BS ISO/IEC 24744:2014

BSI Standards Publication

Software engineering — Metamodel for development methodologies

bsi.

...making excellence a habit.™

Document Sharing Hub

National foreword

This British Standard is the UK implementation of ISO/IEC 24744:2014. It supersedes BS ISO/IEC 24744:2007+A1:2010 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/15, Software and systems engineering.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2014. Published by BSI Standards Limited 2014

ISBN 978 0 580 82003 8

ICS 35.080

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 November 2014.

Amendments issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24744:2014

BS ISO/IEC 24744:2014

INTERNATIONAL STANDARD

ISO/IEC 24744

Second edition
2014-11-15

Software engineering — Metamodel for development methodologies

Ingénierie du logiciel — Métamodèle pour les méthodologies de développement

Reference number
ISO/IEC 24744:2014(E)

Documents
Document Sharing Hub

© ISO/IEC 2014

Full document details: BS ISO/IEC 24744:2014

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2014

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24744:2014

Contents

Page

Foreword v

Introduction vi

1 Scope 1

2 Conformance 1

3 Terms and definitions 1

4 Naming, diagramming and definition conventions, and abbreviated terms 3
4.1 Naming, diagramming and definition conventions 3
4.2 Abbreviations 4

5 Basic Concepts 4
5.1 Method Engineering 5
5.2 Dual-Layer Modelling 5
5.3 Powertypes and Clabjects 5
5.4 Uniting Process and Product 6
5.5 Process Assessment 6

6 Introduction to the SEMDM 7
6.1 Highly Abstract View 7
6.2 Abstract View and Core Classes 7
6.3 Process Classes 8
6.4 Producer Classes 10
6.5 Product Classes 11
6.6 Connection between Process and Product 12
6.7 Support Classes 13

7 Metamodel Elements 14
7.1 Classes 14
7.2 Enumerated Types 61

8 Using the Metamodel 62
8.1 Usage Rules 62
8.2 Usage Guidelines 63

9 Extending the Metamodel 64
9.1 Extension Rules 64
9.2 Extension Guidelines 65

Annex A (informative) Worked Example 66
A.1 SimpleMethod Description 66
A.2 Construction of Process Components 66
A.3 Construction of Producer Components 68
A.4 Construction of Product Components 68
A.5 Connection Between Process and Product Components 70

Annex B (informative) Mappings to Other Metamodelling Approaches 72
B.1 OMG SPEM 1.1 72
B.2 OOSPICE 73
B.3 OPEN 73
B.4 LiveNet 74
B.5 ISO/IEC 12207 and 15288 74
B.6 ISO/IEC 15504 (SPICE) 75
B.7 ISO/IEC 19501 (UML 1.4.2) 75

Annex C (informative) Graphical Notation 76
C.1 Introduction 76

Documents
Document Sharing Hub

© ISO/IEC 2014 – All rights reserved

Full document details: BS ISO/IEC 24744:2014

C.2 Notation Elements **77**
C.3 Diagram Types **88**
C.4 Abbreviation Tables **94**

Bibliography **96**

Table of Figures

Figure 1 – The three areas of expertise, or domains, which act as a context for SEMDM. Arrows mean "is represented by". 4

Figure 2 – Example of a powertype pattern and clabject. The Document class is partitioned by the DocumentKind powertype. The RequirementsSpecificationDocument class plus the rsd object represent a particular kind of document, making up a clabject. The rsd1 object represents a particular requirements specification document. 6

Figure 3 – Highly abstract view of the SEMDM 7

Figure 4 – Abstract view of the SEMDM, showing the core classes in the metamodel 8

Figure 5 – Work units 9

Figure 6 – Stages 10

Figure 7 – Producers 11

Figure 8 – Work product and modelling classes 12

Figure 9 – Actions and constraints 13

Figure 10 – Support classes 13

Figure C.1 – A lifecycle diagram showing the temporal structure of a complete method 89

Figure C.2 – A lifecycle diagram showing the content structure as well as the temporal structure of a method 90

Figure C.3 – An enactment diagram for the “Construction” phase kind of Figure C.2 91

Figure C.4 – A dependency diagram based on a refinement of Figure C.2 92

Figure C.5 – A process diagram showing the details of the “Requirements Engineering” and “Requirements Quality Assurance” processes 93

Figure C.6 – An action diagram showing the interaction between some task kinds pertaining to the “Requirements Engineering” and “Requirements Quality Assurance” processes and some related document kinds 94

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24744:2014

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 7, *Systems and Software engineering*.

This second edition cancels and replaces the first edition (ISO/IEC 24744:2007), which has been technically revised. It also incorporates the Amendment ISO/IEC 24744:2007/Amd.1:2010.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

Introduction

Development methodologies may be described in the context of an underpinning metamodel, but the precise mechanisms that permit them to be defined in terms of their metamodels are usually difficult to explain and do not cover all needs. For example, it is difficult to devise a practice that allows the definition of properties of the elements that compose the methodology and, at the same time, of the entities (such as work products) created when the methodology is applied. This document introduces, as a (potential) standard, the Software Engineering Metamodel for Development Methodologies, a comprehensive metamodel that makes use of a new approach to defining methodologies based on the concept of powertype. The SEMDM is aimed to the definition of methodologies in information-based domains, i.e. areas characterized by their intensive reliance on information management and processing, such as software, business or systems engineering. The SEMDM combines key advantages of other metamodelling approaches with none of their known drawbacks, allowing the seamless integration of process, modelling and people aspects of methodologies. Refer to Annex B where other metamodels are mapped to SEMDM and a brief synopsis of problems is provided.

Various methodologies are defined, used, or implied by a growing number of standards and it is desirable that the concepts used by each methodology be harmonized. A vehicle for harmonization is the SEMDM. Conformance to this metamodel will ensure a consistent approach to defining each methodology with consistent concepts and terminology.

This document also presents a proposed notation for the ISO/IEC 24744 standard metamodel. The notation presented here is mainly graphical and supports most of the classes found in ISO/IEC 24744.

Purpose

The SEMDM follows an approach that is minimalist in depth but very rich in width (encompassing domains that are seldom addressed by a single approach). It therefore includes only those higher-level concepts truly generic across a wide range of application areas and at a higher level of abstraction than other extant metamodels. The major aim of the SEMDM is to deliver a highly generic metamodel that does not unnecessarily constrain the resulting methodologies, while providing for the creation of rich and expressive instances.

In order to achieve this objective, the SEMDM incorporates ideas from several metamodel approaches plus some results of recent research (see [4, 5, 6, 7, 9, 10, 17] for details). This will facilitate:

- The communication between method engineers, and between method engineers and users of methodology (i.e. developers).
- The assembly of methodologies from pre-existing repositories of method fragments.
- The creation of methodology metamodels by extending the standard metamodel via the extension mechanisms provided to this effect.
- The comparison and integration of methodologies and associated metamodels.
- The interoperability of modelling and methodology support tools.

The relation of SEMDM to some existing methodologies and metamodels is illustrated in Annex B.

Audience

Since many classes in the SEMDM represent the endeavour domain (as opposed to the methodology domain), it might look like developers enacting the methodology would be direct users of the metamodel. This is not true. Classes in the SEMDM that model endeavour-level elements serve for the method engineer to establish the structure and behaviour of the endeavour domain, and are not used directly during enactment. Only

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

methodology elements, i.e. classes and objects created by the method engineer from the metamodel, are used by developers at the endeavour level, thus supporting both the creation of “packaged” methodologies as well as tailored, project-specific methodologies.

Here the term “method engineer” refers collectively to either a person constructing a methodology on site for a particular purpose or a person creating a "packaged" methodology as a "shrink-wrapped" process product.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

Documents
Document Sharing Hub

Software engineering — Metamodel for development methodologies

1 Scope

This International Standard defines the Software Engineering Metamodel for Development Methodologies (SEMDM), which establishes a formal framework for the definition and extension of development methodologies for information-based domains (IBD), such as software, business or systems, including three major aspects: the process to follow, the products to use and generate, and the people and tools involved.

This metamodel can serve as a formal basis for the definition and extension of any IBD development methodology and of any associated metamodel, and will be typically used by method engineers while undertaking such definition and extension tasks.

The metamodel does not rely upon nor dictate any particular approach to IBD development and is, in fact, sufficiently generic to accommodate any specific approach such as object-orientation, agent-orientation, component-based development, etc.

2 Conformance

A metamodel is defined in accordance with this International Standard if it:

a. Describes the scope of the concepts in the metamodel in relation to the scope of the elements defined in clause 7.

b. Defines the mapping between the concepts that are addressed in the metamodel, and that are within the scope of this International Standard, and the corresponding elements of this Standard (i.e. the elements of the standard cannot be substituted by others of identical intent but different construction).

A development methodology is defined in accordance with this International Standard if it is generated from a conformant metamodel as defined in the first paragraph of this clause (2 Conformance).

A development or engineering tool is developed in accordance with this International Standard if it implements a conformant metamodel as defined in the first paragraph of this clause (2 Conformance). If the purpose of the tool involves the creation of methodologies, then it is developed in accordance with this Standard if it also implements the necessary features so as to make the mechanisms described in sub-clause 8.1 available to the tool's users. If the purpose of the tool involves the extension of the metamodel, then it is developed in accordance with this Standard if it also implements the necessary features so as to make the mechanisms described in sub-clause 9.1 available to the tool's users.

NOTE 1 The metamodel thus defined does not necessarily have to include all the elements defined in clause 7 – only those that are relevant to the purpose of the said metamodel are required.

NOTE 2 Conformance for methodologies or conformance for tools can be established without any necessity of explicitly including the detailed metamodel for any relevant work product kind or model unit kind. It is adequate to define the mappings of any such work products to the WorkProductKind and ModelUnitKind classes of the SEMDM.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

NOTE This Standard uses a self-consistent set of core concepts that is as compatible as possible with other standards (such as ISO/IEC 12207, ISO/IEC 15504, etc.).

