MOF2 Versioning Final Adopted Specification

This OMG document replaces the submission document (ad/05-05-11). It is an OMG Final Adopted Specification and is currently in the finalization phase. Comments on the content of this document are welcomed and due by September 30, 2005.

Issues should be directed to *issues@omg.org*. You may view the pending issues for this specification from the OMG issues web page *http://www.omg.org/issues/*.

The FTF Report for this specification will be published on February 27, 2006. Please download the appropriate document from the OMG Specifications Catalog.

OMG Adopted Specification
ptc/05-08-01

Date: August 2005

Meta Object Facility (MOF) 2.0 Versioning and Development Lifecycle Specification

Final Adopted Specification

ptc/05-08-01

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR

WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE.
IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 250 First Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

The OMG Object Management Group Logo®, CORBA®, CORBA Academy®, The Information Brokerage®, XMI® and IIOP® are registered trademarks of the Object Management Group. OMG™, Object Management Group™, CORBA logos™, OMG Interface Definition Language (IDL)™, The Architecture of Choice for a Changing World™, CORBAservices™, CORBAfacilities™, CORBAmed™, CORBAnet™, Integrate 2002™, Middleware That's Everywhere™, UML™, Unified Modeling Language™, The UML Cube logo™, MOF™, CWM™, The CWM Logo™, Model Driven Architecture™, Model Driven Architecture Logos™, MDA™, OMG Model Driven Architecture™, OMG MDA™ and the XMI Logo™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.htm).

Table of Contents

1 Scope

During a system's evolution it will go through a number of iterations, driven by changes to a model at some level. The model for an existing deployed system is a valuable asset for understanding that system and making minor changes for bug fixes etc. So it is important to preserve it when making broader changes for new functionality. The problem being addressed by this Specification is to manage the co-existence of multiple versions of such metadata in a Meta Object Facility and their inclusion in different configurations (for example a specific version of a PIM, the corresponding version of the derived PSM, and the corresponding version of the generated system).

A major problem related to the presence of multiple versions is to determine the correct one to use in response to a client request such as obtaining all instances of Package (it would not generally be useful to return many different versions of the same package), or even accessing the Features of a Class (again it would not be useful to return all Features the class has ever had, especially when some have replaced others; and yet again it would not be sensible to return all versions of those Features). An important requirement for version management is support of context: this is a client-specified property related to metadata access requests that is used to select the appropriate versions.

A related problem is the lack of a consistent mechanism for determining the status of versions of models or artifacts with respect to a development lifecycle (for example draft->approved->unit-tested->system-tested->live->withdrawn). This is often the most useful way of specifying a context (see previous paragraph) and to control the operations that may be applied to the metadata (e.g. to prohibit changes to the model for the 'live' version).

2 Conformance

A compliant implementation must implement the APIs generated by applying a MOF language binding to the metamodels in this specification.

The following are optional compliance points:

- Support for version selection by Time
- Support for selection by State

3 Normative References

MOF 2.0 Core Specification: ptc/04-10-15 or successor formal document

SPEM 1.1 Specification: formal//05-01-06

4 Additional Information

4.1 Acknowledgements

The following companies submitted and/or supported parts of this specification:

- Adaptive
- International Business Machines
- MetaMatrix
- Unisys

4.2 Proof of Concept

This specification represents the implementation in Adaptive's Adaptive Repository™ product: a MOF compliant repository that has been extended with this versioning capability and, as part of Adaptive's IT Portfolio Manager ™ linked with SPEM to add a development lifecycle capability. The technology has been in the market for more than 2 years.

This specification is also designed to reflect the experience gained from standardizing different versioning products as part of JSR 147 Workspace Versioning and Configuration Management http://www.jcp.org/aboutJava/communityprocess/review/jsr147/.

4.3 Changes or Extensions to OMG Specifications

There are no required changes or extensions to OMG specifications.

5 Metamodel Specification

This section outlines the public elements of the Versioning package, which is defined as a MOF metamodel depicted in the following UML diagrams. Note that the versioning API is automatically generated from this metamodel, following the normal MOF generation rules which will be specific to the language mapping and so are not contained in this specification. The remainder of this section explains the classes in the metamodel.

