Representing Programming Object Metadata Using the IEEE LOM Methodology

Daniyal M. Alghazzawi, IEEE
Information Systems Department, Faculty of Computing and Information Technology,
King Abdulaziz University, Saudi Arabia
dghazzawi@kau.edu.sa

Abstract-Software engineering is concerned with developing reusable high-quality programming objects such as architectures, source code, data, designs, documentation, templates, human interfaces, plans, requirements, and test cases. In order to deploy such as these objects, we need to maintain a metadata for them. In this paper, we used the term Programming Object Metadata for such a metadata. Currently, there is no standard for this type of metadata. Therefore, this paper focuses on developing a metadata for the programming objects by using IEEE Learning Object Metadata 1484.12.1-2002 methodology. The proposed metadata was validated by providing an example of a Programming Object Metadata (POM) in XML format.

***Index Terms*—Programming Object, Metadata, Learning Object, Reuse, Quality.**

I. INTRODUCTION

Developing high quality software at low cost is a major challenge for the software industry [1]. One way to accomplish this task is to have a reusable *Programming Object* plan [2]. A programming object is defined in this paper as any entity–digital or non-digital–that may benefit software product, such as architectures, source code, data, designs, documentation, templates, human interfaces, plans, requirements, and test cases. Sharing high-quality programming objects between developers in an institution or across institutions will enhance the quality of the software in a short period.

The major barrier in sharing a programming object between parties is the protocol (the language of speech) used to describe the programming object. We can formalize this protocol by developing a metadata called Programming Object Metadata (POM). A metadata instance for a programming object is defined in this paper as a description of relevant characteristics of the programming object to which it applies. Although there are numerous standards in software engineering that are developed by Institute of Electrical and Electronics Engineers (IEEE), none of them addresses the Programming Object Metadata [3,4]. On the other hand, IEEE developed a similar standard in the education field called IEEE Learning Object Metadata. Therefore, this paper proposes a metadata for Programming Objects by using IEEE LOM (1484.12.1) standard methodology.

IEEE LOM (1484.12.1) standard is described in section 2. Section 3 presents the proposed changes to the IEEE LOM standard in order to represent a programming object. At the end, section 4 provides a conclusion about this work.

II. BACKGROUND ON IEEE LOM STANDARD

The IEEE Learning Object Metadata (IEEE LOM) draft standard was developed by the IEEE Learning Technology Standard Committee (IEEE LTSC) in 2002 to represent *learning objects*. A learning object is defined by the standard as "any entity–digital or non-digital–that may be used for learning, education or training" [5], such as, multimedia resource, case studies, and education tools [6]. A *Learning Object Metadata (LOM)* describes the characteristics of a learning object, so it can be shared between individuals or institutions using a Learning Management System (LMS). Also, this standard addresses sharing learning objects between nations, so it permits linguistics diversity of both learning objects and the metadata that describe them.

Sharing learning objects between institutions may result in reducing the time-cost of development of a learning object and enhancing the quality of education. The cost of developing a new learning object from scratch will be reduced because of the capability to use a similar object that was developed by others. Also, the quality of the education in an institution will be enhanced by using high-quality learning objects. In general, sharing high-quality objects will reduce the time-cost of developing similar objects and enhance the quality of the field.

IEEE LOM standard categorizes the characteristics of a learning object in 9 categories: *general*, *life cycle*, *meta-metadata*, *technical*, *educational*, *rights*, *relation*, *annotation*, and *classification categories*. Each category contains data elements that describe the learning object or its metadata in detail. Appendix A shows the 9 categories with their data elements. Each of the data elements that do not include any sub-category (*leaf node*) is defined by name, explanation, size, ordering, value space, and datatype. All of the data types –`LangString`, `DateTime`, `Duration`, and `Vocabulary`– used in describing data elements are defined in the standard.

III. USING IEEE LOM STANDARD TO REPRESENT PROGRAMMING OBJECTS

There are 46 main data elements in the IEEE Learning Object Metadata standard across all the categories used to describe a learning object and its metadata; all of them are used to describe a programming object, while some of which are modified to suit the programming objects. The descriptions of 7 data elements (`1.1 Identifier`, `1.7 Structure`, `2.2 Status`, `4.2 Size`, `4.3 Location`, `9.1 Purpose`, and `9.2 Taxon Path`) are changed slightly to be used in characterizing Programming Objects. On the other hand, 2 of the data elements (`1.6 Coverage` and `5.7 Typical Age Range`) are eliminated because there are not related to any programming object. In addition to the existing data elements, 4 new data elements (`5.x Time Unit`, `5.x Requirement`, `5.x Testing Techniques`, and `6.x Reuse Type`) are added to further specify a programming object. Besides the necessary changes in the data elements, the descriptors for 2 categories (`4 Technical` and `5 Education`) are changed slightly. Fig. 1 presents a diagram for the categories and the data elements are proposed to be modified to suit Programming Object Metadata.

