IMS Learning Design Information Model

Version 1.0 Final Specification

Document Name: IMS Learning Design Information Model

Date: 20 January 2003

Table of Contents

1. Introduction

1.1 Overview

This document contains the Learning Design Information Model. It represents an integration of the Educational Modelling Language (EML) work, submitted to the Learning Design working group (LDWG) by the Open University of the Netherlands [LD1], and existing IMS Specifications, notably Content Packaging [LD2] that this specification extends and builds on, but also Meta-Data [LD3] and Simple Sequencing [LD4].

A key task of the LDWG was "the development of a framework that supports pedagogical diversity and innovation, while promoting the exchange and interoperability of e-learning materials".

The OU Netherlands carried out extensive examination and analysis of a wide range of pedagogical approaches prior to developing EML as a relatively concise 'meta-language' that could capture this diversity. Regardless of the pedagogy involved, in practice every learning design came down to: a Method prescribing various Activities for learner and staff Roles in a certain order. Each activity refers to a collection of specific objects and services (called the 'Environment') needed to perform the activity. In order to support the description of individualized learning designs, learner Properties, Conditions, and Notifications are needed. The designs which can be described by this meta-language might involve a single user or multiple users; the learning and instructional designers and providers might take a behaviorist, cognitivist, constructivist, or some other approach; they might require learners to work separately or collaboratively, but the OUNL studies found these could all be captured in terms of a Method containing Roles, Activity-structures, and Environments and a number of other concepts elaborated around these [LD5].

This meta-language approach has the great advantage that, rather than trying to capture the terminology of each approach, which could lead to an indefinitely large vocabulary or set of vocabularies, a single relatively small vocabulary can be used to express what, in concrete terms, each of these approaches asks of the learners and support staff involved. It also allows different pedagogical approaches to be integrated into a single 'learning design' where different approaches may be appropriate for different types of learners.

The language also supports mixed mode delivery ('blended learning'), enabling traditional approaches such as face-to-face teaching, the use of books and journals, lab work, and field trips to be also specified as learning activities and combined with ICT supported learning. What it brings to mixed mode teaching is the ability to specify both kinds of learning in a unit of learning that is itself in digital form.

By being expressive of pedagogical approaches, rather than prescriptive, the language facilitates the development of new pedagogical approaches. To the developer of learning technologies, the meta-language enables pedagogical diversity to be supported through the implementation of a single engine, rather than either having to implement multiple engines for each approach, or remaining 'pedagogically agnostic' by providing no specific support for any.

The remainder of this document sets out the vocabulary of this language, its syntax expressed in terms of its information structures, and its semantics, taken to mean how designs specified in this language should be interpreted in order to give rise to learning activities when instantiated and engaged with users.

1.2 Three Levels of Implementation and Compliance

Learning Design specifies three levels of implementation and compliance. This document is therefore partitioned to reflect this. However, each level is mapped to separate XML Schemas.

Learning Design Level A includes everything described so far. It thus contains all the core vocabulary needed to support pedagogical diversity. Levels B and C add three additional concepts and their associated capabilities in order to support more sophisticated behaviors.

Learning Design Level B adds Properties and Conditions to level A, which enable personalization and more elaborate sequencing and interactions based on learner portfolios. It can be used to direct the learning activities as well as record outcomes. The separation of Properties and Conditions into a separate Schema also enable it to be used independently of the rest of the Learning Design Specification, typically as an enhancement to IMS Simple Sequencing.

Learning Design Level C adds Notification to level B, which, although a fairly small addition to the specification, adds significantly to the capability, but potentially also to the implementation task where something similar is not already in place.

The approach taken in this specification is therefore not to define a single large schema with a core of mandatory elements and numerous optional elements, but rather to define a complete core that is yet as simple as possible, and then to define two levels of extension that capture more sophisticated features and behaviors.

We expect compliance to be both more rigorous and yet flexible, with Level A being relatively easy to achieve. The option lies with whether and when to implement the higher levels.

Complete compliance will then be expected of implementing systems for any given level. With respect to Learning Design, instance documents implemented in compliance with this specification, do not need to implement every element, so a distinction is made between compliance of content and compliance of supporting systems. Optional elements apply to document instances; systems must implement all the specification at a stated level and thus should be able to run all instances of that level whatever options these choose to make use of. Learning design instances that comply with this specification must be validated by a parser using the supplied XML schemas. However they should specify which Learning Design Level they expect a compliant runtime system to support, so that systems which do not specify all three levels can determine whether they are capable of running any particular learning design instance.

1.3 Learning Design and Other Specifications

Learning Design can be considered as an integrative layer to many existing specifications. The IMS Learning Design Specification makes use of, includes, or is extendable with the following specifications:

- **IMS Content Packaging.** The IMS Learning Design is preferably integrated into an IMS Content Package to create a so called ‘Unit of Learning’. This is explained later in the text [LD2].
- **IMS Simple Sequencing.** The IMS Simple Sequencing Specification can be used to (a) sequence the resources within a learning-object and (b) sequence the different learning-objects and services within an environment. This works in a similar way as the integration of Simple Sequencing in the organization of items in an IMS Content Package. The Simple Sequencing elements can be namespaced into the ‘any’ place holders of the elements learning-object and environment. These place holders are specified in the binding of IMS LD [LD4].
- **IMS/LOM Meta-Data.** Placeholders for meta-data are on various structures within the IMS Learning Design. IMS/LOM Meta-Data can be included at these places [LD3].
- **IMS Question and Test Interoperability.** The IMS QTI can be integrated in two ways. The first way is to integrate QTI elements into the element context environment/learning-object as a separate schema. This is semantically seen as the correct place for tests. Test can than be connected to learning-activities which provide the instruction to complete the test that is present in the environment. Also, the currently used methods, integrating them into IMS Content Packaging as specific Resource types or as separate files are still supported [LD6].
- **IMS Reusable Definition of Competency or Educational Objective (RDCEO).** Learning Objectives and Prerequisites can refer to resources that are defined according to this specification. This is seen as a further refinement when needed. Also supported are simple resources (e.g., textual descriptions) of the learning objectives through the standard ‘item’ mechanism as can be found in IMS Content Packaging [LD7].
- **IMS Learner Information Package.** The structure of IMS Learning Design properties can be mapped fully to the IMS LIP [LD8].
- **IMS Enterprise** can be used for mapping learners and support staff to roles when instantiating a learning design [LD9].
- With the IMS Learning Design Specification it is possible to include **SCORM** content within a learning design. It would be necessary to have its type set and the runtime system would have to be able to deliver and manage SCORM content [LD10].

The standard way to include specifications is through the mechanisms of XML Namespaces. All IMS specifications have their own namespace.

1.4 Scope and Context

This document is the IMS Learning Design Specification. As such it will be used as the basis for the production of the following documents:

- IMS Learning Design XML Bindings for level A, B, and C.
- IMS Learning Design Best Practice and Implementation Guide.

Taken together, the three documents constitute the IMS Learning Design Specification.

This information model describes a model for learning design that contains three primary components:

- A **conceptual model** that presents the vocabulary, the functional relationships between the concepts, and the relationship with IMS Content Packaging. The conceptual model is described from an overall (level C) perspective.
- An **information model** that describes the IMS Learning Design elements for respectively the levels A, B, and C. Also the restricted conceptual model for the different levels is presented.
- A **behavioral model** which describes a set of runtime behaviors that delivery systems must implement.

1.5 Nomenclature

EML	Educational Modelling Language
IMSCP	IMS Content Packaging Specification
IMSMD	IMS/LOM Meta-Data Specification
IMSQTI	IMS Question and Test Interoperability Specification
LOM	Learning Object Metadata (IEEE 1484.12.1 – 2002)
PCDATA	Character Data
UML	Unified Modeling Language
URI	Universal Resource Identifier
W3C	World Wide Web Consortium
XML	Extensible Mark-up Language

1.6 References

[LD1] EML reference manual (http://eml.ou.nl)

[LD2] IMS Content Packaging Specification (http://www.imsglobal.org)

[LD3] IMS Learning Resource Meta-Data Specification (http://www.imsglobal.org). See also IEEE LTSC (http://ltsc.ieee.org) LOM (Learning Object Metadata)

[LD4] IMS Simple Sequencing Specification (http://www.imsglobal.org)

[LD5] Modelling units of study from a pedagogical perspective: the pedagogical metamodel behind EML, Koper E.J.R., 2001: (http://eml.ou.nl/introduction/docs/ped-metamodel.pdf)

[LD6] IMS Question and Test Interoperability Specification (http://www.imsglobal.org)

[LD7] IMS Reusable Definition of Competency or Educational Objective (RDCEO) Specification (http://www.imsglobal.org)

[LD8] IMS Learner Information Package Specification (http://www.imsglobal.org)

[LD9] IMS Enterprise Specification (http://www.imsglobal.org)

[LD10] ADL SCORM (http://www.adlnet.org)

[LD11] CLEO, content aggregation (http://www.lsal.cmu.edu/lsal/expertise/projects/cleo/report20010701/working/aggregation.html)

[LD12] OASIS DOCBOOK (http://www.oasis-open.org/docbook/documentation/reference/html/docbook.html)

[LD13] Unified Modeling Language (http://www.omg.org/technology/documents/formal/uml.htm)

[LD14] W3C HTML 4.0 specification (http://www.w3.org/TR/REC-html40/struct/global.html#h-7.5.2) (for the definition of the ‘class’ attribute)

[LD15] IETF (http://www.ietf.org), relevant specifications: URI, ftp, news, smtp, http

[LD16] W3C (http://www.w3c.org) consortium for Web-related interoperability specifications: HTML, XHTML, XML 1.0, XML schema, XML namespaces, XSLT

[LD17] Vogten, H., Verhooren, M., & Koper, E. UML diagrams (http://eml.ou.nl/introduction/docs/uml.pdf)

2. Conceptual Model

2.1 Objective of Learning Design Specification

Note: The objective of the Learning Design Specification is to provide a containment framework of elements that can describe any design of a teaching-learning process in a formal way. More specifically, the Learning Design Specification meets the following requirements:

R1. Completeness: The specification must be able to fully describe the teaching-learning process in a unit of learning, including references to the digital and non-digital learning objects and services needed during the process. This includes:

- Integration of the activities of both learners and staff members.
- Integration of resources and services used during learning.
- Support for a wide variety of approaches to learning.
- Support for both single and multiple user models of learning.
- Support mixed mode (blended learning) as well as pure online learning.

R2. Pedagogical Flexibility: The specification must be able to express the pedagogical meaning and functionality of the different data elements within the context of a unit of learning. It must be flexible in the description of all different kinds of pedagogies and not prescribe any specific pedagogical approach.

R3. Personalization: The specification must be able to describe personalization aspects within a learning design, so that the content and activities within a unit of learning can be adapted based on the preferences, portfolio, pre-knowledge, educational needs, and situational circumstances of users. In addition, the control over the adaptation process must be given, as desired, to the student, a staff member, the computer, and/or the designer.

R4. Formalization: The specification must describe a learning design in the context of a unit of learning in a formal way, so that automatic processing is possible.

R5. Reproducibility: The specification must describe the learning design abstracted in such a way that repeated execution in different settings with different persons is possible.

R6. Interoperability: The specification must support interoperability of learning designs.

R7. Compatibility: The specification uses available standards and specifications where possible, mainly IMS Content Packaging, IMS Question and Test Interoperability, IMS/LOM Meta-Data and IMS Simple Sequencing.

R8. Reusability: The specification must make it possible to identify, isolate, de-contextualize and exchange useful learning artefacts, and to re-use these in other contexts.

2.2 Conceptual Model

The conceptual model is expressed as a set of UML class models and a definition of the vocabulary used. It represents the overall conceptual model; it is not yet divided into levels A, B, or C, and it contains some elements that are not expressed in the information model but are needed for the better conceptual understanding. There are three basic models: an aggregation model, a structure model and a model representing the integration of the learning design within IMS Content Package to obtain a so-called ‘unit of learning’. The models are all provided before the vocabulary is defined.

2.2.1 Semantic Aggregation Levels in Learning Design

The first figure represents the conceptual model of the semantic aggregations levels in the Learning Design Specification [see also LD11]. The diagrammatic notation conforms to UML version 1.4 [LD13], and represents only aggregation relationships (including compositions) and the specializations of abstract classes (‘types’).

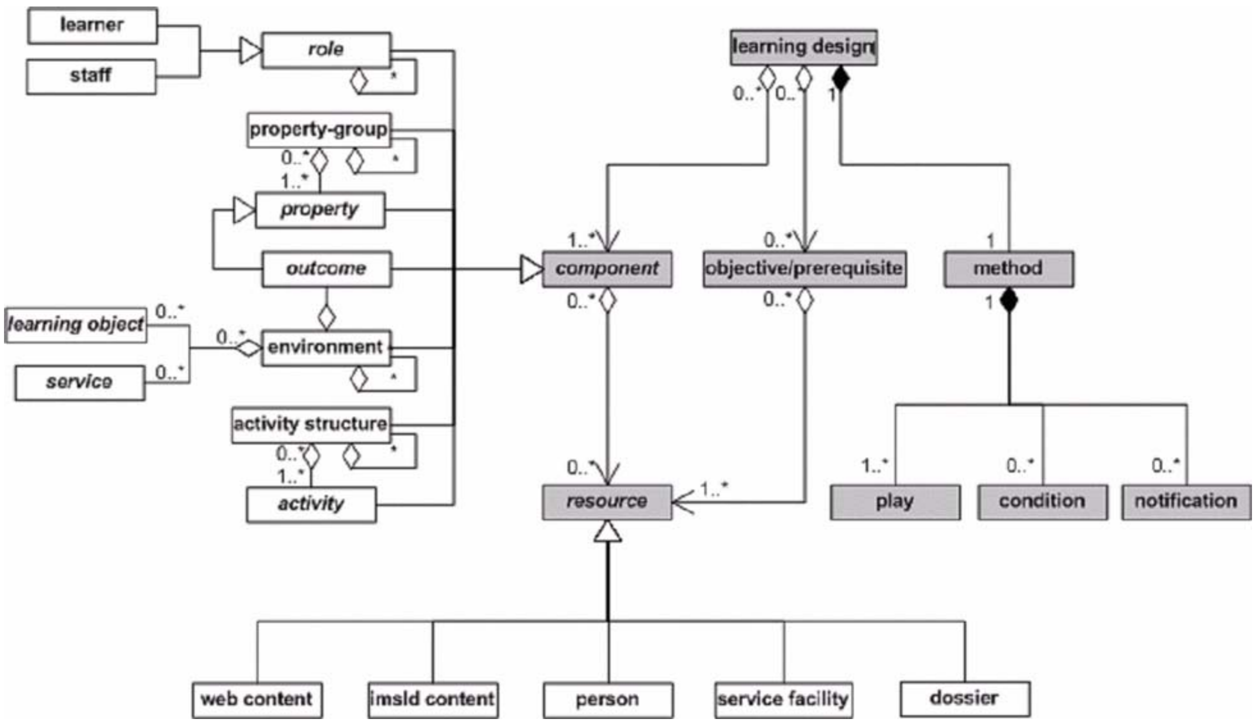


Figure 2.1 - Semantic aggregation levels in the Learning Design Level C specification (grey coloring is only used to increase the readability).

The model shows that learning design provides a semantic view of a collection of *resources* on one hand, and on the other hand it integrates a *method*, specifying the dynamic aspects of the learning design.

The model shows three levels of semantic aggregation (the three horizontal layers of grey colored classes). The semantically highest level is the *learning design*, it aggregates a collection of *components*, *objectives/prerequisites* (short for: learning objectives & prerequisites), and a *method*. The lowest level of aggregation are the *resource*, *play*, *condition*, and *notification*. The *resources* are aggregated into *components* and *objectives/prerequisites*. The *plays*, *conditions*, and *notifications* are aggregated into the *method*.

A *component* can be one of seven different types: role, property group, property, activity structure, activity, environment, or outcome. With the exception of outcome, these are all elements in the LD Information Model. Role can be one of two types: learner or staff.

A *resource* can be one of five different types: web content, imsld content, person, service facility, or dossier. These resources can be referenced from a Learning Design but are not explicitly part of the Information Model.

Specific types of components are bound to specific types of resources. The moment in time when the resource is bound in the learning design differs:

Component	Binds to Resource Type	Moment of Binding
Role	Person	during instantiation and during runtime
Objective/prerequisite	Web content	during design, instantiation or during runtime
Objective/prerequisite	Imsld content	during design
Property	Dossier	during instantiation
Learning object	Web content	during design, instantiation or during runtime
Learning object	Imsld content	during design

Component	Binds to Resource Type	Moment of Binding
Service	Service facility	during design, instantiation or during runtime
Activity	Web content	during design, instantiation or during runtime
Activity	Imsld content	during design

Example: The textual description of a learning objective for instance, can be written during the design and delivered as a fixed resource with the learning design. However also an absolute URL can be provided to a location where the file can be edited at any time.

The resources that are not bound during design time are not part of this specification. This applies to concrete persons and to the actual dossiers of persons. However, the roles that persons can take into the learning design and the properties that have to be present in the dossiers are part of the learning design.

The model also shows that the components, objectives/prerequisites, and resources are independent of the learning design. They can be referenced and used in many other learning designs. However the method has a composite relationship with the learning design meaning that it is an integral part of it and cannot stand on its own and cannot (easily) be re-used in other learning designs.

2.2.2 Conceptual Structure of the Learning Design

Another conceptual view of the learning design is provided in Figure 2.2. In this model, the emphasis is on the functional relationships between the classes.

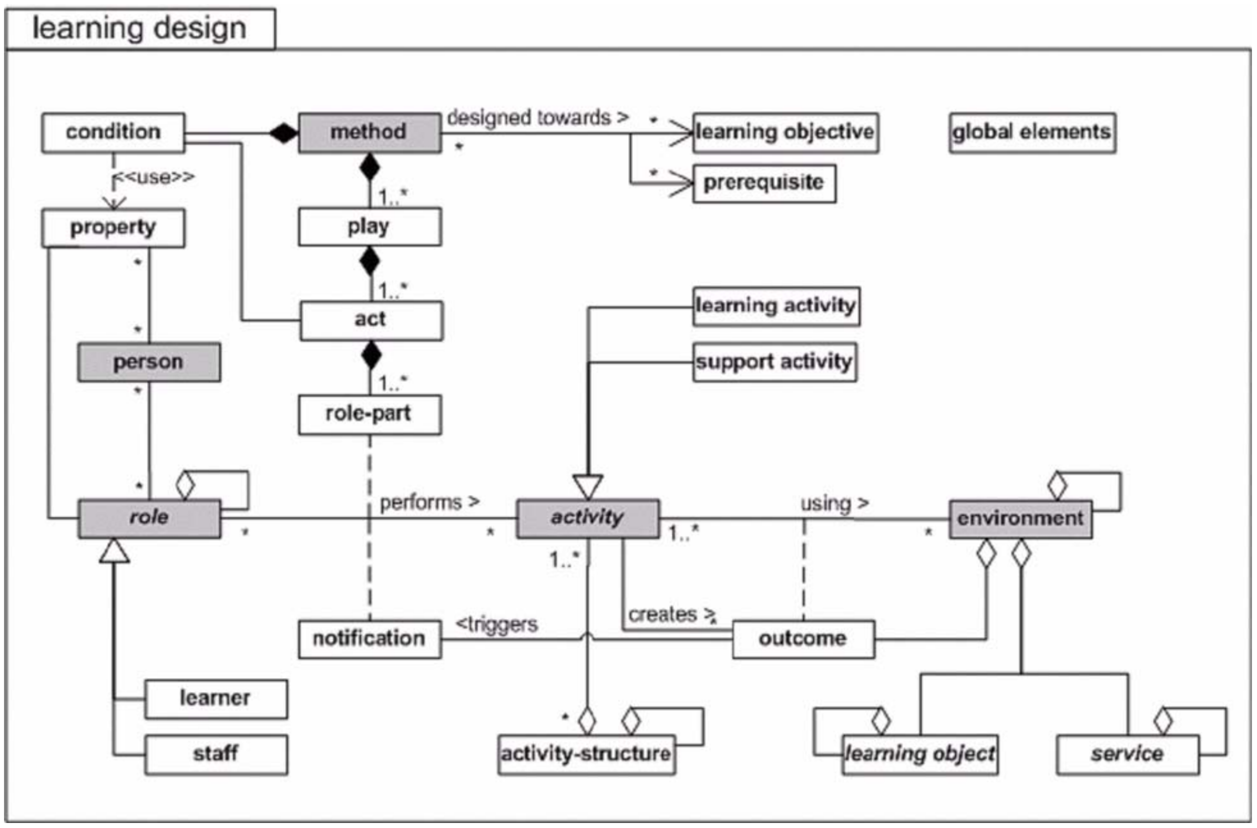


Figure 2.2 - Conceptual model of overall Learning Design (gray coloring is only used to increase the readability).

The core concept of the Learning Design Specification, as expressed in Figure 2.s, is that regardless of pedagogical approach, a *person* gets a *role* in the teaching-learning process, typically a *learner* or a *staff* role. In this role he or she works toward certain *outcomes* by performing more or less structured *learning* and/or *support activities* within an *environment*. The environment consists of the appropriate *learning objects and services* to be used during the performance of the activities. Which role gets which activities at what moment in the process, is determined by the *method* or by a *notification*. Note: most of the concepts mentioned above are reflected in the information model, but some only exist at the conceptual level (person, outcome).

The *method* is designed to meet *learning objectives* (specification of the outcomes for learners), and presupposes certain *prerequisites* (specification of the entry level for learners). The method consists of one or more concurrent *play*(s); a play consists of one or more sequential *act*(s) and an act is related to one or more concurrent *role-part*(s), each role-part associates exactly one role with one activity or activity-structure. The teaching-learning process is modelled in the method on the notion of a theatrical play. A play has acts, and in each act has one or more role-parts. The acts in a play follow each other in a sequence (although more complex sequencing behavior can take place within an act). The role-parts within an act associate each role with an activity. The activity in turn describes what that role is to do and what environment is available to it within the act. In the analogy, the assigned activity is the equivalent of the script for the part that the role plays in the act, although less prescriptive. Where there is more than one role-part within an act, these are run in parallel.

A method may, at level B, contain *conditions* (i.e., If-Then-Else rules that further refine the visibility of activities and environment entities for persons and roles), by defining Boolean expressions on their *properties*. A property can be grouped into property-groups. Properties can be of different types, representing respectively global versus local properties and personal versus role properties. This is explained later.

In order to enable users to set and view the level B properties from content that is presented to them, so-called *global elements* are present in the model. These global elements are designed to be included in any content schema through namespaces. Content that includes these global elements is called '*imsldcontent*'.

A *notification* is triggered by an outcome and can make a new activity available for a role to perform. The person getting the notification is not necessarily the same person who creates the outcome. For instance, when one student completes an activity (= an outcome), then another student or the teacher may be notified and set another activity as a consequence. This mechanism can also be used for learning designs where the supply of a consequent activity may be dependent on the kind of outcome of previous activities (adaptive task setting designs).

The explicit *roles* specified in this language are those of *learner* and *staff* roles. Each of these can be specialized into sub-roles, but no vocabulary is put forward for this. It is left open to the learning designer to name the (sub)-roles and specify their activities. For example, in simulations and games different learners can play different roles, each performing different activities in different environments.

Activities can be assembled into *activity-structures*. An activity-structure aggregates a set of related activities into a single structure, which can be associated to a role in a role-part. A structure can model a sequence or a selection of activities. In a *sequence*, a role has to complete the different activities in the structure in the order provided. In a *selection*, a role may select a given number of activities from the set provided in the activity-structure. This can, for instance, be used to model situations where students have to complete two activities, which they may freely select from a collection of e.g., five activities contained in the activity-structure.

Activity-structures can also reference other Activity-structures and reference external Units of Learning, enabling elaborate structures to be defined if required.

Environments can contain two basic types:

- Located *learning objects*, typically specified by a URL with optional metadata. A user may further classify these learning objects by means of the vocabulary provided in the IMS LOM Meta-Data (5.2 Learning Resource Type) or the generic 'class' attribute that is available on all elements. In EML [LD1], the learning objects are classified in the following types: knowledge-objects, tool-objects, and test-objects.

- Generic *services*. A service relates to a concrete service facility available at runtime. During design a service has no URL assigned to it, but must be given a URL when the Learning Design is instantiated at runtime. Examples of a Service include a discussion forum, chat rooms, monitoring tools, search facilities, etcetera. In Learning Design the conditions for setting up a service at runtime are specified at an abstract level. For example, for discussion groups it specifies which learning design roles have what type of access (participant, observer, moderator, etc.).

Note: if a discussion forum is to be used within a learning design, were it given a predefined URL, then all instances of the Unit of Learning that includes the learning design, wherever and whenever instantiated, would have the same one specific discussion forum. While this may lead to serendipitous learning, it is probably not what was intended by the learning designer! (However, should this be desirable, then a normal Resource element with a fixed URL can be provided.)

For this version of the specification, the types of services specified are limiting these to those that are now found in typical LMS systems. It is possible to inherit from a generic service and thus specify new types as extensions to the vocabulary. As many services are targeted for use in a specific instance of a learning design, the actual members will need to have been assigned to roles before the service is instantiated. As different roles may have different permissions within a service, there is the facility to specify these within the particular service definition.

EML included a complete content vocabulary based on the OASIS DOCBOOK specification [LD12]. For Learning Design it was decided not to include any content specification, but let the users of the Learning Design Specification decide which one to use. In order to allow for runtime interactions with the end users, specific *global learning design elements* are provided separately at level B which can be namespaced into any XML-based content schema. A suggestion is to use XHTML for content and to namespace the global elements of learning design into XHTML.

2.2.3 Unit of Learning = IMS Content Package + IMS Learning Design

The primary use of IMS Learning Design is to model *units of learning* by including an IMS Learning Design in a content package, preferably – but not necessarily - an IMS Content Package. It this specification it is assumed that IMS Learning Design is being used with IMS Content Packages to model units of learning. How this is done is explained in this section.

IMS Content Packages describe their contents in an XML document called the ‘package manifest’. The Manifest may include structured ‘views’ into the resources contained in that package; each ‘view’ is described as a hierarchy of items called an ‘organization’. Each item refers to a Resource that, in turn, can refer to a physical file within the package. It can however also refer to an external resource. Figure 2.3 depicts the entire IMS Content Packaging conceptual model.

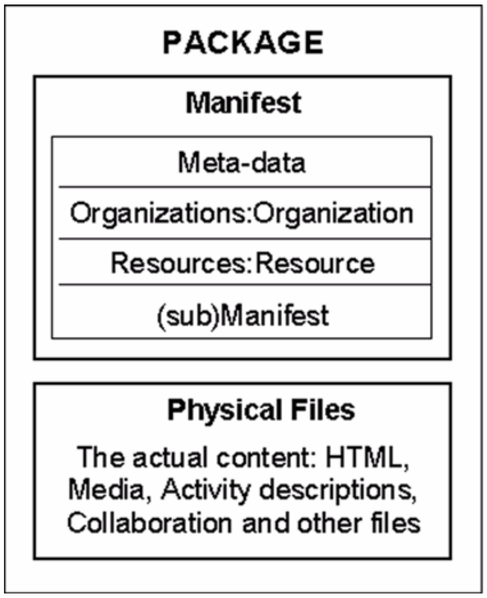

PACKAGE
Manifest
Meta-data
Organizations:Organization
Resources:Resource
(sub)Manifest
Physical Files
The actual content: HTML, Media, Activity descriptions, Collaboration and other files

Figure 2.3 - Structure of an IMS Content Package.

The Manifest is the information structure defined in the Content Packaging Specification. It is contained within a package as an XML file with a fixed, pre-defined name (imsmanifest.xml). This enables it to be found amongst the many other content files that may be contained in a package.

The integration of a Learning Design into the Content Packaging Structure is set out in the Figure 2.4.

