INFORMATION SOCIETY TECHNOLOGIES (IST) PROGRAMME

Diogene

D8.2 Standard Monitoring Second Report

Prepared for the European Commission
under Contract No. IST-2001-33358 as a deliverable from

WP 8: Standard Monitoring and System Upgrading

Date of issue: 1/10/2004
Version: 1.0
Distribution: public

Revision History

Date	Version	Circulation	Reviewed paragraphs	Authors	Responsible Partner	Approved by
1/10/2004	1.0	public	Initial Version	DIIMA	DIIMA	Enver Sangineto

Table of Contents

1. INTRODUCTION 5

1.1. REFERENCES 5
1.2. INVOLVED ORGANIZATIONS 8

2. STANDARDS FOR METADATA AND TEST REPRESENTATION 8

2.1 IMS METADATA 8
2.2 IEEE LEARNING OBJECT METADATA 9
2.3 DUBLIN CORE 9
2.4 IMS QUESTION AND TEST INTEROPERABILITY (QTI) E QTI LITE 11
An Overview of QTI 11
QTI Lite 13
main Differences between QTI and QTI Lite 13
2.5 COMPARISON AND CHOICE 15

3. STANDARDS FOR COURSE REPRESENTATION 16

3.1 IMS CONTENT PACKAGING 16
IMS Content Packaging Conceptual Model 16
3.2 SCORM CONTENT AGGREGATION MODEL 17
IEEE ECMA SCRIPT APPLICATION PROGRAMMING INTERFACE FOR CONTENT TO RUNTIME SERVICES COMMUNICATION (1484.11.2) (PART OF SCORM 1.3) 19
IMS CONTENT PACKAGING SPECIFICATION VERSION 1.1.3 (PART OF SCORM 1.3) 19
IEEE DATA MODEL FOR CONTENT OBJECT COMMUNICATION 1484.11.1 (PART OF SCORM 1.3) 19
3.3 AICC CONTENT STRUCTURE MODEL 20
3.4 COMPARISON AND CHOICE 22

4. STANDARDS FOR LEARNER MODELLING 23

4.1 IMS LEARNER INFORMATION PACKAGE (LIP) 23
Core Data Structures 23
Relationship with the IEEE LTSC PAPI Specification 24
4.2 IMS LEARNER INFORMATION PACKAGE ACCESSIBILITY FOR LIP 25
4.3 IEEE PAPI LEARNER 25
4.4 SABA PROFILE FORMAT 28
4.5 COMPARISON AND CHOICE 29

5. STANDARDS FOR ONTOLOGY MODELLING 30

5.1 W3C RESOURCE DESCRIPTION FRAMEWORK (RDF) AND RDF-SCHEMA 30
5.2 W3C OWL (LITE, DL E FULL) 31
5.3 UNIVERSITY OF MARYLAND SHOE 34
5.4 DARPA DAML AND DAML+OIL 35
OIL 35
5.5 COMPARISON AND CHOICE 36

6. STANDARDS FOR COMPETENCIES MODELLING 36

6.1 IMS REUSABLE DEFINITION OF COMPETENCY OR EDUCATIONAL OBJECTIVE (RDCEO) 36
6.2 IEEE LEARNING TECHNOLOGY COMPETENCY DEFINITIONS 37
6.3 COMPARISON AND CHOICE 37

7. STANDARDS FOR MODELING OF LEARNING ACTIVITIES 38

7.1 IMS SIMPLE SEQUENCING 38
7.2 IMS LEARNING DESIGN 40
Compliance Levels *40*
basic Models *41*
7.3 UNED UNIVERSITY'S PALO 41
7.4 OPEN UNIVERSITY OF NEDERLANDS' EDUCATION MODELLING LANGUAGE (EML) 42
7.5 COMPARISON AND CHOICE 43

8. STANDARDS FOR LMS INTEROPERABILITY 44

8.1 ADL SCORM RUNTIME ENVIRONMENT 44
Runtime Data Model and API *44*
8.2 AICC CMI GUIDELINES FOR INTEROPERABILITY 45
8.3 IEEE DATA MODEL AND ECMA SCRIPT API FOR CONTENT TO LMS COMMUNICATION 46
8.4 IMS ENTERPRISE SPECIFICATION AND ENTERPRISE SERVICES 47
8.5 COMPARISON AND CHOICE 49

9. STANDARDS FOR LEARNING OBJECT REPOSITORIES 49

9.1 IMS DIGITAL REPOSITORIES 49
9.2 EDUTELLA 50
9.3 COMPARISON AND CHOICE 51

10. OTHER STANDARDS FOR E-LEARNING 51

10.1 IMS SHAREABLE STATE PERSISTENCE 51
10.2 IMS VOCABULARY DEFINITION EXCHANGE (VDEX) 53
10.3 IEEE DIGITAL RIGHTS EXPRESSION LANGUAGE 54
10.4 COMPARISON AND CHOICE 55

11. STANDARDS FOR SEMANTIC REPRESENTATION OF SERVICES 55

11.1 DARPA DAML-S / W3C OWL-S 55
11.2 COMPARISON AND CHOICE 57

12. CONCLUSIONS 57

D8.2 ***Standard Monitoring Second Report***

1. INTRODUCTION

For standards already applied in Diogene it will contain the description of novelties introduced by new versions and a cost/benefit estimation of their upgrade
For standards not yet applied in Diogene it will include a critical review and a cost/benefit estimation of their adoption

This survey has the following structure:

- STANDARDS FOR METADATA AND TEST REPRESENTATION
- STANDARDS FOR COURSE REPRESENTATION
- STANDARDS FOR LEARNER MODELLING
- STANDARDS FOR ONTOLOGY MODELLING
- STANDARDS FOR COMPETENCIES MODELLING
- STANDARDS FOR MODELING OF LEARNING ACTIVITIES
- STANDARDS FOR LMS INTEROPERABILITY
- STANDARDS FOR LEARNING OBJECT REPOSITORIES
- OTHER STANDARDS FOR E-LEARNING
- STANDARDS FOR SEMANTIC REPRESENTATION OF SERVICES
- CONCLUSIONS

1.1. REFERENCES

- [1] CRMPA DIOGENE Glossary, 2002.
- [2] IEEE P1484.12 Learning Object Metadata Working Group. LOM Approved Working Draft 4. (http://ltsc.ieee.org/doc/wg12/LOM_WD4.PDF).
- [3] IMS LIP Information Model Specification, 2001, (http://www.imsproject.com/profiles/lipinfo01.html).
- [4] IMS Learning Resource Meta-Data Best Practice and Implementation Guide, 2001, version 1.2.1, Final Specification (http://www.imsproject.org/specifications.html).
- [5] IMS Question and Test Interoperability: an overview, 2001. Public draft specification version 1.2 (http://www.imsproject.org/specifications.html).
- [6] L. Aiello and A. Micarelli, 1990. SEDAF: An intelligent educational system for mathematics. Applied Artificial Intelligence- An International Journal 4 (15), pp. 15-36.
- [7] J. R. Anderson, C. F. Boyle and G. Yost, 1985. The geometric Tutor. Proceedings of the IJCAI,. Los Angeles, CA.
- [8] J. R. Anderson and B. J. Raiser, 1985. The LISP Tutor. Byte 10 (4).
- [9] T. Berners-Lee. Semantic Web Road map. World Wide Web Consortium Archive.

http://www.w3.org/DesignIssues/Semantic.html.
- [10] T. Berners-Lee. Interpretation and Semantics on the Semantic Web. World Wide Web Consortium Archive (http://www.w3.org/DesignIssues/Interpretation.html).
- [11] D. G. Bodrow and T. Winograd, 1977. An overview of KRL, a knowledge representation language. Cognitive Science 1, pp. 3-46.
- [12] B. Bos, 1999. XML in 10 points. World Wide Web Consortium Archive. (http://www.w3.org/XML/1999/XML-in-10-points.html).
- [13] R. J. Brachman and H. J. Levesque, 1984. The tractability of subsumption in frame-based description languages. In Proceedings of AAAI-84, 4th Conference of the American Association of Artificial Intelligence, pp. 34-37.
- [14] R. J. Brachman and H. J. Levesque editors, 1985. Readings in knowledge representation. Morgan Kaufmann, San Matteo CA.
- [15] M. Cialdea, A. Micarelli, D. Nardi, J. C. Spohrer and L. Aiello, 1990. Meta-level reasoning for diagnosis in intelligent tutoring systems. Technical Report, Dipartimento di Informatica e Sistemistica, Universita' di Roma "La Sapienza".
- [16] S. Cranefield, 2001. UML and the Semantic Web.
- [17] S. Cranefield and M. Purvis, 1999. UML as an Ontology Modelling language. Proceedings of the Workshop on Intelligent Information Integration, 16th IJCAI.
- [18] F. Donini, M. Lenzerini, D. Nardi and A. Schaerf, 1992. From subsumption to instance checking. Technical Report, Dipartimento di Informatica e Sistemistica, Universita' di Roma "La Sapienza".
- [19] F. Donini, M. Lenzerini, D. Nardi and A. Schaerf, 1996. Reasoning in Description Logics. In G. Brewka editor, Principles of Knowledge Representation and Reasoning, Studies in Logic, Language and Information, pp. 193-238. CLSI Publications.
- [20] D. Dubois and H. Prade, 1980. Fuzzy sets and systems. Theory and Applications. Academic Press.
- [21] D. Fensel, F. van Harmelen, I. Horrocks: OIL: A Standard Proposal for the Semantic Web. Deliverable 0 in the European IST project OnToKnowledge. (http://www.ontoknowledge.org/oil/downl/otk.del02.pdf).
- [22] Genesereth. Logical Foundation of Artificial intelligence.
- [23] M. R. Genesereth and R. E. Fikes, 1992. Knowledge Interchange Format, Version 3.0 Reference Manual. Computer Science Department, Stanford University, Technical Report Logic-92-1, June 1992.
- (http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps).
- [24] J. R. Hartley and D. H. Sleeman, 1973. Towards intelligent teaching systems. International Journal of Man-Machine Studies, 5, pp. 215-236.
- [25] J. Heflin, J. Hendler, S. Luke. SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078 (UMIACS TR-99-71). 1999. (http://www.cs.umd.edu/projects/plus/SHOE/pubs/techrpt99.pdf).
- [26] R. Iannella, 1998. An idiot's guide to the Resource Description Framework. The new Review of Information Networking 4.
- [27] A. N. Kolomogorov, 1956. Foundations of the theory of probability. Chelsea, NY.
- [28] H. J. Levesque and R. J. Brachman, 1987. Expressiveness and tractability in knowledge representation and reasoning. Computational Intelligence 3, pp. 78-93.
- [29] J. W. Lloyd, 1986. Foundations of Logic Programming. Springer Verlag, Berlin.
- [30] G. F. Luger and W. A. Stubblefield, 1998. Artificial Intelligence. Structures and strategies for complex problem solvers. Addison-Wesley Longman.
- [31] A. Micarelli, F. Mungo, F. S. Nucci and L. Aiello, 1991. SAMPLE: An Intelligent Educational System for Eletrical Circuits. Journal of Artificial Intelligence in Education, 2 (3), pp. 83-99.
- [32] E. Miller, 1998. An introduction to the Resource Description Framework (http://www.dlib.org/dilib/may98/miller/05miller.html).
- [33] M. Minsky, 1975. A framework for representing knowledge. I P. J. Winston editor, The Psychology of Computer Vision, McGraw-Hill NY pp. 211-277.
- [34] J. F. Nicaud and M. Vivet, 1988.Les tuteurs intelligents: Realisations et tendance de recherches. In Technique et Science Informatiques, 7 (1), pp. 21-45.
- [35] W. Paper, 2001. Capitalizing on the Learning Object Economy. Learning Objects Network.
- [36] R. Reiter, 1980. A logic for default reasoning. Artificial Intelligence 13, pp. 81-132.

- [37] E. Rich and K. Knight. Artificial Intelligence, second edition. McGraw-Hill, 1991.
- [38] S. Russel and P. Norvig. Artificial Intelligence, a modern approach. Prentice Hall, 1995.
- [39] F. Sebastiani, 1996. Logica e Rappresentazione della Conoscenza. Servizio Editoriale Universitario di Pisa.
- [40] D. H. Sleeman, 1987. PIXIE: A shell for developing intelligent tutoring systems. In Artificial Intelligence and Education: Learning environments and tutoring systems, R. H. Lawler and M. Yazdani Eds. Norwood NJ: Ablex Publishing, pp. 239-265.
- [41] H. Stuckenschmidt and H. Wache, 2000. Context Modelling and Transformation for Semantic Interoperability. In: Knowledge Representation meets Databases - Proceedings of the Workshop at ECAI 2000. (http://www.tzi.de/~heiner/public/KRDB-00.pdf).
- [42] L. A. Zadeh, 1975. Fuzzy logic and approximate reasoning. Synthese 30.
- [43] W3C. Resource Description Framework (RDF) Schema Specification 1.0, 2000. W3C Candidate Recommendation 27 March 2000. (http://www.w3.org/TR/2000/CR-rdf-schema-20000327).
- [44] Advanced Distributed Learning. SCORM Overview. (http://www.adlnet.org/index.cfm?fuseaction=scormabt).
- [45] Advanced Distributed Learning (ADL). (http://www.adlnet.org/)
- [46] Outline Processor Markup Language (OPML). (http://www.opml.org/)
- [47] J. Allen. Making a Semantic Web. (http://www.xml.com/pub/a/2000/07/17/syndication/rss.html)
- [48] R. E. Kent. Conceptual Knowledge Markup Language: The Central Core. The Ontology Consortium. (http://sern.ucalgary.ca/ksi/kaw/kaw99/papers/Kent1/CKML.pdf)
- [49] R. Wille, 1982. Restructuring lattice theory: an approach based on hierarchies of concepts, in, I. Rival (ed.), Ordered Sets, Reidel.
- [50] B. Ganter and R. Wille, 1989. Conceptual scaling, in, F. Roberts (ed.), Applications of Combinatorics and Graph Theory in the biological and Social Sciences, Springer-Verlag.
- [51] B. Barwise and J. Seligman, 1997. Information Flow: The Logic of Distributed Systems, Cambridge University Press.
- [52] M. Biezunski, M. Bryan, S. R. Newcomb (eds), 1999. Topic Maps: Information Technology - Document Description and Markup Languages, ISO/IEC 13250:2000. (http://www.y12.doe.gov/sgml/sc34/document/0129.pdf)
- [53] Edutella. (http://edutella.jxta.org/)
- [54] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, T. Risch, 2002. Edutella: A P2P Networking Infrastructure Based on RDF. WWW2002, Hawaii. (http://edutella.jxta.org/reports/edutella-whitepaper/)
- [55] Stanford University. Protégé 2000. (http://protege.stanford.edu)
- [56] Stanford University. Planning a Protégé 2000 Project. Protégé 2000 User Guide. (http://protege.stanford.edu/doc/users_guide/)
- [57] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative Ontology Development for the Semantic Web. (http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/ 2002_iswc_ontoedit.pdf)
- [58] S. Bechhofer, I. Horrocks, C. Goble and R. Stevens. OilEd: a Reason-able Ontology Editor for Editor for the Semantic Web. (http://www.cs.man.ac.uk/~horrocks/Publications/ download/2001/oiled-dl.pdf)
- [59] J. C. Apirez, O. Corcho, M. Fernandez-Lopez and A. Gomez-Perez, 2001. WebODE: a Scalable Workbench for Ontological Engineering. K-CAP'01, Canada. (http://mkbeem.elibel.tm.fr/paper/KCAP2001-35.pdf)
- [60] Stanford University Knowledge Systems Laboratory. Ontolingua. (http://www.ksl.stanford.edu/software/ontolingua/)
- [61] Information Sciences Institute. SENSUS Ontosaurus. (http://mozart.isi.edu:8003/sensus2/)
- [62] U. Visser, H. Stuckenschmidt, H. Wache, T. Vogele, 2000. Enabling Technologies for Interoperability. NEC Research Index. (http://citeseer.nj.nec.com/459286.html)
- [63] Stanford University. Scalable Knowledge Composition (SKC). (http://www-db.stanford.edu/SKC/)
- [64] W3C. Requirements for a Web Ontology Language. W3C Working Draft 08 July 2002. (http://www.w3.org/TR/webont-req/)

- [65] MIT. Open Knowledge Initiative. (http://web.mit.edu/oki/)
- [66] m-Learning Project. (http://www.m-learning.org/)

1.2. INVOLVED ORGANIZATIONS

The e-learning standard definition landscape is quite a complex one. Currently, half a dozen organizations are working to develop industry standards in this field. They include the following:

- IMS, a consortium of members that include major software developers and vendors, training and education representatives, and government agencies.
- ADL, an initiative of the U.S. Department of Defence which aims to ensure the interoperability of future e-learning technologies purchased by the U.S. government.
- LTSC, a branch of the Institute of Electrical and Electronic Engineers (IEEE) with a long-standing reputation as an accrediting body for technology standards.
- AICC (the Aviation Industry CBT Committee), an association of technology developers that spans well beyond the aviation industry.

All these groups are increasingly working together in order to harmonize their efforts towards a comprehensive set of standards for distance learning.

2. STANDARDS FOR METADATA AND TEST REPRESENTATION

This section introduces the various standards regarding metadata available together with those used in the Diogene Project.

2.1 IMS METADATA

In 1997 the IMS Project established an effort to develop open market-based standards for online learning, including specifications for learning content meta-data. Also in 1997, groups within the IEEE P.1484 study group (now the IEEE Learning Technology Standards Committee - LTSC) began similar efforts. IMS began collaborating with the ARIADNE Project, a European project with an active meta-data remit.

In 1998, IMS and ARIADNE submitted a joint proposal and specification to the IEEE LTSC, which formed the basis of the IEEE LTSC Working Group 12 work on a draft standard for Learning Object Metadata (LOM). The standard developed by this working group is a multi-part standard: part one being a conceptual data schema (which corresponds to the IMS Learning Resource Meta-data Information Model), parts two, three, and four being bindings of this schema in ISO/IEC11404, XML and RDF respectively.

As a result of significant worldwide interest, the IMS Project was incorporated as the IMS Global Learning Consortium in 1999, publicizing the IEEE work through its stakeholder community in the US, UK, Europe, Australia, and Singapore. This evolving stakeholder community has contributed feedback into the ongoing specification development process. In August 1999, IMS released its Learning Resource Meta-data Specification v1.0 to the public, with minor revisions being released periodically up to v1.2.2 in November 2001. Each of these specification releases was based on updates of the IEEE LOM conceptual data schema with an accompanying XML Binding and Best Practice and Implementation Guide produced by IMS.

IMS Meta-data v1.2.2 specification was based on working draft 6.1 of the IEEE LOM standard. The IEEE standard finally published in June 2002 was based on working draft 6.4 of the LOM. With the release of this specification, IMS Meta-data v1.3, the Information Model of the IMS specification has been realigned with the published IEEE standard. The XML Binding of the IMS Meta-data specification has also been realigned, given changes in the IEEE information model and to allow stricter checking of meta-data instances using an XML schema. The intention is that the IMS Meta-data Specification v1.3 will align to the IEEE 1484.12.1 and 1484.12.3 standards.

Concluding, we won't introduce IMS Metadata here, pointing for details the interested reader to the related material, available at: http://www.imsglobal.org/metadata/index.cfm.

