or more applications 106, and may include program data 107. In one embodiment, application 106 may include a word processor application 120. This basic configuration is illustrated in FIGURE 1 by those components within dashed line 108.

Computing device 100 may have additional features or functionality. For example, computing device 100 may also include additional data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Such additional storage is illustrated in FIGURE 1 by removable storage 109 and non-removable storage 110. Computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. System memory 104, removable storage 109 and non-removable storage 110 are all examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computing device 100. Any such computer storage media may be part of device 100. Computing device 100 may also have input device(s) 112 such as keyboard, mouse, pen, voice input device, touch input device, etc. Output device(s) 114 such as a display, speakers, printer, etc. may also be included. These devices are well known in the art and need not be discussed at length here.

Computing device 100 may also contain communication connections 116 that allow the device to communicate with other computing devices 118, such as over a

network. Communication connection 116 is one example of communication media. Communication media may typically be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. The term computer readable media as used herein includes both storage media and communication media.

A number of program modules and data files may be stored in the system memory 104 of the computing device 100, including an operating system 105 suitable for controlling the operation of a networked personal computer, such as the WINDOWS operating systems from MICROSOFT Corporation of Redmond, Washington. System memory 104 may also store one or more program modules, such as word processor application 120, and others described below. Word processor application 120 is operative to provide functionality for creating, editing, and processing electronic documents.

According to one embodiment of the invention, the word processor application 120 comprises the WORD program from MICROSOFT Corporation. It should be appreciated, however, that word processor application programs from other manufacturers may be utilized to embody the various aspects of the present invention. It should further be appreciated that illustration of a word processing application is for purposes of example only and is not limiting of other types of applications that may produce and operate on documents according to the present invention. For example, other application programs 106 which are capable of processing various forms of content (e.g. text, images, pictures, etc.), such as spreadsheet application programs, database application programs, slide presentation application programs, drawing or computer-aided application programs, etc. are equally applicable to embodiments of the present invention. An example application program 106 that produces and operates on

a variety of different types of documents includes OFFICE from MICROSOFT Corporation.

5 Embodiments of the invention may be implemented as a computer process, a computing system, or as an article of manufacture such as a computer program product or computer readable media. The computer program product may be a computer storage media readable by a computer system and encoding a computer program of instructions for executing a computer process. The computer program product may also be a propagated signal on a carrier readable by a computing system and encoding a computer program of instructions for executing a computer process.

10 Throughout the specification and claims, the following terms take the meanings associated herein, unless the context of the term dictates otherwise.

15 The term "data" may refer to document surface level or presentation level information such as words, sentences, paragraphs and the like, as well as, supplementary information, for example, metadata, which is carried with, referred to, or used by the word processing document. This information is often large and is likely not exposed on the presentation layer of the document.

20 The terms "markup language" or "ML" refer to a language for special codes within a document that specify how parts of the document are to be interpreted by an application. In a word processor file, the markup language specifies how the text is to be formatted or laid out.

The term "element" refers to the basic unit of an XML document. The element may contain attributes, other elements, text, and other content regions for an XML document.

25 The term "presentation" refers to the visible portion of the document – the text and layout that would appear if the document were printed.

The term "tag" refers to a character inserted in a document that delineates elements within an XML document. Each element can have no more than two tags: the start tag and the end tag. It is possible to have an empty element (with no content) in which case one tag is allowed.

The XML content between the tags is considered the element's "children" (or descendants). Hence other elements embedded in the element's content are called "child elements" or "child nodes" or the element. Text embedded directly in the content of the element is considered the element's "child text nodes". Together, the 5 child elements and the text within an element constitute that element's "content".

The term "attribute" refers to an additional property set to a particular value and associated with the element. Elements may have an arbitrary number of attribute settings associated with them, including none. Attributes are used to associate additional information with an element that will not contain additional elements, or be 10 treated as a text node.

"XPath" is an operator that uses a pattern expression to identify nodes in an XML document. An XPath pattern is a slash-separated list of child element names that describe a path through the XML document. The pattern "selects" elements that match the path.

15 The term "XML data store" refers to a container within a document, such as a word processor document, a spreadsheet document, a slide presentation document, etc., which provides access for storage and modification of the data ( in XML format, for example) stored in the document while the file is open. Further definition of XML data store is provided below with respect to Figure 2.