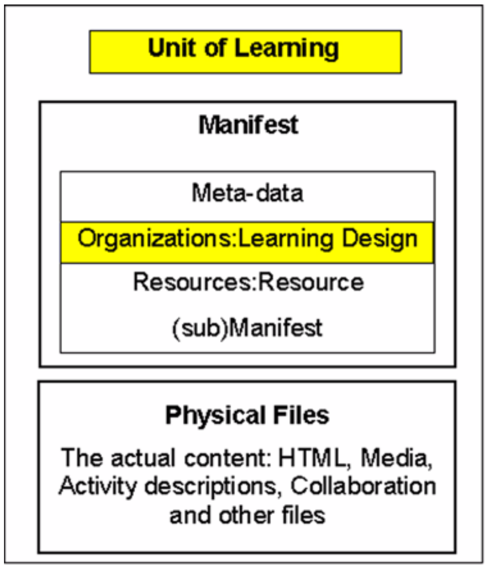

Unit of Learning
Manifest
Meta-data
Organizations:Learning Design
Resources:Resource
(sub)Manifest
Physical Files
The actual content: HTML, Media, Activity descriptions, Collaboration and other files

Figure 2.4 - The structure of a Unit of Learning, composed by including an IMS Learning Design within the Organizations part of IMS Content Packaging

To create a unit of learning, IMS Learning Design is integrated with an IMS Content Package by including the learning design element as another kind of organization within the <organizations> element, using the standard namespace for Learning Design. When the standard namespace is "[standard-namespace-for-learning-design]", then learning design elements are included as follows (ignoring irrelevant elements and attributes):

```
<manifest>
   <metadata/>
   <organizations>
      <learning-design xmlns="[standard-namespace-for-learning-design]">
         [add learning design elements here]
      </learning-design>
   </organizations>
   <resources/>
</manifest>
```

The italics have to be filled in with the appropriate namespace and elements respectively.

In a package that includes a learning design element, the optional organization element within organizations is ignored. This mechanism is in conformance with the extensibility mechanisms IMS Content Packages provide. If an organizations element contains a learning design element, any 'organization' element in the same organizations element is ignored and only the learning design element is read by the runtime system. Where other content organization elements are desired, they can be included in sub manifests, as sub packages may be aggregated in the same way as in normal content packages.

2.3 Conceptual Vocabulary of Learning Design

In this section an overview will be given of the basic conceptual terms of the Learning Design Specification.

Unit of Learning
A learning design is an integral part of any unit of learning. A 'unit of learning' is an abstract term used to refer to any delimited piece of education or training, such as a course, a module, a lesson, etc. It is noted that a 'unit of learning' represents more than just a collection of ordered resources to learn, it includes a variety of prescribed activities (problem solving activities, search activities, discussion activities, peer assessment activities, etcetera), assessments, services and support facilities provided by teachers, trainers and other staff members. Which activities, which resources, which roles and which workflow is dependent on the learning design in the unit of learning. A unit of learning can be modelled as an IMS Content Package by including an IMS Learning Design in the package.

An IMS content package is called a 'Unit of learning' if and only if it includes a valid IMS learning-design element in the organizations part of the package's manifest. A Unit of Learning includes a manifest, a learning design, resources, possible (sub-) manifests and physical files.

Learning Design
A learning design is a description of a method enabling learners to attain certain learning objectives by performing certain learning activities in a certain order in the context of a certain learning environment. A learning design is based on the pedagogical principles of the designer and on specific domain and contexts variables (e.g., designs for mathematics teaching can differ from designs for language teaching; designs for distance education can differ from designs which integrate face-to-face settings). Several hundreds of designs are described in literature, each based on different assumptions about the teaching and learning process [LD5]. In daily practice, most teachers and trainers apply their own principles of learning. This leads to a countless number of possible design solutions for the same content domain. In order to allow all these different designs to be effectively included into e-learning modules, the approach of a meta-language is taken, enabling the description of all kinds of learning designs without forcing a specific solution on the designers.

The Learning Design element is the root element for the Learning Design Specification. It includes the core set of elements added by the Learning Design Specification to the existing Content Packaging Specification. It provides a semantically structured view on the resources along with learning process information. The following terms are the key additions.

Learning Objectives
The learning objectives are the overall learning objectives to be attained by learners who complete the unit of learning. Learning objectives can be specified on several levels of detail. In IMS Learning Design, designers can choose to specify learning objectives at two levels, each with advantages and disadvantages. First, it is possible to define the learning objectives at the global level of the unit of learning. Second, it is possible to specify learning objectives for every single activity in the learning design. Designers can follow several approaches:

- define the learning objectives only at the level of the unit of learning as a whole, not indicating the sub-objectives of the individual learning activities or what they add to the overall objectives.
- define the learning objectives only per learning-activity and not globally for the unit of learning. The learning objective for the unit of learning is nothing more or less than the list of all the learning objectives specified in the different learning activities.
- define the learning objectives on both levels: the learning objectives at the unit of learning level can be described more abstractly than those at the activity level.

Prerequisites
The prerequisites specify the overall entry requirements for learners for doing the unit of learning. As with learning objectives, the prerequisites can be provided at the level of the unit of learning and/or for individual learning activities.

Learning objectives and prerequisites can be described using the IMS Reusable Definition of Competency or Educational Objective (RDCEO) format, but can also refer to simple resources (e.g., a text) with a description of the learning objective.

Components
These are the declarations of the different components that provide the 'building blocks' for the method section of the learning design. In Learning Design Level A these are: roles, activities, and environments In Learning Design Levels B and C these are: roles, properties, activities, and environments. The components are declared separately from the

method to avoid duplication in the method when using the same component more than once. The component and the method sections can be compared to a cooking recipe: the components are the list of ingredients and the method is the preparation instructions.

Roles
Roles allow the type of participant in a unit of learning to be specified. There are two basic Role types: Learner and Staff. These however can be sub-typed to allow learners to play different roles in certain types of learning activity such as task-based, role-play and simulations. Similarly support staff can be sub-typed and given more specialized roles, such as Tutor, Teaching Assistant, Mentor, etc. Roles thus lay the basis for multi-user models of learning.

The name a role is given depends on the pedagogy and setting used. In some instances a learner is called a ‘student’ in others a ‘participants’. The names of staff roles are even more variant, e.g., teacher, trainer, tutor, facilitator, mentor, assessor. Every role can have its own ‘title’ which provides the name for it.

At runtime more than one user can be assigned to the same role, however restrictions can be set on the maximum and minimum number for each role. In this sense roles can be used for grouping purposes.

Properties
Properties are only available in level B and C of the Learning Design Specification. They form the basis on which to build user and role dossiers and portfolios. Properties are an essential part of monitoring, personalization, assessment and for user-interaction. Learning Design supports five types of properties: local properties, local-personal properties, local-role properties, global-personal properties and global properties. Furthermore, properties can be grouped to e.g. create forms. The local properties are declared in the learning design. The global properties are expected to be declared externally, but a mechanism is included to declare new global properties when they are not present.

Global Elements
In Levels B and C, in order for users to be able to set and view properties during the teaching and learning process, global elements are provided as a separate part of the IMS Learning Design Specification. There are four global elements: set-property, view-property, set-property-group, and view-property-group. The set-property element enables a user control in a Web (or other) interface to change the current value of a specific property. The view-property element shows the property value of a selected property to the user as part of the learning content. The set-property-group and view-property-group elements do the same for a set of properties. Global elements are not a part of the learning-design tree, but are provided separately. They are designed to be included in any XML content schema by use of XML namespaces (e.g., for inclusion in XHTML). Without these elements it is not possible to access or set the properties. Content that uses global elements must be given a specific resource type (referring to the type attribute of the resource element of IMS Content Packaging), namely ‘imsldcontent’ instead of ‘webcontent’. In the future, the set of global elements available for inclusion in content schemas can be extended.

Activities
Activities are one of the core structural elements of the ‘learning workflow’ model for learning design. They form the link between the roles and the learning objects and services in the learning environment. They describe the activities a role has to undertake within a specified environment composed of learning objects and services. They also specify their termination conditions and the actions to be taken on termination. There are two basic types of activities: learning activities and support activities. A learning activity is directed at attaining a learning objective per individual user. Any user performs a learning activity only once (until completion). A support activity is meant to facilitate a role performing one or more learning activities. More than one person can be assigned to a role in runtime. In practice this means that a support activity has to be performed as many times as there are users in the supported role.

Activities can be aggregated into an activity-structure which provides the mechanisms to structure activities and referenced units of learning into a sequence or a user-selection.

An activity references the environment in which the activity must be executed. For the execution of any activity, a user needs, at a minimum, an activity description and, optionally, an environment with the learning objects and services needed to perform the activity.

Learning Activity
A Learning Activity consists of a single *activity-description* and several optional elements. The activity-description is the actual cue given to the user (rendered in the user-interface) to describe the activity to be performed by the user. In

most cases the activity-description is a text (of type webcontent). In other cases it can be an audio-file (webcontent), a video file or any other cue to the user. Whatever forms it takes, the activity-description is referenced via an <item> element, derived from Content Packaging, referencing a resource element in the content package.

In addition to Environment reference(s), the other optional elements include:

title, IMS metadata, learning-objectives, prerequisites (see above), and the new elements: *complete-activity* (which specifies when an activity is completed, in Level A either by user-choice, or on reaching a time-limit, and extended by Level B with *when-property-value-is-set*), and *on-completion* (which specifies actions that are to be executed on completion of the activity).

In level A, *on-completion* contains only one element, *feedback-description,* which references content to be displayed to the user when they finish.

on-completion is further extended at level B by the *change-property-value* element, and at Level C by the *notification* element).

Support Activity
A Support Activity consists mostly of the same elements as a Learning Activity, but without the *learning-objectives, prerequisites,* and with a *role-ref* element added. The role-ref element indicates who will be supported by this activity. The supported role can have more than one person in the role. This means in practice that the support activity has to be repeated for every user in the supported role before it is completed. This is a key difference from learning activities, which is only performed once. Example: a staff role has the support activity to grade reports made by persons in the learner role named ‘student’. Every person being a student creates his/her own report. The tutor grades every report (repeating the ‘grade report’ support activity).

Activity-Structure
An Activity-structure in turn consists of references to one or more of:

- A Learning Activity
- A Support Activity
- An (sub) Activity-structure
- Another (separate) Unit of Learning

In the case of the Unit of Learning, the reference is an HREF to the Unique Resource Identifier (URI) of the unit of learning. This URI can be any worldwide unique identifier, including a URL (as it is used in the W3C namespace specification to identify unique namespaces). When using IMS Content Packaging, this means that it refers to the ‘identifier’ attribute of the manifest, which must be a worldwide unique identifier of some format.

Like the single Activities, an Activity-structure may reference one or more environments. This allows for learning design models where a series of different activities are performed within the same environment. When an activity-structure references one or more environments, then these will overrule the environments specified within the referenced activities.

The environments will not be inherited between hierarchical levels of the Activity-structure, allowing environments to be omitted as well. As a consequence, for each hierarchical level of the Activity-structure the appropriate reference to an environment has to be made and possibly repeated.

A structure may contain information. This provides a CP Organization/Item structure which provides links to resources which contain further information about the activity-structure.

Environment
Activities take place in a so-called ‘environment’, which is a structured collection of learning objects, services, and sub-environments. The relationship between an activity and an environment can be derived from the linguistic description of the activities. Most nouns in the activity imply the availability of learning objects in the environment; references to other persons imply the availability of communication services; some verbs imply the availability of supportive services or tools. For instance the activity: ‘read the problem and discuss solutions with your peers’ refers to environment components: ‘the problem’ which must be available for reading; and ‘peers’ which must be available to communicate with (including communication means).

Learning Object
Learning objects are defined here, as any reproducible and addressable digital or non-digital resource used to perform learning activities or support activities. In IMS Content Packaging they are represented with the element 'Resources'. Examples are: web pages, text books, productivity tools (text processors, editors, calculators, …), instruments (microscope, etc.), test items. A classification of different types of learning objects can be found in the LOM specification (element 5.2 Learning Resource Type makes a distinction between: exercise, simulation, questionnaire, diagram, figure, graph, index, slide, table, narrative text, exam, experiment, problem statement, self assessment, and lecture). A Learning Object may reference any of these types. However this assumes a runtime system that will be able to handle them.

Service
Besides resources which can be defined at design time, there are numerous so-called 'service facilities' used during the teaching and learning, for instance, a discussion forum or some other communication facility. Service facilities are resources that cannot be given a URL at design time. They have to be instantiated by a local runtime service. This is because, if a service facility is bound at design time, then that specific service would have to be used by all users of all instances of the learning design. When what is needed is an instance of the service that is unique to the runtime instance of the learning design and its assigned users, (e.g., if a chat forum is to be dedicated to the use of a specific group of learners and support staff associated with an particular instance of a learning design), then this has to be created and the local URL assigned after the instance of the design has been set up and the group of learners and staff associated with it. For this to work, it requires a well defined set of service types, which are known to the runtime service, such as chat, discussion forum, announcement channel etc. These are now commonly found in learning management systems. In a learning design, the use and setup of such a service is declared at an abstract level, so that a runtime facility (or a human) can setup the necessary facility according to the requirements. In the learning design specification, the abstract declaration of a service facility is called a 'service'. The instantiation of a service is called a 'service facility'.

Current service types are: *send-mail*, *conference*, *monitor* (level B), and *index search*. The selection of services to be included needs to be driven by the community. We therefore decided to start with the most widely implemented and used services in online learning environments.

Send-Mail Service
One of the services in any online learning facility is the ability to send and receive mail. This is done with an e-mail client. However, in learning situations it is often needed to know all the e-mail addresses of your fellow students and teachers when sending to groups. This information is available in the runtime system. In order to help users to send e-mail to other users in the same run of the unit-of-learning, the declaration of a send-mail facility is included in the services part. When a send mail service is included as part of a user's environment, the runtime system must provide a facility to edit a message and attachments and to send them to a selected list of e-mail addresses of users in the run of the unit-of-learning. This can either be to all the users in a specific role or to individuals selected from a role. The users receive the messages in their regular inbox in the e-mail client.

Conference Service
A typical communications service is a conference. The conference service, in addition to a title and metadata, specifies four conference system roles: participant, observer, conference-manager, and moderator. These contain references to roles in the learning design. When the learning design roles have been assigned players, this information can be used to automatically set up the dedicated conference space. It is not defined in this Learning Design Specification what permissions the conference roles should have, so this is left up to the implementer. However, these conference roles have commonly understood meanings and learning designers would have the expectation that implementers would stay within this range.

The conference service can be divided into three subtypes: synchronous conferences (like chats and audio/video conferences), asynchronous conferences (like newsgroups, forums), and announcements (one to many asynchronous conferences).

Monitor Service
The monitor service provides a facility for users to look at their own properties or that of others in a structured way. The idea is that the author defines IMSLD content (e.g. XHTML tables with global view-property elements) to view the properties. This IMSLD content is referred to with the 'item' element. When creating a monitor object, the author has to choose between allowing the user to see the properties of their own dossier ('self'), or those of all the users in

the specified role. With a monitor object, one can look at the properties in the dossier of 'self' or in the dossiers of all users in a certain role. When 'self' is selected, every property has exactly one value. When a role is selected, the properties in the dossiers of all the users in the specified role may be viewed. In this case the learning designer has to be careful, because only one view-property is specified, but the effect is recurrent for every user in the role. This means that the list in the user interface must be extended automatically when parsing the content:

- when the view-property is in a text line without other view-properties on the same line (all outside tables), then a list of values is created, each separated with a linefeed and carriage return.
- when there is more than one view-property in a text line (outside tables), then a list of values is created, each separated with a linefeed and carriage return and grouped per line in conformance with the grouping of the view-properties.
- when the view-property is in a table, than for each user in the role a new row in the table is created.

Index-Search Service
The Index-Search service enables a unit-of-learning to be indexed, and searches to be then made across it. In addition to *title* and *metadata* elements, it includes an *index* element and a *search* element.

The *index* element is a wrapper for indexing aspects, used to set up a search service.

The index is made in the background (not visible to users). The visibility is determined with the search element.

The functionality of the index is dependent on the search element:

- when search is free-text-search, then the index is made on the resource pointed at in the index (i.e., the underlying html texts).
- when search is index-with/without-reference, then an index is only made of the elements which share the same class, including underlying items. This has the form of a table of contents.

The *search* element specifies how a user can access the indexed entities. There are three possibilities:

- the user gets a free text search dialog, where he can search the index in a free text format (this also means that the index has to be build for free text retrieval). The syntax for free text retrieval is implementation dependent (e.g., the format found in search engines like Google or Altavista).
- the user is presented a text index (table of content) with (hyper-)linked (or on other media e.g., page numbers) references to the source.
- the user is presented a text index (table of content) without (hyper-)linked references. This provides e.g., information about the structure of the unit of learning.

Method
The method contains two core parts of the Learning Design Specification: the play and conditions, along with some completion and on-completion statements.

Play
The core part of the learning design is represented in the 'play'. A play specifies the actual learning design, the teaching-learning process, referring to the components declared earlier. In the play, it is specified which roles perform what activities in what order. When reading a learning design one basically reads the play. This is true for human readers as well as machines. Components not referenced in the play are not shown in the runtime system. A play is modelled according to a theatrical play with acts and role-parts. In general: a play consists of a sequence of acts. In each act, different activities are set for different roles and are preformed in parallel. When an act is completed, the next act starts until the completion requirements for the learning design are met.

Conditions
Conditions are only available in Levels B and C of the IMS Learning Design. They are used in conjunction with properties to further refinement and to add personalization facilities in the learning design.

Conditions have the basic format: IF [expression] THEN [show, hide, or change something or notify someone].

The expressions are mostly defined on the properties of the dossier of a learner (e.g., IF pre-knowledge-english=”4”). The effects of a condition are mostly different for individual users, although they can be assigned to the same role. Conditions work in the context of the current active act. In practice, conditions are mostly useful within activity-structures of the type ‘selection’.

Notification
Notifications are only available at Level C of the Learning Design Specification. With notifications it is possible to send a message to a role or to assign new learning or support activities to roles based on certain events. These events are:

- the completion of a certain activity.
- the completion of a certain act.
- the completion of a play.
- the completion of the unit of learning.
- when an expression in a certain condition is true.
- when a certain property-value has been changed.

Item
When a component, a learning objective, or a prerequisite needs a resource, an ‘item’ element is used in the similar way as in the organization part of IMS Content Packaging. The learning design provides a semantic context for these items, so that runtime systems can know what to do with the resource. For example in the following case:

<learning-objectives><item identifierref="o123"/></learning-objectives> it is clear that the resource with identifier ‘o123’ is a learning objective description. In the case of:

<activity-description><item identifierref="o345"/></activity-description> the item is an activity description. A runtime system can position the learning objectives in a certain place in the user-interface, different from the activity-description, and activity-descriptions can be handled differently from other learning content (this can vary from implementation to implementation, within the boundaries of the behavior descriptions provided in this information model).

3. Information Model

The information model for the level A, B, and C will be presented in the following format:

- The conceptual UML model, derived from Figure 2.2, showing only the elements that are appropriate for this level. The items not used are deleted.
- Tree diagrams representing successive parts of the learning-design and global-elements tree.
- Information tables of parts of the information structure. The diagrams and tables are successively unpacked.

Note: Due to the table format, the same elements are often described more than once in different tables. The description of the elements in the different tables is kept exactly the same, as they are generated from UML diagrams [LD17]. When the same element occurs more than once in a table, the information is not repeated, but referred to with 'see above'.

The diagrams use the following format:

- Only elements are shown (no attributes).
- The diagrams are tree structures, to be read from left to right. An element to the left contains the elements at the right hand. The element most to the left is the top of the tree.
- Only two trees are represented in this whole document: a tree with top level 'learning-design' and a small tree with top level 'global-elements'. The tree 'learning-design' is successively unpacked for presentation purposes. This is done in the order from left to right and from top to bottom of the tree. Elements that are completely unpacked are not further expanded in subsequent diagrams.
- An OR relationship in the diagrams is represented with **<**
- An AND relationship in the diagrams is represented with [
- * means that the element occurs zero or more times in the container
- + means that the element occurs one or more times in the container
- ? means that the element is optional
- When not one of the symbols (*, +, ?) is placed before the element name, the element occurs exactly one time.

The tables describe the elements and attributes of the diagram under which they are positioned. Tables have the following format:

No.	The number of the element in the hierarchy.
Name	The name of the element or attribute. Attributes are in italic. No annotation element is part of level A LD. (*) element is part of level B LD (B contains also A) (**) element is part of Level C LD (C contains A and B) (cp) element is part of IMS Content Packaging
Explanation	The meaning and function of the element.
Reqd	Indicates whether the element or attribute is mandatory (M) or optional (O).
Mult	Indicates the multiplicity of the element or attribute. 1 element occurs 1 time 0..1 element is optional and occurs zero or 1 time 0..n element may occur zero or more times 1..* element occurs 1 or more times - multiplicity is undetermined at this level. This is true for top level elements of which the multiplicity will be determined in the context of use

Type	Indicates the type of the element of attribute.

- Container: wraps one or more elements of the same type
- Choice: wraps a selection of multiple elements
- Sequence: wraps an ordered collection of multiple elements
- Group: placeholder for a elaborate hierarchy that is re-used multiple times. This hierarchy will be expanded in a separate table
- String: placeholder for character data
- Any: placeholder for any other construct
- Empty: end note not containing further character data

A general convention followed in naming attributes in the information model is the following. In order to distinguish between references (IDREF) *within* the learning-design model and references to resources in the content package, the following rules apply:
1. Attribute name 'ref' (IDREF) refers to an element with an identifier within the learning-design. Example: <act-ref ref=""/> refers to an act element within learning design.
2. Elements with the 'identifierref' attribute, refer to a resource in the content package. Example: <item identifierref="..."/> refers to a resource. The attribute name 'uri' is used for URIs, which are worldwide unique identifiers, and the attribute 'href' is used to refer to URIs.

3.1 Level A Information Model

3.1.1 Conceptual Model

The conceptual UML model for Level A is in Figure 3.1.

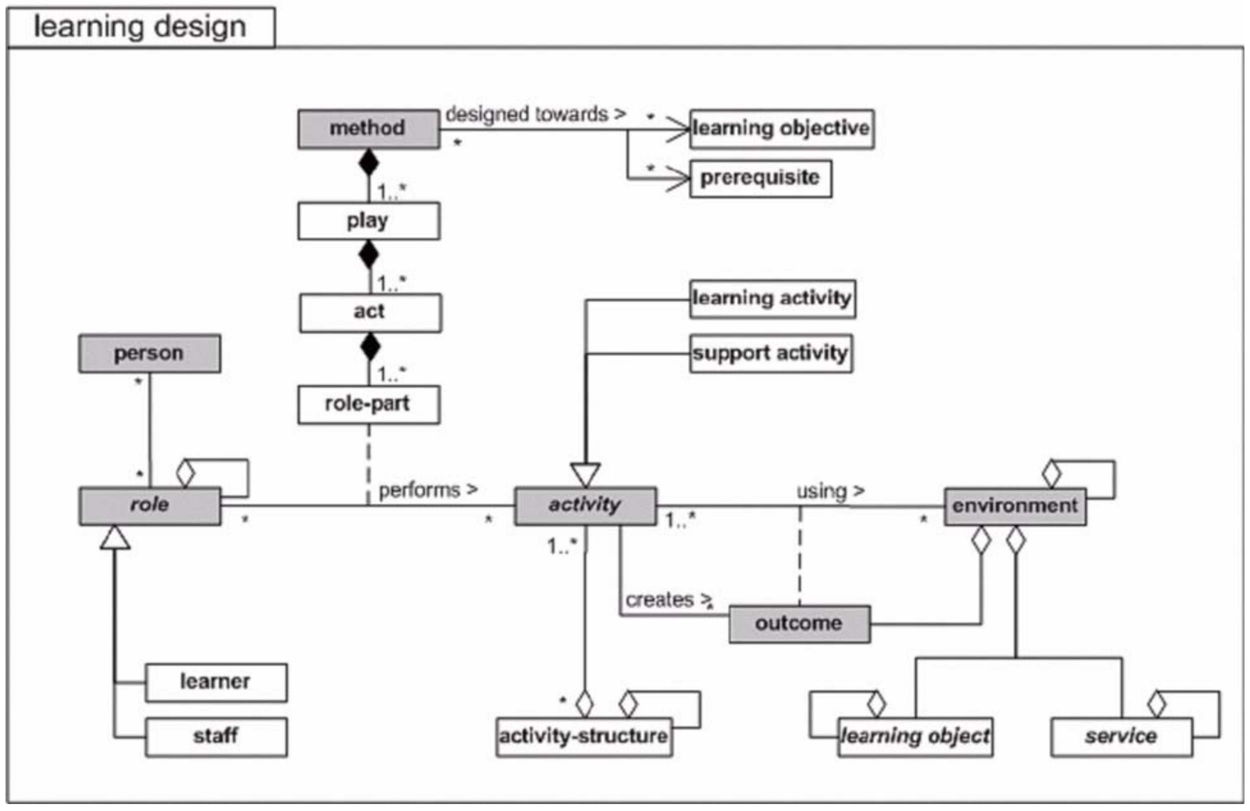
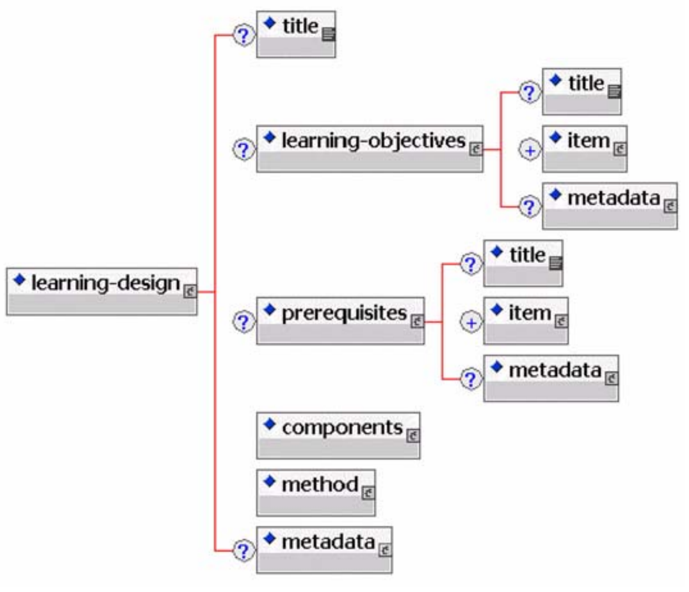


Figure 3.1 - Conceptual model of Level A.

3.1.2 Information Table ‘learning-design’



learning-design					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	learning-design	This element specifies the learning design.	-	-	sequence
0.1	*identifier*	An identifier that is unique within the learning design file (ID).	M	1	ID
0.2	*version*	A version number.	O	1	string
0.3	*uri*	Specifies a URI.	M	1	anyURI
0.4	*level*	Specifies the lowest level of Learning Design that the document instance is valid against. The letter is specified with one of the following characters: A, B, C, a, b or c. Possible *values:A*, B, C, a, b, c	M	1	token
0.5	*sequence-used*	Boolean, when set to 'true' IMS Simple Sequencing is included at the appropriate places in the document instance. Default is false. Possible *values:true*, false *Default value:false*	O	1	boolean
0.6	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.7	*learning-objectives*	Learning objectives describe the intended outcome for learners. Learning-objectives and prerequisites contain a standard organization of items, referring to resources or sub manifests. Resource types connected to learning objectives and prerequisites can be webcontent, imsldcontent or it can point to an IMS RDECO Schema. There are two locations where learning-objectives and prerequisites are specified: - At the level of the learning design (in the root of learning-design) - At the level of learning-activities (within learning-activities). The first ones are a more general description; the second ones are more concrete. There are two types of learning-objectives: 1 human readable descriptions (the items point to text resources) 2 machine-readable specifications. These are addressed through the href attribute of the resources pointed at. The learning-objectives schemas could be user-defined or fixed by an organization. In the latter case, the texts of the learning objectives are referred to (through href).	O	0..1	sequence
0.7.1	{itemmodel}	A schema group.	M	1	group
0.8	prerequisites	Prerequisites are the entry-requirements for students, e.g. the pre-knowledge needed. For the item formats see the description of the element 'learning-objectives'.	O	0..1	sequence
0.8.1	{itemmodel}	See above	M	1	group
0.9	components	Specifies the building blocks used in the method section.	M	1	sequence

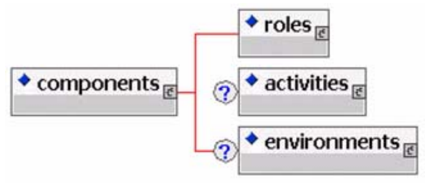
learning-design					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.10	method	The method contains a sequence of elements for the definition of the dynamics of the learning process. It consists of one or more play(s) (which could be interpreted as the runscript for the unit of learning) and a statement for the completion of the unit of learning.	M	1	sequence
0.11	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

3.1.3 Information Table 'item model'

See previous diagram. This information table comes from IMS Content Packaging.