Documents
Document Sharing Hub

© ISO/IEC 2014 – All rights reserved

Full document details: BS ISO/IEC 24744:2014

3.1
information-based domain
realm of activity for which information is the most valuable asset

Note 1 to entry: This means that information creation, manipulation and dissemination are the most important activities within information-based domains. Typical information-based domains are software and systems engineering, business process reengineering and knowledge management.

3.2
methodology
method
specification of the process to follow together with the work products to be used and generated, plus the consideration of the people and tools involved, during an IBD development effort

Note 1 to entry: A methodology specifies the process to be executed, usually as a set of related activities, tasks and/or techniques, together with what work products must be manipulated (created, used or changed) at each moment and by whom, possibly including models, documents and other inputs and outputs. In turn, specifying the models that must be dealt with implies defining the basic building blocks that should be used to construct these models.

Note 2 to entry: The term “methodology” is used throughout this Standard, reserving the term “method” for conventional phrases such as “method engineer” or “method fragment”.

3.3
metamodel
specification of the concepts, relationships and rules that are used to define a methodology

3.4
endeavour
IBD development effort aimed at the delivery of some product or service through the application of a methodology

Note 1 to entry: Projects, programmes and infrastructural duties are examples of endeavours.

3.5
methodology element
simple component of a methodology

Note 1 to entry: Usually, methodology elements include the specification of what tasks, activities, techniques, models, documents, languages and/or notations can or must be used when applying the methodology. Methodology elements are related to each other, comprising a network of abstract concepts. Typical methodology elements are Capture Requirements, Write Code for Methods (kinds of tasks), Requirements Engineering, High-Level Modelling (kinds of activities), Pseudo-code, Dependency Graphs (notations), Class, Attribute (kinds of model building blocks), Class Model, Class Diagram, Requirements Specification (kind of work products), etc.

3.6
endeavour element
simple component of an endeavour

Note 1 to entry: During the execution of an endeavour, developers create a number of endeavour elements, such as tasks, models, classes, documents, etc. Some examples of endeavour elements are Customer, Invoice (classes), Name, Age (attributes), High-Level Class Model number 17 (a model), System Requirements Description (a document), Coding Cycle number 2, Coding Cycle number 3 (tasks), etc.

3.7
generation
act of defining and describing a methodology from a particular metamodel

Note 1 to entry: Generating a methodology includes explaining the structural position and semantics of each methodology element using the selected metamodel. Thus, what methodology elements are possible, and how they relate to each other, are constrained by such a metamodel. Usually, method engineers perform generation, yielding a complete and usable methodology.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

3.8
enactment
act of applying a methodology for some particular purpose, typically an endeavour

Note 1 to entry: Enacting a methodology includes using the existing generated methodology to create endeavour elements and, eventually, obtain the targeted IBD system. Thus, what kinds of endeavour elements can be created, and how they relate to each other, is governed by the methodology being used. Usually, technical managers, together with other developers, perform enactment.

3.9
method engineer
person who designs, builds, extends and maintains methodologies

Note 1 to entry: Method engineers create methodologies from metamodels via generation.

3.10
developer
person who applies a methodology for some specific job, usually an endeavour

Note 1 to entry: Developers apply methodologies via enactment.

3.11
powertype
type, the instances of which are subtypes of another type called the “partitioned type”

Note 1 to entry: This definition must be interpreted in the context of the object-oriented paradigm. For example, the class TreeSpecies is a powertype of the class Tree, since each instance of TreeSpecies is also a subclass of Tree.

3.12
clabject
dual entity that is a class and an object at the same time

Note 1 to entry: This definition must be interpreted in the context of the object-oriented paradigm. Because of their dual nature, clabjects exhibit a class facet and an object facet, and can work as either at any time. Instances of powertypes are usually viewed as clabjects, since they are objects (because they are instances of a type, the powertype) and also classes (subtypes of the partitioned type).

4 Naming, diagramming and definition conventions, and abbreviated terms

4.1 Naming, diagramming and definition conventions

The SEMDM is defined using different kinds of instruments that complement each other. These instruments are:

- Definitions. Each concept in the SEMDM is defined using natural language. Also, a description is given, including the context in which the concept occurs and its most distinctive properties. Examples are also given for each concept.
- Class diagrams. Concepts of interest to the SEMDM are formalized as classes. Consequently, class diagrams are used to show these classes together with their attributes and relationships. UML 1.4.2 (i.e. ISO/IEC 19501) is used throughout with some noticeable exceptions. First, a special notation is used to depict powertype patterns, consisting of a dashed line between the powertype and the partitioned type with a black dot on the side of the powertype. Secondly, “white diamonds” are used to depict whole/part relationships without making any reference to their secondary characteristics (see [2] for more details).
- Text tables. Text tables are included to provide additional descriptions of attributes and relationships.
- Mappings to other approaches. Each concept in the SEMDM is related to equivalent or similar concepts in other metamodelling approaches, so that translation between approaches is easier.

These instruments are used simultaneously.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

Two different types of class diagrams are provided. Clause 6 presents some diagrams that aim to give an overall picture of the structure of SEMDM. These diagrams are designed to give an idea of the main classes and relationships within the metamodel, and are not comprehensive, i.e. do not display every single detail of the metamodel. Clause 7, on the other hand, includes a class diagram for each class in the metamodel. The class under discussion is shown in the centre, and is surrounded by its closest neighbours. Each of these diagrams, together with the accompanying attribute and relationship tables, do contain all the details for the particular class being discussed.

The philosophy of the SEMDM is to offer broad coverage for all the issues often found in methodology definition avoiding, at the same time, unnecessary structural constraints on the resultant methodologies. Therefore, only a minimal set of attributes and associations is provided by the metamodel. Using powertype pattern instantiation (see sub-clause 8.1.2), and thanks to the usage of powertypes in the metamodel, additional attributes and associations can be easily added at the methodology domain.

4.2 Abbreviations

IBD information-based domain

5 Basic Concepts

Metamodels are useful for specifying the concepts, rules and relationships used to define methodologies. Although it is possible to describe a methodology without an explicit metamodel, formalizing the underpinning ideas of the methodology in question is valuable when checking its consistency or when planning extensions or modifications. A good metamodel must address all of the different aspects of methodologies, i.e. the process to follow, the work products to be generated and those responsible for making all this happen. In turn, specifying the work products that must be developed implies defining the basic modelling building blocks from which they are built.

Metamodels are often used by method engineers to construct or modify methodologies. In turn, methodologies are used by developers to construct products or deliver services in the context of endeavours. *Metamodel*, *methodology* and *endeavour* constitute, in this approach, three different areas of expertise that, at the same time, correspond to three different levels of abstraction and three different sets of fundamental concepts. As the work performed by developers at the endeavour level is constrained and directed by the methodology in use, the work performed by the method engineer at the methodology level is constrained and directed by the chosen metamodel. Traditionally, these relationships between “modelling layers”, here called “domains”, are seen as *instance-of* relationships, in which elements in one layer or domain are instances of some element in the layer or domain below (Figure 1).

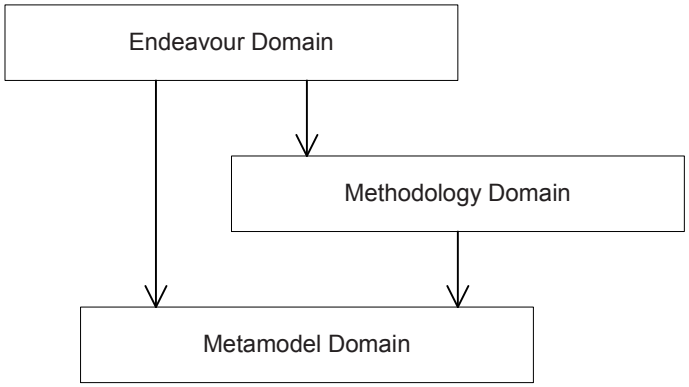


Figure 1 – The three areas of expertise, or domains, which act as a context for SEMDM. Arrows mean "is represented by".

Regarding the methodology domain, it must be noted that more than one “methodology” may exist at this level, interlinked by refinement relationships. For example, it is common that organizations create organization-wide, generic methodologies from a metamodel, and then adjust and customize said methodologies for each particular endeavour. In cases like this, both kinds of methodologies (organization-wide and endeavour-

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

specific) belong in the methodology domain and are connected via a refinement relationship (as opposed to instance-of). Cases with more than two steps of refinement are also possible.

5.1 Method Engineering

In accordance with most of the above-mentioned approaches to metamodelling, the SEMDM accepts the idea of method engineering (see [3, 18] for an introduction), defining the metamodel as a set of classes from which "methodology chunks" can be generated and then composed into a usable methodology [8]. However, the method engineering approach has been used primarily in the process realm (and hence the often-used name of "process engineering"), whereas the SEMDM extends it to the modelling domain as well (see next section).

5.2 Dual-Layer Modelling

Most metamodelling approaches define a metamodel as a model of a modelling language, process or methodology that developers may employ. Following this conventional approach, classes in the metamodel are used by the method engineer to create instances (i.e. objects) in the methodology domain and thus *generate* a methodology. However, these *objects* in the methodology domain are often used *as classes* by developers to create elements in the endeavour domain during methodology *enactment*. This apparent contradiction, not solved by any of the existing metamodelling approaches, is addressed by the SEMDM and solved by conceiving a metamodel as a model of *both the methodology and the endeavour* domains. While offering a strict model of the endeavour domain in the metamodel, the SEMDM maintains a high degree of flexibility, allowing the method engineer to configure the development process and address the modelling issues as necessary.