2.2 IEEE LEARNING OBJECT METADATA

LOM stands for Learning Object Metadata and is a standard defined by the Learning Tecnology Standardization Committee (LTSC) of IEEE. As in the IMS Metadata Specification, the LOM standard has a hierarchical structure. Each of its elements can be mandatory, optional or conditional (in this case, the attribute values depend on the presence or absence of a value for another optionally defined attribute). Some attributes can have, as a value, a list of terms rather than a single term. This list can be ordered or unordered. Finally, the shared vocabularies of terms of the metadata values can be defined as: restricted (only the values of the vocabulary are admissible values for that attribute) or open (the vocabulary groups a suggested list of terms frequently used by applications but also other terms are admissible values for that attribute).
The main elements of LOM schema actually are 47 (against the only 15 of Dublin Core) and are grouped in 9 categories as shown in the following table .

Category	Description	Elements
General	Groups all context independent features together with the semantics descriptors of the resource.	Identifier, Title, CatalogEntry, Language, Description, Keywords, Coverage, Structure, Aggregation Level.
Life Cycle	Groups the features related to the life cycle of the resource.	Version, Status, Contribute.
Meta Metadata	Groups the features of the description itself rather than the resource which is described.	Identifier, CatalogEntry, Contribute, Metadata Scheme, Language.
Technical	Groups the technical features of the resource.	Format, Size, Location, Requirements, Installation Remarks, Other Platform Requirements, Duration.
Educational	Groups the didactic and pedagogical features of the resource.	Interactivity Type, Learning Resource Type, Interactivity Level, Semantic Density, Intended End User Role, Context, Typical Age Range, Difficulty, Typical Learning Time, Description, Language
Rights Management	Groups the resource features depending on the type of use we plan for it.	Cost, Copyright and Other Restrictions, Description.
Relation	Groups the resource features which relate it to other resources.	Kind, Resource, Identifier, Description, CatalogEntry.
Annotation	Allows comments to be made about the resource's educational content.	Person, Date, Description.
Classification	Describes a particular classification system in which the resource is defined (such as taxonomies, conceptual graphs, and so on).	Purpose, TaxonPath, Description, Keywords.

Documents on this standard are available at http://ltsc.ieee.org/wg12/

2.3 DUBLIN CORE

The Dublin Core metadata element set is a standard for cross-domain information resource description.

The Dublin Core Metadata Initiative (DCMI) is an organization born with the aim of defining a metadata

standard composed of a small set of elements (hence the name "core") capable of describing the greatest number of Web resource typologies. Indeed, while several organizations dedicate their efforts to fix standards for specific application domains, Dublin Core is aimed at defining a minimal set of terms and categories usable "horizontally" by all domains to allow interoperability among applications coming from different areas.

The following are the main objectives of Dublin Core:

- Development and management simplicity. The set of elements is small and easy so that non-experts can build the records she/he needs easily and cheaply.
- Shared semantics. Information retrieval is sometimes difficult due to the differences in terminology and descriptive techniques among knowledge domains. Hence, the DCMI has established a shared set of elements, the semantics of which are universally clear, in order to augment the accessibility to all kinds of resources independently from their original field.
- International definition. Initially the DCMI developed the set of elements of Dublin Core in English.

Successively, the standard was translated into more than 20 different languages.
The Dublin Core standard consists of exactly 15 different elements, each of which is optional and can be instantiated multiple times. Each element is an attribute of the metadata (see Section 4.2.1). There is no constraint on the order to use when we instantiate the attributes, and this rule is valid also for multiple instantiated attributes. The admissible set of values for some of the attributes of Dublin Core belongs to a shared vocabulary containing an accurately selected set of terms. A shared vocabulary of terms for attribute values is important to improve search results, avoiding ambiguity errors due to aliases describing the same concept.
The elements (attributes) of Dublin Core (as specified in Dublin Core Version 1.1), are listed in Table 4. All data types must be strings containing English text.

Identifier	Definition	Comment
Title	A name given to the resource.	**Title** is the formal name of the resource.
Creator	The entity which is the principal creator of the resource.	A **Creator** can be a person, an organization, a service, …
Subject	The subject of the resource.	**Subject** can be expressed by means of key words or key sentences or classification codes. It is suggested to choose its value from a formal defined vocabulary.
Description	An explanation concerning the resource content.	For instance, a **Description** can be: an analytical summary, an index, a link to a graphic representation, text about the content, …
Publisher	The entity which is the actual publisher of the resource.	A **Publisher** can be a person, an organization, a service, …
Contributor	The entity which is a minor creator of the resource.	A **Contributor** can be a person, an organization, a service, …
Date	A date associated with an important event of the life-cycle of the resource.	Normally, **Date** is associated with the creation or the availability of the resource. For the value of this attribute is suggested to use the following format: YYYY-MM-DD[1]. If one wants to use another format, this must be univocally identified.
Type	The nature or the kind of the resource content.	**Type** includes terms describing general categories. It is suggested to choose this from a formal vocabulary: for example, the Dublin Core Types list (DCT1).
Format	The physical representation format of the resource.	Normally, **Format** includes the resource support type (software, hardware, …) or its size.
Identifier	The resource identifier in a given context.	It is suggested to identify the resource by means of a sequence of characters belonging to a formal identification system (URL, DOI, ISBN).

[1] See standard ISO 8601, defined in http://www.w3.org/TR/NOTE-datetime.

Source	A link to a resource from which the present resource is derived.	The present resource can be derived from **Source** in a complete or partial way. It is suggested to identify the resource by means of a sequence of characters belonging to a formal identification system.
Language	The language in which the resource content is described.	It is suggested to represent the values of the attribute **Language** following the format defined in RFC 1766[2].
Relation	A link to a related resource.	It is suggested to identify the resource by means of a sequence of characters belonging to a formal identification system.
Coverage	The resource scope.	Normally, **Coverage** includes a spatial location (the name of a place or its geographic coordinates) or a temporal range. It is suggested to take the value of the attribute from a formal vocabulary (for example, TGN).
Rights	Copy-right information.	Normally, **Rights** contains information about the resource management rights. If **Rights** is not instantiated, no hypothesis is made about resource copy-rights.

More details are available at: http://dublincore.org/documents/dces/

2.4 IMS QUESTION AND TEST INTEROPERABILITY (QTI) E QTI LITE

The IMS Question & Test Interoperability Specification provides proposed standard XML language for describing questions and tests. The specification has been produced to allow the interoperability of content within assessment systems. This will be useful for publishers, certification authorities, teachers, trainers, publishers and creators of assessments, and the software vendors whose tools they use. Authoring tools, and publishers, may publish XML and this data can be imported into other authoring tools and delivery systems.

More specifically, the IMS Question & Test Interoperability (QTI) specification describes a basic structure for the representation of question (item) and test (assessment) data and their corresponding results reports. Therefore, the specification enables the exchange of this item, assessment and results data between Learning Management Systems, as well as content authors and, content libraries and collections. The IMS QTI specification is defined in XML to promote the widest possible adoption. XML is a powerful, flexible, industry standard markup language used to encode data models for Internet-enabled and distributed applications. The IMS QTI specification is extensible and customisable to permit immediate adoption, even in specialized or proprietary systems. Leading suppliers and consumers of learning products, services and content contributed time and expertise to produce this final specification. The IMS QTI specification, like all IMS specifications, does not limit product designs by specifying user interfaces, pedagogical paradigms, or establishing technology or policies that constrain innovation, interoperability, or reuse.

AN OVERVIEW OF QTI

Despite its name, the IMS QTI specification details more than how to tag questions, tests and results. The standard Question types e.g. multiple choice, fill in the blank, or true/false choice, etc. can be constructed using a core set of presentation and response structures, and results of questions can be collected and scored by using a variety of methods. To represent these options, the IMS QTI specification defines the 'Item'. Items contain all the necessary data elements required to compose, render, score and provide feedback from questions. Therefore, the key difference between a 'Question' and 'Item' is that an 'Item' contains the 'Question', layout rendering information, the associated response processing information, and the corresponding hints, solutions and feedback.

Similarly, the 'test' is an instance of an Assessment. Assessments are assembled from Items that are contained within a 'Section' to resemble a traditional test. Additionally, Assessments might be assembled from blocks of Items that are logically related. These groups are also defined as 'Sections' and so Assessments are composed of

[2] RFC 1766 is available at: http://www.ietf.org/rfc/rfc1766.txt.

one or more Sections which themselves are composed of Items, or more Sections. Collectively, these three data objects are referred to as the ASI (Assessment, Section, Item) structures. These evaluation objects can be bundled together to create an object bank. This object bank can then be externally referenced and used as a single evaluation object. To avoid limitations associated with words like user, student, or learner the IMS QTI working group adopted the term 'participant' to refer to the person interacting with an assessment. Thus, the key definitions are:

- Item - A combination of interrogatory, rendering, and scoring information;
- Section - A collection of zero or more items and/or other Sections;
- Assessment - A collection of one or more Sections;
- Object Bank - A group of Items and/or Sections that have been bundled e.g. to create an Item-bank;
- Participant - The user interacting with an assessment.

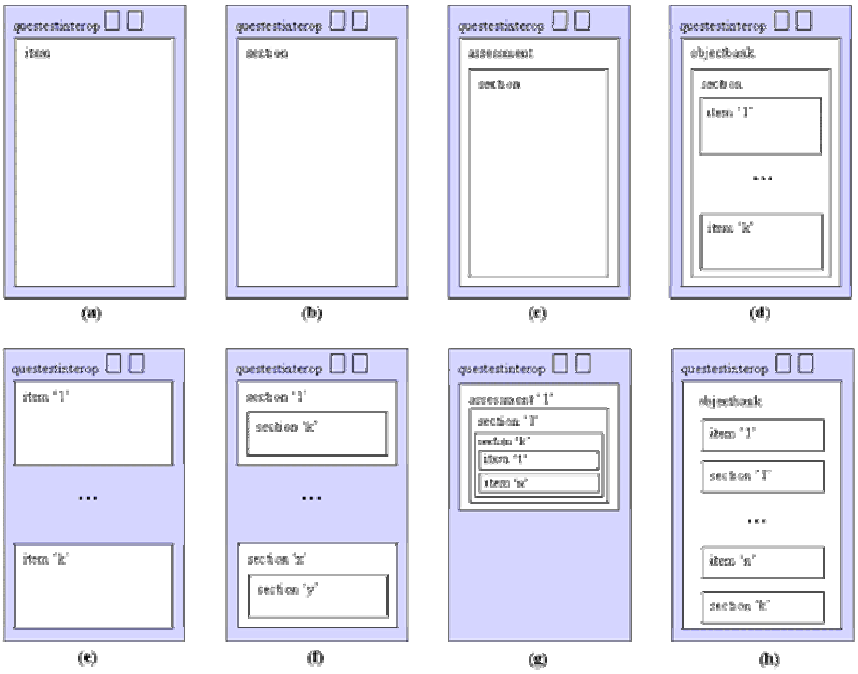
When constructing the results report for an Assessment, a Section or an Item, a similar structure is used. A results report can contain either a summary of the results themselves and/or the detailed set of results with respect to the assessment, section(s) and item(s). Each results report is contained within its own package that also describes the context for the evaluation e.g. participant identifier, etc.
The QTI ASI Information Model document is comprised of several sections. The first section contains use cases in which the underlying usage, processing control, and core data structures of the QTI ASI specification are described. It also details the taxonomy of responses, as well as their relationship to questions type and the larger group of 'items'. The basic information model itself is outlined in conceptual terms by using a tabular layout of the Assessment, Section, and Item objects in terms of their elements, sub-elements and attributes. The Item, Section, and Assessment meta-data, which are used to catalogue these objects, are also described. In addition, the document contains a conformance statement to be used by vendors who plan to implement the specification; we have adopted a descriptive approach to conformance thereby enabling vendors to implement subsets of the full specification.

The core ASI data structures that can be exchanged using IMS QTI are Items, Sections, Assessment and Object bank.
One of more Items can be contained within a QTI-XML instance. The Item is the smallest independent unit that can be exchanged using IMS QTI. An Item cannot be composed of other Items. An Item is more than a 'Question' in that it contains the 'Question', the presentation/rendering instructions, the response processing to be applied to the participant's response(s), the feedback that may be presented (including hints and solutions) and the meta-data describing the Item;

One or more Sections can be contained within a QTI-XML instance. A Section can contain any mixture of Sections and/or Items. A Section is used to support two different needs:-
To represent different grouping constructs as defined by the appropriate educational paradigm e.g. a Section could be equivalent to a subject topic
To constrain the extent of the sequencing instructions and to control the ways in which the different possible sequences may be constructed;

Only one Assessment can be contained within a QTI-XML instance. It is not possible to define relationships between the Assessments. Each Assessment must contain at least one Section, thus it is not possible to have Items housed directly within an Assessment. The Assessment contains all of the necessary instructions to enable variable sequencing of the Items and the corresponding aggregated scoring for all of the Items to produce the final score;

The core ASI data structures that can be exchanged using IMS QTI are shown in the following picture.



Having described the main concepts of QTI we now introduce a slimmed-down reduced of such standard, known as QTILite.

QTI LITE

As we know from the previous section the Question & Test Interoperability (QTI) specification describes a basic structure for the representation of question (item) and test (assessment) data and their corresponding results reports. Therefore, the specification enables the exchange of this test, assessment and results data between Learning Management Systems, as well as content authors and, content libraries and collections. As well as the full QTI, also QTILite specification is defined in XML to promote the widest possible adoption. The QTILite specification is extensible and customizable to permit immediate adoption, even in specialized or proprietary systems. Leading suppliers and consumers of learning products, services and content contributed time and expertise to produce this final specification.

For ease of reading, instead of introducing from the ground up the QTILite specification we proceed with introducing the main differences with the full QTI specification.

MAIN DIFFERENCES BETWEEN QTI AND QTI LITE

The key differences between QTI Lite and the full specification are the following:

The only question-types to be supported within QTI Lite are:

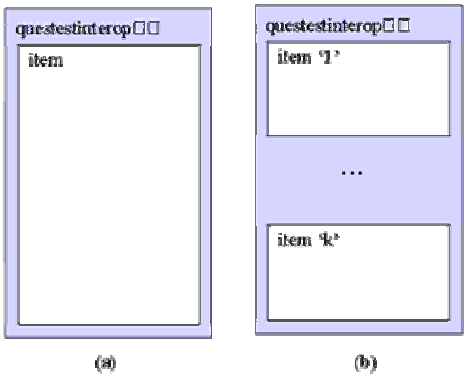
- Yes/No.
- True/false.
- Values from multiple choice lists (i.e. one choice from many).

Simple response processing to provide for a single right answer and using the default mechanisms;
Furthermore, in QTILite there is no support for:

- Hints and solutions
- Meta-data
- Comments
- Extensions

- Options that are not enumerated
- Limited media types and limited text types
- All time-based mechanisms.

The following picture shows the principal QTI Lite interchange data objects.



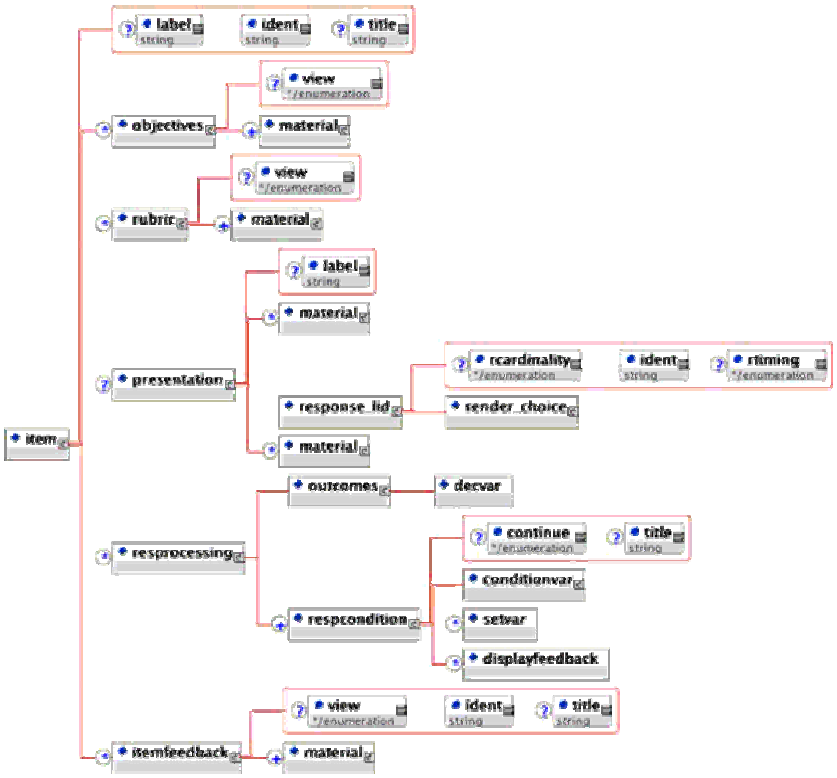
The QTI Lite specification is concerned with the exchange of Items between Assessment systems. The internal representation may conform to the QTI Lite but the adoption in this way is beyond the scope of the specification. Nine distinct 'views' have been identified for each of the core participants:

- Administering,
- Administrator,
- Assessor,
- Author,
- Candidate,
- Invigilator/Proctor,
- Psychometrician,
- Scorer
- Tutor.

Different types of information may be made available to each of these actors.

The core data structures that can be exchanged using the QTILite Specification are shown in the previous Figure. QTILite supports the exchange of Items only, while the full specification also supports the exchange of Assessments and Sections.

As a matter of comparison, the following diagram shows the tree structure of the QTI LIte XML schema binding.



More details on QTI Lite and some related material can be found at:
http://www.imsglobal.org/question/index.cfm.

2.5 Comparison and Choice

From the previous sections the choice about the suitable standards to be adopted in the Diogene project, as regards metadata representation is reduced to the only IMS metadata format (or its equivalent IEEE LOM). In fact, the other viable alternative, Dublin Core, suffers from several shortcomings that make it a sub-optimal choice when compared with IMS metadata. The major shortcomings being:

- While IMS Metadata and IEEE LOM have been expressively designed for e-learning, Dublin Core has being designed for a much larger and general kind of electronic content, and as this it suffers from an over generalization that makes its use in specialized context (such as e-learning) more difficult.
- Despite being around for so many years, the current diffusion of the Dublin Core standard has been visibly slowing down while other newer standards (like IMS Metadata) are catching up quickly.
- Dublin Core has been developed previously than other more sophisticated approaches and it lacks some state-of-the-art features like for example interchangeable XML support.

From all these considerations it is apparent that IMS metadata or IEEE LOM should be preferred over Dublin Core, as regards the adoption within the Diogene project.

As regards the question and interoperability support, the choice is between QTI and QTI Lite.
As a first note, QTI Lite provides all expressiveness power needed by the Diogene Project question representation requirements. Furthermore, the Diogene Project doesn't need any of the features included in the full QTI specification. Implementing the QTILite specification is technically easier and the exported format files result lighter than full QTI.
Given these considerations, as regards the adoption in the Diogene project, QTI Lite should be preferred over full QTI.

3. STANDARDS FOR COURSE REPRESENTATION

This section discusses the main available standards for course material representation (like for instance didactic resource files packaging).

