EXPRESSING PRECONDITIONS IN LEARNING OBJECT CONTRACTS

SALVADOR SÁNCHEZ-ALONSO

Computer Languages and Systems Department, Pontifical University of Salamanca (Madrid campus) Paseo de Juan XXIII 3 – 28040 Madrid, SPAIN.
E-mail: salvador.sanchez@upsam.net

MIGUEL-ANGEL SICILIA

Computer Science Department, Carlos III University, Avda. Universidad 30 – 28911 Leganés (Madrid) SPAIN.
E-mail: msicilia@inf.uc3m.es

Not having a formal semantics for representing metadata instances in a machine understandable way, LOM makes it difficult to implement software modules capable of searching and retrieving data from learning object repositories based only in conforming metadata descriptions. Design by Contract, a well-known object oriented software design technique aimed at helping developers and designers to reuse objects, is revisited in order to find similarities with the paradigm of learning objects. Design by Contract is introduced here as a convenient means of formalizing the information in metadata instances resulting in machine understandable metadata information. In this paper we describe preconditions as an essential part of applying Design by Contract techniques to learning objects. A definition of what a precondition is intended to be in the learning objects arena is described, as well as a syntax for writing formal preconditions. Finally, we map the existing LOM metadata categories that can be useful in expressing preconditions to our machine-readable syntax.

KEYWORDS: learning objects, metadata, design by contract, preconditions.

1 Introduction

Polsani's definition of reusable learning object [7] as "independent and self standing unit(s) of learning content predisposed to reuse in multiple instructional contexts" and other definitions consistent with it as the ones given by Sosteric and Hesemeier [10] and Hamel and Ryan-Jones' [3] evidence the necessity of including metadata together with the objects. The metadata instance attached to a given learning object provides information on its contents, what undoubtedly facilitates its reusability. Among all the current specifications and proposals on metadata, LOM [4] emerges as the most important initiative, basically after being adopted by the more comprehensive SCORM [1] specifications.

But even though there is real enthusiasm in the learning object community with regard to LOM, there exist some problems that have not been solved yet. On the one hand, as Farance says in [2], it remains a few open technical issues derived from LOM being a young standard, still in the process of adoption and subject to changes before it becomes widely used. On the other hand, the approach that permits every single element in a LOM metadata instance to be optional seems to be an excessively optimistic one. According to the declaration of conformance of LOM, "a LOM instance that contains no value for any of the LOM data elements is a conforming instance". In many cases, such a declaration ranging from no value to the full list of values for every item in the category list, makes machine useless many metadata records. Finally, the lack of formalization in LOM affects the way a machine reads, analyzes and processes the information in a learning object, for instance, to automatically elaborate new educational materials from already existing learning objects. It is necessary then to formalize metadata records to obtain useful machine understandable information, what implies the design of a language —or formal representation— of the knowledge. Such representation will guarantee the consistency of the metadata, and therefore the effective reuse of the objects they are attached to.

As Longmire remarks in [5], numerous parallelisms have been identified between the theory of learning objects and object oriented programming. In a previous work [9], we have proposed using design by contract (DBC) [6] —a prominent software design technique from the object oriented software domain— as a means of introducing a formal way of specifying learning object responsibilities (postconditions) and circumstances of use (preconditions). DBC becomes particularly useful when defining metadata instances for learning object preconditions, since preconditions embody the assertions in the object contract to be satisfied by any automated system trying to (re-)use the object.

In section 2 of this paper we will examine preconditions as an essential part of the DBC mechanism applied to learning objects. Later, in section 3 we will sketch a proposal about formalizing LOM elements holding information that can be modeled as preconditions, by using specific idioms. Finally, in section 4 conclusions on this work are described and some future lines of research are briefly outlined.