5.3 Powertypes and Clabjects

Two concepts, new to methodology modelling, must be introduced in order to support the features required by the SEMDM. First of all, modelling the methodology and endeavour domains at the same time gives rise to pairs of classes in the metamodel that represent the same concept at different levels of classification. For example, the Document class in the metamodel represents documents managed by developers, while the DocumentKind class in the metamodel represents different *kinds* of documents that can be managed by developers. Notice how Document represents a concept that belongs in the endeavour domain (documents that people manage) while DocumentKind represents a concept that belongs in the methodology domain (kinds of documents described by the methodology). For example, the concept of ClassDiagram is an instance of DocumentKind, but a given class diagram in the endeavour, with a particular author and creation time, is an instance of Document. In turn, these two classes are related by a classification relationship, since every document (in the endeavour domain) is an example (instance) of some particular kind of document (as defined in the methodology domain). This pattern of two classes in which one of them represents "kinds of" the other is called a *powertype pattern*, since the class with the "kind" suffix is a powertype (see [15, 16] for an introduction to the powertype concept) of the other class, called the partitioned type. In this Standard, the notation Document/*Kind is used to refer to the powertype pattern formed by the powertype DocumentKind and the partitioned type Document.

At the same time, endeavour-level elements must be instances of some methodology-level elements, and methodology-level elements must be instances of metamodel-level elements. This means that (at least some) elements in the methodology domain act *at the same time* as objects (since they are instances of metamodel classes) and classes (since endeavour-level elements are instances of them). This class/object hybrid concept has been described in [1] and named *clabject*. Clabjects have a class facet and an object facet. Within the SEMDM, clabjects are the means to construct a methodology from the powertype patterns found in the metamodel. In this way, a powertype pattern can be "instantiated" into a clabject by making the object facet of the clabject an instance of the powertype class in the powertype pattern, and the class facet of the clabject a subclass of the partitioned type in the powertype pattern. For example, a method engineer wanting to support requirement specification documents in the methodology that he or she is constructing would create the clabject RequirementsSpecificationDocument (in the methodology domain) as an instance of Document-Kind *and* a subclass of Document. By using clabjects at the methodology level, every single element susceptible of being instantiated during enactment is represented by a class, which is appropriate for instantiation, and by an object, which is appropriate for automated manipulation by tools. See Figure 2 for an example.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

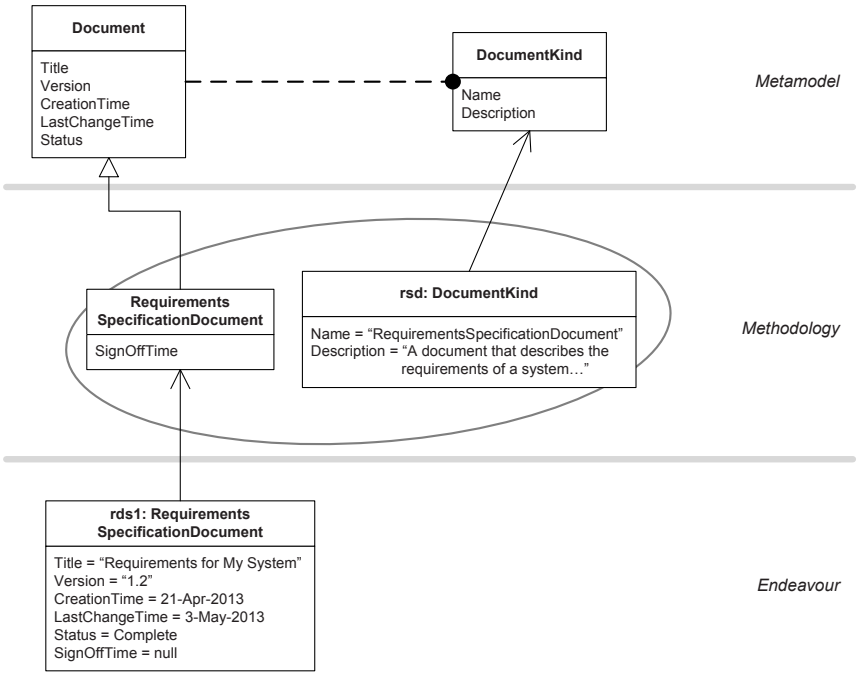


Figure 2 – Example of a powertype pattern and clabject. The Document class is partitioned by the DocumentKind powertype. The RequirementsSpecificationDocument class plus the rsd object represent a particular kind of document, making up a clabject. The rsd1 object represents a particular requirements specification document.

Notice how a given attribute of the powertype class acts as *discriminator* of the powertype pattern, meaning that unique values of that attribute will be assigned to each of the instances of the powertype class, and the same value will be used to name the corresponding subclass of the partitioned type. For example, in the Document/*Kind powertype pattern, DocumentKind.Name is the discriminator. This means that each instance of DocumentKind will have a unique value for Name and its associated class (a subtype of Document) will be named with that value. Following the previous example, a given instance of DocumentKind would have Name = "ClassDiagram", and its corresponding subclass of Document would be called ClassDiagram. The discriminator attribute thus acts as the bond between the two facets of the clabject.

5.4 Uniting Process and Product

Most of the existing metamodelling approaches focus either on the process or on the modelling (i.e. product) side of methodologies. Most of these approaches, however, offer connection points for "plugging in" the complementary, as yet undefined, component of a full-fledged methodology. The SEMDM goes a step beyond by offering a complete metamodel that covers the process and modelling aspects of methodologies evenly. Not doing so would be like trying to define the actions to be performed without defining the concepts on which these actions must act (process focus), or the concepts to use without knowing what to do with them (modelling focus). This approach has the benefit of allowing a rich definition, at the methodology level, of the interactions between a process and the products generated by it.

5.5 Process Assessment

Usually, the maturity or capability of an organization regarding the performance of a process is measured by assigning a *capability level* to its enactment. The SEMDM adopts the concept of capability level and attaches it to work unit kinds, so a method engineer can easily establish the minimum capability level at which each work unit kind may be performed. Although different assessment approaches and standards have slightly different ranges of capability levels (see [11] for an example), the following exemplar list is generic enough to be applicable to nearly every situation:

- **Incomplete** (level 0): the organization fails to successfully execute the process.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

- **Performed** (level 1): the process is successfully executed but may not be rigorously planned and tracked.
- **Managed** (level 2): the process is planned and tracked while it is performed; work products conform to specified standards and requirements.
- **Established** (level 3): the process is performed according to a well-defined specification that may use tailored versions of standards.
- **Predictable** (level 4): measures of process performance are collected and analysed, leading to a quantitative understanding of process capability and an improved ability to predict performance.
- **Optimizing** (level 5): continuous process improvement against business goals is achieved through quantitative feedback.

6 Introduction to the SEMDM

6.1 Highly Abstract View

From the most abstract perspective, the SEMDM defines the classes MethodologyElement and Endeavour-Element that represent, respectively, elements in the methodology and the endeavour domains. Methodology-Element, in turn, is specialized into Resource and Template, corresponding to methodology elements that are used "as is" at the endeavour level (i.e. resources) and methodology elements that are used by instantiation at the endeavour level (i.e. templates) [6]. Since Template is the abstract type of all elements at the methodology level that will have instances at the endeavour level, and EndeavourElement is the abstract superclass of the same elements, these two classes form a powertype pattern in which Template is the powertype, Endeavour-Element is the partitioned type and Template.Name is the discriminant. Powertype patterns and their usage are discussed in sub-clause 5.3. See Figure 3 for a graphical representation.

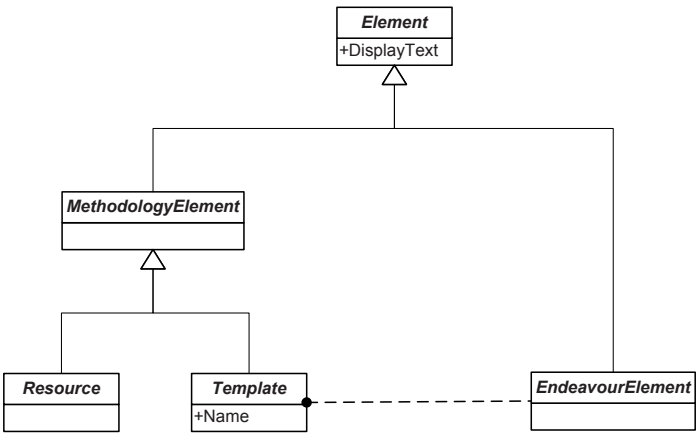


Figure 3 – Highly abstract view of the SEMDM

At the same time, a top class Element is defined to generalize MethodologyElement and EndeavourElement and allow homogeneous treatment of all elements across the methodology and endeavour domains when necessary. The DisplayText attribute of Element gives a short text describing each instance suitable to be shown to the instance's final users.

6.2 Abstract View and Core Classes

There are three clusters of core classes: methodology templates, specializing from Template; methodology resources, specializing from Resource; and endeavour classes, specializing from EndeavourElement.

The powertype pattern formed by Template and EndeavourElement is refined into more specialized powertype patterns formed by subclasses of these two, namely: StageKind and Stage (representing a managed time frame within an endeavour), WorkUnitKind and WorkUnit (a job performed, or intended to be performed, within an endeavour), WorkProductKind and WorkProduct (an artefact of interest for the

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

endeavour), ProducerKind and Producer (an agent that has the responsibility to execute work units) and ModelUnitKind and ModelUnit (an atomic component of a model). See Figure 4 for a graphical depiction.