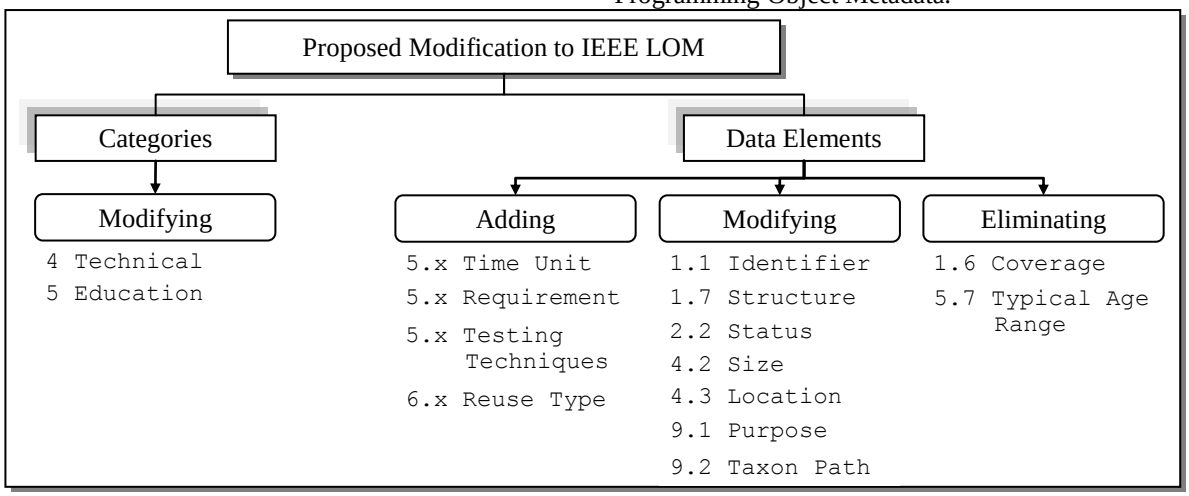


FIG. 1: Proposed Modification to IEEE LOM to suit Programming Object Metadata

The following four subsections describe the proposed modifications to the IEEE LOM Standard to conform to the Programming Object Metadata. Appendix B provides an example of the proposed POM in XML format.

A. Modifying Categories

This paper proposed modifying the descriptors for two categories in IEEE LOM Standard to conform to the metadata of Programming Objects. These categories are:

- `4 Technical`
- `5 Education`

A slight change is proposed in the descriptor for the category `<5 Education>` to suit the Programming Objects. This category may be used in any Programming Object that seeks an education purpose, which implies that most of the data elements in this category can be used to describe Programming Objects. In this paper, it is proposed that the data elements that focus on the quality be merged with the category `<5 Education>`; thus, the time and the space cost data elements would be in this category. Later in section 3.4, it is proposed that two new data elements related to the quality would be added to this category. The name of this category may need to be changed to a different name to meet the new description, such as `<5 Complexity>` or `<5 Quality>`.

The other proposed change is in the descriptor for the category `<4 Technical>`. In LOM, this category describes the technical requirements of a learning object and its source. However, we propose to separate the technical requirements of the object from the technical requirements of the object source. So, this category will focus only on the source of the programming object, while the data elements intended for the programming object will be moved to the previous category `<5 Education>`. The name of this category may also need to be changed to a different name to meet the new description, such as `<4 Source>` or `<4 Technical Source>`.

In the next three sections, this paper proposed either modifying, adding, or eliminating some data elements, some of which are caused by the changes in the descriptors of the two categories `<5 Education>` and `<4 Technical>`.

B. Modifying Data Elements

This paper proposed modifying seven data elements in IEEE LOM Standard to conform to the metadata of Programming Objects, which are:

- `1.1 Identifier`
- `1.7 Structure`
- `2.2 Status`
- `4.2 Size`
- `4.3 Location`
- `9.1 Purpose`
- `9.2 Taxon Path`

The first and second proposed changes in the data elements of the LOM standard are to distinguish clearly the difference between the two data elements: <1.1 Identifier> and <4.3 Location>. In POM, the first data element <1.1 Identifier> refers to the programming object itself, since the source may have more than one programming object while the second one <4.3 Location> refers to the source itself. For example in Fig. 2, if we have a web page that describes more than one programming object, and each of them has a link that refers to it, the data element <4.3 Location> will refer to the web site, and <1.1 Identifier> will refer to one of the programming-object links.

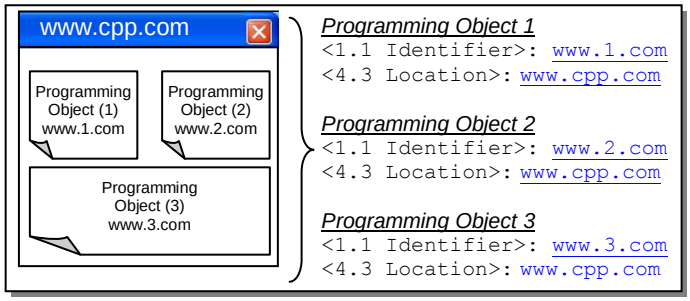


FIG. 2: An example of the two data elements <1.1 Identifier> & <4.3 Location> in POM

The third proposed change in the data elements of the LOM standard is in <1.7 Structure> by eliminating one value from its value space defined in LOM standard. The value space for this data element has 5 values: *atomic*, *collection*, *networked*, *hierarchical*, and *linear*. All of these values except the last one (*linear*) can be used to describe a Programming Object. Table. I shows an example of a Programming Object for each of these values.

