SPEM 2.0 RFP ad/2004-11-04: Revised Submission

Software Process Engineering Meta-Model 2.0
OMG Draft Adopted Specification

ad/2006-06-01

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

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

LICENSES

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

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

PATENTS

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

GENERAL USE RESTRICTIONS

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

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING

LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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

RESTRICTED RIGHTS LEGEND

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

TRADEMARKS

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

COMPLIANCE

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

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

ISSUE REPORTING

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

Table of Contents

Table of Figures

1 Scope

The purpose of this document is to provide a comprehensive definition of the Software Process Engineering Meta-Model 2.0 (hereafter "SPEM 2.0", "Standard" or "Specification"). It serves as a guide for understanding the semantics of this meta-model as well its direct application in all software process modeling and enactment activities.

SPEM 2.0 is a metamodel for defining and enacting processes and their components. A tool based on SPEM would be a tool for process authoring, customizing and enacting. This Specification aims to define the minimal set of process modeling elements necessary to describe any software development process. The Standard aims to accommodate a broad range of existing and described software development processes by avoiding complex and unnecessary features or constraints.

SPEM 2.0 incorporates significant feedback from early adopters of SPEM 1.1 and responds to different methodologies and process modeling approaches. It separates the *definition* of definable process elements from their *usage* and *extension* in process workflows. This concept was ambiguous in SPEM 1.1 and was being interpreted differently by early SPEM 1.1 adopters. This SPEM 2.0 reuse pattern is also applied to process components fostering easier management and reuse of families of processes. The process refinement and tailoring mechanisms provided in this Specification enable process engineers to utilize best practices across a variety of lifecycle models and methodologies. Process enactment is now within the scope of SPEM 2.0. The meta-model supports a progressive definition of enactable process models offering the choices of content heavy processes, enactable processes or a combination of both. These, in effect, constitute logical compliance levels towards process enactment, offering SPEM 2.0 adopters an unambiguous and staged adoption path.

2 Conformance

This Specification defines the following two conformance levels. A SPEM 2.0 implementation must support at least one of these conformance levels. The levels are defined by the implementation and compliance of the two main packages defined for the SPEM 2.0 Meta-Model.

- Level 1 Conformance: Must implement the 'Process Modeling' package only.
- Level 2 Conformance: Must implement Level 1 Conformance and implement the 'Process Enactment' package.

3 References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this Specification. For dated references, subsequent amendments to or revisions of any of these publications do not apply.

[MOF 2.0] OMG, Meta-Object Facility Version 2.0, www.omg.org/mof.

[SPEM 1.1] OMG, *Software Process Engineering Metamodel Specification*, Version 1.1, January 2005, www.omg.org.

[UML 2.0] OMG, *Unified Modeling Language Version 2.0*, www.omg.org/uml, 2005.

The authors of this document would also like to acknowledge the Software Process Engineering Meta-Model 2.0 Revised Submission ad/06-04-05 (http://www.omg.org/cgi-bin/doc?ad/06-04-05). Some of the concepts in this document (ad/2006-06-01) are based on the concepts introduced in that earlier joint submission.

4 Terms and Definitions

This Specification defines a terminology for software development processes. Please refer to the individual sections of this document for definitions for each concept defined in SPEM 2.0. For the purposes of this specification, the terms and definitions given in the normative reference apply.

5 Symbols

There are no symbols defined in this Specification.

6 Additional Information

6.1 Changes to Adopted OMG Specifications

This is the third Specification defined for the Software Process Engineering Meta-Model. The Specification of SPEM 1.0 was released in 2002 (final FTF report in May 2002). SPEM 1.1 incorporated minor updates that were formally released in 2005 (final RTF report in March 2004).

This Specification addresses the following requirements from SPEM 2.0 RFO ad/2004-11-04:

- Update SPEM to be compliant with UML 2.0, taking advantage of the new functionality to improve process modeling techniques and capabilities.
- Define a new SPEM XML Schema, based on MOF 2.0. XML Schemas provide greater richness and control beyond what is available in the SPEM 1.1 XMI DTD.
- Provide guidance on migrating existing process models from SPEM 1.1 to SPEM 2.0.
- Address feedback from first implementers to address identified inconsistencies and concerns regarding the practicality and functional coverage of SPEM 1.1.
- Define extensions to SPEM that will be of use to process automation tools.
- Align SPEM with evolving and emerging standards other than UML; specifically, it may be possible to use the Business Process Definition Meta-model and the Business Process Runtime Interfaces submissions in conjunction with SPEM to provide greater value to the user community.
- Introduce process meta-model extensions that may be used equally in software development processes and systems engineering processes.

6.2 Key New Capabilities of SPEM 2.0

In addressing the SPEM 2.0 RFP requirements, this specification provides the following new capabilities:

6.2.1 Reusable process elements definition and usage in software development processes

SPEM 2.0 separates reusable process elements such as workproducts, roles and activities from their application in process workflows. Process elements are the building blocks that describe how specific development goals are achieved independent of the placement of these elements within a workflow. Process workflows take these process elements and relate them, in effect, creating work execution sequences that are customized to specific types of projects.

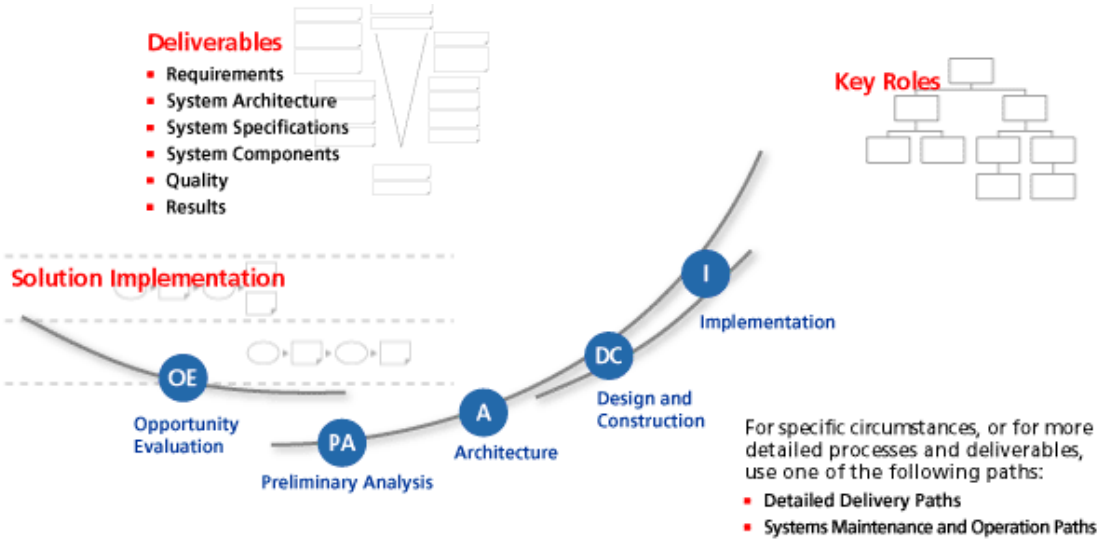

Figure 1 Process Elements from Macroscope

Figure 1 is an example from Macroscope showing common WorkProducts (called Deliverables in Macroscope) that are used by a variety of standard processes. A process determines the scope and level of details of the deliverables. It also orchestrates their production by key roles. Processes must be adapted with regard to the project circumstances and to the system characteristics. Such adaptation may consist of: replacing some streamlined deliverables by detailed ones, adding activities from the more detailed paths, specializing some roles, adding specific guidelines or referring to path-specific fundamentals, and so on.

SPEM 2.0 supports the extension of process elements as well as their usage in process workflows, ensuring a kind of live binding of the base element to the usage element. This ensures the creation of simple and flexible process models that dynamically respond to changes in base elements. This low-overhead reusability mechanism can be utilized by agile self organizing teams as well as structured software process engineering groups.

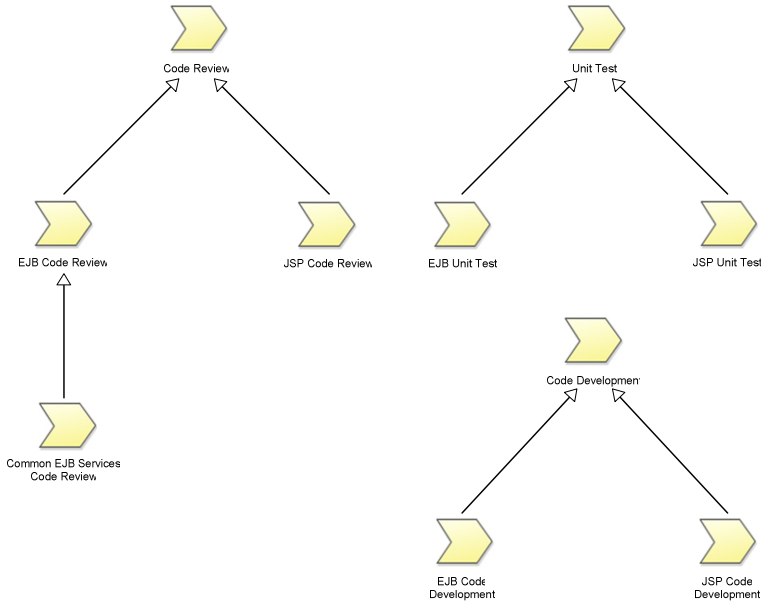

Figure 2 Specialization of base process elements

SPEM 2.0 allows the specialization of these base process elements in accordance to the project characteristics and environment. For a J2EE project, the “Code Development” activity could be specialized as “EJB Code Development” and “JSP Code Development”. Similarly, “Code Review” and “Execute Unit Tests” could be further specialized. This specialization may add, remove or modify input and output work products, performers and guidances to suit the environment context. Any changes in the general process element will automatically be reflected in all specialized elements.

6.2.2 Easy creation and maintenance of enactable development process workflows

The Specification enables the creation of enactable processes through flexible references to process elements. The mechanisms being introduced here support the creation of process workflows while enabling downstream association to process elements consumed in the workflow. It is now easy for a process engineering team to plan enactment of a process while modeling the process. With the separation of process elements definition (Base Process Elements) and their usage in process workflows, process engineers create and validate enactable process workflows with the flexibility of switching process elements used in the workflow without compromising the validity of the process workflow.

The example in Figure 3 illustrates the usage of process elements in a software development process workflow. A usage element acts as a proxy or instance representation of base process element within a process. It maintains all matching associations and content definitions defined in the Base Process Element. Because of these references, changes in the process elements will automatically be reflected in all process elements usage instances within all process workflows. Moreover these changes will not cause an adverse impact on the workflow since the usage is decoupled from the process element definition.

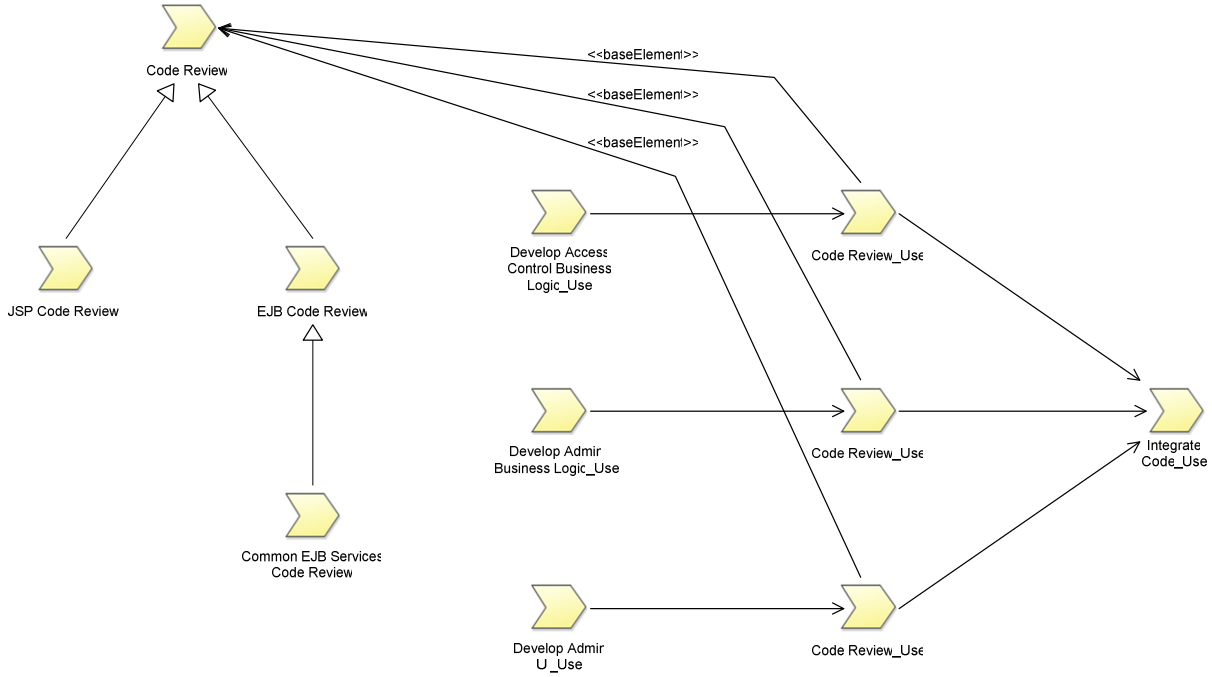

Figure 3 Usage instances application in enactable development processes

Figure 3 illustrates how the same process element "Code Review" has been applied in different places within the same process workflow.

If a new project change necessitates development on a J2EE platform, the "Code Review" activity in Figure 4 can be specialized to "EJB Code Review" and "JSP Code Review". In this way, a process engineer may update the Process Element content by updating the general base element or by changing the specialized base element. The workflow usage elements for "Code Review" would now simply switch and refer to "EJB Code Review" or "JSP Code Review" or remain pointed to "Code Review" without the need to reconstruct the workflow. This adjustment can be made in the existing workflow, retaining any complex work sequence structure that may have been created. This is possible since when a usage element is created, it maintains reference to the base element and its root base element. Usage element may change its reference binding to any child base element extending the root base element. This change in reference may only apply to extending base elements that have the same root base element as previous base reference.

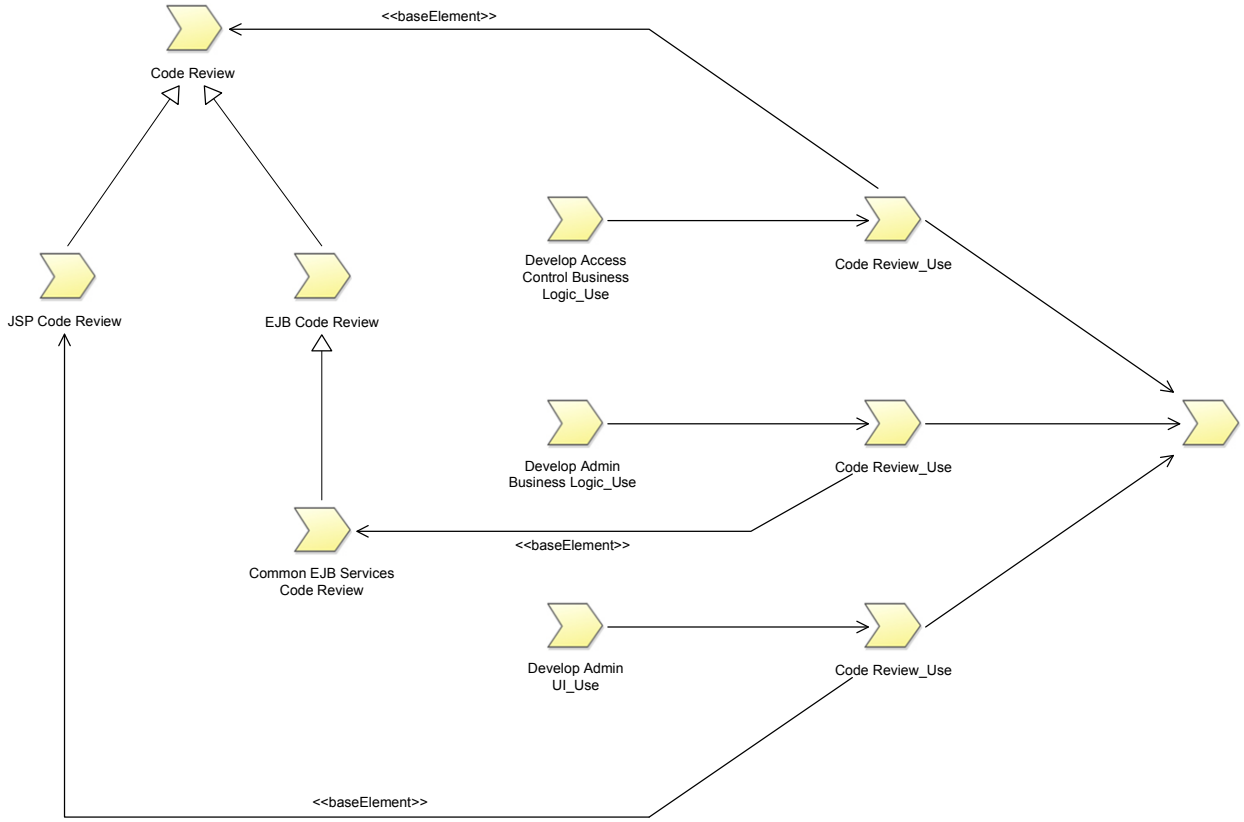

Figure 4 Easy maintenance of development processes

6.2.3 Extensible process components for rapid process assembly

SPEM 2.0's Process Components are reusable building blocks for creating new development processes or larger Process Components. SPEM 2.0 supports reusing process knowledge by factoring out commonly reoccurring work into Process Components which can then be applied over and over again in a process. A SPEM tool could implement a mechanism such that whenever the Process Component is revised or updated; all changes are automatically reflected in all processes that applied that Process Component.

Input and output work products can be defined for each process component, allowing the user to treat the actual definition of the work that produces the outputs as a "black box". However, the process engineer has visibility into the component and can modify it if required. The component approach also allows different styles or techniques of doing work to be replaced with others. For example, a software code output of a component could be produced with model-driven development or a code-centric technique. The component concept encapsulates the actual work and lets the project team choose the appropriate technique.

6.2.4 Support for different lifecycle models

The concepts presented in SPEM 2.0 support different kinds of lifecycle models. The process element and component usage and extensibility patterns introduced here accommodate multiple lifecycle modeling approaches. Figure 5 shows an example of three processes with different lifecycle models from Macroscope.

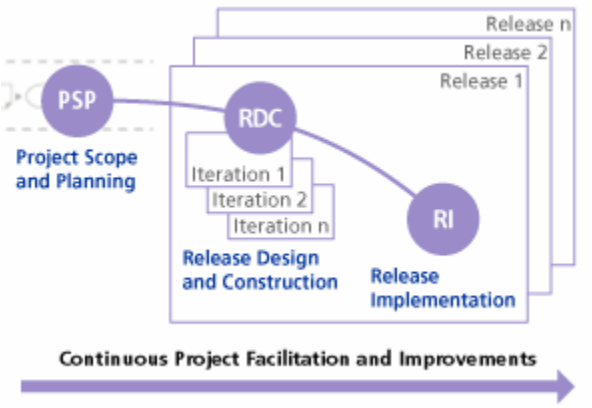

Figure 5 Three processes with different lifecycle models

6.2.5 Support for Enactment

The addition of a Work Breakdown Element state machine in the meta-model provides process engineers increased flexibility and control over their process workflows and their enactment. Work Breakdown element states can now be used alongside Work Product states to create machine readable Work Breakdown Element pre- and post-conditions as well as fork and join node constraints in workflow decisions. If process engineers choose to have machine readable conditions, an enactment system would be able to evaluate these conditions and automate the activity state changes without the need for direct human involvement.