Note: Where 'isReadOnly = true' there are generally specific operations for updating the value either directly or indirectly.

5.1 Package Dependencies

The dependencies with other packages are depicted here.

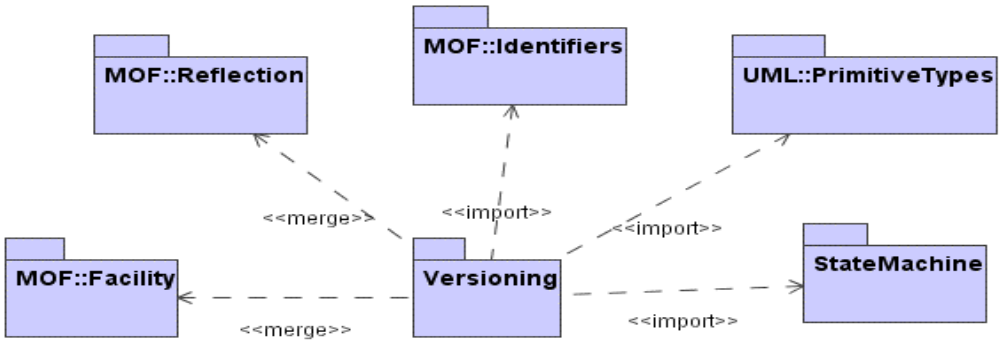

Figure 5.1 - Package Dependencies

Packages Versioning and StateMachine are defined by this specification.

5.2 Versioning Diagram

The package provides the core versioning service. A VersionHistory and an initial Version is created automatically when a new VersionedExtent is created. The VersionHistory of a given VersionedExtent contains all versions of that VersionedExtent.

A VersionedExtent is the object that determines which Version from a given VersionHistory is currently selected in a given Workspace, and whether or not changes are being made to that extent in that workspace (i.e., whether or not that VersionedExtent is checked out). To provide key performance optimizations (such as copy based Workspaces), each Workspace must have its own set of VersionedExtents. This also ensures that each Workspace can decide independently which Version that is selected and whether or not that VersionedExtent is checked out. So each Workspace that wants to select a Version from a given VersionHistory has its own VersionedExtent for that VersionHistory, which means a VersionHistory is related to more than one VersionedExtent (in particular, is related to a different VersionedExtent for each different Workspace that selects a Version from that VersionHistory).

