Preparing SCORM for the Semantic Web

Lora Aroyo[1,2], Stanislav Pokraev[3], and Rogier Brussee[3]

[1] Department of Computer Science, University of Twente
P.O. BOX 217, 7500 AE Enschede, The Netherlands
[2] Department of Mathematics and Computer Science,
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
l.m.aroyo@tue.nl
[3] Telematica Institute,
P.O. BOX 589, 7500 AN Enschede, The Netherlands
{stanislav.pokraev,rogier.brussee}@telin.nl

Abstract. In this paper we argue that the effort within the context of Semantic Web research, such as RDF and DAML-S. will allow for better knowledge representation and engineering of educational systems and easier integration of e-learning with other business processes. We also argue that existing educational standards, such as SCORM and LOM could be mapped to those technologies, providing for more efficient automation of processes like educational resource annotation, and intelligent accessibility management. In this way we can use a successful combination of the technical advances outside of educational context and the existing educational standards, and allow for easier interoperability. To illustrate these issues and a solution approach we present the OntoAIMS educational environment.

1 Introduction

The learning technology community is quickly adopting many of the Web technologies (XML, RDF(S), streaming video, etc.) [Dev01]. Simultaneously, the educational technology standardisation is moving forward at rapid pace, with the IMS [IMS] and the ADL [ADL] having become the specification consortia that are tracked by vendors, implementers and academia. Both bring important contributions with respect to management of educational resources. The emerging educational specification for learning content SCORM addresses semantic annotations, content aggregation and sequencing. However, SCORM [Dod02] has chosen its own XML formats and methodologies, thereby limiting the educational community to a restricted universe, and making it much more difficult to integrate e-learning with other business processes. For example, although SCORM was designed for the military, without special interfacing code, e-learning systems will not be able to tell military planners which soldiers have absolved a course for safe operation of a particular weapon system because the SCORM specific XML schema used to track course results, is almost certainly not used by the (many) military planning systems. It is thus recognised by some, including leaders of the SCORM development community, that there is a benefit in opening up to the larger (semantic) web

R. Meersman et al. (Eds.): CoopIS/DOA/ODBASE 2003, LNCS 2888, pp. 621–634, 2003.

community. By design, these technologies make it easier to integrate learning material with other material by avoiding cumbersome translations of existing vocabulary and semantic descriptions. Moreover, there is the hope of getting better tool support and using a more widely deployed infrastructure by tapping in the resources of a larger community.

Current web-based courseware (both static material and dynamic activities) is usually decomposed into independent modules, which are further combined in complete courses [Che98, Mur99]. This approach stimulates reuse and makes it easier to create variants of courses adapted to different purposes or audiences, up to the level of individual students. This also leads to the possibility that different students can take different learning paths through the content based on navigation requests and guided by their learning goals and current capabilities. Often various limitations are set, such as being given a fixed amount of time or being able to see the material only once e.g. for the answers to quizzes, or expensive commercial material. Therefore, in this area it has become an important technological problem to specify how individual learning units should be found and sequenced based on different conditions. Conditions can come from interaction of students such as queries, navigation requests, test results, or predefined conditions such as learning goals, previous experiences of students, and externally imposed limits. Such (semi-) automatic choices of learning material are at heart didactic choices, and a lot of effort has been put into finding didactic templates that formalise and encapsulate the underlying didactic choices in a simple model [IMS LD]. However, here, we will only consider effective specification of sequencing and annotation of material on a semantic level.

In this paper, we will discuss how the sequencing of the current SCORM standards can be mapped onto OWL [OWL] and DAML-S [DAM02]. Such a DAML-S translation can then be used to integrate SCORM based learning environments with other business processes or webservices that have a DAML-S description. We also refer to existing work on integrating the annotation part of SCORM in the larger Semantic Web framework. In order to exemplify some of the issues related to the use and the authoring of intelligent web-based courseware we have developed the system OntoAIMS [Aro01], which provides intelligent (adaptive) information support to both students and authors. Its instructional goal is to act as an adaptive educational portal for instructional material, services, learning activities and teaching content. We will illustrate the sequencing model of AIMS and it options for reuse and interoperability of content.

2 State-of-the-Art and Related Work

In this section we present a brief description of the current state-of-the-art within educational and relevant semantic web standards and technologies. We will point out the role of ontologies for the improvement of educational content management. We will base our discussion on the ADL SCORM, which incorporates many specifications of the IMS and the IEEE LOM [Nil02] educational metadata scheme. We also describe the DAML-S process specification language, which we propose to use for sequencing to improve interoperability and interchangeability of the educational services with other services.

