

Software Portfolio Management Facility (SPMF) Specification bei/2004-01-03

***Second Revised
Submission***

Submitter:
Adaptive

Supported By:
Interoperability Clearing House

January 12, 2004

Table of Contents

| | |
|--|------------|
| 1. Copyright Waiver | ii |
| 2. Contacts for the submission | iv |
| 2.1 Primary Contact..... | iv |
| 2.2 Acknowledgments..... | iv |
| 3. Structure of This Submission | v |
| 4. Overall Design Rationale | 4-1 |
| 5. Proof of Concept | 4 |
| 6. Resolution of RFP Requirements | 5 |
| 6.1 Mandatory Requirements..... | 5 |
| 6.2 Optional Requirements | 9 |
| 7. Glossary and References | 10 |
| 8. Proposed Metamodel | 11 |
| 8.1 ExtendedPrimitiveTypes Package..... | 11 |
| 8.2 Linkage Package | 11 |
| 8.3 SoftwarePortfolio Package | 14 |
| 9. Kind Library | 32 |
| 10. Compliance Points | 58 |
| 11. Changes or extension to existing OMG Specifications | 58 |

PART I

1. Copyright Waiver

The company listed above hereby grants a royalty-free license to the Object Management Group, Inc. (OMG) for worldwide distribution of this document or any derivative works thereof within OMG and to OMG members for evaluation purposes, so long as the OMG reproduces the copyright notices and the below paragraphs on all distributed copies.

The company listed above has 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.

The copyright holder 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.

NOTICE : The information contained in this document is subject to change with notice.

The material in this document details a submission to the Object Management Group for evaluation in accordance with the license and notices set forth on this page. This document does not represent a commitment to implement any portion of this specification by the submitter.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, THE OBJECT MANAGEMENT GROUP AND THE COMPANY LISTED ABOVE MAKE NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. The Object Management Group and the company listed above shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

The copyright holder 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.

This document contains information that is patented which is protected by copyright. All Rights Reserved. No part of the work covered by copyright hereon 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. All copies of this document must include the copyright and other information contained on this page.

The copyright owner grants member companies of the OMG permission to make a limited number of copies of this document (up to fifty copies) for their internal use as part of the OMG evaluation process.

RESTRICTED RIGHTS LEGEND. Use, duplication, or disclosure by government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Right in Technical, Data and Computer Software Clause at DFARS 252.227.7013

OMG® is a registered trademark of the Object Management Group, Inc.

2. Contacts for the submission

2.1 Primary Contact

Pete Rivett
Adaptive Ltd.
Dean Park House, 8-10 Dean Park Crescent
Bournemouth
BH1 1HL
UK
Tel: +44 (0)1202 449419
Fax: +44 (0)1202 449448
Email: pete.rivett@adaptive.com

2.2 Acknowledgments

The following colleague at Adaptive has helped review and improve the metamodel on which this proposal has been based:

Nick Dowler

Thanks are also due to other consultants at Adaptive and its customers who have contributed to the ideas behind this proposal over a number of years.

3. Structure of This Submission

The submission follows the OMG guidelines for structuring submissions: Part I is introductory material related to the submission process, Part II is the normative

specification and Part III is the impact on other OMG Specifications (which in this case is none).

Part II is a MOF 1.4 metamodel depicted using UML class diagrams. The normative form of the metamodel is the MOF XMI file that accompanies the submission. Part of the specification is also a normative XMI DTD for the interchange format. In addition a XMI document is provided for the library of Kind elements.

4. Overall Design Rationale

Unlike the previous revision, which attempted a detailed model for each of the areas, the fundamental approach taken has been to supply only a core model that can be linked to any external model (or not at all). A Linkage package is specified for supporting this linking. In the future a user of a MOF 2.0 version of SPM could use Model merging to incorporate the SPM classes directly into their metamodels.

The Facility is designed to be extensible and so 'Kind' objects (which are akin to UML Powertypes) are used extensively. To provide commonality of support for frequently used elements, a library of Kind instance elements is also specified.

5. Proof of Concept

Adaptive Ltd. started life as a large in-house project in a major bank, tasked with building an Organization Design Facility to provide dynamic management of the bank's People, Process and System structures following a major Business Process Engineering exercise. During that time it constructed a generic metamodel-driven, repository-supported, software base for building and interacting with organizational information at all levels. Since a management buyout at the start of 1997, Adaptive has taken this software, Adaptive Framework™, to market and has worked with a wide variety of customers in modeling and managing their systems, processes and organizations.