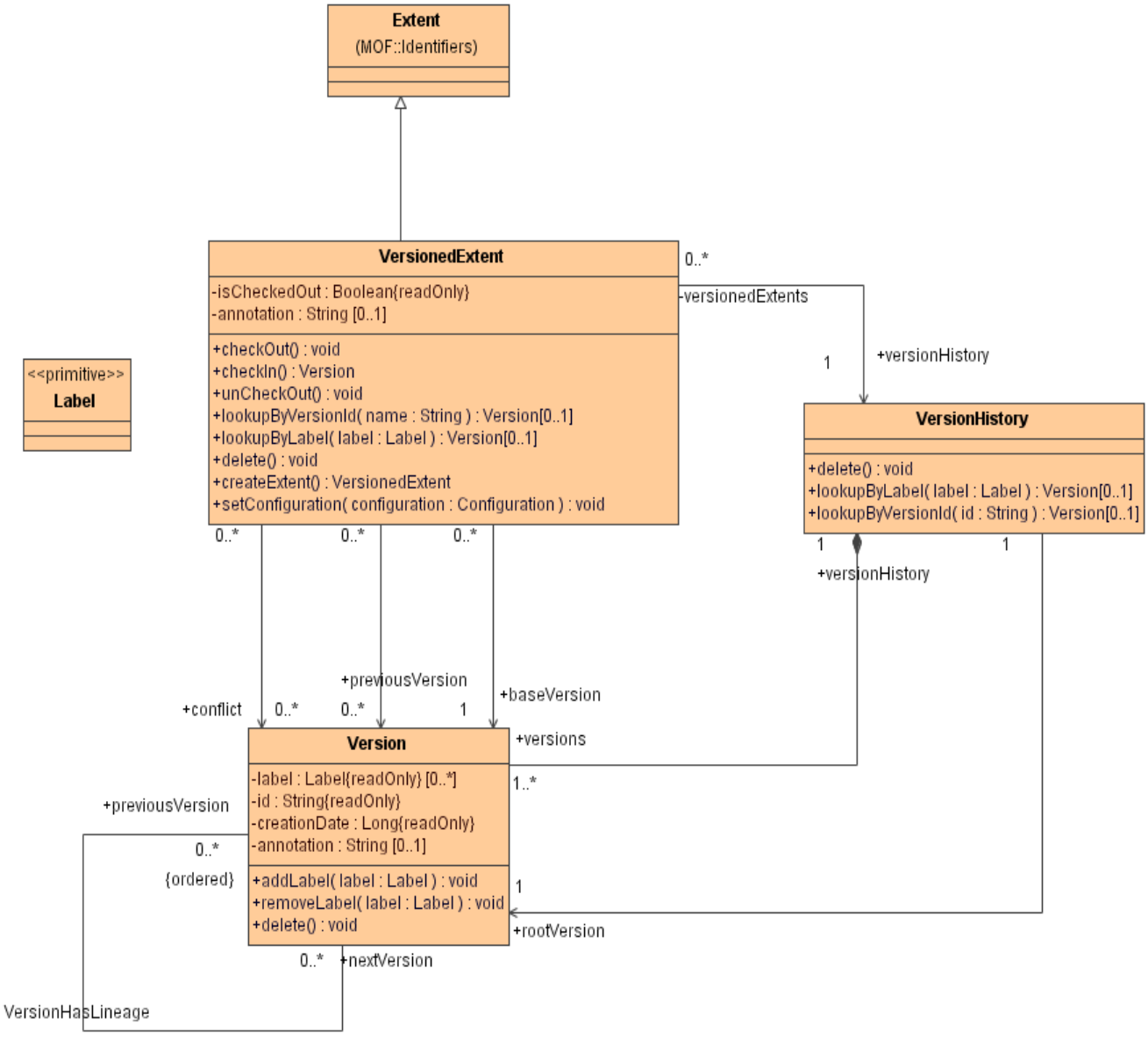

Figure 5.2 - Versioning Diagram

5.3 Class Version

Each instance of Version is a snapshot of a VersionedExtent.

5.3.1 Attributes

id, type = *String*, isReadOnly = true

- This attribute is an id for the version that is unique within the VersionHistory of that version.
- The format of the string is implementation-defined.

label, type = *Label*, multivalued [0..*], isReadOnly = true

- A user assigned string that can be used to identify the version. Several can be applied to one version. Also, version labels must be unique among all the version labels within the VersionHistory of that version.

annotation, type = *String*, [0..1]

- This attribute can contain a description of the version.

creationDate, type = *Long*, isReadOnly = true

- Used to record the timestamp when the Version object was created.

5.3.2 References

versionHistory, type = *VersionHistory*, isReadOnly = *true*

- The VersionHistory for this Version.

previousVersion, type = *Version*, [0..*], isReadOnly = true, inverse = *nextVersion*

- This refers to the immediately previous version (or versions, in case this Version resulted from a merge). Every Version that is not the root version of a VersionHistory has at least one previousVersion.

nextVersion, type = *Version*, [0..*], isReadOnly = true, inverse = *previousVersion*

- This refers to ordered subsequent versions.

baseline, type = *Baseline*, [0..*], isReadOnly = true, inverse = *version*

- This refers to the Baselines that reference this Version.

5.3.3 Operations

addLabel (*newLabel: Label*) operation, return type = *void*

- Adds a label to the version. If that label currently appears on another version in the version history of this Version, it is automatically removed from that other version. See the *label* attribute for further details.

removeLabel (*oldLabel: Label*) operation, return type = *void*

- Removes the indicated label from the version. See the label attribute for further details.

delete (), return type = *void*

- Removes the version object. This method also fixes up the links in the version graph so that it is still valid.

5.4 Class VersionHistory

A VersionHistory contains all versions of a given VersionedExtent.

5.4.1 References

rootVersion, type = *Version*, isReadOnly = true

- This refers to the initial version of the VersionedExtent. The rootVersion has no previousVersion. All other versions have at least one previousVersion.

versions, type = *Version*, [1..*], isReadOnly = true

- This refers to all versions in this VersionHistory, including the rootVersion. Ordering of the versions is defined by the previousVersion/nextVersion relations.

