On the Semantics of Aggregation and Generalization in Learning Object Contracts

Salvador Sánchez
Faculty of Computing
Pontifical University of Salamanca - Spain
salvador.sanchez@upsam.net

Miguel-Angel Sicilia
Computer Science Department
University of Alcalá - Spain
msicilia@uah.es

Abstract

When machine-understandability is required to build software modules that automatically retrieve and combine learning objects, learning object relationships should be carefully considered, as they raise important semantic issues that influence runtime behaviour. In this paper, we analyse how learning object relationships have an effect on learning object contracts and look for analogies with the object-oriented paradigm. Being some of the most common relationships, we focus on the commitments that aggregation and generalization impose on learning object contracts.

1. Introduction

Current learning object metadata specifications support several kinds of relationships between learning resources. Indeed, LOM [4] includes a *Relation* category that groups features to specify the relationship between the learning object being described and other related learning objects. Furthermore, Dublin Core Metadata [2] also contains a *Relation* element as a means of specifying references to related resources.

Currently, both LOM and Dublin Core are not completely unambiguous regarding relationships; e.g. Farance [3] includes *relation* in a list of LOM elements whose definitions are imprecise and thus problematic, while the IMS best practices [5] explicitly state about the Dublin Core *Relation* label: "[...] it is currently under development. Users and developers should understand that use of this element is currently considered to be experimental". This vagueness is mainly due to the fact that there doesn't exist a shared consensus on the kind of relations that can be established between learning objects. Nevertheless, in order to attain consistent LMS behaviour, a clear determination of the runtime implications of the diverse kinds of relationships is critical.

From the origins of learning object technology, parallelisms have been established with the object-oriented paradigm (OOP) [10]. Some elements in LOM and Dublin Core bear some resemblance to the basic relationships in OOP, although a shared analogy has not been explicitly established yet.

In this paper we discuss the nature of learning object relationships, establishing analogies with relationships in the OOP. In section 2, information on relations provided by current metadata specifications is reviewed. In section 3, we reason about semantic relationships by comparing the LOM information on relations to structural relationships in the OOP. Then, in sections 4 and 5 we examine generalization and aggregation and the commitments they entail when applied to learning objects. Finally, conclusions are outlined in section 6.

2. Current metadata information on relations revisited

Metadata specifications, like Dublin Core and LOM, see learning objects relations primarily on a syntactical level, and consequently they do not specify semantic constraints regarding the description of concrete types of relationship. It is not precisely specified whether metadata information has to be added at both ends of the relationship. At the same time, it has not been clearly determined what a learning object can be linked to (other learning objects, descriptions of external resources, books, etc.). The following metadata record from a NETg learning object –provided by IMS as a metadata example– illustrates this case, since it links the learning object to external resources not properly

identified from the point of view of automation:

7.1:Relation.kind = isPartOf
7.2:Relation.Resource.Description = "NETg Course 72475: Microsoft SQL Server 7.0…"
...
7.1:Relation.kind = isBasedOn
7.2:Relation.Resource.Description = "MS Education and Certification Student Workbook… Page 127"
...

The Dublin Core (DC) metadata set includes a *Relation* element for referencing a related resource, and a number of refinement terms each providing support for a specific kind of relation: *hasVersion*, *replaces*, *requires*, *hasPart*, *references*, *hasFormat* and their symmetrical *isVersionOf*, *isReplacedBy*, *isRequiredBy*, *isPartOf*, *isReferencedBy* and *isFormatOf*. Moreover, there is a *Source* element that could be used for subtyping. Excluding symmetrical terms, DC terms supporting relations fall into two categories: referential and semantic. *Referential* terms support metadata information that is mostly syntactical. This category includes *hasVersion* (whether the described resource object has a version, edition, or adaptation), *replaces* (the described resource supplants, displaces or supersedes the referenced resource), *references* (the described resource references, cites, or otherwise points to the referenced resource) and *hasFormat* (the referenced resource is essentially the same intellectual content presented in another format). As this kind of information is not related to the learning object content, we will not deal with it here. *Semantic* terms refer to information that could be used in automated content selection and delivery. DC refinement terms in this category are *requires* (the described resource needs the referenced resource to support its function, delivery or coherence of content) and *hasPart* (the referenced resource is included in the described resource either physically or logically). The term *Source* (work from which the resource was derived) also belongs to this category, even though the forms and implications of derivation are unclear.

