VisBOL2 - Improving Web-based Visualization for Synthetic Biology Designs

Benjamin Hatch,[†] Linhao Meng,[‡] Jeanet Mante,[¶] James A. McLaughlin,[§] James Scott-Brown,[‖] and Chris J. Myers[*,¶]

†*University of Utah, Salt Lake City, 84112, UT, USA*

‡*Eindhoven University of Technology, 5612 AZ Eindhoven, Netherlands*

¶*University of Colorado Boulder, Boulder, 80309, CO, USA*

§*EMBL-EBI, Cambridge CB10 1SD, United Kingdom*

‖*University of Oxford, Oxford OX1 2JD, United Kingdom*

E-mail: chris.myers@colorado.edu

Abstract

VisBOL is a web-based visualization tool used to depict genetic circuit designs. This tool depicts simple DNA circuits adequately, but it has become increasingly outdated as new versions of SBOL Visual were released. This paper introduces VisBOL2, a heavily redesigned version of VisBOL that makes a number of improvements to the original VisBOL, including proper functional interaction rendering, dynamic viewing, a more maintainable code base, and modularity that facilitates compatibility with other software tools. This modularity is demonstrated by incorporating VisBOL2 into a sequence visualization plugin for SynBioHub.

Keywords

synthetic biology, visualization, sequence visualization, SBOL, SBOL Visual

Visual depiction is an essential tool for the development of engineered designs. Standardization of visual depictions enables them to be shared and universally understood. In the synthetic biology field, the *Synthetic Biology Open Language Visual* (SBOLv)[1,2] provides a standardized graphical notation for visualization of genetic circuits. SBOLv is a complementary standard to the *Synthetic Biology Open Language* (SBOL),[3] which provides a data format that stores both functional and structural information for a given synthetic biology design.

Several software tools have been developed that support both the SBOL and SBOLv standards. DNAplotlib[4] enables highly customizable visualization of individual genetic constructs and libraries of design variants. SBOLDesigner[5] is a genetic circuit editor that supports the construction of DNA-level designs encoded in SBOL using SBOLv symbols. GenoCAD[6] is a rule-based DNA design tool that allows users to define domain specific languages to design expression systems for particular applications. SBOLCanvas[7] is a visual 'drag-and-drop' tool for the creation of genetic constructs and related figures.

VisBOL[8] was one of the first visualization tools to directly support both SBOLv and SBOL. Implemented in JavaScript, VisBOL differs from the tools previously mentioned in that it provides automated visualization of SBOL files on the Web, and is designed to integrate into web-based applications such as SynBioHub.[9] Although VisBOL has undoubtedly been useful for visualizing simple DNA circuits, it has become increasingly outdated as new versions of SBOLv were released, such as SBOLv 2,[10] which introduced a standardized format for depicting the functional interactions of a genetic design. This paper presents VisBOL2, a heavily redesigned version that makes a number of improvements to the original VisBOL software. These improvements include extending support to genetic circuits with functional interactions, a dynamic user interface, a more maintainable code base, and improved modularity, which further facilitates incorporation into other web-based software tools.

Results

Here we describe VisBOL2, a re-architected version of VisBOL which resolves many of the issues identified since its publication and introduces a number of new features requested by the synthetic biology community.

Functional Interaction Rendering: The main issue that prompted a redesign of VisBOL was inadequate functional interaction visualization. SBOL 2 introduced the capability to represent functional interactions between different parts of a design. However, the original VisBOL was built to render only simple DNA circuits. Its rendering pipeline did not keep track of the location of glyphs (genetic parts in the design) once they were rendered, making it difficult to correctly position arrows representing functional interactions between the glyphs. In an effort to work around this problem, functional interactions were represented in separate sub-diagrams, alongside the diagram representing the physical structure of the circuit. Figure 1a shows an example of VisBOL attempting to render a genetic design with functional interactions.

