ACS SyntheticBiology

Research Article

pubs.acs.org/synthbio

Proposed Data Model for the Next Version of the Synthetic Biology Open Language

Nicholas Roehner,[†,*] Ernst Oberortner,[‡] Matthew Pocock,[§] Jacob Beal,[∥] Kevin Clancy,[⊥] Curtis Madsen,[§] Goksel Misirli,[§] Anil Wipat,[§] Herbert Sauro,[#] and Chris J. Myers[▽]

[†]Department of Bioengineering, University of Utah, Salt Lake City, Utah, United States

[‡]Department of Electrical and Computer Engineering, Boston University, Boston, Massachusetts, United States

[§]School of Computing Science, Newcastle University, Newcastle upon Tyne, United Kingdom

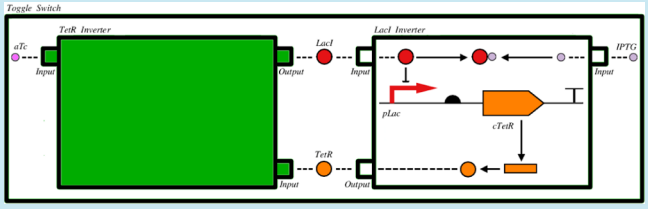
[∥]Raytheon BBN Technologies, Cambridge, Massachusetts, United States

[⊥]Life Technologies, Carlsbad, California, United States

[#]Department of Bioengineering, University of Washington, Seattle, Washington, United States

[▽]Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, Utah, United States

ABSTRACT: While the first version of the *Synthetic Biology Open Language* (SBOL) has been adopted by several academic and commercial *genetic design automation* (GDA) software tools, it only covers a limited number of the requirements for a standardized exchange format for synthetic biology. In particular, SBOL Version 1.1 is capable of representing DNA components and their hierarchical composition via sequence annotations. This proposal revises SBOL Version 1.1, enabling the representation of a wider range of components with and without sequences, including RNA components, protein components, small molecules, and molecular complexes. It also introduces modules to instantiate groups of components on the basis of their shared function and assert molecular interactions between components. By increasing the range of structural and functional descriptions in SBOL and allowing for their composition, the proposed improvements enable SBOL to represent and facilitate the exchange of a broader class of genetic designs.



KEYWORDS: *SBOL, synthetic biology, standards, modular design*

The Synthetic Biology Open Language (SBOL)[1] is an emerging data exchange standard for synthetic biology with growing support among genetic design automation (GDA) software tools.[2−13] SBOL has been developed by various members of the synthetic biology community to document DNA components for the primary purpose of engineering genetic designs. Unlike existing standards that were originally conceived for documenting naturally occurring genetic sequences, such as the FASTA[14] and GenBank[15] formats, SBOL can be used to document partial genetic designs and to recursively annotate the sequences of DNA components with subcomponents in a hierarchical fashion. These capabilities of SBOL address the iterative, modular character of engineering design in a way that current standards for genetic sequences neglect. Furthermore, SBOL is an extensible standard that can be adapted to meet the evolving needs of the synthetic biology community, such as connecting structural, sequence-oriented descriptions of genetic designs with descriptions of their function. Without standards that meet the need for dual representation of genetic structure and function, there can be no exchangeable basis for design automation in synthetic biology,[16] a paradigm that has been applied to great success in electrical and computer engineering.

While a host of GDA tools exist for applications such as biochemical modeling and simulation,[13,17−21] sequence editing and optimization,[2,10,22,23] design composition,[3,4,8,12,24,25] and more recently genetic technology mapping,[5,26−29] not all of these tools use publicly available standards to represent data and none of them use standards to tightly couple descriptions of genetic structure and function. In order for GDA tools to facilitate interdisciplinary collaboration and exchange of genetic designs, they must use standards that represent both genetic structure and function, even if individual tools only focus on one aspect of genetic design. Without such standards, GDA tools with different applications cannot exchange a single design, thus potentially damaging the reproducibility of a designer's intent. In addition, GDA tools that handle both structural and functional aspects of genetic designs cannot use a single standard, thus making it more difficult to document how the structure of a genetic design affects its function and vice versa.

The current SBOL standard, Version 1.1,[1,30] is primarily capable of representing the structural aspects of genetic designs. To serve as an effective medium for the computational exchange of genetic designs, SBOL must be extended to increase the scope of

Special Issue: SEED 2014

Received: March 7, 2014
Published: June 4, 2014

   dx.doi.org/10.1021/sb500176h | *ACS Synth. Biol.* 2015, 4, 57−71

genetic structural and functional information that it can encode and to provide a basis for its use in composing the structural and functional layers of genetic designs. The SBOL data model proposed in this paper provides a roadmap for addressing three of the most pressing needs for expanding SBOL Version 1.1. The first need is the ability to structurally represent components of a genetic design other than DNA components, the second need is the ability to provide functional representations of these components, and the third need is a composition framework for connecting descriptions of genetic struture and function. As long as these capabilities remain outside the scope of SBOL, the SBOL standard is not sufficiently expressive to provide hierarchical, modular representations of both the intended structure and function of genetic designs.

This paper reviews the current capabilities of SBOL Version 1.1 before describing a proposed data model that was recently presented at the SBOL 10 workshop held at the University of California, Berkeley and voted on as a starting point for the next version of SBOL. It is important to emphasize that this data model does not represent the final, community-approved specification for the next version of SBOL. Rather, this data model is a proposal that draws from both discussions within the SBOL community and the original contributions of the authors of this paper. It is an intermediate result in a larger development process, one in which feedback is being gathered from the synthetic biology community at large in order to reflect, fulfill, and standardize the data exchange requirements of the community in the next version of the SBOL standard.

SBOL VERSION 1.1

Figure 1 illustrates an example of the current capabilities of SBOL Version 1.1 using symbols taken from the SBOL Visual standard.[31] In this example, the DNA component for a *genetic toggle switch*[32] is hierarchically composed from a TetR-repressible gene and a LacI-repressible gene, which are in turn composed from the *pTet* promoter, the *cLacI* coding sequence (CDS), ribosome binding sites (RBS), terminators, the *pLac* promoter, and the *cTetR* CDS. In the case of the toggle switch component, one of its subcomponents (the TetR-repressible gene) is located on its negative/reverse complement strand.

A more detailed SBOL representation of the TetR-repressible gene of the genetic toggle switch is shown in Figure 2 using a *Unified Modeling Language* (UML)[34] diagram. DNA components are the core of SBOL Version 1.1 and represent abstractions of a particular DNA sequence for engineering design. Each DNA component has a *uniform resource identifier* (URI) [35] (a URI is used by software and databases to uniquely identify objects across the World Wide Web) and a *display ID* and can have at most one name, description, and DNA sequence. Each DNA component can also have one or more *types*, at least one of which must refer to a term from the Sequence Ontology (SO).[36] An ontology is a controlled vocabulary that captures terms and relationships between terms from a specific knowledge domain, thereby enabling machine reasoning over the domain.[37] In the case of the SO, the captured knowledge domain is the annotation of biological sequences with sequence features. The central DNA component shown in Figure 2 is a TetR-repressible gene that has the display ID "UU_001," a type of "*gene*", and a DNA sequence that is partially shown in the figure.

DNA components can be composed hierarchically using sequence annotations that indicate their absolute or relative position on the DNA sequence of their parent DNA component. Each sequence annotation can have a single pair of *bioStart* and *bioEnd* integers or a *precedes* relation to another sequence annotation. When present, the bioStart and bioEnd integers bound the position of a subcomponent on the DNA sequence of the parent DNA component. When one or more subcomponents do not have a DNA sequence, however, a complete DNA sequence cannot be assigned to the parent DNA component and the positions of its subcomponents cannot be exactly specified by its sequence annotations. In this case, a partial genetic design can be specified using precedes references between sequence annotations to indicate the relative positions of their subcomponents. This capability is necessary to satisfy the iterative nature of engineering, in which some details of a design cannot be specified

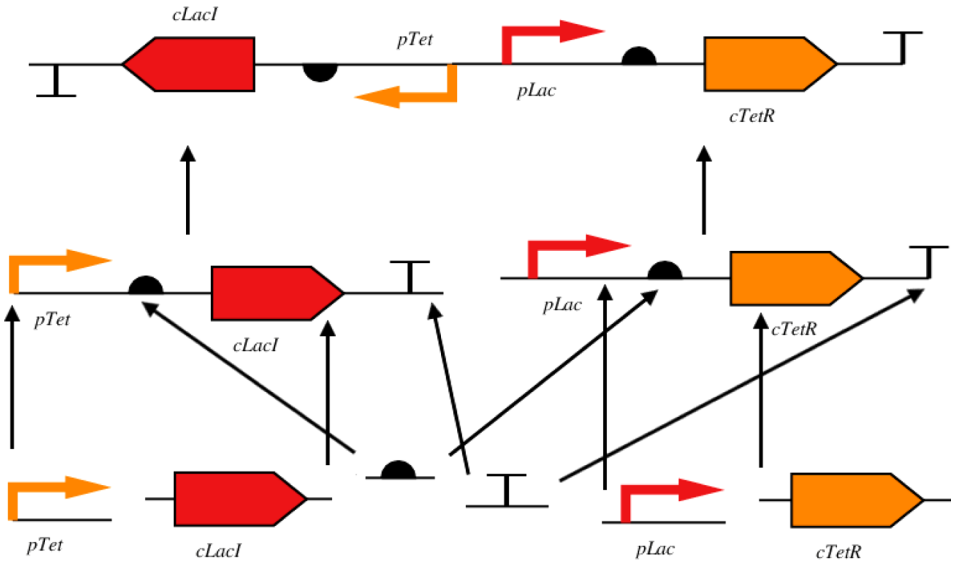


Figure 1. Hierarchical composition of the DNA component for a genetic toggle switch in SBOL Version 1.1. Each grouping of subcomponent symbols along a solid line represents a single composite DNA component. Of these symbols, each bent arrow represents a promoter, each semicircle represents a RBS, each box arrow represents a CDS, and each T-shape represents a terminator (see the SBOL Visual standard[31]). This figure was partly constructed using Pigeon,[33] an SBOL Visual-compliant tool.