{itemmodel}					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.2	item	A node in a structure, referring to a resource.	M	1..*	sequence
0.2.1	*identifier*	An identifier that is unique within the learning design file (ID).	O	1	ID
0.2.2	*identifierref*	Refers to an identifier of a resource in the content package (outside the learning design).	O	1	IDREF
0.2.3	*isvisible*	Initial visibility attribute; possible values: true (default) or false. *Possible values:*true, false *Default value:*true	O	1	boolean
0.2.4	*parameters*	Parameters to be passed during runtime.	O	1	string
0.2.5	title	See above	O	0..1	string
0.2.6	item	See above	O	0..*	sequence
0.2.7	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence
0.3	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

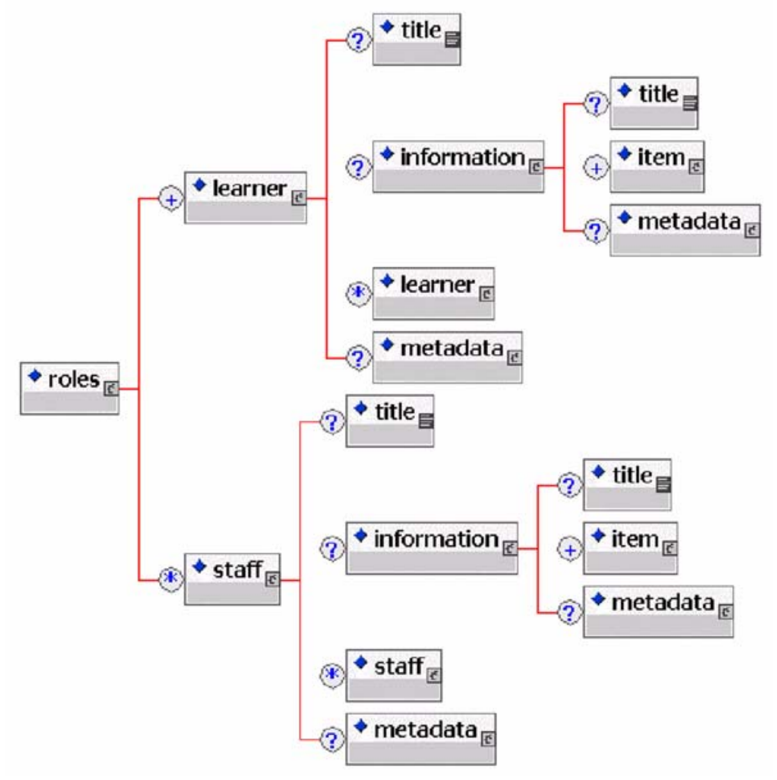
3.1.4 Information Table 'components'



components					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	components	Specifies the building blocks used in the method section.	-	-	sequence
0.1	roles	This element specifies the roles distinguished in the learning design. Roles contains a sequence that declares the two general roles: learner & staff. A href may be provided when referring to a global role (e.g. a role defined by an institute). This is obligatory when specifying a global role and connected globrole-properties. Global roles are specified with the href attribute. The rest of the declaration, like information, is local. It is not possible to declare global roles in a learning design. This is just an organizational issue and is nothing more or less than providing absolute URIs for roles. The URI doesn't necessarily point to a resource on the location of the address, its just used as a worldwide unique identifier. The attribute 'identifier' on roles can be used to refer to the whole group of all roles within the learning-design (learners and staff). In every learning-design at least one learner role is specified. In institutional installations the role names are fixed. For instance in most universities, the role-identifier for learners is: 'student'.	M	1	sequence

components					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1.1	*identifier*	An identifier that is unique within the learning design file (ID).	O	1	ID
0.2	activities	This element contains a choice for different activity definitions, including 'activity-structure'.	O	0..1	choice
0.3	environments	Container for the environment elements.	O	0..1	container

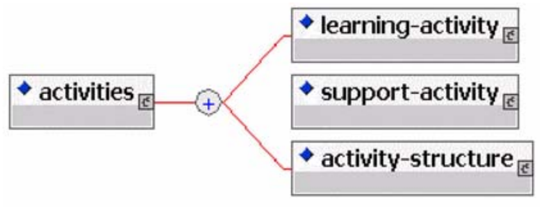
3.1.5 Information Table 'roles'



roles					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	roles	This element specifies the roles distinguished in the learning design. Roles contains a sequence that declares the two general roles: learner & staff. A href may be provided when referring to a global role (e.g. a role defined by an institute). This is obligatory when specifying a global role and connected globrole-properties. Global roles are specified with the href attribute. The rest of the declaration, like information, is local. It is not possible to declare global roles in a learning design. This is just an organizational issue and is nothing more or less than providing absolute URIs for roles. The URI doesn't necessarily point to a resource on the location of the address, its just used as a worldwide unique identifier. The attribute 'identifier' on roles can be used to refer to the whole group of all roles within the learning-design (learners and staff). In every learning-design at least one learner role is specified. In institutional installations the role names are fixed. For instance in most universities, the role-identifier for learners is: 'student'.	-	-	sequence
0.1	*identifier*	An identifier that is unique within the learning design file (ID).	O	1	ID
0.2	learner	In every learning design there is at least one learner-role. Learners can be 'nested', meaning that a role may be divided in sub roles. The title in the learner model is used to provide the name for the role. E.g. in an educational game you can distinguish roles like chair and participant as sub roles of student.	M	1..*	sequence

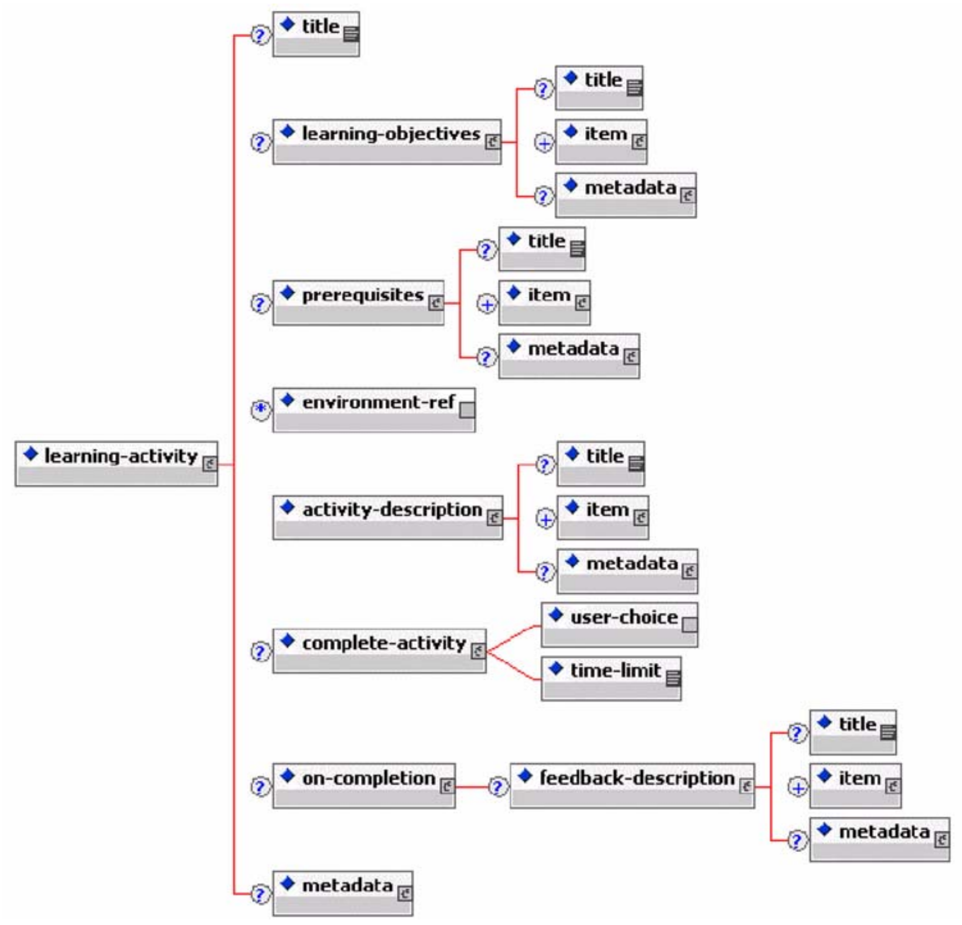
roles					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.2.1	*create-new*	This attribute indicates whether multiple occurrences of this role may be created during runtime. When the attribute has the value "not-allowed" then there is always one and only one instance of the role. If the value is "allowed" (default), a mechanism in the runtime system should be provided to create new instances of this role. If a new instance of a role is created, new instances for all available sub-roles of that role are created as well. Possible *values:allowed*, not-allowed *Default value:allowed*	O	1	token
0.2.2	*href*	Refers to a URI.	O	1	anyURI
0.2.3	*identifier*	See above	M	1	ID
0.2.4	*match-persons*	This attribute is used when there are several sub roles (e.g. chair, secretary, member). Persons can be matched exclusively to the sub roles, meaning that a person, who has the role of chair, may not be bound to one of the other roles at the same time. When it is not exclusive, persons may be bound to more than one sub role (this is the default situation). Possible *values:exclusively-in-roles*, not-exclusively	O	1	token
0.2.5	*max-persons*	Specifies the maximum number of persons bound to the role before starting the run. When the attribute min-persons and max-persons are empty, there are no restrictions. When used, the following rule applies: 0 <= min-persons <= max-persons	O	1	nonNegativeInteger
0.2.6	*min-persons*	Specifies the minimum number of persons bound to the role before starting a run. When the attribute min-persons and max-persons are empty, there are no restrictions. When used, the following rule applies: 0 <= min-persons <= max-persons	O	1	nonNegativeInteger
0.2.7	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.2.8	*information*	The information element may be used to provide additional information about the activity-structure. It specifies the set of items pointing to the resource(s) where the information can be found.	O	0..1	sequence
0.2.8.1	{itemmodel}	A schema group.	M	1	group
0.2.9	learner	See above	O	0..*	sequence
0.2.10	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence
0.3	staff	Staff members can be 'nested', meaning that a role may be divided in sub roles. E.g. in an educational institute you may distinguish roles like tutors, assessors, mentors, evaluators, etc.	O	0..*	sequence
0.3.1	create-new	See above	O	1	token
0.3.2	*href*	See above	O	1	anyURI
0.3.3	*identifier*	See above	M	1	ID
0.3.4	*match-persons*	See above	O	1	token
0.3.5	*max-persons*	See above	O	1	nonNegativeInteger
0.3.6	*min-persons*	See above	O	1	nonNegativeInteger
0.3.7	*title*	See above	O	0..1	string
0.3.8	information	See above	O	0..1	sequence
0.3.8.1	*{itemmodel}*	See above	M	1	group
0.3.9	staff	See above	O	0..*	sequence
0.3.10	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

3.1.6 Information Table 'activities'



activities					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	activities	This element contains a choice for different activity definitions, including 'activity-structure'.	-	-	choice
0.1	learning-activity	This element contains a sequence of elements for learning-activity definitions.	M	1	sequence
0.1.1	*identifier*	An identifier that is unique within the learning design file (ID).	M	1	ID
0.1.2	*isvisible*	Initial visibility attribute; possible values: true (default) or false. Possible values:true, false Default *value:true*	O	1	boolean
0.1.3	*parameters*	Parameters to be passed during runtime.	O	1	string
0.2	support-activity	This element contains a sequence of elements to define support-activities. A support activity can optionally be connected to a role. This means that the activity is repeated for every member in the supported role (learner/staff). Support activities are typically performed by staff members (e.g. tutors) to support learners. In some pedagogical models however learners can support learners (peer to peer teaching). It is also possible that staff members support staff members. When the optional role-ref element is set, it is expected that the support activity will act for every single user in the specified role(s). That is: the same support activity is repeated for every user in the role(s). When the role-ref is not available, the support activity is a single activity (like the learning-activity).	M	1	sequence
0.2.1	*identifier*	See above	M	1	ID
0.2.2	*isvisible*	See above	O	1	boolean
0.2.3	*parameters*	See above	O	1	string
0.3	activity-structure	An activity-structure groups activities in sequences or selections. The tree is handled depth first (and not breadth first).	M	1	sequence
0.3.1	*identifier*	See above	M	1	ID
0.3.2	*number-to-select*	When the attribute 'number-to-select' is set, the activity-structure is completed when the number of activities completed equals the number set. The number-to-select must be the same as or smaller than the number of activities (including unit-of-learnings) which are at the immediate child level. When the number-to-select isn't set, the activity-structure is completed when all the activities in the structure are completed.	O	1	nonNegativeI nteger
0.3.3	*sort*	The attribute 'sort' determines the sort-order in relation to the visibility. Default the order in which activities are made visible is in the order specified in the activity-structure. Possible *values:as-is,* visibility-order *Default value:as-is*	O	1	token
0.3.4	*structure-type*	Indicates whether the activity-structure represents a sequence or a selection. Possible *values:sequence,* selection	O	1	token

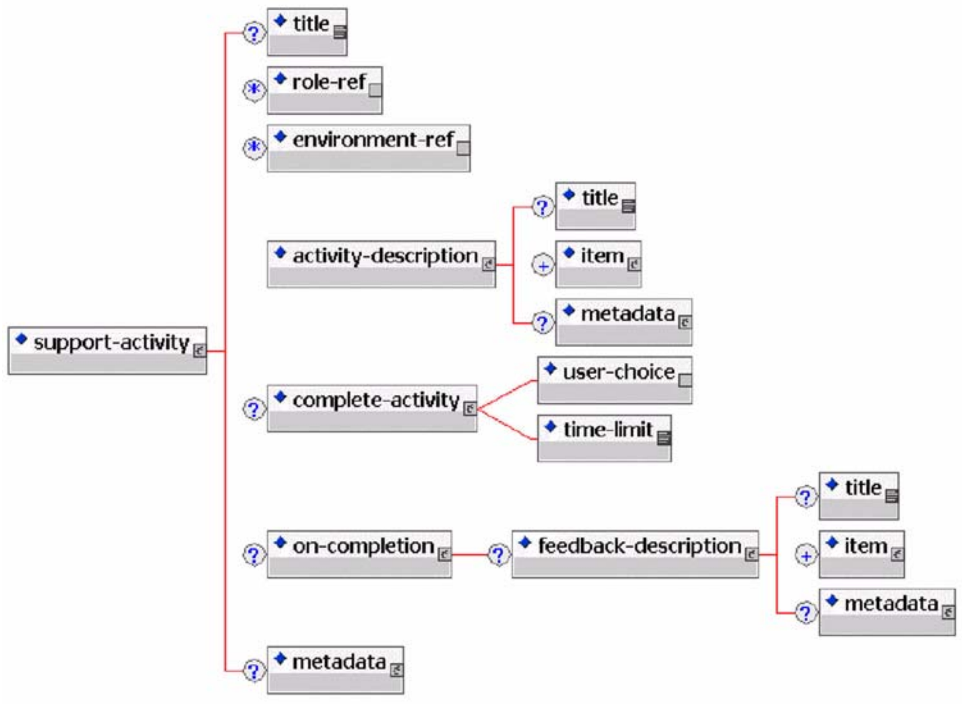
3.1.7 Information Table ‘learning-activity’



learning-activity					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	learning-activity	This element contains a sequence of elements for learning-activity definitions.	-	-	sequence
0.1	*identifier*	An identifier that is unique within the learning design file (ID).	M	1	ID
0.2	*isvisible*	Initial visibility attribute; possible values: true (default) or false. Possible values:true, false Default *value:true*	O	1	boolean
0.3	*parameters*	Parameters to be passed during runtime.	O	1	string
0.4	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.5	*learning-objectives*	Learning objectives describe the intended outcome for learners. Learning-objectives and prerequisites contain a standard organization of items, referring to resources or sub manifests. Resource types connected to learning objectives and prerequisites can be webcontent, imsldcontent or it can point to an IMS RDCEO Schema. There are two locations where learning-objectives and prerequisites are specified: - At the level of the learning design (in the root of learning-design) - At the level of learning-activities (within learning-activities). The first ones are a more general description; the second ones are more concrete. There are two types of learning-objectives: 1 human readable descriptions (the items point to text resources) 2 machine-readable specifications. These are addressed through the href attribute of the resources pointed at. The learning-objectives schemas could be user-defined or fixed by an organization. In the latter case, the texts of the learning objectives are referred to (through href).	O	0..1	sequence
0.5.1	{itemmodel}	A schema group.	M	1	group

learning-activity					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.6	prerequisites	Prerequisites are the entry-requirements for students, e.g. the pre-knowledge needed. For the item formats see the description of the element 'learning-objectives'.	O	0..1	sequence
0.6.1	{itemmodel}	See above	M	1	group
0.7	environment-ref	Refers to an environment in this package.	O	0..*	empty
0.7.1	ref	Refers to an identifier within the learning design.	M	1	IDREF
0.8	*activity-description*	Alias: task. The activity-description is the actual cue given to the user (rendered in the user-interface) to describe the activity to be performed by the user. In most cases the activity-description is text (of type webcontent or imsldcontent). In other cases it can be an audio-file (webcontent), a video file or any other cue to the user. Activity-descriptions also define the environment for the activity. Every noun mentioned in the description refers to a resource in the environment. It is up to the author to have a strict representation of the nouns in the environment or a more open one (leaving nouns implicit).	M	1	sequence
0.8.1	{itemmodel}	See above	M	1	group
0.9	complete-activity	Contains a choice of elements to specify when an activity is completed. When this element doesn't occur, the activity is set to 'completed'.	O	0..1	choice
0.9.1	user-choice	This element is used in the completed element of activities and specifies that the user may decide him or herself when the activity is completed. This means that a control must be available in the user-interface to set the activity status to 'completed'. A user can do this once (no undo). Once he/she indicated the activity to be completed, then this activity stays completed in the run.	M	1	empty
0.9.2	time-limit	The time limit specifies that it is completed when a certain amount of time has passed, relative to the start of the run of the current unit of learning. The datatype time is expressed in the 'duration' format (explained elsewhere in the information model). The time is always counted relative to the time when the run of the unit-of-learning has been started (see element: 'time-unit-of-learning-started'. Authors have to take care that the time limits set on role-parts, acts and plays are logical. In runtime the time limit of the play overrules the time limit on act and that one on the role-parts. In level B and C, the time-limit may be specified in a property (property-ref attribute, of type loc-property, datatype=string, to be declared by the author). In that case an author may set controls (set-property) on this property for users to control the value of the property. When a property-ref is specified, content in the element is ignored: the property overrules.	M	1	string
0.10	on-completion	When an activity, act, play or unit-of-learning is completed, the optional actions contained in this element are executed. In level A it contains only one element. The wrapper is available for the extensions of level B and C.	O	0..1	container
0.10.1	*feedback-description*	The underlying item elements point to a resource (of type webcontent or imsldcontent), where the feedback description can be found. After completion this text becomes visible.	O	0..1	sequence
0.10.1.1	{itemmodel}	See above	M	1	group
0.11	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

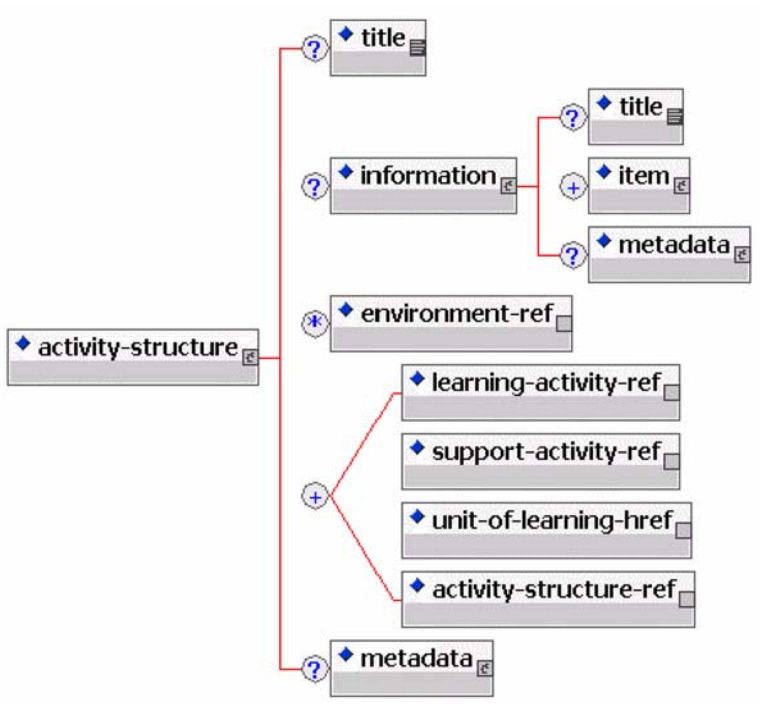
3.1.8 Information Table 'support-activity'



support-activity					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	support-activity	This element contains a sequence of elements to define support-activities. A support activity can optionally be connected to a role. This means that the activity is repeated for every member in the supported role (learner/staff). Support activities are typically performed by staff members (e.g. tutors) to support learners. In some pedagogical models however learners can support learners (peer to peer teaching). It is also possible that staff members support staff members. When the optional role-ref element is set, it is expected that the support activity will act for every single user in the specified role(s). That is: the same support activity is repeated for every user in the role(s). When the role-ref is not available, the support activity is a single activity (like the learning-activity).	-	-	sequence
0.1	*identifier*	An identifier that is unique within the learning design file (ID).	M	1	ID
0.2	*isvisible*	Initial visibility attribute; possible values: true (default) or false. Possible values:true, false Default *value:true*	O	1	boolean
0.3	*parameters*	Parameters to be passed during runtime.	O	1	string
0.4	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.5	*role-ref*	Refers to the identifier of the resource of the role. The element can be used as an operand in an expression.	O	0..*	empty
0.5.1	ref	Refers to an identifier within the learning design.	M	1	IDREF
0.6	*environment-ref*	Refers to an environment in this package.	O	0..*	empty
0.6.1	ref	See above	M	1	IDREF

support-activity					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.7	*activity-description*	Alias: task. The activity-description is the actual cue given to the user (rendered in the user-interface) to describe the activity to be performed by the user. In most cases the activity-description is text (of type webcontent or imsldcontent). In other cases it can be an audio-file (webcontent), a video file or any other cue to the user. Activity-descriptions also define the environment for the activity. Every noun mentioned in the description refers to a resource in the environment. It is up to the author to have a strict representation of the nouns in the environment or a more open one (leaving nouns implicit).	M	1	sequence
0.7.1	{itemmodel}	A schema group.	M	1	group
0.8	complete-activity	Contains a choice of elements to specify when an activity is completed. When this element doesn't occur, the activity is set to 'completed'.	O	0..1	choice
0.8.1	user-choice	This element is used in the completed element of activities and specifies that the user may decide him or herself when the activity is completed. This means that a control must be available in the user-interface to set the activity status to 'completed'. A user can do this once (no undo). Once he/she indicated the activity to be completed, then this activity stays completed in the run.	M	1	empty
0.8.2	time-limit	The time limit specifies that it is completed when a certain amount of time has passed, relative to the start of the run of the current unit of learning. The datatype time is expressed in the 'duration' format (explained elsewhere in the information model). The time is always counted relative to the time when the run of the unit-of-learning has been started (see element: 'time-unit-of-learning-started'. Authors have to take care that the time limits set on role-parts, acts and plays are logical. In runtime the time limit of the play overrules the time limit on act and that one on the role-parts. In level B and C, the time-limit may be specified in a property (property-ref attribute, of type loc-property, datatype=string, to be declared by the author). In that case an author may set controls (set-property) on this property for users to control the value of the property. When a property-ref is specified, content in the element is ignored: the property overrules.	M	1	string
0.9	on-completion	When an activity, act, play or unit-of-learning is completed, the optional actions contained in this element are executed. In level A it contains only one element. The wrapper is available for the extensions of level B and C.	O	0..1	container
0.9.1	*feedback-description*	The underlying item elements point to a resource (of type webcontent or imsldcontent), where the feedback description can be found. After completion this text becomes visible.	O	0..1	sequence
0.9.1.1	{itemmodel}	See above	M	1	group
0.10	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

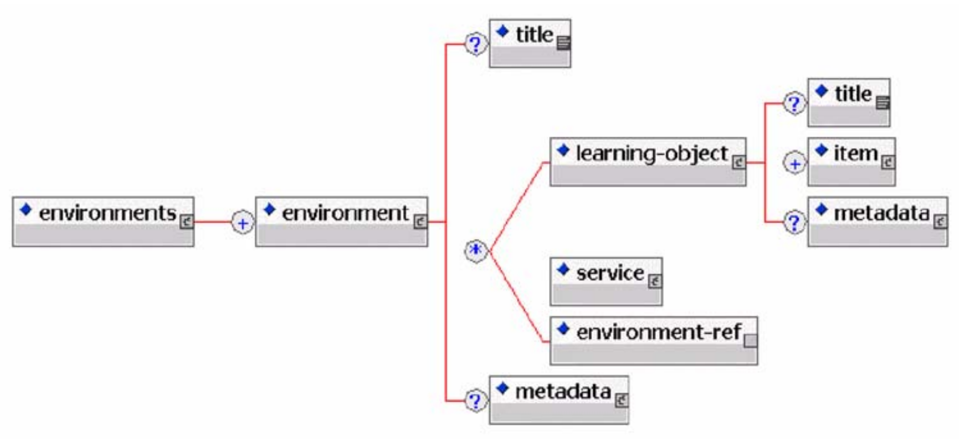
3.1.9 Information Table 'activity-structure'



activity-structure					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	activity-structure	An activity-structure groups activities in sequences or selections. The tree is handled depth first (and not breadth first).	-	-	sequence
0.1	*identifier*	An identifier that is unique within the learning design file (ID).	M	1	ID
0.2	*number-to-select*	When the attribute 'number-to-select' is set, the activity-structure is completed when the number of activities completed equals the number set. The number-to-select must be the same as or smaller than the number of activities (including unit-of-learnings) which are at the immediate child level. When the number-to-select isn't set, the activity-structure is completed when all the activities in the structure are completed.	O	1	nonNegativeInteger
0.3	*sort*	The attribute 'sort' determines the sort-order in relation to the visibility. Default the order in which activities are made visible is in the order specified in the activity-structure. *Possible values:*as-is, visibility-order *Default value:*as-is	O	1	token
0.4	*structure-type*	Indicates whether the activity-structure represents a sequence or a selection. *Possible values:*sequence, selection	O	1	token
0.5	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.6	information	The information element may be used to provide additional information about the activity-structure. It specifies the set of items pointing to the resource(s) where the information can be found.	O	0..1	sequence
0.6.1	{itemmodel}	A schema group.	M	1	group
0.7	environment-ref	Refers to an environment in this package.	O	0..*	empty
0.7.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.8		Choice	M	1..*	choice
0.8.1	learning-activity-ref	Refers to a learning-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.8.1.1	*ref*	See above	M	1	IDREF

activity-structure					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.8.2	support-activity-ref	Refers to a support-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.8.2.1	ref	See above	M	1	IDREF
0.8.3	unit-of-learning-href	The element can be used as an operand in a calculation or expression. This element is used to reference the appropriate elements of an external unit-of-learning (uol). This may be contained in the same package (the href is then a relative URI) or a resource pointing to a unit-of-learning outside of the package (the href is an absolute URI). It requires the use of a fragment identifier (#ID) added to the file reference. This is used, in the same way that an IDREF is used internally in an XML document, to point to the ID of an activity-structure, learning-activity, support-activity or environment element contained in the referenced external unit-of-learning. Note: this is equivalent to a simple or 'bare name' XPointer, which has the format: URI#ID and is the XML equivalent of an HTML fragment identifier. In XML Schema this format is supported by the anyURI construct.	M	1	empty
0.8.3.1	*href*	Refers to a URI.	M	1	anyURI
0.8.4	activity-structure-ref	Reference to an activity-structure.	M	1	empty
0.8.4.1	*ref*	See above	M	1	IDREF
0.9	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

