Promotion of Data Reuse in Synthetic Biology

by

J. V. Mante

B.A., University of Cambridge, 2018

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Biomedical Engineering

2022

Mante, J. V. (Ph.D., Biomedical Engineering)

Promotion of Data Reuse in Synthetic Biology

Thesis directed by Prof. Chris J. Myers

Synthetic biology is a movement to standardize genetic engineering and make it more reproducible and accessible by using functional descriptions of desired circuits. Such descriptions can then be converted to genetic designs via genetic design automation tools. Subsequently, the genetic designs can be used to generate models for *in silico* experimentation using automatic model generators. Both of these technologies rely on access to libraries of genetic part information encoded in standard, machine-readable ways. The Synthetic Biology Open Language (SBOL) can be used together with SynBioHub (a genetic part repository) to encode and store the information. However, the use of SynBioHub for the storage and reuse of parts is still very limited. This is due to insufficient metadata (making it difficult to find parts or judge their usefulness) and the effort required to submit parts to the repository. This dissertation aims to decrease the barriers to part reuse and thus enable a more automated synthetic biology workflow. Hence, an integrated curation workflow is proposed based on the contributions of the dissertation. The contributions are: a proposed SBOL Data Content Standard, tools for working with genetic parts in spreadsheets, a framework to modularly extend the SynBioHub part repository, and the lessons learned from the analysis and curation of data from existing genetic data repositories.

Dedication

For my brothers Hein and Eltjo, may it inspire you

Acknowledgements

This dissertation would not have been possible without the support of many people and lots of cups of peppermint tea. First, I would like to thank my advisor Chris Myers for giving me the opportunity to conduct the research presented in this dissertation. I learnt a great deal from him, both academic (particularly developing my computer science knowledge) and non-academic. Chris gave me the freedom to explore, supported conference attendance, pushed me to write publications, and gave me the chance to teach and mentor my own students. Furthermore, he was very supportive during times of crisis including, but not limited to, the difficulties of the coronavirus pandemic.

In addition to Chris Myers, I want to thank my supervisory committee: Mirela Alistar, Brian DeDecker, Duygu Dikicioglu, and Ryan Layer. Their feedback helped steer and improve work.

As a member of the Genetic Logic Lab, I have worked alongside many talented students: Leandro Watanabe, Tramy Nguyen, Michael Zhang, Zach Zundel, Pedro Fontanarrosa, Lukas Buecherl, Ben Hatch, Eric Yu, Julian Abam, Sai Samineni, Payton Thomas, Thomas, Stoughton, James Scholz, Logan Terry, Sam Bridge, Oliver Flatt, and Meher Samineni. I would like to thank them for collaborations, discussion of ideas, and the “lab social activities”. They provided a wonderful atmosphere and much laughter. I would also particularly like to thank Zach, Lukas and Pedro for helping me through deaths in the family, the NCAR fire, and bouts of imposter syndrome.

Additionally, I would like to thank the Google Summer of Code students who I had the opportunity to work with: Isabel Pötzsch and Linhao Meng.

I would also like to thank those who are involved in the development of the software tools that are used in this research: SynBioHub, SYNBICT, and the community representing SBOL. In particular, Jake Beal, Bryan Bartley, Nic Roehner, and Tom Mitchell. The work presented in this research could not have been possible without these developers.

Furthermore, I would like to thank my family and friends for their encouragement and good-natured teasing. The support they provided both intra and inter time zones was much appreciated. The Martis and Raos helped me find my feet in Salt Lake City and the Bishop Cohort gave me a home during the pandemic. My flatmates provided a lot of late night support including overlooking dirty dishes and dishing dirt. The ONLs kept me grounded with the slightly sporadic weekly calls. My parents, brothers, and grandparents helped me get this far and continued to believe in me even in moments when I didn't. I wouldn't have been able to do it without you.

I would also like to thank a few influential teachers who helped me develop and follow my love of learning: Mrs. Herringer-Brock, Mrs. Baker, Dr. Welford, Mrs. Betts, Valerie Chock, Jim Haseloff, and Tim Weil.

This research is based upon work supported by the National Science Foundation under Grants CF-1748200, and 1939892. It was also supported by a Dean's Graduate Assistantship at the University of Colorado Boulder.

Contents

1 Introduction **1**

1.1 Synthetic Biology Workflow 2

1.2 Contributions 4

1.3 Dissertation Outline 6

2 Background **8**

2.1 Genetic Components 8

2.2 Standards 14

2.3 Repositories 16

2.4 Metadata in Databases 17

2.4.1 Library Science Principles 18

2.4.2 Minimum Information Standards 18

2.5 Automated Curation 19

2.6 Search 20

2.7 Terminology Equivalence 22

3 The Extension of SynBioHub via a Plugin Interface **23**

3.1 Plugin Specification 24

3.1.1 Submit 27

3.1.2 Visualization 32

3.1.3 Download 35

3.2 Plugin Templates 37

3.2.1 Templates and Docker 37

3.2.2 Python Templates 38

3.2.3 JavaScript Templates 41

3.3 New Plugin Specifications 41

3.3.1 Curation 43

3.3.2 Search 44

3.3.3 Index 50

3.3.4 Link 53

3.4 Example Plugins 54

3.5 Further Work 54

4 Excel-SBOL Converter **56**

4.1 Excel-to-SBOL Evolution 58

4.1.1 Iteration 1: Fixed Column Templates 58

4.1.2 Iteration 2: Flexible Column Templates 63

4.1.3 Iteration 3: Multi-Library Templates 67

4.2 Excel-to-SBOL Case Study 72

4.3 SBOL-to-Excel 74

4.4 SBOL-to-Excel Case Study 74

4.5 Submit and Download Plugins 75

4.6 Conclusions 76

5 Post Hoc Curation **79**

5.1 Post-Hoc Curation Pipeline 80

5.2 Example Applications of the Post-hoc Curation Pipeline 82

5.2.1 iGEM 82

5.2.2 ACS Dataset 90

5.2.3 Addgene 96

5.3 Conclusions 100

6 A Structure for Integrated Curation **104**

6.1 The SBOL Data Content Standard 106

6.2 Research Workflow with Integrated Curation 109

6.3 Realization of the Integrated Curation Workflow 115

6.3.1 Plugins 115

6.3.2 Excel-SBOL Converter 119

6.4 Challenges for the Integrated Curation Workflow 119

6.4.1 Gaps Between Workflow Stages 119

6.4.2 Resistance to Uptake 121

7 Conclusions **122**

7.1 Summary 122

7.2 Future Work 124

7.2.1 Plugins 125

7.2.2 Search 125

7.2.3 User Interface Development 125

7.2.4 Excel Templates 126

7.2.5 Further Curation Libraries 126

7.2.6 Further Curation of iGEM, ACS, and Addgene Libraries 126

7.2.7 SBOL Data Content Standard Extension 126

7.2.8 Framework to Assess Data Reuse 127

7.2.9 Community Uptake 127

Bibliography **128**

A Supplemental iGEM Figures **149**

B GitHub Repositories **153**

B.1 Plugins 153

B.2 Excel-SBOL 154

B.3 Post-hoc Curation 155

List of Tables

3.1 Currently Available Plugins . . . 55

5.1 iGEM SYNBICT: Frequency of Annotations per Sequence . . . 85

5.2 iGEM SYNBICT: Number and Percentage Cover of Specific Annotations . . . 86

5.3 iGEM: Real versus Spurious Descriptive Field Lengths . . . 88

5.4 iGEM: Basic Unique Descriptive Field Lengths . . . 89

5.5 ACS: Top 10 Most Common Supplemental File Types . . . 91

5.6 ACS: Types of Named Entity Recognised . . . 94

5.7 ACS: Sequences Annotation by File Type. . . . 94

5.8 ACS: Top Annotations of Supplemental Sequences . . . 95

5.9 ACS: Top Ten Terms Extracted by NER Type . . . 95

5.10 Addgene: Frequencies of Annotation Properties . . . 99

5.11 Addgene: Species Annotations Frequency of Occurrence . . . 101

5.12 Addgene: Top 50 Sequence Annotations . . . 102

6.1 SBOL Data Content Standard . . . 108

6.2 SBOL Data Content Standard None and Other Terms . . . 110

Algorithms

3.1 Submit Plugin API 31

3.2 Visualization Plugin API 34

3.3 Download Plugin API 36

3.4 Curation Plugin API: STATUS 43

3.5 Curation Plugin API: EVALUATE 45

3.6 Curation Plugin API: RUN 46

3.7 Curation Plugin API: SAVE 47

3.8 SEARCH Plugin API: STATUS 48

3.9 SEARCH Plugin API: PARAMETERS 49

3.10 Search Plugin API: RUN 51

4.1 Excel-to-SBOL Iteration 1: Part Library Processing 60

4.2 Excel-to-SBOL Iteration 1: Composite Library Processing 62

4.3 Excel-to-SBOL Iteration 2: Flexible Column Templates 64

4.4 Excel-to-SBOL Iteration 3: Multi-Library Templates 69

4.5 SBOL-to-Excel Iteration 2: RDF Conversion 75

List of Figures

1.1 Design-Build-Test-Learn Cycle and Its Implementation 2

2.1 Central Dogma of Biology: DNA to RNA to Protein 9

2.2 DNA Binding Proteins OR Gate 10

2.3 Epigenetic NOT Gate 10

2.4 Recombinase AND Gate 11

2.5 CRISPRi NOR Gate and CRISPRa OR Gate 12

2.6 RNA IN/OUT NOT Gate 13

2.7 Comparison of Different Genetic Part Standards: FASTA, GenBank, and SBOL . . . 15

3.1 Plugin Administrative Interface 25

3.2 SynBioHub-Plugin API Diagram 26

3.3 SynBioHub Submit Interface with Plugin Dropdown 28

3.4 SynBioHub View Endpoint with Visualisation Plugins 29

3.5 SynBioHub endpoint with Download Plugins 30

3.6 Visual Template 40

3.7 Download Template 42

3.8 Search Plugin Facet Example 52

4.1 Excel-SBOL Converter Overview 57

4.2 Evolution of the Excel-to-SBOL pipeline over time 59

4.3 SD2 Library Template 60

4.4 SD2 Composite Template . . . 61

4.5 Column Definitions Sheet for Flexible Column Templates . . . 66

4.6 Initialization Sheet for Multi-Sheet Templates . . . 68

4.7 Column Definitions Sheet for Multi-Sheet Templates . . . 70

4.8 Excel-SBOL Example: Flapjack Spreadsheet . . . 73

4.9 Spreadsheet Output by the SBOL-to-Excel library . . . 76

4.10 Integration of the Converter Plugins with SynBioHub . . . 77

5.1 Post-hoc Curation Pipeline . . . 80

5.2 iGEM: Word Cloud of Species Frequency . . . 89

5.3 Example of an Addgene Plasmid Page . . . 97

6.1 Elements of an Interactive Paper . . . 105

6.2 Research Workflow with Integrated Automation . . . 111

6.3 Knowledge Enabled Search . . . 113

6.4 Sequence Curation Interface . . . 116

6.5 Publication Curation Interface . . . 117

6.6 Plugins in the Research Workflow with Integrated Automation . . . 118

6.7 Excel-SBOL Converter in the Research Workflow with Integrated Automation . . . 120

A.1 Variation of iGEM description by Year . . . 150

A.2 Variation of iGEM Description by Month . . . 151

A.3 Variation of iGEM description over time by Group . . . 152

Chapter 1

Introduction

Synthetic biology is a movement to standardize genetic engineering and make it more reproducible. Founded on the idea of composable DNA parts, synthetic biology seeks to enable construction of complex genetic devices with predictable functions [149]. This has largely proved possible [108, 157, 42], underpinned by the Design-Build-Test-Learn (DBTL) cycle. The DBTL cycle aims to decouple specialist biology knowledge from the design of a circuit: by separating the specification of function from the choosing of the parts to implement the function. However, the choice of parts remains a difficult problem as there are few effective means to communicate part information amongst designers. Early in the development of synthetic biology, parts databases were recognized as key for information exchange. Yet, many of these have been ineffective in communicating data and metadata like function, intended host, and assembly method. Thus, databases are rarely used in genetic design, especially for organisms that are not *Escherichia coli*. This leaves the field in a state where relevant part performance data is distributed among results and methods sections, paper supplemental files, and tables of sequences — shifting the work of a genetic designer from design to searching through disparate sources for part information. Many designers simply use a custom set of parts curated from past experience or screen new parts rather than use a database or mine the literature. This work aims to reduce the effort required to find biological parts, and thus increase their reuse.

1.1 Synthetic Biology Workflow

The Synthetic biology workflow is based on the DBTL cycle (Figure 1.1). It is iterative, and the goal is to be increasingly focused on the functional description of circuits rather than the Build and Test phases. For this to be possible, part selection, modeling, assembly, and screening should all require little user input and be increasingly automated. To enable this, tools are required to go from functional descriptions to genetic design descriptions [155, 154, 158, 42], and from design descriptions to models [68]. In order to design software applications that can automate the process of designing and modelling a genetic circuit, biological part information must be encoded using data standards and stored in accessible repositories.

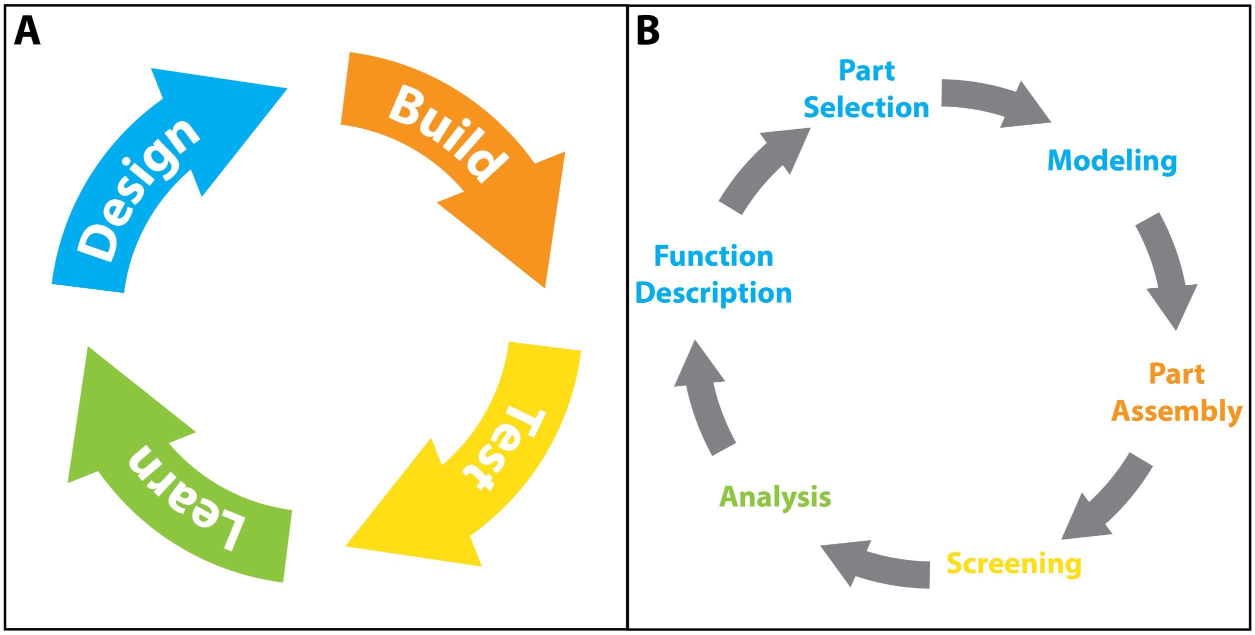


Figure 1.1: The Synthetic Biology Workflow. **A**: The Design, Build, Test, Learn workflow. **B**: How the Design, Build, Test, Learn workflow relates to implementation. The Design stage requires the description of circuit function, the selection of parts to implement the function, and the modelling of the chosen design. If the user is happy with the model predictions, parts can then be assembled *in vitro* or *in vivo* and the resulting circuits can be screened for functionality. The results of screening are analyzed and inform the next design.

There are several data standards used to store genetic information (see Chapter 2), however the one developed specifically for synthetic Biology is the Synthetic Biology Open Language

(SBOL) [9]. It is a Resource Description Framework (RDF) standard that stores all information as triples: subject, predicate, object. For example, Gene A (subject) up-regulates (predicate) Gene B (object). SBOL was developed as a standard to support the specification and exchange of biological design information. Complex relationships can be described using the SBOL format, including the description of functional genetic sequences (e.g. a promoter), their conglomeration into a composite part (e.g. a promoter, ribosome binding site, coding sequence, and terminator), and the interaction of composite parts to create a design (e.g. composite one produces protein one, which prevents composite two producing protein two). Furthermore, the standard has also expanded to allow the description of experimental data. The standard specifically aims to be machine readable. This is achieved through many explicit property (predicate) fields and the use of hierarchical controlled vocabularies (ontologies) [56, 95, 179, 33, 121, 62, 50].

Whilst data standards are required for the automation of genetic design, they are not sufficient. The design information must not only be described in comparable, machine readable, and standardized formats, but must also be accessible. Designs can be made accessible via the use of open, centralized, internet repositories. There are several different systems for storing and sharing data about engineered biological designs, most notable are JBEI-ICE [82] and the iGEM Registry of Standard Biological Parts (http://parts.igem.org) (for more information see Chapter 2). However, SynBioHub is the only repository designed to leverage the benefits of SBOL for storing information about genetic designs [138]. SynBioHub was initially developed as an aggregation facility to unite information from many different repositories and store clear provenance information together with genetic design information. However, it has fallen short of this goal, due to complexity of getting information into the SBOL data format, and the lack of part reuse making depositing parts not worth the effort [217].

Unfortunately, part reuse and the use of genetic design repositories are bidirectionally linked. Parts would be easier to reuse if they were stored in a repository, and more parts would be deposited if they were reused more. However, there are also other barriers to part reuse. The main barriers to data reuse are low efficacy and low efficiency, i.e. how well data can be used to answer the question posed (usability) and how easy it is to find the data (discoverability). When data is easier to find and has more complete metadata on which its suitability for answering a research question can be judged, then data reuse is more common [54, 166, 248, 49, 30, 71]. Thus, increasing metadata, reducing the effort required to submit parts to repositories, and increasing the ability to search for parts increases part submission and reuse. In turn, this decreases the time and effort required to create new genetic designs.

1.2 Contributions

The contributions of this dissertation are all steps towards reducing the effort required to find and use genetic parts. Some are more direct steps, others less so.

The **framework to modularly extend the SynBioHub part repository** enables the customization and extension of SynBioHub functionality. The plugin framework allows users to develop simple servers, in a programming language of their choice, to provide additional functionality on SynBioHub. Several templates are provided to make the threshold to development as low as possible. The additional functionality users might add includes: visualizations comparing genetic part use, sequence views, downloading genetic part information in the form they are most comfortable with, or uploading information from whatever format they generally use. The framework makes the part repository better able to fit into the disparate existing workflows. Additionally, the extended plugin framework was specifically designed to support enhanced curation of parts and make finding curated parts easier.

The **tools to work with genetic parts in spreadsheets**, the Excel-SBOL Converter, enables researchers to submit and view their part and experimental information in a form that they are comfortable with. It consists of two Python libraries, one to convert spreadsheets to the SBOL data standard and one to convert SBOL data objects to spreadsheets. These libraries are designed so that users unfamiliar or uncomfortable with ontologies and RDF can still submit their data to SBOL repositories. Thus, the users can work in spreadsheets, which are often already part of experimental workflows, but still benefit from the curation and search functionality offered by a language such as SBOL.

We performed analysis and curation of data from existing genetic data repositories. This led to the creation of a **list of issues with the genetic data sets and curated genetic data sets**. The data sets are often difficult to search, present information in non-standard ways, or lack the metadata required to judge the usefulness of the data. The analysis consisted of pulling data from three existing repositories iGEM (a repository of genetic parts mostly submitted by high school students and undergraduates), ACS Synthetic Biology (a journal), and Addgene (a non-profit plasmid repository with an online database of the stored plasmids). The data was then processed to pull sequences out and annotate them with common sub-sequences. Additionally, the records were analysed to retrieve and standardize the sequence metadata. The contribution is three-fold: the semi-curated data sets, information about the data sets, and the elucidation of challenges to curation. The data sets are each curated to different levels due to the varying levels of manual curation required. None of the data sets are fully curated as in every case there are terms which are ambiguous to the point of requiring input from the original author, and there are gaps in the provenance information. The information about the data sets includes the kinds of genetic parts often seen, the types of metadata recorded, and some of the most common metadata values (e.g. most common species). The challenges preventing curation are shown in the difficulty of generalising a workflow that would enable disparate data sources to be combined into a single well-curated searchable database. In particular, places where manual curation is required are major hurdles to the implementation of curation.

A **proposed SBOL Data Content Standard** was created. This is a list of metadata properties: their name, description, and any ontologies they should use. Such a standard enables the conversion of data from other formats and repositories to SBOL in a more standardized manner. Additionally, it can inform experimental design to ensure any new part libraries are usable in synthetic biology workflows. The standard can form the basis of new search algorithms, can be used for *in silico* modeling, and ensure part data can be evaluated and utilized by others. The standard was developed based on the analysis mentioned in the previous contribution.

An **integrated curation workflow** is a workflow that utilises curation to enhance genetic design metadata and reduce the effort required to find, create, and share genetic designs. The workflow presented illustrates how the plugin framework, spreadsheet tools, and SBOL Data Content Standard can be brought together to create a more automated and efficient synthetic biology workflow.

1.3 Dissertation Outline

This dissertation is composed of seven themed chapters.

Chapter 2 provides the background information on data reuse methodologies, and the technologies used to support tool development. This includes: genetic parts and circuits, standards (FASTA, Genbank, SBOL), repositories (WormBase, UniProt, GenBank, JBEI-ICE, SynBioHub), metadata in databases, automated curation, and APIs.

Chapter 3 focuses on the development of the plugin framework. It explains the plugin specification for submit, visualisation and download Plugins. Then, it explains the development of plugin templates to make the development and testing of plugins easier. Next, a further set of plugins is proposed to extend the plugin framework based on user feedback and tooling wishes. Subsequently, the specific implementations of plugins that currently exist are described. Finally, a further work section indicates the next steps in plugin framework development. The framework discussed here is also used in Chapter 6 for an integrated curation structure.

Chapter 4 describes the principles behind the Excel-SBOL Converter. It considers the generalization of spreadsheets by going through the different iterations of Excel-to-SBOL and indicating the challenges that prompted further development. This is supported by a case study using Excel-to-SBOL together with experimental data. The development of SBOL-to-Excel is then discussed together with a case study that was used to develop the converter. Next, the integration of the converter into the plugin framework (discussed in Chapter 3) is shown. Finally further work is discussed. The converter is used in the post-hoc curation discussed in Chapter 5, and the proposed integrated curation framework in Chapter 6.

Chapter 5 shows a proposed post-hoc curation pipeline and its application to three genetic part data sets: iGEM, ACS Synthetic Biology, and Addgene. After the initial description of the post-hoc curation workflow, the three example applications are discussed. Each application explains how the sequences were extracted, had sub-components annotated, and how metadata about the sequences was extracted. The results of the sequence and metadata extraction are then discussed along with the challenges of applying the pipeline to the particular case study. A summary of the lessons learned is given and this provides the motivation for the work discussed in Chapter 6.

Chapter 6 proposes an integrated curation workflow and its applications to advancing the way in which data and information is exchanged in synthetic biology. The SBOL Data Content Standard is proposed to support the realization of the integrated curation workflow. The pieces required for the proposed workflow are then discussed, including how the plugin framework (Chapter 3) and the Excel-SBOL Converter (Chapter 4) can be used. Additionally, the chapter highlights the challenges that must be overcome in order to implement the proposed framework.

Finally, Chapter 7 summarizes the accomplishments that were presented and discusses future directions for this work including: expanding search capabilities to take advantage of improved metadata, creating a framework to assess the impact of better metadata on part reuse, making the development of user interfaces for the tools simpler, implementing the extended plugin framework in SynBioHub, further expanding the SBOL Data Content Standard, providing more Excel-to-SBOL templates, creating further libraries for sequence annotation, and further curating the genetic part information gathered in the post-hoc curation pipeline.

Chapter 2

Background

This chapter provides some background to the material presented in this dissertation. Section 2.1 describes genetic components. Section 2.2 discusses different standards for storing genetic part information. Section 2.3 provides background on different genetic data storage repositories. Section 2.4 explains the role of metadata in repositories. Section 2.5 gives some background on automated curation. Section 2.6 discusses search principles. Finally, Section 2.7 provides a section on terminology equivalence.

2.1 Genetic Components

Synthetic biology aims to implement engineering principles of standardization, abstraction, and modularity to genetic engineering. To do so, the idea of **genetic components** was developed. Components may be used by themselves or combined to create a more complex component with sub-components. Common components are based on the central dogma of biology "DNA to RNA to Protein", and the main sequence features required for transcription of DNA to RNA, and translation of RNA to protein. An example of these features is shown in Figure 2.1. Key sequence features are: **operators**, **promoters**, **ribosome binding sites**, **coding sequences** and **terminators**. An operator is the element of an operon to which activators or repressors bind thereby effecting translation of genes in that operon. A promoter is the region of DNA to which RNA polymerase binds, to begin transcription. A ribosome binding site is the region in the 5' un-transcribed region that pairs with the 16S ribosomal RNA during the start of translation. A coding sequence is a region which contains the information about the amino acid sequence in the final protein. A terminator is a region of sequence that causes RNA polymerase to terminate transcription.

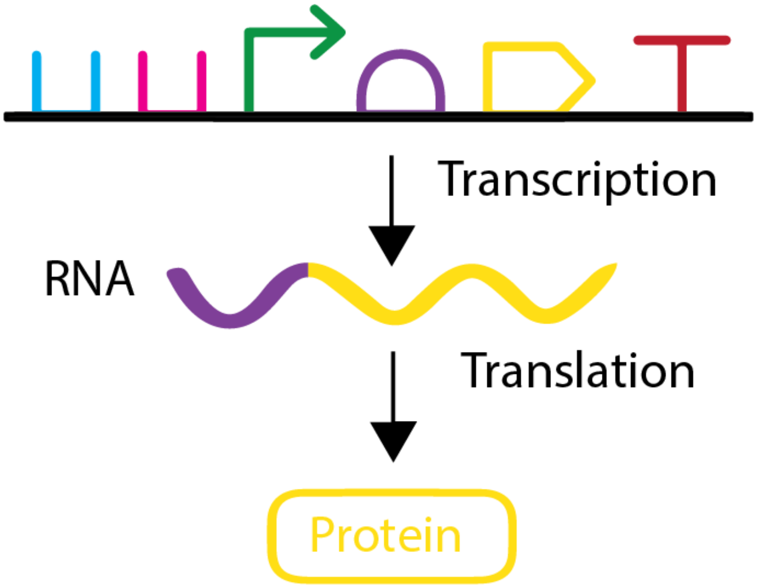


Figure 2.1: Central Dogma of Biology. DNA is transcribed to RNA and translated to protein. The figure also shows some key annotations used in synthetic biology. The blue (⊔) and pink (⊔) symbols are the SBOL Visual symbols for operators. An operator is the element of an operon to which activators or repressors bind thereby effecting translation of genes in that operon. The promoter (↱) is the region of DNA to which RNA polymerase binds, to begin transcription. The ribosome binding site (⌓) is the region in the 5' un-transcribed region that pairs with the 16S ribosomal RNA during the start of translation. The coding sequence (▷) is a region which contains the information about the amino acid sequence in the final protein. Finally, the terminator (⊤) is a region of sequence that causes RNA polymerase to terminate transcription.

Genetic components can be modeled as circuits [154, 34]. This kind of abstraction allows techniques from electrical design automation to be used in genetic circuit design. To allow this modeling, circuits are abstracted into boolean (i.e. True/False) digital logic gates (AND, OR, NOR, NAND, NOT) the building blocks of electrical circuits and computers. These circuits may be implemented in several different ways including: DNA binding proteins (Figure 2.2), epigenetic binding proteins (Figure 2.3), recombinases (Figure 2.4), CRISPR-based (Figure 2.5), and RNA-IN/OUT (Figure 2.6).

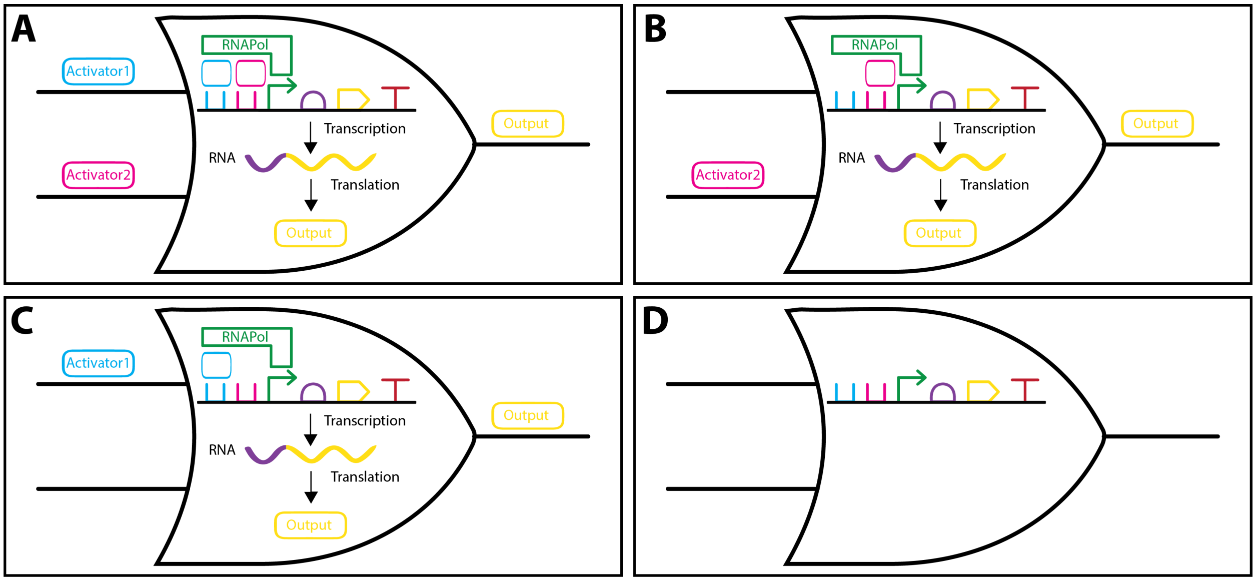


Figure 2.2: OR Gate implemented via DNA Binding Proteins. OR gates are gates that provide an output if one, or both of the inputs is present. As depecited in A-D the or gate provides the output protein only if one or both of the activators are present. This is achieved by having activator proteins bind to operator sites. If the activator is present then RNA polymerase can bind and mRNA can be transcribed. Otherwise, no transcript is made and thus no protein will be created.

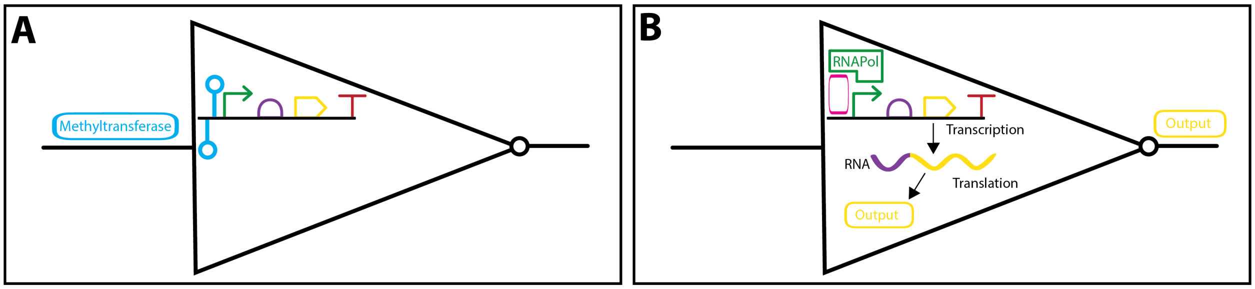


Figure 2.3: Epigenetic NOT gate. NOT gates invert the input given. Epigenetic gates work by changing the methylation state of the DNA. **A**: If methylation is present then the zinc finger protein cannot bind so RNA polymerase is not recruited. **B**: If the methylation is not present then the zinc finger protein (with RNA polymerase recruitment domain) can bind to the sequence and recruit the RNA polymerase. Then transcription and translation can occur and the output protein is produced.

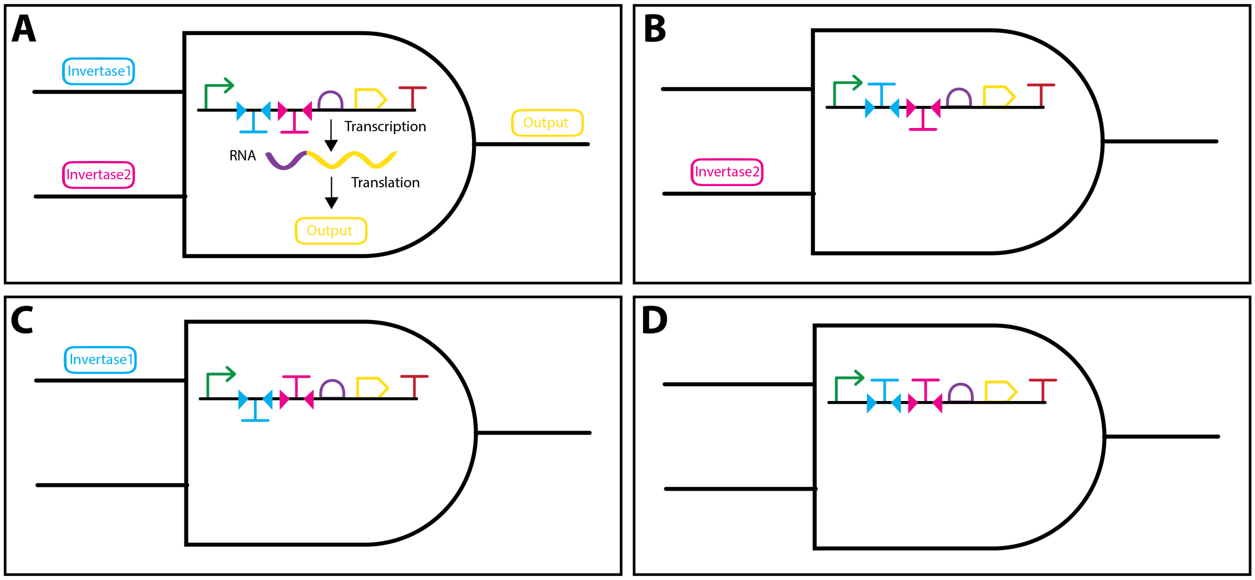


Figure 2.4: Recombinase based AND gate. Recombinase gates work by changing sequence parts from one DNA strand to the other. In this example terminators are inactivated by transfering them from one strand to the other. Thus, the output is only produced if both invertases are present.

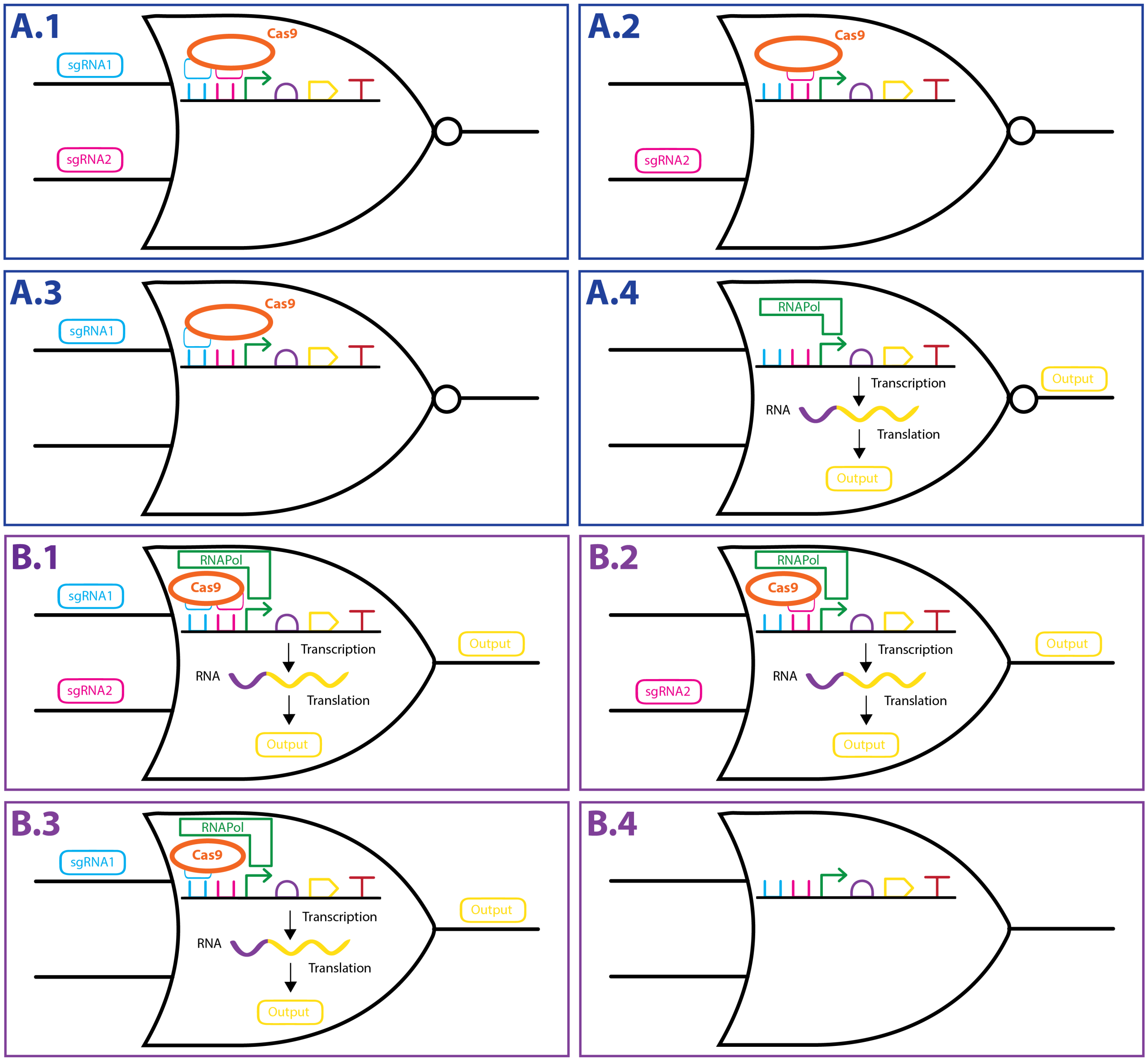


Figure 2.5: CRISPR based gates. Panel A: CRIPSRi based NOR gate. Sterhic hinderence means RNA polymerase can only bind if neither of the guide RNAs (sgRNA) are present, otherwise the dCas9 blocks the RNA polymerase from binding. Panel B: CRISPRa based OR gate. Here dCas9 has an RNA polymerase recruiting domain fused to it. Thus when it is present RNA polymerase is recruited and transcription occurs. Thus if either, or both, of the guide RNAs (sgRNA) are present the output protein will be produced.

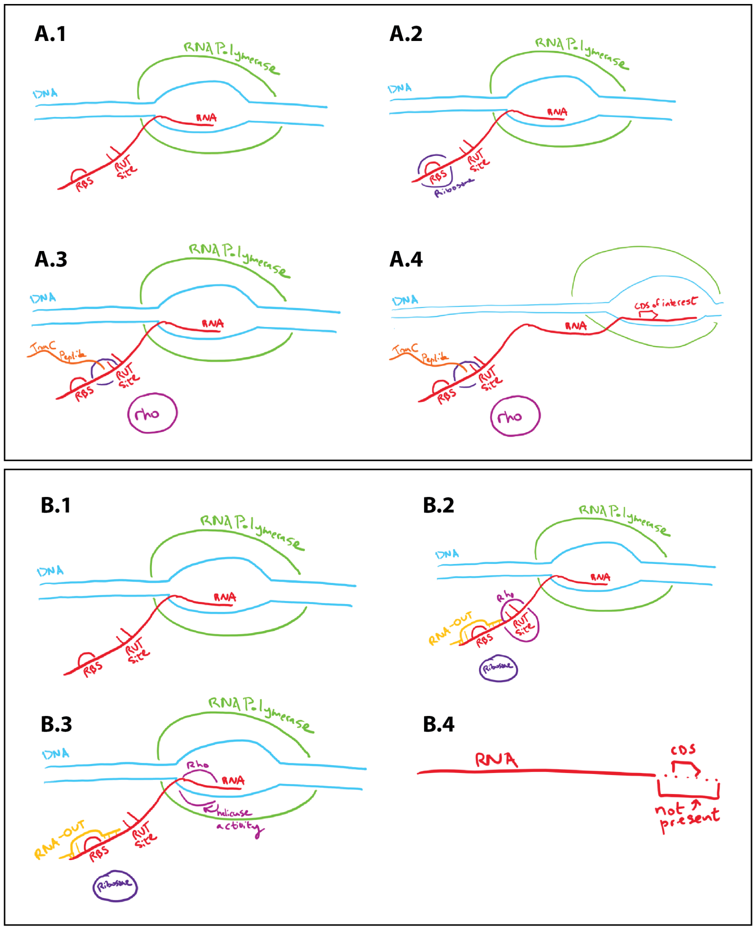


Figure 2.6: NOT Gate implemented via the RNA IN/OUT system. NOT gates take a single input and provide the opposite output. Here the input is RNA-out. If the RNA-OUT mRNA is present then the CDS is not transcribed and thus no protein is produced. In prokaryotes rho dependent termination occurs when the ribosome encounters the rho protein at the rho utilization site (RUT), if this is blocked then termination does not occur and read through to the CDS occurs. **Panel A**: RNA-OUT transcript is not present and the protein of interest is produced. **A.1** The transcript is initiated. The RNA polymerase moves along the DNA strand producing the mRNA transcript. **A.2** Whilst transcription continues a ribosome binds to the ribosome binding site of the transcript. **A.3** As transcription of the tna leader peptide (TnaC) occurs, the last 12 Tna peptide residues lead to exposure of the tryptophan site in the ribosome. If tryptophan binds to the ribosomal tryptophan site then ribosome release is inhibited and the ribosome stalls blocking the RUT site. **A.4** If the RUT site is blocked then the rho protein cannot bind, and thus cannot terminate the RNA polymerase transcription. The CDS is transcribed and translated into protein. **Panel B**: RNA-OUT mRNA is produced and so the CDS region is not transcribed and no protein is produced. **B.1** The transcript is initiated. The RNA polymerase moves along the DNA strand producing the mRNA transcript. **B.2** RNA-OUT mRNA binds to the complimentary RNA-IN site in the produced transcript. The binding of RNA-OUT blocks the ribosome binding site. Thus the ribosome cannot bind. This leaves the RUT site free, so the Rho protein can bind. **B.3** The rho protein moves along the transcript and uses its helicase activity to unwind the mRNA transcript from the DNA. This terminates transcription. **B.4** As the transcription was terminated the CDS was not transcribed and cannot be translated.

2.2 Standards

Standards are integral to the success of any engineering field as standards make data findable, accessible, interoperable, and reproducible data (FAIR data) [177, 229]. Standards were developed to enable tools and workflows to be developed around genetic part data storage and genetic circuit design. This section describes three standards for storing genetic designs and associated data: FASTA, GenBank, and SBOL (Figure 2.7).

The FASTA file format was originally an ad hoc format developed for a tool suite [167]. It has a single '>' to start the definition line which contains a unique sequence identifier/name, optional information about the source organism, and a title containing a brief description of the sequence [152]. The next line is the string of letters that comprise the sequence. There are no annotations for sub-components, and it is a plain text format.

GenBank is a format developed for use in the NCBI GenBank database [20]. However, due to the lack of documentation it is supported differently by different software making it more ad hoc. The closest to a definition of the format is the example file (`https://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html`) which shows the fields found in GenBank. It includes more annotation properties including referencing publications than FASTA. Additionally, it provides the ability to annotate sequence features. However, the terms used for annotations are free text. Thus a ribosome binding site may be called a ribosome entry site despite the function being the same.

