Basic model of the EML

Onderwijstechnologisch expertisecentrum OTEC
Open Universiteit Nederland

Basic Model of the EML

OTEC 2002/20

Colofon

Titel:	Basic Model of the EML
Auteurs:	Arjan Loeffen, Jocelyn Manderveld, Rob Koper, Hubert Vogten en Marc Verhooren
Projectleiding	Jocelyn Manderveld
Projectondersteuning	Mieke Haemers
Uitgifte	OTEC
Datum druk	7 juni 2002

Table of contents

Basic model of the EML

Approach

The following sections describe the UML classes that represent the core structures of EML. In the following sections they are introduced using UML class diagrams. In the text descriptions, a class (property) is referenced by a name prefixed by underscore, e.g. _*class* and _*class.property*.

The EML structure is centered around the concept of a 'play'. Most of the components are named after the elements in the EML specification; some have been added to model implied constructs, such as 'task', and others have received a more explicit name, such as 'property'. This however does not alter the fundamental mapping of the EML to the model presented.

UML class diagrams

_play — UML — Play diagram

A "play" is the primary mechanism of collecting separate learning components into a meaningful whole, just as a set of theatrical movements are collected into a play on stage, a narration. Plays consist of acts (_*act*) in turn consist of tasks by an actor (_*role.performs*). The task is always associated with exactly 1 role (_*role*). A task is set in an environment (_*environment*) and is constrained by a start- and end-level (_*level*). Finally, the activity or set of activities that is inherent to the perfomance is described explicitly (_*activity-list*).

Note 1	The EML constructs model intended behavior (like a script), not actual behavior (like a performance). Class names reflext this consideration.

In the current XML binding the relation of play, act and task can be expressed by *implicit substructures for the <play> element* (p. 8).

Figure 1 - Mapping of UML play related classes onto EML element structure

Below we show the class diagram.

The UML model classes are described here.

- *_play* (p. 7) — A play is a particular scheduling of tasks. A unit of study is an envelope that holds the composites that are required for performing a play. The unit of study may also be taken as a task by its own.

  This element maps onto `<play>`

- *_act* — A set of role performances that, as a group, must be finished before the next set of tasks can start. The act therefore has an attribute *_act.continue* which holds a condition.
- *_role.performs* — The intended performance of a task by a role. Role performances are scheduled by a single act. This class is therefore part of the workflow model of the EML.
- *_role* — A role is a description of a person or group of persons that perform a task in the play. Roles may have subroles (composite roles), which means each more specific role takes on the properties of the composite role. The role may be
  - A learner, i.e. a role that is filled by persons that need to learn by perfoming tasks.
  - A staff member, i.e. a role that is filled by persons that support learners in perfoming learning tasks. For staff, no learning objectives are set; a staff task may only have general objectives.
- *_task* — tasks are intended operations to be performed by persons or groups in a role (*actors*). A task has an environment in which it is performed, and a context.

  A task is scheduled by an *_act* within a *_play* (p. 7). The *_role.performs* specifies which actors must perform which activities.

  A task is bound by a *_level*. If an actor does not comply with the prerequisites, the task is not assigned to that actor. If the task is completed, the specified objective is reached.

  A task is done in an *_environment*. Objects in this environment are available for the actor to use in doing the task.

  A task emits a task result. This may be a completion signal, and/or a assigned task result, and/or an object (deliverable, such as a report). This is not expressed in the UML model, as it models *the dossier function* (p. 16).

  The task always has a completion state *_task.completion-state*; it is succesfully performed or not. Based on completion states, new tasks can be scheduled in the play.

  Subtypes are:

  - *_unit-of-study* — Unit of study is modelled as a task, because it can be made part of a play (a play within a play!). The unit of study does however not describe the operations to be performed, but 'packs' the task related objects into a reusable composite.
  - *_activity-list* — An activity list is build out of 1 or more activities, that have to be perfomed and finished succesfully (“completed”). The

activities are taken as a choice or as a sequence: a choice means that one or more of the tasks specified must be performed; a sequence means that all tasks must be perfomed in the specified order.

- *_activity* — An activity is the description of what to do and how to do it.

  The *_activity.completion-type* is any of the enumeration: unrestricted / user-choice / property-set.

