INTERNATIONAL STANDARD

ISO/IEC 24744

First edition
2007-02-15

Software Engineering — Metamodel for Development Methodologies

Ingénierie du logiciel — Métamodèle pour les méthodologies de développement

Reference number
ISO/IEC 24744:2007(E)

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

Contents

Page

Table of Figures

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 24744 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

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 International Standard introduces the Software Engineering Metamodel for Development Methodologies SEMDM, a comprehensive metamodel that makes use of a new approach to defining methodologies based on the concept of powertype. The SEMDM is aimed at 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.

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 work 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.

1.1 Purpose

This International Standard 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 [1-7] 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; and
- The interoperability of modelling and methodology support tools.

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

1.2 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

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.

2 Conformance

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

- describes the scope of the concepts in the metamodel in relation to the scope of the elements defined in Clause 7; and
- 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 International Standard (i.e. its elements 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 International Standard if it also implements the necessary features so as to make the mechanisms described in 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 International Standard if it also implements the necessary features so as to make the mechanisms described in 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. Unless otherwise noted, the definitions are specific to this International Standard.

The following concepts are defined only for their usage throughout this International Standard.

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

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

NOTE 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
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 A methodology specifies the process to be executed, usually as a set of related activities, tasks and/or techniques, together with the work products that 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.

3.3
method
synonym of methodology

NOTE The term “methodology” is used throughout this International Standard, reserving the term “method” for conventional phrases such as “method engineer” or “method fragment”.

3.4
metamodel
specification of the concepts, relationships and rules that are used to define a methodology

3.5
endeavour
IBD development effort aimed at the delivery of some product or service through the application of a methodology

EXAMPLES Projects, programmes and infrastructural duties are examples of endeavours.

3.6
methodology element
simple component of a methodology

NOTE 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.

3.7
endeavour element
simple component of an endeavour

NOTE 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.

3.8
generation
act of defining and describing a methodology from a particular metamodel. 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.

3.9
enactment
act of applying a methodology for some particular purpose, typically an endeavour

NOTE 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.

3.10
method engineer
person who designs, builds, extends and maintains methodologies

NOTE Method engineers create methodologies from metamodels via generation.

3.11
developer
person who applies a methodology for some specific job, usually an endeavour

NOTE Developers apply methodologies via enactment.

3.12
powertype
A powertype of another type, called the *partitioned type*, is a type the instance of which are subtypes of the partitioned type. This definition is 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.

3.13
clabject
dual entity that is a class and an object at the same time

NOTE This definition is 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 [8] 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.

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

SEMDM software engineering metamodel for development methodologies

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).

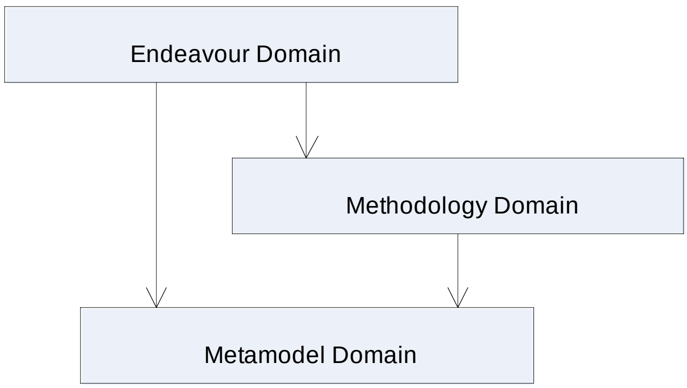

Figure 1 – The three areas of expertise, or domains, which act as a context for SEMDM

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-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 [9, 10] 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 [11]. 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 5.2).

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 [12] for an introduction to the powertype concept) of the other class, called the partitioned type. In this International 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 [13] 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 DocumentKind *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.

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 expressed using the MinCapabilityLevel attribute of class WorkUnitKind, 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 [14] 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.
- **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 EndeavourElement that represent, respectively, elements in the methodology and the endeavour domains. MethodologyElement, 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) [3]. 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, EndeavourElement is the partitioned type and Template.Name is the discriminant. Powertype patterns and their usage are discussed in sub-clause 5.3. See Figure 2 for a graphical representation.