2 Design by Contract in Learning Objects

Applying design by contract to learning objects consists, in the simplest possible scenario, in specifying a formula in the form *{C}LO{O} [θ]* for each learning object meaning the following: using the learning object LO in a learning context *C* that includes a description of a specific learner profile, it facilitates the acquisition of some kind of learning outcome *O* to a certain degree of credibility θ [9]. This formula using preconditions, postconditions and invariants, allows the learning object designer to define formal contracts that represent the behaviour of an individual object in a learning object system. Of course deciding whether a learning object is appropriate for a particular learning objective or not has to be based on the outcomes the object ensures (postconditions), but once the learning object has been chosen and before it can be used, preconditions accomplishment is essential. We can then define precondition in the learning objects domain as follows:

Preconditions state the constraints under which a learning object can be delivered and used.

Clients —a Learning Management System (LMS), or any other software— are required to satisfy the preconditions of a learning object before they can use it, otherwise clients will not be granted its use, in such a way that the learning object will become useless for those clients under the given circumstances. We will focus on preconditions as a way of improving the way we design reusable learning objects.

2.1 Weak preconditions vs. strong preconditions

Weak preconditions, understood as excessively open or permissive ones, represent "bad news" for the learning object designer, since the laxness of such an assertion implies that the content has to be appropriate for a wider educational scenario. A learning object either described by a metadata instance including only a few pairs entry-value in the full set of LOM categories, or empty —which is also LOM conformant— complicates the job of the automatic learning object search engines and delivery systems. Furthermore, designing learning objects whose preconditions are too weak or even non existent, will involve significantly more effort in order to ensure the fulfillment of their postconditions (a LMS delivering a learning object about the battle of Waterloo without assuming any previous knowledge on the learner's part, will hardly be able to guarantee that the learner will fully understand what happened given that the learner maybe doesn't know anything about the historical context where the battle took place).

However, too restrictive —strong— preconditions also bring disadvantages. They can seriously affect the number of times a learning object is reused. If the learning object was designed with an economic aim, as for example charging a utilization fee, fixing too strong preconditions in its contract could be harmful and thus, should be reconsidered. The learning object designer is then forced to tradeoff between the economic benefit derived from a 'weak' design and the appropriateness to future and unknown contexts of use derived from a 'strong' design.

Subsequently, on the designer's side the weaker the preconditions the more difficult it becomes to design reusable learning objects. This is due to the fact that objects defined by excessively weak preconditions will hardly accomplish the goals stated in their contracts so, in accordance with the relation between usability and reusability in certain contexts given in [8], learning object global "usability" will be lower. In conclusion, we should always opt for learning objects designed under the strongest possible (and reasonable) preconditions, even if there are tempting reasons —as the mentioned— not to do so.

But in order to get fully machine understandable metadata, it is also important to enforce the compulsory character of preconditions in metadata instances because optional elements lead to obtaining ambiguous or non-informative metadata instances. In order to write these new kind of metadata we suggest: a) mapping LOM elements into a formal language which obliges providing values for at least an essential set of preconditions (providing values for all of them is highly recommended); and b) establishing default values representing the lack of a preference: if the learning object you are designing does not have any specific requirements about the browser needed to display it, we recommend setting the *browser* element to 'any' instead of avoiding the *browser* clause in the object contract.

2.2 A taxonomy of preconditions

After an analysis of the data elements in LOM conformant metadata instances, learning object preconditions can be classified into three categories as follows:

- System requirements: most of the elements under the LOM *Technical* category can be seen as system preconditions from the DBC perspective. Requirements in this category point out to system settings where the learning object is due to be executed. Surprisingly when the lack of

information about the system requirements can impede the object utilization, LOM permits not including requirements of this kind in conforming metadata instances. In our opinion, including these elements in metadata instances has to be mandatory.

- Context requirements, some of which fall into the compulsory group (for instance, information about the human language in which a text is written) while some others can be seen as more flexible items and even optional (as the information about the kind of interaction, average estimated duration, etc.)
- Learner requirements, understood as the ideal learner profile. Learner requirements are mainly considered as optional, but this non-compulsory nature can be shaded in a scale having different levels, as we will see later in this paper. Of course, the closer the learner profile is to what the object contract states, the more suitable the learning object will be and the higher the usability for that object in the given situation, considering [8].