20 Figure 2 is a block diagram illustrating a relationship between one or more client applications and a data store and the contents of the data store according to embodiments of the present invention. Referring to Fig. 2, the document data 220 includes XML structure data and associated document data representing the surface or presentation level view of a document. For example the document data 220 may 25 include XML structure (e.g., heading tags, body tags, conclusion tags) and associated surface view data (e.g., words, sentences, paragraphs) of a word processing document, spreadsheet document, slide presentation document, and the like.

The data store 208 is a document data repository for storing one or more pieces of structured data associated with one or more types of data associated with a given 30 document. The metadata1 225 (structured data item) may include XML structure data

and associated data for a first piece of metadata associated with the document. For example, the metadata1 225 may include XML structure data (e.g., date tags, name tags, etc.) applied to metadata listing the document author, date of document creation, date of document last change/save, and the like. The metadata2 230 (structured data item) may 5 include XML structure data (tags) and associated metadata representing a second piece of metadata associated with the document. As should be understood, the metadata1 and metadata2 are for purposes of example and are not limiting of the variety and number of different types of data that may be maintained in the data store 208 in association with a given document. For example, as described herein, arbitrary data may be structured and 10 added to the document by one or more software application as desired by solution providers or users having access to the document data.

Referring still to Fig. 2, a schema file 240, 245 may be attached to each piece of data stored in the data store 208 for dictating the syntax and validation rules associated with Extensible Markup Language (XML) data applied to each piece of data 225, 230. 15 As known to those skilled in the art, XML schema files provide a way to describe and validate data in an XML environment. A schema file states what XML markup data, including elements and attributes, are used to describe content in an XML document, and the schema file defines XML markup syntax, including where each element is allowed, what types of content are allowed within an element and which elements can 20 appear within other elements. The use of schema files ensures that the document (or individual piece of data in this case) is structured in a consistent and predictable manner. Schema files 240, 245 may be created by a user and generally supported by an associated markup language, such as XML.

This schematization of the document allows the data store to provide the ability 25 to “guarantee” the structural validity of the document by rejecting any change that violates a given schema file at the data store level. According to an embodiment, the data store 208 utilizes a schema validation module 260 for validating XML structure added to or changes made to a given piece of data against an associated schema file. For example, if a document creator or editor makes XML structural changes to a given 30 piece of data, for example, the metadata1, wherein the editor adds or removes a given

XML tag, the data store 208 will utilize the schema validation module to check the XML structural changes against the associated schema file to ensure the validity of the change. If the change is not valid, an error can be generated to the editor. As is understood, such control of the XML structure applied to a given piece of data allows 5 for structural consistency and predictability which is especially important for allowing client and third party applications to interact with associated data.

According to an embodiment of the invention, the data store 208 provides one or more application programming interfaces (API) 270 which can be accessed by client applications 205 (e.g., word processing applications, spreadsheet applications, slide 10 presentation applications, etc.), as well as, third party applications 210, 215 via the object models (OM) of the respective applications 205, 210, 215. These interfaces allow client applications and third party applications to load any existing XML file into a given document's data store 208, thus ensuring that that data is now part of the document and will travel within that document for its lifetime (e.g., through 15 opening/editing/saving/renaming/etc.) or until the data is deleted from the data store. According to one embodiment, the data in the data store is available in its XML format even when a source application for a given piece of data 225, 230 is closed or is otherwise not available. That is, a given piece of data 225, 230 may be accessed via the APIs 270 by other applications (other than a source application). As described below, 20 the APIs also allow client and third party applications to make changes to the XML markup data applied to the data items 225, 230.