dx.doi.org/10.1021/sb500176h | *ACS Synth. Biol.* 2015, 4, 57−71

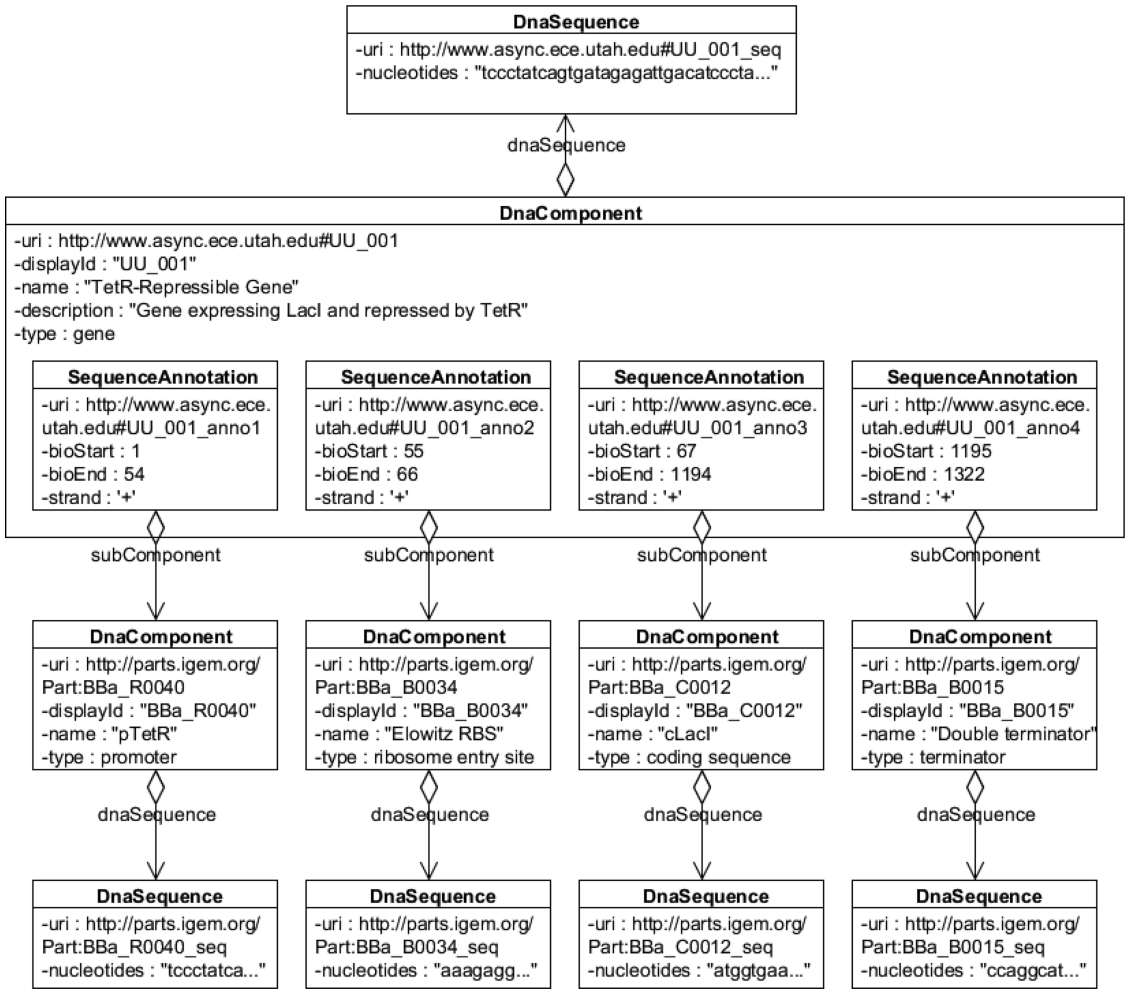


Figure 2. SBOL Version 1.1 UML for a TetR-repressible gene that expresses the TF protein LacI. Sequence annotations are placed inside the DNA component UU_001 to show that they are owned by the component. These sequence annotations indicate that four DNA subcomponents are located side by side on UU_001's DNA sequence, including the promoter BBa_R0040, the RBS BBa_0034, the CDS BBa_C0012, and the terminator BBa_0015. Accordingly, the DNA sequence of UU_001 is the concatenation of the sequences of its subcomponents.

immediately and must be revisited later in the design cycle. Furthermore, each sequence annotation can have either a '+' or "−" character to indicate whether its subcomponent is located on the positive or negative strand of its parent DNA component. The TetR-repressible gene shown in Figure 2 has four sequence annotations that specify that this gene is composed of four DNA components representing a promoter, RBS, CDS, and terminator in that order.

The general UML data model for SBOL Version 1.1 is shown in Figure 3. It includes one additional class, *Collection*, which is a container for DNA components having common characteristics. For example, a collection could be the result of querying a database to find all promoter DNA components. Currently, SBOL Version 1.1 does not support the specification of non-DNA components, such as the LacI and TetR *transcription factor* (TF) proteins of the genetic toggle switch. In addition, SBOL 1.1 does not support functional description of the toggle switch as a whole, such as the assertion of qualitative regulatory interactions between its components, or linking to mathematical models that provide information on its dynamic function in a particular organism. One of the goals of the proposed data model is to overcome each of these limitations.

RESULTS AND DISCUSSION

The primary goal of the proposed data model is to make SBOL a more comprehensive standard for genetic design. Since synthetic biology encompasses research into a broad range of entities and materials, SBOL must grow to represent a similarly broad range of structural components for genetic design. In order to more fully support the representation of genetic structure, the proposed data model generalizes the DNA component class of SBOL Version 1.1 to represent components with and without sequences. As a consequence, this data model can be used to represent RNA components, such as mRNA, tRNA, and small interfering RNA (siRNA),[38] as well as protein components, such as transcription factors (TF) and enzymes. Furthermore, the proposed data model can be used to represent potentially nongenetic components of a design, such as environmental factors, small molecules, molecular complexes, nonbiological polymers, and even light.

Because synthetic biology is increasingly concerned with the intended function of genetic designs, SBOL must also be extended to support minimalistic, qualitative representations of genetic function and to reference more detailed, quantitative representations written in specialized, external standards. To meet these needs, the proposed data model introduces classes

dx.doi.org/10.1021/sb500176h | *ACS Synth. Biol.* 2015, 4, 57−71

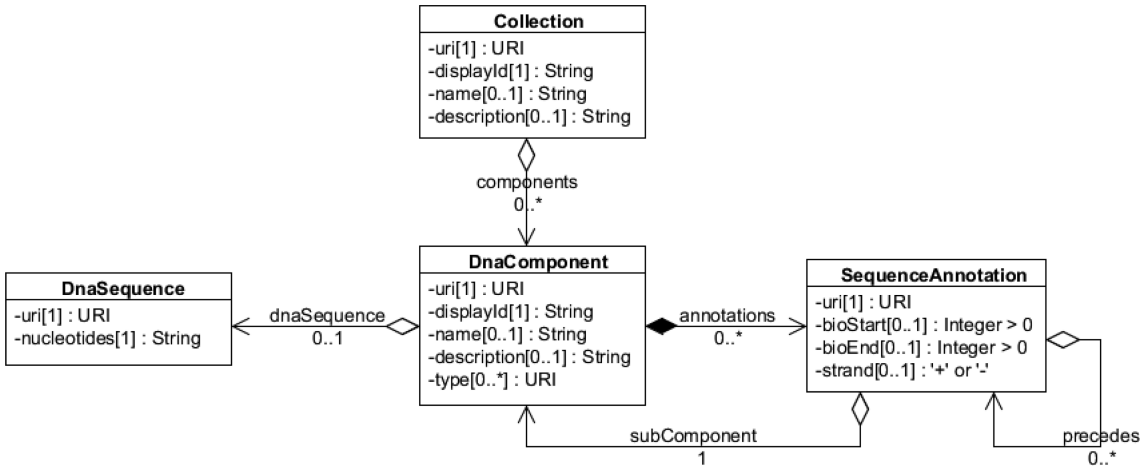


Figure 3. UML class diagram for SBOL Version 1.1,[30] consisting of the Collection, DNA Component, DNA Sequence, and Sequence Annotation classes. Each data object that belongs to these classes contains a variety of data fields, including strings of characters that identify, name, and describe the object and URIs that type and uniquely identify the object. A white diamond arrow indicates that objects of one class refer to and aggregate objects of other classes, while a black diamond arrow indicates ownership as well. For example, if a DNA component is deleted, then all of its sequence annotations are deleted, since they are owned by that DNA component. The same is not true of a DNA component and its DNA sequence, since another DNA component may share the same sequence.

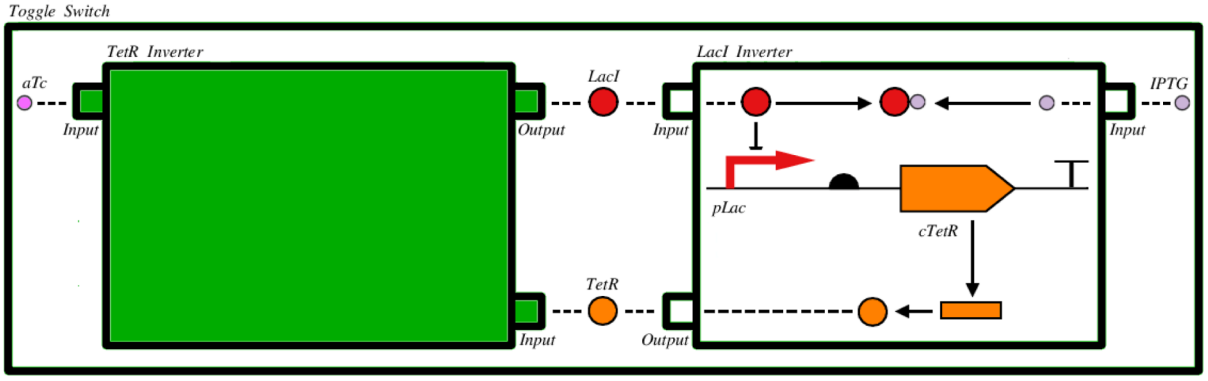


Figure 4. Design for the genetic toggle switch that captures its qualitative structure and function. The design consists of three functional modules in the form of a composite toggle switch module that contains connected copies of a TetR inverter module and a LacI inverter module. The LacI inverter module contains copies of a composite DNA component for the LacI-repressible gene, a TF protein component for LacI (red circle), a TF protein component for TetR (orange circle), an mRNA component for TetR (orange rectangle), a small molecule component for IPTG (pink circle), and a molecular complex component for LacI bound to IPTG. This module asserts a variety of molecular interactions between its contained components (solid arrows), including the repression of pLac by LacI, transcription of cTetR to TetR mRNA, translation of TetR mRNA to the TetR TF, and noncovalent binding of LacI to IPTG. While these modules allow different parts of the design to be treated as "black boxes" that have most of their contents ignored (see the TetR inverter), the ports on these modules allow connections between them (dashed lines). For example, the toggle switch is connected to the LacI inverter through mapping of the latter's input port to copies of LacI contained by both modules. In turn, the TetR inverter is connected to both the toggle switch and LacI inverter through mapping of its output port to the copy of LacI in the toggle switch.

for functional *modules*, molecular *interactions*, and mathematical models. Examples of functional modules include genetic logic gates, oscillators, sensors, and signaling cascades, while examples of molecular interactions include transcription, translation, activation/repression, noncovalent binding, and phosphorylation.