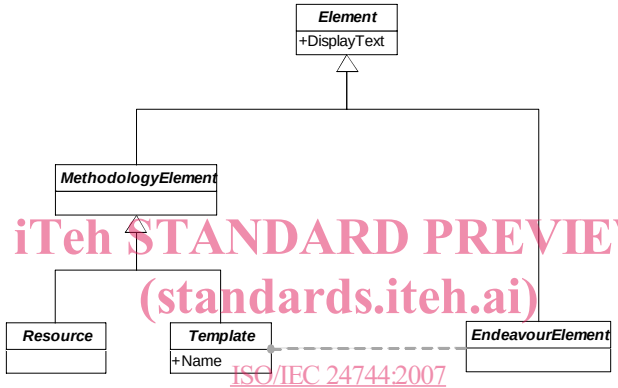

Figure 2 – Highly abstract view of the SEMDM

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 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 3 for a graphical depiction.

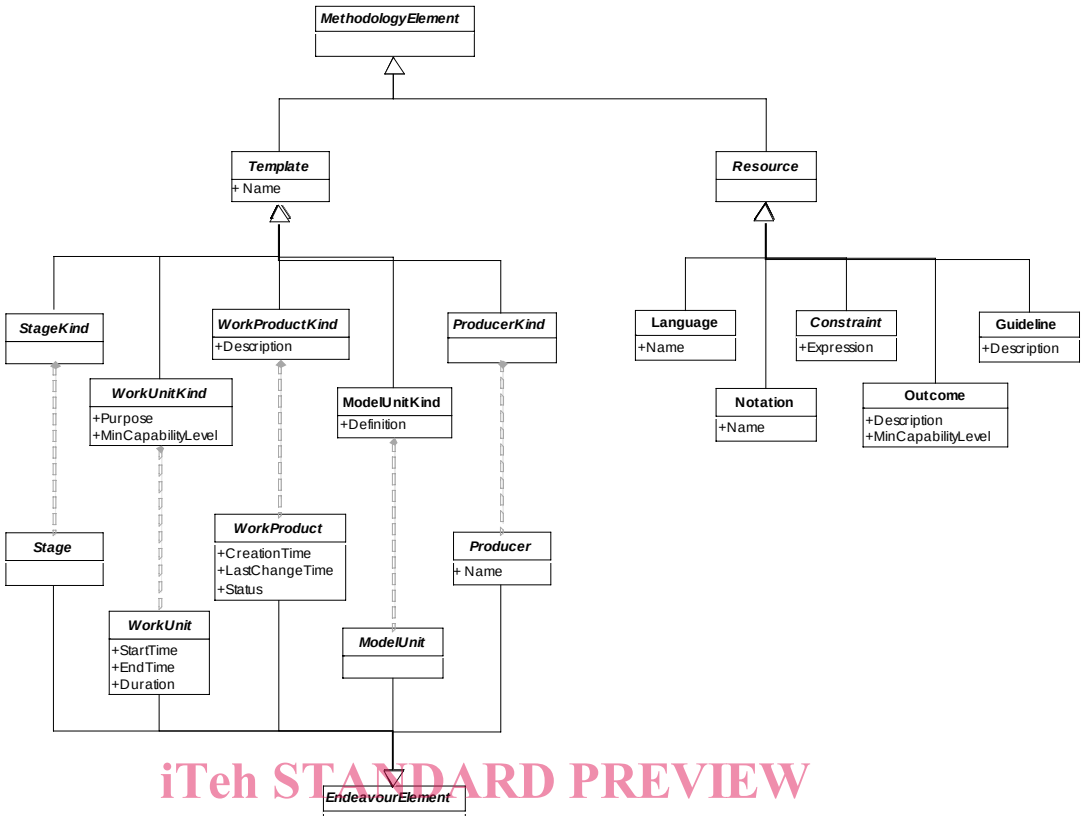

Figure 3 – Abstract view of the SEMDM, showing the core classes in the metamodel

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), Constraint (a condition that holds or must hold at certain point in time) and Outcome (an observable result of the successful performance of a work unit).

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 Outcome in a one-to-many fashion, so a set of outcomes 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 4 for a graphical depiction.