3.1.10 Information Table ‘environments’

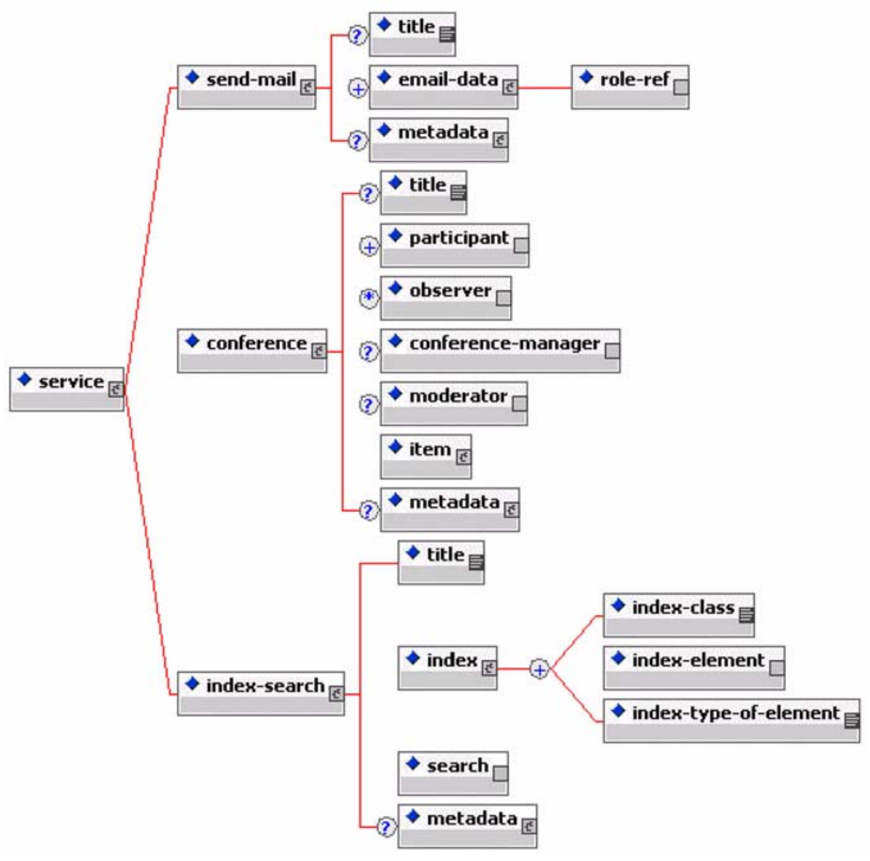


environments					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	environments	Container for the environment elements.	-	-	container
0.1	environment	Contains a sequence of elements to model an environment. IMS Simple Sequencing elements can be namespaced into the environment, to support the further sequencing of the environment elements. When there is no sequencing information provided, all elements that are specified in the environment will be shown to the user in the order and hierarchy provided, given that there are no further conditions defined that influences the visibility of the environment or the elements it contains.	M	1..*	sequence
0.1.1	*identifier*	An identifier that is unique within the learning design file (ID).	M	1	ID
0.1.2	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.1.3		Choice	O	0..*	choice

environments					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1.3.1	learning-object	Learning objects are incorporated either by using an included schema (e.g. IMS QTI) or by referencing resources through the item elements. IMS Simple Sequencing elements can be namespaced into the learning-object, to support the further sequencing of the item elements. When there is no sequencing information provided, all items that are specified in the learning-object will be shown to the user in the order and hierarchy provided, given that there are no further conditions defined that influences the visibility of the items or the learning-object.	M	1	choice
0.1.3.1.1	*class*	The class attribute refers to the value of class attributes available in learning-design or content elements. Contains a CDATA string. Just as in HTML more than one class may be specified in one CDATA string, each separated with a blank space. The priority order for classes is the same as specified in the CSS specification (see http://www.w3.org/style/css). Any element can in principle have the class attribute. 'Class' is a global defined W3C attribute for HTML 4.0 and XHTML [LD14]. This attribute assigns a class name or set of class names to an element. Any number of elements may be assigned the same class name or names. Multiple class names must be separated by white space characters. The class element can be used for semantic grouping of elements and be manipulated by IMSLD conditions and stylesheets. When sending a learning object to a web client, include the class attribute and value(s).	O	1	string
0.1.3.1.2	*identifier*	See above	M	1	ID
0.1.3.1.3	*isvisible*	Initial visibility attribute; possible values: true (default) or false. *Possible values:*true, false *Default value:*true	O	1	boolean
0.1.3.1.4	*parameters*	Parameters to be passed during runtime.	O	1	string
0.1.3.1.5	*type*	The type of learning object (e.g. knowledge-object, tool-object test-object). Vocabulary used can be the one of 'learning resource type' element from the IEEE LTSC LOM.	O	1	string
0.1.3.1.6		Sequence	M	1	sequence
0.1.3.1.6.1	title	See above	O	0..1	string
0.1.3.1.6.2	item	A node in a structure, referring to a resource.	M	1..*	sequence
0.1.3.1.6.2.1	*identifier*	See above	O	1	ID
0.1.3.1.6.2.2	*identifierref*	Refers to an identifier of a resource in the content package (outside the learning design).	O	1	IDREF
0.1.3.1.6.2.3	*isvisible*	See above	O	1	boolean
0.1.3.1.6.2.4	*parameters*	See above	O	1	string
0.1.3.1.6.2.5	title	See above	O	0..1	string
0.1.3.1.6.2.6	item	See above	O	0..*	sequence
0.1.3.1.6.2.7	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence
0.1.3.1.6.3	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence
0.1.3.1.7		Sequence	M	1	sequence
0.1.3.1.7.1	schema	Indicate the schema to be used.	O	0..1	string
0.1.3.1.7.2	schemaversion	Indicate the version of the schema to be used.	O	0..1	string
0.1.3.1.8	{itemmodel}	A schema group.	M	1	group

environments					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1.3.2	service	A service is a declaration of a service facility which has to be bound during instantiation of a run of a unit of learning. To automate the set up process of a service facility from a service declaration, the runtime data from the instantiated learning design would be translated into a configuration format used by the conference system if the conference is to be automatically set up. This is an implementation issue. It is also possible that a system manager could read this information and set up the required conference space manually, but the intent is to alleviate the manager of this task by enabling it to be automated. The service specification is extensible by namespacing in additional services. When instantiating a service, the runtime systems needs to maintain a handle on the 'context' to which the service is to be bound and determine the users to whom the service is being made available. A service will be referenced by an item element's identifierref attribute. The item will be within an environment. The environment in turn will be associated with an activity, or possibly directly with a role-part, associating it with a role. The activity or role-part forms the context for the use of the service. The users playing the role are those that have access to the service.	M	1	choice
0.1.3.2.1	*class*	See above	O	1	string
0.1.3.2.2	*identifier*	See above	M	1	ID
0.1.3.2.3	*isvisible*	See above	O	1	boolean
0.1.3.2.4	*parameters*	See above	O	1	string
0.1.3.3	environment-ref	Refers to an environment in this package.	M	1	empty
0.1.3.3.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.1.4	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

3.1.11 Information Table ‘service’

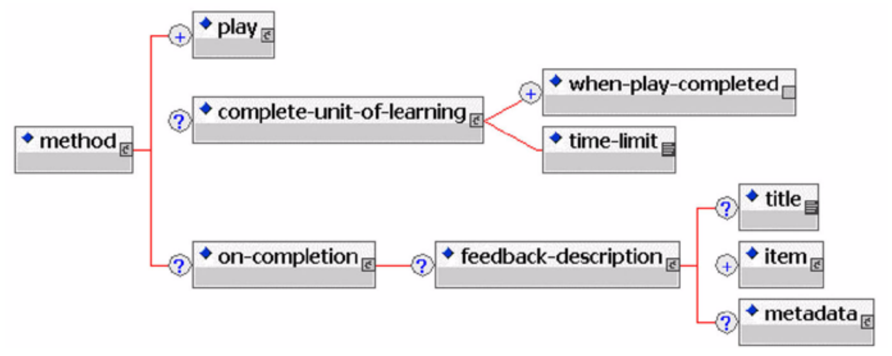


service					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	service	A service is a declaration of a service facility which has to be bound during instantiation of a run of a unit of learning. To automate the set up process of a service facility from a service declaration, the runtime data from the instantiated learning design would be translated into a configuration format used by the conference system if the conference is to be automatically set up. This is an implementation issue. It is also possible that a system manager could read this information and set up the required conference space manually, but the intent is to alleviate the manager of this task by enabling it to be automated. The service specification is extensible by namespacing in additional services. When instantiating a service, the runtime systems needs to maintain a handle on the 'context' to which the service is to be bound and determine the users to whom the service is being made available. A service will be referenced by an item element's identifierref attribute. The item will be within an environment. The environment in turn will be associated with an activity, or possibly directly with a role-part, associating it with a role. The activity or role-part forms the context for the use of the service. The users playing the role are those that have access to the service.	-	-	choice
0.1	*class*	The class attribute refers to the value of class attributes available in learning-design or content elements. Contains a CDATA string. Just as in HTML more than one class may be specified in one CDATA string, each separated with a blank space. The priority order for classes is the same as specified in the CSS specification (see http://www.w3.org/style/css). Any element can in principle have the class attribute. 'Class' is a global defined W3C attribute for HTML 4.0 and XHTML [LD14]. This attribute assigns a class name or set of class names to an element. Any number of elements may be assigned the same class name or names. Multiple class names must be separated by white space characters. The class element can be used for semantic grouping of elements and be manipulated by IMSLD conditions and stylesheets. When sending a learning object to a web client, include the class attribute and value(s).	O	1	string
0.2	*identifier*	An identifier that is unique within the learning design file (ID).	M	1	ID
0.3	*isvisible*	Initial visibility attribute; possible values: true (default) or false. Possible values:true, false Default *value:true*	O	1	boolean
0.4	*parameters*	Parameters to be passed during runtime.	O	1	string
0.5	send-mail	This service is used to send mail to users in roles (with mail address in property for level b/c).	M	1	sequence
0.5.1	*select*	Fixed choice: 'all-persons-in-role' or 'persons-in-role'. With the first choice, the user agent only allows messages to be sent to the role, indicating that all persons in the role get the message. With the second choice, the user agent allows a user to select one or more individuals within the specified role to send the message to. Possible *values:all-persons-in-role,* persons-in-role	M	1	token
0.5.2	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.5.3	*email-data*	This is used for send-mail purposes (as a service in the environment, or in notifications). In level B, the properties on this element refer to the property resources where the relevant e-mail data can be found for the connected role. In level A, the source is not specified explicitly and is left to implementers to decide how to address the data needed. Both properties (email, username) should be available for all persons assigned to the role and also for the sending party.	M	1..*	container
0.5.3.1	role-ref	Refers to the identifier of the resource of the role. The element can be used as an operand in an expression.	M	1	empty
0.5.3.1.1	ref	Refers to an identifier within the learning design.	M	1	IDREF
0.5.4	*metadata*	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

service					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.6	conference	The elements participant, observer, conference-manager, moderator facilitate the setting of the user rights in the conferences. They each contain a role-ref which associates them with a role in the learning design. If more than one role is to be assigned to a conference role (e.g. several LD roles are to be participants) then several instances of the conference role are needed, one for each LD role. It depends on the implementation how a conference is set up and managed: 1. When the conference system is an integral part of the runtime system, it is expected to be set up automatically; 2. When the conference is external, the user-rights can be set manually by the conference manager. The conference manager must be able to get a list from the runtime agent about which conferences of what type, for what users, with what rights, need to be set up. 3. Using the data in this conference element, the conferences can also be set up by a generating scripts, configuration files or a legacy interface to the rights management system of the conferencing system. In all instances the runtime system must be able to provide this information in a structured way. The item element refers to the resource where the conferencing system is to be found or identified. External conferencing systems can be of any kind accessible through the internet (resource type is webcontent). Examples: netmeeting, placeware (synchronous), first-class, lotus notes, news groups (asynchronous). An announcement object sets the rights: creator of announcement = participant. Reader of announcements = observer.	M	1	sequence
0.6.1	conference-type	Fixed choice to specify the type of conference facility that is expected to be present at runtime: synchronous, asynchronous or announcement. Possible values:synchronous, asynchronous, announcement	M	1	token
0.6.2	title	See above	O	0..1	string
0.6.3	participant	Specifies who the participants are in the conference. Participants can read (listen/see) the information, and can contribute to the conference. This element has an effect on setting the user rights in the conference. At least one role must be specified to identify the participants in the conference.	M	1..*	empty
0.6.3.1	*role-ref*	Refers to a role identifier.	M	1	IDREF
0.6.4	observer	Specifies who the observers are in the conference. Observers have only reading rights; they may not contribute. This element has an effect on setting the user rights in the conference.	O	0..*	empty
0.6.4.1	*role-ref*	See above	M	1	IDREF
0.6.5	conference-manager	The conference manager is allowed to create new sub conferences and delete conferences he/she created. The new conferences are children of the existing base conference. The conference manager may not delete the base conference. It is deleted by system management when deleting the information of the (completed) run of the unit of learning. The conference manager has all the rights of observer, participant.	O	0..1	empty
0.6.5.1	*role-ref*	See above	M	1	IDREF
0.6.6	moderator	Specifies who the moderators are in the conference. Moderators are persons who have the right to control and change the contributions of participants before they are made visible to other participants or observers. When a moderator is specified it means that participants may not contribute directly to the conference, but via the moderator. The moderator can reject, adapt or accept a proposed contribution of a participant. In all cases the contributor is notified of the judgment of the moderator. When there are more users in the role connected to the moderator, all have the same rights, but always the first one who did the job decides. This element has an effect on the setting of the user rights in the conference.	O	0..1	empty
0.6.6.1	*role-ref*	See above	M	1	IDREF
0.6.7	item	A node in a structure, referring to a resource.	M	1	sequence
0.6.7.1	*identifier*	See above	O	1	ID
0.6.7.2	identifierref	Refers to an identifier of a resource in the content package (outside the learning design).	O	1	IDREF
0.6.7.3	*isvisible*	See above	O	1	boolean
0.6.7.4	*parameters*	See above	O	1	string

service					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.6.7.5	*title*	See above	O	0..1	string
0.6.7.6	*item*	See above	O	0..*	sequence
0.6.7.7	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence
0.6.8	*metadata*	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence
0.7	index-search	Contains a sequence of elements that declare an index and/or search service facility.	M	1	sequence
0.7.1	title	See above	O	0..1	string
0.7.2	index	A choice of elements to specify indexing aspects, used to set up a search service. The index is made in the background (not visible to users). The visibility is determined with the search element. The functionality of the index is dependent on the search element: - When search is free-text-search, then the index is made on the resource pointed at in the index (i.e. the underlying html texts). - When search is index-with/without-reference, than only an index is made of the elements which share the same class, including underlying items. This has the form of a table of content.	M	1	choice
0.7.2.1	index-class	This element selects the class to make the index on. Only one class item per element may be provided. Example: <index-class index-class="problemdescription"/> makes an index on all objects in the design which have one of the strings in the class attribute assigned to "problemdescription".	M	1	string
0.7.2.2	index-element	This element selects the element to make the index on. The index attribute specifies the element to index-on (only one reference per index-element). This indexing only makes sense when there is a structure to index on, or underlying text to index for free-text-search.	M	1	empty
0.7.2.2.1	*index*	Refers to the element to make the index of.	M	1	IDREF
0.7.2.3	index-type-of-element	In this element the type of element to index on is entered. Only one element name per index-type-element occurrence. The element names much match the element names uses in the IMSLD schema. E.g.: <index-type-of-element>learning-activity</index-type-of-element>	M	1	string
0.7.3	search	This element specifies how a user can access the indexed entities. There are three possibilities: 1. The user gets a free text search dialog, where he can search the index in a free text format (this also means that the index has to be build for free text retrieval). The syntax for free text retrieval is implementation dependent, e.g. the format found in search engines like Google or Yahoo. 2. The user is presented a text index (table of content) with (hyper-) linked (or on other media e.g. page numbers) references to the source. 3. The user is presented a text index (table of content) without (hyper-) linked references. This provides e.g. information about the structure of the unit of learning.	M	1	empty
0.7.3.1	search-type	Fixed choice to indicate the type of search facility that is expected at runtime: free-text-search, index-with-reference, index-without-reference. A free text search uses a free text search index. An index without reference is a list of terms without page-numbers or hyperlinks. An index with references use page-numbers or hyperlinks (depending on the publication medium used). *Possible values:free-text-search*, index-with-reference, index-without-reference	M	1	token
0.7.4	*metadata*	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

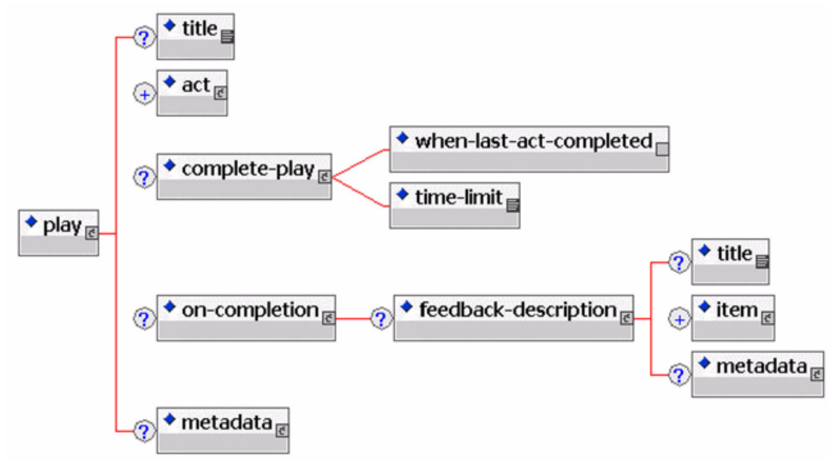
3.1.12 Information Table ‘method’



method					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	method	The method contains a sequence of elements for the definition of the dynamics of the learning process. It consists of one or more play(s) (which could be interpreted as the runscript for the unit of learning) and a statement for the completion of the unit of learning.	-	-	sequence
0.1	play	The play is the root element when interpreting the learning design. It represents the flow of activities during the learning process (the 'workflow' or better: the 'learningflow'). A play consists of a series of acts and an act consists of a series of role-parts. There is always at least one play in every learning design (and every unit-of-learning). In runtime the play is interpreted to show and hide activities, (other)units-of-learning, environments and resources to the users. When there is more than one play, these are interpreted concurrently and independent of each other. The same user can see the results of more than one play in the user-interface. Practical experience has shown that a lot of designs use multiple plays, to represent the flows of activities per role, e.g. a play for the learners and a play for staff. However this can only be done when the activities are independent of each other.	M	1..*	sequence
0.1.1	*identifier*	An identifier that is unique within the learning design file (ID).	O	1	ID
0.1.2	*isvisible*	Initial visibility attribute; possible values: true (default) or false. *Possible values:*true, false *Default value:*true	O	1	boolean
0.2	complete-unit-of-lea rning	A choice of elements to specify when a unit-of-learning is completed. When this element doesn't occur, the completed status is set to 'unlimited'.	O	0..1	choice
0.2.1	when-play-complete d	This element states that an unit-of-learning is completed when the referenced play(s) is (are) completed. More than one play can be selected, meaning that all the referenced plays must be completed before the unit-of-learning is completed. When a unit-of-learning is completed this should be made aware in the runtime environment to the managers of the system.	M	1..*	empty
0.2.1.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.2.2	time-limit	The time limit specifies that it is completed when a certain amount of time has passed, relative to the start of the run of the current unit of learning. The datatype time is expressed in the 'duration' format (explained elsewhere in the information model). The time is always counted relative to the time when the run of the unit-of-learning has been started (see element: 'time-unit-of-learning-started'. Authors have to take care that the time limits set on role-parts, acts and plays are logical. In runtime the time limit of the play overrules the time limit on act and that one on the role-parts. In level B and C, the time-limit may be specified in a property (property-ref attribute, of type loc-property, datatype=string, to be declared by the author). In that case an author may set controls (set-property) on this property for users to control the value of the property. When a property-ref is specified, content in the element is ignored: the property overrules.	M	1	string

method					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.3	on-completion	When an activity, act, play or unit-of-learning is completed, the optional actions contained in this element are executed. In level A it contains only one element. The wrapper is available for the extensions of level B and C.	O	0..1	container
0.3.1	feedback-description	The underlying item elements point to a resource (of type webcontent or imsldcontent), where the feedback description can be found. After completion this text becomes visible.	O	0..1	sequence
0.3.1.1	{itemmodel}	A schema group.	M	1	group

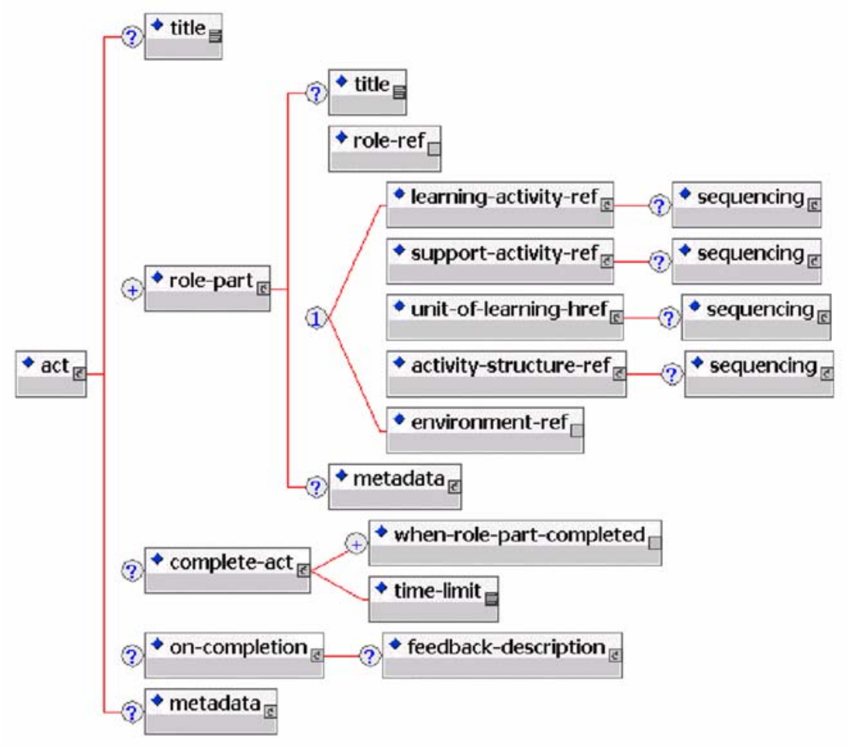
3.1.13 Information Table ‘play’



play					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	play	The play is the root element when interpreting the learning design. It represents the flow of activities during the learning process (the 'workflow' or better: the 'learningflow'). A play consists of a series of acts and an act consists of a series of role-parts. There is always at least one play in every learning design (and every unit-of-learning). In runtime the play is interpreted to show and hide activities, (other)units-of-learning, environments and resources to the users. When there is more than one play, these are interpreted concurrently and independent of each other. The same user can see the results of more than one play in the user-interface. Practical experience has shown that a lot of designs use multiple plays, to represent the flows of activities per role, e.g. a play for the learners and a play for staff. However this can only be done when the activities are independent of each other.	-	-	sequence
0.1	*identifier*	An identifier that is unique within the learning design file (ID).	O	1	ID
0.2	*isvisible*	Initial visibility attribute; possible values: true (default) or false. Possible values:true, false Default *value:true*	O	1	boolean
0.3	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string

play					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.4	*act*	A play consists of a series of acts and an act consists of a series of role-parts. An act represents a series of concurrent role-parts. There is at least one act in a play. When there is more than one act in a play, these are presented in sequence from first act to last act. Only one act in a play is the active act at any moment in time, starting with the first. When the first act is completed, the second act is made the active act. When the second act is completed, the third act is made active, etc. Acts, which are sequenced in row after the current active act, are never visible. Conditions cannot overrule this, meaning that the act is of higher priority than conditions. However, completed acts are still left visible and accessible, but the interface should sharply distinguish them from the current act, and make it clear that they are only available for reference as part of the learner's history.	M	1..*	sequence
0.4.1	identifier	See above	O	1	ID
0.5	*complete-play*	A choice of elements to specify when a play is completed. When this element doesn't occur, the completed is set to 'unlimited'.	O	0..1	choice
0.5.1	when-last-act-completed	This element states that a play is completed when the last act is completed.	M	1	empty
0.5.2	time-limit	The time limit specifies that it is completed when a certain amount of time has passed, relative to the start of the run of the current unit of learning. The datatype time is expressed in the 'duration' format (explained elsewhere in the information model). The time is always counted relative to the time when the run of the unit-of-learning has been started (see element: 'time-unit-of-learning-started'. Authors have to take care that the time limits set on role-parts, acts and plays are logical. In runtime the time limit of the play overrules the time limit on act and that one on the role-parts. In level B and C, the time-limit may be specified in a property (property-ref attribute, of type loc-property, datatype=string, to be declared by the author). In that case an author may set controls (set-property) on this property for users to control the value of the property. When a property-ref is specified, content in the element is ignored: the property overrules.	M	1	string
0.6	on-completion	When an activity, act, play or unit-of-learning is completed, the optional actions contained in this element are executed. In level A it contains only one element. The wrapper is available for the extensions of level B and C.	O	0..1	container
0.6.1	*feedback-description*	The underlying item elements point to a resource (of type webcontent or imsldcontent), where the feedback description can be found. After completion this text becomes visible.	O	0..1	sequence
0.6.1.1	{itemmodel}	A schema group.	M	1	group
0.7	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

3.1.14 Information Table 'act'



act					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	act	A play consists of a series of acts and an act consists of a series of role-parts. An act represents a series of concurrent role-parts. There is at least one act in a play. When there is more than one act in a play, these are presented in sequence from first act to last act. Only one act in a play is the active act at any moment in time, starting with the first. When the first act is completed, the second act is made the active act. When the second act is completed, the third act is made active, etc. Acts, which are sequenced in row after the current active act, are never visible. Conditions cannot overrule this, meaning that the act is of higher priority than conditions. However, completed acts are still left visible and accessible, but the interface should sharply distinguish them from the current act, and make it clear that they are only available for reference as part of the learner's history.	-	-	sequence
0.1	*identifier*	An identifier that is unique within the learning design file (ID).	O	1	ID
0.2	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.3	role-part	A play consists of a series of acts and an act consists of a series of role-parts. A role-part relates exactly one role to exactly one type of activity (including the performance of another unit-of-learning and activity-structures). Role-parts within one act, are performed concurrently. When an activity or item attribute isvisible is set to 'false', the link in the activity-tree may be made visible when the role-part sets the activity for a role (implementation dependent), but the content isn't accessible.	M	1..*	sequence
0.3.1	*identifier*	See above	O	1	ID
0.3.2	title	See above	O	0..1	string
0.3.3	role-ref	Refers to the identifier of the resource of the role. The element can be used as an operand in an expression.	M	1	empty
0.3.3.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.3.4		Choice	M	1	choice

act					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.3.4.1	learning-activity-ref	Refers to a learning-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.3.4.1.1	*ref*	See above	M	1	IDREF
0.3.4.2	support-activity-ref	Refers to a support-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.3.4.2.1	*ref*	See above	M	1	IDREF
0.3.4.3	unit-of-learning-href	The element can be used as an operand in a calculation or expression. This element is used to reference the appropriate elements of an external unit-of-learning (uol). This may be contained in the same package (the href is then a relative URI) or a resource pointing to a unit-of-learning outside of the package (the href is an absolute URI). It requires the use of a fragment identifier (#ID) added to the file reference. This is used, in the same way that an IDREF is used internally in an XML document, to point to the ID of an activity-structure, learning-activity, support-activity or environment element contained in the referenced external unit-of-learning. Note: this is equivalent to a simple or 'bare name' XPointer, which has the format: URI#ID and is the XML equivalent of an HTML fragment identifier. In XML Schema this format is supported by the anyURI construct.	M	1	empty
0.3.4.3.1	*href*	Refers to a URI.	M	1	anyURI
0.3.4.4	activity-structure-ref	Reference to an activity-structure.	M	1	empty
0.3.4.4.1	*ref*	See above	M	1	IDREF
0.3.4.5	environment-ref	Refers to an environment in this package.	M	1	empty
0.3.4.5.1	*ref*	See above	M	1	IDREF
0.3.5	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence
0.4	complete-act	A choice of elements to specify when an act is completed. When this element doesn't occur, the completed is set to 'unlimited'.	O	0..1	choice
0.4.1	when-role-part-completed	This element states that an act is completed when the referenced role-part(s) is (are) completed. More than one role-part can be selected, meaning that all the referenced role-parts must be completed before the act is completed. NB: all role-part references must be in specified in the current act!	M	1..*	empty
0.4.1.1	*ref*	See above	M	1	IDREF
0.4.2	time-limit	The time limit specifies that it is completed when a certain amount of time has passed, relative to the start of the run of the current unit of learning. The datatype time is expressed in the 'duration' format (explained elsewhere in the information model). The time is always counted relative to the time when the run of the unit-of-learning has been started (see element: 'time-unit-of-learning-started'. Authors have to take care that the time limits set on role-parts, acts and plays are logical. In runtime the time limit of the play overrules the time limit on act and that one on the role-parts. In level B and C, the time-limit may be specified in a property (property-ref attribute, of type loc-property, datatype=string, to be declared by the author). In that case an author may set controls (set-property) on this property for users to control the value of the property. When a property-ref is specified, content in the element is ignored: the property overrules.	M	1	string
0.5	on-completion	When an activity, act, play or unit-of-learning is completed, the optional actions contained in this element are executed. In level A it contains only one element. The wrapper is available for the extensions of level B and C.	O	0..1	container
0.5.1	feedback-description	The underlying item elements point to a resource (of type webcontent or imsldcontent), where the feedback description can be found. After completion this text becomes visible.	O	0..1	sequence
0.5.1.1	{itemmodel}	A schema group.	M	1	group

act					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.6	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

3.1.15 Standard Name for the Unit of Learning Manifest File

Like any IMS Content Package, the manifest file of a unit of learning has a pre-defined name and location. The file (named 'imsmanifest.xml') is placed in the root of the package interchange file or any other packaging image like a CD-ROM. It is required that the name is kept in all lowercase letters. It would be desirable for a runtime system to know what specifications can be expected to be present in the content package (QTI, LD, SS, etc.). At the moment, no profiling mechanism is present to inform the parser or runtime before reading the file what facilities have to be present. This is a possible future specification to be provided by IMS.