2.1 Ontology Research, DAML, and DAML-S

Ontologies will play a crucial part in facilitating the sharing of information between communities, both of people and software agents. A number of representational formats have been proposed, e.g. RDF Schema [RDF], the Ontology Interchange Language [OIL], the DARPA Agent Markup Language [DAML], later in the form of DAML+OIL [DAM01]. The latter provides the basis for the ontology web language [OWL], currently proposed as a W3C standard for ontology and metadata representation. DAML+OIL exploits existing web standards such as XML, RDF and RDFS and adds the primitives of the object-oriented and frame-based systems, as well as the formal thoroughness of description logic. Its formal and rational basis provides powerful means for reasoning services.

The DAML-S coalition [DAM02] has developed the DAML+OIL ontology for Web Services, DAML-S, aiming at making web services computer-interpretable, to enable discovery of services, invocation of an identified service by an agent or other service and interoperation, i.e. breaking down interoperability barriers through semantics, composition of new services through automatic selection, composition and interoperation of existing services, verification of service properties and execution monitoring. In DAML+OIL, abstract categories of entities, events, etc. are defined in terms of classes and properties. DAML-S defines a set of classes and properties, specific to the description of services, within DAML+OIL.

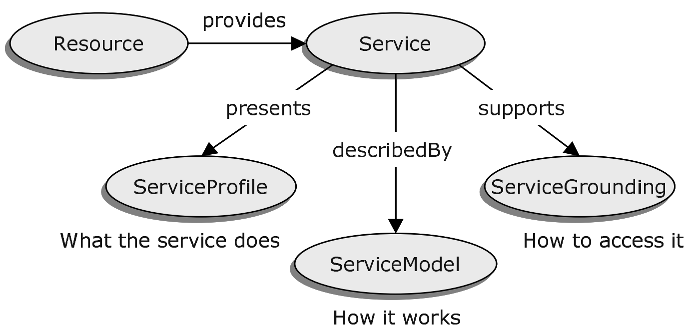

Fig. 1. Top level of the service ontology

Figure 1 resents the top-level service ontology of DAML-S. At the top level is the class *Service*, where all defined properties are very general. The idea is to provide a conceptual basis for structuring the services taxonomy, but it is expected that the taxonomy itself will be created according to functional and domain-specific needs. A service *presents* a *ServiceProfile.* This is a high level description of the service and its provider. A *ServiceProfile* describes the service in a human readable way, specifies the functionalities provided by the service and its functional attributes (e.g. requirements and capabilities). A *Service* is formally described by a *ServiceModel.* It facilitates automated services invocation, composition, interoperation and execution monitoring. The *ServiceModel* provides means for describing the data flow and the control flow in case of a composite service. Finally, a Service supports a *ServiceGrounding*, which is a specification of service access information. The

ServiceGrounding specifies the communication protocols, transport mechanisms, etc. Web services are web-accessible programs or devices. Their operation is described in terms of a process model, e.g. by means of control and data flows. The service model shows the possible steps (typically initiated by messages sent by the client) required to execute a service.