Once XML data 225, 230 is loaded into the data store for association with a document 220, it can be manipulated as standard XML using the data store interfaces designed to provide similar methods to existing XML editing interfaces in order to 25 leverage developers' existing knowledge of the XML programming standard. This allows users to perform standard XML operations on XML data added to the data store for a document, such as adding elements and attributes, removing elements and attributes, changing the value of existing elements/attributes, and reading the values of any existing part of the associated XML tree. Using these XML standard operations, 30 solutions may store structured complex metadata with a document subject to none of the

previous restrictions on the length/size of the data or structure of the data, which enables the use of this XML data store for significantly more structured solutions than prior solutions. For example, a third party application 215 may be written for locating and extracting document author names and document creation dates from a number of 5 documents 204 by reading the metadata1 225 added to the data store 208 for each document. The example third party application may be a spreadsheet application programmed for making a list of document author names and document creation dates for all documents created by a given organization. In accordance with embodiments of the present invention, the third party application may utilize the XML structure applied 10 to the metadata1 for efficiently locating and extracting the desired data. For example, the third party application may be written to parse the XML structure of the metadata1 file to locate XML tags, such as <docauthor> and <doccreationdate> for obtaining and using data associated with those tags. As should be appreciated, the forgoing is just one example of the many ways one or more applications may interact with structured data 15 that is associated with the document via the data store 208.

In addition, the data store 208 provides any number of API interfaces 270 to any individual piece of XML data 220, 225, 230 (also known as a store item) to enable multiple applications 205, 210, 215 to work with the same piece of data. For example, several solutions, such as a client application (e.g., word processing application) and 20 third party application solutions (e.g., the example spreadsheet application described above), may work with the same set of document properties (e.g., properties contained in the metadata2 230 file). Using the data store 208, each of these applications receive separate access to the desired XML data 230 through their own data store API interface 270 for allowing each application to communicate with the data via its own OM 25 without having to deal with the complexity of having multiple processes accessing the same piece of data.

In order to allow for these multiple applications 205, 210, 215 to access the same data, the data store 208 notifies each of these applications when any part of the XML data is changed by another application so that a given application may respond to 30 that change (both internally to its own process and externally by other changes to the

same data). When one application requests a change to a given data item, that request is automatically sent to all other applications to allow other applications to decide how or if to respond to the requested change. According to one embodiment, this is accomplished by allowing each application to register to “listen” to any part of the

5 XML data to which it has an interface so that a given application solution/program only receives those messages which are pertinent to its own logic. For example, one type of application 210 may wish to register to listen to all changes made to a given XML data in order to provide detailed business logic capabilities to a third party solution, but another type of application 215 may wish to only listen to changes to one or two

10 specific XML elements within the same data because its logic does not care about changes to any other part of the XML data.

According to this embodiment, the multiple applications 205, 210, 215 may access and edit the same piece of document data, and any conflicting changes to a given piece of data are resolved. For example, “side effects” to any given change may be

15 made when one change by one application causes a side effect change by another application. For example, a first application 210 may be tasked with extracting company names from one or more data items 225, 230 associated with a given document for translating those names into corresponding stock symbols, if available, for compiling a list of company stock symbols related to a given document. If a second

20 application 215 causes a given company name in a given piece of metadata to be added or to be changed, for example, changing a company name from “Company ABC” to Company XYZ,” the first application may listen to this change for automatically updating its list of stock symbols to include the stock symbol for “Company XYZ” instead of “Company ABC.” In addition, such changes and any associated side effects

25 may be bundled by the data store 208 so that undoing one or more changes reverses all related changes.

As described herein, embodiments of the invention provide a data store for storing, relating and for allowing use of data associated with a computer-generated document. It will be apparent to those skilled in the art that various modifications or

30 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.

51028-90

CLAIMS:

1. A computer-implemented method for generating an accessible metadata store in a source document, the method comprising:

generating the source document on a source application, wherein the source

5 document includes surface view data that is exposed on a presentation layer when the source document is rendered and metadata that is unexposed on the presentation layer when the source document is rendered;

structuring the metadata of the source document into a mark-up language (ML) document;

10 storing the ML document that includes the metadata of the source document in the metadata store separate from the surface view data of the source document, wherein the metadata store and the surface view data are stored in the source document;

attaching an ML schema file to the ML document that includes the metadata of the source document, wherein the ML schema file is stored in the metadata store of the source

15 document;

exposing application programming interfaces (APIs) to the metadata store, wherein the APIs provide other applications having application types different than the source application access to the ML document that includes the metadata of the source document via object models of the other applications, wherein the exposed APIs for the metadata store

20 permit the other applications to:

load the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

monitor modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

51028-90

modify the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

when a modification of the metadata is received at the metadata store, receive a change notification from the metadata store of the modification to the metadata, stored in the  
5 metadata store that is stored in the source document, independent of the surface view data, and

undo modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data.