3.1.16 Standard Namespace for IMS Learning Design Elements

The namespace to use for the learning design schema elements is: http://www.imsglobal.org/xsd/imsld_v1_p0

3.2 Level B Information Model

Level B provides additional elements, which significantly extend the ability of a learning designer to control the learning flow within a Unit of Learning. The main elements added are:

- Properties
- Conditions

The addition of properties and conditions affect different models:

1) The model of *components* is extended with the element *properties*, this is the place where the properties are declared.
2) The model of *complete-activity*, *complete-act*, *complete-play* and *complete-unit-of-learning* are extended to include the element *when-property-value-is-set*.
3) The model of *on-completion* is extended to include the element *change-property-value*.
4) The model of *service* is extended to include the element *monitor.*
5) The model of *email-data* is extended with two attributes (*email-property-ref* and *username-property-ref*) referring to global properties with data.
6) The model of *time-limit* is extended with one attribute (*property-ref*) referring to a property with data.
7) The element *method* is extended to include the element *conditions*.
8) The model of *complete-act* is extended to include the element *when-condition-true*.
9) A separate group of *global-elements* are included to read and set properties from all sorts of XML-based content schemas (e.g. XHTML).
10) Use is made of the W3C global attribute *class* to enable show and hide conditions on content elements in all sorts of XML-based content schemas (e.g. XHTML).

3.2.1 Conceptual Model

The conceptual UML model for Level B is in Figure 3.2. The grey marked classes are added to the model of Level A.

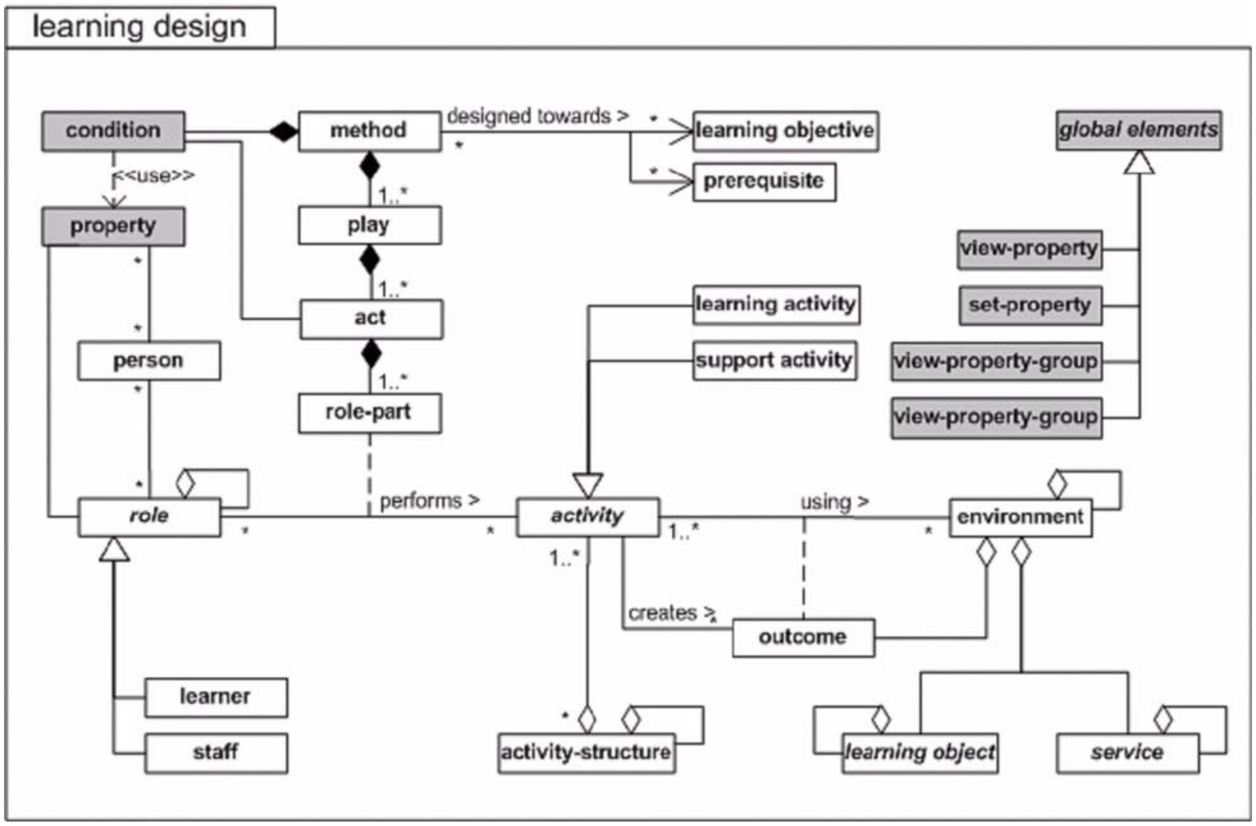


Figure 3.2 - Conceptual model of Level B.

The runtime system, or 'user-agent' is expected to keep record of property-values and property-definitions for users and roles in a so-called 'dossier'.

Properties are defined and or declared (for already defined global properties) under learning-design/components/properties and operated upon with property-operation elements (view-property, set-property, conditions, change-property-value, etc.).

There are several types of properties.

1) *Local properties* (element name: *loc-property*) are stored with a scope local to the run of a unit of learning. They are defined and used in the unit-of-learning. The value of this property is the same for every user in the run of the unit-of-learning, but can differ in different runs.
2) *Global properties* (element name: *glob-property*) are accessible outside the context of a unit of learning (e.g., by more than one unit of learning). They can be defined in one unit of learning and used in another one. In IMSLD global properties can be defined. Runtimes are expected to control whether a defined global property URI already exists or not. Global properties - once defined - may never change definition. So when the property already exists the definition is ignored.
3) *Personal properties* (element name: *locpers-property* and *globpers-property*) are owned by a person (local or global). These properties are used for personalization. For example, a portfolio that works across units of learning can be modeled with globpers (global personal) properties. The personal properties can be stored in a personal, portable 'dossier'.
4) *Role properties* (element name: *locrole-property*) are owned by a role and are always local. Every user in a specific role can access this property and has the same value in the same run of the unit of learning.

User-agents are expected to operate on properties in a secure way and with a maximum performance (to be detailed by the implementer).

3.2.1.1 The Scope of Global Properties

Global properties have to be maintained in a persistent storage. The organization or institution that controls the persistent storage effectively determines the scope of global properties by allowing or denying access to the storage.

Typically, a runtime system will have access to the persistent storage. However, there may be a number of different runtime systems accessing the same storage. The scope of the global properties is therefore extended to all these runtime systems.

A distinction can be made between global personal properties and the generic global properties.

The generic global properties are typically under the control of the organization or institution that provides the learning, so the learning provider determines their scope.

If at some point in the future, there is worldwide access to learner’s progress files, and these are used to maintain the data generated during learning activities, then the scope of global personal properties (globpers-property) is potentially actually global, assuming the runtime systems that a learner is concurrently using all have access to the same persistent data. An example might be a person who, as an employee, is taking training courses at work, but on his/her own personal time is registered as a part-time distance learner in a university across the globe.

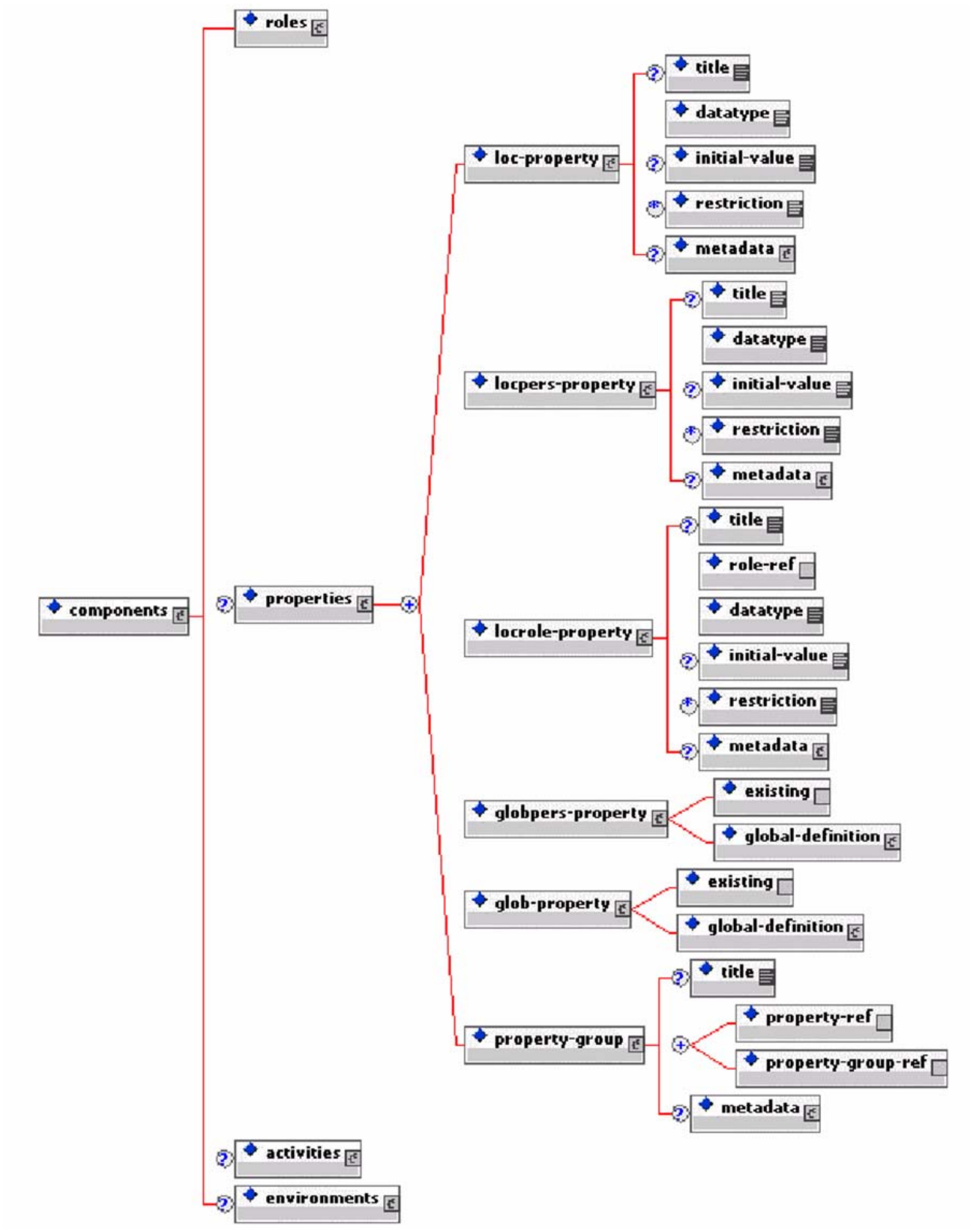
However, the issues of architectures, security, ownership, and control all need to be worked out and agreed on before this can happen and this is part of a larger problem that faces the uptake and use of the IMS LIP Specification for lifelong learning.

So, for the near- and perhaps medium-term future, personal learner information is likely to be maintained separately by each organization or institution that provides the learning (despite the problems this creates for lifelong learners). So for the time being, the learning provider will be likely to also determine the scope of global personal properties.

The other large issue is that of gaining widespread agreement as to the names, type, and vocabulary of global learner properties that will allow them to be used across systems.

3.2.2 Information Table ‘properties’

The element *properties* is added to the content model of the element *components*.


components
roles
properties
loc-property
title
datatype
initial-value
restriction
metadata
locpers-property
title
datatype
initial-value
restriction
metadata
locrole-property
title
role-ref
datatype
initial-value
restriction
metadata
globpers-property
existing
global-definition
glob-property
existing
global-definition
property-group
title
property-ref
property-group-ref
metadata
activities
environments

properties					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	properties (*)	Definition and declaration of new and existing properties. All properties referred to in the learning-design are declared in this section, including referred global-properties that may be defined elsewhere. However, global-properties that are not declared, nor referenced in the learning-design, can still be set or viewed with the global-elements available in the content resources of type 'imsldcontent'. All properties can be addressed in property-operations (property-ref, view-property, view-property-group, etc.).	-	-	choice
0.1	loc-property (*)	Local property, alias: run-property. This property has the same value for every user in a run. The property is owned by the run of the unit-of-learning. The identifier can be used to refer to the property in this unit-of-learning-package. Property-operations can refer to this identifier to operate on the value.	M	1	sequence
0.1.1	*identifier*	An identifier that is unique within the learning design file (ID).	M	1	ID
0.1.2	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string
0.1.3	*datatype* (*)	This is where the datatype is provided. The value is set with the attribute containing a fixed set of predefined datatypes. These are also predefined in the attribute datatype. For extensions use the 'other' value and specify the content in the element self.	M	1	string
0.1.3.1	datatype	The fixed set of datatypes to select from. The fixed values to choose from are: boolean, integer, real, string, datetime, duration, text, file, uri and other. The format is described elsewhere in the information model. Possible values:string, boolean, integer, uri, datetime, file, real, text, duration, other	M	1	token
0.1.4	*initial-value* (*)	The initial value of the property is set to the value of this element when specified. When this value isn't specified the initial value is '<no value>'.	O	0..1	string
0.1.5	restriction (*)	Zero or more restrictions of different type may be set on the property-values, meaning that the property-value is valid when it is of the specified data type and its value is within the specified restriction rules. Zero or more restrictions can be specified (these have the same format as specified in the W3C XML schema 1.0 specification) in the attribute: 'restriction-type'. However properties may not contain arrays (lists) of data, but can only contain a single value. So restrictions only apply to this single value. (Also the 'whitespace' restriction is not supported in IMSLD). The restriction types are specified elsewhere in the information model.	O	0..*	string
0.1.5.1	restriction-type	The fixed set of restriction types supported: minExclusive, minInclusive, maxExclusive, maxInclusive, totalDigits, fractionDigits, length, minLength, maxLength, enumeration, whiteSpace, pattern (see elsewhere in the information model for an explanation). Possible values:minExclusive, minInclusive, maxExclusive, maxInclusive, totalDigits, fractionDigits, length, minLength, maxLength, enumeration, whiteSpace, pattern	O	1	token
0.1.6	*metadata*	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence
0.2	locpers-property (*)	Local personal property. This property can have a different value for every user in all the roles for a run of this unit-of-learning. The property is owned by the run of the unit-of-learning, specifying a value per user. The identifier can be used to refer to the property in this unit-of-learning-package. Property operations can refer to this identifier to operate on the value.	M	1	sequence
0.2.1	identifier	See above	M	1	ID
0.2.2	*title*	See above	O	0..1	string
0.2.3	datatype (*)	See above	M	1	string
0.2.3.1	*datatype*	See above	M	1	token
0.2.4	initial-value (*)	See above	O	0..1	string
0.2.5	*restriction* (*)	See above	O	0..*	string
0.2.5.1	restriction-type	See above	O	1	token
0.2.6	*metadata*	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

properties					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.3	locrole-property (*)	Alias: group-property. Local role property. This property has the same value for every user in the specified role during the run of a unit-of-learning. The property is owned by the role in the run of the unit-of-learning. The identifier can be used to refer to the property in this unit-of-learning-package. Property operations can refer to this identifier to operate on the value.	M	1	sequence
0.3.1	identifier	See above	M	1	ID
0.3.2	*title*	See above	O	0..1	string
0.3.3	role-ref	Refers to the identifier of the resource of the role. The element can be used as an operand in an expression.	M	1	empty
0.3.3.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.3.4	datatype (*)	See above	M	1	string
0.3.4.1	datatype	See above	M	1	token
0.3.5	*initial-value* (*)	See above	O	0..1	string
0.3.6	restriction (*)	See above	O	0..*	string
0.3.6.1	*restriction-type*	See above	O	1	token
0.3.7	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence
0.4	*globpers-property* (*)	Global personal property, alias: portfolio-property. This property can have a different value for every user, independent of the different runs of units of learning (it specifies the portfolio of the user). The person owns the property. The identifier can be used to refer to the property in this unit-of-learning-package. Property operations can refer to this identifier to operate on the value.	M	1	choice
0.4.1	identifier	See above	M	1	ID
0.4.2	*existing* (*)	Refers to a property already declared (e.g. in another unit-of-learning, or in the global dossier) to the knowledge of the author (see 'global-definition' what happens if the author defines a new global property which in practice already exists). The property is referred to with href, specifying an absolute URI. NB: when validating this unit-of-learning, the URI doesn't have to be present. The declaration of the URI by an external unit-of-learning can happen at any time. So this is only under the control of the author.	M	1	empty
0.4.2.1	href	Refers to a URI.	M	1	anyURI
0.4.3	*global-definition* (*)	The global definition can be used to declare and define global properties. Global properties can be defined once and can never be changed from a declaration within the context of a learning design (only outside in the database where the properties reside). Global properties are preferably to be defined with an external definition mechanism. To force consistency, the following rule applies. Once a global property has been defined in whatever context, it can never be changed! This is also true for re-publications of the same unit-of-learning. So the definition is only used when the URI (href) doesn't exist yet. Otherwise it is ignored. The URI must be an identifier which identifies the global property globally unique. It must be an absolute URI. When the URI is an URL, the URI doesn't need to point to the property location, but can be interpreted as an identifier.	M	1	sequence
0.4.3.1	uri	Specifies a URI.	M	1	anyURI
0.4.3.2	title	See above	O	0..1	string
0.4.3.3	*datatype* (*)	See above	M	1	string
0.4.3.3.1	datatype	See above	M	1	token
0.4.3.4	initial-value (*)	See above	O	0..1	string
0.4.3.5	*restriction* (*)	See above	O	0..*	string
0.4.3.5.1	restriction-type	See above	O	1	token
0.4.3.6	*metadata*	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

properties					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.5	glob-property (*)	A global property is a globally unique property, which stores one value, independent of user, units-of-learning and role. The identifier can be used to refer to the property in this unit-of-learning-package. Property operations can refer to this identifier to operate on the value.	M	1	choice
0.5.1	*identifier*	See above	M	1	ID
0.5.2	existing (*)	See above	M	1	empty
0.5.2.1	*href*	See above	M	1	anyURI
0.5.3	global-definition (*)	See above	M	1	sequence
0.6	*property-group* (*)	A definition of a group of properties that belong together (and are edited in e.g. a form). It can only contain properties of the same type. The identifier can be used to refer to the property-group in this unit-of-learning-package. Operations can refer to this identifier to operate on the value.	M	1	sequence
0.6.1	identifier	See above	M	1	ID
0.6.2	*title*	See above	O	0..1	string
0.6.3		Choice	M	1..*	choice
0.6.3.1	*property-ref* (*)	Refers to a property. This can be a property of any kind: local property, global property, local personal property, local role property, global personal property, local role property. The ref attribute refers to the property declaration in the learning-design. The element can be used as an operand in a calculation or expression.	M	1	empty
0.6.3.1.1	ref	See above	M	1	IDREF
0.6.3.2	property-group-ref (*)	Refers to a property-group.	M	1	empty
0.6.3.2.1	*ref*	See above	M	1	IDREF
0.6.4	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

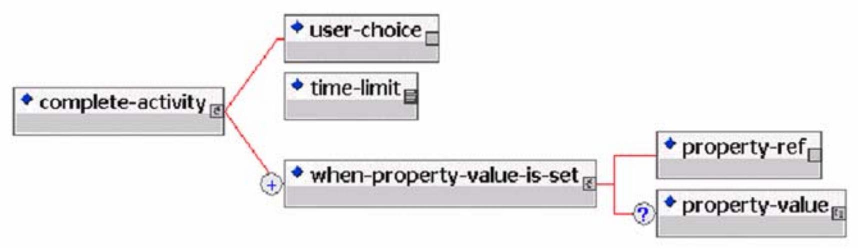
3.2.3 Information Table ‘when-property-value-is-set’

The element *when-property-value-is-set* is added to the content models of the following Level A elements:

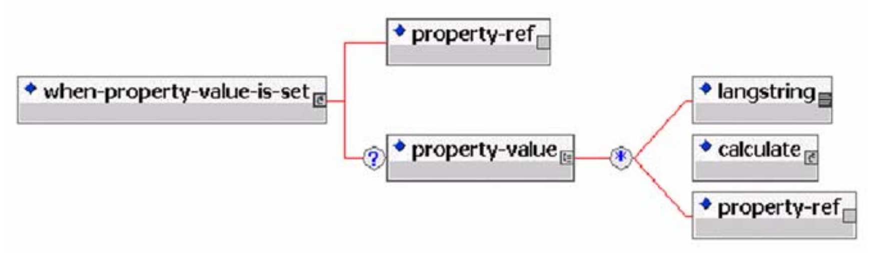
- complete-activity
- complete-act
- complete-play
- complete-unit-of-learning

In all four it is added as the last element in the group.

Example for complete-activity:



The model for *when-property-value-is-set* is:

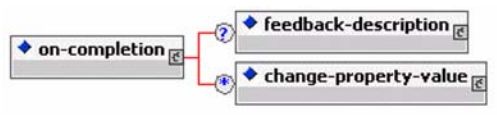


when-property-value-is-set					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	when-property-value -is-set (*)	Simple expression, containing two child elements: a property and an optional property-value. The condition evaluates to true when: 1) the property is set to the specified property-value; 2) the property is not NULL and the property-value is omitted.	-	-	sequence
0.1	property-ref (*)	Refers to a property. This can be a property of any kind: local property, global property, local personal property, local role property, global personal property, local role property. The ref attribute refers to the property declaration in the learning-design. The element can be used as an operand in a calculation or expression.	M	1	empty
0.1.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.2	property-value (*)	The element can be used as an operand in a calculation or expression. This element specifies the value a property is set or compared to. This depends on the context. For instance within an If statement the property is compared to the value. In a change-property-value context, the property is set to this value. Depending on the property-type this value is of type PCDATA or langstring. Property values may be calculated from the values of other properties. It is also possible to take over the property value of another property (with property-ref).	O	0..1	string
0.2.1	langstring	This is identical to the XHTML <p> element. The binding comes from the IMS Meta-Data. The attribute xml:lang may be added to all elements according to the W3C specifications. It is specifically needed on this element.	M	1	string
0.2.2	calculate (*)	This is the container for the elements to perform calculations. This container is also used in expressions.	M	1	choice
0.2.2.1	{expression} (*)	A schema group.	M	1	group
0.2.3	property-ref (*)	See above	M	1	empty
0.2.3.1	*ref*	See above	M	1	IDREF

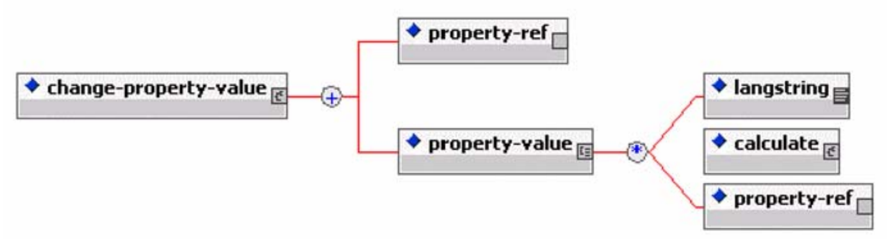
3.2.4 Information Table ‘change-property-value’

The element *change-property-value* is added to the content model of the element of Level A element *on-completion*. It also occurs in the Level B element *then*.

The extension of Level A *on-completion*:

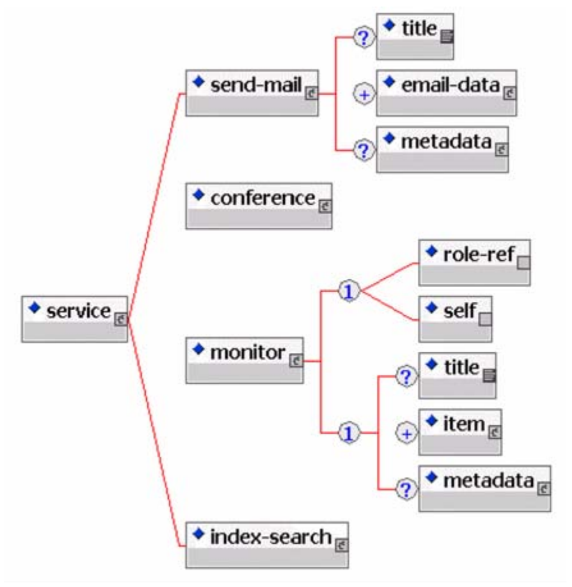


The model of change-property-value is:



change-property-value					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	change-property-value (*)	This element is used to change values of properties after an event (e.g. completion of something). E.g. when the activity is completed, a property value may be changed to reflect this fact. In the dossier also an automated record of completed activities is kept, so it isn't necessary to record the completion as such. It can be used to register (or change) other things.	-	-	sequence
0.1	property-ref (*)	Refers to a property. This can be a property of any kind: local property, global property, local personal property, local role property, global personal property, local role property. The ref attribute refers to the property declaration in the learning-design. The element can be used as an operand in a calculation or expression.	M	1	empty
0.1.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.2	property-value (*)	The element can be used as an operand in a calculation or expression. This element specifies the value a property is set or compared to. This depends on the context. For instance within an If statement the property is compared to the value. In a change-property-value context, the property is set to this value. Depending on the property-type this value is of type PCDATA or langstring. Property values may be calculated from the values of other properties. It is also possible to take over the property value of another property (with property-ref).	M	1	string
0.2.1	langstring	This is identical to the XHTML <p> element. The binding comes from the IMS Meta-Data. The attribute xml:lang may be added to all elements according to the W3C specifications. It is specifically needed on this element.	M	1	string
0.2.2	calculate (*)	This is the container for the elements to perform calculations. This container is also used in expressions.	M	1	choice
0.2.2.1	{expression} (*)	A schema group.	M	1	group
0.2.3	property-ref (*)	See above	M	1	empty
0.2.3.1	*ref*	See above	M	1	IDREF

3.2.5 Information Table ‘monitor’



monitor					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	monitor (*)	The monitor service provides a facility for users to look at their own properties or that of others in a structured way. A monitor service uses global properties in resources of type 'imsldcontent' to view the properties of one-self or of all users in a role.	-	-	sequence
0.1		Choice	M	1	choice
0.1.1	role-ref	Refers to the identifier of the resource of the role. The element can be used as an operand in an expression.	M	1	empty
0.1.1.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.1.2	self (*)	Refers to the properties of the person him/her self instead of the role the person is in.	M	1	empty
0.2	{itemmodel}	A schema group.	M	1	group

3.2.6 Extension of ‘email-data’

The element *email-data* has two additional attributes in Level B. In Level A, the element does not have attributes.

No.	Name	Explanation	Reqd	Mult	Type
.1	*email-property-ref*	Attribute contains a reference to the property containing the email address of the users being notified.	M	1	anyURI
.2	*username-property-ref*	This attribute contains a reference to the property containing the user name of the users being notified.	O	1	anyURI

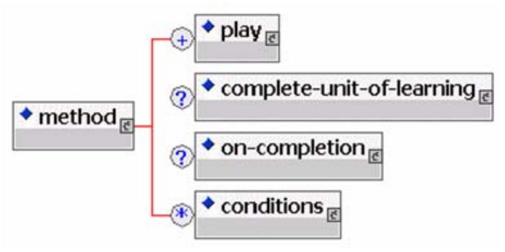
3.2.7 Extension of ‘time-limit’

The element *time-limit* has one additional attribute in Level B. In Level A, the element does not have attributes.