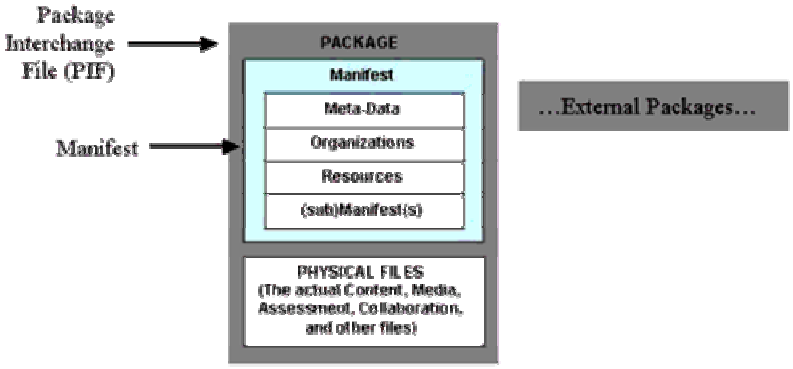
3.1 IMS CONTENT PACKAGING

The IMS Content Packaging Specification provides the functionality to describe and package learning materials, such as an individual course or a collection of courses, into interoperable, distributable packages. Content Packaging addresses the description, structure, and location of online learning materials and the definition of some particular content types.
The Content Packaging Specification is aimed primarily at content producers, learning management system vendors, computing platform vendors, and learning service providers. Learning materials described and packaged using the IMS Content Packaging XML format should be interoperable with any tool that supports the Specification. Content creators can develop and distribute material knowing that it can be delivered on any compliant system, thereby protecting their investment in rich content development.

This specification is included in the SCORM standard and will be discussed in detail in the related section.

IMS CONTENT PACKAGING CONCEPTUAL MODEL

The following Figure is a conceptual diagram that illustrates the components of the IMS Content Packaging Information Model. As indicated in the IMS Content Packaging Best Practice Guide, this is part of the larger IMS Content Framework, which forms the basis for this and future specifications.



The IMS Package depicted in the previous Figure consists of two major elements: a special XML file describing the content organization and resources in a Package, and the physical files being described by the XML. The special XML file is called the IMS Manifest file, because course content and organization is described in the context of 'manifests'. Once a Package has been incorporated into a single file for transportation, it is called a Package Interchange File. The relationship of these parts to the content container is described below:

- Package Interchange File - a single file, (e.g., '.zip', '.jar', '.cab') which includes a top-level manifest file named "imsmanifest.xml" and all other physical files as identified by the Manifest. A Package Interchange File is a concise Web delivery format, a means of transporting related, structured information. PKZip v2.04g (.zip) is recommended as the default Package Interchange File format. Any ZIP file format MUST conform to RFC1951.
- Package - a logical directory, which includes a specially named XML file, any XML control documents it references (such as a DTD or XSD file), and contains the actual physical resources. The physical resources may be organized in sub-directories.
- Top-level Manifest - a mandatory XML element describing the Package itself. It may also contain optional (sub)Manifests. Each instance of a manifest contains the following sections:
    - Meta-data section - an XML element describing a manifest as a whole;
    - Organizations section - an XML element describing zero, one, or multiple organizations of the content within a manifest;

- Resources section - an XML element containing references to all of the actual resources and media elements needed for a manifest, including meta-data describing the resources, and references to any external files;
- (sub)Manifest - one or more optional, logically nested manifests;
- Physical Files - these are the actual media elements, text files, graphics, and other resources as described by the manifest(s). The physical resources may be organized in sub-directories.
- Package - A Package represents a unit of usable (and reusable) content. This may be part of a course that has instructional relevance outside of a course organization and can be delivered independently, as an entire course or as a collection of courses. Once a Package arrives at its destination to a run time service, such as an LMS vendor, the Package must allow itself to be aggregated or disaggregated into other Packages. A Package must be able to stand alone; that is, it must contain all the information needed to use the contents for learning when it has been unpacked.
  Packages are not required to be incorporated into a Package Interchange File. A Package may also be distributed on a CD-ROM or other removable media without being compressed into a single file. An IMS Manifest file and any other supporting XML files required by it (DTD, XSD) must be at the root of the distribution medium.

A manifest is a description in XML of the resources comprising meaningful instruction. A manifest may also contain zero or more static ways of organizing the instructional resources for presentation. A manifest can describe part of a course that can stand by itself outside of the context of a course (an instructional object), an entire course, or a collection of courses. The decision is given to content developers to describe their content in the way they want it to be considered for aggregation or disaggregation. The general rule is that a Package always contains a single top-level manifest that may contain one or more (sub)Manifests. The top-level manifest always describes the Package. Any nested (sub)Manifests describe the content at the level to which the (sub)Manifest is scoped, such as a course, instructional object, or other.

For example, if all content comprising a course is so tightly coupled that no part of it may be presented out of the course context, a content developer would want to use a single manifest to describe that course's resources and organization. However, content developers who create "instructional objects" that could be recombined with other instructional objects to create different course presentations would want to describe each instructional object in its own manifest, then aggregate those manifests into a higher-level manifest containing a course organization. Finally, a content developer who wants to move multiple courses in a single Package (a curriculum), would use a top-level manifest to contain each course-level manifest and any instructional object manifests that each course might contain.

The resources described in the manifest are physical assets such as Web pages, media files, text files, assessment objects or other pieces of data in file form. Resources may also include assets that are outside the Package but available through a URL, or collections of resources described by (sub)Manifests. The combination of resources is generally categorized as content. Each resource may be described in a <resource> element within a manifest's XML. This element includes a list of all the assets required to use the resource. The files included in the Package are listed as <file> elements within such <resource> elements.

Related documentation is available at: http://www.imsglobal.org/content/packaging/index.cfm

3.2 SCORM CONTENT AGGREGATION MODEL

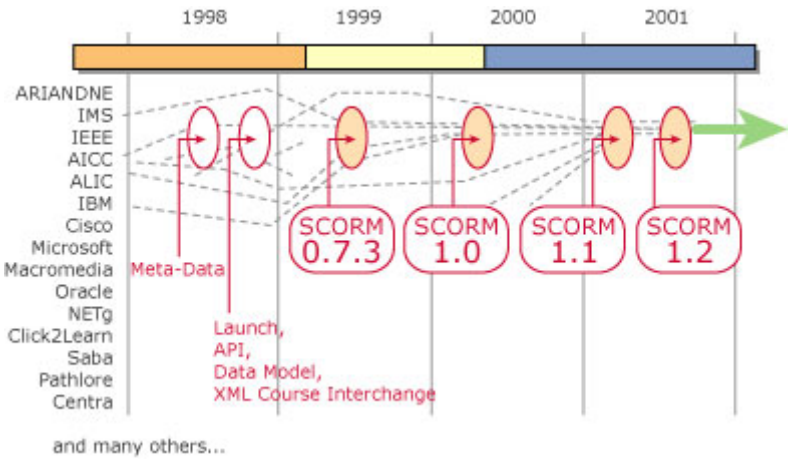
The SCORM (Sharable Content Object Reference Model) [44] is a set of specifications for developing, packaging and delivering education and training materials whenever and wherever they are needed. SCORM is a product of the U.S. Government's initiative in Advanced Distributed Learning (ADL) [45], which aims to provide access to high-quality materials that are easily tailored to individual learner needs.
The SCORM applies current technology developments – from groups such as the IMS Global Learning Consortium, the Aviation Industry Computer-Based Training Committee, the Alliance of Remote Instructional Authoring and Distribution Networks for Europe (ARIADNE) and the Institute of Electrical and Electronics

Engineers (IEEE) Learning Technology Standards Committee (LTSC) – to a specific content model to produce recommendations for consistent implementations by the vendor community.

Although choice and competition in the marketplace are generally a good thing, the rapid growth of LMS vendors has created a significant problem for content authors. Without a common specification for packaging online courses, LMS vendors organize their content databases in any fashion they choose. As a result, every vendor is using a different format for packaging their courses. Although they are all delivered by http, or some other standard protocol, if an author tries to move a course from one LMS to another, they find that this task is very time-consuming. In many cases, moving from one LMS vendor to another requires complete reconstruction of the course materials.

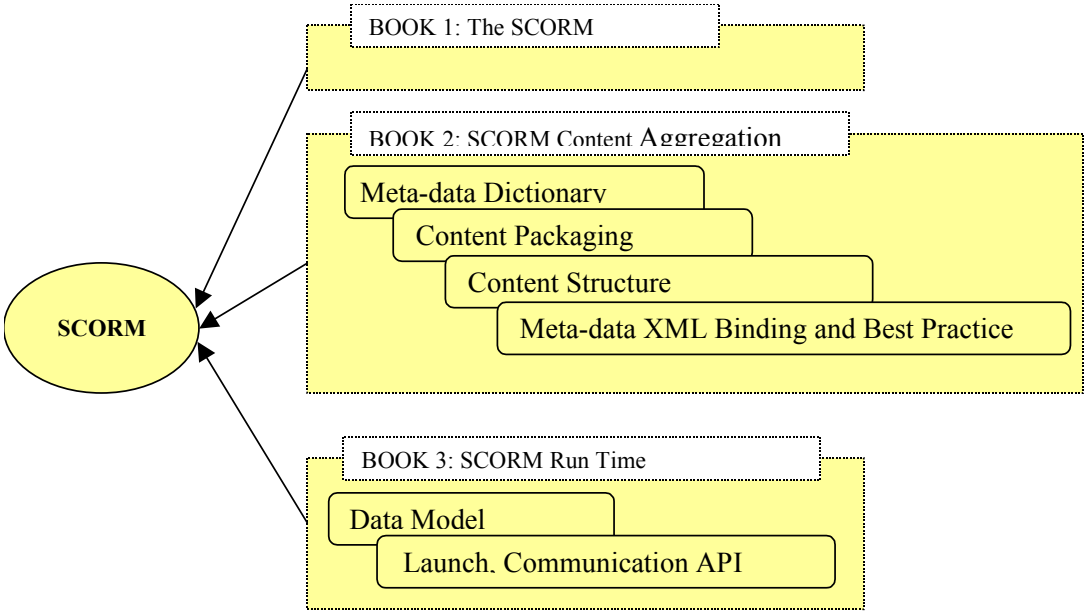
ADL is evolving a set of specifications for packaging online courses that will not only make it easier to transport a course from one LMS to another, it will also achieve other desirable goals as well. A course packaged following the SCORM specifications can be transported from one LMS to another with minimum modifications. SCORM-compliant courses leverage course development investments by ensuring that compliant courses are:

- Accessible: the ability to locate and access instructional components from one remote location and deliver them to many other locations.
- Interoperable: the ability to take instructional components developed in one location with one set of tools or platform and use them in another location with a different set of tools or platforms.
- Durable: the ability to withstand technology changes without redesign, reconfiguration or recoding.
- Reusable: the flexibility to incorporate instructional components in multiple applications and contexts.



Some of the main organizations involved in SCORM**.**

As shown in the following figure, all of the specifications and guidelines contained or referenced can be viewed as separate “books” gathered together into a growing library.



SCORM Organization.

IEEE ECMA SCRIPT APPLICATION PROGRAMMING INTERFACE FOR CONTENT TO RUNTIME SERVICES COMMUNICATION (1484.11.2) (PART OF SCORM 1.3)

The ECMAScript API for content-to-runtime-services communication defined in the AICC "CMI Guidelines for Interoperability" version 3.4, has broad applicability to systems used for learning management. The purpose of this Standard is to build consensus around, resolve ambiguities, and correct defects in this ECMAScript API for exchanging data between learning-related content and a runtime service used to support learning management. See the section “STANDARDS FOR LMS INTEROPERABILITY” in this same document for more details.

IMS CONTENT PACKAGING SPECIFICATION VERSION 1.1.3 (PART OF SCORM 1.3)

This specification (updated on July 1, 2003) corrects an editorial error that inadvertently left the **`<variation>`** element set in Table 4.1 of the IMS CP Information Model; this element is now removed from the specification.

The related documentation is available at: http://www.imsglobal.org/content/packaging/index.cfm

IEEE DATA MODEL FOR CONTENT OBJECT COMMUNICATION 1484.11.1 (PART OF SCORM 1.3)

This Standard describes a data model to support the interchange of agreed upon data elements and their values between a learning-related content object and a runtime service (RTS) used to support learning management.

This Standard does not specify the means of communication between a content object and an RTS nor how any component of a learning environment shall behave in response to receiving data in the form specified. This Standard is based on a related data model defined in the "Computer Managed Instruction (CMI) Guidelines For Interoperability," version 3.4, defined by the Aviation Industry CBT Committee (AICC). To balance the need to support existing implementations with the need to make technical corrections and support emerging practice, this Standard selectively includes those data elements from the CMI specification that are commonly implemented; renames some data elements taken from the CMI specification to clarify their intended meaning; modifies the data types of data elements taken from the CMI specification to reflect ISO standard data types and internationalization requirements; removes some organizational structures used in the CMI specification to group data elements that are specific to the AICC community of practice and not generally applicable; and introduces some data elements not present in the CMI specification to correct known technical defects in data elements taken from that specification.

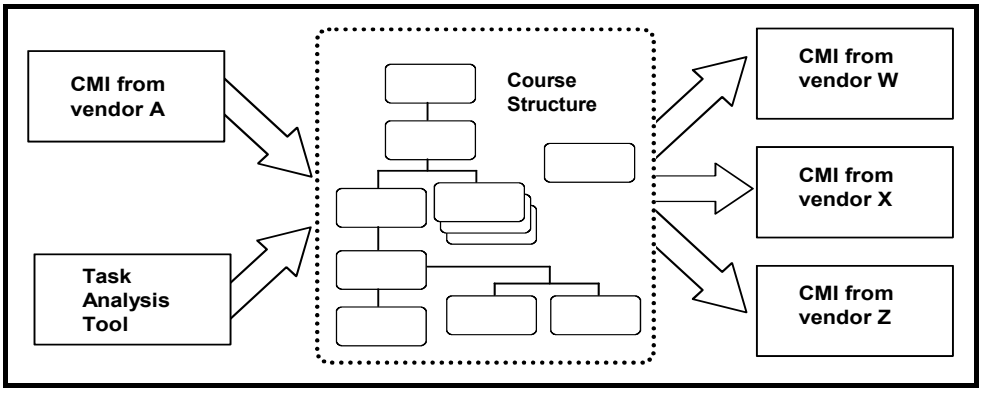
There is widespread acknowledgement that the data model for content object communication defined in the AICC "Computer Managed Instruction (CMI) Guidelines for Interoperability", version 3.4, has broad applicability to systems used for learning management. The purpose of this standard is to build consensus around, resolve ambiguities, and correct defects in this data model for the data exchanged between learning-related content and a runtime service used to support learning management.

More details are available at: http://ltsc.ieee.org/wg11/par1484-11-1.html

3.3 AICC CONTENT STRUCTURE MODEL

A course may be as simple as a few lessons to be viewed sequentially, or it may be as complex as hundreds of lessons, some of which are prerequisites to others and some of which may be experienced in any order. Basically, courses have two components: instructional elements and structure.

The instructional elements are all the lessons, tests, and other assignable units (AUs) in the course. Frequently, the content elements also include all of the objectives to be mastered in the course. In defining a structure, the developer frequently groups lessons for assignment. In other cases the designer defines complex lesson hierarchies.



Course Structure Interchange

The structure determines the order in which these are to be experienced by each student. The part of the CMI system that sequences the course content, is referred to as the router.

There are at least two circumstances in which guidelines for moving courses from one environment to another are useful. The first assumes a course is complete and is being transferred from a vendor or manufacturer to an airline -- moving from one CMI system to another. The second assumes a course is being designed in a tool other than a CMI system -- moving course design into CMI. Having a standardized mechanism for describing course content and structure, enables CMI systems to "ingest" a new course with minimal manual effort.

The AICC has identified seven files (some optional) that can be used to describe a course's content and structure. Additionally, the AICC guidelines define five levels of complexity in describing the course structure. Increasing the level of complexity should result in:

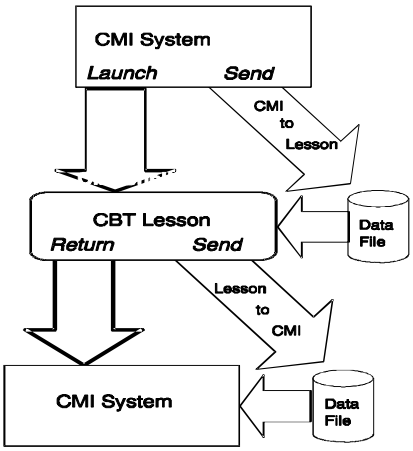
- Less effort to review and modify the CMI system after importing the data.
- More complete description of the designer's intended usage of the course material.

The level of complexity determines the number of files required and the amount of information required in each file.

There are two aspects of the AICC approach to enabling interoperability of CMI systems with different CBT systems.

- Lesson launch: The CMI should have a standard approach to CBT lesson initiation, and

- Communication: The CMI should have a standard approach to providing information to the CBT lessons, and receiving information from the CBT lessons.



CMI Management of CBT

The seven file types are described below.

Course Description File	Information about the course as a whole including a textual description of the course, and general makeup of the course -- the number and type of elements.
Assignable Unit Table	Information about the assignable units (AUs) in the course. Each AU has its own record (or row in the table). The information includes the name of the AU, its ID, and the mastery score for that AU.
Descriptor Table	A complete list of every course element in the course including: AUs, Blocks, Objectives, Complex Objectives. It is used as the basic cross reference file showing the correspondence of system-generated IDs with user-defined IDs for every element.
Course Structure Table	The basic data on the structure of the course including all of the AUs and blocks in the course, showing how they are organized. Finally, it implies the order in which these should be taken.
Objectives Relationships File	Objectives have complex and variable relationships to other elements of a course. This file defines all of these relationships. This file is optional, depending on the level of the course description.
Prerequisite Listing	Sometimes it may be desirable to prevent a student from entering a lesson until he has met certain prerequisites. This file allows that sort of constraint to be placed on each block or AU in a course. There are three levels of complexity that may be used in describing prerequisites: • a single prerequisite AU or block to be defined for each element in the course • prerequisites to be defined in the form of a logic statement (with "ands" and "ors") • the definition of prerequisites for each mode (Review, Browse, Normal) for the lesson
Completion Requirements	While lesson and objective status is determined within the lesson by the logic designed into it, this is not true of blocks. Blocks are created specifically to describe a course structure. Similarly Complex Objectives are defined in terms of other structure elements. Therefore, block and complex objective status must be determined by the CMI system. The Completion Requirements file is designed to allow the explicit specification of when a block or objective is complete when it does not conform to the defaults for completion. It is essentially an exception file.

Also lesson evaluation data is described in these specifications.
Lesson evaluation data includes information that a CBT lesson or test generates on the behavior of a student. It may include such items as a student's responses, latency, and path through a lesson. Standardizing the format of the student records permits multiple tools to use the information. Lesson evaluation data is contained in several files. File names for this data are passed to the lesson from the CMI system. If the file already exists, the lesson appends the data. If the file does not exist, the file is created and the data deposited.

3.4 COMPARISON AND CHOICE

First of all, it is important to note that SCORM integrates AICC's content structure into its model. In addition SCORM can be though of an aggregation of several standards. It tries to combine the best of what is available in Literature to come with a global, unique solution.
Furthermore SCORM is what is called an "application profile" in that it is based on several other standards and specifications, but specifies a "profile" of how to combine and use those in a coherent way to provide interoperability.

To recap, SCORM 2004 is composed by the following:

- Content structure and packaging: Content Packaging spec from IMS Global Learning Consortium
- Sequencing: Simple Sequencing spec from IMS
- Metadata: IEEE Metadata standard (XML binding based on an IEEE draft standard)
- SCO to LMS communication data model: IEEE draft standard
- SCO to LMS API: IEEE standard

And in this aspect is a much more complete and homogeneous standard for content structure than any other proposal currently available.

