Meeting Minutes 29 March 2001

Metadata Background

Author:
Melanie Nelson (Physiome Sciences Inc.)
Contributor:
Warren Hedley (Bioengineering Research Group, University of Auckland)

1 Introduction

This document provides background information about metadata, its role in the CellML project, and the possible implementations of metadata in CellML.

1.1 Need for Metadata in CellML

Metadata is usually defined as "data about data". It is the supporting information that provides context to a resource. In CellML, the model (i.e., the structure and mathematics of the model) is the resource. Information that puts the model into the larger scientific context is metadata. Metadata in CellML includes information such as the literature reference that supports the model, the identity of the creator(s) of the model, and the species for which the model is relevant.

The CellML metadata project needs metadata for two primary reasons:

- It will be difficult to reuse other people's models and components without metadata to provide the scientific context for these objects. A modeller considering reusing someone else's model component will need to know things such as: what biological entity the component represents, for which species the component is relevant, and when the component was created and last modified (to help determine whether it is likely to incorporate the most recent experimental results).
- As the number of models and components grows, metadata will provide the only scalable method for locating particular models and components. Experience in other biological fields shows that as a field grows, powerful search techniques are needed to enable researchers to find relevant resources. These search techniques require structured metadata.

Metadata in CellML can be used in many different ways, such as:

- To support searches of a model repository (or at least to make it possible to automate loading of a database that will support such searches).
- To enable automatic discovery of models published on remote websites, such as laboratory websites.
- To allow the documentation for a model to be kept in the same document as the model itself, which will keep the documentation from becoming obsolete as work continues on the model.

The metadata structure should be flexible and extensible, because it is almost certain that we have not thought of all possible uses of CellML metadata.

1.2 The Larger Metadata Picture

Metadata has become a bit of a buzzword lately. This is because people are starting to realise that we cannot get the maximum use out of the information stored on the web without metadata. It is currently not particularly easy to find a specific piece of information on the web, and once you have found the information, it is not easy to determine whether or not you should trust it. Metadata can address both of these problems. Therefore, there is a push to begin to incorporate metadata into web resources. Tim Berners-Lee has been particularly active in pushing for a "semantic web", in which resources on the web would include the semantic information necessary to allow machines to *understand* (not just read) them. The W3C has set up a semantic web activity[1]. Some software projects, such as Mozilla[2], have begun trying to take advantage of the metadata that is currently available about web resources.

The "semantic web" vision is one of the future, and not of today. Several projects are beginning to take tentative steps towards realising Tim Berners-Lee's dream, but success is by no means certain. The library science community is leading the way in implementing metadata. A consequence of this is that the tools being provided for handling metadata on the web (such as the Resource Description Framework[3], or RDF) have come from the knowledge management community. Like any academic discipline, that community has its own jargon, which can be a hindrance to the rest of us when we try to understand and use these tools. However, several projects are now using RDF, and a variety of tools have been created for it. These will be discussed in Section 3.2.

None of the problems faced by the nascent metadata community are insurmountable. It seems very likely that something resembling the "semantic web" will come into existence, if for no reason other than the importance of the problem it is attempting to address. Therefore, we should at least consider how we can make metadata in CellML compatible with the semantic web activity.

2 Metadata in CellML

The initial step in incorporating metadata into CellML was to determine what sorts of information modellers might want to store about their models, and what sort of information software developers might find useful to be able to store. This was done as part of the requirements gathering for version 1.0 of CellML. A list of the metadata requirements for CellML is included in the requirements document[4].

The necessary metadata can be split into three broad categories:

- Metadata that can be mapped easily onto the Dublin Core elements (see Section 3.1).
- Literature citations, which we might be able to handle using an existing standard such as DocBook[5] or the Object Mangement Group's Bibliographic Query Service[6].
- Metadata that is specific to biology and/or CellML, an implementation of which we will probably need to develop ourselves.

