IT Portfolio Management Facility Specification

This OMG document replaces the OMG Final Adopted Specification (dtc/04-11-03). Comments on the content of this document are welcomed, and should be directed to *issues@omg.org*.

You may view the pending issues for this specification from the OMG revision issues web page *http://www.omg.org/issues/*; however, at the time of this writing there were no pending issues.

OMG Available Specification
dtc/06-05-01
May 2006

Date: May 2006

IT Portfolio Management Facility (ITPMF)
Available Specification

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

Table of Contents

1 Scope

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 ITPMF could use Model merging to incorporate the ITPMF classes directly into their metamodels.

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

It is a design aim that the specification should be implementable using conventional database approaches - so the design decision was taken not to assume the presence of UML.

Overall there are a number of options for extensibility:

- Extend the supplied model directly - by defining subclasses.
- Link with other models - using the Linkage package or just defining new MOF Associations.
- Use the generic Kind and Property mechanisms within the specification itself: this means that this ITPMF specification can be used to create a stand-alone facility that is still extensible.

2 Conformance

There are 2 compliance points.

1. To be compliant software must import and export XMI documents compliant with the XMI DTD generated from the ITPMF metamodel
2. An optional compliance point is to provide the instances specified in the Kind Library.

3 Normative References

None.

4 Terms and Conventions

For the purposes of this specification, the following terms and conventions apply.

4.1 Glossary

The following conventions and terms are used in this document.

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. Available OMG technology: formal/04-03-26
EDOC	UML Profile for Enterprise Distributed Object Computing - key specification bridging business and component architectures. Available OMG Technology: formal/04-02-05
ITPMF	IT Portfolio Management Facility. This specification.
MOF	Meta Object Facility. Available OMG technology: formal/06-01-01
ODP	Open Distributed Processing. Architectural approach advocated by International Standards Organization (ISO).
RAS	Reusable Asset Specification covering the classification and cataloguing of (usually software) assets. Available OMG technology: formal/05-11-02.
SPEM	Software Process Engineering Metamodel. Available OMG technology: formal/05-01-06. 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/05-07-04. 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/05-09-01. 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.

5 Symbols

None.

6 Additional Information

6.1 Acknowledgements

The following individuals are acknowledged for their contribution to this specification:

6.1.1 Submitters

- Adaptive

The following colleague at Adaptive has helped review and improve the metamodel on which this specification 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 specification over a number of years.

6.1.2 Supporters

- Deere & Company
- Interoperability Clearing House

7 The Metamodel

The metamodel is illustrated through a set of UML diagrams. Two supplementary and reusable packages are first presented as generic MOF extensions.

7.1 ExtendedPrimitiveTypes Package

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

7.1.1 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.

7.1.2 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.

7.1.3 Blob

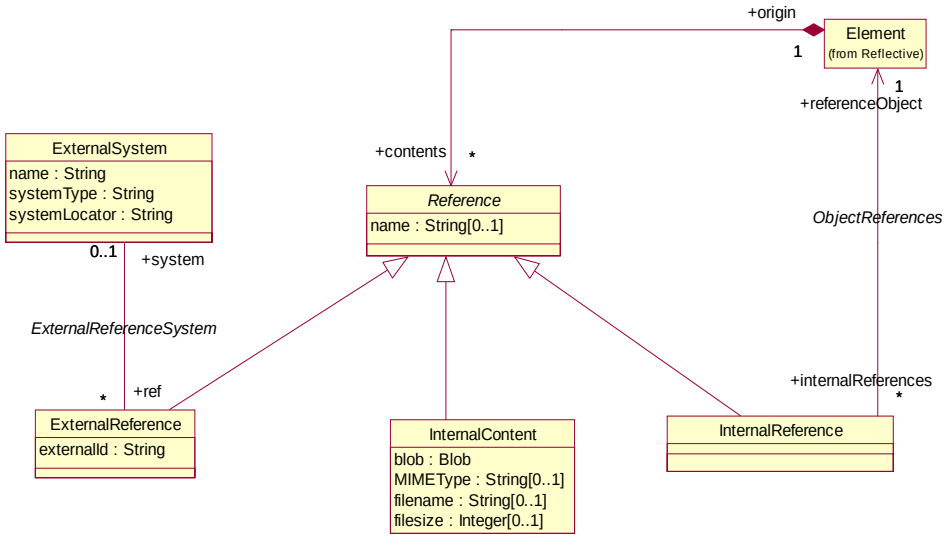
This represents an opaque Binary Large Object as supported by most database systems. For XMI it is mapped to an href referencing an external file that will typically be local and the name of which will typically be system-allocated. For remote communication the external files will typically be transmitted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

7.2 Linkage Package

This provides some generic modeling capability to allow linking between ITPMF 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).

Issue 7921 - Use Element instead of RefObject

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



7.2.1 Reference

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

7.2.1.1 Attributes

String name[0..1]

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

7.2.1.2 References

Issue 7921 - Use Element instead of RefObject

Element origin [1]

This is the object owning the reference.

Issue 7922 - Deleted extraneous line

7.2.2 ExternalReference

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

7.2.2.1 Attributes

String externalId

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

7.2.2.2 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.

7.2.3 ExternalSystem

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

7.2.3.1 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.

7.2.4 InternalContent

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

7.2.4.1 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 that 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.