The main differences between AICC and SCORM can be stated as follows. From an overall perspective, the AICC is governed by representatives of the aviation industry and focused on the needs of that industry, which has a great deal of legacy content, and is traditionally English speaking only.
On the other side, SCORM is focused on the needs of corporate, vocational, government and military training as well as education in general, and was developed and is maintained under the umbrella of the Advanced Distributed Learning initiative (ADL). SCORM is more complete (it has a robust manifest-based content packaging model and metadata), it was from the start designed to function in international settings with support other languages, and content development for SCORM can be simpler because the burden of bi-directional communication across the internet is left to the LMS and does not have to be built into every SCO.

The SCORM API model also allows content to play offline without requiring a HTTP server on the offline computer (that also works using the AICC JavaScript API, which is basically the same as the SCORM API).
Most LMS vendors have been focusing on SCORM for any new implementation work, although many still have support for at least some versions of the AICC spec, and at least one is actively promoting the AICC specification.
Furthermore SCORM 1.2 has proven to be very robust and practical, with a very high success rate for "plug and play", but if the LMS and the content are on different servers, the security enforced by browsers may be an issue. Finally, the network infrastructure needs more thought and care than using the AICC HTTP communication model, which ignores cross-domain security.

As regards internationalization (the ability of running on Locales different than English) the SCORM API intrinsically supports characters required for many languages, because it uses ECMAScript, which means that all parameters are passed and returned as Unicode strings without having to do anything special to encode and decode them. SCORM package manifests and metadata use an XML binding that intrinsically support international character sets.

A conformant LMS should be able to handle Unicode string values as required for communication to and from the SCO; however what a LMS does with such strings, or how it generates them, is outside the scope of SCORM. AICC instead has been developed without this character set flexibility in mind making it dangerous to use for International (i.e. multilingual) applications.

Concluding, IMS Content Packaging should be preferred over the AICC standard for a number of reasons (the main ones are: better overall technical architecture, richer underlying design community, complete internationalization support and an more robust security model).

4. STANDARDS FOR LEARNER MODELLING

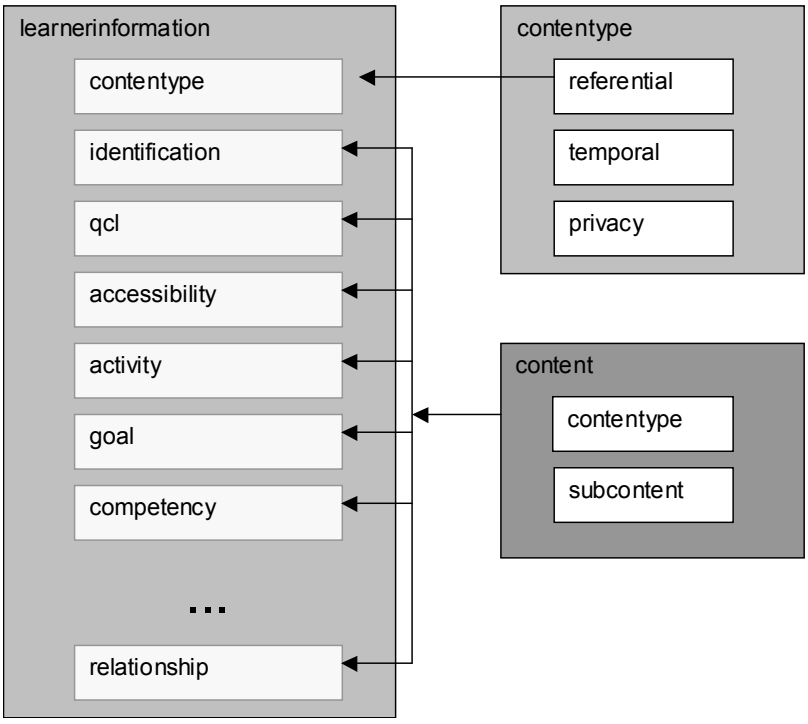
In this section we discuss the main standards for modeling learners' features.

4.1 IMS LEARNER INFORMATION PACKAGE (LIP)

The Learner Information Package (LIP) comes from the IMS, the consortium of institutions including government agencies, software developers and vendors, and training and education representatives. Version 1.0 of the IMS Learner Information Package Specification was released to the public in March 2001. The IMS LIP has partly been derived from the IEEE PAPI Learner (versions 5.0 and 6.0).

The LIP specification provides a way of packaging learner information for exchange between disparate systems. It focuses on learner information, that is, the wide range of information that can be used by different systems to support the learner's activities. The semantics of the packages being exchanged may vary depending on the context; this is determined by the services participating in the exchange. Furthermore, learner information can be packaged from a variety of environments, not only human resources, student information and learning management systems.
An important aspect of the implementation of the XML-based specification to note is that nearly all LIP elements are optional. Depending on needs, data can be packaged to match the basic LIP segment structure or to match the structure of information on either side of the exchange. Either approach is acceptable.

LIP can be used for individual learner information packaging (for example, a student submitting his or her resume to an e-learning website) or for organizational exchange (both intra-organization, like data about employees, or extra-organization, like the certification of a student's achievements to a third-party institution).



The IMS LIP data structure

CORE DATA STRUCTURES

LIP is structured around eleven core data structures, as follows:

1. Accessibility – Data regarding the accessibility of learner's information as defined through:
   - Language: the definition of a learner's language proficiencies.

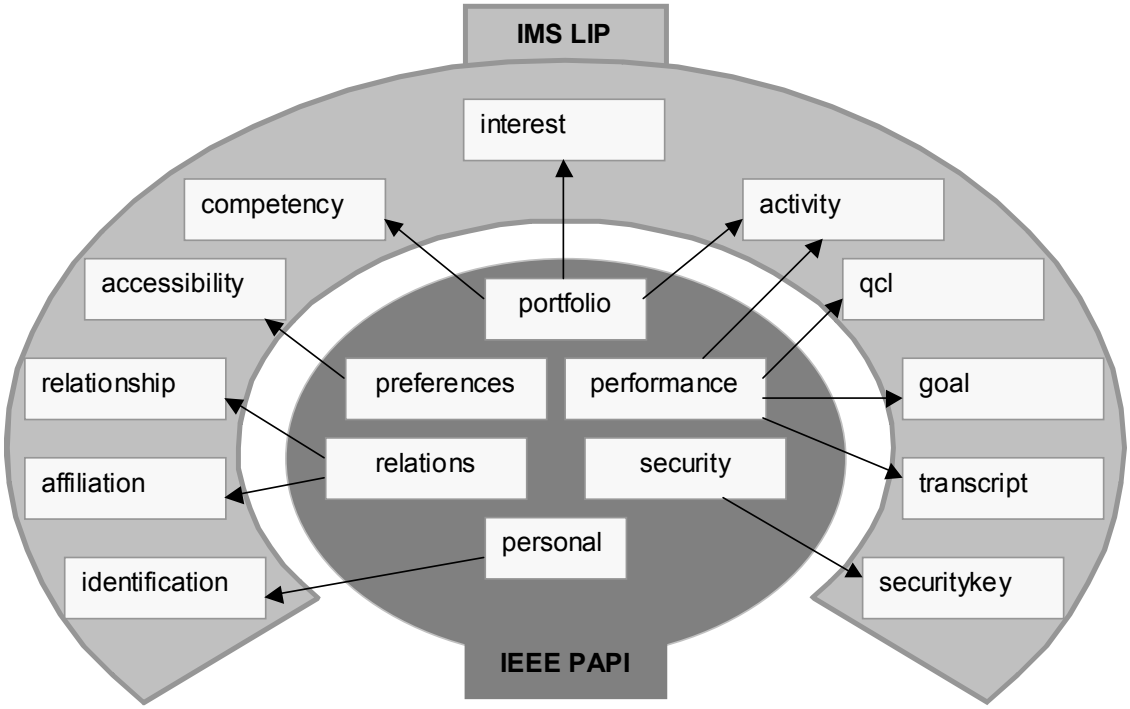
- Preference: the definition of a learner's cognitive, physical and technological preferences.

2. Activity – The activity the learner is engaging in, comprising:
   - Learning activity reference: an external reference mechanism to the learning materials.
   - Definition: the definition of the materials studied.
   - Product: the materials developed by the learners themselves.
   - Testimonial: statements attesting to the capabilities of the learner.
   - Evaluation: the results of the evaluations undertaken.
3. Affiliation – The learner's professional affiliations and associated roles.
   - Competency – The competencies of the learner.
   - Goal – The learner's goals and sub-goals.
4. Identification – The learner identification data. They comprise:
   - Formatted Name: the learner's name, formatted.
   - Name: the learner's name.
   - Address: the learner's addresses.
   - Contact info: electronic-based contact information about the learner.
   - Demographics: demographics information about the learner.
   - Agent: the representatives permitted to act on behalf of the learner.
5. Interest – Hobbies and recreational interests of the learner.
6. Qcl – A description of the qualifications, certifications and various licenses of a learner.
7. Relationship – the set of relationships that are to be defined between the learner and their identification, accessibility, qualifications, competencies, goals, activities, interests, transcripts, security keys and affiliations.
8. Security key – the security-related information for the given learner.
9. Transcript – the transcripts that summarize the performance of the learner.

A full, detailed list of all LIP data elements would be of little interest. What is important is that the standard has been designed to be extensible, in order to accommodate any possible learner data. Of course, the extensions obtained would be proprietary additions.

Relationship with the IEEE LTSC PAPI Specification

As mentioned earlier, the IMS LIP work incorporated the IEEE PAPI specification. The following Figure describes such the relationship.



Relationship between IMS Lip and IEEE PAPI.

An arrow in Figure 4 indicates the mapping between one data structure and another. Hence, data belonging to the IEEE PAPI personal group can be put in the identification IMS LIP data group when using the latter specifications.

4.2 IMS LEARNER INFORMATION PACKAGE ACCESSIBILITY FOR LIP

Defines two new sub-schemas for the IMS Learning Information Package that define a means to specify accessibility preferences and learner accommodations.

The Accessibility for LIP defines two new sub-schemas for the IMS Learning Information Package that define a means to specify accessibility preferences and learner accommodations. These preferences go beyond support for disabled people to include kinds of accessibility needs such as mobile computing, noisy environments, etc. The <accessForAll> element defines accessibility preferences that were left for future work in the IMS Learner Information Package (LIP) specification version 1.0. The "accessibility" data structure includes the following elements: <language>, <preference>, <eligibility>, and <disability> in the LIP. This specification adds the <accessForAll> element under <accessibility> because it is intended to address the needs of learners beyond those with disabilities. The <disability> element is deprecated henceforth.

More details are available at: http://www.imsglobal.org/accessibility/index.cfm

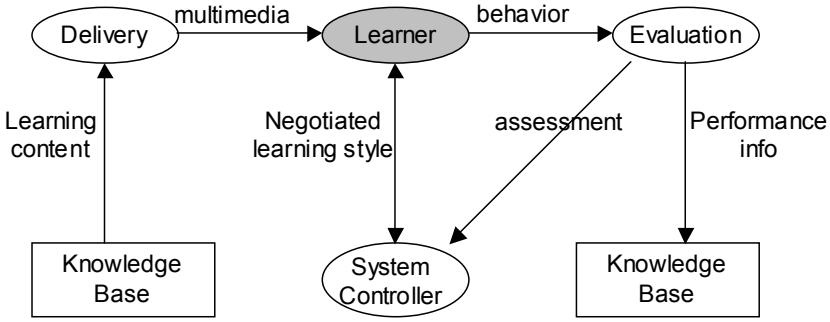
4.3 IEEE PAPI LEARNER

Public and Private Information (PAPI) for Learners (PAPI Learner) is a standard effort aimed at providing the syntax and semantics of a student model, including knowledge, learning styles, skills, abilities, records and

personal information, all at multiple levels of granularity. This standard specifies the syntax and semantics of a "Learner Model", which characterizes a learner (either a student or knowledge worker) and his or her knowledge/abilities. This will include elements such as knowledge (from coarse to fine-grained), skills, abilities, learning styles, records, and personal information. The standard will allow these elements to be represented in multiple levels of granularity, from a coarse overview, down to the smallest conceivable sub-element. It will allow different views of the Learner Model (learner, teacher, parent, school, employer, etc.) while addressing the sensitive issues of privacy and security.
The working group for the Learner Model (P1848.2) has the following purposes:

- To enable learners (students or knowledge workers) to build lifelong personal learner models.
- To enable personalized instruction and effective instruction.
- To provide educational researchers with a standardized source of data.
- To provide a foundation for the development of additional educational standards, from a student-centred learning focus.
- To provide architectural guidance to developers of education systems.

A simple view of the IEEE 1484.2 overall organization is provided in the following Figure.



Overall IEEE 1484.2.

The main architectural feature of the PAPI Learner standard is its logical division. It separates the security and the administration of several types of learner information (also called Profile Information or Learner Profiles):

- Personal information like name, address and social security number. It is not directly related to the measurement and recording of learner performance and is primarily concerned with administration. Usually this type of information is private and secure.
- Relations information, e.g., cohorts, classmates. This concerns the learner's relationship to other users of learning technology systems, such as teachers, practitioners, and other learners.
- Security information. This is concerned with the learner's security credentials, such as passwords, challenges/responses, private and public cryptographic keys, and biometrics.
- Preference information: useful and unusable I/O devices, learning styles and physical limitations. It describes preferences that may improve human-computer interactions.
- Performance information, like grades, interim reports, log books. This pertains to the learner's history, current work or future objectives and is created and used by learning technology components to supply enhanced learning experiences.
- Portfolio information: accomplishments, works and so on. This information is a representative collection of a learner's works or references to them that is intended to illustrate and justify the student's abilities and attainments.

The PAPI Learner Standard is not limited to these six types of information and may be integrated with other systems, protocols, formats, and technologies.

Generally speaking, standards that attempt to be omni-comprehensive risk becoming large and unmanageable. The designers of PAPI Learner aimed at concrete diffusion by avoiding catch-all specifications and focusing on essential, learning-related data while providing an extension mechanism for customization. Another aim of this standard is to ensure different levels of granularity within information definitions. Hence, there are no coarse-

grained information definitions, but rather a continuum, with many gradations and variations. For example, learner information can vary in "distance" from the trainee; local information, typically, is characterized by online availability, higher performance access, and fewer security restrictions. There are also various possible degrees regarding privacy.

PAPI Learner information has been designed in order to follow a well-defined set of requisites. In particular PAPI Learner information must support:

- Cultural conventions (like measure units, currencies and other linguistic conventions).
- Institutional conventions. This mainly concerns learning institutions, each supporting its own conventions (like grading systems, or course denominations). An engineering goal of PAPI Learner is to "let the market solve the problem" of choosing/reducing the number of coding schemes to the "right" level.
- Simple application paradigms, in order to promote adoption. That is, it should require minimal effort to incorporate PAPI Learner codings, APIs, and protocols into existing real-world applications.
- Other engineering requirements include:
  - Controlling access to information to the extent necessary.
  - Maximizing performance of accessing data.
  - Supporting varying data and information structures.
  - Supporting varying coding and extension mechanisms.
  - Supporting varying information partitioning schemes.
  - Letting the market choose the best coding scheme(s).
  - Supporting varying types of online, "sometimes", and offline connectivity.
  - Supporting varying geographic (nomadic) access to information.

Learners' data can be exchanged in three different ways: (i) by means of the external specification, i.e., only PAPI Learner coding bindings are used while some other data communication method is mutually agreed upon by data exchange participants; (ii) by using the control transfer mechanism to facilitate data interchange, e.g., PAPI Learner API bindings and (iii) by employing data and control transfer mechanisms. e.g., PAPI Learner protocol bindings.

Also security features are part of the conceptual model definition, as follows:

- Session-View-Based. Security features are provided on a per-session, per-view basis. Each security session is initiated by an accessor (a user or agent that requests access). The accessor provides security credentials that authenticate the accessor, authorize the accessor, or both. A view represents a portion of PAPI Learner information and is similar to the notion of a database "view". Each view established represents a session, i.e., the "session" represents the duration of access and the "view" represents the scope of access.
- Security Parameter Negotiation. Data interchange participants negotiate security parameters prior to, during, and after each session. The security parameters are defined in the PAPI Learner bindings.
- Security Extension. Additional security features may be used that were not foreseen. The method of incorporating security extensions is defined in the PAPI Learner bindings.
- Access Control. Accessors may attempt read or write access to data elements, to create new data elements (separately or within aggregates), to destroy data elements (separately or within aggregates), and/or to change attributes of data elements. Other access methods, if any, are implementation-defined.
- Identification. The methods for identifying learners are implementation-defined. A related standard, IEEE 1484.13 ("Simple Human Identifiers"), defines the data type associated with a learner identifier.
- Authentication. The methods of authenticating users are outside the scope of the PAPI Learner Standard.
- De-identification. PAPI Learner prescribes that all information, except learner personal information, should be de-identified (that is, students should not be identified). The methods of de-identifying learners and their information are outside the scope of this Standard.
- Authorization. The methods of authorizing operations are implementation-defined.
- Delegation. The methods of delegating administration, authority, or credentials are implementation-defined.

- Non-Repudiation. The methods of non-repudiation are implementation-defined.
- Repudiation. The methods of repudiating data, users, or credentials are implementation-defined.
- Privacy. This Standard supports security frameworks and approaches that permit the implementation of a wide variety of privacy frameworks.
- Confidentiality. This Standard supports access controls and the partitioning of information types that permit the implementation of a wide variety of confidentiality frameworks.
- Encryption. PAPI Learner supports several security frameworks and techniques that permit the integration of various encryption models and technologies.
- Data Integrity. This Standard supports information assurance frameworks and approaches that permit the implementation of a wide variety of data integrity frameworks.
- Validation of Certificates. PAPI Learner does not require validation of student performance information, but supports the parameterisation of automated validation.
- Digital Signature. This Standard adopts third-party signing frameworks harmonized with ISO/IEC 15945.
- IEEE 1484.2.3, PAPI Learner Information Security Notes, contains information about applying security techniques and technologies to PAPI Learner implementations.

4.4 SABA PROFILE FORMAT

Profile Format (http://www.saba.com/standards/ulf/Specification/specPROF.htm) is an XML-based representation for describing learner profile information. Learner profiles comprise a variety of data about learners, including personal and job information, learning history, goals and plans, and held competencies and certifications. Profile Format captures this information in an XML-based format using RDF to define metadata for describing learners. Profile Format incorporates several existing metadata standards, including the Dublin Core and vCard, which ensures compatibility with existing person/profile descriptions.

By employing Profile Format to describe the learners in a system, learning providers can extend their learning management architecture to support all of the following:

- Searches of critical learner metadata such as name, title, role, learning results, and held competencies and certifications
- Tracking the learning history of individual learners
- Assignment of competencies (with proficiency levels) and certifications to learners
- Assignment of learning goals to learners and tracking of progress towards fulfilment of those goals
- Creation of distributed profiles, where portions of a learner's profile are provided by different sources
- Compatibility with standard web search engines
- Profile Format is based on open standards and is designed to reflect the following principles:
- Compatibility with emerging industry standards for learning profiles, including ongoing work in IMS and IEEE.
- Extensibility to easily accommodate future growth and change.

Profile Format employs an XML standard known as RDF (Resource Description Framework), the standard for defining metadata to describe web-based resources. The use of RDF makes it possible to define a set of unique RDF properties and merge these properties with properties defined in existing standards, such as vCard and Dublin Core. RDF also provides a unified mechanism for manipulating and querying this merged metadata. Furthermore, the use of RDF allows Profile Format to support distributed profiles, where portions of a learner's profile are provided by different sources.

A Profile Format document is an RDF document that contains one or more Description elements, where each element describes a learner in the system. Each Description element contains a unique identifier and a set of property/value pairs that fully describe the learner. These properties can draw from any of the Profile Format RDF schemas.
Each Description element has an attribute that unambiguously identifies the learner being described. This

attribute can be either of the following:

- id
- about

The Description element can also include the xml:lang attribute for specifying the language in which the metadata description is authored. The xml:lang attribute contains the ISO 639 /RFC 1766 language code with an optional geographic identifier, such as en for English, or fr for French.