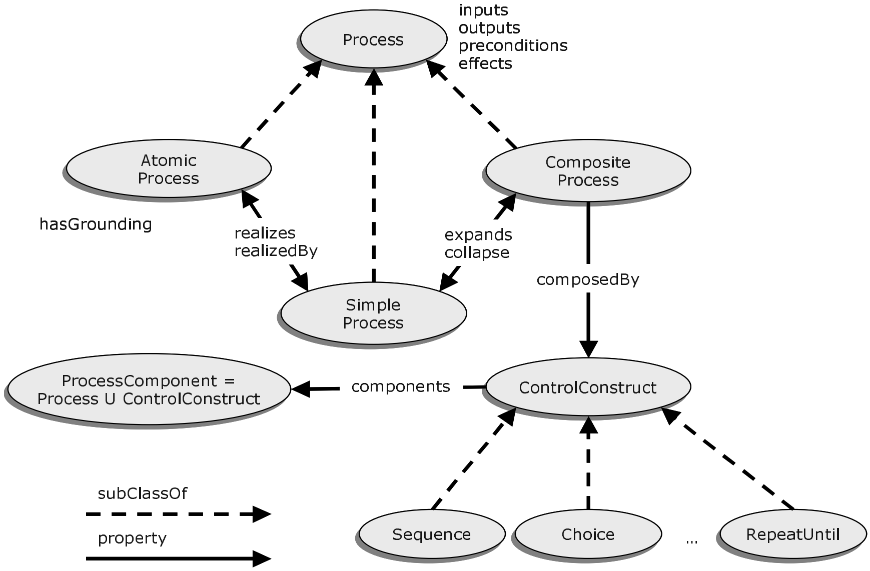

Fig. 2. Top level of the process ontology

The process model comprises of subclasses and properties of the *ProcessModel* class. The two chief components of the process model are the *Process Ontology* that describes a service in terms of its *inputs*, *outputs*, *preconditions*, *effects*, and, where appropriate, its *component subprocesses*; and the *Process Control Ontology* which describes each process in terms of its state, including initial activation, execution, and completion. The DAML-S process model is presented in Fig. 2.

2.2 Sharable Content Object Reference Model (SCORM)

The SCORM or Sharable Content Object Reference model is a collection of existing or further and newly developed specifications driven by the Advanced Distributed Learning (ADL) Initiative [ADL]. The ADL initiative is sponsored by the department of defence (DoD) and works in close cooperation with the industry forum IMS [IMS] and academia. The goal is to establish a new distributed learning environment that permits the interoperability of learning tools and course content on a global scale. Its roots in the DoD have produced specifications that work [SCO1.2] but which have been criticised for the rigid model they enforce on the creators and users of educational content. The main parts of the SCORM specification are the metadata model borrowed from the IEEE-LOM [LOM] the IMS content packaging model developed further together with the IMS [IMS CP], the IMS simple sequencing model co-developed with the IMS for SCORM 1.3 [SCO1.3] and a SCORM specific runtime and data model. The SCORM model assumes that content is build up out of small learning units, the SCO's (Sharable Content Objects)[1]. A SCO is intended to be a self-contained unit of learning. Courses are designed by creating a IMS packaging manifest document describing *organisations.* An organisation is comparable to a document tree structure, whose leaves consist of SCO's and whose nodes (items) represent the course internal structure. Each structuring level (manifest, organisation, item and SCO) can be annotated with metadata. The separation of content and organisation means that the same SCO can be reused in different organisation documents.

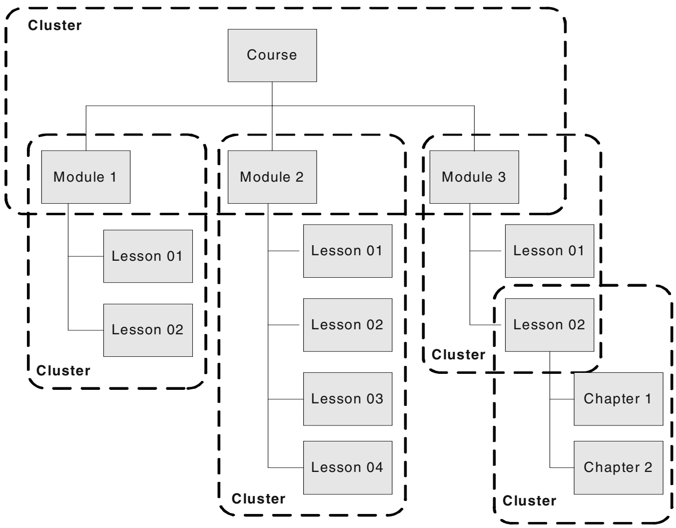

Fig. 3. An organisation document with clusters of activities © ADL

The other assumption of the SCORM model is that the only communication to the outside world is with a so-called earning management system (LMS) through a well-defined interface. In particular, it is the sole responsibility of the LMS to launch SCO's for presentation to the user, which means that the LMS is aware of the material the user has seen and of his or her responses. It also means that SCO's cannot by them selves launch other "related" SCO's which make then independent and therefore more reusable.