We use preconditions in the learning object contract to formally indicate the requirements in the above categories. The following syntax has been purposely defined to it:

```
preconditionId.element <relationalOperator> requestedValue
```

where the kind of preconditions can be seen in table 1:

Table 1. Formal specification for preconditions.

	preconditionId	**example**
System preconditions	`sys`	`sys.operating_system = unix`
Context preconditions	`ctx`	`ctx.language = en-GB`
Learner preconditions	`lrn`	`lrn.language = en`

Requested values will be valid values defined in the domain of the element being qualified. When more than a value for the precondition is admitted, an assertion containing a list of OR separated values in order of preference will be used. Use '|' as OR separator as follows:

```
sys.browser > v5
ctx.language = en-GB | en //indicates english, preferably british
```

This is an example employing the syntax defined in [9] to describe a metadata instance corresponding to an introductory lesson on the use of irregular verbs in English intended for a German speaking audience:

```
rlo <URI>
      require
            sys.browser = any;
            ctx.language = en;
            lrn.language = ge;
```

2.3 Compromise levels

One of the targets when formalizing contract preconditions is creating rigid rules that improve learning object reusability. In the preceding taxonomy, not all the categories should have the same weight when requesting a precondition to be accomplished. By now, we have mentioned that system preconditions must be mandatory due to their particular connotations, but dealing with learner preconditions is different, since determining how well a learner fits into some previously defined learner category is not trivial.

When we said that lists of values could be used to express a precondition, we set a basic rule: similar preconditions can be prioritised depending on their position in the list. Unfortunately, this is not enough. We need a way of better expressing how strong a requirement really is, because sometimes *we need* to have an element but sometimes just *it would be nice* to have it. We introduce what we call *compromise levels* as a way to fix this problem:

- `mandatory`: the element is essential. A system that needs using the learning object must accomplish what the assertion states.

- `recommended`: it is good to have the element. Not accomplishing what is stated in the assertion can lead, for instance, to an inaccurate visualization of the learning object[1].
- `optional`: the element is elective, but accomplishing what the assertion states drives the learner to a higher 'score' in the total output count. In a strict system of preconditions this level can be avoided, however it proved to be necessary for mapping a number of LOM specifications into our syntax.

Information on the levels of compromise must be added to every assertion in the learning object contract. So, our previous syntax needs a slight amendment:

```
[level] preconditionId.element <relationalOperator> requestedValue
```

Assertions not preceded by a compromise level tag are assumed to be `mandatory`, as this is the default value. In fact, writing an assertion without a compromise tag —or having a `mandatory` tag— is considered a good practice in order to ensure strong preconditions. This way, our earlier example about a course on irregular verbs in English would be written as:

```
rlo <URI>
      require
            mandatory   sys.browser = any;
            mandatory   ctx.language = en;
            recommended lrn.language = ge;
```

3 Mapping LOM specifications into preconditions

Formalising LOM metadata information aims the metadata instances to become machine understandable. This is to be achieved by mapping LOM categories to the above defined categories and idioms:

1. *Context*: information on context preconditions is not grouped as a single category in LOM. The following elements get involved when mapping LOM context information into preconditions:

Table 2. Context preconditions.

LOM entry	Description	Element	Minimum compromise level
5.6 Context	Principal environment within the use of the object is due to take place	`type`	`recommended`
5.7 Typical age range	Age of the intended user, understood as development age if different from the chronological age.	`age`	`recommended`

2. *System*: most system preconditions easily map to elements in the LOM category "4. Technical", which describes technical requirements and characteristics of a given learning object:

Table 3. System preconditions.