Although process engineers can extend or tailor a process for a particular project type or project instance, Activities are not necessarily the units of work that get assigned to project practitioners during enactment. Tasks and Work Items (new in SPEM 2.0) enable process engineers and project managers to create enactment specific entities that are assignable to practitioners. Associating Tasks with Work Breakdown Elements makes the process more transparent to the practitioner in an enactment system. The enactment system would help practitioners focus on their current assignments and avoid process information overload by filtering the process for practitioners, ensuring that they are provided only the information that they need when they need it. For example, if a practitioner is assigned to a Task that is only responsible for a subset of an Activity's steps, the enactment system could filter the Activity's associated WorkProducts and Guidances for that user based on their associations with those steps. Linking these enactment Tasks and Work Items with the process also facilitates data rollup and process feedback and improvement by allowing enactment data and metrics to be fed back to the process engineers and project managers. To continue the previous example, when the Task is complete, the enactment system could mark the steps as complete or possibly change the Activity's state depending on the status of the remainder of its steps.

6.3 Architecture Alignment and MDA Support

This Specification documents the Software Process Engineering Meta-Model 2.0 Meta-Model (SPEM 2.0 Meta-Model) and the Software Process Engineering Meta-Model 2.0 UML 2.0 Profile (SPEM 2.0 Profile).

This document provides a MOF 2.0 compliant meta-model Specification for the SPEM 2.0 Meta-Model as depicted on the left-hand side in Figure 6.

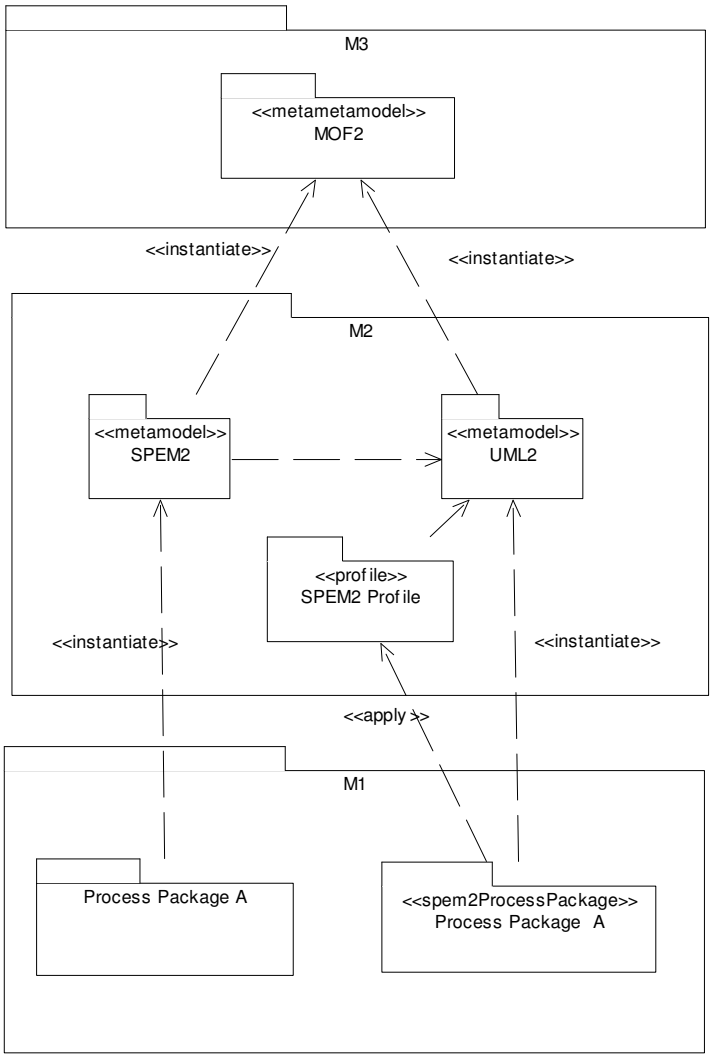

Figure 6 Model Layers for UML and SPEM 2.0

The SPEM 2.0 Meta-Model is a MOF 2.0 compliant Meta-Model

A model defined on a higher layer defines the language to be used on the immediate layer below it. MOF is the universal language that can be used on any layer, but in our case, MOF is instantiated from the M3 layer by SPEM 2.0 on the M2 layer. The UML 2.0 meta-model itself, as depicted on the right-hand side of the M2 layer, instantiates MOF2 defined on M3 layer in the exact same way. "Process Package A" is an example of a concrete instance of the SPEM 2.0 meta-model using SPEM 2.0 as a schema to represent its content. In that sense, "Process Package A" represents a method model. For example, SPEM 2.0 defines the concepts of Roles, Work WorkProducts and Activities, as well as relationships between them. Process Package A on the M1 layer provides concrete instances of Roles and Artifacts from the M2 layer such as "System Analyst" and "Use Case". As illustrated in Figure 7, "Use Case" is a direct instance of the meta-class "Artifact", which is an instance of the Meta-Meta Class "Class" from the M3 layer. A use-case instance that one would create during a development project, such as "Browse Catalog" for a Web-based sales system, would now be an instance of the class "Use Case" on the M0 layer.

The SPEM 2.0 Meta-Model reuses parts of UML 2.0

The SPEM 2.0 meta-model describes all structures and attributes needed to represent SPEM 2.0-based methods and processes. However, SPEM 2.0 does not define all of its elements from scratch, but actually reuses elements from the UML 2.0 meta-model. Figure 6 shows a dependency on the M2 layer from SPEM 2.0 to UML2. This dependency expresses that parts of the SPEM 2.0 are based on definitions in the UML 2.0. For example, core elements of SPEM 2.0 such as "Process Element" and "Process Package" have been derived through specialization from classes from the UML 2.0 Infrastructure Library inheriting relationships that allow the definition of packages and packageable elements. SPEM 2.0 also uses UML 2.0 state machines.

The SPEM 2.0 Meta-Model has a reference implementation

The SPEM 2.0 meta-model can be directly instantiated for an implementation, i.e. a CASE tool that now represents all classes from the M2 layer as Java classes and database tables. Although, the SPEM 2.0 MOF meta-model is defined in UML, an instance of the model (i.e. a concrete method or process) can be represented independent of the UML.

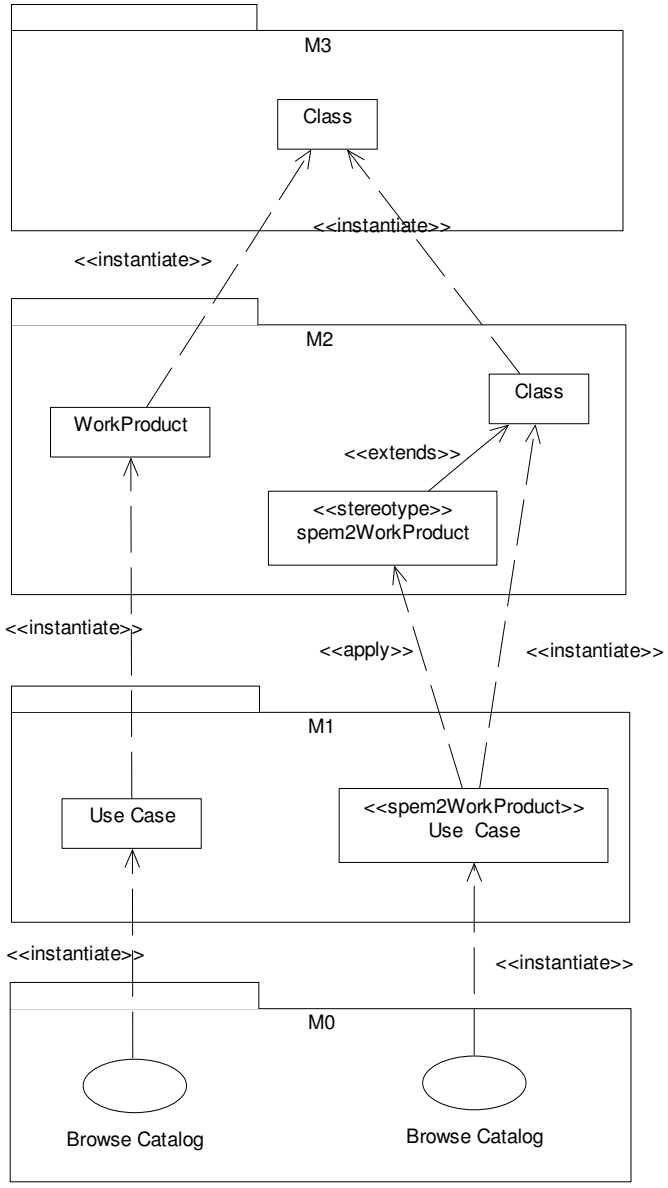

Figure 7 Exemplary instantiations of the modeling layers

The SPEM 2.0 Profile is a UML 2.0 Profile that provides an alternative representation to the SPEM 2.0 Meta-Model

In addition to representing a method library such as 'Process Package A" with these structures creating your own implementation of these classes (e.g. using the Java classes mentioned above), one could also decide to represent classes from the M1 layer with a generic UML 2.0 modeling tool. In this case, one would use UML Superstructure classes on the M2 layer and extend these with the official UML2 extension mechanisms by providing profiles with stereotypes and OCL constraints as depicted in Figure 7. For example, in Figure 7, a stereotype declaration for WorkProduct extends the UML 2.0 class concept. An instance on the M1 layer would be a UML 2.0 class which has this stereotype assigned. An M0 instance would still look the same. The only difference is the formalized representation used for the meta-model (M2) and model (M1).

SPEM 2.0 defines a Meta-Model as well as a UML Profile

This Specification provides definitions for both representations:

- The SPEM 2.0 Meta-Model: Defines all structures and structuring rules and is in itself complete.
- The SPEM 2.0 UML Profile: Defines a set of UML 2.0 stereotypes which allow presenting SPEM 2.0 process elements and processes using the UML 2.0. However, the definition of these stereotypes in this Specification only covers their presentation, but relies on the SPEM 2.0 Meta-Model for all semantic definitions and constraints. In other words, the profile does not contain any OCL constraints, but relies on the SPEM 2.0 Meta-Model semantics to define all of its constraints.

6.4 How to Read this Specification

The rest of this document contains the technical content of this Specification.

Although the chapters are organized in a logical manner and can be read sequentially, this is a reference Specification intended to be read in a non-sequential manner. Consequently, extensive cross-references are provided to facilitate browsing and search.

6.5 Acknowledgements

The following companies submitted, supported or contributed to parts of the SPEM 2.0 Specification:

- Borland Software Corporation (Submitter)
- Osellus Inc. (Submitter)
- Bell Canada
- Microsoft Corporation
- Sun Microsystems

The following individuals are acknowledged for their contribution to SPEM 2.0:

Omid Afnan, Kamal Ahluwalia, April Alibayan, Reda Bendraou, Ming Chan, Aaron deVries, Karl Frank, David Gowan, Payman Hodaie, Rajendra Kanakamedala, Jim Lu, Randy Miller, Daryl Spencer, Harvey Stein, Sammy Wahab.

7 Package Structure

The SPEM 2.0 Meta-Model reuses the UML 2.0 Infrastructure library. It derives all of its classes from classes defined in this library

The SPEM 2.0 Meta-Model utilizes the UML 2.0 diagram interchange for the presentation of various diagrams.

To generate the SPEM 2.0 XMI schema, all packages in Software Process Engineering Meta Model 2, UML2 Diagram Interchange, and Infrastructure Library will be collapsed into the spem2 package. As a result all classes will be defined in a flat list.

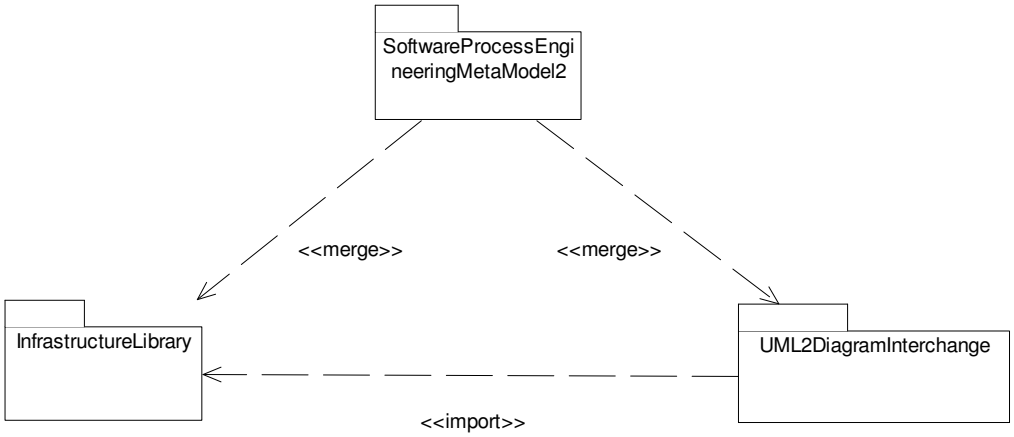

Figure 8 SPEM 2.0 reuses the UML 2.0 Infrastructure and Diagram Interchange standards

The SPEM 2.0 Meta-Model is structured into two main packages as depicted in Figure 9. The structure divides the model into two logical units. The UML 2.0 package merge mechanism applied to the packages realizes a gradual extension of the capabilities.

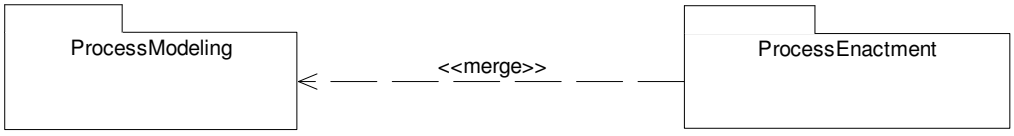

Figure 9 Structure of the SPEM 2.0 Meta-Model

The packages depicted in Figure 9 provide the following capabilities:

Process Modeling: This package defines reusable core process elements such as Work Products, Roles and Activities. It supports the creation of process workflows that take these process elements and relate them to create work execution sequences that are customized to specific types of projects. It introduces concepts for the extension of process elements as well as their usage in process workflows ensuring live binding between base elements and usage elements. This ensures the creation of simple and flexible process models that dynamically respond to changes in base elements. This low-overhead reusability mechanism can be utilized by agile self organizing teams as well as structured software process engineering groups.

Process Enactment: The Process Enactment package does not attempt to describe a full enactment system, but introduces process element extensions to guide process engineers and project managers in tailoring and facilitating process enactment and improvement. Software development processes that are created using the Process Modeling package alone are not always sufficient for enactment. Project managers and practitioners typically have difficulty applying these processes to their daily activities, making process enforcement and improvement difficult. By allowing process engineers and project managers to decompose the process into discrete units of work that can be assigned to individuals and tracked based on the workflow and events defined in the Process Modeling package, this package enables the formal process to be abstracted from project practitioners.

Project enactment is typically carried out by creating and assigning a series of tasks and work items to project practitioners. Adding Tasks and Work Items to the meta-model and allowing them to be associated with Work Breakdown Elements allows the process to become transparent to the practitioner in an enactment system. The enactment system would help practitioners focus on their current assignments and avoid process information overload by filtering the process for practitioners, ensuring that they are provided only the information that they need when they need it. These Tasks and Work Items also facilitate data rollup and increased process feedback and improvement by allowing enactment data and metrics to be fed back to the process engineers and project managers during enactment.

8 Process Modeling

The Process Modeling package contains the basic structural process elements for defining software development processes. It provisions the definition of lifecycle-independent reusable process elements that provide a base of documented knowledge of software development methodologies, techniques, and concrete realizations of best practices. Process workflows organize these process elements into sequences that are customized to specific types of projects.

A development process defines how development projects shall be executed. It focuses on the actions to be performed, and their sequence, to deliver the expected result or series of events. Each activity that makes up a process involves participants and may be triggered by preconditions and results in postconditions.