The IMS simple sequencing specification provides the model of learning activities organised as tree of clusters of more primitive activities. This tree of activities is mapped directly onto the items in the "document tree" of the organisation document (Fig. 3) but with a separate specification what the learning path through the tree is. By default, the learning path is a depth first walk through the tree, but one can specify an

---

[1] There also exist SCA's (Sharable Content Assets) as learning unit. The difference between SCO's and SCA's is that SCO's register themselves as being launched, but this will not be relevant on this level of detail.

action for each item depending on conditions. The actions allowed are either a move to a different leaning activity or a change of state of the learning management system. Conditions can either refer to the state of the learning management system or to a navigation request of the user. Only the simplest of sequencing controls are available: “choice” (with a variant called “choice exit”) and “if condition then action” with various conditions and actions being hardwired in the specification.

The possible action of changing the state of the learning management system is used for the specification of the progress of *objectives*. Conditioning on objectives is the primary mechanism to define more subtle behavior linking activities in different clusters. Objectives can be specified separately from the learning activity and can be either local to specific activities or global and involving several activities. The specification does not specify which objectives exist, but only standardizes on a scoring scheme, a syntax and a run time model.

2.3 Learning Objects Metadata (LOM)

The Learning Objects Metadata [LOM] standard has been developed specifically for the educational domain. It focuses on the minimal set of properties needed to allow learning objects to be found, evaluated, acquired, and used. LOM only provides metadata fields for specifying descriptive tokens related to areas such as security and commerce. The standard supports security, privacy, commerce, and evaluation, but does not concern itself with the implementation of these features. The LOM has been given an RDFS binding [Nil02] which makes the (IEEE specified) interoperability with the Dublin Core more apparent by simply using the Dublin Core RDFS vocabulary where appropriate.

3 Authoring in OntoAIMS

OntoAIMS (its processor AIMS [Aro01]) is designed on the one hand to support students in such as discovery-based learning and task-based search, browsing and retrieving of educational information. On the other hand OntoAIMS is also designed to support authors (instructors, experts, courseware designers) in the construction, description and conceptualisation of content items and in the generation and composition of the course sequence (structure) [Aro03]. In the context of the latter, the main goal of OntoAIMS is to facilitate the creation, editing, maintenance and reusability of teaching material and its efficient application within various courseware. This is achieved by applying ontology-driven course decomposition process within a web-based learning architecture building upon existing instructional models. OntoAIMS is based on the AIMS reference model specifying the strict separation between content, data and their application and defines three main application modules (e.g. domain, course and resources) and in this way allows authoring within three main system modules – *Domain Editor*, *Course Composition* and *Resources Management*, where the main goal is to construct a course sequence structure on the basis of already defined conceptual model of the subject domain and semantically annotated resources. The course composition offers a. wizard-like tool to support the author through the entire process of defining course sequence structure and to associate the related teaching material. OntoAIMS sequence model (based on SCORM) allows the author define the learning process and to organise all the course relevant metadata. In order to create the course structure the author should (1) select concepts from the domain model and assign them to course topics (components); (2) select a specific sequence of course topics realising the specified learning goals; (3) assign course activities for each topic; and (4) link educational resources to each course topic. Some concepts can already be linked to resources in the domain. The author is therefore guided by OntoAIMS to organise and annotate his educational materials and his intentions to create a specific course.

We will illustrate how these three modules are implemented within OntoAIMS architecture and how they are mapped to semantic web technologies. We will also illustrate how to use SCORM and DAML-S in order to facilitate the course sequencing process.

3.1 OntoAIMS Architectural Description

To describe the OntoAIMS architecture (Fig. 4) we introduce five roles to interact with the system: the authoring roles of the domain, resource and course authors, and the user roles of instructor and learner. Every user (designer, content expert, author, instructor or student) can play each of the roles in different stages in the process. One and the same user can also perform in all of the roles alone. Each role is defined with a set of authoring or learning activities.

