

# **Software Portfolio Management Facility (SPMF) Specification bei/2004-04-04**

## ***Final Submission***

---

***Submitter:***

**Adaptive**

***Supported By:***

**Interoperability Clearing House**

April 5, 2004

# Table of Contents

<b>1. Copyright Waiver</b>	<b>1-2</b>
<b>2. Contacts for the submission</b>	<b>2-4</b>
2.1 Primary Contact .....	2-4
2.2 Acknowledgments .....	2-4
<b>3. Structure of This Submission</b>	<b>3-4</b>
<b>4. Overall Design Rationale</b>	<b>4-5</b>
<b>5. Proof of Concept</b>	<b>6</b>
<b>6. Resolution of RFP Requirements</b>	<b>7</b>
6.1 Mandatory Requirements .....	7
6.2 Optional Requirements.....	9
<b>7. Glossary and References</b>	<b>10</b>
<b>8. Proposed Metamodel</b>	<b>11</b>
8.1 ExtendedPrimitiveTypes Package.....	11
8.2 Linkage Package.....	11
8.3 SoftwarePortfolio Package.....	14
<b>9. Kind Library</b>	<b>38</b>
<b>10. Compliance Points</b>	<b>44</b>
<b>11. Changes or extension to existing OMG Specifications</b>	<b>44</b>

# PART I

## 1. Copyright Waiver

Copyright © 2001-2004 Adaptive Ltd.

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.

## Section 1. Copyright Waiver

---

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](mailto: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 earlier revisions of this submission, 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 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 SPMF specification can be used to create a stand-alone facility which is still extensible.

## 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*  
**The impact of project tasks on portfolio elements is represented through EventOccurrence objects (which may represent future intentions); projects themselves can be represented through higher level tasks or, organizationally, as a Party.** 6.5.2.7  
*Technology types and vendor products*  
**This is addressed through the class PlatformElement.**

### 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 to 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 EventOccurrence class: see response to requirement 6.5.2.6.**

# PART II

## 7. Glossary and References

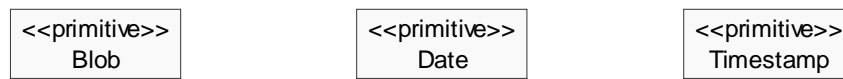
<b>BPDM</b>	<b>Business Process Definition Metamodel - in-process OMG specification: RFP is bei/03-01-06.</b>
<b>CCA</b>	Collaborating Component Architecture – part of EDOC (qv) covering components at any level of granularity and their composition.
<b>CWM</b>	Common Warehouse Metamodel.- despite its name covers all all aspects of information resources. Available OMG technology: formal/01-10-01
<b>EAI</b>	UML Profile and Interchange Models for Enterprise Application Integration. Adopted OMG technology: ptc/02-02-02
<b>EDOC</b>	UML Profile for Enterprise Distributed Object Computing – key specification bridging business and component architectures. Adopted OMG Technology: ptc/02-02-05
<b>MOF</b>	Meta Object Facility. Available OMG technology: formal/02-04-03
<b>ODP</b>	Open Distributed Processing. Architectural approach advocated by International Standards Organization (ISO).
<b>RAS</b>	Reusable Asset Specification covering the classification and cataloguing of (usually software) assets. <b>Undergoing adoption through the Request For Comments process: ad/03-10-10</b> <a href="http://www.rational.com/rda/ras/preview/index.htm">http://www.rational.com/rda/ras/preview/index.htm</a>
<b>SPMF</b>	Software Portfolio Management Facility. This specification.
<b>SPEM</b>	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.
<b>UML</b>	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.
<b>XMI</b>	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.
<b>XML</b>	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. Two supplementary and reusable packages are first presented as generic MOF extensions.*