- *_level* — A task level is the set of prerequisites and objectives of a task. This may be a learning task (student must write paper), a staff task (e.g. tutor must assesses student), or a task in general (person must buy book in store).

  The level may be any of:

  - a prerequisite, i.e. a formal or informal specification of an entry level, needed to perform a task.
  - an objective, i.e. a formal or informal specification of an exit level, reached by succesfully perform a task. Objectives may be set for all roles (learner and staff).
  - a learning objective, i.e. a particular kind of objective, which defines the educational goal of the task. Learning objectives are only set for the learner role.

  The *_level.type* is any of the enumeration: skill / knowledge / insight / attitude / competence / situational / other.

- *_environment* — An environment is a collection of objects that are required or useable for performing a task. Environments may include other environments. See also the *object diagram* (p. 12).

_component — UML — Components and references

The EML elements are based on some basic abstract classes that mainly implement the reference mechanism that underlies reuse. In EML there are two kinds of references:

- References to inherently internal objects, such as a reference to parts of the text (a figure, a formula, a table, a lemma), but also parts of a unit of study (a learner or staff, a role, an activity structure, a play or a set of conditions). These references are based on an identifier value, a unique name.
- References to inherently external object, such as units of study and knowledge-objects. These references add versioning information.

Note 2	In EML 1.0, for practical purposes identifiers have been separated into local and global identifiers. Conceptually, these are equivalent and the distinction is not made in the UML representation.

The UML model classes are described here.

- *_identifiable* — An object that is identifiable and therefore may be referenced. For example, a *_figure*, an *_article*, a *_lemma*, a *_role*.

  Subtypes are:

  - *_reusable* — a reusable object. This object may be stored within a large repository of objects and referenced for inclusion. Examples are all questions, *_bookmark*, *_section*, *_prerequisite*.

    Subtypes are:

    - *_component* (p. 10) — A component is a typed reusable object. Though reuse is inherent by technical nature (it has all features needed for reuse), component reuse may be constrained by design. In that case, it should not be reused as the contents of this component would be invalid or meaningless in a different context. For example, an *_environment*, any object in the environment, *_activity*.
- *_reference* — a reference to any instance that is not a component. Examples are: reference to roles, activity-structures, play, articles, books and chapters.

  Subtype is:

- *_reuse* — A reference to a reusable object. This reference implies a specification of what version is intended, and which (minor) changes to that version are acceptable before the link is deemed invalid (because the contents of that component may have changed too much).

_object — UML — Object diagram

Objects are items that are part of the environment of a *play* (p. 7). Here we specify the different object subtypes. The questionnaire is specified separately because it has a rather complex structure and is typically processed in a very specific way.

The classes that make up the object model are described here.

- *_object* (p. 12) — An object is any internal or external *_component* (p. 10) that can be part of one or more environments. Typical objects are books, articles, websites, software tools, CBT, communication tools.

  All objects have a meta-data specification.

  Subtypes are:

  - *_tool-object* — This object is a representation of a tool, available on the internet, that can be used in performing a task.
  - *_announcement-object* — An announcement object is a formal registration showing what kind of messages may be sent to, or received from, other actors by the person to which the object is assigned.

- *_mail-object* — object that defines a communication channel using mail.
- *_conference* — object that defines a conference. By default the conference is asynchronous

  Subtypes are:

  - *_synchronous-conference* — object that defines a synchronous conference.

    The *_synchronous-conference.medium* is any of the enumeration: text / video / audio / face2face / animated

- *_role-information-object* — This object specifies what kind of information should be shown about an actor in the unit of study.

  The *_role-information-object.show* is any of the enumeration: name / street / zip / city / country / email / telephone / roles / misc.

  The *_role-information-object.monitor-access* is any of the enumeration: first / last / all.

- *_knowledge-object* — An object that contains information to be used in performing activities.
- *_personal-object* — An object representing the assets that are available to the actor.
- *_index-search-object* — This object shows what kind of searches can be performed on the electronic materials that make up the unit of study.

- *_timespan* — A span of time, i.e. a combination of minutes, hours, days or the like.
- *_access-code* — Any access code.

_metadata — UML — Metadata diagram

Meta-data is data about data, which makes it possible to retrieve the object from a large collection. It holds information on any kind of component that is reusable. The meta-data diagram shows all properties and associations of a meta-data object.

The classes that make up the meta-data model are described here.

