

scribing file may be a word processing file, a diagramming file, a spreadsheet file, an image file, or any other type of file that an application is capable of accessing and/or modifying. Furthermore, in embodiments, the self-describing file may be written in any number of formats without departing from the spirit of the embodiments disclosed herein. The self-describing file type may be written in an XML format, an HTML format, in a binary format, or any other type of format known to the art. While the method 200 describes a discrete number of steps occurring in a particular order, one of skill in the art will appreciate that the method 200 may be performed in a different order or may comprise fewer or additional steps.

FIG. 3 is an illustration of a flowchart representing an embodiment of a method 300 performed by an application capable of properly recalculating properties values stored in a self-describing file. For example, embodiments of the method 300 may be performed by a full featured application that is capable of supporting and/or understanding the objects, properties, and/or capabilities provided in the self-describing file. Flow begins at operation 302 where an application opens or otherwise accesses the self-describing file. In embodiments, the step of opening the self-describing file may occur after another application accessed and/or modified the self-describing file. However, the other application may not have correctly preserved the content of the file when modifying it. This may occur due to limitations in the other application's capabilities, the other application's inability to understand the content of the file, an error encountered by the other application when writing information to the self-describing file, or for any other reason.

Flow continues to operation 304 where the application determines that at least one portion of the content of the self-describing file has not been properly maintained. For example, a second property of an object that is dependent upon a first property may not have been correctly recalculated upon modification of the first property. In one embodiment, the application may make such a determination by parsing the file to check for missing and/or incorrect data. In another embodiment, the determination may be made by checking for an indicator within the file that the file contents have not been correctly updated. For example, the indicator may be a flag or an error message placed within or associated with the self-describing file. Although specific examples are provided as to the process and mechanisms used by the application to make the determination at operation 304, one of skill in the art will appreciate that any manner of determining that that at least a portion of the content of the self-describing file is incorrect may be employed at operation 304.

Flow continues to operation 306 where the application recalculates any of the file content that was improperly updated during a previous modification of the self-describing file. In one embodiment, the application may recalculate a value for a property, object, or other portion of the content based upon a relationship, function, or formula provided in an extension included in the extension section of the self-describing file. In another embodiment, the application performing operation 306 may be a full featured application that is capable of supporting all objects, properties, and/or capabilities of the self-describing file. As such, the full featured application may be capable of natively recalculating any errant data without reliance upon an extension element in the self-describing file. In such an embodiment, the recalculation of the file content at operation 306 may further include the addition of information (e.g., object, properties, formulas, and/or functions provided by the extension elements described with respect to FIG. 1) to the self-describing file. For example, the content of the self-describing file may not

have been properly preserved by the last application due to the fact that an extension defining a relationship of the content was missing from the file. Upon recalculating the file content at operation 306, the full featured application may prevent future miscalculations by updating an extension section included in or associated with the self-describing file.

Flow continues to operation 308 where the application stores the recalculated file content (e.g., property values, objects, etc.) in the self-describing file. In embodiments, the recalculated values may be stored by writing the values to the self-describing files at operation 308. In further embodiments, operation 308 may also include writing or otherwise storing information related to a portion of a file (e.g., extension objects) to an extension section that is part of or associated with the self-describing file at operation 308.

In embodiments, flow continues to operation 310 where the application displays the file contents from the self-describing file to the user. For example, an object and its properties may be displayed to the user. Returning to the example involving the diagramming application, a shape may be displayed to the user in a manner such that its properties are correctly calculated and displayed. In embodiments, because the application recalculated and stored any improperly maintained file contents at operations 306 and 308, the file is correctly displayed to the user at operation 310. By doing so, the problems that occur when a less featured application modify a file shared between applications are avoided, thereby providing a user of a less featured application with an enhanced experience by providing a more complete calculation of data for the file.

While operation 310 is described as displaying the content of the self-describing file to the user, one of skill in the art will appreciate that the application may perform other functionality at operation 310 such as, but not limited to, playing a video, playing audio, or otherwise executing the self-describing file to perform a function or task. Additionally, while the method 300 describes a discrete number of steps occurring in a particular order, one of skill in the art will appreciate that the method 300 may be performed in a different order or with more or fewer steps.

FIG. 4 is an illustration of flowchart representing an embodiment of a method 400 for preserving unknown file contents. In embodiments, an application performing the method 400 may be a less featured application that does not natively support or understand all of the capabilities or file contents (e.g., objects, properties, etc.) of the self-describing file. Flow begins at operation 402 where an application opens a self-describing file, such as the self-describing file 100 of FIG. 1. Upon opening the file, flow continues to operation 404 where the application receives information related to a portion of the self-describing file that the application may not natively support. For example, in one embodiment the application may receive information defining an object, property, characteristic, or other portion of the file that the application does not support. In another embodiment the application receives an indication that at least one portion of data in the file is dependent upon a first portion of the file. For example, a second property of an object may be dependent upon a first property of an object. In yet another embodiment, the application receives an indication that certain data is to be updated in the self-describing file upon performing an action.

In embodiments, the application receives an indication by processing an extension section that may be part of or associated with the self-describing file. In embodiments, the application may examine one or more extension elements (or one or more children of an extension element) in order to derive information to calculate values for extended portions of the file that the application does not natively support. An