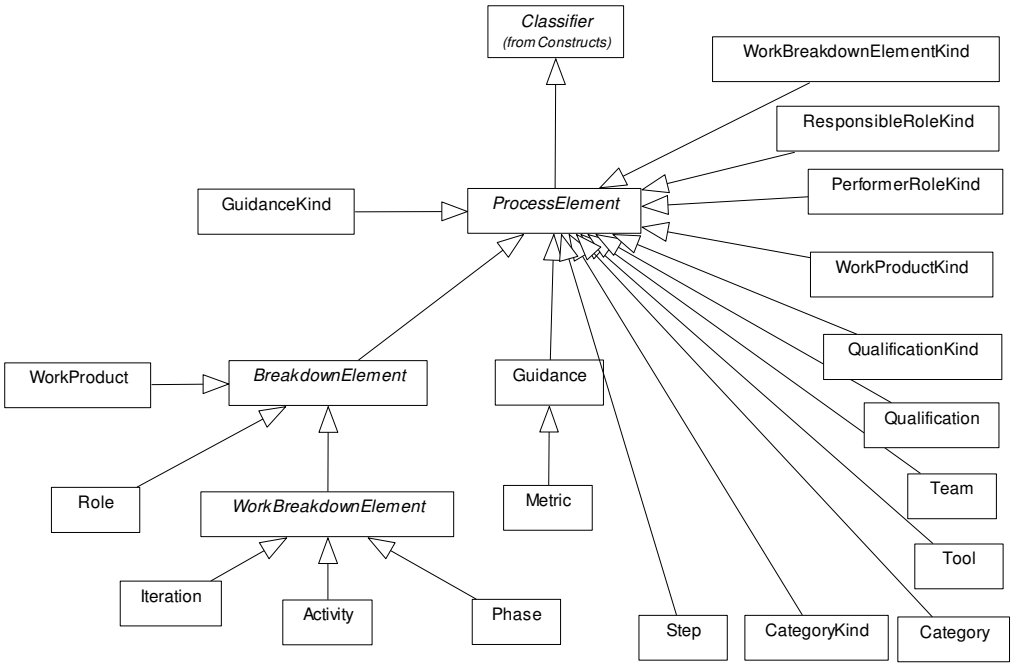

Figure 10 Taxonomy of classes defined in Process Modeling

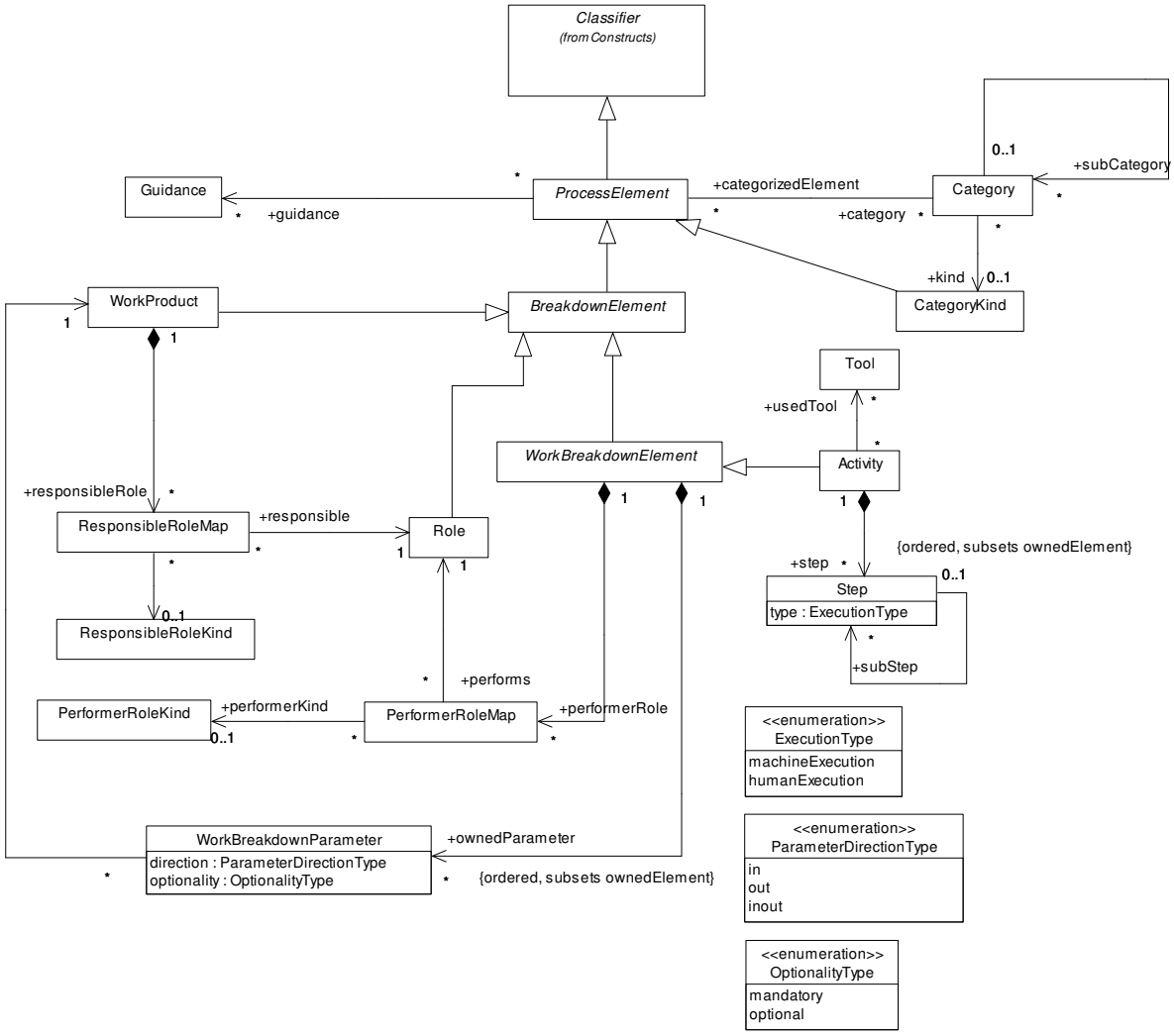

Figure 11 Base Process Elements

Process Elements are the basic structural building blocks of a process. They are described by defining Activities that have Steps, input and output Work Products and are performed by Roles. Roles also define an important responsibility relationship for work products. Figure 11 depicts these core process elements with their relationships.

8.1 Process Element

Super Class

Classifier

Description

A Process Element is an abstract generalization that represents basic structural process elements that can be used to define a software process engineering model. Process Elements provide the information that is necessary to construct a process.

Attributes

- description: String — A Process Element can have a human readable description of the element. Normally this description is expected to be one or two paragraphs in length; however there is no restriction on size specified by this document.

Associations

- guidance: Guidance — A Process Element can be related to many guidances.
- category: Category — Process Elements can be categorized by many categories.

Rationale

Process Elements enable the definition of basic structural elements needed to define a process model based on any methodology. A Process Element could be described using the description attribute and may have related guidances providing additional information as needed.

Changes from previous SPEM

Process Element is new in SPEM 2.0.

8.2 Category

Super Class

Process Element

Description

A Category is a Process Element used to categorize, i.e. group any number of Process Elements of any subtype and Process Components based on user-defined criteria. Because Categories are Process Elements themselves, Custom Categories can be used to recursively categorize Categories as well. Categories can also be nested with any Category.

Attributes

No additional attributes.

Associations

- categorizedElement: ProcessElement — A Category groups together any number of Process Elements (including other Categories).
- subCategory: Category — A Category can have any number of Categories defined as nested sub-categories. Therefore, one could nest Category into n-level hierarchies.
- kind: CategoryKind — Every Category instance can be classified by a Category Kind.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Category	Process Element	spem2Category	n/a

Examples

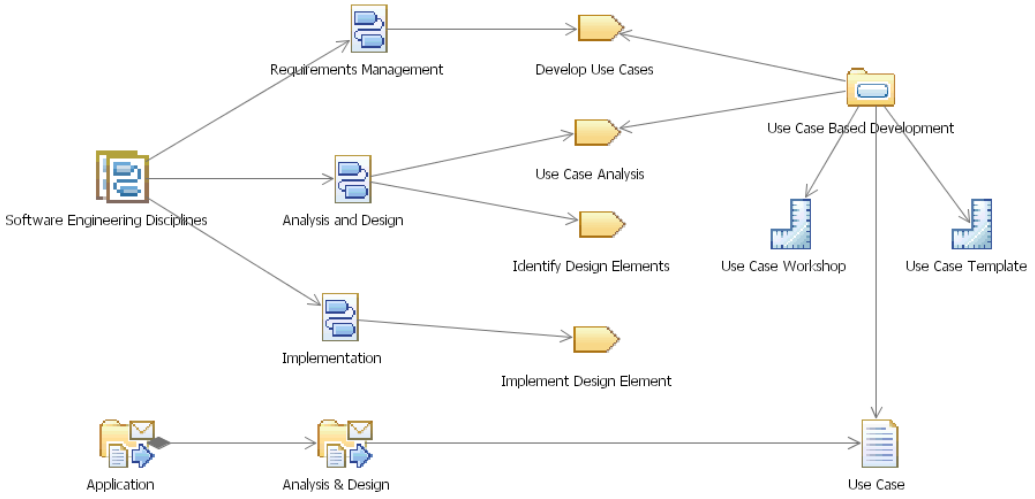

Figure 12 RUP example for categories and categorized process elements

Figure 12 shows examples for categorized process elements. It defines the Discipline Grouping "Software Engineering Discipline" that categorizes three Disciplines. The Disciplines categorize Activities such as "Develop Use Case", "Use Case Analysis", etc. Two of these Activities have also been categorized by a Custom Category "Use Case Based Development" that a user might have defined to provide a 'filter' on the process elements related to use cases. The user has also assigned the use case work product from the "Analysis & Design" domain, as well as specific Guidance (template and workshop guideline) to this category.

Rationale

Enable filtering process elements based on user-defined categories for inclusion in process components and processes or generating presentation structures by a tool implementation.

Changes from previous SPEM

Category is new in SPEM 2.0. It replaces Categorizes Dependency from SPEM 1.1. This SPEM 1.1 dependency did not work well with packages, because process engineers wanted to define multiple categories that all categorized the same element.

8.3 Category Kind

Super Class

Process Element

Description

A Category Kind is a flexible way of defining different groupings for Categories on the M1 level, i.e. the level on which a user defines process models. Category Kind is Process Element itself that contains descriptions about the characteristics of the Categories that it groups. Every Category can be associated with a Category Kind element. The name of the Category Kind indicates what kind of Category it is.

Attributes

No additional attributes.

Associations

No additional associations.

Examples

See Appendix D for Category Kind examples.

Changes from previous SPEM

Category Kind is new in SPEM 2.0.

8.4 Guidance

Super Class

Process Element

Description

Guidance is a Process Element that provides additional information related to process elements such as Roles, Activities, and Work Products. The particular Guidance is classified with a guidance kind that indicates a specific type of guidance. Examples for Guidance are Guidelines, Templates, Checklists, Tool Mentors, Estimates, Supporting Materials, Reports, Concepts, etc.

Attributes

No additional attributes.

Associations

- kind: GuidanceKind A Guidance element can be associated to a Guidance Kind.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Guidance	Process Element	spem2Guidance	

Changes from previous SPEM

Guidance is unchanged from SPEM 1.1.

8.5 Guidance Kind

Super Class

Process Element

Description

A Guidance Kind is a flexible way of defining different groupings for guidance. A Guidance element can be associated with a Guidance Kind element. The name of the Guidance Kind indicates what kind of Guidance it is.

Attributes

No additional attributes.

Associations

No additional associations.

Examples

See Appendix D.

Rationale

Apart from constraining the values of Guidance Kinds to a known set of instances it also enables describing how a kind of Guidance may be used and maintained. It is essential for keeping the meta-model flexible and inclusive of methodologies.

Changes from previous SPEM

Guidance Kind is unchanged from SPEM 1.1.

8.6 Metric

Super Class

Guidance

Description

A Metric is special Guidance that contains a number of constraints which provide measurements for any Process Element. Because Metric is Guidance, different Guidance Kinds can be defined for Metrics to distinguish different groups of Metrics such as Productivity, Quality, or Scale (example for Guidance Kinds for Metrics).

Attributes

No additional attributes.

Associations

No additional associations.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Metric	Guidance	spem2Metric	

Changes from previous SPEM

Metric is new in SPEM 2.0.

8.7 Breakdown Element

Super Class

Process Element

Description

Breakdown element is an abstract generalization for any type of Process Element that is part of a breakdown structure. When creating a process or process component, a Breakdown Element is never used directly; instead, instance representation of Breakdown Element will be used to define a process or process component. Multiple instances of a Breakdown Element may be defined within the same process or process component.

Attributes

- hasMultipleOccurrences: Boolean — When this attribute is set to True for a Breakdown Element then it will typically occur multiple times within the same process or process component. For example, an Activity such as "Detail Use Case" would be performed for every use case identified for a particular Iteration in a process or process component.
- isOptional: Boolean — The isOptional attribute indicates that the Breakdown Element describes work, a work result, or even work resources, the inclusion of which is not mandatory when performing a project that is planned based on a process containing this element.

Associations

- suppressed: ProcessElement — Suppressed defines the process element associations that will be hidden from the extending breakdown element. In addition, suppressed process element associations will not be visible to use elements that trace to the breakdown element.

Semantics

Extension is a taxonomic relationship between a more general Breakdown Element and a more specific Breakdown Element. It provides a capability for Breakdown element content variation; the specific Breakdown instance may be viewed as an instance of general Breakdown instance, hence, all visible features of the general breakdown element are implicitly defined for the specific Breakdown element. The Extension association is only instantiated between two Breakdown Element sub classes with the same concrete type. In addition, each concrete Breakdown Element may only maintain one specific reference to a general Breakdown Element.

When defining Extension between Breakdown Elements, the specific Breakdown element attribute interpretation would apply following rules.

0..1-association instances	Association instance of the based-on general Element is kept if it is visible (not suppressed at the general Element) to the specific breakdown element. If the Specialized element has defined its own association, the supplier element's association is ignored.
0..n-association instances	Associations of the specialized element are added to the already existing association instances of the general element. If both general associations and specialized associations refer to the same object, the general associations are ignored. By default, associations from specialized element and general element are not suppressed.

The specific Element may suppress any association defined in supplier through the suppress association. For any association defined in a general and current specific element; if the associated element is suppressed, then it will not be visible to the referenced use element (Breakdown Element Use) and the specific element that references it.

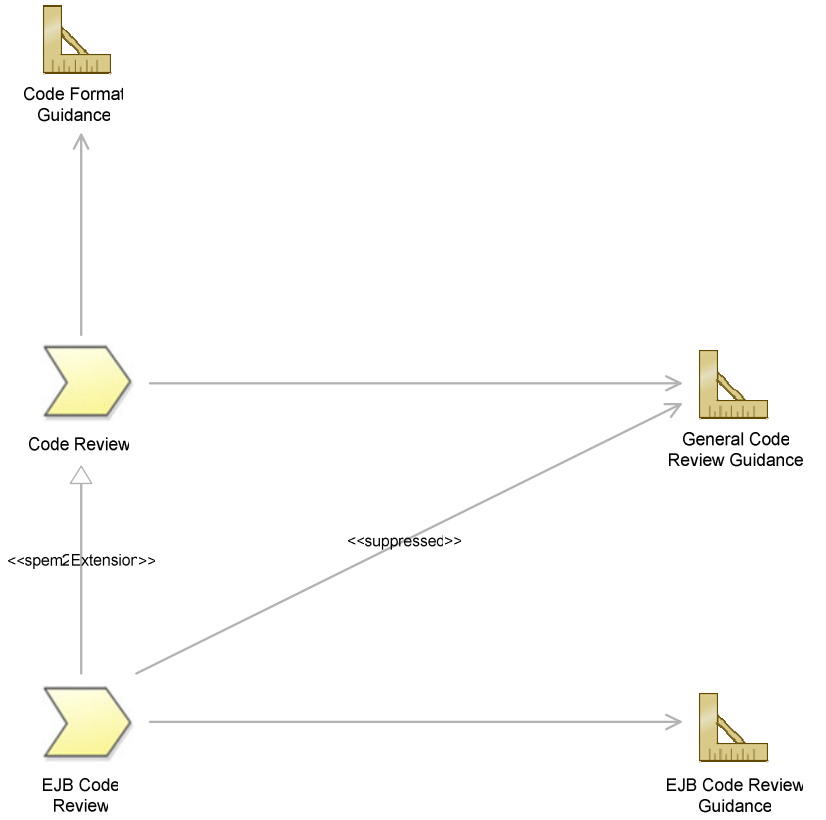

Figure 13 Breakdown Element Definition View

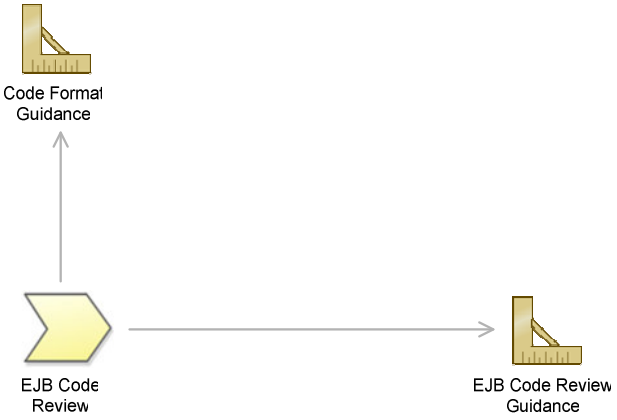

Figure 14 Consolidated View

Rationale

Breakdown Element is an abstract class that defines a set of attributes available to its specializations.

Changes from previous SPEM

Breakdown Element is new in SPEM 2.0.

8.8 Work Product

Super Class

Breakdown Element

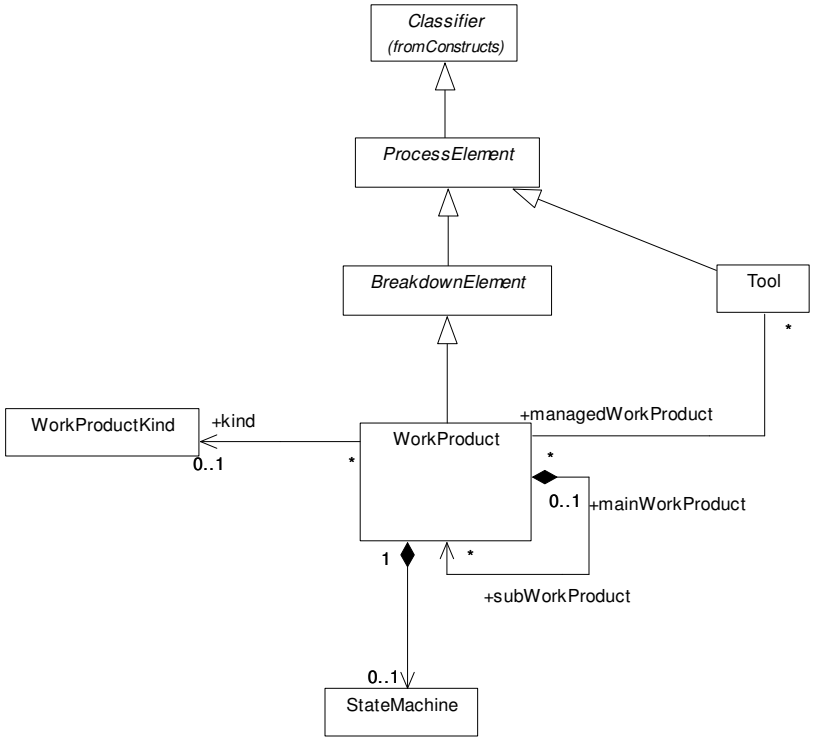

Figure 15 WorkProduct taxonomy

Description

WorkProduct provides a description and definition for all work product artifacts. WorkProducts may be composed of other WorkProducts.

Attributes

- isDeliverable: Boolean — The isDeliverable attribute is true if that WorkProduct is defined as a formal deliverable of the process. Deliverable is not a basic structural process element in SPEM 2.0 because not all WorkProducts are deliverable, and whether a WorkProduct is delivered or not may change during enactment.

Associations

- subWorkProducts: WorkProduct — WorkProducts may be composed by other WorkProducts. For example, a use case model is composed of use cases and actors.
- workProductKind: WorkProductKind — Defines WorkProductKind information based on a modeler's needs. For example, Artifact, Deliverable, Code, Document and so on.
- stateMachine: StateMachine — A WorkProduct be may associated with a state machine that describes the states that the work product may be in, and the transitions allowed

between those states. UML 2.0 State Machine has been reused.

- responsibleRole: Role — Multiple roles may be responsible for the same Work Product. See ResponsibleRoleMap.

Semantics

Work Products are artifacts consumed, produced, or modified by an Activity. Multiple roles may be responsible for the same Work Product.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
WorkProduct	Breakdown Element	spem2WorkProduct	
subWorkProduct	Composite Aggregation	n/a	n/a

Examples

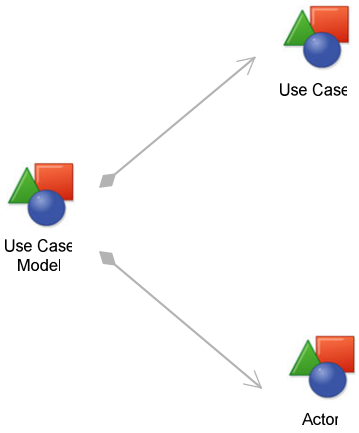

Figure 16 WorkProduct composition example

Changes from previous SPEM

WorkProduct is unchanged from SPEM 1.1.

8.9 WorkProduct Kind

Super Class

Process Element

Description

A WorkProduct Kind describes a type of work product, such as Deliverable, Artifact, Text Document, UML Model, Executable and so on. The range of WorkProduct kinds is dependent on the process being modeled.

Attributes

No additional attributes.

Associations

No additional associations.

Rationale

Apart from constraining the values of WorkProduct Kinds to a known set of instances it also enables describing how a kind of WorkProduct may be used and maintained. It is essential for keeping the meta-model flexible and inclusive of methodologies.

Changes from previous SPEM

WorkProduct Kind is unchanged from SPEM 1.1.

8.10 Responsible Role Map

Super Class

Classifier

Description

A Responsible Role Map declares a Work Product's responsible Roles; these are the Roles that are associated to the Work Product with varying levels of responsibilities.

Attributes

No additional attributes.

Associations

- responsibleKind: ResponsibleRoleKind — This association represents the kind of relationship a Role has with the Work Product. If no ResponsibleRoleKind is specified the associated role is considered to be responsible for the Work Product.
- role: Role — Defines the work product's responsible role.