TABLE I
AN EXAMPLE OF THE DATA ELEMENTS <1.7 STRUCTURE> IN POM

<1.7 Structure>	Example	Programming Objects

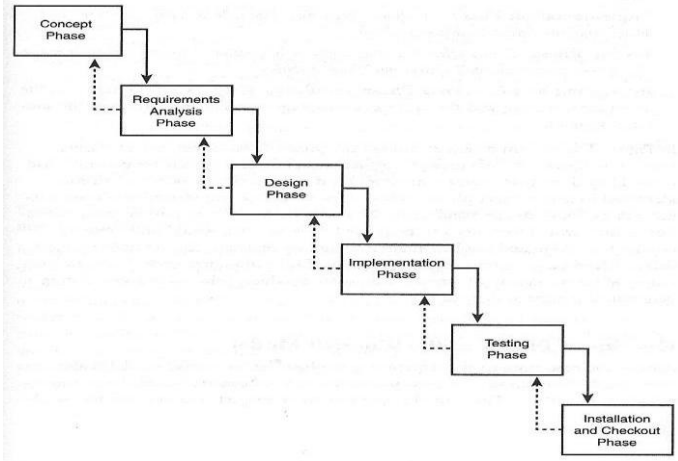
Atomic	Define a variable	Struct StudentInfo
		{
		long ID;
		char Name[100];
		}

Collection	A function. Each function may have more than one object without relationship between them.	Function Init()
		{
		StudentInfo S1;
		StudentInfo S2;
		}

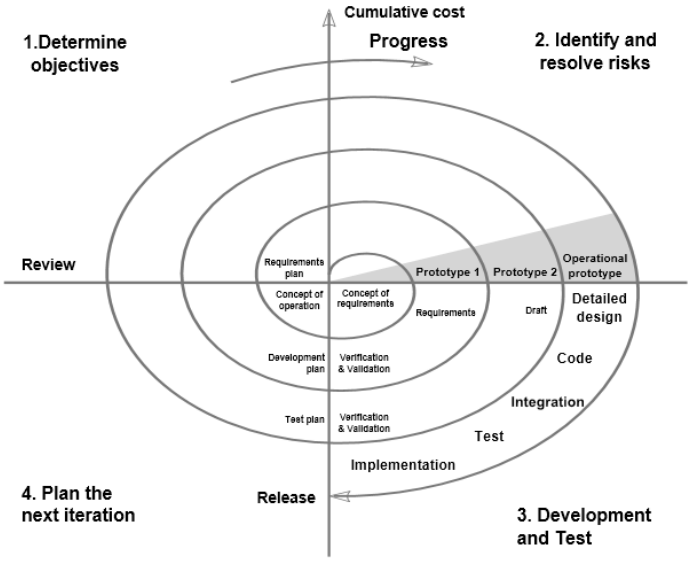
Networked	A Class. It has more than one object inside with relationship between them.	Class NumCalculation
		{
		Private:
		int X;
		int Y;
		void PrintSum(int,int);
		void PrintMul(int,int);
		}

Hierarchical	Class inheritance	Class New::NumCalculation
		{
		:
		}

The fourth proposed change in the data elements of the LOM standard is in <2.2 Status> by using different value spaces. This data element <2.2 Status> can provide considerable support to programming objects [7]. The old value space consists of the following four values: draft, final, revised, and unavailable. On the other hand, the new value space can be one of the value spaces that are provided by the IEEE software life cycle standards to specify the phase of a programming object. An example of IEEE software life cycle standard is IEEE Std. 1012 for the waterfall model [3] shown in Fig. 3 (a). The IEEE Std. 1012 has the following life cycle value space: concept phase, requirements analysis phase, design phase, implementation phase, testing phase, and installation and checkout phase. Another example of IEEE software life cycle standard is IEEE Std. 1490 for the spiral model [3] shown in Fig. 4 (b). Some of the life cycle values in IEEE Std. 1490: Final Design, Physical Design, Second Build, Test, and System Requirement. Fig. 5 illustrates the use of this data element in Programming Objects with two different value spaces.



(a)



(b)

FIG. 3: Two life cycle models [3]: (a) The Water Fall Model, and (b) The Spiral Model.

```
2.2 Status
2.2.1 Source: IEEE Std. 1012
2.2.2 Value: Testing Phase
```

(a)

```
2.2 Status
2.2.1 Source: IEEE Std. 1490
2.2.2 Value: Final Design
```

(b)

FIG. 4: Two examples of the data element <2.2 Status> in POM.

Due to the proposed change of the descriptor for category <4 Technical> to focus on the source of a programming object only, the data element <4.2 Size> needs to have the size that reflects the source. This designates that the download/bandwidth for a source can be specified accurately and consistently which is not feasible in IEEE LOM Standard [8].

The last two proposed changes in the data elements of the LOM standard are in <9.1 Purpose> and <9.2 Taxon Path>. These two data elements indicate the classification of a programming object. The data element <9.1 Purpose> describes the purposed of the classifying, and the data element <9.2 Taxon Path> describes the taxonomic path by defining the path in <9.2.2 Taxon Path> depending on a specific classification system defined in <9.2.1 Source>. The proposed change in <9.1 Purpose> is to add the new value "programming language" in its value space. This new value will add a new purpose to classify a programming object contained a source code. The new value "programming language" implies another change in the data element <9.2 Taxon Path>. If the value of the data element <9.1 Purpose> is "programming language", this implies the data element <9.2 Taxon Path> will indicate the programming language used in the programming object. An example of classification system focused on programming objects is ACM Computing Classification System [9]. Fig. 5 illustrates the use of these two data elements <9.1 Purpose> and <9.2 Taxon Path> for a programming object used "Visual C++" as a programming language.

```
9.1 Purpose: Programming Language
9.2 Taxon Path
9.2.1 ID: D, 9.2.2: Software
9.2.1 ID: D.1, 9.2.2: Programming Techniques
9.2.1 ID: D.1.7, 9.2.2: Visual Programming
9.2 Taxon Path
9.2.1 ID: D, 9.2.2: Software
9.2.1 ID: D.3, 9.2.2: Programming Language
9.2.1 ID: D.3.2, 9.2.2: Language Classification
9.2.1 ID: Subject, 9.2.2: Object-Oriented language
9.2.1 ID: Noun, 9.2.2: C++
```

FIG. 5: An example of the two data elements <9.1 Purpose> & <9.2 Taxon Path> in POM.