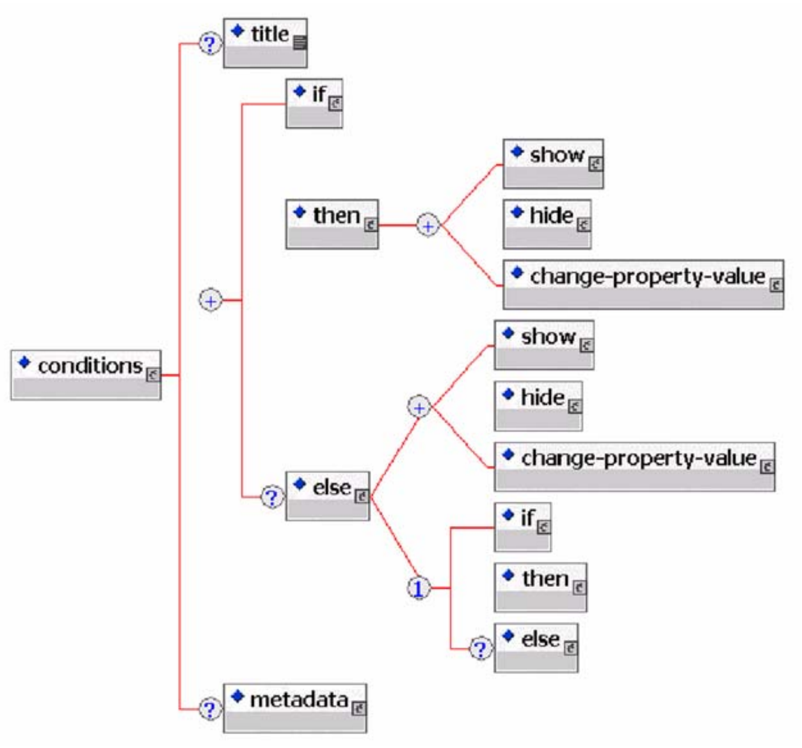
No.	Name	Explanation	Reqd	Mult	Type
.1	*property-ref*	Refers to a property identifier. In level B and C, the time-limit may be specified in a property (property-ref attribute, of type loc-property, datatype=string, to be declared by the author). In that case an author may set controls (set-property) on this property for users to control the value of the property. When a property-ref is specified, content in the element is ignored: the property overrules.	O	1	IDREF

3.2.8 Information Table 'conditions'

The element *conditions* is added to the content model of the Level A element *method*.



The model of *conditions* is:



conditions					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	conditions (*)	All conditions are pre-conditions and must be evaluated: - when entering the run of a unit of learning (new session); - every time when the value of a property has been changed. This applies only to the following properties: a) properties where the person has access to in the context of the run of the unit of learning, and b) the property has to be evaluated in one of the expressions in the unit of learning. These properties include properties, which are available in the expression, but are set automatically (e.g. time-unit-of-learning-started). An action is performed (fired) according to the success (true) or failure (false) of the condition. The action is to show, hide, change-property-value (level B) or notify (level C) a role. The show and hide actions set the visibility attribute (isvisible) of different objects: activities, environments, items, plays, activity-structures, units-of-learning and different classes of objects (set with the 'class' attribute).	-	-	sequence
0.1	title	A short name given to the resource, suitable for rendering in user-agents.	O	0..1	string

conditions					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.2		Sequence	M	1..*	sequence
0.2.1	if (*)	If refers to an expression-schema which evaluation results in the value: true or false. Different expression schemas may be used, however it is preferred to use the expression (and calculation) schema provided with IMSLD. The expression elements (e.g. a random number function) may be extended given that a separate namespace is used. When the expression resolves to 'true', the 'then' rule fires. When it resolves to 'false' the 'else' rule fires when it is present (otherwise nothing happens in this rule).	M	1	choice
0.2.1.1	{expression} (*)	A schema group.	M	1	group
0.2.2	then (*)	When expression specified in the if element is true, the statements after the then element are executed.	M	1	choice
0.2.2.1	{thenmodel} (*)	A schema group.	M	1	group
0.2.3	else (*)	Is executed when the If expression is false.	O	0..1	choice
0.2.3.1	{thenmodel} (*)	See above	M	1	group
0.2.3.2		Sequence	M	1	sequence
0.2.3.2.1	if (*)	See above	M	1	choice
0.2.3.2.1.1	{expression} (*)	See above	M	1	group
0.2.3.2.2	then (*)	See above	M	1	choice
0.2.3.2.2.1	{thenmodel} (*)	See above	M	1	group
0.2.3.2.3	else (*)	See above	O	0..1	choice
0.3	metadata	Placeholder for metadata. Include IMS Meta-Data here, using its namespace.	O	0..1	sequence

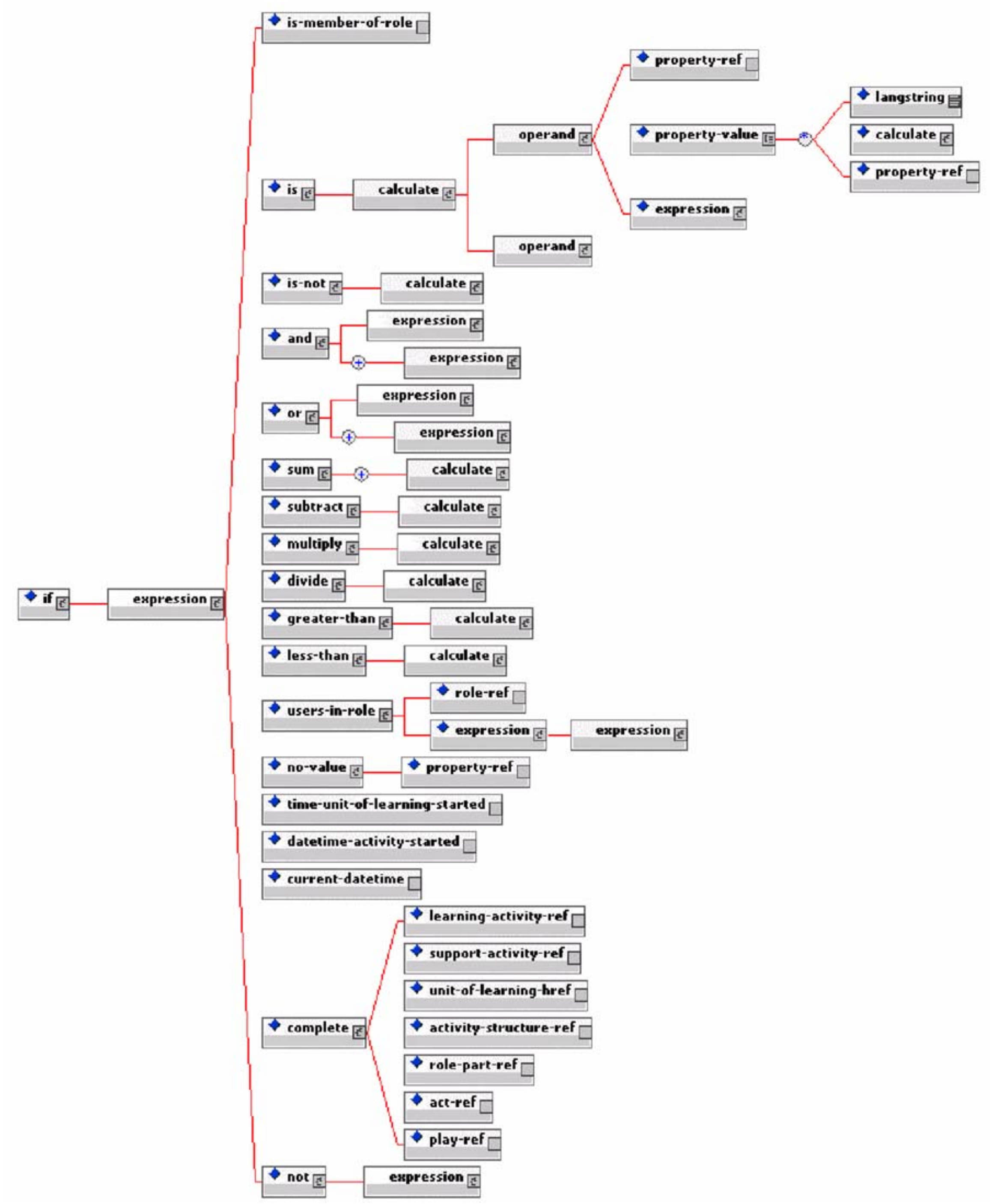
3.2.9 Information Table '{thenmodel}'

See previous diagram, {thenmodel} is a schema group.

{thenmodel}					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1	show (*)	A choice of elements to specify what has to be shown when the condition (if) is true. This has effect on the 'isvisible' status of the entity (set to true).	M	1	choice
0.1.1	{show-hide}	A schema group.	M	1	group
0.2	hide (*)	A choice of elements to specify what must be made hidden when the condition (if) is true. This affects the 'isvisible' status of the entity (set to false).	M	1	choice
0.2.1	{show-hide}	See above	M	1	group
0.3	change-property-value (*)	This element is used to change values of properties after an event (e.g. completion of something). E.g. when the activity is completed, a property value may be changed to reflect this fact. In the dossier also an automated record of completed activities is kept, so it isn't necessary to record the completion as such. It can be used to register (or change) other things.	M	1	sequence

3.2.10 Information Table 'if'

The model for *if* contains two grouping entities named *expression* and *calculate*. The entity *expression* is also available as an element with the same content in the model of elements *when-condition-true* (see later) and *users-in-role*. The entity *calculate* exists as an element with the identical structure within the content model of the element *property-value* (see: *change-property-value*). To avoid repeating the same information several times, the models are only expressed here.



if					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	if (*)	If refers to an expression-schema which evaluation results in the value: true or false. Different expression schemas may be used, however it is preferred to use the expression (and calculation) schema provided with IMSLD. The expression elements (e.g. a random number function) may be extended given that a separate namespace is used. When the expression resolves to 'true', the 'then' rule fires. When it resolves to 'false' the 'else' rule fires when it is present (otherwise nothing happens in this rule).	-	-	choice

if					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1	{expression} (*)	A schema group.	M	1	group

3.2.11 Information Table ‘{expression}’

See previous diagram, {expression} is a schema group.

{expression}					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1	is-member-of-role (*)	Is true when the person is a member of the role referenced with 'ref'.	M	1	empty
0.1.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.2	is (*)	Regular logical operator (true when an expressions is true)	M	1	sequence
0.2.1	{calculate} (*)	A schema group.	M	1	group
0.3	is-not (*)	Regular logical operator (true when an expression is false)	M	1	sequence
0.3.1	{calculate} (*)	See above	M	1	group
0.4	and (*)	Regular logical operator (true when both expressions are true)	M	1	sequence
0.4.1	{expression} (*)	See above	M	1	group
0.4.2	{expression} (*)	See above	M	1..*	group
0.5	or (*)	Regular logical operator (true when at least one of both expressions is true)	M	1	sequence
0.5.1	{expression} (*)	See above	M	1	group
0.5.2	{expression} (*)	See above	M	1..*	group
0.6	sum (*)	Regular calculation operator (returns the sum of the values provided)	M	1	container
0.6.1	{calculate} (*)	See above	M	1..*	group
0.7	subtract (*)	Regular calculation operator (returns the result of the first value minus the second value provided)	M	1	sequence
0.7.1	{calculate} (*)	See above	M	1	group
0.8	multiply (*)	Regular calculation operator (returns the product of the values provided)	M	1	sequence
0.8.1	{calculate} (*)	See above	M	1	group
0.9	divide (*)	Regular calculation operator (returns the result of the division of the first value by the second value)	M	1	sequence
0.9.1	{calculate} (*)	See above	M	1	group
0.10	greater-than (*)	Regular logical operator (is true when the first value is greater than the second value)	M	1	sequence
0.10.1	{calculate} (*)	See above	M	1	group
0.11	less-than (*)	Regular logical operator (is true when the first value is less than the second value)	M	1	sequence
0.11.1	{calculate} (*)	See above	M	1	group
0.12	users-in-role (*)	Contains a sequence of elements that specify that the expression applies to all the individual members of a referenced role (and not to the role itself).	M	1	sequence
0.12.1	role-ref	Refers to the identifier of the resource of the role. The element can be used as an operand in an expression.	M	1	empty
0.12.1.1	*ref*	See above	M	1	IDREF
0.12.2	expression (*)	Contains a choice of different expression elements.	M	1	choice
0.12.2.1	{expression} (*)	See above	M	1	group
0.13	no-value (*)	True when a property is empty, <no-value>.	M	1	container

{expression}					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.13.1	property-ref (*)	Refers to a property. This can be a property of any kind: local property, global property, local personal property, local role property, global personal property, local role property. The ref attribute refers to the property declaration in the learning-design. The element can be used as an operand in a calculation or expression.	M	1	empty
0.13.1.1	*ref*	See above	M	1	IDREF
0.14	time-unit-of-learning -started (*)	Refers to the time that the run of the unit of learning has been started. This is a fixed time, stored during the instantiation of the learning design. Uses the datetime format (see datatype).	M	1	empty
0.14.1	*unit-of-learning-uri*		M	1	anyURI
0.15	datetime-activity-sta rted (*)	Date and time when an activity has been first accessed by an individual user. Uses the datetime format (see datatype).	M	1	empty
0.15.1	*ref*	See above	M	1	IDREF
0.16	current-datetime (*)	Current date and time. Uses the datetime format (see datatype).	M	1	empty
0.17	complete (*)	Is true when the corresponding element (e.g. activity) has been completed.	M	1	choice
0.17.1	learning-activity-ref	Refers to a learning-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.17.1.1	*ref*	See above	M	1	IDREF
0.17.2	support-activity-ref	Refers to a support-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.17.2.1	*ref*	See above	M	1	IDREF
0.17.3	unit-of-learning-href	The element can be used as an operand in a calculation or expression. This element is used to reference the appropriate elements of an external unit-of-learning (uol). This may be contained in the same package (the href is then a relative URI) or a resource pointing to a unit-of-learning outside of the package (the href is an absolute URI). It requires the use of a fragment identifier (#ID) added to the file reference. This is used, in the same way that an IDREF is used internally in an XML document, to point to the ID of an activity-structure, learning-activity, support-activity or environment element contained in the referenced external unit-of-learning. Note: this is equivalent to a simple or 'bare name' XPointer, which has the format: URI#ID and is the XML equivalent of an HTML fragment identifier. In XML Schema this format is supported by the anyURI construct.	M	1	empty
0.17.3.1	*href*	Refers to a URI.	M	1	anyURI
0.17.4	activity-structure-ref	Reference to an activity-structure.	M	1	empty
0.17.4.1	*ref*	See above	M	1	IDREF
0.17.5	role-part-ref (*)	Reference to a role-part.	M	1	empty
0.17.5.1	*ref*	See above	M	1	IDREF
0.17.6	act-ref (*)	Refers to an act (in method/play/act).	M	1	empty
0.17.6.1	*ref*	See above	M	1	IDREF
0.17.7	play-ref (*)	Reference to a play.	M	1	empty
0.17.7.1	*ref*	See above	M	1	IDREF
0.18	not (*)	Regular logical expression.	M	1	choice
0.18.1	{expression} (*)	See above	M	1	group

3.2.12 Information Table '{calculate}'

See previous diagram, {calculate} is a schema group.

{calculate}					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1	{operand} (*)	A schema group.	M	1	group
0.2	{operand} (*)	See above	M	1	group

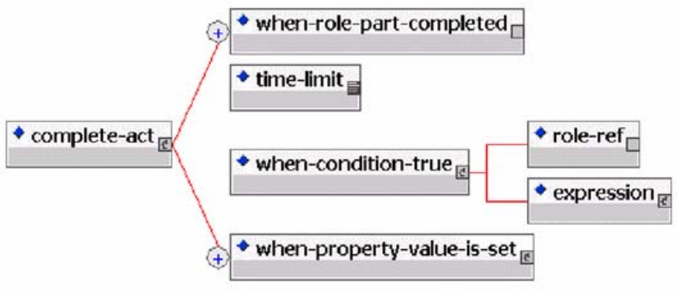
3.2.13 Information Table '{operand}'

See previous diagram, {operand} is a schema group.

{operand}					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1	property-ref (*)	Refers to a property. This can be a property of any kind: local property, global property, local personal property, local role property, global personal property, local role property. The ref attribute refers to the property declaration in the learning-design. The element can be used as an operand in a calculation or expression.	M	1	empty
0.1.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.2	property-value (*)	The element can be used as an operand in a calculation or expression. This element specifies the value a property is set or compared to. This depends on the context. For instance within an If statement the property is compared to the value. In a change-property-value context, the property is set to this value. Depending on the property-type this value is of type PCDATA or langstring. Property values may be calculated from the values of other properties. It is also possible to take over the property value of another property (with property-ref).	M	1	string
0.2.1	langstring	This is identical to the XHTML <p> element. The binding comes from the IMS Meta-Data. The attribute xml:lang may be added to all elements according to the W3C specifications. It is specifically needed on this element.	M	1	string
0.2.2	calculate (*)	This is the container for the elements to perform calculations. This container is also used in expressions.	M	1	choice
0.2.2.1	{expression} (*)	A schema group.	M	1	group
0.2.3	property-ref (*)	See above	M	1	empty
0.2.3.1	*ref*	See above	M	1	IDREF
0.3	{expression} (*)	See above	M	1	group

3.2.14 Information Table 'when-condition-true'

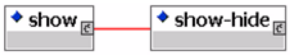
The element *when-condition-true* is added to the content model of *complete-act*, which was already extended with the element *when-property-value-is-set.*



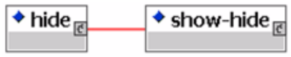
when-condition-true					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	when-condition-true (*)	Simple expression for a condition. This condition applies to all the individual users mentioned in the containing role-ref. When the contained expression is true for all users in the specified roles, this condition is true.	-	-	sequence
0.1	role-ref	Refers to the identifier of the resource of the role. The element can be used as an operand in an expression.	M	1	empty
0.1.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.2	expression (*)	Contains a choice of different expression elements.	M	1	choice
0.2.1	{expression} (*)	A schema group.	M	1	group

3.2.15 Information Table ‘show & hide’

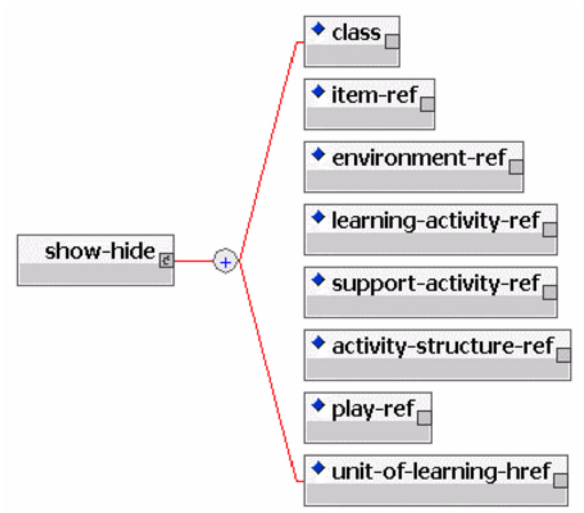
The elements *show* and *hide* have identical content models:



show					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	show (*)	A choice of elements to specify what has to be shown when the condition (if) is true. This has effect on the 'isvisible' status of the entity (set to true).	-	-	choice
0.1	{show-hide}	A schema group.	M	1	group



hide					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	hide (*)	A choice of elements to specify what must be made hidden when the condition (if) is true. This affects the 'isvisible' status of the entity (set to false).	-	-	choice
0.1	{show-hide}	A schema group.	M	1	group

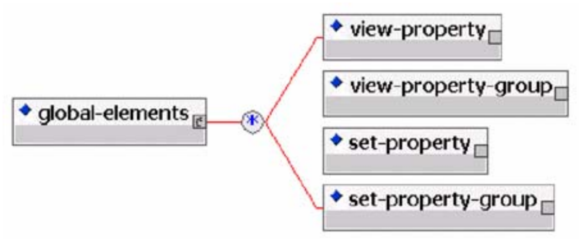


{show-hide}					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1	class (*)	Indicates that elements with a certain class attribute value must be shown or hidden, depending on the context of the element (in show or hide). The class attribute is a global attribute and may be set in resources of type 'imsldcontent' and is available on the environment elements in the learning-design model. Note that the classes can be used for style sheet like functions (e.g. set visibility), but they can also have a semantic classification purpose (just as in HTML) not connected to style sheets or automated processing at all. It is used to identify classes of common objects in order to manipulate them once.	M	1	empty
0.1.1	*class*	The class attribute refers to the value of class attributes available in learning-design or content elements. Contains a CDATA string. Just as in HTML more than one class may be specified in one CDATA string, each separated with a blank space. The priority order for classes is the same as specified in the CSS specification (see http://www.w3.org/style/css). Any element can in principle have the class attribute. 'Class' is a global defined W3C attribute for HTML 4.0 and XHTML [LD14]. This attribute assigns a class name or set of class names to an element. Any number of elements may be assigned the same class name or names. Multiple class names must be separated by white space characters. The class element can be used for semantic grouping of elements and be manipulated by IMSLD conditions and stylesheets. When sending a learning object to a web client, include the class attribute and value(s).	O	1	string
0.1.2	*title*	When the content is collapsed (see 'with-control') a title has to be given. The title is provided in the 'title' attribute on the class element.	O	1	string
0.1.3	*with-control*	Boolean: when true the content elements are hidden, but in the user-interface a collapse and expand control (like the [+] controls in Windows Explorer) is provided. With this control a user can decide to hide or show the content in the element him or herself.	O	1	boolean
0.2	item-ref (*)	Refers to the identifier of an item in the design context.	M	1	empty
0.2.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.3	environment-ref	Refers to an environment in this package.	M	1	empty
0.3.1	*ref*	See above	M	1	IDREF
0.4	learning-activity-ref	Refers to a learning-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.4.1	*ref*	See above	M	1	IDREF
0.5	support-activity-ref	Refers to a support-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.5.1	ref	See above	M	1	IDREF
0.6	*activity-structure-ref*	Reference to an activity-structure.	M	1	empty
0.6.1	ref	See above	M	1	IDREF
0.7	*play-ref* (*)	Reference to a play.	M	1	empty
0.7.1	ref	See above	M	1	IDREF
0.8	*unit-of-learning-href*	The element can be used as an operand in a calculation or expression. This element is used to reference the appropriate elements of an external unit-of-learning (uol). This may be contained in the same package (the href is then a relative URI) or a resource pointing to a unit-of-learning outside of the package (the href is an absolute URI). It requires the use of a fragment identifier (#ID) added to the file reference. This is used, in the same way that an IDREF is used internally in an XML document, to point to the ID of an activity-structure, learning-activity, support-activity or environment element contained in the referenced external unit-of-learning. Note: this is equivalent to a simple or 'bare name' XPointer, which has the format: URI#ID and is the XML equivalent of an HTML fragment identifier. In XML Schema this format is supported by the anyURI construct.	M	1	empty
0.8.1	href	Refers to a URI.	M	1	anyURI

3.2.16 Information Table ‘global elements’

There are four global elements defined in the IMSLD Specification:

In Level B these elements are all empty. The global elements must each be used separately – that is *without* the container element *global-elements*, within any XML content schema (like XHTML). The element *global-elements* has no other function than temporary grouping the different global elements. Note: global elements are not part of the *learning-design* model! Use the standard namespace.



global-elements					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	global-elements	Abstract wrapper for a choice of elements, used for temporary grouping of the global elements. It has no other function at all. It never occurs in content or any document instances. When global-elements, like set-property are used, they are always used on their own and not in the context of this wrapper.	-	-	choice
0.1	view-property	Global element, to be included in an external XML content schema. The resource type is 'imsldcontent'. With this element a specified property-value may be viewed. It works outside of the context of a text line (e.g. outside the context of a <p> element. The view attribute sets whether the value or the title+value should be delivered. It refers to the property with an ref or href. In order to avoid confusions it is good practice to include the imsldcontent with property-operations in the unit-of-learning-package. In case of personal properties, the property value of the user himself is returned. Only in the context of a monitor object (with the role-ref specified) the values of all the users in the specified role(s) are returned.	M	1	empty
0.1.1	*href*	Refers to a URI.	O	1	anyURI
0.1.2	*property-of*	Fixed selection: 'self' or 'supported-person'. When 'self' is selected only the value of the property in the person using the property is referenced. When 'supported-person' is selected, the properties of all the persons selected are referenced. Possible *values:self*, supported-person *Default value:self*	O	1	token
0.1.3	*ref*	Refers to an identifier within the learning design.	O	1	IDREF
0.1.4	*view*	Fixed selection: 'value' or 'title-value'. With 'value' only the value is returned for the display of the property. With 'title-value', the title of the property and the value is returned; in a property-group also the title of the group is returned. Possible *values:value*, title-value *Default value:value*	O	1	token
0.2	view-property-group	Global element, to be included in an external XML content schema. The resource type is 'imsldcontent'. With this element the values of the properties in a specified property-group may be viewed. It works outside of the context of a textline (e.g. outside <p>). The view attribute determines whether the titles of the containing values should be shown. The group-title is always shown. It refers to the property-group identifier or URI with a ref or href. In order to avoid confusions it is good practice to include the imsldcontent with property-operations in the unit-of-learning-package. In case of personal properties, the property value of the user himself is returned. Only in the context of a monitor object (with the role-ref specified) the values of all the users in the specified role(s) are returned.	M	1	empty
0.2.1	*href*	See above	O	1	anyURI
0.2.2	*property-of*	See above	O	1	token

global-elements					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.2.3	*ref*	See above	O	1	IDREF
0.2.4	*view*	See above	O	1	token
0.3	set-property	Global element, to be included in an external XML content schema. The resource type is 'imsldcontent'. With this element a specified property-value may be set by the user. It works outside of the context of a textline (e.g. outside <p>). The view attribute sets whether the value or the title+value should be delivered. The user gets a control in the user-interface to set the value of the property. The type of control is dependent on the property datatype and the restrictions. In the control the current value is shown and the datatype and restrictions are made explicit so that the user knows exactly what values are valid and which are not. This allows for client-side checking of the input (dependent on implementation this may also be dealt with at the server side). The element refers to the property URI or identifier with a ref or href. In order to avoid confusions it is good practice to include the imsldcontent with property-operations in the unit-of-learning-package. In case of personal properties, the property value of the user himself is set. Only in the context of a monitor object (with the role-ref specified) the values of all the users in the specified role can be set.	M	1	empty
0.3.1	*href*	See above	O	1	anyURI
0.3.2	*max-transactions*	The attribute max-transactions represents the number of times a property may set by a user. Technical (upload) errors do not count as a trial, but only successful transactions. When the attribute isn't specified, the number of attempts is set to unlimited.	O	1	nonNegativeI nteger
0.3.3	*property-of*	See above	O	1	token
0.3.4	*ref*	See above	O	1	IDREF
0.3.5	*transaction-type*	The attribute transaction-type is included for future extension, e.g. for secure transactions.	O	1	string
0.3.6	*view*	See above	O	1	token
0.4	set-property-group	Global element, to be included in an external XML content schema. The resource type is 'imsldcontent'. With this element the values of the properties contained in a specified property-group may be set by the user. It works outside of the context of a text line (e.g. outside <p>). The view attribute determines whether the titles of the containing values should be shown. The group-title is always shown. The user gets a control in the user-interface to set the value of properties in the property-group. The type of control per property is dependent on the property datatype and the restrictions set on the property. In the control the current value of the properties are shown and the data type and restrictions are made explicit so that the user knows exactly what values are valid and which are not. This allows for client-side checking of the input (dependent on implementation this may also be dealt with at the server side). All values of all properties in the group are set by the user before updating. The transaction is always counted for the group of properties, not for single properties. The element refers to the property-group identifier or URI with a ref or href. In order to avoid confusions it is good practice to include the imsldcontent with property-operations in the unit-of-learning package. In case of personal properties, the property values of the user himself is set. Only in the context of a monitor object (with the role-ref specified) the values of all the users in the specified role can be set.	M	1	empty
0.4.1	*href*	See above	O	1	anyURI
0.4.2	*max-transactions*	See above	O	1	nonNegativeI nteger
0.4.3	*property-of*	See above	O	1	token
0.4.4	*ref*	See above	O	1	IDREF
0.4.5	*transaction-type*	See above	O	1	string
0.4.6	*view*	See above	O	1	token

3.2.17 Global Attribute 'class'

Conditions can show or hide elements with the attribute *class*. This attribute is a global attribute – defined by the W3C in the context of Cascading Style Sheets (CSS) – that is available at the following elements within the learning-design model:

- learning-object
- service

Outside the context of learning design it can be added to any XML content schema. It is available on all elements in XHTML. Conditions do not only affect elements with the class attribute within the learning design, but also in the content, when this content is of resource type 'imsldcontent'.