7.2.5 InternalReference

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

7.2.5.1 References

Issue 7921 - Use Element instead of RefObject

referenceObject Element

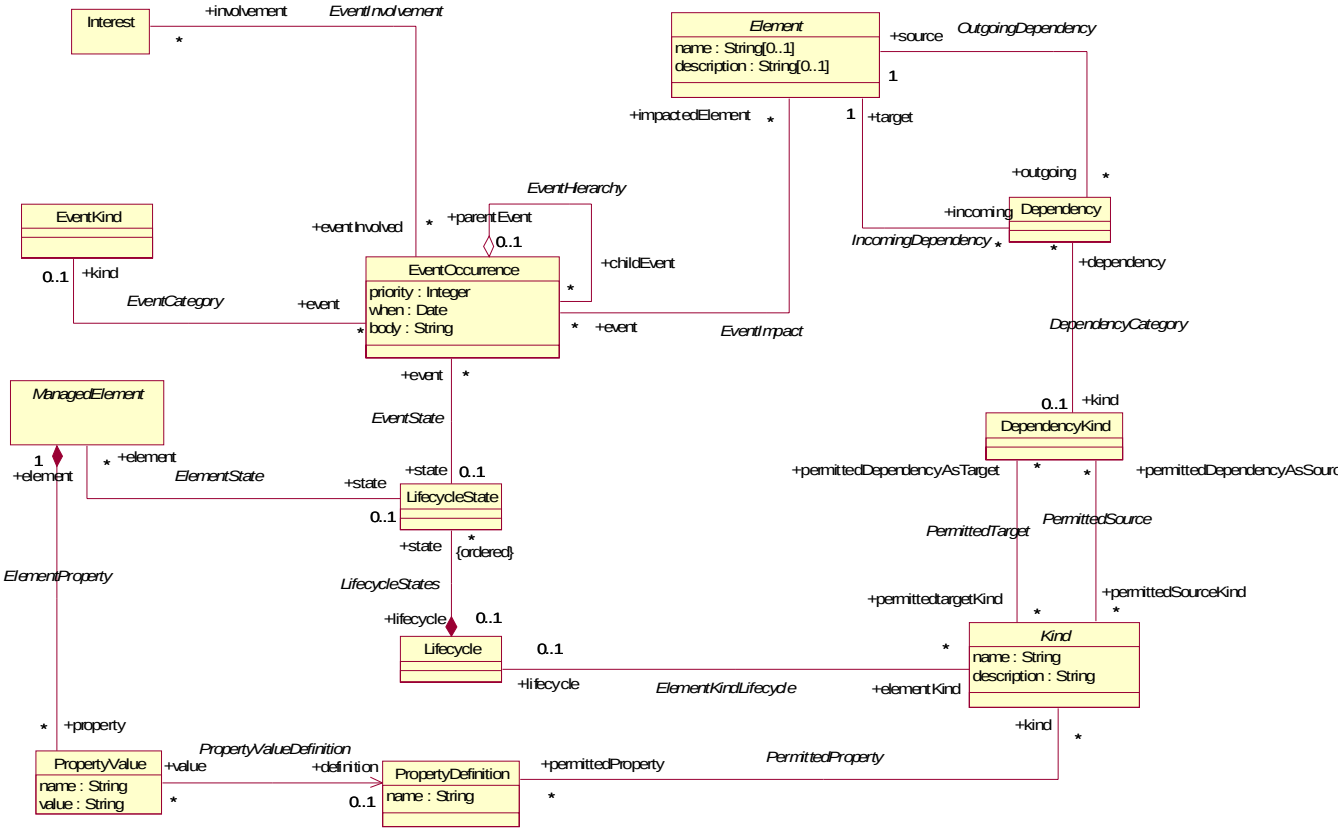
The element referenced.

7.3 ITPortfolio Package

This constitutes the bulk of the metamodel. It is represented here for convenience in a number of separate diagrams. The diagrams are shown first, then the classes in alphabetical order. Note that xKind objects (any class ending in 'Kind') are covered in section 8.4 except for the class Kind itself. In terms of metamodel, each Kind class has a standard association with the element it classifies.