### 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

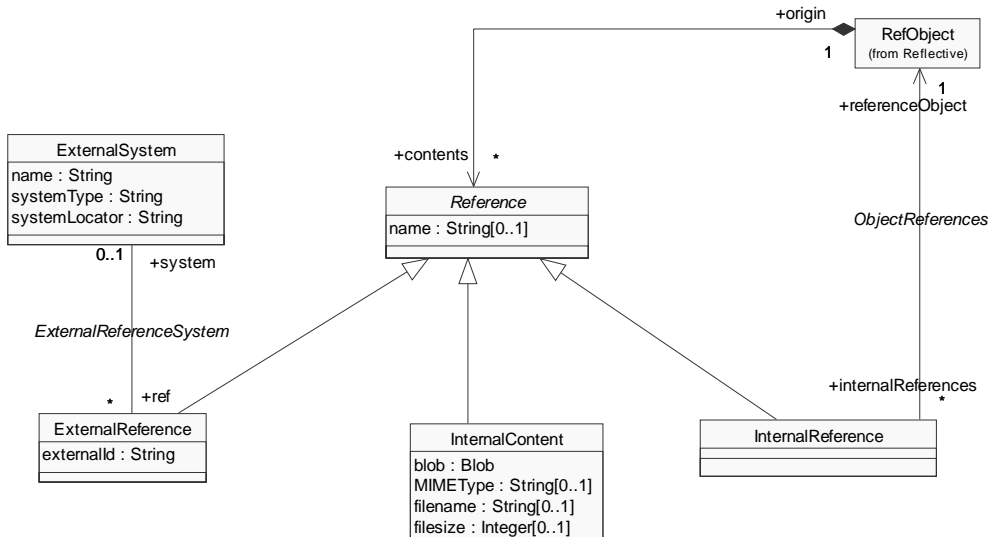
#### 8.1.3 Blob

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

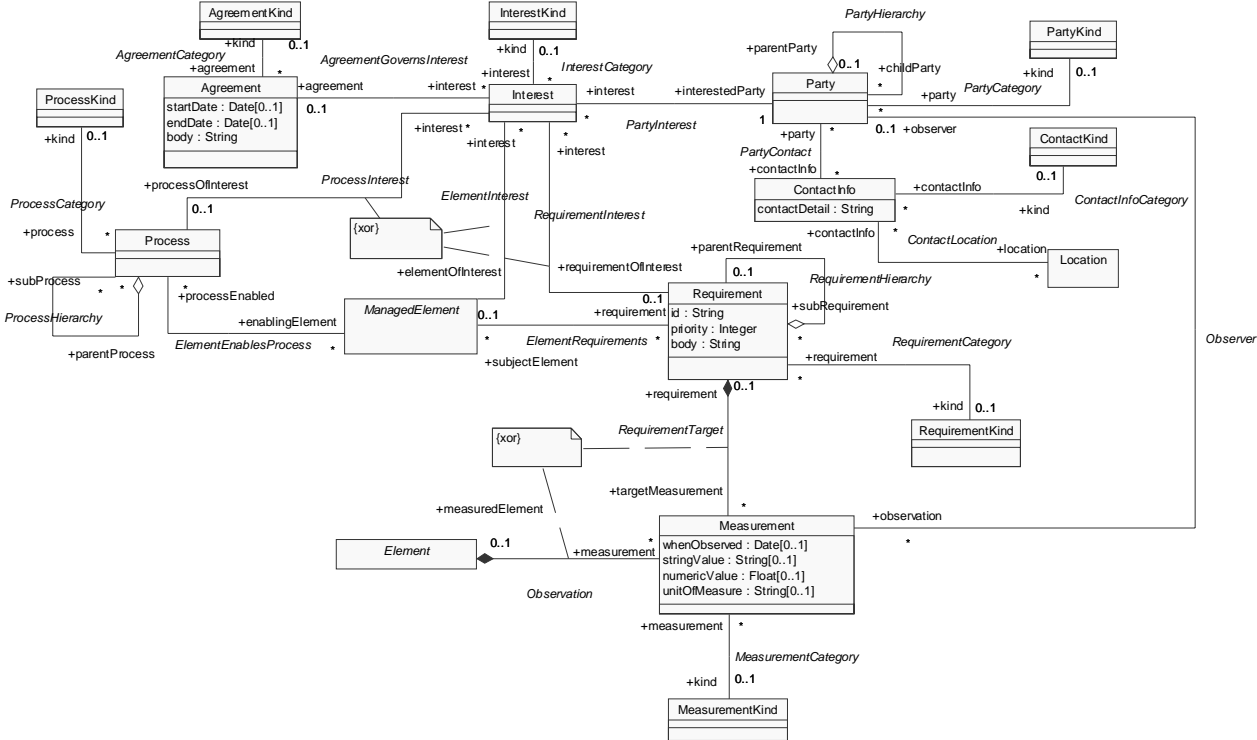
This constitutes the bulk of the metamodel. It is represented here for convenience in a number of separate diagrams. The diagrams are shown first, then the classes in alphabetical order. Note that xKind objects (any class ending in 'Kind') are covered in section 8.4 except for the class Kind itself.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