Profile Format subdivides learner information into the following categories:

- Personal information. This includes information such as name, address, title, role(s), and organisation membership. All personal information is represented using RDF mappings of the vCard specification.
- Learning information. Profile Format defines a set of RDF properties that capture information on a learner's current learning (current enrolments) and learning history (transcript).

The learning property specifies a URL to a learning resource described in a Catalog Format document. The specified resource is a held offering in the learner's transcript.
In its simplest form, the learning property contains only the URL of the held learning offering, for example:
<profile:learning rdf:resource="http://www.saba.com/learning/catalog.rdf#JAVA101"/>
The learning property can also be a structured property, with substructure properties that qualify the status of the learning offering and the conditions under which it was attained. For qualified instances of the learning property, the URL of the learning resource is captured using the rdf:value property.

Profile Format defines a set of RDF properties that capture information about a learner's goals. Goals can encompass both business and professional objectives for a learner and include the following additional information:

- planned actions for achieving the goal
- learning interventions
- accomplishments

The goal property defines a learner's goal. The goal property can also be a structured property, with substructure properties that provide details about the goal and its status. For qualified instances of the goal property, the URL of a qualified goal is captured using the rdf:value property.

- Profile Format defines a set of RDF properties that capture information reflecting a learner's progress towards specific goals. These observations track specific, measurable milestones on the path towards achieving a goal.
- Competency information includes information on held competencies. This category contains a pointer to a competency defined in a Competency Format document, with optional properties describing how the competency was attained.
- Certification information contains a pointer to a certification track defined in a Certification Format document, with optional properties describing how the certification was attained.
- Preference information Profile Format defines a set of RDF properties that capture information reflecting the learning preferences of a learner. Includes information on learner preferences, such as home language and country.
- Profile Information includes information about the profile itself, such as the date it was generated and the language it is in. All profile information is represented using the RDF mappings of Dublin Core.

4.5 COMPARISON AND CHOICE

Before discussing the technical details could be useful to trace briefly some of the history of the specification introduced in the previous sections. Originally, IMS was working on PAPI Learner from 1997 through 1999. By the end of 1999, IMS decided that the work should be finally transferred to IEEE (previously, there were "joint" activities on the work). In mid-2000, the IMS enterprise committee, which all along had not been working on learner information, decided to start work in learner information. They produced a document in 2001 called the Learner Information Profile (LIP), including a data model, XML binding, and other supporting documents. Unfortunately, the LIP and (formerly-called) PAPI specs are competing specifications in that they both describe the same topic called namely "learner information".

As a difference between the two standards, the PAPI Learner specifications use a registry-based approach for long-term maintenance of the “value domains” and their "permissible values" (often called "vocabularies"). Moreover, LIP uses XML while the (formerly-called) PAPI Learner and its related standards uses ISO/IEC 11179 and ISO/IEC 20944 for a complete suite of harmonized bindings. Finally, LIP organizes data according to “purposes” while PAPI Learner does not. This is a useful characteristic in those application contexts where data needs to be organized in customized ways.

The previous considerations make LIP a better candidate than PAPI Learner, and as this it is currently used within the Diogene project for student management.

5. STANDARDS FOR ONTOLOGY MODELLING

This section describes the main standards available for ontology modeling.

5.1 W3C RESOURCE DESCRIPTION FRAMEWORK (RDF) AND RDF-SCHEMA

W3C proposed the Resource Description Framework (RDF), a general-purpose metadata standard for cross-discipline applications. An RDF resource is any object uniquely identifiable by a Uniform Resource Identifier (URI). RDF extends XML to be specific for describing resources.

RDF is designed to facilitate semantic interoperability by representing a domain model in terms of simple object-relation-object triples, and techniques from Knowledge Representation can be used to help find mappings between two RDF descriptions. Of course this does not solve the general interoperability problem (i.e., finding semantic-preserving mappings between objects), but the usage of RDF for data interchange raises the level of potential reuse much beyond the parser reuse which is all that one would obtain from using plain XML. Moreover, since RDF describes a layer independent of XML, an RDF domain model (and software using the RDF model) could still be used, even if XML syntax changes or becomes obsolescent.

One of the most important peculiarities of RDF is that it provides the ability to unambiguously identify the semantics of shared vocabularies of terms used to describe data. This is done through the XML Namespace mechanism, which uniquely establishes the governing authority of the vocabulary. Thus, if you read the attribute (“property”) author in RDF metadata, through the Namespace mechanism it is possible to understand its semantic simply by checking the vocabulary source.

If people want to share their resource descriptions, there has to be an agreement on a standard core of vocabulary in terms of modelling primitives that should be used to describe metadada. RDF schemas (RDF/S) attempt to provide this standard vocabulary. RDF provides a syntactical convention and a simple data model for representing machine-processable semantics of data; RDFS defines basic ontological modelling primitives on top of RDF. RDF schemas provide a notion of concepts (class), slots (property), inheritance (SubClassOf) and range restrictions (ConstraintProperty). However, at the moment, the semantics are not well-defined. It provides a common syntax + basic vocabulary, but needs an additional "logical level" that defines a clear semantics for RDF expressions and provides a basis for integration mechanisms.
In their current states neither XML nor RDF provides sufficient support for the integration of heterogeneous structures or different meanings of terms. The Semantic Web won’t be possible until agents have the means to figure out some things by themselves, given the data they have to work with. Fortunately, artificial intelligence gives us two tools to help make this possible. First, knowledge representation is a field that defines how we might represent, in computers, some of what is stored in our brains. This would give computers a chance of synthesizing unclassified data at a useful speed. Second, inference is a way of using formal logic to approximate further knowledge from that which is already known. All of this forms a system of representing and synthesizing knowledge that is often referred to as an ontology. A further representation layer is needed on top of the currently available layers.

For sharing information and knowledge (i.e. for interoperability) between different applications, a shared set of terms describing the application domain with a common understanding is needed. More flexibility is gained if not just a flat set of terms is defined, but also the relationships between these terms. Such a set of terms is called

an "ontology".
A broker has access to information (metadata) describing data sources registered with the broker. When a user submits a request for information, the broker uses this metadata to check which data source is able to handle the request. If there is a match, the broker establishes a connection between the user and the source. This semantic mapping can be performed using ontologies (detailed descriptions).
For each data source, the respective ontology holds information about the underlying technology and concepts. Data sources A and B may be linked by automatically matching the concepts and terms described in Ontology A with those described in Ontology B. This way, no static filter function Filter A-B is needed (filters contain translation rules for mapping one database to another). The main advantages of this approach are:

- each data source has one ontological description, and the growth in the number of ontologies needed to link multiple data sources is linear, e.g. to link 10 data sources, 10 ontologies are needed.
- if a data source changes, only the respective ontology has to be updated.
- even without an update of the respective ontology, the semantic mapping with respect to the concepts and terms specified for the "old" data source remain functional.
- to integrate a new data source into an existing network of data sources, the new data source simply has to be registered with the broker. No new filters have to be written.
- implicit knowledge hidden in the data sources can be made explicit through automatic inference. This applies also to knowledge that is the result of the combined knowledge stored in many distributed data sources.

Using the syntax of XML and RDF various proposals for ontologies of Web contents have been formulated.

As regards the RDF schema, one limitation of the above described mechanism lies in the way application-specific classes and properties can be defined. As we know RDF provides a way to express simple statements about resources, using named properties and values. However, RDF user communities also need the ability to define the vocabularies (terms) they intend to use in those statements, specifically, to indicate that they are describing specific kinds or classes of resources, and will use specific properties in describing those resources. RDF itself provides no means for defining application-specific classes and properties. Such classes and properties are described as an RDF vocabulary, using extensions to RDF provided by the RDF Vocabulary Description Language 1.0: RDF Schema.
RDF Schema does not provide a vocabulary of application-specific classes and properties. Instead, it provides the facilities needed to describe such classes and properties, and to indicate which classes and properties are expected to be used together. In other words, RDF Schema provides a type system for RDF. The RDF Schema type system is similar in some respects to the type systems of object-oriented programming languages such as Java. For example, RDF Schema allows resources to be defined as instances of one or more classes. In addition, it allows classes to be organized in a hierarchical fashion. However, RDF classes and properties are in some respects very different from programming language types. RDF class and property descriptions do not create a straightjacket into which information must be forced, but instead provide additional information about the RDF resources they describe.
The RDF Schema facilities are themselves provided in the form of an RDF vocabulary; that is, as a specialized set of predefined RDF resources with their own special meanings. The resources in the RDF Schema vocabulary have URI refs with the prefix http://www.w3.org/2000/01/rdf-schema#. Vocabulary descriptions (schemas) written in the RDF Schema language are legal RDF graphs. Hence, RDF software that is not written to also process the additional RDF Schema vocabulary can still interpret a schema as a legal RDF graph consisting of various resources and properties, but will not "understand" the additional built-in meanings of the RDF Schema terms. To understand these additional meanings, RDF software must be written to process an extended language that includes not only the rdf: vocabulary, but also the rdfs: vocabulary, together with their built-in meanings.

5.2 W3C OWL (LITE, DL E FULL)

The OWL Web Ontology Language [64] is being designed by the W3C Web Ontology Working Group as a revision of DAML+OIL. It takes DAML+OIL as its basis and its expressiveness will be close to that of DAML+OIL.
The following introduction accounts for the latest extensions (as of 18 August 2003).

OWL can be thought of being composed of three sublanguages. Such languages are designed for use by specific communities of implementers and users. We report them here in order of increasing expressiveness:

- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL.
- *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computed) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL.
- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

Each of these languages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. Hence, for example every legal OWL Lite ontology is a legal OWL DL ontology, while every legal OWL DL ontology is a legal OWL Full ontology and every valid OWL Lite conclusion is a valid OWL DL conclusion. Finally, every valid OWL DL conclusion is a valid OWL Full conclusion.

Ontology developers adopting OWL should consider which sublanguage best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more-expressive constructs provided by OWL DL and OWL Full. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modeling facilities of RDF Schema (e.g. defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable since complete OWL Full implementations do not currently exist.

OWL Full can be viewed as an extension of RDF, while OWL Lite and OWL DL can be viewed as extensions of a restricted view of RDF. Every OWL (Lite, DL, Full) document is an RDF document, and every RDF document is an OWL Full document, but only some RDF documents will be a legal OWL Lite or OWL DL document. Because of this, some care has to be taken when a user wants to migrate an RDF document to OWL. When the expressiveness of OWL DL or OWL Lite is deemed appropriate, some precautions have to be taken to ensure that the original RDF document complies with the additional constraints emposed by OWL DL and OWL Lite. Among others, every URI that is used as a class name must be explicitly asserted to be of type owl:Class (and simililarly for properties), every individual must be asserted to belong to at least one class (even if only owl:Thing), the URI's used for classes, properties and individuals must be mutually disjoint.

In particular DAML+OIL is extended in two ways:

- A lower step-in threshold (which is perceived to be too high for DAML+OIL) is provided.
- A human-readable presentation syntax is provided, besides an RDF/XML-based syntax.

The W3C working draft specifies eight design goals for the Web ontology language:

- Shared ontologies. Ontologies should be publicly available and different data sources should be able to commit to the same ontology for shared meaning. Also, ontologies should be able to extend other ontologies in order to provide additional definitions.
- Ontology evolution. An ontology may change during its lifetime. A data source should specify the version of an ontology to which it commits. An important issue is whether or not documents that commit to one version of an ontology are compatible with those that commit to another. Both compatible and incompatible revisions should be allowed, but it should be possible to distinguish between the two. Note that since formal descriptions only provide approximations for the meanings of most terms, it is possible for a revision to change the intended meaning of a term without changing its

formal description. Thus determining semantic backwards-compatibility requires more than a simple comparison of term descriptions. As such, the ontology author needs to be able to indicate such changes explicitly.

- Ontology interoperability. Different ontologies may model the same concepts in different ways. The language should provide primitives for relating different representations, thus allowing data to be converted to different ontologies and enabling a "web of ontologies."
- Inconsistency detection. Different ontologies or data sources may be contradictory. It should be possible to detect these inconsistencies.
- Balance of expressivity and scalability. The language should be able to express a wide variety of knowledge, but should also provide for efficient means to reason with it. Since these two requirements are typically at odds, the goal of the web ontology language is to find a balance that supports the ability to express the most important kinds of knowledge.
- Ease of use. The language should provide a low learning barrier and have clear concepts and meaning. The concepts should be independent from syntax.
- Compatibility with other standards. The language should be compatible with other other commonly used Web and industry standards. In particular, this includes XML and related standards (such as XML Schema and RDF), and possibly other modeling standards such as UML.
- Internationalisation. The language should support the development of multilingual ontologies, and potentially provide different views of ontologies that are appropriate for different cultures.

These design goals motivate a number of requirements for a Web Ontology Language. The Working Group currently feels that the requirements described below are essential to the language:

- Ontologies must be objects that have their own unique identifiers, such as a URI reference.
- Two terms in different ontologies must have distinct absolute identifiers (although they may have identical relative identifiers). It must be possible to uniquely identify a term in an ontology using a URI reference.
- Ontologies must be able to explicitly extend other ontologies in order to reuse terms while adding new classes and properties.
- Resources must be able to explicitly commit to specific ontologies, indicating precisely which set of definitions and assumptions are made.
- It must be possible to provide meta-data for each ontology, such as author, publish-date, etc. The language should provide a standard set of common metadata properties. These properties may or may not be borrowed from the Dublin Core element set.
- The language must provide features for comparing and relating different versions of the same ontology. This should include features for relating revisions to prior versions, explicit statements of backwards-compatibility, and the ability to deprecate terms.
- The language must be able to express complex definitions of classes. This includes, but is not limited to, sub classing and Boolean combinations of class expressions (i.e., intersection, union, and complement).
- The language must be able to express the definitions of properties. This includes, but is not limited to, sub properties, domain and range constraints, transitivity, and inverse properties.
- The language must provide a set of standard data types. These data types may be based on XML Schema data types.
- The language must include features for stating that two classes or properties are equivalent.
- The language must include features for stating that pairs of identifiers represent the same individual.
- In general, the language will not make a unique names assumption. That is, distinct identifiers are not assumed to refer to different objects (see the previous requirement). However, there are many applications where the unique names assumption would be useful. Users should have the option of specifying that all of the names in a particular namespace or document refer to distinct objects.
- The language must provide a way to allow statements to be "tagged" with additional information such as source, timestamp, confidence level, etc.
- The language must support the ability to treat classes as instances. This is because the same concept can often be seen as a class or an individual, depending on the perspective of the user.
- The language must support the definition and use of complex/ structured data types. These may be used

to specify dates, coordinate pairs, addresses, etc.
- The language must the support the specification of cardinality restrictions on properties. These restrictions set minimum and maximum numbers of object that any single object can be related to via the specified property.
- The language should have an XML serialisation syntax.
- The language should support the specification of multiple alternative user-displayable labels for the objects within an ontology. This can be used, for example, to view the ontology in different natural languages.
- The language should support the use of multilingual character sets.
- In some character encodings, e.g. Unicode based encodings, there are some cases where two different character sequences look the same and are expected, by most users, to compare equal. Given that the W3C I18N WG has decided that early uniform normalization (to Unicode Normal Form C) as the usual approach to solving this problem, any other solution needs to be justified.

An OWL ontology is a sequence of axioms and facts, plus inclusion references to other ontologies which are considered to be included in the ontology. OWL ontologies are web documents, and can be referenced by means of a URI.
Axioms are used to associate class and property IDs with either partial or complete specifications of their characteristics, and to give other logical information about classes and properties.
Class axioms include the common, widely understood, frame idiom. The abstract syntax used here is meant to look somewhat like the syntax used in some frame systems. Each frame-like class axiom contains a collection of more-general classes; a collection of local property restrictions, in the form of restriction constructs; and a collection of descriptions. The restriction construct gives the local range of a property, how many values are permitted, and a collection of required values. Descriptions are used to specify boolean combinations of restrictions and other descriptions as well as construct sets of individuals. Classes can also be specified by enumeration or be made the same or disjoint.
Properties can be the equivalent to or subproperties of others; can be made functional, inverse functional, or transitive; and can be given global domains and ranges. However, most information about properties is more naturally expressed in restrictions, which allow local cardinality and range information to be specified.
Facts state information about particular individuals in the form of a class that the individual belongs to plus properties and values. Individuals can either be given an individualID or be anonymous (blank nodes in RDF terms).

To recap, OWL is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. OWL is a revision of the DAML+OIL web ontology language incorporating lessons learned from the design and application of DAML+OIL.

In the following section we will introduce an older knowledge representation format, still interesting for several aspects.

5.3 UNIVERSITY OF MARYLAND SHOE

Simple HTML Ontology Extensions (SHOE) is a language for knowledge representation based on the ontology approach and XML syntax. Like XML, SHOE is also an extension of HTML.
In the present work on knowledge representation methods, we have often underlined the necessity of choice, in defining a representation formalism, between large expressive power or good computational costs. SHOE is not as expressive as other ontology systems such as Ontolingua, based on the logic language KIF. On the contrary, it is quite simple and has a tractable computational complexity. SHOE allows representing not only metadata, composed of a set of attributes, but it also allows the definition of relationships between the individual objects and the classes of the domain. It inherits the expressive power of semantic networks characteristic of the ontology-based knowledge representation methods.

An example of SHOE is reported below.

```
<INSTANCE KEY="http://www.somesite.com/somepath/somepage.html">
<USE-ONTOLOGY ID="id-ontology" URL="http://www.somesite.com/somepath/id-ontology.html"
        VERSION="1.0" PREFIX="id">
<CATEGORY NAME="id.Some">
        <RELATION NAME="id.name">
        <ARG POS=1 VALUE="value">
        <ARG POS=2 VALUE="Another value">
</RELATION>
    <RELATION NAME="id.member">
        <ARG POS=1 VALUE="another data">
        <ARG POS=2 VALUE="http://www.somesite.com">
    </RELATION>
   <RELATION NAME="id.attr">
        <ARG POS=1 VALUE="http://www.somesite.com/some">
        <ARG POS=2 VALUE="data">
     </RELATION>
        <RELATION NAME="id.value2">
        <ARG POS=1 VALUE="another value">
     </RELATION>
</INSTANCE>
```

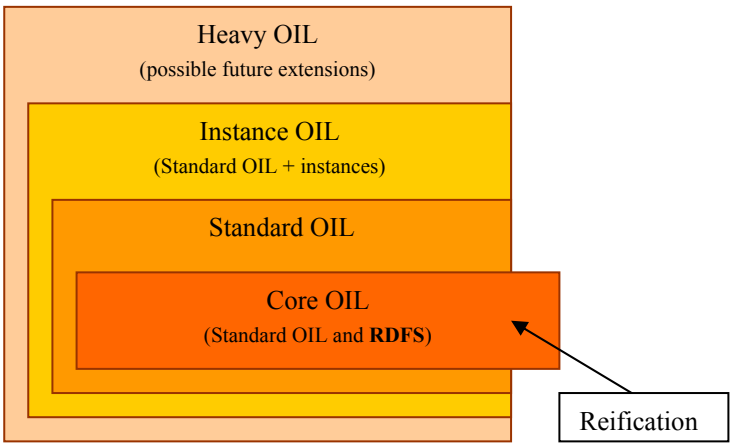
The above fragment (added in the BODY section of an HTML page) allows to index the HTML page providing relationships structure with other existing ontologies (in this case the ontology referred as “id-ontology”).