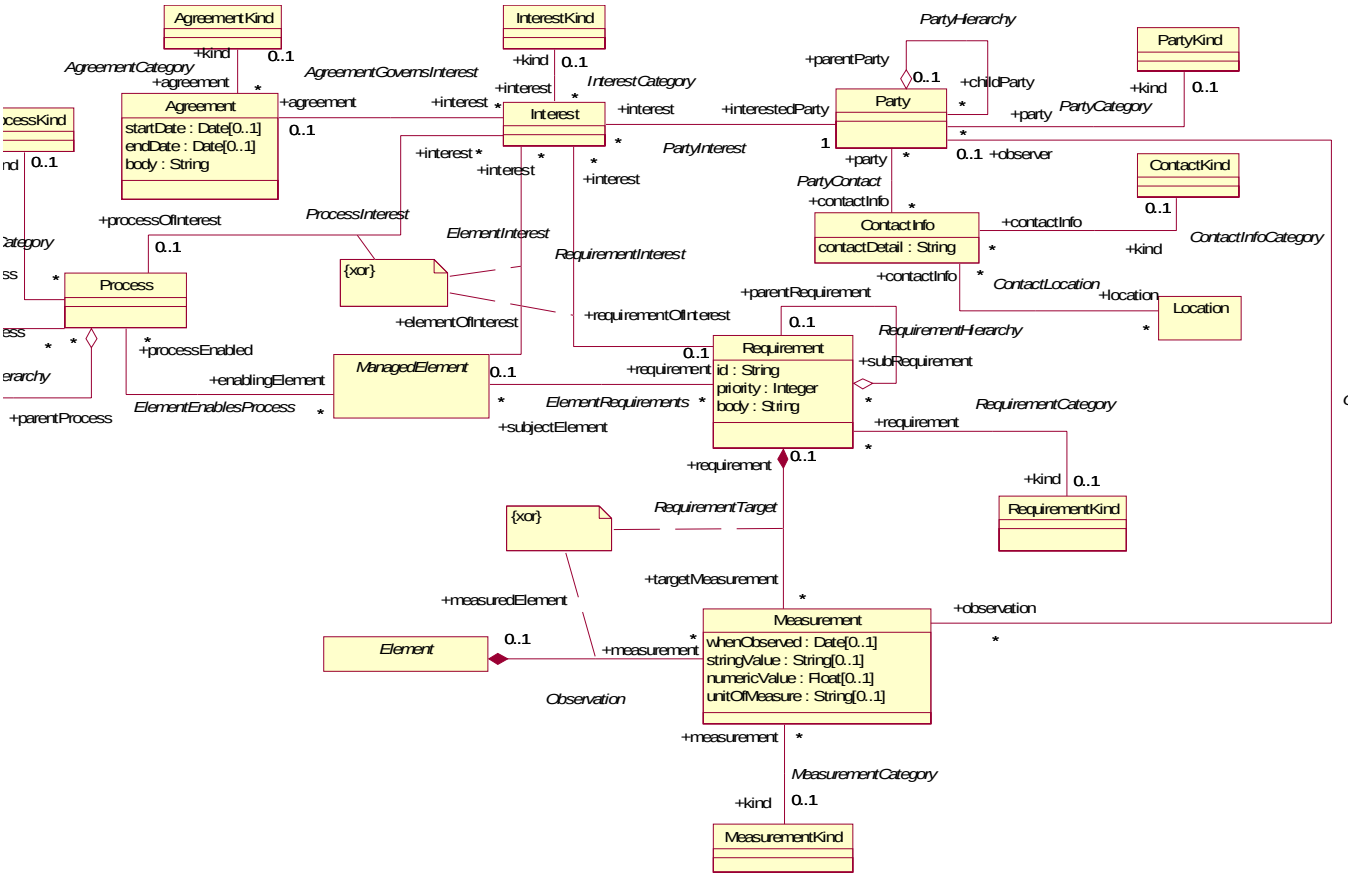
7.3.1 Diagrams

7.3.1.1 Control Diagram



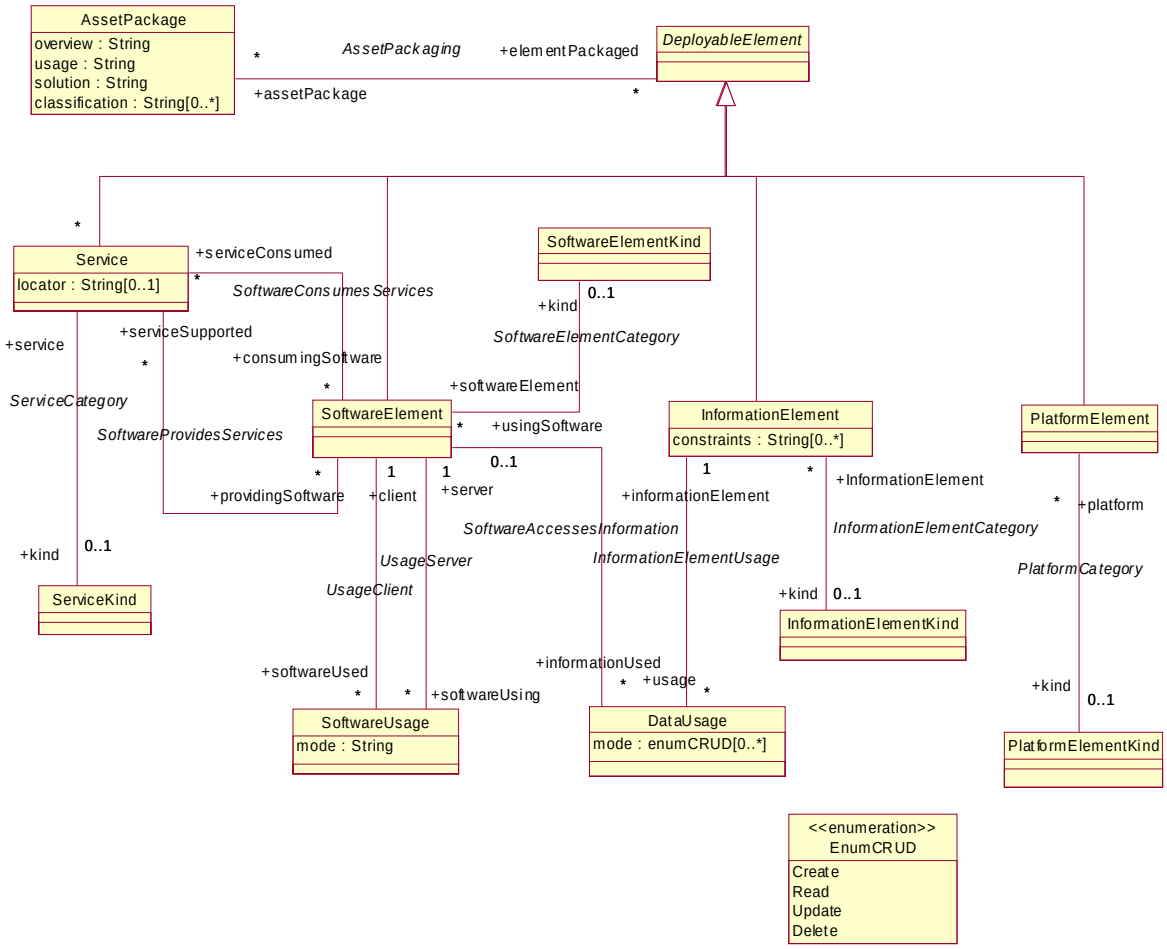
7.3.1.2 Context Diagram

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



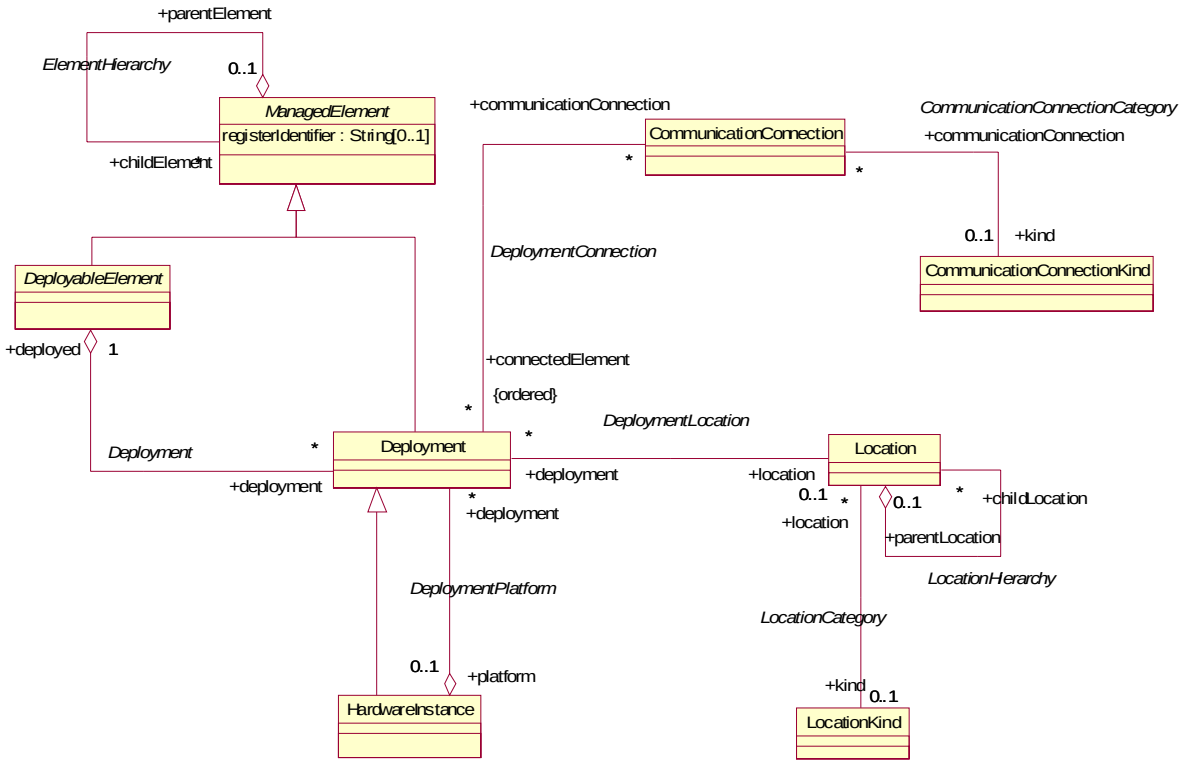