This proposal is firmly based on product functionality and a common set of metamodels that have been deployed at more than 20 blue-chip customer sites. This has now been packaged as a product **Adaptive IT Portfolio Manager™**. Compared with the product this proposal just represents the core linking elements.

6. Resolution of RFP Requirements

This section describes how this submission meets the key requirements identified in the RFP. The section numbers in the left hand column refer to the RFP. In some cases the statement of requirement has been summarized, so this should not be taken as the definitive definition.

6.1 Mandatory Requirements

6.1.1 Modeling elements

These aspects are all addressed in the metamodel.

Submissions shall:

- 6.5.1.1 *Define the elements used to describe and manage the software portfolio: e.g., application, application component, units of software delivery etc.*
This is represented through the class SoftwareElement.
- 6.5.1.2 *Define the relationships among the elements, which will be of various types, including but not limited to: mappings, dependencies, trace links, associations (these terms are as defined in UML)*
These links appear throughout the metamodel.
- 6.5.1.3 *Provide a semantic model over the software portfolio and related elements specifying the types of relationships among entities and all multiplicities.*
SPMF is defined as a MOF metamodel with all multiplicities defined.
- 6.5.1.4 *Support unrestricted multiple levels of granularity and composition (bill of materials) and allow the user to choose which to use.*
Generally the important metamodel elements in each area support decomposition e.g. SoftwareElement, BusinessProcess, InformationElement.

6.1.2 Mapping from Software Portfolio to other elements

These aspects are all addressed in the metamodel.

Submissions shall define the relationships of software portfolio elements to the following elements, without providing for detailed definition of those elements and their internals or constraining the level of refinement at which the reference is made (e.g. geographical information shall permit references at level of cities or down to individual desks):

- 6.5.2.1 *Business processes which are supported by the software elements*
This is addressed through the class BusinessProcess which may be used to link to a full process model such as BPDm.
- 6.5.2.2 *Data stores and streams external to the software application elements – for example corporate databases, data feeds from partners*
This is addressed through the class InformationElement which may be used to link to a full information model such as CWM.
- 6.5.2.3 *Geographical/site information – for example cities, offices, individual desks*
This is addressed through the class Location which is used to link to a full location model.

- 6.5.2.4 *Responsible parties within IT Operations*
This is addressed through the classes Party, Interest, Agreement, License, ContactInfo.
- 6.5.2.5 *Business organization*
This is addressed through the class Party which may be used to link to a full organization structure model .
- 6.5.2.6 *Projects that affect the Software Portfolio*
This is addressed through the class ImpactElement which may be used to link to a full change management model
- 6.5.2.7 *Technology types and vendor products*
This is addressed through the class XXX.

6.1.3 Management

- Submissions shall:*
- 6.5.3.1 *Provide the information necessary for organizational ownership (at individual or organization unit level) of any element. This shall be available in the same way as other mapping information.*
This is addressed by the Interest class which can apply between any ManagedElement and any Party and which provides for different types of ownership.
- 6.5.3.2 *Provide the information necessary to facilitate access control at the element instance level using standard security mechanisms. This is because different aspects of the portfolio will typically be managed by different parts of the organization.*
This can also be addressed by the above Interest class.
- 6.5.3.3 *Provide the information necessary to facilitate notifications whereby individuals are informed of proposed or actual changes to elements in which they have expressed an interest. For example, it is expected that in most cases owners will wish to be informed of changes. This shall include changes to linked elements. The mechanism shall provide the ability to register interest at various levels of granularity at which this should be specified - for example whether it should be for specific relationships (“inform me if there are any changes to the mapped Business Process”) or a specified level of indirection (“inform me if anything linked to this element by up to 2 links changes”). The facility shall provide a means of defining and recording events, and subsequently triggering them.*
This area is being specified by the MOF 2.0 Facility and Object Lifecycle RFP ad/03-01-35 and so is not addressed by this submission.

6.1.4 Views and reporting

- Submissions shall:*
- 6.5.4.1 *Provide the information necessary to facilitate aggregated, high-level information views across a number of different linkages or mappings, including end-to-end traceability of the linkages.*
This area is being specified by the MOF 2.0 Queries, Views and Transformations RFP ad/02-04-10 and so is not addressed by this submission. However this submission does specify the linkages, and recursive decompositions, on which such views may be defined.
- 6.5.4.2 *Provide the information necessary to facilitate expressing complex queries including queries over relationships and mappings.*
See response to previous requirement.