The Synthetic Biology Open Language (SBOL) is an open community based standard for sharing genetic design information [137]. One of the major improvements that SBOL provides is the use of **ontologies**. Ontologies are controlled hierarchical vocabularies. They provide precise definitions of terms, and provide less specific parent, and more specific child terms (e.g. Enterobacteriacaea is the parent of Escherichia which is the parent of *E. coli*). Ontologies are generally machine readable and their use prevents multiple variations of the same term. SBOL uses ontologies such as the Dublin Core [227] and the sequence ontology [62]. Apart from the use of ontologies, SBOL also allows clear part sub-part relationships to be defined. Later versions,

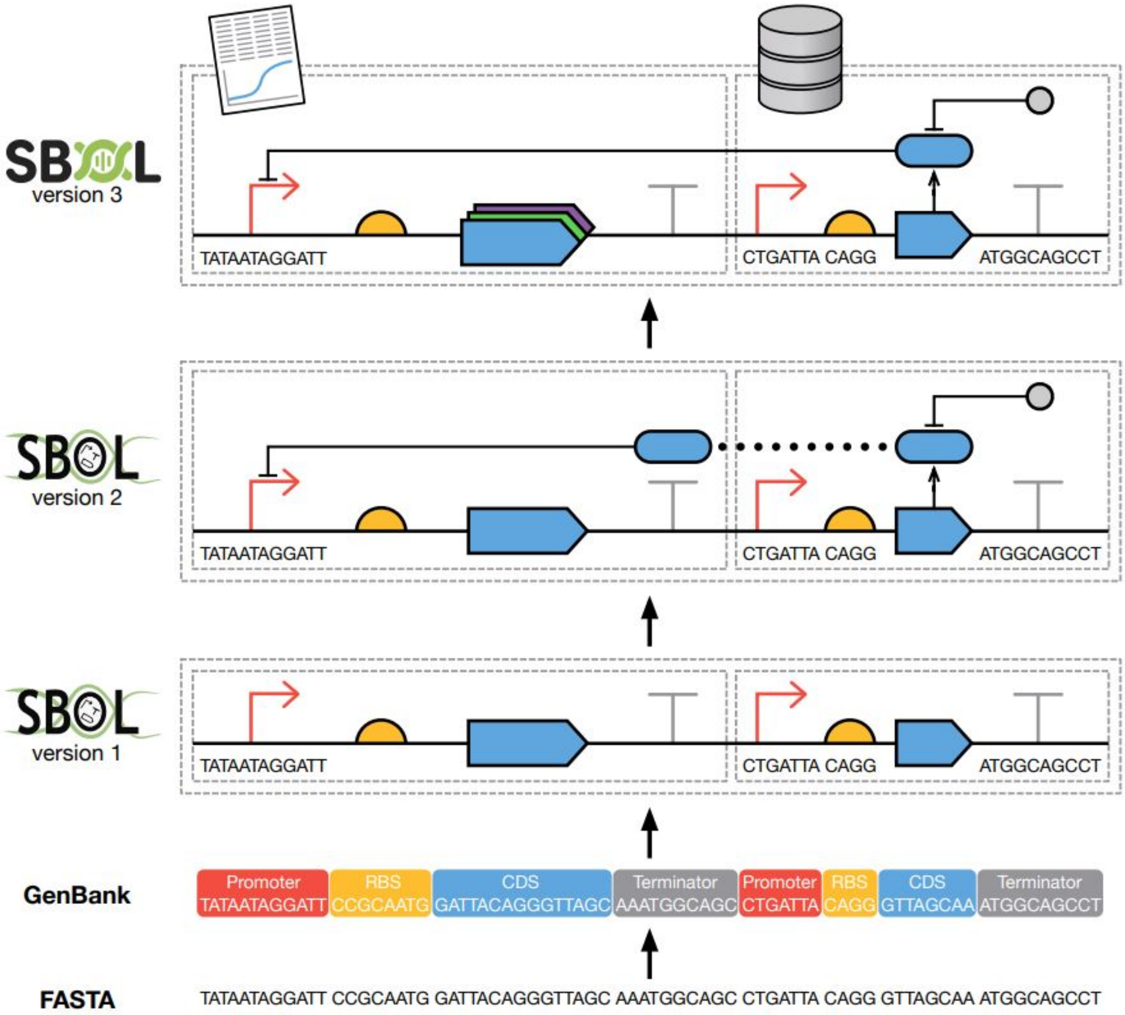


Figure 2.7: Variation in genetic part data standards. Courtesy of [137]. FASTA was an ad hoc data format that allowed no annotations to the sequence. GenBank allowed annotations but only within the file. SBOL v1 allowed annotations of parts as sub-parts of other sequences and incorporated ontologies to standardized terms. SBOL v2 also allowed interactions between parts to be described. SBOL v3 condensed and simplified SBOL v2 based on feedback from experimental and industrial users.

also include interactions between parts, and the ability to link to other kinds of data. SBOL objects work in a similar way to programming objects where a class can be derived from another class type and inherits all its properties. For example, the TopLevel class is used to derive the Sequence, ComponentDefinition, Activity, and CombinatorialDerivation classes (for more see the SBOL specification: `https://sbolstandard.org/docs/SBOL2.3.0.pdf`). This means that each of these classes requires a URI to define it, may have an optional persistentIdentity, displayId, version, wasDerivedFrom, name, description, and annotation. Additional properties can be added to specific classes. For example a Sequence object must also have elements (the nucleic acid or amino acid sequence) and an encoding property (defines if it is DNA, RNA, protein, or a small molecule). Similarly, ComponentDefinitions must have a types property to define the kind of component. It may also have roles (like promoter), link to a sequence or sub-components, and have sequence annotations and constraints.

2.3 Repositories

As part of the engineering principles that synthetic biology embraces it aims to promote the reuse of genetic components by other researchers. The researchers search for components to use in a synthetic biology database, and in turn submit their new findings or component creations into the database. Examples of databases used are: WormBase, UniProt, GenBank, JBEI-ICE, and SynBioHub.

WormBase is a data repository for Nematodes (with an inital focus on *C. elegans*) [84]. It includes genetic annotations with annotations from several ontologies: Gene Ontology [212], Anatomy Ontology [125], Human Disease Ontology [187], Life Stage Ontology, and Phenotype Ontology [185]. It is focused on the existing biology of Nematodes and not on synthetic biology.

UniProt is a protein repository [47, 48]. It is particularly good at recording provenance information and cross-linking as human and machine curated sources are added to most properties. Additionally, it cross links with the **Protein Data Bank** (PDB) [23], European Molecular Biology Laboratory Nucelotide Sequence Database [10], GenBank [20], **DNA Data Bank of Japan**

(DDBJ) [134], and **Protein Information Resource** (PIR) [236]. It also has a good mix between ontology based annotations and free text, and a search interface that makes use of faceted searching.

The BioModels Database is a repository of mathematical models of biological systems [128]. It supports the FAIR data standard and offers curation services. The website says “Before being publicly available in the BioModels curated section, a model passes through a stringent curation pipeline”. The repository uses curation, minimum information standards (MIRIAM the minimum information Required in Annotation of Models [160]), cross-linking with other databases, and ontological annotation (Gene Ontology [212], Systems Biology Ontology [99], and ChEBI [56]). It is an easy to use database of biological models.

The NCBI GenBank database stores nucleotide sequences [21]. It has strong filtering capabilities by species, source, and molecule type. It is organized based on nucleotide sequences (generally on a gene level) rather than the function of the sequence. Additionally, the experimental annotations are limited (no temperature, cloning methods etc).

The **Joint BioEnergy Institute Inventory of Composable Elements** (JBEI-ICEs) is an open source repository for biological parts [82]. It requires an account to access, so it is less easy to browse than the other databases mentioned. It does not make use of ontologies and has limited filtering (based on type and biosafety level). It does allow the storage of experimental data [218].

SynBioHub is an open source repository based on the SBOL data standard [138]. It uses ontologies like the Sequence Ontology. Filtering is possible via the advanced search interface, but not via the normal search interface. Filtering can occur based on the provenance information, the object type or the specific predicate names. Whilst it has the ability to store experimental data and information, there are currently few examples of such data being stored.

2.4 Metadata in Databases

Library Sciences is the field that develops and applies methods and tools, to the collection, organization, preservation, and dissemination of information. This can be both in the form of the traditional library or more contemporary data repositories. Some key principles of library science influence the work proposed in this dissertation.

2.4.1 Library Science Principles

In 1931 S. R. Ranganathan proposed the five laws of library science [176]:

(1) Books are for use

(2) Every reader his or her book

(3) Every book its reader

(4) Save the time of the reader

(5) A library is a growing organism

These five laws have been adapted to many different contexts including: knowledge curation [78], the internet [159], the variety of media mediums [192], and knowledge systems [188].

Similarly, these principles can be used to guide synthetic biology repositories. **Parts are for use**: Users should be able to access parts, both information about them and to use in experiments. **Every user his or her part**: A wide collection of parts should be stored with the information necessary for them to be used by different users. This means non-model organisms should be considered, and what part information experimentalists as well as modelers need. **Every part its user**: All parts deserve to be stored even if they may be very niche. **Save the time of the user**: Users should be able to quickly and easily find the parts and part information they need. **A repository is a growing organism**: A repository is a dynamic storage solution. It should not be static, and the repository should be updated to store new types of information, support new formats, and expand storage methods as time progresses.

2.4.2 Minimum Information Standards

To enact the principles of library sciences metadata is used. Metadata is “information about information”. It provides information about the information in a record, for example the author, publication date, number of pages, and language, of a book. Metadata has been shown to be important to data reuse [54, 210]. Many researchers report the inability to understand data set content and/or the quality of the data due to a lack of metadata and that this impairs their ability to judge whether it is useful for analysis [54, 210, 31]. “Good digital collections should contain metadata in the descriptions of the stored data” is the second principle provided in ‘A Framework of Guidance for Building Good Digital Collections’ as published by the National Information Standards organization [79]. Whilst there are alternative ways to find data when metadata is poor, such as by literature searches [247], metadata makes data more discoverable as specified by the FAIR principles [177, 229].

The kind of metadata provided can be standardized through the use of **minimum information standards**. Minimum information standards generally have two aspects: what metadata fields must be present, and how the metadata should be stored (the data format). Minimum information standards are used in other areas. An overview of minimum information standards was given in 2008 [207]. Some current standards are: MIABE (Minimum Information About a Bioactive Entity) [163], MIGS (Minimum information about a genome sequence) [67], MIAFGE (minimum information associated with a functional genomics experiment) [201], MIRIAM (minimum information required in the annotation of models) [160], and STRENDA (standards for reporting enzymology data) [216].

2.5 Automated Curation

Automated integrity checks help curation. The curation of data includes the use of metadata. To help increase the use of metadata checklists or templates for different experiment types have been suggested [114, 169, 113, 196]. Currently, no metadata templates cover all of the data stored in SynBioHub, the one that came closest was the CEDAR Genetic Construct metadata [146].

Comparison to templates can highlight missing information. There are many different data validation technologies (document content descriptions) that can be implemented depending on the format of the SBOL: XML Schema [32], JSON Schema [61], or ShEx/SHACL for RDF [171, 200, 214, 111]. Many of these schemas have associated schema creation and schema validation libraries in several coding languages including Python, Java, Scala, and JavaScript [45, 90, 164, 6, 197].

Curation is important. The more complete data records that are submitted, the more reusable the data and the more useful the database becomes [91]. The idea of having submitters curate their own data rather than having a special curator is becoming increasingly popular [140, 242, 89]. However, to do this, automated integrity checks, citation mechanisms, and editorial oversight is suggested [69].

2.6 Search

To be able to find and reuse genetic parts good search algorithms are required. There are different ways of improving search algorithms including: tailoring to the mode of searching, providing ranked results, providing result filtering, and providing query expansion and optimization suggestions.

There are different 'modes' of searching. Often searches are simple phrases, which could have multiple different intentions. For example, "GFP" could have several different expected results:

- "BBa_E0040" the CDS which produces GFP
- All components which include BBa_E0040
- Any literature about GFP use in different organisms
- Any reporter proteins

The nature of the results that are expected to depend on the users background (do they know the term reporter protein or only the concept of using GFP in such a way) and the users intention (e.g. is it an exploratory search or a page-find search). Previous work has indicated that users have different types of search which may vary based on device they are using to search, for example searches performed on phones are more likely to be looking for nearby amenities than desktop searches [80]. The way 'modes' of search are classified varies. They can be split into content-search vs target-search (named-page finding) [245], split along 4 axis: ambiguity, authority sensitivity, temporal sensitivity, and spatial sensitivity [153], or split by navigational search (finding a document) vs research search (finding information) [80]. The use of context to improve query results is a current practice for major search engines such as Google [202, 83] and an active area of research [3, 38, 37, 11]. It does however, raise the issues of privacy, polarization, and search bubbles [250, 175].

Faceted search and result ranking can help reduce ‘information overload’. Faceted search is the presentation of search results along with different ‘facets’ of the data to allow filtering of the results. It is widely used by companies like Amazon, eBay, Google (image search for example), and even biological databases (as shown above) [19, 226]. Information overload occurs when too many results are returned by a broad or exploratory query resulting in a lot of time and effort being required to find the desired information. The time spent can be addressed by ranking or by faceted search. Ranking is effective when assumptions used by ranking are aligned with user preferences, but often does not perform well for exploratory queries [103]. Faceted search to allow filtering of results is a form of query scoping (query rewriting) where a search like: “GFP protein” becomes GFP refined by the facet value ComponentType:Protein, such rewriting of the query can be done by leveraging SBOL elements [137, 7]. Implementation of faceted search query scoping could solve a current issue with SynBioHub search where an exact part id does not return the part in the top results (for example the search for “BBa_J70011” only returns the part as the 14th result when a regular search is performed and advanced search is required to make it the top result).

Query expansion can be used to produce more matching results. Query expansion is the practice of searching not only for ‘documents’ that contain the exact keywords provided by the search but also returning documents that contain similar keywords. For SynBioHub an example might be searching for ribosome entry site and seeing results returned for the query string ribosome binding site too. On popular search engines, query expansion often takes the form of suggestions starting with “Did you mean. . . ” or “Few results found for x try y instead”. Query expansion can be carried out using synonyms or hierarchically related terms provided by an ontology of faceted terms [3, 226, 7, 17]. Query expansion could be implemented in SynBioHub to present similar more ‘popular’ components to users.

Most search improvements (ranking, facet filtering, expansion) rely on a combination of improved search algorithms and good metadata for the records that are being searched over.

2.7 Terminology Equivalence

The nature of this work is interdisciplinary. This poses a problem when terminology is different across the disciplines. The terminology used here was chosen as most accessible across fields. However, there is an equivalence in terms that is outlined below.

The SBOL descriptions of sequences are digital twins of the biological reality. Each SBOL sequence has a unique identifier but can also store other identifiers such GenBank or Addgene. This allows mapping between the SynBioHub database and other databases which in turn allows interoperability and helps overcome the problem of data silos (data bases that are not interconnected).

The plugin framework discussed in Chapter 3 uses microservices. Each plugin is a microservice and the microservices enrich and extend the functionality of the SynBioHub platform beyond the core Graph Database.

Post-hoc curation in Chapter 5 focuses on improving data quality during the **Extract-Transform-Load** (ETL) data pipeline. It discusses extracting data and adding metadata to older formats such as PDFs which are legacy formats. This is done in order to work towards linked open data (a version of which is proposed in Chapter 6).

Chapter 3

The Extension of SynBioHub via a Plugin Interface

SynBioHub is a repository for synthetic genetic designs represented in SBOL. Despite the many advantages to machine readable SBOL, the interface for interacting with stored designs is still limited. Going to and from SBOL constructs is difficult and requires an in depth understanding of the SBOL language and XML or a programming language, such as Python or Java to use packages like pySBOL [13] and libSBOLj [244] to create the XML. Additionally, there are limited visualizations or ways to interact with the data once it is stored in SynBioHub. Whilst these problems could be individually tackled by SynBioHub developers, the many different workflows supported by SynBioHub make it difficult to keep up with the variety of user needs. As such, the plugin interface has been developed. The use of plugins allows third party developers to write simple modular extensions to be used with SynBioHub. A single plugin can be used across multiple instances of SynBioHub and be written in any language that supports API (application programming interface) creation.

APIs are a set of defined rules that explain how servers can communicate with each other. A client server sends an API call to a web server. If the request is valid, the API calls a programme or carries out an action. The response of this action is returned via the API to the client. Open APIs are APIs that are published on the internet and can be accessed and shared freely without being bound to any individual company or network. One of the most common API architectures is REST (Representational State Transfer). It is a set of principles, not a standard. The principles are: uniform interface, client-server decoupling, statelessness, cacheability, layered system architecture, and optionally: code on demand. RESTful APIs were chosen to implement the plugin framework as they can be written in many different programming languages, can create stand alone servers that can be hosted anywhere, and require few additional computer science concepts to understand and use.

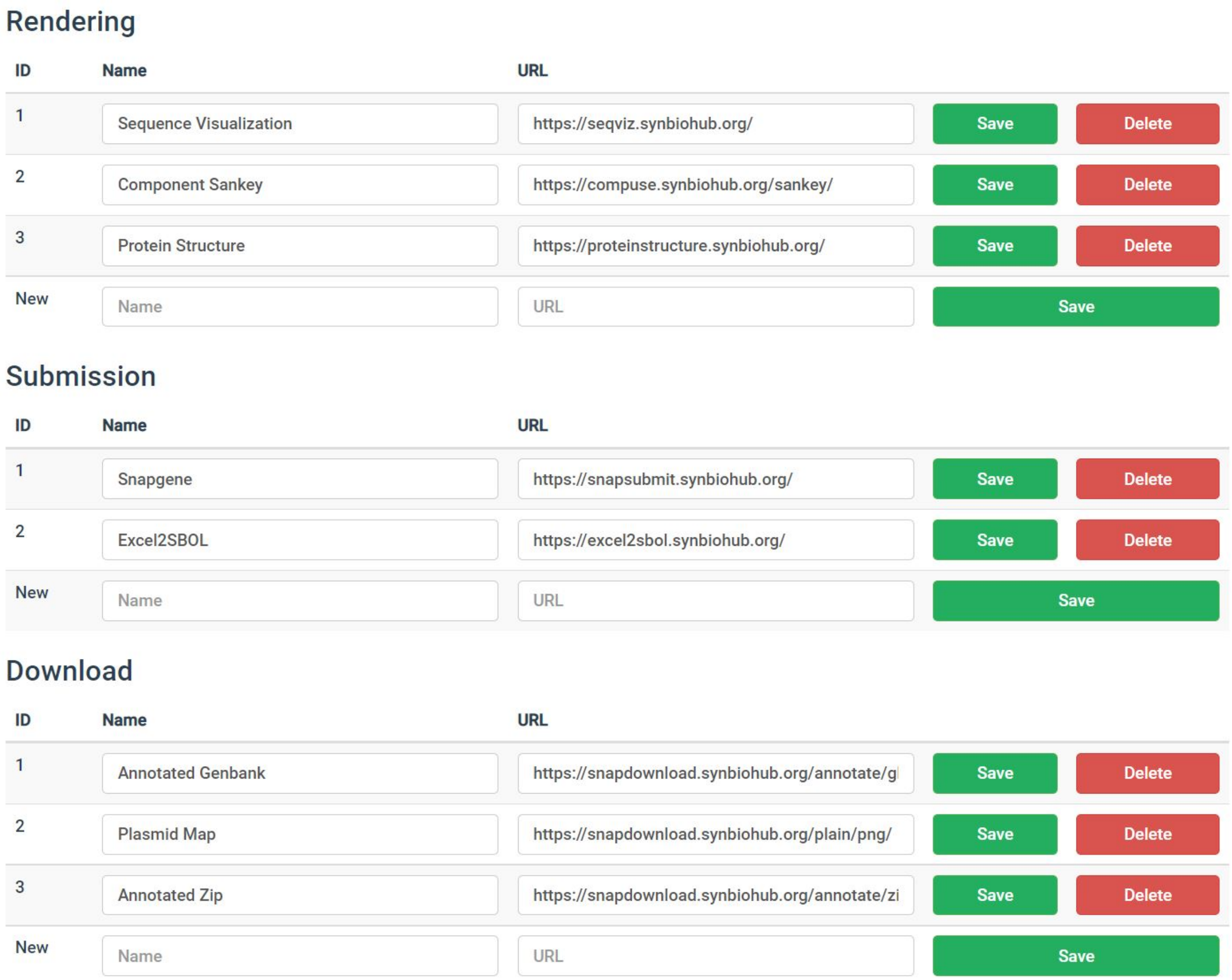
The development of the plugin framework includes: plugin API specification (Section 3.1), plugin templates (Section 3.2), further plugins (Section 3.3), example plugins (Section 3.4), and further work (Section 3.5). Note that this is not user documentation, for that please see the GitHub repositories in Appendix B.

3.1 Plugin Specification

The work presented in this section was originally published as [131]. SynBioHub was extended with a plugin engine. The plugin interface is based on a microservices design specification. Each plugin is deployed as a web service that is accessible by SynBioHub. Plugins are added to SynBioHub via the administrative interface. The plugin server can be run anywhere as long as the base URL is accessible to the SynBioHub server. To add a plugin, the URL and plugin name are put in the interface as shown in Figure 3.1. SynBioHub can then call the different plugin endpoints to interact with them.

The SynBioHub instance interacts with the plugins via a predefined API specification that is broadly the same across all kinds of plugins. Once an appropriate action (e.g. a page is rendered) is taken on SynBioHub, the plugin is engaged. The steps for this process are illustrated in Figure 3.2. First, the status of the plugin is checked to ensure it is ready to serve requests. Second, the plugin evaluates its ability to run by comparing the data type received to the data types it can handle. If the evaluation is positive, SynBioHub sends an HTTPS POST request to the plugin run endpoint. Finally, SynBioHub waits for a response and asynchronously reports it to the user.

Whilst the three endpoints of the plugin API are the same, the information sent to the different endpoints varies depending on the type of plugin. Currently, there are three types of plugins available:



Rendering

ID	Name	URL		
1	Sequence Visualization	https://seqviz.synbiohub.org/	Save	Delete
2	Component Sankey	https://compuse.synbiohub.org/sankey/	Save	Delete
3	Protein Structure	https://proteinstructure.synbiohub.org/	Save	Delete
New	Name	URL	Save	

Submission

ID	Name	URL		
1	Snapgene	https://snapsubmit.synbiohub.org/	Save	Delete
2	Excel2SBOL	https://excel2sbol.synbiohub.org/	Save	Delete
New	Name	URL	Save	

Download

ID	Name	URL		
1	Annotated Genbank	https://snapdownload.synbiohub.org/annotate/g	Save	Delete
2	Plasmid Map	https://snapdownload.synbiohub.org/plain/png/	Save	Delete
3	Annotated Zip	https://snapdownload.synbiohub.org/annotate/zi	Save	Delete
New	Name	URL	Save	

Figure 3.1: Plugin Administrative Interface. Adding a plugin to SynBioHub requires registering the name and URL in the administrative interface. A plugin added here will show up on the appropriate page. Only administrative users have access to this page.

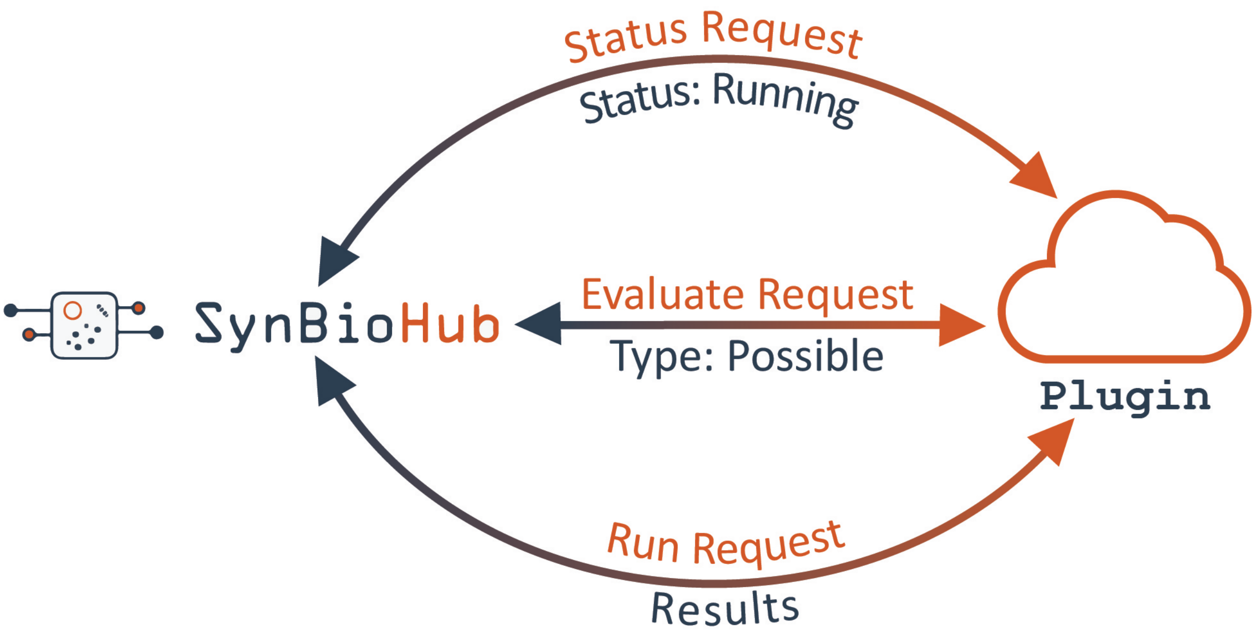


Figure 3.2: Flow diagram illustrating the different steps of communication between the plugin server and SynBioHub. Orange signifies messages from SynBioHub to the Plugin, and blue from the Plugin to SynBioHub. First, SynBioHub checks the status of the plugin. If the plugin is running, the type of object is sent to the plugin. If the plugin responds that it can handle the object type, the full data is sent to the plugin to run and return results to SynBioHub.

- Submit: Submit plugins are available to use from the submit endpoint (Figure 3.3). They work by taking in the file that is uploaded in the submit and processing it to return SBOL to the SynBioHub endpoint.

- Visual: Visual plugins are available on all object pages, for example pages for ComponentDefinitions, Sequences, Activities, etc (Figure 3.4). Visual plugins return HTML to be displayed on the page.

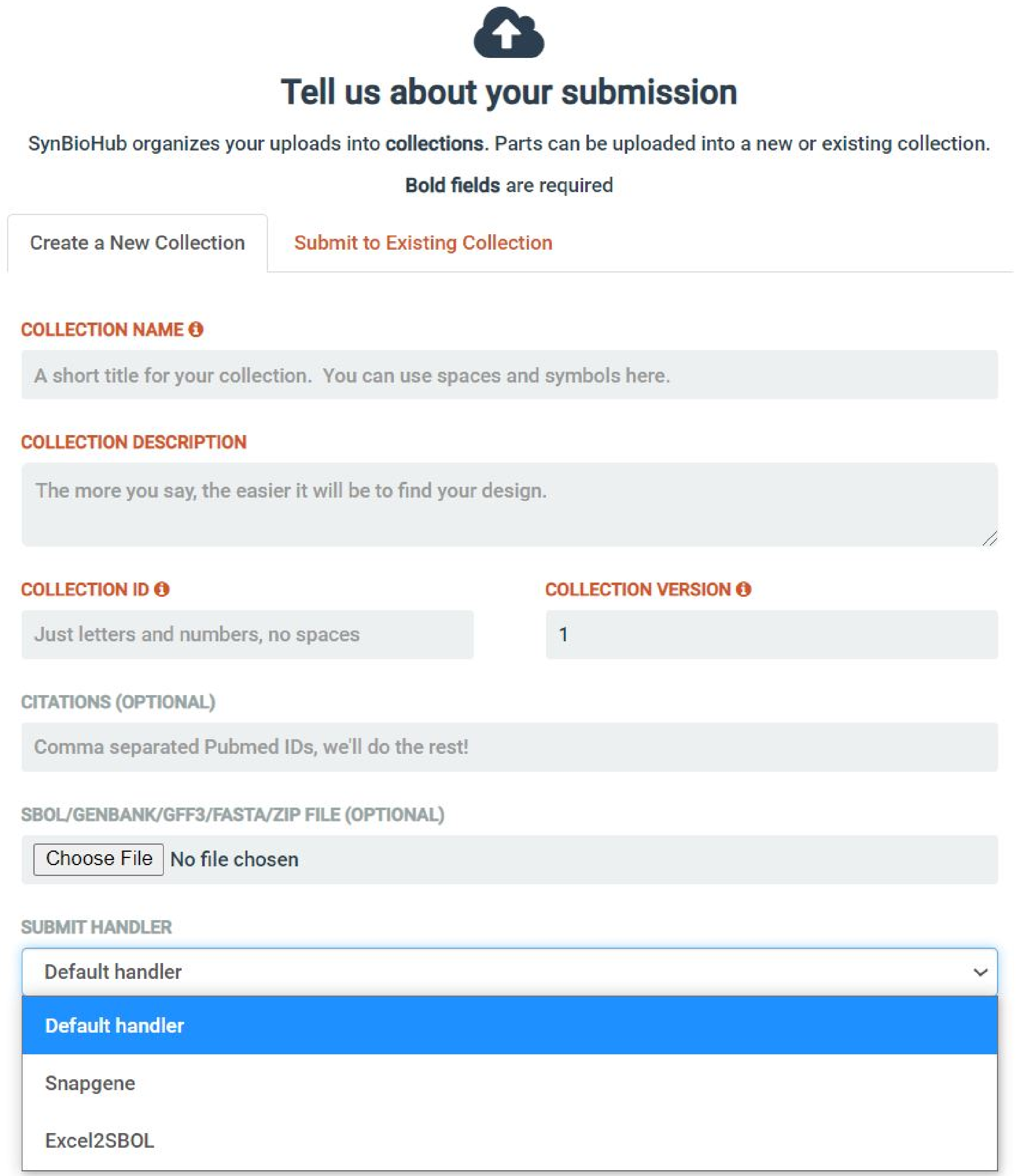
- Download: Download plugins are available on all 'endpoint' pages, for example pages for Components, Sequences, Activities, etc (Figure 3.5). Download plugins return SBOL data that has been processed, generally into a different format, which is then downloaded by the user.

3.1.1 Submit

Submit plugins were designed to allow multiple files to be sent to the plugin at once including some 'helper' files. The plugins operate on entire submissions, rather than individual SBOL constructs. When a user submits something to SynBioHub, they can select a plugin to handle that submission. A manifest for the submission is prepared, describing each file in the submission.

The **status endpoint** (Algorithm 3.1) simply allows get requests and returns a status of 200 and a message that the plugin is up and running. If an error is received when trying to access the plugin, then SynBioHub will not display the particular submit plugin as an option for the user to select. The endpoint is designed to be the simplest possible check that the plugin is up and running.

The **evaluate endpoint** (Algorithm 3.1) is designed to allow the plugin to determine which files are necessary and which can be converted. If the evaluate endpoint responds negatively to a file, then SynBioHub falls back to the default submission handler. The evaluate endpoint was designed so that in future several different submit plugins might be run dynamically. This requires a submission to be sent to multiple files and each plugin returning a list of files it could



Tell us about your submission

SynBioHub organizes your uploads into **collections**. Parts can be uploaded into a new or existing collection.

Bold fields are required

COLLECTION NAME

A short title for your collection. You can use spaces and symbols here.

COLLECTION DESCRIPTION

The more you say, the easier it will be to find your design.

COLLECTION ID

Just letters and numbers, no spaces

COLLECTION VERSION

1

CITATIONS (OPTIONAL)

Comma separated Pubmed IDs, we'll do the rest!

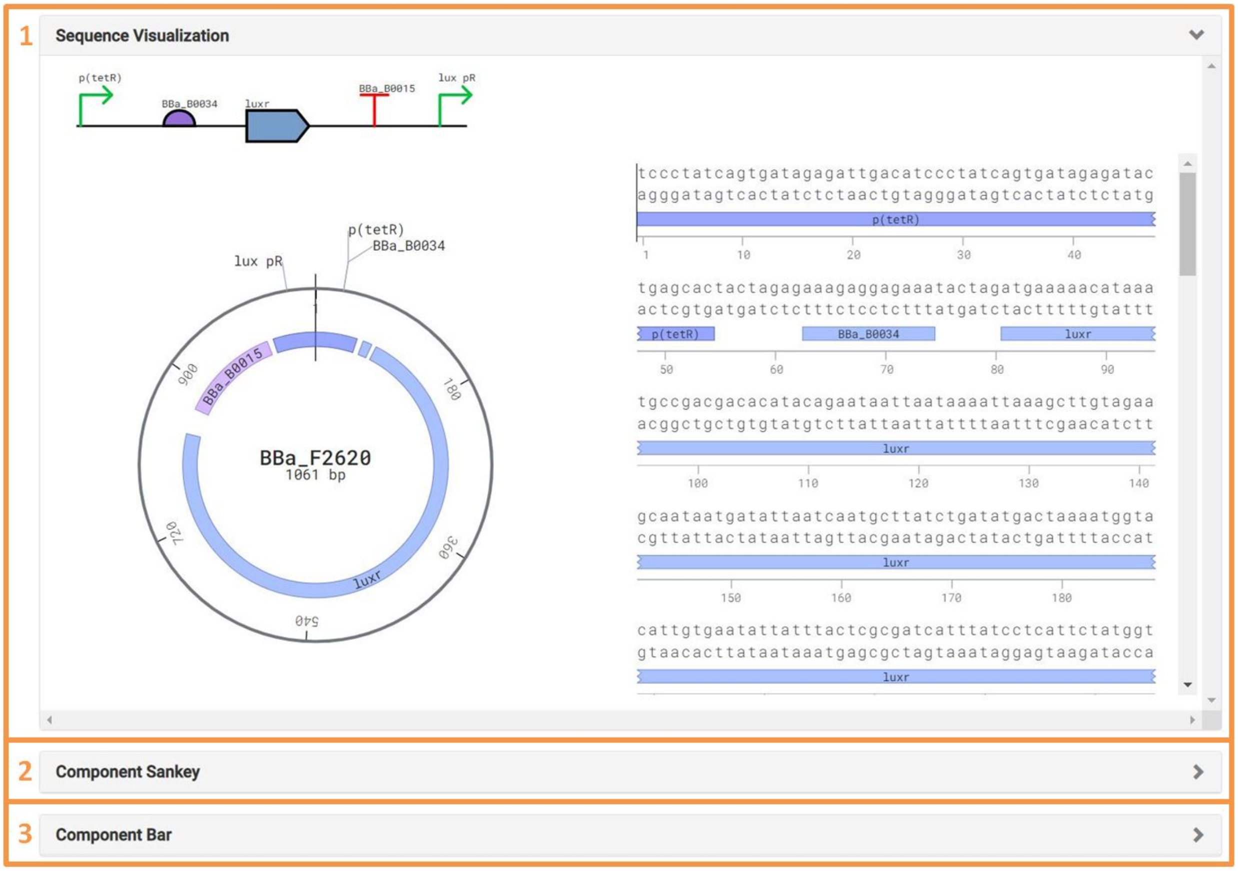
SBOL/GENBANK/GFF3/FASTA/ZIP FILE (OPTIONAL)

Choose File No file chosen

SUBMIT HANDLER

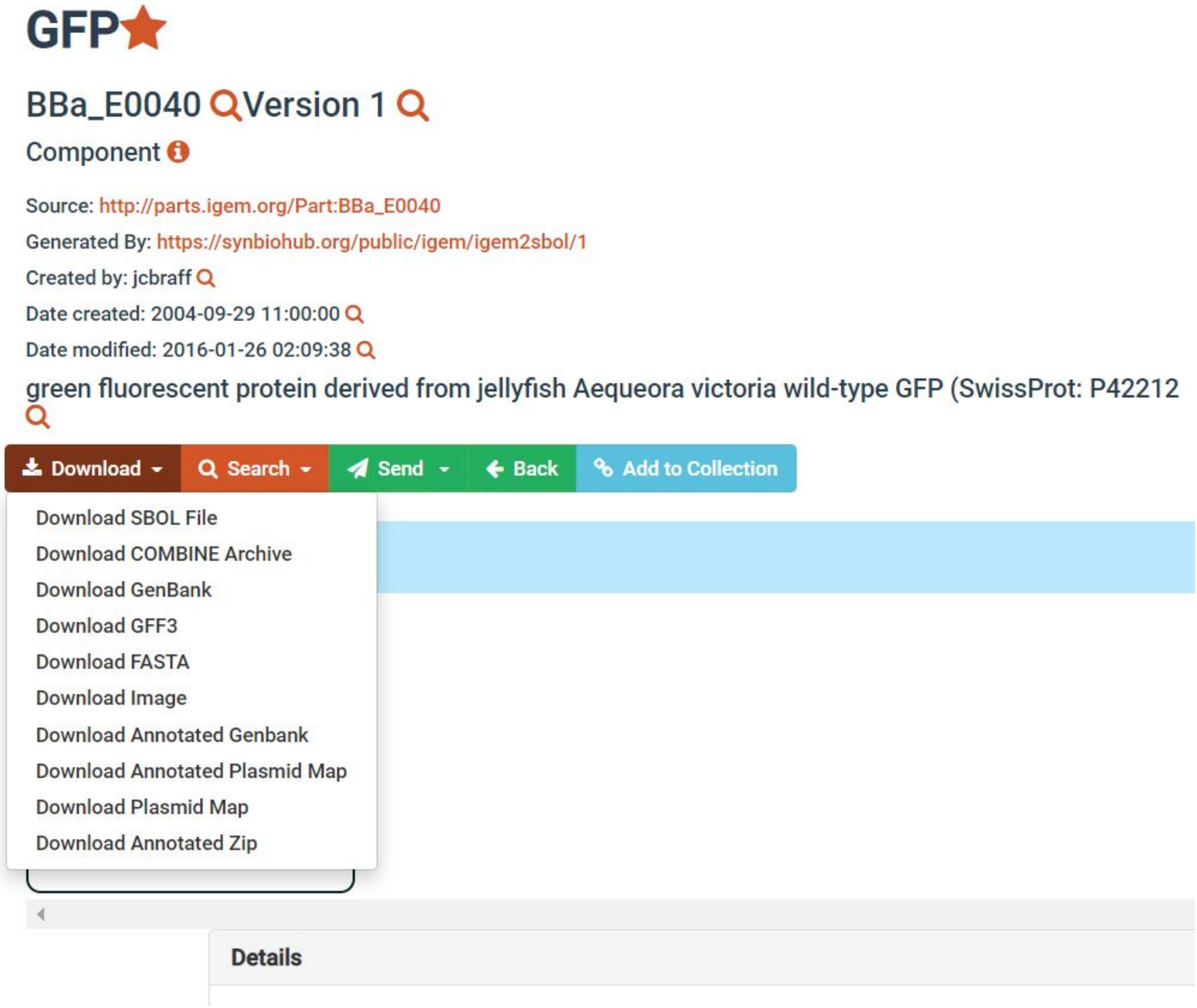
Default handler

Figure 3.3: SynBioHub Submit Interface. This is how new genetic parts are uploaded to SynBioHub. The drop down menu can be used to select submit plugins. There are two plugins: Snapgene and Excel2SBOL in the example.



```
tccctatcagtgatagagattgacatccctatcagtgatagagatac
agggatagtcactatctctaactgtagggatagtcactatctctatg
                      p(tetR)
1       10        20        30        40

tgagcactactagagaaagaggagaaatactagatgaaaaacataaa
actcgtgatgatctctttctcctctttatgatctactttttgtattt
 p(tetR)        BBa_B0034          luxr
  50        60        70        80        90

tgccgacgacacatacagaataattaataaaattaaagcttgtagaa
acggctgctgtgtatgtcttattaattattttaatttcgaacatctt
                      luxr
     100       110       120       130       140

gcaataatgatattaatcaatgcttatctgatatgactaaaatggta
cgttattactataattagttacgaatagactatactgattttaccat
                      luxr
        150       160       170       180

cattgtgaatattatttactcgcgatcatttatcctcattctatggt
gtaacacttataataaatgagcgctagtaaataggagtaagatacca
                      luxr
```

Figure 3.4: Part of a SynBioHub View Endpoint. It shows three visualization plugins. The Sequence Visualisation Plugin is expanded whilst the other two are collapsed.



GFP★

BBa_E0040 Version 1

Component

Source: http://parts.igem.org/Part:BBa_E0040

Generated By: https://synbiohub.org/public/igem/igem2sbol/1

Created by: jcbraff

Date created: 2004-09-29 11:00:00

Date modified: 2016-01-26 02:09:38

green fluorescent protein derived from jellyfish Aequeora victoria wild-type GFP (SwissProt: P42212

Figure 3.5: Download plugin options. The drop down shows the download plugins available for this Component page.

Algorithm 3.1: Submit Plugin API

```
STATUS Endpoint
    SynBioHub: GET Request
    Plugin: Response
        200 and up and running message
EVALUATE Endpoint
    SynBioHub: POST Request
        {'manifest': {'files': [List of dictionaries (one for every file)]}}
        Each file dictionary has the keys:
            • 'url': the single-use URL for the file submitted
            • 'filename': the encrypted file name (with correct extension)
            • 'type': the mime type of the file
    Plugin: Response Data
        {'manifest':[ Dictionary for each file]}
        Each dictionary has the keys:
            • 'filename': the encrypted file name (with correct extension) that matches the
              original filename sent in the manifest from synbiohub
            • 'requirement': an integer 0-2 which indicates whether or not the file can be
              used. 2 means the file will be converted to SBOL, 1 the file will be used to
              convert other files to SBOL, 0 the file cannot be handled/is not useful
RUN Endpoint
    SynBioHub: POST Request
        {'manifest': {'files': [List of dictionaries (one for every file)]}}
        Each dictionary has the keys:
            • 'url': the single-use URL for the file submitted
            • 'filename': the encrypted file name (with correct extension)
            • 'type': the mime type of the file
            • 'instanceUrl': the top-level URL of the synbiohub instance
    Plugin: Response Data
        A zip file which contains the generated SBOL files and a file called manifest.json
          which contains a JSON response manifest. The JSON takes the form:
          {'results':[List of dictionaries (one for every file)]}
        Each dictionary has the keys:
            • 'filename': name of the file within the zip file
            • 'sources' a list of filenames received from SynBioHub that were used to generate the file
```

handle. SynBioHub would then send the files to different plugins in the most efficient manner. Files submitted may be 'convertable' or necessary for conversion. The additional 'informational' files might include a set of parameters for the plugin to use in conversion. This is a way to pass parameters to a plugin without providing a user interface, which is more difficult to create and maintain for plugin developers. The information sent to the evaluate endpoint about each different file is the URL, the encrypted file name (with the correct extension), and the **MIME** (Multipurpose Internet Mail Extension) type of the file. The URL is given so that the full file may be read to determine whether or not conversion can be carried out. A URL is given rather than sending the file as this reduces the size of the data transmission and not all evaluate endpoints will look at file contents, for some, the file type may be enough. The encrypted file name with the correct extension is given to allow the appropriate response to be constructed. Additionally, it allows a plugin to be written that makes decisions based on file extension types (a way of parameterising) or the combination of the file extension and the MIME type. The MIME type is given as file extensions are not always agreed on but the MIME type is the official internet protocol for defining the file type being transferred.

The **run endpoint** (Algorithm 3.1) is designed to actually call the plugin for conversion. If the plugin's run endpoint is not successful, SynBioHub falls back to the default submission handler. The run endpoint is sent almost the same information as the evaluate endpoint. However, any files that were considered not useful by the evaluate endpoint are omitted from the dictionary. Additionally, each file also has the 'instanceURL' given. This URL is given so that the plugin knows which SynBioHub instance is making the request. This information might be necessary if any searching of the SynBioHub instance is done before submission. For example a more advanced submission plugin could link all new components to existing sequence instances found in SynBioHub.

3.1.2 Visualization

Visualization plugins were designed to allow additional information to be depicted on SBOL object pages. An example of this is shown in [87]. They may display text, images, or even interactive windows.

The **status endpoint** (Algorithm 3.2) simply allows get requests and returns a status of 200 and a message that the plugin is up and running. If an error is received when trying to access the plugin, then SynBioHub will not contact the evaluate endpoint. It was made as simple as possible and thus uses the GET request method rather than exchanging any data.

The **evaluate endpoint** (Algorithm 3.2) of a visualization plugin is again kept as simple as possible. It receives a single RDF-type (e.g. 'Activity', or 'Component') and then returns a 200 status if the plugin will accept the type and 4XX (e.g. 404, 415) otherwise. The RDF-type was chosen as the information as all information will be SBOL and stored as RDF triples. Thus the type provides the best way to differentiate between objects. If a 200 status is received by SynBioHub, then the plugin 'box' will be shown on the endpoint page, otherwise the 'box' will be removed.

The **run endpoint** (Algorithm 3.2) receives a number of URLs so it can fetch the information required to create the visualization. Again URLs are sent rather than files as it reduces the size of the RUN request. The URLs sent are limited to reduce the complexity of the request and maintenance of the plugin architecture, whilst maintaining the flexibility of the visualization plugins. The complete SBOL gives the SBOL description of the object, as well as all top-level objects that are referenced by the original object. This is useful for visualizations which consider child objects, e.g. the sub-components of an object. The shallow SBOL only provides the information about the top-level object and its associated child objects and thus is more compact than the complete SBOL. Fetching this takes less time than fetching the complete SBOL and is the more efficient route if the complete SBOL is not needed. The GenBank version of a file is provided for ComponentDefinitions. This is provided as many tools do not yet support SBOL, but rather only support GenBank. Thus, providing the Genbank URL ensures the visualization plugins can be used to integrate visualizations from existing workflows. The top-level URL of an object is provided so that, if needed, more information can be pulled from the object page. This instance URL is needed as the top level URL will be the same for synbiohub.org and dev.synbiohub.org as these two instances spoof each other. The size of the object is a number that indicates the number of triples in the triple store

Algorithm 3.2: Visualization Plugin API

```
STATUS Endpoint
    SynBioHub: GET Request
    Plugin: Response
        200 and up and running message
EVALUATE Endpoint
    SynBioHub: POST Request
        {'type': RDF-Type of the top-level object}
    Plugin: Response Data
        200 if the type is acceptable and 4XX status if it isn't
RUN Endpoint
    SynBioHub: POST Request
        {'complete_sbol': URL, 'shallow_sbol': URL, 'genbank': URL, 'top_level': URL,
          'instanceUrl': URL, 'size': number, 'type': RDF-type}
        The meaning of the different keys is explained below:
            • 'complete_sbol': The single-use URL for the complete SBOL of the object
            • 'shallow_sbol': The single-use URL for a truncated SBOL file of the the object
            • 'genbank': The single-use URL for the Genbank of the object (Note: This will
              lead to a blank website for all RDF-types other than ComponentDefinition)
            • 'top_level': The top-level URL of the SBOL object
            • 'instanceUrl': The top-level URL of the SynBioHub instance
            • 'size': A number representing an estimate of the size of the object (the number
              of triples about an object)
            • 'type': The RDF type of the top-level object (this is the same as was sent to
              the evaluate endpoint
    Plugin: Response Data
        HTML that contains the visual to be displayed. It is sent as a text response not a
          file attachment.
```

(e.g. Virtuoso) that relate to the object. It is provided so that a plugin can choose to use the shallow SBOL for large objects and the complete SBOL otherwise. Doing this, allows plugins to pull additional information as needed for large objects and pull a larger selection of all available information in the case of small objects. The type of the object is provided as some visualization plugins may be able to handle multiple different object types but have specific behaviors for different types.

3.1.3 Download

Download plugins are designed to allow SBOL data to be downloaded in other formats. For example, the download may be a ZIP file containing multiple files inside, visualizations generated based on the SBOL object, or other formats such as the Snapgene DNA format.

The **status endpoint** (Algorithm 3.3) allows get requests and returns a status of 200 and a message that the plugin is up and running. If an error is received when trying to access the plugin, then SynBioHub will not contact the evaluate endpoint. This is a simple check to see if the plugin is up and running.

The **evaluate endpoint** (Algorithm 3.3) of a download plugin is relatively simple. It receives a single RDF-type (e.g. 'Activity', or 'ComponentDefinition') and then returns a 200 status if the plugin will accept the type and 4XX (e.g. 404, 415) otherwise. If a 200 status is received, the plugin download button is shown on the endpoint page, otherwise the button is removed. The reasoning for using the RDF-type is the same as for the visualization evaluate endpoint. Keeping, the two evaluate endpoints the same between visualization and download also makes it easier for users to learn to use the second kind of plugin after understanding the first.

The **run endpoint** (Algorithm 3.3) receives the same information as the visualization endpoint. The reason behind each of the URLs is the same. The use of the same information between the visualization and download run endpoints means there is less for users to understand when introduced to the plugin structure. Additionally, it is easier to maintain on the SynBioHub side.