### 8.3.1 Diagrams



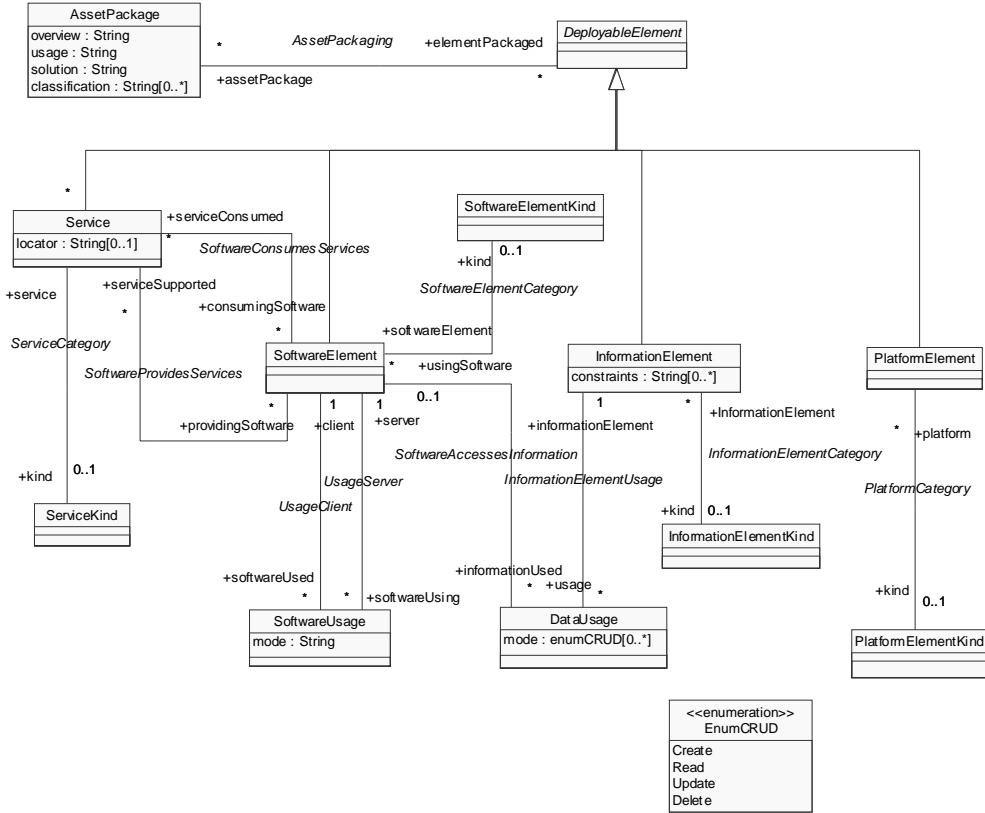
8.3.1.2 Context Diagram

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



8.3.1.3 Deployable Elements Diagram

This diagram represents the elements that may be deployed.