2. The computer-implemented method of claim 1, wherein the ML document is an Extensible Mark-up Language (XML) document.

10 3. The computer-implemented method of claim 1, wherein said structuring the metadata of the source document into the ML document includes structuring the metadata into a plurality of ML documents, wherein each of the plurality of ML documents includes a portion of the metadata of the source document.

4. The computer-implemented method of claim 1, wherein the ML document is  
15 validated by the ML schema file.

5. The computer-implemented method of claim 1, wherein the surface view data is stored in the source document in a separate data store from the metadata store.

6. The computer-implemented method of claim 1, wherein the exposed APIs permit other applications to simultaneously:

20 load the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

monitor modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

51028-90

modify the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

when the modification of the metadata is received at the metadata store receive the change notification from the metadata store of the modification to the metadata, stored in  
5 the metadata store that is stored in the source document, independent of the surface view data, and

undo modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data.

7. A computer-readable storage medium having stored thereon computer  
10 executable instructions for generating an accessible metadata store in a source document, the instructions comprising:

generating the source document on a source application, wherein the source document includes surface view data that is exposed on a presentation layer when the source document is rendered and metadata that is unexposed on the presentation layer when the  
15 source document is rendered;

structuring the metadata of the source document into a mark-up language (ML) document;

storing the ML document that includes the metadata of the source document in the metadata store separate from the surface view data of the source document, wherein the  
20 metadata store and the surface view data are stored in the source document;

attaching an ML schema file to the ML document that includes the metadata of the source document, wherein the ML schema file is stored in the metadata store of the source document;

exposing application programming interfaces (APIs) to the metadata store,  
25 wherein the APIs provide other applications having application types different than the source application access to the ML document that includes the metadata of the source document via

51028-90

object models of the other applications, wherein the exposed APIs for the metadata store permit the other applications to:

load the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

5 monitor modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

modify the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

when a modification of the metadata is received at the metadata store, receive a  
10 change notification from the metadata store of the modification to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, and

undo modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data.

8. The computer-readable storage medium of claim 7, wherein the ML document  
15 is an Extensible Mark-up Language (XML) document.

9. The computer-readable storage medium of claim 7, wherein said structuring the metadata of the source document into the ML document includes structuring the metadata into a plurality of ML documents, wherein each of the plurality of ML documents includes a portion of the metadata of the source document.

20 10. The computer-readable storage medium of claim 7, wherein the ML document is validated by the ML schema file.

11. The computer-readable storage medium of claim 7, wherein the surface view data is stored in the source document in a separate data store from the metadata store.

51028-90

12. The computer-readable storage medium of claim 7, wherein the exposed APIs permit other applications to simultaneously:

load the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

5 monitor modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

modify the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

when the modification of the metadata is received at the metadata store, receive  
10 the change notification from the metadata store of the modification to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, and

undo modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data.

15 13. A system for generating an accessible metadata store in a source document, the system comprising:

a processor; and

a memory having computer-executable instructions stored thereon, wherein the computer executable instructions are configured for:

20 generating the source document on a source application, wherein the source document includes surface view data that is exposed on a presentation layer when the source document is rendered and metadata that is unexposed on the presentation layer when the source document is rendered;

51028-90

structuring the metadata of the source document into a mark-up language (ML) document;

storing the ML document that includes the metadata of the source document in the metadata store separate from the surface view data of the source document, wherein the

5 metadata store and the surface view data are stored in the source document;

attaching an ML schema file to the ML document that includes the metadata of the source document, wherein the ML schema file is stored in the metadata store of the source document;

exposing application programming interfaces (APIs) to the metadata store,

10 wherein the APIs provide other applications having application types different than the source application access to the ML document that includes the metadata of the source document via object models of the other applications, wherein the exposed APIs for the metadata store permit the other applications to:

load the metadata, stored in the metadata store that is stored in the source

15 document, independent of the surface view data,

monitor modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

modify the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

20 when a modification of the metadata is received at the metadata store, receive a change notification from the metadata store of the modification to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, and

undo modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data.

51028-90

14. The system of claim 13, wherein the ML document is an Extensible Mark-up Language (XML) document.

15. The system of claim 13, wherein said structuring the metadata of the source document into the ML document includes structuring the metadata into a plurality of ML 5 documents, wherein each of the plurality of ML documents includes a portion of the metadata of the source document.

