Specification

1. Overview

BrainML describes a format and set of conventions for defining data models for neuroscience data exchange. It also specifies formats for representing data conforming to these models. These two classes of documents are illustrated in the architecture diagram as layers in a multi-level structure. Models use the XML Schema language and define models and components of models for neuroscience data. Instance documents use the XML language and describe data or content-based queries for data. The markup tags instance documents contain and the arrangements they can occur in is determined by the model schemas.

BrainML is the centerpiece of a larger effort to enable the creation of neuroscience data resources and maximize the utility of the data they provide. BrainML-X is a set of protocols for the query and transfer of BrainML data between applications and across networks. The Neurodatabase Construction Kit is an open-source Java software library for building BrainML-aware applications. This document focuses on describing the BrainML markup language itself.

Below, we first outline the meta model in which BrainML data models are described - that is, the types of definitional structure that are available. Then we describe BrainML model documents and instance documents in more detail, and finally we discuss the various special facilities provided as part of the BrainMetaL and BrainML base schemas.

2. Model Structure

The data model is object-oriented, consisting of entities with fields, which may be of primitive type: *free text*, *number*, *controlled vocabulary*, *unit of measure*, or *data block*. The last three are all special types that are discussed later in this document. *Controlled vocabulary* consists of hierarchical lists of defined terms, which are currently independent but may be linked to the UMLS or any other controlled system. *Unit of measure* fields provide for standardized, metadata enriched tagging of SI and other units for numeric data. *Data blocks* support structured, multi-dimensional data.

To accomodate structure and support component reuse, three types of relationships between entities are permitted: (single-) *inheritance*, *aggregation* (container-contained), and *m:n link*. For *inheritance*, fields may be added, or fields may be given default or required values. In

aggregation and *m:n relations*, a subentity may always be substituted for an entity. A given entity may be marked as *abstract*, either globally, or in the context of a particular usage. Abstract entities do not hold data directly.

3. Model Documents

Model documents are written in XML Schema. BrainML.org both accepts and publishes schemas in either regular XML syntax or in an alternative compact format. In this document we will provide examples in both formats.

Model documents are structured in a layered fashion, leveraging the capability of XML Schema to support the definition of reusable object-like components. Each component may reference others to build its definition. A set of very basic definitions useful in a variety of contexts, and not specific to neuroscience is provided in the BrainMetaL package. The BrainML "base" package adds additional basic definitions that *are* specific to neuroscience.

Component reuse is best clarified by an example. Among the basic set of data structures provided in BrainML, entities are defined for some common data set types encountered in neurophysiology: `'time series trace'`, `'spike train trace'`, and `'histogram trace'`. Each of these has its specific characteristics, but all contain labels, sequence numbers, and data blocks. These are only defined once, in an abstract `'trace'` entity that the specific trace types all inherit from. Likewise, the data block content item itself is defined only once and incorporated by reference.

Reference between model components is aided by a *namespace* mechanism. Each model declares an identifier, called a namespace, that is unique to itself. Elements inside the model can then be unambiguously referred to by combining the namespace with the name of the element. Thus, it is not a conflict if two different models each define their own element called "neuron". One of these may be defining a model for invertebrate studies which describe neurons in different ways the vertebrate work. The invertebrate one would declare a namespace of "invertebrate" (or, more likely, something more specific). Thus, a third model could specify whether it reuses the 'neuron' component from "invertebrate" or "vertebrate" (for example), or it could even mix uses of both, without conflict.

Note:

XML Schema is a complex language that attempts to accomodate both inheritance / aggregation structure derived from object-oriented programming and grammar-based structure deriving from the XML and SGML markup language traditions. The metamodel that BrainML provides simplifies this to make models more manageable and comprehensible. Therefore not every XML schema is a valid BrainML schema, and certain conventions should be used. These are discussed further below. In the future an authoring tool incorporating the constraints will be provided so that schemas need not be edited directly.

4. Instance Documents

Instance documents are written in XML that conforms to BrainML model documents (schemas).

An XML document generally conforms *one* schema, and thus *one* BrainML model, however schemas and BrainML models may incorporate others in whole or in part by reference.

The XML document states which primary schema it conforms to using the following syntax:

```
<?xml version="1.0"?>
<submission xmlns="urn:bml/brainml.org:uni.edu/OwlNeurophys/1">
  ...
</submission>
```

Here, `"submission"` is just the top-level tag for this particular model; it could be something different for other models. The schema is specified indirectly by the URN in the `xmlns` attribute. The structure of the URN follows BrainML conventions and is used by BrainML-aware software to find the appropriate schema. In particular, the `"urn:"` indicates this is a URN, the `"bml/brainml.org"` indicates that this is a BrainML schema hosted at brainml.org, and the `"uni.edu/OwlNeurophys/1"` indicates the originating institution, model name, and version. BrainML aware servers (such as `brainml.org`) know how to resolve a request for a schema given just this portion. A URL of the form `http://brainml.org/isr.do?namespace="urn:bml/brainml.org:uni.edu/OwlNeuroph` will return the appropriate schema, inside of which appropriate references will be given for retrieving additional component schemas directly.