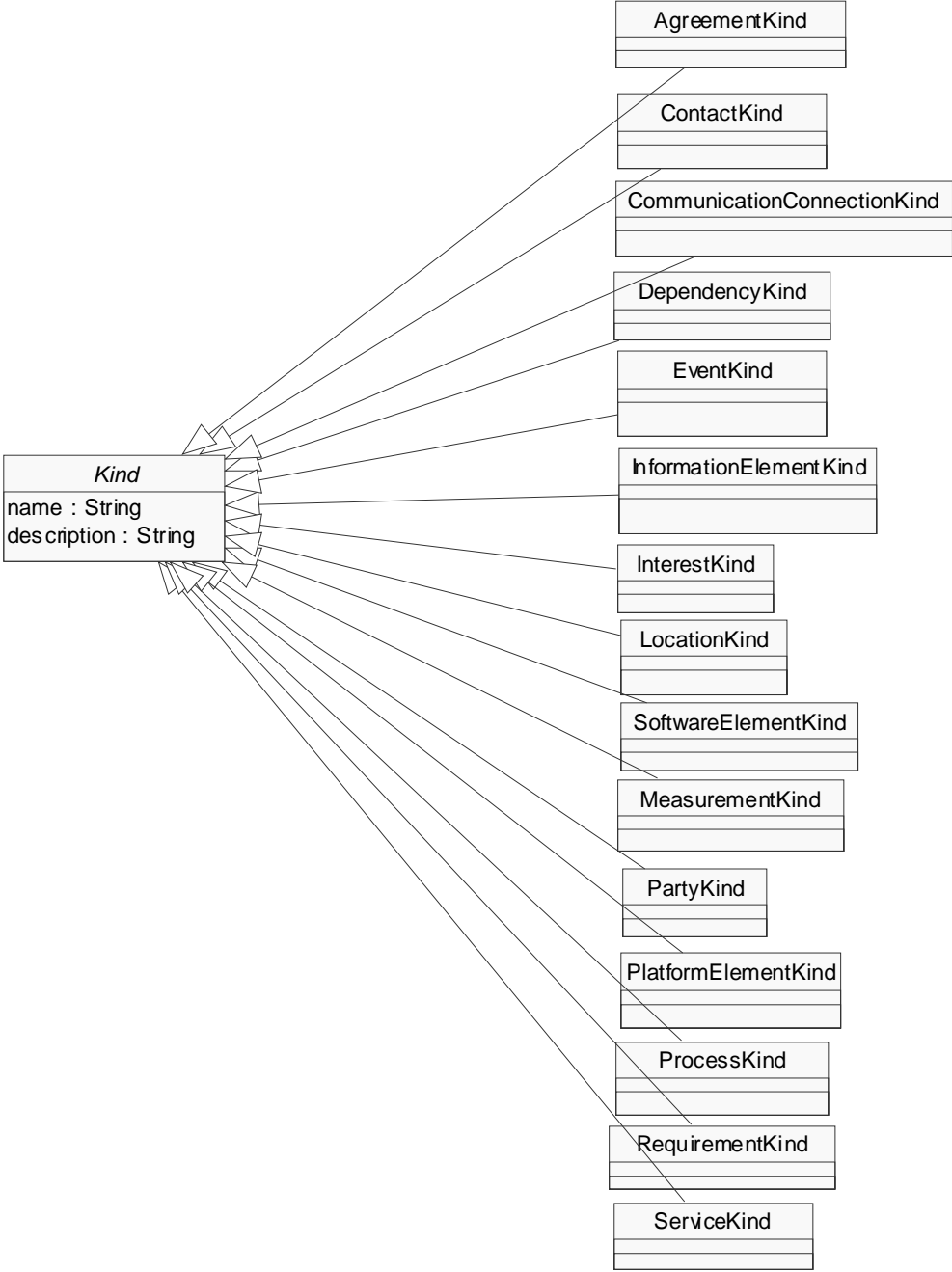






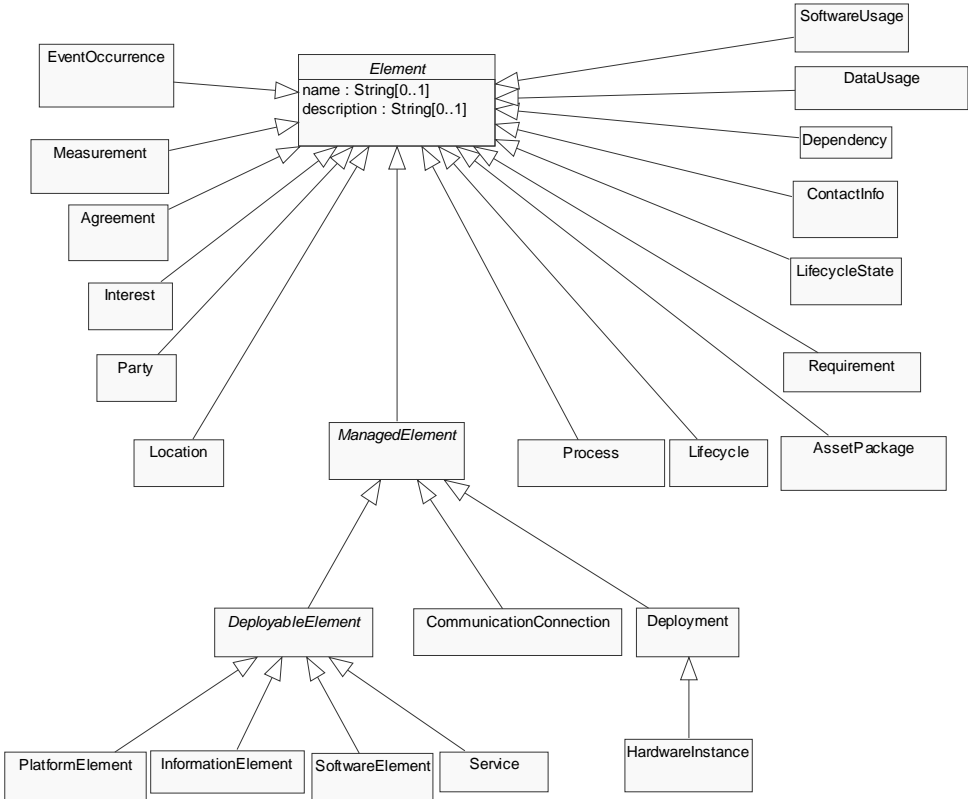
### 8.3.1.5 Kinds Diagram

This diagram shows the different subclasses of Kind.



8.3.1.6 Inheritance Diagram

This shows the inheritance structure, excluding the Kind classes.



### 8.3.2 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.

#### body

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

#### References

##### Interest interest [0..\*]

The Interests which 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 which also links to another Interest of InterestKind 'Technical Support' and a Party representing a HelpDesk team.

### 8.3.3 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 environments 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.

### 8.3.4 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 CommunicationConnections.

As a subclass of ManagedElement 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.

### 8.3.5 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.

### 8.3.6 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 of Create, Read, Update, Delete.

#### References

##### informationElement[1]

The informationElement being used/accessed.

##### usingSoftware[0..1]

The software doing the accessing.

### 8.3.7 Dependency

Represents a design-time or run-time dependency between elements which is not tracked by the specifically modeled associations. So it can link dependencies between Elements (e.g. Application X depends on an Application Serder) 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.

### 8.3.8 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.

### 8.3.9 Deployment

This represents an instance of a DeployableElement placed on a HardwareInstance. Importantly, it is itself a ManagedElement. And can participate in Dependencies 9e.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 which 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).

### 8.3.10 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.

### 8.3.11 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 which will determine how the events should be processed.

Not 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 even in which this is nested e.g a major upgrade may require a number of sub-events e.g. adding more RAM, upgrading Windows, upgrading Office, downloading and applying latest patches etc.

##### EventOccurrence childEvent[\*]

The inverse of parentEvent.