Regarding relations, LOM includes all the DC terms plus a few additional elements, grouped in a specific category (*7:Relation*) whose aim is *grouping features that define the relationship between the learning object and other related learning objects*. In subcategory *7.1:Relation.kind*, the LOM lists the value space for relations, which is basically a mapping of the already mentioned DC refinement terms plus *isBasedOn* (together with its complement *isBasisFor*) that maps to the DC *Source* term. Unfortunately, this is not the only category in LOM where information on relations can be included: *1.7:Structure, 1.8:Aggregation.level* and *5.2:LearningResourceType* can also be considered as implicit relationships.

Summing up: in general terms, current metadata information on relations lacks both structure and a clear definition. Moreover, in LOM this information is scattered over several categories, which makes its use unclear. It can be concluded that a) metadata information on relationships is not unambiguously specified and b) the discussed specifications do not seem to be ready for extensive use. This situation may mislead learning object designers and users, and it might be one of the reasons why, in many cases, this kind of element is not included in metadata records.

3. Semantic learning object relationships

Relating a learning object to another has semantic implications (understood as runtime commitments for the system using or delivering the objects), because some relations affect objects and create dependencies between them. As stated in [1], learning resources should be more accurately described by structural relationships in order to a) achieve better results when searching and b) to allow better queries. We will examine whether LOM can be used to successfully adapt structural relationships in the OOP to learning objects specificities.

In previous works [7][9], we have referred to preconditions as the constraints under which a learning object can be delivered and used, and postconditions as the object expected learning outcomes, thus introducing the concept of *contract* as a set of pre- and post-conditions for a given learning object. This notion of contract will be used to support formal metadata specifications.

Four major relationships exist in OOP –as defined in the UML– and all of them entail meaningful semantic responsibilities in the elements involved:

- *Association* and *aggregation* describe semantic connections between objects.
- *Dependency* relates two objects, the changes in one of which affect the other.
- *Generalization* relates a more general class and a more specific one.

It can be noticed that not all these relationships are defined at the same level: while some describe links between instances, others show connections between classes, and thus can be classified into instance-level (association, aggregation and dependency) and class-level relationships (generalization).

OOP relationships can be mapped to LOM by using the *7.1:Relation.kind*, as described in Table 1.

Table 1. OOP Relationships as LOM elements

OOP Relationship	LOM element
Association	`requires`
Generalization	`isBasedOn`
Aggregation	`hasPart`
Dependency	`references`

However, generalization can also be mapped to the LOM *5.2:LearningResourceType*. Utilizing *isBasedOn* suggests the possibility of an indeterminate number of user-defined available learning object types. On the other hand, using *5.2:LearningResourceType* suggests that a limited list of universally acknowledged learning object types exist, whose structure is known.

Relationships bring in *runtime commitments* that will affect the whole type (class-level relationships) or just the individual objects that participate in the relationship (instance-level relationships). From a learning object point of view, the most important commitment is *availability*, present in almost every relationship. Availability means that the referenced resource must be available whenever the current learning object is used or delivered. If a learning object –say A– that is being delivered to a learner includes a reference to another object –say B–, availability of B can be proved in two different forms: a) the referenced object must be effectively available, or b) the learner must provide evidence of a level of knowledge greater than or equal to that stated in the learning outcomes published in the contract of A [9]. Other runtime commitments are:

- *Propagation*: some properties propagate from the aggregate to the parts.
- *Acyclicness*: chains of relationships are not allowed to form cycles.
- *Contract inheritance*: subtypes inherit the contract defined for a type.
- *Reference validity*: the weakest form of *availability*, understood as a way of validating the existence of the referenced resource.

Table 2 shows an inclusive set of commitments for learning object relationships.

Table 2. Relationship commitments

Relationship	Commitments
Association	Availability
Aggregation	Propagation Availability Acyclicness
Generalization	Availability Contract inheritance
Dependency	Reference validity

In what follows, the discussion is intentionally focused on the most common ones: generalization and aggregation.

4. Generalization: discussing about the concept of learning object type

In the OOP, every object is an instance of a class –a declarative element that describes the behaviour and structure of a set of objects–. As objects have their own entity and values, and are identifiable during execution, the descriptions of all the objects in a system must be available in order to know about their behaviour and structure.

Although learning objects are assumed to be instances, LOM incorporates the concept of type as the value set in *5.2:LearningResourceType* (*simulation*, *exercise*, *diagram*, etc.). As this list can be extended, one can presume that there is no limit to the number of types, but given the current approach the structure and implications of the existing types is unclear.

To belong to a certain learning object type should entail a number of commitments. For example, let's suppose that the LOM 5.2 entry value is *diagram* in a given metadata record; in this case the learning object will be probably required to include information on the formal language of representation –*UML* for instance, or *none* if it is a free notation diagram–. On the other hand, if the value is *questionnaire*, information on the number of questions is expected, and this time any reference to the formal language of representation will be probably considered an error. This example shows how, despite the fact that LOM does not provide information on the structure of the types it supports, details about the structure of the types are needed. The problem is that it is not possible to select a subset of specific metadata elements depending on the learning object type.

