

perform the precondition constraint checks and the access constraint checks against a given instance, and a postcheck-(instance) that performs the postcondition constraint checks against a given instance.

[0218] A parameter event is fired with the results of the precondition check. A second parameter event is fired with the results of the postcondition check (if the operation is successful enough to reach the post condition check).

[0219] A method accessor of the present invention is an operation accessor and a feature accessor on a model accessor. A method accessor has one-to-one association relationship with an operation descriptor that is the operation descriptor for which method accessor provides access to an implementation. A method accessor has one-to-one association relationship with a method implementation that is the operation implementation that the method accessor uses to perform the action. A method accessor has zero-to-many association relationship with signal accessors that use the signal implementations they describe to access the appropriate operations in the model implementation to register for event notifications. Signal accessors also provide 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. A method accessor has zero-to-many association relationship with precondition constraints that are each an implementation of a constraint to restrict some environmental condition or state of the model implementation before the execution of the operation. These types of constraints may check for the installation of a security manager, the existence of a database connection, or some other environmental condition. These constraints may also check to see if a model implementation has entered into the correct state to allow the execution of this operation. Constraints related to the parameter values are held by the parameter descriptors. A method accessor has zero-to-many association relationship with postcondition constraints that are each an implementation of constraints to check for certain conditions at the end of the operation. Similar to precondition constraints in checking for environmental and model related assumptions.

[0220] A method accessor may include the following operation: execute(ParameterList) that attempts to perform the method by executing the method implementation. First all access and precondition constraints are check 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.

[0221] A method accessor adds no additional events.

[0222] A signal accessor of the present invention is a feature accessor that describes a notification and provides the mechanism to register an appropriate instance's interest in receiving notification when the model generates an event. A signal accessor 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. Note that signal accessors do not require access constraints as the operations accessors impose these constraints. Also note the existence of the listener operation that was contained in the signal

descriptor. The implementation places the burden of implementing the listener operations on the listener. The signal accessor can still contain the description of how to perform those operations on a listener, rather than pushing these operation descriptors to a model accessor.

[0223] A signal accessor has a one-to-one association relationship with a signal descriptor that is the signal descriptor for which this accessor provides access to an implementation. A signal accessor has a one-to-one association relationship with a signal implementation that is the signal implementation that this accessor uses to perform the action. A signal accessor has a one-to-many aggregation relationship with listener operation accessors that call upon the operation implementation in the listener implementation. A signal accessor has a one-to-many aggregation relationship with registration operation accessors that call upon operation implementations in a signal implementation. These operations are usually operation implementations on the model implementation without a composite "signal" object held by the model. A signal accessor has a one-to-one aggregation relationship with deregistration operation accessors that call upon operation implementations in a signal implementation. These operations are usually operation implementations on the model implementation without a composite "signal" instance held by the signal source model. A signal accessor has a one-to-many aggregation relationship with listener access operation accessors that call upon operation implementations in the signal implementation. These operations are usually operation implementations on the model implementation without a composite "signal" instance held by the listener model. A signal accessor has a zero-to-many aggregation relationship with listener registration operation accessors that call upon the operation held by the signal implementation to perform registration for event notification. A signal accessor has a zero-to-many aggregation relationship with listener deregistration operation accessors that call upon the operation held by the signal implementation to remove an instance from registration for event notification. A signal accessor has a zero-to-many aggregation relationship with listener access operation accessors that call upon the operation held by the signal implementation to list all instances registered for event notification. A signal accessor has a zero-to-many association relationship with registration constraints that are each an implementation of constraints to check for certain conditions of the listener instances being registered. These constraints may also be used to restrict the type and number of listeners.

[0224] A signal accessor may include the following operations: register(Instance) that attempts to register the given instance as a listener (Registration constraints are applied to the listener before performing the registration event), deregister(Instance) that attempts to remove the given instances from listening to event notifications, and listListeners() that returns an enumeration to the listeners currently listening for events from this signal.

[0225] A registration event with the status of the registration constraints is fired to all interested parties after the register(Instance) method performs all the precondition tests. A second event is sent at the successful completion of the registration operation. Events are also fired when deregister(Instance) and listListener() methods are called.