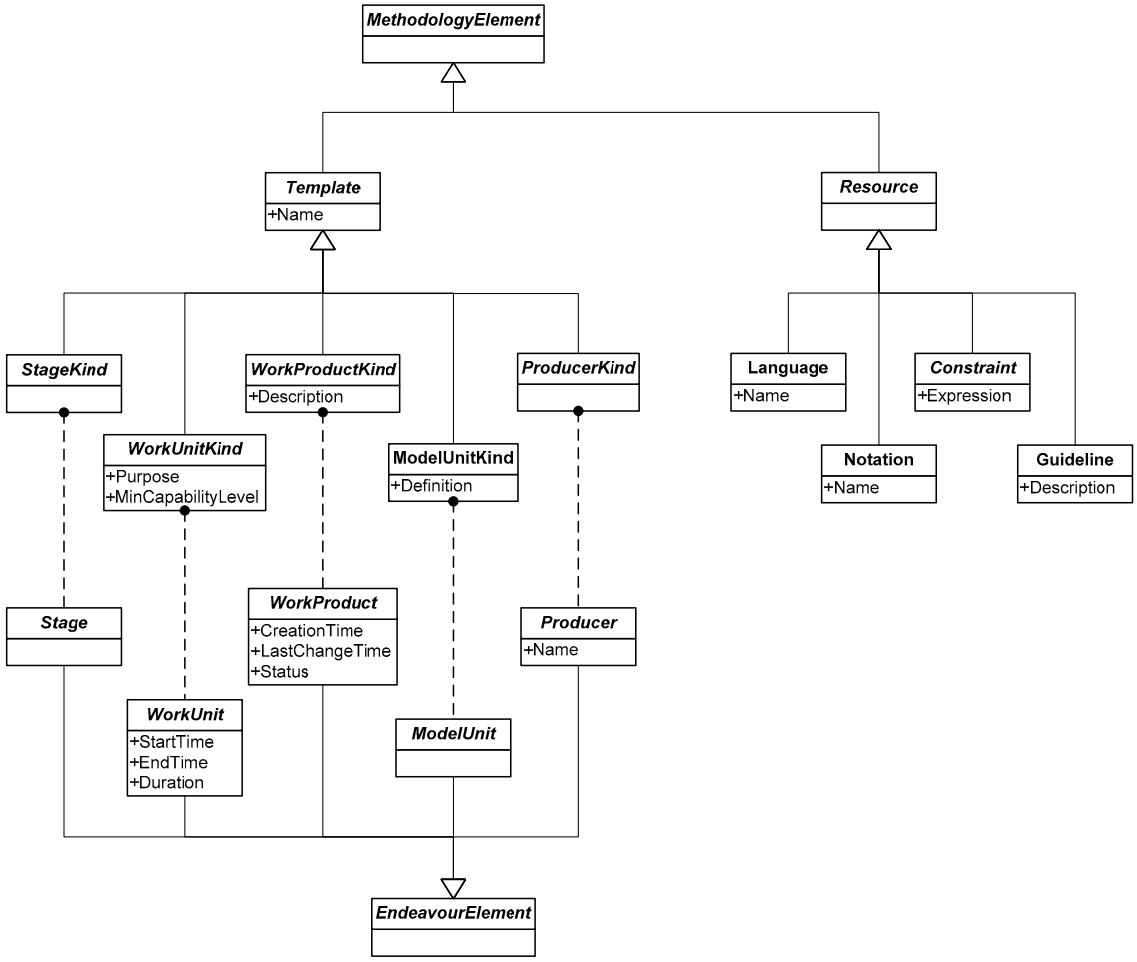


Figure 4 – Abstract view of the SEMDM, showing the core classes in the metamodel

At the same time, Resource is specialized into Language (a structure of model unit kinds that focus on a particular modelling perspective), Notation (a concrete syntax, usually graphical, which can be used to depict models created with certain languages), Guideline (an indication of how some methodology elements can be used) and Constraint (a condition that holds or must hold at certain point in time).

6.3 Process Classes

The WorkUnit/*Kind powertype pattern is specialized into Process/*Kind (large-grained, operating within a given area of expertise), Task/*Kind (small-grained, focusing on *what* must be done in order to achieve a given purpose) and Technique/*Kind (small-grained, focusing on *how* the given purpose may be achieved).

WorkUnitKind is characterized by a purpose and a minimum capability level at which it makes sense to be performed, and is related to OutcomeKind in a many-to-many fashion, so a set of outcome kinds can be defined for each specific kind of work unit. Also, WorkUnit/*Kind holds a whole/part relationship to Task/*Kind, so any work unit or work unit kind can be defined as a collection of tasks or task kinds, respectively. This allows for the recursive definition of units of work down to the necessary level of detail.

Since individual work units happen at the endeavour domain within a particular temporal frame (see below), the WorkUnit class incorporates the necessary attributes to describe this. The WorkUnitKind class, however, is only a specification of what must be done and does not contain any reference to any particular time frame; therefore, no time-related attributes are present. See Figure 5 for a graphical depiction.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

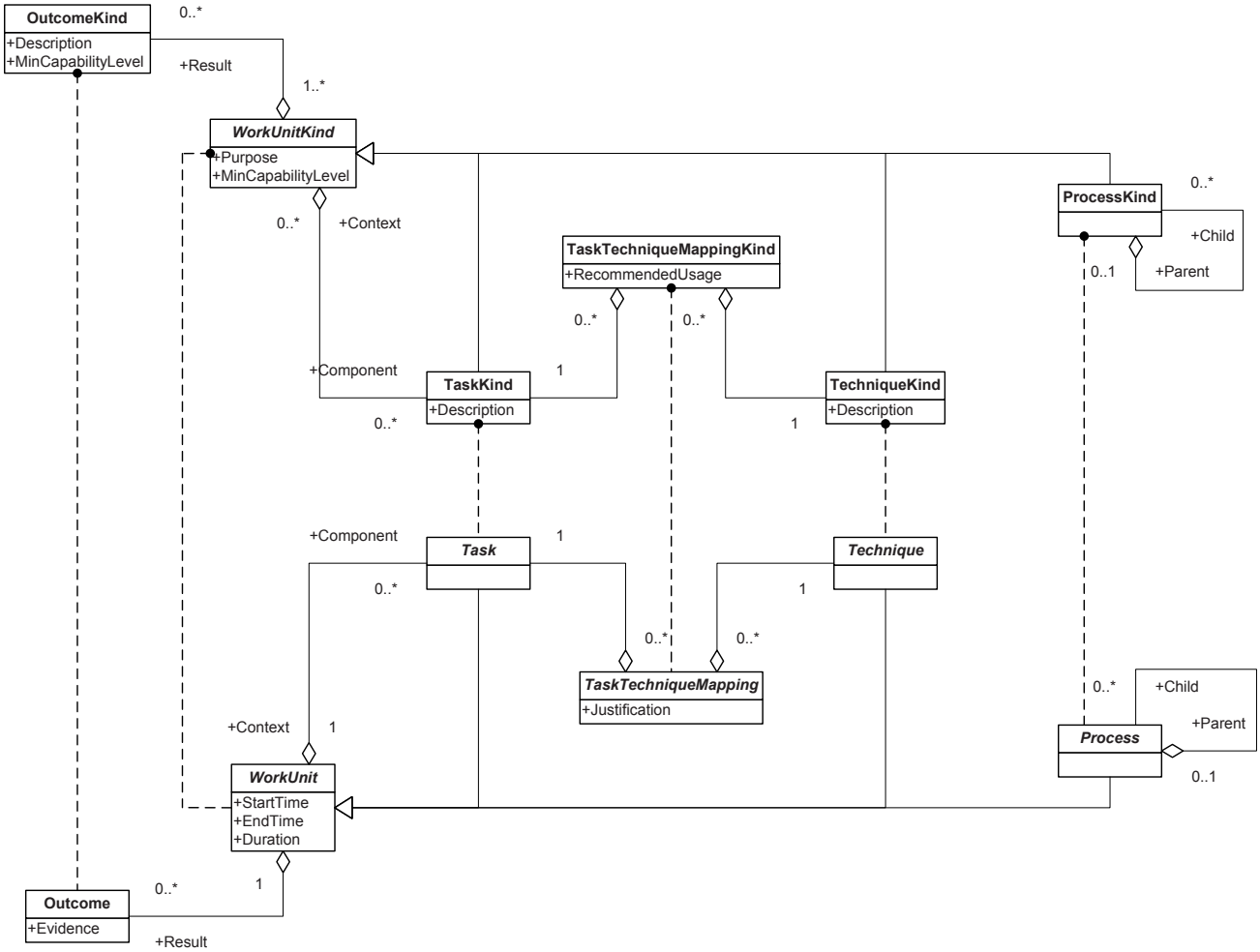


Figure 5 – Work units

On the temporal side, Stage/*Kind is specialized into StageWithDuration/*Kind (a managed interval of time within an endeavour) and InstantaneousStage/*Kind (a managed point in time within an endeavour). Stage-WithDuration/*Kind is, in turn, specialized into TimeCycle/*Kind (having as objective the delivery of a final product or service), Phase/*Kind (having as objective the transition between cognitive frameworks) and Build/*Kind (having as major objective the delivery of an incremented version of an already existing set of work products). TimeCycle/*Kind also holds a whole/part relationship to Stage/*Kind, allowing for the recursive composition of time cycles and other stages. Phase/*Kind, on the other hand, holds a whole/part relationship to Build/*Kind so any phase or phase kind can be linked to the corresponding builds or build kinds, respectively, that occur within it. At the same time, StageWithDuration/*Kind is associated with Process/*Kind so the temporal side of the process can be related to the appropriate elements on the job side. See Figure 6 for a graphical depiction.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