C. Eliminating Data Elements

This paper proposed eliminating two data elements from IEEE LOM Standard to conform to the metadata of Programming Objects, which are:

- 1.6 Coverage
- 5.7 Typical Age Range

The first data element proposed to be eliminated from the LOM standard to suit POM is <1.6 Coverage>. This data element focuses on the time, culture, geography or region to which the learning object applies. This data element can not be applied to programming objects because a programming object is supposed to be applied at any place or time depending on the right of using it defined in <6. Rights>.

Another data element proposed to be eliminated from the LOM standard to suit POM is <5.7 Typical Age Range>. This data element defines the age of the typical intended user. Therefore, this data element can not be applied to any programming object because the programming objects are not supposed to be limited to any age.

D. Adding Data Elements

This paper proposed adding four new data elements, are not in IEEE LOM Standard, to conform to the metadata of Programming Objects, which are:

- 5.x Time Unit
- 5.x Testing Strategy
- 5.x Requirement
- 6.x Reuse Type

In the LOM standard, the data element <4.7 Duration> indicates the time that a continuous learning object takes when played at intended speed. According to this principle, a new proposed data element may be added to the category <5 Education> which is <Time Unit>, which will reveal the quality of a source code. The time unit designates the number of the statements in a source code. So, the datatype of this new data element will be Number. Fig. 6 demonstrates a programming object intend to calculate the sum of the square of three numbers in an array with 5 time units. The new data element <Time Unit> can be used as a metric to measure the time and space cost.

<Time Unit> Programming Object

```
1  int numbers[3]={2,5,9};
2  int i;
3  int sum=0;
4  for (i=0; i<3; i++)
5     sum = sum + number[i]^2;
```

FIG. 6: An example of using the data element <Time Unit> in POM.

Another quality metric may be added to the category <5 Education> is <5.x Testing Techniques>. This new data element will focus on the types of the testing techniques that used to test the source code. Jorgensen [10] and Pressman [11] indicated many testing techniques. The testing techniques used in a programming object may benefit software engineering's users who measure the quality according to this metric. The source code may be tested using

more than one testing technique; therefore, this data element may be repeated in the metadata to present all of them. An example of this data element is shown in Fig. 7.

```
5.x Testing Techniques
5.x.1 Type: Black Box
5.x.2 Technique: Boundary – Worst Case
5.x.1 Type: Black Box
5.x.2 Technique: Decision Table
5.x.1 Type: White Box
5.x.2 Technique: Data Flow-Based – All Uses
```

FIG. 7: An example of the new data element <5.x Testing Techniques> in POM

The third proposed new data element is <Requirement> that may add to the category <5 Education>. This data element needs to be added due to the proposed change in the description of the category <4 Technical> to focus on the source of a programming object only. The data element <4.4 Requirement> indicates the requirements of the source of the programming objects, while the new data element <5.x Requirement> indicates the requirements to execute the programming object. Fig. 8 reveals the difference between the two data elements assuming that a programming object uses Visual C++ displayed in a browser using Flash.

```
4.1 Formant: application/zip
4.1 Formant: application/x-shockwave-flash
4.1 Formant: text/html
4.4 Requirement
4.4.1 OrComposite
4.4.1.1 Type
4.4.1.1.1 Source: LOMv1.0
4.4.1.1.2 Value: browser
4.4.1.2 Name
4.4.1.2.1 Source: LOMv1.0
4.4.1.2.2 Value: netscape communicator
4.4.1.3 Minimum Version: 6.0
4.4.1 OrComposite
4.4.1.1 Type
4.4.1.1.1 Source: LOMv1.0
4.4.1.1.2 Value: browser
4.4.1.2 Name
4.4.1.2.1 Source: LOMv1.0
4.4.1.2.2 Value: netscape communicator
4.4.1.3 Minimum Version: 6.0
5.x Requirement
5.x.1 OrComposite
5.x.1.1 Type
5.x.1.1.1 Source: ACM http://www.acm.org/class/1998
5.x.1.1.2 Value: Software
5.x.1.2 Name
5.x.1.2.1 Source: ACM http://www.acm.org/class/1998
5.x.1.2.2 Value: Visual C++
5.x.1.3 Minimum Version: 2005
```

FIG. 8: An example of the new data element <5.x Requirement> in POM

The last proposed new data element that may be added to the category <6. Right> is <Reuse Type>. Frakes, William, and Terry [12] provided a faceted classification of reuse definitions shown in Table II that can be used as a value space for this new data element. The datatype of this data element is Vocabulary, which means two values will describe this data element {source, value} as defined in the LOM standard. An example of this data element for a source code intends to be for public users shown in Fig. 9.

