# BBF RFC 31: Provisional BioBrick Language (PoBoL)

Michal Galdzicki, Deepak Chandran, Alec Nielsen, Jason Morrison, Mackenzie Cowell, Raik Grünberg, Sean Sleight, Herbert Sauro

15 May 2009

## 1. Purpose

This BioBricks Foundation Request for Comments (BBF RFC) describes a semantic markup language for publishing and sharing information about BioBricks on the World Wide Web. This BBF RFC includes the recommendation for the minimal information expected when creating a Provisional BioBrick Language (PoBoL) description of BioBricks and for the implementation of the language using Web Ontology Language (OWL).

## 2. Relation to other BBF RFCs

BBF RFC 31 UPDATES BBF RFC 30 as it proposes a standard conforming to BBF RFC 30.

## 3. Copyright Notice



## 4. Acknowledgments

This document is the result of discussions from the Standards and Specifications in Synthetic Biology Workshop held in Seattle April 26-27, 2008. This workshop was sponsored by the Microsoft Computational Challenges in Synthetic Biology Initiative.

We especially want to thank the participants for their contribution to developing this standard:
J. Christopher Anderson, Frank Bergmann, Josh Bishop, Michael Blinov, Eric Butterworth, Javier Carrera, Deepak Chandran, Mackenzie Cowell, John Cumbers, Shivang Dave, Kim de Mora, Michal Galdzicki, Jonathan Goler, Raik Grüenberg, Jim Haseloff, Brandi House, Mike Hucka, Kyung Kim, Simon Mercer, Andrew Miller, Jason Morrison, Jean Peccoud, Michael Pedersen, Sarah Richardson, Guillermo Rodrigo, Vincent Rouilly, Ralph Santos, Herbert Sauro, Sean Sleight, Lucian Smith, Ingrid Swanson, Alan Villalobos.

# 5. Table of Contents



# 6. Motivation

Computational access to information about biological parts, especially about BioBricks from the Registry of Standard Biological Parts (http://partsregistry.org), is currently restricted. Exchange of this information in a common semantic markup language can reduce the barrier for creating interoperable software to support the information needs of synthetic biologists. Such markup language standards leverage software implemented by the community to facilitate reuse of previously generated knowledge across independent research efforts. To build such a format we propose PoBoL as a starting point for a community discussion. The ideas proposed here have admitted flaws, and these MUST be addressed prior to adopting PoBoL for use.

# 7. Introduction

PoBoL in its current state has been defined using the Web Ontology Language (OWL), a type of markup language used to model domain knowledge. The choice to use OWL was based

on the theory that formally modeling knowledge in a computable, standardized, and community supported format will provide a benefit. Use of OWL is suggested by BBF RFC 30, which explains its advantage for an evolving model. BBF RFC 31 attempts to follow the guidelines set forth in BBF RFC 30 to define the *core data model* described therein.

PoBoL defines the terminology used specifically when using standardized biological parts known as BioBricks. These definitions constitute an ontology and therefore the concepts that are defined within this RFC attempt to formalize this vocabulary by making explicit the meaning thereof. The purpose of this ontology is the electronic storage and transmission of information about BioBricks. It is limited in scope to only those ideas thought of as central to BioBricks. Therefore it leaves out much of the knowledge about biological complexity of BioBricks and focuses on concepts created to support BioBricks themselves.

All of the PoBoL elements for this project were created using the Protégé Editor and therefore it follows a typical pattern of development guided by its user interface. However, the structure of the OWL definitions, examples, and intended use are aimed at independent users. It is expected that most users will not consume PoBoL using Protégé, but rather by using third-party applications. These tools SHOULD be developed specifically to leverage the standard specification of BioBricks information in PoBoL using OWL.

# 8. Background

A selection of the many features of OWL are used to define PoBoL. Terms used to refer to their meaning in the OWL syntax specification are italicized in the remainder of this document to distinguish them. Subsumption is a fundamental method, organizing concepts into classes and their subclasses. In OWL, *subClassOf* means that every member of the subclass is also a member of the class. This implies that a member of a subclass inherits the properties of the class. This structure allows the organization of concepts into a hierarchy starting with the general and moving towards more specific distinctions between types of things.