Note:

As a special case, leaving the final version number off of the URN returns the latest version of the schema.

5. Data Containers

BrainML instance documents contain raw data (or references to raw data) from neuroscience experiments, plus *metadata* that describes the experiment -- what preparation and protocol was used, where / what type of recording was done, who collected the data, etc.. The latter is used to index the experiment so that its data may be located in a content-based search.

BrainML is designed to support both the data and metadata portions of neuroscience data documents. While custom structures may be defined within BrainML for both of these, data is expected to be less diverse than metadata. Spike trains and voltage traces, for example, can be collected in many circumstances but remain lists of floating-point values. BrainML therefore provides a set of tags defining a basic set of *data containers* that we hope will be reusable across many data models. These come in two chunks, the ***dataset*** definitions in BrainMetaL, and the ***experiment-view-trace*** hierarchy in BrainML base.

5.1. Dataset Containers

Data set containers support lists and single- and multi-dimensional arrays of numerical and text data, in several formats. Supported data types are: "integer", "decimal", and "string". Integer and decimal types can be of arbitrary size and precision, however BrainML applications MAY process these as 4-byte signed integers and 8-byte IEEE doubles respectively. Strings are in whatever encoding the XML document itself declares, which is usually UTF-8, so can support non-English text and even non-Roman alphabets.

The data can be formatted in one of four different ways: as ASCII in comma-separated or similar text-delimited format, as ASCII structured by XML tags, as a binary, Base64-encoded block, or as a reference to an external resource.

In terms of the XML representation, BrainMetaL defines the `<dataset>` abstract element, together with one concrete "subclass" elements for each of the four data formats. Wherever a data model specifies use of the abstract element, any of the concrete instances can be used. These four concrete elements are `<datasetC>`, `<datasetX>`, `<datasetB>`, and `<datasetR>`.

Each of these takes '`type`' and '`dimensions`' attributes. The `<datasetC>` and `<datasetR>` elements take additional attributes relevant to their format as well. See the BrainMetaL definitions for a more complete description.

5.2. Experiment-View-Trace Hierarchy

The Experiment-View-Trace definitions in the BrainML base distribution define higher level structure than the Dataset elements, but in a generic fashion applicable to hopefully a wide variety of data models. In particular, a concrete element `<experiment>` is defined, which may contain one or more `<view>` elements, each of which may contain one or more `<trace>` elements.

6. Controlled Vocabulary

Fields of BrainML entities may specify a particular set of values that may be used. These values, called *controlled vocabulary* items, are declared in special XML documents that conform to a schema in BrainMetaL called `vocabulary.xsd`, with namespace `urn:bml/brainml.org:internal/BrainMetaL/1` (this namespace is shared by all BrainMetaL entities). A fragment of a vocabulary document looks like this:

```
<lexicon xmlns="urn:bml/brainml.org:internal/BrainMetaL/1">
    <domain id="_10" name="trace">
        <domain id="_20" name="recording technique">
            <term id="_50" name="extracellular">
```

```
                <term id="_60" name="single electrode">
                    <term id="_70" name="single-unit source"/>
                    <term id="_71" name="multi-unit source"/>
                </term>
                <term id="_61" name="multielectrode array"/>
            </term>
        </domain>
    </domain>
</lexicon>
```

Here, terms are specified hierarchically in *term* tags. The *domain* tags specify the domains of applicability for the terms beneath.

BrainML models that wish to use controlled vocabulary elements should specify the *name* of the domain the terms should come from. This allows a limited form of consistency checking without strictly preventing future documents from using terms in other domains if this proves desirable. The method to specify the domain name is use XML Schema "derivation by restriction" as shown in the following example:

```
element recording_technique restricts bmtl:vocab-type {
    attribute domain {xs:token} = "trace.data class"
}
```

Or, in XML syntax:

```
<xs:element name="recording_technique">
   <xs:complexType>
      <xs:complexContent>
         <xs:restriction base="bmtl:vocab-type">
            <xs:attribute name="domain" type="xs:token"
                          fixed="trace.data class"/>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
```

This requires, in an instance document, that the 'domain' attribute either be absent or be set to the specified value. Applications check this value against the domain actually found containing the referenced term in its vocabulary document, which is accessed through the link to the ID. For example, the following might occur in an instance document.

```
<data_class domain="trace.data class"
    xlink:href="urn:bml/brainml.org:internal/BrainML/2/
     vocabulary.xml#_60"/>
```

The schema validation processor will first check that "trace.data class" matches what is specified in the schema. The BrainML application will then follow the href to pick up the declaration of the term used, then the domain the term occurs in, which it will check against "trace.data class". Note that this check does not verify that the "trace.data class" domain is in the same document as originally intended; this is done deliberately to provide an additional measure of flexibility.