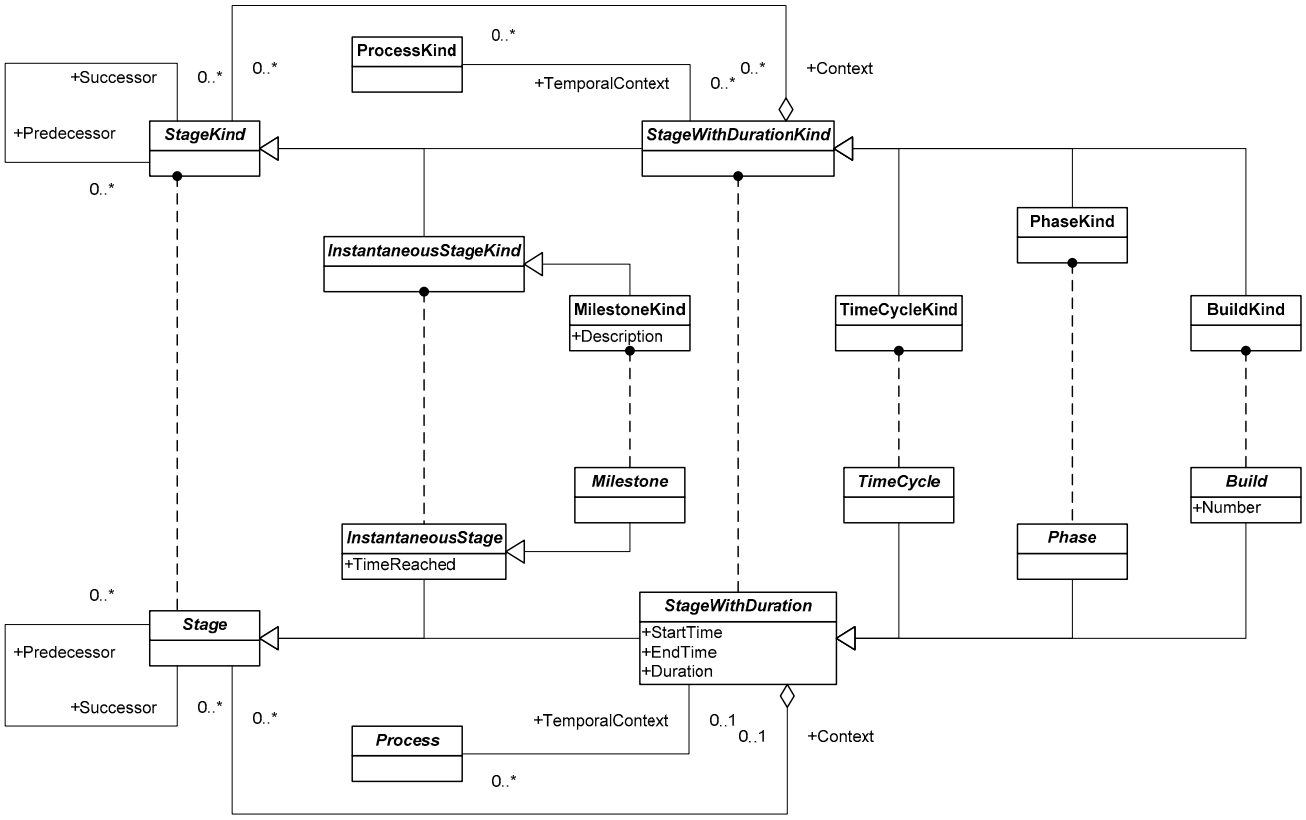


Figure 6 – Stages

NOTE Temporal ordering and sequencing are achieved in SEMDM in two different ways. At a high level of abstraction, stage kinds allow the methodologist to specify the overall temporal structure of a methodology. Stage kinds, in this sense, are “empty containers” that can be “filled” with work unit kinds in order to specify when things are to be done. At a detailed level, however, time ordering and sequencing is not explicitly specified, but emerges from the collections of action kinds associated to each task kind. Action kinds of any given task kind determine what kinds of work products are necessary in order to accomplish the associated task. Thus, at any point in time during enactment, the set of “executable” tasks can be determined by looking at the pool of existing work products and the action kinds associated to each candidate task kind.

6.4 Producer Classes

Producer/*Kind is specialized into Role/*Kind (a collection of responsibilities that a producer can take), Tool/*Kind (an instrument that helps another producer to execute its responsibilities in an automated way). Producer has an additional subclass, Person, which allows taking into account individual persons at the endeavour level. Producer/*Kind is also related to WorkUnit/*Kind through WorkPerformance/*Kind, so links between units of work and the assigned and/or responsible producers are possible. See Figure 7 for a graphical depiction.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

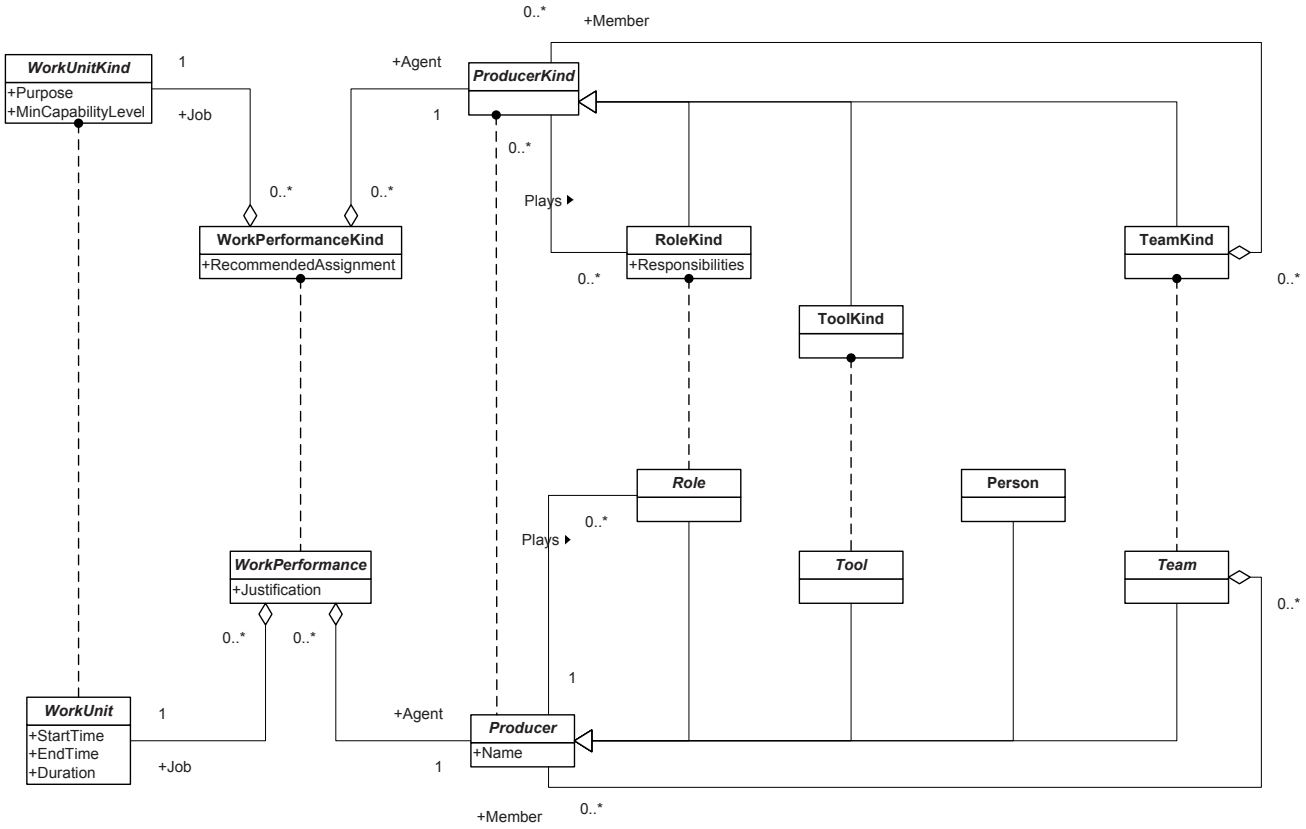


Figure 7 – Producers

6.5 Product Classes

WorkProduct/*Kind has five subtypes: SoftwareItem/*Kind, HardwareItem/*Kind (a piece of software or hardware, respectively, that is of interest to the endeavour), Model/*Kind (an abstract representation of some subject that acts as the subject's surrogate for some well defined purpose), Document/*Kind (a durable depiction of a fragment of reality) and CompositeWorkProduct/*Kind (an aggregate of other elements). Although documents would usually depict models, they can also depict other entities of interest or even other documents. In fact, Document/*Kind has an association to WorkProduct/*Kind to represent this fact. Also, Document/*Kind has a recursive whole/part relationship with itself so a given work product or work product kind can be defined as a collection of other work products or work product kinds, respectively. See Figure 8 for a graphical depiction.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