Finally, in order to be more useful for the purpose of engineering design, the proposed data model enables the hierarchical composition of separate yet connected descriptions of genetic structure and function. The data model addresses this need by introducing classes for *instantiation* and *port mapping*—two abstract, proven, and well-established concepts borrowed from the domain of electrical and computer engineering. As explained later on, instantiation allows the creation of a modular hierarchy by incorporating one or more copies of a subdesign in a composite design, while port mapping allows the specification of connections between designs by asserting the equivalence of elements within these designs. These concepts simplify the process of creating a large, complex design by facilitating the reuse of previous designs in its construction, factoring out reoccurring design patterns that would otherwise be redundant, and splitting a design into multiple distinct layers that warrant separate consideration.

As an example, consider the design for a genetic toggle switch shown in Figure 4. The proposed data model captures not only this design's structure but also its basic function. First, generalized components allow the representation of RNA components such as the mRNA coding for TetR, protein components such as TetR and LacI, and small molecules such as IPTG. Next, interactions can be specified between these components, such as the transcription of the CDS *cTetR* to TetR mRNA and the latter's translation to TetR protein. Other examples include the repression of the pLac promoter by LacI

dx.doi.org/10.1021/sb500176h | *ACS Synth. Biol.* 2015, 4, 57−71

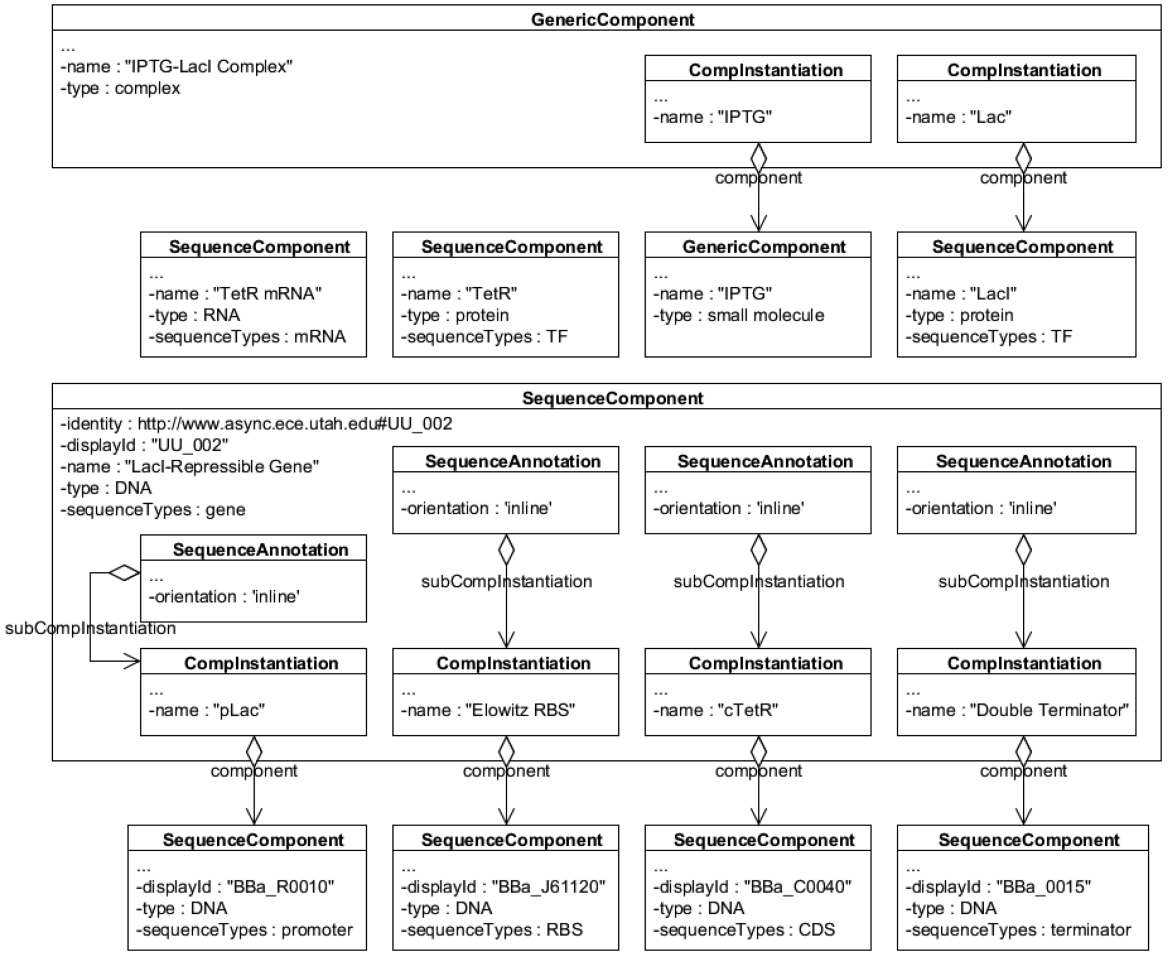


Figure 5. UML example of components under the proposed data model, including components referenced by the LacI Inverter module of the genetic toggle switch. "Comp" is short for "component" wherever it appears in this figure.

and the binding of LacI by IPTG to form a complex. In turn, these components and their interactions can be grouped into functional modules, such as a LacI inverter. Finally, these modules can be instantiated as part of larger modules, such as the instantiation of the TetR inverter and LacI inverter to form the genetic toggle switch. The points of connection between modules are specified using ports, while the connections between modules are established using port maps. The rest of this section describes each of these new features in greater detail.

Structural Representation. To support an increased range of structural representation, the proposed data model generalizes DNA components to sequence components and adds generic components to capture components without a sequence. Figure 5 presents a UML object diagram that presents the components for one-half of the genetic toggle switch, including sequence components that are engineering abstractions of DNA, RNA, and protein, and generic components that represent small molecules and molecular complexes. Of these components, only the LacI-repressible gene and IPTG-LacI complex have any substructure. In particular, the gene's sequence is annotated with four other sequence components of type "DNA," while the complex is composed of a generic component of type "small molecule" and a sequence component of type "protein."

Functional Representation. To address the need for functional descriptions in SBOL, the proposed data model adds classes for modules, interactions, and models. These classes provide a firm basis for functional representation in SBOL without going so far as to create a new standard for mathematically modeling biology, as there already exist several established languages for doing so, from the *Systems Biology Markup Language* (SBML)[39] to CellML[40] and even MatLab.[41] Rather, these classes enable users of SBOL to group components that function together, describe the basic qualitative interactions between these components, and document references to standard mathematical models that are external to SBOL and that provide more detailed descriptions of component function. In other words, a module gathers together a set of component instantiations, a set of interactions between these component instantiations, and a set of references to external models that are expected to be consistent with the module's interactions.

Figure 6 provides a UML example of the interactions between the component instantiations of the LacI inverter module of the genetic toggle switch. In this diagram, the binding of LacI to IPTG is represented using a noncovalent binding interaction that has three participants, including LacI and IPTG participating as reactants and the IPTG-LacI complex participating as a product. The repression of transcription at the pLac promoter is represented using a repression interaction, with LacI serving as the repressor participant and pLac serving as the repressed participant. Lastly, the transcription and translation of TetR are represented in this module using a single genetic production interaction that abstracts away the presence of the intermediate TetR mRNA. If this additional detail becomes necessary, then a new module could be created that instantiates the same

dx.doi.org/10.1021/sb500176h | *ACS Synth. Biol.* 2015, 4, 57−71

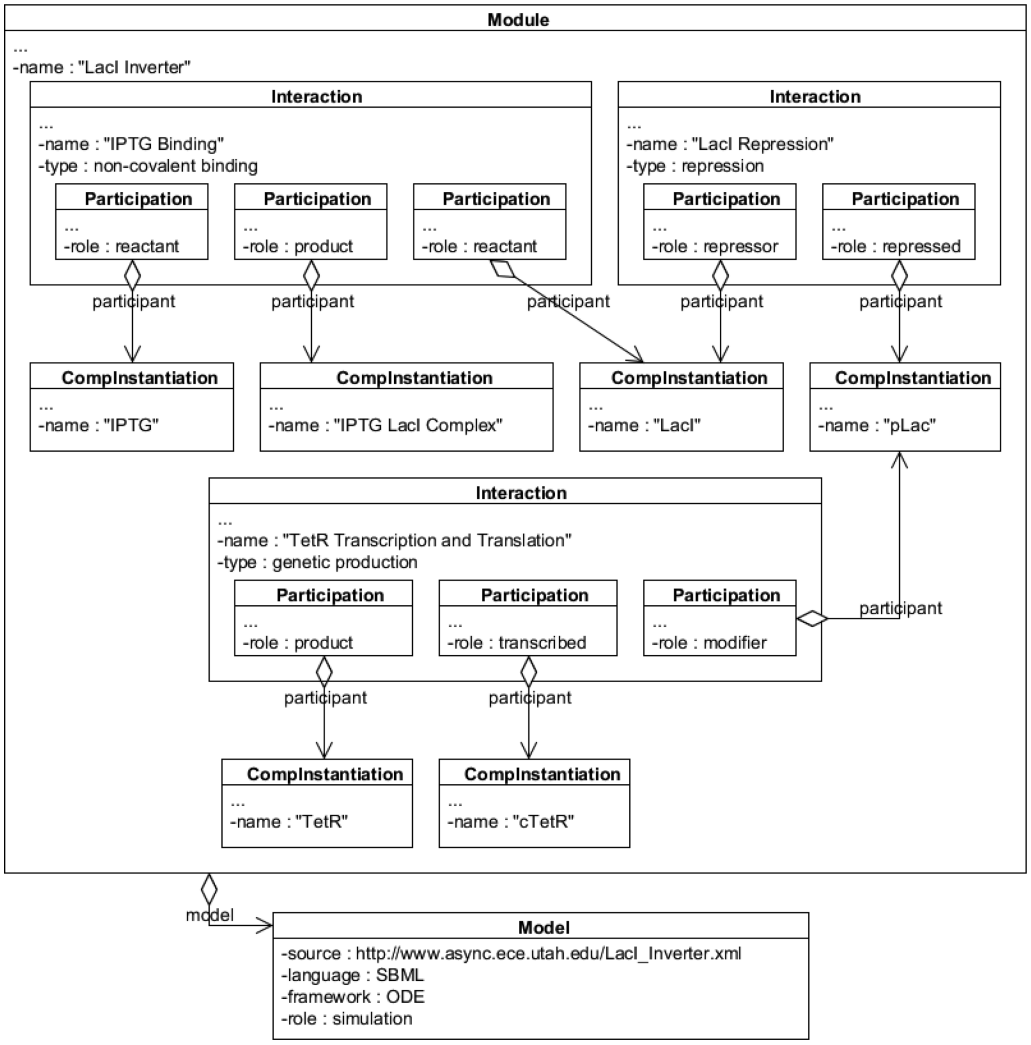


Figure 6. UML example displaying the interactions between the component instantiations in the LacI inverter module. In particular, there is an interaction representing the noncovalent binding of IPTG with the LacI protein, an interaction representing repression of the pLac promoter by LacI, and an interaction representing the production of TetR as coded for by the cTetR CDS and initiated by the pLac promoter. This module also references an external mathematical ODE model written in SBML for detailed simulation.

components alongside a TetR mRNA component instantiation and includes both transcription and translation interactions. In the current example, the genetic production interaction has three participants: pLac as a modifier, cTetR as a transcribed participant, and TetR as a product. Finally, the module can reference a Model object that points to an external model. In this example, the model source file is "LacI_Inverter.xml", it is written in the SBML language, it is an *ordinary differential equation* (ODE) model, and it is to be used for simulation.