6.1. External References

Many vocabularies already exist in neuroinformatics and bioinformatics, and some of these contain definitions and structure that go beyond the simple name and hierarchical position supported by BrainML vocabulary. BrainML does not attempt to accomodate directly all of the possible ways that such information could be represented, but it does not preclude the use of such vocabulary. The terms themselves can be declared in a voacabulary document together with links connecting them to an external resource containing material on their definitions and relations to other terms. For this purpose, the `external-equivalent` tag is used, as in the following example:

```
<domain id="_13" name="cortical location">
    <domain id="_27" name="cytoarchitectural area">
        <term id="_540" name="1">
            <external-equivalent xlink:href="urn:UMLS/CID=C0037658"/>
            <external-equivalent xlink:href=
    "http://braininfo.rprc.washington.edu/Scripts/search.aspx?
          searchstring=area%201"/>
        </term>
    </domain>
</domain>
```

Here, `external-equivalent` is used to link a term both to a Concept ID in the UMLS and to an information page at BrainInfo. The contents of `external-equivalent` are not strictly formalized by BrainML, however subcommunities are free to require, for example, that every vocabulary term includes a link in a certain format to a UMLS term.

7. Units of Measure

Units of measure are handled in an analogous fashion to controlled vocabulary. A BrainML field may declare that it accepts units of measure as valid values. Units are declared in documents conforming to a schema in BrainMetaL called `units.xsd`, with namespace `urn:bml/brainml.org:internal/BrainMetaL/1` (this namespace is shared by all BrainMetaL entities). A fragment of a units document looks like this:

```
<unit-definition id="millivolt">
    <name>
        <base-name>volt</base-name>
        <prefix>milli</prefix>
    </name>
    <class>electric-potential</class>
    <symbol>mV</symbol>
    <external-equivalent
         xlink:href="http://iso.org/si-units.xsd,millivolt" />
</unit-definition>
```

The `external-equivalent` tag serves a similar purpose here to the controlled vocabulary case. (*Note the contents are fake; there is no XML format for units standardized under an organization at the level of ISO or NIST.*)

BrainML models that wish to use units elements for fields do so like this:

```
element horizontal_axis_units restricts bmtl:unit-type { }
```

Or, in XML syntax:

```
<xs:element name="horizontal_axis_units">
  <xs:annotation>
     <xs:documentation>Unit of measure for horizontal axis (often a
         temporal unit).</xs:documentation>
  </xs:annotation>
  <xs:complexType>
     <xs:complexContent>
        <xs:restriction base="bmtl:unit-type"/>
     </xs:complexContent>
  </xs:complexType>
</xs:element>
```

In an instance document this would appear as:

```
<horizontal_axis_units
   xlink:href="urn:bml/brainml.org:internal/BrainMetaL/1
      /units.xml#millisecond"/>
```

A unit of measure can also be specified for a numerical value directly. This is done using a `measured_quantity` type that incorporates both a unit reference and a number:

Definition:

```
element electrode_tip_diameter { bmtl:measured_quantity-type }
```

Use:

```
<electrode_tip_diameter
    xlink:href="urn:bml/brainml.org:internal/BrainMetaL/1
        /units.xml#_millimeter">
  0.04
</electrode_tip_diameter>
```

8. Links and Relationships

Links connect elements with others that are related to them, but either exist in a many-to-many relationship with elements of this type, so that they cannot be expressed using containment in XML, or are not included in the current document (for brevity when they are of secondary interest). For example, a time series `trace` will link to the `site` it was recorded from. This is

better than including the site under the trace: since a data document may contain multiple traces, such an approach would not only be redundant, but require an additional mechanism for identifying unique sites.