7.3.1.3 Deployable Elements Diagram

This diagram represents the elements that may be deployed.



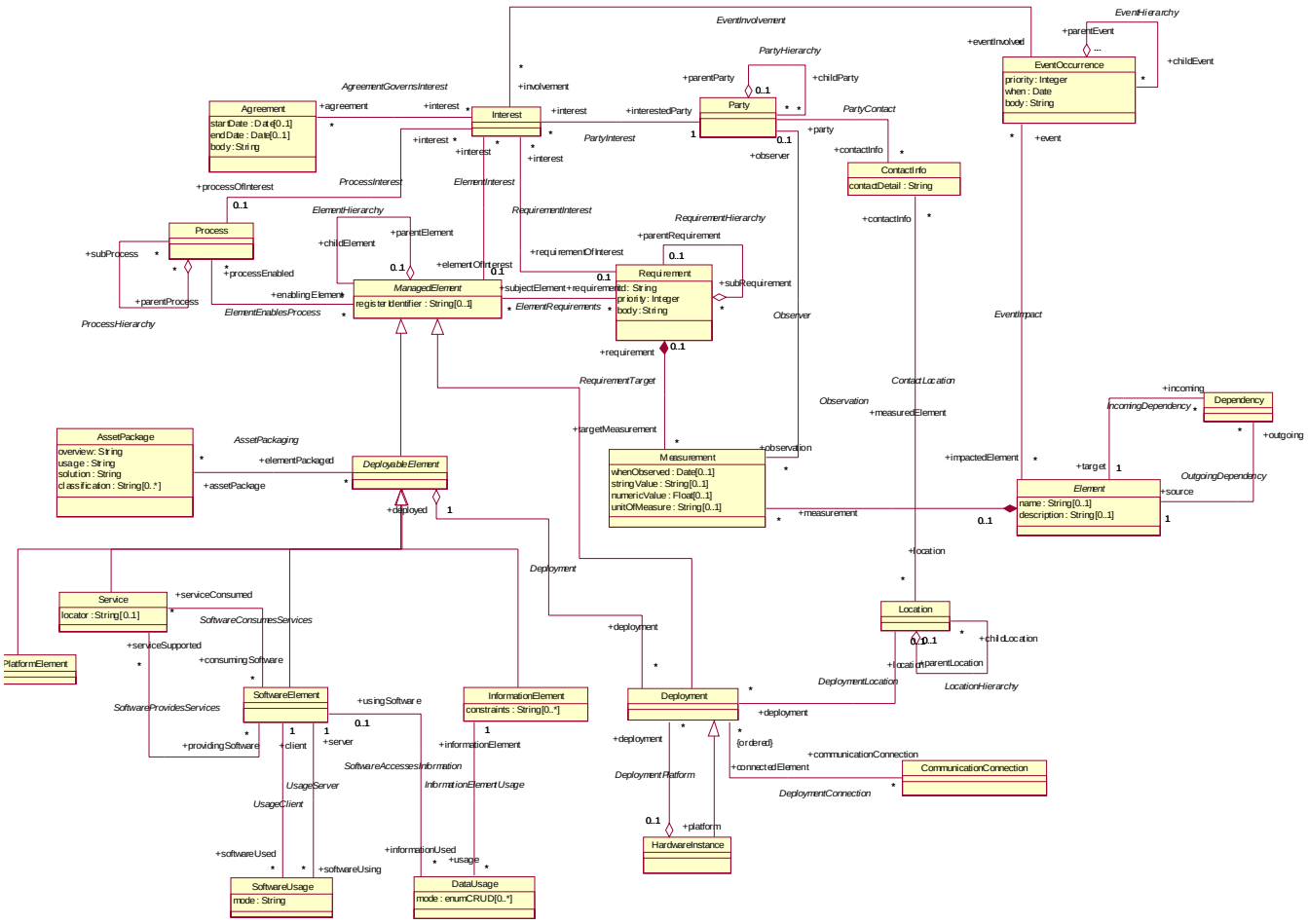
7.3.1.4 Deployment Diagram

This diagram covers the deployment of elements onto hardware.