TABLE II
TYPES OF SOFTWARE REUSE (FRAKES AND TERRY, 1996)

Development Scope	Modification	Approach	Domain Scope	Management	Reused Entity
Internal (Private)	White Box	Generative	Vertical	Systematic (Planned)	Code
External (Public)	Black Box (verbatim)	Compositional	Horizontal	Ad Hoc	Abstract Level
	Adaptive (porting)	In-the-Small			Instance Level
		In-the-Large			Customization Reuse
		Indirect			Generic
		Direct			Source Code
		Carried Over			
		Leveraged			

```
6.x Reuse Type
6.x.1 Source: ACM, Vol. 28, No. 2, June 1996 (415-435)
6.x.2 Value: External
6.x Reuse Type
6.x.1 Source: ACM, Vol. 28, No. 2, June 1996 (415-435)
6.x.2 Value: White Box
6.x Reuse Type
6.x.1 Source: ACM, Vol. 28, No. 2, June 1996 (415-435)
6.x.2 Value: Source Code
```

FIG. 9: An example of the new data element <6.x Reuse Type> in POM

IV. CONCLUSION

In this paper, we described how Programming Object Metadata (POM) could be defined based on the previous IEEE LOM standard. Due to the similarity between Learning Objects and Programming Objects in their metadata, this paper contributed POM using IEEE LOM methodology. By adopting a formal method such as POM, we feel that software industry can develop reusable high-quality software at low costs.

REFERENCES

[1] Osterweil, Leon. (1996), "Strategic Directions in Software Quality." ACM Comput. Surv. 28(4), 738-50.

[2] Banker, R. D., R. J. Kauffman, and D. Zweig. (1993), "Repository Evaluation of Software Reuse." IEEE Trans. Softw. Eng. 19(4), 379-89.

[3] Schmidt, Michael. (2000), Implementing the Ieee Software Engineering Standards. Indianapolis, Ind: Sams.

[4] IEEE. (2003), "IEEE Software Engineering Collection on Cd-Rom".

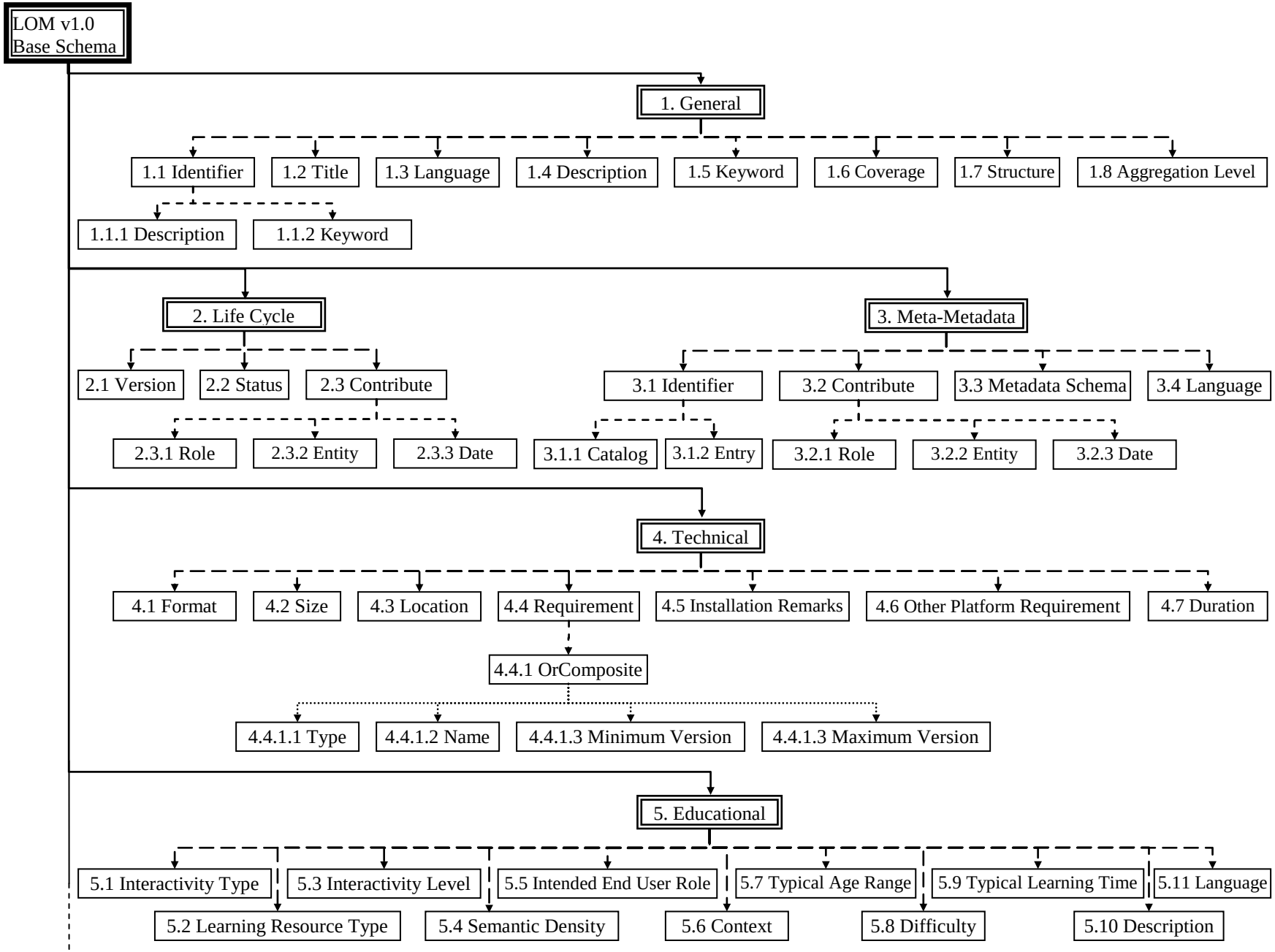
[5] IEEE. (2005), "Draft Standard for Learning Object Metadata". 2002. 1484.12.1-2002. Septemper 1, <http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf>.