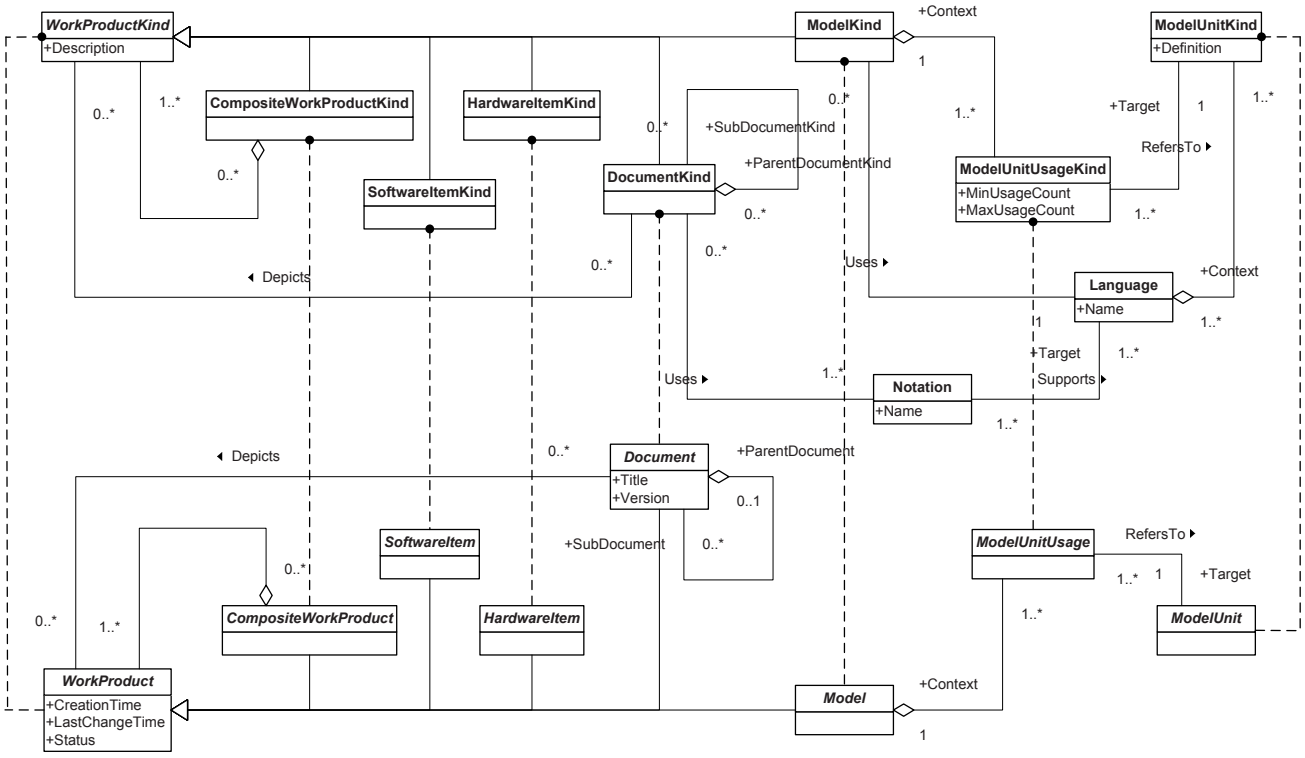


Figure 8 – Work product and modelling classes

Model/*Kind, in turn, holds a whole/part relationship to ModelUnitUsage/*Kind, which in turn is associated to a specific ModelUnit/*Kind. This chain of relationships makes it possible to describe what model units are used in which models, and how they are employed. In addition, every ModelUnitKind is always defined in the context of at least one Language. Different languages sharing the same model unit kinds allow for a single and interconnected network of model units and models across a system rather than having different separate, isolated models. In addition, ModelKind and Language are directly linked by an association to support cases in which a method engineer wishes to specify what language is used by a certain model kind without detailing the component model unit kinds. Also, Language is associated with Notation to represent the fact that different notations support (or can depict) different languages. Finally, Notation is also related to DocumentKind to represent that each document kind makes use of at least one notation.

Note that Language and ModelUnit/*Kind can generate any required modelling language.

6.6 Connection between Process and Product

The interaction between the process and the product sides of the metamodel is achieved by the powertype pattern Action/*Kind. An Action/*Kind is always performed in the context of a given Task/*Kind (process side), and acts upon a given WorkProduct/*Kind (product side). The ActionKind.Type attribute takes values showing whether actions of a specific kind create, modify or only read work products of a given kind. Note that some task kinds may not perform any action kinds whatsoever. See Figure 9 for a graphical depiction.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

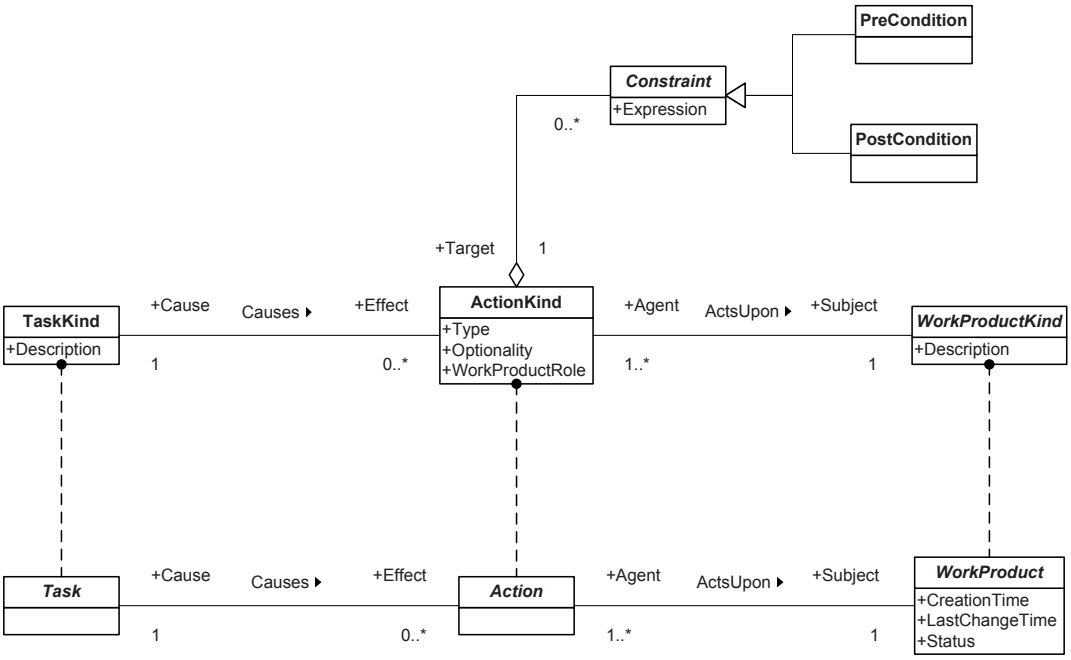


Figure 9 – Actions and constraints

ActionKind is also related to Constraint, which is specialized into PreCondition (a condition that must hold in order for the associated action to proceed) and PostCondition (a condition that is guaranteed to hold after the associated action has been successfully performed).

6.7 Support Classes

In addition to the classes necessary to construct methodologies, some support classes exist for the convenience of method engineers using the SEMDM. See Figure 10 for a graphical depiction.

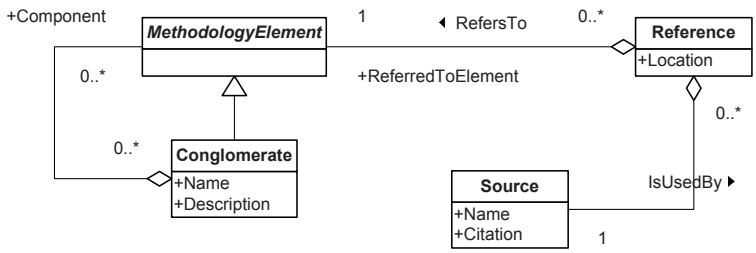


Figure 10 – Support classes

Some mechanism for specifying reusable methodology chunks is necessary. The Conglomerate class is defined to represent collections of related methodology elements (i.e. instances of the MethodologyElement class) that can be defined by a method engineer and then reused in different methodological contexts. Note that Conglomerate is also a subtype of MethodologyElement, so recursive composition of conglomerates is possible.

Also, some means of managing references to bibliographic sources and best practices is needed. The Source class represents literature items or other sources of information and experience that a method engineer may want to use when defining methodology elements. The Reference class acts as a link between Source and Element so any number of linkages between methodology elements and sources can be specified.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

7 Metamodel Elements

7.1 Classes

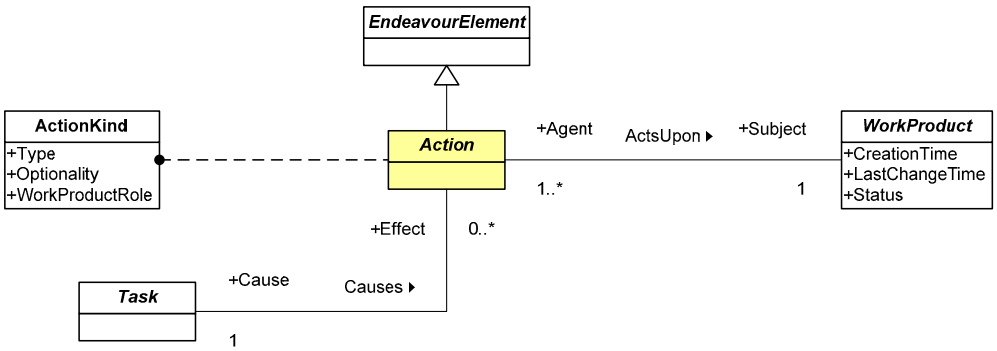
In this section, the classes in the SEMDM are described in alphabetical order. For each class, a definition is given in italics, and a description optionally follows. A diagram for each class is also included, showing the class being defined in the context of its immediate neighbours in the metamodel. Then, the attributes of the class are listed, including, for each one, its name, data type and semantics. Attribute data types are always one of the basic primitive types (Boolean, Integer, Timestamp or String) or an enumerated type defined in sub-clause 7.2. Finally, the relationships that the class is involved in are listed from the class' perspective, including, for each one, the name of the relationship if there is one, the role that the class being described plays in said relationship if there is one, the target class to which the class is associated, and its semantics.

7.1.1 Action

An action is *a usage event performed by a task upon a work product*. Actions represent the fact that specific tasks use specific work products.

Action is an abstract subclass of EndeavourElement.

This is a process- and product-related class.



7.1.1.1 Attributes

This class has no attributes of its own.

7.1.1.2 Relationships

Name	Role	To class	Semantics
IsClassifiedBy	n/a	ActionKind	An action in the endeavour domain is always of some action kind defined in the methodology domain.
ActsUpon	Agent	WorkProduct	An action always acts upon a particular work product.
n/a	Effect	Task	An action is always the effect of a particular task.

7.1.1.3 Example

In a software development project, developer John executes a programming task (a task) that involves making modifications to the source code file "Invoice.cs" (a work product). The event of said task changing said work product is an action.

7.1.2 ActionKind

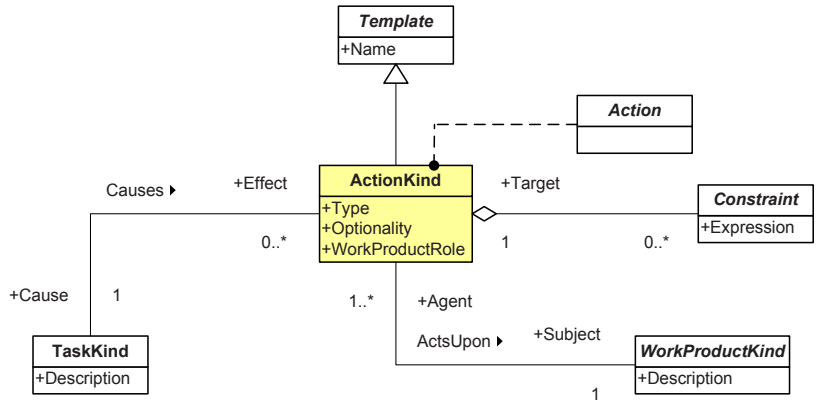
An action kind is *a specific kind of action, characterized by a given cause (a task kind), a given subject (a work product kind) and a particular type of usage.* Action kinds describe how tasks of specific kinds use work products of specific kinds, including the nature of such usage, i.e. creation, modification, etc.; its optionality (whether tasks of the associated kind always use work products of the associated kind via this action kind, or whether this is optional to some degree); and, optionally, the role that each work product kind plays when

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

acted upon by the associated task kind. This last characterization is useful to differentiate work products when a single task kind is linked (via action kinds) to the same work product kind multiple times. In cases like this, it is expected that each action kind will be marked with a distinctive work product role.