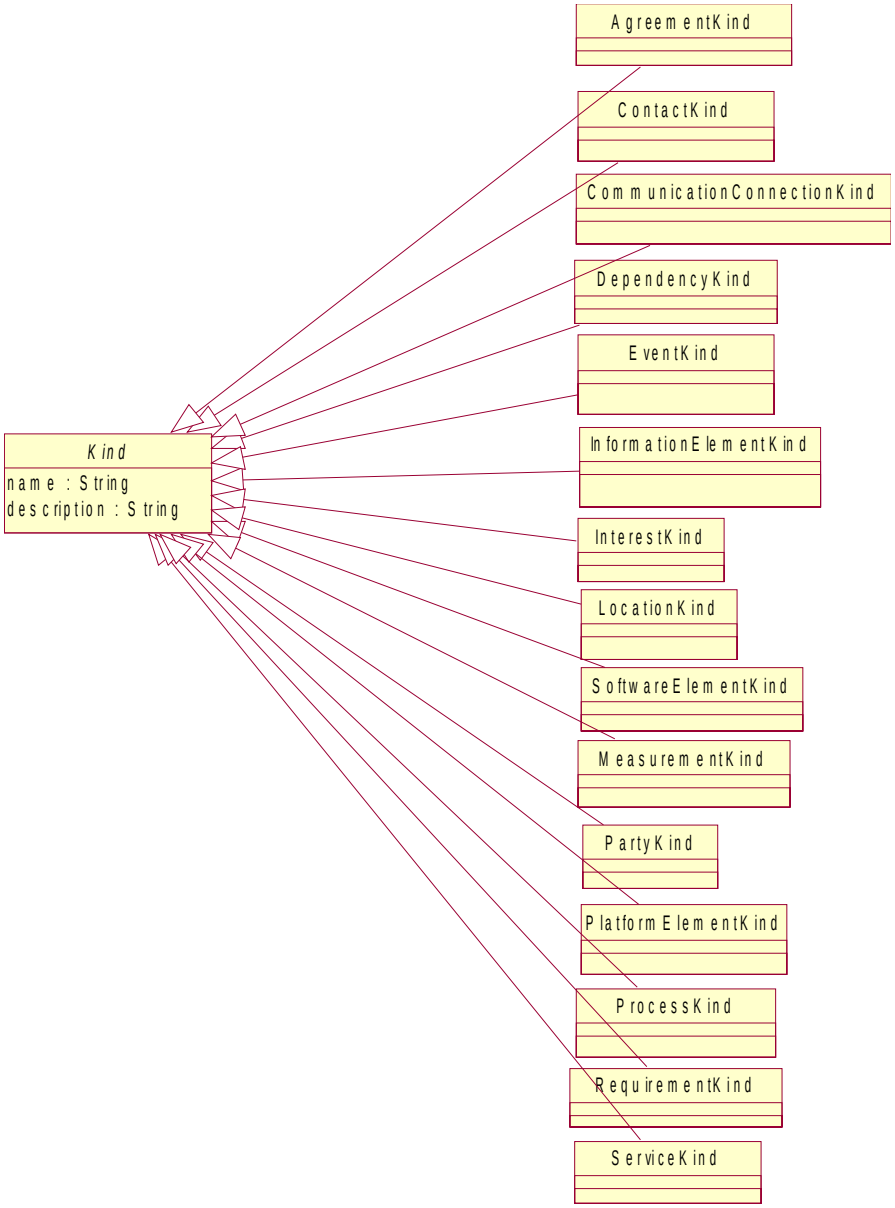
7.3.1.5 The ‘Essential’ Diagram

This diagram combines all the key classes (excluding less important ones such as xKind).



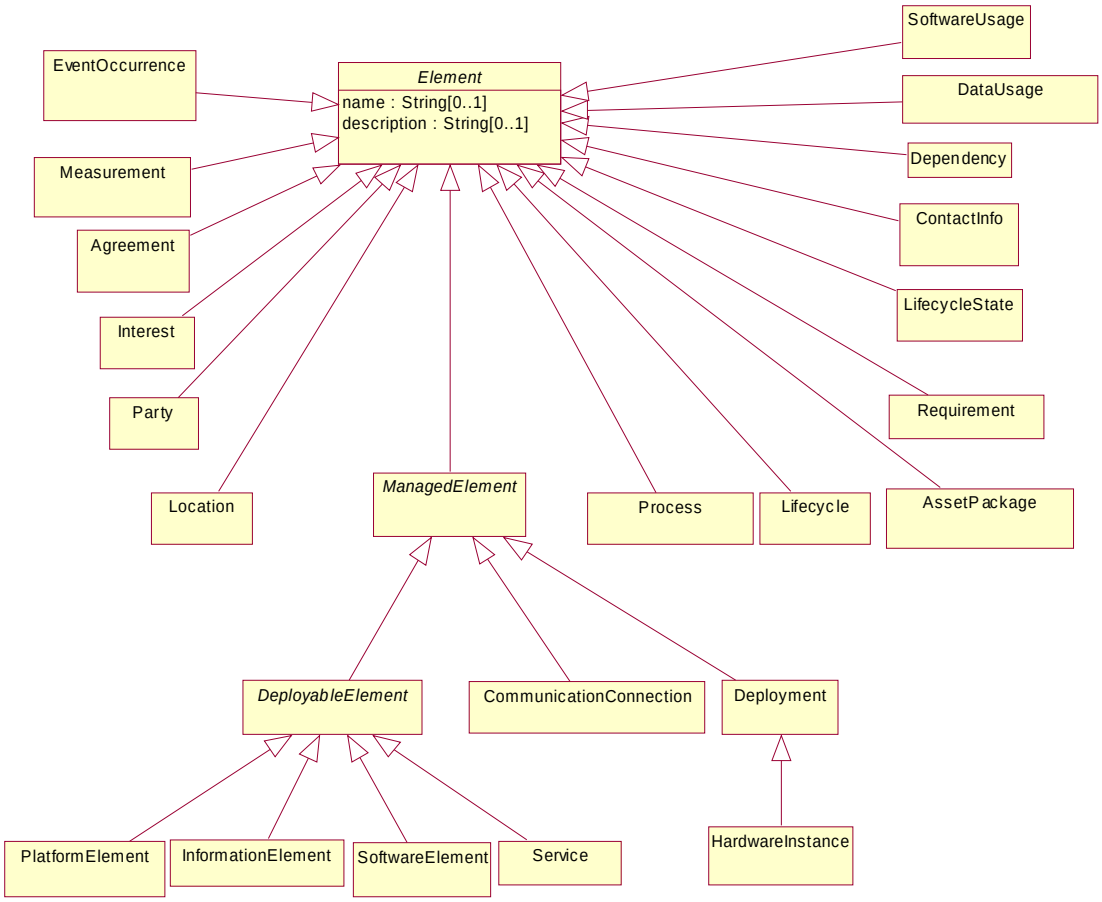
7.3.1.6 Kinds Diagram

This diagram shows the different subclasses of Kind.