6.1.5 Maintenance

- Submissions shall address the issue of how the information in the Software Portfolio is captured and maintained, and specifically the following points:*
- 6.5.5.1 *How semantic linkages to external elements are kept up to date with or reconciled with the external systems that source those elements*
Links to external systems are explicitly represented via ExternalLink and Externalsystem Objects. It would therefore be possible to create an agent that, knowing about an external system change, used MOF capabilities to find the

- ExternalReferences to it and performed whatever synchronization was needed – which might include deleting the SPM object.**
- 6.5.5.2 *How clients of the SPM Facility discover the currency of the information*
The link to the external system is explicitly modeled, together with a unique locator for the external object – so these can be interrogated, assuming appropriate interfaces to the external system.
- 6.5.5.3 *How the lifecycle of elements is managed*
Each ManagedElement can be linked to a LifecycleState which can be linked to a detailed process definition such as in SPEM.
- 6.5.5.4 *Allow semantic equivalence to be mapped (e.g. different suppliers have different descriptions of the same underlying capability)*
This is not addressed.

6.1.6 Submission Compatibility

- Submissions shall:*
- 6.5.6.1 *Address overlaps in functionality with other specifications as mentioned in 6.4, by using the other specification's interface or by explaining intentional differences, where appropriate.*
See section 4, Rationale: the approach is not have a fairly minimal core model that may be used to link to existing models but not overlap with them.
- 6.5.6.2 *Use the MOF as the meta-metamodel, the UML Notation as the graphical notation (if required), and the XMI as the stream-based interchange format*
This is the approach taken.

6.2 Optional Requirements

6.2.1 Change Management

- Submissions may:*
- 6.6.1.1 *Provide time-based views that can be used to show both element history and a future 'roadmap' of proposed changes. For the latter it should be possible to construct and compare different alternative roadmaps.*
This is covered by the MOF 2.0 Versioning RFP so has not been addressed by this submission.
- 6.6.1.2 *Provide a project management or planning capability that allows specific changes to be linked to tasks in a project plan. This may offer facilities for:*
- one- or two-way interchange with project planning tools
 - visibility of progress
 - visibility of cross-project dependencies
- This is provided in a generic manner through the ImpactElement class.**

PART II

7. Glossary and References

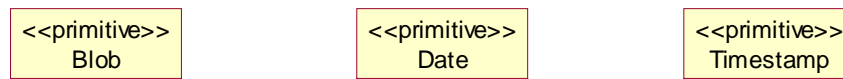
| | |
|-------------|---|
| BPDM | Business Process Definition Metamodel - in-process OMG specification: RFP is bei/03-01-06. |
| CCA | Collaborating Component Architecture – part of EDOC (qv) covering components at any level of granularity and their composition. |
| CWM | Common Warehouse Metamodel.- despite its name covers all all aspects of information resources. Available OMG technology: formal/01-10-01 |
| EAI | UML Profile and Interchange Models for Enterprise Application Integration. Adopted OMG technology: ptc/02-02-02 |
| EDOC | UML Profile for Enterprise Distributed Object Computing – key specification bridging business and component architectures. Adopted OMG Technology: ptc/02-02-05 |
| MOF | Meta Object Facility. Available OMG technology: formal/02-04-03 |
| ODP | Open Distributed Processing. Architectural approach advocated by International Standards Organization (ISO). |
| RAS | Reusable Asset Specification covering the classification and cataloging of (usually software) assets. Undergoing adoption through the Request For Comments process: ad/03-10-10 http://www.rational.com/rda/ras/preview/index.htm |
| SPMF | Software Portfolio Management Facility. This specification. |
| SPEM | Software Process Engineering Metamodel. Available OMG technology: ptc/01-12-06. Covers how software development is organized and representation of the formal processes deployed. Covers organization, planning and responsibilities. |
| UML | Unified Modeling Language, Available OMG technology: formal/01-09-67. Originally designed for object-oriented analysis and design, it has evolved into a general-purpose language for modeling information and systems. It is the core of many different modeling efforts including CWM and SPEM. It is MOF compliant. |
| XMI | XML Metadata Interchange. Available OMG technology: formal/00-06-01 Defines a mapping of any MOF-compliant metamodel to a XML Document Type Definition and content documents. |
| XML | eXtensible Markup Language. A very widely used tag-based format for exchanging information. |

8. Proposed Metamodel

The metamodel is illustrated through a set of UML diagrams. In fact the metamodel is a MOF 1.4 metamodel represented via UML using the UML Profile for MOF (documented in EDOC).