5.4.2 Operations

lookupByVersionId (*id : String*), return type = *Version*, [0..1], isQuery = true

- Returns the Version, if any, in this VersionHistory having the specified *id*.

lookupByLabel (*label : Label*), return type = *Version*, [0..1], isQuery = true

- Returns the version, if any, in this VersionedHistory having the specified *label*.

delete (), return type = *void*

- Removes the VersionHistory object. This method also deletes all Versions in the VersionHistory.

5.5 Class VersionedExtent

Represents a MOF Extent for which versioning has been enabled.

5.5.1 Superclasses

MOF:Extent

5.5.2 References

configuration, type = *Configuration*, isReadOnly = true

- The Configuration to which this VersionedExtent belongs.

5.5.3 Operations

checkOut () operation, return type = *null*, isQuery = false

- Allows modifications to be performed on the content of this VersionedExtent.

checkIn (), return type = *Version*, isQuery = false

- Creates a new Version object with the current content of this VersionedExtent, disallows modifications to be made to the content of this VersionedExtent until it is subsequently checked out. This Version is now available to other Workspaces.

lookupByVersionId (*id : String*), return type = *Version*, [0..1] isQuery = true

- Returns the version, if any, of this VersionedExtent having the specified *id*.

lookupByLabel (*label : String*), return type = *Version*, [0..1], isQuery = true

- Returns the version, if any, of this VersionedExtent having the specified *label*.

delete (), return type = *void*

- Removes the VersionedExtent object.

setConfiguration (*configuration : Configuration*), return type = *void*

- Changes the configuration to which this VersionedExtent belongs.

5.5.4 Attributes

isCheckedOut, type = *Boolean*, isReadOnly = true

- A VersionedExtent is modifiable only if isCheckedOut is true.

annotation, type = *String*, [0..1]

- When isCheckedOut is true, this attribute contains the annotation attribute for the version that will be created when this VersionedExtent is checked in. If the isCheckedOut is false that attribute does not exist and cannot be written to.

5.5.5 References

versionHistory, type = *VersionHistory*, isReadOnly = true

- The VersionHistory that contains the versions for this VersionedExtent.

baseVersion, type = *Version*, isReadOnly = true

- The currently selected version for this VersionedExtent. This is the version that this VersionedExtent is currently based on, but when checked-out, is not necessarily equal to. In particular, the base version of a checked-out VersionedExtent is the Version content that was started with (i.e, that the current checked-out content is 'based on').

previousVersion, type = *Version*, [1..*], isReadOnly = true

- When isCheckedOut is true, this reference identifies the versions that will be the previousVersion for the version that will be created when this VersionedExtent is checked in.

conflict reference, type = *Version*, [0..*], isReadOnly = true, composite

- These are the conflicts created by a merge. After the client resolves the conflict (by appropriately modifying the content of the VersionedExtent), the client indicates that the conflict has been resolved by adding the version to the previousVersion list of the VersionedContext, and removing the version from the conflict list.

workspace, type = *Workspace*, [1], isReadOnly = true

- This reference identifies the Workspaces that contain this VersionedExtent.

5.6 Context Diagram

This section defines a metamodel for selecting specific Versions of Extents.