Similarly to vocabulary terms, a generic link type is defined in BrainMetaL which should be restricted for use in particular situations. It is based upon the (completed parts of the) XLink specification.

```
element link_experiment restricts bmtl:link-type {
    required attribute dest { xs:anyURI } =
        "urn:bml/brainml.org:internal/BrainML/1,experiment"
  }
```

Or, in XML syntax:

```
<xs:element name="link_experiment">
  <xs:complexType>
     <xs:complexContent>
        <xs:restriction base="bmtl:link-type">
           <xs:attribute
               fixed="urn:bml/brainml.org:internal/BrainML/1,experiment"
               name="dest" type="xs:anyURI" use="required"/>
        </xs:restriction>
     </xs:complexContent>
  </xs:complexType>
</xs:element>
```

Notice that unlike in the vocabulary case, the 'dest' attribute cannot be absent; however, a link can be provided using the default BrainMetaL `<link>` element instead if flexibility is required. Thus, an instance document could have either:

```
<link_experiment xlink:href="#data-1"
          dest="urn:bml/brainml.org:internal/BrainML/1,experiment"/>

          -or-
<link xlink:href="#data-1"/>
```

In the first case, applications can verify, during a validation stage, that the link goes to the appropriate type, and subsequent processing can safely assume that things are as expected.

9. Bibiolographic Citations

BrainML handles bibliographic citations by defining a standard format capable of supporting journal article, book chapter / in-proceedings, monograph, and thesis reference types. If these are not sufficient, or use of a different format is preferred, this is also supported, through a special generic citation-container tag.

The BrainML format may be viewed here. An example that could occur in an instance document

is:

```
<citation id="ref-22" type="article">
    <link xlink:role="urn:bml/brainml.org:internal/BrainMetaL/
        1/relations.xml#full-text"
          xlink:href="http://springerlink.metapress.com/app/home/
            content.asp?wasp=hpb9ypyuqk2tpcuq9evl
             &referrer=contribution&format=2
             &page=1&pagecount=26"/>
    <author>
        <first>Esther</first>
        <middle>P.</middle>
        <last>Gardner</last>
    </author>
    <author>
        <initials>J.Y.</initials>
        <last>Ro</last>
    </author>
    <author>
        <initials>D.</initials>
        <last>Debowy</last>
    </author>
    <author>
        <initials>S.</initials>
        <last>Ghosh</last>
    </author>
    <title>Facilitation of neuronal activity in somatosensory and
        posterior parietal cortex during prehension</title>
    <year>1999</year>
    <journal>Exp. Brain Res.</journal>
    <volume>127</volume>
    <pubmedID>99408147</pubmedID>
    <pages>329-354</pages>
</citation>
```

An example of the same reference using an external format is:

```
<citation_external id="ref-23" type="article">
    <link xlink:role="urn:bml/brainml.org:internal/BrainMetaL/
      1/relations.xml#full-text"
          xlink:href="http://springerlink.metapress.com/app/home/
            content.asp?wasp=hpb9ypyuqk2tpcuq9evl
             &referrer=contribution&format=2
             &page=1&pagecount=26"/>
    <bib:article xmlns:bib="http://brainml.org/schemas/internal
       /reference/bibtex-adaptor.xsd">
        <bib:author>
            ...
        </bib:author>
        ...
    </bib:article>
</citation_external>
```

10. BrainML Base

The BrainML *base model* provides a set of definitions that are specific to neuroscience, unlike the more generic BrainMetaL structures we have been discussing. These include entities for describing the major components of a typical neurophysiology experiment: protocol, recording site, and data traces. The latter are grouped into *views*, corresponding to graphs depicting related data. The complete package is grouped under an `<experiment>` tag. All of these elements may be subclassed in order to define more specific contents.

Please refer to the model documentation given at the above link for more information.

11. BrainML Models

The BrainMetaL and BrainML base models are not specific enough to describe the contents of a neuroscience repository. Instead, further models are defined in the BrainML language that build on these basic structures to define a particular data model. Examples may be seen by clicking Model Schemas at left. In particular, the "cortex" model defines the data model used for neurodatabase.org.

BrainML is intended to be extended by neuroscience research communities in a modular fashion. For example, the Cortex model just mentioned incorporates other models for describing animal subjects and protocols, and basic neuronal structures. This approach avoids the need to redefine common structures, and makes integration of multiple repositories easier when they share terminology.

In the typical scenario, a community wishing to set up a repository first browses the available models at BrainML.org to see which structures are available for describing their data. Then, for those aspects for which there is no matching structure, they create their own definitions and place these into a new BrainML model. This model incorporates by reference those components in other models that they need. They then set up their repository to be based on this model (see Neurodatabase Construction Kit). By contributing this model to brainml.org, they ensure that other groups wishing to model similar data can benefit from their work, and that any additional repositories they set up can readily be linked to theirs.

12. Conclusion

BrainML is a framework utilizing the object-oriented and modular facilities in XML schema to aid the process of setting up and interlinking neuroscience data repositories. In addition to the features described here, a set of protocols is defined for transferring BrainML data between software agents, whether these are clients, servers, or peer systems. Finally, an open-source software repository and toolset implementation conforming these protocols will be provided in

the summer of 2006.