8.1 ExtendedPrimitiveTypes Package

In order to represent the business information and artifacts involved in SPM, it is necessary to add additional datatypes to those provided by MOF. These are contained in a separate package which is imported by the SoftwarePortfolio package as follows:



8.1.1 Date

This represents dates. In a language binding it should be mapped to a type that allows ordered comparison. For XMI it is mapped to the XML Schema **date** type.

8.1.2 Timestamp

This represents a point in time: for example a combination of a date and a time within the day. For XMI it is mapped to the XML **dateTime** type

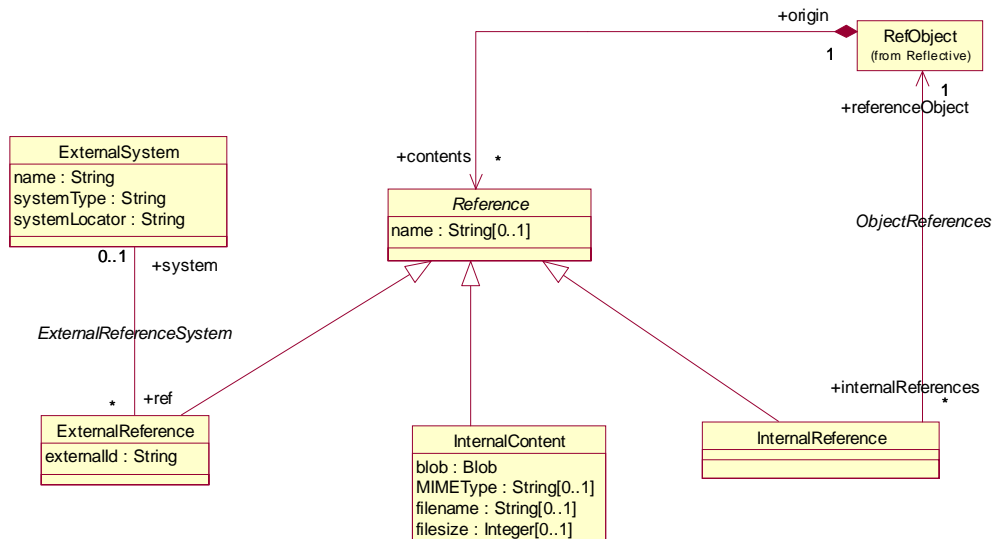
8.1.3 Blob

This represents an opaque Binary Large Object as supported by most database systems. For XMI it is mapped to a href referencing an external file which will typically be local and the name of which will typically be system-allocated. For remote communication the external files will typically be transmitted together with the XMI file as SOAP attachments. Y

8.2 Linkage Package

This provides some generic modeling capability to allow linking between SPM elements and other elements that they may represent. These may be held as fully modeled MOF elements (InternalReference), opaque internal BLOBs (InternalContent) or references to external systems (ExternalReference).

This package imports the class RefObject which is the implicit supertype of all MOF classes: thus references may be attached to any element or may reference any element.



8.2.1 Reference

This abstract class represents an objectified reference that can be owned by any object.

Attributes

String name[0..1]

A reference may optionally have a name – for example to allow different references to be distinguished.

References

RefObject origin [1]

This is the object owning the reference.

The system used to resolve the link.

8.2.2 ExternalReference

Where a ModelElement is ‘mastered’ in an external system, this provides a link back to that system.

Attributes

String externalId

An identifier unique within the context of the externalSystem that can be used to locate the object.

References

System externalSystem[0..1]

The external system used to resolve the object referenced. If absent it is assumed that the externalId can be resolved using normal internet protocols.

8.2.3 ExternalSystem

This represents an instance of an external system which can be used to access the objects indicated by ExternalReferences.

Attributes

String name

A label for the external system.

String systemType

A string that represents the method/protocol used to access the external system.

String systemLocator

A locator for the external system within the context of the systemType, and which can be used to access it for resolution of external references.

8.2.4 InternalContent

This represents an artifact held internally to the SPM Facility as a Blob.

Attributes

Blob blob

The actual content.

String MIMEType[0..1]

Optional MIME type for the content, as a convenience to simplify client-side processing

String filename[0..1]

Optional file name for the content when externalized, as a convenience to simplify client-side processing. If not specified then a system-generated name will be used. The filename may include an extension which may be used to guide client-side processing in the absence of a MIMEType

Integer fileSize[0..1]

Optional size of the content when externalized, as a convenience to simplify client-side processing.

8.2.5 InternalReference

Where an element is held in the same Facility, this provides a the ability to reference that directly, without ‘corrupting’ either the SPM or the external metamodel..