Algorithm 3.3: Download Plugin API

```
STATUS Endpoint
    SynBioHub: GET Request
    Plugin: Response
        200 and up and running message
EVALUATE Endpoint
    SynBioHub: POST Request
        {'type': RDF-Type of the top-level object}
    Plugin: Response Data
        200 if the type is acceptable and 4XX status if it isn't
RUN Endpoint
    SynBioHub: POST Request
        {'complete_sbol': URL, 'shallow_sbol': URL, 'genbank': URL, 'top_level': URL,
         'instanceUrl': URL, 'size': number, 'type': RDF-type}
        The meaning of the different keys is explained below:
            • 'complete_sbol': The single-use URL for the complete SBOL of the object
            • 'shallow_sbol': The single-use URL for a truncated SBOL file of the the object
            • 'genbank': The single-use URL for the Genbank of the object (Note: This will
              lead to a blank website for all RDF-types other than ComponentDefinition)
            • 'top_level': The top-level URL of the SBOL object
            • 'instanceUrl': The top-level URL of the SynBioHub instance
            • 'size': A number representing an estimate of the size of the object (the number
              of triples about an object)
            • 'type': The RDF type of the top-level object (this is the same as was sent to
              the evaluate endpoint
    Plugin: Response Data
        An HTTP response that contains the file to be downloaded as an attachment to the
         request.
```

3.2 Plugin Templates

Plugin templates were created to facilitate plugin development. These templates serve three roles: 1) to provide the most basic example of the different kinds of plugin, 2) to make SynBioHub testing easier, and 3) to provide a simple example for new developers to edit to enhance their understanding of plugins, thus lowering the bar for plugin development. The templates are stored as GitHub repository templates to allow them to be easily cloned. The templates not only contain code but also a set of GitHub Actions which automate commonly performed operations associated with plugin development. Currently there are 6 templates, three python templates (submit, visualization, and download), and three JavaScript templates (submit, visualization, and download). More detail is given about the templates below. Additionally, for the testing of plugins a standard set of API calls was created (`https://github.com/SynBioHub/Postman`). This can be used together with a tool such as Postman (`https://www.postman.com/`) to test plugins without needing to understand how to run a personal SynBioHub. This makes plugin development more modular and testing simpler.

3.2.1 Templates and Docker

All three types of plugins are API servers. Thus, they can stand as independent applications that are run via the API tools provided by different languages (e.g. Python's Flask and JavaScript's Node). However, this may require installing Python or JavaScript and appropriate tools on the server on which plugins are to be run. This can be difficult with versioning of the language, or clashes in package dependencies. Thus, Docker containers were created for each plugin to make it as easy as possible for plugins to be used by inexperienced programmers. This allows all plugins to be installed and run by installing a Docker image. To make this easier the templates provide examples of the necessary Dockerfiles and provide GitHub Actions that can automatically create a Docker image and push it to DockerHub for the repository.

For Python a standard Docker base image is used (synbiohub/docker-base-python:snapshot) which has Python3 installed, as well as the Flask and Waitress package. The 5000 port is exposed (which is the standard Flask port), all requirements from the requirements.txt file are installed, and the Flask application called app.py is run.

The JavaScript DockerFile is based on the node:12 Docker image. It copies in all required packages from files named package*.json (e.g. package.json and package-lock.json). Then it installs all the listed dependencies required for production (not the development dependencies). It copies in all files from the folder it is in, exposes port 5000 (this is used to make it more similar to the Python version which has a default), and finally runs the node app found in app.js.

The use of Docker images also allows multiple plugins to be run together via Docker compose files. This gives developers the opportunity to run plugins alongside a local SynBioHub instance without setting a domain name for every plugin. It is still possible to run plugins as stand alone images if that works better with debugging infrastructure.

3.2.2 Python Templates

All python templates use the same structure to make it easier for developers to use each of the different kinds of plugin. Each template contains a folder of python test files, a License file, a ReadMe file, a Dockerfile, a requirements.txt file, an app.py file and 3 GitHub actions (automatically run tasks for software development workflows (`https://github.com/features/actions`)).

The License file indicates that the plugin is open source. The ReadMe file provides background information. The Dockerfile is used together with the requirements.txt file to create the Docker image. The app.py file contains the template Flask application. There are three GitHub Actions, all three run when changes are pushed to the main branch:

- release: to automatically create a Docker image with the repository name and push it to Docker Hub.
- linting: this checks the Python code follows the flake8 format. Doing so provides additional standardization between plugins.

- testing: This runs all the pytest modules found in the test folder. It checks the plugin has a status, evaluate, and run endpoint.

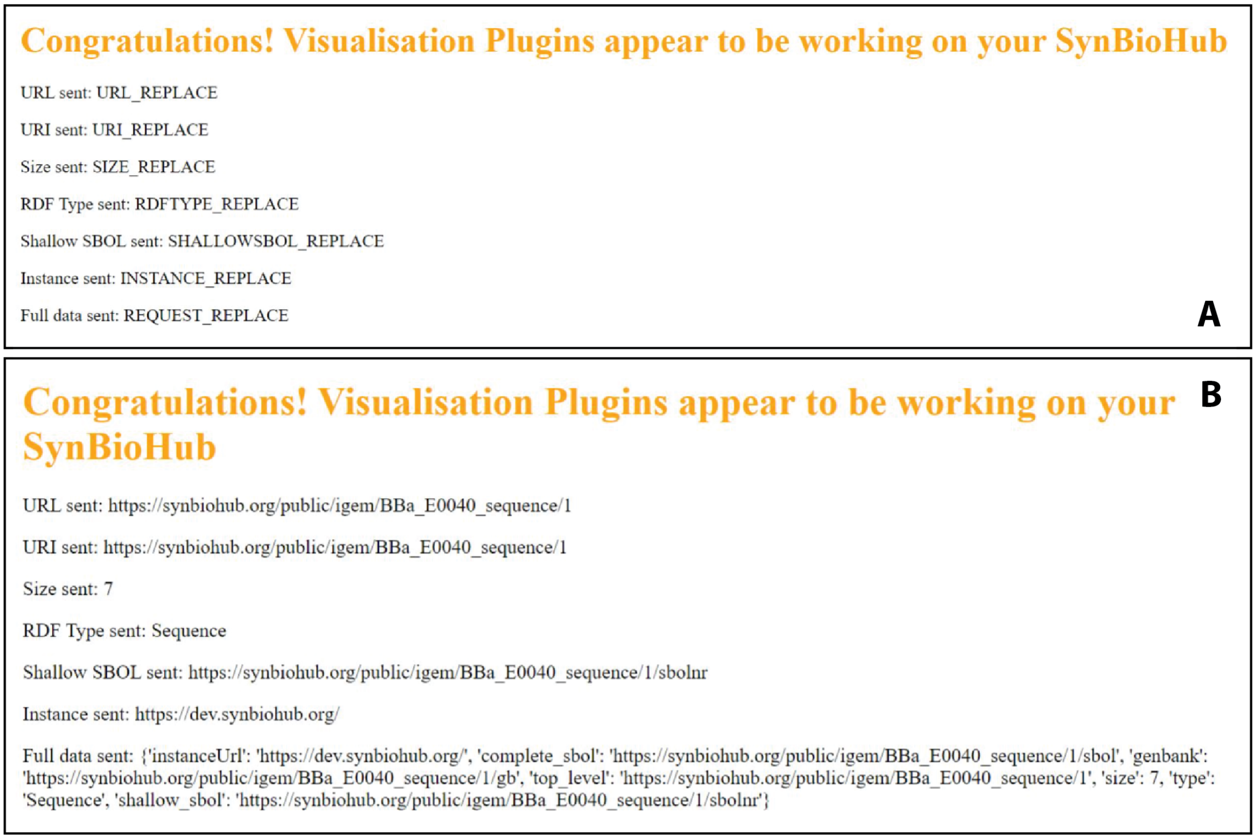
Whilst many things are standardized across the three types of plugin, the actual Flask templates are necessarily slightly different.

3.2.2.1 Submit

The submit test plugin has a status endpoint that only allows GET requests and returns the message “The Submit Test Plugin Flask Server is up and running”. The evaluate endpoint iterates over each of the file entries found in the manifest it receives and creates a response to be returned. This endpoint is easy to edit as the developer simply has to ensure the accepted types are listed in the acceptable_types dictionary and the additionally useful types are listed in the useful_types dictionary. The lines where these two dictionaries are defined are clearly indicated by a demarcation in the code of “REPLACE THIS SECTION WITH OWN RUN CODE” and “END SECTION ” For the run endpoint a temporary directory is created. Then, for every file the appropriate manifest is created and the variable sbolcontent is written out to a file in the temporary directory. After all the files have been processed, the manifest file is added to the temporary directory, it is zipped and sent back to SynBioHub. For the example, all files are replaced with a Test.xml file which provides all the information sent to SynBioHub as the mutable description. The GitHub repository for the Python Submit template is found at `https://github.com/SynBioHub/Plugin-Submit-Test`.

3.2.2.2 Visualisation

The visual test plugin has a status endpoint that returns the message “The Visual Test Plugin Flask Server is up and running”. The evaluate endpoint checks if the RDF-type given is a type that the plugin can handle. This endpoint is easy to edit as the developer simply has to ensure the accepted types are listed in the accepted_types dictionary. The lines where these two dictionaries are defined are clearly indicated by a demarcation in the code of “REPLACE THIS SECTION WITH OWN RUN CODE” and “END SECTION ” For the run endpoint, HTML is created and returned based on the input URLs. For the template plugin, the URL returns a message that the plugin has run successfully and the data sent in the request is as shown in Figure 3.6. The Github repository for the Python visualization template is found at `https://github.com/SynBioHub/Plugin-Visual-Test`. Note there is also a visual-serve-test repository at `https://github.com/SynBioHub/Plugin-Visual-Serve-Test`. This repository is the same as the basic repository, however, it has the code to allow images to be returned in the HTML.



Congratulations! Visualisation Plugins appear to be working on your SynBioHub

URL sent: URL_REPLACE

URI sent: URI_REPLACE

Size sent: SIZE_REPLACE

RDF Type sent: RDFTYPE_REPLACE

Shallow SBOL sent: SHALLOWSBOL_REPLACE

Instance sent: INSTANCE_REPLACE

Full data sent: REQUEST_REPLACE

Congratulations! Visualisation Plugins appear to be working on your SynBioHub

URL sent: https://synbiohub.org/public/igem/BBa_E0040_sequence/1

URI sent: https://synbiohub.org/public/igem/BBa_E0040_sequence/1

Size sent: 7

RDF Type sent: Sequence

Shallow SBOL sent: https://synbiohub.org/public/igem/BBa_E0040_sequence/1/sbolnr

Instance sent: https://dev.synbiohub.org/

Full data sent: {'instanceUrl': 'https://dev.synbiohub.org/', 'complete_sbol': 'https://synbiohub.org/public/igem/BBa_E0040_sequence/1/sbol', 'genbank': 'https://synbiohub.org/public/igem/BBa_E0040_sequence/1/gb', 'top_level': 'https://synbiohub.org/public/igem/BBa_E0040_sequence/1', 'size': 7, 'type': 'Sequence', 'shallow_sbol': 'https://synbiohub.org/public/igem/BBa_E0040_sequence/1/sbolnr'}

Figure 3.6: Visual template. **A**: The template HTML response. **B**: Specific example of the response. Note how each of the REPLACE found in the template has been replaced with specific values. These values are the values that were sent in the API call the the plugin. Thus, this plugin can be used to check that the API call is happening as expected.

3.2.2.3 Download

The download test plugin has a status endpoint that returns the message "The Visual Test Plugin Flask Server is up and running". The evaluate endpoint checks if the RDF-type given is a type that the plugin can handle. This endpoint is easy to edit as the developer simply has to ensure the accepted types are listed in the accepted_types dictionary. The lines where these two dictionaries are defined are clearly indicated by a demarcation in the code of "REPLACE THIS SECTION WITH OWN RUN CODE" and "END SECTION " For the run endpoint, a temporary directory is created. The file with the name stored in the variable 'download_file_name' is sent back to SynBioHub as an attachment. For the download template, the file returned is an HTML file that contains the information sent to the run endpoint as shown in Figure 3.7. The Github repository for the Python download template is found at `https://github.com/SynBioHub/Plugin-Download-Test`.

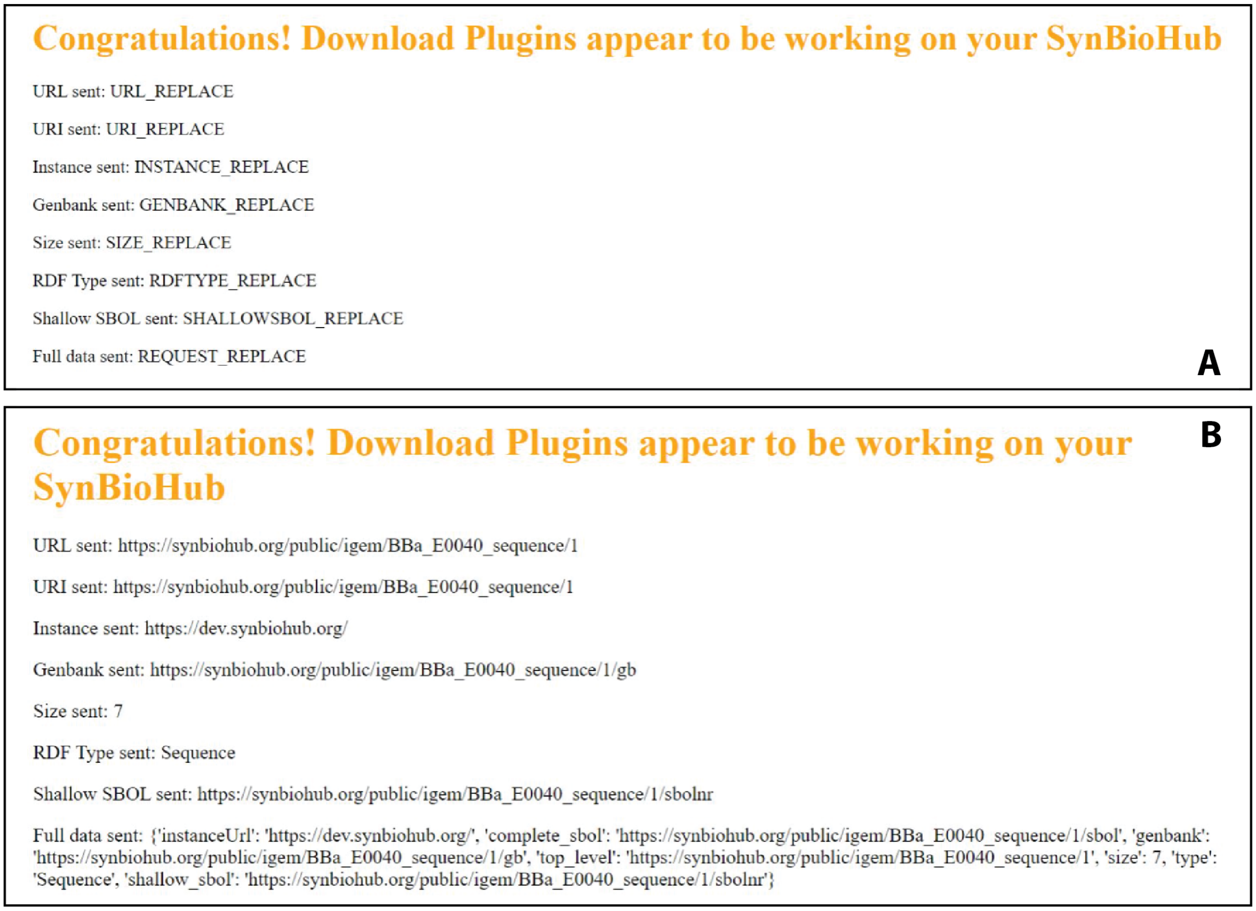
3.2.3 JavaScript Templates

The javascript template repositories all contain automatic actions to push docker images on a pull request. The docker image will be pushed to "synbiohub/lowercase_repository_name:snapshot" e.g. synbiohub/plugin-submit-test-js:snapshot. The repositories closely mimic the Python ones including the use of 'REPLACE THIS SECTION' headers. The repositories can be found at:

- Submit: `https://github.com/SynBioHub/Plugin-Submit-Test-js`
- Visualiation: `https://github.com/SynBioHub/Plugin-Visual-Test-js`
- Visualisation-serve: `https://github.com/SynBioHub/Plugin-Visual-Serve-Test-js`
- Download: `https://github.com/SynBioHub/Plugin-Download-Test-js`

3.3 New Plugin Specifications

Apart from the three original types of plugins (submit, visualization, and download), four additional types of plugin are proposed: curation, search, index, and link. These plugin types are proposed to fill gaps in the SynBioHub pipeline that are not addressed by the original three plugin types: **curation** to add metadata to SBOL and make the SBOL files more informative, **search** to



Congratulations! Download Plugins appear to be working on your SynBioHub

URL sent: URL_REPLACE

URI sent: URI_REPLACE

Instance sent: INSTANCE_REPLACE

Genbank sent: GENBANK_REPLACE

Size sent: SIZE_REPLACE

RDF Type sent: RDFTYPE_REPLACE

Shallow SBOL sent: SHALLOWSBOL_REPLACE

Full data sent: REQUEST_REPLACE

Congratulations! Download Plugins appear to be working on your SynBioHub

URL sent: https://synbiohub.org/public/igem/BBa_E0040_sequence/1

URI sent: https://synbiohub.org/public/igem/BBa_E0040_sequence/1

Instance sent: https://dev.synbiohub.org/

Genbank sent: https://synbiohub.org/public/igem/BBa_E0040_sequence/1/gb

Size sent: 7

RDF Type sent: Sequence

Shallow SBOL sent: https://synbiohub.org/public/igem/BBa_E0040_sequence/1/sbolnr

Full data sent: {'instanceUrl': 'https://dev.synbiohub.org/', 'complete_sbol': 'https://synbiohub.org/public/igem/BBa_E0040_sequence/1/sbol', 'genbank': 'https://synbiohub.org/public/igem/BBa_E0040_sequence/1/gb', 'top_level': 'https://synbiohub.org/public/igem/BBa_E0040_sequence/1', 'size': 7, 'type': 'Sequence', 'shallow_sbol': 'https://synbiohub.org/public/igem/BBa_E0040_sequence/1/sbolnr'}

Figure 3.7: Download Template. **A**: The template HTML response. Note the template stored in the plugin directory has a lot of REPLACE placeholders. **B**: Specific example of the response. his is an example of the file returned by the download plugin. It uses the template, shown above, and replaces the place holders with the information sent in the API call. This can then be used to verify that the API call is happening as expected.

extend the search capabilities, **index** to provide better search result interfaces and allow temporary storage of data that has not been integrated into the SBOL standard, and **link** to make it easier to communicate with other SBOL tools. These four types of plugins are described below. They have not been implemented in SynBioHub, but templates have been created to varying extents.

3.3.1 Curation

Curation plugins are designed to improve SBOL via curation. They prompt users to provide more metadata or additional annotations on existing SBOL objects. These plugins will allow users to input parameters, return proposed changes, and save the changes the user decides on. A template for the plugin has been created in the repository: `https://github.com/SynBioHub/Plugin-Curation-Test`.

The **status endpoint** checks that the plugin is up and running before providing the curation button on the SBOL object page (Algorithm 3.4).

Algorithm 3.4: Curation Plugin API: STATUS

```
STATUS Endpoint
    SynBioHub: GET Request
    Plugin: Response
        200 and up and running message
```

The **evaluate endpoint** checks if the object type can be curated by the plugin (Algorithm 3.5). If it can, then a parameter interface can be returned by the plugin. This is either HTML to allow users to fill out the required parameters, or a standard formatted JSON to allow SynBioHub to create an HTML form. This endpoint allows users to provide parameters to use in the curation process. The ability to send HTML allows fully custom interfaces to be designed for parameter collection. The alternative of sending JSON ensures that developers with little user interface development experience can still create curation plugins. Additionally, providing JSON rather than HTML to SynBioHub gives the opportunity to standardize the layout of the parameters endpoint between forms by using similar style sheets. Note in the case of calling plugins via

API rather than through SynBioHub the evaluate endpoint could be skipped and the appropriate parameters sent directly to the RUN endpoint.

The **run endpoint** receives any parameters that the user entered in the evaluate form (Algorithm 3.6). Curation is then run and an interface can be returned to allow the user to select which changes to use. The interface is created in much the same way as the evaluate endpoint; allowing either HTML or the list to create HTML. If no curation interface is required then SynBioHub passes directly to the SAVE endpoint. This endpoint allows users to provide feedback about the automatic curation carried out. For example, if sequence annotations were created this endpoint is how users can decide which sequence annotations to accept.

Finally, the **save endpoint** takes in any user data entered into the RUN endpoint form (Algorithm 3.7). The save endpoint uses this data to create the finished SBOL. This SBOL object is returned to SynBioHub for validation. The validation will check the SBOL is valid as well as the SBOL scope (this prevents a plugin overwriting a large chunk of SynBioHub data and only allows it to update a small number of objects). The evaluate parameters are again given to this endpoint in case the run endpoint had no interaction and the user input to create the SBOL is required at the save endpoint instead.

3.3.2 Search

Search plugins are designed to provide an intake of search parameters, perform a search, and return the search output. They should make extending search functionality (like in this work [239]) easier. For both the intake of queries and the output of parameters, a default is provided as well as allowing users to specify their own HTML intake or results view. This allows multiple different parameters to be taken in, and results to be displayed as networks or anything else the user might wish. The search endpoint also provides facetting to display the top choices in results. The facetting interface is based off of the Elasticsearch (`https://www.elastic.co/elasticsearch/`) facet interface to make it compatible with Elasticsearch queries. The default output is based on the SPARQL JSON output to make it compatible with SPARQL queries. An example template is

Algorithm 3.5: Curation Plugin API: EVALUATE

```
EVALUATE Endpoint
    SynBioHub: POST Request
        {‘complete_sbol’: URL, ‘shallow_sbol’: URL, ‘genbank’: URL, ‘top_level’: URL,
         ‘instanceUrl’: URL, ‘size’: number, ‘type’: RDF-type, ‘submit_link’:url}
        The meaning of the different keys is explained below:
            • ‘complete_sbol’: The single-use URL for the complete SBOL of the object
            • ‘shallow_sbol’: The single-use URL for a truncated SBOL file of the the object
            • ‘genbank’: The single-use URL for the Genbank of the object (Note: This will
              lead to a blank website for all RDF-types other than Component)
            • ‘top_level’: The top-level URL of the SBOL object
            • ‘instanceUrl’: The top-level URL of the SynBioHub instance
            • ‘size’: The number of RDF triples about an object
            • ‘type’: The RDF type of the top-level object
            • ‘submit_link’: the SynBioHub link to which the HTML form should submit
    Plugin: Response Data
        {‘needs_interface’: Boolean, ‘own_interface’: Boolean, ‘submit_link’:url, ‘interface’:
         HTML or list}
        The meaning of the different keys is explained below:

            • ‘needs_interface’: Boolean that tells SynBioHub whether any parameters need
              to be added. If this is false then all the other keys can be disregarded and can
              pass on to the run endpoint
            • ‘own_interface’: Boolean to tell SynBioHub what kind of interface to expect
              back. If True than HTML will be sent as the interface, if False than a list
              object indicating an HTML form will be sent back.
            • ‘submit_link’: The URL for SynBioHub to use if creating a form from the
              single use object. It can be the same as the link SynBioHub sent or can point
              to another plugin endpoint. It is not used if the own_interface variable is set
              to True.
            • ‘interface’: An HTML object or list to allow parameter entry. If a list, it has
              an entry for every variable of the HTML form to be created. Each entry is a
              dictionary with the keys:

                ∗ type: string, HTML-form input type
                ∗ label: string, heading to give above the input entry
                ∗ description: string, description/help text to provide to the user
                ∗ options: list of options, empty list unless the input is a radio or checkbox
                ∗ default: list, list providing the default selection
                ∗ restrictions: dictionary providing any further options that can be pro-
                  vided to HTML forms, e.g. pattern is a key for the regex expression used
                  to check the input
```

Algorithm 3.6: Curation Plugin API: RUN

```
RUN Endpoint
    SynBioHub: POST Request
        {‘complete_sbol’: URL, ‘shallow_sbol’: URL, ‘genbank’: URL, ‘top_level’: URL,
         ‘instanceUrl’: URL, ‘size’: number, ‘type’: RDF-type, ‘submit_link’: URL,
         ‘eval_params’: dictionary}
        The meaning of the different keys is explained below:
            • ‘complete_sbol’: The single-use URL for the complete SBOL of the object
            • ‘shallow_sbol’: The single-use URL for a truncated SBOL file of the the object
            • ‘genbank’: The single-use URL for the Genbank of the object (Note: This will
              lead to a blank website for all RDF-types other than Component)
            • ‘top_level’: The top-level URL of the SBOL object
            • ‘instanceUrl’: The top-level URL of the SynBioHub instance
            • ‘size’: A number of RDF triples about the object
            • ‘type’: The RDF type of the top-level object
            • ‘submit_link’: the SynBioHub link to which the HTML form should submit
            • ‘eval_params’: A dictionary providing the output values from the evaluate
              form. Is empty if there is no output. Otherwise of the form {‘variable 1’:
              [‘text input’], ‘variable 2’: [‘option 1’], ‘variable 3’: [‘option 1’, ‘option 2’]}
    Plugin: Response Data
        {‘needs_interface’: Boolean, ‘own_interface’: Boolean, ‘submit_link’:url, ‘interface’:
         HTML or list}
        The meaning of the different keys is explained below:
            • ‘needs_interface’: Boolean that tells SynBioHub whether any parameters need
              to be added. If this is false then all the other keys can be disregarded and can
              pass on to the save endpoint
            • ‘own_interface’: Boolean to tell SynBioHub what kind of interface to expect
              back. If True than HTML will be sent as the interface if False than a list
              object indicating an HTML form will be sent back.
            • ‘submit_link’: The URL for SynBioHub to use if creating a form from the
              single use object. It can be the same as the link SynBioHub sent or to another
              plugin endpoint. Is not used if the own_interface variable is set to True.
            • ‘interface’: An HTML object or list to allow parameter entry. If a list, it has
              an entry for every variable of the HTML form to be created. Each entry is a
              dictionary with the keys:
                  ∗ type: string, HTML-form input type
                  ∗ label: string, heading to give above the input entry
                  ∗ description: string, description/help text to provide to the user
                  ∗ options: list of options, empty list unless the input is a radio or checkbox
                  ∗ default: list, list providing the default selection
                  ∗ restrictions: dictionary providing any further options that can be pro-
                    vided to HTML forms, e.g. pattern is a key for the regex expression used
                    to check the input
```

Algorithm 3.7: Curation Plugin API: SAVE

```
SAVE Endpoint
    {‘complete_sbol’: URL, ‘shallow_sbol’: URL, ‘genbank’: URL, ‘top_level’: URL,
     ‘instanceUrl’: URL, ‘size’: number, ‘type’: RDF-type, ‘submit_link’: URL,
     ‘eval_params’: dictionary, ‘run_params’: dictionary}
    The meaning of the different keys is explained below:
        • ‘complete_sbol’: The single-use URL for the complete SBOL of the object
        • ‘shallow_sbol’: The single-use URL for a truncated SBOL file of the the object
        • ‘genbank’: The single-use URL for the Genbank of the object (Note: This will
          lead to a blank website for all RDF-types other than Component)
        • ‘top_level’: The top-level URL of the SBOL object
        • ‘instanceUrl’: The top-level URL of the SynBioHub instance
        • ‘size’: A number representing an estimate of the size of the object (the number
          of triples about an object)
        • ‘type’: The RDF type of the top-level object (this is the same as was sent to
          the evaluate endpoint
        • ‘eval_params’: A dictionary providing the output values from the evaluate
          form. Is empty if there is no output. Otherwise of the form {‘variable 1’:
          [‘text input’], ‘variable 2’: [‘option 1’], ‘variable 3’: [‘option 1’, ‘option 2’]}
        • ‘run_params’: A dictionary providing the output values from the run form. Is
          empty if there is no output. Otherwise of the form {‘variable 1’: [‘text input’],
          ‘variable 2’: [‘option 1’], ‘variable 3’: [‘option 1’, ‘option 2’]}

    Plugin: Response Data
        SBOL is returned (inline not as a file)
```

provided in the repository: `https://github.com/SynBioHub/Plugin-Search-Test`. Note these plugins also allow the caching of SPARQL queries to make advanced querying more repeatable and more accessible to graphical interface users.

The **status endpoint** checks that the plugin is up and running before providing the search button on the search home page (Algorithm 3.8).

Algorithm 3.8: SEARCH Plugin API: STATUS

```
STATUS Endpoint
    SynBioHub: GET Request
    Plugin: Response
        200 and up and running message
```

The **parameters endpoint** provides the search interface to SynBioHub (Algorithm 3.9). Search may be carried out with only a button click, it may require input from the standard search bar, it may require an HTML form generated by SynBioHub based on the list of parameters sent to SynBioHub, or it may require the display of the HTML returned to SynBioHub. The different intial input options are provided to enable a wide range of search plugins to be developed. A simple button may be used in the case of searches like "top ten genetic components" which requires no input. The standard search bar may be used with a plugin such as "organism search". The plugin can take in "E. coli" and specifically match the term against organism properties. An HTML form may be used for more complex queries which require multiple parameters, including ones that are not free text. For example, the advanced search interface could be turned into a plugin via this kind of form. Finally, an HTML file might be used for complex graphical search input. For example, by allowing the selection of parameters based off of relationship maps. This endpoint manages to support each of these possibilities whilst receiving and returning relatively few parameters. Additionally, the standard HTML forms that can be used to collect search parameters are the same kind as are used in the proposed curation plugins. This standardization between plugin types helps reduce the difficulty of developing new plugins and the complexity of implementation in SynBioHub.

The **run endpoint** receives parameters from the parameters endpoint and returns a search

Algorithm 3.9: SEARCH Plugin API: PARAMETERS

PARAMTERS Endpoint

SynBioHub: POST Request

{'instanceUrl': URL, 'submit_link': URL, 'sparql_link': URL, 'es_link': URL}

The meaning of the different keys is explained below:

- 'instanceUrl': The top-level URL of the SynBioHub instance
- 'submit_link': the SynBioHub link to which the HTML form should submit
- 'sparql_link': URL at which the sparql endpoint may be accessed, could build in authorization if desired
- 'es_link': URL at which the elastic search endpoint may be accessed, could build in authorization if desired (this endpoint allows the plugin to do fuzzy string matching).

Plugin: Response Data

{'search_on_click': Boolean, 'standard_search': Boolean, 'own_interface': Boolean, 'submit_link':url, 'interface': HTML or list}

The meaning of the different keys is explained below:

- 'search_on_click': Boolean that tells SynBioHub whether to search on selection of the plugin or whether parameters are needed. If this is True, then all the other keys can be disregarded and can pass on to the run endpoint
- 'standard_search': Boolean that tells SynBioHub whether the search bar will be used as input. If True, the search bar is used as input, otherwise the interface returned will be used to gather parameters.
- 'own_interface': Boolean to tell SynBioHub what kind of interface to expect back. If True, than HTML will be sent as the interface if False, than a list object indicating an HTML form will be sent back.
- 'submit_link': The URL for SynBioHub to use if creating a form from the single use object. It can be the same as the link SynBioHub sent or to another plugin endpoint. Is not used if the own_interface variable is set to True.
- 'interface': An HTML object or list to allow parameter entry. If a list, it has an entry for every variable of the HTML form to be created. Each entry is a dictionary with the keys:
    - type: string, HTML-form input type
    - label: string, heading to give above the input entry
    - description: string, description/help text to provide to the user
    - options: list, list of options, empty list unless the input is a radio or checkbox
    - default: list, list providing the default selection
    - restrictions: dictionary, dictionary providing any further options that can be provided to HTML forms, e.g. pattern is a key for the regex expression used to check the input

results visualization or table to be visualized by SynBioHub (Algorithm 3.10). It also returns a list of all URIs returned by the search, any columns, and facets to allow for filtering. The two kinds of data that are allowable ensure search plugins can be used in many different kinds of search. The list of URIs with facets allows visualization of the search results in a tabular manner similar to existing biological databases like UniProt. Additionally, the facet format is designed to integrate with the popular Elasticsearch tool. Whilst this is not an official standard, it is a *de facto* standard. The option to return HTML instead of URIs enables custom search result visualizations. This allows search results to be visualized as networks, or even interactive graphical elements. Whilst there may not be many use cases for this yet, allowing HTML results display provides search plugins with the flexibility to be used for scenarios that have not yet been encountered.

3.3.3 Index

Index plugins are more theoretical than the two other plugins mentioned so far. They are designed to provide offline indexing that can be queried later. They can provide additional information that may be migrated to the SynBioHub central data store at a later stage. This functionality may be used to cache difficult to calculate parameters (like the GC content of a sequence) or to store parameters that do not yet have an SBOL best practice associated with them (like Source Organism). Additionally, this kind of plugin might be adapted to do large scale curation on an offline basis overnight. An initial example might be the maintenance of the page-rank index.

The plugin API has not yet been developed. However, an initial idea of how the plugin might work is described. SynBioHub sends a list of URIs to the plugin and the plugin creates an index of URIs mapped to columns. These columns might be empty (in which case the plugin works as a filter) or might contain information. The information does not have to be SBOL encoded. This means information can be stored in a temporary format before incorporation into the SBOL standard, if the column proves to be popular. The plugin returns the index to SynBioHub for storage. The plugin should be able to incrementally update an index if the index is returned, or perhaps if just a subset of URIs are provided.

Algorithm 3.10: Search Plugin API: RUN

RUN Endpoint

SynBioHub: POST Request

{‘instanceUrl’: URL, ‘sparql_link’: URL, ‘es_link’: URL, ‘search_parameters’: dictionary }

The meaning of the different keys is explained below:

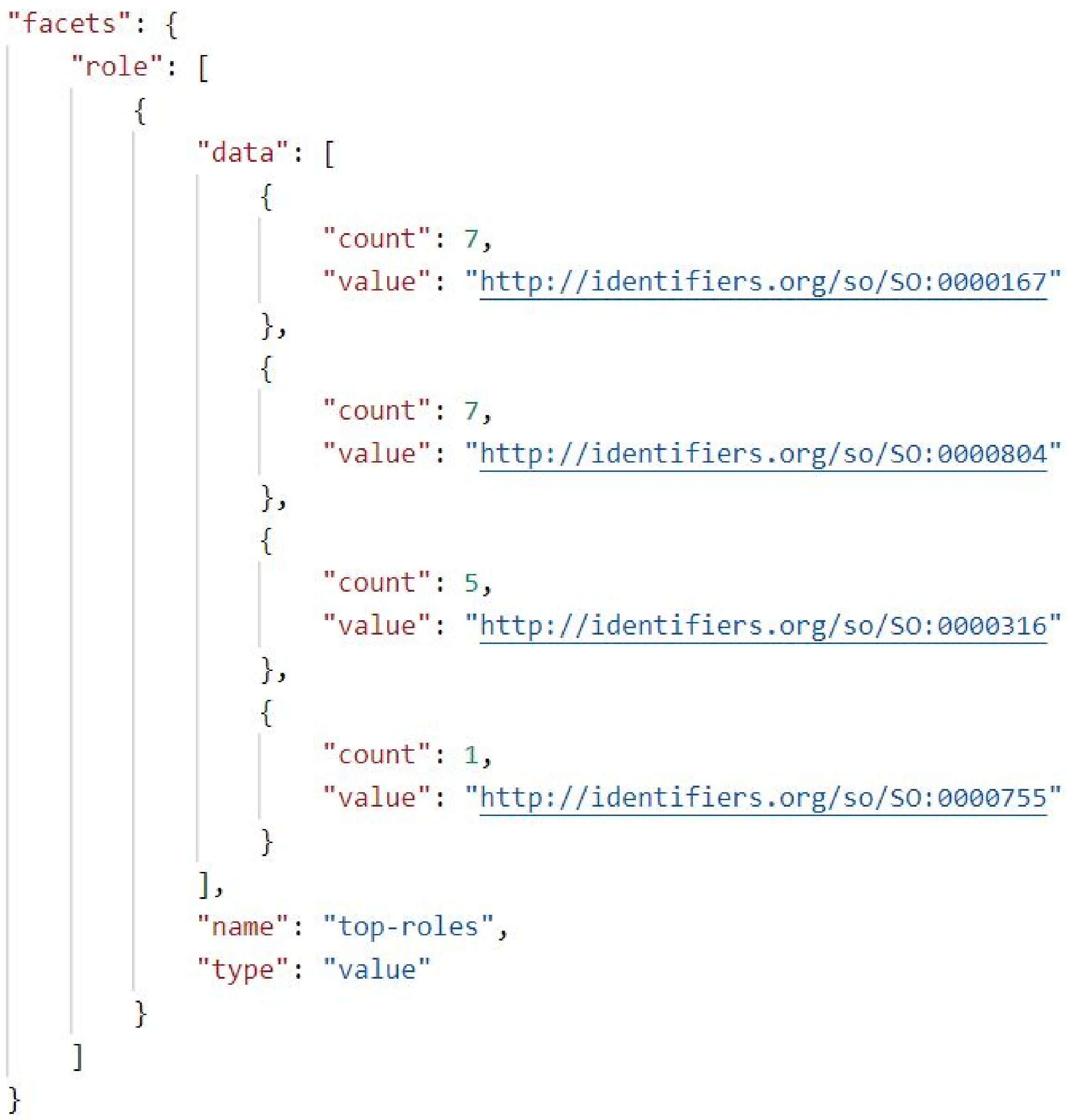
- ‘instanceUrl’: The top-level URL of the SynBioHub instance
- ‘submit_link’: the SynBioHub link to which the HTML form should submit
- ‘sparql_link’: URL at which the sparql endpoint may be accessed, could build in authorization if desired
- ‘es_link’: URL at which the elastic search endpoint may be accessed, could build in authorization if desired (this endpoint allows the plugin to do fuzzy string matching).
- ‘search_parameters’: A dictionary providing the output values from the evaluate form. Is empty if there is no output. Otherwise of the form {‘variable 1’: [‘text input’], ‘variable 2’: [‘option 1’], ‘variable 3’: [‘option 1’, ‘option 2’]}

Plugin: Response Data

{‘table_output’: list, ‘uris’: list, ‘facets’: dictionary, ‘columns’: list, ‘HTML’: string}

The meaning of the different keys is explained below:

- ‘table_output’: List with a dictionary for each row in the table. The dictionary has column names as keys and cell values as the value. One of the columns is ‘uri’ and contains the uri of the object. If using own output, then this will be an empty dictionary
- ‘uris’: list of all uris returned by the search. If using own search results display then this list will be empty
- ‘columns’: list of a columns in ‘table_output’. If using own search results display, then this list will be empty
- ‘HTML’: may be a blank string if the standard table output is being used. Otherwise, this is the HTML displaying the search results output.
- ‘facets’: Dictionary of facets based on the elastic search idea of facets. This summarizes the data and provides a way of filtering to use on the next query. The dictionary has a key for every column that is being filtered. The value is a list containing all the facets. Each facet is represented by a dictionary containing the keys type (value or range), name (name to be displayed), and data (list of dictionaries each containing the value (name) and the count (number)). For an example see Figure 3.8.



```
"facets": {
    "role": [
        {
            "data": [
                {
                    "count": 7,
                    "value": "http://identifiers.org/so/SO:0000167"
                },
                {
                    "count": 7,
                    "value": "http://identifiers.org/so/SO:0000804"
                },
                {
                    "count": 5,
                    "value": "http://identifiers.org/so/SO:0000316"
                },
                {
                    "count": 1,
                    "value": "http://identifiers.org/so/SO:0000755"
                }
            ],
            "name": "top-roles",
            "type": "value"
        }
    ]
}
```

Figure 3.8: Example of the facet dictionary returned for the role column. This complies with the ElasticSearch standard. The roles listed use the sequence ontology URLs.

3.3.4 Link

Link plugins are designed to allow SynBioHub data to be seamlessly transferred between SBOL tools. For example, visiting a design page and opening it in SBOLCanvas to make edits after which the user returns to SynBioHub again where the new edited version is present. This could in theory be done using a visualisation plugin, however this poses problems with sharing authorization. The visual plugin could display a button which when clicked sends a request and redirects to a new page (e.g. SBOLCanvas). However, how the user log-in info is transferred is difficult. Additionally, it means redundant code will have to be written to update any SBOL Objects sent to the 'link application'.

Link plugins do not yet have a full API as some of the decisions about the API implementation are still being considered. The proposed options, their benefits, and their drawbacks are discussed here. The 'link application' should receive authorization (if necessary), an instance URL from SynBioHub, and a submission link where altered SBOL can be sent back to SynBioHub for validation. However, whether the request can take in additional parameters from the plugin is still being considered. The benefit would be allowing requests to applications that need additional parameters and to which the developer has little access other than the API (e.g. Benchling). However, adding this would make creating the plugin more complex. Additionally, the plugin should have a status and evaluate endpoint. The status endpoint can be the standard endpoint seen in all plugins. The evaluate endpoint would be used to decide if the plugin shows up on SBOL object pages. The evaluate endpoint should receive the RDF-type. It might also receive the SBOL links. This would be useful as an application might be unable to process SBOL objects without particular triples (e.g. attachment of an archive file). The evaluate endpoint could directly return all the information needed by SynBioHub, but there could also be a separate run endpoint. The benefit of a separate endpoint is to decrease the complexity of the evaluate endpoint and make the different endpoints more modular. The downside is if the evaluate endpoint is simply adding an additional API call to a run endpoint that might be needlessly time consuming. As discussions continue, an initial repository has been set up to start testing different configurations of the endpoints. The repository can be found at: `https://github.com/SynBioHub/Plugin-Link-Test`.

3.4 Example Plugins

Table 3.1 is a table of all plugins created to date. The column "External" indicates whether the plugin was developed by the author or other developers. Note the number of plugins developed by others. This highlights the ability of others to use the plugin framework and the interest in doing so.

3.5 Further Work

There is still further work to be done on plugins. First and foremost, the proposed new plugins must be further developed and incorporated into the SynBioHub interface. They could be incorporated into SynBioHub3, which is currently under active development.

Additionally, plugins are currently not accessible via the SynBioHub API. For example, a submission via the API does not allow submit plugins to be used. Making plugins accessible via the API is an important step. This is particularly important as SynBioHub3 separates front and back-end code and has the front end call the back-end via the API. Additionally, the new front end is developed to allow multiple file downloads from a 'cart' option. Enabling multi-file calls for download plugins (either via multiple API calls or by updating download plugins to allow multiple downloads at once) should be considered.

Finally, the creation of a plugin registry/marketplace where plugins are listed in a format similar to Table 3.1 is a next step. Having a registry aware of all running plugins would allow users to more easily add plugins to their SynBioHub instances. This would increase the ease of plugin use. It would work something like a plugin App store or a browser extension manager.

Table 3.1: A table summarising all plugins currently available. Note the number of plugins developed by External developers (i.e. not this author).

Name	**External**	**Language**	**Test**	**Description**
CurationSynbict	No	Python	No	Runs SYNBICT to add component annotations to component definitions
DownloadSBOL2Excel	No	Python	No	Downloads an SBOL file into an excel template
DownloadSnapgene	No	Python	No	Returns one of: genbank file, snapgene visualization, zip of both. Either of the component or the component after annotation with snapgene
SubmitExcel2SBOL	No	Python	No	Turns an excel template submission into SBOL
SubmitSnapgene	No	Python	No	Takes in a snapgene file and converts it to sbol
VisualComponentUse	No	Python	No	Shows a co-use component sankey diagram, and the most used components bar graph endpoints
VisualIgem	No	TypeScript	No	The iGEM Main Page, iGEM Design Page, and iGEM Experience Page
CurationTest	No	Python	Yes	Allows developers to play with the curation plugins and provides a simple example to check that they are up and running
DownloadTest	No	Python	Yes	Indicates that download plugins are working and provides a framework to play with for plugin developers
DownloadTestEval	No	Python	Yes	Designed to show what parameters the evaluate endpoint recieves. Was a way of testing and developing
DownloadTestJs	No	JavaScript	Yes	Indicates that download plugins are working and provides a framework to play with for plugin developers
IndexTest	No	Python	Yes	Allows developers to play with the index plugins and provides a simple example to check that they are up and running
LinkTest	No	Python	Yes	Allows developers to play with the link plugins and provides a simple example to check that they are up and running
SearchTest	No	Python	Yes	Allows developers to play with the search plugins and provides a simple example to check that they are up and running
SubmitTest	No	Python	Yes	Indicates that submit plugins are working and provides a framework to play with for plugin developers
SubmitTestJs	No	Python	Yes	Indicates that submit plugins are working and provides a framework to play with for plugin developers
VisualServe	No	Python	Yes	Allows testing of file serving and provides a framework to play with for plugin developers
VisualServelet	No	JavaScript	Yes	Allows testing of file serving and provides a framework to play with for plugin developers
VisualTest	No	Python	Yes	Indicates that visualisation plugins are working and provides a framework to play with for plugin developers
VisualTestJS	No	JavaScript	Yes	Indicates that submit plugins are working and provides a framework to play with for plugin developers
DownloadiBioSim	Yes	Python	No	Runs an iBioSim simultaion based on a COMBINE archive
DownloadShortbol	Yes	Python	No	Downloads a shortbol file
SubmitExcelComposition	Yes	Python	No	Allows submissions of excel composite templates
SubmitExcelLibrary	Yes	Python	No	Allows submissions of excel library templates
SubmitShortbol	Yes	Python	No	Turns a shortbol file into SBOL
VisualFlapjack	Yes	Python	No	Visualizes measurement plots for experimental data with a Flapjack id
VisualProteinStructure	Yes	Python	No	Visualizes protein structures
VisualSeqviz	Yes	JavaScript	No	Shows the plasmid view and sequence view of components
VisualVisBOLJs	Yes	JavaScript	No	Creates a VisBOL image

Chapter 4

Excel-SBOL Converter

Shared representations for data and metadata, grounded in well-defined ontology terms, can help make it easier to share materials between biologists [232]. If sharing is made easier, then parts can be reused more. However, using formal representations, such as SBOL, typically requires either a thorough understanding of these standards or a suite of tools developed in concurrence with the ontologies [127]. Unfortunately, this poses a significant barrier to use for scientists not trained to work with such abstractions. One approach to lowering the barrier to the use of ontologies, was demonstrated in the Systems Biology for Micro-Organisms (SysMO) consortium [29]. In SysMO, the MicroArray Gene Expression Markup Language (Mage-ML) was set up as an XML schema [199], and users were expected to submit data to the SysMO Assets Catalogue (called SEEK) in XML format in order to publish work. To allow the use of the Mage-ML language without having to understand XML, the RightField tool was created [234], an ontology annotation and information management application that can add constrained ontology term selection to Excel spreadsheets. This tool enables administrators to create templates with controlled vocabularies, such that the scientists utilizing the tool would never actually see the raw RightField, only the more familiar Excel spreadsheet interface. Spreadsheets are a popular interface as many biological workflows already use spreadsheets and comma seperated values. Furthermore, several popular tools use spreadsheets and CSV files as inputs or outputs, including Addgene (`https://www.addgene.org/`), and opentrons (`https://opentrons.com/`).