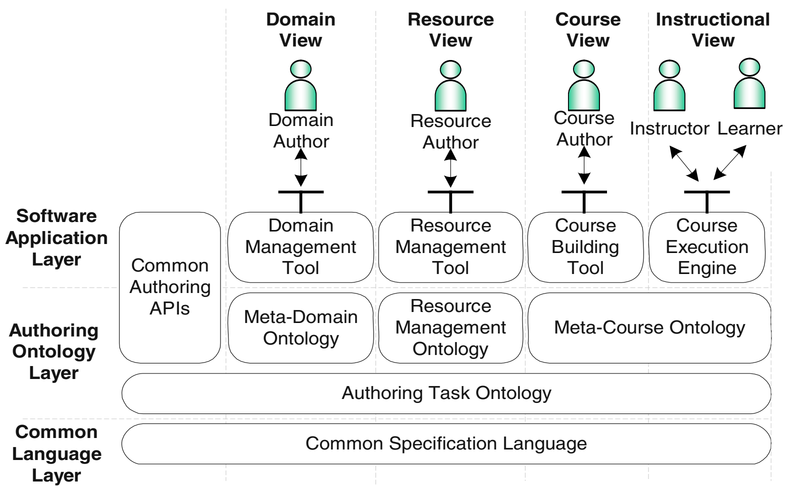

Fig. 4. Enterprise view on OntoAIMS

For the realisation of the *Common Specification Language* we choose OWL because of its openness and because it provides a large amount of flexibility for reasoning. This allows us to use semantic web tools and methodologies within the context of OntoAIMS. It also allows for the modularisation and semantic organisation of courseware authoring process and offers openness and interoperability of the authoring tools. Based on the results of organising functional knowledge in a task ontology [Fuk95, Miz96], we define the concepts and the semantics of the authoring process within the upper *Authoring Task Ontology* (*ATO*), where each of the phases (domain, course and resource views) define sub-sets of specific authoring concepts (e.g. meta-domain ontology, meta-course ontology and resource management ontology).

3.2 OntoAIMS Course Composition

A course in OntoAIMS is defined as a set of course components (*activities*) interconnected by composition (instructional) operators in order to achieve a coherent didactic sequence of learning activities and educational resources. In the following sections we will present the definitions of the composite operators, the learning activities and their mapping to DAML-S concepts.

3.3 Learning Activities

The learning activities (Fig. 5) capture learning-specific tasks within the context of a selected course and the formal definitions of these activities are presumed to support their interpretation. In analogy with the DAML-S process ontology (Fig. 2), we view the learning activity as an *atomic* or *composite* process, where the composite process is constructed from atomic or other composite processes by applying the composition operators, based on the DAML-S control constructs (Table 1). Each primitive learning activity $\mathcal{LA}$ is determined by a tuple

<Activity_ID, Input, Output, PreCond, Effect, Composite_Components>.

Thus it specifies some prerequisites (pre-conditions) and some input and produces some effect (post-conditions) and some output (Fig. 5), which are further used for the selection of the following activity, and it may have an internal structure.

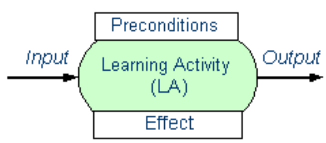

Fig. 5. Learning activity definition

Chains of atomic and composite learning activities (Fig. 6) are realised by applying the same composition operators. Figure 6 describes the sequencing process corresponding to the course clustering shown in Fig. 3.

The *primitive* (atomic activities) functional concepts are basic for the learning sequence process, and build OntoAIMS instructional vocabulary. They are generic for every educational process and are used to form *composite* semantic rich learning activities. This way we modularise all the learning activities within their specific context of use. They are also independent of the system's domain, the educational strategy and the educational goal. Examples of learning activities include: 'read', 'write', 'compute', 'exercise', 'proceed', etc.

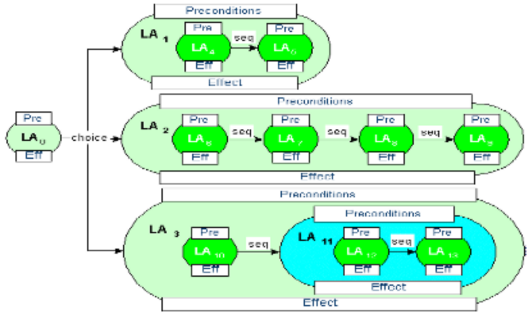

Fig. 6. Sequencing process of atomic and composite learning activities

3.4 Composition Operators

We define composition operators based on the DAML-S process ontology definition of control constructs. This will allow us to build composite activities and course components as sequences of both primitive and composite activities.

In Figure 6 we illustrate the composition process of learning activities by applying the composition operators. For example the learning activity $\mathcal{LA}_3$ is expressed as a sequence of two learning activities $\mathcal{LA}_{10}$ seq $\mathcal{LA}_{11}$, where $\mathcal{LA}_{11}$ is also a composite activity of two other learning activities also organized in a sequence $\mathcal{LA}_{12}$ seq $\mathcal{LA}_{13}$ On the other hand the main course activity in this example $\mathcal{LA}_0$ allows for applying the *choice* operator by offering a choice of three other activities $\mathcal{LA}_1$ or $\mathcal{LA}_2$ or $\mathcal{LA}_3$.