VisBOL2 overcomes this issue by redesigning the rendering process and utilizing a different backing data structure that keeps track of the locations of all parts in the design relative to each other. This backing data structure is a directed graph, where individual parts (glyphs) are regarded as nodes and functional interactions are regarded as edges. For the purposes of determining the layout, all edges incident to a glyph on the nucleic acid backbone are considered to be directed away from that glyph, regardless of the physical direction of the functional interaction. Each glyph is assigned its own unique identifier, enabling seamless mapping of functional interaction participants to glyphs in the rendering. The back-end then identifies the "root glyphs," which are glyphs that have no incoming edges, and assigns a position to each. A recursive process is then called on each root glyph. In this process, the position of each glyph that is connected to the root glyph through an edge is determined relative to the root glyph's position. By the end of this process, all glyphs and their functional interactions are positioned in such a way that there is minimal to no

overlapping in the visualization. Figure 1b demonstrates a VisBOL2 rendering of the same genetic design that the original VisBOL attempts to render in Figure 1a. Figure 2 illustrates the VisBOL2 rendering process.

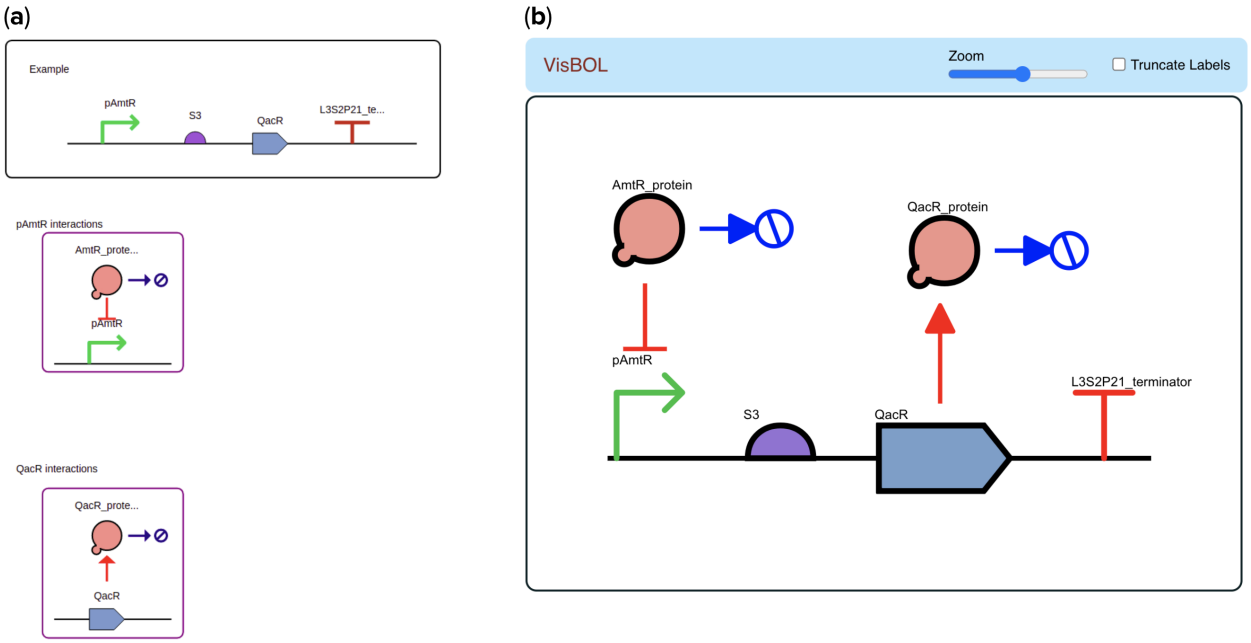

Figure 1: This figure shows the original VisBOL and VisBOL2's rendering of the same genetic design. **(*a*)** A screenshot of the original VisBOL rendering of a simple genetic design with functional interactions on SynBioHub. Note that functional interactions are rendered as separate sub-diagrams with large gaps between them. The promoter and coding sequence in the rendered functional interaction sections correspond to the promoter and coding sequence in the main circuit. **(*b*)** A VisBOL2 rendering of the same genetic design seen in Figure 1a. Note that functional interactions are rendered in the correct locations on a single diagram rather than being rendered separately. Also note that VisBOL2 allows the zoom level and part label truncation of the visualization to be dynamically adjusted.