From a given set of interactions, different GDA tools can derive different mathematical models at separate levels of mechanistic detail. In the near future, the proposed data model can be extended with the capacity to store data on measurements and basic parameters, thereby providing a firmer foundation for GDA tools to generate complementary mathematical models that nevertheless conform to the same basic data set. This capability is important because it enables function-oriented GDA tools to perform different functional design tasks with respect to the same genetic design.

Composition of Structure and Function. To enable the hierarchical, modular composition of genetic structure and function in SBOL, the proposed data model introduces classes for instantiation and port mapping. An instantiation is a documented reference to a specific component or module that effectively serves as a distinct copy and can be composed with other instantiations into a composite component or module. Currently, the proposed data model includes component instantiations and module instantiations. While a module can only be instantiated by another module, a component can be instantiated by either a module or another component, depending on its intended use. When a component is instantiated by another component, it is effectively referred to as a structural entity for the purpose of physical composition. When a component is instantiated by a module, on the other hand, it is referred to as a functional entity for the purpose of playing a role in an interaction.

In turn, ports and port maps enable connections between composite components and modules. Currently, port mapping serves two specific use cases related to the composition of genetic designs. The first use case is to indicate with greater fidelity how a module describes the function of a composite component, namely by asserting that particular component instantiations within the module are equivalent to particular component instantiations within the component.

dx.doi.org/10.1021/sb500176h | *ACS Synth. Biol.* 2015, 4, 57−71

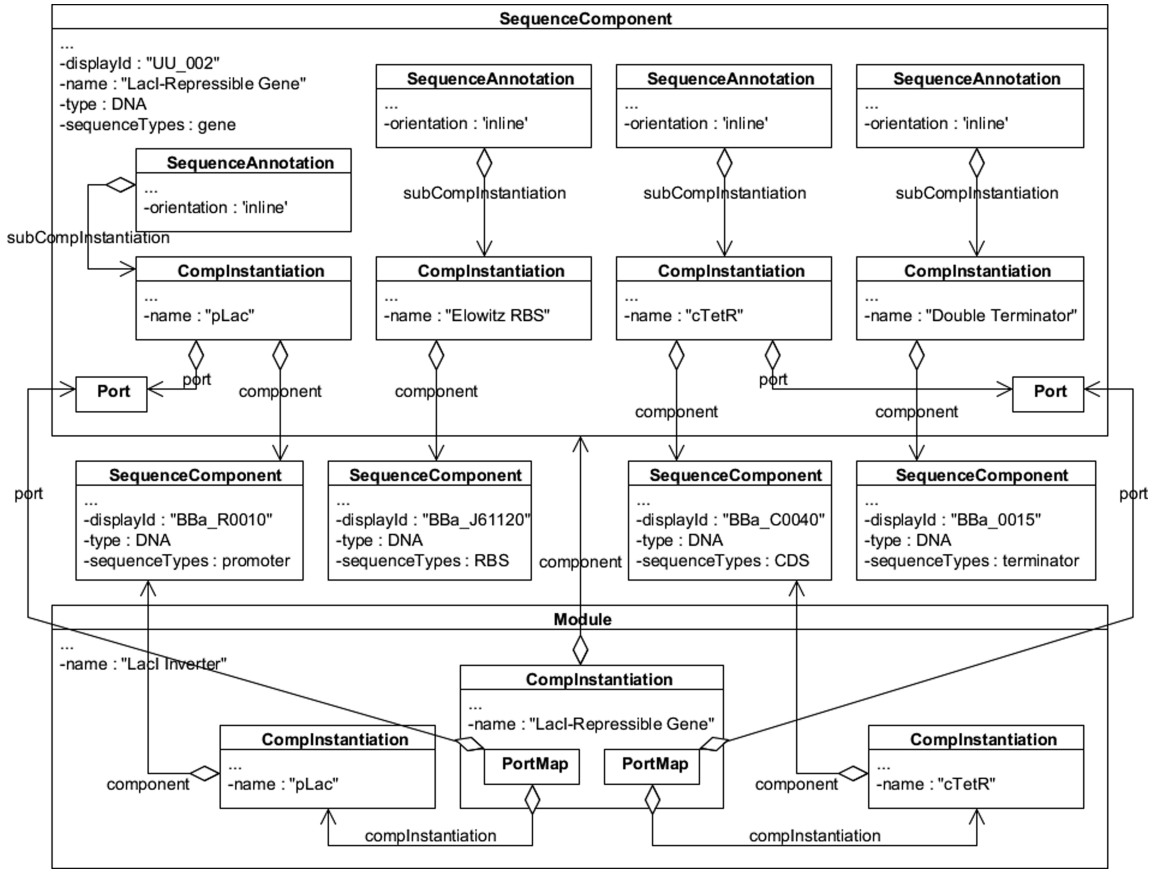


Figure 7. UML example of instantiating the LacI-repressible gene within the LacI inverter module. Port maps are used to indicate that the component instantiations of the pLac promoter in the LacI-repressible gene and in the LacI inverter module are equivalent. Similarly, port maps are used to indicate that the structural and functional component instantiations of the cTetR CDS are equivalent. "Comp" is short for "component" wherever it appears in this figure.

As an example of this use case, Figure 7 shows how one might compose the structure and function for the LacI-repressible gene of the genetic toggle switch. In this example, the LacI-repressible gene and two of its subcomponents are to be composed with the LacI inverter module, namely the pLac promoter and cTetR CDS. In order to compose these components with the LacI inverter module and indicate that it describes their behavior, they are functionally instantiated inside the module. In addition, port maps are placed on the functional instantiation of the LacI-repressible gene to connect between its subcomponent instantiations and the corresponding functional component instantiations in the module. Doing so makes it clear which component instantiations in the gene are being described by which component instantiations in the module.

This use case is most relevant when there is reason to believe that two structural instantiations of the same component should function differently based on physical location or other environmental context. For example, a polycistronic gene could contain two copies of a CDS, with one copy experiencing transcriptional repression due to its position downstream of the first copy. To capture such a scenario, there would need to be two functional component instantiations in a module that participate in different interactions and are separately mapped to the gene's two structural component instantiations.

The second use case of port mapping is to connect modules by asserting the equivalence of their component instantiations, effectively sharing these instantiations between modules. Figure 8 demonstrates how the LacI and TetR inverter modules can be composed into a toggle switch module using instantiation and connected using port mapping. In this example, the output of the LacI inverter is an input of the TetR inverter and vice versa. Also, both inverters accept the instantiation of a small molecule component as input, IPTG in the case of the LacI inverter and aTc in the case of the TetR inverter.

The primary reason for distinguishing between components and modules and port mapping between their instantiations is to promote the reuse of components. When the structural and functional layers of genetic design are kept separate, different researchers can use the same component in different modules to document its intended function for different engineering tasks and under different environmental conditions.

Ultimately, the concepts of instantiation and port mapping are not intended to directly represent biological reality. Rather, they are abstract artifacts that engineers use to organize their designs and enable reasoning over these designs by software. Without these concepts, it is very difficult to introduce the simplifying notions of hierarchy and modularity to genetic design in a manner that is conducive to the application of GDA software tools and the exchange of data between them. As progress in synthetic biology continues and the scale of genetic design becomes more ambitious, GDA tools that support hierarchical, modular standards will be useful, if not necessary, for managing the complexity of synthetic biological systems.

Examples. As a further demonstration of the utility of the proposed data model, the next two subsections present examples of designs for real-world synthetic biological systems

dx.doi.org/10.1021/sb500176h | ACS Synth. Biol. 2015, 4, 57−71

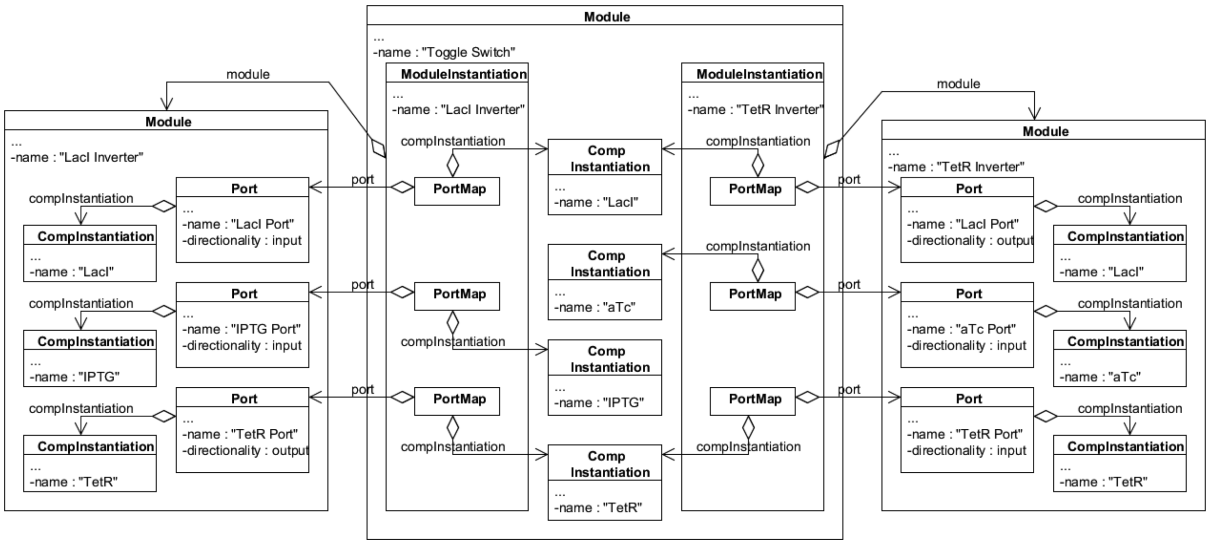


Figure 8. UML example of composing the LacI and TetR inverter modules into a toggle switch module. The LacI, IPTG, and TetR component instantiations of the LacI inverter module are mapped to the equivalent component instantiations in the toggle switch module, as are the TetR, aTc, and LacI component instantiations of the TetR inverter module. The end result is a composite toggle switch module that, if flattened into a noncomposite module, would include a single copy of each of these component instantiations and their accompanying interactions (see Figure 6). "Comp" is short for "component" wherever it appears in this figure.

that it can represent. These include a *RNA replicon* expression system[42] and a regulatory cascade based on *clustered regularly interspaced short palindromic repeat* (CRISPR) systems.