References

referenceObject RefObject

The element referenced.

8.3 SoftwarePortfolio Package

This constitutes the bulk of the metamodel. It is represented here for convenience in a number of separate diagrams.

8.3.1 Element

This is the abstract root class **from which all others inherit** (apart from Kind and its derivatives – see below)

Attributes

String name[0..1]

An optional name for the element, used as the label for display purposes. Note that it is not required to be unique.

String description[0..1]

An optional description for the element, used as the label for explanatory purposes.

8.3.2 Kind

This is the abstract root class **from which all xKind elements inherit.**

Attributes

String name

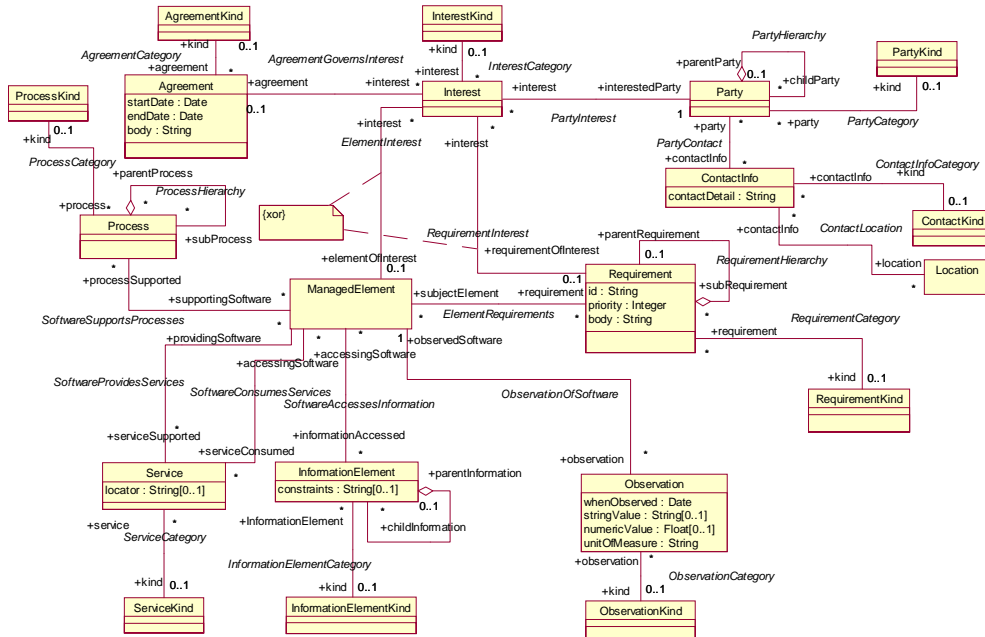
A name for the kind, used as the label for display purposes. Note that it is required to be unique amongst all other instances of the same Kind class within a Facility.

String description

A description for the kind, used as the label for explanatory purposes.

8.3.3 Context Diagram

This diagram addresses the context of software elements within the business.



8.3.4 Interest

This represents a vested interest or accountability that a party has in a managed element or requirement – for example use of software, business ownership, support responsibility. The Interest may be subject to an Agreement

Subclass of Element

References

InterestKind kind [0..1]

An optional reference to a category for the Interest, allowing similar interests to be grouped.

Party interestedParty

The Party with the interest.

Requirement requirementOfInterest[0..1]

The requirement in which the Party has an interest.

ManagedElement elementOfInterest[0..1]

The requirement in which the Party has an interest.

Constraints

Exactly one of requirementOfInterest, elementOfInterest, impactOfInterest must be set.

8.3.5 InterestKind

This represents a category of Interest

Subclass of Kind

References

Interest interest [*]

The interests which are categorized by this InterestKind.

8.3.6 Party

This represents a human entity – person or organization – of relevance to the software portfolio. It could also represent a position – allowing links by role rather than the person in the position. It will include users, vendors, support staff, business owners.

Subclass of Element

References

PartyKind kind [0..1]

An optional reference to a category for the Party.

ContactInfo contactInfo[*]

How to contact the Party. If not specified it can be assumed that the contactInfo on the parentParty (or its parent etc.) will apply.

Party parentParty[0..1]

The organization (usually) or position of which this party is a part.

Party childParty[*]

The people, organizations and positions which comprise this party.

Constraints

Exactly one of requirementOfInterest and elementOfInterest must be set.