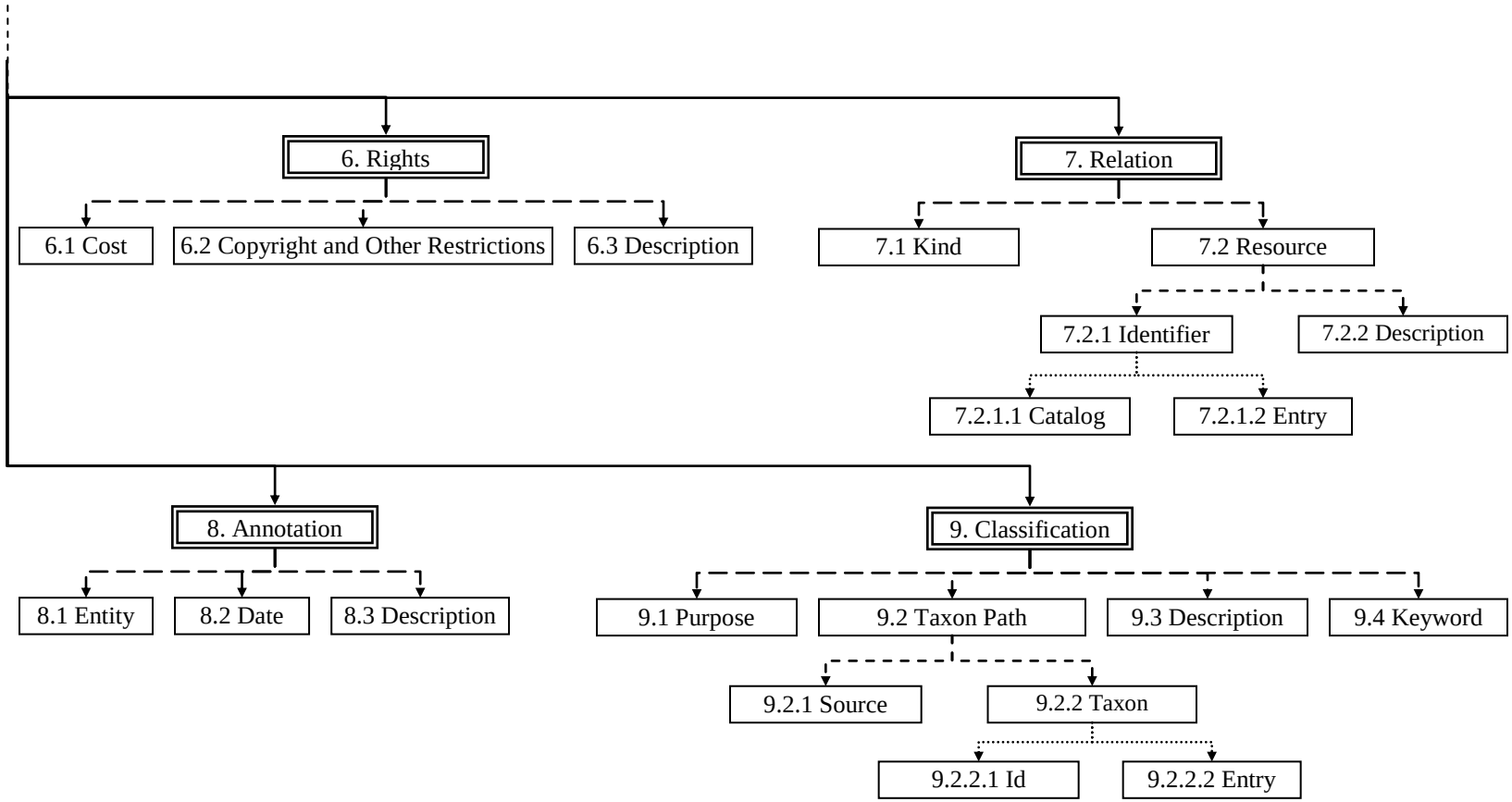
[6] Hirata, Kenji, Mamoru Ohta Yoshiyuki Takaoka, and Mitsuru Ikeda. (2001), "The Meaning of Lom and Lom Authoring Tool on Hrd." International Conference on Dublin Core and Metadata Applications.

Tokyo, Japan: National Institute of Informatics. 259-62. Ed. Keizo Oyama and Hironobu Gotoda (NII).
[7] Adrion, W. Richards, Martha A. Branstad, and John C. Cherniavsky. (1982), "Validation, Verification, and Testing of Computer Software." ACM Comput. Surv. 14(2): 159-92.
[8] Fisher, Sue, et al. (2002), "Metadata Guidelines." CanCore Initiative. Vol. Version 1.1.
[9] ACM. (1998), "Computing Classification System (CCS)". <http://www.acm.org/class/1998/>.
[10] Jorgensen, Paul C. (2002), Software Testing: A Craftsman's Approach. 2nd edn: CRC Press.
[11] Pressman, Roger S. (2005), Software Engineering: A Practitioner's Approach. 6th edn. New York, NY: McGraw-Hill.
[12] Frakes, William, and Carol Terry. (1996), "Software Reuse: Metrics and Models." ACM Comput. Surv. 28(2), 415-35.