5.4 DARPA DAML AND DAML+OIL

This section describes the standards DARPA DAML and DAML+OIL

OIL

The Ontology Inference Layer (OIL) is another ontology representation language compatible with Web standards. It derives from Frame representation languages (Section 4.1) and Description Logics (Section 3.4); indeed, its objective is to build on top of RDF Schema by adding modelling constructs for Description Logics. OIL is based on a multi-layer architecture. Each layer is an extension of the level below, obtained by adding expressiveness to it. The aim of this architecture is to allow intelligent agents not able to perform complex inferences to access only low layers of a description, in order to be able to partially understand it. The following figure shows the relation between OIL and RDF.



Relation between OIL and RDF

As shown in the figure, Core OIL largely overlaps with RDFS (exception for reification features). This means that simple RDF-based agents can also process OIL ontologies, although they will not be able to completely understand them.
Standard Oil includes a set of modeling primitives which ensure an expressive power enough to perform inferences. This layer allows individual variables to be expressed in term definitions.
Instance OIL includes a complete instance integration and database functionalities.
Heavy OIL, finally, is a layer providing possible future extension.
An OIL ontology is a structure whose elements can themselves be sub-structures. Moreover, these elements can be mandatory, optional (indicated by the symbol "?") or multiple (indicated by the symbol "+" or "*", depending on whether the empty value is admissible).

5.5 Comparison and Choice

Having seen the main characteristics of each of the standards regarding ontology modeling, the choice regarding the adoption of SHOE and DAML+OIL seems the most reasonable for the Diogene project. On the other hand, in a long-term perspective a shift towards the adoption a newer language like OWL (and within the OWL family, OWL Lite) seems the natural choice, on the behalf of the Diogene project.

Finally, given the important interoperability features that enables, an important extension that should be incorporated in the project is the compliance with the RDF Schema specification.

6. STANDARDS FOR COMPETENCIES MODELLING

This section discusses the currently available standards for modeling competencies in e-learning systems.

6.1 IMS Reusable Definition of Competency or Educational Objective (RDCEO)

This standard provides a means to create common understandings of competencies that appear as part of a learning or career plan, as learning pre-requisites, or as learning outcomes.

The Reusable Definition of Competency or Educational Objective (RDCEO) specification provides a means to create common understandings of competencies that appear as part of a learning or career plan, as learning pre-requisites, or as learning outcomes. The information model in this specification can be used to exchange these definitions between learning systems, human resource systems, learning content, competency or skills repositories, and other relevant systems. RDCEO provides unique references to descriptions of competencies or objectives for inclusion in other information models.

This specification defines an information model for describing, referencing, and exchanging definitions of

competencies, primarily in the context of online and distributed learning. In this specification, the word competency is used in a very general sense that includes skills, knowledge, tasks, and learning outcomes. This specification gives a way to formally represent the key characteristics of a competency, independent of its use in any particular context. It enables interoperability among learning systems that deal with competency information by providing a means for them to refer to common definitions with common meanings.

The core information in a Reusable Definition of Competency or Educational Objective (RDCEO) is an unstructured textual definition of the competency that can be referenced through a globally unique identifier. This information may be refined using a user-defined model of the structure of a competency.

The RDCEO specification provides a means to create common understandings of competencies that appear as part of a learning or career plan, as learning pre-requisites, or as learning outcomes. The information model in this specification can be used to exchange these definitions between learning systems, human resource systems, learning content, competency or skills repositories, and other relevant systems. RDCEO provides unique references to descriptions of competencies or objectives for inclusion in other information models.

The RDCEO that conform to this specification are intended for interchange by machines, but the information they contain is currently intended for human interpretation. This specification does not address the aggregation of smaller competencies into larger competencies and does not address how competencies are to be assessed, certified, recorded, or used as part of a process such as instructional design or knowledge management. It also does not specify how records of competencies associated with an individual are structured, stored, or exchanged.

More details are available at: http://www.imsglobal.org/competencies/index.cfm

6.2 IEEE LEARNING TECHNOLOGY COMPETENCY DEFINITIONS

This standard provides a means to create common understandings of competencies that appear as part of a learning or career plan, as learning pre-requisites, or as learning outcomes. It is a set of recommended practices for Digital Rights Expression Languages Suitable for e-Learning Technologies.

This standard is still in an early stage of development. This standard shall specify the mandatory and optional data elements that constitute a Competency Definition as used in a Learning Management System, or referenced in a Competency Profile. This standard is intended to satisfy the following objectives:

- Provide a standardized data model for reusable Competency Definition records that can be exchanged or reused in one or more compatible systems
- Reconcile various existing and emerging data models into a widely acceptable model
- Provide a standardized way to identify the type and precision of a Competency Definition
- Provide a unique identifier as the means to unambiguously reference are usable Competency Definition regardless of the setting in which this Competency Definition is stored, found, retrieved, or used. For example, metadata that describe learning content may contain a reference to one or more Competency Definition records that describe the learning objectives for the content.
- Provide a standardized data model for additional information about a Competency Definition, such as a title, description, and source, compatible with other emerging learning asset metadata standards
- Provide a controlled vocabulary to express how competency definitions are semantically related.

Further documentation is available at: http://ltsc.ieee.org/wg20/par1484-20.html.

6.3 COMPARISON AND CHOICE

The following standards are discussed in this section:

- IMS Reusable Definition of Competency or Educational Objective (RDCEO)
- IEEE Learning Technology Competency Definitions

At the end of 2003 IMS donated the Reusable Definition of Competency or Educational Objective (RDCEO) specification to the IEEE Learning Technology Standards Committee (LTSC) for development as a standard.

The LTSC Competency Definitions standards could usefully used in the Diogene Project even if its adoption could be postponed for other more important extensions to the project.

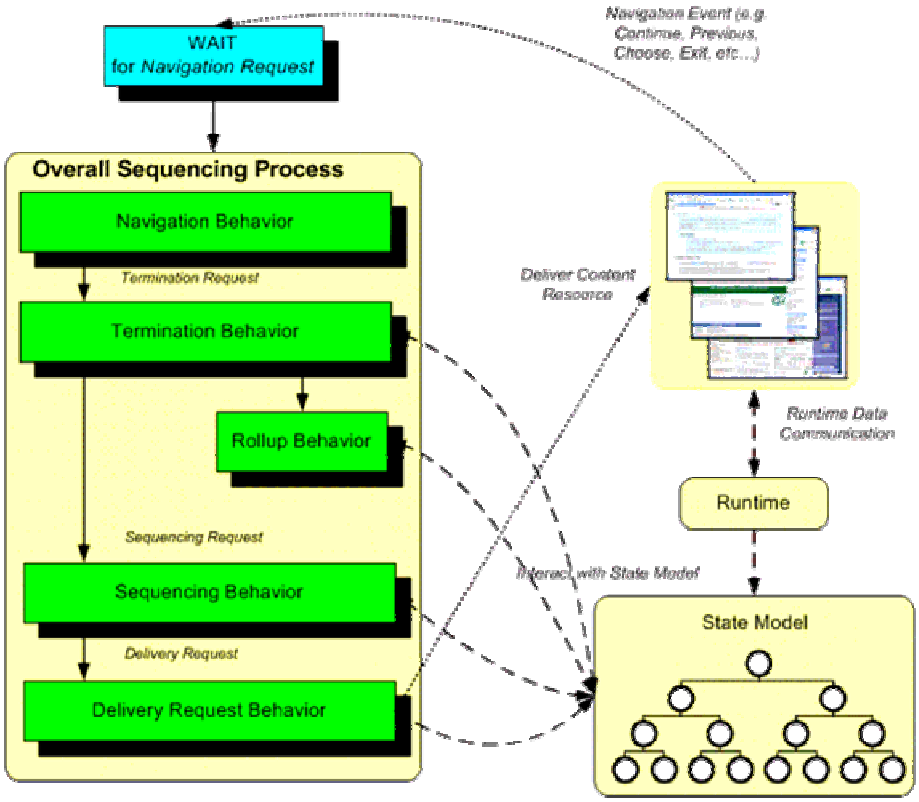
7. STANDARDS FOR MODELING OF LEARNING ACTIVITIES

This section discusses the available standards for modeling learning activities within e-learning systems.

7.1 IMS SIMPLE SEQUENCING

The IMS Simple Sequencing standard defines a method for representing the intended behavior of a learning experience such that any learning technology system can sequence discrete learning activities in a consistent way. Because of its complexity we report here just few details. The interested reader can see the related documentation available at: http://www.imsproject.org/simplesequencing/.

The following figure describes the sequencing loop devised in the underlying conceptual model.



The sequencing loop and its main steps throughout the sequencing process.

The Figure above shows the various steps in the sequencing process. The control process is shown on the left. In normal operation, the overall sequencing process flows from navigation behavior to termination behavior to sequencing behavior to delivery behavior, followed by a wait while the learner interacts with the content resource.

The right side of the previous Figure shows the learner's view of the learning experience. A content resource is delivered to the learner. The learner interacts with the content resource and results may be returned to the tracking model. The learner triggers an event that maps to a navigation request. The navigation request triggers the various steps in the sequencing process.

Throughout all sequencing processes, a collection of state and tracking model data is maintained. Content resources may directly set values in the tracking model through a runtime communications interface to tracking model; this interface is not part of the specification and is not required. All of the other sequencing processes access and update elements of the tracking models.

Changes to the state of an activity occur because of learner interaction with a content resource delivered for the activity or one of its descendents. If an external event affects the tracking, such as an instructor changing a grade, the model assumes processes are invoked as required to update the activity state model.

Changes to the activity state model and tracking model that occur outside of the scope of delivering the learning experience via the sequencing process are outside of the scope of the specification. Systems that implement such side effects and additional capabilities must deal with these conditions accordingly.

The specification describes a collection of sequencing data that comprises the following information:

- **controlMode**
  describes the control that can be exerted over the particular activity (for example whether is a forward-only activity)

- **sequencingRules**
  specifies detailed rules for the following possible rules:

  - precondition
  - exit condition
  - postcondition

- **limitConditions**
  defines the limits for the fruition of the given activity, like for example the attempt limit (for example this activity can be attempted for no more than 3 times)

- **auxiliaryResources**
  specifies auxiliary resources, if any.

- **rollupRules**
  defines the set of rollup rules that are to be applied to an activity. Each activity may have an unlimited number of rollup rules and the order in which they are defined is not significant.

- **objectives**
  defines the set of objectives that are to be associated with an activity. Each activity must have at least one primary objective and may have an unlimited number of objectives.

- **randomizationControls**
  specifies the descriptions of how children of an activity should be ordered during the sequence process. Simple Sequencing processes may reference the randomization control data for any activity in the activity tree and when this is absent the default values should be used.

- **deliveryControls**
  represents the description of the actions and controls used when an activity is delivered. Simple Sequencing processes may reference the delivery control data for any activity in the activity tree. The delivery controls are optional and if these are absent then the default values must be used.

- **#wildcard**
  This is the Simple Sequencing Binding extension facility, for expanding the sequencing definition as needed.

7.2 IMS LEARNING DESIGN

The Learning Design Information Model can be thought of as an integration of the Educational Modelling Language (EML) work and the existing IMS Specifications, notably Content Packaging, Meta-Data and Simple Sequencing.

As introduced before, 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'.
- IMS Simple Sequencing.
  The IMS Simple Sequencing Specification can be used to
  - sequence the resources within a learning-object and
  - 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.
- 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.
- 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.
- 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.
- IMS Learner Information Package.
  The structure of IMS Learning Design properties can be mapped fully to the IMS LIP.
- IMS Enterprise can be used for mapping learners and support staff to roles when instantiating a learning design.

Finally, it's worth noting that with the IMS Learning Design Specification it is possible to include SCORM content within a learning design.

COMPLIANCE LEVELS

Learning Design specifies three levels of implementation and compliance. 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.

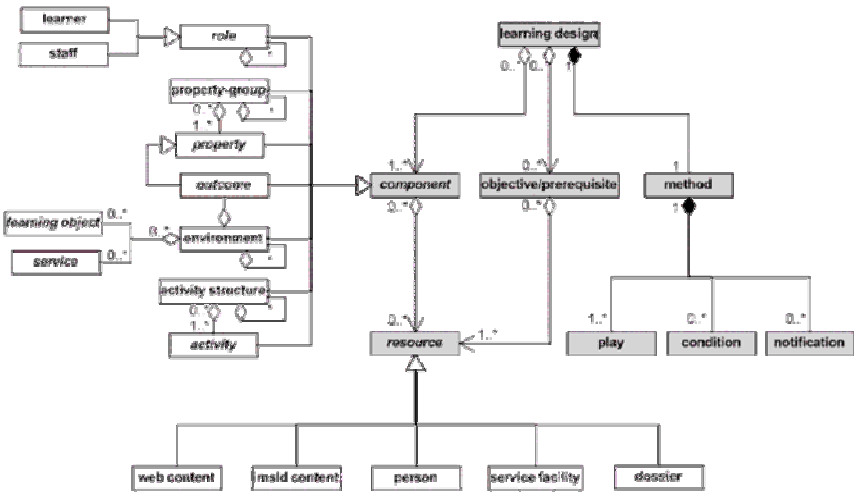
- Learning Design Level C adds Notification to level B, which, although a fairly small addition to the specification, adds significantly to the capability, but potentially also to the implementation task where something similar is not already in place.

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. Compliance with Level A would be relatively easy to achieve. The option for implementors 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.

BASIC MODELS

There are three basic models in the Learning Design Specification: 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 UML diagram in the following Figure represents only aggregation relationships (including compositions) and the specializations of abstract classes ('types') of the conceptual model of the semantic aggregations levels in the Learning Design Specification.



Semantic aggregation levels in the Learning Design specification (Level C)

Further information about the IMS Learning Design standard is available at http://www.imsglobal.org/learningdesign/index.cfm.

7.3 UNED UNIVERSITY'S PALO

PALO Language is an Educational Modeling Language developed at the Department of Languages and Computer Systems of UNED University, in exploitation since 1998. PALO defines a cognitive approach of an

EML that describe courses structured in modules. Each module includes a declaration of the structure, the activities students and tutors undertake, and the scheduling of activities and content. Module and task sequencing is scheduled by mean of attributes of the language, providing deadlines and dependences between modules and tasks based on different types of prerequisites.

PALO defines learning scenarios by mean of instructional templates. An instructional template defines a type of learning scenario with certain pedagogical properties. Pedagogical domain is defined by mean of elements of the language that provides different functionality.
Instructional templates, however, do not define different "languages" by themselves, but a subset of the element of the language that provides a certain type of pedagogical functionality. A group of consistent elements of PALO that provide an instructional or pedagogical purpose constitute a template. Furthermore, PALO is intended to be pedagogically neutral. Elements of the language provide, however, the capability of configuring learning scenarios based on both behaviourist and constructivist approaches.
PALO is based on a reference framework to design educative content based in levels of abstraction. Each level identifies a certain group of related components or elements of a learning resource.
The language allows defining teaching strategies by mean of the definition of specific DTD's called instructional templates. Such templates are a general type of PALO document that specially suits for a given instructional or teaching purpose.

7.4 OPEN UNIVERSITY OF NEDERLANDS' EDUCATION MODELLING LANGUAGE (EML)

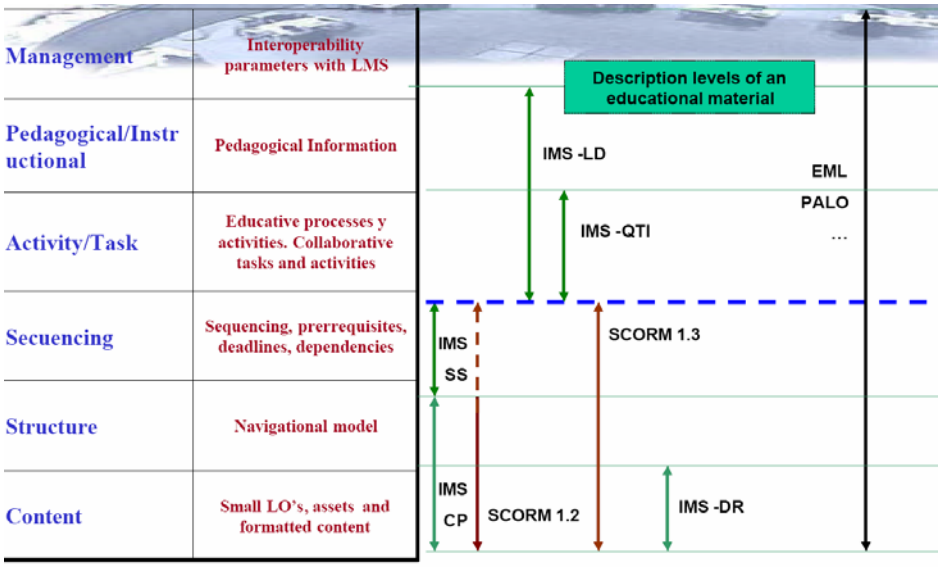
The work carried out by the Open University of the Netherlands (OUNL) on educational modeling comes from an R&D project funded by the Dutch national government through their structural funds for universities. The R&D work on learning technologies is paid from these funds with the objective of innovating education through the use of ICT.
OUNL research is academic and independent of any vendor or other commercial stakeholder. Besides work on Educational Modeling Language (EML), the OUNL's research and development activities in learning technologies include: competency based learning, new models of assessment, printing on demand, and others. The main outputs are: specifications, prototypes and publications.

To date no comprehensive notational system exists that allows one to codify units of study (e.g. courses, course components and study programs), in an integral fashion. EML is the first system to achieve precisely this. EML describes not just the content of a unit of study (texts, tasks, tests, assignments) but also the roles, relations, interactions and activities of students and teachers. The major EML implementation is in XML (eXtensible Mark-up Language), an internationally accepted meta-language for the structured description of documents and data. In EML, people engage in Activities, for which they may use:

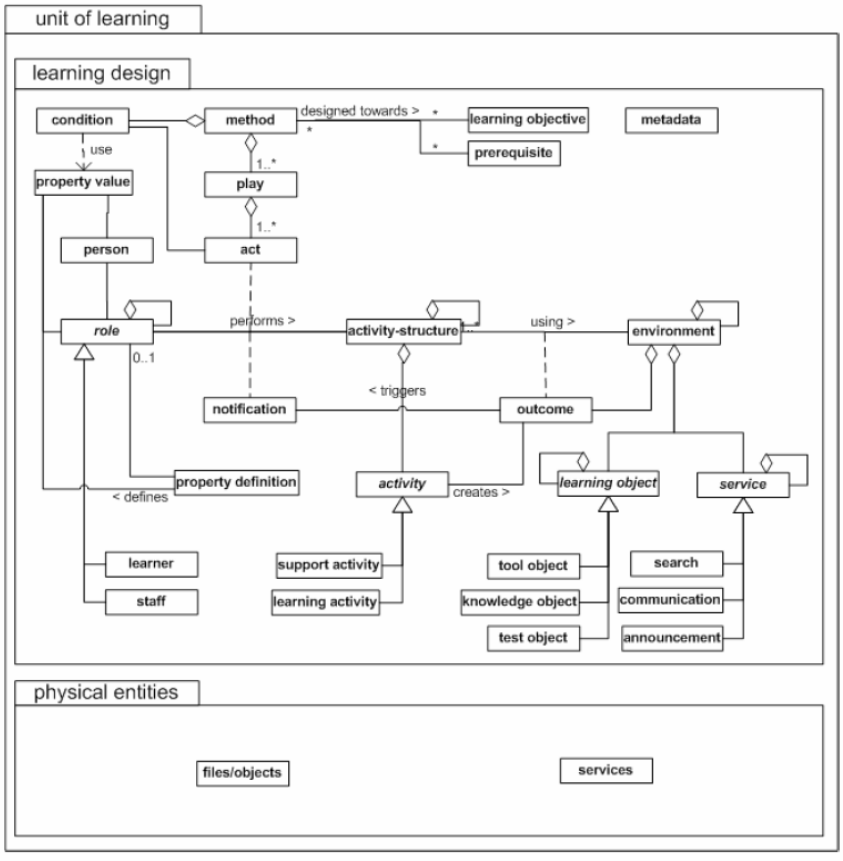
- People: one or many, learner or staff roles
- Activities: description, structured
- Resources: learning objects & services (chat, etc.)

