

present invention, these external sources are UML documents or XMI documents. In another embodiment, these are text files in a predetermined format. In yet another embodiment, these are table structures in a database. It is only required, that these “descriptions” have a format understood by a handler designed for that format and contain data related to the descriptor the handler will produce.

[0453] Many simple handlers simply copy a string value from the description to the descriptor. Other handlers create a feature descriptor of the output descriptor and set attributes based on string values in the description. In this way, a description is converted a descriptor to participate in the meta-implementation layer.

[0454] Handlers mapping a descriptor to an implementation have a very well defined input and a very well defined output. The handler constructs the output implementation, then sets the attributes, signals, operations, etc. of that implementation based on the description given in the descriptor. In many cases, the handler is able to construct a fully functional implementation based on the descriptor. In cases where the descriptor is not fully defined, the handler will complete as much of the implementation as possible based on the available data and return a warning to the user. In this way, a descriptor is converted into an implementation to be used in the meta-implementation layer and component integration engine.

[0455] Several handlers occur repeatedly in the figures. The name handler simply copies a string of input representing a name to the appropriate attribute in the output. The documentation handler simply copies a string of input representing a description to the appropriate attribute in the output. The tagged values handler copies a pair of strings, one representing a name the other representing a value, to a hint instance with a name attribute and a value attribute. The constraint handler tries to interpret the description input to construct a constraint descriptor. In many cases, the input source is insufficient to construct a constraint descriptor. In these cases the constraint handler will simply construct a constraint descriptor with the appropriate name from the input description and as much detail as possible and warn the user of the incomplete constraint.

[0456] Some descriptions will contain a string representing the preferred language in which to be implemented. This string is unimportant in the context of a meta-implementation layer, but is preserved as a hint for code generators to convert the meta-implementation to compilable source code. The language string is used as the name of a pre-defined role containing zero-to-many hints used by these code generators.

[0457] Handlers dealing with input with the string “Type” in the name use the input string to lookup a datatype, interface, or model to use as the classifier type in the handler’s output. The output consists of the identity of the type as the type is stored in the metamodel repository. Handlers dealing with input with the string “Kind” in the name use the input string to lookup a value in an enumeration. The name of the input generally matches the name of the enumeration and the value of the input matches the name of the value selected in the enumeration. For example, an “Owner scope” input with a value of “classifier” is used to lookup the “Owner scope” enumeration in the metamodel repository and select the “classifier” value from that enu-

meration as output from the handler. Several other pre-defined enumerations exist for use by the handlers including “Changeability”, “Visibility”, “Concurrency”, “Parameter Kind”, “Model Stereotype”, and “Operation Kind”. More enumerations may be defined by other embodiments of the present invention or by users of the present invention to handle other current and future descriptor inputs.

[0458] In some of the descriptors, the attribute name ends with the string “Indicator” or “Flag”. These attributes hold a Boolean (“true” or “false”) value. Handlers which create these outputs determine if the input represents a “true” value (“T”, “true”, “True”, 1, etc.) or a false value (“F”, “false”, “False”, 0, <null>, etc.) and set the attribute accordingly.

[0459] In the figures described below, implementations contain a pointer to the descriptor for which they are an implementation. By holding a descriptor, the implementation allows a user of the system to understand what the implementation does and why it does it. By providing its own metadata, the implementation allows a user to understand how the implementation does what it does. Instances also hold a pointer to the implementation that created them, allowing a user (human or otherwise) to retrieve the implementation and through the implementation retrieve the descriptor.

[0460] As shown in **FIG. 8**, Static Diagrams (Static Diagram<n>) include zero-to-many DataTypes (DataType<1>, DataType<2>, DataType<n>), zero-to-many Packages (Package<n>), zero-to-many Models (Model<1>, Model<2>, Model<n>), zero-to-many Interfaces (Interface<n>), zero-to-many Signals, and zero-to-many Associations. Each class (Class<n>) belongs to a single package (Package<n>). Some classes will implement interfaces (Class<1>, Interface<n>) but not all classes implement interfaces (Class<3>). Not all classes implement the same interfaces (Class<2> and Interface<2> and Class<1> and Interface<n>). Some classes participate in association relationships (Class<2>, Class<3>, Association<n>) but not all classes participate in association relationships (Class<1>). Some classes generate signals (Class<2>, Signal) but not all classes generate signals (Class<1>, Class<3>). Some classes receive signals (Class<3>), but not all classes receive signals (Class<2>, Class<1>). Although not pictured, classes may send or receive signals without necessarily being in an association relationship.

[0461] As shown in **FIG. 9**, the Metamodel Repository of the Component Integration Engine includes DataType Descriptors, Model Descriptors, Signal Descriptors, Enumerations, Other Element Descriptors, Package Descriptors, Interface Descriptors, Hint Descriptors, and Role Descriptors. As shown in **FIG. 9**, a Static Diagram Handler of the Component Integration Engine includes a DataTypes Handler, a Packages Handler, an Interfaces Handle, a Models Handler, an Associations Handler, and a Signals Handler.

[0462] As indicated by connector A of **FIGS. 8 and 9**, State Diagram<n> is read by Static Diagram Handler shown in **FIG. 9**. The static diagram handler is composed of a datatypes handler, packages handler, interfaces handler, models handler, associations handler, and signals handler. These handlers are described in the following sections. Each handler handles a specific description in the static diagram and creates the appropriate descriptor which is stored in the metamodel repository. The datatypes handler handles each