- *_metadata* (p. 13) — A complete description of a part of the EML that provides vital information for reuse of that component. The *_metadata.object-type* is any of the enumeration: electronic / nonelectronic / mix.
- *_status* — The status of the component. The status is typed and described.
- *_copyright* — Copyright information in terms of the year, owner and description.
- *_contributor* — Contributor to the component. The contributor is typed as e.g. "reviewer" or "author", and described.

_questionnaire-object — UML — Questionnaire-diagram

A questionnaire is a list of self-tests and interactions that are dealt with as a whole, and processed in a particular way. Questions are modelled as separate items.

The classes that make up the questionnaire model are described here.

- *_questionnaire-object* (p. 14) — A questionnaire, i.e. a set of questions, with specifications on how to present the question and how to deal with responses. The questionnaire consists of a set of questionnaire-items.
- *_test-group* — A group of questions to be handled as a whole, such as a complete test. The difference with a questionnaire is that the latter may be processed in a particular way, and to this end it holds the options and declaration of processing strategy. The test group is typically a set of questions that cover a particular field, and that must be handled in accordance with the processing defined on the containing questionnaire.
- *_questionnaire-item* (abstract) — A single modelled "question". It is the common supertype of question types such as MC, short-answer and true-false. Any question is represented texually. It has meta-data (and therefore can be part of a repository of questions). Questions may have hints and feedback information, and the score of the question is stored in a named property.

  Subtypes are:

  - *_matching-question* — This question type requires the user to match items taken from two categories to be placed in the right order, every n-th item in the first category being associated with every n-th item in the second category.

    - o _*short-answer-question* — A question which requires a short answer to be specified (for example, typed in). The answer is matched against one or more answer patterns. All matching patterns are correct
    - o _*sequence-question* — This question type requires the student to place a number of items in the correct sequence. First item is the correct one.
    - o _*mc-question* — A multiple-choice question, where the first answer is correct, and all next answersare incorrect.
    - o _*mr-question* — Multiple response question. Much like the MC-question, but this question type allows for more than one (rather than only one) correct answer to be specified
    - o _*true-false-question* — A question that represents one or more statements. These statements are true or false. The user must determine truth or falsity. The way the truth is represented is expressed by the options-representation attribute. Values are: true / right / correct
- _*score-condition* — This is a specification of how to process the collective result of answering all questions. If the number of correct answers is within the specified range, set the target-property given. Optionally, send a notification.
- _*notification* — A message sent to a particular actor, optionally requiring that actor to perform an activity.
- _*pattern* — A specification of a string pattern.
- _*item-pair* — Represents a pair of items, taken from the first and second category in a _*matching-question*.

_dossier — UML — Dossier diagram

Every actor has a dossier. Dossier information is available as a set of singular _*property* instances or as groups of properties. Properties for actors are declared as part of the _*role*. Properties receive a value during the run. An actor may also have properties that are not associated with the role (and therefore a run), but that are personal and persistent thoughout all runs of all units of study that she/he is involved in.

There is also a dossier that is shared by all actors in a role within a particular unit of study. The set of properties within that dossier typically concerns information that is role-independent but should not be recorded in the unit of study. An example is a list of Internet search systems (URLs). This list will be compiled as a property-group, with a separate property for each URL.

The classes that make up the dossier are:

- Role — A role defined for one or more actors.
- Property — An association between a name and value, associated with all actors in a role.
- Role.property — a property of a role.

_activity-logger — UML — Activity logger

The activity logger is an object that is used to record activity results. It determines if the activity is done and accesses the dossier to record the results.

- *_activity-logger* (p. 17) — an object that logs an activity by getting or setting a value in order to 1) determine that the activity is done ([activity completion-type=property-set] enumerated value) or 2) store the result of

the activity. This class references *_charseq*, *_property-accessor*, *_calculation* (p. 18) and *_activity* are treated in separate diagrams.

_calculation — UML — Calculation diagram

A calculation is an object that generates a value by calculation or comparison It accesses values specified in the UOS as well as those in the dossier.

The classes that make up the textual diagram are:

- *_operand* — A part of a calculation. In "A + B", A and B are operands.

  Subtypes are:

  - *_expression* — a typed value or value list. This inludes the subtypes integer, real, file, and such.