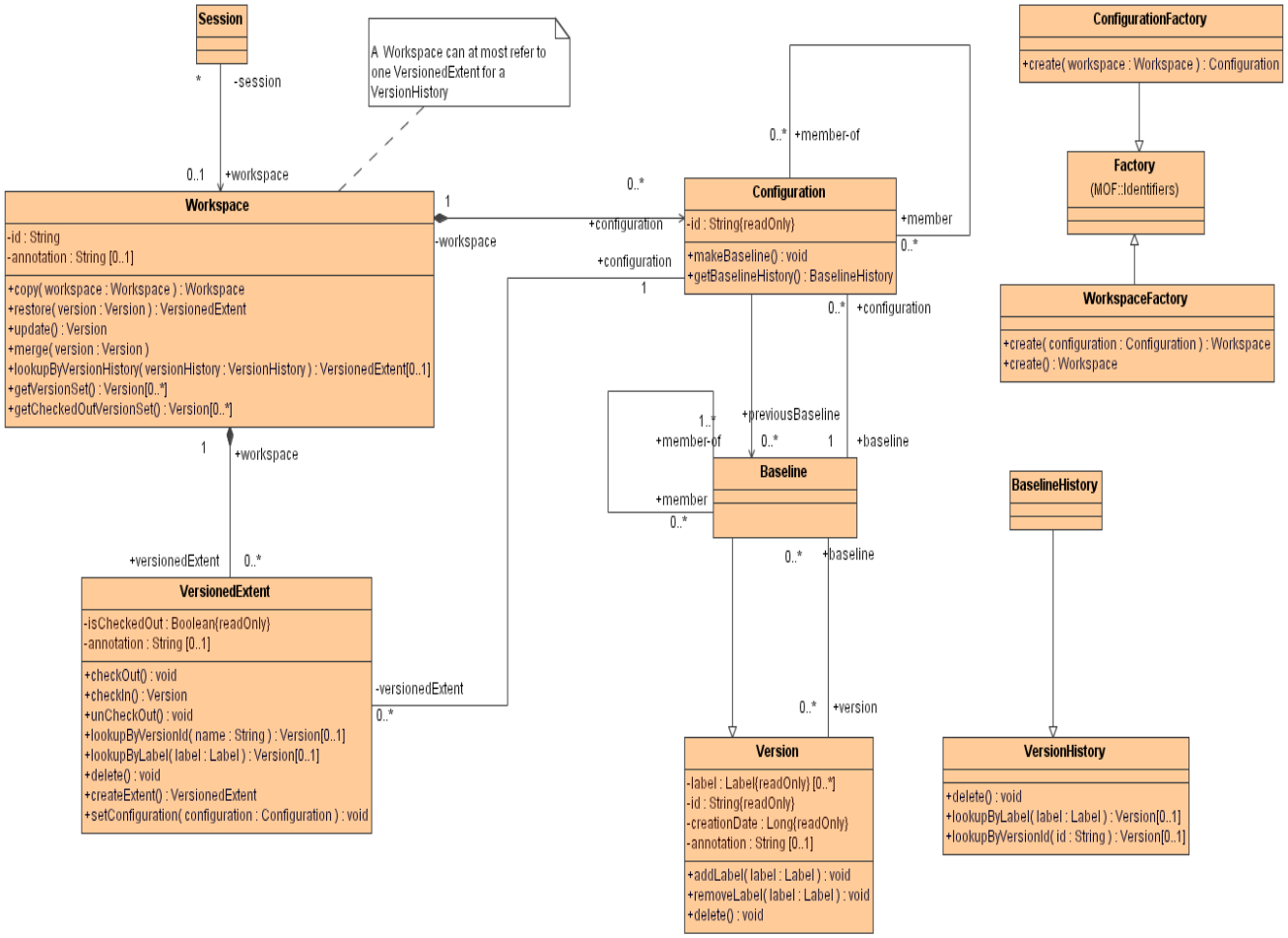

Figure 5.3 - Context Diagram

5.6.1 Class Session

This logically represents the client's current session. By first attaching and selecting the Workspace through the Session, the other MOF 2 APIs can transparently access specific versions without being aware of it.

5.6.2 Class Workspace

This selects a set of versions that are accessed together for a Session or Sessions. Unlike Session this is a persistent object.

A Workspace selects versions via VersionedExtent objects.

An important constraint is that a Workspace may contain at most one VersionedExtent for a given VersionHistory and at most one Configuration for a given BaselineHistory.

5.6.2.1 References

versionedExtent, type = VersionedExtent, [0..*], isReadOnly = true

- This refers to all VersionedExtents in this Workspace (inclusive of checked out extents).

configuration, type = Configuration, [0..*], isReadOnly = true

- This refers to all Configurations in this Workspace.