Similarly, users of SBOL and SynBioHub have faced a steep learning curve for understanding the underlying ontology: as assessed in [217], "For successful use and interpretation of metadata presented in SynBioHub, the semantic annotation process should be biologist-friendly and hide the underlying RDF predicates." Recently, SynBio2Easy was published as a command-line tool to convert Excel spreadsheets of plasmids to SBOL [246]. Similarly, the Excel-SBOL Converter presented here has been designed to provide a simple way for users to generate and visualize SBOL data without needing a detailed understanding of the underlying ontology and associated technologies. Unlike SynBio2Easy, the converter focuses on multiple kinds of SBOL data and allows customization of the templates. The converter provides a simple way for users to manage data by allowing users to download SBOL into Excel templates and submit Excel templates for conversion into SBOL (Figure 4.1). Additionally, it was tested and expanded based on two case studies: experimental data [241], and the existing Cello genetic part library [158].

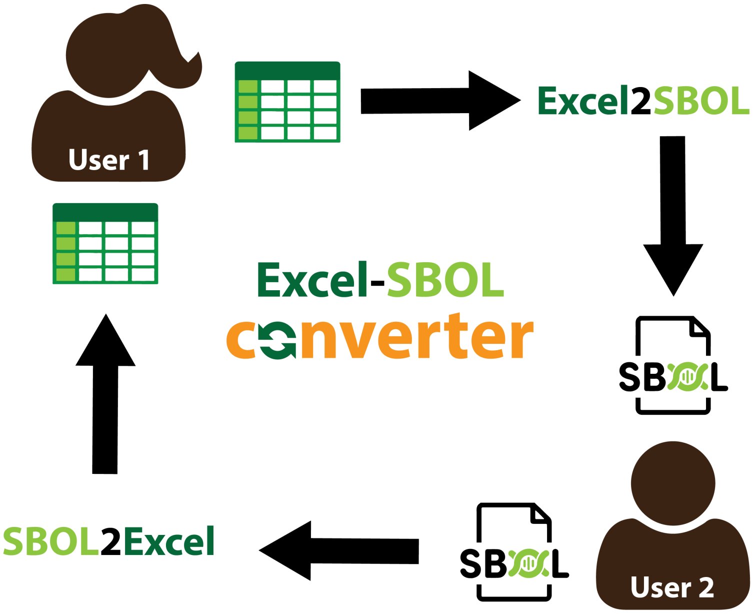


Figure 4.1: The Excel-SBOL Converter consists of two libraries: 1) to allow the use of Excel Templates to create SBOL data, and 2) to allow SBOL data to be downloaded as Excel Spreadsheets. Using the two libraries together allows the creation, uploading, downloading, editing, and re-uploading of data. Additionally, it allows collaboration between users who prefer spreadsheets and those who prefer SBOL.

4.1 Excel-to-SBOL Evolution

The Excel-to-SBOL converter evolved via several iterations as the scale and capability of the converter expanded. Over time, the imagined workflow pipeline changed as depicted in Figure 4.2. To allow more flexibility in the types of SBOL data that could be created, the templates became more complex and general. This means that to design a template, knowledge of the SBOL data structure is now required. However, it should still be possible to hide the complexities of the template from the novice SBOL user. To justify the design decisions, the sections below cover the different iterations, benefits, and limitations that led to the next version of the converter.

4.1.1 Iteration 1: Fixed Column Templates

The original library was created based on Excel templates used by the DARPA SD2 project (Defense Advanced Research Projects Agency Synergistic Discovery and Design (`https://sd2e.org/`)). There was a library sheet (See Figure 4.3) for the creation of single component parts, and a composite sheet (Figure 4.4) for the creation of components from sub-components. Due to the very different sheet layouts and types of SBOL they contained, two separate code libraries were required, one for processing the part library sheet and one for processing the composite library sheet. Each is described below.

4.1.1.1 Part Library

The part library sheet is processed using Algorithm 4.1. Note the full code can be found in the archived github repository: `https://github.com/SynBioHub/Plugin-Submit-Excel-Library`.

The code was very fixed to the template. There was some flexibility in column value extraction as they are extracted based on column names, which are a variable (i.e., not hard coded). However, the default names are the function defaults and a user has to go in and edit the code should they wish to update the names (there is no smart way of deciding how to process the columns). Additionally, all of these columns must be present and no additional columns are possible.

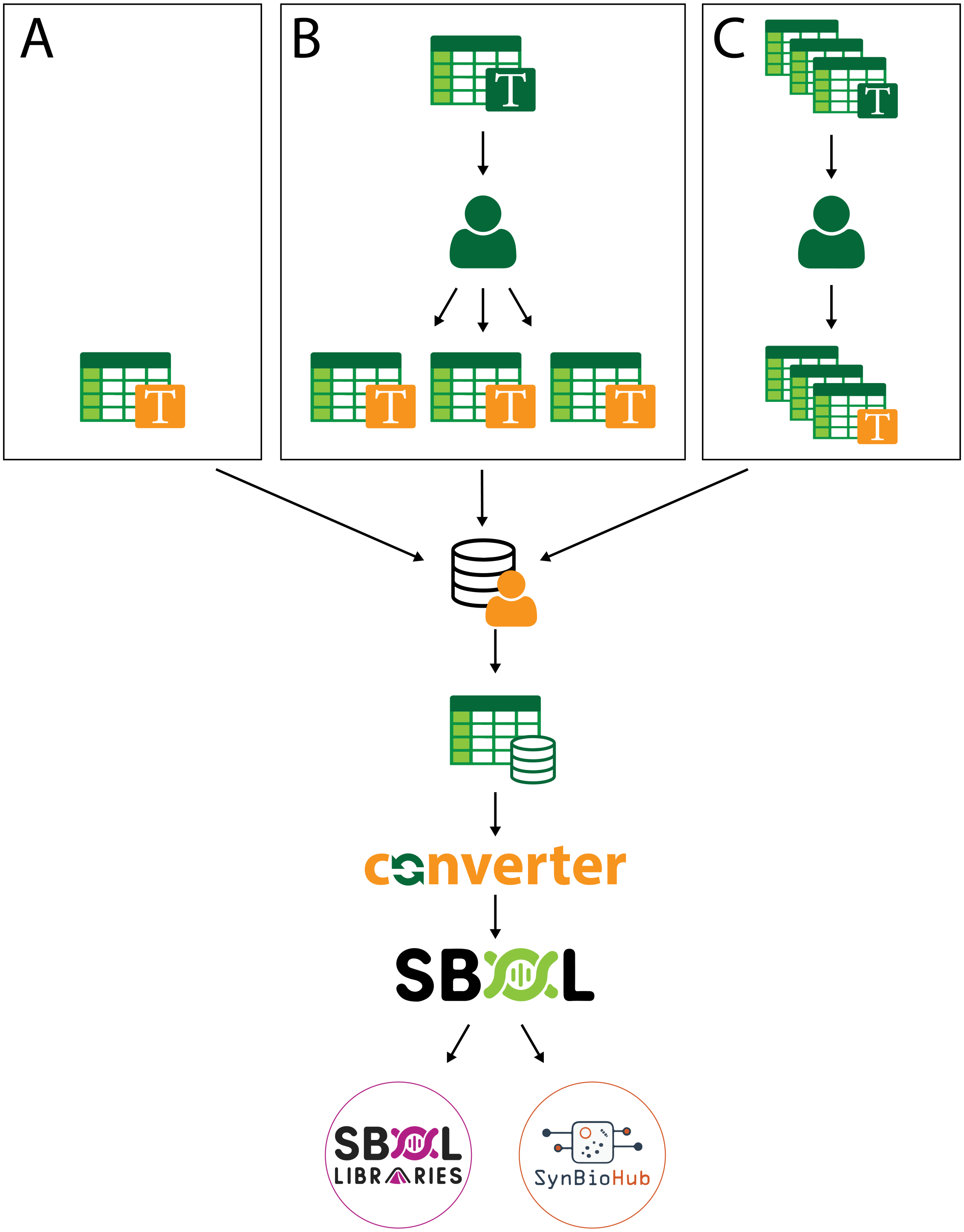
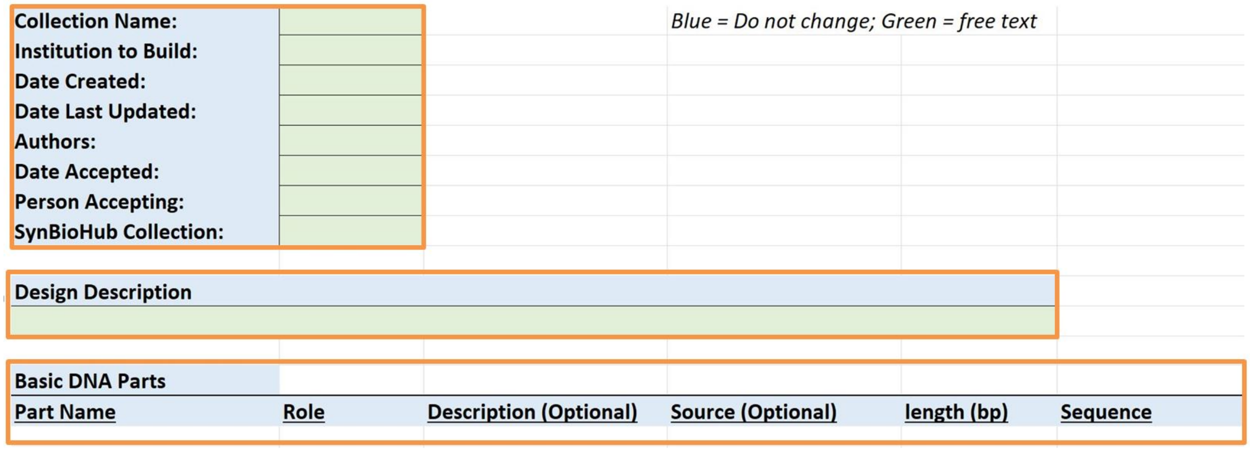


Figure 4.2: Evolution of the Excel-to-SBOL pipeline over time. In all cases once the templates are created, they are filled in by the user. These templates are then converted to SBOL using the Converter. This SBOL can be uploaded to SynBioHub or used in other tools. **A**: Initially a single rigid template was used for data collection. **B**: The converter was designed to be more flexible in the conversion of different templates. It became possible for someone familiar with the SBOL specification to design new templates for novice users. **C**: Templates now allow multi-sheet interactivity whilst maintaining the flexibility created in pipeline B.



Blue = Do not change; Green = free text

Collection Name:	
Institution to Build:	
Date Created:	
Date Last Updated:	
Authors:	
Date Accepted:	
Person Accepting:	
SynBioHub Collection:	

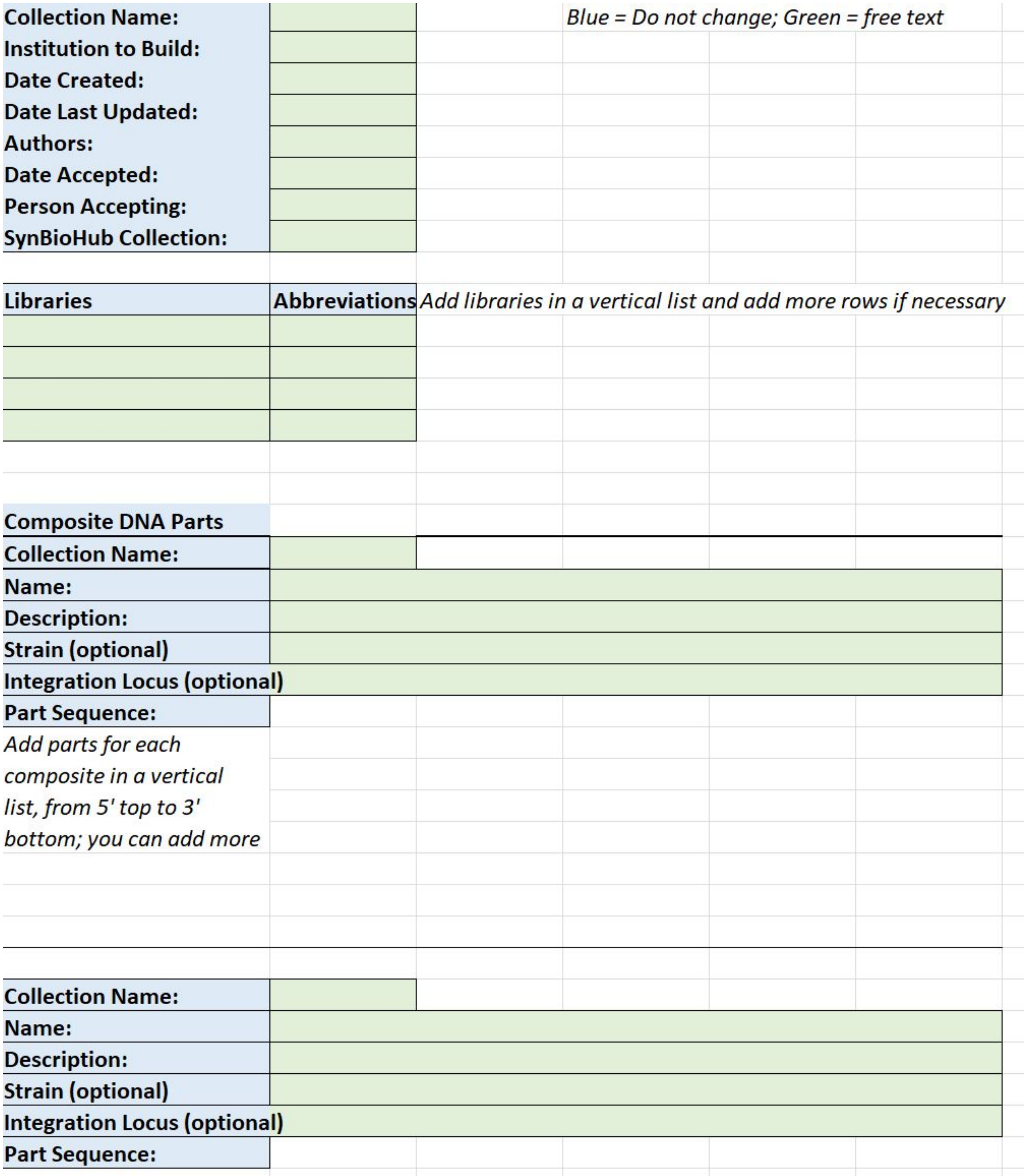
Design Description

Basic DNA Parts

Part Name	**Role**	**Description (Optional)**	**Source (Optional)**	**length (bp)**	**Sequence**

Figure 4.3: SD2 Library Template. This is the template for the import of single components. Note the three sections: collection information at the top, the design description, and the headers for the part information.

```
Algorithm 4.1: Excel-to-SBOL Iteration 1: Part Library Processing
  Input: Excel Library Sheet Filled, Excel Library Sheet Empty
  Output: SBOL File
  initialization
  if blank template structure = filled template structure then
      SBOL Document Name ← Metadata Value: Collection Name
      SBOL Document Description ← Metadata Value: Description
      for row in library do
          Create ComponentDefinition
          ComponentDefinition role ← value in role column
          ComponentDefinition name ← value in name column
          ComponenDefinitiont description ← value in description column
          Sequence Object Created based on value in sequence column, with spaces removed
          ComponentDefinition sequence references the newly created Sequence Object
```



Collection Name:		*Blue = Do not change; Green = free text*
Institution to Build:		
Date Created:		
Date Last Updated:		
Authors:		
Date Accepted:		
Person Accepting:		
SynBioHub Collection:		

Libraries	Abbreviations	*Add libraries in a vertical list and add more rows if necessary*

Composite DNA Parts	
Collection Name:	
Name:	
Description:	
Strain (optional)	
Integration Locus (optional)	
Part Sequence:	

Add parts for each composite in a vertical list, from 5' top to 3' bottom; you can add more

Collection Name:	
Name:	
Description:	
Strain (optional)	
Integration Locus (optional)	
Part Sequence:	

Figure 4.4: SD2 Composite Template. Note the many separate tables for each collection of composites that are created.

4.1.1.2 Composite Library

The algorithm for processing composite libraries is shown in Algorithm 4.2. Note the full code can be found in the archived github repository: `https://github.com/SynBioHub/Plugin-Submit-Excel-Composition`. This code allows for the creation of composite parts. Additionally, the (reversible) conversion function to SBOL compliant collection names can be used in other contexts too. Unfortunately, a lot of code is necessary to read in the part libraries that are already being read in to convert the simple part library. Additionally, the code is even more complex because of the mutli-table input style of the composites sheet. Finally, the composite sheets do not allow any custom columns to be input by the user or any flexibility in the object type that is being created.

Algorithm 4.2: Excel-to-SBOL Iteration 1: Composite Library Processing

```
Input: Excel Library Sheet Filled, Excel Library Sheet Empty
Output: SBOL File
initialization
if blank template structure = filled template structure then
    collection start list ← list of all rows containing ‘Collection Name:’ and the number of
      filled columns in the ‘Part Sequence:’ row
    Extract Parts from Sheet (collection start list):
        for collection in collection start list do
            for column in collection do
                Add collecion name: part name: part list to Part Dictionary
                Add parts list to Part List
    Check Part Names are SBOL compliant
    Create SBOL (Part List, Part Dictionary):
        Pull all part definitions from their urls
        for collection in Part Dictionary do
            Create Collection
            for Part Name in collection do
                Create ComponentDefinition
                Assemble Composite ComponentDefinition
                Add Component Description
                Add ComponentDefinition to Collection
```

4.1.2 Iteration 2: Flexible Column Templates

Based on the limitations of the first version, a new version of the template and reader was developed. The aims were: a single library to process library and composite templates, reduced code redundancy, more machine friendly template layouts, and more flexibility in the kinds of SBOL objects that could be encoded in the sheet. To achieve these aims, a single template type, like the library template used in the DARPA SD2 project, was used. This was supplemented by a column definitions sheet. The column definitions sheet provides two sets of information about each column: whether the value in the column should be converted to a machine readable version, and if yes then how, and the SBOL encoding the converter should use for the column. This allows a template creator to add any properties they might want (SBOL or otherwise), name the columns in a human readable way in the sheet, and only provide a machine readable version behind the scenes in the column definitions sheet. A more complete description of the columns sheet is given below. Hypothetically, one of the property columns could be the object type which would then be used to create different SBOL object types. However, the next version was created before this was implemented. The full code can be found at the github repository merge instance found here: `https://github.com/SynBioDex/Excel-to-SBOL/tree/787e088a7c916bb69c6b167932c3219674075d8d`. The code works as outlined in Algorithm 4.3.

4.1.2.1 Column Definitions Sheet

The column definitions sheet is used for two things: conversion of cell values to a machine readable format, and conversion of cell values to SBOL. The conversion to a machine readable format happens first. This is based on the “Sheet Lookup” and “Replacement Lookup” Columns (as shown in Figure 4.5). Sheet lookup takes the cell value and converts it to another value like a lookup dictionary. For example, promoter to `http://identifiers.org/so/SO:0000167`. This is done by creating a conversion dictionary from a sheet and going from the value in one column

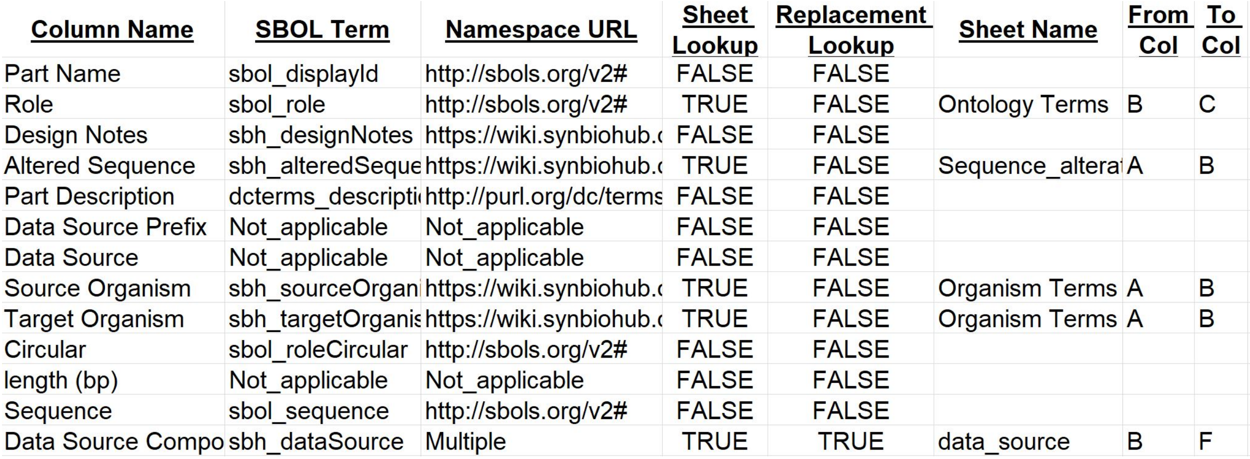
Algorithm 4.3: Excel-to-SBOL Iteration 2: Flexible Column Templates

```
Input: Excel Filled Template
Output: SBOL File
Initialization
Use metadata on Library Sheet to Set SBOL Document Properties
for row in library do
    Create ComponentDefinition
    for column in row do
        Convert cell value using column definitions sheet Ontologies (may use sheet lookup or
          replacement lookup)
        Pull SBOL Term from column definitions sheet
        switch SBOL Term do
            case Not Applicable do
                continue
            case Altered Sequence do
                ComponentDefinition was Generated by ← cell value
            case Data Source do
                ComponentDefinition was Derived From ← cell value
                if 'PubMed' in cell value then
                    ComponentDefinition PubMed Property ← cell value
            case Source Organism do
                Define Source Organism Property
                ComponentDefinition Source Organism ← cell value
            case Target Organism do
                Define Target Organism Property
                ComponentDefinition Target Organism ← cell value
            case Role do
                if ComponentDefinition has role(s) then
                    Add cell value to ComponentDefinition roles
                else
                    ComponentDefinition Role ← cell value
            case Role Circular do
                if cell value is True and ComponentDefinition has role(s) then
                    Add Circular to ComponentDefinition roles
                else if cell value is True then
                    ComponentDefinition Role ← Circular
            case Display ID do
                ComponentDefinition Display ID ← cell value
            case Sequence do
                Remove spaces from cell value and make it lowercase
                Create SBOL Sequence Object
                Sequence Name ← ComponentDefinition Name
                Add Sequence to Document
                Link Sequence to ComponentDefinition
            otherwise do
                Add cell value to ComponentDefinition as a text property using the name
                  space provided in column definitions sheet
    Add ComponentDefinition to SBOL Document
```

to the value in another column. It provides a simple and versatile way for human readable values to be converted to ontology values. It can be used for true ontologies and for any other kind of conversion. For example, TRUE can convert to a specific value or a nickname can be used to call a full name. The downside is it does require the full list of value and conversion pairs to be listed. For values that are being pulled from ontologies this can be a long list which is mostly not used. Additionally, it does not allow the conversion of IDs to URLs.

The replacement lookup is a special case of sheet lookup. In this case, the cell value is expected to be prefix:value, e.g. PMID:24295448, representing the data source is a PubMed id with the value 24295448. Here the lookup works by using the prefix as the key and pulls a value from the "To Column" and inserts the cell suffix in the place of {REPLACE_HERE}. So for the case PMID:24295448, the PMID key gives the string `https://pubmed.ncbi.nlm.nih.gov/{REPLACE_HERE}`, which is changed to `https://pubmed.ncbi.nlm.nih.gov/24295448`. This kind of lookup is useful in the case where several different kinds of information are put in one column, e.g. data source might be GenBank, PubMed, AddGene, etc. Additionally, it allows the formatting of a URL rather than being a direct 'translation' in the way a simple sheet look up is.

After the cell value conversion to a machine readable format, the column definitions sheet is used to convert cell values to SBOL. This is done using the "SBOL Term" and "Namespace URL" column. If the SBOL term is a defined case in the switch method, then it is implemented via the code in the case statement (see the switch statement in Algorithm 4.3). For example, sbh_sourceOrganism creates the SBH namespace and the URI property sourceOrganism, and then sets the property equal to the value: `https://identifiers.org/taxonomy:{cell_value}`. For sbol_role, after checking the type, the SBOL2 ComponentDefinition property role is used to add the information. If the SBOL Term is not a defined case, then a fall back method is called that adds the namespace to the document, creates the text attribute sbol_suffix, and sets the attributes value equal to the cell value. For example, sbh_designNotes leads to the adding of the sbh namespace using the value found in the Namespace URL column. Then, the attribute designNotes is added to the ComponentDefinition object, and it is set equal to the cell value. This allows new properties to be defined by users whilst still having standard processing methods for commonly used properties.



Column Name	SBOL Term	Namespace URL	Sheet Lookup	Replacement Lookup	Sheet Name	From Col	To Col
Part Name	sbol_displayId	http://sbols.org/v2#	FALSE	FALSE			
Role	sbol_role	http://sbols.org/v2#	TRUE	FALSE	Ontology Terms	B	C
Design Notes	sbh_designNotes	https://wiki.synbiohub.o	FALSE	FALSE			
Altered Sequence	sbh_alteredSeque	https://wiki.synbiohub.o	TRUE	FALSE	Sequence_altera	A	B
Part Description	dcterms_descripti	http://purl.org/dc/terms	FALSE	FALSE			
Data Source Prefix	Not_applicable	Not_applicable	FALSE	FALSE			
Data Source	Not_applicable	Not_applicable	FALSE	FALSE			
Source Organism	sbh_sourceOrgani	https://wiki.synbiohub.o	TRUE	FALSE	Organism Terms	A	B
Target Organism	sbh_targetOrganis	https://wiki.synbiohub.o	TRUE	FALSE	Organism Terms	A	B
Circular	sbol_roleCircular	http://sbols.org/v2#	FALSE	FALSE			
length (bp)	Not_applicable	Not_applicable	FALSE	FALSE			
Sequence	sbol_sequence	http://sbols.org/v2#	FALSE	FALSE			
Data Source Compo	sbh_dataSource	Multiple	TRUE	TRUE	data_source	B	F

Figure 4.5: Example of the Column Definitions Sheet used in the Flexible Column Templates. The names of the columns found on the library sheet are seen in the Column titled "Column Name". The SBOL Term column is used together with the Namespace URL to determine how cell values will be encoded in SBOL. The final five columns are used to convert cell values to more machine readable formats via the use of ontologies.

4.1.2.2 Additional features

Apart from the addition of the column definitions sheet, the GitHub repository also underwent improvements. GitHub actions were added to automatically push a new Python library to PyPi whenever a new release is created on GitHub. This makes it easier to automatically role out new versions of the library when changes are made. Additionally, a unit test library was created using pytest (`https://docs.pytest.org/en/7.0.x/#`). This library is automatically run on pull requests together with flake8 linting (`https://flake8.pycqa.org/en/latest/`). The pytests ensure that when edits are made, the original functionality is still retained. The linting ensures that standards of the Python language are maintained, which makes it easier for others to work on the existing code base.

4.1.2.3 Weaknesses

Despite the additional flexibility and reduction of code redundancy in this version, there are still weaknesses. Notably, the SBOL object type is hard coded rather than being read in from the sheet so only ComponentDefinitions can be created. Furthermore, most SBOL properties that are added require the definition of a new case for the converter. Additionally, every library sheet must be processed individually, and only one library sheet can be processed per workbook. This means that no objects can be created referencing objects in another sheet or workbook, and that the ontology sheets are duplicated between workbooks. The latter is particularly inefficient as all ontology sheets have to contain the entire ontology and every ontology gets its own sheet. Each of these weaknesses reflect a lack of flexibility in the code.

There are also a second set of issues related to the way single columns are processed. As every column must be processed on its own, a property which has multiple values has to be combined to a single column. Such a combined column must then be split in the code and processed using a special case property function. Setting up a standard splitting procedure for columns would decrease the special case code. Automatic splitting would also make it easier to allow checking of cell inputs via template specified regular expressions. Allowing regular expression based validation creates an additional checking mechanism for the more flexible code.

A final issue is that the sheet metadata (top left table in Figure 4.3) is required to be filled in, but it is not used by SynBioHub, and is not always relevant.

4.1.3 Iteration 3: Multi-Library Templates

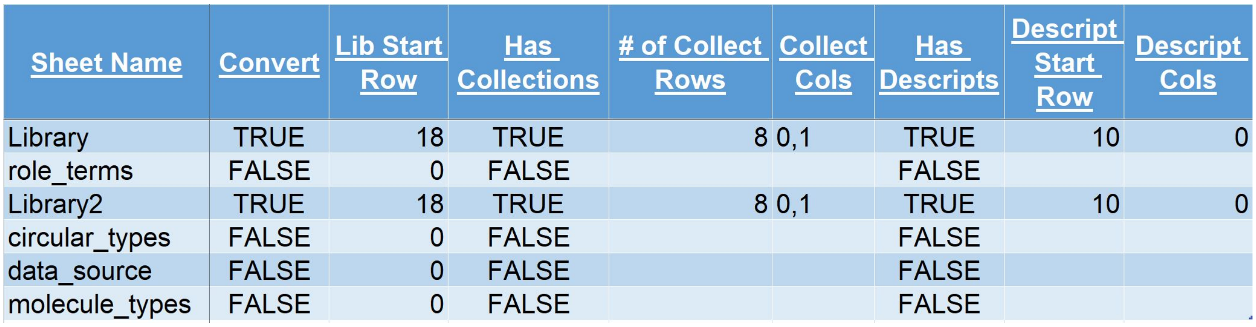
Based on the weaknesses of the flexible column templates, the next version was developed: multi-library templates. The main innovations are the ‘Init’, or initialize, sheet that allows multiple sheets to be read and the multiple scanning of sheets that allows SBOL objects to be referenced across sheets. The new ‘Init’ sheet and updated column definitions sheet are explained further below, after an overview of how the processing works now. The full code can be found at `https:`

//github.com/SynBioDex/Excel-to-SBOL. The algorithm used is shown in Algorithm 4.4.

4.1.3.1 Init Sheet

The "Init" sheet is a list of all sheets to be processed (Figure 4.6). The sheet name is given in the first column and the kind of conversion in the second column (if True it contains SBOL objects, if false it is only to be used to convert cell values). The next columns indicate how the structure of the sheet works. These columns indicate whether each of the three elements found in the initial DARPA SD2 template (Figure 4.3) exists. The library section of SBOL objects or conversion is given a start row. Then, the collection metadata is said to be present or absent. If present, the number of rows are specified and which columns it can be found in. Finally, the collection description is present or absent, and if present, the start row and column are given.

The use of the Init sheet in this way means multiple sheets can be converted and none of the sheets needs to have redundant collection or description information. It allows for templates where each sheet takes a different kind of SBOL object, and for referencing of SBOL objects between sheets without worrying about the order of conversion of the objects.



Sheet Name	Convert	Lib Start Row	Has Collections	# of Collect Rows	Collect Cols	Has Descripts	Descript Start Row	Descript Cols
Library	TRUE	18	TRUE	8	0,1	TRUE	10	0
role_terms	FALSE	0	FALSE			FALSE		
Library2	TRUE	18	TRUE	8	0,1	TRUE	10	0
circular_types	FALSE	0	FALSE			FALSE		
data_source	FALSE	0	FALSE			FALSE		
molecule_types	FALSE	0	FALSE			FALSE		

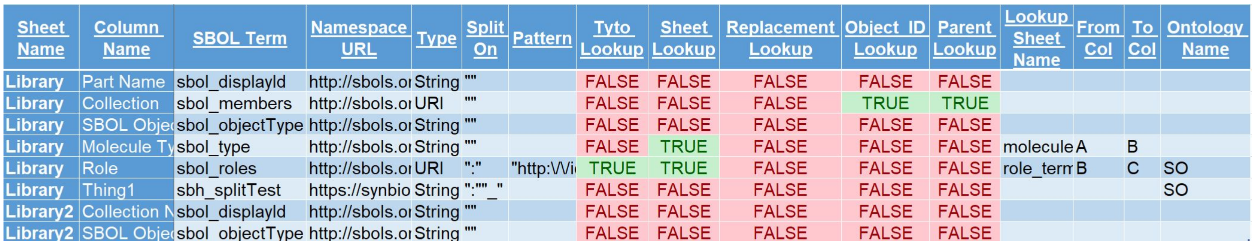
Figure 4.6: Example of the initialization sheet used in the multi-sheet templates. The left most column contains the names of the sheets to be processed. Next the 'Convert' column indicates whether the sheet contains SBOL objects (contains SBOL objects if TRUE). The 'Lib Start Row' column indicates in which row the information on the sheet starts (note this is zero-indexed). The final columns are about the elements of sheet, is there metadata and/or a description field, and if so where?

Algorithm 4.4: Excel-to-SBOL Iteration 3: Multi-Library Templates

```
Input: Excel Filled Template
Output: SBOL File
Initialization
    Read in Init Sheet and create a list of sheets with SBOL objects in them
    Read in all Sheets based on Init Sheet
    Read in Column Definitions Sheet
Parse Objects
    Create SBOL Document
    for sheet in SBOL Object Sheets do
        Find Display ID Column and SBOL Object Type Column
        for SBOL Object in sheet do
            Create SBOL Object
            Save SBOL Object in a dictionary common name:URI
Parse Columns
    for sheet in SBOL Object Sheets do
        for Row in sheet do
            for Column in Row do
                Split the cell value based on Column Definitions Sheet split column
                Convert cell value using Ontology Lookups found in Column Definitions Sheet
                Check that the converted cell value match the pattern specified in the
                  Column Definitions Sheet
                Convert the cell value based on SBOL Term, Namespace URL and Parental
                  Lookup values from the Column Definitions Sheet
```

4.1.3.2 Column Definitions Sheet

The column definitions sheet is similar to the previous version, but it is extended (Figure 4.7). There are now roughly five sections of the sheet: the name, the SBOL Type, splitting information, pattern information, and lookup information. The name now requires two columns: the sheet name and the column name. This means column names do not have to be unique and can be shared across different sheets and processed in different ways. The SBOL Type has three columns: “SBOL Term”, “Namespace URL”, and “Type”. The addition of the type column indicates if a new property should be initiated as a text or URI property. The addition of this column is part of what allows this version to be more flexible in the basic code and require fewer special cases to be written out for column processing.



Sheet Name	Column Name	SBOL Term	Namespace URL	Type	Split On	Pattern	Tyto Lookup	Sheet Lookup	Replacement Lookup	Object ID Lookup	Parent Lookup	Lookup Sheet Name	From Col	To Col	Ontology Name
Library	Part Name	sbol_displayId	http://sbols.or	String	""		FALSE	FALSE	FALSE	FALSE	FALSE				
Library	Collection	sbol_members	http://sbols.or	URI	""		FALSE	FALSE	FALSE	TRUE	TRUE				
Library	SBOL Objec	sbol_objectType	http://sbols.or	String	""		FALSE	FALSE	FALSE	FALSE	FALSE				
Library	Molecule Ty	sbol_type	http://sbols.or	String	""		FALSE	TRUE	FALSE	FALSE	FALSE	molecule	A	B	
Library	Role	sbol_roles	http://sbols.or	URI	":"	"http:\/\/Wi	TRUE	TRUE	FALSE	FALSE	FALSE	role_term	B	C	SO
Library	Thing1	sbh_splitTest	https://synbio	String	":""_"		FALSE	FALSE	FALSE	FALSE	FALSE				SO
Library2	Collection N	sbol_displayId	http://sbols.or	String	""		FALSE	FALSE	FALSE	FALSE	FALSE				
Library2	SBOL Objec	sbol_objectType	http://sbols.or	String	""		FALSE	FALSE	FALSE	FALSE	FALSE				

Figure 4.7: Example of the Column Definitions Sheet used in the Multi-Sheet Templates. Unlike the single-sheet templates, this sheet had the sheet name column as the first column. This is required to identify the column name, as the same column name may be repeated. Next a column indicates the encoding property for the cell. The Namespace and Type columns are used in conjunction with this. The Split On column is used to split a single column into multiple values for a property. The Pattern column contains regex expressions to check the cell values after they have undergone conversion. The following columns contain conversion information to go from human to machine readable values. The new columns here are the object_ID lookup and the Parent lookup which are used to reference an SBOL object defined in the workbook, and the direction of the reference.

The splitting information is a single column that contains split characters surrounded by double quotes. For example, Library2, Thing1 has cells which are split both on the : character and the _ character as indicated by the Split On value of “:” “_”.

Pattern Information is also contained in a single column. It contains a series of regular expressions to apply separated by quotations. For example: “ $\wedge [a - zA - Z\backslash s*] + \$$” “$https : \backslash/\backslash/www.ncbi.nlm.nih.gov/nuccore/.*$” is used for checking sequence entries. This indicates that entries should either contain only alphabetical characters and spaces or be URLs starting with `https://www.ncbi.nlm.nih.gov/nuccore/`.

The lookup information consists of nine columns. First, a series of TRUE/FALSE columns and then columns with further information. The sheet lookup and replacement lookup work the same way as before (working in tandem with the lookup sheet name and from_col and to_col columns). The additional lookups that were added are: Tyto (Python Ontology Package), Object_ID Lookup, and Parent Lookup. The Tyto lookup uses the Tyto (`https://github.com/SynBioDex/tyto`) library rather than sheets to perform lookups. The Tyto column indicates that the Tyto library should be used and the ontology to look the terms up in is given by the ontology name column. The Object_ID Lookup means that the cell value is converted to the URI. For example, a part called GFP promoter can be referenced using that name, but this reference can then be converted to the URI http://www.examples.org/gfp_promoter using this lookup. The lookup only works for SBOL objects. If the parent lookup column is also true, then it indicates that the current object is added as a property to the referenced object rather than the standard, which is vice versa. For example, in a row for a ComponentDefinition called GFP promoter, there may be a collection column with the collection name: Test collection. If the Parent Lookup is true, then rather than adding Test collection to GFP promoter using the sbol_member attribute, the GFP promoter URI is added to the Test collection object using the sbol_member attribute.

4.1.3.3 Strengths

This new version allows multiple sheets with different SBOL object types to be processed. Additionally, the converter can now output either SBOL2 or SBOL3 documents depending on a parameter given to the converter. This makes it more flexible to integrate within synthetic biology workflows.

This version solves the issues of: redundant code cases for column conversion, redundant collection metadata, and the difficulty of special cases for all column splitting. The feature to check column conversion output using regular expressions was also added. Additionally, the inclusion of

Tyto allows ontologies to be used without creating a full ontology sheet. However, if the input is to be limited for the user then an ontology sheet still needs to be provided. Reducing code redundancy makes the library easier to understand and maintain by other developers. No longer requiring redundant collection metadata makes it less confusing for users to use. Making it easier to split columns and add ontologies increases the flexibility of the converter to handle disparate templates.

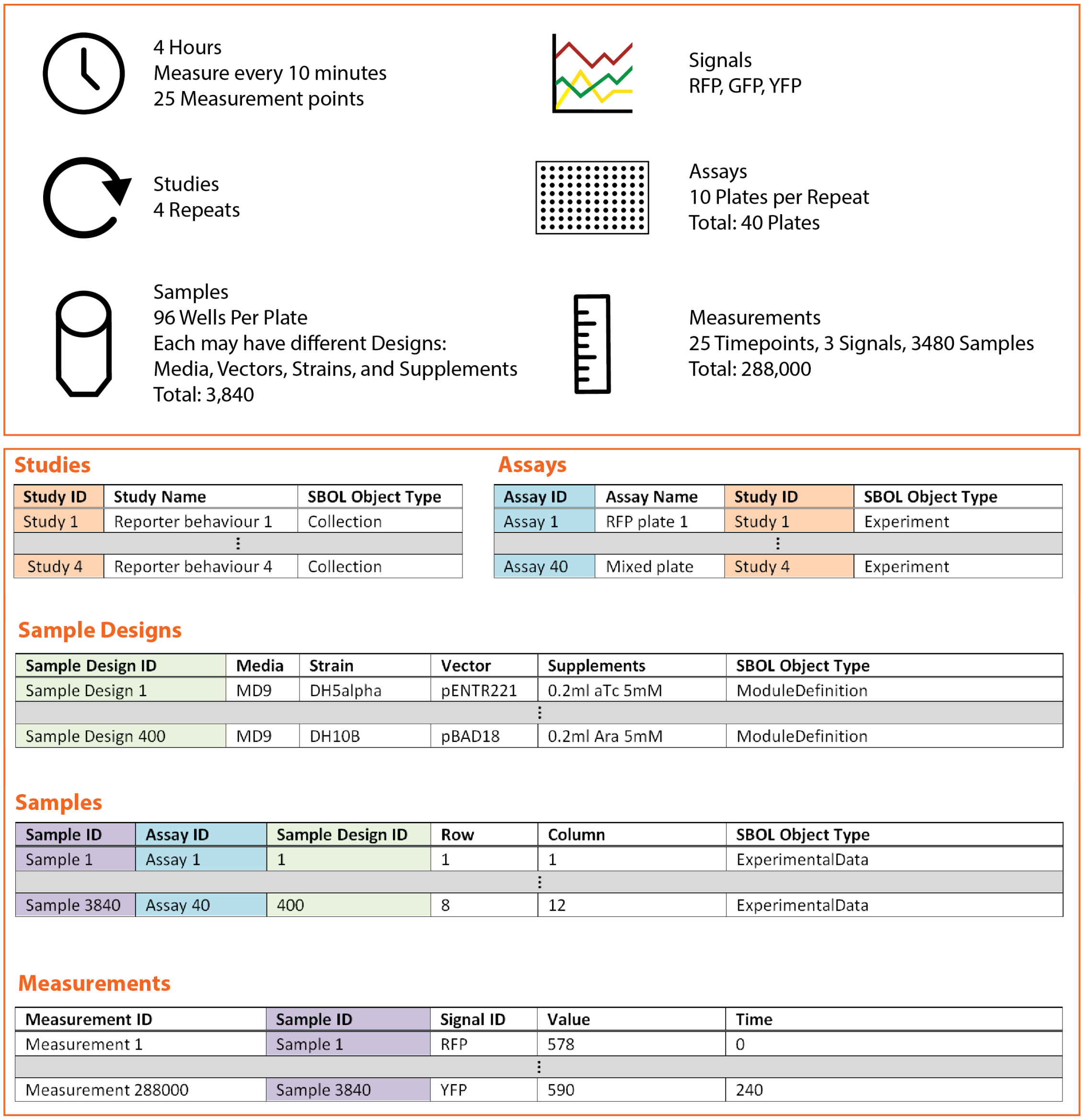
4.2 Excel-to-SBOL Case Study

The development of the converter was driven by an experimental data case study.

Flapjack is an experimental data tool to store metadata (e.g. experimental conditions, assay types) and experimental results obtained from genetic components [241]. It provides an interactive front-end for the analysis and plotting of experimental results. Additionally, a Python package and REST API is provided to allow Flapjack to integrate with other software. Data can be uploaded to Flapjack from spreadsheets. Flapjack uses SBOL to store the genetic component information, but does not store the full range of information that SynBioHub does. A system is envisioned where experimental data is stored in Flapjack and cross referenced with genetic design data in SynBioHub. To fit this system into existing synthetic biology workflows a single spreadsheet may be used to store all experimental data. This spreadsheet is then processed by two converters: Excel-to-SBOL to upload the information to SynBioHub, and the Flapjack converter to upload the information to Flapjack. Here we test the Excel-to-SBOL processing part of the proposed workflow.

Flapjack spreadsheets contain several different sheets in order to explain an experiment. There are sheets for the studies carried out, the 96 well plates used in the studies (assays), the samples (wells in the plate), and measurements (a value for a signal at a time for a particular sample).

The processing of Flapjack spreadsheets required several additional functions to be added to the Excel-to-SBOL converter. The converter had to be able to process multiple sheets with SBOL objects on them, process ExperimentalData SBOL objects, and be able to link SBOL objects



4 Hours
Measure every 10 minutes
25 Measurement points

Signals
RFP, GFP, YFP

Studies
4 Repeats

Assays
10 Plates per Repeat
Total: 40 Plates

Samples
96 Wells Per Plate
Each may have different Designs:
Media, Vectors, Strains, and Supplements
Total: 3,840

Measurements
25 Timepoints, 3 Signals, 3480 Samples
Total: 288,000

Studies

Study ID	Study Name	SBOL Object Type
Study 1	Reporter behaviour 1	Collection
	⋮	
Study 4	Reporter behaviour 4	Collection

Assays

Assay ID	Assay Name	Study ID	SBOL Object Type
Assay 1	RFP plate 1	Study 1	Experiment
		⋮	
Assay 40	Mixed plate	Study 4	Experiment

Sample Designs

Sample Design ID	Media	Strain	Vector	Supplements	SBOL Object Type
Sample Design 1	MD9	DH5alpha	pENTR221	0.2ml aTc 5mM	ModuleDefinition
			⋮		
Sample Design 400	MD9	DH10B	pBAD18	0.2ml Ara 5mM	ModuleDefinition

Samples

Sample ID	Assay ID	Sample Design ID	Row	Column	SBOL Object Type
Sample 1	Assay 1	1	1	1	ExperimentalData
				⋮	
Sample 3840	Assay 40	400	8	12	ExperimentalData

Measurements

Measurement ID	Sample ID	Signal ID	Value	Time
Measurement 1	Sample 1	RFP	578	0
			⋮	
Measurement 288000	Sample 3840	YFP	590	240

Figure 4.8: Example of a Flapjack Setup. **Top**: An example experiment for how the Flapjack multisheets may be utilized shows studies with four repeats, which have measurements taken over a period of four hours every ten minutes. For each repeat, there are ten plates per repeat with one assay per plate. With use of the standard 96 wells for each plate, we have a total of 3,840 samples across ten plates. Finally, each sample is measured at each of the 25 time points. **Bottom**: Given this experimental setup, the five different sheets are used to describe and connect information for Study, Assay, Sample Designs, Samples, and Measurements. In the Excel-to-SBOL converter, there are also seven additional sheets for the following information: “Signal” , “Media” , “Strain” , “Supplement” , “Vector” , “Chemical” , and “DNA” (which are omitted in this Figure). Note the way the sheets are linked via IDs. For example, assays identify the study they are part of using the “Study ID” column and similarly, samples to assay with “Assay ID” and measurements to sample with “Sample ID”.

across sheets. The objects had to be able to link from parent to child and from child to parent. For example, assays are able to identify the studies they are a part of and samples identify the assay they are a part of (Figure 4.8).

4.3 SBOL-to-Excel

To compliment the Excel-to-SBOL converter, work was also done to create an SBOL-to-Excel converter (`https://github.com/SynBioDex/SBOL-to-Excel`). One intended use is to allow novice users to download SBOL data in Excel format to provide an example of data that can be submitted to the Excel-to-SBOL library. This takes in an RDF file and for every top-level item creates a row with a column for every property (Algorithm 4.5).

