

association relationship with an attribute implementation that is the attribute implementation that this accessor uses to perform the actual storage and retrieval of values. An attribute accessor has a zero-to-many association relationship with access signal accessors that each provide the mechanism to register an appropriate instance's interest in receiving notification when the attribute value is accessed. Also provides the mechanism to remove interest in receiving notification for an event. The instance registering interest in receiving event notification must implement the appropriate interface to match the listener type described in the signal descriptor. An attribute accessor has a zero-to-many association relationship with prechange signal accessors that each provide the mechanism to register an appropriate instance's interest in receiving notification when the attribute value about to be changed. Prechange signal accessors also provided the mechanism to remove interest in receiving notification for an event. The instance registering interest in receiving event notification must implement the appropriate interface to match the listener type described in a signal descriptor. An attribute accessor has a zero-to-many association relationship with occurrence constraints that are each an implementation of a constraint to restrict the number of values assigned to the attribute implementation. An attribute accessor has a zero-to-many association relationship with value constraints that are each an implementation of a constraint to restrict the values allowed to be held by an attribute implementation. An attribute accessor has a zero-to-many association relationship with access constraints that are each an implementation of a constraint to restrict access to the attribute implementation unless the user holds certain credentials.

[0209] Attribute accessors may include the following operations a `getValue()` operation that returns the current value(s) after checking access constraints, a `setValue(Instance)` operation that removes any previous value and sets it to the new value after checking all constraints, `addValue(Instance)` that adds the instance given to the current values after checking all constraints, `removeValue(Instance)` that removes the given instance from the current values after checking to see if the value is being held and checking access and occurrence constraints, and `clearValues()` that remove all values after checking access and occurrence constraints.

[0210] An attribute data event is fired when the value of an attribute implementation is changed to a new value.

[0211] An operation accessor of the present invention is an accessor that provides a mechanism to execute the operation. If the operation accessor is static, an instance is not required to perform the execution. Otherwise, an instance of the model containing the operation is needed to perform the operation.

[0212] An operation accessor has a one-to-one association relationship with an operation descriptor that is the operation

descriptor for which this accessor provides access to an implementation. An operation accessor has a one-to-one association relationship with an operation that is the operation implementation that the operation accessor uses to perform the action. An operation accessor has a zero-to-many association relationship with a parameter implementation that is represented by parameter accessors. Parameter accessors provide augmented constraints above those performed by the operation implementation. An operation accessor has a zero-to-many association relationship with parameter accessors that are used by the operation accessor to impose occurrence, value, and access constraints on the parameter values. An operation accessor has a zero-to-many association relationship with access constraints that are each an implementation of a constraint to restrict access to the operation unless the user holds certain credentials.

[0213] An operation accessor may include the following may include the following operations `execute(ParameterList)` that attempts to perform the operation by executing the operation implementation. First all access and precondition constraints are checked for the operation accessor and parameter accessors. Then the operation is attempted. If the operation implementation is successful, the postcondition constraints are checked for the operation accessor and parameter accessors.

[0214] An operation event with the status of the preconditions is fired to all interested parties after `execute(ParameterList)` method performs all the precondition tests. A second event is sent at the successful completion of the operation. A third event is fired with the status of the postconditions or the operation. A fourth event is fired only if a failure occurs during the execution of the operation.

[0215] A parameter accessor of the present invention is a feature accessor that performs constraint checks on the values passed to an operation accessor. These constraints will fire a failure if they are not satisfied, canceling execution.

[0216] A parameter accessor has a zero-to-many association relationship with a parameter descriptor that is represented by the parameter accessor. Parameter accessors provide augmented constraints above those performed by the operation implementation. Since most languages do not have a separate construct for parameters, parameters refer to the value assigned to their position in the operation parameter list. A parameter accessor has a zero-to-many association relationship with precondition constraints that are each an implementation of a constraint to restrict the values allowed to be held by the parameter before execution of the operation. A precondition constraint may also enforce a constraint, like a no modification constraint, by adding a wrapper around the parameter value. A parameter accessor has a zero-to-many association relationship with postcondition constraints that are each an implementation of a constraint to restrict the values allowed to be held by the parameter after execution of the operation. A post condition constraint can also remove wrappers added by precondition constraints as necessary. A parameter accessor has a zero-to-many association relationship with occurrence constraints that are each an implementation of a constraint to restrict the number of values assigned to the parameter.

[0217] A parameter accessor of the present invention may include the following operations: a `precheck(instance)` that