Daniyal Alghazzawi has completed his Ph.D in Computer Science from University of Kansas in 2007, Master of Science in Teaching & Leadership in 2004 and Master of Science in Computer Science in 2003 from University of Kansas. He has worked as Web Programmer at ALTec (Advanced Learning Technologies) . Dr. Daniyal is currently Chairman of the Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University. His research interest includes Smart e-Learning, Information Security, and Video Processing.

APPENDIX A: IEEE LOM Data Elements




6. Rights
6.1 Cost
6.2 Copyright and Other Restrictions
6.3 Description
7. Relation
7.1 Kind
7.2 Resource
7.2.1 Identifier
7.2.2 Description
7.2.1.1 Catalog
7.2.1.2 Entry
8. Annotation
8.1 Entity
8.2 Date
8.3 Description
9. Classification
9.1 Purpose
9.2 Taxon Path
9.3 Description
9.4 Keyword
9.2.1 Source
9.2.2 Taxon
9.2.2.1 Id
9.2.2.2 Entry

APPENDIX B: A POM in XML Format

```
<general>
   <identifier>
      <catalog>URI</catalog>
      <entry>http://hdkaau.com/courses/cs305/lessons/p/downloads/animals.zip</entry>
   </identifier>
   <title>
      <string language="en">Create a class </string>
      <string language="ar">Class إنشاء</string>
   </title>
   <language>en</language>
   <description>
      <string language="en">In this example program, you will learn how to create a
         class, instantiate objects from the class, and access member functions and
         attributes of the class.
      </string>
      <string language="ar"> البرنامج المعروض في هذا المثال سيعلمك طريقة إنشاء Class وعمل
         object منه. كما سيعلمك طريقة طلب function أو متغير منه
      </string>
   </description>
   <keyword>
      <string language="en">Class</string>
   </keyword>
   <structure>
      <source>LOMv1.0</source>
      <value>collection</value>
   </structure>
   <aggregationLevel>
      <source>LOMv1.0</source>
      <value>2</value>
   </aggregationLevel>
</general>

<lifeCycle>
   <version>
      <string language="en">1.0</string>
   </version>
   <status>
      <source>IEEE Std. 1012</source>
      <value>Design Phase</value>
   </status>
   <contribute>
      <role>
         <source>LOMv1.0</source>
         <value>author</value>
      </role>
      <entity>
         BEGIN:VCARD
         VERSION:3.0
         N:Templeman;Julian;
         FN:Julian Templeman
         ORG: Microsoft .NET. Redmond
         END:VCARD
      </entity>
      <entity>
         BEGIN:VCARD
         VERSION:3.0
         N:Smith;Mary;
         FN:Mary Smith
         ORG: Microsoft .NET. Redmond
```

```
            END:VCARD
         </entity>
         <date>
            <dateTime>2002</dateTime>
         </date>
      </contribute>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>editor</value>
         </role>
         <entity>
            BEGIN:VCARD
            VERSION:3.0
            N:Alghazzawi;Daniyal;M.;Mr.;
            FN:Mr. Daniyal M. Alghazzawi
            ORG:King Abdulaziz University
            END:VCARD
         </entity>
         <date>
            <dateTime>2003-07</dateTime>
         </date>
      </contribute>
   </lifeCycle>

   <metaMetadata>
      <identifier>
         <catalog>URI</catalog>
         <entry>www.kau.edu.sa</entry>
      </identifier>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>creator</value>
         </role>
         <entity>
            BEGIN:VCARD
            VERSION:3.0
            N:Alghazzawi;Daniyal;M.;Mr.;
            FN:Mr. Daniyal M. Alghazzawi
            ORG:King Abdulaziz University
            END:VCARD
         </entity>
         <date>
            <dateTime>2005-10-01</dateTime>
         </date>
      </contribute>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>validator</value>
         </role>
         <entity>
            BEGIN:VCARD
            VERSION:3.0
            N:Gauch;John;Dr.;
            FN:Dr. John Gauch
            ORG:None
            END:VCARD
         </entity>
         <date>
            <dateTime>2005-11-01</dateTime>
         </date>
```

```
    </contribute>
    <contribute>
       <role>
          <source>LOMv1.0</source>
          <value>validator</value>
       </role>
       <entity>
          BEGIN:VCARD
          VERSION:3.0
          N: Saiedian; Hossein;Dr.;
          FN:Dr. Hossein Saiedian
          ORG:None
          END:VCARD
       </entity>
       <date>
          <dateTime>2003-11-30</dateTime>
       </date>
    </contribute>
    <metadataSchema>LOMv1.0</metadataSchema>
    <language>en</language>
</metaMetadata>

<technical>
    <format>application/x-shockwave-flash</format>
    <format>text/html</format>
    <size>9099~20000</size>
    <location>http://hdkaau.com/courses/cs305/lessons/program/prog2/index.htm</location>
    <requirement>
       <orComposite>
          <type>
             <source>LOMv1.0</source>
             <value>browser</value>
          </type>
          <name>
             <source>LOMv1.0</source>
             <value>netscape communicator</value>
          </name>
          <minimumVersion>6.0</minimumVersion>
          <maximumVersion/>
       <orComposite>
          <type>
             <source>LOMv1.0</source>
             <value>browser</value>
          </type>
          <name>
             <source>LOMv1.0</source>
             <value>ms-internet explorer</value>
          </name>
          <minimumVersion>5.5</minimumVersion>
          <maximumVersion/>
       </orComposite>
    </requirement>
    <installationRemarks>
       <string language="en">Unzip the zip file and launch index.html in your web
          browser.</string>