# 9. Description

PoBoL is defined in terms of a hierarchy of classes and properties. The classes correspond to concepts about BioBricks. The properties specify relationships between actual BioBricks, termed instances. There are five proposed root classes *BioBrick*, *BioBrickFamily*, *BioBrickFormat*, *DNA*, and *Sample* which correspond to five concepts for potential use in the unambiguous description of information about BioBricks. Each instance of a *BioBrick* class refers to an actual BioBrick. The *BioBrick* class has subclasses, which refer to more specific concepts of *BioBrickBasic*, *BioBrickComposite*, and *BioBrickVector*. When using PoBoL to describe data about BioBricks, an instance of one of the subclasses of *BioBrick* MUST be created. The *BioBrick* class MAY be extended if the subclasses described here are not descriptive enough.

The classes and their properties are shown in Figure 1 and described in more detail in section 10.1.

## 10.1 Core Data Standard

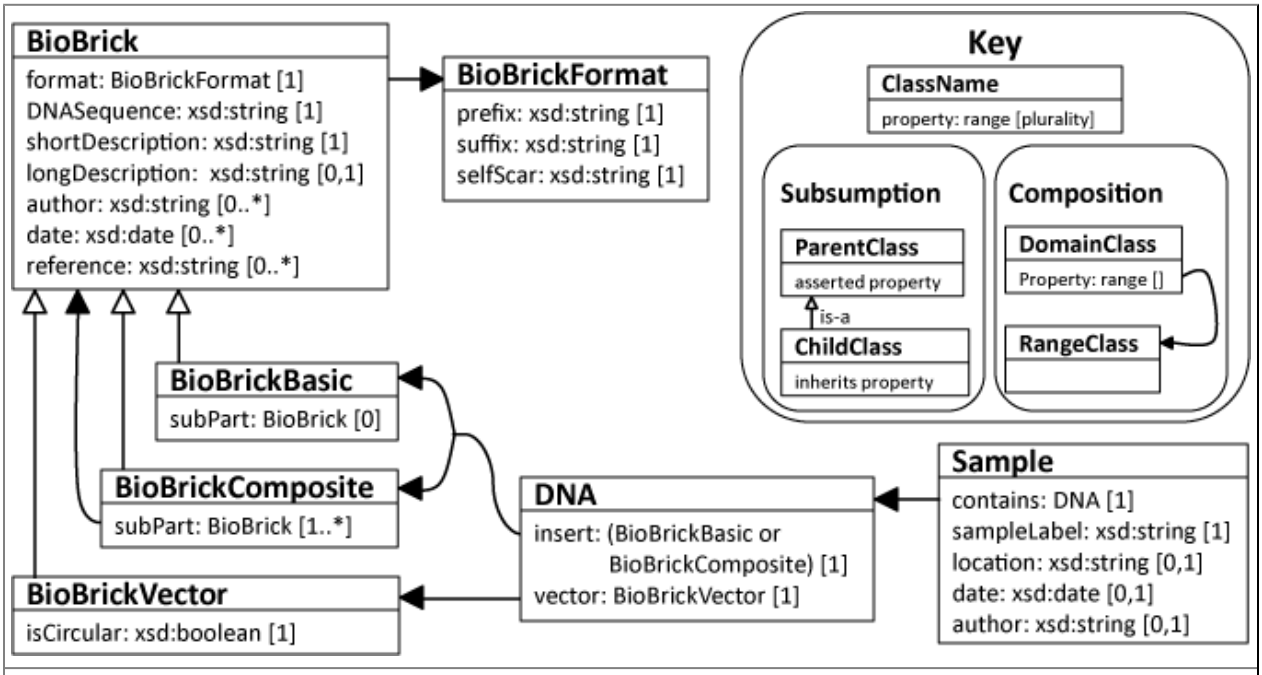


**Figure 1.** PoBoL is described using a UML like diagram. Each class may have many properties. Each property has a range type and a plurality, such as "exactly one DNA" *DNA* [1] or "one or more BioBrick" *BioBrick* [1..*]. The subsumption hierarchy is depicted using open arrows, this illustrates how an instance of the child class will inherit all properties of the parent class. Intended composition of instances is shown using closed arrows.
RDF Namespaces: (default) http://www.biobricks.org/rdf/pobol.owl#; xsd:http://www.w3.org/2001/XMLSchema#

### 10.1.1 Class *BioBrick*:

A root class describing the general concept of BioBrick in terms of properties which MUST be present at a minimum to describe any BioBrick.

REQUIRED Property: *format* MUST be associated with one and only one instance of the *BioBrickFormat* class. This property refers the BioBrick to a physical composability standard's description (see the *BioBrickFormat* class description for more detail). There can only be one *format* property for each *BioBrick.*

REQUIRED Property: *DNASequence* MUST have one value of type *xsd:string*. The DNA sequence for the BioBrick is specified here. There can only be one *DNAsequence* property for each *BioBrick*.