7.3.1.7 Inheritance Diagram

This shows the inheritance structure, excluding the Kind classes.



7.3.2 Agreement

This represents an agreement entered into by one or more Parties with respect to some sort of Interest in a managed Element.

Subclass of Element

Attributes

Date startDate[0..1]

The effective start date of the Agreement. If no value is present it can be assumed to have always been in effect.

Date endDate[0..1]

The effective end date of the Agreement. If no value is present, it can be assumed either to be indefinite or be terminable by some action or event described in the Agreement itself.

Issue 7923 - Add missing type String

String body

The text of the agreement. This might be a summary with the full text held in a linked document (see Linkage package).

References

Interest interest [0..*]

The Interests that this Agreement governs. For example if the Agreement represents a Software License for a software package P, the Interests will have InterestKind of ‘User’ and link P with the different Parties licensed to use P.

A further Support Agreement might link the same ‘User’ Interests with a Support Agreement that also links to another Interest of InterestKind ‘Technical Support’ and a Party representing a HelpDesk team.

7.3.3 AssetPackage

This represents the package of information that might accompany a ManagedElement ready for deployment. It is intended to match the structure for an Asset in the Reusable Asset Specification (RAS), and to be extensible to a full RAS implementation. The package will typically contain overview, classification, and usage information. The actual deployable artifacts will typically be linked using the Linkage package.

Subclass of Element

Attributes

String Overview

Textual overview of the asset and its purpose.

String Usage

Textual overview of how to use the asset in different environrnents including development.

String Solution

Information about what gets deployed.

String Classifications[0..*]

Classification keywords in order to facilitate searching for reuse.

References

Interest elementPackaged [0..*]

The DeployableElements packaged.

7.3.4 CommunicationConnection

This represents a connection between managed elements through which they can communicate– for example a network segment or a telephone line. It can link any number of elements. Network elements such as bridges and routers will typically appear as the intersection between two CommuncationConnections.

As a subclass of ManagedEement this ‘network’can be managed in its own right.

Subclass of ManagedElement

References

CommunicationConnectionKind kind [0..1]

An optional reference to a category for the **Connection**.

Deployment connectedElement [* ordered]

The deployed elements connected via this connection.

7.3.5 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.

7.3.6 DataUsage

This represents the usage of information by software.

Subclass of Element

Attributes

EnumCRUD mode[0..*] unique

How the software is accessing the information: this will be one or more or Create, Read, Update, Delete.

References

informationElement[1]

The informationElement being used/accessed.

usingSoftware[0..1]

The software doing the accessing.

7.3.7 Dependency

Represents a design-time or run-time dependency between elements that is not tracked by the specifically modeled associations. So it can link dependencies between Elements (e.g., Application X depends on an Application Sender) and between Deployments (the deployment of Application X on Server S123 is dependent on the Tomcat Application Server deployed n machine S124) needs a dependency is directional and links one source to one target Element. It may be types by a DependencyKind. This is designed for fairly dynamic extensibility. To make the solution manageable it is possible to limit which Kinds of Element each DependencyKind can link.

Subclass of Element

References

Element source

The source, or client, of the dependency.

Element target

The target, or supplier, of the dependency.

DependencyKind kind[0..1]

The nature of the dependency.

7.3.8 DeployableElement

This is an abstract class that represents elements that are deployable onto hardware at a specific location.

Abstract subclass of Element

References

Deployment deployment [0..*]

Where the element is deployed.

AssetPackage assetPackage [0..1]

The package of asset information to help use and deploy it, if present.

7.3.9 Deployment

This represents an instance of a DeployableElement placed on a HardwareInstance. Importantly, it is itself a ManagedElement. And can participate in Dependencies (e.g., one Deployment is emergency backup for another). Deployments can also be grouped hierarchically. This allows one to compose a System as a combination of deployed hardware and software elements. And, further still, allows the creation of 'environments' such as Production and Development.

Subclass of ManagedElement

References

DeployableElement deployed

The element deployed in this case. Note that this is really a type-instance relationship and allows the same software element (for example) to be instantiated/copied and deployed many times.

HardwareInstance platform [0..1]

The specific hardware where the deployment occurs. Note that HardwareInstance, as a ManagedElement, can be recursively decomposed. So it would be possible to track the disk or even folder within a machine.

location Location[0..1]

Where the deployment is located. Note that this will normally be used for the hardware itself since HardwareInstance inherits from Deployment.

CommunicationConnection communicationConnection[0..*]

The networks or other communications that this deployment participates in. Amongst other things the communicationConnections should be used to realize dependencies (e.g., one deployment is dependent on another) there usually needs to be a mutual CommunicationConnection.

7.3.10 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.

7.3.11 EventOccurrence

This represents something that has happened or may happen which may have an impact on one or more elements. It could range from a Project through to a particular Change or Issue or Risk, or just an installation/upgrade. Such events have their own Kind, which in turn can have a Lifecycle that will determine how the events should be processed.

Note that Events can be nested inside other 'larger' events and have Dependencies between them.

Subclass of Element

Attributes

Integer priority

The priority of dealing with the Event, with larger numbers indicating greater priority.

Date when

The date associated with the event – the interpretation will depend on the nature of the ImpactElementKind.

String body

The full description of the event.

References

EventKind 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 event with respect to the lifecycle of the EventKind.

EventOccurrence parentEvent[0..1]

A larger event in which this is nested. For example, a major upgrade may require a number of sub-events, such as: adding more RAM, upgrading Windows, upgrading Office, downloading and applying latest patches, etc.

EventOccurrence childEvent[*]

The inverse of parentEvent.

7.3.12 HardwareInstance

This represents a physical platform, for example, a particular item of hardware or a particular (set of) directory on a specific hard disk.