However, a different solution could have been chosen: introducing the concept of learning object type as an object whose content is not directly usable but a description of other objects (a class). As in OOP, type descriptions would be used whenever an object of the class is delivered or used. This new approach suggests expressing learning object types in their metadata record, for instance, by adding the value *NULL* to the *5.2:LearningResourceType* vocabularies. This would allow us to represent a *Diagram* type as follows:

5.2:LearningResourceType = NULL
5.2.1:LearningResourceTypeName = Diagram
5.2.1.1.TypeDependantCategory = 5.2.2
5.2.1.2.TypeDependantElementName = RepresentationLanguage
5.2.1.3.EntryType = CharacterString

Category 5.2.2 is not actually part of LOM, and it is automatically generated to accommodate the new type dependant elements. According to this model, a *Diagram* instance could be described as:

5.2: LearningResourceType = Diagram
5.2.2: RepresentationLanguage = UML

Adopting this model implies, on the one hand, modifying LOM to allow type descriptions based on including type dependant metadata elements for those resources whose *5.2:LearningResourceType* is NULL; and on the other hand, extending the formal notation given in [9] to support the new approach. With regard to learning object contracts, generalization is not easy to represent since learning object design by contract is defined at instance level. It seems obvious that at least contract inheritance should be assured, which means that all the pre- and post- conditions of the parent apply to the subtype. The following assertions could be part of the contract of a questionnaire subtype:

```
rlo <http://... /VideoBasedQuestionnaire>
  require
    mandatory <http://... /Questionnaire>
    mandatory sys.requiresVideoFormat ≠ NULL
    ...
  ensure
    <http://... /Questionnaire>
    ...
```

In this example, contract inheritance is guaranteed by adding the parent type contract to the current type list of postconditions. Major shortcomings of this approach are the lack of support by current metadata standards, the inexistence of a common language for type description and the need for a means of making accessible the available classes. However, such definitions would enable more specialized metadata schemas to be defined.

5. Aggregation

Most learning objects are compositions of others: this nature is on the basis of reusable learning resources. LOM category *1.8:AggregationLevel* identifies four levels of aggregation, numbered from 1 (raw media data or fragments) to 4 (a set of courses that lead to a certificate). A level n object can contain a number of level n-1 objects or can recursively contain objects of level n. Likewise, category *1.7:Structure* classifies the different types of aggregation from their internal structure: collection, linear, hierarchical or networked. However, being merely a way of classifying objects by their granularity, LOM information in categories 1.7 and 1.8 does not enforce any dependency on the aggregate or its constituent parts. In contrast, aggregation should be seen as a semantic relationship, which implies two major constraints: properties propagate from aggregate to parts (and vice versa), and no cycles of aggregation links are permitted.

The contract of a learning object that is an aggregate of others is often affected by the contracts of its parts. Consider a LOM-conformant level 3 learning object (course) in Spanish: as the course is a composition of lower-level objects, their language will have to be Spanish as well. So, whenever the elements comprising an object are other learning objects, their contracts will need to be conformant to what the aggregate contact states: this is semantic aggregation. When representing semantic aggregation in metadata records, a number of data items depend on information in other objects, as stated in Table 4.

Table 4. Aggregation constraints

LOM Category	Constraint
1.3:Language 5.1:InteractivityType 5.7:TypicalAgeRange 6.1:Cost 6.2:Copyright&Restrictions	The desired value for the aggregate restricts the value of the parts to be chosen
1.7:Structure 1.8:AggregationLevel 5.3:InteractivityLevel 5.4:SemanticDensity 5.8:Difficulty	The value (level) in the aggregate must be greater or equal than the value of its parts
2.2:Status	*Completion* on the aggregate enforces completion on every part
4.1:Format	Aggregate incorporates all the formats in its parts
4.2:Size 4.7:Duration 5.9:TypicalLearningTime	Aggregate values calculate from the sum of the values of its parts
4.3:Location	Location of the parts has to be publicly accessible
4.4:Requirement 4.6:OtherPlatformRequirements	Aggregate value is made up of all the requirements in its parts