RNA Replicons. In this system, three different RNA replicons based on the Sindbis virus[43] are transfected into the same host. Consequently, the expression of these replicons is modulated via their competition for the same translation resources, in a manner that is dependent on their relative initial dosages. The expression of the payload of each individual replicon is accomplished in two phases. In the first phase, the nonstructural proteins (nsPs) at the 5′ end of the replicon are translated by the host to form a replicase. In the second phase, the replicase transcribes copies of the replicon, including shortened copies that only contain the payload and are produced when the replicase binds to the subgenomic promoter (SGP) at the end of the nsP block. Lastly, the third phase concerns the translation of the shortened copies, thereby expressing the payload (in this case, a fluorescent protein) in the place of structural proteins that would form the capsid of the virus.

As shown in Figure 9, the basic genetic structure and function of the replicon expression system can be represented using the proposed data model. In this design, an RNA component with an unspecified payload sequence serves as a structural template for the three RNA replicons. In turn, this RNA component is instantiated within a module that serves as a functional template for the replicons and asserts the key interaction of the host translation resources with their payload CDS. Finally, the mixed replicon expression system as a whole is composed by instantiating three submodules, each of which maps its fluorescent protein payload and CDS to the appropriate ports on an instantiation of the generic replicon expression module. This effectively documents that the mixed replicon expression module contains three separate copies of the generic replicon expression module, each with a different fluorescent protein payload. While the initial dosages for each replicon are outside the scope of the proposed data model, they can still be captured as custom annotations on the mixed expression module or within a mathematical model that is referenced by the module via the SBOL Model class.

CRISPR Cascade. The second example is drawn from a system originally constructed and presented by Kiani et al.[44] In this CRISPR-based regulatory cascade, transcriptional repression is accomplished using catalytically inactive Cas9 protein (Cas9m). Like many other TFs, Cas9m sterically blocks transcription initiation, but unlike other TFs, it is targeted to specific promoters via guide RNA (gRNA) molecules that allow for easier generation of orthogonal regulators. In the present example, there are two promoters that are serially repressed in this manner but are targeted via different gRNA molecules. More specifically, CRP-a is targeted by gRNA-a and initiates transcription of gRNA-b, which is coexpressed with the fluorescent protein mKate as intronic gRNA (igRNA). In turn, CRP-b is targeted by gRNA-b and initiates the transcription of EYFP. Since gRNA-a is constitutively transcribed in this system, the expression of gRNA-b and mKate are repressed and EYFP is produced in relatively larger quantities.

Figure 10 demonstrates one possible way in which the CRISPR cascade can be specified using the proposed data model. In this design, there are four submodule instantiations, three of which encompass a DNA component with an unspecified CDS and an RNA or protein product. These submodules are connected in series via port mapping so that the unspecified CDS and product of one submodule are equivalent to the specified CDS and product of the next submodule and the overall parent CRISPR cascade module. The latter module also instantiates DNA components that produce Cas9m and the activator TF Gal4VP16, which are then mapped as inputs to the two modules that represent CRISPR-based repression at CRP-a and CRP-b. In this way, the CRISPR cascade module serves a common source of regulators for any and all CRISPR-based modules that it instantiates.

DISCUSSION

This paper presents a set of extensions to the SBOL Version 1.1 data model which, if adopted by the community, should

dx.doi.org/10.1021/sb500176h | ACS Synth. Biol. 2015, 4, 57−71

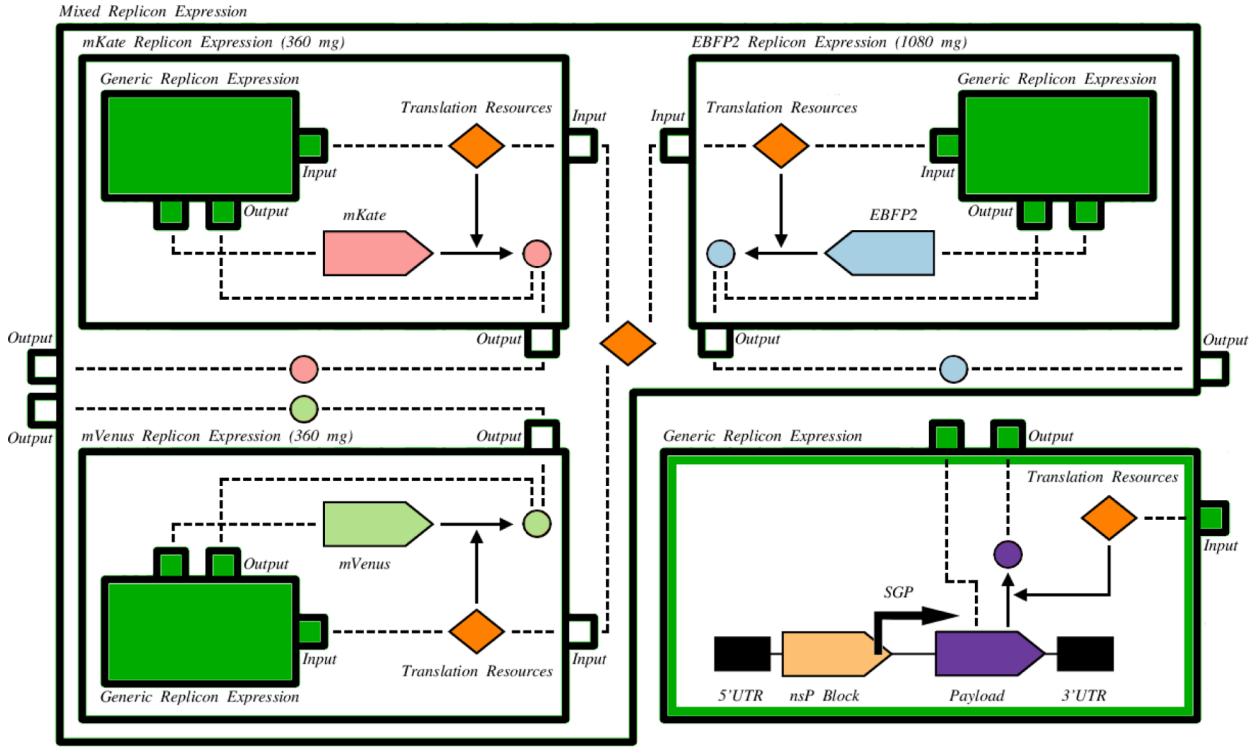


Figure 9. Mixed replicon expression module that instantiates three different replicon expression submodules, which in turn instantiate copies of a generic replicon expression module. Port mapping is used to customize each copy of the generic expression module so that it contains a different payload CDS and produces a different fluorescent protein. Port mapping is also used to indicate that these submodule instantiations share translation resources.

provide a means of expressing and composing genetic designs exhibiting a wide range of structure and function. This data model represents a conservative extension of the current model, striking a balance between expressiveness and minimization of complexity. In particular, this proposal avoids, to the extent possible, either making representational commitments where there is not yet scientific consensus or duplicating other modeling and standardization efforts.

In order to test the utility of this new data model, a new version of the Java library, libSBOLj, has been implemented and is being utilized to construct the above-described use cases and other genetic designs from the literature. In conjunction with further discussions in the community, this experimentation will hopefully allow for the resolution of any remaining details so that a formal specification can be written and ratified by the SBOL Developers Group. Once ratified, the specification becomes official when at least two software tools have implemented the standard and demonstrated the exchange of data.

Even if this proposal is accepted, there are still important aspects of engineering genetic designs not yet captured by SBOL. In particular, the proposed extensions to SBOL do not explicitly address the complex relationship between environmental context and its influence on the intended function of a design. Such specifications can become quite important when composing modules, as not all of them function correctly when deployed in the same environment or host organism, nor are they amenable to the same experimental techniques. Furthermore, the proposed data model does not capture protocols for experiments or physical assembly of designs. More research is necessary to identify the types of data related to context, assembly, and experiments that can be incorporated into SBOL and reasoned over by software. With these additions, SBOL will be able to better facilitate the specification of genetic designs and their deployment and testing in the lab.

METHODS

This section describes in detail the proposed data model for the next version of SBOL. In order to provide a more comprehensive standard for design in synthetic biology, this data model extends the range of genetic structure and function that can be represented in SBOL by including more general component classes and classes for modules, interactions, and models. In addition, this data model enables the hierarchical, modular composition of descriptions of genetic structure and function by introducing the abstract concepts of instantiation and port mapping from electrical and computer engineering.

Identified, Documented, and Collection. One minor improvement made by the proposed data model is the creation of two abstract classes, the Identified and Documented classes. As shown in Figure 11, these classes enable more efficient representation and implementation of SBOL by separating out data fields that are common to many classes and placing them into super classes that other classes may extend. The Identified class contains two data fields. The first is a URI that serves to identify the objects of any class that implements the Identified class, in the same way that data objects are identified with URIs in SBOL Version 1.1. The second is an annotation string that may contain a user's custom data that is not explicitly captured by SBOL. This string must take the form of one or more predicate-object pairs that adhere to the guidelines for the *Resource Description Framework* (RDF)[45] language in which SBOL is written.

The Documented class contains three data fields: a display ID, a name, and a description. The contents of these data fields