[1] http://www.w3.org/2001/sw/
[2] http://www.mozilla.org/
[3] http://www.w3.org/RDF/
[4] http://www.cellml.org/private/fundamentals/requirements.html
[5] http://www.docbook.org
[6] http://www.omg.org/homepages/lsr/

3 Existing Standards for Metadata

There are two existing metadata standards that warrant our attention: The Dublin Core element set[7] (which defines standard types of metadata) and the Resource Description Framework[8] (which defines a means to specify metadata).

3.1 The Dublin Core Metadata

The Dublin Core Metadata Initiative[9] came out of the library science community. It is an attempt to identify metadata that is common across a variety of types of resources, and provide a standard way to refer to this metadata. The Dublin Core group is primarily concerned with identifying the common metadata objects (such as "creator" or "copyright"), rather than specifying how the metadata should be stored. However, they have released a document entitled Using Dublin Core[10] that gives guidelines for storing Dublin Core metadata in HTML and XML/RDF. The Dublin Core documents most relevant for the CellML project are:

- The Dublin Core metadata element set[11] (the core elements)
- The Dublin Core qualifiers[12] (qualifiers that extend the core elements)
- A proposal called Guidance on expressing the Dublin Core within the Resource Description Framework (RDF)[13] (not published by dublincore.org, so of uncertain status)

Why should we care about the Dublin Core? The Dublin Core metadata element set is probably the most widely used set of metadata elements. Most projects listed on the RDF Project list[14] use the Dublin Core metadata elements in some way or another. Using a standard vocabulary wherever possible increases the chances that our metadata will be accessible to general purpose metadata tools. For instance, someone could use the RDF Crawler[15] to discover some basic information about CellML resources. They are more likely to be able to interpret metadata stored in a common vocabulary such as the Dublin Core element set. There are also a growing number of metadata tools designed to work with the Dublin Core element set. For instance, the DC-Assist[16] tool provides a browsable set of descriptions and examples for the Dublin Core elements and qualifiers. See the Dublin Core's tool section[17] for a list of other tools.

3.2 The Resource Description Framework

The Resource Description Framework[18] (RDF) is a W3C recommendation for storing and exchanging metadata. It specifies a general data model for metadata and provides an XML syntax for storing metadata in this data model. The companion RDF schema recommendation[19] specifies a syntax for defining the detailed data model for a specific set of metadata. It is expected and encouraged that people will draw from a variety of RDF schema when marking up the metadata about their documents. Because all metadata stored in RDF

[7] http://dublincore.org/documents/dces/
[8] http://www.w3.org/RDF/
[9] http://dublincore.org
[10] http://dublincore.org/documents/usageguide
[11] http://dublincore.org/documents/dces
[12] http://dublincore.org/documents/dcmes-qualifi ers
[13] http://www.ukoln.ac.uk/metadata/resources/dc/datamodel/WD-dc-rdf/
[14] http://www.w3.org/RDF/#projects
[15] http://ontobroker.semanticweb.org/rdfcrawl/index.html
[16] http://www.ukoln.ac.uk/metadata/dcassist
[17] http://dublincore.org/tools/
[18] http://www.w3.org/RDF/
[19] http://www.w3.org/TR/2000/CR-rdf-schema-20000327/

uses the same basic data model (described below), the various vocabularies and schema that people develop are all interoperable. In fact, an RDF schema can be used to define a new metadata element that is a subtype of an element defined in a different schema.

RDF came from the knowledge representation community, and therefore has a frame of reference that is quite different from more computer science driven standards such as XML itself. The basic data model of RDF is a directed labelled graph, which can equivalently be expressed as a "three-tuple" (or triple). Readers wanting a formal definition of this model should refer to Section 5 of the RDF Model and Syntax recommendation[20]. What follows here is an informal explanation.