Dynamic Viewing: Another important issue with the original VisBOL tool was its lack of visual customization. VisBOL did not support resizing or scaling of its depictions. This limitation became a problem when rendering complex genetic designs containing hundreds of glyphs, as their size made viewing the entire design at once impossible. An inability to zoom and scroll the diagram made it frustratingly difficult for users to track what section of the design they were viewing, thereby compounding the negative side effects of not being able to resize large renderings. VisBOL also did not support user control over the truncation

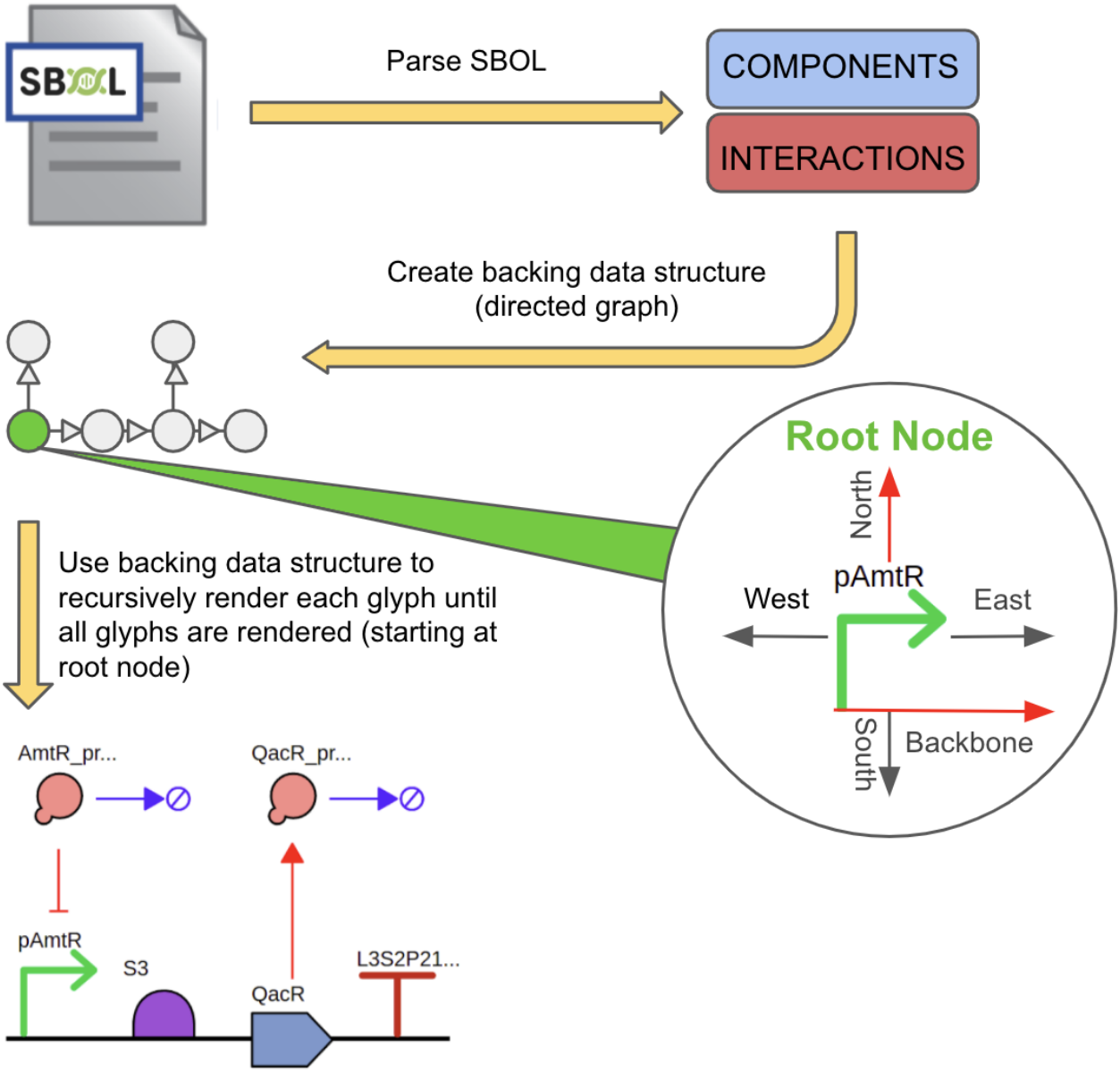

Figure 2: The VisBOL2 rendering process for an SBOL file. VisBOL2 creates a graph data structure, regarding part glyphs as nodes and interactions as edges. Edges are limited to four compass directions (as shown). The edge direction is determined by the roles of functional interaction participants and functional interaction types (e.g., a production interaction would have a "Northern" edge direction). The glyphs are rendered recursively, starting at the root node. In this example, the root node is pAmtR. It has two connected edges: an edge representing an inhibition interaction to the north, and an edge indicating that the S3 RBS resides east of pAmtR along the circuit's backbone. After the root node (pAmtR) is rendered, the nodes connected to the root node (S3 and the AmtR protein) are regarded as root nodes. This recursive process continues until all nodes have been rendered.

of part labels in its visualizations. This caused confusion when parts had similar names.

VisBOL2 supports dynamic scaling of its depictions by utilizing parametric SVG[11] in the display during rendering. Parametric SVG enables VisBOL2 to resize glyphs without restarting the entire rendering process, making resizing of the display efficient and seamless. To further reduce confusion when viewing the diagram, VisBOL2 