

**ENHANCED COMPOUND DOCUMENT  
PROCESSING ARCHITECTURES AND  
METHODS THEREFOR**

**COPYRIGHT NOTICE**

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The owner has no objection to the facsimile reproduction by any one of the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

**BACKGROUND OF THE INVENTION**

The present invention relates generally to computer systems and to document processing systems in computers. More particularly, the present invention relates to improved apparatuses and methods for communicating attribute data between objects in a compound document in a computer.

Traditionally, documents are processed using stand-alone application programs. Each document in the application-centered approach typically utilizes a single application program to render and manipulate its data. Further, data within an application-centered document is homogenous throughout. By way of example, a text document typically contains only text and is rendered and manipulated by a word processor application program. In the application-centered world, if a computer user wants to create a graphics image, he or she would typically have to switch to a different document that is associated with, say a graphics application program.

In contrast, compound documents are documents whose contents are non-homogenous. A compound document may have within it different types of data, such as text, graphics, sound, or other types of data that are displayable or manipulable using a computer. Representative existing compound document architectures include OpenDoc™ by Apple Computer, Inc. of Cupertino, Calif. and OLE™ by Microsoft Corporation of Redmond, Wash.

Furthermore, a compound document embeds, or incorporates, multiple applications in a single compound document framework. Each application program so embedded is responsible for rendering and manipulating its associated data in a discrete content area of the compound document. As such, a computer user may move among the discrete content areas to use the respectively associated embedded applications to edit the compound document's non-homogeneous contents without having to switch documents. For this reason, many people find that compound documents are easy to work with.

To facilitate a discussion of compound documents, FIG. 1 shows a traditional compound document along with its various constituting elements. The compound document of FIG. 1 may be created by, for example, the aforementioned OpenDoc™ compound document software. Referring now to FIG. 1, there is shown a window 200 representing a window within which a compound document may be displayed and manipulated. As is well known to those of skill in the art, window 200 may include a menu bar 202 and a display area 204. There may optionally be provided a scroll bar 206 for permitting the computer user to scroll through different portions of the compound document.

Within display area 204 there is shown a root view of the compound document. In a compound document, the aforementioned different types of data, e.g., text, graphics, sound, and the like, co-exist within a document "container." The

root view is a visual representation of these different types of data in the aforementioned compound document container.

Within the document container, a plurality of objects may be embedded, i.e., incorporated or contained. As the term is used herein, an object is defined as an area in a document that contains information or "content." Programs that manipulate object information are called object editors. Visual representations of objects on screen or in an electronic document are called data objects. In a typical compound document architecture objects may contain other objects in an embedding hierarchy, where the first object present in a document is referred to as the root object. Since the root object is an embedded object, it delineates the content area within which an intrinsic text content associated with the root object is rendered. An example of the intrinsic text content associated with the root object is illustrated in the sentence that reads: "This is a South American iguana."

To render and manipulate this intrinsic text content of the root object, there is associated with the root object a root object editor, also known as a root editing component, representing the underlying program that manipulates object content. In the example of FIG. 1, this root object editor is shown as object editor 214 and may be implemented by, for example, a word processor.

Besides the root object, the document container is further capable of embedding, i.e., incorporating, other objects. Each embedded object, whether or not a root object, delineates a discrete, mutually exclusive content area within the compound document. Content areas are mutually exclusive because each content area only contains one type of data. By way of example, there is shown in FIG. 1 an embedded object 208, which serves to delineate the content area of the compound document that is associated with a graphic image 210. Note that only graphics data is shown in the content area associated with embedded object 208. In contrast with the root object, which represents the first level of embedding in the document container, embedded object 208 represents a deeper level of embedding.

Each embedded object has associated with it an object editor, which is used for rendering and manipulating the content that is intrinsic to that embedded object. An example of an object editor, i.e., root object editor 214, has already been discussed in connection with the embedded root object. As a further example, object editor 216 of FIG. 1 may represent the graphics object editor associated with embedded object 208 for rendering and manipulating graphics image 210 within object 208. Object editor 216 may represent, for example, a simple drawing program.

In general, each object editor has a proprietary user interface, which is typically furnished by the developer of the application program code underlying that embedded object editor. The user interface represents, among other things, the overall "look" of the content area with which an object editor is associated, the manner in which the content is rendered, as well as the manner in which editing tools are implemented. The portion of the user interface for laying out the editing tools is referred to herein as a UI container since its role, be it a menu bar, a floating palette, a dialog box, or the like, is to provide an environment for implementing editing tools.

Within object 208 of FIG. 1, there is further embedded an object 212, representing a yet further level of embedding within the document container. Embedded object 208 serves to delineate the content area associated with the text content inside it. This text content is represented by, for example, the