The RDFlib python library was used to allow the processing of SBOL2 and SBOL3 files in a consistent manner. It allows conversion of many different object types whilst maintaining code simplicity. This library is able to parse any kind of RDF file (including SBOL2 XML, SBOL3 XML, and SBOL3 turtle files). This library provides simple code that can be generalized to many different file types. The code creates a list of top-level SBOL objects and then finds all of the property-value pairs for these objects. The property names are processed to make them more human readable. The columns are reordered based on a provided column order and any unnecessary columns are dropped. Each of the values are converted to be more human readable. Finally, the result is output to a spreadsheet (Figure 4.9).

4.4 SBOL-to-Excel Case Study

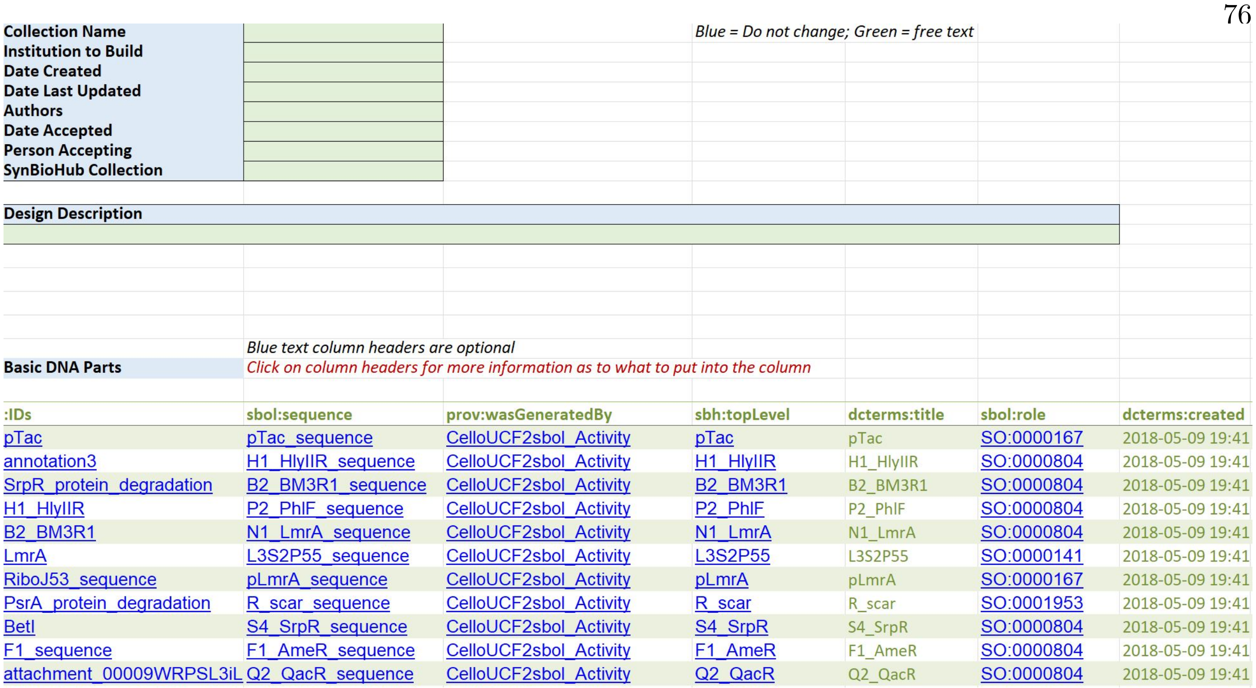
The Cello library was chosen as a case study due to the variety of SBOL objects it contains, including objects with a diverse set of custom annotations [158]. In order to achieve compliance with these varied SBOL types, it was key that the Converter had a way of dynamically manipulating these types. With the XML parsing capabilities offered by RDFLib, the converter is able to successfully process and organize all of the data into an Excel spreadsheet, facilitating the analysis of the SBOL document for the user. The results of the Cello conversion can be seen in Figure 4.9.

Algorithm 4.5: SBOL-to-Excel Iteration 2: RDF Conversion

```
Input: SBOL File
Output: Excel Filled Template
Initialization
    Load SBOL document
    Initialize output path; Initialize human readable role and organism
Parse Objects
    Initialize Data Frame
    for Subject, Predicate, Object in SBOL document do
        Add items to Data Frame, if Subject is the toplevel object
Process Subjects
    for Key, Value in Data Frame
        Process each key to a human readable form
    return Data Frame
Process Column Names
    for Column Name in Predicate
        Process the property URLs into human readable values to be set as the column
          names
    return processed Column Name
Reorder and Drop Columns
    Any Data Frame columns found in the column list are reordered according to the order
      in the column list and are moved to the front. The rest of the columns maintain the
      same order.
    Get Data Frame with list of new columns
    Drop unnecessary columns from Data Frame.
    return Data Frame with list of dropped columns
Dataframe to Excel Processing
    for Column in Work Sheet
        Initialize the column name
        for cell in Column
            Process the value within the worksheet to a human readable and hyperlinked
              form, otherwise iterate of the value
```

4.5 Submit and Download Plugins

Based on the plugin interface developed in Chapter 3 and the PyPi library developed in this chapter, it was straightforward to develop a submit plugin (see Figure 4.10) which takes in Excel sheets, constructed using the templates as described in this chapter, and uploads the converted SBOL to SynBioHub. This provides a key piece of a post-hoc curation pipeline discussed in Chapter 5. Similarly, a download plugin was developed using the SBOL-to-Excel PyPI library.



Collection Name		*Blue = Do not change; Green = free text*
Institution to Build		
Date Created		
Date Last Updated		
Authors		
Date Accepted		
Person Accepting		
SynBioHub Collection		

Design Description

Blue text column headers are optional

Basic DNA Parts *Click on column headers for more information as to what to put into the column*

:IDs	sbol:sequence	prov:wasGeneratedBy	sbh:topLevel	dcterms:title	sbol:role	dcterms:created
pTac	pTac_sequence	CelloUCF2sbol_Activity	pTac	pTac	SO:0000167	2018-05-09 19:41
annotation3	H1_HlyIIR_sequence	CelloUCF2sbol_Activity	H1_HlyIIR	H1_HlyIIR	SO:0000804	2018-05-09 19:41
SrpR_protein_degradation	B2_BM3R1_sequence	CelloUCF2sbol_Activity	B2_BM3R1	B2_BM3R1	SO:0000804	2018-05-09 19:41
H1_HlyIIR	P2_PhlF_sequence	CelloUCF2sbol_Activity	P2_PhlF	P2_PhlF	SO:0000804	2018-05-09 19:41
B2_BM3R1	N1_LmrA_sequence	CelloUCF2sbol_Activity	N1_LmrA	N1_LmrA	SO:0000804	2018-05-09 19:41
LmrA	L3S2P55_sequence	CelloUCF2sbol_Activity	L3S2P55	L3S2P55	SO:0000141	2018-05-09 19:41
RiboJ53_sequence	pLmrA_sequence	CelloUCF2sbol_Activity	pLmrA	pLmrA	SO:0000167	2018-05-09 19:41
PsrA_protein_degradation	R_scar_sequence	CelloUCF2sbol_Activity	R_scar	R_scar	SO:0001953	2018-05-09 19:41
BetI	S4_SrpR_sequence	CelloUCF2sbol_Activity	S4_SrpR	S4_SrpR	SO:0000804	2018-05-09 19:41
F1_sequence	F1_AmeR_sequence	CelloUCF2sbol_Activity	F1_AmeR	F1_AmeR	SO:0000804	2018-05-09 19:41
attachment_00009WRPSL3iL	Q2_QacR_sequence	CelloUCF2sbol_Activity	Q2_QacR	Q2_QacR	SO:0000804	2018-05-09 19:41

Figure 4.9: Spreadsheet output by the SBOL-to-Excel library. This is an example output for the Cello library. Note the hyperlinks in the cells and the human readable column names.

4.6 Conclusions

This chapter presents two Python libraries: Excel-to-SBOL and SBOL-to-Excel. The development of these libraries makes it easier for SBOL to be incorporated into existing synthetic biology workflows that make use of spreadsheets for data capture.

The Excel-to-SBOL converter allows multiple sheets with different SBOL object types to be processed. Additionally, the converter can output either SBOL2 or SBOL3 documents depending on a parameter given to the converter. The converter can be used to convert spreadsheet columns into any RDF properties. Furthermore, it can check column conversion output using user supplied regular expressions. Finally, the inclusion of lookups allows user friendly names to be used in the spreadsheet that are then processed into ontology terms by the converter.

The next steps in the development of the Excel-to-SBOL converter interface would be the development of sheet creation checking. For example, the ability to check sheet ontologies are correct using Excel Plugins. Excel plugins could also be used to automate the cell input check-

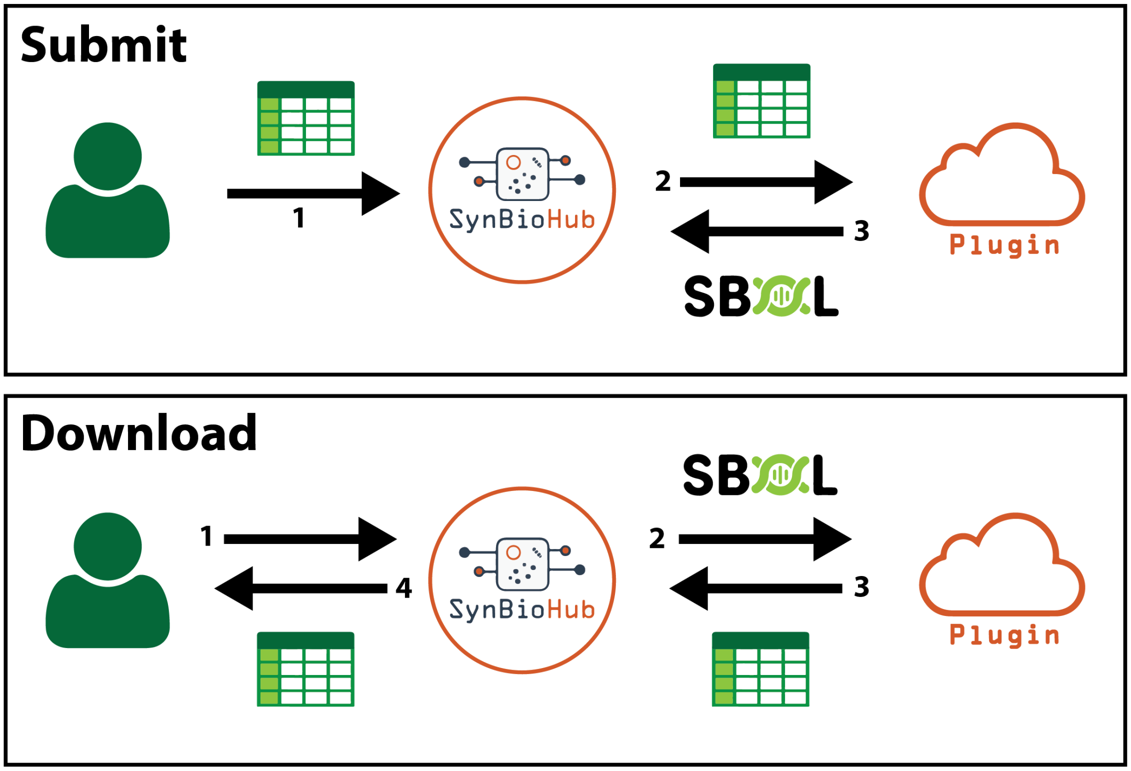


Figure 4.10: Integration of Excel-to-SBOL and SBOL-to-Excel Plugins with SynBioHub. **Submit**: When a user uploads an Excel Template to the submit endpoint, SynBioHub sends it to the Excel-to-SBOL plugin, which returns SBOL. SynBioHub then processes the returned SBOL and continues with submission. **Download**: If a user requests an Excel sheet download from SynBioHub, SynBioHub sends the appropriate SBOL to the SBOL-to-Excel plugin. The plugin returns an Excel Spreadsheet to SynBioHub, which in turn is passed to the user.

ing, rather than using the current built-in data validation based on a template list. This would be particularly useful when multiple inputs are allowed. For the development of the ecosystem, more standard templates with example data should be created. This provides a starting point for template creators as well as template users.

The SBOL-to-Excel converter can process any RDF file and turn it into a spreadsheet. This allows both SBOL2 and SBOL3 documents to be converted to spreadsheets. The conversion makes the values more human readable, and adds hyperlinks to the sheets. In order to promote greater organization, the next step for this library is to output the columns into separate sheets. Additionally, the process of conversion should be recorded in a column definitions sheet. Such a sheet would also provide the basis for the round-tripping of sheets.

The round-tripping of sheets still requires further work. Whilst the output of SBOL-to-Excel is similar to the input for Excel-to-SBOL there are a few issues. The spreadsheet output has no “Init” or “Column Definitions” sheet. Additionally, the spreadsheet uses hyperlinks that the Excel-to-SBOL converter cannot process. The creation of “Init” and “Column” sheets needs to be incorporated into the SBOL-to-Excel converter, otherwise, there is no way to reverse the property to cell value conversion. The processing of hyperlinks may be added as a feature to Excel-to-SBOL or as a stand-alone utility. Finally, to be able to round trip with SynBioHub the namespace must be considered. SynBioHub URLs cannot be re-uploaded to SynBioHub. Thus, SBOL object URLs will require a change of namespace if the objects are to be reuploaded to SynBioHub (e.g. `https://synbiohub.org/public/igem/BBa_E0040/1` to `https://www.examples.org/BBa_E0040/1`).

The case studies both indicated that more thought should be put into a core information standard. Whilst, there are currently best practices, there is still a lot of information that people wish to encode with no standard way of doing so. As further Excel templates are developed, the templates will serve as a *de facto* standards for information to be collected. Thus, the development of further templates should be done bearing this in mind. This idea is discussed further in Chapter 6.

Chapter 5

Post Hoc Curation

For parts to be reused, the effective sharing of genetic design information is required. Well characterized biological parts are needed to increase the accuracy of network level simulation [5], allow programmatic access to registry databases from multiple client applications [168], and allow the design of scalable circuits without looking at individual reactions [135]. Several attempts have been made to define what a well characterized part is. These include the Provisional BioBrick Lanugage (PoBoL) [73], the use of electrical engineering inspired data sheets for genetic devices [36, 135], behavioral characteristics of a part (for example, polymerase operations per second) [172], and atomic (non-composite/basic) parts [168]. However, due to the rapid growth of synthetic biology the standards are continually growing and expanding. This then poses the issue of components submitted at different times having different metadata associated with them.

To enable synthetic biology through the effective sharing of reusable genetic design information several databases were created. These databases have to contend with the variation in metadata over time. A method of dealing with such variation is to carry out **post-hoc curation** at intervals after submission. Three large very different databases in synthetic biology are: the **International Genetically Engineered Machines** (iGEM) Registry of Standard Biological Parts (`http://parts.igem.org/`), the AddGene database (`https://www.addgene.org/`), and the article database for ACS Synthetic Biology journal (ACS Dataset) (`https://pubs.acs.org/journal/asbcd6`). While this last one may not seem like a genetic design database, significant information about genetic designs are only shared via these articles and their corresponding supplemental data files. These three databases were used to test the post-hoc curation of genetic part information. The general pipeline is described below and then the examples of the curation being applied to each of the databases.

5.1 Post-Hoc Curation Pipeline

The post-hoc curation pipeline (Figure 5.1) is made up of three sub-workflows. There is the creation of a sequence library to use for annotating sequences pulled from records, the sequence extraction and annotation with components, and the extraction of descriptors.

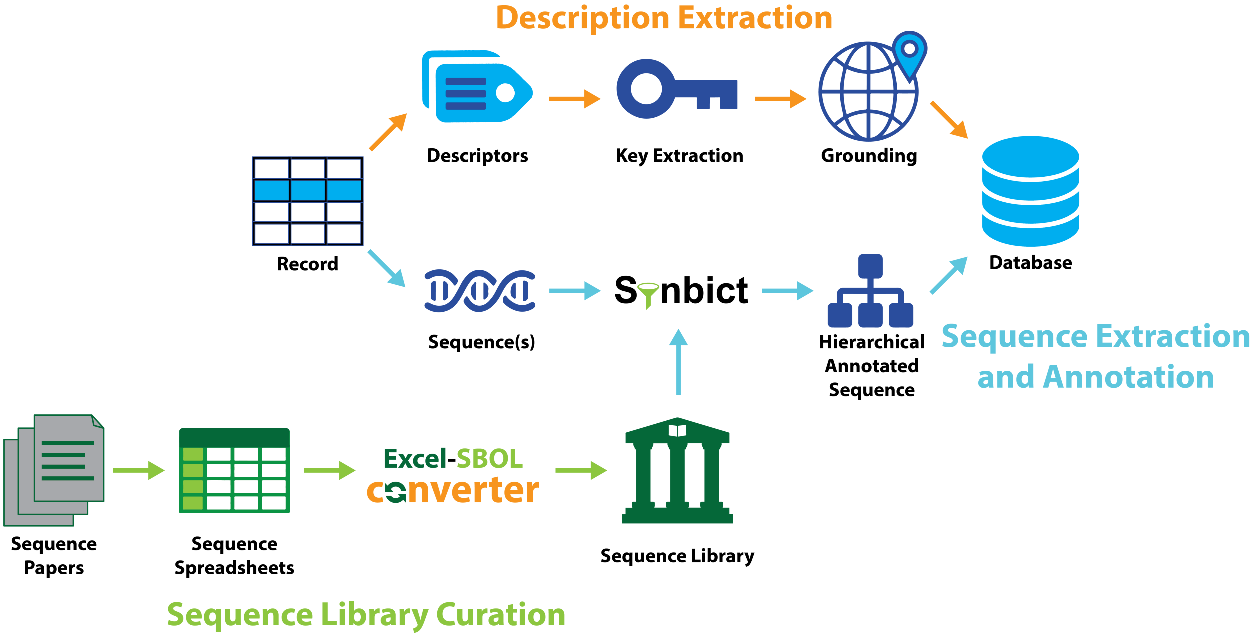


Figure 5.1: Post-hoc Curation Pipeline. The pipeline has three sub-workflows: Sequence library curation, Sequence extraction and annotation, and description extraction. Together these three workflows lead to genetic part records that have well annotated sequences, and machine readable metadata. Sequence Library Curation requires manual sequence extraction from papers with sequence libraries (e.g. a promoter library being generated). These sequences are then used to annotate sequences extracted from a record of interest (e.g. an ACS paper or iGEM registry). The record of interest is also mined for key words such as species names and cell lines. These are then grounded to machine readable terms and linked to the extracted sequences.

For the creation of a sequence library, papers are used that contain many sequences. These papers have their information manually entered into spreadsheets that are converted to SBOL via the Excel-to-SBOL Converter described in Chapter 4. These SBOL documents together form a library for annotating sequences used by the sequence extraction workflow.

The sequence extraction workflow varies somewhat depending on the record from which the sequence must be extracted. Note, a single record may contain one or more sequences. In the iGEM and Addgene case, it is much easier then in the ACS dataset as sequences in iGEM and Addgene are more separated, standardized, and machine readable. For the ACS dataset, the sequences have to be extracted using machine learning and often require parsing PDF files. Regardless of how the sequence is extracted, once it is extracted, the sequence is run through SYNBICT [181]. SYNBICT is a tool that performs automation-assisted annotation, curation, and functional inference for genetic designs. We use it to sequence match a library of parts against a test sequence to allow hierarchical annotation of sequences with the previously prepared sequence library. For each of the three case studies the same sequence libraries are used. Resources mined for yeast parts included the Yeast Toolkit [124], Pichia Toolkit [162], and a combinatorial design paper [238]. Parts for Gram-negative bacteria are drawn from the CIDAR MoClo kit [96], the CIDAR Extension Kit Volume I [145], the Voigt Lab terminator collection [43], and the *Bascillus subtilis* collection [141]. The data mined from these papers is input into Excel spreadsheets and converted to SBOL using Excel2SBOL (`https://github.com/SynBioDex/Excel-to-SBOL`). The collections were uploaded to a SynBioHub instance and used as the feature libraries in SYNBICT.

The extraction of the descriptors is the most non-standard part of the workflow. It generally requires a lot of manual input. The idea is to go from free text fields to key terms (e.g. the description: “This part is designed for use in E. coli” to “Target Organism: E. coli”). The next step is the grounding of the terms to ontologies, e.g. “Target Organism: E. coli” to “Target Organism: `https://identifiers.org/taxonomy:562`”. The grounding of terms ensures that disparate spellings (e.g. E. coli, E coli, and Escherichia coli) all are seen as the same by the computer. Additionally, grounding to an ontology can provide hierarchical relationships allowing searches for parent terms (e.g. Enterobacteriaceae) and child terms (e.g. Escherichia coli O1:HNT).

Finally, one or more records are created in a database like SynBioHub [138]. For example, a single ACS paper can be seen as a collection with properties based on the grounded terms and component items based on the annotated sequences. For an iGEM or Addgene record, a single component may be created with grounded tags added as properties to the component. How the same workflow can be applied to different data sets is discussed in more detail below.

5.2 Example Applications of the Post-hoc Curation Pipeline

5.2.1 iGEM

The work described here, as well as a more in depth analysis of the data set, was originally published as [130]. One of the largest and oldest repositories of parts is the iGEM Registry of Standard Biological Parts (`http://parts.igem.org/`). Since its inception in 2003, the iGEM competition has followed principles aimed at the advancement of synthetic biology via education, competition, and development of an open and collaborative community [195], including the submission of parts created by iGEM participants to the iGEM Registry. There are now tens of thousands of parts, the vast majority of which use the **BioBricks** format and are thus, at least in principle, able to be composed in a modular fashion [190]. Due to the wealth of genetic parts deposited in the registry and the role of the iGEM Registry in training undergraduate synthetic biologists, the registry has been used by some as an indicator to measure the progress of the field of synthetic biology [100]. When parts are found and reused they can significantly reduce the cost of creating new circuits [215]. While there is part reuse, a small core set of parts accounts for most of the reuse, and this core set remains constant from year to year [193]. Likewise, while it is possible to create full circuits based only off of registry parts [12], finding reliable parts is difficult. The large number of parts, a variety of issues around assembly methods, and issues with quality control mean that there is uncertainty with part reuse, leading many iGEM teams to choose to create and submit new parts rather than reuse old ones [221]. The use of a manual submission process until 2010 and the time pressure associated with the competition contribute to the variation of quality level in documentation and annotations [144]. There have also been a number of reviews of the iGEM data set that highlight issues such as: the lack of part reuse [168, 170, 221], the lack of annotation of sub-sequences [168], incomplete or inaccurate part descriptions [72], and lack of part validation [115, 221]. However, of these papers only [168] provides in depth statistics about the iGEM data set and an attempt to create a library of basic iGEM parts, though it does not analyze the descriptions of the parts.

To address these issues with the iGEM data set, the post-hoc curation pipeline was used to create a new iGEM library and analyze the existing one. The aim was to create a library of thoroughly documented, well annotated, and easily searchable basic parts from the iGEM data set. The aim of a library is to be a set of parts that are well documented enough that researchers can make judgements about the usefulness of components to their work with confidence. Additionally, the library must be encoded in such a way that even at a large scale the appropriate parts, if they exist, can still be easily found. To this end, data records must be machine readable. Machine readability also increases the ease of data set curation [39].

As the first step in the post-hoc curation pipeline, the registry was converted into SBOL [180] data format.

To convert the iGEM Registry to SBOL a simple automatic conversion method was used:

(1) Each part is converted into a ComponentDefinition.

(2) The Sequence Ontology (SO) role for that part is mapped from the iGEM part type

(3) A composite part composed of other BioBricks is constructed using Component instantiations.

(4) A composite part not composed of BioBricks, but rather simply described with annotations has these annotations converted into SequenceAnnotations with roles.

(5) The categories are converted into Collections. Each sub-category is mapped into a member of its parent category. All 'top-level' categories are mapped into a category_collection, and this collection and all iGEM parts are members of an iGEM collection. Each collection that a part is a member of has been annotated as an iGEM annotation within the SBOL

record for the part.

(6) Most fields available in the iGEM SQL database that have not mapped as above have been mapped into iGEM custom annotations. There are a few exceptions, but these are mostly fields that map into some other table that is not currently accessible. One example is the GroupId field that somehow maps to a Group that provided the part, but this mapping has not been shared by iGEM. Further annotations include descriptions from the registry page being added to the SBOL component via dcterms:description and sbh:mutableDescription.

The result of this procedure was: 372 Collections, 38,365 ComponentDefinitions, and 36,595 Sequences. These data were uploaded to `https://synbiohub.org`.

5.2.1.1 Pipeline Application

Sequence Extraction and Annotation

Due to the conversion to SBOL, the sequence extraction and annotation was straightforward. The SYNBICT sequences_to_features module was used to annotate the 'basic unique' ribosome binding sites with a minimum feature length of 10 bp. An overview of the annotations was created using a Python script (`https://github.com/JMante1/iGem-Data-Cleaning`).

Descriptor Workflow

As the description fields were unstructured and varied widely, expert manual curation was used for the grounding of terms. For expert curation, an abbreviated version of the SBOL version of the iGEM data set was used. It contained only parts classified as 'basic unique' by earlier analysis. The data was converted to a CSV format with the fields/properties being converted to columns. The columns included: long description, short description, notes, and source as the free text fields. The CSV was viewed and analyzed using OpenRefine `https://github.com/OpenRefine`. This allowed the reading of all of the descriptions to compile a list of all mentioned species. This list was then used together with the text filter functionality to count the number of rows/parts that contained mentions of a particular species. The ability to flag rows meant any particularly silly descriptions were flagged for further analysis and discussion.

For the analysis of the difference between real and spurious parts, a sample of 400 random ones of each was taken. The random selection was done by using Excel to generate a random number column and sorting based on that. The first 400 'real' parts and first 400 spurious parts were then selected for the analysis.

5.2.1.2 Results

Sequence Extraction and Annotation

SYNBICT was used to annotate 'basic unique' Ribosome Binding Sites (RBS) (Table 5.1). SYNBICT uses an exact match algorithm. Whilst most RBS parts were not annotated (92%), 8% were annotated, and seven sequences even had multiple annotations. There are nine annotations for a single sequence labelled as an RBS, which is surprising as RBS are typically quite short. Inspection of this sequence reveals that it is a composite rather than an RBS. The un-annotated RBS sequences may be unique new parts, or may indicate a need to expand the libraries used for SYNBICT annotation. Further manual inspection of these is required to determine whether they are suitable for adding to the annotation library.

Table 5.1: SYNBICT annotations seen in the 'basic unique' ribosome binding sites. Note that most RBS seem to not be annotated (92%), but 4 RBS sequences have 9 annotations.

Number of Annotations per Sequence	Frequency
0	411
1	30
2	2
3	1
9	4
Total	**448**

Parts in the 'basic unique' sequence set that had at least one feature annotated by SYNBICT were further inspected (see Table 5.2). Several annotations were used multiple times (e.g., BBa_B0034 was used 13 times). It seems that BBa_B0034 was cut out in several different ways (i.e.

there was more or less flanking DNA taken across different uses of the part) as the percentage cover of the annotation was quite high. On the other hand J23100 was used 12 times but had a very low percentage cover so is likely used in combination with other parts/features, some of which may be unknown. It is important to note that there are also non-RBS type annotations. In particular J23100 (promoter), the CamR Terminator, and the CamR Promoter stand out as not being RBS. This suggests that there was mislabeling of the original iGEM part type.

Table 5.2: SYNBICT annotations seen in the 'basic unique' ribosome binding sites. Note that most annotations are reused which suggests that some of the ribosome binding sites are not as unique as expected. Additionally, the annotation names suggest that not all of the sequences annotated are RBSs, e.g., finding the CamR terminator in an RBS is unexpected. The "Number of Uses" is how often the annotation is used across the corpus. The "% Cover" is the fraction of the sequence that the annotation covers.

Annotation Name	Number of Uses	Annotation Length	% Cover Min, Average, Max
BCD12	2	83	0.91, 0.93, 0.95
B0034m	3	20	0.69, 0.73, 0.8
BCD2	3	83	0.91, 0.95, 0.99
BCD8	2	83	0.91, 0.93, 0.95
BBa_B0034	13	11	0.17, 0.49, 0.85
BCD14	1	83	0.99, 0.99, 0.99
BCD13	1	83	0.91, 0.91, 0.91
B0033m	2	19	0.7, 0.75, 0.79
BBa_B0034	2	11	0.23, 0.51, 0.79
B0015	3	128	0.3, 0.3, 0.3
BBa_B0011	4	45	0.1, 0.1, 0.1
BBa_B0012	2	40	0.09, 0.09, 0.09
T7 consensus	1	22	0.27, 0.27, 0.27
UTR1	1	33	0.4, 0.4, 0.4
J23100	12	34	0.01, 0.02, 0.04
B0032m	2	21	0.72, 0.77, 0.81
ChlorR	3	659	0.29, 0.3, 0.3
BBa_B0057	3	41	0.02, 0.02, 0.02
CamR Promoter	6	104	0.05, 0.05, 0.05
BBa_B0062-R	3	40	0.02, 0.02, 0.02
CamR Terminator	3	108	0.05, 0.05, 0.05
BCD16	1	83	0.91, 0.91, 0.91
Average	**3.3±3.2**	**83±133**	**0.44±0.37, 0.48±0.37, 0.52±0.39**

Descriptor Workflow

Expert annotation was used to process the part descriptors found in the ‘basic unique’ iGEM data sets. Despite some sequences being long enough to be plausible for a part type, it might not actually be a plausible sequence. Deciding what kind of sequence is plausible is difficult, but looking at the sequence description fields often gives a better idea. For example, at the time of the iGEM to SBOL conversion BBa_K1740001 has a sequence of length 672 base pairs, however the description fields say ‘This is a test’, ‘none’, and ‘.’ which suggests that the sequence likely is not ‘real’. For this reason, a mostly qualitative analysis was carried out to identify patterns in the sequence descriptions which could form the basis for further curation efforts. The main observations are as follows:

- Sequence descriptions and the relevant information is split across four fields: long description, notes, source, and description (Table 5.4). The split is by no means equal though with 7530 (39%) having no text in at least one of the fields. The length of the text varies widely between parts and does not necessarily correlate with valuable information (for example, a single Addgene reference conveys much more than ‘test test test test’). Additionally, the descriptions do not appear to improve over the years, or over the months per year as the iGEM jamboree gets closer (see the Appendix A for the related figures).
- Many of the parts are temporary/test parts. They are often indicated via descriptions including ‘test’, ‘temporary’, ‘none’, ‘kill’, ‘bla’, ‘blah’, or a keyboard smash.
- Subheadings have been implemented within single fields to break out information more clearly such as: notes, references, source, see also, design notes, mutagenicity, assembly, and functional parameters.
- In many cases a lot of information is present in descriptions (e.g., source organism, target organism, paper citations, assembly methods) however how it is presented varies widely (e.g., species information may include genus, species, or be a common name, as shown in Figure 5.2).

- There are many descriptions stating they will be edited or finished later when more information is known/gathered, or the part has been tested.

To compare such spurious parts to ‘real’ parts, 400 of the spurious parts were randomly selected and 400 of the ‘real’ parts were randomly selected (full list is given in the supplemental). A significant difference was seen in the length of descriptive fields between the two sample groups. The ‘real’ parts had more information in each of the fields and generally had significantly longer sequences too (Table 5.3). These results are not surprising as part of the method of determining whether a part is ‘real’ is looking at the description provided. Additionally, spurious parts are expected to generally have shorter made up sequences. SYNBICT was used to annotate the sequences for the two groups. In the ‘real’ group, the average number of annotations was 0.340 ± 1.133 per sequence (i.e. on average 0.340 (± standard deviation) sequence library parts are found as sub-sequence per iGEM sequence), whilst in the spurious group it was 0.375 ± 1.246 per sequence. The student’s t-test indicates no significant difference between the two groups with a p-value of 0.678. The lack of difference in annotation may be explained by the annotation library used in SYNBICT (a larger set of libraries might have led to more annotations). It should be noted that all parts selected had no annotations by the authors so the lack of significant difference between ‘real’ and spurious annotation number does not indicate a lack of correlation between ‘good’ descriptions and ‘good’ sequence annotations in general.

	Long Description	**Notes**	**Sequence Length**	**Description**
‘Real’ Sample	264 ± 320	117 ± 208	1028 ± 2298	32 ± 20
Spurious Sample	30 ± 69	11 ± 43	629 ± 1216	23 ± 21
p-value	<0.00001	<0.00001	0.00234	<0.00001

Table 5.3: Comparison of a random sample of 400 ‘real’ vs 400 spurious parts. The length of each of the descriptive fields is given in characters (mean ± standard deviation). The p-value for the student’s t-test between the two groups is given. Each of the fields show a significant difference between the ‘real’ and spurious group at the 0.05 alpha level.

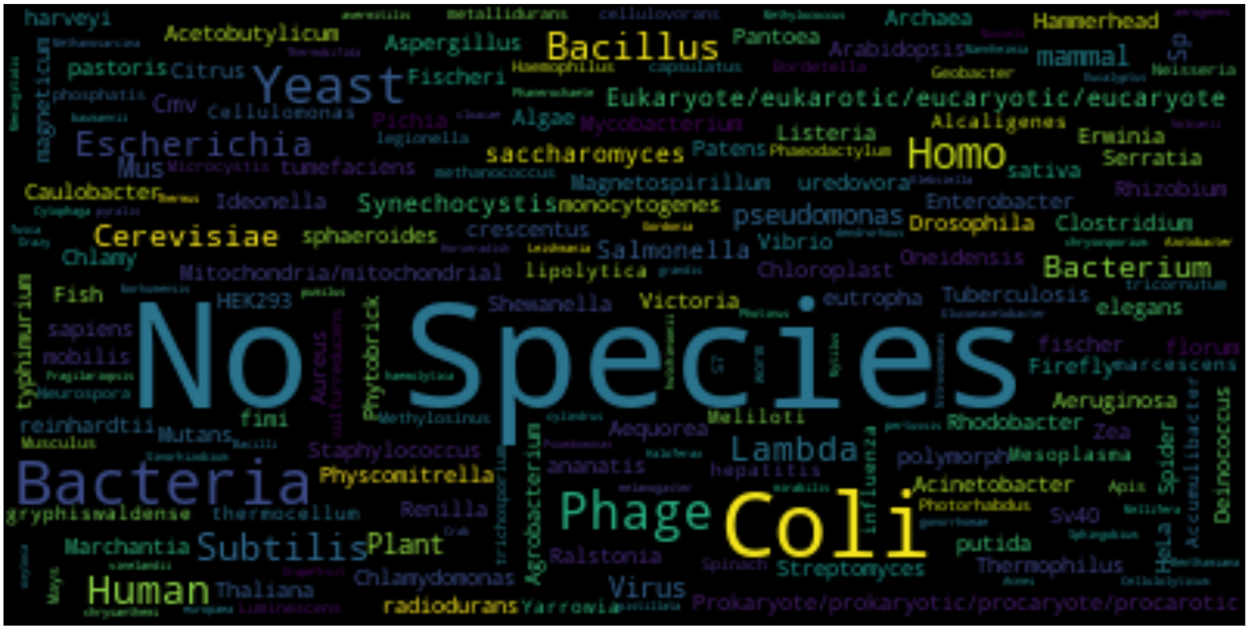


Figure 5.2: This is a word cloud of 'species' manually extracted from the long description of basic iGEM parts. The frequency of a genus name versus a species name were counted separately to account for *E. coli* or *Clostridium sp.* There are 271 species (not including 'No species') across the data set. However, of the 19825 parts 10978 have no species mentioned (55%). On average every species is mentioned by 41 ± 159 parts.

Table 5.4: Statistics about the length of sequence description fields for the iGEM 'basic unique' data set. There are 4 sequence description fields: Long Description, Notes, Source, and Description. The 'Fields Std.Dev.' column indicates the standard deviation between the length of the four description column fields, and 'Fields Mean' calculates the mean length of the four fields.

	Long Description	Notes	Source	Description	Fields Std.Dev.	Fields Mean
Mean	263	102	67	31	135	116
Std.Dev.	736	218	149	20	371	210

5.2.1.3 Challenges

While the registry is a very large and well known repository of synthetic biology information, it does have several design flaws. The flaws which hinder machine readability, and the curation of an iGEM library are: 1) its over-reliance on free text, 2) insufficient part provenance, 3) part duplication, 4) the lack of part removal, and 5) insufficient continuous curation. To automate post-hoc curation these five concerns must be considered and addressed. We suggest a submission structure to nudge participants towards the submission of well characterized parts.

5.2.2 ACS Dataset

This work was originally presented in [129]. ACS Sytnethic Biology (`https://pubs.acs.org/journal/asbcd6`) is a peer-reviewed scientific journal published by the American Chemical Society. It started accepting articles in 2011. It is a major publishing outlet in the synthetic biology community with special issues for synthetic biology conferences (such as IWBDA, SEED, and the International Meeting on Synthetic Biology), topics (e.g. cell free systems, genome engineering, and circuits in metabolic engineering), and geographical overviews (e.g. Europe and Asia).

The ACS Synthetic Biology corpus consists of all the articles that have been published in ACS Synthetic Biology up to November 2019. These articles were provided to us in annotated JATS XML format, which includes a rich set of metadata, the full article text, and structured references to cited papers.

Additionally, supplemental files are parsed to identify and extract genetic sequences of parts. From the 1597 articles (these were all articles published since the inception of the journal at the time of our analysis), there is at least one supplemental file available for 82 percent of the articles, and there are 10,070 supplemental files in total. These supplemental files are provided in various formats ranging from structured data, such as GenBank files, to PDF documents. The top 10 most frequent file types are shown in Table 5.5.

Whilst parts databases are recognized as key for information exchange, many of the databases

Rank	File Type	Count	Rank	File Type	Count
1	PDF	1434	6	SBML	475
2	TSD	1091	7	PNG	403
3	XML	1079	8	JPG	337
4	GenBank	707	9	TXT	330
5	HTML	499	10	JS	325

Table 5.5: Top 10 most common supplemental file types

have been ineffective in communicating data and metadata (like function, intended host, and assembly method). Thus, databases are rarely used in genetic design, especially for organisms that are not *E. coli*. This leaves the field in a state where relevant part performance data is distributed among results and methods sections, paper supplemental files, and tables of sequences — shifting the work of a genetic designer from design to searching through disparate sources for part information.

The aim of using the annotation pipeline on the ACS corpus was to combine the richness of information in literature with the searchability and standardization of databases. This will save the genetic designer search time and reduce the risk of missing valuable information about parts. To do this the *ACS Synthetic Biology* corpus is semantically annotated and sequences are extracted from the papers and supplemental files.

5.2.2.1 Pipeline Application

For the ACS Dataset an article is the record from which descriptors and sequences will be extracted (as per the pipeline in Figure 5.1).

Descriptor Workflow

The metadata and citation elements of the structured article file are harvested and converted into SBOL-compliant RDF/XML with Dublin Core annotations suitable for ingestion into SynBioHub. Among the steps taken during this process is the employment of Python scripts to match article DOIs to corresponding PubMed IDs. Additionally, various **natural language processing** (NLP) techniques to extract salient content from published articles. Each article is provided in the

Journal Article Tag Suite (JATS) XML format, and it is parsed to extract text and relevant metadata. Each supplemental file is also parsed to extract genetic sequences of parts. The article text is then processed using **named entity recognition** (NER) to identify terms [151, 51, 2], **relation extraction** (RE) to identify relationships between terms [228, 110], and **topic modeling** (TM) to identify topics [219, 104]. These discovered terms and concepts are then used to tag the article, providing a way to link the context of each article to data elements.

Sequence Extraction and Annotation

The supplemental files are processed as well as the main article text. Sequences were extracted using regular expressions looking for a combination of ATGC and spaces that is longer than 4 characters (to rule out cat, act, etc). The sequence extraction process found at least one sequence from 8 percent of the supplemental files, and 89,620 genetics sequences were found in total. Of these 89,620 entries, 1,732 sequences are skipped due to blank entries or a lack of publications to match them to. As a result, only 87,888 sequences are processed.

Database Upload

For every sequence, an annotated SBOL Componentdefinition is created which is linked to an SBOL Collection object for the paper it comes from. The SBOL Collections are all uploaded as part of a single ACS Collection in SynBioHub.

5.2.2.2 Results

The ACS Synthetic Biology corpus extracted sequences are converted to SBOL so they can be annotated using SYNBICT (`https://github.com/SD2E/SYNBICT`).

The annotation of genetic parts was hindered by the accuracy of sequence scraping. Table 5.7 indicates that some file types do better for scraping than others. Notably PDFs are a very popular supplemental type (93 percent) for presenting sequences, but they fair poorly when automating sequence extraction for annotation. This is partially due to the fragmentation of sequences (any spaces or numbers between rows of a sequence will lead to a sequence being split into several sequences by the scraping method as indicated by the shorter average sequence length). GenBank files put into PDFs were found particularly difficult to scrape. However, when GenBank files are provided in their native file format, they are more effectively annotated. The use of SBOL and FASTA standard formats also significantly increases the machine readability of sequences. SBOL shows annotation within shorter sequences, which may be indicative of the better coverage of the part libraries in these formats or the shorter sequences due to sequences being of parts not whole constructs. The top annotations for the sequences are shown in Table 5.8.

Using the descriptor workflow, the paper "entities" were also extracted. For the initial round of NER, standard biological entity categories (e.g., genes and chemicals) are used, since there is no labelled dataset for synthetic biology entities. Results from this initial round are reviewed and corrected by domain experts to create a more refined dataset with entities more specific to synthetic biology that can be used to fine tune the NER models. Named entities expected to be detected within synthetic biology articles are also added to the articles as suggested annotations, to be confirmed by expert annotators in order to facilitate the creation of gold-standard synthetic biology-specific training data. Table 5.6 shows the entity types, total number of annotations and average annotation per document for each entity type over the ACS Synthetic Biology dataset. The table also shows the total number of entities and the unique number of mentions annotated by the NER component. Table 5.9 shows the top ten terms identified by the NER component from the ACS Synthetic Biology dataset for each of the entity types. These results suggest that while NER is not perfect, it is able to identify many of the entity types correctly. The next step in the process is to refine the model based on human corrected annotations. For example, the NER system identified LB and $M9$ as $Cell\ Line$ whereas they are formulations of media, so therefore should probably be annotated as $Chemical$.

5.2.2.3 Challenges

There are two main issues with the application of the pipeline to the ACS Dataset: the requirement of expert curation and the difficulty of extracting sequences from the ACS corpus.

The first problem is difficult to address, though improved machine learning should decrease

Entity Type	Total Annotations	Average per Document
Cell Line	12,084	13.47
Chemical	86,180	96.07
Gene	169,061	188.47
Species	31,785	35.43
Total Number of Mentions	299,110	
Number of Unique Mentions	43,439	

Table 5.6: Entity statistics from named entity recognition component. Entity Type is the entity category. Total Annotations are the number of entities extracted from the dataset. Average per Document is the average number of entities per document over all the documents in the ACS Synthetic Biology dataset. Total Number of Mentions is the total number of entities, and Total Number of Unique Mentions are the number of unique mentions identified in the ACS Synthetic Biology dataset.

File Type	No. of Sequences	% of Sequences Annotated	Avg. Sequence Length	Avg. Annotated Sequence Length
FASTA/TXT	679	0.88	397	5014
GenBank	686	79.88	4419	4286
XML/SBOL	5098	26.19	399	988
PDF	81426	0.38	56	1829
Total	87888	2.5	113	1825

Table 5.7: Annotation of sequences from ACS Synthetic Biology Supplemental Files. There are different supplemental file types used with the majority (93 percent) being PDFs. Unlike FASTA, GenBank, and SBOL file types, PDFs need to be scraped to extract sequences. The lack of standard page breaks and formatting of the sequence leads to errors in sequence extraction meaning a single sequence is often read in as several shorter fragments. This is noticeable as the average sequence length for PDFs is significantly lower than the other file types. The other feature of note is that the percentage annotation is much higher for FASTA, GenBank, and SBOL, which are standard sequence formats. Of these, SBOL shows annotation within shorter sequences which may be indicative of the better coverage of the part libraries in these formats or the shorter sequences due to sequences being of parts not whole constructs.

the expert annotation required over time. Additionally, continuous curation where the work is done gradually, over time, by the experts submitting the data, rather than retrospectively should decrease the total time required for annotation. Furthermore, making the formats more machine readable will also improve the accuracy of computer aided descriptor annotation and grounding.

Similarly, sequence extraction would greatly improve from standardized, machine readable sequence submission. The annotation could also be improved by having the submitting authors check the machine annotations in a continuous manner.

All	Promoters	CDS	Terminators	Other
RiboJ (397)	AmpR_promoter (302)	AmtR (173)	L3S2P21 (386)	RiboJ (397)
L3S2P21 (386)	pCONST (288)	PhlF (172)	AmpR_terminator (226)	ColE1 (153)
AmpR_promoter (302)	pTet (157)	SrpR (146)	BBa_B0015 (214)	RiboJ53 (86)
pCONST (288)	pTac (123)	HlyIIR (146)	CamR_Terminator (135)	BydvJ (85)
AmpR_terminator (226)	pBAD (122)	BetI (117)	ECK120033737 (92)	RiboJ51 (73)
BBa_B0015 (214)	CamR_Promoter (116)	CamR (111)	L3S2P55 (86)	RiboJ10 (73)
AmtR (173)	pPhlF (92)	AraC (107)	ECK120033736 (81)	RiboJ57 (57)
PhlF (172)	pSrpR (73)	TetR (98)	ECK120019600 (79)	RiboJ54 (19)

Table 5.8: Top annotations of ACS Synthetic Biology Supplemental sequences. These are the top annotations of different SO:Role types (e.g. Promoter) that were used to annotate the ACS Synthetic Biology supplemental sequences.

All		Cell Line		Species		Gene		Chemical	
#Doc.	Term	#Doc.	Term	#Doc.	Term	#Doc.	Term	#Doc.	Term
7682	E. coli	305	LB	7682	E. coli	4549	GFP	1725	glucose
4549	GFP	292	cells	2965	yeast	1970	Cas9	1391	peptide
2965	yeast	185	HEK293T cells	1202	S. cerevisiae	1298	mCherry	1206	amino acid
1970	Cas9	181	BL21	822	human	1200	dCas9	1180	amino acids
1725	glucose	180	MG1655	564	B. subtilis	1013	CRISPR	835	glycerol
1391	peptide	172	M9	430	P. putida	938	LacI	829	kanamycin
1298	mCherry	125	cultures	367	Escherichia coli	839	sfGFP	789	NaCl
1206	amino acid	118	KT2440	332	C. glutamicum	786	YFP	781	peptides
1202	S. cerevisiae	100	HeLa cells	321	S. oneidensis	748	TetR	737	tetracycline
1200	dCas9	96	strains	299	Synechocystis	673	RFP	657	ampicillin

Table 5.9: Top ten terms extracted from the ACS Synthetic Biology dataset by the NER component for each entity type (Cell Line, Species, Gene and Chemical) and across all of the entity types (All).