OCL: requirementOfInterest->size() = 1 xor elementOfInterest->size() = 1

8.3.7 PartyKind

This represents a category of Party for example Person, Unit, Position, Company.

Subclass of Kind

References

Party party [*]

The parties which are categorized by this PartyKind.

8.3.8 ContactInfo

This represents a means of contacting a Party. A Party may have many different ways of being contacted. A contact may be associated with Locations.

Subclass of Element

Attributes

String contactDetail

The detail of the contact information e.g. an email address or a physical address. The interpretation will depend on the ContactInfoKind.

References

ContactInfoKind kind [0..1]

An optional reference to a category for the ContactInfo.

Party party[*]

The Parties that can be contacted using this ContactInfo

Location location[*]

The Locations corresponding to this ContactInfo.

8.3.9 ContactInfoKind

This represents a category of ContactInfo for example Office Email, Emergency Phone, Office Fax.

Subclass of Kind

References

ContactInfo contactInfo [*]

The parties which are categorized by this PartyKind.

8.3.10 Requirement

This represents a need that is applicable to the software portfolio. Requirements can be decomposed in a hierarchy.

Subclass of Element

Attributes

String id

A formal identifier for the requirement, typically multi-part to represent the hierarchical structure e.g. 2.3.6.

Integer priority

The priority of the requirement, with larger numbers indicating greater priority.

String body

The full expression of the requirement.

References

RequirementKind kind [0..1]

An optional reference to a category for the Requirement.

ManagedElement subjectElement[*]

The portfolio elements that are subject to this requirement.

Interest interest[*]

Those who have a vested interest in this requirement – either because they want to see it met or they are responsible for meeting it.

Requirement parentRequirement[0..1]

The higher level requirement of which this is a part.

Requirement childRequirement[*]

The lower level requirements into which this is decomposed.

8.3.11 RequirementKind

This represents a category of requirement for example Performance, Cost, Functionality.

Subclass of Kind

References

Requirement requirement [*]

The requirements which are categorized by this kind.

8.3.12 Observation

This represents an actual measure or observation of relevance to the software portfolio, for example the number of actual users, the Total Cost of Ownership.

Subclass of Element

Attributes

Date whenObserved [0..1]

When the observation was made.

String stringValue [0..1]

The observation value if a String.

Float integerValue [0..1]

The observation value if a number.

String unitOfMeasure [0..1]

What the value represents e.g. Dollars, 100 users.

References

ObservationKind kind [0..1]

An optional reference to a category for the Observation.

ManagedElement observedSoftware

The portfolio element that has been observed.

Constraints

Exactly one of stringValue and numericValue must be set

8.3.13 ObservationKind

This represents a category of observation for example TCO, Users.

Subclass of Kind

References

Observation observation [*]

The Observations which are categorized by this kind.

8.3.14 InformationElement

This represents information that may be accessed by portfolio elements. It may represent anything from databases to individual database columns, and logical or physical information. Information elements may be decomposed: for example a database into tables into columns.

Subclass of Element

Attributes

String constraints[0..1]

Any rules applying to the information.

References

InformationElementKind kind [0..1]

An optional reference to a category for the Information Element.

ManagedElement accessingSoftware[*]

The portfolio elements that are access this information.

InformationElement parentInformation[0..1]

The higher level/aggregated information of which this is a part.

InformationElement childInformation[*]

The lower level information into which this is decomposed.

8.3.15 InformationElementKind

This represents a category of information element for example Table, Record, File, Database, XML Schema.

Subclass of Kind

References

InformationElement informationElement [*]

The elements which are categorized by this kind.

8.3.16 Service

This represents a service that a software portfolio element provides to the business. It may be a logical service (business function) or a technical service (e.g. a web service or a more traditional API).

Subclass of Element

Attributes

String locator[0..1]

For an enactable service, how to access it on a network – including full details of their interfaces (e.g. location of a WSDL file).

References

ServiceKind kind [0..1]

An optional reference to a category for the Service.

ManagedElement providingSoftware[*]

The portfolio elements that are used to provide the service.

ManagedElement consumingSoftware[*]

The portfolio elements that access the service.

8.3.17 ServiceKind

This represents a category of service for example Web Service, Transaction Service, Business Function.

Subclass of Kind

References

Service service [*]

The services which are categorized by this kind.

8.3.18 Process

This the business processing that is supported by software elements. It may represent a high level business process, a largely manual procedure, or an automated workflow.

Subclass of Element

References

ProcessKind kind [0..1]

An optional reference to a category for the process.

ManagedElement accessingSoftware[*]

The portfolio elements that are access this information.