HardwareInstance itself inherits from Deployment – indicating that each item of hardware is the deployment of the PlatformElement representing the Model of server. Even if at a specific time it is ‘deployed’ onto a shelf in a warehouse for storage.

Note that it inherits PropertyValues from ManagedElement.

Subclass of Deployment

References

Deployment deployment [*]

The Deployments deployed in this platform.

7.3.13 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..*]

Any rules applying to the information.

References

InformationElementKind kind [0..1]

An optional reference to a category for the **Information Element**.

DataUsage accessingSoftware[*]

The software access to 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.

7.3.14 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.

Process processOfInterest[0..1]

The process in which the Party has an interest.

Constraints

Exactly one of requirementOfInterest, elementOfInterest, eventOfInterest, processOfInterest must be set.

7.3.15 Kind

This is the abstract root class **from which all xKind elements inherit.** It is akin to the class Stereotype in UML. It is possible to add some type checking – for the DependencyKinds that can apply either as source or target.

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.

References

DependencyKind permittedDependencyAsTarget[0..*]

Declares that instances of this Kind are permitted to be the target of instances of the referenced DependencyKind.

DependencyKind permittedDependencyAsSource[0..*]

Declares that instances of this Kind are permitted to be the source of instances of the referenced DependencyKind.

PropertyKind permittedProperty[0..*]

Declares that instances of this Kind are permitted to have values for the referenced PropertyKind.

7.3.16 Lifecycle

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

Subclass of Element

References

LifecycleState state [*] {ordered}

The states that belong to the Lifecycle.

EventKind eventKind [*]

The EventKinds making use of this Lifecycle.

Kind elementKind [*]

The Kinds making use of this Lifecycle.

7.3.17 LifecycleState

This represents a state which is part of a Lifecycle. For example ‘under development,’ ‘testing,’ ‘live,’ ‘complete’ (for an event).

Subclass of Element

References

Lifecycle lifecycle [0..1]

The Lifecycle to which this state belongs.

EventOccurrence eventOccurrence [*]

The Events that currently have this state.

ManagedElementKind element [*]

The ManagedElements that currently have this state.

7.3.18 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 “4^{th} Floor of HQ Building.”

Subclass of Element

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.

7.3.19 ManagedElement

This abstract class represents those elements which may be managed as part of the ITPMF. It allows the grouping of elements into hierarchies, which allows management at a higher level of granularity (e.g., System).

Abstract subclass of Element

Attributes

String registerIdentifier [0..1]

The identifier used for the element in a corporate asset register or similar.

References

ManagedElement parentElement[0..1]

The element containing or composing this element.

ManagedElement childElement[0..*]

The elements contained or composed by this element.

Process processEnabled[0..*]

The organization processes enabled by this managed element – in order to track business impact and achieve traceability.

PropertyValue property[0..*]

A dynamically extensible set of values relevant to the Kind of the element. This can represent factors such as capacity (memory, disk) for hardware, protocols, etc.

7.3.20 Measurement

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

An instance of Measurement may either be linked to an Element as an actual measurement, or linked to a Requirement as a target.

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

MeasurementKind kind [0..1]

An optional reference to a category for the **Measurement**.

Party observer[0..1]

Who made the observation/measurement.

Element measuredElement[0..1]

The element that has been observed.

Requirement requirement[0..1]

The requirement for which this measure is a target.

Constraints

Exactly one of stringValue and numericValue must be set.

7.3.21 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 that comprise this party.

Measurement observation[0..*]

The measurements observed by the party.

7.3.22 PlatformElement

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) or a class of platforms – for example ‘Departmental Server’ or ‘Mainframe’. It may be deployed as a HardwareInstance.

Subclass of ManagedElement

7.3.23 Process

This is the business processing that is enabled by managed 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 enablingElement[*]

The portfolio elements that enable the process.

Interest interest[0..*]

The stakeholders for the process across the organization(s).

Process parentProcess[*]

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

Process subProcess[*]

The subprocesses into which this is decomposed.

7.3.24 PropertyValue

This represents a value assocated with ManagedElement. This is provided as a generic extensible capability, though can also be controlled by element Kind by linking the Value to a PropertyValueKind.

Attributes

String name

The name of the property.

String value

The value of this property for the managed element

References

ManagedElement element

The element that the property is for.

PropertyDefinition definition[0..1]

The definition for the property.

Constraints

If there is an attached definition, its name must be the same as the PropertyValue name.

7.3.25 PropertyDefinition

Attributes

String name

The name of the property.

References

Kind kind[0..*]

The element Kinds that the property is permitted for.

7.3.26 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.

Measurement targetMeasurement[0..*]

Target measurements to quantify the requirement.

7.3.27 Service

This represents a service that a software 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 ManagedElement

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**.

SoftwareElement providingSoftware[*]

The software elements that are used to provide the service.

SoftwareElement consumingSoftware[*]

The software elements that access the service.

7.3.28 SoftwareElement

Represents software providing some aspect of business related functionality (middleware, etc. may be represented as PlatformElement). Can represent different levels of granularity from components/libraries up to whole ERP suites. The inherited association fromManagedElement allows the use of aggregation hierarchies.

Subclass of ManagedElement

References

SoftwareElementKind kind [0..1]

An optional reference to a category for the **Software**.

SoftwareUsage softwareUsed[0..*]

Other software elements that are used by this one.

SoftwareUsage softwareusing[0..*]

Other software elements that use this one.

7.3.29 SoftwareUsage

Represents the use of one software element by another. This is at the design level and does not represent a specific runtime connection – though it will usually prompt the need for one.

Subclass of Element

Attributes

String mode

The nature of the usage.