REQUIRED Property: *shortDescription* MUST have one and only one value of type *xsd:string*. This property is intended to allow free text for users such as a title or name. There MUST be one and only one *shortDescription* property for each *BioBrick.*

RECOMMENDED Property: *longDescription* MUST have one value of type *xsd:string*. This property is intended to allow a longer free text based description for users. There can be zero or one *longDescription* property for each *BioBrick.*

OPTIONAL Property: *author* MUST have one value value of type *xsd:string*. This property is intended to hold the name of the creator of the BioBrick being described. There can be zero or more *author* properties for each *BioBrick.*

OPTIONAL Property: *date* MUST have one value of type *xsd:date*. This property is intended to store the date of the creation of the BioBrick. There can be zero or one *date* property for each *BioBrick.*

OPTIONAL Property: *reference* MUST have one value of type *xsd:string*. This property is intended to store citation for this BioBrick. There can be zero or more *reference* properties for each *BioBrick.*

PoBoL is extensible and therefore the *BioBrick* class is not limited to only having the properties listed above. Consequently, to describe characteristics important to goals of specific users or groups properties MAY be added. The *BioBrick* class MAY be extended at any time.

### 10.1.2 Class *BioBrickBasic*:

A BioBrick which is not composed of other BioBricks itself. These basic BioBricks are intended to serve as the unique parts used to make a particular construct. The *BioBrickBasic* class is a subclass of the *BioBrick* class and therefore inherits all of its properties. An instance of *BioBrickBasic* is also a *BioBrick* instance.

MUST NOT have property *subPart*: This type of *BioBrick* instance is a *BioBrickBasic* when it is not composed of other BioBricks.

MUST be disjoint from *BioBrickComposite:* PoBoL documents SHALL NOT have instances of *BioBrickBasic* which are members of *BioBrickComposite.*

### 10.1.3 Class *BioBrickComposite*:

A BioBrick made from multiple sub part BioBricks. This class is a subclass of the *BioBrick* class and therefore inherits all of its properties. An instance of *BioBrickComposite* is also a *BioBrick* instance.

REQUIRED Property: *subPart* MUST have one or more *BioBricks* instances as values. There MUST be one or more *subPart* properties for each *BioBrickComposite.*

### 10.1.4 Class *BioBrickVector:*

A special BioBrick into which the construct of interest is inserted to be transfected into cells. This class is a subclass of the *BioBrick* class. Typically in synthetic biology, vectors are plasmids used in cell engineering, but could potentially be entire chromosomes. The *BioBrickVector* serves to provide information about the entire molecule used to engineer cells. It inherits the properties of generalized *BioBricks* and therefore an instance of *BioBrickVector* is also a *BioBrick* instance.

RECOMMENDED Property: *isCircular* MUST be set using an *xsd:boolean* value as True or False. Indicates wether the vector being described is circular or not. There MUST be one and only one *isCircular* property for each *BioBrickVector.*

### 10.1.5 Class *BioBrickFormat*:

Describes the sequence features which designate a specific physical composition standard. Instances of this class MUST be named as specified by BBF RFC 29. These standards, from the point of view of information about it, is defined by its prefix, suffix, and self scar DNA sequences.

REQUIRED Property: *prefix* MUST have one value of type *xsd:string.* DNA sequence prefix. There MUST be one and only one *prefix* property for each *BioBrickFormat.*

REQUIRED Property: *suffix* MUST have one value of type *xsd:string.* DNA sequence suffix. There MUST be one and only one *suffix* property for each *BioBrickFormat.*

REQUIRED Property: *selfScar* MUST have one value of type *xsd:string.* DNA sequence which remains after assembly is complete. There MUST be one and only one *selfScar* property for each *BioBrickFormat.*

### 10.1.6 Class *DNA:*

Describes the DNA molecule as a combination of the Vector and the BioBrick insert used within the construct. It serves as a vehicle relating the abstract description of components manipulated as BioBricks to the actual continuous DNA molecule, which can be present in a test tube.

REQUIRED Property: *insert* MUST have one *BioBrickBasic* or *BioBrickComposite* instance as a value. There MUST be one and only one *insert* property for each *DNA.*

REQUIRED Property: *vector* MUST have one *BioBrickVector* instance as value. There MUST be one and only one *vector* property for each *DNA.*

### 10.1.7 Class *Sample*:

Represents the physical container and the BioBrick DNA that is stored.