5.2.3 Addgene

Addgene is a global nonprofit plasmid repository (`https://www.addgene.org/`). It has as its mission: "Accelerate research and discovery by improving access to useful research materials and information". They provide their materials in a searchable database, and sequence all items in their repository for quality control. To submit new plasmids to the repository, researchers can either use a graphical user interface or upload their parts via a spreadsheet (similar to the idea described in Chapter 4). The use of a spreadsheet standardizes what information is collected, reduces free text, and restricts information expression.

The post-hoc curation pipeline was applied to the Addgene data set to look for frequent sub-component use, analyze the effectiveness of Addgene's spreadsheet design, and make the Addgene library more accessible to design tools such as iBioSim and SBOLCanvas.

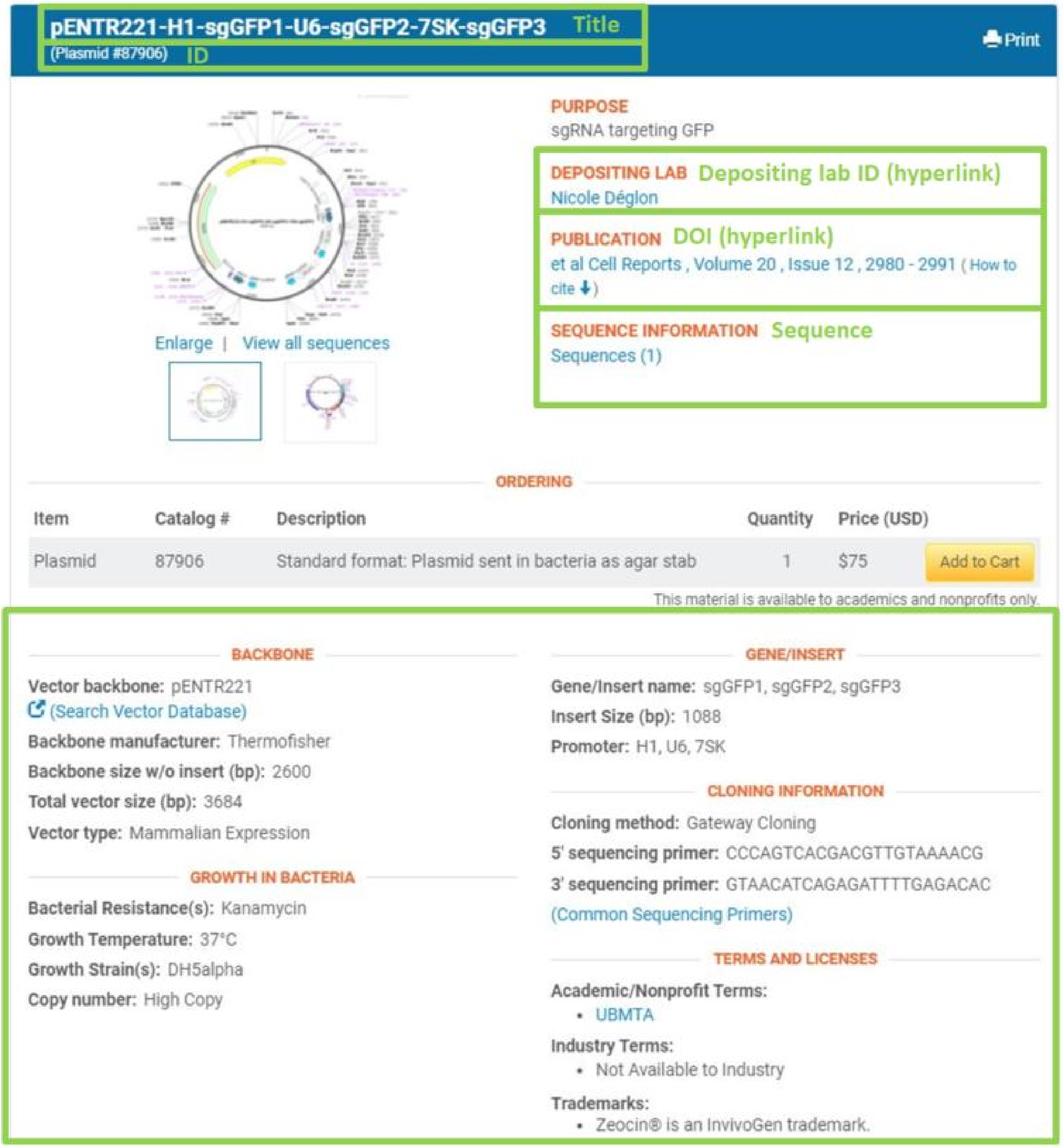
5.2.3.1 Pipeline Application

For the Addgene Dataset, a plasmid is the record from which descriptors and sequences will be extracted (as per the pipeline in Figure 5.1)

Descriptor Workflow

For the descriptors, all 181,796 plasmid records were pulled as HTML, of these 50,953 were successfully scraped. The plasmid title and Addgene id were extracted together with publication DOI, sequence, and depositing lab. Any further headings in the section below ordering are extracted as heading:sub-heading:term (Figure 5.3). Each of these are then added to an SBOL document using the Addgene namespace and a property name based on the sub-heading (with spaces, 3, and 5 re-coded to "_", "three", and "five", respectively).

After the initial descriptor pass a list of all Addgene predicates was created (see Table 5.10), as well as individual CSV files for each predicate. The predicate CSVs contain all Addgene IDs and their corresponding value for the predicate of interest. These files were used to assess the variability of responses, and to attempt to ground the species predicate. The results of this grounding were then



pENTR221-H1-sgGFP1-U6-sgGFP2-7SK-sgGFP3
(Plasmid #87906)

PURPOSE
sgRNA targeting GFP

DEPOSITING LAB
Nicole Déglon

PUBLICATION
et al Cell Reports , Volume 20 , Issue 12 , 2980 - 2991 (How to cite)

SEQUENCE INFORMATION
Sequences (1)

ORDERING

Item	Catalog #	Description	Quantity	Price (USD)
Plasmid	87906	Standard format: Plasmid sent in bacteria as agar stab	1	$75

This material is available to academics and nonprofits only.

BACKBONE

Vector backbone: pENTR221
(Search Vector Database)
Backbone manufacturer: Thermofisher
Backbone size w/o insert (bp): 2600
Total vector size (bp): 3684
Vector type: Mammalian Expression

GROWTH IN BACTERIA

Bacterial Resistance(s): Kanamycin
Growth Temperature: 37°C
Growth Strain(s): DH5alpha
Copy number: High Copy

GENE/INSERT

Gene/Insert name: sgGFP1, sgGFP2, sgGFP3
Insert Size (bp): 1088
Promoter: H1, U6, 7SK

CLONING INFORMATION

Cloning method: Gateway Cloning
5' sequencing primer: CCCAGTCACGACGTTGTAAAACG
3' sequencing primer: GTAACATCAGAGATTTTGAGACAC
(Common Sequencing Primers)

TERMS AND LICENSES

Academic/Nonprofit Terms:
- UBMTA

Industry Terms:
- Not Available to Industry

Trademarks:
- Zeocin® is an InvivoGen trademark.

Figure 5.3: Example of an Addgene plasmid page. For the descriptors plasmid records were pulled as HTML. The plasmid title (pENTR221-H1-sgGFP1-U6-sgGFP2-7SK-sgGFP3) and Addgene id (87906) were extracted together with publication DOI, sequence, and depositing lab. Any further headings in the section below ordering are extracted as heading:sub-heading:term. For example, backbone:vectorBackbone and growthInBacteria:copyNumber.(This page can be found at: https://www.addgene.org/87906/)

applied by adding a custom annotation called SBH:Source_Organism to the SBOL files (Table 5.11). The code for this can be found at `https://github.com/JMante1/Addgene-Annotation`.

Sequence Extraction and Annotation

As part of the HTMLhtml parsing the sequence links were extracted. The first of these links is used to obtain a GenBank file which can then be converted to SBOL using the SBOL Validator. SYNBICT is then used to annotate the sequences with subparts.

5.2.3.2 Results

Descriptor Workflow

Fifty-two different descriptor properties were seen (Table 5.10), however, some of these are plurals of each other. For example: articleCitingThisPlasmid vs articlesCitingThisPlasmid. The presence of the different descriptors varied per record with properties like bacterialResistances being seen on every record and addgeneNotes only seen on 390 of the 50,953 (0.77%).

Some properties were often seen together such as fiveSequencingPrimer, and threeSequencingPrimer (co-variance of 0.1338). On the other hand, some properties were generally not seen together such as purpose and termsAndLicenses (co-variance of -0.0809).

Most of the work on descriptors was done looking at the “species” property. 353 distinct species were seen in the annotations. An additional 74 were considered ungroundable. Terms were ungroundable when they were too vague (e.g. hamster), pointed to multiple species (e.g. C. griseus), or did not contain species information (e.g. metagenomic). Variation in species entries were seen due to punctuation (e.g. Synechocystis PCC 6803, Synechocystis PCC6803, Synechocystis sp. PCC 6803, and Synechocystis sp. PCC6803), not fully writing out the genus name (e.g. B. taurus (bovine) vs. Bos taurus), adding a common name to the end (e.g. A. Victoria (Jellyfish) vs. A. Victoria), adding protein information (GFP is from Aequorea victoria), and mis-spellings (e.g. Aequorea victoria vs. Aequoria victoria). 523 of the 50,953 records (10.26%) had multiple species in the “species” property. The top 50 species annotations and their occurence are shown in Table 5.11. Interestingly, synthetic is often used. Additionally, whilst *E. coli* is generally used

Property name	Percentage Filled	Number Filled
addgeneId	100	50953
bacterialResistances	100	50953
copyNumber	100	50953
depositingLab	100	50953
doi	100	50953
growthStrains	100	50953
growthTemperature	100	50953
plasmidTitle	100	50953
refBib	100	50953
refMethod	100	50953
vectorBackbone	100	50953
geneInsertName	99.99	50946
industryTerms	99.79	50846
vectorType	97.51	49685
academicNonprofitTerms	89.52	45615
cloningMethod	88.03	44853
species	87.78	44725
purpose	80.57	41054
fiveSequencingPrimer	73.03	37211
insertSizeBp	55.83	28448
backboneSizeWoInsertBp	55.77	28414
fiveCloningSite	53.03	27020
threeSequencingPrimer	49.96	25458
threeCloningSite	49.74	25343
promoter	48.13	24526
entrezGene	46.55	23717
tagFusionProtein	39.4	20076
backboneManufacturer	38.63	19682
altName	33.91	17276
totalVectorSizeBp	32.27	16444
mutation	32.09	16353
selectableMarkers	30.52	15552
supplementalDocuments	28.29	14415
genBankID	18.38	9364
aPortionOfThisPlasmidWasDerivedFromAPlasmidMadeBy	13.19	6720
termsAndLicenses	10.27	5231
tagsFusionProteins	8.41	4285
modificationsToBackbone	7.72	3935
gRNAshRNASequence	6.42	3269
articleCitingThisPlasmid	6.19	3152
articlesCitingThisPlasmid	5.84	2976
growthInstructions	4.59	2339
addgeneNotes	0.77	390
pricing	0.26	132
storage	0.26	132
shipment	0.22	113
titer	0.22	113
volume	0.22	113
amount	0.04	19
guaranteedConcentration	0.04	19
reference	0.03	16
depositingLabs	0	1

Table 5.10: All properties scraped from Addgene. A subset of the full addgene dataset was used (because there were errors in scraping some of the data). For the list of all Addgene IDs used see https://github.com/JMante1/Addgene-Annotation/blob/master/Analysis/all_addgene_files.csv.

as a target organism (not what the species property indicates) the genes used generally come from other model organisms.

Sequence Extraction and Annotation After SYNBICT annotation 346 unique sequences were used to annotate the library a total of 98,340 times. The annotations ranged in length from 4,163 base pairs to 39 base pair. The annotations ranged from less than .01 percent to 65 percent of the sequence. The top annotations seen are listed in Table 5.12. The number of annotations per sequence range from 0 to 28 with an average of 1.93 and a standard deviation of 1.75. The plasmids with more annotations are ones that were well annotated before SYNBICT annotation was run.

5.2.3.3 Challenges

Despite less free text being used (compared to iGEM and ACS) the Addgene data set still faces issues. For some of the properties (e.g. species and growthStrains), the use of ontologies would strengthen the data set by making it more uniform. Additionally, the use of ontologies might standardize the way in which multiple tags are added (currently many different delimiters are used including ";", ":", "&", and "and"). An alternative solution would be to allow sub-part annotations, as often multiple annotations are used when different parts of the plasmid are being described (e.g. GFP came from *A. victoria* and Cas9 from *S. pyogenes M1*).

Further work should also be done on the conversion of HTML to SBOL as only 50,953 of the 181,796 (28.03%) of the Addgene records were successfully converted to SBOL.

5.3 Conclusions

The initial expectation was to produce a single reusable pipeline for the curation of data sets as well as sets of well annotated part libraries. This was not achieved due to the variation and intense manual curation required in post-hoc curation.

However, whilst well annotated reusable parts libraries were not achieved uses of annotations were found. For example, BBa_B0034 was a common annotation across the three data sets indicating it is used and may be reliable to reuse. This suggests that it may not be possible to judge a part

Ontology term	Human Readable	Occurance
9606	Homo sapiens	3463
32630	Synthetic	2437
10090	Mus musculus (mouse)	975
4932	Saccharomyces cerevisiae	514
7955	Danio rerio (zebrafish)	331
10116	Rattus norvegicus (rat)	270
3702	Arabidopsis thaliana	235
562	Escherichia coli	226
1314	Streptococcus pyogenes	138
6183	Schistosoma mansoni	114
7227	Drosophila melanogaster (fly)	99
9031	Gallus gallus (chicken)	98
11676	Human immunodeficiency virus 1	84
6239	Caenorhabditis elegans (nematode)	79
Ungroundable		74
37296	Kaposi's Sarcoma Associated Herpesvirus (HHV-8)	73
4577	Zea mays	49
6100	Aequorea victoria	43
8355	Xenopus laevis (frog)	26
32644	unspecified	24
130404	Piper methysticum	23
10684	Bacteriophage PBS2	22
487	Neisseria meningitidis	20
11036	venezuelan equine encephalitis	20
1140	Synechococcus elongatus PCC 7942	18
9483	Callithrix jacchus	18
2886926	Bacteriophage P1	18
3847	Glycine max	17
9913	Bos taurus (bovine)	17
666	Vibrio cholerae	15
7049	Lampyridae	14
4952	Yarrowia lipolytica	13
693140	Tiarina fusus	13
1820202	Phagocata morgani	13
7054	Photinus pyralis	12
89184	Silicibacter pomeroyi	12
135777	Dugesia dorotocephala	12
386891	Moraxella bovoculi	12
1354672	Phagocata gracilis	12
4896	Schizosaccharomyces pombe (fission yeast)	11
28285	Adenovirus type 5	11
264483	Xanthophyllomyces dendrorhous	11
1148	Synechocystis PCC 6803	10
7159	Aedes aegypti	10
30538	Vicugna pacos	10
86600	Disocoma sp.	10
2884423	Bacillus phage PBS1	10
264	Francisella tularensis subsp. Novicida	9
4498	Avena sativa	9
272131	Nostoc punctiforme	9

Table 5.11: The top 50 species annotations seen in the Addgene descriptor analysis. The first column gives the NCBI:txid, the next a human readable version, and the occurrence is the number of times the label is seen. Note that a label such as *Homo sapiens*: *A. thaliana* is counted as an occurence of *Homo sapiens* and of *Arabidopsis thaliana*.

Annotation Name	Annotation Length (bp)	Frequency of Occurrence
AmpR promoter	96	17299
AmpR terminator	130	14336
ColE1	763	10789
AmpR promoter	96	10679
R0010	199	10311
AmpR terminator	130	10108
ColE1	763	6077
R0010	199	4399
B0015	128	1446
BBa_B0010	79	1162
ECK120051383	47	728
BBa_B0062-R	40	623
BBa_B0057	41	585
Bba_R0040 TetR promoter	53	578
CamR Promoter	104	454
CamR Terminator	108	440
SpecR Promoter	133	435
araC	878	418
URA3 terminator	77	350
ECK120034551	52	336
URA3 terminator	77	304
KanR promoter	147	289
CmR	659	254
ECK120029600	89	244
CamR Promoter	104	200
BBa_B0053	71	195
B0015	128	194
No DisplayID	74	191
Tagtef1	238	189
L3S2P21	60	184
CmR	659	158
BBa_B0011	45	122
LEU2 Promoter	647	121
BBa_B0010	79	118
AOX1 terminator	246	117
ECK120051382	63	116
TetR	623	104
Bba_R0040 TetR promoter	53	102
pBAD	330	100
ECK120051383	47	94
KanR promoter	147	91
KanR	815	88
PTac	70	78
BBa_B0062-R	40	77
BBa_B0057	41	75
L3S3P21	52	68
Tagtef1	238	64
AmpR	857	63
tSNR52	74	61
BBa_B0011	45	60

Table 5.12: The top 50 sequence annotations seen after annotation with SYNBICT. Note that repeated names can indicate the same name was given to two different sequences e.g. that there are two different AmpR promoter sequences found.

as reliable or not depending on the number of annotations from the part library. However, a part from the part library may be seen as reliable if it is used to annotated several Addgene plasmids, which are known implementations.

Additionally, the post-hoc curation process did give insights into the challenges of post-hoc curation and ways in which it could be made easier. Additionally, it provided information on the metadata contained in three repositories that are very important to synthetic biology. Knowing what information to encode, together with how to make the information more machine readable will help when designing new information intake systems. Such systems may incorporate curation during information input, or at least ensure the information is easier to curate post-hoc.

Chapter 6

A Structure for Integrated Curation

As scientific output continues to grow at phenomenal rates [120], it is becoming more challenging to keep track of all relevant advances in a field. The sheer volume of work, as well as the variety of publication venues (the range of journals, conferences, and languages) make it easy to miss a single piece of scientific work. In 2000, it was estimated that to keep up with breast cancer research alone, 130 different journals would need to be subscribed to, and more than 27 papers would need to be read per day [8]. Several solutions have been proposed to address this problem including: subject or object specific databases of curated 'facts' [8, 47, 101, 85], **Structured Digital Abstracts** (SDAs) [40, 126, 70, 86], and interlinking of data with articles [1]. Each of these solutions makes use of advances in computer science to make articles, and the information they contain, more machine readable and thus, crucially, more machine findable. However, each of the solutions also propose a post-hoc form of curation where the creation of additional facts, abstracts, and links occurs at the time of publication, or even after an article has been approved by reviewers. The use of machine learning to make 'fact' extraction automatic and quick is suggested based on the PDF submitted for publication. Yet, based on the issues with post-hoc PDF curation, outlined in Chapter 5, and the increasing integration of digital technology in day-to-day research, we suggest considering building more machine readable papers from the ground up. Such 'future' papers may start to be assembled as soon as research commences.

A 'future' paper might contain links to several specialist repositories (see Figure 6.1). Each repository can store aspects of the information discussed in the paper itself including: sequences, models, experiments, computer code, and the method. Each repository can be easily searched and contains data records with metadata specialized to the type of information stored in the repository. The cross-linking with the paper then allows the specialized knowledge to be placed in a wider context. This structure might also allow metadata from specific repositories to be used to search for and find a particular paper. Yet, for repository metadata to make finding a paper easier, curation must take place. For example, a sequence file that contains information about whether the sequence works in eukaryotes or prokaryotes is both easier to find in the database and can provide the tag organism:eukaryote or organism:prokaryote for the paper to be found under. However, at the time of sequence submission, authors might not remember to add eukaryote or prokaryote tags to their sequences. Curation can help prompt authors to remind them of the useful pieces of information they may have forgotten to add. For curation to be possible, standards are required.

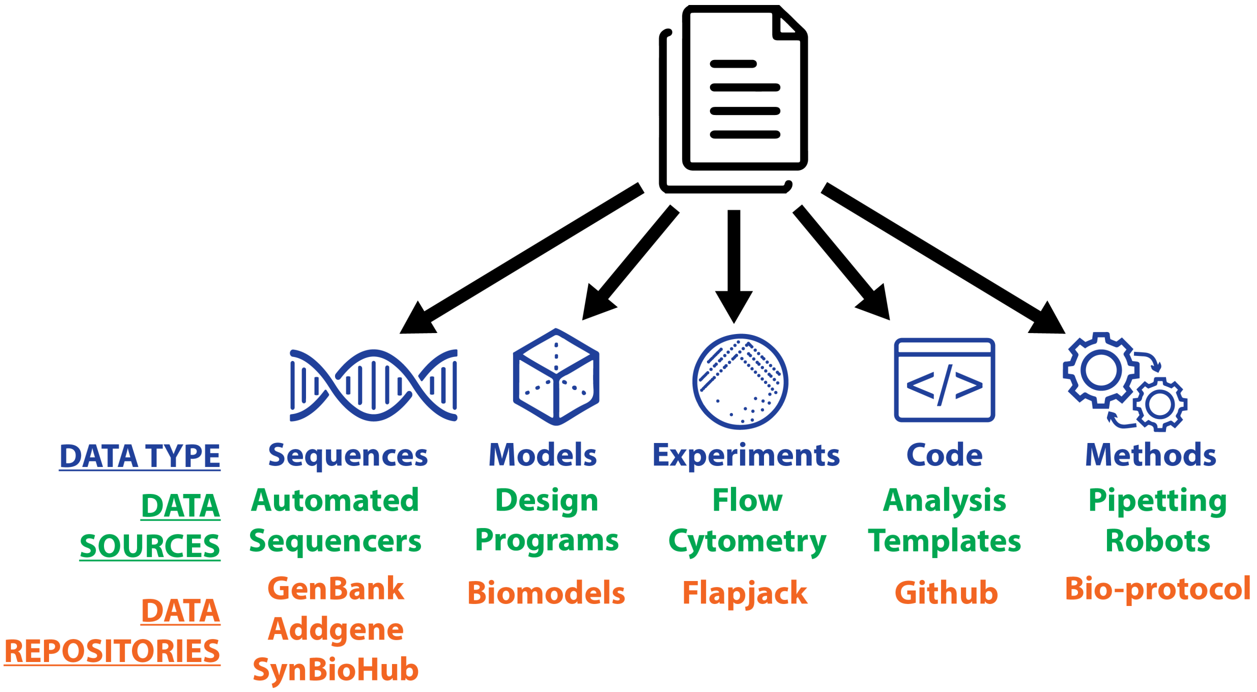


Figure 6.1: The Elements of an Interactive Paper are shown. In blue the names of the different data types, in green the possible research pipeline sources of the data, and in orange repositories where such data might be stored.

Standards may be either *de facto* or *de jure*. A *de facto* standard is one that is not explicitly stated, but it is created by the nature of data collection and/or storage. For example by providing submission via a spreadsheet, Addgene acts as a *de facto* standard despite not explicitly stating anywhere that it is a standard. A *de jure* standard is and explicitly stated or codified standard. For example, Python is a *de jure* standard with many explicit rules to follow for something to be a valid Python file. The standard type can also be classified based on what it regulates, i.e. data encoding, data content, or a mixture of the two. Languages such as SBOL and programming languages are encodings in the same way a file format such as CSV or docx is. They provide rules about how data should be written and what kind of data can be stored (e.g. no JPEGs) but not what data is stored. On the other hand, metadata and minimum information standards provide rules about what information a data record can, or must, contain. For example, MIRIAM (**Minimum Information Required In the Annotation of Models**) provides a list of properties for models such as metadata [160]. There are standards that provide a combination of encoding and data content standards. This generally happens with *de facto* standards, where a tool is importing a standard in the way the data is being collected. Such a tool may have required fields, and convert the input into a standard encoding. In effect, Excel template sheets for the Excel-SBOL Converter, create both a data content standard, as well as use the SBOL encoding standard. The issue with *de facto* standards is that they are often less machine readable than *de jure* standards as they are a result of tool development rather than data management considerations. To prevent the pitfalls observed with the de facto iGEM and Addgene data standards (Chapter 5), and to align with the goal of findable components, we outline a metadata standard for SBOL data. Then we indicate how this standard can, together with SynBioHub plugins and the Excel-SBOL converter, be part of a research workflow with integrated curation. Finally, we examine factors preventing the workflow from being realized, both missing links and researcher reticence.

6.1 The SBOL Data Content Standard

The **SBOL Data Content Standard** (SDCS) is based on the patterns observed while curating the iGEM, Addgene, and ACS data sets (Chapter 5), the needs of the Excel-SBOL Converter (Chapter 4), and the aim of supporting better search functionality (for example via search plugins as mentioned in Chapter 3).

The first draft of SDCS is shown in Table 6.1. The table contains the property and value type expected for a range of properties based on looking at various existing data sources. The data source that suggested the property is shown in the column titled: “Source/Reason”. For example, GenBank ID is useful for cross-linking with the GenBank database, whilst promoter strength was something often mentioned in the iGEM description fields. For properties that have names in other databases, those names are listed in the “Same As” column. For example, Source organism is the same as the Addgene term Species and the UniProt term Organism. Generally, the chosen properties are not free text to limit the inputs. However for some, such as Cloning Method, free text is used as there is no obvious ontology to use. Hopefully, as the standard continues to be used, the cloning methods can be summarized and grounded to machine readable terms.

Table 6.1: Proposed SBOL Data Content Standard. The table contains the property and value type expected for a range of properties based on looking at various existing data sources. The data source that suggested the property is shown in the column titled: "Source/Reason".

Name	**Description**	**Value Type**	**Namespace**	**Term**	**Same As**
GenBank ID	The GenBank ID	URI	GenBank	genbankId	
Addgene ID	The Addgene plasmid Catalog Number	URI	Addgene	addgeneId	
Entrez ID	Entrez Gene ID	URI	Entrez	entrezID	
iGEM ID	Biobrick Name	URI	iGEM	igemID	
Antibiotic Resistance	What antibiotic resistance genes are present	Ontology Term	SDCS	antibioticReistance	Addgene: Bacterial Resistance
Target Organism	The organism for which the construct was designed	Ontology Term	SDCS	targetOrganism	Addgene:Vector Type
Source Organism	The organism from which a gene was taken	Ontology Term	SDCS	sourceOrganism	Addgene:Species, UniProt:Organism, GenBank: Source Organism, Entrez:Organism
Cloning Method	What kind of cloning procedure this component is designed for	Free text	SDCS	cloningMethod	Addgene: Cloning Method
Codes for	Link to protein objects	URI	SDCS	codesFor	UniProt: Protein
References	URL of sources (e.g. publications)	URL	Prov	wasDerivedFrom	Addgene:Publication, Entrez:Primary source, Genbank:(Reference, authors, title, journal), UniPort:Publication
Design Notes	Any notes about how the design was constructed: e.g. Higher GC content, reduced mutagenicity, etc	Free text	SDCS	designNotes	
Mutagenicity	The potential for genetic constructs to undergo mutation	Free text	SDCS	mutagenicity	
Notes	Any further comments	Free text	SDCS	notes	
Paper Topic	The topic of a journal article	Free text	dcterms	subject	
Cell Line	Cell lines used in a journal article	URI	dcterms	subject	
Gene	Genes mentioned in a journal article	URI	dcterms	subject	
Chemical	Chemicals mentioned in a journal article	URI	dcterms	subject	
Species	Species mentioned in a journal article or species annotation on a component definition where it is not clear if it is a source or target organism	Ontology Term	dcterms	subject	
Altered Sequence	How a sequence has been altered from the original source material	Free text	SDCS	alteredSequence	
Growth Temperature	The temperature at which the target organism should be grown (in celsius)	Number	SDCS	growthTemperature	Addgene: Growth Temperature
Growth Instructions	Any notes on how to grow the target organism once modified	Free text	SDCS	growthInstructions	Addgene: Growth Instructions
Copy Number	High or Low?	Free text	SDCS	copyNumber	Addgene: Copy Number
Contributor/Author	The orcid ids of any contributing authors	URI	dc	creator	Addgene: Deposition Lab
Gate Type	What kind of electrical gate it is e.g. NOR	Free text	Cello	gate_type	
Family	What repressible homolog is used e.g. Homolog to TetR	Free text	Cello	family	
Group Name	The output protein group	Free text	Cello	group_name	
Cello n	The Hill coefficient	Number	Cello	notes	
Cello response function	An equation showing the on off thresholds of the circuit response to input, based on [158]	Free text	Cello	response_function	
Cello x off threshold	The concentration at which a promoter output above minimum starts to be seen	Number	Cello	x_off_threshold	
Cello x on threshold	The lowest conentration at which the maximum promoter output is seen	Number	Cello	x_on_threshold	
Cello ymax	Maximum observed promoter output value	Number	Cello	ymax	
Cello ymin	Minimum observed promoter output value	Number	Cello	ymin	
Cello K	The repression threshold (the input value at which the output is half of maximum)	Number	Cello	k	
Burden	Quantifies the burden a construct imposes on a cell based on the method in [41]	Number	SDCS	burden	
Fluoresence	Fluorescence measurements based on the methods described in [15]	Number	SDCS	fluorescence	
RBS Strength	RBS strength, measured using the method in [237]	Number	SDCS	rbsStrength	
Promoter Strength	Measured in RPU (relative promoter units) according to [105]	Number	SDCS	promoterStrength	

For the terms that use ontologies, a separate table has been made to show more detail (Table 6.2). Note the “None Term” and “Other Term” Columns. These columns indicate if the ontology has terms to indicate the property does not exist, or it does exist but there is no specific term for it in the ontology. For example, adding the term synthetic to a genetic sequence indicates that it was not taken from an organism. This is important as, if the sequence has no label, then a user cannot differentiate between a sequence that comes from an organism but is lacking the label, or a sequence that is synthetic. Similarly, for antibiotic resistance, there is an important distinction between the absence of antibiotic resistance and the absence of information about antibiotic resistance. Unfortunately, the **Antibiotic Resistance Ontology** (`https://obofoundry.org/ontology/aro.html`) has no none term. For “Gene” and “Chemical”, a None Term is not important as an article mentioning no genes does not require an annotation that no genes are mentioned. The “Other Term” is similarly important. It indicates that there is information that lacks detail or cannot be expressed using the ontology. For example, it may be that a construct is known to have antibiotic resistance but not to what antibiotic. Alternatively, a construct can have antibiotic resistance but to an antibiotic not currently listed in the ontology. For properties using the NCBI Taxonomy, the use of “unidentified” can indicate information such as: works in non-human primates or does not work in prokaryotes. The use of an “other” term should always be accompanied by further explanation in a notes field.

6.2 Research Workflow with Integrated Curation

Integrating curation into the existing research workflow relies on leveraging automation. A, somewhat idealized, research workflow is shown in Figure 6.2. It divides roughly into 5 stages: literature study, experimental planning, sequence planning, experimental work, and paper writing.

Literature study: A researcher starts with a question that they answer by searching over the different databases and repositories. Example questions include:

- What are some papers about metabolic engineering in *E. coli*? What are some papers

Name	**Description**	**Ontology**	**None Term**	**Other Term**
Antibiotic Resistance	What antibiotic resistance genes are present	Antibiotic Resistance Ontology	N/A	N/A
Target Organism	The organism for which the construct was designed	NCBI Taxonomy	NCBI:txid32630 (synthetic)	NCBI:txid32644 (unidentified)
Source Organism	The organism from which a gene was taken	NCBI Taxonomy	NCBI:txid32630 (synthetic)	NCBI:txid32644 (unidentified)
Cell Line	Cell lines used in a journal article	NCBI Taxonomy	NCBI:txid32630 (synthetic)	NCBI:txid32644 (unidentified)
Gene	Genes mentioned in a journal article	Uniprot	N/A	N/A
Chemical	Chemicals mentioned in a journal article	ChEBI	N/A	N/A
Species	Species mentioned in a journal article or species annotation on a component definition where it is not clear if it is a source or target organism	NCBI Taxonomy	NCBI:txid32630 (synthetic)	NCBI:txid32644 (unidentified)

Table 6.2: This table looks at the SDCS proposed properties with ontologies. Note the “None Term” and “Other Term” Columns. These columns indicate if the ontology has terms to indicate the property does not exist, or it does exist but there is no specific term for it in the ontology.

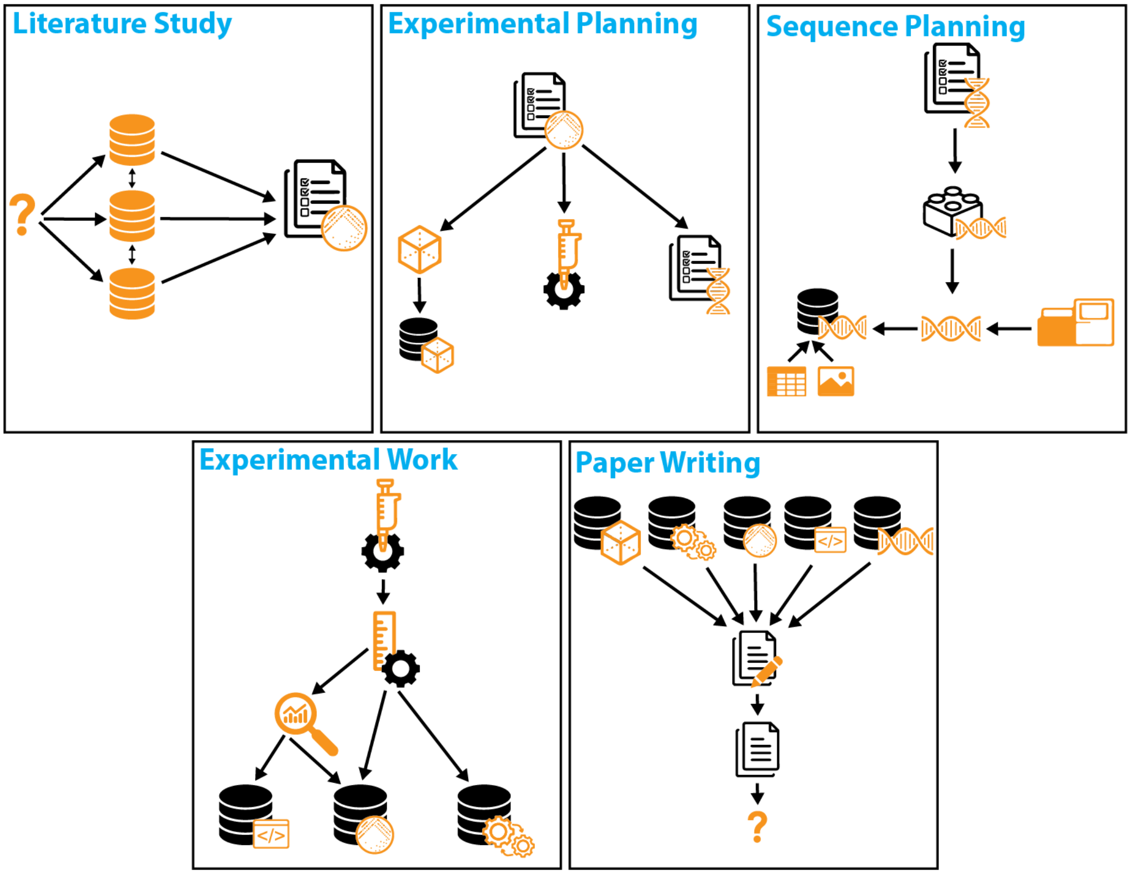


Figure 6.2: Research workflow with integrated automation. The steps in the workflow are shown over several connected panels. **Literature Study**: A researcher starts with a question which they answer by querying the different databases and repositories. The cross-linking of the databases provides the combination of information required to create an experimental plan. **Experimental planning**: This includes models which are stored in modeling databases, the creation of code for pipetting robots and other automated laboratory equipment, and genetic circuit design. **Sequence Planning**: This leads to the creation of genetic parts (which can be used in the experimental work) as well as information about the part sequences. Information about part sequences can also be gained from sequencers. This sequence information can then be deposited into a sequence repository. Information can also be deposited into the repository via manual entry into a GUI or a spreadsheet, rather than an automated output. **Experimental Work**: Automated laboratory equipment includes the pipetting work which can build genetic constructs and add reagents, and measurement work (including flow cytometry). The results of the automated measurements can be analysed using code. The code is stored in a code repository like GitHub whilst the results of analysis, as well as the direct outputs from the measurement work, are stored in an experimental results repository. The code used for the automated pipetting and measurement steps is stored in a methods database. **Paper Writing**: Information from the different repositories can be compiled together to form a "shell" paper. For example, the code for automated laboratory equipment can be converted to a rudimentary methods section. The "shell" paper requires further details added by the author and checking of the automatically provided details. Then the paper can be published and the process can start again with new questions raised by the publication.

about metabolic engineering in *Pseudomonas putida*?

- What are some promoters that can be used in *E.coli*? What are some promoters I can use in *S.cerevisiae*?

- What parts were used to construct the strains in this paper about a fluoride biosensor?

- Which papers use the pLac promoter sequence?

- What promoter is generally used with this CDS?

- What are some inducible promoters for *E. coli*? What are some strong constitutive promoters for *S. cerevisiae*?

These questions are answerable due to the curation of parts which enables efficient searching over cross-linked databases (Figure 6.3). For example, the question about metabolic engineering in *E. coli* might be answered using a natural language search in a journal, after which the search results can be filtered down using ontological suggestions such as a specific *E. coli* strain or synonyms of metabolic engineering. Using the information gained through searching, experimental planning can begin.

Experimental Planning: Experimental planning includes models which are stored in modeling databases. These models may be created using genetic part parameters extracted in the search stage of the research pipeline. The *in silico* experiments then inform the creation of code for automated laboratory equipment (such as pipetting robots), and genetic circuit designs.

Sequence Planning: The genetic circuit design leads to the creation of genetic parts (which can be used in the experimental work) as well as information about the part sequences. Information about preexisting parts can be carried through from the literature search stage. This maintains the provenance chain. Any new parts can be designed *in silico* and implemented, or come from natural sources via automated sequencers. This sequence information can then be deposited into a sequence repository. Note that, sequence information can also be deposited into the repository via manual entry (into a GUI or a spreadsheet), rather than an automated output. This is important

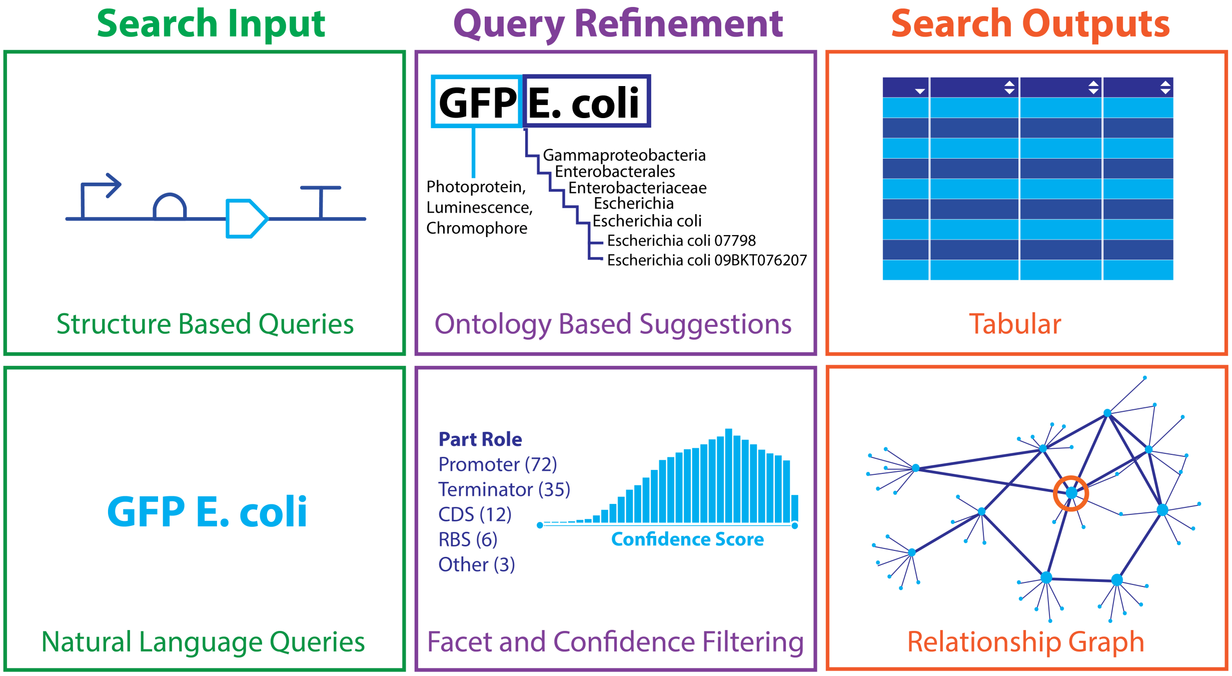


Figure 6.3: Knowledge enabled search overview. **Search input** may use structure-based queries where the types of parts required for a design (e.g. repressible promoter) are outlined, however the exact parts (i.e. lac promoter) are not defined, or natural language queries. **Query refinement** can be carried out via ontology based suggestions for similar, more specific, or broader terms, and facet filtering where a property can be used to refine the search results. Such a property could include a confidence metric for each part based on the quantity and quality of data about the part. **Search outputs** can be either tabular with sorting enabled per column and the ability to choose various columns, or relationship based graphs that depict how closely objects are related.

as it provides a way for researchers with less lab automation to still deposit sequences into the sequence repository. This increases the diversity of people who can add to the repository, including researchers from other areas of biology with less of a focus on automation.

Experimental Work: Automated laboratory equipment includes the pipetting robots (which can build genetic constructs and add reagents), and measurement robots (including flow cytometry). The results of the automated measurements can be analyzed using code. The code is stored in a code repository like GitHub whilst the results of analysis, as well as the direct outputs from the measurement work, are stored in an experimental results repository. The code used for the automated pipetting and measurement steps is stored in a methods database.

Paper Writing: Information from the different repositories can be compiled together to form a "shell" paper. For example, the code for automated laboratory equipment can be converted to a rudimentary methods section. The "shell" paper requires further details added by the author, and checking of the automatically provided details. Then, the paper can be published, and the process can start again with new questions which were raised during the creation of the publication.

In the presented research workflow, many of the transfers of information are from one machine readable format to another. Thus, provenance information should be preservable. This reduces the amount of curation required. When curation is needed, it can be incorporated in the transfer of information from one format to another. An ideal point is when information is added to repositories. For code, this can be done in the form of linting via automatic Github actions, whenever new code is pushed. For methods, experimental results, and sequences this curation can be done in a machine assisted manner. For example, for sequences, machine annotation can occur, using models trained on post-hoc curation data sets (Chapter 5). These results can then be returned in an interactive interface where they can be tweaked by the submission author (Figure 6.4). Similarly, once the paper "shell" has been filled out it can undergo curation to ensure all the relevant cross-linking has occurred. This cross-linking is made easier by the integration of database references into the initial "shell" paper. The combination of the "provenance" annotations, with the machine learning additions can be returned to the submission authors for approval before the "interactive" paper is submitted to a journal (Figure 6.5).

6.3 Realization of the Integrated Curation Workflow

The research workflow with integrated curation is not yet fully realized. However, the development of SynBioHub plugins and the Excel-SBOL Converter have brought the workflow a step closer to realization.

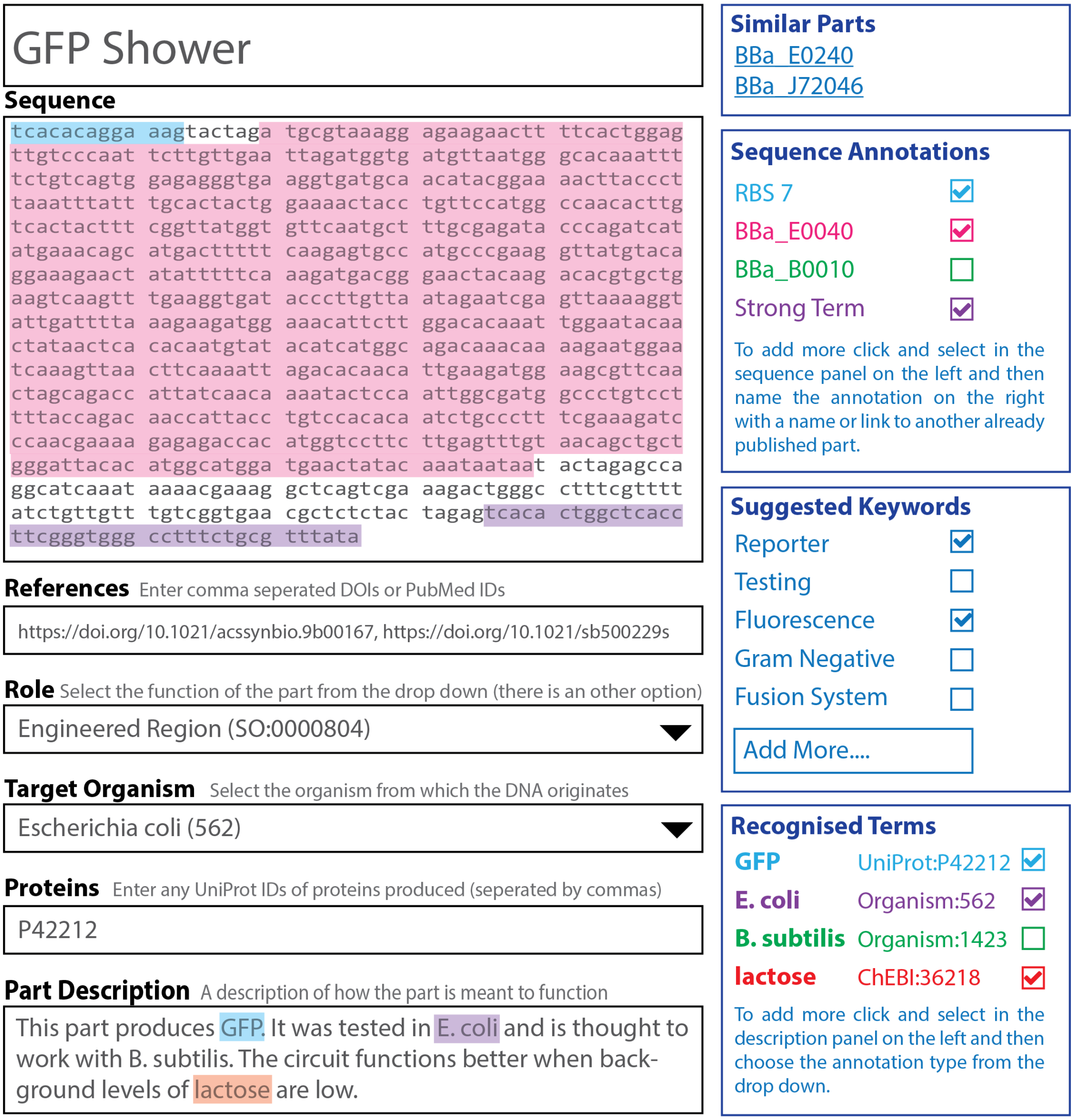
6.3.1 Plugins

SynBioHub plugins have been designed to help realize the research workflow with integrated curation (Figure 6.6). They can help convert data between SBOL and other formats to facilitate data input and retrieval from SynBioHub. Additionally, they can aid in curation for publication data, and query selection and refinement for the literature search.