The thing about which you want to store metadata is the *resource*. The type of metadata you are storing is the *property*. The value of the metadata is either another resource (which can itself have metadata) or a *literal*. An RDF three-tuple contains a predicate, subject, and object. The subject is a resource, the predicate is a property, and the object is the literal or resource that is the value of the metadata. RDF also provides grouping mechanisms that allow you to unambiguously specify the correct interpretation of multivalued objects. There are three types of grouping containers: a bag (an unordered group of objects), sequence (an ordered group of objects), and alternative (a group of objects that specify alternative values for a single-valued object).

Unlike XML (which restricts the allowed syntax of the data in a resource), RDF attempts to encode the semantics, or meaning, of the data in a resource in a machine-understandable manner. The RDF elements are all concerned with identifying the subject, predicate, and object for each metadata statement. With this information, RDF parsers can construct directed graphs and 3-tuples of the metadata, which is itself useful. The 3-tuples can be thought of as attribute-value pairs about the subject, and simply presenting these attribute-value pairs to the end user is a useful thing to do with metadata.

The information in an RDF schema allows software to do more meaningful things with the metadata. For instance, an RDF schema could define a type of metadata called "author", and declare it to be a subclass of the Dublin Core type "creator". RDF-savvy software could then infer that an author has all of the properties of a creator. An RDF schema can also limit the types of resources to which a particular property can be applied. For instance, a "hair_color" property would probably not be applicable to a resource of type "car". Note that the RDF Schema recommendation only has "candidate" status, and there don't seem to be many implementations of it yet.

Why should we care about RDF? The simple answer is that it is a W3C recommendation. It also provides a robust and flexible framework in which metadata can be stored, and allows people working in diverse subject areas to use each other's metadata in an interoperable way. However, it has a steep learning curve, due in large part to a difficult specification. Furthermore, because it is encoding the "subject-predicate-object" information for each piece of metadata, RDF is a bit verbose (see Section 4 for a defense of this verbosity).

Perhaps the most powerful argument for paying attention to RDF is that people are beginning to use it. The RDF home page at the W3C[21] has a list of projects and tools using RDF. Dave Beckett also has a list of RDF resources[22], which includes many tools and projects. The following is a list of some of the most interesting projects from these lists:

- RDF Crawler[23]: A Java based tool that downloads RDF from the internet and constructs a knowledge-base.
- Jena[24]: a Java API for manipulating RDF models.

[20] http://www.w3.org/TR/REC-rdf-syntax/
[21] http://www.w3.org/RDF/
[22] http://www.ilrt.bris.ac.uk/discovery/resources/
[23] http://ontobroker.semanticweb.org/rdfcrawl/index.html
[24] http://www-uk.hpl.hp.com/people/bwm/rdf/jena

- SiRPAC[25]: a set of Java for parsing RDF. There is also an online RDF parsing service, which can return the 3-tuples and directed graph from RDF.
- Mantis[26]: a toolkit for developing catalogue systems.

There are at least two XSLT RDF parsers, as well as tools using Java, C, Perl, Tcl, and Prolog. Anyone implementing CellML metadata support should look through these lists and see if there is anything useful there.

4 Verbosity of RDF

A common criticism of RDF is that it is much more verbose than an "XML-only" language to store the same information. The content/markup ratio seems quite low. However, this misses the fact that the information content of an RDF statement is more than just the content of the RDF elements. It is the set of 3-tuples that these elements encode, as well as additional structure for the metadata content. These 3-tuples provide some very basic machine-understandable *semantics* of the information, whereas equivalent XML would only provide machine-understandable *syntax*. The information that supports the semantic interpretation of metadata must be stored somewhere if processing software is going to do anythign reasonable with the metadata. In XML, this semantic information is stored in the specification of the XML vocabulary, and therefore must also be coded into the processing applications' logic if the metadata is going to be machine-understandable as opposed to simply machine-readable. In RDF, more of the semantic information resides in the document itself and in the (machine-understandable) RDF schema, making the metadata machine-understandable to any RDF-enabled processor.