REQUIRED Property: *contains* MUST have a value of instance of the class *DNA*. There MUST be one and only one *contains* property for each *Sample.*

REQUIRED Property: *sampleLabel* MUST have one value of type *xsd:string* listing the exact text found on the tube itself. Effectively connecting the data record in the computer to laboratory storage. There MUST be one and only one *sampleLabel* property for each *Sample.*

RECOMMENDED Property: *location* MUST have one value of type *xsd:string* describing the location of the sample within laboratory storage. There can be zero or one *location* property for each *Sample.*

OPTIONAL Property: *author* MUST have one value value of type *xsd:string*. This property is intended to hold the name of person who created the sample in the laboratory. There can be zero or one *author* property for each *Sample.*

OPTIONAL Property: *date* MUST have one value of type *xsd:date*. This property is intended to store the date of the creation of the sample. There MUST be one and only one *date* property for each *Sample.*

# 11. Example BioBrick BBa_I0462

Publishing BioBricks information using PoBoL allows unambiguous exchange of information. As an example we demonstrate the markup of BBa_I0462 using PoBoL. Information from the Registry of Standard Biological Parts (http://partsregistry.org) is used to describe this simple composite BioBrick coding for a protein generator of the LuxR protein. The entry used can be found at http://partsregistry.org/Part:BBa_I0462. This BioBrick is composed of four previously described BioBrick parts: a ribosome binding site (BBa_B0034), the coding sequence for LuxR (BBa_C0062), and two terminators (BBa_B0010 and BBa_B0012) (Figure 2). BBa_I0462 and each of its parts conform to BBF Assembly Standard 10 (as defined in BBF RFC 29). An instance of BBa_I0462, which demonstrates key features of PoBoL, is displayed below. The PoBoL schema can be found in Appendix A of this document.

```
@prefix : <http://www.biobricks.org/rdf/pobol.owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@base <http://www.biobricks.org/rdf/pobol.owl>

:Assembly_Standard_10_non_coding rdf:type :BioBrickFormat ;
      :prefix "GAATTCGCGGCCGCTTCTAGAG"^^xsd:string ;
      :suffix "TACTAGTAGCGGCCGCTGCAG"^^xsd:string .

:BBa_I0462 rdf:type :BioBrickComposite ;
      :date "2009-04-19"^^xsd:date ;
      :author "Caitlin Conboy"^^xsd:string ,
              "Jennifer Braff"^^xsd:string ;
      :longDescription "Produces LuxR protein which can sense 3OC6HSL in
          the media and activate transcription from R0062, the right hand
          Lux promoter"^^xsd:string ;
      :DNASequence "aaagaggag ... ttctgcgtttata"^^xsd:string ;
      :shortDescription "luxR Protein Generator"^^xsd:string ;
      :format :Assembly_Standard_10_non_coding ;
      :subPart :BBa_B0034 ,
               :BBa_C0062 ,
               :BBa_B0010 ,
               :BBa_B0012 .
```

```
:DNA_55 rdf:type :DNA ;
      :insert :BBa_I0462 ;
      :vector :pSB1A2 .

:Sample_56 rdf:type :Sample ;
           :date "2009-04-27"^^xsd:date ;
           :author "John smith"^^xsd:string ;
           :sampleLabel "Sample 56 E. coli strain MG1655"^^xsd:string ;
           :location "Stocks Box 25 D7"^^xsd:string ;
           :contains :DNA_55 .
```

PoBoL instance of BBa_I0462 is expressed using OWL/Turtle syntax, designed to be more easily read by people in contrast to XML. This is an excerpt and only includes a truncated DNA sequence, for better visual display. Additionally, the instances of sub parts are not specified themselves in this excerpt. For a full PoBoL entry using OWL/Turtle and OWL/XML syntax see Appendix A of this document.

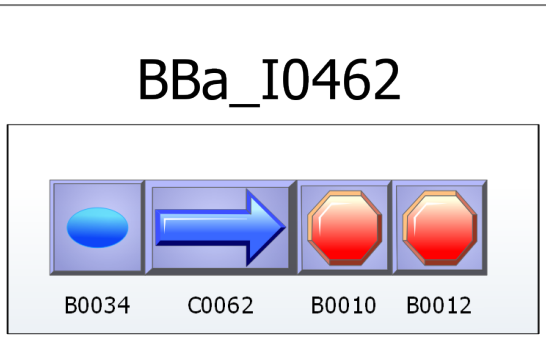


**Figure 2.** Graphic of BBa_I0462 designed using TinkerCell (www.tinkercell.com) as composed of BBa_B0034, BBa_C0062, BBa_B0010, and BBa_B0012.