    Subtypes are:

    - *_calculation* (p. 18) — A calculation, which emits a value. The type is any of the enumeration: is / is-not / sum / subtract / multiply / divide / greater-than / less-than
    - *_logical* — a boolean comparison. The type is any of the enumeration: and / or / not.
    - *_time-accessor* — an accessor for timing information. The [time-accessor task-start] is a *_task* instance which is timed. If none specified, the accessor returns the current time.
    - *_role-accessor* — an accessor for role information. The [role--accessor role] is a *_role* which is filled with a number of actors that is expressed as a proportion.

- o *_value* — a typed value or value list. This includes the subtypes integer, real, file, and such.
- o *_task* — see *play diagram* (p. 7).
- o *_property-accessor* — see *textual diagram* (p. 19).

_textual — UML — Textual diagram

Textual objects may occur as part of a sequence of paragraph-level instances. These instances, when rendered, constitute a structured text. This sequence may contain paragraphs, lists, graphics, tables and other such objects.

Note 3	Textual and *inline* (p. 20) objects all may have content, that is: the inner text stored within the outer text element. Note that the typing mechanism is based on where a text object may occur (*substitution inheritance*) and not what content a text object may have.

The classes that make up the textual diagram are:

- *_textual* (p. 19) — An object that may occur within a sequence of paragraph-level objects.

  Subtypes are:

- *_p* — A single paragraph. Instances of this class have inline content.
- *_emphasis* — An emphasises piece of text. Instances of this class have inline content.
- *_graphical* — the representation of a figure or formula with caption, as expressed by the type attribute. This class is composed of a *_graphic-source* class, see *inline diagram* (p. 20).
- *_table* (p. 21) — A table. See *table diagram* (p. 21).
- *_literature* (p. 22) — A bibliographical reference. See *literature diagram* (p. 22).
- *_list* — a list. The list consists of *_list-item* components.
- *_lemma* — A lemma, i.e. an association between a term and a description of the term.
- *_internet-source* — A resource available on the internet, accessible by url..
- *_special* — A special part of a text, which may be typed and may be hidden when a condition is set that hides the associated content-type.
- *_code-line* — a single "line" of programming code. Instances of this class have string content.
- *_av-object* — an audio-visual component within a textual structure.

  _av-object.type is any of enumeration: image / animation / running-video / story-board / speech / music / source-sound / mix

  _av-object.nature is any of enumeration: live / streaming-file / file

- *_section* — a reusable portion of a text.
- *_questionnaire-item* — a single question.
- *_comment* — Comments provided by the author of the specification.
- *_property-accessor* — access to a property. The type of access is either 'get' or 'set'. The property is known by *_property-accessor.name*. When the property is set, any notifications specified will be activated.

_inline — UML — inline diagram

Inline objects may occur within a sequence of *characters*. This class covers concrete classes like bookmarks, terms, and emphasized phrases.

The classes that make up the inline diagram are:

- *_inline* (p. 20) — An object that may occur as part of a character sequence.

  Subtypes are:

  - *_charseq* ISA *_inline* (p. 20) — a sequence of 0..n characters.
  - *_property-accessor* — see *textual diagram* (p. 19)
  - *_emphasis* — see *textual diagram* (p. 19).
  - *_bookmark* — see *textual diagram* (p. 19).
  - *_special-inline* — a specially marked part sequence of inline objects. The content type and flow position of this part may be explicated and may affect the rendition.
  - *_graphic-source* — a reference to a source of a figure or formula.
  - *_term* — a term, as possibly defined in a corresponding *_lemma*.
  - *_comment-inline* — Comments entered inline.
  - *_internet-ref* — reference to an internet location through URL.

_table — UML — CALS Table diagram

A table is part of any EML that intends to incorporate textual structrures. The model therefore must provide for the *_table* (p. 21) class and constituent classes that represent table headers, rows and cells.

The diagram shown attempts to clarify the complex relations between the classes that cooperate in visualizing a CALS table. The classes are not described further here.

_literature — UML — Literature

_literature (p. 22) models a bibliographic reference. It is part of *_textual* (p. 19) description.

The classes that make up the literature diagram are:

- *_literature* (p. 22) — A single bibliographic reference . This is an abstract class.

  Subtypes are:

  - *_book* — A book.
  - *_article* — An article.
  - *_chapter* — A chapter within a book.
- *_author-role* — an author and her/his role.