allows users to dynamically toggle part label truncation. This allows users to distinguish between parts that have similar names. By enabling users to zoom in and out on the visualization and toggle part label truncation, VisBOL2 makes it easier for users to keep track of what section of the design is being viewed.

Maintainability: As the original VisBOL adapted to add capabilities of new SBOLv versions, the open-source codebase became increasingly difficult to maintain. Implementing significant changes or adding new features such as functional interaction rendering became challenging as there was not a strict separation between different concerns.

VisBOL2 implements a separation of concerns, strictly dividing the tasks of parsing design information, creating the backing data structure, and rendering the display. This abstraction barrier allows significant changes and new features to be implemented seamlessly, and should enable VisBOL2 to quickly add support for all future SBOLv versions.

Modularity: The strict separation between the VisBOL2 front-end and back-end enables other software tools to integrate with VisBOL2 for improved visualization. This is illustrated by VisBOL2 integration into the SynBioHub sequence view plugin[12] (SeqViz). In order to achieve this integration, the plugin first uses the VisBOL2 back-end to create a backing data structure. SeqViz traverses this data structure, mapping DNA sequences to specific parts (glyphs) in the genetic design. After passing the data structure to the VisBOL2 front-end, SeqViz then uses the VisBOL2 front-end API (see VisBOL2 documentation in supporting information) to highlight corresponding sequences when parts in the VisBOL2 rendering are selected. SeqViz also uses the front-end API to highlight corresponding parts in the VisBOL2 rendering when DNA sequences are hovered over. Figure 3 illustrates the

integration of SeqViz with VisBOL2 to create a sequence visualization plugin for SynBioHub.