Table 1. Composition operators

DAML-S Control Constructs	Description
Sequence	Sequence defines a set of learning activities (processes) to be executed in order one after each other. The learning activities in this sequence can be either *atomic* (primitive), *simple* (clustered) or *composite* with the same or other *composition operators*
Choice	Choice is a composition operator defining a set of activities with two property states: (1) *chosen* and (2) *chooseFrom*. It allows a choice between alternative activities and afterwards executes the 'chosen' one(s). By this operator the following constructs are also allowed 'choose at least *n* activities from total *m*', 'choose exactly *n* activities from total *m*', 'choose at most *n* activities from total *m*'.
Conditional (***If Then Else***)	The conditional construct is an operator with *if Condition*, *then* and *else* properties. Its semantics is intended as 'Test If-condition; if True do Then, if False do Else' (class CONDITION is defined as a class of logical expressions.)

In the following section we illustrate the application of DAML-S for learning activities sequencing, based on the same sequencing example of Figure 6 and the course clustering of Figure 3.

4 Mapping SCORM Simple Sequencing onto DAML-S

Mapping the simple sequencing specification onto DAML-S is based on a separation between the processes corresponding to learning activities and the processes corresponding to actions performed by the learning management system to support the sequencing model. This separation is inherent to the SCORM model in that the flow of learning activities can only be changed by the state of the LMS and navigation requests of the user. Written in N3 notation [BL03] we have

```
scorm:LearningActivity rdfs:subClassOf daml-s:Process
scorm:LMSActivity      rdfs:subClassOf daml-s:Process
```

Learning activities can be either `daml-s:AtomicProcess` es or `daml-s:CompositeProcess` es. The DAML-S specification requires that an atomic process has a grounding in a web service specified in a WSDL document. However in this context, a grounding would be a specific learning object (SCO or SCA). Although a SCO is not (currently) a web service it has a well-defined interface, and is conceptually close to a realisation of a service. Note that it is easy to imagine "atomic" learning activities that are not grounded in this sense but that may still rightfully be called leaning activities e.g. discussions or traditional courses, and may result in some grading and a corresponding change of state of the LMS.

The internal structure of learning activities as clusters can be modelled by the DAML-S vocabularies. In addition, the simple sequencing specification implicitly contains standard conditions and state changes of the LMS like timeouts that can be given an owl/DAML-S binding. These can then be used in the specification of the learning process flow. The cluster structure of Fig.3 together with some imaginary expression of the way the different modules are sequenced is illustrated in Sec. 4.1

Given the distinction between learning and LMS activities, a mapping of the sequencing model to DAML-S is further a matter of identifying the relevant pre- and post conditions, input output and effect. For the purpose of the sequencing model a `scorm:LearningActivity` can have three sorts of input: a user navigation request, state from the learning management system, and parameters specified in the organisations document. Pre and post conditions are determined by the state of the LMS and by input parameters in the organisation document. The sequencing specification defines several such pre and post conditions. Of all the inputs, the navigation requests are the most problematic because they arrive asynchronously and the sequencing model takes timeouts into account. It is probably desirable to not consider the learner part of this process, because due to its inherent human complexity, and only consider input to the learning activity object. However it seems to be worthwhile to have the notion of a `scorm:theLearner` so that statements about the behaviour of "the learner" like passing or failing course modules can be made. Due to its special role in the SCORM specification we also propose to define an instance `scorm:LMS` of a `daml-s:process`. This process can then be used for low level specification of the SCORM run time model and in higher level constructs that can be used in course specifications For example in this way the navigation requests can be considered as a time stamped message that a learning activity can direct to the LMS process. Likewise the precise effect of a higher level `scorm:LMSActivity` can be specified in terms of this process.