EML allows modeling a variety of pedagogies for education. One may use EML to model for instance competence based pedagogy, problem based learning, performance support, self study packages or even traditional face-to-face teaching. Furthermore EML guarantees that investments in content will last for a long time; because of the uniformity of notation that EML brings, an instrument for comparative research on the effectiveness of educational structures emerges.

The following figure illustrates the role of EML in relation with other e-learning standards.



The following figure describes the information model of EML 1.1.



7.5 Comparison and Choice

Despite the several interesting ideas behind the PALO language, its diffusion didn't get beyond a restricted community and substantially did not catch up as a major learning model language. PALO doesn't support generic model bindings, while EML does. For each instructional template PALO defines its own schema for a subset of PALO elements for a certain type of pedagogical functionality. Furthermore, EML has been selected as

the basis for the IMS Learning Design specification, and as this, would represent a better investment over other modeling languages, like PALO.
As regards extensibility, PALO is open in the sense that it can easily be extended each time there is a need for a certain pedagogical functionality. However, that the scope of the current version is restricted to a small selection of extension features.
As regards the support of other e-learning standards, while IMS Learning Design (and EML) supports LOM, IMS Content Packaging, IMS Simple Sequencing and others, PALO supports only Dublin Core and some other file formats like LaTex source files and HTML.
Concluding, the adoption of IMS Learning Design Level A in the Diogene Project should be preferred over PALO or other competing specifications. Anyway, despite the massive impact that the adoption of IMS Learning Design Level A will have on the existing Diogene project it is highly advisable that its adoption in Diogene would be scheduled carefully over time for avoiding troublesome reverse engineering issues.

8. STANDARDS FOR LMS INTEROPERABILITY

This section describes the various standards for LMS interoperability.

8.1 ADL SCORM RUNTIME ENVIRONMENT

In discussing the runtime environment it is convenient to begin with a simple case e.g. a lesson represented as a single SCO, how this is launched and how it interacts with the LMS. The objective of this specification is to make learning resources to be reusable and interoperable across multiple Learning Management Systems (LMS). Moreover, the launch mechanism defines a common way for LMSs to start Web-based learning resources. Such a mechanism defines the procedures and responsibilities for the establishment of communication between the delivered learning resource and the LMS. The communication protocols are standardized through the use of a common API. The API is the communication mechanism for informing the LMS of the state of the learning resource (e.g., initialized, finished or in an error condition), and is used for getting and setting data (e.g., score, time limits, etc.) between the LMS and the Sharable Content Object (SCO).
A Data Model is a standard set of data elements used to define the information being communicated, such as, the status of the learning resource. In its simplest form, the data model defines elements that both the LMS and SCO are expected to "know" about. The LMS maintains the state of required data elements across sessions, and the learning content must utilize only these predefined data elements if reuse across multiple systems is to occur.

This specification provides a common launch scheme that in turn enables consistency of learning resource delivery behavior across LMSs; the two SCORM Content Model components that can be launched by an LMS are Assets and SCOs. There are different launching requirements depending on the type of learning resource being launched. The procedures and responsibilities for the establishment of communication between the delivered learning resource and the LMS vary depending on the type of SCORM learning resource being launched. For learning resources that represent Assets, the SCORM launch model only requires that an LMS launch the Asset using the HTTP protocol. Instead, for SCOs the SCORM launch model requires that an LMS only launch one SCO at a time and that only one SCO is active at a time. The launch model also requires that only LMSs may launch SCOs. SCOs may not launch other SCOs. The LMS must launch the SCO in a browser window that is a child window or a child frame of the LMS window that exposes the API Adapter as a Document Object Model (DOM) Object. The API Adapter must be provided by the LMS. It is the responsibility of the SCO to recursively search the parent and/or opener window hierarchy until the API Adapter is found. Once the API Adapter has been found the SCO may initiate communication with the LMS.

RUNTIME DATA MODEL AND API

The Runtime Environment consists of two other major components, Runtime metadata, and Runtime commands. These describe how and what sort of information is communicated between a SCO and an LMS. The data model and API are a subset of the AICC interoperability guidelines, and only a relatively small number of the commands and metadata are mandatory.

The runtime metadata consists of 49 items organized into 6 categories as follows:

- cmi.core items (15 items): 12 of these items are the SCORM mandatory items. These items are used to capture the learning state of the SCO in the event that the learner leaves and then returns to the SCO. Every SCORM-compliant LMS is required to maintain a database that can be used to store and retrive learner data for the 12 mandatory items.
- cmi.objectives (8 optional items): Used for describing and tracking each of the learning objectives associated with a SCO.
- cmi.student_preference (5 optional items): Used to describing the interface preferences for an individual student, such as language, audio volume, etc.
- cmi.student_data (4 optional items): Used to tracking a learners progress and any time limits associated with the SCO.
- cmi.interactions (13 optional items): Used for describing and tracking a student's responses to a quiz (interactions).
- communications(4 optional items): Four items used for SCO-to-SCO and SCO-to-LMS communications, e.g. cmi.suspend_data, cmi.launch_data, cmi.comments, cmi.comments_from_lms.

There are only 8 run-time commands (grouped in execution state commands, data transfer commands, or SCO state commands); usually most SCORM-compliant courses will not use all 8 run-time commands. However, even if a course is not designed to communicate with the LMS, it must meet the following minimum run-time requirements:

- LMSInitialize must be called before using any other run-time command.
- Every SCO must call LMSInitialize at the start of the SCO to signal to the LMS that the SCO is initialized and ready to communicate, if needed, with the LMS.
- LMSFinish must be called when the SCO has finished communicating with the LMS.

A SCO is not required to use any of the other 6 run-time commands. That is, if there is no need to communicate with the LMS, the SCO does not need to use the data transfer commands. Likewise, if there is no need to trap or investigate run-time errors, the SCO does not need to use the SCO state commands. The only commands the SCO is required to use are the LMSInitialize and LMSFinish commands. However, it is important to understand the LMSInitialize does not direct the LMS to load and run the SCO. In fact, the SCO must have already been loaded before LMSInitialize is issued. Additionally, LMSFinish must be issued to the LMS when the SCO is completed and before any other SCO is loaded by the LMS.

To recap, in practice the current SCORM run-time environment is implemented through the availability of 8 special commands, which are normally communicated to the LMS using javascript embedded within course webpages.

8.2 AICC CMI GUIDELINES FOR INTEROPERABILITY

In the following, the term Assignable Unit (AU) defines a module of computer based learning content (or CBT) that can be launched and tracked by a CMI system (the smallest logical unit of learning content in a course).

All that data that may be communicated between the CMI and the AU is composed of data elements that may appear in one or more of the following bindings:

- File-Based.
  A text -file binding for use in LAN/CD-ROM based systems.
- HACP.
  An HTTP-based binding which may be used for Web implementations.
- API
  A JavaScript API binding which may also be used for Web implementations.

In general, a data element is used in the same manner across all bindings, but there are some important distinctions to be made for each binding:

1. Each binding has different rules for formatting data
2. Each binding also operates in a different environment with different transport mechanisms.
3. Some data elements may be specific only to a particular binding.

The data elements in this model are arranged hierarchically (in a “parent/child” relationship). Hierarchy levels are delimited by period (“.”) in the data element name.

Complex guidelines for interoperability are defined within this specification for every aspect of electronic learning material. Further details can be found at http://www.aicc.org/docs/tech/cmi001v4.pdf.

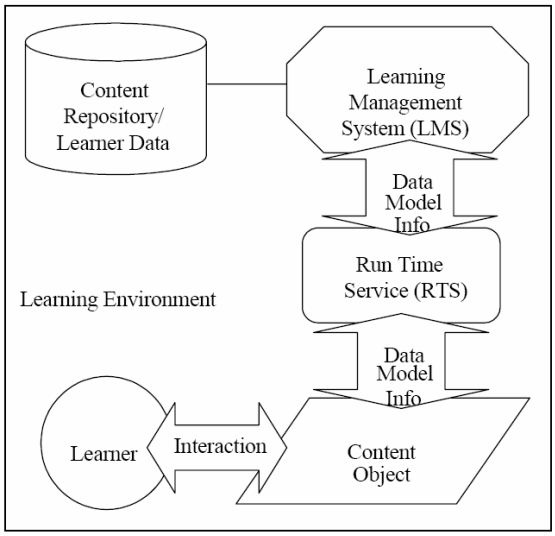
8.3 IEEE DATA MODEL AND ECMA SCRIPT API FOR CONTENT TO LMS COMMUNICATION

This Standard describes an ECMAScript application programming interface (API) for content-to- runtime-services communication. It is based on a current industry practice called "CMI—Computer Managed Instruction."
This API enables the communication of information between content and a runtime service (RTS) typically provided by a learning management system (LMS) via common API services using the ECMAScript language. The purpose of this Standard is to build consensus around, resolve ambiguities, and correct defects in existing specifications for an ECMAScript API for exchanging data between learning-related content and an LMS.

A possible scenario of use of this specification is as follows. An RTS implements an API in the content object's environment. The implementer incorporates in the content object the capability to discover and communicate with an API implementation. An LMS or the front-end to a content repository (local or remote) provides an RTS for the learner. The RTS either delivers a content object to the learner and starts it, or launches a URI to initiate the content object. The sequence of operations for this scenario is reported below.
The RTS initiates the launch of a content object. As the content object starts up, it searches for the API implementation. After verifying that the API implementation is accessible in the content object's environment, the content object initializes a communication session through the instance that has been located.
All subsequent communication is part of this communication session until it is ended. The content object may request data through the API implementation. Through the API implementation, the RTS returns the requested data or a message identifying an error condition. While running, content may send or set data-model data elements for storage across communication sessions. The RTS may use data elements or other data in reports on a learner's status with that content. The content object may elicit a more detailed error message.

The content object may continue communicating in this fashion, requesting and sending data until a learner finishes a content object, a learner terminates the communication session before finishing, or the communication session is abnormally terminated (e.g., loss of power, system crash). In the first two cases, the content object tells the API implementation that it is closing the communication session. In the last case, the RTS will not receive a signal through the API implementation.

As regards the data model, the conceptual data model diagram is reported below.



The Conceptual Model Diagram.

Further details can be found at: http://ltsc.ieee.org/wg11/materials.html.

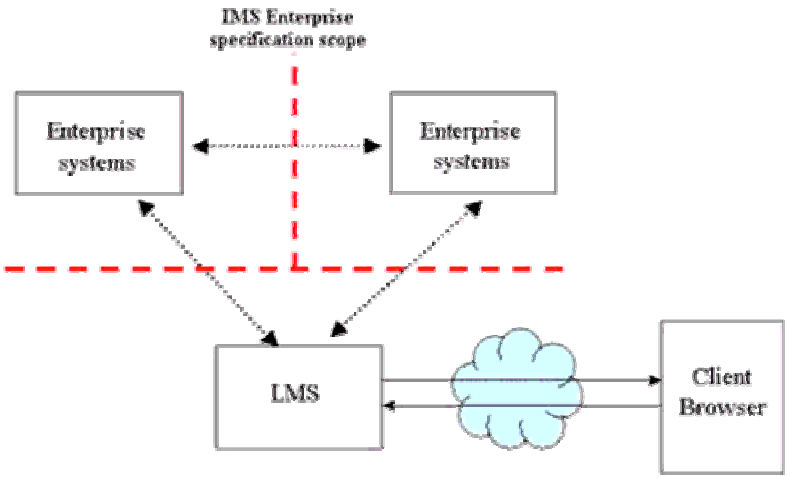
8.4 IMS ENTERPRISE SPECIFICATION AND ENTERPRISE SERVICES

The scope of information included in this version of the specification is intended to support interoperability between Learning Management Systems (LMS) and the following classes of Enterprise Systems:

- Human Resource Systems track skills and competencies and define eligibility for training programs;
- Student Administration Systems support the functions of course catalog management, class scheduling, academic program registration, class enrollment, attendance tracking, grade book functions, grading, and many other education functions;
- Training Administration Systems support course administration, course enrollment, and course completion functions for work force training;
- Library Management Systems track library patrons, manage collections of physical and electronic learning objects, and manage and track access to these materials.

The scope of the IMS Enterprise Specification is focused on defining interoperability between systems residing within the same enterprise or organization. Data exchange may be possible between separate enterprises, but the documents comprising the IMS Enterprise Specification are not targeted at solving the issues of data integrity, communication, overall security, and others inherent when investigating cross-enterprise data exchange.

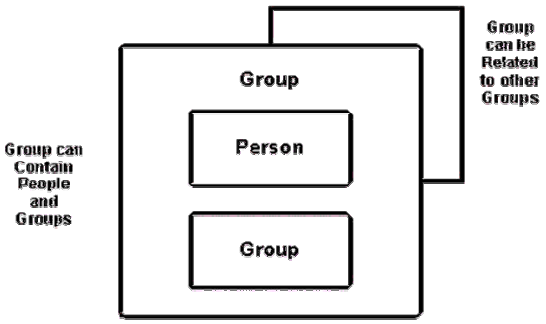
The IMS Enterprise Information Model is designed to support interoperability of the following four business process components, which typically require interaction between Learning Management systems and these types of Enterprise systems. The basic architectural model for the Enterprise V1.1 Specification is shown in the Figure below.



The basic Enterprise system architectural model.

In this architecture the scope of the IMS Enterprise Specification is shown as the dotted line. The scope of the interoperability is the data model of the objects being exchanged and not the associated behavioral model or the required communications infrastructure.

A schematic representation of the core data objects exchanged using the IMS Enterprise Specification is given in the following Figure.



The principal Enterprise data objects

The core objects are:

- Person - the individuals who are undertaking some form of study and/or group related activity e.g., they are members of a particular course. The personal structure only contains information that is used to identify the individual and that is needed in learning systems. It is not designed to be a data repository for all of the personal information about an individual;
- Group - a collection of objects related to learning activities or individuals. The group is a generic container used to define any set of related activities e.g., a tutor group, a class, a curriculum, etc. There is no restriction on how the Group and sub-group structures can be used with respect to containing other groups, persons, etc.;
- Group Membership - the membership structure is used to define the members of a Group. A member can be a Person or another Group. A Group or Person can be a member of any number of groups. The Group and the member are identified using their “sourcedids”.

An Enterprise XML instance is designed to contain any number of Person, Group and Membership structures but the order in which these can occur are limited to all of the Person objects, followed by all of the Group objects followed by all of the Membership objects.

Other details are available at: http://www.imsglobal.org/enterprise/entv1p1/imsent_infov1p1.html.

8.5 COMPARISON AND CHOICE

Given the requisites of the Diogene project and its adoption of the SCORM specification, the compliance with the SCORM runtime environment is an essential prerequisite for complete compliance with SCORM.

As regards the other specifications discussed in this section, their adoption in the Diogene project can be seen as an interesting extension, but not as important and necessary as the conformance with the SCORM runtime environment model.

9. STANDARDS FOR LEARNING OBJECT REPOSITORIES

This section describes the currently standards for enabling learning object repositories.

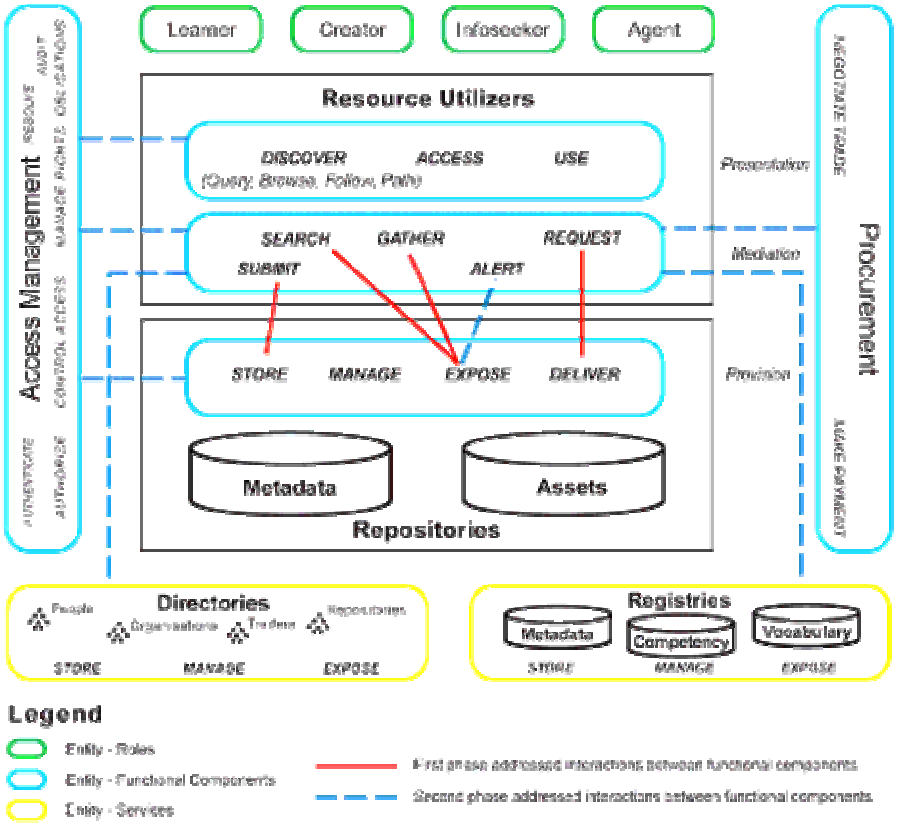
9.1 IMS DIGITAL REPOSITORIES

Provides recommendations for the interoperation of metadata indexed collections of resources accessible via a network without prior knowledge of their structure

The diagram in the Figure below depicts the DRI functional architecture. The interactions are shown by the red (solid) lines. The diagram maps out three entity types that define the space where e-learning, digital repositories, and Information Services interact, and which provide a context for exploration of the problem space.