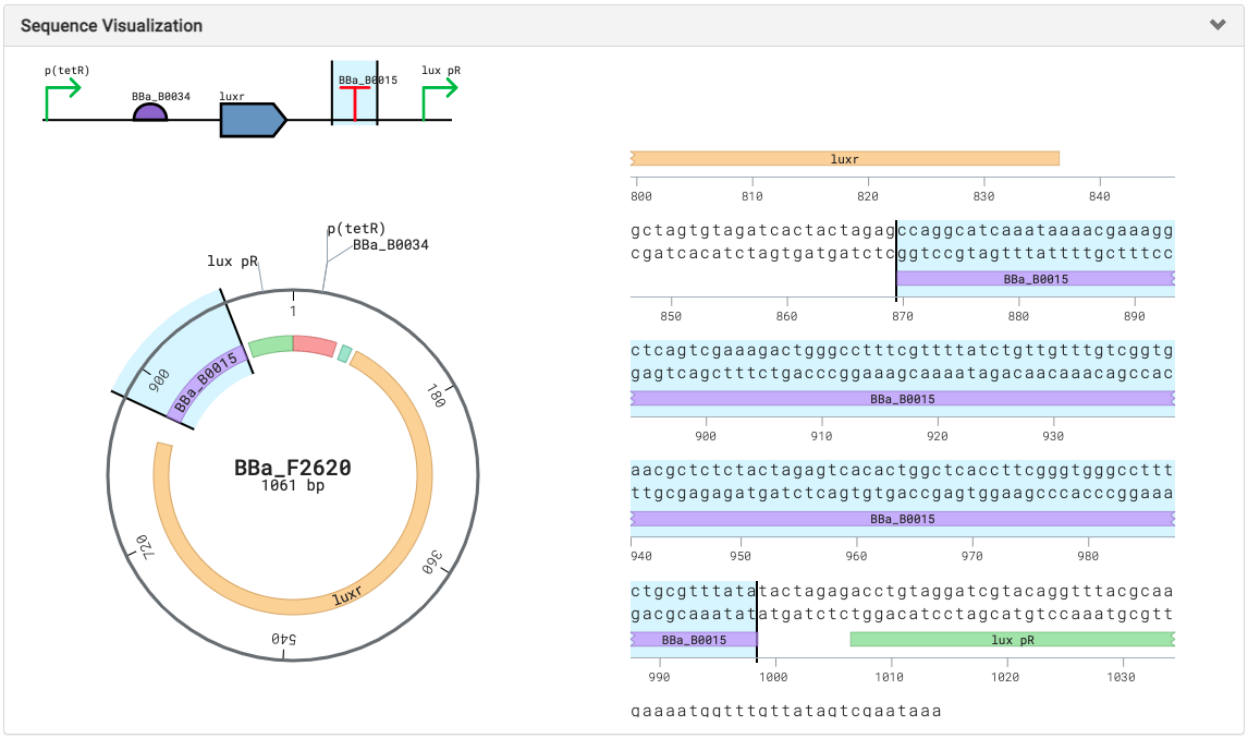

Figure 3: This figure shows the SynBioHub sequence visualization plugin rendering a genetic design. Note that along with rendering the sequence and plasmid view, the plugin also uses the VisBOL2 front-end to render the VisBOL2 view of the sequence. This view can be found in the top left corner of the visualization. The terminator in the VisBOL2 rendering has been clicked, causing its corresponding sequence to be highlighted in the plasmid and sequence views.

Methods

The VisBOL2 architecture adheres to a strict separation of concerns between the front-end and back-end. The web-based front-end utilizes the data produced by the back-end to create an interactive visualization. The back-end produces the data the front-end needs to render the display for a specific SBOL file, such as the position, color, and corresponding SVG for each glyph.

The front-end was built using React,[13] a JavaScript framework developed by Facebook. React was chosen as the framework for the VisBOL2 front-end for two main reasons. First,

React simplified the challenge of making the VisBOL2 visualization interactive. To enable interactive resizing of glyphs in the VisBOL2 display, react-parametric-svg,[14] an open-source react component specifically designed for rendering and resizing parametric SVGs, was used. React's use of a "virtual" document object model enabled these parametric SVGs to resize in response to user interaction without requiring that the entire web page to refresh. Second, building the front-end in React facilitated easier integration between the VisBOL2 rendering software and other web-based tools. The integration of VisBOL2 with the SynBioHub sequence view plugin was aided by the fact that the sequence view plugin, along with many modern web applications, is also built using React. However, it is important to note that integration with the VisBOL2 front-end is not only available for other React applications, since the React framework allows React components to easily be injected into websites that do not use React. The fact that VisBOL2 is now integrated with SynBioHub, a non-React based web application, illustrates this principle.