### 8.3.12 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 I a warehouse for storage.

Note that it inherits PropertyValues from ManagedElement.

### Subclass of Deployment

#### References

##### Deployment deployment [\*]

The Deployments deployed in this platform..

### 8.3.13 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.

### 8.3.14 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.

### 8.3.15 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 which 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.

## 8.3.16 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.

## 8.3.17 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 which currently have this state.

### **ManagedElementKind element [\*]**

The ManagedElements which currently have this state.

### **8.3.18 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<sup>th</sup> 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.

### **8.3.19 ManagedElement**

This abstract class represents those elements which may be managed as part of the SPMF. 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 value relevant to the Kind of the element. This can represent factors such as capacity (memory, disk) for hardware, protocols etc.

### 8.3.20 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

### 8.3.21 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 which comprise this party.

**Measurement observation[0..\*]**

The measurements observed by the party.

### 8.3.22 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

### 8.3.23 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.

### 8.3.24 PropertyValue

This represents a value associated 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.

## **8.3.25 PropertyDefinition**

### **Attributes**

#### **String name**

The name of the property

### **References**

#### **Kind kind[0..\*]**

The element Kinds that that the property is permitted for

## **8.3.26 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.

### 8.3.27 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 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.

### 8.3.28 SoftwareElement

Represents software providing some aspect of business related functionality (middleware etc may be represented as PlatformElement). Can represent different level of granularity from components/libraries up to whole ERP suites. The inherited association from ManagedElement 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.