Search plugins can help with literature search. They allow common queries like "papers with organism x" or "protein y in source organism z" to be saved. This means users do not need to understand or write SPARQL to be able to carry out common complex queries. Additionally, link plugins allow interactions between different databases making it easier to move from one source of information to another. Finally, download plugins enable found information to be pulled from SynBioHub and exported in a format that is compatible with the experimental planning and genetic design software.

When methods have been developed and results generated and analyzed, submit and curation plugins can be used. Submit plugins can convert different data formats to SBOL for data submission. Curation plugins can add additional annotations on the SBOL to make submitted records more complete. The curation interface depicted in Figure 6.4 could be implemented via curation plugins. Similarly, SYNBICT, which was used for sequence annotation in post-hoc curation (Chapter 5), could be incorporated into a curation plugin. An initial draft of this plugin has been developed (`https://github.com/SynBioHub/Plugin-Curation-Synbict`).

Finally, in the publication stage, download plugins could be used to export data into a "shell"



GFP Shower

Sequence

```
tcacacagga aagtactaga tgcgtaaagg agaagaactt ttcactggag
ttgtcccaat tcttgttgaa ttagatggtg atgttaatgg gcacaaattt
tctgtcagtg gagagggtga aggtgatgca acatacggaa aacttaccct
taaatttatt tgcactactg gaaaactacc tgttccatgg ccaacacttg
tcactacttt cggttatggt gttcaatgct ttgcgagata cccagatcat
atgaaacagc atgacttttt caagagtgcc atgcccgaag gttatgtaca
ggaaagaact atatttttca aagatgacgg gaactacaag acacgtgctg
aagtcaagtt tgaaggtgat acccttgtta atagaatcga gttaaaaggt
attgatttta aagaagatgg aaacattctt ggacacaaat tggaatacaa
ctataactca cacaatgtat acatcatggc agacaaacaa aagaatggaa
tcaaagttaa cttcaaaatt agacacaaca ttgaagatgg aagcgttcaa
ctagcagacc attatcaaca aaatactcca attggcgatg gccctgtcct
tttaccagac aaccattacc tgtccacaca atctgccctt tcgaaagatc
ccaacgaaaa gagagaccac atggtccttc ttgagtttgt aacagctgct
gggattacac atggcatgga tgaactatac aaataataat actagagcca
ggcatcaaat aaaacgaaag gctcagtcga aagactgggc ctttcgtttt
atctgttgtt tgtcggtgaa cgctctctac tagagtcaca ctggctcacc
ttcgggtggg cctttctgcg tttata
```

References Enter comma seperated DOIs or PubMed IDs

https://doi.org/10.1021/acssynbio.9b00167, https://doi.org/10.1021/sb500229s

Role Select the function of the part from the drop down (there is an other option)

Engineered Region (SO:0000804) ▼

Target Organism Select the organism from which the DNA originates

Escherichia coli (562) ▼

Proteins Enter any UniProt IDs of proteins produced (seperated by commas)

P42212

Part Description A description of how the part is meant to function

This part produces GFP. It was tested in E. coli and is thought to work with B. subtilis. The circuit functions better when background levels of lactose are low.

Similar Parts

BBa_E0240
BBa_J72046

Sequence Annotations

RBS 7 ☑
BBa_E0040 ☑
BBa_B0010 □
Strong Term ☑

To add more click and select in the sequence panel on the left and then name the annotation on the right with a name or link to another already published part.

Suggested Keywords

Reporter ☑
Testing □
Fluorescence ☑
Gram Negative □
Fusion System □

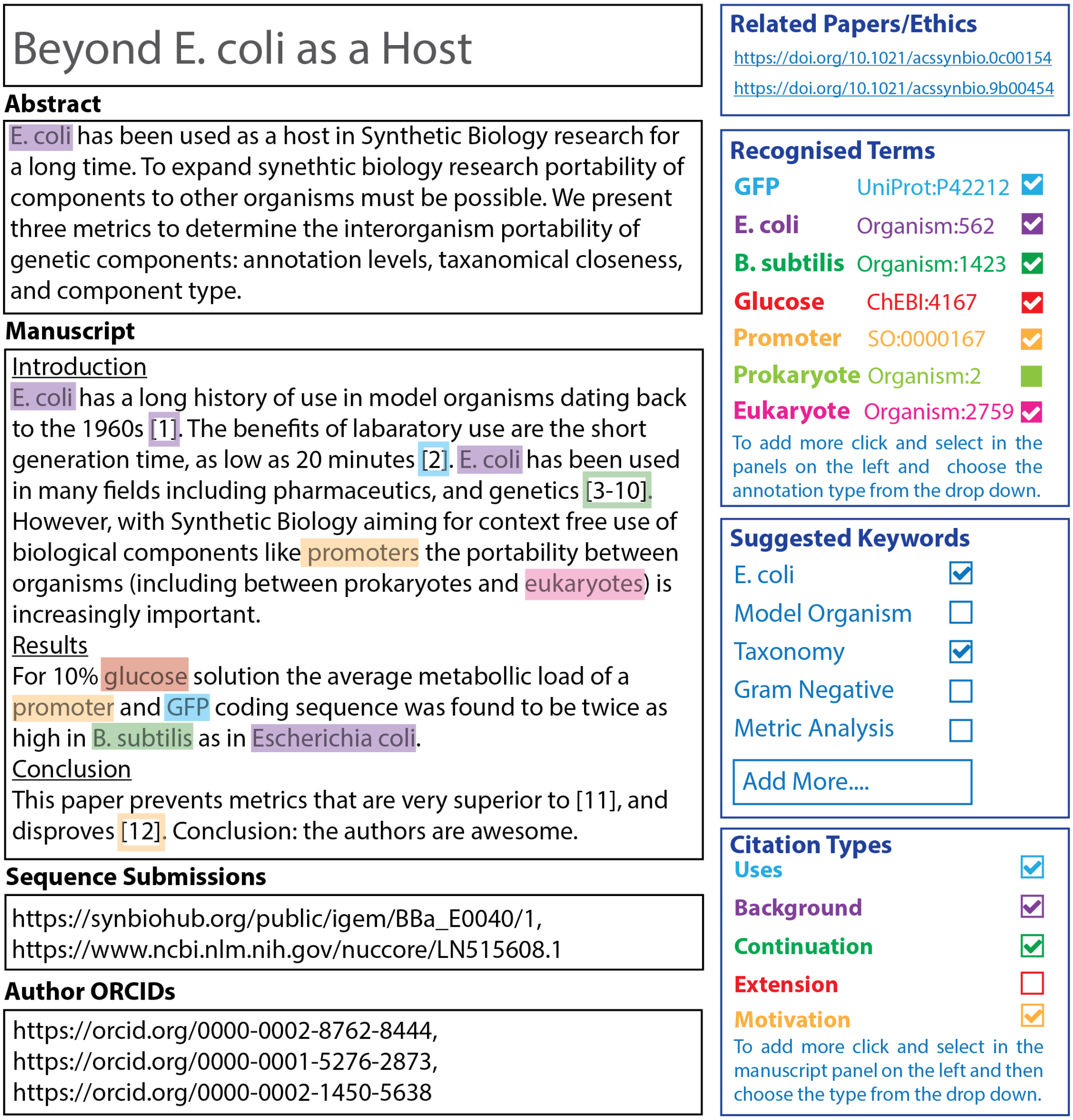
Add More....

Recognised Terms

GFP	UniProt:P42212	☑
E. coli	Organism:562	☑
B. subtilis	Organism:1423	□
lactose	ChEBI:36218	☑

To add more click and select in the description panel on the left and then choose the annotation type from the drop down.

Figure 6.4: Sequence curation interface mock-up. The sequence and part information on the left is initially annotated using a data mining pipeline. The pipeline results are shown on the right. Submission authors can use this interface to correct and approve these annotations. For example, whilst the term *B. subtilis* was recognized in the part description, the author has decided the annotation was incorrect and thus removed it. Similarly, of the suggested keywords only two have been chosen.



Beyond E. coli as a Host

Abstract

E. coli has been used as a host in Synthetic Biology research for a long time. To expand synethtic biology research portability of components to other organisms must be possible. We present three metrics to determine the interorganism portability of genetic components: annotation levels, taxanomical closeness, and component type.

Manuscript

Introduction

E. coli has a long history of use in model organisms dating back to the 1960s [1]. The benefits of labaratory use are the short generation time, as low as 20 minutes [2]. E. coli has been used in many fields including pharmaceutics, and genetics [3-10]. However, with Synthetic Biology aiming for context free use of biological components like promoters the portability between organisms (including between prokaryotes and eukaryotes) is increasingly important.

Results

For 10% glucose solution the average metabollic load of a promoter and GFP coding sequence was found to be twice as high in B. subtilis as in Escherichia coli.

Conclusion

This paper prevents metrics that are very superior to [11], and disproves [12]. Conclusion: the authors are awesome.

Sequence Submissions

https://synbiohub.org/public/igem/BBa_E0040/1,
https://www.ncbi.nlm.nih.gov/nuccore/LN515608.1

Author ORCIDs

https://orcid.org/0000-0002-8762-8444,
https://orcid.org/0000-0001-5276-2873,
https://orcid.org/0000-0002-1450-5638

Related Papers/Ethics

https://doi.org/10.1021/acssynbio.0c00154
https://doi.org/10.1021/acssynbio.9b00454

Recognised Terms

GFP	UniProt:P42212	☑
E. coli	Organism:562	☑
B. subtilis	Organism:1423	☑
Glucose	ChEBI:4167	☑
Promoter	SO:0000167	☑
Prokaryote	Organism:2	■
Eukaryote	Organism:2759	☑

To add more click and select in the panels on the left and choose the annotation type from the drop down.

Suggested Keywords

E. coli	☑
Model Organism	□
Taxonomy	☑
Gram Negative	□
Metric Analysis	□

Add More....

Citation Types

Uses	☑
Background	☑
Continuation	☑
Extension	□
Motivation	☑

To add more click and select in the manuscript panel on the left and then choose the type from the drop down.

Figure 6.5: Publication curation interface mock-up. The text on the left is initially annotated using "provenance" annotations, supplemented by models trained in the text mining pipeline (Chapter 5). Submission authors can use this interface to correct and approve these annotations. For example, the term prokaryote was omitted, and citation 11 was not considered to be worth annotating.

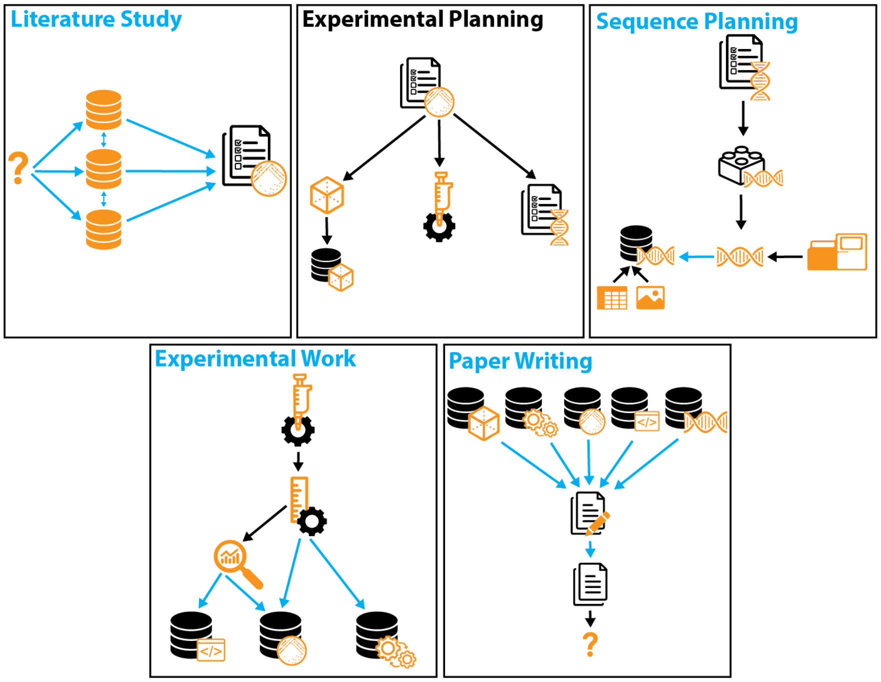


Figure 6.6: Plugins in the research workflow with integrated curation. This figure is the same as Figure 6.2, however, here the blue arrows represent links where plugins might be used. Plugins can help convert data between SBOL and other formats to facilitate data input and retrieval from SynBioHub. Additionally, they can aid in curation for publication data, and query selection and refinement for the literature search.

paper, with its associated metadata annotations. Then, curation plugins could be used to add and modify annotations on the final publication. An example of the proposed interface for publication curation is shown in Figure 6.5.E

6.3.2 Excel-SBOL Converter

The Excel-SBOL Converter (Chapter 4) was also designed to help realize the research workflow (Figure 6.7). In particular, the expansion of the scope of the converter was done to allow different types of data to be read from spreadsheets. This means that not only can it be used to upload sequences to SynBioHub, but also to upload experimental, or possibly even method data. The test case where the Excel-SBOL Converter was used for Flapjack data shows an initial proof of concept. Additionally, many lab robots already take Excel sheets or CSVs as an input, or return these formats as output. Thus, it seems like there is a clear path for the expansion of the Excel-SBOL Converter in other parts of the workflow.

6.4 Challenges for the Integrated Curation Workflow

There are still problems left to overcome before the research workflow with integrated curation (Figure 6.2) is realized. The problems come in two forms: gaps between stages of the workflow, and resistance to uptake.

6.4.1 Gaps Between Workflow Stages

The plugins provide the opportunity to connect different data formats to different repositories (i.e. linked open data). However, though the plugin framework has been created, many of the actually required plugins are not yet realized. For example, the curation plugins, suggested earlier in the chapter, do not exist. Though their elements (such as the machine learning models) exist, they have not been put together and given a front end. Additionally, whilst plugins can be integrated into SynBioHub, they are not currently integrated into any journal websites or other repositories. Thus, any improvements made to search or for data exchange may not be carried through to other

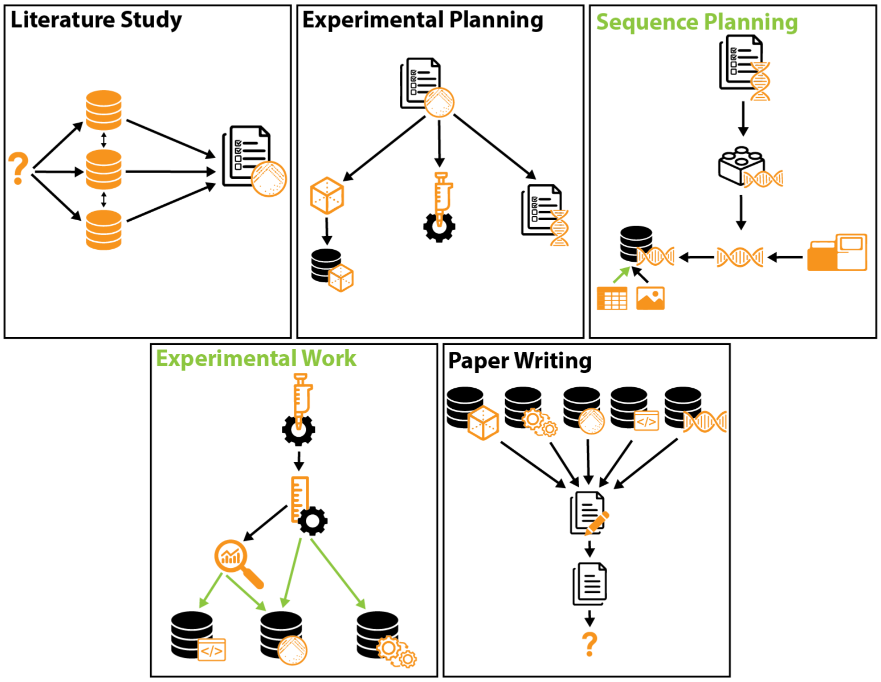


Figure 6.7: Excel-SBOL Converter in the research workflow with integrated curation. This figure is the same as Figure 6.2, however, here the green arrows represent links where the Excel-SBOL Converter might be used. It could be used to upload sequences to SynBioHub, but also to upload experimental, or possibly even method data. The test case where the Excel-SBOL Converter was used for Flapjack data shows an initial proof of concept. Additionally, many lab robots already take Excel sheets or CSVs as an input, or return these formats as output. Thus, it seems like there is a clear path for the expansion of the Excel-SBOL Converter in other parts of the workflow.

data stores.

A second hurdle is methodology storage. Methodology mark-up languages have been proposed [132, 133], however they appear to have gained little traction, possibly due to the papers being in Spanish. Methods are currently stored in journals specifically for methods, such as Nature Methods (`https://www.nature.com/nmeth/`) and Bio-Protocol (`https://bio-protocol.org/Default.aspx`). To enable the automation and curation proposed, this is a major concern that must be addressed.

6.4.2 Resistance to Uptake

Apart from missing elements, this research workflow is also hindered by resistance from the potential user group. This resistance may be due to retraining being required, the extra work expected from curation, or a mismatch between the standards used and the laboratory experience. These problems may be addressed by making the workflow as intuitive as possible, decreasing the work required in stages other than curation (for example the paper publication), and working closely with a variety of wet-lab scientists to continue to develop the tools and standards (both through direct feedback and interaction statistics). This may be insufficient to overcome resistance, thus we suggest further detailed investigation into barriers preventing uptake of the workflow.

Chapter 7

Conclusions

Synthetic biology aims to implement engineering principles of standardization, abstraction, and modularity to genetic engineering [147]. These principles are applied in order to make synthetic biology accessible to a greater number of researchers, as well as to address the reproducibility crisis [97]. To this end, a variety of standards were created across the discipline (including SBOL [180], SBML [92], SEDML [112]) as well as developing tools to help automate the research workflow (SBOLCanvas [211], SynBioHub [138], OpenTrons `https://opentrons.com/`, iBioSim [148], Flapjack [241]). However, as these tools and standards increase the rate of research, the amount of data being produced increases. Thus, synthetic biology must also embrace the principles of library and information sciences in order to allow efficient finding and use of the data being produced. This dissertation proposes extensions and methodologies for the creation of data that adheres to the principles of library sciences. This chapter concludes the dissertation by highlighting the main contributions of this research, which are summarized in Section 7.1, and future directions of this research, which are discussed in Section 7.2.

7.1 Summary

This dissertation presents methods and tools for the creation of a synthetic biology research workflow promoting data reuse via principles of curation and automation. Through its embrace of engineering principles, synthetic biology already leans into many of the principles required for curation and further automation. The use of machine readable standards (and their verification) is already wide spread. The tools and methods proposed build on work done by the synthetic biology community and the library and data science communities. Specifically, this dissertation describes a SynBioHub plugin framework, the Excel-SBOL Converter, post-hoc curation methodologies, and an integrated curation workflow.

The SynBioHub plugin framework de-centralizes SynBioHub development. It allows users to create plugins to adapt SynBioHub to better fit into their existing workflows. Submit and download plugins enable the exchange of new file formats with SynBioHub. An example is a researcher from Newcastle University creating a plugin for the upload of ShortBOL to SynBioHub. Visualization plugins provide users with customization options for the viewing of the data. Several examples have been developed including for molecule 3D structure visualization and more traditional sequence visualization. Search and index plugins provide more ways to query the data. They may be simple implementations of SPARQL queries or include more complex cases such as implementing ElasticSearch or GraphQL. Curation plugins incorporate automated suggestions with user feedback to improve sequence annotations and metadata. Finally, link plugins were conceived to provide seemingly in-place editing of data and models. These six kinds of plugins, together with the examples and documentation enable workflows based on the finding and exchange of data between different standards.

The Excel-SBOL Converter was designed to make it possible to benefit from the SBOL standard without understanding it or being able to write code. Excel was chosen as a specific data format to concentrate on as CSVs and Excel spreadsheets are already widely used in biological research. The flexibility and generalization of the Excel-SBOL Converter was a priority to enable many different kinds of data (experimental, sequences, models, interactions) to be converted to the SBOL standard. Additionally, the flexible column naming and sheet layout allow users to create templates that are very similar to ones they currently use. The idea is to make the change for users as minimal as possible whilst leading to more data collection and introducing more users to SBOL. Similarly, the SBOL-to-Excel Converter is designed to hide the complexities of SBOL and provide an output similar to the input. Thus, users should be able to download data that others created in a spreadsheet format that is familiar to them. The overarching goal of the converter is to increase the compatibility of SBOL and related tools with existing workflows.

To demonstrate the benefits of curation, the post-hoc curation workflow was developed. This workflow aimed to curate several existing data sets and show that they were more useful and searchable once they had been curated. Additionally, the aim was to develop a reusable methodology so others could use post-hoc curation on their previously produced data sets. This goal is not fully realized as the three case studies (iGEM, ACS, and Addgene) showed great variety in the underlying data they contained. The variation was both in terms of how the data was stored in the individual databases and the kinds of labeling used in free text fields. This work did produce three partially curated data sets, and a set of principles to produce data more conducive to post-hoc curation. Additionally, it supported the idea that the integration of curation into existing workflows is more useful than the creation of new, and separate, curation workflows.

Finally, a synthetic biology research workflow promoting data reuse via principles of curation and automation is proposed. The methodology of this workflow relies on lessons learned from post-hoc curation of data sets. This includes the proposal of a SBOL Data Content Standard (SDCS), as well as suggestions about the implementation of the workflow via: leveraging of the machine learning models generated in the post-hoc curation, the Excel-SBOL Converter, and SynBioHub plugins. This workflow is designed to be accessible to users who do not commit to the full workflow, but only use parts of it (including users outside of synthetic biology). Additionally, it is intended to integrate with existing practices in biological research. The realization of this workflow should increase the ease of biological research and promote the reuse of previously generated data.

7.2 Future Work

While the research presented is promising, there is room for improvements. Future work sections were included in each of the chapters, however this section goes over longer term or bigger picture future work.

7.2.1 Plugins

Several kinds of plugins were proposed: submit, visualization, download, search, curation, index, and link. Whilst the majority of these are fully developed, some require further thought. Additionally, not all have been incorporated into SynBioHub. As SynBioHub3 is being developed, further work should be done to consider how plugins are integrated with SynBioHub (the front or back-end) and how the plugins might handle multi-file requests (currently submit does, but download does not as it was previously called from a single part page).

7.2.2 Search

Currently there is a search plugin framework, but it is not integrated with SynBioHub. Integration with SynBioHub includes being able to use returned facts to create follow up queries. Potentially, the type of query run could also be logged. This would provide information as to the kinds of queries that are most used and thus help shape future search development. Additionally, there are no functioning example plugins. Example plugins could be created to answer the example queries given in Chapter 6, and to incorporating GraphQL queries into SynBioHub. Furthermore, SBOLExplorer could be rebuilt to be less complicated to maintain, and fit better with the proposed plugin framework.

7.2.3 User Interface Development

A common theme across future work is the requirement of user interfaces. GUI interfaces are needed for curation plugins, reuse data, and possibly specialist search result interfaces (e.g. graph data representations). Some work has been done, as part of search and curation plugins, to develop a framework for easy HTML form generation. However, this work could be expanded to make generating simple user interfaces easier.

7.2.4 Excel Templates

An initial set of templates was created for the Excel-SBOL Converter, however further templates should be developed to fit more data types and workflows. Additionally, developing templates together with laboratory researchers will help increase documentation blind spots. Furthermore, templates should be developed to help enforce the SDCS.

7.2.5 Further Curation Libraries

Curation libraries were used for SYNBICT annotation in the post-hoc curation pipeline. These same libraries could be reused for SYNBICT curation plugins. However, expanding the range and number of these libraries would make curation more effective. New libraries could include further model organisms, and the most common parts observed in the post-hoc curation case studies.

7.2.6 Further Curation of iGEM, ACS, and Addgene Libraries

For the post-hoc curation, the initial aim was to generate well annotated libraries. Due to the difficulty of this task, the libraries generated were not as well-annotated as hoped for. It would be good to go back and annotate the libraries based on the SDCS proposed. These collections with more detailed annotations could then also feed back into the curation of sub-component annotations in the integrated curation workflow.

7.2.7 SBOL Data Content Standard Extension

The SDCS is proposed. However, it is a first draft that still has clear gaps. These gaps include experimental conditions (e.g. medium and machine used) and modeling parameters. The latter is particularly important as to close the design, build, test, learn cycle, the part information submitted must be sufficient to produce models for *in silico* modeling before *in vivo* experimentation. Adding modeling parameters to SDCS, highlights the importance of the parameters. This may induce more experimental researchers to capture modeling parameters in their experiments. The ability of SDCS to influence experimental design is increased if SDCS is incorporated into the SBOL workflows. Additionally, this provides testing of the standard which will naturally highlight areas for expansion (i.e. further properties to add). Additionally, preforming a post-hoc curation exercise on data submitted with SDCS compliance will indicate which fields might require restricted entry or clearer None and Other terms.

7.2.8 Framework to Assess Data Reuse

Each of the measures proposed so far hopes to promote data reuse. However, the methods to assess reuse over time are not currently available. Thus, a workflow could be developed to assess data reuse. This could be a simple set of SPARQL queries looking at the number of times components are used that are run at regular time intervals. More complex queries could be developed looking for improvements in sub-component annotations and the effect this has on component reuse statistics. Finally, a GUI interface could be developed to return visualizations of the reuse measurements.

7.2.9 Community Uptake

A standard relies on enforcement. This can be *de facto* enforcement via community normailization or *de jure* enforcement via institutions like ACS or NSF. To benefit from the proposed integrated curation workflow the underlying standards must be enforced. This requires reducing the barriers to using the standards, increasing the benefits to using them, and obtaining *de jure* enforcement. To achieve the radical shift in publishing workflow proposed in Chapter 6, the community has to be convinced of the benefits of the workflow and early adopters must be curated. This requires a lot of community outreach and involvement.

Bibliography

[1] IJsbrand Jan Aalbersberg, Sophia Atzeni, Hylke Koers, Beate Specker, and Elena Zudilova-Seinstra. Bringing digital science deep inside the scientific article: the elsevier article of the future project. LIBER Quarterly: The Journal of the Association of European Research Libraries, 23(44):274–299, Apr 2014.

[2] Arda Akdemir and Tetsuo Shibuya. Analyzing the Effect of Multi-task Learning for Biomedical Named Entity Recognition. arXiv:2011.00425 [cs], November 2020. arXiv: 2011.00425.

[3] G. Akrivas, M. Wallace, G. Andreou, G. Stamou, and S. Kollias. Context-sensitive semantic query expansion. In Proceedings 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002), page 109–114, Sep 2002.

[4] allenai. Spacy models for biomedical text processing. `https://allenai.github.io/scispacy/`.

[5] G. Alterovitz, T. Muso, and M. F. Ramoni. The challenges of informatics in synthetic biology: from biomolecular networks to artificial organisms. Briefings in Bioinformatics, 11(1):80–95, Jan 2010.

[6] Andrawaag. Sulab/genewiki-shex.

[7] Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov, Sarunas Marciuska, and Dmitriy Zheleznyakov. Faceted search over ontology-enhanced rdf data. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14, page 939–948. Association for Computing Machinery, Nov 2014.

[8] Rudeina A. Baasiri, Stanley R. Glasser, David L. Steffen, and David A. Wheeler. The breast cancer gene database: a collaborative information resource. Oncogene, 18(56):7958–7965, Dec 1999.

[9] Hasan Baig, Pedro Fontanarrosa, Vishwesh Kulkarni, James Alastair McLaughlin, Prashant Vaidyanathan, Bryan Bartley, Jacob Beal, Matthew Crowther, Thomas E. Gorochowski, Raik Grünberg, Goksel Misirli, James Scott-Brown, Ernst Oberortner, Anil Wipat, and Chris J. Myers. Synthetic biology open language (sbol) version 3.0.0. Journal of Integrative Bioinformatics, 17(2–3), Jun 2020.

[10] Wendy Baker, Alexandra van den Broek, Evelyn Camon, Pascal Hingamp, Peter Sterk, Guenter Stoesser, and Mary Ann Tuli. The embl nucleotide sequence database. Nucleic Acids Research, 28(1):19–23, Jan 2000.

[11] Ziv Bar-Yossef and Naama Kraus. Context-sensitive query auto-completion. In Proceedings of the 20th international conference on World wide web, WWW '11, page 107–116. Association for Computing Machinery, Mar 2011.

[12] Federico Barone, Francisco Dorr, Luciano E Marasco, Sebastián Mildiner, Inés L Patop, Santiago Sosa, Lucas G Vattino, Federico A Vignale, Edgar Altszyler, Benjamin Basanta, and et al. Design and evaluation of an incoherent feed-forward loop for an arsenic biosensor based on standard igem parts. Synthetic Biology, 2(ysx006), Jan 2017.

[13] Bryan A. Bartley, Kiri Choi, Meher Samineni, Zach Zundel, Tramy Nguyen, Chris J. Myers, and Herbert M. Sauro. pysbol: A python package for genetic design automation and standardization. ACS Synthetic Biology, 8(7):1515–1518, Jul 2019.

[14] G. I. Baylor. Up, up and away. Proc. Roy. Soc., London A, 294:456–475, 1959.

[15] Jacob Beal, Traci Haddock-Angelli, Markus Gershater, Kim de Mora, Meagan Lizarazo, Jim Hollenhorst, and Randy Rettberg. Reproducibility of fluorescent expression from engineered biological constructs in e. coli. PLoS ONE, 11(3):e0150182, Mar 2016.

[16] Jacob Beal, Tramy Nguyen, Thomas E. Gorochowski, Angel Goñi-Moreno, James Scott-Brown, James Alastair McLaughlin, Curtis Madsen, Benjamin Aleritsch, Bryan Bartley, Shyam Bhakta, Mike Bissell, Sebastian Castillo Hair, Kevin Clancy, Augustin Luna, Nicolas Le Novère, Zach Palchick, Matthew Pocock, Herbert Sauro, John T. Sexton, Jeffrey J. Tabor, Christopher A. Voigt, Zach Zundel, Chris Myers, and Anil Wipat. Communicating structure and function in synthetic biology diagrams. ACS Synthetic Biology, 8(8):1818–1825, 2019. PMID: 31348656.

[17] Andre Bechara, Maria Luiza Machado, and Vanessa Braganholo. Applying biomedical ontologies on semantic query expansion. Nature Precedings, page 1–1, Aug 2009.

[18] Carolin Becker-Leifhold and Samira Iran. Collaborative fashion consumption – drivers, barriers and future pathways. Journal of Fashion Marketing and Management: An International Journal, 22(2):189–208, Jan 2018.

[19] Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har'El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. Beyond basic faceted search. In Proceedings of the 2008 International Conference on Web Search and Data Mining, WSDM '08, page 33–44. Association for Computing Machinery, Feb 2008.

[20] Dennis A. Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. Genbank. Nucleic Acids Research, 41(Database issue):D36–42, Jan 2013.

[21] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. Genbank. Nucleic Acids Research, 39(suppl_1):D32–D37, Jan 2011.

[22] Frank T. Bergmann, Richard Adams, Stuart Moodie, Jonathan Cooper, Mihai Glont, Martin Golebiewski, Michael Hucka, Camille Laibe, Andrew K. Miller, David P. Nickerson, and et al. Combine archive and omex format: one file to share all information to reproduce a modeling project. BMC bioinformatics, 15:369, Dec 2014.

[23] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. Nucleic Acids Research, 28(1):235–242, Jan 2000.

[24] Ivelize R Bernardo, Matheus S Mota, and André Santanchè. Extracting and semantically integrating implicit schemas from multiple spreadsheets of biology based on the recognition of their nature. Journal of Information and Data Management, 4(2):104–104, 2013.

[25] David M. Blei and John D. Lafferty. Correlated topic models. In Proceedings of the 18th International Conference on Neural Information Processing Systems, NIPS'05, page 147–154, Cambridge, MA, USA, 2005. MIT Press.

[26] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. the Journal of machine Learning research, 3:993–1022, 2003.

[27] Guest Blogger. Identifying sequence elements with snapgene's feature database.

[28] Jerven Bolleman, Alain Gateau, Sebastien Gehant, and Nicole Redaschi. Provenance and evidence in uniprotkb. arXiv:1012.1660 [cs], Dec 2010. arXiv: 1012.1660.

[29] Ian R. Booth. Sysmo: back to the future. Nature Reviews Microbiology, 5(8):566–566, Aug 2007.

[30] Christine L. Borgman. The conundrum of sharing research data. Journal of the American Society for Information Science and Technology, 63(6):1059–1078, 2012.

[31] Philip E. Bourne, Jon R. Lorsch, and Eric D. Green. Perspective: Sustaining the big-data ecosystem. Nature, 527(7576):S16–S17, Nov 2015.

[32] Tim Bray, Charles Frankston, and Ashok Malhotra. Document content description for xml, Jul 1998.

[33] Alvis Brazma, Maria Krestyaninova, and Ugis Sarkans. Standards for systems biology. Nature Reviews Genetics, 7(88):593–605, Aug 2006.

[34] Lukas Buecherl, Riley Roberts, Pedro Fontanarrosa, Payton J. Thomas, Jeanet Mante, Zhen Zhang, and Chris J. Myers. Stochastic hazard analysis of genetic circuits in ibiosim and stamina. ACS Synthetic Biology, 10(10):2532–2540, Oct 2021.

[35] Robert C. Cannon, Padraig Gleeson, Sharon Crook, Gautham Ganapathy, Boris Marin, Eugenio Piasini, and R. Angus Silver. Lems: a language for expressing complex biological models in concise and hierarchical form and its use in underpinning neuroml 2. Frontiers in Neuroinformatics, 8:79, 2014.

[36] Barry Canton, Anna Labno, and Drew Endy. Refinement and standardization of synthetic biological parts and devices. Nature Biotechnology, 26(7):787–793, Jul 2008.

[37] Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. Context-aware query classification. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09, page 3–10. Association for Computing Machinery, Jul 2009.

[38] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08, page 875–883. Association for Computing Machinery, Aug 2008.

[39] José María Cavanillas, Edward Curry, and Wolfgang Wahlster. New Horizons for a Data-Driven Economy. Springer International Publishing, 2016.

[40] Arnaud Ceol, Andrew Chatr-Aryamontri, Luana Licata, and Gianni Cesareni. Linking entries in protein interaction database to structured text: The febs letters experiment. FEBS Letters, 582(8):1171–1177, Apr 2008.

[41] Francesca Ceroni, Rhys Algar, Guy-Bart Stan, and Tom Ellis. Quantifying cellular capacity identifies gene expression designs with reduced burden. Nature Methods, 12(5):415–418, May 2015.

[42] Ye Chen, Shuyi Zhang, Eric M. Young, Timothy S. Jones, Douglas Densmore, and Christopher A. Voigt. Genetic circuit design automation for yeast. Nature Microbiology, 5(11):1349–1360, Nov 2020.

[43] Ying-Ja Chen, Peng Liu, Alec A. K. Nielsen, Jennifer A. N. Brophy, Kevin Clancy, Todd Peterson, and Christopher A. Voigt. Characterization of 582 natural and synthetic terminators and quantification of their design constraints. Nature Methods, 10(7):659–664, July 2013. Number: 7 Publisher: Nature Publishing Group.

[44] Peter J. A. Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. Nucleic Acids Research, 38(6):1767–1771, Apr 2010.

[45] Samuel Colvin. pydantic, May 2017.

[46] Kim Y Hiller Connell. Exploration of second-hand apparel acquisition behaviors and barriers. In itaa 2009 Proceedings, page 3, 2009.

[47] The UniProt Consortium. Uniprot: a hub for protein information. Nucleic Acids Research, 43(D1):D204–D212, Jan 2015.

[48] The UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. Nucleic Acids Research, 47(D1):D506 – D515, Jan 2019.

[49] Mark J. Costello, William K. Michener, Mark Gahegan, Zhi-Qiang Zhang, and Philip E. Bourne. Biodiversity data should be published, cited, and peer reviewed. Trends in Ecology & Evolution, 28(8):454–461, Aug 2013.

[50] Mélanie Courtot, Nick Juty, Christian Knüpfer, Dagmar Waltemath, Anna Zhukova, Andreas Dräger, Michel Dumontier, Andrew Finney, Martin Golebiewski, Janna Hastings, Stefan Hoops, Sarah Keating, Douglas B Kell, Samuel Kerrien, James Lawson, Allyson Lister, James Lu, Rainer Machne, Pedro Mendes, Matthew Pocock, Nicolas Rodriguez, Alice Villeger, Darren J Wilkinson, Sarala Wimalaratne, Camille Laibe, Michael Hucka, and Nicolas Le Novère. Controlled vocabularies and semantics in systems biology. Molecular Systems Biology, 7(1):543, Jan 2011.

[51] Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. A neural network multi-task learning approach to biomedical named entity recognition. BMC Bioinformatics, 18:368, August 2017.

[52] M. E. Crow. Aerodynamic sound emission as a singular perturbation problem. Stud. Appl. Math., 29:21–44, 1968.

[53] Fiona Cunningham, Barry Moore, Nicole Ruiz-Schultz, Graham RS Ritchie, and Karen Eilbeck. Improving the sequence ontology terminology for genomic variant annotation. Journal of Biomedical Semantics, 6(1):32, Jul 2015.

[54] Renata Gonçalves Curty, Kevin Crowston, Alison Specht, Bruce W. Grant, and Elizabeth D. Dalton. Attitudes and norms affecting scientists' data reuse. PLOS ONE, 12(12):e0189288, Dec 2017.

[55] Davis Daniel, Jane Burry, and Mark Burry. Untangling parametric schemata: Enhancing collaboration through modular programming. CAAD Futures, Jan 2011.

[56] Kirill Degtyarenko, Paula de Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. Chebi: a database and ontology for chemical entities of biological interest. Nucleic Acids Research, 36(suppl_1):D344–D350, Jan 2008.

[57] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805, 2018.

[59] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 [cs], May 2019. arXiv: 1810.04805.

[60] Julian D. Dole. Perturbation Methods in Applied Mathematics. Winsdell Publishing Company, 1967.

[61] Michael Droettbooom. Understanding json schema, Dec 2020.

[62] Karen Eilbeck, Suzanna E. Lewis, Christopher J. Mungall, Mark Yandell, Lincoln Stein, Richard Durbin, and Michael Ashburner. The sequence ontology: a tool for the unification of genome annotations. Genome Biology, 6(5):R44, Apr 2005.

[63] Karen Eilbeck, Suzanna E Lewis, Christopher J Mungall, Mark Yandell, Lincoln Stein, Richard Durbin, and Michael Ashburner. The sequence ontology: a tool for the unification of genome annotations. Genome biology, 6(5):R44, 2005.

[64] Joshua D. Eisenberg, Deya Banisakher, Maria Presa, Kalli Unthank, Mark A. Finlayson, Rene Price, and Shu-Ching Chen. Toward semantic search for the biogeochemical literature. In 2017 IEEE International Conference on Information Reuse and Integration (IRI), page 517–525, Aug 2017.

[65] J. S. Fabnis, H. J. Giblet, and H. McDormand. Navier-stokes analysis of solid rocket motor internal flow. J. Prop. and Power, 2:157–164, 1980.

[66] James R. Faeder, Michael L. Blinov, and William S. Hlavacek. Rule-Based Modeling of Biochemical Systems with BioNetGen, page 113–167. Methods in Molecular Biology. Humana Press, 2009.

[67] Dawn Field, George Garrity, Tanya Gray, Norman Morrison, Jeremy Selengut, Peter Sterk, Tatiana Tatusova, Nicholas Thomson, Michael J Allen, Samuel V Angiuoli, Michael Ashburner, Nelson Axelrod, Sandra Baldauf, Stuart Ballard, Jeffrey Boore, Guy Cochrane, James Cole, Peter Dawyndt, Paul De Vos, Claude dePamphilis, Robert Edwards, Nadeem Faruque, Robert Feldman, Jack Gilbert, Paul Gilna, Frank Oliver Glöckner, Philip Goldstein, Robert Guralnick, Dan Haft, David Hancock, Henning Hermjakob, Christiane Hertz-Fowler, Phil Hugenholtz, Ian Joint, Leonid Kagan, Matthew Kane, Jessie Kennedy, George Kowalchuk, Renzo Kottmann, Eugene Kolker, Saul Kravitz, Nikos Kyrpides, Jim Leebens-Mack, Suzanna E Lewis, Kelvin Li, Allyson L Lister, Phillip Lord, Natalia Maltsev, Victor Markowitz, Jennifer Martiny, Barbara Methe, Ilene Mizrachi, Richard Moxon, Karen Nelson, Julian Parkhill, Lita Proctor, Owen White, Susanna-Assunta Sansone, Andrew Spiers, Robert Stevens, Paul Swift, Chris Taylor, Yoshio Tateno, Adrian Tett, Sarah Turner, David Ussery, Bob Vaughan, Naomi Ward, Trish Whetzel, Ingio San Gil, Gareth Wilson, and Anil Wipat. The minimum information about a genome sequence (migs) specification. Nature biotechnology, 26(5):541–547, May 2008.

[68] Pedro Fontanarrosa. AUTOMATED GENERATION OF DYNAMIC MODELS FOR GENETIC REGULATORY NETWORKS. PhD thesis, University of Utah, Dec 2019.

[69] International Society for Biocuration. Biocuration: Distilling data into knowledge. PLoS Biology, 16(4), Apr 2018.

[70] Ad Hoc Working Group for Critical Appraisal of the Medical Literature. A proposal for more informative abstracts of clinical articles. Annals of Internal Medicine, 106(4):598–604, Apr 1987.

[71] Katharina Frey, Alenka Hafner, and Boas Pucker. The reuse of public datasets in the life sciences: Potential risks and rewards. Preprints, Feb 2020.

[72] Brett M. Frischmann, Michael J. Madison, and Katherine J. Strandburg. Governing Medical Knowledge Commons. Cambridge University Press, Oct 2017. Google-Books-ID: Ai02DwAAQBAJ.

[73] Michal Galdzicki, Deepak Chandran, Alec Nielsen, Jason Morrison, Mackenzie Cowell, Raik Grünberg, Sean Sleight, and Herbert Sauro. Provisional biobrick language (pobol), May 2009. Accepted: 2009-05-15T18:05:32Z.

[74] Michal Galdzicki, Kevin P. Clancy, Ernst Oberortner, Matthew Pocock, Jacqueline Y. Quinn, Cesar A. Rodriguez, Nicholas Roehner, Mandy L. Wilson, Laura Adam, J. Christopher Anderson, and et al. The synthetic biology open language (sbol) provides a community standard for communicating designs in synthetic biology. Nature Biotechnology, 32(6):545–550, Jun 2014.

[75] Michal Galdzicki, Kevin P Clancy, Ernst Oberortner, Matthew Pocock, Jacqueline Y Quinn, Cesar A Rodriguez, Nicholas Roehner, Mandy L Wilson, Laura Adam, J Christopher Anderson, et al. The synthetic biology open language (sbol) provides a community standard for communicating designs in synthetic biology. Nature biotechnology, 32(6):545–550, 2014.

[76] John H. Gennari, Maxwell L. Neal, Michal Galdzicki, and Daniel L. Cook. Multiple ontologies in action: composite annotations for biosimulation models. Journal of Biomedical Informatics, 44(1):146–154, Feb 2011.

[77] Padraig Gleeson, Sharon Crook, Robert C. Cannon, Michael L. Hines, Guy O. Billings, Matteo Farinella, Thomas M. Morse, Andrew P. Davison, Subhasis Ray, Upinder S. Bhalla, and et al. Neuroml: A language for describing data driven models of neurons and networks with a high degree of biological detail. PLOS Computational Biology, 6(6):e1000815, Jun 2010.

[78] Michael Gorman. Five new laws of librarianship. American Libaries, Sep 1995.

[79] NISO Framework Working Group. A framework of guidance for building good digital collections - 3rd edition. National Information Standards Organization (NISO), page 100, Dec 2007.

[80] R. Guha, Rob McCool, and Eric Miller. Semantic search. In Proceedings of the 12th international conference on World Wide Web, WWW '03, page 700–709. Association for Computing Machinery, May 2003.

[81] F. Guillot and Z. Javalon. Acoustic boundary layers in propellant rocket motors. J. Prop. and Power, 5:331–339, 1989.

[82] Timothy S. Ham, Zinovii Dmytriv, Hector Plahar, Joanna Chen, Nathan J. Hillson, and Jay D. Keasling. Design, implementation and practice of jbei-ice: an open source biological part registry platform and tools. Nucleic Acids Research, 40(18):e141, Oct 2012.

[83] Aniko Hannak, Piotr Sapiezynski, Arash Molavi Kakhki, Balachander Krishnamurthy, David Lazer, Alan Mislove, and Christo Wilson. Measuring personalization of web search. In Proceedings of the 22nd international conference on World Wide Web, WWW '13, page 527–538. Association for Computing Machinery, May 2013.

[84] Todd W. Harris, Igor Antoshechkin, Tamberlyn Bieri, Darin Blasiar, Juancarlos Chan, Wen J. Chen, Norie De La Cruz, Paul Davis, Margaret Duesbury, Ruihua Fang, Jolene Fernandes, Michael Han, Ranjana Kishore, Raymond Lee, Hans-Michael Müller, Cecilia Nakamura, Philip Ozersky, Andrei Petcherski, Arun Rangarajan, Anthony Rogers, Gary Schindelman, Erich M. Schwarz, Mary Ann Tuli, Kimberly Van Auken, Daniel Wang, Xiaodong Wang, Gary Williams, Karen Yook, Richard Durbin, Lincoln D. Stein, John Spieth, and Paul W. Sternberg. Wormbase: a comprehensive resource for nematode research. Nucleic Acids Research, 38(suppl_1):D463–D467, Jan 2010.

[85] Todd W. Harris, Igor Antoshechkin, Tamberlyn Bieri, Darin Blasiar, Juancarlos Chan, Wen J. Chen, Norie De La Cruz, Paul Davis, Margaret Duesbury, Ruihua Fang, Jolene Fernandes, Michael Han, Ranjana Kishore, Raymond Lee, Hans-Michael Müller, Cecilia Nakamura, Philip Ozersky, Andrei Petcherski, Arun Rangarajan, Anthony Rogers, Gary Schindelman,

Erich M. Schwarz, Mary Ann Tuli, Kimberly Van Auken, Daniel Wang, Xiaodong Wang, Gary Williams, Karen Yook, Richard Durbin, Lincoln D. Stein, John Spieth, and Paul W. Sternberg. Wormbase: a comprehensive resource for nematode research. Nucleic Acids Research, 38(suppl_1):D463–D467, Jan 2010.