3.2.18 Datatypes

The following datatypes are used in the property declaration. The format of each datatype is specified:

1) *Boolean*: Represents binary logic, true or false (aliases: yes/no; 1/0). NB: like all other datatypes, booleans can have <no-value>.
2) *Integer*: Represents the standard mathematical concept of integer numbers, representing whole positive and negative numbers (including zero), ranging from: -9223372036854775898 to 922372036854775807 (alias: longinteger).
3) *Real*: Represents the standard mathematical concept representing arbitrary precision decimal numbers, and must be capable of handling a number to 18 decimal places at least.
4) *String*: Represents any legal character strings. The minimal maximum number of characters is 2000.
5) *File*: Represents any binary file as datatype. The property stores this file.
6) *Uri*: Represents an URI according to the IETF's RFC 2396. Note: according to the W3C only URI should be used in future and not URL or URN. (see: http://www.w3.org/TR/uri-clarification/).
7) *Datetime*: Specifies the date and time in the form: CCYY-MM-DDThh:mm:ss. CC is the century; YY is the year (year 0000 is prohibited); MM is the month; dd is the day. T is the date/time separator; hh are the hours; mm are the minutes; ss are the seconds. (see ISO 8601). There is also an optional timezone separator. Partial productions of the lexical expression are not allowed.
8) *Duration*: Specifies an amount of time: the duration of an event in relative terms (e.g. the duration given the start datetime of the run of a unit-of-learning. The format - also used in the W3C XML schema specification - is: PnYnMnDTnHnMnS where:
   P is the designator that must always be present.
   n is a variable where an integer is filled in.
   nY represents the number of years.
   nM represents the number of month.
   nD represents the number of days.
   T is the date/time separator which must always be present when representing time.
   nH is the number of hours.
   nM is the number of minutes.
   nS is the number of seconds.
   Example: P2Y0M1DT20H10M55S. Meaning that the duration is: 2 years and 0 month and 1 day and 20 hours and 10 minutes and 55 seconds. Limited forms of lexical production are also allowed: For example, a duration of 40 minutes is expressed as PT40M; a duration of 30 days is P30D.
9) *Text*: represents any legal character strings. The minimal maximum number of characters is 64000 (about 10 pages of A4 text).

3.2.19 Restriction Types

For properties, certain restrictions on the data values can be specified. The restriction types are:

1) *length*: constrains the length of the property value of a textual datatype (string, text or uri) in terms of the number of characters that it can have.
2) *minLength*: constrains the minimum number of characters that a property of textual datatype can have.
3) *maxLength*: constrains the maximum number of characters that a property of textual datatype can have.
4) *enumeration*: constrains the value of a property to a specific value (use for value alternative lists).
5) *maxInclusive*: constrains the value of an ordered (integer, real, datetime) property to a specific inclusive upper bound.
6) *minInclusive*: constrains the value of a ordered property to a specific inclusive lower bound.
7) *maxExclusive*: constrains the value of a ordered property to a specific exclusive upper bound.
8) *minExclusive*: constrains the value of a ordered property to a specific exclusive lower bound.
9) *totalDigits*: constrains the value of a decimal property to a specific number of digits it must contain.
10) *fractionDigits*: constrains the value of a decimal property to the maximum number of digits it may have after the decimal point.
11) *pattern*: constrains the literals comprising the value of a property to a pattern defined by a regular expression.

3.3 Level C Information Model

Level C adds the capability for a learning designer to specify the sending of messages and setting of new activities based on certain events. The runtime system, or ‘user-agent’ is expected to support a notification mechanism. Notifications are event-driven mechanisms, which can be directed towards elements in the system or to human users.

Notifications affect the following content models of Level B elements:

1) The *on-completion* model is extended with a notification element.
2) The *then* model is extended with a notification element.
3) Global elements *set-property* and *set-property-group* are both extended with a notification element.

3.3.1 Conceptual Model

Figure 3.3 provides the conceptual UML model for Level C. The grey marked class is added to the model of Level B.

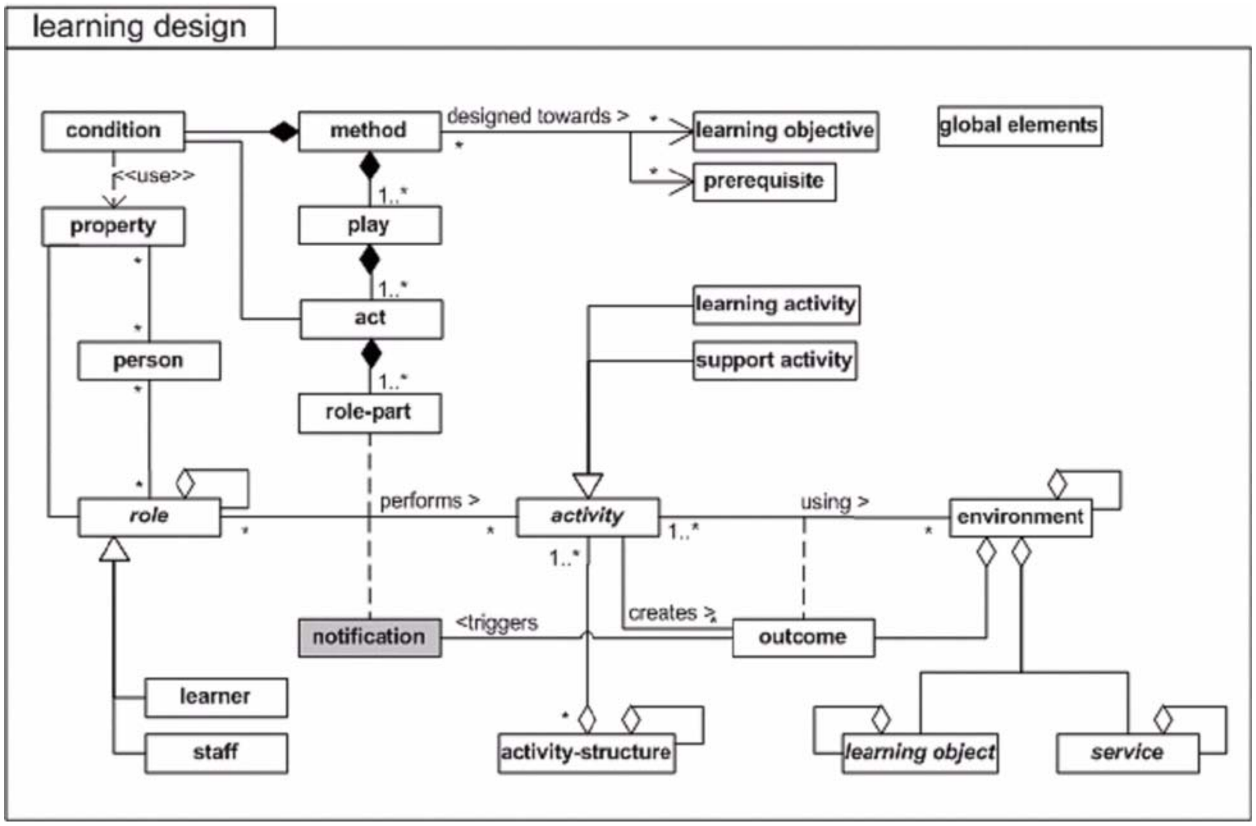
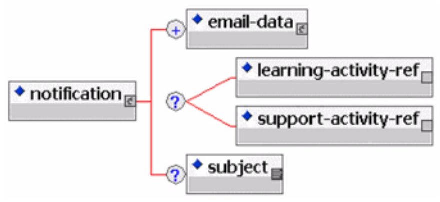


Figure 3.3 - Conceptual model of Level C.

3.3.2 Information Table 'notification'

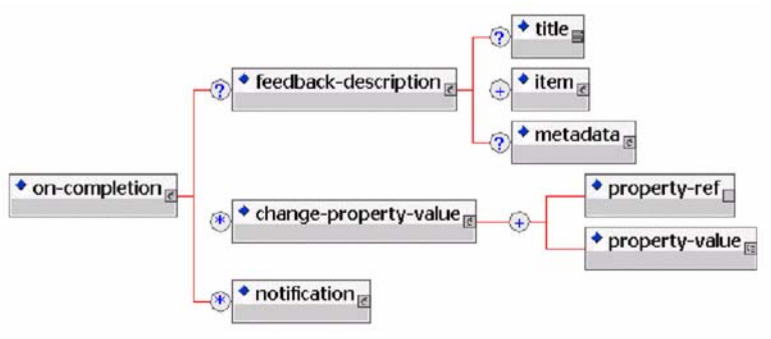


notification					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0	notification (**)	A notification happens after an event that is known by the runtime environment. Such an event can be e.g.: the completion of an activity, an expression evaluates to true, or a property value is set. The notification makes a new learning activity or a new support activity active for a role or it only sends a message. A notification is of the highest priority, meaning that an otherwise invisible item will be made visible and accessible to the user. Depending on the implementation an email message can be send to the user, notifying that a new activity has arrived (with a link to that activity in the message). The subject field can be set to a specific value (otherwise a standard message will be send). A notification can be inserted in external vocabularies (after an event like set-property), however, then the content must be provided in the package (because it contains references to identifiers in the package). When the identifier cannot be resolved the notification is ignored (but doesn't prevent the xhtml content from being presented).	-	-	sequence

notification					
No.	**Name**	**Explanation**	**Reqd**	**Mult**	**Type**
0.1	email-data	This is used for send-mail purposes (as a service in the environment, or in notifications). In level B, the properties on this element refer to the property resources where the relevant e-mail data can be found for the connected role. In level A, the source is not specified explicitly and is left to implementers to decide how to address the data needed. Both properties (email, username) should be available for all persons assigned to the role and also for the sending party.	M	1..*	container
0.1.1	*email-property-ref*	Attribute contains a reference to the property containing the email address of the users being notified.	M	1	anyURI
0.1.2	*username-property-ref*	This attribute contains a reference to the property containing the user name of the users being notified.	O	1	anyURI
0.1.3	role-ref	Refers to the identifier of the resource of the role. The element can be used as an operand in an expression.	M	1	empty
0.1.3.1	*ref*	Refers to an identifier within the learning design.	M	1	IDREF
0.2		Choice	O	0..1	choice
0.2.1	learning-activity-ref	Refers to a learning-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.2.1.1	*ref*	See above	M	1	IDREF
0.2.2	support-activity-ref	Refers to a support-activity. The element can be used as an operand in a calculation or expression.	M	1	empty
0.2.2.1	ref	See above	M	1	IDREF
0.3	*subject* (**)	It specifies the subject of a notification, to be presented to the notified actor when the notification is activated. E.g. in the mail-header (subject field).	O	0..1	string

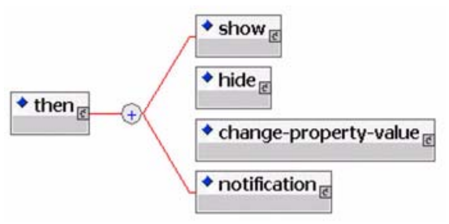
3.3.3 Extension of ‘on-completion’

Notifications are added to the *on-completion* model:



3.3.4 Extension of ‘then’

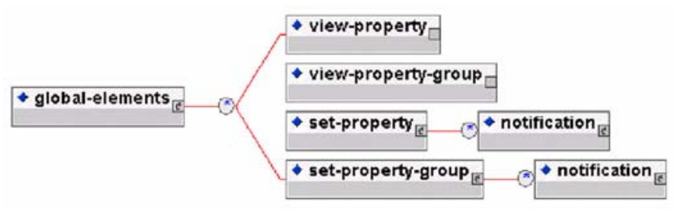
Notifications are added to the *then* model:



3.3.5 Extension of 'global elements'

Notifications are added to the *global-elements:*

- set-property
- set-property-group



4. Behavioral Model

4.1 Behavioral Model Overview

There are two broad parts of the Learning Design behavioral model:

- Instantiation
- Runtime

4.1.1 Instantiation

There are two main sub-parts to instantiation, in addition to the instantiation of the unit of learning itself:

- Instantiating Roles
- Instantiating Services

4.1.1.1 Instantiating Roles

As a learning design specifies Roles but no specific individuals playing these roles, the main part of instantiating a learning design is the association of particular people with the roles specified in the design. Learning Design Level B Properties may have to be initialized for the various Roles.

4.1.1.2 Instantiating Services

Once the roles are known, generic services can be set up for or with actual members according to the roles associated with the use of the service, taking into account the permissions accorded to each role.

4.1.2 Runtime

The three different Learning Design levels provide increasing levels of sophistication to the runtime behavior of a unit of learning.

4.1.2.1 Learning Design Level A

For Level A, runtime also divides into two broad aspects:

- Activity-structure Sequence and Selection
- Method

Activity-structure Sequence and Selection
An Activity-structure may contain as sub-elements any combination of Activity references, Activity Structure references, and/or Unit of Learning external references. Mechanisms are provided for determining how these are to be delivered.

For Learning Design Level A, an activity-structure element contains three attributes that affect sequencing behavior:

1) *Structure–type*
This has a fixed vocabulary: {sequence, selection}.
This determines whether the sub-elements are to be delivered as a sequence or as a selection. A selection is a structure where users may select and complete the activities contained in any order. As activity-structures may be nested, selections may be nested within other sequences or selections.

2) *Number-to-select*
This is an integer. It determines the number of sub-elements that are to be delivered before the structure-type selection is considered completed. If the structure-type is selection and the number-to-select is 1, then (any) one of the sub-elements is to be delivered. If the number-to-select is not set (or equals the number of sub-elements), then they must all be delivered but in any order chosen by the user.

3) *Sort*
This has a fixed vocabulary of {as-is, visibility-order}
The attribute 'sort' determines the sort-order in relation to the visibility. Default the order in which activities are made visible is in the order specified in the activity-selection structure. When the value is set to 'visibility-order', activities are presented in the order they were made visible by means of conditions or notifications. This can be thought of as imitating a kind of inbox where new activities are 'posted' (made visible) and hence become available over time to users playing associated roles. This is only supported in Levels B and C.

IMS Learning Design Level B extends the control options of an Activity-structure by adding Properties and Conditions, while Level C adds the Notification mechanism, which can be used to create dynamic sequencing.

Method
The *method* governs the running of (the learning design in) a Unit-of-Learning. A Method has one or more *plays*. If there is more than one *play*, they represent logically independent parts of the learning design. The *plays* are therefore always run concurrently. They are always available to their participating actors while the *unit-of-learning* is live.

A *play* has one or more *acts*. The *act*s always run in sequence. This is a basic, top-level, linear sequencing method analogous to the sequence of *acts* in a *play.*

An *act* has one or more *role-part*s. These are associated with one and only one *role*, specifying the part that this *role* plays in the *act* and some activity type which describes the actions that have to be performed and the learning resources that are available for this. A *role* can be played by one or more actors (e.g., actual persons playing the role of learners or support staff). *Role-parts* are always run concurrently, enabling multiple actors to participate in the same *act*. A *role-part* associates a *role* with either an *activity* or with an *environment*, which contains one or more *learning objects* or *services*. The association is always made by a reference to a *role*, and to an *activity* or *environment* element held in the *components* section.

complete-activity = "activity completion rule". This is part of the learning and support activity declarations and specifies when an activity is completed. In Level A completion can either by user-choice, or on reaching a time-limit. In Level B it is extended with *when-property-value-is-set.*

on-completion specifies actions that are to be executed on completion of the activity. In Level A it contains only one element, *feedback-description* which references content to be displayed to the user when they complete the activity. On-completion is further extended by the Level B *change-property-value* element, and by the Level C *notification* element.

4.1.2.2 Learning Design Level B

Level B includes Properties and Conditions. Conditions are used to personalize the presentation of the unit-of-learning. All conditions are pre-conditions defined at design time and must be evaluated during a run:

- When entering the unit of learning (new session).
- Every time the value of a property has been changed. This applies only to the following situations:
  - properties the person has access to in the context of the unit of learning, and
  - the property has to be evaluated in one of the expressions in the unit of learning.

These properties include properties, which are available in the expression, but are set automatically (e.g., time-unit-of-learning-started).

An action is performed (fired) according to the success (true) or failure (false) of the condition. The action is to show or hide various objects, to change a property value, or to notify a role.

The show and hide actions set the visibility attribute (isvisible) of different objects: activities, environments, items, plays, activity-structures, units-of-learning, and different classes of objects (set with the 'class' attribute).

Properties belong to roles (local-role properties), the individual persons in the role (local-personal and global-personal properties), to the run of a unit of learning (local properties) or are global (global properties). They are of five different types and can be collected into property-groups. A property-group can also contain other property-groups, thus allowing arbitrary tree structures to be created.

Properties can be personal or non-personal
Personal properties are key to personalizing a learning design. Given that many actors can play the same role, personal properties, while defined as a part of the role, are assigned to the individual dossier (learning profile or learning record) of each person playing that role. They can then take on separate values for each person according to their state and outcomes when playing that role.

Non-personal properties can be fixed global properties (same value for every person, independent of role or unit-of-learning) or local properties that belong the run of a unit of learning and affect all users, either playing that role, or participating in the unit of learning, respectively. The local properties are always ‘local’, relating to a particular instance or run of a unit of learning (see next).

Properties can also be local or global
Local properties are those created during a particular run of a unit of learning and cease to exist when the unit of learning terminates. They are thus under the complete control of the learning designer who can determine their name, their data type, and the values to be assigned under given conditions.

Global properties persist beyond the duration of the run of a unit of learning. While these can be used by a learning designer to persist value across runs and across different units of learning, they can also be used to access values of persistent learner information. This however is beyond the scope of the Learning Design Specification, and it would look instead to LIP and more immediately Accessibility to provide elements and value types for these global properties.

The five different types of property are:

1) *Local property*, alias: run-property.
This property has the same value in a run for every user. The property is owned by the run of the unit-of-learning. The property has an identifier that can be used to refer to it in this unit-of-learning-package. Operations can refer to this identifier to operate on its value.

2) *Local personal property*
This property can have a different value for every user in all the roles for a run of the unit-of-learning. The property is owned by the user in the context of a run of the unit-of-learning, specifying a value per user. The property has an identifier that can be used to refer to it in this unit-of-learning-package. Operations can refer to this identifier to operate on its value.

3) *Local role property*, alias: group-property.
This property has the same value for every user in the specified role during the run of a unit-of-learning. The property is owned by the role in the run of the unit-of-learning. The property has an identifier that can be used to refer to it in this unit-of-learning-package. Operations can refer to this identifier to operate on the value.

4) *Global personal property, alias: portfolio-property*
This property can have a different value for every user, independent of the different runs of units of learning (its specifies the portfolio of the user).The property is owned by the person. The property has an identifier that can be used to refer to it in any unit-of-learning-package. Operations can refer to this identifier to operate on the value.

5) *Global property*
This property is a globally unique property, which stores one value, independent of user, units-of-learning and role. The property has an identifier that can be used to refer to it in any unit-of-learning-package. Operations can refer to this identifier to operate on the value.

4.1.2.3 Learning Design Level C

Level C adds notification. A notification happens after an event, which is known by the runtime environment. Examples of events, which can trigger a notification, include the completion of an activity, an expression evaluates to true, or a property value is set.

A runtime notification can set the visibility of an activity, which is then activated for a role. If a runtime notification to a particular role sets the isvisible property of a given activity to true, this activity is immediately made available to the role, regardless of any other setting in act, sequence, or condition.

A notification is of the highest priority, meaning that an otherwise invisible item will be made visible and accessible to the user.

The runtime system sends notification messages to roles (affecting all actors playing the role). Notifications are triggered either when certain conditions are met, or when an actor, in a role (typically support staff) with appropriate permissions, sends one. At runtime, depending on the implementation, it may be possible to select a particular actor as the recipient of a notification, but this would be a runtime decision and not a design-time decision as actors are not known at design time. Support activities can select from which actors, playing a given role, it is acceptable to receive notifications from e.g., only the learners in the tutor's tutor group.

A runtime system should keep track of the originator of the notification and make this visible to the receiver of the notification. This context could be used when launching the associated (support) activity, making a choice for an actor superfluous and enabling a sort of challenge/response interaction.

Depending on the implementation, an e-mail message can be sent to the user, notifying them that a new activity has arrived (with a link to that activity contained in the message). The subject field can contain a specific value (otherwise a standard message will be sent).

A notification can be inserted in external vocabularies (after an event like set-property). However, the content must then be provided in the package (because it contains references to identifiers in the package). When the identifier cannot be resolved, the notification is ignored (but does not prevent the xhtml content being presented).

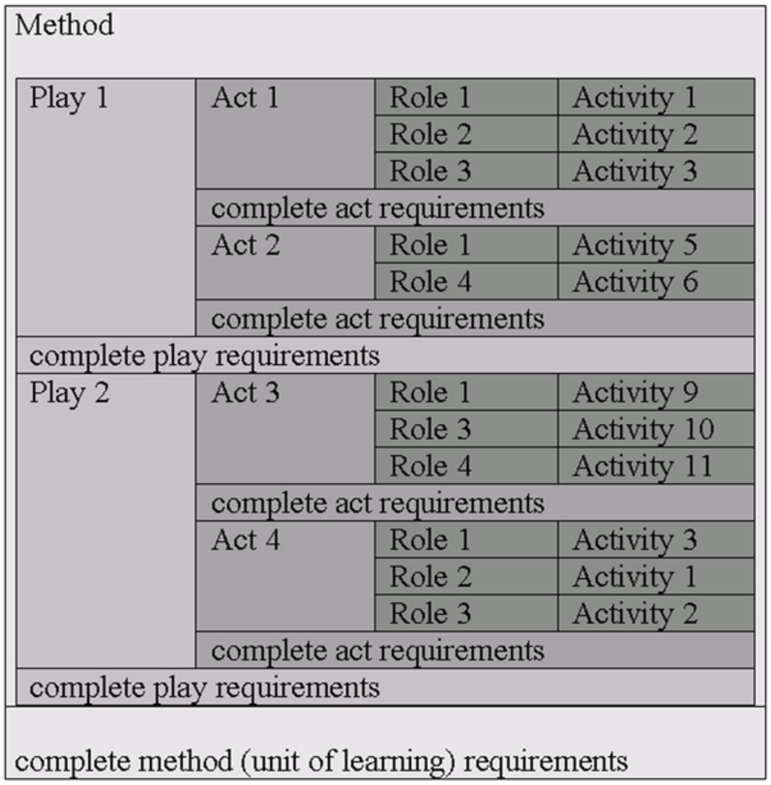
4.1.3 Hierarchy of Control

There are several structures in the Learning Design Specification that have an influence on the visibility of learning activities or other entities. This could mean that there are conflicting visibilities from different mechanisms. To solve possible conflicts there is a hierarchy of control:

- **notify** (LD Level C) a runtime notification can set the visibility of an activity which is then activated.
  - **acts** (LD Level A) determine whether, when, and for what roles an activity, resource structure, or item is to be used.
    - **sequence** (LD Level A) is a type of activity-structure and sets the order of completion for the activities in the sequence. This resets the isvisible attribute settings, regardless of conditions and isvisible attribute initial values.
      - **condition** (LD Level B) may reset the isvisible property of an activity, resource structure, or item, regardless of its current setting.
        - **isvisible** (LD Level A) attribute determines whether an activity, resource structure, or item is displayed to the learner.

This means that 'isvisible' values can be overruled by conditions; conditions can be overruled by sequences; sequences by acts; and acts by notifications. This means that notifications – as the strongest mechanism – can make all types of activities visible that are defined in the components section, independent of any statement in the method section.

A method can be expressed as a table:

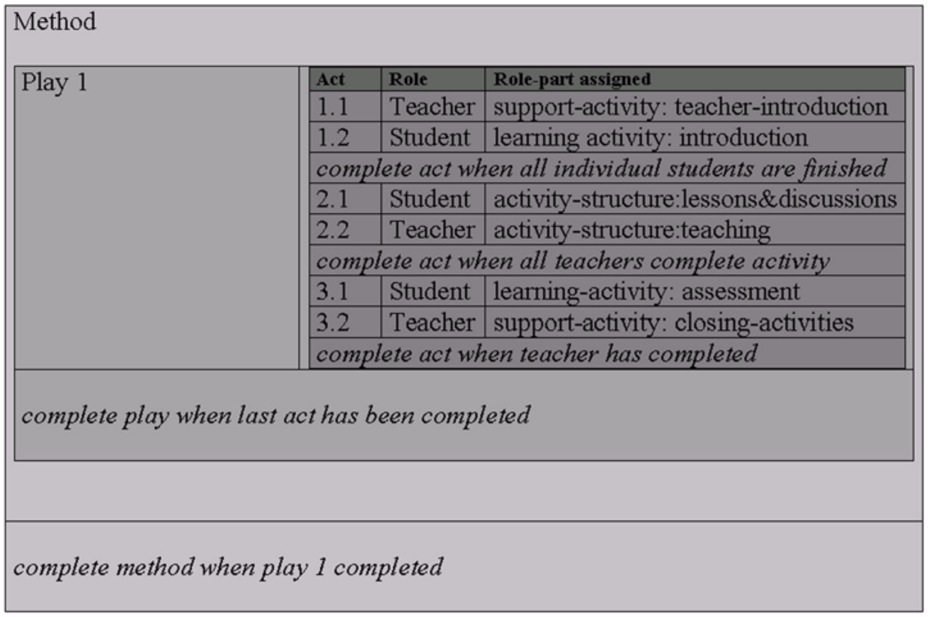


Method

Play 1	Act 1	Role 1	Activity 1
		Role 2	Activity 2
		Role 3	Activity 3
	complete act requirements		
	Act 2	Role 1	Activity 5
		Role 4	Activity 6
	complete act requirements		
complete play requirements			
Play 2	Act 3	Role 1	Activity 9
		Role 3	Activity 10
		Role 4	Activity 11
	complete act requirements		
	Act 4	Role 1	Activity 3
		Role 2	Activity 1
		Role 3	Activity 2
	complete act requirements		
complete play requirements			

complete method (unit of learning) requirements

For example a play can specify the following:



Method

Play 1

Act	Role	Role-part assigned
1.1	Teacher	support-activity: teacher-introduction
1.2	Student	learning activity: introduction
complete act when all individual students are finished		
2.1	Student	activity-structure:lessons&discussions
2.2	Teacher	activity-structure:teaching
complete act when all teachers complete activity		
3.1	Student	learning-activity: assessment
3.2	Teacher	support-activity: closing-activities
complete act when teacher has completed		

complete play when last act has been completed

complete method when play 1 completed

This is represented in a Learning Design Method as follows:

```
<method>
   <play id="play1">
      <act id="act1">
         <role-part id="part11"><role-ref ref="Teacher"/><support-activity-ref
```

```
ref="teacher-introduction"/></role-part>
         <role-part id="part12"><role-ref ref="Student"/><learning-activity-ref
ref="introduction"/></role-part>
         <complete-act><when-role-part-completed ref="part11"/></complete-act>
      </act>
      <act id="act2">
         <role-part id="part21"><role-ref ref="Student"/><activity-structure-ref
ref="lessons&discussions"/></role-part>
         <role-part id="part22"><role-ref ref="Teacher"/><activity-structure-ref
ref="teaching"/></role-part>
         <complete-act><when-role-part-completed ref="part22"/></complete-act>
      </act>
      <act id="act3">
         <role-part id="part31"><role-ref ref="Student"/><learning-activity-ref
ref="assessment"/></role-part>
         <role-part id="part32"><role-ref ref="Teacher"/><support-activity-ref
ref="closing-activities"/></role-part>
         <complete-act><when-role-part-completed ref="part32"/></complete-act>
      </act>
      <complete-play><when-last-act-completed/></complete-play>
   </play>
   <complete-unit-of-learning><when-play-completed ref="play1"/> </complete-unit-of-learning>
</method>
```

Note: id in the example is short for ‘identifier’. The specific names of the identifiers are arbitrary.

Translated in text this means: When entering the unit of learning the play starts with act number 1. All persons assigned to the role ‘Teacher’ get the support-activity ‘teacher-introduction’. At the same time all persons assigned to the role ‘Student’ get a learning activity called ‘introduction’.

