

have different features and functionality. Models in the same package cannot solely by virtual of being in the same package be used interchangeably. Instead models in the same package are models that are likely to be used together while performing some functionality. A package does not eliminate the possibility of models implementing polymorphic behavior; it simply does not endow that characteristic.

**[0068]** For the purposes of the present invention, the term “polymorphism” refers to the ability of a software program to use different implementations that share a common base class or interface.

**[0069]** For the purposes of the present invention, the term “primitive” refers to a basic type that has no base model from which the basic type may of the present invention extend. A primitive is data used in a software program that is not an object and therefore has no class definition. A primitive is a built in part of the language and is usually simple. Primitives include Boolean, character, date, integer, number, rootclassifier, rootinterface, time, and void. Boolean types may only contain “true” or “false”; no other values are possible. Character types may contain a value representing a language character. Date types represent a day in the current calendar. Integer types are whole numbers with no fraction or decimal part. Number types may contain any numeric value. Time types contain an indicator of a particular division within a date such as hours, minutes, and seconds. The void type represents no type at all. Void is used to indicate that no type is legal or that no object is returned (from an operation). The RootClassifier type is used for base models that do not of the present invention extend any other previously defined model. The RootInterface type is used for interfaces of the present invention that do not extend from any previously defined interface.

**[0070]** For the purposes of the present invention, the term “problem domain” refers to a problem or the subject matter of interest and defines what details are included and what details are eliminated during the process of abstraction of a software definition.

**[0071]** For the purposes of the present invention, the term “relationship” refers to a link or connector between models.

**[0072]** For the purposes of the present invention, the term “resource” refers to an available reserve or supply of anything that can be drawn on when needed. Components use available resources to accomplish tasks. Resources are different from components in that they are not classifiers in the component integration engine; rather the component integration engine uses them. Examples include hard drives, network cards, memory, databases and files.

**[0073]** For the purposes of the present invention, the term “service context” refers to a logical grouping of services available for a particular group of users within a virtual host. Service groups share the resources of that virtual host, but each service context holds different services. While all users in a virtual host have access to the shared resources, the services available to make use of those resources is determined by the service context. In this way the “administration” service context might make services available to add new services to service contexts, while the “marketing” service context does not.

**[0074]** For the purposes of the present invention, the term “service” refers to a task on a server performed on behalf of a user. The type of task is determined by the service.

**[0075]** For the purposes of the present invention, the term “software definition” refers to a description of a software program that includes sufficient detail to allow the software to be implemented. Software definitions may take the form of a plurality of textual descriptions, a plurality of diagrams, a plurality of requirements, or a plurality of descriptors. A software definition may include any combination of these of these things. A software definition may capture diagrams and textual descriptions as descriptors and may translate descriptors to text descriptions and diagrams.

**[0076]** For the purposes of the present invention, the term “thread of execution” or “thread” refers to the ability of a computer to appear to be executing multiple programs or parts of a program simultaneously. The computer rapidly switches execution between threads, making it appear that the threads are executing simultaneously. Computers with more than one central processing unit may execute a thread on each central processing unit simultaneously.

**[0077]** For the purposes of the present invention, the term “type” refers to a category such as a base model, an interface, a primitive, etc. Type is synonymous with “classifier”.

**[0078]** For the purposes of the present invention, the term “virtual host” refers to a grouping of services and resources made available for a particular group of users. It serves as a logical separation when certain services and resources should be available to one particular group of users, but not another group.

**[0079]** For the purposes of the present invention, the term “virtual implementation” refers to assemblies of meta-implementations (including other virtual implementations and accessors) that behave as an implementation for a specific descriptor. A virtual implementation fulfills the implementation for a specific descriptor. Rather than creating a “real” implementation layer using source code generators and mapping the meta-implementation to this layer, a set of virtual implementations exist for each descriptor of the present invention. There is a one-to-one relationship between the virtual implementation and the descriptor for the virtual implementation. Therefore, a virtual implementation is directly traceable to the descriptors describing it. For example, a dog metamodel might describe two attributes (owner and fur color) and one operation (bark). A virtual implementation for this metamodel contains a VirtualModelImplementation (dog) assembled from two VirtualAttributeImplementations and a VirtualOperationImplementation and mapped back to the dog metamodel. The first VirtualAttributeImplementation instance holds a string value representing the owner’s name and mapped to the owner descriptor in the metamodel. The second VirtualAttributeImplementation instance holds a value of class “color” and map to the fur color descriptor. The bark operation is assembled from a VirtualDataType instance holding the value “bark” and a SystemPrintImplementation to print the word “bark” to the screen. In a similar manner, any software implementation can be created out of a very small set of virtual implementation instances without requiring code compilation. However, since the virtual implementation classes are compiled, these virtual implementations perform only slightly slower than a natively compiled implementation.

**[0080]** For the purposes of the present invention, the term “computer system” refers to any type of computer system