dx.doi.org/10.1021/sb500176h | *ACS Synth. Biol.* 2015, 4, 57−71

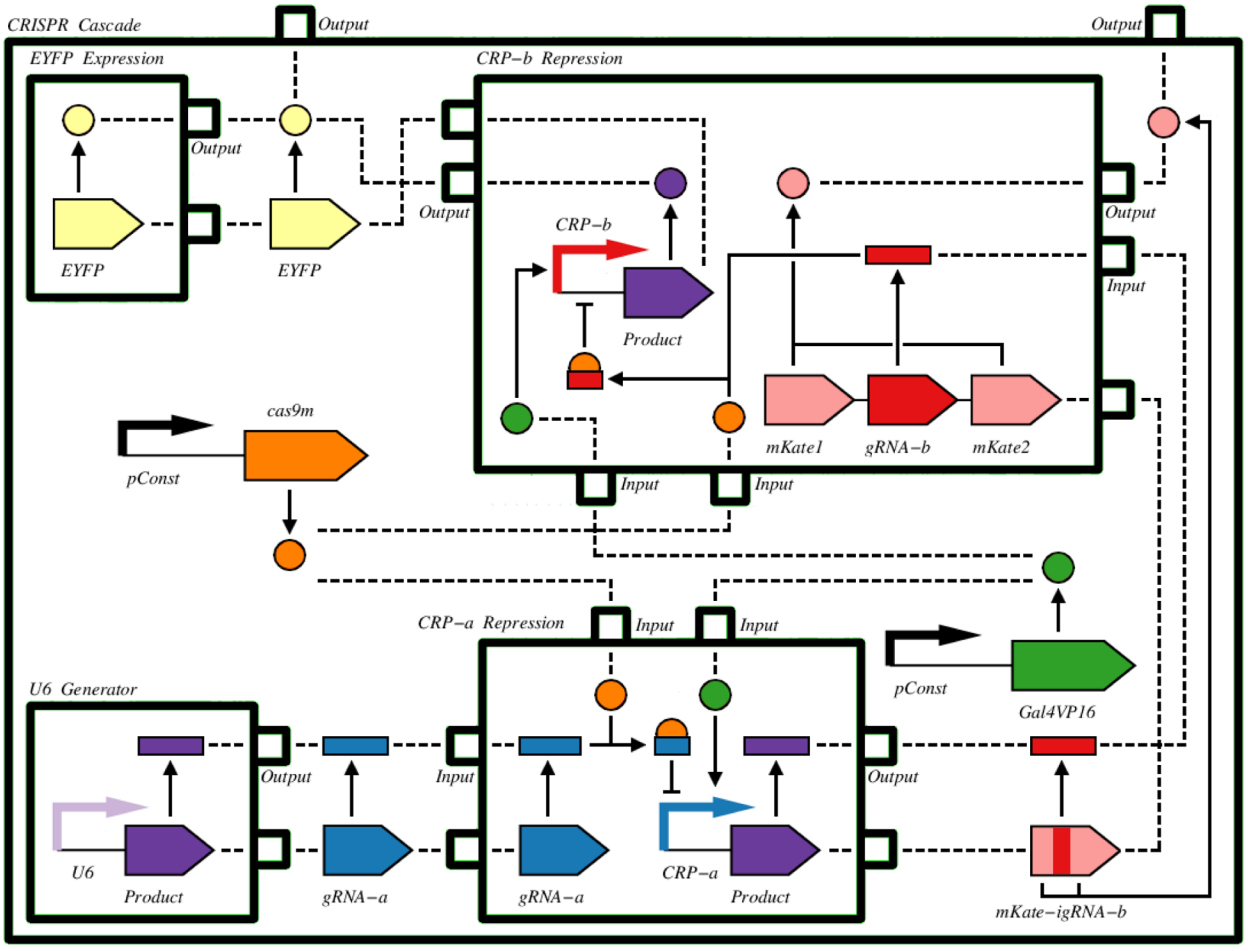


Figure 10. CRISPR cascade module that instantiates four submodules and several components. In this example, port mapping is used to specify the precise downstream components that are the outputs of each module.

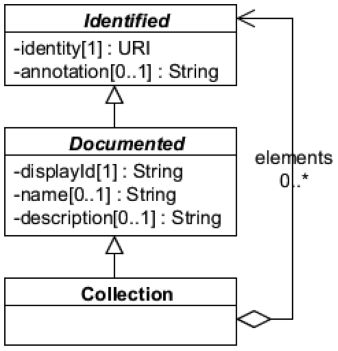


Figure 11. UML diagram for the Identified, Documented, and Collection classes of the proposed data model. Note that classes with italicized names are abstract classes that are meant to be extended by other classes and not used directly.

are identical to those of the same name in SBOL Version 1.1. Note that the Documented class inherits from the Identified class since, while all classes in SBOL are Identified classes, not all of them are Documented classes, such as the Sequence class. Rather, sequences are effectively documented by the sequence components that abstract them for the purpose of engineering design.

Finally, Figure 11 also contains an example of class from SBOL Version 1.1 that is now documented: the Collection class. Under the proposed data model, objects of this class can contain one or more objects that inherit from the Identified class. In other words, a collection may now contain one or more SBOL objects of any class from the proposed data model.

Components. Under the proposed data model, DNA components have been generalized to components with a sequence, or sequence components. The Sequence Component class captures previously unrepresented genetic components, such as RNA and protein components, but is also sufficiently general to represent nongenetic components with a sequence, such as nonbiological polymers. In order to capture components without a sequence, such as small molecules, molecular complexes, and light, a Generic Component class has also been introduced. As shown in Figure 12, both classes inherit from an abstract Component class that may refer to one or more subcomponent instantiations and must contain a type URI that refers to a term from an appropriate ontology, such as *Chemical Entities of Biological Interest* (ChEBI).[46] This type URI documents the basic sort of biochemical or physical entity (for example, DNA) that a component abstracts for the purpose of engineering design. The sequence type URIs of a sequence component, on the other hand, are analogous to the type URIs of a DNA component in SBOL Version 1.1 (see Figure 3). When possible, the sequence type URIs are expected to reference SO[36] terms to clarify the role or nature of the sequence that is abstracted by the component. For example, a sequence component of type DNA may have a sequence type of "promoter"

dx.doi.org/10.1021/sb500176h | *ACS Synth. Biol.* 2015, 4, 57–71

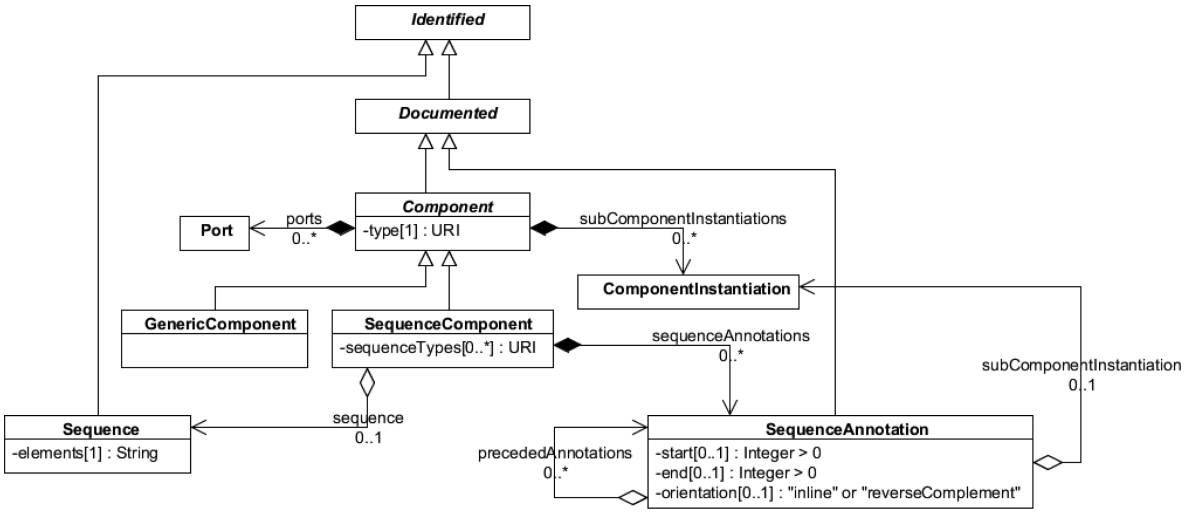


Figure 12. UML diagram for the proposed generalized Component classes. Since Generic Component and Sequence Component inherit from the abstract Component class, objects belonging to these classes can aggregate one or more objects that belong to the Port and Component Instantiation classes. Sequence components may additionally aggregate one or more objects of the Sequence Annotation class and up to one object of the Sequence class. In turn, a sequence annotation can refer to a component instantiation to effectively position it on the sequence of its parent sequence component.

or "terminator", while a sequence component of type protein may have a sequence type of "binding site" or "protease site."

Similar to a DNA component in the SBOL Version 1.1 data model, a sequence component can refer to sequence annotations to document the absolute or relative positions of subcomponent instantiations along its sequence. Unlike SBOL Version 1.1, sequence annotations do not directly refer to subcomponents but rather to instantiations or usages of these subcomponents that may be exposed via ports and mapped to other component instantiations for the purpose of design composition. Finally, a sequence component can refer to an object of the Sequence class that contains a string of characters encoding its elements. A sequence's string encoding must adhere to the *IUPAC codes* for the types of sequence components that refer to the sequence. For example, a sequence that is referred to by DNA components should contain a string of IUPAC-approved characters[47] that represent different nucleotides.

While this data model can be further extended by dividing sequence components into DNA, RNA, and protein components and adding data structures for small molecules and environmental factors, care must be taken to avoid creating a data model that is overly refined. Such a data model would have many classes, but no data-specific reason to distinguish between them. In the case of DNA, RNA, and protein components, however, there may be near-term reasons to distinguish among them, such as the different elements that make up their sequences and the single-strandedness of protein components, reasons that restrict the contents of the proposed Sequence and Sequence Annotation classes.

The alternative approach is to supplement the proposed data model with validation rules. For example, rules for checking that sequence components of type "protein" are only annotated with other sequence components of type "protein," that the orientation of their sequence annotations is always set to "inline," and that their sequences only contain characters taken from the IUPAC amino acid code. As the proposed data model continues to be implemented for testing, the SBOL community intends to explore both approaches.

Structural Instantiation. The structural composition of components is enabled through component instantiations.

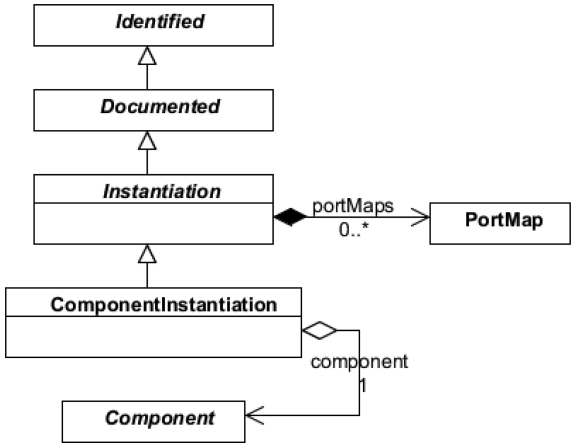


Figure 13. UML diagram for the proposed Component Instantiation class. As an Instantiation class object, a component instantiation is allowed to aggregate port maps to connect any ports on the component that it instantiates.

Under the SBOL Version 1.1 data model, composite DNA components are composed by annotating their sequences with other DNA components. As shown in Figures 12 and 13, this composition pattern is also true under the proposed data model, but the Sequence Annotation class now refers to an object of the Component Instantiation class, thereby explicitly documenting that a sequence annotation positions a particular instance or usage of a component, rather than the component itself. This distinction is necessary to allow different copies of a component to be referred to and treated differently on the basis of their physical location or other environmental context. In addition, by generalizing the concept of component instantiation, the proposed data model allows generic components without a sequence to be composed from instances of other components.