For instance, think about the simple case where we want to store the following information: "the book's author is John Doe". We could define an XML syntax for this, as shown in Figure 1. We look at this and know that it is representing the metadata about the author of a book. However, a computer could not automatically understand that. The fact that the **<author>** element is contained in the **<book>** element might mean that the information in the **<author>** element is a property of the thing represented by the **<book>** element. However, it could mean many other things, too. For instance, it might mean that the **<author>** element represents an object that is contained in the object represented by the **<book>** element (think about how one might represent the relationship between and engine and its parts in XML).

```
<book>
  <author>John Doe</author>
</book>
```

FIGURE 1: Example XML to store author metadata about a book.

If we store the same information in RDF (see Figure 2), the fact that the author information is a property of the book object is explicit, because this is defined by the RDF data model. Additional semantics about the relationship between an author and a book could be provided in the RDF schema for the vocabulary indicated by the "s" namespace.

In fact, we can go one step further, and use a standard RDF vocabulary, such as the Dublin Core, to maximize the utility of our metadata for other applications. This is shown in Figure 3.

[25] http://www.w3.org/RDF/Implementations/SiRPAC
[26] http://orc.rsch.oclc.org:6464

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:s="my_example_namespace_uri">
  <rdf:Description about="book_identifier">
    <s:author>John Doe</s:author>
  </rdf:Description>
</rdf:RDF>
```

FIGURE 2: Example RDF to store author metadata about a book.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.0/">
  <rdf:Description about="book_identifier">
    <dc:creator>John Doe</dc:creator>
  </rdf:Description>
</rdf:RDF>
```

FIGURE 3: Example RDF that stores author metadata using the Dublin Core creator element.

The basic RDF data model also provides useful grouping semantics. Consider the example of a resource that has three creators. There are three interpretations of the meaning of this:

- The three creators created the resource together. All three are considered to be equally responsible for the resource.
- The three creators created the resource together. One is considered to be the “primary author”, another to be the “second author”, and the third to be the “third author”.
- The three creators worked on the resource, but may have worked on it independently (for instance, at different times).

We could define an XML vocabulary that allows us to differentiate between these three options. However, the meaning of the elements and attributes that allow this differentiation would have to be coded into processing applications’ logic. It would not be enough to simply parse the XML. If we had another type of metadata that needed the same set of options (for instance, editors), we would either have to store the differentiation twice, or generalize the XML elements that represent the two types of metadata.

We can also differentiate between these three options in RDF. The first option is stored in an construct called a *bag*. The second option is stored in a construct called a *sequence*. The third option is the default assumption, and is represented by simply repeating the RDF element that stores the creator metadata. The meaning of these constructs is specified by RDF, and an application could understand which of the three possibilities is correct for a particular instance of metadata simply by parsing the RDF. If we had another type of metadata that needed the same set of options, we could use the same three constructs to represent the different options.

5 Storing Metadata in CellML

We have three main options for storing metadata in CellML:

- Use RDF, and follow all recommendations for existing storing existing standards such as the Dublin Core elements and vCard where possible (we'd still need to create our own RDF schema for some of our metadata).
- Use RDF, but create our own RDF schema for all metadata.
- Create our own XML encoding for our metadata.

5.1 The CellML Philosophy

The decision about how to implement metadata in CellML needs to take into account:

- The overall CellML philosophy of re-using other people's standards as much as possible.
- The need for a flexible, robust metadata system that can unambiguously store the majority of metadata that modellers might need.
- The ease of implementing software that uses the metadata. Metadata is no use if no one implements it.
- The "elegance" of the XML design. This includes considerations about the conciseness, clarity, and consistency of the XML.
- The probability that our metadata solution will be interoperable with existing and future general purpose metadata tools and projects.

