

[0383] A version instance of the present invention is a grouping of the four data instances defined by a version implementation.

[0384] A version instances refers back to a version implementation detailing the four attribute instances it must possess.

[0385] A version instance has a one-to-one aggregation relationship with a major release attribute instance that is an instance of data that contains the value for the major release according to the major release data implementation from the version implementation. A version instance has a one-to-one aggregation relationship with a minor release attribute instance that is an instance of data that contains the value for the major release according to the major release data implementation from the version implementation. The minor release attribute is reset to its initial value (which may be zero) when the major release is incremented. A version instance has a one-to-one aggregation relationship with a build release attribute instance that is an instance of data that contains the value for the major release according to the major release data implementation from the version implementation. The build release attribute is reset to its initial value (which may be zero) when the major release or minor release is incremented. A version instance has a one-to-one aggregation relationship with a revision release attribute instance that is an instance of data that contains the value for the major release according to the major release data implementation from the version implementation. The revision release attribute is reset to its initial value (normally zero) when the major release, minor release, or build release is incremented.

[0386] A version instance may include the following operations: `incrementMajorRelease()` that is used to increase the major release version, `incrementMinorRelease()` that is used to increase the minor release version, and `incrementRevisionRelease()` that is used to increase the revision release version.

[0387] An attribute change event is fired whenever a value is added, changed, or removed.

[0388] Hint implementations must be implemented in the model descriptor layer in order to be added to the meta-model. A hint implementation has a one-to-one association relationship with a hint descriptor is the hint descriptor that describes the hint implementation. A hint implementation has a zero-to-one association relationship with a description that is a description of the purpose of the hint implementation. A hint implementation has a zero-to-many association relationship with attribute implementations that are the implementation of the various attributes to hold the values for a hint.

[0389] A hint implementation includes a `newInstance()` operation to create a new hint instance. Hints may not take constructor arguments.

[0390] An implementation change event is fired when the hint descriptor is changed.

[0391] A hint instance holds the value or values fitting the definition of its hint implementation.

[0392] A hint instance has one-to-one association relationship with an implementation that is the hint implementation that describes the hint instance. A hint instance has one-to-

many association relationship with attribute instances that are the attribute instances for capturing important details not captured by any other element of the metamodel layer.

[0393] An attribute change event is fired whenever the value held by the hint is changed.

[0394] A role implementation must be implemented in the model descriptor layer in order to be added to the meta-model. A role implementation has a one-to-one association relationship with a role descriptor that is the role descriptor that describes this implementation. A role implementation has a zero-to-many association relationship with role implementations that are storage locations for other roles that participate in this role. A role implementation has a zero-to-many association relationship with hint implementations that are storage locations for hints that participate in this role.

[0395] A role implementation may include the following operations: a `newInstance()` to create a new role instance. Roles may not take constructor arguments, `getHints()` that returns the current hints(s), `addHint(HintImplementation)` that add the hint given to the current hint implementations, `removeHint(HintImplementation)` that removes the given hint from the current hint values, `clearHints()` that remove all hints, `getRoles()` that return the current roles, `addRole(RoleImplementation)` that add the role given to the current role implementations, `removeRole(RoleImplementation)` that remove the given role from the current role values, and `clearRoles()` that remove all roles.

[0396] An implementation change event is fired whenever the hints or roles are added or removed. An implementation event is also fired when the role descriptor is changed to a different role descriptor.

[0397] A role instance of the present invention is a container for hint instances and other role instances. A role instance has a one-to-one relationship with a role descriptor that is the role descriptor that describes the role implementation. A role instance has a zero-to-many relationship with role instances that are storage locations for other role instances that participate in this role. A role instance has a zero-to-many relationship with hint instances that are storage location for a related group of hints that are applied in several locations to capture details about a model that are not captured in any other part of the metamodel layer.

[0398] A role instance may include the following operations: `getHints()` that returns the current hint instances, and `getRoles()` that return the current role instances

[0399] An attribute change event is fired whenever hints or roles are added or removed.

[0400] A component integration engine of the present invention assembles metamodels describing how a software applications work. The tool then generates a meta-implementation or a virtual implementation directly without intermediate source code and compilation. Meta-implementations are combined to perform all the necessary software processes that make up an application. These processes are made available as client-server applications or web-based applications through user-access points designed for each type of application. Other user-access points can be developed for other types of applications like email list-serve or peer-to-peer applications.