Modules and Module Instantiations. Beyond the component-based representation of genetic structure in the proposed data model, modules are used to group components that work together to provide an intended function. As displayed

dx.doi.org/10.1021/sb500176h | ACS Synth. Biol. 2015, 4, 57−71

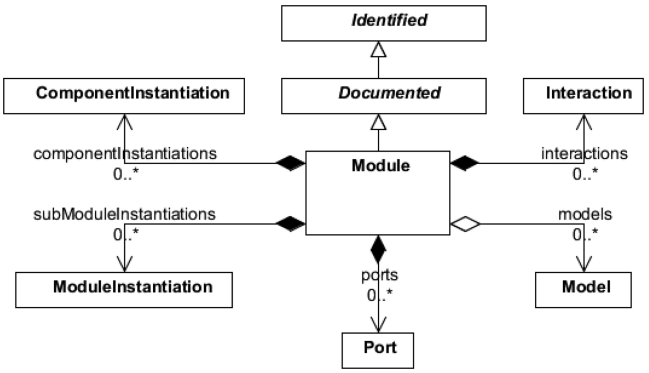


Figure 14. UML diagram for the proposed Module class. Note that data objects belonging to the Component Instantiation, Module Instantiation, Interaction, and Port classes are owned by a given module and no other object. Data objects belonging to the Model class, however, may be aggregated by more than one module.

in Figure 14, the Module class forms the hub for functional description of genetic designs. A module aggregates zero or more component instantiations, module instantiations, interactions, models, and ports. A component instantiation inside a module refers to a component as a functional entity for the purpose of playing a role in an interaction (described in more detail below). In this way, a module instantiates components that work together to perform an intended function.

Module instantiations (see Figure 15) enable the composition of modules from other modules. As described later on, the

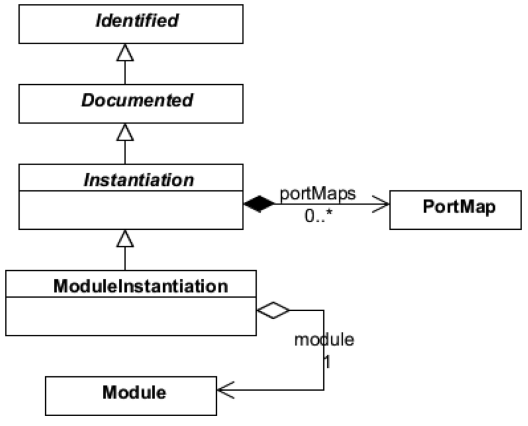


Figure 15. UML diagram for the proposed Module Instantiation class. A module instantiation is allowed to aggregate port maps to connect any ports on the module that it instantiates.

connection of the module instantiations within these modules is accomplished via ports and port mapping.

Ports and Port Maps. Connections between instantiations are achieved using ports and port maps. As depicted in Figure 16, a port refers to a component instantiation, thereby exposing it for port mapping. In addition, a port is allowed to have a directionality URI that indicates whether it is an input or output port by referencing the appropriate term from the Systems Biology Ontology (SBO).[48] However, owing to the reversibility of many biochemical reactions and the tight integration of genetic components with their environment, it is important to note that the directionality of a port is only expected to document a designer's intent and does not necessarily reflect biological reality.

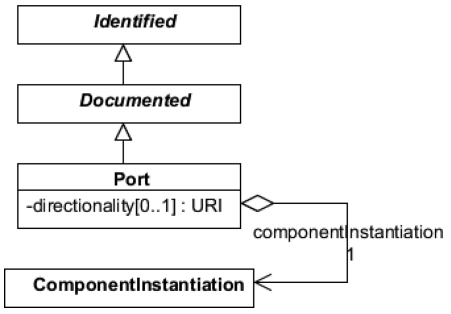


Figure 16. UML diagram for the proposed Port class. Note that a port is documented to better describe a designer's intent in exposing a given component instantiation.

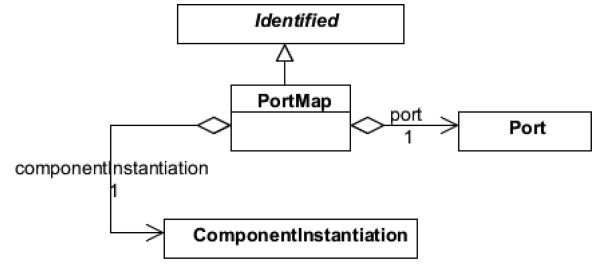


Figure 17. UML diagram for the proposed Port Map class. Unlike a port, a port map is identified rather than documented, as it simply represents a connection between a component instantiation and a port.

Figure 17 indicates that a port map refers to a component instantiation and a port, thereby asserting that its component instantiation is equivalent to that referenced by the port. When components referenced by mapped component instantiations have different identities, their respective data fields are to be interpreted in combination. While this interpretation may be ambiguous in the case of two sequence components with different sequences, it is useful when one of the two sequence components lacks a sequence, in which case a port mapping effectively supplies a sequence to fill in a partial design.

Interactions. Interactions provide a qualitative basis for asserting the intended function of a genetic design. The proposed data model supports regulatory interactions, such as activation or repression, and processes from the central dogma of biology, such as transcription and translation. Other supported interaction types include noncovalent binding between a small molecule and TF and phosphorylation of a TF by an enzyme. Each interaction is a nonempty set of participating component instantiations, each having a specific role in the interaction. As illustrated in the UML class diagram of Figure 18, each interaction must document its type by referencing a SBO term.

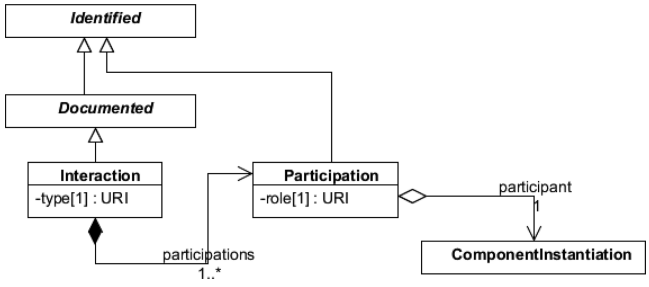


Figure 18. UML diagram for the proposed Interaction classes. The Interaction class aggregates one or more objects of the Participation class, which in turn reference objects of the Component Instantiation class.

dx.doi.org/10.1021/sb500176h | ACS Synth. Biol. 2015, 4, 57−71

In the proposed data model, an interaction refers to the involved components indirectly via the Participation class. A participation has a role URI that is expected to also reference an SBO term to specify the role of each component in an interaction. For example, a protein component instantiation that participates in a repression interaction has the role of a "repressor," while the role of a promoter DNA component instantiation in the same interaction is "repressed."

Models. Instead of introducing a new language for the specification of mathematical models of biology, the proposed data model leverages existing standards and refers to them via the Model class. As shown in Figure 19, each object that belongs to the Model class is required to refer by means of URIs to a source model and ontology terms that document the source model's language, framework, and role. In this way, there is minimal duplication of standardization efforts and users of SBOL can specify the quantitative function of their modules in a well-developed language of their choice. A module can refer to more than one model since each model can encode different levels of functional detail and play different roles in engineering design. Examples of languages for mathematically modeling for biology include SBML[39] and CellML.[40] Modeling frameworks include ODEs, stochastic processes, and Boolean networks. Lastly, examples of modeling roles include simulation, verification, and synthesis (building composite models from simpler models). One possible source of terms for modeling frameworks and roles is the Mathematical Modeling Ontology (MAMO),[49] though it is currently in the early stages of its development.

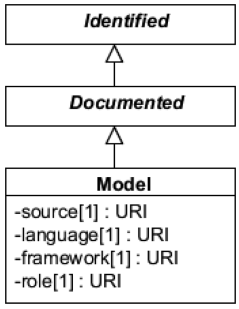


Figure 19. UML diagram for the proposed Model class. A SBOL model class object documents and refers to an external mathematical model.

Summary of Proposed Data Model. As summarized in the UML class diagram shown in Figure 20, the proposed data model expands the total number of classes in SBOL from four to 17 (four of these classes, the Identified, Documented, Collection, and Generic Component classes, are omitted from the figure for clarity). Central to this data model are the Component and Module classes, which are the basic exchangeable units for composing descriptions of genetic structure and function. A module composes components and other modules by means of the Component Instantiation and Module Instantiation classes and describes their function by aggregating objects belonging to the Interaction and Model classes. A component that belongs to the Sequence Component class refers to an object of the Sequence class and composes its subcomponent instantiations along its sequence via objects of the Sequence Annotation class. Once components and modules have been composed using the various Instantiation classes, their component instantiations can be connected using the Port and Port Map classes.

AUTHOR INFORMATION

Corresponding Author

*Email: n.roehner@utah.edu.

Notes

The authors declare no competing financial interest.

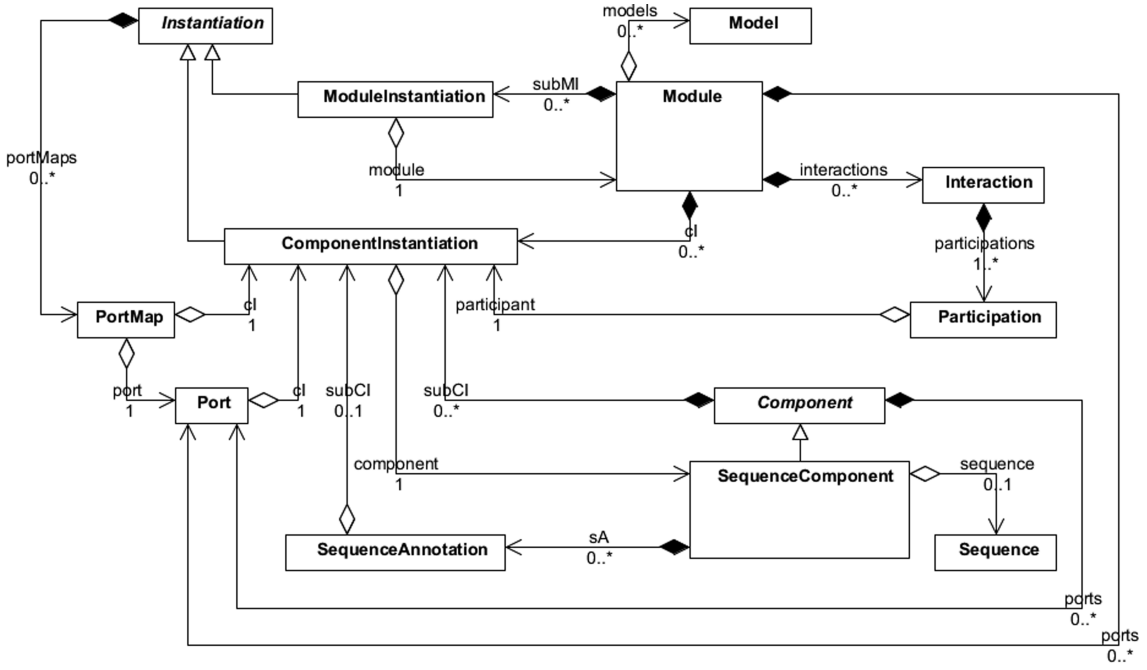


Figure 20. UML diagram that summarizes the proposed data model. In this figure, "cI" stands for "componentInstantiation," "subCI" stands for "subComponentInstantiation," and "subMI" stands for "subModuleInstantiation".