The back-end was built primarily using Node.js,[15] an open-source, cross-platform JavaScript run-time environment. Like the original VisBOL, VisBOL2 uses sboljs[16] to parse SBOL files and extract relevant data, such as components and interactions in the design.

Discussion

The strict separation of concerns in VisBOL2 enables new features to be implemented relatively quickly. The tool is under active development: there are improvements planned for the future to improve the longevity and usability of the tool.

Like the original VisBOL, VisBOL2 must manually update the glyphs it uses for depiction as SBOLv changes. This forces web applications that utilize VisBOL2 to ensure that VisBOL2 is updated to its latest version if consistency with the SBOL Visual standard is desired. To overcome this issue, future plans for VisBOL2 include subscribing to a standardized library of parametric SVG representations of SBOLv glyphs. This will allow updates to

the SBOL Visual standard, such as the addition of new glyphs, to be implemented by the VisBOL front-end without having to update the tool to a newer version.

Another planned feature is the ability to save layout information in a standard format. This will allow VisBOL2 to exchange relevant layout information with other tools. Adding this feature will enable synthetic biologists to have greater control over how their SBOL designs are depicted. Currently, users of SBOLCanvas are able to export the designs they create to SynBioHub, which will represent them using a SBOLv diagram, but this diagram typically has a different layout than the initial diagram created in SBOLCanvas. By supporting a standard layout format, VisBOL2 would be able to depict designs exported to SynBioHub from SBOLCanvas exactly as they were constructed. This standard layout format could also ensure depiction consistency for designs that SBOLCanvas imports from SynBioHub. Thus, SynBioHub users could import their SBOL designs in SBOLCanvas, edit the designs' layouts to their liking, and then upload them back to SynBioHub in order to ensure their SBOL designs are depicted on the website exactly as intended.

In rare cases, interaction arrows in VisBOL2's rendering may overlap as a result of complex functional interactions and overlapping components in genetic designs. To address this, two changes will be made to VisBOL2. First, (as discussed previously) VisBOL2 will support the saving and loading of layout information in a standard format. This will allow users to manually adjust positions of glyphs and interaction arrows, thereby enabling users to reduce overlap in their depictions. Second, VisBOL2 will reduce overlapping by using a more advanced graph layout algorithm and relaxing the constraint that glyphs be positioned in a grid.

Data Availability Statement

VisBOL2 is available under the BSD open-source license. Its code and issue tracker can be found on GitHub at

https://github.com/VisBOL/VisBOL2.

Documentation for VisBOL2 can be found at

https://github.com/VisBOL/visbol2/wiki.

To render SBOL files using VisBOL2, visit the VisBOL website at

https://visbol.org/.

The SeqViz source code and issue tracker can be found on Github at

https://github.com/SynBioHub/Plugin-Visual-Seqviz.

Acknowledgement

The authors of this work are supported by DARPA FA8750-17-C-0229, National Science Foundation Grant No. 1939892, and a Dean's Graduate Fellowship at the University of Colorado Boulder. The SeqViz plugin is hosted on an Azure server provided by Microsoft Research. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Author Contributions

Linhao Meng and Jeanet Mante developed the sequence visualization plugin on SynBioHub and facilitated VisBOL2's integration with the plugin. James A. McLaughlin developed the original VisBOL and provided guidance on the development of VisBOL2. James Scott-Brown developed react-parametric-svg and provided guidance on VisBOL2's usage of parametric SVG. Chris Myers managed the VisBOL2 project. Benjamin Hatch is the primary developer of VisBOL2 and the primary author of this paper. All authors contributed to the writing of this manuscript.

Conflicts of Interest

The authors declare no conflicts of interest.

