

tively, of the Constraint Descriptor. The Tagged Values Handler of the Constraint Descriptor Handler creates Hint Instances of the Constraint Descriptor. The Language Handler of the Constraint Descriptor Handler looks up a role in the Metamodel Repository for the language named by the language name description. The language handler retrieves the Role Implementation for that language and uses it to create one of the Role Instances of the Constraint Descriptor. The language role may be used by code-generators to generate source code from the Constraint Descriptor in the appropriate language. The language role may contain one or more hint instances to aid the code-generation process.

[0484] The Configuration of the Constraint Descriptor is read by the Constraint Impl Generator, as indicated by connector O. The OCL rule Descriptor is read by the OCL Rule Handler of the Constraint Impl Generator. The Constraint Impl Generator gets a model from the Metamodel Repository. If the model is found, an implementation is added to the Model Implementation of the Constraint Implementation. If no model is found, a model is created in the Model Implementation. The Model Implementation then adds the Model to the Metamodel Repository. The OCL Rules Handler of the Constraint Impl Generator creates an operation in the Operation Implementation of the Model Implementation. In one embodiment of the present invention, an OCL parser interprets the OCL language to construct virtual operations corresponding to the OCL language. The Model Implementation also includes Other Model Parts as described in the Model Implementations described in FIG. 30 and FIG. 31.

[0485] In FIG. 18, the Constraint Implementation includes a pointer to the Constraint Descriptor for which the Constraint Implementation is an implementation. During the creation process, the constraint implementation uses the Model Implementation to create the Model Instance used by the Constraint Instance. The constraint implementation uses the configuration from this constraint descriptor when creating constraint instances to configure the model instance held by the constraint instance. The Constraint Instance also includes a pointer to the Constraint Implementation that created it of which the Constraint Instance is an instance.

[0486] In FIGS. 20 and 21, a Method description is read by a Method Handler to create a Method Descriptor. The method descriptor is read by a Method Impl(ementation) Generator, as indicated by connector Q, to create a Method Implementation. The Method Implementation can then be used to create a Method Instance. A Name, Documentation, an Operation Kind, a Scope, Visibility, Call Concurrency, Polymorphic, Query, Return Type, Parameters, Failures, Constraints, Receptor Descriptors, Tagged Values and Code Language in the Method description are handled by a Name Handler, Documentation Handler, Operation Kind Handler, Scope Handler, Visibility Handler, Concurrency Handler, Polymorphic Handler, Query Handler, Return Type Handler, Parameters Handler, Failures Handler, Constraints Handler, Tagged Values Handler, and Language Handler, respectively, of the Method Handler. The Name Handler, Documentation Handler, Operation Kind Handler, Scope Handler, Visibility Handler, Concurrency Handler, Polymorphic Handler, Query Handler, and Return Type Handler of the Method Handler map to the Name, Description, the Operation Kind, the Scope, the Visibility, the Concurrency, the Polymorphic Indicator, the Query Indicator, and the Return

DataType Indicator, respectively, of the Method Descriptor. The Parameters Handler, Failures Handler, Constraints Handler, Tagged Values Handler, and Language Handler of the Method Handler create the Parameter Descriptors, the Failure Descriptors, the Constraint Descriptors, the Hint Instances, and the Role Instances, respectively, of the Method Descriptor.

[0487] The Parameter Descriptors, Constraint Descriptors, the Operation Descriptors, and the Signal Descriptors of the Method Descriptor are read by the Parameters Handler, as indicated by connector R, by the Constraints Handler, as indicated by connector S, by the Operations Handler, as indicated by connector T, and by the Signals Handler as indicated by the connector U, respectively, of the Method Impl Generator. The Parameters Handler, the Constraints Handler, the Operations Handler, and the Signals Handler of the Method Impl Generator create the Parameter Implementations, the Constraint Implementations, the Operation Implementations and the Signal Implementations, respectively, of the Method Implementation. The Method Implementation also includes a pointer to the Method Descriptor of which the Method Implementation is an implementation.

[0488] The Parameter Implementations, the Constraint Implementations, the Operation Implementations and the Signal Implementations of the Method Implementation create the Parameter Instances, the Constraint Instances, the Operations Instances, and the Signal Instances, respectively, of the Method Instance. The Method Instance also includes a pointer to the Method Implementation of which the Method Instance is an instance.

[0489] In FIGS. 22 and 23, an Attribute is read by an Attribute Handler to create an Attribute Descriptor. The attribute descriptor is read by an Attribute Impl(ementation) Generator, as indicated by connector V, to create an implementation of the an Attribute Implementation. The Attribute Implementation can be used to create Attribute Instances. A Name, Documentation, Visibility, Owner Scope, Multiplicity, Type, Changeability, Constraints, an Initial Value, Tagged Values and a Language Name of the Attribute description are handled by a Name Handler, a Documentation Handler, a Visibility Handler, an Owner Scope Handler, a Multiplicity Handler, a Type Handler, a Changeability Handler, a Constraints Handler, an Initial Value Handler, a Tagged Values Handler, and a Language Handler, respectively, of the Attribute Handler.

[0490] The Name Handler, Documentation Handler, Visibility Handler, Owner Scope Handler, and Multiplicity Handler of the Attribute Handler map to a Name, Description, Visibility, Owner Scope and Multiplicity, respectively, of the Attribute Descriptor. The Type Handler creates the Datatype Descriptor of the Attribute Descriptor. The Changeability Handler used the Attribute Changeability Enumeration, which is predefined in the metamodel repository, to create an access constraint to prevent certain types of changes to the attribute instances. The constraints handler creates constraint descriptors and the tagged values handler creates hint instances. The language handler reads in the name of a language to create a role descriptor used by code generators for that specific language. The initial value handler converts its input into a configuration for constructing an instance of the initial value. Generally, descriptions from an external source will limit initial values to primitives or