4.1 Example of DAML-S Specification of a Complex Learning Path

Here we give an example of how the SCORM-based course clustering presented in Fig. 3 and the corresponding course sequencing in Fig. 6 can be translated into DAML-S specification of a complex learning path (we view clustering as DAML-S simple process, modules as composite and chapters as atomic). We use the N3 notation for the specification with daml-s denoting the DAML-S, and scorm denoting the SCORM namespace. A partial binding of SCORM concepts expressed in an OWL ontology used in the example is also included.

```
:Course a scorm:LearningActivity;
     daml-s:consistsOf     [a daml-s:Sequence;
                            daml-s: components
                                   (:Module1 :Module2 :Module3)
                           ].
:Module1 a scorm:LearningActivity;
     daml-s:consistsOf     [a daml-s:Choice;
                            daml-s:components
                                   (:Lesson01 :Lesson02)
                           ] .
:Module2 a scorm:LearningActivity;
     daml-s:consistsOf     [a daml-s:Unordered;
                            damls-s:components
                            (:Lesson01 :Lesson02 :Lesson03: Lesson04)
                           ].
:Module3 a scorm:LearningActivity
     daml-s:consistsOf  [a daml-s: If-Then-Else;
                         daml-s:ifCondition
                    {:Lesson02 scorm:objectiveMeasureGreater "0.75"};
                         daml-s:then
                             :Lesson01;
                         daml-s:else
                             :Lesson02
                        ].
:Lesson02 a scorm:LearningActivity
     daml-s:consistsOf [a daml:Sequence;
                        daml-s:components
                      ([a scorm:BeginTimelimit scorm:timeLimit "10M"]
                    :Chapter1
                    :Chapter2
                    [a scorm:EndTimeLimit]
                    [a scorm:RollUp;
                     daml-s:ifCondition
                           {scorm:theLearner scorm:satisfied
                    [scorm:activityObjective :Chapter1]};
                      daml-s:then {
                           scorm:theLearner scorm:satisfied
                              [scorm:activityObjective :Lesson02;
```

```
                                   scorm:objectiveMeasure "0.5"]
                                }
                           ]
                         [a scorm:RollUp;
                          daml-s:ifCondition
                             {scorm:theLearner scorm:satisfied :Chapter2}
                          daml-s:then {
                                scorm:theLearner scorm:satisfied
                                 [scorm:activityObjective :Lesson02;
                                 scorm:objectiveMeasure "1.0"]
                         }
                         )
                         ].
   :Chapter1 a scorm:LearningActivity.
   :Chapter2 a scorm:LearningActivity.
```

The following would be the start of a SCORM ontology that formalises and binds
the SCORM spec. Namespaces left out for conciseness

In this ontology Learners are not processes.

```
scorm:Learner a owl:Class.
scorm:theLearner a scorm:Learner.
```

Normalised scorings cf. SCORM v1.3, 7.1.1.3, maybe this is better done with a new
XML schema datatype.

```
scorm:ObjectiveNormalisedMeasure rdfs:subClassOf xsd:double,
            [a owl:Restriction;
             owl:onProperty xsd:maxInclusive;
                owl:toValue "1.0"],
            [a owl:Restriction;
             owl:onProperty xsd:minInclusive;
             owl:toValue "-1.0"].

scorm:objectiveMeasure a owl:DataTypeProperty;
     rdfs:range scorm:ObjectiveNormalisedMeasure.
scorm:objectiveMeasureGreater a owl:DataTypeProperty;
     rdfs:range scorm:ObjectiveNormalisedMeasure.
```

Many things can be objectives so objectives are orthogonal to learning activities

```
scorm:Objective a owl:Class.
scorm:activityObjective a rdf:Property;
     rdfs:domain scorm:Objective.
scorm:satisfied a rdf:Property;
     rdf:domain scorm:Learner;
     rdf:range scorm:Objective.

scorm:timeLimit a owl:DataTypeProperty;
     rdfs:range xsd:duration.
```

Learning activities are considered processes and can be handled as such

```
scorm:LearningActivity rdfs:subClassOf daml-s:Process.
scorm:LMSActivity rdfs:subClassOf daml-s:Process.

scorm:theLMS a daml-s:Process.
```

Starting and stopping "the stopwatch" is a very explicit LMSactivity
a process ontology could specify the precise description of

`scorm:BeginTimeLimit` in terms of interaction with the `scorm:theLMS`
`process`

```
scorm:BeginTimeLimit a scorm:LMSActivty, daml-s:AtomicProperty.
scorm:EndTimeLimit a scorm:LMSActivity, daml-s:AtomicProperty
```

The RollUp process gathers the "conclusion" of the learning activity, cf. SCORM v.
1.3 7.1.2. Many other rollup actions can be defined besides the ones used in the
example

```
scorm:RollUp a scorm:LMSActivity, daml-s:If-Then-Else.
```

5 Conclusions

We argued that the adoption of semantic web technologies is useful for learning technologies due to the expected broader adoption and easier interoperability with infrastructure, tools and business processes outside the educational area. We focussed on the specification of the sequencing of learning activities, and attempted to map these on the general purpose process and service specification language DAML-S. We found that that the concepts used in the IMS simple sequencing specification and the SCORM run time model, leading specifications in the educational area, as well as the model used in the experimental OntoAIMS platform seem to map readily onto the concepts and vocabulary in the DAML-S specification. The added value of using the DAML-S based specification is that it will allow for integration of e-learning in larger business processes (e.g. to facilitate the interoperability and information exchange information with a human resource management system).