       <string language="ar">فك ضغط الملف المضغوط بواسطة zip ثم أفتح الملف index.html من
          متصفحك.</string>
    </installationRemarks>
    <otherPlatformRequirements>
       <string language="en">Flash 5.0 or greater</string>
    </otherPlatformRequirements>
    <duration/>
</technical>
```

```
<Educational>
   <interactivityType>
      <source>LOMv1.0</source>
      <value>mixed</value>
   </interactivityType>
   <learningResourceType>
      <source>LOMv1.0</source>
      <value>narrative text</value>
   </learningResourceType>
   <interactivityLevel>
      <source>LOMv1.0</source>
      <value>high</value>
   </interactivityLevel>
   <semanticDensity>
      <source>LOMv1.0</source>
      <value>very low</value>
   </semanticDensity>
   <intendedEndUserRole>
      <source>LOMv1.0</source>
      <value>learner</value>
   </intendedEndUserRole>
   <intendedEndUserRole>
      <source>LOMv1.0</source>
      <value>manager</value>
   </intendedEndUserRole>
   <context>
      <source>LOMv1.0</source>
      <value>training</value>
   </context>
   <difficulty>
      <source>LOMv1.0</source>
      <value>very difficult</value>
   </difficulty>
   <typicalLearningTime/>
   <timeUnit>1</timeUnit>
   <description>
      <string language="eng">This resource can be very effective when utilized as a
         generator for discussion in a grade two classroom. However, it can also be
         used for individual writing assignments for grade 4 students, or for grade 5
         students who are challenged.</string>
   </description>
   <language>en</language>
   <requirement>
      <orComposite>
         <type>
            <source>ACM http://www.acm.org/class/1998</source>
            <value>Software</value>
         </type>
         <name>
            <source>ACM http://www.acm.org/class/1998</source>
            <value>Visual C++</value>
         </name>
         <minimumVersion>6.0</minimumVersion>
         <maximumVersion/>
      </orComposite>
   </requirement>
</Educational>

<right>
   <cost>
      <source>LOMv1.0</source>
      <value>no</value>
```

```
    </cost>
    <copyrightAndOtherRestrictions>
       <source>LOMv1.0</source>
       <value>yes</value>
    </copyrightAndOtherRestrictions>
    <reuseType>
       <source>ACM Computer Surveys, Vol. 28, No. 2, June 1996 (415-435)</source>
       <value>External</value>
    </reuseType>
    <reuseType>
       <source>ACM Computer Surveys, Vol. 28, No. 2, June 1996 (415-435)</source>
       <value>White Box</value>
    </reuseType>
    <reuseType>
       <source>ACM Computer Surveys, Vol. 28, No. 2, June 1996 (415-435)</source>
       <value>Source Code</value>
    </reuseType>
    <description>
       <string language="en">Read the copyright section in the website.</string>
       <string language="ar">أقرأ حقوق النسخ التي في الموقع</string>
    </description>
</right>

<relation>
    <kind>
       <source>LOMv1.0</source>
       <value>ispartof</value>
    </kind>
    <resource>
       <identifier>
          <catalog>URI</catalog>
          <entry>http://msdn.microsoft.com/library/default.asp?url=/library/en-
             us/cpref/html/frlrfsystemgcmemberstopic.asp</entry>
       </identifier>
       <description>
          <string language="en">The class is a member of GC Class.</string>
          <string language="ar">GC Class يكون عضو في class ال هذا</string>
       </description>
    </resource>
</relation>

<annotation>
    <entity>
       BEGIN:VCARD
       VERSION:3.0
       N:Alghazzawi;Daniyal;M.;Mr.;
       FN:Mr. Daniyal M. Alghazzawi
       ORG:King Abdulaziz University
       END:VCARD
    </entity>
    <date>
       <dateTime>2004-01-15T12:00:00.0</dateTime>
    </date>
    <description>
       <string language="en">I highly recommend the Web page for any novice learning
          class.</string>
       <string language="ar">classال في للمبتدئين الإنترنت هذه صفحة بشدة أنصح أنا</string>
    </description>
</annotation>

<classification>
    <purpose>
       <source>~~LOMv1.0~~</source>
```

```
      <value>programming language</value>
   </purpose>
   <taxonpath>
      <source>
         <string language="en">ACM http://www.acm.org/class/1998</string>
      </source>
      <taxon>
         <entry>
            <id>D</id>
            <string language="en">Software</string>
         </entry>
      </taxon>
      <taxon>
         <entry>
            <id>D.3</id>
            <string language="en">Programming Languages</string>
         </entry>
      </taxon>
      <taxon>
         <entry>
            <id>D.3.2</id>
            <string language="en">Language Classification</string>
         </entry>
      </taxon>
      <taxon>
         <entry>
            <id>Subject</id>
            <string language="en">Object-oriented languages</string>
         </entry>
      </taxon>
      <taxon>
         <entry>
            <id>Noun</id>
            <string language="en">C++</string>
         </entry>
      </taxon>
   </taxonpath>
   <taxonpath>
      <source>
         <string language="en">ACM http://www.acm.org/class/1998</string>
      </source>
      <taxon>
         <entry>
            <id>D.1</id>
            <string language="en">Programming Techniques</string>
         </entry>
      </taxon>
      <taxon>
         <entry>
            <id>D.1.7</id>
            <string language="en">Visual Programming</string>
         </entry>
      </taxon>
   </taxonpath>
   <keyword>
         <string language="en">class</string>
   </keyword>
   <keyword>
         <string language="en">object-oriented</string>
   </keyword>
</classification>
```