

by an implementation hold all instance data described by the descriptor for that implementation.

[0054] For the purposes of the present invention, the term “inheritance” refers to the ability of one or more software models to of the present invention extend a base model in order to receive all the features and functionality of that parent model while adding new features and functionality. Models that inherit from a base model may also override features inherited from the parent model to change functionality or further restrict features. When common features and functionality may be logically grouped, but for technical reasons cannot be implemented in a parent model, an interface may be appropriate. When models share a type, either through a parent model or an interface, software processes can use these models interchangeably.

[0055] For the purposes of the present invention, the term “instance” refers to a single entity following the definition given by an implementation. The instances created by an implementation hold all instance data described by the descriptor for that implementation. The instance has access to the static data held by its implementation and can execute operations through the mechanism provided by the implementation.

[0056] For the purposes of the present invention, the term “interface” refers to a set of operations that must be implemented by a model in order for that model to participate in the interface’s type for polymorphic functionality. An interface defines operations that must be implemented by any models that participate in that interface. Interfaces are a way of communicating over the boundary between two or more components. The components know how to communicate with each other because they all understand the interface used for this communication.

[0057] For the purposes of the present invention, the term “layer” or “software layer” refers to an indirection consisting of a plurality of components related to a specific task. An example “security layer” might consist of several components used to filter or restrict access to sensitive data. The security layer serves as an indirection between the user and the data.

[0058] For the purposes of the present invention, the term “manager” refers to a mechanism for sharing components between processes to improve scalability and centralize configuration. Each manager is responsible for holding a plurality of a single type of component. These components can be reused thousands of times before shutdown and may even be used by multiple users at the same time. Managers associate a unique identity or name with each component for easy retrieval.

[0059] For the purposes of the present invention, the term “member” is a synonym for attribute.

[0060] For the purposes of the present invention, the term “metadata” refers to data that describes other data.

[0061] For the purposes of the present invention, the term “meta-implementation layer” refers to a software layer that serves as an indirection between those entities that use an implementation and the implementation itself. The meta-implementation layer allows the actual implementation to be changed or replaced with another implementation without the need for the user of the meta-implementation to be aware

of that change. A meta-implementation layer provides a connection between a descriptor and the implementation of that descriptor allowing users of the meta-implementation layer a greater understanding of the implementation and its design (the “why does it work this way”) while simultaneously buffering the user from the implementation specifics (the “how exactly does it work to accomplish that task”). There exists a one-to-one mapping between a descriptor’s parts and the parts in the meta-implementation layer for that descriptor.

[0062] For the purposes of the present invention, the term “metamodel” refers to a description of a model and is synonymous with the term “model descriptor”. A metamodel may be used to describe any model. Since a metamodel is a model that describes models, a metamodel may be constructed to describe another metamodel. A metamodel for a model may include the structure, operations, and constraints on the use of the model.

[0063] For the purposes of the present invention, the term “model view controller” refers to a design pattern used to separate data storage from business logic and display logic. The model is the object that holds the data. The view is the component that displays the data. The controller interprets input from the view to determine what action to take. It is a misconception to believe that the controller contains the business logic. The controller could more accurately be called the “input controller” since object-oriented programming should correctly place related business logic in the model.

[0064] For the purposes of the present invention, the term “model” refers to a software definition containing enough detail to be useful to a software application. A model does not need to include every detail of a real object. A model of the present invention may extend only one base model, but may also implement zero to many interfaces.

[0065] For the purposes of the present invention, the term “object oriented programming” refers to the customary definition of the term object oriented programming i.e. a type of programming in which programmers define not only the datatype of a data structure, but also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes a class that includes both data and functions. In addition, programmers can create relationships between one class and another. For example, classes can inherit characteristics from other classes.

[0066] For the purposes of the present invention, the term “object” refers to a software representation that follows the definition given by a model to represent one instance of that class. For example, if a software metamodel describes a blueprint for a “dog”, a class implements that blueprint, and an object instance can be used to hold the information for a specific dog named “Spot” with brown fur owned by “John”. Generally, an object is any item that can be individually selected and manipulated. This can include shapes and pictures that appear on a display screen as well as less tangible software entities. In object-oriented programming, for example, an object is a self-contained entity that consists of both data and procedures to manipulate the data.

[0067] For the purposes of the present invention, the term “package” refers to a logical grouping of related models that