Examples

A WorkProduct-based methodology may define explicit relationships between WorkProducts and Roles instead of relying on indirect relationships through formal activities. For example, a methodology may use the following values for ResponsibleRoleKind: Owned, Reviewed, Approved, and Produced. This methodology may then have a Quality Plan that has the following associations:

- Quality Manager – Owned, Produced
- Project Manager – Reviewed
- SEPG – Approver

Changes from previous SPEM

Responsible Role Map is new in SPEM 2.0.

8.11 Responsible Role Kind

Super Class

Process Element

Description

A Responsible Role Kind describes a type of Responsible Role.

Attributes

No additional attributes.

Associations

No additional associations.

Changes from previous SPEM

Responsible Role Kind is new in SPEM 2.0.

8.12 State Machine

Super Class

Protocol State Machine

Description

State Machine extends Protocol State Machine from UML 2.0 Meta Model; it inherits all behavior model elements and allows trigger definition for different transition expression.

8.13 Tool

Super Class

Process Element

Description

A Tool is a special Process Element that can be used to specify a tool's participation in an Activity.

Attributes

• version: String	Avoids confusion between different tool versions used.

Associations

• managedWorkProduct: WorkProduct	A Tool can manage instances of one or more Work Products. For example a Tool can be modeled that specializes in managing Use Case Models or another Tool that manages Analysis and Design Models.

Semantics

A Tool describes the capabilities of a CASE tool, general purpose tool, or any other automation unit that supports the associated instances of Roles in performing the work defined by an Activity. A Tool can represent a resource useful, recommended, or necessary for an activity's completion.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Tool	Process Element	spem2Tool	n/a

Changes from previous SPEM

Tool is new in SPEM 2.0.

8.14 Role

Super Class

Breakdown Element

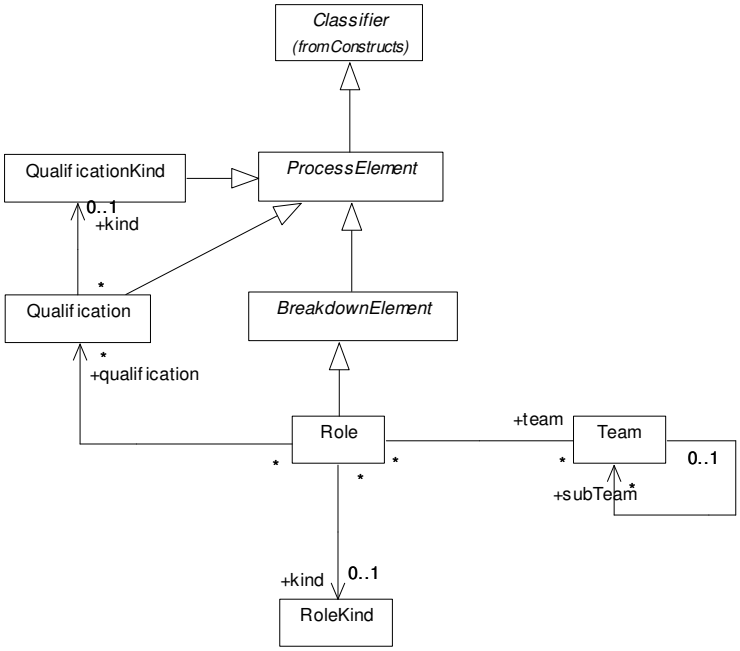

Figure 17 Role taxonomy

Description

A Role is a Breakdown Element that defines a set of related skills, competencies, and responsibilities. Roles are used by activities to define who performs them as well as define a set of workproducts they are responsible for.

Attributes

No additional attributes.

Associations

- /responsible: WorkProduct — Responsibility for a workproduct indicates the role's ownership of all instances of this workproduct. For example, the role responsible for a workproduct answers to management about its quality state. Being responsible for the workproduct does not mean that the role is the only one modifying it, but it means that the role makes ultimate decisions about accepting or rejecting these modifications when reviewing these modifications. Every workproduct has none or one responsible role.
- /performs: WorkBreakdownElement — Performance of a workbreakdown element indicates the role's involvement of all instances of this workbreakdown element. For example, a role may be involved in a consultative capacity for some work whereas as a primary contributor for others.
- roleKind: RoleKind — This association defines role grouping information based on

modeler's needs.

- qualification: Qualification — This association defines the role's qualifications.
- team: Team — This association defines the team membership.

Semantics

A Role defines a set of related skills, competencies, and responsibilities for that role. Roles are not individuals or resources. Individual members of the development organization will wear different hats, or perform different roles. The mapping from individual to role, performed by the project manager when planning and staffing for a project, allows different individuals to act as several different roles, and for a role to be played by several individuals.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Role	Breakdown Element	spem2Role	

Rationale

A role definition needs to describe the responsibilities and associations qualifications for a meaningful role assignment to activities and work products. This is even more critical for staffing individuals to those roles during enactment.

Changes from previous SPEM

Process Role in SPEM 1.1 has been renamed to Role in SPEM 2.0.

8.15 Role Kind

Super Class

Process Element

Description

A Role Kind describes a type of Role.

Attributes

No additional attributes.

Associations

No additional associations.

Rationale

Apart from constraining the values of Role Kinds to a known set of instances it also enables describing how a kind of Role may be used and maintained. It is essential for keeping the meta-model flexible and inclusive of methodologies.

Changes from previous SPEM

Role Kind is new in SPEM 2.0.

8.16 Team

Super Class

Process Element

Description

A team is an organized grouping of roles that collectively focus on common work units.

Attributes

No additional attributes.

Associations

- subTeam: Team — A team may consist of sub-teams.
- teamRole: Role — A team may be associated with many roles.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Team	Process Element	spem2Team	

Changes from previous SPEM

Team is new in SPEM 2.0.

8.17 Qualification

Super Class

Process Element

Description

Qualifications define a skills profile that will help a user in fulfilling the responsibilities of the role.

Attributes

No additional attributes.

Associations

No additional associations.

Changes from previous SPEM

Qualification is new in SPEM 2.0.

8.18 Qualification Kind

Super Class

Process Element

Description

A Qualification Kind describes types of Qualifications, such as Skills, Certifications, Job Role (HR Level) and so on. The range of Qualification kinds is dependent on the project environment for which the process is being modeled.

Attributes

No additional attributes.

Associations

No additional associations.

Changes from previous SPEM

Qualification Kind is new in SPEM 2.0.

8.19 Work Breakdown Element

Super Class

Breakdown Element

Description

A Work Breakdown Element is a special Breakdown Element that provides specific properties for Breakdown Elements that are part of a Work Breakdown Structure.

Attributes

- isRepeatable: Boolean — This attribute is used to define repetition of work, e.g. iterations. A Work Breakdown Element with this attribute set to True can be repeated more than once on the same set of Work Products.
- isOngoing: Boolean — If the isOngoing attribute is set to True for a Work Breakdown Element, then the element describes an ongoing piece of work without a fixed duration or end state. For example, the Activity could represent work of an administrator continuously (e.g. 3h a day) working to ensure that systems are kept in a certain state. Another example would be program management work overseeing many different projects being scheduled for one particular project at specific reoccurring intervals during the whole lifecycle of the project.
- isEventDriven: Boolean — The isEventDriven attribute indicates that the Work Breakdown Element describes an instance of work which is triggered because another specific event has occurred and not because it has been scheduled to start at a certain point of time or because preceding work is being completed, or because input work products are available.

  Examples for such events are exceptions or problem situations which require specific work to be performed as a result. Also change management work can be modeled as event driven work analyzing a change request or defect and allocating work dynamically to resources to

deal with it following the work described with such Activity.

- hasMilestone: Boolean — A Milestone describes a significant event in a development project, such as a major decision, completion of a deliverable, or meeting of a major dependency.
- milestoneDescription: String — A human-readable brief description of the milestone.

Associations

- Kind :WorkBreakdownElementKind — WorkBreakdownElement Kind provides the capability to introduce user-defined kinds of Work Breakdown Elements.
- performerRole: PerformerRoleMap — This association defines the Roles that may provide information or otherwise contribute in completing the work. Performer Role Kind is used to classify the manner in which the role(s) contribute to the work being performed.
- ownedParameter: WorkBreakdown Element Parameter — Work Breakdown Elements can define an ordered set of parameters for input and output Work Product Use declarations.
- stateMachine: StateMachine — A Work Breakdown Element may define a state machine. For example an Activity may have the following states: Pending, Ready, In-progress, Bypassed and Completed. The state machine definition would depend on project characteristics and enactment environment. All work breakdown element instances share the same state machine.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
MandatoryInput	Association	spem2MandatoryInput	n/a
OptionalInput	Association	spem2OptionalInput	n/a
MandatoryOutput	Association	spem2MandatoryOutput	n/a
OptionalOutput	Association	spem2OptionalOutput	n/a

Changes from previous SPEM

Work Breakdown Element replaces WorkDefinition from SPEM 1.1. Work Breakdown Element is an abstract class to avoid confusion in SPEM 1.1 on when to use Activity and when to use Work Definition. User defined Work Breakdown Elements like Workstreams, Modules etc can be created when used with Work Breakdown Element Kind.

8.20 Work Breakdown Element Kind

Super Class

Process Element

Description

Work Breakdown Element Kind is a Process Element that defines a specific type for work breakdown elements.

Attributes

No additional attributes.

Associations

No additional attributes.

Semantics

Work Breakdown Element Kind provides the capability to introduce user-defined kinds of work breakdown elements.

Examples

If a process engineer would like to represent a special kind of work breakdown element he can define instances of work breakdown element kinds and assign these to his work breakdown elements. 'Phase' and 'Iteration' are popular examples for work breakdown element kinds. Another example can be found in Microsoft's Microsoft Solution Framework (MSF). The first level of MSF's breakdown structure is referred to as a 'Workstream'. The work breakdown element kind class allows defining and modeling such specific interpretation of the breakdown level.

Rationale

Apart from constraining the values of work breakdown element kinds to a known set of instances it also enables describing how a user-defined kind work breakdown element may be used and maintained. It is essential for keeping the meta-model flexible and inclusive of methodologies.

Changes from previous SPEM

Work Breakdown Element Kind is new in SPEM 2.0.

8.21 Performer Role Map

Super Class

Classifier

Description

A Performer Role Map declares a Work Breakdown Element's participating Roles; these are the Roles that are associated to the Work Breakdown Element with varying levels of participation.

Attributes

No additional attributes.

Associations

- performerKind: PerformerRoleKind — This association represents the kind of relationship a Role has with the Work Breakdown Element.

Examples

Microsoft's MSF for CMMI Process Improvement uses the following values for PerformerRoleKind: Responsible, Accountable, Consulted, Informed. The Create Product Requirements Workstream has the following performers:

Responsible:

- Business Analyst
- User Experience Architect

- Solution Architect

Accountable:

- Business Analyst

Consult:

- Subject Matter Expert
- Sponsor

Informed:

- IPM Officer
- Development Manager
- Project Manager
- Test Manager
- Release Manager

Changes from previous SPEM

Performer Role Map is new in SPEM 2.0.

8.22 Performer Role Kind

Super Class

Process Element

Description

Performer Role Kind defines the level of participation a role has when assigned to a work breakdown element.

Attributes

No additional attributes.

Associations

No additional associations.

Examples

See Performer Role Map.

Changes from previous SPEM

Performer Role Kind is new to SPEM 2.0

8.23 Work Breakdown Parameter

Super Class

Classifier

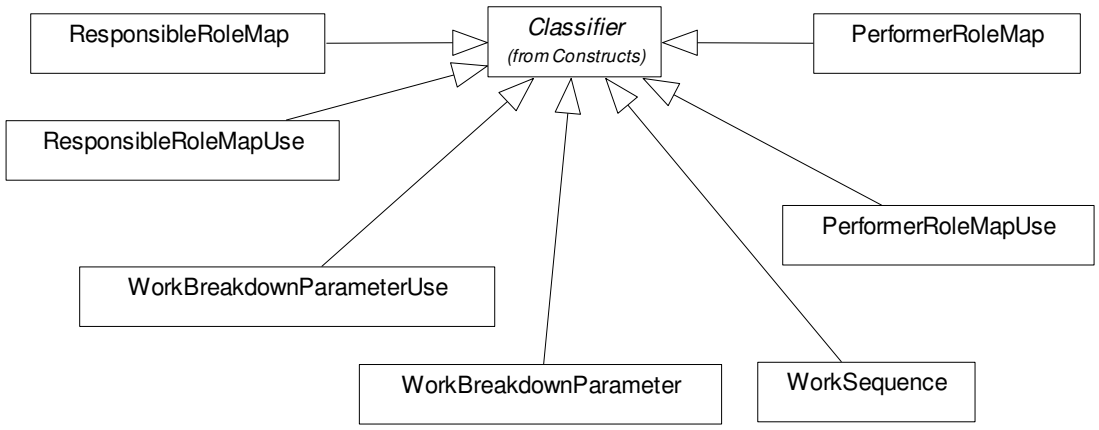

Figure 18 General Element Taxonomy

Description

A Work Breakdown Parameter declares the input/output associations between Work Breakdown Element and WorkProduct.

Attributes

- optionality:OptionalityType — This attribute represents the type of optionality as specified by the enumeration Optionality Type.
- direction: ParameterDirectionType — This attribute represents the direction type as specified by the enumeration Parameter Direction Type.

Associations

No additional associations.

Changes from previous SPEM

WorkBreakdown Parameter replaces Activity Parameter from SPEM 1.1.

8.24 Parameter Direction Type

Super Class

n/a: Enumeration

Description

This enumeration defines for Work Breakdown Element Parameter instances if the parameter represents an input, output, or input as well as output.

Enumeration Literals

- in — A Work Breakdown Element Parameter instance with this direction value represents an input.
- out — A Work Breakdown Element Parameter instance with this direction value represents an output.
- inout — A Work Breakdown Element Parameter instance with this direction value represents an input and output.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Work Breakdown Element Parameter (in)	Association	spem2Input	n/a
Work Breakdown Element Parameter (out)	Association	spem2Output	n/a
Work Breakdown Element Parameter (inout)	Association	spem2InOutput	n/a

8.25 Optionality Type

Super Class

n/a: Enumeration

Description

This enumeration provides the values for the Work Breakdown Element Parameter attribute optionality.

Enumeration Literals

- mandatory — It is mandatory to provide the Work Product specified in this parameter as input or to provide an instance of the Work Product as output respectively.
- optional — It is optional to provide the Work Product specified in this parameter as input or to provide an instance of the Work Product as output respectively.

8.26 Phase

Super Class

Work Breakdown Element

Description

Phase is a special Work Breakdown Element for which the default value for its attribute isRepeatable is 'False'. It has been included into the meta-model for convenience because it represents a very commonly used kind of Work Breakdown Element.

Attributes

No additional attributes.

Associations

No additional associations.

Semantics

Phase represent a significant period in a project, ending with major management checkpoint, milestone or set of Deliverables. It is included in the model as a predefined special Work Breakdown Element, because of its significance in defining breakdowns.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Phase	WorkBreakdownElement	spem2Phase	

Changes from previous SPEM

Phase is unchanged from SPEM 1.1 except it is represented as a work breakdown element for which its work breakdown element kind is set to "Phase".

8.27 Iteration

Super Class

Work Breakdown Element

Description

Iteration is a special Work Breakdown Element for which the default value for its attribute isRepeatable is 'True'. It has been included into the meta-model for convenience because it represents a very commonly used kind of Work Breakdown Element.

Attributes

No additional attributes.

Associations

No additional associations.

Semantics

Iteration groups a set of nested Activities that are repeated more than once. It represents an important structuring element to organize work in repetitive cycles.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Iteration	WorkBreakdownElement	spem2Iteration	

Examples

The concept of Iteration can be associated with different rules in different methodologies. For example, the IBM Rational Unified Process method framework (RUP) defines a rule that Iterations are not allowed to span across Phases. In contrast IBM Global Services Method (GS Method) based method frameworks this rule does not apply and Iteration can be defined which nest Phases. Rules like these, which play an important role for each individual method and are therefore not enforced by this meta-model. Instead, process authors are expected to follow and check these rules manually.

Changes from previous SPEM

Iteration is unchanged from SPEM 1.1 except it is represented as a work breakdown element for which its work breakdown element kind is set to "Iteration".

8.28 Activity

Super Class

Work Breakdown Element

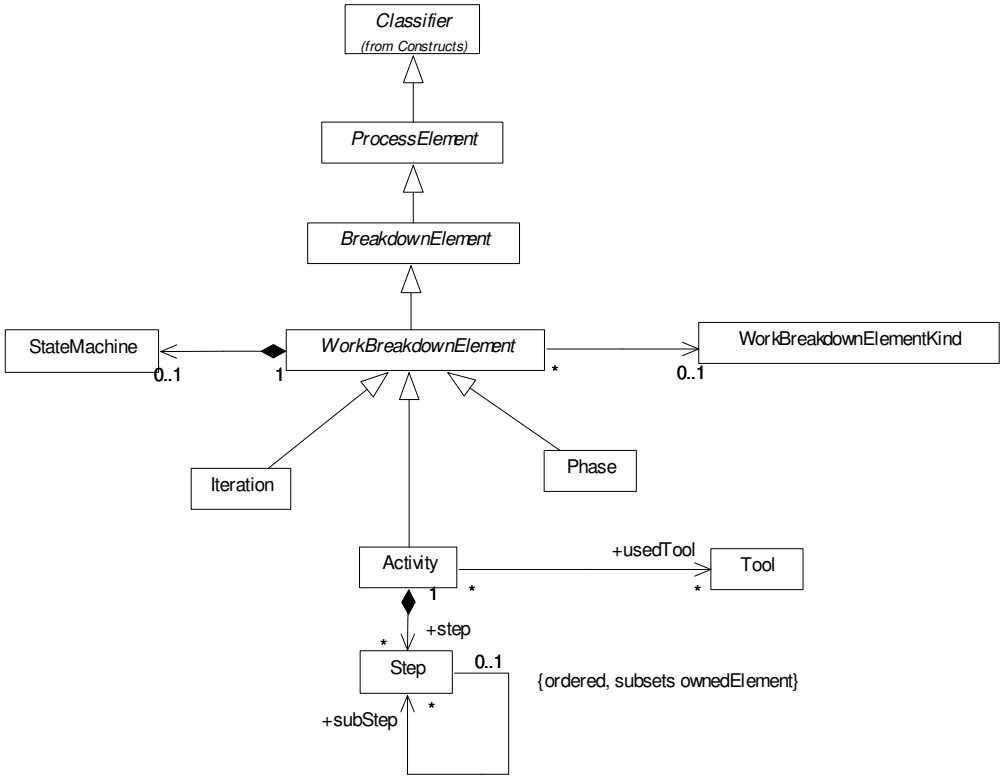

Figure 19 Activity taxonomy and key relationships

Description

An Activity is a concrete Work Breakdown Element that describes the work being performed by Roles. An Activity may consist of atomic elements called Steps.

Attributes

No additional attributes.