Process parentProcess[*]

The higher level/aggregated process or processes of which this is a part.

Process subProcess[*]

The subprocesses into which this is decomposed.

8.3.19 ProcessKind

This represents a category of process for example Line Of Business, BusinessProcess, Automated Workflow, Procedure.

Subclass of Kind

References

BusinessProcess businessProcess [*]

The elements which are categorized by this kind.

References

ImpactElementKind kind [0..1]

An optional reference to a category for the ImpactElement.

ManagedElement impactedElement[*]

The portfolio elements that are impacted.

LifecycleState state [0..1]

The current state of the impact element with respect to the lifecycle of the ImpactElementKind

ImpactElement sourceImpact[*]

Impact elements that are related as the 'source' of this one e.g. a Risk may be the source of a Change.

ImpactElement targetImpact[*]

The inverse of sourceImpact.

Constraints

The state must belong to the Lifecycle associated with the Kind.

OCL: state.lifecycle=kind.lifecycle

8.3.22 ImpactElementKind

This represents a category of ImpactElement for example Change, Risk, Project.

Subclass of Kind

References

ImpactElement impactElement [*]

The ImpactElements which are categorized by this kind.

lifecycle[0..1]

The optional lifecycle which applies to the ImpactElements that have this Kind.

8.3.23 Lifecycle

This represents a sequence of states through which elements with certain Kinds may progress.

Subclass of Element

References

LifecycleState state [*]

The states that belong to the Lifecycle.

ImpactElementKind impactKind [*]

The ImpactElementKinds making use of this Lifecycle.

ManagedElementKind elementKind [*]

The ManagedElementKinds making use of this Lifecycle.

8.3.24 LifecycleState

This represents a state which is part of a Lifecycle.

Subclass of Element

References

Lifecycle lifecycle [0..1]

The Lifecycle to which this state belongs.

ImpactElement impactElement [*]

The ImpactElements which currently have this state.

ManagedElementKind element [*]

The ManagedElements which currently have this state.

8.3.25 ManagedElementKind

This represents a category of ManagedElement for example Application, Component.

Subclass of Kind

References

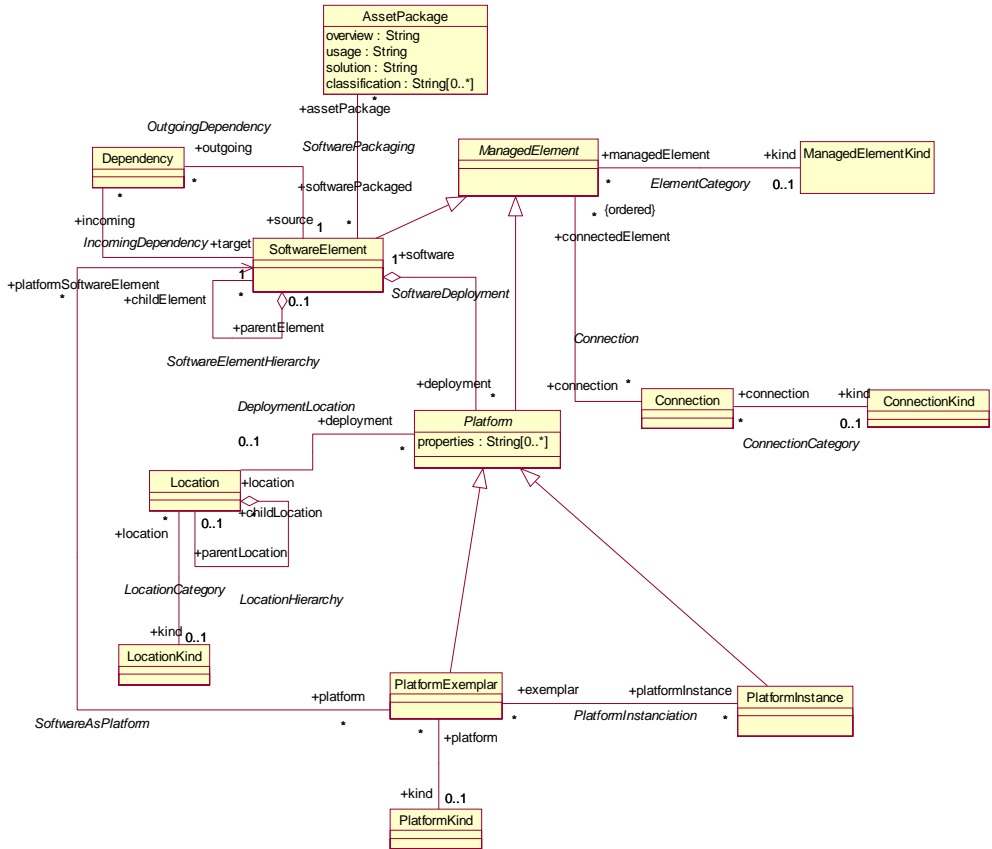
ManagedElement managedElement [*]

The ManagedElements which are categorized by this kind.