5.6.2.2 Operations

copy (*workspace : Workspace*), return type = *Workspace*, isQuery = false

- Returns a new workspace that selects the same configuration of Versions and Baselines as the specified *workspace*.

restore (*version: Version*), return type = *VersionedExtent*, isQuery = false

- Create a new VersionedExtent in this Workspace for the specified Version, or if the Version is a Baseline, a new Configuration in this Workspace for the specified Baseline. This Workspace must not already select a Version for the VersionHistory of the specified Version.

update (*version : Version*), return type = *null*, isQuery = false

- Updates the VersionedExtent in this Workspace for the VersionHistory of this Version to select the specified Version, or if the Version is a Baseline, updates the Configuration in this Workspace for the BaselineHistory of this Baseline to select the specified Baseline (which then updates the Workspace with each of the versions in that Baseline).

merge (*version : Version*), return type = *null*, isQuery = false

- If the Version is not a Baseline, this merges the specified Version into the VersionedExtent in this Workspace for the VersionHistory of this Version. If the version is a predecessor or equal to the baseVersion of the VersionedExtent, the VersionedExtent is left unmodified. If the version is a successor of the baseVersion of the VersionedExtent, the VersionedExtent is updated to select the version. If the version is not a predecessor of, equal to, or a successor of the baseVersion of the VersionedExtent, the specified version is added to the conflict attribute of the VersionedExtent.
- If the Version is a Baseline, this merges the specified Baseline into the Configuration in this Workspace for the BaselineHistory of this Baseline. If the Baseline is a predecessor or equal to the +baseline of the Configuration, the Configuration is left unmodified. If the Baseline is a successor of the +baseline of the Configuration, the Configuration is updated to select the Baseline. If the Baseline is not a predecessor of, equal to, or a successor of the +baseline of the Configuration, the specified Baseline is added to the previousBaseline property of the Configuration, and each of the versions of the specified Baseline are merged into the Workspace.

lookupByVersionHistory (*versionHistory : VersionHistory*), return type = *VersionedExtent*, [0..1] isQuery = true

- Returns the VersionedExtent, if any, in this Workspace for the specified *versionHistory*.

getVersionSet (), return type = *VersionedExtent*, [0..*] isQuery = true

- Returns the VersionedExtents, inclusive of those that are checked out, that are being referenced by this Workspace.

getCheckedOutVersionSet (), return type = *VersionedExtent*, [0..1] isQuery = true

- Returns only the VersionExtents referenced by this Workspace that have a status of checked out.

5.6.3 Class Configuration

Selects a Baseline for the Workspace and determines the other Configurations that are the members of this Configuration.

5.6.3.1 Attributes

annotation, type = *String*

- The annotation of a new Configuration that is used for the annotation of a new Baseline that is created through the makeBaseline operation.

5.6.3.2 References

baseline, type = Baseline, [1..1], isReadOnly = true

- The Baseline that is used by this Configuration to select a set of Versions.

previousBaseline, type = *Baseline*, [1..*], isReadOnly = true

- When isCheckedOut is true, this reference identifies the baselines that will be the previousVersion for the baseline that will be created when makeBaseline is applied to this Configuration.

member, type = Configuration, [0..*], isReadOnly = true, inverse = member-of

- Configurations which are logically included in this Configuration.

member-of, type = Configuration, [0..*], isReadOnly = true, inverse = member

- Configurations which logically include this Configuration.

versionedExtents, type = VersionedExtents, [0..*], isReadOnly = true, inverse = configuration

- The VersionedExtents that belong to this Configuration.

5.6.3.3 Operations

makeBaseline (), return type = *void* isQuery = true

- Creates a new Baseline based upon the Versions selected in the Workspace by the versionedExtents of this Configuration. This also copies the annotation into the new Baseline and removes the annotation from the Configuration.

getBaselineHistory (), return type = *BaselineHistory* isQuery = true

- Returns the BaselineHistory for this Configuration.

5.6.4 Class Baseline

Selects a set of Versions, with no two versions from the same VersionHistory.

5.6.4.1 Superclasses

Version

5.6.4.2 References

configuration, type = Configuration, [0..*], isReadOnly = true, inverse = baseline

- Configurations that refer to this Baseline.

member, type = Configuration, [0..*], isReadOnly = true, inverse = member-of

- Baselines which are logically included in this Baseline.

member-of, type = Configuration, [0..*], isReadOnly = true, inverse = member

- Baselines which logically include this Baseline.