Associations

- ownedStep: Step — Steps to be performed for Activity.
- ownedParameter: WorkBreakdownParameter — Define associated input/output Work Product elements.
- usedTool: Tool — An Activity can recommend a specific set of tools to be used to support the Activity.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Activity	WorkBreakdownElement	spem2Activity	

Examples

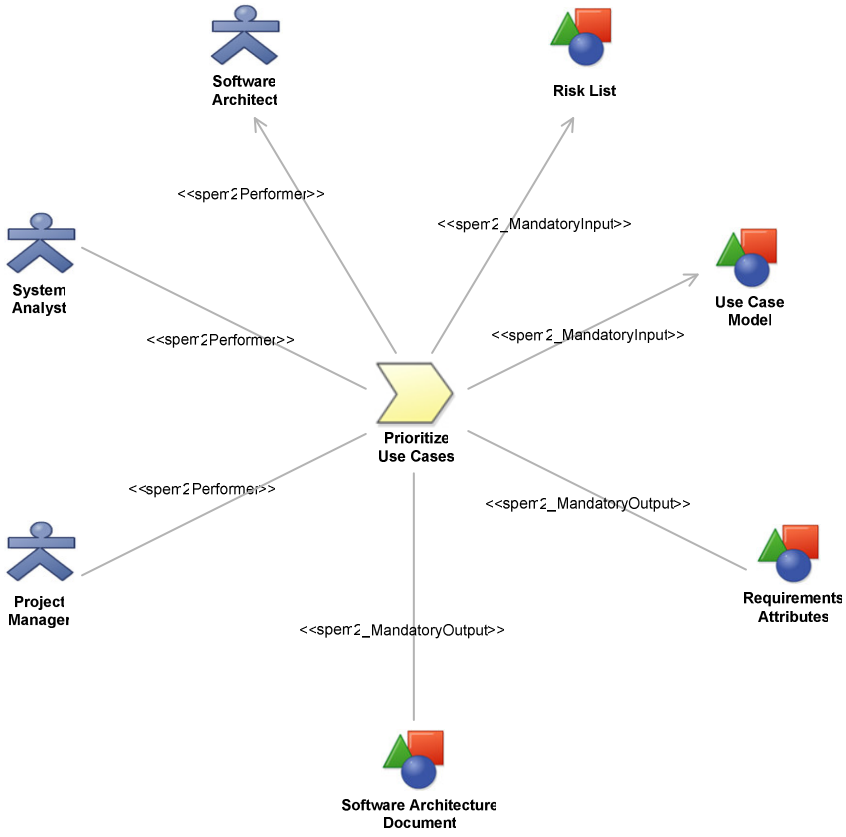

Figure 20 Activity with related content elements represented using the UML 2.0 SPEM 2.0 Profile.

Changes from previous SPEM

Activity in SPEM 2.0 can have nested steps and also be associated to tools.

8.29 Step

Super Class

Process Element

Description

A Step organizes an Activity's description into parts or sub-units of work. Steps can describe sub-steps nested into Steps.

Attributes

- Execution Type — This attribute represents the execution type as specified by the enumeration execution type.
- isOptional: Boolean — Indicates whether the step is optional or mandatory.

Associations

- subStep: Step — Steps can describe sub-steps nested into steps.

Semantics

A Step describes a meaningful and consistent part of the overall work described for an Activity. The collection of Steps defined for an Activity represents all the work that should be done to achieve the overall development goal of the Activity. Typical kinds of steps an Activity author should consider are:

- Thinking steps: where the individual roles understand the nature of the activity, gathers and examines the input workproducts, and understands the expected output workproducts.
- Performing steps: where the individual roles create or update some workproducts.
- Reviewing steps: where the individual roles inspects the results against some criteria.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Step	Process Element	spem2Step	

Changes from previous SPEM

Step is unchanged from SPEM 1.1 except that Steps may be nested.

8.30 Execution Type

Super Class

n/a: Enumeration

Description

This enumeration provides the values for the Step attribute Execution Type.

Enumeration Literals

- Machine execution — Execution may be performed by a system.
- Human execution — Execution requires human intervention.

8.31 Process Package

Super Class

Package

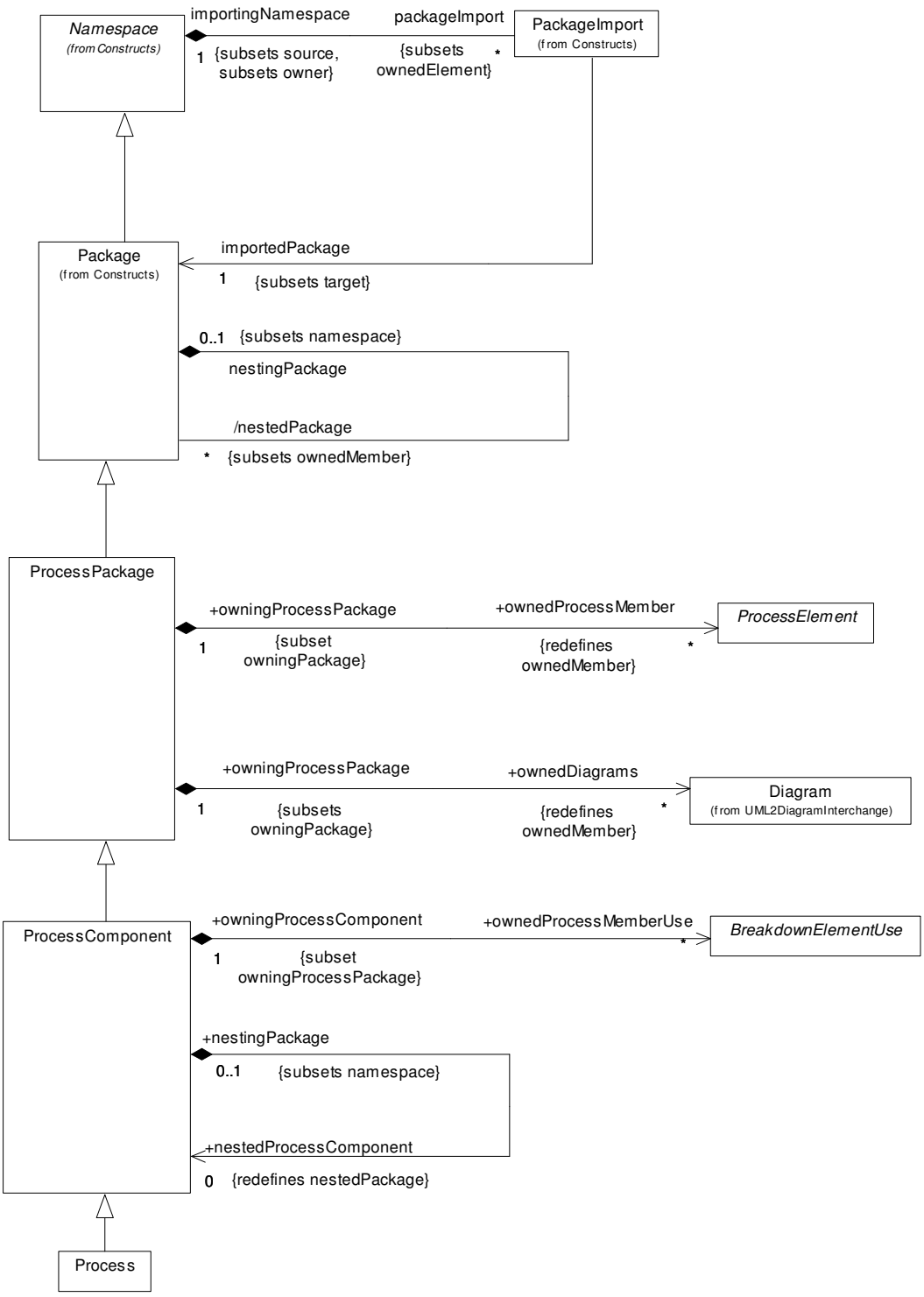

Figure 21 Process Package structure

Description

Process Package is a special Package that contains Process Elements and UML 2.0 Diagram Interchange elements. It redefines the owned Member association to allow elements of these two types. Process Package extends from the Package element from UML 2.0. It is a container that maintains base process elements used to define process structure. In addition, Process Package may also contain sub Process Packages, Process Components and Processes as it may be used to package an entire process library.

Attributes

- description: String — A Process Package can have a human readable description of the package. Normally this description is expected to be one or two paragraphs in length; however there is no restriction on size specified by this document.

Associations

- ownedProcessElements: ProcessElement — A Process Package can contain Process Elements which are used to define Processes. A Process Element instance can be part of only one Process Package instance.
- ownedDiagrams: Diagram — A Process Package can contain UML 2.0 Diagram Interchange diagrams which contain any number of diagram elements.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
ProcessPackage	Package	spem2ProcessPackage	

Changes from previous SPEM

Package is renamed to Process Package in SPEM 2.0.

8.32 Process Component

Super Class

Process Package

Description

A Process Component describes a reusable cluster of work breakdown elements. Process Components express and communicate process knowledge for a key area of interest such as a Discipline. They are also used as building blocks to assemble Processes or larger Process Components ensuring optimal reuse and application of the key practices they express.

A Process Component defines a set of Work Product Ports that define the inputs and outputs for a Process Component. There might be many components defining the same Work Product Ports, but using different workflows to achieve similar outputs for similar inputs. Acting as gray box reusable assets, Process Components can be utilized in Processes or be combined with other process components using the Work Product Ports with or without modifying the process workflow within the Process Component.

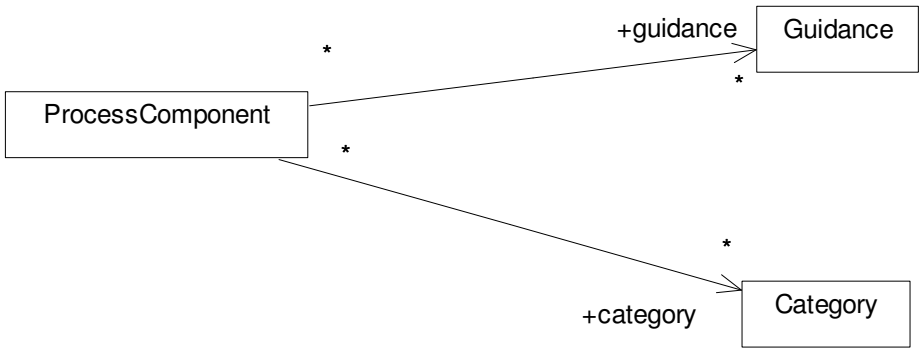

Figure 22 Process Component categories and guidance association

Attributes

No additional attributes

Associations

- ownedUseElement:BreakdownElementUse — This association defines a specialized Process Component's new Breakdown Element Uses and Work Sequences that contribute to the general Process Component.
- suppressed:BreakdownElementUse — This defines any use element that will be hidden from the process component workflow structure interpretation. In addition, a suppressed use element is not visible to any extending Process Components.
- ownedPort: WorkProductPort — This association defines the ports required or provided by the Process Component. They define work product types used, produced, or changed by the Process Component.
- guidance:Guidance — A Process Component can be related to many guidances process elements.
- category:Category — A Process Component can be categorized by many categories.

Semantics

Process Component Extension

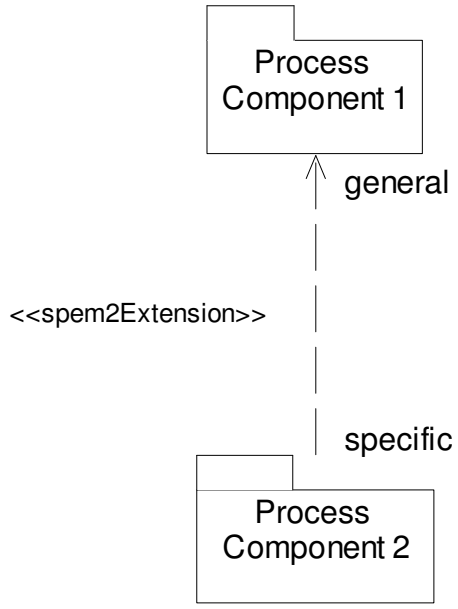

Figure 23 Process Component extension

Process Component work flow cluster definition may be inherited and extended by another Process Component through Extension much the same as extension between Breakdown Elements. The specific Process Component may extend the general process component workflow structure by defining new work sequence or breakdown use element instances within the workflow. The specific Process Component may also suppress inherited work sequence or breakdown use element instances. No modification is allowed to attribute values and association instances for inherited breakdown elements. However, a breakdown use element may have its base element redefined in the specific Process Component. New breakdown use elements defined within the specialized Process Component will contribute to the general Process Component work breakdown structure.

Specialized Process Components apply the following rules for attribute and association derivation:

attribute values	Values from the general Process Component are inherit and used to populate the specific Process Component. If a value is defined in the specific Process Component, then value from general Process Component is ignored.
association instances	Association instances defined in the general Process Component are inherited by the specific Process Component. The specific Process Component may add additional association instances or suppress inherited instances.

Process Components may also define a suppress association to its breakdown usage elements so they are not included in a consolidated view of the Process Component's workflow. The suppress association may reference to all usage element owned by existing Process Component. In addition, it may include all visible usage element reference from extending general Process Component. The inherited usage elements that have been suppressed will no longer be visible or used for all process components extending current component package. For all work breakdown use element, if it is defined as part of the suppress association, then all sub work breakdown use element associated with it are automatically set as suppressed.

A Process Component may not reuse itself or any process component that extends from it. Such a constraint is defined to avoid a recursive loop within the work break down structure hierarchy.

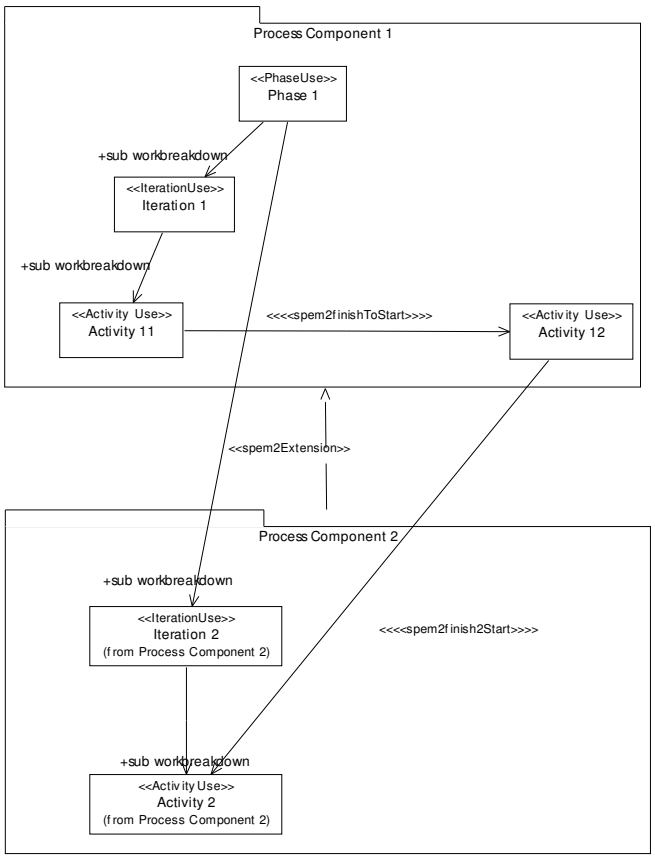

Figure 24 Definition View

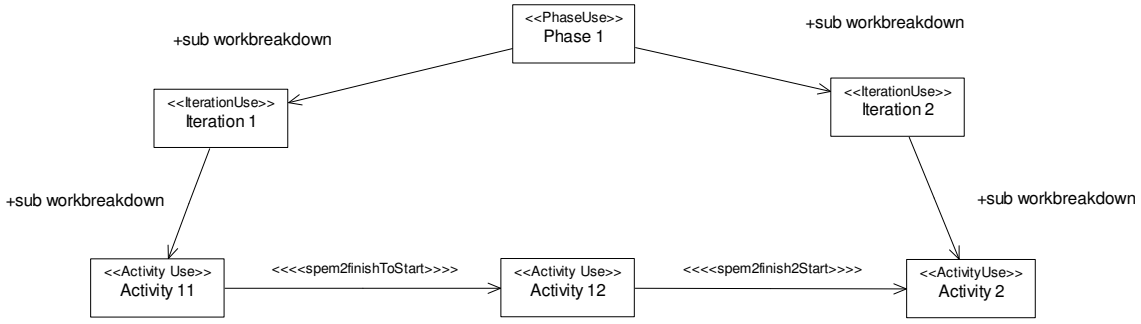

Figure 25 Consolidated Workflow View for Specific Process Component

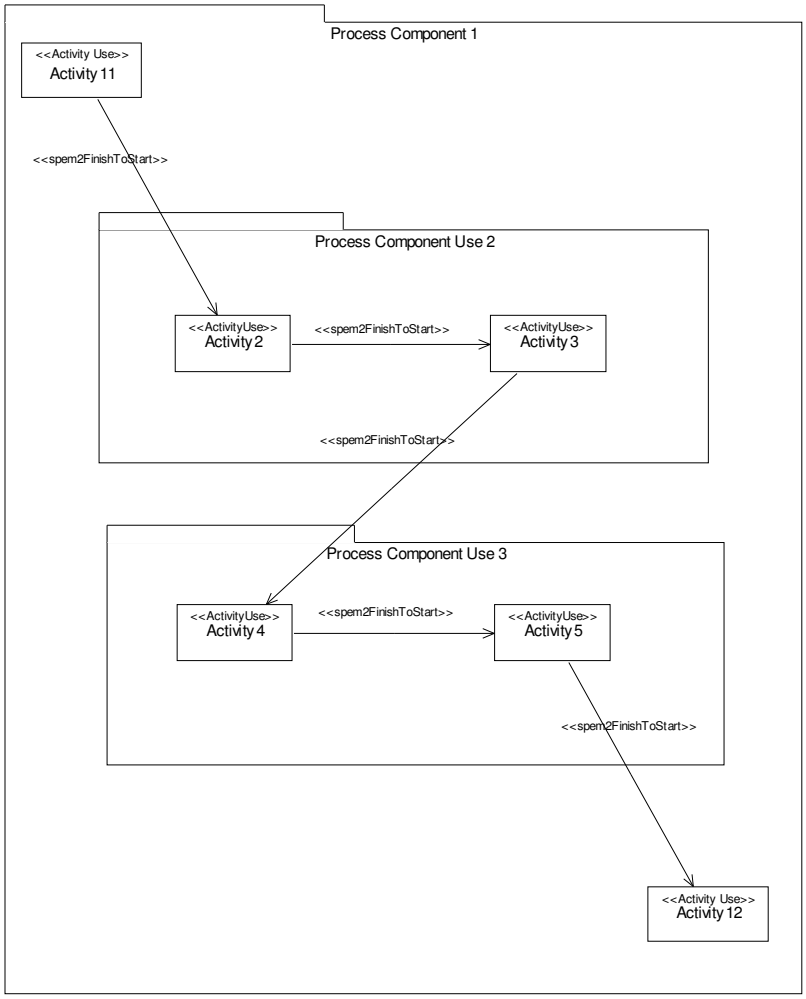

Figure 26 Definition View

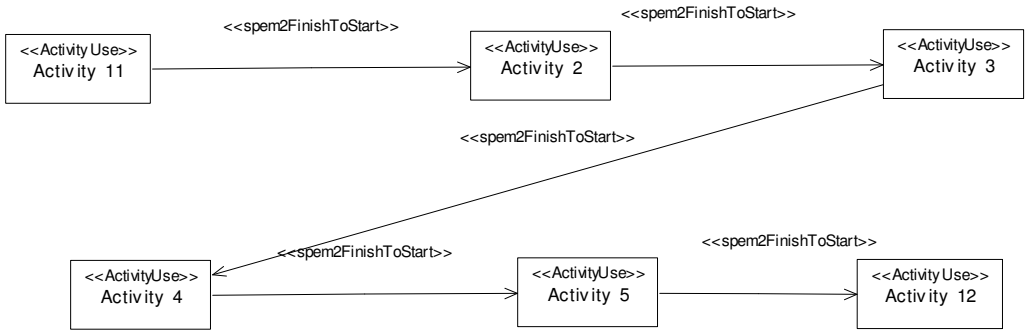

Figure 27 Consolidated View

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
ProcessComponent	Process Package	spem2ProcessComponent	

Examples

Examples of process components include "use case-based requirements management," "develop components," "validate build," or "ongoing management and support."