References

(1) Beal, J.; Nguyen, T.; Gorochowski, T. E.; Goñi-Moreno, A.; Scott-Brown, J.; McLaughlin, J. A.; Madsen, C.; Aleritsch, B.; Bartley, B.; Bhakta, S. et al. Communicating Structure and Function in Synthetic Biology Diagrams. *ACS Synth. Biol.* **2019**, *8*, 1818–1825, PMID: 31348656.

(2) Quinn, J. Y.; Iii, R. S. C.; Adler, A.; Beal, J.; Bhatia, S.; Cai, Y.; Chen, J.; Clancy, K.; Galdzicki, M.; Hillson, N. J. et al. SBOL Visual: A Graphical Language for Genetic Designs. *PLoS Biol.* **2015**, *13*, e1002310.

(3) Roehner, N.; Beal, J.; Clancy, K.; Bartley, B.; Misirli, G.; Grünberg, R.; Oberortner, E.; Pocock, M.; Bissell, M.; Madsen, C. et al. Sharing Structure and Function in Biological Design with SBOL 2.0. *ACS Synth. Biol.* **2016**, *5*, 498–506, PMID: 27111421.

(4) Der, B. S.; Glassey, E.; Bartley, B. A.; Enghuus, C.; Goodman, D. B.; Gordon, D. B.; Voigt, C. A.; Gorochowski, T. E. DNAplotlib: Programmable Visualization of Genetic Designs and Associated Data. *ACS Synth. Biol.* **2017**, *6*, 1115–1119.

(5) Zhang, M.; McLaughlin, J. A.; Wipat, A.; Myers, C. J. SBOLDesigner 2: an intuitive tool for structural genetic design. *ACS synthetic biology* **2017**, *6*, 1150–1160.

(6) Czar, M. J.; Cai, Y.; Peccoud, J. Writing DNA with genoCAD™. *Nucleic acids research* **2009**, *37*, W40–W47.

(7) Terry, L.; Earl, J.; Thayer, S.; Bridge, S.; Myers, C. J. SBOLCanvas: A Visual Editor for Genetic Designs. forthcoming paper.

(8) McLaughlin, J. A.; Pocock, M.; Mısırlı, G.; Madsen, C.; Wipat, A. VisBOL: Web-Based Tools for Synthetic Biology Design Visualization. *ACS Synth. Biol.* **2016**, *5*, 874–876, PMID: 26808703.

(9) McLaughlin, J. A.; Myers, C. J.; Zundel, Z.; Mısırlı, G.; Zhang, M.; Ofiteru, I. D.; Goñi-Moreno, A.; Wipat, A. SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology. *ACS Synth. Biol.* **2018**, *7*, 682–688, PMID: 29316788.

(10) Sidney Cox, R.; Madsen, C.; McLaughlin, J.; Nguyen, T.; Roehner, N.; Bartley, B.; Bhatia, S.; Bissell, M.; Clancy, K.; Gorochowski, T. et al. Synthetic Biology Open Language Visual (SBOL Visual) Version 2.0. *Journal of Integrative Bioinformatics* **2018**, *15*.

(11) parametric.svg – SVG on rocket fuel. `https://parametric-svg.js.org/`.

(12) Mante, J.; Zundel, Z.; Myers, C. Extending SynBioHub's Functionality with Plugins. *ACS Synth. Biol.* **2020**, *9*, 1216–1220.

(13) React – A JavaScript library for building user interfaces. `https://reactjs.org/`.

(14) react-parametric-svg. `https://www.npmjs.com/package/react-parametric-svg`.

(15) About Node.js. `https://nodejs.org/en/about/`.

(16) McLaughlin, J.; Myers, C.; Zundel, Z.; Wilkinson, N.; Atallah, C.; Wipat, A. sboljs: Bringing the Synthetic Biology Open Language to the Web Browser. *ACS Synth. Biol.* **2019**, *8*, 191–193.

Graphical TOC Entry