ActionKind is a subclass of Template.

This is a process- and product-related class.



7.1.2.1 Attributes

Name	Type	Semantics
Type	ActionType	The nature of the usage that the associated task kind performs on the associated work product kind. See sub-clause 7.2.1 for possible values.
Optionality	DeonticValue	The degree of obligation for the associated task kind to act upon the associated work product kind. See sub-clause 0 for possible values.
WorkProductRole	String	The role that work products of the associated kind will play during enactment within actions of this kind.

7.1.2.2 Relationships

Name	Role	To class	Semantics
Classifies	n/a	Action	An action in the endeavour domain is always of some action kind defined in the methodology domain.
ActsUpon	Agent	WorkProductKind	An action kind is always the agent that acts upon a particular work product kind.
n/a	Effect	TaskKind	An action kind is always the effect of a particular task kind.
n/a	Target	Constraint	An action kind may be constrained by some constraints.

7.1.2.3 Example

In a given methodology, a task kind “Determine business concepts” is defined, together with a work product kind “Business Concept Dictionary”. Both are related by the fact that tasks of the “Determine business concepts” kind will, when executed, create work products of the “Business Concept Dictionary” kind. Said relationship is modelled as an action kind with Type = Create and Optionality = Mandatory.

7.1.3 Build

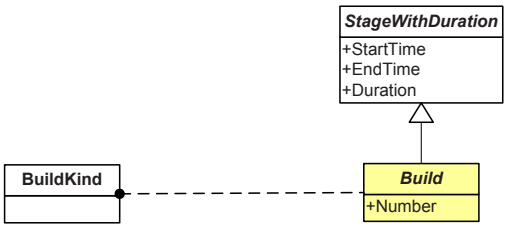
A build is *a stage with duration for which the major objective is the iteration of a collection of component stages.* Builds are often used to implement incremental, iterative time cycles and to deliver incremented versions of already existing work products.

Build is an abstract subclass of StageWithDuration.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

This is a process-related class.



7.1.3.1 Attributes

Name	Type	Semantics
Number	String	The number of the build. Since builds are, by definition, incremental, some kind of numbering system is strongly recommended.

7.1.3.2 Relationships

Name	Role	To class	Semantics
IsClassifiedBy	n/a	BuildKind	A build in the endeavour domain is always of some build kind defined in the methodology domain.

7.1.3.3 Example

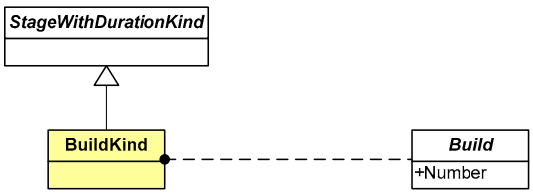
In a software development project, John's team spends two weeks focusing on analysing, modelling, implementing and testing a few new system features. After these two weeks, they deliver a partial implementation of the final system. Once this is done, they pick a new bunch of features and repeat the analysis, modelling, implementation and testing to construct an incremented version of the system. They keep doing this until the system contains all the required features. Each of the short time spans in which a new set of features is incrementally incorporated to the system, as performed by John's team, is a build.

7.1.4 BuildKind

A build kind is *a specific kind of build, characterized by the kind of stages over which it iterates.*

BuildKind is a subclass of StageWithDurationKind.

This is a process-related class.



7.1.4.1 Attributes

This class has no attributes of its own.

7.1.4.2 Relationships

Name	Role	To class	Semantics
Classifies	n/a	Build	A build in the endeavour domain is always of some build kind defined in the methodology domain.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

7.1.4.3 Example

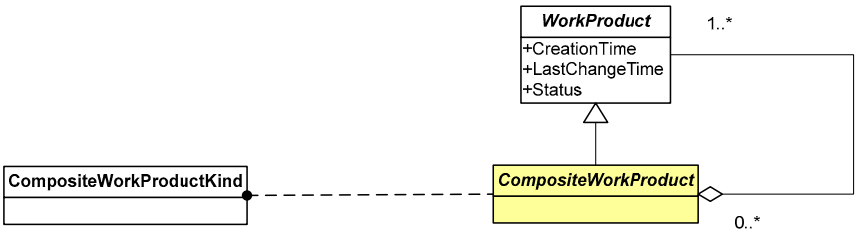
In a given methodology, a build kind "Construction Build" is defined to represent the fact that, when said methodology is enacted, a sequence of construction builds will be performed in order to construct the product incrementally.

7.1.5 CompositeWorkProduct

A composite work product is *a work product composed of other work products*.

CompositeWorkProduct is an abstract subclass of WorkProduct.

This is a product-related class.



7.1.5.1 Attributes

This class has no attributes of its own.

7.1.5.2 Relationships

Name	Role	To class	Semantics
IsClassifiedBy	n/a	CompositeWork-ProductKind	A composite work product in the endeavour domain is always of some composite work product kind defined in the methodology domain.
IsMadeOf	Composite	WorkProduct	A composite work product is made of one or more work products.

7.1.5.3 Example

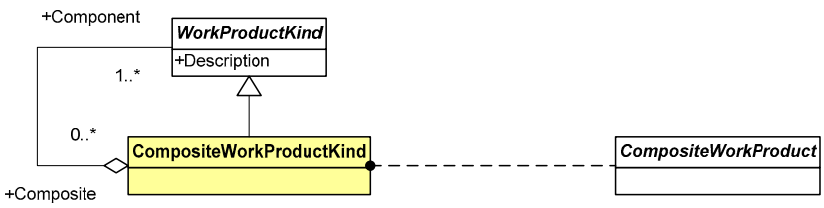
Upon finalization of a systems development project, a complex configuration of hardware, software and documentation is delivered to the customer. Each individual released work product can be modelled as either a hardware item, a software item or a document; the complete, final product delivered to the customer is a composite work product.

7.1.6 CompositeWorkProductKind

A composite work product kind is *a specific kind of composite work product, characterized by the kinds of work products that are part of it*.

CompositeWorkProductKind is a subclass of WorkProductKind.

This is a product-related class.



© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

7.1.6.1 Attributes

This class has no attributes of its own.

7.1.6.2 Relationships

Name	Role	To class	Semantics
Classifies	n/a	CompositeWork-Product	A composite work product in the endeavour domain is always of some composite work product kind defined in the methodology domain.
IsMadeOf	Composite	WorkProductKind	A composite work product kind is made of one or more work product kinds.

7.1.6.3 Example

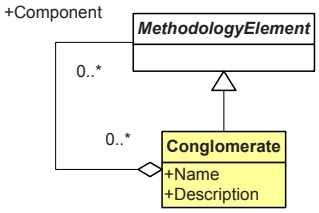
In a given methodology, the final product to be delivered to the customer upon completion of a project is modelled as a composite work product kind. Such a composite includes software and hardware item kinds plus the associated document kinds.

7.1.7 Conglomerate

A conglomerate is *a collection of related methodology elements that can be reused in different methodological contexts.* Conglomerates provide the basic reuse mechanism in the SEMDM.

Conglomerate is a subclass of MethodologyElement.

This is a support class.



7.1.7.1 Attributes

Name	Type	Semantics
Name	String	The name of the conglomerate.
Description	String	The description of the conglomerate, usually explaining the methodological contexts for which it has been designed.

7.1.7.2 Relationships

Name	Role	To class	Semantics
n/a	n/a	Methodology-Element	A conglomerate is composed of a collection of methodology elements.

7.1.7.3 Example

In a given methodology, the process kind “Quality Assurance”, the document kinds “Quality Standard” and “Quality Report”, and the team kind “Quality Assurance Team” (all of them methodology elements) are put into a conglomerate named “Quality-Related Fragments” so a methodologist can, easily and in a single step, incorporate it to a customized methodology or remove it from a methodology.

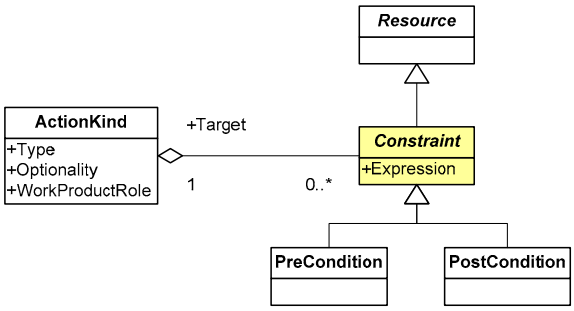
7.1.8 Constraint

A constraint is *a condition that holds or must hold at certain point in time*. Constraints are often used to declaratively characterize the entry and exit conditions of actions.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

Constraint is an abstract subclass of Resource, specialized into PreCondition and PostCondition.

This is a process- and product-related class.



7.1.8.1 Attributes

Name	Type	Semantics
Expression	String	The expression that must evaluate as true for the constraint to hold. Note that this is an abstract attribute and therefore different subclasses attach different specific semantics to it.

7.1.8.2 Relationships

Name	Role	To class	Semantics
n/a	n/a	ActionKind	A constraint always characterizes a given action kind.