Rationale

Process Components provide a key process modeling mechanism to construct reusable building blocks of work breakdown structures.

Changes from previous SPEM

Process Component in SPEM 2.0 replaces the flawed self-containment constraints and unification mechanism of SPEM 1.1 with ports concepts.

8.33 Work Product Port

Super Class

Process Element

Description

A Work Product Port defines the work products input and outputs for a Process Component. It is defined based on exactly one type of Work Product and defines for exactly one Process Component if this Work Product is to be expected as a required (input) or supplied (output) by the Process Component. It also specifies if this input or output is optional or not.

Attributes

- portKind : WorkProductPortKind — This attribute defines if the port represents and input or output Work Product.
- isOptional: Boolean — This attribute specifies if the port represents a mandatory or optional Work Product input or output.

Associations

- portType: WorkProduct — This association defines the exact type of the Work Product Port.

Changes from previous SPEM

WorkProduct Port is new in SPEM 2.0.

8.34 WorkProduct Port Connector

Super Class

Process Element

Description

A WorkProduct Port Connector is used to connect Work Product Ports for assembling Process Components.

Changes from previous SPEM

WorkProduct Port Connector is new in SPEM 2.0.

8.35 Process

Super Class

Process Component

Description

A Process is a special Process Component intended to stand alone as a complete, end-to-end process. It is distinguished from normal process components by the fact that it is not intended to be composed with other components. In a tooling context, the instance of Process is the root of the process model, from which a tool can start to compute the transitive closure of an entire process.

Attributes

No additional attributes.

Associations

- includesProcessComponents: Process Component — Provides traceability for a Process document which Process Components have been used for defining the process.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
Process	Process Component	spem2Process	

Changes from previous SPEM

Process is unchanged from SPEM 1.1.

8.36 Breakdown Element Use

Super Class

Classifier

Description

A Breakdown Element Use is an abstract generalization that references a concrete Breakdown Element. A Breakdown Element Use provides a proxy-like representation of a Breakdown Element within process component workflows. A Breakdown Element Use can modify the Breakdown Element's structural relationships upon usage in a process component.

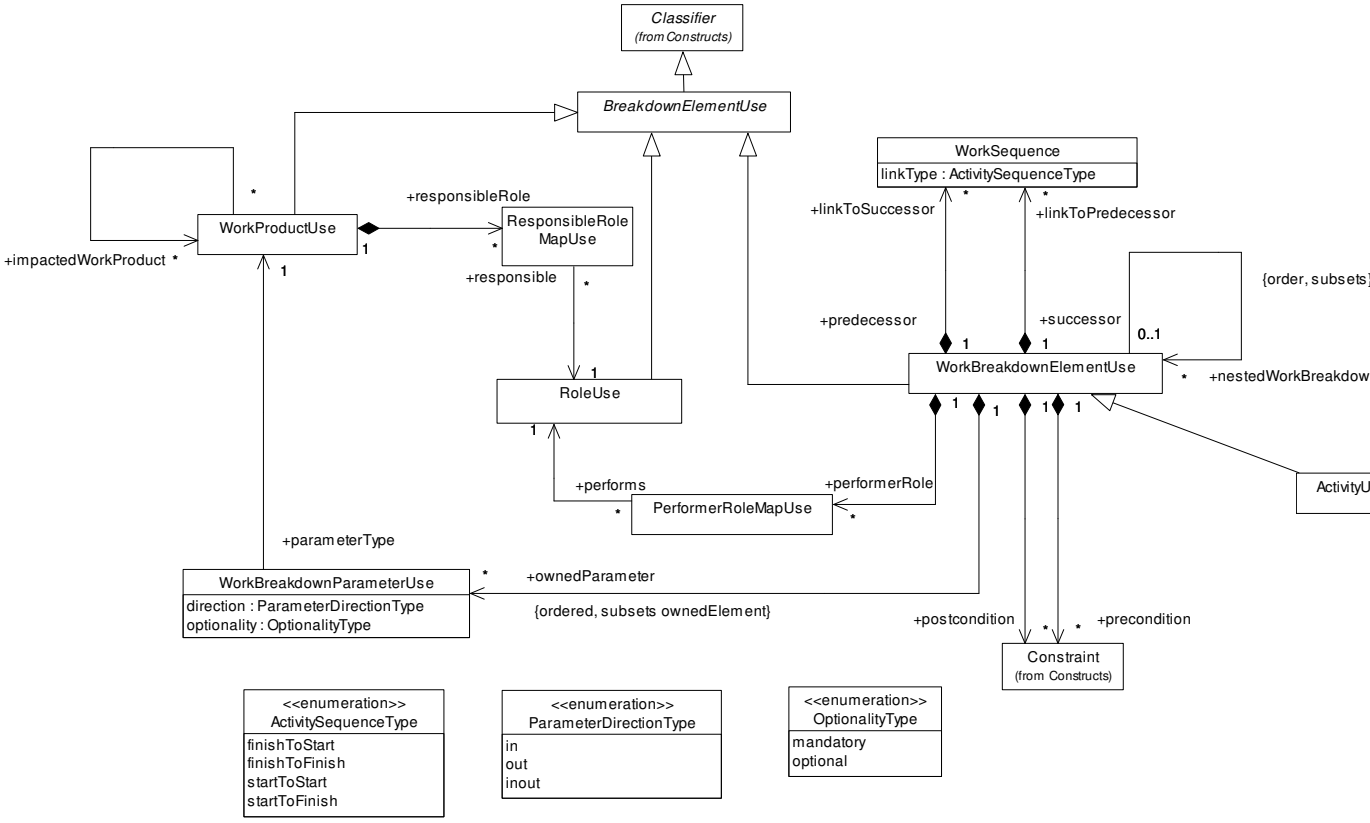

Figure 28 Use Elements

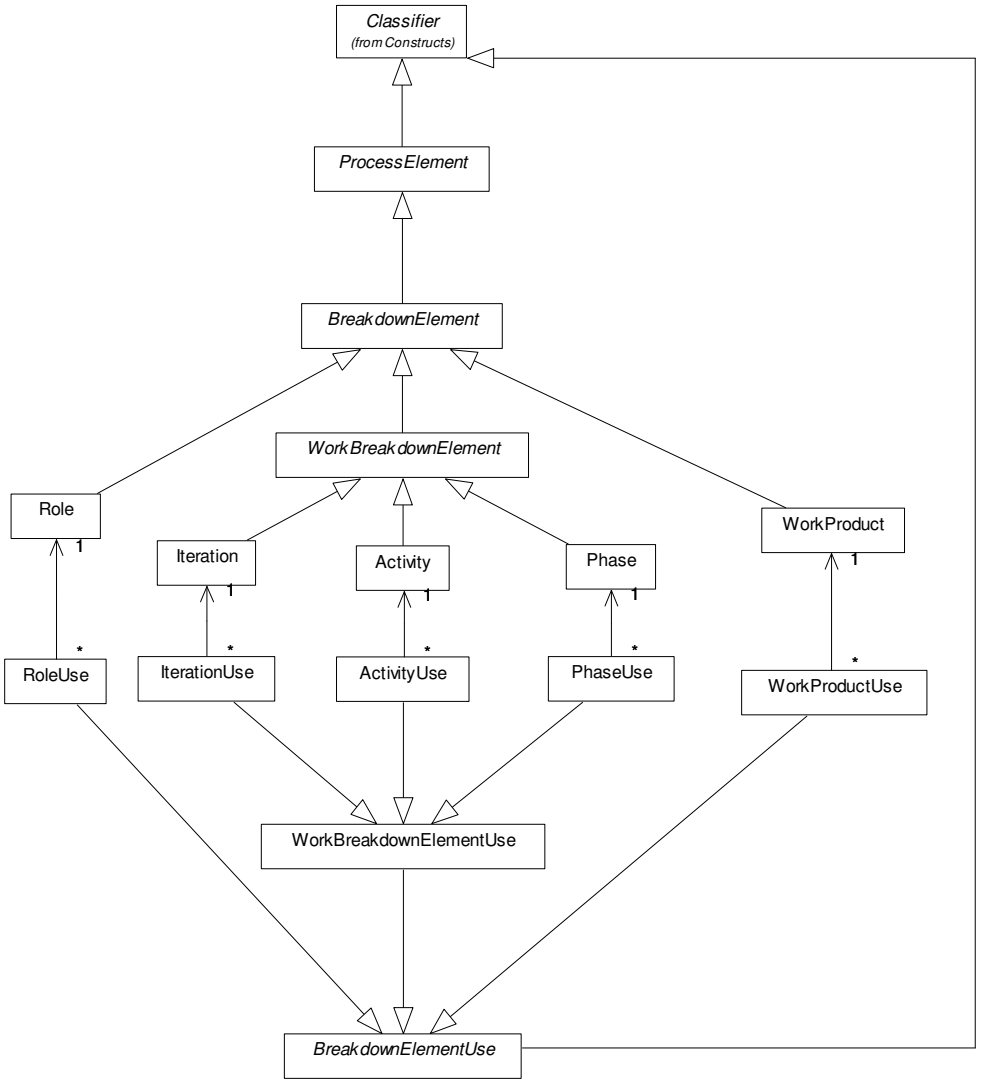

Figure 29 Key relationships of Breakdown

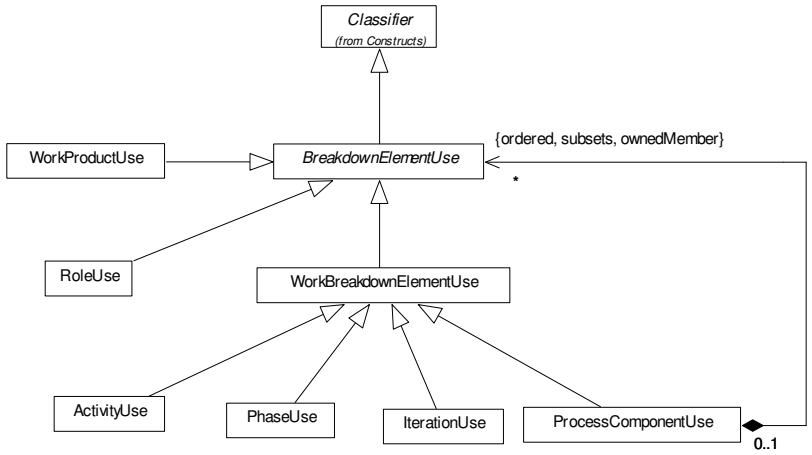

Figure 30 Breakdown Element Use structure

Attributes

- presentationName: String — Every Breakdown Element Use can maintain a presentation name which is externally visible/published name of the element, which might be localized.
- hasMultipleOccurrences: Boolean — This attribute maps to the respective attribute in Breakdown Element and is modifiable for the Use element.
- isOptional:Boolean — This attribute maps to the respective attribute in Breakdown Element and is modifiable for the Use element.

Associations

- baseElement: BreakdownElement — Each Breakdown Element Use references one Breakdown Element with the same element type. For example, Role Use can only reference Role.
- suppressed: BreakdownElementUse — Suppressed defines the breakdown element use associations that will be hidden from the process component workflow interpretation.

Semantics

Breakdown Element Use can be characterized as a reference or instance object to a Breakdown Element which has its own properties and associations. When a breakdown element use is created it shall be provided with attribute values and association instances defined for the referenced breakdown element. In other words, each association defined for a referenced breakdown element will be represented as associations for the breakdown element use in the process. Multiple breakdown element use elements may reference the same base breakdown element when constructing a process. Each breakdown element use instance can be part of only one process instance. Since sequencing breakdown elements is always in context of a process or process component, the pattern used here has separate classes for usage (Breakdown Element Use) and reusable definition (Breakdown Element). The meta-model supports tailoring at the use level to allow process engineers to customize a use instance as it is being applied in a process.

Examples

In this example from Macroscope, the “Define Owner Requirement” activity has an optional step “Define Owner Models” with circumstances of use presented in the process description.

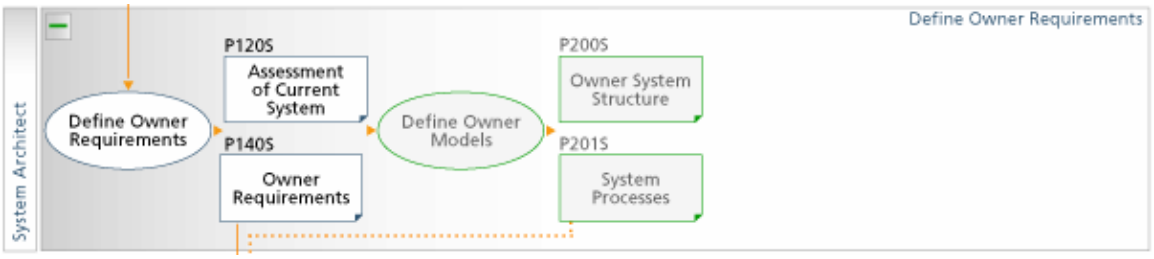

Figure 31 Optional work breakdown elements

Changes from previous SPEM

Breakdown Element Use is new in SPEM 2.0.

8.37 Work Product Use

Super Class

Breakdown Element Use

Description

A Work Product Use is a concrete Process Element Use object that represents an instance of a Work Product within a process or process component. A Work Product Use represents the occurrence of a real work product consumed, produced or modified when performing an activity when the process is instantiated.

Attributes

No additional attributes.

Associations

- baseElement: Work Product — This association represents the reference from Work Product Use to the Work Product it refers to. Every Work Product Use can reference only one WorkProduct. However, a WorkProduct can be represented by many Work Product Uses.
- subWorkProducts: WorkProductUse — This association maps to the respective association in Work Product and is modifiable for the Use element.
- workProductKind: WorkProductKind — This association maps to the respective association in Work Product and is modifiable for the Use element.
- stateMachine: StateMachine — This association maps to the respective association in Work Product and is not modifiable for the Use element.
- impactedWorkProduct: WorkProduct — This association represents an impacts dependency amongst Work Product Uses. It indicates that content or information of one referenced Work Product Use is being used to produce, refine, validate, etc. for the other referenced Work Product Use. For example, a link from one Work Product Use to another could indicates that one Work Product is required as input to an Activity that produces the related Work Product ('impacts' role end). It could also mean that a change on one Work Product instance may affect a change on the related product.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
WorkProductUse	Breakdown Element Use	spem2WorkProductUse	
subWorkProduct	Composite Aggregation	n/a	n/a
impactedWorkProduct	Dependency	spem2ImpactedWorkProduct	n/a
baseElement	Dependency	trace	n/a

Examples

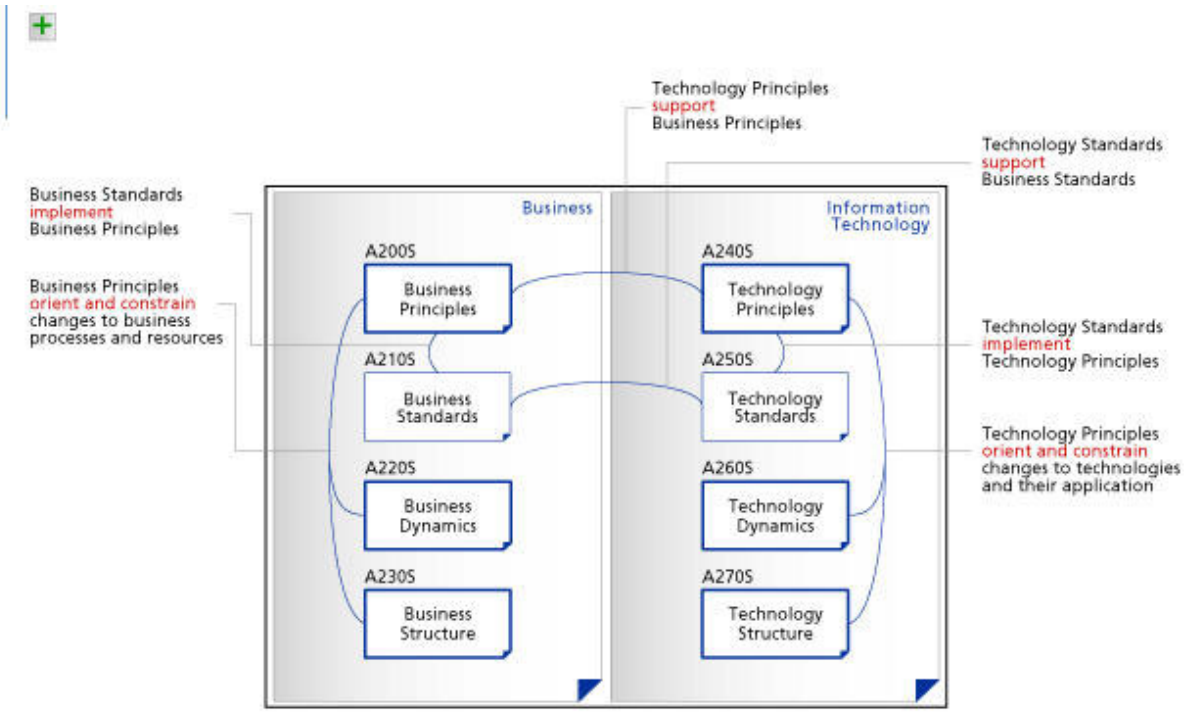

Figure 32 Macroscope example showing work product impact dependencies

Rationale

A WorkProduct definition can be reused as multiple usage instances. See Breakdown Element Use also.

Changes from previous SPEM

WorkProduct Use is new in SPEM 2.0.

8.38 Responsible Role Map Use

Super Class

Classifier

Description

A Responsible Role Map Use is an instance representation of a Responsible Role Map element in context of a role's association to work product.

Attributes

No additional attributes.

Associations

- responsibleKind: ResponsibleRoleKind — This association maps to the respective association in Responsible Role Map and is modifiable for the Use element.
- role: Role — This association maps to the respective association in Responsible Role Map and is not modifiable for the Use element.

Changes from previous SPEM

Responsible Role Map Use is new in SPEM 2.0.

8.39 Role Use

Super Class

Breakdown Element Use

Description

A Role Use is an instance representation of a role element. Each Role Use element may be defined as the performer role of a Work Breakdown Element Use or a responsible role of a Work Product Use.

Attributes

No additional attributes.

Associations

- baseElement: Role — This association represents the reference from Role Use to the Role it refers to. Every RoleUse can reference only one Role. However, a Role can be represented by many Role Uses.
- /responsible: WorkProduct — This association maps to the respective association in Role and is modifiable for the Use element.
- /performs: WorkBreakdownElement — This association maps to the respective association in Role and is modifiable for the Use element.
- roleKind: RoleKind — This association maps to the respective association in Role and is modifiable for the Use element.
- qualification: Qualification — This association maps to the respective association in Role and is modifiable for the Use element.
- team: Team — This association maps to the respective association in Role and is modifiable for the Use element.