■ ACKNOWLEDGMENTS

We thank all of the attendees (which include the authors) of the 10th SBOL Workshop held at UC Berkeley, January 7−9, 2014, namely Nathan Hillson, Kevin Costa, Evan Appleton, Leandro Watanabe, Jeff Johnson, Robert Sidney Cox, Joanna Chen, Deepak Chandran, Cesar Rodriguez, David Lomelin, Michal Galdzicki, Linh Huynh, Darren Platt, Jacqueline Quinn, Chris Anderson, Aaron Berliner, and Mike Fero for their valuable comments on this proposal during the workshop. We also thank Douglas Densmore of Boston University and all members of the SBOL Developers Group for their intellectual contributions to the development of the SBOL standard. This material is based upon work supported by the National Science Foundation under Grant No. CCF-1218095. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

■ REFERENCES

(1) Galdzicki, M., et al. (2014) The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat. Biotechnol. 32*, 545−550.

(2) Villalobos, A., Ness, J., Gustafsson, C., Minshull, J., and Govindarajan, S. (2006) Gene Designer: A synthetic biology tool for constructing artificial DNA segments. *BMC Bioinf. 7*, 285.

(3) Chandran, D., Bergmann, F. T., and Sauro, H. M. (2009) TinkerCell: Modular CAD tool for synthetic biology. *J. Biol. Eng. 3*, 19.

(4) Cai, Y., Wilson, M. L., and Peccoud, J. (2010) GenoCAD for iGEM: A grammatical approach to the design of standard-compliant constructs. *Nucleic Acids Res. 38*, 2637−2644.

(5) Beal, J.; Lu, T., and Weiss, R. (2011) Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. *PLoS One 6*, DOI: 10.1371/journal.pone.0022490.

(6) Bilitchenko, L.; Liu, A.; Cheung, S.; Weeding, E.; Xia, B.; Leguia, M.; Anderson, J. C., and Densmore, D. (2011) Eugene—A domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLoS One 46*, DOI: 10.1371/journal.pone.0018882.

(7) Galdzicki, M.; Rodriguez, C.; Chandran, D.; Sauro, H. M., and Gennari, J. H. (2011) Standard Biological Parts Knowledgebase. *PLoS One 6*, DOI: 10.1371/journal.pone.0017005.

(8) Misirli, G., Hallinan, J. S., Yu, T., Lawson, J. R., Wimalaratne, S. M., Cooling, M. T., and Wipat, A. (2011) Model annotation for synthetic biology: Automating model to nucleotide sequence conversion. *Bioinformatics 27*, 973−979.

(9) Xia, B., Bhatia, S., Bubenheim, B., Dadgar, M., Densmore, D., and Anderson, J. C. (2011) Developer's and user's guide to Clotho v2.0. *Methods Enzymol. 498*, 97−135.

(10) Ham, T. S.; Dmytriv, Z.; Plahar, H.; Chen, J.; Hillson, N. J., and Keasling, J. D. (2012) Design, implementation and practice of JBEI-ICE: An open source biological part registry platform and tools. *Nucleic Acids Res. 40*, DOI: 10.1093/nar/gks531.

(11) Hillson, N. J., Rosengarten, R. D., and Keasling, J. D. (2012) j5 DNA assembly design automation software. *ACS Synth. Biol. 1*, 14−21.

(12) Chen, J., Densmore, D., Ham, T. S., Keasling, J. D., and Hillson, N. J. (2012) DeviceEditor visual biological CAD canvas. *J. Biol. Eng. 6*, 1.

(13) Madsen, C., Myers, C., Patterson, T., Roehner, N., Stevens, J., and Winstead, C. (2012) Design and test of genetic circuits using iBioSim. *IEEE Design Test 29*, 32−39.

(14) Pearson, W. R., and Lipman, D. J. (1988) Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. U.S.A. 85*, 2444−2448.

(15) Bilofsky, H. S., and Christian, B. (1988) The GenBank genetic sequence data bank. *Nucleic Acids Res. 16*, 1861−1863.

(16) Densmore, D., and Hassoun, S. (2012) Design automation for synthetic biological systems. *IEEE Design Test Comput. 29*, 7−20.

(17) Goler, J. BioJADE: A design and simulation tool for synthetic biological systems. M.Sc. thesis, Massachusetts Institute of Technology, Cambridge, MA, 2004.

(18) Rodrigo, G., Carrera, J., and Jaramillo, A. (2007) Asmparts: Assembly of biological model parts. *Syst. Synth. Biol. 1*, 167−170.

(19) Funahashi, A., Matsuoka, Y., Jouraku, A., Morohashi, M., Kikuchi, N., and Kitano, H. (2008) CellDesigner 3.5: A versatile modeling tool for biochemical networks. *Proc. IEEE 96*, 1254−1265.

(20) Hill, A. D., Tomshine, J. R., Weeding, E. M., Sotirpoulos, V., and Kaznessis, Y. N. (2008) SynBioSS: The synthetic biology modeling suite. *Bioinformatics 24*, 2551−2553.

(21) Mirschel, S., Steinmetz, K., Rempel, M., Ginkel, M., and Gilles, E. D. (2009) ProMoT: Modular modeling for systems biology. *Bioinformatics 25*, 687−689.

(22) Richardson, S. M., Wheelan, S. J., Yarrington, R. M., and Boeke, J. D. (2006) GeneDesign: Rapid, automated design of multikilobase synthetic genes. *Genome Res. 16*, 550−556.

(23) Wu, G., Bashir-Bello, N., and Freeland, S. J. (2006) The Synthetic Gene Designer: A flexible web platform to explore sequence manipulation for heterologous expression. *Protein Expression Purif. 47*, 441−445.

(24) Umesh, P., Naveen, F., Rao, C., and Nair, A. (2010) Programming languages for synthetic biology. *Syst. Synth. Biol. 4*, 265−269.

(25) Roehner, N., and Myers, C. J. (2014) A methodology to annotate Systems Biology Markup Language Models with the Synthetic Biology Open Language. *ACS Synth. Biol. 3*, 57−66.

(26) Pedersen, M., and Phillips, A. (2009) Towards programming languages for genetic engineering of living cells. *J. R. Soc. Interface 6*, S437−S450.

(27) Yaman, F., Bhatia, S., Adler, A., Densmore, D., and Beal, J. (2012) Automated selection of synthetic biology parts for genetic regulatory networks. *ACS Synth. Biol. 1*, 332−344.

(28) Huynh, L., Tsoukalas, A., Koppe, M., and Tagkopoulos, I. (2013) SBROME: A scalable optimization and module matching framework for automated biosystems design. *ACS Synth. Biol. 2*, 1073−1089.

(29) Roehner, N., and Myers, C. J. (2014) Directed acyclic graph-based technology mapping of genetic circuit models. *ACS Synth. Biol.*, DOI: 10.1021/sb400135t.

(30) Galdzicki, M., et al. (2012) *Synthetic Biology Open Language (SBOL) Version 1.1.0. BBF RFC 87*, DOI: 1721.1/73909.

(31) Quinn, J., Beal, J., Bhatia, S., Cai, P., Chen, J., Clancy, K., Hillson, N. J., Galdzicki, M., Maheshwari, A., Umesh, P., Pocock, M., Rodriguez, C., Stan, G.-B., and Endy, D. (2013) *Synthetic Biology Open Language Visual (SBOL Visual), Version 1.0.0. BBF RFC 93*, DOI: 1721.1/78249.

(32) Gardner, T. S., Cantor, C. R., and Collins, J. J. (2013) Construction of a genetic toggle switch in *Escherichia coli*. *Nature 403*, 339−342.

(33) Bhatia, S., and Densmore, D. (2013) Pigeon: A design visualizer for synthetic biology. *ACS Synth. Biol. 2*, 348−350.

(34) Booch, G.; Rumbaugh, J., and Jacobson, I. (2005) *The Unified Modeling Language User Guide*, 2nd ed.; Addison-Wesley, Boston, MA.

(35) Berners-Lee, T.; Fielding, R., Masinter, L. Uniform Resource Identifier (URI): Generic syntax. IETF RFC 3986, 2005; http://tools.ietf.org/html/rfc3986, accessed on May 31, 2014.

(36) Eilbeck, K., Lewis, S. E., Mungall, C. J., Yandell, M., Stein, L., Durbin, R., and Ashburner, M. (2005) The Sequence Ontology: A tool for the unification of genome annotations. *Genome Biol. 6*, R44.

(37) Lord, P., and Stevens, R. (2010) Adding a little reality to building ontologies for biology. *PLoS One 5*, e12258 DOI: 10.1371/journal.pone.0012258.

(38) Fire, A., Xu, S., Montgomery, M. K., Kostas, S. A., Driver, S. E., and Mello, C. C. (1998) Potent and specific genetic interference by double-stranded RNA in *Caenorhabditis elegans*. *Nature 391*, 806−811.

(39) Hucka, M., et al. (2003) The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics 19*, 524−531.
(40) Hedley, W. J., Nelson, M. R., Bellivant, D. P., and Nielsen, P. F. (2001) A short introduction to CellML. *Philos. Trans. R. Soc. London A 359*, 1073−1089.
(41) *MATLAB*, version 8.3 (R2014a). (2014) The MathWorks Inc.: Natick, MA.
(42) Beal, J.; Wagner, T. E.; Kitada, T.; Krivoy, A.; Azizgolshani, O.; Parker, J. M.; Densmore, D., Weiss, R. Model-driven engineering of gene expression from RNA replicons. Personal communication, on May 29, 2014.
(43) Frolov, I., Hardy, R., and Rice, C. M. (2001) Cis-acting RNA elements at the 5′ end of Sindbis virus genome RNA regulate minus- and plus-strand RNA synthesis. *RNA 7*, 1638−1651.
(44) Kiani, S., Beal, J., Ebrahimkhani, M. R., Huh, J., Hall, R. N., Xie, Z., Li, Y., and Weiss, R. (2014) CRISPR transcriptional repression devices and layered circuits in mammalian cells. *Nat. Methods*, DOI: 10.1038/NMETH.2969.
(45) Lassila, O., Swick, R. RDF/XML syntax specification (revised). *W3C Recommendation*, 2004; http://www.w3.org/TR/rdf-syntax-grammar/, accessed on May 21, 2014.
(46) Hastings, J., de Matos, P., Dekker, A., Ennis, M., Harsha, B., Kale, N., Muthukrishnan, V., Owen, G., Turner, S., Williams, M., and Steinbeck, C. (2013) The ChEBI reference database and ontology for biologically relevant chemistry: Enhancements for 2013. *Nucleic Acids Res. 41*, D456−D463.
(47) Cornish-Bowden, A. (1985) Nomenclature for incompletely specified bases in nucleic acid sequences: Recommendations 1984. *Nucleic Acids Res. 13*, 3021−3030.
(48) Juty, N.; and Novere, N. (2013) *Encyclopedia of Systems Biology*. Springer, New York, pp 2063−2063.
(49) Waltemath, D.; Zhukova, A.; Swat, M.; Lefranc, Y.; Vik, J.-O., and Novere, N. L. *Mathematical Modelling Ontology*. Available online: http://sourceforge.net/projects/mamo-ontology/, accessed on May 31, 2014.