References

SoftwareElement client

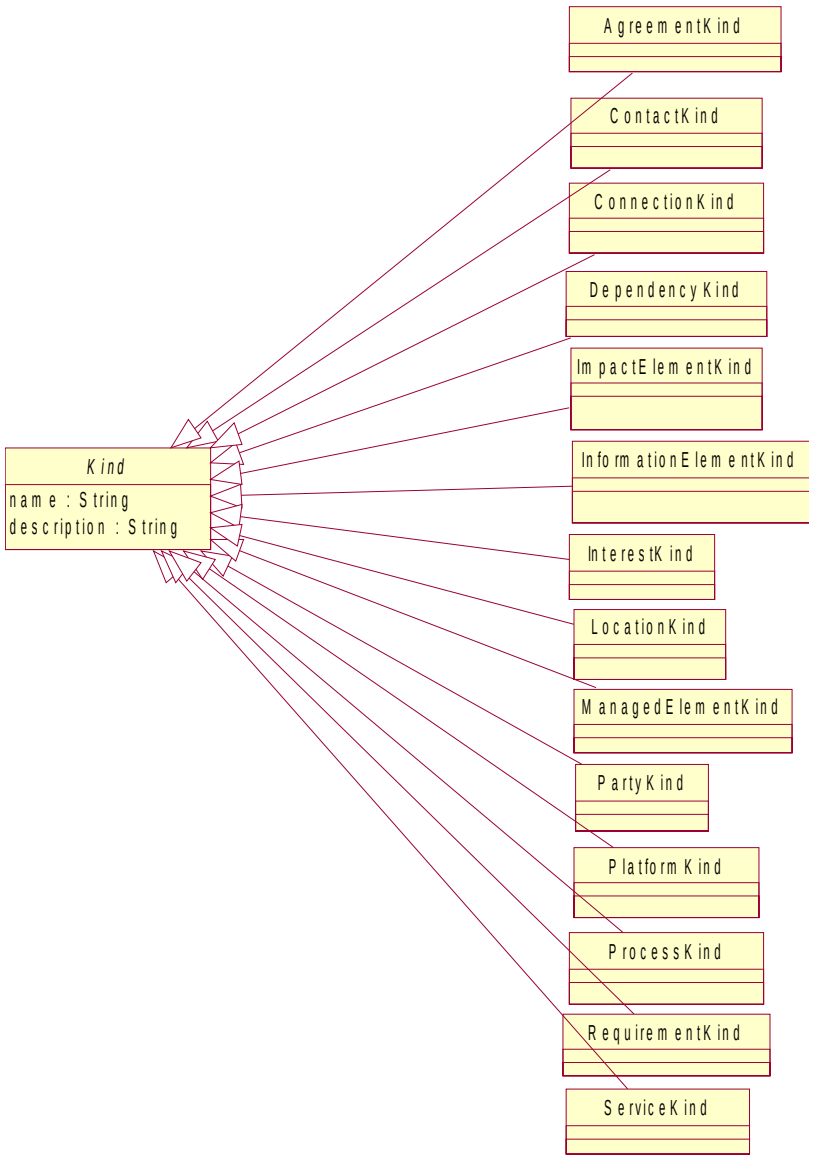
The client of this usage requesting capability.

SoftwareElement server

The server of this usage providing a capability to the client.

8 Kind Library

The metamodel is illustrated through a set of UML diagrams. Two supplementary and reusable packages are first presented as generic MOF extensions.



8.1 AgreementKind

Service Level Agreement	Maintain an ongoing quantifiable level of service to a set of users according to a set of agreed parameters/objectives.
Software License Agreement	Provision of software for a specified number of users.
Support Agreement	Provision of incident resolution.
Services Agreement	Carry out specified service activities.
Maintenance Agreement	Maintain a capability by providing updates (patches, new releases) as needed.

8.2 ContactKind

Sales Contact	Maintain an ongoing quantifiable level of service to a set of users according to a set of agreed parameters/objectives.
Support Contact	Provision of software for a specified number of users.
Emergency Contact	Provision of incident resolution.
Management Contact	Carry out specified service actitivities.
HQ Contact	Maintain a capability by providing updates (patches, new releases) as needed.
Local Contact	
Mobile Contact	

8.3 CommunicationConnectionKind

Connections between Platforms will tend to be more physical in nature (e.g., LAN Connection) whereas those between SoftwareElements will tend to be more software-oriented (e.g., Middleware Connection).

LAN Connection	Connection over Local Area Network (with no intervening firewalls).
Public Internet Connection	
VPN Connection	
Private WAN Connection	
Wireless Connection	
Middleware Connection	
Message Queue Connection	
Web Services Connection	
SAN Connection	Storage Area Network (typically fiber-optic).

8.4 DependencyKind

Platform Dependency	An element depends on a platform to run (not exclusive – there may be alternatives).

8.5 EventKind

This represents

Change	
Risk	
Installation	
Decommissioning	
Commissioning	

8.6 InformationElementKind

Relational Table	Maintain an ongoing quantifiable level of service to a set of users according to a set of agreed parameters/objectives.
Record	
File	
XML Schema	Provision of incident resolution.
Database	Carry out specified service actitivities.
Maintenance Agreement	Maintain a capability by providing updates (patches, new releases) as needed.

8.7 InterestKind

Technical Owner	
Technical Assigneee	
Business Owner	
Steward	
Customer	
Sponsor	
Owner	
User	
Vendor	
Developer	
Supporter	
Reviewer	

8.8 LocationKind

Country	
State	
City	
District	
Campus	
Building	
Floor	
Room	
Position	

8.9 ObservationKind

Total Cost of Ownership	
Number of Users	
Response Time	
Acquisition Cost	

8.10 PartyKind

Person	A human being.
Company	A legally-constituted company.
Organization Unit	Part of a company such as a Department or Division.
Position	The role to which one or more people are assigned.
Consortium	A grouping of companies.
Project	A temporary grouping of people constituted for a specific duration and a specific purpose.

8.11 PlatformKind

Server	
Desktop	
Laptop	
Database	
Networking Hardware	
Firewall	
Operating System	
Application Server	
Web Server	

8.12 ProcessKind

Business Process	
Procedure	
Development Method	
Outsourced Process	
Line of Business	
Governance Process	
Automated Workflow	

8.13 RequirementKind

Functional Requirement	Provision of functionality.
System Use Case	Requirement specified as one or more system use cases.
Business Use Case	Requirement specified as one or more business use cases.
Non Functional Requirement	
Performance Requirement	
Cost Requirement	

8.14 ServiceKind

Business Function	
Web Service	
Service Product	Service provided on a commercial basis (internally or externally).
API	