16. The system of claim 13, wherein the ML document is validated by the ML schema file.

17. The system of claim 13, wherein the surface view data is stored in the source 10 document in a separate data store from the metadata store.

18. The system of claim 13, wherein the exposed APIs permit other applications to simultaneously:

load the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

15 monitor modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

modify the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data,

when the modification of the metadata is received at the metadata store, receive 20 the change notification from the metadata store of the modification to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data, and

undo modifications to the metadata, stored in the metadata store that is stored in the source document, independent of the surface view data.

1/2

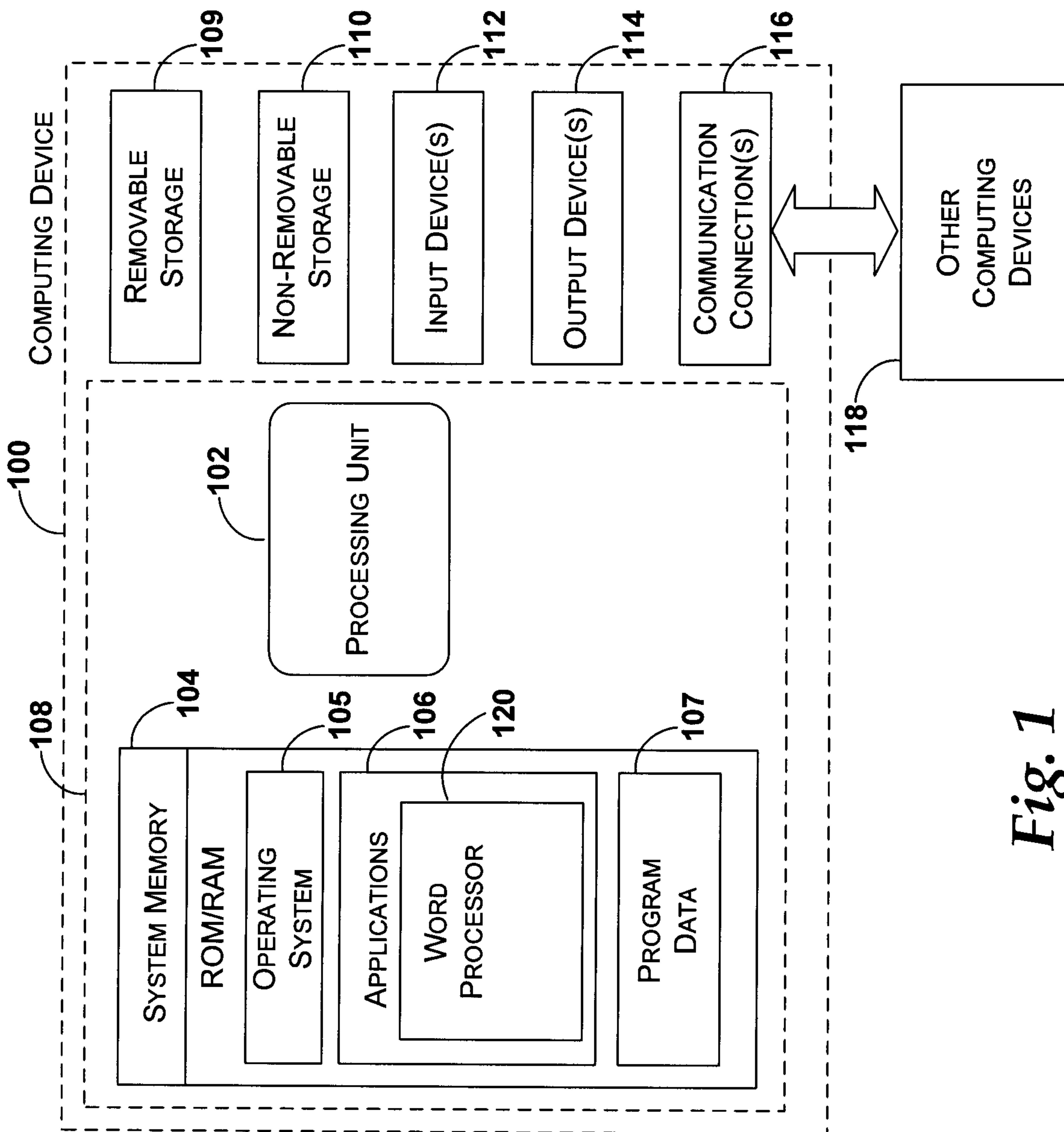


Fig. 1

2/2

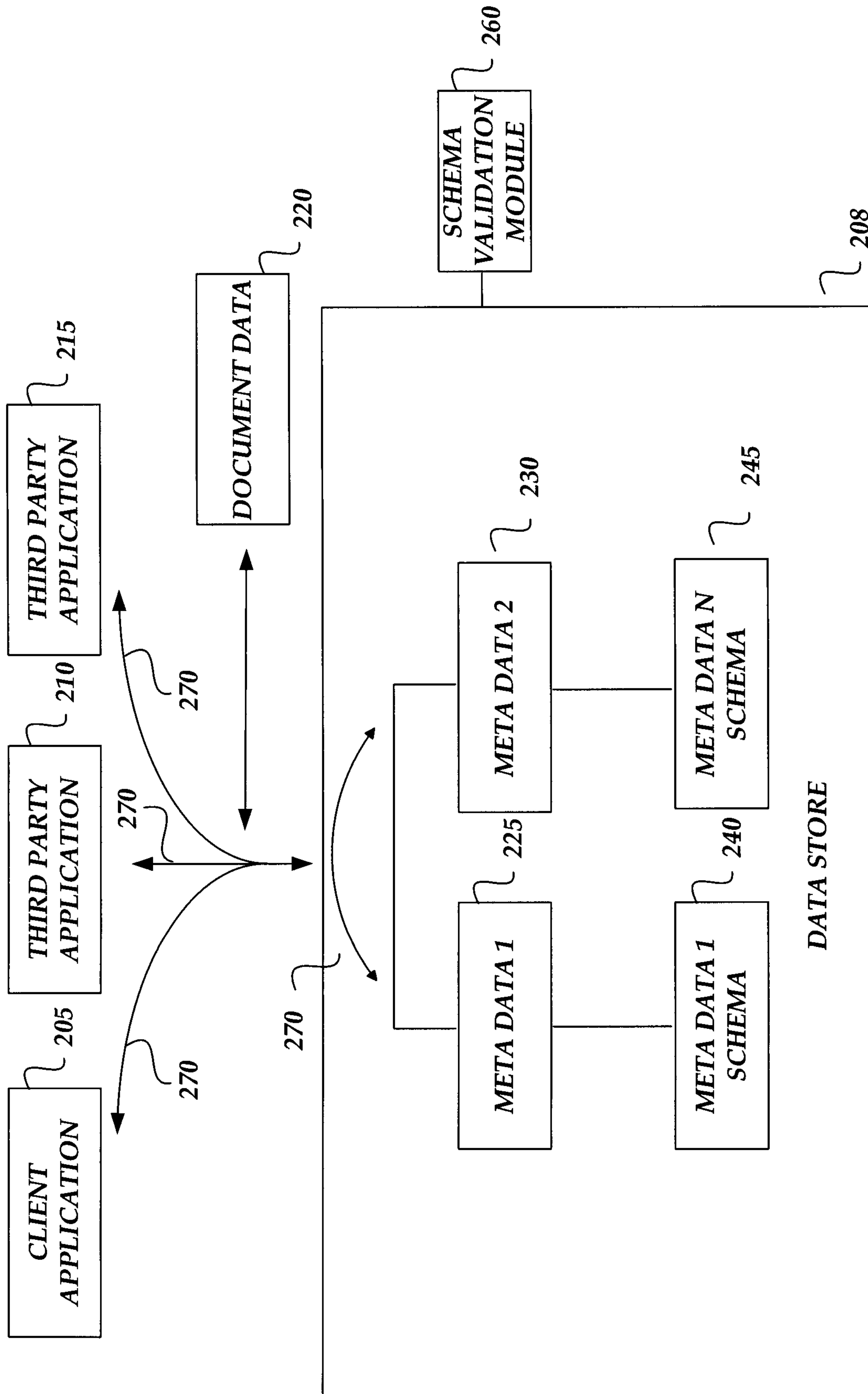


Fig. 2

