

various data object classes, the attributes and the relationships between them operated by the application, and a document generated by the application is an instance of the data model of the application.

[0037] FIG. 3 shows an example of data models of different versions of an application. As shown, by comparing the data model of version 2.0 with the data model of version 1.0, it can be seen that the class Person is renamed as Individual, the attribute gender is renamed as sex, and the type of the attribute grade, EInt, is changed into Grade, additionally a new class, Grade, is added. By comparing the data model of version 3.0 with the data model of version 2.0, it can be seen that the attributes of the class School, grades and courses, are added, and their types are designated as classes Grade and Course, respectively, and the attribute salary of the class School is deleted; in addition, a new class, Course, is added.

[0038] During operation, the application can create a document according to its data model, that is, forming an instance of the data model in the memory, namely, an object graph conforming to the data model; and store the document on a disk, that is, serializing the object graph in the memory to a disk file; or load the document from the disk, that is, read the contents of the disk file into the memory and parse the document contents according to the data model, so as to form an object graph conforming to the data model in the memory and display the object graph in a user interface (UI).

[0039] In order for the application to open, read and parse the document contents correctly, the data model used by the application when parsing the document contents must be consistent with the data model used when the document is generated and stored; otherwise, the document cannot be opened, read or parsed, thus arising a compatibility problem. FIG. 4 schematically shows an exemplary editor application editing a document in a user interface and an object graph of the document in the memory. FIG. 5 shows an exemplary data model instance represented in XML format.

[0040] The data model of an application is usually described using a formal model description language, e.g., UML, XSD, Ecore, etc. The model extractor may use any one of a plurality of methods known in the art to acquire data models of different versions of an application. For example, a reverse engineering technique can be used to automatically extract the data model of an application from the source code of the application or, for those applications which were developed based on existing data models, the data models of the applications can also be acquired directly.

[0041] The model comparator can use a model comparison technique known in the art to compare the data models of different versions, and automatically generate a differentiation model representing the differences between the data models of different versions. For example, an existing model weaving tool, e.g., AMW, can be used to generate the differentiation model. Additionally or alternatively, the user can manually modify or create a differentiation model.

[0042] For example, modern model comparison techniques generally use a “minimum edit-distance” hypothesis to find the differences between two models; especially when the number of different elements is very small as compared with the number of total elements, these techniques work well. However, there exist a few cases where the results generated by such a method of automatically finding differences do not reflect the truth. When an attribute “label” in an old data model is deleted, and a new attribute “weight” is created in a new data model, according to the principle of minimum edit-

distance, the differentiation model will view “weight” as a substitution for “label”, which is not correct. Therefore, the model comparator may allow the user to modify or edit an automatically generated differentiation model.

[0043] For example, with respect to a data model described using UML, the model comparator may acquire the information of adding, deleting or modifying various UML elements, such as, Class, Attribute, Aggregation, Composition, and Generalization between data models of different versions, and record the information in a differentiation model. For example, for the exemplary data models of version 1.0 and version 2.0 shown in FIG. 3, a differentiation model including the following information may be generated: 1) the class “Person” was renamed as “Individual”; 2) the attribute “gender” of the class “Person” was renamed as “sex”; 3) a class “Grade” with an attribute “name” was added; 4) the type of the attribute “grade”, “EInt”, was modified as “Grade”.

[0044] The differentiation model may use any known data structure in the art. As long as the differentiation model includes the differences between document data models of different versions so that it can be used for inter-version document conversion, it is feasible. According to an embodiment of the present invention, the conversion stack is any data structure which includes differentiation models between adjacent versions and verification models for verifying the validity of documents.

[0045] The conversion module 202 for performing conversion between documents of different versions of an application by using the conversion stack 201 is further configured for performing the following operations: obtaining a document of a higher version; successively converting the document of the higher version into zero, one or more documents of intermediate versions and a document of a lower version by using one or more differentiation models between data models between the higher version and the lower version in the conversion stack 201, and storing the document of the higher version, and the zero, one or more documents of the intermediate versions between the higher version and the lower version or relevant information therein (for example, information lost when the document of one version is converted into the document of another version); sending the document of the low version to the application of the lower version, so as to perform required modifications to the document of the lower version by using the application of the lower version; receiving the modified document of the lower version from the application of the lower version; successively converting the modified document of the lower version into zero, one or more modified documents of intermediate versions and a modified document of the higher version by using the one or more differentiation models between the data models between the higher version and the lower version in the conversion stack 201, and by merging with the stored zero, one or more documents of the intermediary versions between the higher version and the lower version and the document of the higher version or the relevant information therein by a merging module 204; wherein the apparatus 200 further optionally includes a merging module 204 for merging zero, one or more incomplete documents of intermediate versions and an incomplete document of the higher version, which are successively converted from the lower version of documents, with the zero, one or more stored documents of the intermediate versions between the higher version and the lower version and the stored document of the higher version or the related information therein, so as to generate the zero, one or