## 11.1 Use of PoBoL in an experimental setting

A potential use of PoBoL is the storage of relevant wet-lab information in a standardized format. PoBoL is amenable to the organization of physical samples, documentation of DNA constructs, and allows for the possibility of manipulation by laboratory automation equipment.

An instance of the *DNA* class stores not only the associated BioBrick, but the composition of an entire vector, including the backbone. This is useful for inferring the plasmid copy number, the vector's antibiotic resistance, sequencing primers, and other elements relevant for physical composition, cloning, sequencing, and PCR amplification. The DNA class, therefore, contains the property *insert*, which stores a BioBrick part, and the property *vector*, which stores relevant information for physical manipulation of the BioBrick.

An instance of the *Sample* class represents a physical manifestation of DNA, and is used largely for laboratory documentation, organization, and opens the possibility for laboratory automation. The properties of a physical sample include a label, location, date, and author. These properties are derived from being commonly included on containers in laboratory settings. However, the efficacy of PoBoL allows samples to be sorted and searched automatically. In addition, the property *location* SHOULD be included in PoBoL to aid in the organization of laboratory samples, and to provide laboratory automation equipment the necessary information needed to locate and manipulate physical samples.

As an example of how PoBoL might be used to aid wet-lab organization, an instance of the *DNA* class could have an insert, BBa_I0462, and a vector backbone, pSB1A2. This information indicates that this plasmid DNA contains a part with some desired functionality and that the part is present on a backbone vector. For example, the production of luxR protein which, in the presence of $3OC_6HSL$, up-regulates expression downstream of the right hand Lux promoter (BBa_R0062), confers ampicillin resistance, and can be sequenced using the VF2 and VR primers.

Continuing with the wet-lab example, an instance of the *Sample* class could represent physical DNA containing part BBa_I0462 and backbone pSB1A2. Furthermore, properties of this *Sample* class instance would indicate that the sample was prepared by John Smith on April 27, 2009. Additional information could be stored in the *sampleLabel* property; the label might indicate the host organism with a *sampleLabel* value "Sample 56 *E. coli* strain MG1655."

The *DNA* class allows for the storage of pertinent information regarding plasmid composition. The *Sample* class allows for documentation of physical samples, including location, label, date, author and the sample's contents. These classes provide an extra layer of utility to the PoBoL structure: applicability to wet-lab organization and manipulation.

## 11.2 Use of PoBoL in a software tool

We demonstrate potential uses of PoBoL by using a specific software example. The TinkerCell application (www.tinkercell.com) was used to draw the composite part, BBa_I0462 (Figure 2), is able to model synthetic networks as well as store information required to assemble the part. However, the software does not have any *standard* method to output the part that has been constructed. The following Python script was used inside this application in order to interrogate the BBa_I0462 part.

```
>>part = find("BBa_I0462")
>>print annotation(part)
('Caitlin Conboy and Jennifer Braff', '12/04/2003', 'luxR Protein Generator',
'uri', 'reference')

>>print getTextAttribute(part,"format")
Assembly_Standard_10_non_coding

>>print getTextAttribute(part,"type")
composite

>>subparts = getChildren(part)
>>for i in subparts: print getName(i) + " is a " + getFamily(i);
BBa_I0462_B0012 is a Terminator
```

```
BBa_I0462_B0010 is a Terminator
BBa_I0462_C0062 is a Coding
BBa_I0462_B0034 is a RBS

>>for i in subparts: print getName(i) + " : "; print annotation(i);
BBa_I0462_B0012 :
('Reshma Shetty', 'NA', 'Transcription terminator for the E.coli RNA
polymerase.', 'uri', 'reference')

BBa_I0462_B0010 :
('Randy Rettberg', '11/19/2003', 'Transcriptional terminator consisting of a 64
bp stem-loop', 'uri', 'reference')

BBa_I0462_C0062 :
('Vinay S Mahajan, Voichita D. Marinescu, Brian Chow, Alexander D Wissner-Gross
and Peter Carr', '2003', 'luxR repressor/activator', 'uri', 'reference')

BBa_I0462_B0034 :
('Vinay S Mahajan, Voichita D. Marinescu, Brian Chow, Alexander D Wissner-Gross
and Peter Carr', '2003', 'RBS based on Elowitz repressilator', 'uri',
'reference')

>>for i in subparts: print getName(i) + " : "; print
getTextAttribute(i,"sequence");
BBa_I0462_B0012 :
tcacactggctcaccttcgggtgggcctttctgcgtttata

BBa_I0462_B0010 :
ccaggcatcaaataaaacgaaaggctcagtcgaaagactgggcctttcgttttatctgttgtttgtcggtgaacgctctc

BBa_I0462_C0062 :
aaagaggag ... ttctgcgtttata

BBa_I0462_B0034 :
aaagaggagaaa

>>print getParameter(subparts[3],"strength")
```