Semantics

A RoleUse represents an instance object to a role without defining the specifics of the role itself in a process.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
RoleUse	Breakdown Element Use	spem2RoleUse	
Responsible	Association	spem2_Responsible	n/a
role	Dependency	trace	n/a

Rationale

A Role definition can be reused as multiple usage instances. See Breakdown Element Use also.

Changes from previous SPEM

Role Use is new in SPEM 2.0.

8.40 Work Breakdown Element Use

Super Class

Breakdown Element Use

Description

A Work Breakdown Element Use is an abstract generalization that references a concrete Work Breakdown Element. A Work Breakdown Element Use provides a proxy-like representation of a Work Breakdown Element within process component workflows. A Work Breakdown Element Use can modify the Work Breakdown Element's structural relationships upon usage in a process component. It uses work sequence and nested work breakdown elements to create the process component's work breakdown structure and workflow.

Attributes

- kind:WorkBreakdownElementKind — This attribute maps to the respective attribute in work breakdown element and is modifiable for the Use element.
- isRepeatable: Boolean — This attribute maps to the respective attribute in work breakdown element and is modifiable for the Use element.
- isOngoing: Boolean — This attribute maps to the respective attribute in work breakdown element and is modifiable for the Use element.
- isEventDriven: Boolean — This attribute maps to the respective attribute in work breakdown element and is modifiable for the Use element.
- hasMilestone: Boolean — This attribute maps to the respective attribute in work breakdown element and is modifiable for the Use element.
- milestoneDescription: String — This attribute maps to the respective attribute in work breakdown element and is modifiable for the Use element.

Associations

- nestedWork Breakdown Element: Work Breakdown ElementUse — This association represents work breakdown element structure nesting. It defines an n-level hierarchy of work breakdown element grouping.
- linkToPredecessor: WorkSequence — This association links a work breakdown element use to its predecessor. Every work breakdown element use can have predecessor information associated to it. Specific predecessor information is stored in instances of the class work sequence, which defines the kind of predecessor

another work breakdown element use represents for another work breakdown element use.

- linkToSuccessor: WorkSequence — This association links a work breakdown element use to its successor. Every work breakdown element use can have successor information associated to it. Specific successor information is stored in instances of the class work sequence, which defines the kind of successor another work breakdown element use represents for another work breakdown element use.
- performerRole: Role — This association maps to the respective association in work breakdown element and is modifiable for the Use element.
- ownedParameter: Work Breakdown Element Parameter Use — This association maps to the respective association in work breakdown element and is modifiable for the Use element.
- stateMachine:State Machine — This association maps to the respective association in work breakdown element and is not modifiable for the Use element.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
MandatoryInput	Association	spem2MandatoryInput	n/a
OptionalInput	Association	spem2OptionalInput	n/a
MandatoryOutput	Association	spem2MandatoryOutput	n/a
OptionalOutput	Association	spem2OptionalOutput	n/a

Changes from previous SPEM

Work Breakdown Element Use is new in SPEM 2.0.

8.41 Work Sequence

Super Class

Process Element

Description

Work Sequence is a Process Element that represents a relationship between two Work Breakdown Element use instances in which one Work Breakdown Element Use instance depends on the start or finish of the other Work Breakdown Element instance(s) in order to begin or end.

Attributes

- workSequenceType: WorkSequenceType — This attribute expresses the type of the Work Sequence relationship by assigning a value from the Work Sequence Type enumeration.

Associations

- successor: WorkBreakdownElementUse — This association links a Work Breakdown Element to its predecessor. Every Work Breakdown Element can have predecessor information associated to it
- predecessor: WorkBreakdownElementUse — This association links a Work Breakdown Element to its successor. Every Work Breakdown Element can have successor information

associated to it.

Semantics

The Work Sequence class defines predecessor and successor relations amongst Work Breakdown Elements.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
WorkSequence	Classifier	spem2WorkSequence	n/a

Changes from previous SPEM

Work Sequence replaces Precedes Dependency from SPEM 1.1. Made a bi-directional association for improved navigability and added the missing fourth type (start-finish).

8.42 Work Sequence Type

Super Class

n/a: Enumeration

Description

Work Sequence represents a relationship between two Work Breakdown Element Use instances in which one Work Breakdown Element Use (referred to as (B) below) depends on the start or finish of another Work Breakdown Element Use (referred to as (A) below) in order to begin or end. This enumeration defines the different types of Work Sequence relationships available in SPEM 2.0 and is used to provide values for Work Sequence's linkType attribute.

Enumeration Literals

- finishToStart — Work Breakdown Element Use (B) cannot start until Work Breakdown Element Use (A) finishes. For example, if you have two Work Breakdown Element Uses, "Construct fence" and "Paint fence," "Paint fence" can't start until "Construct fence" finishes. This is the most common type of dependency and the default for a new Work Sequence instance.
- finishToFinish — Breakdown Element Use (B) cannot finish until Work Breakdown Element Use (A) finishes. For example, if you have two Work Breakdown Element Uses, "Add wiring" and "Inspect electrical," "Inspect electrical" can't finish until "Add wiring" finishes.
- startToStart — Breakdown Element Use (B) cannot start until Work Breakdown Element Use (A) starts. For example, if you have two Work Breakdown Elements Uses, "Pour foundation" and "Level concrete," "Level concrete" can't begin until "Pour foundation" begins.
- startToFinish — Breakdown Element Use (B) cannot finish until Work Breakdown Element Use (A) starts. This dependency type can be used for just-in-time scheduling up to a milestone or the project finish date to minimize the risk of a Work Breakdown Element Use finishing late if its dependent Work Breakdown Element Uses slip. If a related Work Breakdown Element Use needs to finish before the milestone or project finish date, but it doesn't matter exactly when and you don't want a late finish to affect the just-in-time Work Breakdown Element Use, you can create an SF dependency between the Work Breakdown Element Use you want scheduled just in time (the predecessor) and its related Work Breakdown Element Use (the successor). Then if you update progress on the successor Work Breakdown Element Use, it won't affect the scheduled dates of the predecessor Work Breakdown Element Use.

8.43 Performer Role Map Use

Super Class

Classifier

Description

A Performer Role Map Use is an instance representation of a Performer Role Map element in context of role's associated Work Breakdown Element Use.

Attributes

No additional attributes.

Associations

- performerKind: PerformerRoleKind — This attribute maps to the respective association in Performer Role Kind and is modifiable for the Use element.

Changes from previous SPEM

Performer Role Map Use is new in SPEM 2.0.

8.44 Work Breakdown Parameter Use

Super Class

Classifier

Description

A Work Breakdown Parameter Use defines input/output association between Work Breakdown Element Use and WorkProduct Use. It is reference to Work Breakdown Parameter defined by the base Work Breakdown Element.

Attributes

- optionality:OptionalityType — This attribute maps to the respective attribute in work breakdown parameter and is modifiable for the Use element.
- direction: ParameterDirectionType — This attribute maps to the respective attribute in work breakdown parameter and is modifiable for the Use element.

Associations

- parameterType: WorkProductUse — This association links zero or one Work Product Use instances to a parameter.

Changes from previous SPEM

Work Breakdown Parameter Use is new in SPEM 2.0.

8.45 Phase Use

Super Class

Work Breakdown Element Use

Description

A Phase Use is an instance representation of a phase element in context of a process or process component. A Phase Use can form work breakdown structures by nesting and logical grouping of related sub Activities.

Attributes

No additional attributes.

Associations

No additional associations.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
PhaseUse	WorkBreakdownElementUse	spem2PhaseUse	
baseElement	Dependency	trace	n/a

Rationale

A Phase definition can be reused as multiple usage instances. See Work Breakdown Element Use also.

Changes from previous SPEM

Phase Use is new in SPEM 2.0.

8.46 Iteration Use

Super Class

Work Breakdown Element Use

Description

An Iteration Use is an instance representation of an iteration element in context of a process or process component. An Iteration Use can form work breakdown structures by nesting and logical grouping of related Work Breakdown Element Uses.

Attributes

No additional attributes.

Associations

No additional associations.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
IterationUse	WorkBreakdownElementUse	spem2IterationUse	
baseElement	Dependency	trace	n/a

Rationale

An Iteration definition can be reused as multiple usage instances. See Work Breakdown Element Use also.

Changes from previous SPEM

Iteration Use is new in SPEM 2.0.

8.47 Activity Use

Super Class

Work Breakdown Element Use

Description

An Activity Use is an instance representation of an Activity element in context of a process or process component. An Activity Use can form work breakdown structures by nesting and logical grouping of related sub Activities.

Attributes

No additional attributes.

Associations

- precondition: Constraint — This association adds an optional pre-condition to an Activity. A pre-condition defines any kind of constraint that must evaluate to true before the work described for the Activity can start. For example, a pre-condition could define that an input Work Product needs to be in a specific state or that other related work must be in a certain state (e.g. 'Input document X has been reviewed and signed by customer AND the work defined by Activity "Management Review" is complete') before the work can begin.
- postcondition: Constraint — This association adds an optional post-condition to an Activity. A post-condition defines any kind of constraint that must evaluate to true before the work described for the Activity can be declared completed or finished and which other Activities might depend upon (e.g. for their pre-conditions). For example, a post-condition could define that a work product defined to be the output must be in a specific state before the Activity can end (e.g. 'Use Case must be in state fully described and reviewed by System Analyst').
- ownedStep: Step — This association maps to the respective association in Activity and is modifiable for the Use element.
- ownedParameter: Work BreakdownParameterUse — This association maps to the respective association in Activity and is modifiable for the Use element.
- usedTool: Tool — This association maps to the respective association in Activity and is modifiable for the Use element.

Semantics

Activity Use represents a general unit of work assignable to specific performers represented by Role Use. An Activity can rely on inputs and produce outputs represented by Work Product Uses.

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
ActivityUse	WorkBreakdownElementUse	spem2ActivityUse	
baseElement	Dependency	trace	n/a

Rationale

An Activity definition can be reused as multiple usage instances. See Work Breakdown Element Use also.

Changes from previous SPEM

Activity Use is new in SPEM 2.0.

8.48 Process Component Use

Super Class

Work Breakdown Element Use

Description

A Process Component Use represents a Process Component application in any other Process or Process Component defined by a breakdown structure. In other words, it represents a reference of the Process Component from within another Process. The Process Component Use can define its own set of relationships such as it own predecessors and successors as well its own Ports that can be connected within the breakdown structure in which it is defined.

Attributes

- presentationName: String — Every Process Component Use can maintain a presentation name which is externally visible/published name of the process component, which might be localized.

Associations

- baseProcessComponent: ProcessComponent — This association represents the reference from the Process Component Use to the Process Component it refers to.
- suppressed:BreakdownElementUse — This defines any use element that will be hidden from the process component workflow structure interpretation.
- guidance:Guidance — This association maps to the respective association in Process Component and is modifiable for the Use element.
- category:Category — This association maps to the respective association in Process Component and is modifiable for the Use element.

Semantics

If there are three Process Components as shown below in Figure 33, when Component A and Component B are used within Component C, both Component A use and Component B use can extend their corresponding Process Components. However, when Component C is used in My Process Component C can extend its corresponding Process Component, but Component C > Component A use and Component C > Component B use cannot be further extended although they can be exchanged for another Process Component Use. Note that My Activity > Component A and My Activity > Component B can be extended from their corresponding Process Components.

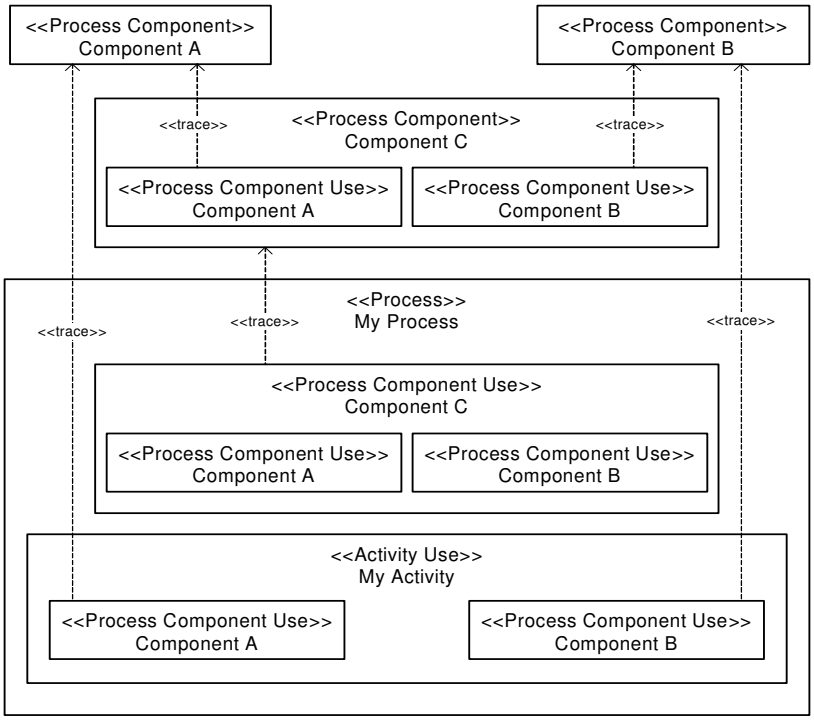

Figure 33 Process Component Use

SPEM 2.0 Profile Notation

Process Element	*Extended Meta-Class*	*Textual Stereotype*	*Graphical Stereotype*
ProcessComponentUse	WorkBreakdownElementUse	spem2ProcessComponentUse	

Changes from previous SPEM

Process Component Use is new in SPEM 2.0.

9 Process Enactment

The Process Enactment package introduces process element extensions to guide process engineers and project managers in tailoring and facilitating process enactment and improvement, but does not attempt to describe a full enactment system. Software development processes that are created using the Process Modeling package alone are not always sufficient for enactment. Project managers and practitioners typically have difficulty applying these processes to their daily activities, making process monitoring and improvement difficult. By allowing process engineers and project managers to decompose the process into discrete units of work that can be assigned to individuals and tracked based on the workflow and events defined in the Process Modeling package, this package enables the formal process to be abstracted from project practitioners.

Project enactment is typically done by creating and assigning a series of tasks and work items for project practitioners. Adding Tasks and Work Items to the meta-model and allowing them to be associated with Work Breakdown Elements allows the process to become transparent to the practitioner in an enactment system. The enactment system would help practitioners focus on their current assignments and avoid process information overload by filtering the process for practitioners, ensuring that they are provided only the information that they need when they need it. These Tasks and Work Items also facilitate data rollup and increased process feedback and improvement by allowing enactment data and metrics to be fed back to the process engineers and project managers during enactment.

An automated enactment system also needs to be able to interpret the workflow and events defined by process engineers and project managers to ensure timely information is relayed to project practitioners. A process engineer is not required to formally define the entire workflow sequence in order for a process to be enactable. An automated enactment system should be continuously listening for and enacting breakdown element state changes to help facilitate dynamic process execution. For example, the starting of one Activity may trigger the start or finish of another Activity. Alternatively, a state change of a WorkProduct may trigger a state change of another WorkProduct, Activity, or Task. An enactment system should also allow activities and decision nodes that have constraints (e.g. precondition or postcondition) defined to listen for events that would allow their constraints to evaluate to true. Process engineers may also decide to not have all state changes and constraints be automated. For those constraints that are not able to be evaluated by the system, an enactment system would still be able to notify a practitioner that their attention is required to help facilitate process enactment.

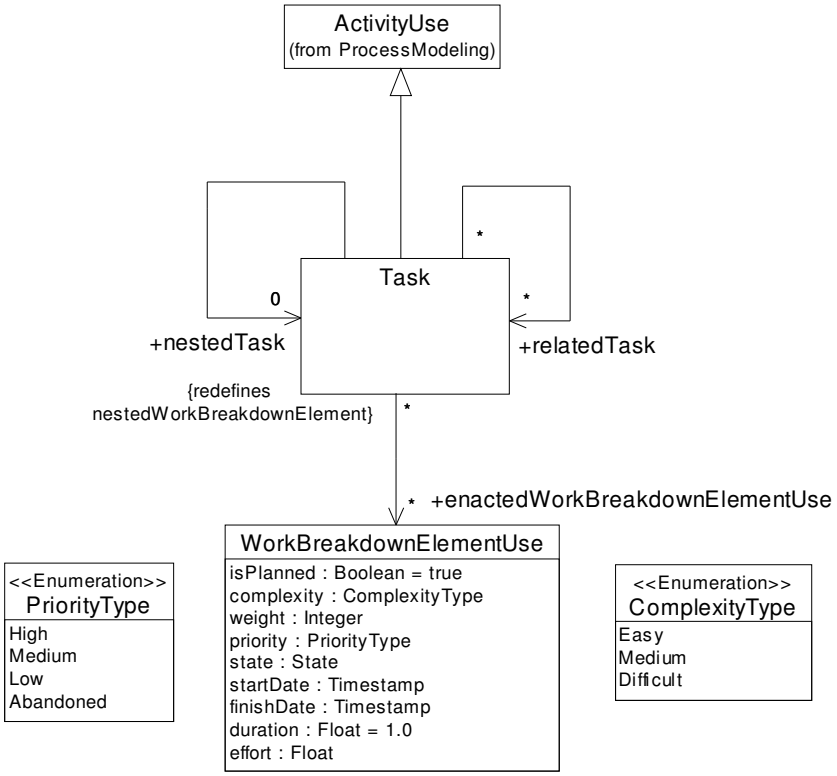

Figure 34 Task Activity Use

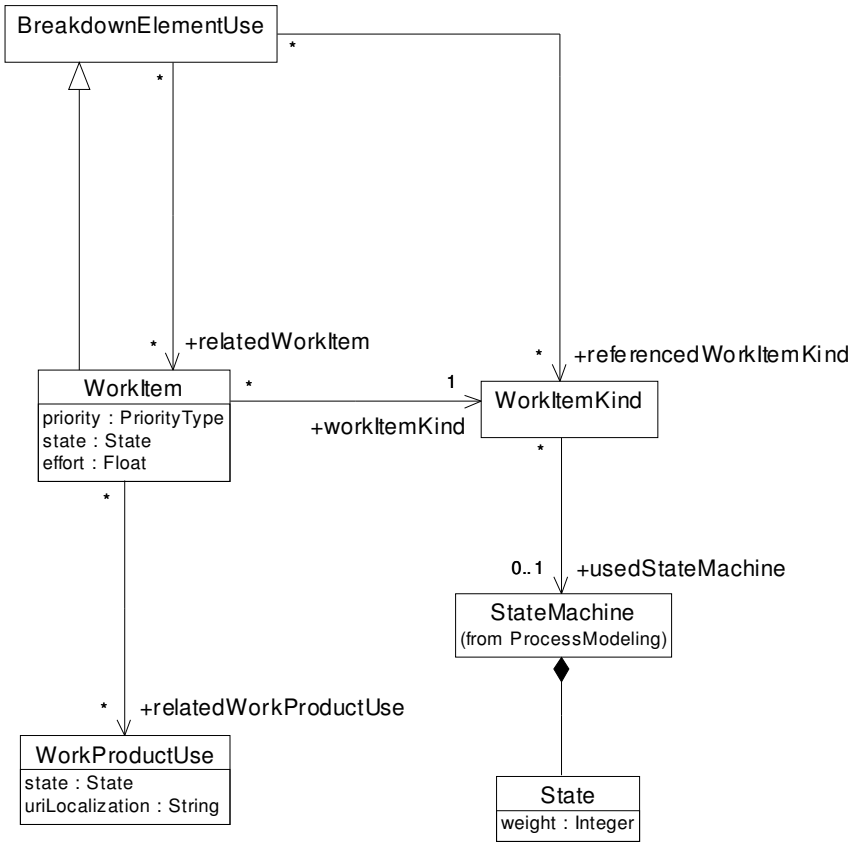

Figure 35 Work Item Breakdown Element Use

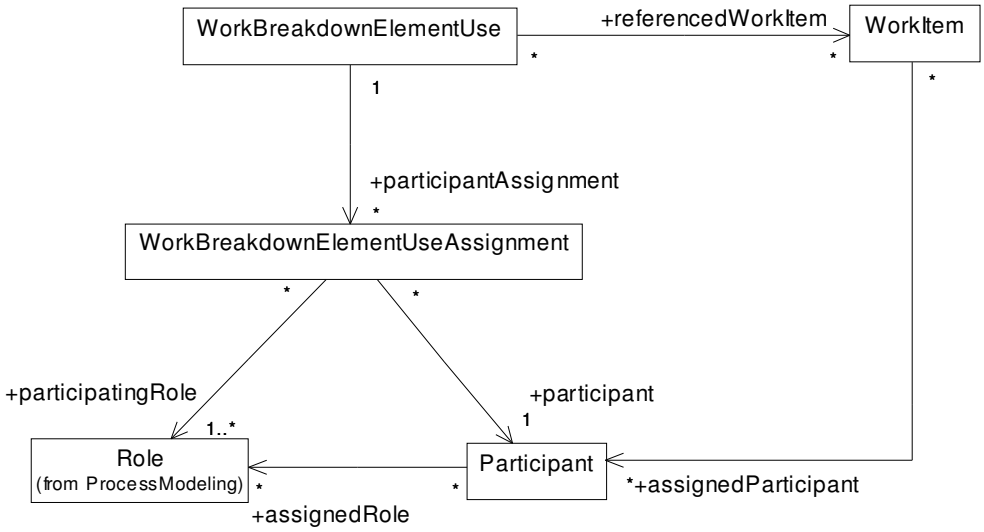

Figure 36 Participant Assignments

9.1 Work Breakdown Element Use

Super Class

Breakdown Element Use

Description

A Work Breakdown Element Use for enactment extends the modeling Breakdown Element Use by adding additional attributes and associations to capture enactment specific data and relationships for planning and reporting purposes.