Acknowledgements. The research work for this paper is realized within the scope of Topia project funded by the Telematica Instituut, the Netherlands. We want to tank Christian Tzolov for his valuable comments on the AIMS architecture.

References

[ADL] Advanced Distributed Learning Network. http://www.adlnet.org/ (visited 19-6-2003)
[Aro01] Aroyo, L. and Dicheva, D. (2001). AIMS: Learning and Teaching Support for WWW-based Education. IJCEELL, 11(1/2), 152–164.
[Aro03] Aroyo, L, and Mizoguchi, R. (2003). Authoring Support Framework for Intelligent Educational Systems. In Proc. AIED'03 Conference (in print).
[BL03] Berners Lee, T. Semantic Web Tutorial using N3 , tutorial at WWW 2003 Budapest April 2003 available at http://www.w3.org/2000/10/swap/doc/ (Visited 29-8-2003)
[Che98] Chen, W. Hayashi, Y., Kin, L. Ikeda, M. and Mizoguchi, R. (1998) Ontological Issues in an Intelligent Authoring Tool, in Chan T-W., Collins A. & Lin J. (Eds.), In Proc. of ICCE'98, 1, 41–50.
[DAM01] DAML+OIL Reference Description, http://www.w3.org/TR/daml+oil-reference
[DAM02] DAML-S: Web Service Description for the Semantic Web, DAML-S coalition, ICSW 2002, Sardinia, Italy, http://www.daml.org/services/ISWC2002-DAMLS.pdf
[DAML] DARPA Agent Markup Language (DAML), http://www.daml.org/

[Dev01] Devedzic, V. (2001). The Semantic Web – Implications for Teaching and Learning, In Proc. of ICCE 2001Conference, Seoul, Korea. [OIL] Ontology Inference Layer (OIL), http://www.ontoknowledge.org/oil/

[Dod02] Dodds, P., Hoberny, A., Blackmon, B. SCORM Next Generation, available at http://www.lsal.cmu.edu/lsal/expertise/papers/presentations/plugfest703122002joint/plugfest703122002joint.pdf (visited 19-6-2003)

[Fuk95] Fukuhara, Y., Kimura, F., Kohama, C., and Nakamura, Y. (1995). A Knowledge-based Educational Environment Integrating Conceptual Knowledge and Procedural Knowledge in Telecommunication Service Field. In Proc.ED-MEDIA'95, 229-234.

[IMS] IMS Global Leaning Consortium Inc. http://www.imsproject.org/ (visited 19-6-2003)

[IMS LD] IMS Learning design. http://www.imsproject.org/learningdesign/index.cfm (visited 19-6-2003)

[LOM] IEEE P1484.12 Learning Object Metadata Working Group, http://ltsc.ieee.org/wg12/

[Miz96] Mizoguchi, R., Sinitsa, K., and Ikeda, M. (1996). Task ontology design for intelligent educational/training systems. In Proc. of Architectures and Methods for designing Cost-Effective and Reusable ITSs Workshop, at ITS'96 Conference.

[Mur99] Murray, T. (1999). Authoring Intelligent Tutoring Systems: An analysis of the state of the art. International Journal of Artificial Intelligence in Education, 10, 98–129.

[Nil02] Mikael Nilsson (ed.). IEEE LOM RDF binding, http://kmr.nada.kth.se/el/ims/md-lomrdf.html

[OWL] Web Ontology Language (OWL), http://www.w3.org/TR/owl-ref/

[RDF] RDF Vocabulary Description Language: RDF Schema, http://www.w3.org/TR/rdf-schema/

[SCO1.2] ADL Initiative, SCORM 1.2 specification, (visited 19-6-2003) http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=840

[SCO1.3] ADL Initiative SCORM 1.3 specification, (visited 19-6-2003) http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=836