As the script illustrates, the application contains relevant information about the part. However, if a user needs to export the data from this application to another, say a sequence analysis application, then the user would need to do this transfer manually. Manual transfer can result is loss of information and errors, because the user may only transfer the information he or she requires. As a result, the information stored with the original software is lost.

**The solution using PoBoL would be as follows:**

The Python script would generate the instance of the *BioBrickComposite* as define above, immediately under the Example BioBrick section title. The PoBoL file will serve as a link between two or more software applications, rather than requiring the user to do this manually. Such automated transfer would be easier for the user and would preserve the minimum information needed to define a BioBrick. Passing the BioBrick from one application

to another as a PoBoL file would never cause a loss of data in this manner. An application that can preserve the PoBoL file is guaranteed to provide sufficient information to construct the part.

Such exchange format between software applications has been employed in related fields, such as the Systems Biology Markup Language [1] (SBML). It has permitted numerous software to exchange information between one another without ambiguity. The idea behind the application of PoBoL is similar, except that it stores part information whereas SBML stores kinetic model information.

# 12. Validation OWL Syntax

Pellet 1.5.2 (Clark & Parsia, LLC) was used to perform an independent validation of the OWL syntax. The output of the run is displayed below.

```
C:\Documents and
Settings\mgaldzic\Desktop\synBio\pobol\RFC31_pobol\0.5\examples>java -Xss4m
-Xms30m -Xmx200m -jar "C:\Program Files\pellet-1.5.2\lib\pellet.jar" -if
pobol_BBa_I0642.owl

Input file: file:/C:/Documents%20and%20Settings/mgaldzic/Desktop/synBio/
pobol/RFC31_pobol/0.5/examples/pobol_BBa_I0642.owl
OWL Species: DL
DL Expressivity: ALCF(D)
Consistent: Yes
Time: 2485 ms (Loading: 2203 Species Validation: 172 Consistency: 110 )

Non OWL-Lite features used:
Disjoint Classes: owl:disjointWith construct is used
DisjointClasses(pobol:BioBrickBasic pobol:BioBrickComposite)
Disjoint Classes: owl:disjointWith construct is used
DisjointClasses(pobol:BioBrickComposite pobol:BioBrickBasic)
Union Class: owl:unionOf construct is used unionOf(pobol:BioBrick
pobol:Sample)
Union Class: owl:unionOf construct is used unionOf(pobol:BioBrickBasic
pobol:BioBrickComposite)
```

## 12.1 Use of OWL syntax for PoBoL

As described in RFC30, an OWL-based architecture enables the extension of the PoBoL data model and facilitates a "rolling standardization" process for these extensions, which enables future expansion in scope of BioBrick description documents while adhering to a core standard like PoBoL. Additionally, "OWL is a good citizen of the web" because it supports sharing (in the sense of synchronizing) and interlinking (distributing and connecting databases) information.

# 13. Author's Contact Information

Michal Galdzicki mgaldzic@u.washington.edu
Alec Nielsen alecn@u.washington.edu
Deepak Chandran deepakc@u.washington.edu
Jason Morrison jason.p.morrison@gmail.com
Mackenzie Cowell mac@diybio.org
Raik Grünberg raik.gruenberg@crg.es
Sean Sleight sleight@u.washington.edu
Herbert Sauro hsauro@u.washington.edu

# 14. References

[1] M. Hucka, A. Finney, HM Sauro, H. Bolouri, JC Doyle, H. Kitano, AP Arkin, BJ Bornstein, D. Bray, A. Cornish-Bowden, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models, 2003

# 15. Appendix A

Other files associated with this BBF RFC
pobol_BBa_I0642.owl
pobol_BBa_I0642.turtle
pobol.owl
pobol.turtle