5.6.5 Class BaselineHistory

A BaselineHistory contains all the Baselines, just as a VersionHistory contains all Versions of a given Versioned Extent.

5.6.5.1 Superclasses

VersionHistory

5.6.6 Class WorkspaceFactory

Creates instances of Workspace

5.6.6.1 Superclasses

MOF::Identifiers::Factory

5.6.6.2 Operations

create (), return type = *Workspace*, isQuery = false

- Returns a new workspace that selects no Versions.

create(*configuration : Configuration*), return type = *Workspace*, [0..1] isQuery = false

- Creates a Workspace based upon a Configuration. The Versions that are returned from the Configuration Baseline are restored into a new Workspace and Configuration. The new Workspace Configuration has its base set to the Configuration supplied.

5.6.7 Class ConfigurationFactory

Creates instances of Configuration.

5.6.7.1 Superclasses

MOF::Identifiers::Factory

5.6.7.2 Operations

create (*workspace : Workspace*), return type = *Configuration*, isQuery = false

- Returns a new Configuration with no Baseline for the specified Workspace.

5.7 Development Lifecycle Diagram

This links to a very simple metamodel containing just two classes to represent the current State of an Element; and to represent the Lifecycle related to a class. This could be left very simple and stand-alone or it could be merged/extended with something like the SPEM 1.1 metamodel.

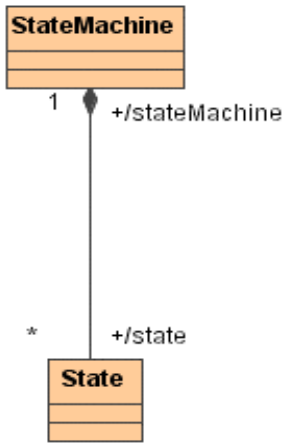

Figure 5.4 - State Machine Package

Above is the StateMachine package. Note that the properties are derived - hiding exactly how a more detailed metamodel might link States to a Statemachine.

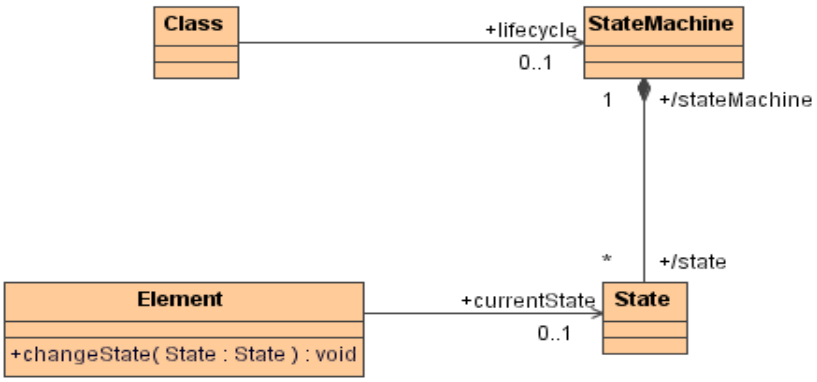

Figure 5.5 - State Machine Package with Properties

The Properties added by this specification are shown here: note that no Properties are added to the StateMachine package.

Class and Element are merged with MOF::Reflection allowing any element to have a State and any (meta)Class to have an associated StateMachine as its Lifecycle.

This shows that a lifecycle StateMachine may be associated with a MOF Class, and each instance (represented by MOF's Reflective::Element) may have a State from that lifecycle.

5.7.1 Class Element

5.7.1.1 Attributes

None

5.7.1.2 References

currentState: State {readOnly}

- This represents the current state of development of the element (at this version) within the lifecycle associated with its class.

5.7.1.3 Operations

changeState (state : State)

- This operation is required to change the state of an object. This enforces any permitted transitions defined by the StateMachine.

5.7.1.4 Constraints

The currentState of an Element is a member of the states in the Lifecycle for that Element's metaclass.

inv: getMetaclass().lifecycle.state includes self.state

5.7.2 Class Class

5.7.2.1 Attributes

None

5.7.2.2 References

lifecycle: StateMachine

- This represents the set if States (and possibly transitions) that the instances of the class may undergo.