The three entities are:

- Roles (e.g., Learner, Creator, Infoseeker, Agent)
- Functional Components for Resource Utilizers, Repositories, Access Management, and Procurement Services
- Services, such as Registries and Directories



The DRI functional architecture

The solid red lines, between a number of functions between Resource Utilizers and Repositories, indicate the interactions between core functional components that support interoperability, including:

- SEARCH, GATHER, (ALERT)/EXPOSE
- REQUEST/DELIVER
- SUBMIT/STORE
- DELIVER /STORE between two repositories

More details available at: http://www.imsproject.org/digitalrepositories/index.cfm

9.2 EDUTELLA

Peer-to-peer (P2P) networks are a relatively new method of sharing information across distributed, heterogeneous networks, without the requirement of central point of control. The Edutella project [53] was begun in response to the fact that information in P2P networks is no longer organised in navigable hypertext-like structures, but is stored on numerous peers waiting to be queried. We need to know what we want to retrieve and which peer is able to provide the resources in order to submit a query [54]. In some cases, especially concerning the exchange of educational resources, queries are complex, building on standards like IEEE-LOM/IMS metadata with multiple metadata entries.
To complicate matters further, concentrating on domain specific formats has meant that current P2P

implementations are fragmenting into niche markets rather than developing unifying mechanisms for future P2P applications. There is, in fact, a danger that unifying interfaces and protocols introduced by the WWW will become lost in the forthcoming P2P arena.
The Edutella project addresses these shortcomings of current P2P applications by building on the W3C metadata standard RDF. The project is a multi-staged effort to scope, specify, architect and implement an RDF-based metadata infrastructure for P2P-networks. To enable interoperability and reusability of educational resources, the infrastructure must be flexible enough to accommodate complex and varying metadata sets, and avoid creating another special purpose application suitable only for a specific application area which is outdated as soon as metadata requirements and definitions change.
The aim is to provide the metadata services required to enable interoperability between heterogeneous JXTA (Juxtapose) applications. The first application will focus on a P2P network for the exchange of educational resources (using schemas like IEEE LOM, IMS, and ADL SCORM to describe course materials).
JXTA is a set of XML-based protocols covering typical P2P functionalities. It provides a Java binding offering a layered approach for creating P2P applications. JXTA enables remote service access and provides additional P2P protocols and services, such as peer discovery, peer groups, peer pipes, and peer monitors.
In a typical P2P-based scenario in which learning resources are being exchanged, universities have two roles:

· Content provider. They do not lose control over their learning resources but still provide them for use within the network.
· Content consumer. Teachers and students benefit from having access not only to a local repository, but also to a whole network, using queries over the metadata distributed within the network to retrieve required resources.

Edutella connects highly heterogeneous peers, each making its metadata information available as a set of RDF statements. By implementing a set of Edutella services, the distributed nature of the individual RDF peers connected to the network can be made completely transparent. Edutella services include a query service, replication, annotation and mapping:

· Query service. The query service is intended to be a standardized query exchange mechanism for RDF metadata stored in distributed RDF repositories and serves as both a query interface for individual RDF repositories located at peers, and as a query interface for distributed queries spanning multiple RDF repositories.
· Replication service. This service complements local storage by replicating data in additional peers to achieve data persistence and availability while maintaining data integrity and consistency. Replication of metadata is the initial focus.
· Mapping, mediation and clustering service. While groups of peers will usually agree on using a common schema (e.g. SCORM or IMS/LOM for educational resources), extensions or variations may be needed in some locations. The purpose of the Edutella Mapping service is to manage mappings between different schemata and use them to translate queries over one schema to queries over another schema. Mapping services will also provide interoperation between RDF- and XML-based repositories. Mediation services actively mediate access between different services, while clustering services use semantic information to set up semantic routing and semantic clusters.
· Annotation service. In order to be applicable in a wide range of application scenarios, the annotation tool must be independent from any particular domain, and support a wide range of semantic definitions.

9.3 COMPARISON AND CHOICE

While Edutella appears as a not-so-widespread way of storing educational material, IMS Digital repositories is an interesting candidate for the adoption into the Diogene project. In fact, thanks to the technically simpler and easier-to-adopt model of the IMS Digital Repositories specification, the Diogene project can implement it in a relatively effortlessly way.

10. OTHER STANDARDS FOR E-LEARNING

In this section are discussed other standards related to e-learning.

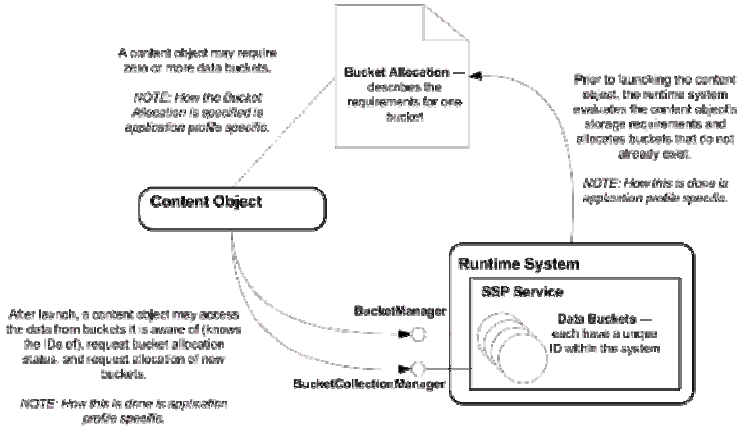
10.1 IMS SHAREABLE STATE PERSISTENCE

The Shareable State Persistence (SSP) specification describes an extension to e-learning runtime systems (e.g.,

SCORM, but not limited to) that enables the storage of and shared access to state information between content objects.
The capability of having a method for a content object to store (arbitrarily complex) state information in the runtime system that can later be retrieved by itself or by another content object, is crucial to the persistence of the sometimes complex state information that is generated by a variety of interactive content (e.g., simulations) and, without this standard would be stored and retrieved in proprietary formats and through proprietary methods. Information is stored in Buckets; data units persisted locally on client machines, similarly to Web browsers' Cookies.

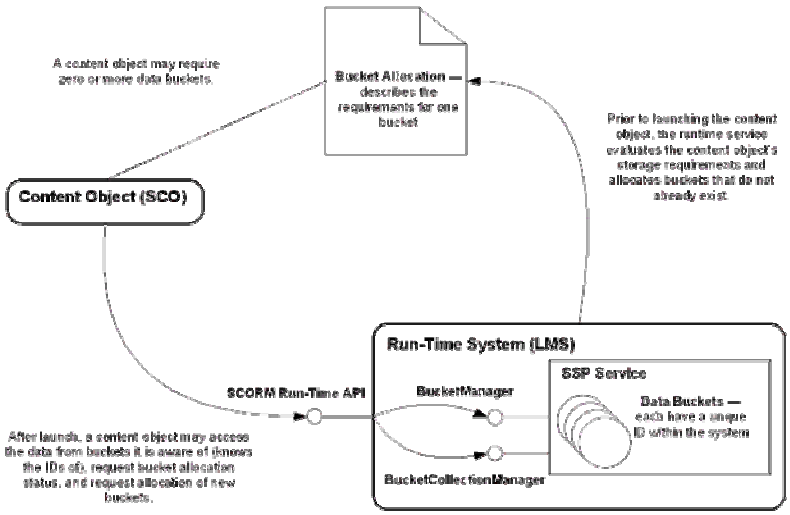
The Shareable State Persistence specification does not include a method to discover bucket identifiers or specific information about allocated buckets; however, content objects created by a vendor or community of practice can use a predefined identifier for a bucket that would contain information about other buckets.

The SSP general conceptual model is shown in the following figure.



This specification intends to address two fundamental technical requirements: enabling a content object to declare its storage needs and providing an interface for a content object to access that storage. A content object may require space to store its state data or it may require access to a known (through ID) set of state data created by another content object. A content object will declare these requirements by defining a set of Bucket Allocations either prior to or after launch. The runtime system, prior to launching a content object, will attempt to ensure storage space is made available to the content object based on that object's defined Bucket Allocations. After launch, a content object will access its storage space through well-defined interfaces.

The previous diagram (and the related general model) can be further specialized for the SCORM model, as follows:



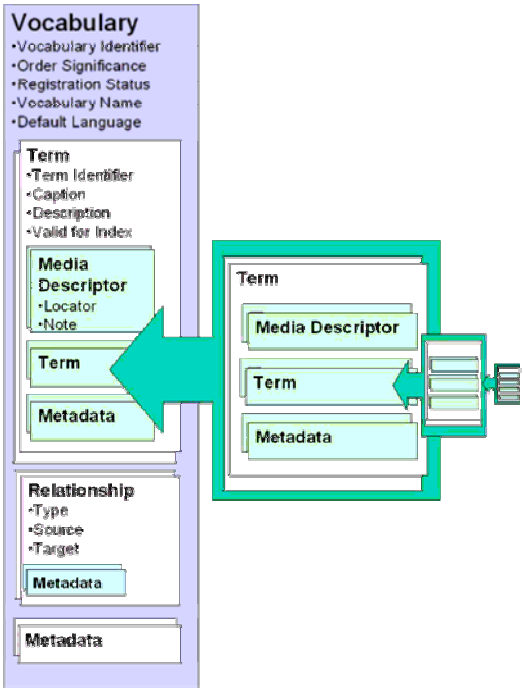
The SCORM Run-Time Environment (SCORM RTE) defines an interoperable mechanism for SCOs to communication information to the LMS -the SCORM Run-Time API. It is assumed that once a SCO is launched, it can initiate communication and then exchange information with an LMS. All communication across the API is initiated by the SCO. In other words, the communication is always initiated in one direction, from the SCO to the LMS. The LMS cannot invoke any functions defined by the SCO; there are currently no supported mechanisms for LMSs to initiate calls to functions implemented by a SCO.

The documentation is available at: http://www.imsglobal.org/ssp/index.cfm

10.2 IMS VOCABULARY DEFINITION EXCHANGE (VDEX)

This standard defines a grammar for the exchange of simple machine-readable lists of values, or terms, together with information that may aid a human being in understanding the meaning or applicability of the various terms, such collections often denoted "vocabularies".
Specifically, VDEX defines a grammar for the exchange of simple machine-readable lists of values, or terms, together with information that may aid a human being in understanding the meaning or applicability of the various terms. VDEX may be used to express valid data for use in instances of IEEE LOM, IMS Metadata, IMS Learner Information Package and ADL SCORM, etc, for example. In these cases, the terms are often not human language words or phrases but more abstract tokens. VDEX can also express strictly hierarchical schemes in a compact manner while allowing for more loose networks of relationship to be expressed if required.

The following figure defines the overall structure of the information model for the VDEX specification. Terms are intended to be units that may be repeated and nested recursively.



More details are available at: http://www.imsglobal.org/vdex/index.cfm

10.3 IEEE DIGITAL RIGHTS EXPRESSION LANGUAGE

This specification focuses on the study of existing specifications that may be considered to be rights expression languages, more specifically the DREL that are most general and that are getting the most attention in industries related to learning, education and training.
Some examples of such DREL's exist in:

- ODRL – Open Digital Rights Language
- MPEG REL – Extensible Rights Markup Language
- OeB – Open E-book
- IEEE Std 1420.1b-1999 – IEEE Standard for Information Technology – Software Reuse, IP Rights Framework

In addition, the DREL subcommittee is surveying a number of the initiatives that are developing or proposing standards for digital rights management. This can be used as a good source of technical documentation and references to the LTSC-DREL WG. The initiatives chosen here are either globally significant or significant within the learning technology community:

- MPEG-21 (Motion Picture Experts Group – 21)
- OASIS
- CEN/ISSS Workshop on Learning Technology
- IMS Global Learning Consortium
- The Open Knowledge Initiative
- Indecs2rdd Rights Description Dictionary
- ONIX (Online Information Exchange) Metadata Specification
- OMA (Open Mobile Alliance) Rights Expression Language

Anyway, the DREL initiative seem controversial to many in the e-learning standards community. In fact, many of the ideas and concepts behind the rights, roles, constraints and workflow or sequencing representations present in IEEE DREL are already present in learning technology systems and associated standards.

Consider the following simple example of a naive implementation of digital right management (expressed using DREL) in a learning content delivery system, like for example a standard Learning Management system.

- A learning designer has specified a set of limits and sequences on content, e.g., (1) the learner must view a lecture (a content object) and then take an assessment (a different content object), (2) the learner may review the lecture an unlimited number of times, (3) the learner is limited in how many times he may take the assessment. These are expressed in some learning experience language.
- There is a trading agreement with the content author that specifies the number of times the lecture and assessment may be accessed, expressed with DREL.
- The learner (already authenticated to the learning delivery system) requests one of the content objects.
- The learning delivery system evaluates the learning designer's conditions on use of the content. The conditions are met, and the learning delivery system requests the content resource from the content repository for display through a media player.
- The content management system applies the DREL conditions and determines that the request cannot be honored.

What happens next? Does the learning delivery system capture the failure and react or does the learner get an error directly from the media player? Have we confused the learner by exposing similar types of behavior limits at multiple points in the learning experience and through multiple components?

By splitting the management of the behavior into different system components and representations (like DREL and sequencing for example), we have introduced the complexity of coordinating a consistent behavior and user experience across the components.

This overlapping definitions potential problems are the reason why extra care should be used when deciding of implementing DREL. Further documentation is available at: http://ltsc.ieee.org/wg4/par1484-4.html

10.4 COMPARISON AND CHOICE

The standards discussed in this section are:

- IMS SHAREABLE STATE PERSISTENCE
  This standard would be employed as part of the runtime specification compliance.
- IMS VOCABULARY DEFINITION EXCHANGE
- IEEE DIGITAL RIGHTS EXPRESSION LANGUAGE

These standards are discussed in the previous sections and their adoption in Diogene would be advisable, because they are the only mechanism available for addressing the respective problems available as of today. As regards the IEEE DIGITAL RIGHTS EXPRESSION LANGUAGE, though, both because its still early development stage and the doubts expressed above, it is recommended to postpone its adoption within the Diogene Project.

11. STANDARDS FOR SEMANTIC REPRESENTATION OF SERVICES

This section illustrates the available standards for semantic representation of (Web) Services.

11.1 DARPA DAML-S / W3C OWL-S

The Semantic Web Services arm of the DAML program is developing an OWL-based Web Service Ontology, OWL-S (formerly DAML-S), as well as supporting tools and agent technology to enable automation of services on the Semantic Web.

With the term ``service'' is meant dynamic information such as allowing one agent to effect some action or

change in the world, such as the sale of a product or the control of a physical device. The Semantic Web should enable users to locate, select, employ, compose, and monitor Web-based services automatically. To make use of a Web service, a software agent needs a computer-interpretable description of the service, and the means by which it is accessed. An important goal for Semantic Web markup languages, then, is to establish a framework within which these descriptions are made and shared. Web sites should be able to employ a set of basic classes and properties for declaring and describing services, and the ontology structuring mechanisms of OWL provide the appropriate framework within which to do this.

OWL-S supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. OWL-S markup of Web services will facilitate the automation of Web service tasks including automated Web service discovery, execution, interoperation, composition and execution monitoring. Following the layered approach to markup language development, the current version of OWL-S builds on top of OWL.

OWL-S is a OWL-based Web service ontology, which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. OWL-S markup of Web services will facilitate the automation of Web service tasks, including automated Web service discovery, execution, composition and interoperation. Following the layered approach to markup language development, the current version of OWL-S builds on the Ontology Web Language (OWL) Candidate Recommendation produced by the Web-Ontology Working Group at the World Wide Web Consortium.

Owl-S has seen designed to perform at least the following four important task: Automatic Web service discovery, Automatic Web service invocation, Automatic Web service composition and interoperation and Automatic Web service execution monitoring.
These tasks are explained and discussed in the following paragraphs.

Automatic Web service discovery. Automatic Web service discovery involves the automatic location of Web services that provide a particular service and that adhere to requested constraints. For example, the user may want to find a service that sells airline tickets between two given cities and accepts a particular credit card. Currently, this task must be performed by a human who might use a search engine to find a service, read the Web page, and execute the service manually, to determine if it satisfies the constraints. With OWL-S markup of services, the information necessary for Web service discovery could be specified as computer-interpretable semantic markup at the service Web sites, and a service registry or ontology-enhanced search engine could be used to locate the services automatically. Alternatively, a server could proactively advertise itself in OWL-S with a service registry, also called middle agent [4,25,15], so that requesters can find it when they query the registry. Thus, OWL-S must provide declarative advertisements of service properties and capabilities that can be used for automatic service discovery.

Automatic Web service invocation. Automatic Web service invocation involves the automatic execution of an identified Web service by a computer program or agent. For example, the user could request the purchase, from a particular site, of an airline ticket on a particular flight. Currently, a user must go to the Web site offering that service, fill out a form, and click on a button to execute the service. Alternately, the user might send an HTTP request directly to the service with the appropriate parameters in HTML. In either case, a human in the loop is necessary. Execution of a Web service can be thought of as a collection of function calls. OWL-S markup of Web services provides a declarative, computer-interpretable API for executing these function calls. A software agent should be able to interpret the markup to understand what input is necessary to the service call, what information will be returned, and how to execute the service automatically. Thus, OWL-S must provide declarative APIs for Web services that are necessary for automated Web service execution.

Automatic Web service composition and interoperation. This task involves the automatic selection, composition, and interoperation of Web services to perform some task, given a high-level description of an objective. For example, the user may want to make all the travel arrangements for a trip to a conference. Currently, the user must select the Web services, specify the composition manually, and make sure that any software needed for the

interoperation is custom-created. With OWL-S markup of Web services, the information necessary to select and compose services will be encoded at the service Web sites. Software can be written to manipulate these representations, together with a specification of the objectives of the task, to achieve the task automatically. Thus, OWL-S must provide declarative specifications of the prerequisites and consequences of individual service use that are necessary for automatic service composition and interoperation.

Automatic Web service execution monitoring. Individual services and, even more, compositions of services, will often require some time to execute completely. A user may want to know during this period what the status of his or her or request is, or plans may have changed, thus requiring alterations in the actions the software agent takes. For example, a user may want to make sure that a hotel reservation has already been made. For these purposes, it would be good to have the ability to find out where in the process the request is and whether any unanticipated glitches have appeared. Thus, OWL-S should provide declarative descriptors for the state of execution of services.

For more details one can see: http://www.daml.org/services/owl-s

11.2 COMPARISON AND CHOICE

The introduction of OWL-S into Diogene is related to the architecture of the System and the importance of Web Services within it, impacting deeply on its structural architecture. Following this advice the introduction of the OWL-S specification into the project should be carefully considered, on a tight cost/benefit approach.

12. CONCLUSIONS

This survey aimed at discussing the main standard involved in the Diogene project and those available outside the Project. In this document we explored the main currently available e-learning standards, covering also knowledge representation format of interest to e-learning.

The findings are reported for convenience in the following recapping table.

Category	Best Standard	Standard Currently used in Diogene	Update Priority Status
METADATA AND TEST REPRESENTATION	IMS Metadata and QTILite	IMS Metadata 1.2.2 and QTI Lite	-
COURSE REPRESENTATION	IMS Content Packaging	IMS Content Packaging 1.1.2	-
LEARNER MODELLING	IMS LIP	IMS LIP 1.0	-
ONTOLOGY MODELLING	OWL Lite	SHOE, DAML+OIL and OWL Lite	-
COMPETENCIES MODELLING	LTSC Competency Definitions	-	Medium

LEARNING ACTIVITIES REPRESENTATION	IMS Learning Design Level A	-	High
LMS INTEROPERABILITY	SCORM Runtime Environment	-	Medium
LEARNING OBJECT REPOSITORIES	IMS Digital Repositories	-	Low
OTHER STANDARDS FOR E-LEARNING	IMS SHAREABLE STATE PERSISTENCE[3] and IMS VOCABULARY DEFINITION EXCHANGE	-	Low
SEMANTIC REPRESENTATION OF SERVICES	OWL-S	-	Medium

The previous table shows the best standards (Second Column) among the categories (First Column) mentioned in the previous sections. The Third Column takes into account their current adoption state in the Diogene Project (for further details refer to the concluding discussions in each section of this document). Finally, the Fourth Column shows the priority value we have associated to those standards needed to be incorporated in Diogene. It is worth noting that we have recently updated the Diogene tools involved in the Ontology management (such as the KMS) in order to be able to export and import OWL-Lite formats. This has been done during the last phases of the project since we have considered the necessity to be compatible with OWL-Lite with a very high priority status.

Concluding, our aim in this work was to provide a document focused on monitoring the most important e-learning standards, to describe the tools available for each of them and providing a critical discussion about the pros and cons of each standard.

[3] This standard would be employed as part of the runtime specification compliance.