When a learning object is part of a higher-level object, it is important to consider whether this should be set in its metadata record or not. Reusability asks for the components of an aggregation to be built without any knowledge about it, in order to facilitate its future reuse, so it must be the aggregate responsibility to compose all the parts and to keep track of them. Also, learning object authors (or automated systems) have to carefully pick out the parts constituting an aggregate, since its contract will impose conditions to be satisfied by the parts before they can work together. In the following example, a learning object *MyLO* that displays a Flash-animated example of the *Quicksort* algorithm, is under consideration by a composer agent [8] in order to integrate it into a Java course object:

```
rlo <http://... /MyLO>
 require
   mandatory   lrn.language = es
   mandatory   sys.browser >= V5_browser
   mandatory   sys.requirement = FlashPlugIn
   mandatory   ctx.cost = true
   recommended ctx.time = 0.5h
 ensure
   lrn.knows(qSort)[90]
```

Due to *propagation*, the aggregate learning time will remain unknown until the total number of pieces are assembled since it is calculated from them. *Availability* asks for MyLO to be publicly accessible whenever the aggregate is being delivered or used, but also for its usage fee to be paid. As MyLO is not composed of other learning objects (it is a leaf in the aggregation tree), *acyclicness* is guaranteed given that all the potential cycles would end here. The aggregate contract could be something like:

```
rlo <http://... /AgentGeneratedAggregate>
  require
    mandatory   lrn.language = es
    mandatory   sys.browser > V5_browser
    recommended ctx.time = 10h
    mandatory   ctx.cost = true
    mandatory   ctx.hasPart = "MyLO"
    ...
  ensure
    lrn.knows(qSort) [90]
    ...
```

As the example shows, MyLO design is not constrained by pre- or post- conditions in the aggregates it would be part of. It is the responsibility of the author of the aggregate to check the consistency of the parts with the aggregate contract and use some features to calculate the final values of some assertions (as *learning time* in our example). Also, some preconditions in the aggregate contract may be required to be stronger than those of its parts. For example, browser requirements for the above aggregate are stricter than the ones in MyLO, possibly due to the existence of other parts with stronger requirements. The new precondition *hasPart* appears after choosing MyLO and as a consequence of the relationship.

We suggest stating information on relationships only in the aggregates by using LOM item *hasPart* (and not in the parts, thus avoiding the use of *isPartOf*). This will allow to include information on semantic aggregations, as it forces systems to check the related resources to a given one. This solution is consistent with Dublin Core and ensures compatibility with it.

6. Conclusions

As it is currently defined, information on relations is inessential because it leans on an imprecise notion of learning object relationship. Generalization and aggregation, the most common relationships in the OOP, have been used as the material for a discussion on LOM representation of meaningful relationships. Learning object design by contract is a means of formalizing metadata records that helps us to represent the runtime commitments that these relationships bring in when taken into the learning objects arena.

Future work should detail the implications of learning object relationships so that full consistency in LMS behaviour could be achieved.

7. References

[1] Brase, J., Painter, M. and Nejdl, W. “Completing LOM – How additional axioms increase the utility of learning object metadata”, in Proceedings of the *3rd IEEE International Conference on Advanced Learning Technologies*, ICALT’03.

[2] Dublin Core Metadata Initiative, http://dublincore.org [Last accessed in February 2004]

[3] Farance, F. “IEEE LOM standard not yet ready for ‘prime time’“, *IEEE Learning Technology Newsletter*, Jan. 2003, vol. 5, issue 1, pp. 21-23.

[4] IEEE Learning Technology Standards Committee, “Learning Object Metadata (LOM)”, IEEE 1484.12.1-2002.

[5] IMS Global Learning Consortium, “IMS Learning Resource Meta-data Best Practice and Implementation Guide”, version 1.2.1 (final specification), Sept. 2001. Online at http://www.imsglobal.org/metadata

[6] Polsani, P. “The use and abuse of reusable learning objects”, *Journal of digital information*, vol. 3, issue 4, 2002. Online at http://jodi.ecs.soton.ac.uk/Articles/v03/i04/Polsani

[7] Sánchez, S. and Sicilia, M.A. “Expressing preconditions in learning object contracts”, in proceedings of the *Second International Conference on Multimedia and Information & Communication Technologies in Education*, 2003, pp.1656-1660. Online at: http://ssanchez.colimbo.net/papers_en.htm

[8] Sánchez S. et al. “Learning object repositories as contract-based Web services”, *IEEE Learning Technology Newsletter*, Jan. 2004, vol. 6, issue 1, pp. 16-18. Online at: http://lttf.ieee.org/learn_tech/issues/january2004

[9] Sicilia, M.A. and Sánchez, S. “On the concept of Learning Object Design by Contract”, *WSEAS Transactions on systems*, Oct. 2003, vol. 2, issue 3, pp. 612-617. Online at: http://ssanchez.colimbo.net/papers_en.htm

[10] Sosteric, M. and Hesemeier, S. “When is a learning object not an object: a first step towards a theory of learning

objects", *International Review of Research in Open and Distance Learning, Oct. 2002, vol. 3, issue 2.*