Lifecycle lifecycle[0..1]

The optional lifecycle which applies to the ManagedElements that have this Kind.

8.3.26 Managed Elements Diagram



8.3.27 ManagedElement

Abstract subclass of Element

Connection connection[*]

The connections for this Managed element.

8.3.28 ManagedElementKind

This represents a category of managed element for example

Subclass of Kind

References

ManagedElement managedElement [*]

The elements which are categorized by this kind.

8.3.29 Location

This represents a physical or logical location where software or hardware may be deployed: for example “Regional Office” (of which there are several) or “4th Floor of HQ Building”

Subclass of Element

Attributes

String constraints[0..1]

Any rules applying to the information.

References

LocationKind kind [0..1]

An optional reference to a category for the Location.

ContactInfo contactInfo [*]

The contact information corresponding to this location e.g. a fax number or full physical address.

Deployment deployment [*]

The deployments for this location.

Location parentLocation[0..1]

The containing location of this location.

Location childLocation [*]

The locations into which this is decomposed.

8.3.30 LocationKind

This represents a category of location for example Office, City, Country, Machine Room.

Subclass of Kind

References

Location location [*]

The elements which are categorized by this kind.

8.3.31 Connection

This represents a physical or logical connection between managed elements – for example a network segment or client-server communication. It can link any number of elements.

Subclass of Element

References

ConnectionKind kind [0..1]

An optional reference to a category for the Connection.

ManagedElement connectedElement [* ordered]

The contact information corresponding to this location e.g. a fax number or full physical address.

Deployment deployment [*]

The deployments for this location.

8.3.32 ConnectionKind

This represents a category of connection for example Local Area Network, Internet.

Subclass of Kind

References

Connection connection [*]

The elements which are categorized by this kind.

8.3.33 Platform

This represents a physical or logical platform on which software may be deployed. It may represent a hardware device or a software environment (e.g. combination of operating system and database).

Abstract subclass of ManagedElement**Attributes****String properties[*]**

A set of values which categorize this platform: for example “RAM > 512”. Each property is a separate value in the multivalued attribute.

References**Location location [0..1]**

The Location of the platform.

SoftwareElement software [*]

The software deployed to the platform.

8.3.34 PlatformExemplar

This represents a participant in a deployment pattern – the platform is representative of a class of platforms – for example ‘Departmental Server’ or ‘Mainframe’. The inherited properties attribute will represent criteria and may involve greater than/less than operators.

Subclass of Platform**References****PlatformKind kind [0..1]**

An optional reference to a category for the Platform.

PlatformInstance platformInstance [*]

The actual platform instances that conform to this exemplar. Note that the same instance may correspond to more than one exemplar e.g. a physical machine may be both a Print Server and a File Server.

SoftwareElement platformSoftwareElement [*]

This allows the platform itself, if it represents software, to be managed as SoftwareElements in their own right. Examples might be in-house developed security software or a commercial database package.

8.3.35 PlatformKind

This represents a category of platform, for example Database, Operating System, Laptop.

Subclass of Kind

References

PlatformExemplar platform [*]

The elements which are categorized by this kind.

8.3.36 PlatformInstance

This represents a physical platform for example a particular item of hardware or a particular (set of) files on a specific hard disk. Any categorization is given by the PlatformExemplar which this is an instance of. The inherited properties attribute will represent actual value so will use '='.

Subclass of Platform

References

PlatformExemplar exemplar [*]

The platform exemplar(s) which this particular instance represents.

Note that the same instance may correspond to more than one exemplar e.g. a physical machine may be both a Print Server and a File Server.

9. Kind Library

This section will enumerate generally useful instances of the various Kind classes. This will be developed in Finalization but the examples quoted in the body of the specification provide an illustration of what's intended.

10. Compliance Points

There are 2 compliance points.

- A) to be compliant software must import and export XMI documents compliant with the XMI DTD generated from the SPM metamodel
- B) An optional compliance point is to provide the instances specified in theKind Library.

PART III

11. Changes or extension to existing OMG Specifications

None.