7.1.8.3 Example

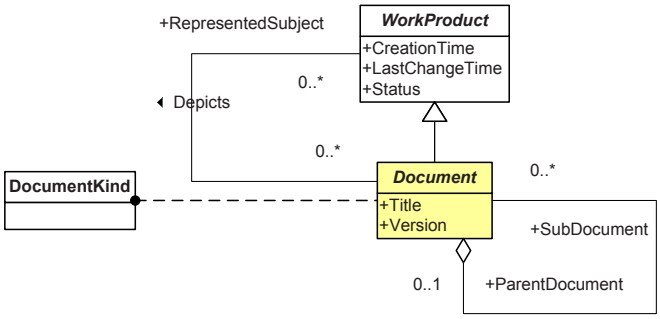
In a given methodology, a task kind “Deliver user documentation” is defined, together with a work product kind “User Documentation”. Both are related by the fact that tasks of the “Deliver user documentation” kind will, when executed, use work products of the “User Documentation” kind. Said relationship is modelled as an action kind with Type = ReadOnly. However, the method engineer wants to capture the need that user documentation is only delivered after the associated “User Interface Specification” work product has been approved. This is done by creating a constraint with Expression = (UserInterfaceSpecification.WorkProduct-Status is Approved) (a precondition) and attaching this constraint to the above mentioned action. Thus, the action will not be allowed to execute unless the required condition is met.

7.1.9 Document

A document is *a durable depiction of a fragment of reality*. Documents often represent models, but they can also represent other subjects.

Document is an abstract subclass of WorkProduct.

This is a product-related class.



© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

7.1.9.1 Attributes

Name	Type	Semantics
Title	String	The title of the document.
Version	String	The version identifier of the document. Since documents are durable by definition, some version control is strongly recommended.

7.1.9.2 Relationships

Name	Role	To class	Semantics
IsClassifiedBy	n/a	DocumentKind	A document in the endeavour domain is always of some document kind defined in the methodology domain.
n/a	ParentDocument	Document	A document may be the parent of sub-documents.
n/a	SubDocument	Document	A document may be a sub-document of a number of parent documents.
Depicts	n/a	WorkProduct	A document may depict a number of work products.

7.1.9.3 Example

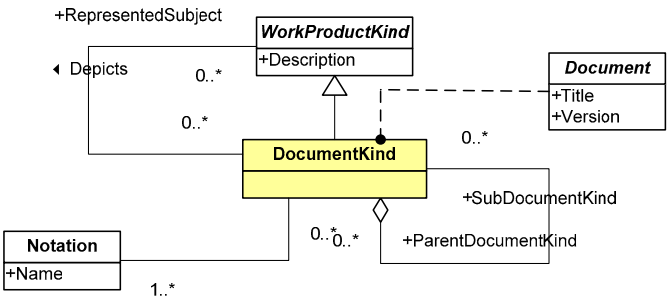
In order to organize a code inspection, Mary prints out the code to be inspected plus a copy of the inspection checklist. During the inspection, Mary takes notes of the defects found in the code and then she compiles these results into an inspection report. The code to be inspected, the inspection checklist and the inspection report, as used by Mary, are documents.

7.1.10 DocumentKind

A document kind is *a specific kind of document, characterized by its structure, type of content and purpose.*

DocumentKind is a subclass of WorkProductKind.

This is a product-related class.



7.1.10.1 Attributes

This class has no attributes of its own.

7.1.10.2 Relationships

Name	Role	To class	Semantics
Classifies	n/a	Document	A document in the endeavour domain is always of some document kind defined in the methodology domain.
n/a	Parent-DocumentKind	DocumentKind	A document kind may be the parent of sub-document kinds.
n/a	SubDocument-Kind	DocumentKind	A document kind may be a sub-document kind of a number of parent document kinds.
Depicts	n/a	WorkProductKind	A document kind may depict a number of work product kinds.
n/a	n/a	Notation	A document kind uses some given notations.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

7.1.10.3 Example

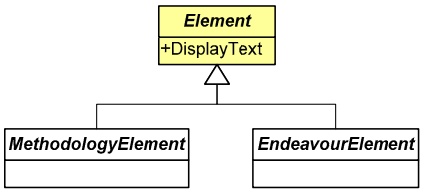
In a given methodology, the document kind “System Requirements Specification” is defined to represent the fact that, when said methodology is enacted, documents of such a kind will be created or used.

7.1.11 Element

An element is *an entity of interest to the metamodel.* Since the SEMDM addresses both the methodology and the endeavour domains (see sub-clause 5.2), any entity in either of these realms is subject to being modelled by the SEMDM and therefore to becoming represented by an element.

Element is an abstract class, specialized into MethodologyElement and EndeavourElement.

This is a high-level class.



7.1.11.1 Attributes

Name	Type	Semantics
DisplayText	String	Name or description suitable to be displayed to final users of the element. The value of this attribute can be, in many of the subclasses of Element, computed from other attributes.

7.1.11.2 Relationships

This class has no relationships of its own.

7.1.11.3 Example

This class is too abstract to give a concrete example. Please see the examples for any of the subtypes of Element.

7.1.12 EndeavourElement

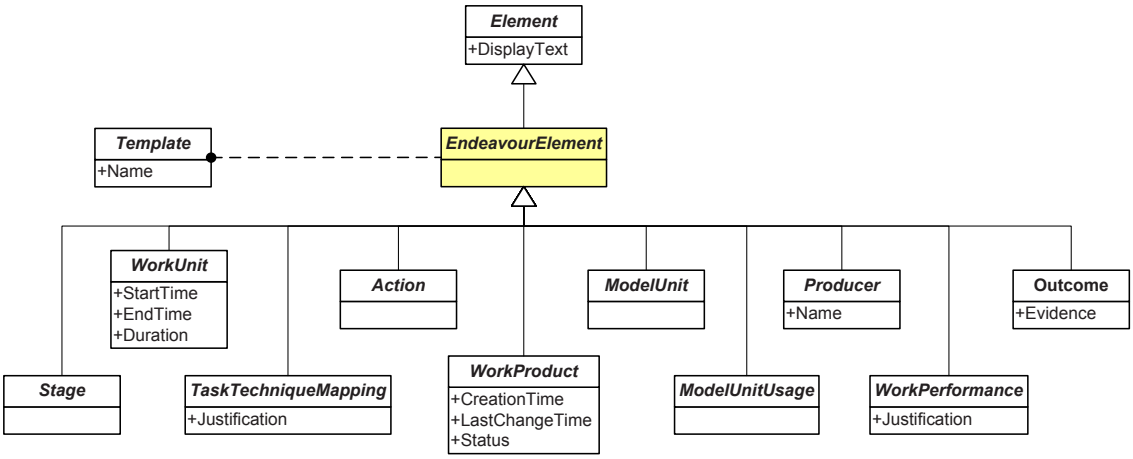
An endeavour element is *an element that belongs in the endeavour domain*. Any element created by a developer while using a methodology is represented by EndeavourElement.

EndeavourElement is an abstract subclass of Element, specialized into Stage, WorkUnit, Outcome, Task-TechniqueMapping, Action, WorkProduct, ModelUnit, ModelUnitUsage, Producer and WorkPerformance.

This is a high-level class.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014



7.1.12.1 Attributes

This class has no attributes of its own.

7.1.12.2 Relationships

Name	Role	To class	Semantics
IsClassifiedBy	n/a	Template	An endeavour element is always of some template defined in the methodology domain.

7.1.12.3 Example

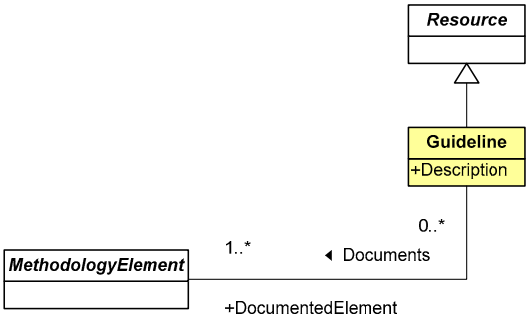
This class is too abstract to give a concrete example. Please see the examples for any of the subtypes of EndeavourElement.

7.1.13 Guideline

A guideline is *an indication of how a set of methodology elements can be used during enactment.*

Guideline is a subclass of Resource.

This is a high-level class.



7.1.13.1 Attributes

Name	Type	Semantics
Description	String	The description of the usage of the associated methodology elements.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014

7.1.13.2 Relationships

Name	Role	To class	Semantics
Documents	n/a	Methodology-Element	A guideline always documents some methodology elements.

7.1.13.3 Example

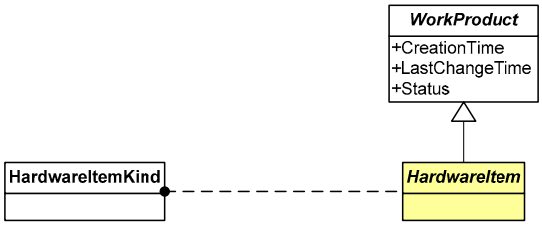
In a particular agent-oriented software development methodology, the model unit kind “Role” is defined to represent roles that agents may play at run-time. Since the concept of “role” in this context is different to the concept of “role” in object-oriented methodologies, the method engineer decides to create a guideline explaining how the model unit kind “Role” in this particular methodology is intended to be used, and attaches said guideline to said model unit kind.

7.1.14 HardwareItem

A hardware item is *a piece of hardware of interest to the endeavour*.

HardwareItem is an abstract subclass of WorkProduct.

This is a product-related class.



7.1.14.1 Attributes

This class has no attributes of its own.

7.1.14.2 Relationships

Name	Role	To class	Semantics
IsClassifiedBy	n/a	HardwareItem-Kind	A hardware item in the endeavour domain is always of some hardware item kind defined in the methodology domain.

7.1.14.3 Example

During an IT infrastructure deployment project, a number of sub-networks are organized and then interconnected via routers. Each sub-network and each router is a hardware item.

7.1.15 HardwareItemKind

A hardware item kind is *a specific kind of hardware item, characterized by its mechanical and electronic characteristics, requirements and features.*

HardwareItemKind is a subclass of WorkProductKind.

This is a product-related class.

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24744:2014