5.2 Comparison of Possible Implementations

5.2.1 RDF Using All Possible Recommendations

Figure 4 shows an example of CellML creator, creation date, annotation, biological entity metadata for a fictitious model component stored in RDF, using the Dublin Core draft recommendation for storing qualified Dublin Core metadata in RDF[27]. Note that we have not worked out how to store information about people, so only the creators' names are provided. A more structured set of information about the creators could be included. The encoded metadata is:

- The component was created by Betty Smith and Al Jones, working together. They have equal status (i.e., neither is the primary author).
- The component was created on 5 October 2000.
- Betty Smith created a limitation annotation on 5 October 2000, with content of "This component is only valid for temperatures above 20 degrees C".
- This component represents a biological entity with the name "calmodulin", that is identified by the SWISS-PROT database entry `CALM_HUMAN`. The GeBank database entry `P02593` is an alternative identifier for this entity.

The advantages of this approach are:

- It takes full advantage of all relevant standards and recommendations.

[27] http://www.ukoln.ac.uk/metadata/resources/dc/datamodel/WD-dc-rdf

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:cmeta="http://www.cellml.org/2001/03/metadata#"
    xmlns:dc="http://purl.org/dc/elements/1.0/"
    xmlns:dcq="http://purl.org/dc/qualifiers/1.0/">