Act number 1 is completed when all persons in the role Student have completed the introduction. Than act 2 starts by assigning an activity-structure called ‘lessons & discussions’ to the persons in the role Student and assigning at the same time the activity structure ‘teaching’ to the Teachers. This act is closed when the teacher completes the act (assuming there is a restriction of one person in the teacher role). Than act 3 starts. Etcetera. The play is closed when the last act has been completed.

More than one play can be specified in a method. These plays run in parallel, independent from each other. This is a necessary facility when modelling more complex designs.

Instead of a table a UML activity diagram or a Gantt Chart or (relative) timetable can be used. These means can be a handy starting point for the design.

A UML activity diagram, using a swimlane for each role is illustrated in Figure 4.1.

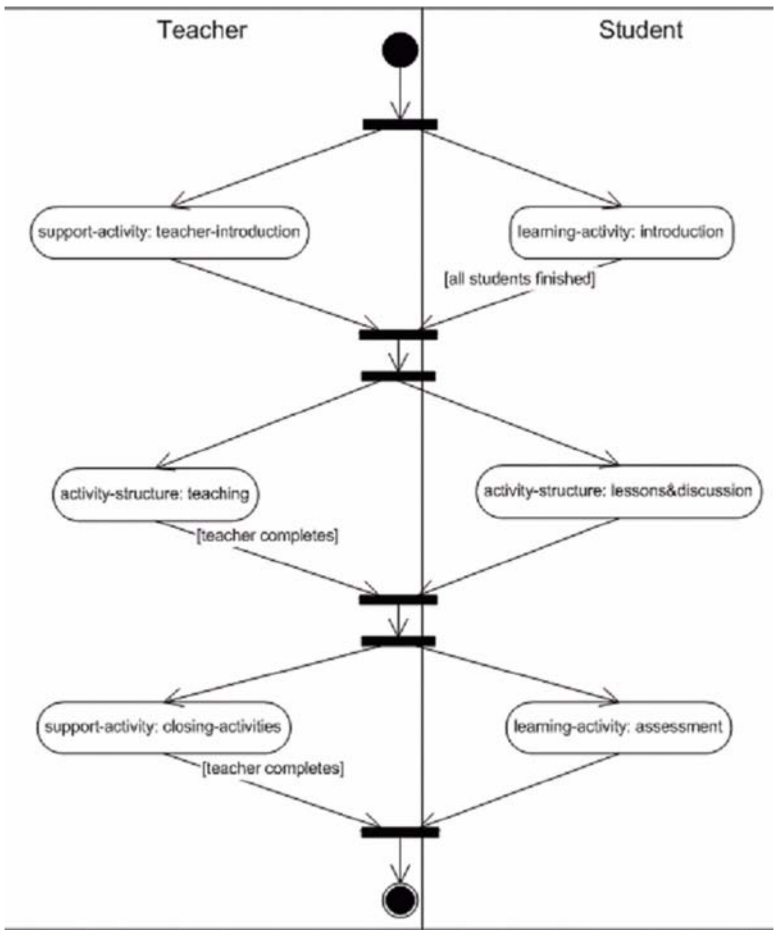

Teacher
Student
support-activity: teacher-introduction
learning-activity: introduction
[all students finished]
activity-structure: teaching
activity-structure: lessons&discussion
[teacher completes]
support-activity: closing-activities
learning-activity: assessment
[teacher completes]

Figure 4.1 - UML activity diagram, using a swimlane for each role.

A Gantt Chart is illustrated in Figure 4.2:

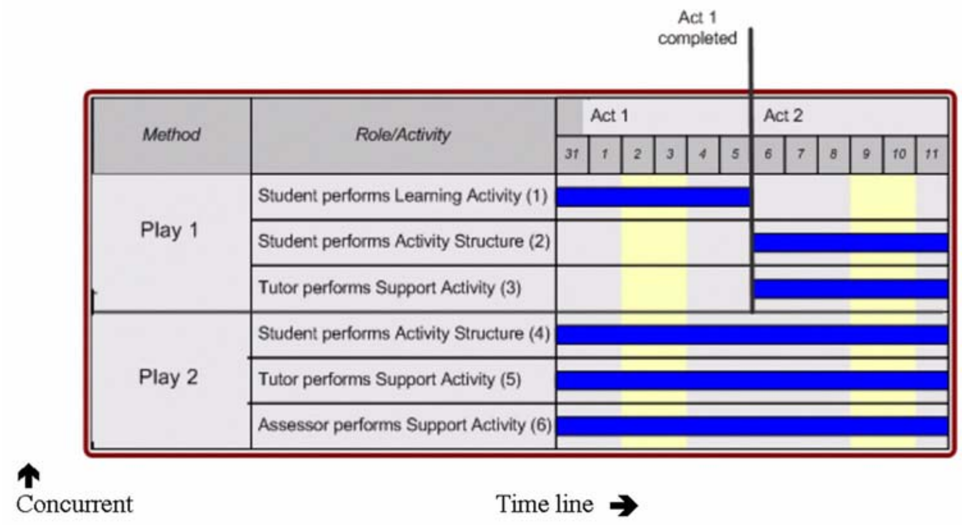


Figure 4.2 - An illustration of a Gantt Chart.

4.2 Instantiating a Unit of Learning

A unit of learning, including a learning design, can be used many times on the same or different systems, in the same or in different training organizations or educational institutions. Each time an instance is created, the runtime service will need to create a certain amount of runtime specific data structures, such as runtime identifier, start-time etc., in order to support the instance and its life-cycle.

A number of steps are needed to translate an IMS Learning Design XML document into a live ‘course’ that a student (and teacher) interact with through, for instance, a browser. These steps have to be implemented in an IMS Learning Design ‘user agent’ or runtime delivery system. It is necessary to specify which medium is to be used to publish the unit (typically the Web, but a unit of learning could be delivered through some other medium or a combination of different media), what media specific elements (video, audio, pictures, etc.) there are, and the availability of support for them, what language is to used, who will participate, and in what roles, the start and end date for the run. Any services, such as announcement channels, chat, or discussions forums have to be set up with the actors that are playing the various roles and their ‘service permissions’ set, as specified in the design. Also, database fields have to be set up to track the values of the properties of the participants and to maintain their ‘dossiers’.

Some of this information is defined in the IMS Learning Design Specification but implementers are likely to want to provide additional runtime information.

4.3 Setting Up by Instantiating Roles and Services

The instance will also be intended for use with particular persons. It will therefore also need to be supplied with information about the people who will be playing the roles of learners and support staff.

For IMS Learning Design Level B, when assigning a person to a role, any properties belonging to a person need to be located in his/her dossier. Any local personal properties, ascribed by the role that each person plays, need to be set up and their default values, if any, set. Role properties, which apply to all members playing that Role, also need to be set up, as do properties that belong to the unit of learning, which apply to all persons playing any role within the unit of learning. There are also global properties, which persist across all units of learning and users, that have to be located and instantiated if this has not been done already.

Once the participants and their roles are known, the services, which depend on this information, can be set up. A unit of learning may be pre-scanned to discover these services and instances of these services set up and allocated a (runtime) URL. An example would be a discussion forum that is dedicated to a particular instance of a unit of learning or to a particular activity within a unit of learning. Within such a forum, different roles may have different permissions with respect to reading, writing, deleting, and editing entries. The IMS Learning Design Specification provides a format for this kind of information, enabling the setup of such services to be automated.

4.4 Activation Process

When activating a unit of learning, the *method* element has to be located within the unit of learning. There must always be one but only one *method* element. This and its sub-elements control the behavior of the unit of learning as a whole, coordinating the activities of the players in the various roles and their use of resources.

This creates a ‘learning flow’, similar to the coordination of activities in a workflow groupware system (but not to the passage of documents in a document-oriented workflow system).

Functionally, a method is made up of one or more play elements. Play elements are functionally independent and run in parallel, so each play element has to be instantiated when the unit of learning is first initialized.

The terminology used to describe the various sub-parts of a Method, Play, Act, Role, and Role-part are drawn from the metaphor of theatre, with the Environment equivalent to the stage set and stage props. The Method, Plays, Acts, and Role-parts are all nested within each other, providing three levels within a Method.

At the top level, the method consists of the 2 elements, play and complete-unit-of-learning. The latter, as described in the Completion Rules section that follows, holds both the condition on which the unit-of-learning is completed and optional actions to be taken when it is completed.

On initialization, all play elements are made active for the members of the roles that participate in it.

Play elements unfold in a series of one or more acts, which are always run in sequence. No act is made visible to the role/s that play a part in it until the previous act has completed. This factor can be used to synchronize the activities of those playing roles within the play.

The activities associated with the roles within a play may have complex sequencing, so it is possible to have a ‘one-act’ play which has internally complex sequencing, as long as the activities of the players do not need synchronizing (e.g., when the use of a learning resource is completed, all participants move to a discussion forum).

A play has an identifier and an isvisible property. It also has a title and metadata. The act or acts constitute the main body of the play, and the complete-play specifies the conditions of completion and the optional actions to be taken when the play is complete.

Although within an act there may be complex sequencing, and there may be only one active act, if there is more than one act within a play, the acts are run in sequence.

Thus acts can be used as synchronization points, either waiting for all players to finish before starting the next, or forcing an end when a certain number of players have finished, when terminated by a support staff member or under some other condition. All role participants in the next act may then start together at the same time, subject to their being simultaneously logged into the runtime service. When the last act is completed, the unit of learning is also complete.

An act brings together one or more role-parts. This is the mechanism that allows more than one role to perform at the same time. Therefore role-parts within an act are always run in parallel.

Role-parts enable several users, playing the same or different roles, to participate in the same act. Each role-part associates exactly one role with exactly one type of activity (including the performance of another unit-of-learning and activity-structures), or with one environment (equivalent to an organization in Content Packaging). Multiple role-parts within one act, are performed concurrently.

The same role can be associated with different activities or environments in different role-parts, and the same activity or environment can be associated with different roles in different role-parts. However, the same role may only be referenced once in the same act. If multiple activities or environments need to be associated for the same role an activity-structure or wrapper environment should be used.

When an act within a play is activated, all the role-parts in the act go ‘on-stage’ or become live. Depending on the implementation, players of the roles referenced by the role-parts may then have their associated activities (or environments) made visible in their ‘activity-tree’ and any content associated with the activity made accessible. However, if an activity or items isvisible attribute is set to ‘false’, the link in the activity-tree might be made visible, but the content not made accessible.

4.5 Completion Rules

At every level within a method, it is possible to specify the rules when a role-part, act, play, or unit-of-learning is completed. It is expected that the runtime system keep a record of the completion status of these different entities. The completion status can be retrieved using certain constructs in this specification.

At the lowest level within a method role-parts have to be completed. The activity which a role-part references may be a learning-activity, a support-activity, an activity-structure, or (sub) unit-of-learning. They are completed when the *complete-activity* conditions are met (for learning/support activities), or when an activity-structure is completed or when a referenced unit of learning is completed.

Activity-structures, while they may include (sub)activity-structures and (sub)unit-of-learning references, ultimately resolve down to basic learning or support activities. An activity-structure with type set to *sequence* is completed when the last referenced entity is completed. An activity-structure with type set to *selection* is completed when all the containing referenced entities are completed or when the number of entities set in *number-to-select* are completed. The completion of an activity-structure or (sub)unit-of-learning is thus determined by the completion of the 'atomic' learning and support activities contained within it. Where there are sub-structures, their completion in turn is 'rolled up' from the completion of their constituent parts.

The completion of an 'atomic' learning or support activity is determined either by user choice or when a time limit is reached (Level A). When no explicit completion rule is specified the completion is set to unlimited, meaning that it is always completed. In Learning Design Level A, the next three levels up, act, play, and unit-of-learning, each have three completion options.

At the next level up from a role-part, an act is completed either when one or more referenced role-parts are completed, or by user choice, or when a time limit is reached. At the next level up again, a play is completed either when the last (final) act is completed, or by user choice or when a time limit is reached. Finally, at the top level, the unit-of-learning is completed either when one or more referenced plays are completed, or by user choice, or when a time limit is reached.

Note: Completion of a higher-level element, by either user choice or time limit, terminates all lower level components contained within it.

Learning Design Level B adds the when-property-value-is-set option as an addition to the three above Level A options for the completion of an act, a play, and a unit-study. The when-property-value-is-set element contains a property-reference and a property-value. When the referenced property is set to the value specified in the property-value, the condition evaluates to true and the act, play, or unit-study is set to completed.

The completion conditions for the act, play, and unit-of-learning are all the same, with the exception of the first condition which is unique to each.

Level B adds the further option of completion being triggered by a property being set to a specified value. The when-property-value-is-set completion condition is triggered when the specified property is set to the specified value. The property value to test against can be specified either as a literal value, as a calculated value, or as the value of another property.

4.6 On Completion

When a learning activity, support activity, act, play, or unit-of-learning is completed, actions to be performed are contained in the on-completion element. The on-completion options are the same for a learning activity, a support activity, an act, a play, and a unit-of-learning. A role-part is considered complete when its referenced activities are complete and has no on-completion element of its own. Likewise, an activity-structure is considered complete when its constituent sub activities are completed and has no on-completion element of its own.

In Learning Design Level A, the only task that can be specified to be carried out on completion is that of providing feedback to the user through the feedback-description element. This points to a resource where the feedback description can be found. After completion of an activity this Web page is to be displayed to the user.

Learning Design Level B adds the option of changing one or more property values through the change-property-value element. The change-property-value element (also) contains a property-reference and a property-value. This specifies that on completion, the referenced property is to be set to the value specified in the property-value.

Learning Design Level C adds the option of sending one or more notifications. (See the Behavioral Model Overview for a description of Notifications).

4.7 Recording Outcomes and their Mapping to IMS LIP

The results or outcomes of each learner's activities in the Unit of Learning will typically be recorded and maintained by the runtime service. The learning design presupposes some form of 'dossier' or learner record that is used to hold and maintain the various personal properties that are part of Learning Design Level B. Local personal properties need to be maintained only for the duration of the unit of learning, although this could run across multiple sessions, which together might span many days. Global personal properties are intended to persist indefinitely and should become part of a learner’s permanent learning record.

The learning design does not specify how properties and their values and their aggregation for each learner should be recorded by the runtime service.

However it is worth noting that the IMS Learning Design properties and property groups map directly into the IMS LIP Specification’s Activities and Evaluations elements. This means that the IMS LIP Specification can be used to transport the outcomes generated by learners in their use of units of learning between different systems should this be needed.

5. Extensibility

The XML binding of a Learning Design may be extended through the use of XML Namespaces and XML Schemas, to allow developers the most flexibility possible. Elements that contain data types (e.g., string, integer) and elements with a “closed” data model may not be extended. Extensions must provide references (via namespacing) to the source of the extensions.

The information model already indicate placeholders, which could be replaced and/or extended with other schemas.

There are at least two cases where extensions can cause problems for developers. The first case is when interoperability with other content packaging tools and vendors is required. Custom extensions must then be agreed upon between individual parties making global interoperability very difficult. The second case is when a developer wishes to add extensions and also provide or alter a schema that will allow document validation. Each schema (DTD, XSD) requires a different approach to handle extensions that can be validated.

Appendix A – Glossary

In this document, the following terms are used with these associated specific meanings. If you are used to a different usage for the term, or a different term for what is being defined, please translate accordingly as you read. The terms defined here are no part of the learning design vocabulary as they are specified in the conceptual model elsewhere in this text.

Announcement Conference	An announcement is a message sent to users to inform them about new events or relevant information. Announcements are declared in the environment/service/conference object with the conference-type set to ‘announcement’.
Asynchronous Conference	Asynchronous conferences are group-messaging systems that use a store (inbox) for incoming messages. These are normally ordered in (nested) topics (conferences). The most primitive asynchronous conferencing system is internet news (nntp).
Attribute	An attribute is a parameter to an element declared in the DTD. An attribute’s type and value range, including a possible default value, are defined in the DTD.
Document	A document is a stream of data that, after being combined with any other streams it references, is structured such that it holds information contained within elements that are organized as defined in the associated DTD.
DTD	A DTD, or document type definition, is a collection of XML declarations that, as a collection, defines the legal structure, elements, and attributes that are available for use in a document that complies to the DTD.
Element	An element is a document-structuring unit declared in the DTD. The element’s content model is defined in the DTD, and additional semantics may be defined in the prose description of the element.
Facilities	Functionality includes elements, attributes, and the semantics associated with those elements and attributes. An implementation supporting that functionality is said to provide the necessary facilities.
Implementation	An implementation is a system that provides collection of facilities and services that supports this specification.
Parsing	Parsing is the act whereby a document is scanned, and the information contained within the document is filtered into the context of the elements in which the information is structured.
Property Operation	A term used to refer to one of the following operations on properties or property groups: set-property, view-property, set-property-group, view-property-group, change-property-value.
Rendering	Rendering is the act whereby the information in a document is presented. This presentation is done in the form most appropriate to the environment (e.g. aurally, visually, in print).
Run of a Unit of Learning	A unit-of-learning describes a class of possible instances. These instances of a unit of learning are called a ‘run’. In a run, concrete persons are bound to the roles defined in the unit of learning and a concrete start date of the learning process is defined. The same unit of learning with the same identifier can have an unlimited number of runs. When having the same identifier it is expected to have exactly the same structure and content. The identifier of the units of learning discriminates between versions of content and structure (learning design). Note that the runtime system is also likely to create separate identifiers for each runtime instance. This however is an implementation design issue.
Runtime	The facility used to interpret the IMS LD Specification for users.

Synchronous Conference	Synchronous conferences are group communication systems which enables groups to communicate and work with each other in real time. Mostly through a variety of media, but the most primitive once use one media type (e.g. chat and telephone conferences). More complex systems combine synchronous and asynchronous conferences. These are also classified here as synchronous conferencing systems.
URI	Unique Resource Identifier. Specification from IETF, annotated by W3C (see references). In this DTD URIs are used in the meaning of the W3C annotation. This annotation does not make a strict distinction between URLs and URNs. Every URI can be a URL and a URN depending on implementation specific conventions. URIs can be absolute or relative. Absolute URIs are global, relative URIs are local. For resources with local URIs it is expected that the resource is available in the unit-of-learning package when the resource is dependent on one or more files.
User Agent	A user agent is an implementation that retrieves and processes IMSLD documents. It is identical to a runtime system. It is indifferent where processing takes place: at the client or server side.
Validation	Validation is a process whereby documents are verified against the associated DTD, ensuring that the structure, use of elements, and the use of attributes are consistent with the definitions in the DTD.
Web Content	Is a data type for a specific resource: any content which could be hosted in, or launched within a Web browser, like html, xml, flash, applets, text processor/spreadsheet files, etcetera. Whether files are launched in the browser which cannot be hosted in the browser, depends on the user client. etcetera. Webcontent does not necessarily have to be well-formed XML (e.g., HTML is not).
Well-Formed	A document is well-formed when it is structured according to the rules defined in Section 2.1 of the XML 1.0 Recommendation (http://www.w3.org/TR/xhtml1/#sec-well-formed). Basically, this definition states that elements, delimited by their start and end tags, are nested properly within one another.

Appendix B – User Agent Compliance

Learning Design offers three levels of compliance, referred to as Levels A, B, and C. Level A builds on IMS Content Packaging. Therefore compliance with level A implies support for all IMS Content Packaging constructs. Learning design Level B builds on Level A. Therefore compliance with Level B also implies compliance with Level A. Similarly, Level C builds on Level B. Therefore compliance with Level C implies compliance with Levels B and A.

A user agent is IMSLD compliant when it has facilities for all of the elements and attributes specified in the XML Schema provided for Profile 3, 4, or 5 (see next appendix) in conformance with the descriptions contained in this specification. For compliance at each Level, IMS expects certain runtime behaviors from a user agent:

B.1 - Level A Compliance

1) A unit-of-learning-package has all the files needed to create one or more runs from this unit-of-learning. The URI of a unit-of-learning identifies the package uniquely, including the update versioning.
   During the run of a unit-of-learning-package the learning-design may not be updated, but the organization structure and the physical local files delivered in the package may be updated without affecting the run status for users.
   When a new version of the local files and/or the learning-design has been created, a new URI has to be created for the unit-of-learning-package. Depending on the implementation several types of information can be stored in the URI (e.g. identifier + type + version). During the run, the new package with the new URI can be published over the run (updating resources and files) or a new run can be created. Facilities to import, publish, set start-dates, manage users in roles, update existing runs and create new runs are expected to be provided with the runtime system.
2) When interpreting IMSLD, the runtime reads the learning-design/method/play element structure (called ‘play’). When more than one play is specified these are interpreted concurrently. A play has one or more acts and an act has one or more role-parts. At every level there are explicit rules specified how a role-part, act, play and unit-of-learning is completed. It is expected that the runtime keeps record of the completion status of these different entities. The completion status is retrieved by some IMSLD constructs.
   The role-part completion has to be derived from activity completions or unit-of-learning completion (when the role-part refers to a unit-of-learning). When no explicit completion rule is specified the completion is set to unlimited, meaning that it is always completed.
3) Per play the acts are interpreted in the order specified. Only one act per play can have the focus at any time, starting with the first act in the play. When the act is completed, the next act gets the focus, until all the acts in the play are completed.
4) The role-parts specify which roles should be able to access what activities. When there is more then one role-part in a play, the role-parts are accessible concurrently by the different roles.
5) Every user that is in runtime bound to a role should have access to the activities that are made accessible and visible for that role. Users can be bound to roles at runtime (new users added, existing users deleted).
6) When a role is tagged to allow for the creation of new roles, the visibility rules and the users for the parent are applied to the children. Within these rules the creator of the new role is allowed to regroup the existing users in the parent role over the newly created roles.
7) The isvisible attribute on elements is interpreted as the initial visibility value. (Level B Conditions and Level C Notifications can change this value to true, when show then applies, or to false, when hide then applies).
8) Meta-data of the learning-design are always accessible in the user-interface. In the user-interface specification a format for all the elements has to be specified. When not specified it is shown in the format [element-name]: [element-value].
9) Meta-data of other elements can be made accessible and visible, depending on the user-interface design (implementation dependent).
10) The title of the learning design is always made visible somewhere in the user interface and can never be changed by templates.

11) Other title elements can be rendered in the interface when needed in the implementation. Templates can overrule the existing value.
12) The learning-design/learning-objectives/item(s) and /prerequisites/items(s) must be accessible for all the roles, at all times in the user-interface. They may require a user-action (e.g. opening a menu, clicking a button or link). They must semantically be presented as learning objectives respectively prerequisites.
13) A user must always be aware in which role he/she is in. When a user is assigned to more than one role, he/she must be able to see to which roles he/she is assigned and be able to switch roles at any time. The information-for-role/item(s) must always be accessible with the roles for a user.
14) Learning-activities and support-activities are rendered in the user-interface in such a way that users always know in which learning activity they are (by rendering the activity title), where it fits in the sequence or selection of a series of activities, what the activity-description is, which learning-objectives and prerequisites are connected, which environment (and content of the environment) is connected, how to complete the activity (including controls when needed) and have access to any feedback-description on completion of an item.
15) When activity-structures are presented to the user, the user-agent must interpret an activity-structure with structure-type=sequence in such a way that: the content of the 'information' element is made visible to the users, the connected environment (and its content) is available for the user and the activities are made accessible one by one in the specified sequence. The next comes available when the previous one is completed. An activity-selection is presented in such a way that the user sees the 'information', the connected environment and can access all the containing activities including a cue to inform the user when restrictions are set on the maximum number of items to be selected. The sequence is completed when the last activity or unit of learning in the row is completed (including sub-sequences or sub-selections). Activity-structures with type=selection are completed when the user has completed the number of items that should be completed (when the number-to-select is not set, all the activities in the selection must be completed). In all situations a user must see what type of activity (learning, support, substructure or unit-of-learning) he/she can access at the moment he/she sees the link to the activity.
16) Support activities can be recurrent for every user in the supported role (when the role-ref is specified), the user-interface must represent this fact and make explicit at any time for which user or group of users the support-activity is performed. The user may select one or more of the users in the supported role.
17) Environments are connected to activities, activity-structures or roles (in a role-part). When an activity-description is visible, always the connected environment (including the content structure of the environment) must be made visible. It must be possible to access and see the activity-description and the content of one of the objects or services within the environment at the same time.
18) Every learning-object or service in the environment has its own design requirements. The implementer is free to implement the learning-objects and services, taking care of the representation of all the elements and the functionality they represent as described with the elements in this information model.
19) Resources are not shown directly in the user-interface. They define the collection of resources where the 'item' elements in the learning-design refers to. Locally defined resources and files (by using relative URIs in the href) are expected to be present in the unit-of-learning-package. Absolute URIs refer to resources outside of the package. A change of an external resource does not affect the unit-of-learning-package (the package is not aware of the presence and change of external resources).
20) The element 'item' occurs at different places in the information model. The items are referring to resources in the unit of learning package. The content of the resources must be rendered in the user-interface in such a way that the content itself is visible, including the item structure when there is more then one item. Also the item titles must be visualized. Providing a table of content and sectioned text can do the latter.
    The nesting level of the items must be made visible to the user (e.g. by providing nested headings).

B.2 - Level B Compliance

1) Conditions work within the scope of an act that has the focus, plus the history acts. Conditions can never make activities visible, which are not specified in one of the role-parts of the current act in the current plays. So in priority the acts have a higher priority then the conditions.

2) Conditions are evaluated any time the user accesses trees or content, when the system has an online connection. But only those conditions have to be evaluated which affect the selected content or trees. Also only those conditions have to be evaluated which refer to properties in the IF expression, which values have been changed after the last time the same property value was accessed by the user.

3) The user-agent is expected to keep record of property-values and property-definitions for users and roles in a so-called ‘dossier’. There are several types of properties.
Local properties are stored with a scope local to the run of a unit of learning. They are defined and used in the unit-of-learning. Global properties are accessible outside the context of a unit of learning (e.g., by more than one unit of learning). They can be defined in one unit of learning en used in another one. In IMSLD global properties can be defined. Runtimes are expected to control whether a defined global property URI already exists or not. Global properties - once defined - may never change definition. So when the property already exists the definition is ignored.
Personal properties are owned by a person (local or global) and role properties are owned by a role (local or global). Dossier properties are defined and/or declared (for already defined global properties) under learning-design/roles/properties and operated upon with property-operation elements (view-property, set-property, conditions, etc.). User-agents are expected to operate on properties in a secure manner and with a maximum performance (to be detailed by the implementer).
When a property value changes a Level C notification may be sent to a role (depending on the declaration).

B.3 - Level C Compliance

1) When a notification is sent, the connected activity is always made visible to the users in the role indicated by the notification. If the role is playing a role-part in the current act (which has the focus), the activity is made visible within the structure (tree) of the current act. If the role is not included in the current or an earlier act, the activity is added to the tree as a separate node directly above or below the activities from the current act. The user is to be alerted when a new notification has arrived.

2) Notifications have a higher priority than acts and acts have a higher priority than sequences. Sequences have a higher priority than conditions.

3) When there are conflicting show and hide conditions, the rule is that show is of a higher priority then hide (and will overrule the hide).

About This Document

Title	IMS Learning Design Information Model
Editors	Rob Koper (Open University of the Netherlands), Bill Olivier (CETIS/JISC), Thor Anderson (IMS)
Team Co-Leads	Chuck Barritt (Cisco), Katy Campbell (University of Alberta/Industry Canada)
Version	1.0
Version Date	20 January 2003
Status	**Final Specification**
Summary	This document describes the conceptual, information, and behavioral models of the IMS Learning Design Specification.
Revision Information	20 January 2003
Purpose	Defines the IMS Learning Design Information Model.
Document Location	http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html

List of Contributors

The following individuals contributed to the development of this document:

Name	**Organization**
Steve Grocott	Can Studios, Ltd.
Mike Halm	Penn State University
Paul Lefrere	JISC
Jocelyn Manderveld	Open University of the Netherlands
Jon Mason	IMS Australia - DEST
Mark Norton	IMS
Claude Ostyn	Click2Learn, Inc
Dan Rehak	Carnegie Mellon University
Peter Sloep	Open University of the Netherlands
GT Springer	Texas Instruments
Brian Taliesin	Microsoft
Robert Todd	Digital Think
Brendon Towle	NETg
Hubert Vogten	Open University of the Netherlands
Carol Washburn	University of Pittsburgh
Bria White	Texas Instruments

Revision History

Version No.	Release Date	Comments
Base 1.0	01 April 2002	
Public Draft 1.0	16 September 2002	The first formally released version of the IMS Learning Design Specification.
Final 1.0	20 January 2003	Minor edits and typographical changes were made to address comments raised during public draft review.

Index