Attributes

- isPlanned: Boolean

  Defines if the element is to be included in a project plan.

  When the isPlanned attribute is set to False for an instance of a Work Breakdown Element Use, then the element is not included when a project plan is generated from a work breakdown structure that contains the element.

- complexity: ComplexityKind

  Defines the complexity of the Work Breakdown Element Use to show its degree of difficulty.

  This attribute is useful for participant assignments during enactment. For example, an Activity that is of medium priority and high complexity would suggest more skilled people and less rigor during enactment.

- weight: Integer

  Defines the weighting of the Work Breakdown Element Use to show its representation of relative completion of its nesting Work Breakdown Element Use. For example, a weight of 10 can be used to indicate that the completion of the nested Work Breakdown Element Use represents 10% completion of its nesting Work Breakdown Element Use.

- priority: PriorityType

  Defines the priority of the Work Breakdown Element Use to show its importance relative to other Work Breakdown Element Uses in the process.

- state: State

  Defines the current state of the Work Breakdown Element Use.

  An enactment system would be able to determine the state of a Work Breakdown Element Use based on the actions and constraints defined for the Work Breakdown Element Use State Machine defined in the Process Modeling package.

- startDate: Timestamp

  Defines the start date for the Work Breakdown Element Use. If the Work Breakdown Element Use has started being enacted, then this date can be considered as the actual start date for the Work Breakdown Element Use, otherwise the date is considered as a planned date.

- finishDate: Timestamp

  Defines the finish date for the Work Breakdown Element Use. If the Work Breakdown Element Use has completed its enactment, then this date can be considered as the actual finish date for the Work Breakdown Element Use, otherwise the date is considered as a planned date.

- duration: Float

  Defines the duration of the Work Breakdown Element Use. Until the Work Breakdown Element Use has completed its enactment, duration could be an exact recommended duration, or an approximate relative duration. Duration can be used to derive a project timeline when generating a project plan if no startDate or finishDate have been defined.

- effort: Float — Defines the effort required to complete the Work Breakdown Element Use.

Associations

- participantAssignment: Work Breakdown Element Use Assignment — This association defines the Participants that are assigned to the Work Breakdown Element Use.

9.2 Task

Super Class

Activity Use

Description

Task is a special kind of Activity Use for enactment that does not have a Base Element or any nested Work Breakdown Element Uses.

Attributes

No additional attributes

Associations

- enactedWorkBreakdownElementUse: Work Breakdown Element Use — This association defines a link between the Task and the Work Breakdown Element Uses that it enacts.

  Creating a link between the Task and its Work Breakdown Element Uses allows a process engineer, project manager, or enactment system to provide participants with a concise view of the work that they are to perform.
- relatedTask: Task — This association links a Task and its related Tasks.

Rationale

Task is a special light-weight enactment Activity Use that allows process engineers and project managers to further define/tailor their work breakdown structures to meet their specific enactment needs and facilitate accurate data capture, process monitoring, and enactment. Tasks are created to guide and capture the real tasks that project participants perform during process enactment. When a Task is defined and associated with Work Breakdown Elements, the Task's definition is constructed such that it only contains the information necessary to perform the task reducing the effort required for participants to distill large amounts of process documentation and relate it to their daily tasks.

9.3 Breakdown Element Use

Super Class

Classifier

Description

A Breakdown Element Use for enactment extends the modeling Breakdown Element Use by adding additional associations to capture enactment specific relationships.

Attributes

No additional attributes.

Associations

- referencedWorkItemKind: WorkItemKind — This association defines a link between the Breakdown Element Use and the kinds of work items that are to be referenced during its enactment.
- relatedWorkItem:Work Item — This association links a Breakdown Element Use and its related Work Items.

9.4 Work Item

Super Class

Breakdown Element Use

Description

Work Item is a general purpose light-weight enactment Breakdown Element Use that does not have a Base Breakdown Element. It allows process engineers and project managers to identify and track work to be done during process enactment that does not justify the creation of Activities, Tasks, or Work Products. A Work Item instance has its state machine and custom properties dictated by its workItemKind.

Attributes

- priority: PriorityType — Defines the priority of the Work Item to show its importance relative to other Work Items.
- state: State — Defines the current state of the Work Item.
- effort: Float — Effort required to implement the Work Item.

Associations

- relatedWorkProductUse: Work Product Use — This association links a Work Item and its related Work Products.
- workItemKind: Work Item Kind — This association defines the Work Item's kind. The kind of work item dictates the State Machine and properties that are used by the Work Item (see Section XX, Work Item Kind).
- assignedParticipant: Participant — This association defines the Participants that are assigned to implement the Work Item.

Rationale

Work Items are flexible in that they can represent bugs, requirements, change requests, ideas, suggestions, risks, issues, etc. and can be created or assigned to one or more participants at any point in time. Instead of incorporating a full-fledged work item management system, an enactment system may choose to reference an external system for the management of such work items.

Examples

As bugs are discovered during enactment, a work item could be created as a bug in Bugzilla or Microsoft Visual Studio Team System and then later assigned to participants to fix and then test while being monitored by the enactment system.

9.5 Work Item Kind

Super Class

Process Element

Description

A Work Item Kind describes the type of a Work Item. When a Work Item is associated to a Work Item Kind, the Work Item Kind dictates its state machine and custom properties to the Work Item. This ensures that all Work Items of the same kind have the same state machine and share a common set of custom properties.

Attributes

No additional attributes.

Associations

- usedStateMachine: State Machine — This association defines the State Machine that is to be used by the Work Items of the Work Item Kind.

Examples

The Microsoft Solution Framework for CMMI Process Improvement has the following Work Item Types: Bug, Change Request, Issue, Requirement, Review, and Risk. Each Work Item Type has its own state machine and set of fields.

9.6 Work Product Use

Super Class

Breakdown Element Use

Description

A Work Product Use for enactment extends the modeling Work Product Use by adding additional attributes to aid enactment.

Attributes

- uriLocalization: String — Represents the location where the Work Product and its accompanying files can be located.
- state: State — Defines the current state of the Work Product.

Associations

No additional associations.

9.7 State

Super Class

State (from UML 2.0)

Description

A State for enactment extends the UML 2.0 State by adding an additional attribute to aid enactment.

Attributes

- weight: Integer — Defines the weighting of the State to show its representation of relative

completion of its associated element. For example, a Work Product state Draft may have a weight of 50 to indicate that a Work Product Use is considered 50% complete when its state is set to Draft. This attribute can be used by an enactment system to estimate the percentage completion of WorkProducts and their composite WorkProducts.

Associations

No additional associations.

9.8 Participant

Super Class

Classifier

Description

A Participant represents a person, system, or team that is assigned to a Work Breakdown Element Use or Work Item during process enactment.

Attributes

No additional attributes.

Associations

No additional attributes.

9.9 Work Breakdown Element Use Assignment

Super Class

Classifier

Description

Work Breakdown Element Use Assignment represents a relationship between a Work Breakdown Element Use and its assigned Participants.

Attributes

No additional attributes.

Associations

- participant: Participant — This association defines the link between the Work Breakdown Element Use and a Participant.
- participatingRole: Role — This association defines the Roles that the Participant is to perform during the enactment of the Work Breakdown Element Use.

10 Using SPEM 2.0 as a UML 2.0 Superstructure Profile

TBA

Appendices

Appendix A Migrating SPEM 1.1 Models to SPEM 2.0

SPEM version 2.0 is a major revision of SPEM version 1.1. Many issues and defects have been fixed in version 2, new capabilities have been added, as well as gaps and limitations of version 1.1 have been closed. SPEM 1.1 was based on UML 1.4 and SPEM 2.0 is now based on UML version 2.0, which represents a major revision of the UML.

The following table provides an overview to how the concrete SPEM 1.1 meta-classes defined in this Specification map to SPEM 2.0 classes. It lists the SPEM 1.1 concepts in the order as they have been defined in the SPEM 1.1 Specification.

SPEM 1.1	**SPEM 2.0**	**Comment**
External Description	<deprecated>	Description is now an attribute for Process Elements and Process Components.
Guidance	Guidance	No change from SPEM 1.1.
Guidance Kind	Guidance Kind	No change from SPEM 1.1.
Categorizes Dependency	<deprecated>	Made a direct association in SPEM 2.0 for simplicity.
Impacts Dependency	<deprecated>	Made a direct association in SPEM 2.0 for simplicity.
Import Dependency	<UML 2.0 Infrastructure> Element Import	Reused from UML 2.0 Infrastructure.
Precedes Dependency	Work Sequence	Made a bidirectional association for improved navigability. Added missing fourth type (start-finish).
Refers To Dependency	<deprecated>	This dependency required modeling redundant information in the process model which leads to maintenance problems.
Trace Dependency	<deprecated>	Deprecated because of lack of semantics and overlap with other concepts (such as Impacts).
Work Product	Work Product	No change from SPEM 1.1.
Work Product Kind	Work Product Kind	No change from SPEM 1.1.
Work Definition	WorkBreakdownElement	Made it abstract class to avoid the confusion of SPEM 1.1 of when to use Activity and when to use Work Definition. Introduced more subclasses to support commonly understood WorkBreakdownElements such as phase, iteration and activity.
Activity Parameter	Work Breakdown Parameter	Renamed Work Breakdown Parameter in SPEM2.0 because every Work Breakdown Element can have parameters not only activities.
Activity	Activity	No change from SPEM 1.1.
Step	Step	No change from SPEM 1.1.
Process Performer	<deprecated>	SPEM 2 does not separate the Process Performer and Process Role anymore, but allows assigning Roles to any Work Breakdown Element reducing the number of objects defined for processes as well as increasing reusability of these concepts. For modeling more general roles or role groupings, SPEM 2.0 introduces the concepts of Role Kinds

		and Team.
Process Role	Role	Renamed Role in SPEM 2.0.
Package	Process Package	Renamed Process Package in SPEM 2.0. Derived from the UML 2.0 package.
Process Component	Process Component	Removed flawed self-contained constraints and "Unification" mechanism of SPEM 1.1 and replaced with Ports as part of the component definition.
Process	Process	No change from SPEM 1.1.
Discipline	Category	Defining discipline as a specialization of Package in SPEM 1.1 created unnecessary restrictions on the way elements could be packaged. In SPEM 2 Disciplines can be represented as a category of kind discipline that allows any element in any package to be categorized to be part of a specific Discipline independent of its physical location.
Phase	Phase	No change from SPEM 1.1.
Lifecycle	<deprecated>	Each process would represent a complete end to end life cycle and it would be redundant to maintain lifecycle element.
Iteration	Iteration	No change from SPEM 1.1.
Precondition	Precondition	No change except for reusing UML 2.0 constraints.
Goal	Postcondition	Renamed Postcondition in SPEM2.0.

Appendix B SPEM 2.0 UML 2.0 Profile Summary

TBA

Appendix C Process Diagrams for SPEM 2.0

SPEM 2.0 utilizes the UML 2.0 Diagram Interchange. Diagrams defined in this format can be stored in a Process Package.

TBA

Appendix D Standard Guidance Kinds and Category Kinds

The following section provides some examples of Guidance Kinds and Category Kinds:

D.1 Standard Guidance Kinds

The following table provides the semantics for commonly used guidance kinds.

Guidance Kind	*Description*
Checklist	A Checklist is a specific type of guidance that identifies a series of items that need to be completed or verified. Checklists are often used in reviews such as walkthroughs or inspections.
Concept	A Concept is a specific type of guidance that outlines key ideas associated with basic principles underlying the referenced item. Concepts normally address more general topics than Guidelines and span across several work product and/or activities.
Estimate (metric kind)	An Estimate is a specific type of Guidance that provides sizing measures, or standards for sizing the work effort associated with performing a particular piece of work and instructions for their successful use. It may be comprised of estimation considerations and estimation metrics.
Estimation Considerations (metric kind)	Estimation Considerations qualify the usage and application of estimation metrics in the development of an actual estimate.
Estimating Metric (metric kind)	Estimation Metric describes a metric or measure that is associated with an element and which is used to calculate the size of the work effort as well as a range of potential labor.
Example	An Example is a specific type of Guidance that represents a typical, partially completed, sample instance of one or more work products or scenario like descriptions of how Task may be performed. Examples can be related to Work Products as well as Tasks that produce them as well as any other Content Element.
Guideline	A Guideline is a specific type of guidance that provides additional detail on how to perform a particular activity or grouping of activities (e.g. grouped together as iterations) or that provides additional detail, rules, and recommendations on work products and their properties. Amongst others, it can include details about best practices and different approaches for doing work, how to use particular types of work products, information on different subtypes and variants of the work product and how they evolve throughout a lifecycle, discussions on skills the performing roles should acquire or improve upon, measurements for progress and maturity, etc.
Practice	A Practice represents a proven way or strategy of doing work to achieve a goal that has a positive impact on work product or process quality. Practices are defined orthogonal to methods and processes. They could summarize aspects that impact many different parts of a method or specific processes. Examples for practices would be "Manage Risks", "Continuously verify quality", "Architecture-centric and component-based development", etc.

Guidance Kind	*Description*
Report	A Report is a predefined template of a result that is generated on the basis of other work products as an output from some form of tool automation. An example for a report would be a use case model survey, which is generated by extracting diagram information from a graphical model and textual information from documents and combines these two types of information into a report.
Reusable Asset	A Reusable Asset provides a solution to a problem for a given context. The asset may have a variability point, which is a location in the asset that may have a value provided or customized by the asset consumer. The asset has rules for usage which are the instructions describing how the asset should be used.
Roadmap	A Roadmap is a special Guidance Kind which is only related to Activates. A Roadmap represents a linear walkthrough of a Process. An instance of a Roadmap represents important documentation for the Activity or Process it is related to. Often a complex Activity such as a Process can be much easier understood by providing a walkthrough with a linear thread of a typical instantiation of this Activity. In addition to making the process practitioner understand how work in the process is being performed, a Roadmap provides additional information about how Activities and Tasks relate to each other over time. Roadmaps are also used to show how specific aspects are distributed over a whole process providing a kind of filter on the process for this information.
Supporting Material	Supporting Materials is catch all for other types of guidance not specifically defined elsewhere. It can be related to all kinds of Process Elements, i.e. including other guidance elements.
Template	A Template is a specific type of guidance that provides for a work product a predefined table of contents, sections, packages, and/or headings, a standardized format, as well as descriptions how the sections and packages are supposed to be used and completed. Templates cannot only be provided for documents, but also for conceptual models or physical data stores.
Term Definition	Term Definitions define concepts and are used to build up the Glossary. They are not directly related to Process Elements, but their relationship is being derived when the Term is used in the Process Elements description text.
Tool Mentor	A Tool Mentor is a specific type of guidance that shows how to use a specific tool to accomplish some piece of work a Work Product either in the context of or independent from an Activity.
Whitepaper	Whitepapers are a special Concept guidance that has been externally reviewed or published and can be read and understood in isolation of other process elements and guidance.

D.2 Standard Category Kind

The following table provides the semantics for commonly used category kinds.

Category Kind	*Description*
Discipline Grouping	Discipline Groupings are used to group Disciplines. For example, the Discipline Grouping "Software Disciplines" would be the group of all disciplines related to developing software such as “Requirements Management” or “Testing”; "IT Infrastructure Management" would be a Disciplines Grouping for disciplines such as “IT Operational Services”, “IT Customer Relationships”, or “IT Enabling Services”. Disciplines can be associated to more than one Discipline Grouping.
Discipline	A Discipline is a categorization of work based upon similarity of concerns and cooperation of work effort. A discipline is a collection of Activities that are related to a major 'area of concern' within the overall project. The grouping of Activities into disciplines is mainly an aid to understanding the project from a 'traditional' waterfall perspective. Typically, for example, it is more common to perform certain requirements activities in close coordination with analysis and design activities. Separating these activities into separate disciplines makes the activities easier to comprehend.
Role Set Grouping	Role Sets can be categorized into Role Set Groupings. For example, different methods might define similar Role Sets which may need to be distinguished from each other on a global scale. Thus, Role Set Groupings allow distinguishing, for example, Software Services Manager Role Sets from Software Development Organization Manager Role Sets.
Role Set	A Role Set organizes Roles into categories. Role Set is used to group roles together that have certain commonalities. For example, the “Analysts” Role Set could group the “Business Process Analyst”, “System Analyst”, as well as “Requirements Specifier” roles. All of these work with similar techniques and have overlapping skills, but may required as distinct roles for activities.
Domain	Domain is a refineable hierarchy grouping related work products. In other words, Domains can be further divided into sub-domains, with work product elements to be categorized only at the leaf-level of this hierarchy.
Work Product Type	Work Product Type is a second category for work products, which in contrast to Domain is more presentation oriented. A work product can have many Work Product Types. An example for one Work Product Type is “Class Diagram”, which categorizes the Artifacts Analysis Model, Design Model, and User Experience Model. Another example is “Specification”, which categorizes requirements specifications that define a system with a well-defined system boundary, such as use case or functional requirements specification. A Work Product can be categorized as many Work Product Types. For example, a use case model can be categorized as a Specification as well as Diagram Work Product Type.
Tool Category	A Tool Category is a container/aggregate for Tool Mentors. It can also provide general descriptions of the tool and its general capabilities.