  <rdf:Description about="a_cellml_component">
    <dc:creator>
      <rdf:Bag>
        <rdf:li>Betty Smith</rdf:li>
        <rdf:li>Al Jones</rdf:li>
      </rdf:Bag>
    </dc:creator>
    <dc:date>
      <rdf:Description>
        <dcq:dateScheme>W3C-DTF</dcq:dateScheme>
        <dcq:dateType>created</dcq:dateType>
        <rdf:value>2000-10-05</rdf:value>
      </rdf:Description>
    </dc:date>
    <cmeta:annotation>
      <rdf:Description>
        <cmeta:annotation_type>limitation</cmeta:annotation_type>
        <rdf:value>
        This component is only valid for temperatures above 20 degrees C
        </rdf:value>
        <dc:creator>Betty Smith</dc:creator>
        <dc:date>
          <rdf:Description>
          <dcq:dateScheme>W3C-DTF</dcq:dateScheme>
          <dcq:dateType>created</dcq:dateType>
          <rdf:value>2000-10-05</rdf:value>
          </rdf:Description>
        </dc:date>
      </rdf:Description>
    </cmeta:annotation>
    <cmeta:bio_entity>
      <rdf:Description>
        <dc:title>calmodulin</dc:title>
        <cmeta:identifier>
          <rdf:Description>
            <cmeta:identifier_scheme>SWISS-PROT</cmeta:identifier_scheme>
            <rdf:value>CALM_HUMAN</rdf:value>
          </rdf:Description>
        </cmeta:identifier>
        <cmeta:identifier>
          <rdf:Description>
            <cmeta:identifier_type>alternative</cmeta:identifier_type>
            <cmeta:identifier_scheme>GenBank</cmeta:identifier_scheme>
            <rdf:value>P02593</rdf:value>
          </rdf:Description>
        </cmeta:identifier>
      </rdf:Description>
    </cmeta:bio_entity>
  </rdf:Description>
</rdf:RDF>
```

FIGURE 4: Example CellML metadata in RDF, using the draft recommendation for encoding qualified Dublin Core elements in RDF. Qualifications of CellML-specific metadata elements (those in the `cmeta` namespace) are implemented in a manner consistent with the recommendations for qualified Dublin Core elements.

- The use of RDF and the Dublin Core increases the probability that this metadata will be interoperable with general purpose metadata tools.
- The use of RDF increases the probability that any extensions the core CellML metadata set will be interoperable with CellML metadata compliant processors.

The disadvantages of this approach are:

- The RDF is verbose, and it is not immediately obvious what it represents.

5.2.2 RDF Using an Entirely New Schema

Figure 5 shows an example of the same CellML creator, creation date, annotation, biological entity metadata stored in RDF, but using only a CellML-specific schema. We can still use the Dublin Core elements and qualifiers, but have devised our own system for encoding them in RDF. Note that we have not worked out how to store information about people, so only the creators' names are provided. A more structured set of information about the creators could be included. The encoded metadata is the same as for Figure 4.

The advantages of this approach are:

- The RDF is less verbose.
- The use of RDF increases the probability that any extensions the core CellML metadata set will be interoperable with CellML metadata compliant processors.

The disadvantages of this approach are:

- It is less likely to be interoperable with general pupose metadata tools, since a tool would have to consult the RDF schema to discover that some of the elements are from the Dublin Core.

5.2.3 A New XML Application

Figure 6 shows an example of the same CellML creator, creation date, annotation, biological entity metadata stored in a new XML application, which is given the `cmeta` namespace. We can still udraw from the Dublin Core elements and qualifiers, but have devised our own system for encoding them in XML. Note that we have not worked out how to store information about people, so only the creators' names are provided. A more structured set of information about the creators could be included. The encoded metadata is the same as for Figure 4.

The advantages of this approach are:

- It is much less verbose.
- It would probably be easier for CellML software to implement metadata facilities if this method is used.

The disadvantages of this approach are:

- It is not interoperable with general pupose metadata tools.
- A modeller wishing to define an extension to the metadata would do so in his or her own XML namespace. Their XML would not need to bear any resemblance to the metadata XML we developed, decreasing the chances that CellML processing software would be able to do anything reasonable with the extension metadata.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:cmeta="http://www.cellml.org/2001/03/metadata#">

  <rdf:Description about="a_cellml_component">
    <cmeta:author>
      <rdf:Bag>
        <rdf:li>Betty Smith</rdf:li>
        <rdf:li>Al Jones</rdf:li>
      </rdf:Bag>
    </cmeta:author>
    <cmeta:date
          cmeta:dateScheme="W3C-DTF"
          cmeta:dateType="created">2000-10-05
    </cmeta:date>
    <cmeta:annotation
        cmeta:annotation_type="limitation">
      This component is only valid for temperatures above 20 degrees C
      <cmeta:author>Betty Smith</cmeta:author>
      <cmeta:date
            cmeta:dateScheme="W3C-DTF"
            cmeta:dateType="created">2000-10-05
      </cmeta:date>
    </cmeta:annotation>
    <cmeta:bio_entity>
      <rdf:Description>
        <cmeta:name>calmodulin</cmeta:name>
        <cmeta:identifier
              cmeta:identifier_scheme="SWISS-PROT">CALM_HUMAN
        </cmeta:identifier>
        <cmeta:identifier
              cmeta:identifier_scheme="GenBank"
              cmeta:identifier_type="alternative">P02593
        </cmeta:identifier>
      </rdf:Description>
    </cmeta:bio_entity>
  </rdf:Description>
</rdf:RDF>
```

FIGURE 5: Example CellML metadata in RDF, using a CellML-specifi c schema.

```
<component>
  <metadata xmlns="http://www.cellml.org/2001/03/metadata#">
    <author_group>
      <author>Betty Smith</author>
      <author>Al Jones</author>
    </author_group>
    <creation_date scheme="W3C-DTF">2000-10-05</creation_date>
    <annotation
        type="limitation">
      This component is only valid for temperatures above 20 degrees C
      <author>Betty Smith</author>
      <creation_date scheme="W3C-DTF">2000-10-05</creation_date>
    </annotation>
    <bio_entity>
      <name>calmodulin</name>
      <identifier
            scheme="SWISS-PROT">CALM_HUMAN
      </identifier>
      <identifier
            scheme="GenBank"
            type="alternative">P02593
      </identifier>
    </bio_entity>
  </metadata>
</component>
```

FIGURE 6: Example CellML metadata in a new XML application.