### 8.3.29 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**

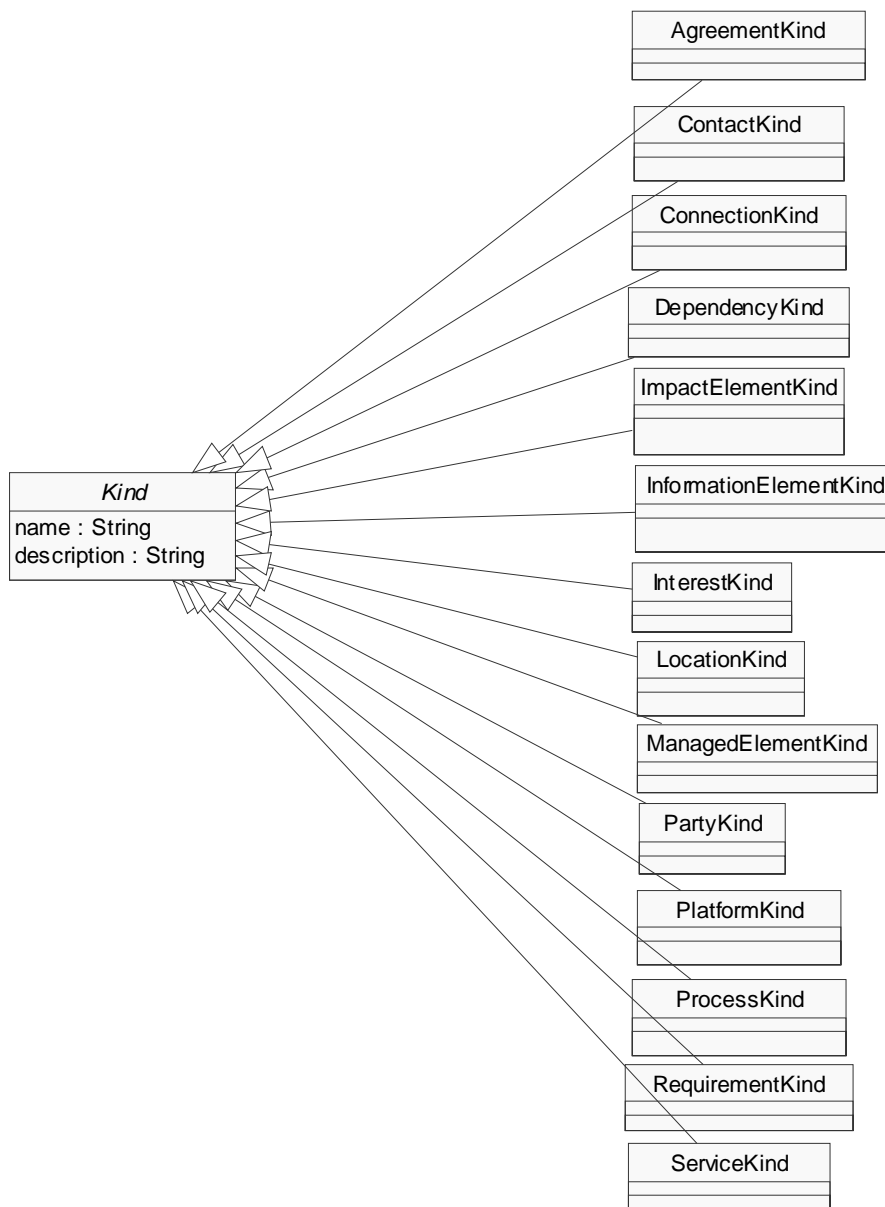
The client of this usage requesting capability.

**SoftwareElement server**

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

## 9. Kind Library

This section enumerates generally useful instances of the various Kind classes. Each subsection enumerates a set of instances by stating their name and description. These lists are not intended to be exhaustive. The provision of this Library is an optional compliance point; if they are supplied the actual text of the descriptions is non-normative.



## 9.1 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

## 9.2 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 activities
HQ Contact	Maintain a capability by providing updates (patches, new releases) as needed
Local Contact	
Mobile Contact	

## 9.3 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)

## 9.4 DependencyKind

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

## 9.5 EventKind

This represents

Change	
Risk	
Installation	
Decommissioning	
Commissioning	

## 9.6 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 activities
Maintenance Agreement	Maintain a capability by providing updates (patches, new releases) as needed

## 9.7 InterestKind

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

## 9.8 LocationKind

Country	
State	
City	
District	
Campus	
Building	
Floor	
Room	
Position	

## 9.9 ObservationKind

Total Cost of Ownership	
Number of Users	
Response Time	
Acquisition Cost	

## 9.10 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

## 9.11 PlatformKind

Server	
Desktop	
Laptop	
Database	
Networking Hardware	
Firewall	
Operating System	
Application Server	

Web Server	

### 9.12 ProcessKind

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

### 9.13 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	

### 9.14 ServiceKind

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

--	--

## 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 the Kind Library.

## PART III

## 11. Changes or extension to existing OMG Specifications

None.