LOM entry	Description	Element	Minimum compromise level
1.3 Language	Human language of the learning object[2]. Provided that an object may have no text, the appropriate value for these data would be "`none`".	`language`	`mandatory`
4.1 Format	Technical data types of the components in the learning object	`format`	`mandatory`
4.3 Location	URL or URI.	`accessTo`	`mandatory`
4.4 Requirement	Technical capabilities for using the object, expressed by type, name and version (min and max).	`requirement`	`mandatory`
4.6 Other platform requirements	Other software and hardware requirements.	`requirement`	`mandatory`
5.2 Learning resource type	Specific kind of learning object	`resource`	`mandatory`

[1] In learning objects with more than a contract the element will be probably linked to a concrete output O_i.

[2] The system must be able to represent every symbol in the language of the object.

3. *Learner*: on the user side, several LOM categories entail preconditions the learner must hold before a given learning object can be delivered to her.

Table 4. Learner preconditions.

LOM entry	Description	Element	Minimum compromise level
5.1 Interactivity type	Predominant mode of learning in the user side.	`interactivity`	`optional`
5.5 Intended end user role	User role the learning object is designed to.	`role`	`optional`
5.8 Difficulty	How difficult or hard is to work with the LO for the intended audience.	`ability`	`optional`
5.11 Language	Human language used by the user of the learning object.	`language`	`recommended`
9.2 Taxon path	Taxonomic path in a specific classification system.	`knows`	`recommended`

We highly recommend including precondition clauses for all the preceding categories and entries while defining learning object contracts, as a means of obtaining relevant metadata instances that are as accurate as possible with the learning object content. This way, metadata authors will help computer systems to better identify relevant learning objects to satisfy given needs.

4 Conclusions and future work

Learning object metadata records require well-defined semantics in order to become machine-understandable and enable automated retrieval, delivery and aggregation of learning resources. In this paper, the Design by Contract philosophy has been described as a means towards that end. Concretely, the purposeful specification of LOM metadata elements as preconditions has been described. Future work will address the complete coverage of the syntax here outlined and also the development of software modules that read and handle learning object contracts in diverse decision-making steps that arise in the design of learning experiences for specific audiences and contexts.

References

1. Advanced Distributed Learning (ADL) Initiative (2003). Sharable Courseware Object Reference Model (SCORM). ADL SCORM Version 1.3 Application Profile Working draft 1.0. Available at http://www.adlnet.org/Scorm/
2. Farance, F. (2003) IEEE LOM Standard Not Yet Ready For "Prime Time". *IEEE LTTF Learning Technology Newsletter*, special issue, Vol. 5, issue 1.
3. Hamel, C. J. and Ryan -Jones, D. (2002). Designing instruction with learning objects. *International Journal of Education Technology*, 3 (1).
4. IEEE Learning Technology Standards Committee (2002). Learning Object Metadata (LOM). IEEE 1484.12.1 — 2002.
5. Longmire, W. (2000). A primer on learning objects. American Society for Training & Development Learning Circuits, March 2000. Available at http://www.learningcircuits.org/mar2000/primer.html
6. Meyer, B. (1997). Object Oriented Software Construction. 2nd Edition, Prentice Hall. pp. 331-410.
7. Polsani, P.R. (2002) The Use and Abuse of Reusable Learning Objects. *Journal of Digital information*, volume 3 issue 4. Available at http://jodi.ecs.soton.ac.uk/Articles/v03/i04/Polsani
8. Sicilia, M. A. and García, E. On the Concepts of Usability and Reusability of Learning Objects. *International Review of Research in Open and Distance Learning* (to appear).
9. Sicilia, M. A. and Sánchez -Alonso, S. (2003). On the Concept of Learning Object Design by Contract, *Proceedings of the 5th Telecommunications and Informatics WSEAS Conference.*
10. Sosteric, M. and Hesemeier, S. (2002). When is a learning objec t not an object: a first step towards a theory of learning objects. *International Review of Research in Open and Distance Learning Journal*, 3 (2).