[86] James Hartley. Current findings from research on structured abstracts: an update. Journal of the Medical Library Association: JMLA, 102(3):146–148, Jul 2014.

[87] Benjamin Hatch, Linhao Meng, Jeanet Mante, James A. McLaughlin, James Scott-Brown, and Chris J. Myers. Visbol2—improving web-based visualization for synthetic biology designs. ACS Synthetic Biology, 10(8):2111–2115, Aug 2021.

[88] John R Hayes. Modular programming in c. Embedded Systems Programming, 14(13):18–24, 2001.

[89] Lynette Hirschman, Karën Fort, Stéphanie Boué, Nikos Kyrpides, Rezarta Islamaj Doğan, and Kevin Bretonnel Cohen. Crowdsourcing and curation: perspectives from biology and natural language processing. Database, 2016, Jan 2016.

[90] Michal Horejsek. Fast json schema for python — fastjsonschema documentation, 2016.

[91] Doug Howe, Maria Costanzo, Petra Fey, Takashi Gojobori, Linda Hannick, Winston Hide, David P. Hill, Renate Kania, Mary Schaeffer, Susan St Pierre, Simon Twigger, Owen White, and Seung Yon Rhee. The future of biocuration. Nature, 455(7209):47–50, Sep 2008.

[92] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, and et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. Bioinformatics (Oxford, England), 19(4):524–531, Mar 2003.

[93] Michael Hucka, David P. Nickerson, Gary D. Bader, Frank T. Bergmann, Jonathan Cooper, Emek Demir, Alan Garny, Martin Golebiewski, Chris J. Myers, Falk Schreiber, and et al. Promoting coordinated development of community-based information standards for modeling in biology: The combine initiative. Frontiers in Bioengineering and Biotechnology, 3:19, 2015.

[94] IBM. SPSS Statistics. download from vendor site, 2012. version 21.

[95] Jon Ison, Matúš Kalaš, Inge Jonassen, Dan Bolser, Mahmut Uludag, Hamish McWilliam, James Malone, Rodrigo Lopez, Steve Pettifer, and Peter Rice. Edam: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. Bioinformatics, 29(10):1325–1332, May 2013.

[96] Sonya V. Iverson, Traci L. Haddock, Jacob Beal, and Douglas M. Densmore. CIDAR MoClo: Improved MoClo Assembly Standard and New E. coli Part Library Enable Rapid Combinatorial Design for Synthetic and Traditional Biology. ACS Synthetic Biology, 5(1):99–103, January 2016. Publisher: American Chemical Society.

[97] Mathew M Jessop-Fabre and Nikolaus Sonnenschein. Improving reproducibility in synthetic biology. Frontiers in Bioengineering and Biotechnology, 7, 2019.

[98] Abayomi Oluwanbe Johnson, Miriam Gonzalez-Villanueva, Kang Lan Tee, and Tuck Seng Wong. An Engineered Constitutive Promoter Set with Broad Activity Range for Cupriavidus necator H16. ACS Synthetic Biology, 7(8):1918–1928, August 2018. Publisher: American Chemical Society.

[99] Nick Juty. Systems biology ontology: Update. Nature Precedings, page 1–1, Oct 2010.

[100] Linda J. Kahl and Drew Endy. A survey of enabling technologies in synthetic biology. Journal of Biological Engineering, 7(1):13, May 2013.

[101] Minoru Kanehisa and Susumu Goto. Kegg: Kyoto encyclopedia of genes and genomes. Nucleic Acids Research, 28(1):27 – 30, Jan 2000.

[102] Jonathan R. Karr, Wolfram Liebermeister, Arthur P. Goldberg, John A. P. Sekar, and Bilal Shaikh. Objtables: structured spreadsheets that promote data quality, reuse, and integration. arXiv:2005.05227 [cs, q-bio], Aug 2020. arXiv: 2005.05227.

[103] Abhijith Kashyap, Vagelis Hristidis, and Michalis Petropoulos. Facetor: cost-driven exploration of faceted query results. In Proceedings of the 19th ACM international conference on Information and knowledge management, CIKM '10, page 719–728. Association for Computing Machinery, Oct 2010.

[104] Spyridon Kavvadias, George Drosatos, and Eleni Kaldoudi. Supporting topic modeling and trends analysis in biomedical literature. Journal of Biomedical Informatics, 110:103574, 2020.

[105] Jason R. Kelly, Adam J. Rubin, Joseph H. Davis, Caroline M. Ajo-Franklin, John Cumbers, Michael J. Czar, Kim de Mora, Aaron L. Glieberman, Dileep D. Monie, and Drew Endy. Measuring the activity of biobrick promoters using an in vivo reference standard. Journal of Biological Engineering, 3:4, Mar 2009.

[106] Jason R. Kelly, Adam J. Rubin, Joseph H. Davis, Caroline M. Ajo-Franklin, John Cumbers, Michael J. Czar, Kim de Mora, Aaron L. Glieberman, Dileep D. Monie, and Drew Endy. Measuring the activity of biobrick promoters using an in vivo reference standard. Journal of Biological Engineering, 3:4, Mar 2009.

[107] Richard JR Kelwick, Alexander J Webb, and Paul S Freemont. Biological materials: the next frontier for cell-free synthetic biology. Frontiers in Bioengineering and Biotechnology, 8, 2020.

[108] Ahmad S. Khalil and James J. Collins. Synthetic biology: applications come of age. Nature Reviews. Genetics, 11(5):367–379, May 2010.

[109] Weston Kightlinger, Katherine F Warfel, Matthew P DeLisa, and Michael C Jewett. Synthetic glycobiology: parts, systems, and applications. ACS synthetic biology, 9(7):1534–1562, 2020.

[110] Halil Kilicoglu, Graciela Rosemblat, Marcelo Fiszman, and Dongwook Shin. Broad-coverage biomedical relation extraction with semrep. BMC bioinformatics, 21:1–28, 2020.

[111] Holger Knublauch and Dimitris Kontokostas. Shapes constraint language (shacl), Jul 2017.

[112] Dagmar Köhn and Nicolas Le Novère. Sed-ml – an xml format for the implementation of the miase guidelines. In Monika Heiner and Adelinde M. Uhrmacher, editors, Computational Methods in Systems Biology, Lecture Notes in Computer Science, page 176–190. Springer, 2008.

[113] Eugene Kolker and Elizabeth Stewart. Omics studies: How about metadata checklist and data publications? Journal of Proteome Research, 13(3):1783–1784, Mar 2014.

[114] Eugene Kolker, Vural Özdemir, Lennart Martens, William Hancock, Gordon Anderson, Nathaniel Anderson, Sukru Aynacioglu, Ancha Baranova, Shawn R. Campagna, Rui Chen, John Choiniere, Stephen P. Dearth, Wu-Chun Feng, Lynnette Ferguson, Geoffrey Fox, Dmitrij Frishman, Robert Grossman, Allison Heath, Roger Higdon, Mara H. Hutz, Imre Janko, Lihua Jiang, Sanjay Joshi, Alexander Kel, Joseph W. Kemnitz, Isaac S. Kohane, Natali Kolker, Doron Lancet, Elaine Lee, Weizhong Li, Andrey Lisitsa, Adrian Llerena, Courtney MacNealy-Koch, Jean-Claude Marshall, Paola Masuzzo, Amanda May, George Mias, Matthew Monroe, Elizabeth Montague, Sean Mooney, Alexey Nesvizhskii, Santosh Noronha, Gilbert Omenn, Harsha Rajasimha, Preveen Ramamoorthy, Jerry Sheehan, Larry Smarr, Charles V. Smith, Todd Smith, Michael Snyder, Srikanth Rapole, Sanjeeva Srivastava, Larissa Stanberry, Elizabeth Stewart, Stefano Toppo, Peter Uetz, Kenneth Verheggen, Brynn H. Voy, Louise Warnich, Steven W. Wilhelm, and Gregory Yandl. Toward more transparent and reproducible omics studies through a common metadata checklist and data publications. OMICS: A Journal of Integrative Biology, 18(1):10–14, Jan 2014.

[115] Roberta Kwok. Five hard truths for synthetic biology. Nature, 463(7279):288–290, Jan 2010.

[116] Leslie Lamport. LATEX: a document preparation system: user's guide and reference manual. Addison-wesley, 1994.

[117] Henry Lao. Linear Acoustic Processes in Rocket Engines. PhD thesis, University of Colorado at Boulder, 1979.

[118] Q. Lao, M. N. Cassoy, and K. Kirkpatrick. Acoustically generated vorticity from internal flow. J. Fluid Mechanics, 2:122–133, 1996.

[119] Q. Lao, D. R. Kassoy, and K. Kirkkopru. Nonlinear acoustic processes in rocket engines. J. Fluid Mechanics, 3:245–261, 1997.

[120] Peder Olesen Larsen and Markus von Ins. The rate of growth in scientific publication and the decline in coverage provided by science citation index. Scientometrics, 84(3):575–603, 2010.

[121] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. Prov-o: The prov ontology. World Wide Web Consortium, Apr 2013.

[122] Jinhyuk Lee. BioBERT. DMIS Laboratory - Korea University, Feb 2022.

[123] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, 36(4):1234–1240, 2020.

[124] Michael E. Lee, William C. DeLoache, Bernardo Cervantes, and John E. Dueber. A Highly Characterized Yeast Toolkit for Modular, Multipart Assembly. ACS Synthetic Biology, 4(9):975–986, September 2015. Publisher: American Chemical Society.

[125] Raymond Y. N. Lee and Paul W. Sternberg. Building a cell and anatomy ontology of caenorhabditis elegans. Comparative and Functional Genomics, 4(1):121–126, 2003.

[126] Florian Leitner, Andrew Chatr-aryamontri, Arnaud Ceol, Martin Krallinger, Luana Licata, Dr Lynette Hirschman, Gianni Cesareni, and Alfonso Valencia. Enriching publications with structured digital abstracts: The human-machine experiment. MITRE, Sep 2013.

[127] Alessandro Maccagnan, Mauro Riva, Erika Feltrin, Barbara Simionati, Tullio Vardanega, Giorgio Valle, and Nicola Cannata. Combining ontologies and workflows to design formal protocols for biological laboratories. Automated Experimentation, 2(1):3, Apr 2010.

[128] Rahuman S Malik-Sheriff, Mihai Glont, Tung V N Nguyen, Krishna Tiwari, Matthew G Roberts, Ashley Xavier, Manh T Vu, Jinghao Men, Matthieu Maire, Sarubini Kananathan, Emma L Fairbanks, Johannes P Meyer, Chinmay Arankalle, Thawfeek M Varusai, Vincent Knight-Schrijver, Lu Li, Corina Dueñas-Roca, Gaurhari Dass, Sarah M Keating, Young M Park, Nicola Buso, Nicolas Rodriguez, Michael Hucka, and Henning Hermjakob. Biomodels—15 years of sharing computational models in life science. Nucleic Acids Research, 48(D1):D407–D415, Jan 2020.

[129] Jeanet Mante, Yikai Hao, Jacob Jett, Udayan Joshi, Kevin Keating, Xiang Lu, Gaurav Nakum, Nicholas E. Rodriguez, Jiawei Tang, Logan Terry, Xuanyu Wu, Eric Yu, J. Stephen Downie, Bridget T. McInnes, Mai H. Nguyen, Brandon Sepulvado, Eric M. Young, and Chris J. Myers. Synthetic biology knowledge system. ACS Synthetic Biology, Aug 2021.

[130] Jeanet Mante, Nicholas Roehner, Kevin Keating, James Alastair McLaughlin, Eric Young, Jacob Beal, and Chris J. Myers. Curation principles derived from the analysis of the sbol igem data set. ACS Synthetic Biology, Sep 2021.

[131] Jeanet Mante, Zach Zundel, and Chris Myers. Extending synbiohub's functionality with plugins. ACS Synthetic Biology, 9(5):1216–1220, May 2020.

[132] Carlos Marcondes. From scientific communication to public knowledge: the scientific article web published as a knowledge base. In Proc. 9th ICCC ElPub, Jan 2005.

[133] Carlos Henrique Marcondes. Scientific methodology elements structure of Health Science e-journal articles. PhD thesis, Universidade Federal Fluminense - Brasil, Sep 2005.

[134] Jun Mashima, Yuichi Kodama, Takatomo Fujisawa, Toshiaki Katayama, Yoshihiro Okuda, Eli Kaminuma, Osamu Ogasawara, Kousaku Okubo, Yasukazu Nakamura, and Toshihisa Takagi. Dna data bank of japan. Nucleic Acids Research, 45(D1):D25–D31, Jan 2017.

[135] Yukiko Matsuoka, Samik Ghosh, and Hiroaki Kitano. Consistent design schematics for biological systems: standardization of representation in biological engineering. Journal of the Royal Society Interface, 6(Suppl 4):S393–S404, Aug 2009.

[136] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

[137] James Alastair McLaughlin, Jacob Beal, Göksel Mısırlı, Raik Grünberg, Bryan A. Bartley, James Scott-Brown, Prashant Vaidyanathan, Pedro Fontanarrosa, Ernst Oberortner, Anil Wipat, Thomas E. Gorochowski, and Chris J. Myers. The synthetic biology open language (sbol) version 3: Simplified data exchange for bioengineering. Frontiers in Bioengineering and Biotechnology, 8:1009, 2020.

[138] James Alastair McLaughlin, Chris J. Myers, Zach Zundel, Göksel Mısırlı, Michael Zhang, Irina Dana Ofiteru, Angel Goñi-Moreno, and Anil Wipat. Synbiohub: A standards-enabled design repository for synthetic biology. ACS Synthetic Biology, 7(2):682–688, Feb 2018.

[139] James Alastair McLaughlin, Chris J. Myers, Zach Zundel, Göksel Mısırlı, Michael Zhang, Irina Dana Ofiteru, Angel Goñi-Moreno, and Anil Wipat. SynBioHub: A standards-enabled design repository for synthetic biology. ACS Synthetic Biology, 7(2):682–688, 2018. PMID: 29316788.

[140] Peter McQuilton, Alejandra Gonzalez-Beltran, Philippe Rocca-Serra, Milo Thurston, Allyson Lister, Eamonn Maguire, and Susanna-Assunta Sansone. Biosharing: curated and crowd-sourced metadata standards, databases and data policies in the life sciences. Database: The Journal of Biological Databases and Curation, 2016, May 2016.

[141] Goksel Misirli, Anil Wipat, Joseph Mullen, Katherine James, Matthew Pocock, Wendy Smith, Nick Allenby, and Jennifer S. Hallinan. Bacillondex: An integrated data resource for systems and synthetic biology. Journal of Integrative Bioinformatics, 10(2):103 – 116, Jun 2013.

[142] F. C. Mulick. Rotational axisymmetric mean flow and damping of acoustic waves in a solid propellant. AIAA J., 3:1062–1063, 1964.

[143] F. C. Mulick. Stability of four-dimensional motions in a combustion chamber. Comb. Sci. Tech., 19:99–124, 1981.

[144] Kristian Müller and Katja Arndt. Standardization in synthetic biology. Methods in molecular biology (Clifton, N.J.), 813:23–43, Jan 2012.

[145] Richard Murray. Addgene: Cidar moclo extension, volume i, Oct 2019.

[146] Mark A Musen, Carol A Bean, Kei-Hoi Cheung, Michel Dumontier, Kim A Durante, Olivier Gevaert, Alejandra Gonzalez-Beltran, Purvesh Khatri, Steven H Kleinstein, Martin J O'Connor, Yannick Pouliot, Philippe Rocca-Serra, Susanna-Assunta Sansone, and Jeffrey A Wiser. The center for expanded data annotation and retrieval. Journal of the American Medical Informatics Association: JAMIA, 22(6):1148–1152, Nov 2015.

[147] Chris J. Myers. Computational synthetic biology: Progress and the road ahead. IEEE Transactions on Multi-Scale Computing Systems, 1(1):19–32, Jan 2015.

[148] Chris J. Myers, Nathan Barker, Kevin Jones, Hiroyuki Kuwahara, Curtis Madsen, and Nam-Phuong D. Nguyen. ibiosim: a tool for the analysis and design of genetic circuits. Bioinformatics, 25(21):2848–2849, Nov 2009.

[149] Chris J. Myers, Jacob Beal, Thomas E. Gorochowski, Hiroyuki Kuwahara, Curtis Madsen, James Alastair McLaughlin, Göksel Misirli, Tramy Nguyen, Ernst Oberortner, Meher Samineni, Anil Wipat, Michael Zhang, and Zach Zundel. A standard-enabled workflow for synthetic biology. Biochemical Society Transactions, 45(3):793–803, Jun 2017.

[150] Göksel Mısırlı, Jennifer Hallinan, Matthew Pocock, Phillip Lord, James Alastair McLaughlin, Herbert Sauro, and Anil Wipat. Data integration and mining for synthetic biology design. ACS synthetic biology, 5(10):1086–1097, Oct 2016.

[151] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. Lingvisticae Investigationes, 30(1):3–26, 2007.

[152] NCBI. Fasta format for nucleotide sequences, Feb 2021.

[153] Bang Viet Nguyen and Min-Yen Kan. Functional faceted web query analysis. In WWW2007, page 8, 2007.

[154] Tramy Nguyen, Timothy S. Jones, Pedro Fontanarrosa, Jeanet V. Mante, Zach Zundel, Douglas Densmore, and Chris J. Myers. Design of asynchronous genetic circuits. Proceedings of the IEEE, 107(7):1356–1368, Jul 2019.

[155] Tramy T Nguyen. ASYNCHRONOUS GENETIC CIRCUIT DESIGN. PhD thesis, University of Utah, Dec 2019.

[156] David Nickerson, Koray Atalag, Bernard de Bono, Jörg Geiger, Carole Goble, Susanne Hollmann, Joachim Lonien, Wolfgang Müller, Babette Regierer, Natalie J. Stanford, and et al. The human physiome: how standards, software and innovative service infrastructures are providing the building blocks to make it achievable. Interface Focus, 6(2):20150103, Apr 2016.

[157] Alec A. K. Nielsen, Bryan S. Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A. Strychalski, David Ross, Douglas Densmore, and Christopher A. Voigt. Genetic circuit design automation. Science (New York, N.Y.), 352(6281):aac7341, Apr 2016.

[158] Alec A. K. Nielsen, Bryan S. Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A. Strychalski, David Ross, Douglas Densmore, and Christopher A. Voigt. Genetic circuit design automation. Science, Apr 2016.

[159] Alireza Noruzi. Application of ranganathan's laws to the web: the five laws of the web.

[160] Nicolas Le Novère, Andrew Finney, Michael Hucka, Upinder S. Bhalla, Fabien Campagne, Julio Collado-Vides, Edmund J. Crampin, Matt Halstead, Edda Klipp, Pedro Mendes, Poul Nielsen, Herbert Sauro, Bruce Shapiro, Jacky L. Snoep, Hugh D. Spence, and Barry L. Wanner. Minimum information requested in the annotation of biochemical models (miriam). Nature Biotechnology, 23(12):1509–1515, Dec 2005.

[161] Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I. Aladjem, Sarala M. Wimalaratne, and et al. The systems biology graphical notation. Nature Biotechnology, 27(8):735–741, Aug 2009.

[162] Ulrike Obst, Timothy K. Lu, and Volker Sieber. A Modular Toolkit for Generating *Pichia pastoris* Secretion Libraries. ACS Synthetic Biology, 6(6):1016–1025, June 2017.

[163] Sandra Orchard, Bissan Al-Lazikani, Steve Bryant, Dominic Clark, Elizabeth Calder, Ian Dix, Ola Engkvist, Mark Forster, Anna Gaulton, Michael Gilson, Robert Glen, Martin Grigorov, Kim Hammond-Kosack, Lee Harland, Andrew Hopkins, Christopher Larminie, Nick Lynch,

Romeena K. Mann, Peter Murray-Rust, Elena Lo Piparo, Christopher Southan, Christoph Steinbeck, David Wishart, Henning Hermjakob, John Overington, and Janet Thornton. Minimum information about a bioactive entity (miabe). Nature Reviews Drug Discovery, 10(9):661–669, Sep 2011.

[164] JSON Schema Org. Implementations.

[165] Jason A. Palmer. pdftotext: Simple pdf text extraction. `https://github.com/jalan/pdftotext`.

[166] Irene V. Pasquetto, Christine L. Borgman, and Morgan F. Wofford. Uses and reuses of scientific data: The data creators' advantage. Harvard Data Science Review, 1(2), Nov 2019.

[167] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. Proceedings of the National Academy of Sciences, 85(8):2444–2448, Apr 1988.

[168] Jean Peccoud, Megan F. Blauvelt, Yizhi Cai, Kristal L. Cooper, Oswald Crasta, Emily C. DeLalla, Clive Evans, Otto Folkerts, Blair M. Lyons, Shrinivasrao P. Mane, and et al. Targeted development of registries of biological parts. PloS One, 3(7):e2671, Jul 2008.

[169] Klara Piletič and Tanja Kunej. Minimal standards for reporting microrna:target interactions. OMICS: A Journal of Integrative Biology, 21(4):197–206, Apr 2017.

[170] Manuel Porcar and Juli Peretó. Are we doing synthetic biology? Systems and Synthetic Biology, 6(3–4):79–83, Dec 2012.

[171] Eric Prud'hommeaux, Iovka Boneva, Jose Emilio Labra Gayo, and Gregg Kellogg. Shape expressions language 2.1, Oct 2019.

[172] Priscilla E. M. Purnick and Ron Weiss. The second wave of synthetic biology: from modules to systems. Nature Reviews Molecular Cell Biology, 10(6):410–422, Jun 2009.

[173] Isabel M. Pötzsch, Jeanet Mante, Jacob Beal, and Chris J. Myers. Creating sbol designs with excel. In Computational Modeling in Biology Network Forum (COMBINE 2020), Oct 2020.

[174] Benjamin Raimbault, Jean-Philippe Cointet, and Pierre-Benoît Joly. Mapping the emergence of synthetic biology. PLOS ONE, 11(9):1–19, 09 2016.

[175] A T Ramitha and J S Jayasudha. Personalization and privacy in profile-based web search. In 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS), page 1–4, May 2016.

[176] S. R. Ranganathan. The Five Laws of Library Science. Madras Library Association (Madras, India) and Edward Goldston (London, UK), 1931. Accepted: 2006-10-18T00:00:01Z.

[177] Marc L. Resnick and Misha W. Vaughan. Best practices and future visions for search user interfaces. Journal of the American Society for Information Science and Technology, 57(6):781–787, 2006.

[178] R. S. Richards and A. M. Brown. Coupling between acoustic velocity oscillations and solid propellant combustion. J. Prop. and Power, 5:828–837, 1982.

[179] Hajo Rijgersberg, Mark van Assem, and Jan Top. Ontology of units of measure and related concepts. Semantic Web, 4(1):3–13, Jan 2013.

[180] Nicholas Roehner, Jacob Beal, Kevin Clancy, Bryan Bartley, Goksel Misirli, Raik Grünberg, Ernst Oberortner, Matthew Pocock, Michael Bissell, Curtis Madsen, Tramy Nguyen, Michael Zhang, Zhen Zhang, Zach Zundel, Douglas Densmore, John H. Gennari, Anil Wipat, Herbert M. Sauro, and Chris J. Myers. Sharing structure and function in biological design with sbol 2.0. ACS Synthetic Biology, 5(6):498–506, Jun 2016.

[181] Nicholas Roehner, Jeanet Mante, Chris J. Myers, and Jacob Beal. Synthetic biology curation tools (synbict). ACS synthetic biology, 10(11):3200–3204, Nov 2021.

[182] Marc-Sven Roell and Matias D Zurbriggen. The impact of synthetic biology for future agriculture and nutrition. Current Opinion in Biotechnology, 61:102–109, 2020.

[183] Alberto Santos-Zavaleta, Heladia Salgado, Socorro Gama-Castro, Mishael Sánchez-Pérez, Laura Gómez-Romero, Daniela Ledezma-Tejeida, Jair Santiago García-Sotelo, Kevin Alquicira-Hernández, Luis José Muñiz-Rascado, Pablo Peña-Loredo, Cecilia Ishida-Gutiérrez, David A. Velázquez-Ramírez, Víctor Del Moral-Chávez, César Bonavides-Martínez, Carlos-Francisco Méndez-Cruz, James Galagan, and Julio Collado-Vides. Regulondb v 10.5: Tackling challenges to unify classic and high throughput knowledge of gene regulation in e. coli k-12. Nucleic Acids Res., 47(D1):D212–D220, 2018.

[184] Eric W. Sayers, Mark Cavanaugh, Karen Clark, James Ostell, Kim D. Pruitt, and Ilene Karsch-Mizrachi. Genbank. Nucleic Acids Research, 47(D1):D94 – D99, Jan 2019.

[185] Gary Schindelman, Jolene S. Fernandes, Carol A. Bastiani, Karen Yook, and Paul W. Sternberg. Worm phenotype ontology: Integrating phenotype data within and beyond the c. elegans community. BMC Bioinformatics, 12(1):32, Jan 2011.

[186] Conrad L. Schoch, Stacy Ciufo, Mikhail Domrachev, Carol L. Hotton, Sivakumar Kannan, Rogneda Khovanskaya, Detlef Leipe, Richard Mcveigh, Kathleen O'Neill, Barbara Robbertse, and et al. Ncbi taxonomy: a comprehensive update on curation, resources and tools. Database: The Journal of Biological Databases and Curation, 2020:baaa062, Jan 2020.

[187] Lynn M Schriml, Elvira Mitraka, James Munro, Becky Tauber, Mike Schor, Lance Nickle, Victor Felix, Linda Jeng, Cynthia Bearer, Richard Lichenstein, Katharine Bisordi, Nicole Campion, Brooke Hyman, David Kurland, Connor Patrick Oates, Siobhan Kibbey, Poorna Sreekumar, Chris Le, Michelle Giglio, and Carol Greene. Human disease ontology 2018 update: classification, content and workflow expansion. Nucleic Acids Research, 47(D1):D955–D962, Jan 2019.

[188] B. Shadrach. S r ranganthan's five laws of library science: A foundation for democratising knowledge basheerhamad shadrach. Informatics Studies, Jan 2019.

[189] Phili Shapira, Seokbeom Kwon, and Jan Youtie. Tracking the emergence of synthetic biology. Scientometrics, 112:1439–1469, 2017.

[190] Reshma P. Shetty, Drew Endy, and Thomas F. Knight. Engineering biobrick vectors from biobrick parts. Journal of Biological Engineering, 2(1):5, Apr 2008.

[191] David Shotton. Cito, the citation typing ontology. Journal of Biomedical Semantics, 1(1):S6, Jun 2010.

[192] Carol Simpson. Editor's notes. Library Media Connection, page 6, May 2008.

[193] Payam Siyari, Bistra Dilkina, and Constantine Dovrolis. Evolution of hierarchical structure and reuse in igem synthetic dna sequences. In Joao M. F. Rodrigues, Pedro J. S. Cardoso, Janio Monteiro, Roberto Lam, Valeria V. Krzhizhanovskaya, Michael H. Lees, Jack J. Dongarra, and Peter M.A. Sloot, editors, Computational Science - ICCS 2019, Lecture Notes in Computer Science, pages 468 – 482. Springer International Publishing, 2019.

[194] T. M. Smitty, R. L. Coach, and F. B. Höndra. Unsteady flow in simulated solid rocket motors. In 16st Aerospace Sciences Meeting, number 0112 in 78. AIAA, 1978.

[195] Christina D. Smolke. Building outside of the box: igem and the biobricks foundation. Nature Biotechnology, 27:1099 – 1102, Dec 2009.

[196] Michael Snyder, George Mias, Larissa Stanberry, and Eugene Kolker. Metadata checklist for the integrated personal omics study: Proteomics and metabolomics experiments. Big Data, 1(4):202–206, Dec 2013.

[197] Harold Solbrig. hsolbrig/PyShEx. GitHub, Sep 2020.

[198] Axel J Soto, Piotr Przybyła, and Sophia Ananiadou. Thalia: semantic search engine for biomedical abstracts. Bioinformatics, 35(10):1799–1801, May 2019.

[199] Paul T. Spellman, Michael Miller, Jason Stewart, Charles Troup, Ugis Sarkans, Steve Chervitz, Derek Bernhart, Gavin Sherlock, Catherine Ball, Marc Lepage, and et al. Design and implementation of microarray gene expression markup language (mage-ml). Genome Biology, 3(9):research0046.1, Aug 2002.

[200] W3C staff members. Examples of rdf validation, 2012.

[201] Christian J Stoeckert, John Quackenbush, Alvis Brazma, and Catherine A Ball. Minimum information about a functional genomics experiment: the state of microarray standards and their extension to other technologies. Drug Discovery Today: TARGETS, 3(4):159–164, Aug 2004.

[202] Jonathan Strickland and John Donovan. How google works, May 2019.

[203] Cong Sun, Zhihao Yang, Leilei Su, Lei Wang, Yin Zhang, Hongfei Lin, and Jian Wang. Chemical-protein Interaction Extraction via Gaussian Probability Distribution and External Biomedical Knowledge. Bioinformatics, 05 2020. btaa491.

[204] Olivier Taboureau, Sonny Kim Nielsen, Karine Audouze, Nils Weinhold, Daniel Edsgärd, Francisco S Roque, Irene Kouskoumvekaki, Alina Bora, Ramona Curpan, Thomas Skøt Jensen, et al. Chemprot: a disease chemical biology database. Nucleic acids research, 39(suppl1):D367–D372, 2010.

[205] Tzu-Chieh Tang, Bolin An, Yuanyuan Huang, Sangita Vasikaran, Yanyi Wang, Xiaoyu Jiang, Timothy K. Lu, and Chao Zhong. Materials design by synthetic biology. Nature Reviews Materials, 6(44):332–350, Apr 2021.

[206] Joseph D. Taum. Investigation of flow turning phenomenon. In 20th Aerospace Sciences Meeting, number 0297 in 82. AIAA, 1982.

[207] Chris F Taylor, Dawn Field, Susanna-Assunta Sansone, Jan Aerts, Rolf Apweiler, Michael Ashburner, Catherine A Ball, Pierre-Alain Binz, Molly Bogue, Tim Booth, Alvis Brazma, Ryan R Brinkman, Adam Michael Clark, Eric W Deutsch, Oliver Fiehn, Jennifer Fostel, Peter Ghazal, Frank Gibson, Tanya Gray, Graeme Grimes, John M Hancock, Nigel W Hardy, Henning Hermjakob, Randall K Julian, Matthew Kane, Carsten Kettner, Christopher Kinsinger, Eugene Kolker, Martin Kuiper, Nicolas Le Novère, Jim Leebens-Mack, Suzanna E Lewis, Phillip Lord, Ann-Marie Mallon, Nishanth Marthandan, Hiroshi Masuya, Ruth McNally, Alexander Mehrle, Norman Morrison, Sandra Orchard, John Quackenbush, James M Reecy, Donald G Robertson, Philippe Rocca-Serra, Henry Rodriguez, Heiko Rosenfelder, Javier Santoyo-Lopez, Richard H Scheuermann, Daniel Schober, Barry Smith, Jason Snape, Christian J Stoeckert, Keith Tipton, Peter Sterk, Andreas Untergasser, Jo Vandesompele, and Stefan Wiemann. Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the mibbi project. Nature biotechnology, 26(8):889–896, Aug 2008.

[208] Chris F Taylor, Dawn Field, Susanna-Assunta Sansone, Jan Aerts, Rolf Apweiler, Michael Ashburner, Catherine A Ball, Pierre-Alain Binz, Molly Bogue, Tim Booth, et al. Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the mibbi project. Nature biotechnology, 26(8):889–896, 2008.

[209] Jessica D Tenenbaum, Susanna-Assunta Sansone, and Melissa Haendel. A sea of standards for omics data: sink or swim? Journal of the American Medical Informatics Association: JAMIA, 21(2):200–203, Mar 2014.

[210] Carol Tenopir, Suzie Allard, Kimberly Douglass, Arsev Umur Aydinoglu, Lei Wu, Eleanor Read, Maribeth Manoff, and Mike Frame. Data sharing by scientists: Practices and perceptions. PLOS ONE, 6(6):e21101, Jun 2011.

[211] Logan Terry, Jared Earl, Sam Thayer, Samuel Bridge, and Chris J. Myers. Sbolcanvas: A visual editor for genetic designs. ACS Synthetic Biology, Jun 2021.

[212] The Gene Ontology Consortium. The gene ontology resource: 20 years and still going strong. Nucleic Acids Research, 47(D1):D330–D338, Jan 2019.

[213] Philippe Thomas, Johannes Starlinger, Alexander Vowinkel, Sebastian Arzt, and Ulf Leser. Geneview: a comprehensive semantic search engine for pubmed. Nucleic Acids Research, 40(W1):W585–W591, Jul 2012.

[214] Katherine Thornton, Harold Solbrig, Gregory S. Stupp, Jose Emilio Labra Gayo, Daniel Mietchen, Eric Prud'hommeaux, and Andra Waagmeester. Correction to: Using shape expressions (shex) to share rdf data models and to guide curation with rigorous validation. In Pascal Hitzler, Miriam Fernández, Krzysztof Janowicz, Amrapali Zaveri, Alasdair J.G. Gray, Vanessa Lopez, Armin Haller, and Karl Hammar, editors, The Semantic Web, Lecture Notes in Computer Science, page C1–C1. Springer International Publishing, 2019.

[215] Joshua J. Timmons and Doug Densmore. Repository-based plasmid design. PLOS ONE, 15(1):e0223935, Jan 2020.

[216] Keith F. Tipton, Richard N. Armstrong, Barbara M. Bakker, Amos Bairoch, Athel Cornish-Bowden, Peter J. Halling, Jan-Hendrik Hofmeyr, Thomas S. Leyh, Carsten Kettner, Frank M. Raushel, Johann Rohwer, Dietmar Schomburg, and Christoph Steinbeck. Standards for reporting enzyme data: The strenda consortium: What it aims to do and why it should be helpful. Perspectives in Science, 1(1):131–137, May 2014.

[217] Uriel Urquiza-García, Tomasz Zieliński, and Andrew J Millar. Better research by efficient sharing: evaluation of free management platforms for synthetic biology designs. Synthetic Biology, 4(1):ysz016, 2019.

[218] Uriel Urquiza-García, Tomasz Zieliński, and Andrew J. Millar. Better research by efficient sharing: evaluation of free management platforms for synthetic biology designs. Synthetic Biology (Oxford, England), 4(1):ysz016, 2019.

[219] Allard J van Altena, Perry D Moerland, Aeilko H Zwinderman, and Sílvia D Olabarriaga. Understanding big data themes from scientific biomedical literature through topic modeling. Journal of Big Data, 3(1):1–21, 2016.

[220] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.

[221] Cristina Vilanova and Manuel Porcar. igem 2.0 - refoundations for engineering biology. Nature Biotechnology, 32(5):420 – 424, May 2014.

[222] Dagmar Waltemath, Richard Adams, Daniel A. Beard, Frank T. Bergmann, Upinder S. Bhalla, Randall Britten, Vijayalakshmi Chelliah, Michael T. Cooling, Jonathan Cooper, Edmund J. Crampin, Alan Garny, Stefan Hoops, Michael Hucka, Peter Hunter, Edda Klipp, Camille Laibe, Andrew K. Miller, Ion Moraru, David Nickerson, Poul Nielsen, Macha Nikolski, Sven Sahle, Herbert M. Sauro, Henning Schmidt, Jacky L. Snoep, Dominic Tolle, Olaf Wolkenhauer, and Nicolas Le Novère. Minimum information about a simulation experiment (miase). PLOS Computational Biology, 7(4):e1001122, Apr 2011.

[223] Dagmar Waltemath, Richard Adams, Frank T. Bergmann, Michael Hucka, Fedor Kolpakov, Andrew K. Miller, Ion I. Moraru, David Nickerson, Sven Sahle, Jacky L. Snoep, and et al. Reproducible computational biology experiments with sed-ml–the simulation experiment description markup language. BMC systems biology, 5:198, Dec 2011.

[224] Beibei Wang, Huayi Yang, Jianan Sun, Chuhao Dou, Jian Huang, and Feng-Biao Guo. Biomaster: An integrated database and analytic platform to provide comprehensive information about biobrick parts. Frontiers in Microbiology, 12, 2021.

[225] Leon Weber, Jannes Münchmeyer, Tim Rocktäschel, Maryam Habibi, and Ulf Leser. Huner: improving biomedical ner with pretraining. Bioinformatics, 36(1):295–302, 2020.

[226] Bifan Wei, Jun Liu, Qinghua Zheng, Wei Zhang, Xiaoyu Fu, and Boqin Feng. A survey of faceted search. Journal of Web Engineering, 12:25, 2013.

[227] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin core metadata for resource discovery, Sep 1998.

[228] Maxwell A Weinzierl, Ramon Maldonado, and Sanda M Harabagiu. The impact of learning unified medical language system knowledge embeddings in relation extraction from biomedical texts. Journal of the American Medical Informatics Association, 27(10):1556–1567, 2020.

[229] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The fair guiding principles for scientific data management and stewardship. Scientific Data, 3, Mar 2016.

[230] Ulrike Wittig, Renate Kania, Martin Golebiewski, Maja Rey, Lei Shi, Lenneke Jong, Enkhjargal Algaa, Andreas Weidemann, Heidrun Sauer-Danzwith, Saqib Mir, and et al. Sabio-rk–database for biochemical reaction kinetics. Nucleic Acids Research, 40(Database issue):D790–796, Jan 2012.

[231] Katherine Wolstencroft, Olga Krebs, Jacky L. Snoep, Natalie J. Stanford, Finn Bacall, Martin Golebiewski, Rostyk Kuzyakiv, Quyen Nguyen, Stuart Owen, Stian Soiland-Reyes, and et al. Fairdomhub: a repository and collaboration environment for sharing systems biology research. Nucleic Acids Research, 45(D1):D404–D407, Jan 2017.

[232] Katherine Wolstencroft, Stuart Owen, Matthew Horridge, Simon Jupp, Olga Krebs, Jacky Snoep, Franco Du Preez, Wolfgang Mueller, Robert Stevens, and Carole Goble. Stealthy annotation of experimental biology by spreadsheets. Concurrency and Computation: Practice and Experience, 25(4):467–480, 2013.

[233] Katherine Wolstencroft, Stuart Owen, Olga Krebs, Wolfgang Mueller, Quyen Nguyen, Jacky L Snoep, and Carole Goble. Semantic data and models sharing in systems biology: The just enough results model and the seek platform. In International Semantic Web Conference, pages 212–227. Springer, 2013.

[234] Katy Wolstencroft, Stuart Owen, Matthew Horridge, Olga Krebs, Wolfgang Mueller, Jacky L Snoep, Franco du Preez, and Carole Goble. Rightfield: embedding ontology annotation in spreadsheets. Bioinformatics, 27(14):2021–2022, 2011.

[235] Katy Wolstencroft, Stuart Owen, Matthew Horridge, Olga Krebs, Wolfgang Mueller, Jacky L. Snoep, Franco du Preez, and Carole Goble. Rightfield: embedding ontology annotation in spreadsheets. Bioinformatics, 27(14):2021 – 2022, Jul 2011.

[236] Cathy H. Wu, Lai-Su L. Yeh, Hongzhan Huang, Leslie Arminski, Jorge Castro-Alvear, Yongxing Chen, Zhangzhi Hu, Panagiotis Kourtesis, Robert S. Ledley, Baris E. Suzek, C. R. Vinayaka, Jian Zhang, and Winona C. Barker. The protein information resource. Nucleic Acids Research, 31(1):345–347, Jan 2003.

[237] Yongfu Yang, Wei Shen, Ju Huang, Runxia Li, Yubei Xiao, Hui Wei, Yat-Chen Chou, Min Zhang, Michael E. Himmel, Shouwen Chen, Li Yi, Lixin Ma, and Shihui Yang. Prediction and characterization of promoters and ribosomal binding sites of zymomonas mobilis in system biology era. Biotechnology for Biofuels, 12:52, 2019.

[238] Eric M. Young, Zheng Zhao, Bianca E. M. Gielesen, Liang Wu, D. Benjamin Gordon, Johannes A. Roubos, and Christopher A. Voigt. Iterative algorithm-guided design of massive strain libraries, applied to itaconic acid production in yeast. Metabolic Engineering, 48:33–43, July 2018.

[239] Eric Yu, Jeanet Mante, and Chris J. Myers. Sequence-based searching for synbiohub using vsearch. ACS Synthetic Biology, 11(2):990–995, Feb 2022.

[240] Tommy Yu, Catherine M. Lloyd, David P. Nickerson, Michael T. Cooling, Andrew K. Miller, Alan Garny, Jonna R. Terkildsen, James Lawson, Randall D. Britten, Peter J. Hunter, and et al. The physiome model repository 2. Bioinformatics (Oxford, England), 27(5):743–744, Mar 2011.

[241] Guillermo Yáñez Feliú, Benjamín Earle Gómez, Verner Codoceo Berrocal, Macarena Muñoz Silva, Isaac N. Nuñez, Tamara F. Matute, Anibal Arce Medina, Gonzalo Vidal, Carlos Vidal Céspedes, Jonathan Dahlin, Fernán Federici, and Timothy J. Rudge. Flapjack: Data management and analysis for genetic circuit characterization. ACS Synthetic Biology, 10(1):183–191, Jan 2021.

[242] Amrapali Zaveri, Wei Hu, and Michel Dumontier. Metacrowd: Crowdsourcing biomedical metadata quality assessment. Human Computation, 6:98–112, Sep 2019.

[243] Robert A. Zeddini. Injection-induced flows in porous-walled ducts. AIAA Journal, 14:766–773, 1981.

[244] Zhen Zhang, Tramy Nguyen, Nicholas Roehner, Göksel Misirli, Matthew Pocock, Ernst Oberortner, Meher Samineni, Zach Zundel, Jacob Beal, Kevin Clancy, Anil Wipat, and Chris J. Myers. libsbolj 2.0: A java library to support sbol 2.0. IEEE Life Sciences Letters, 1(4):34–37, Dec 2015.

[245] Yun Zhou and W. Bruce Croft. Query performance prediction in web search environments. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ’07, page 543–550. Association for Computing Machinery, Jul 2007.

[246] Tomasz Zieliński, Johnny Hay, Andrew Romanowski, Anja Nenninger, Alistair McCormick, and Andrew J Millar. Synbio2easy—a biologist-friendly tool for batch operations on sbol designs with excel inputs. Synthetic Biology, 7(1):ysac002, Oct 2022.

[247] Ann Zimmerman. Not by metadata alone: the use of diverse forms of knowledge to locate data for reuse. International Journal on Digital Libraries, 7(1):5–16, Oct 2007.

[248] Anneke Zuiderwijk, Rhythima Shinde, and Wei Jeng. What drives and inhibits researchers to share and use open research data? a systematic literature review to analyze factors influencing open research data adoption. PLOS ONE, 15(9):e0239283, Sep 2020.

[249] Zach Zundel, Meher Samineni, Zhen Zhang, and Chris J. Myers. A validator and converter for the synthetic biology open language. ACS Synthetic Biology, 6(7):1161–1168, Jul 2017.

[250] Marko Ćurković and Andro Košec. Bubble effect: including internet search engines in systematic reviews introduces selection bias and impedes scientific reproducibility. BMC Medical Research Methodology, 18(1):130, Nov 2018.

[251] Radim Řehůřek. Gensim: Topic modelling for humans. `https://radimrehurek.com/gensim/index.html`.

Appendix A

Supplemental iGEM Figures

- Figure A.1: Variation in different part description field lengths over different iGEM years.
- Figure A.2: Variation in part submission and description by month and year. a) The number of submissions being made per month. The spikes indicate the approximate time of the jamboree each year. b) Part Description Variation by Month.
- Figure A.3: Variation in the average part description field lengths over time for three different iGEM submission groups. No clear pattern is visible here or in any of the simple scatter plots.

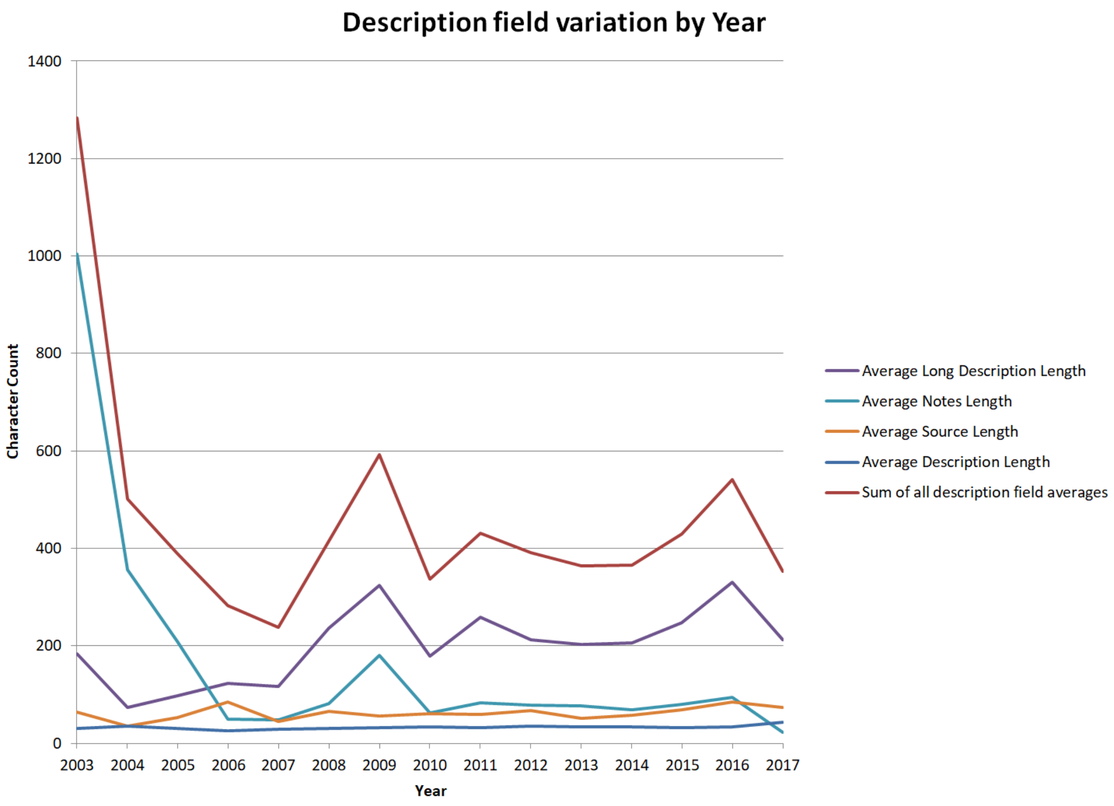
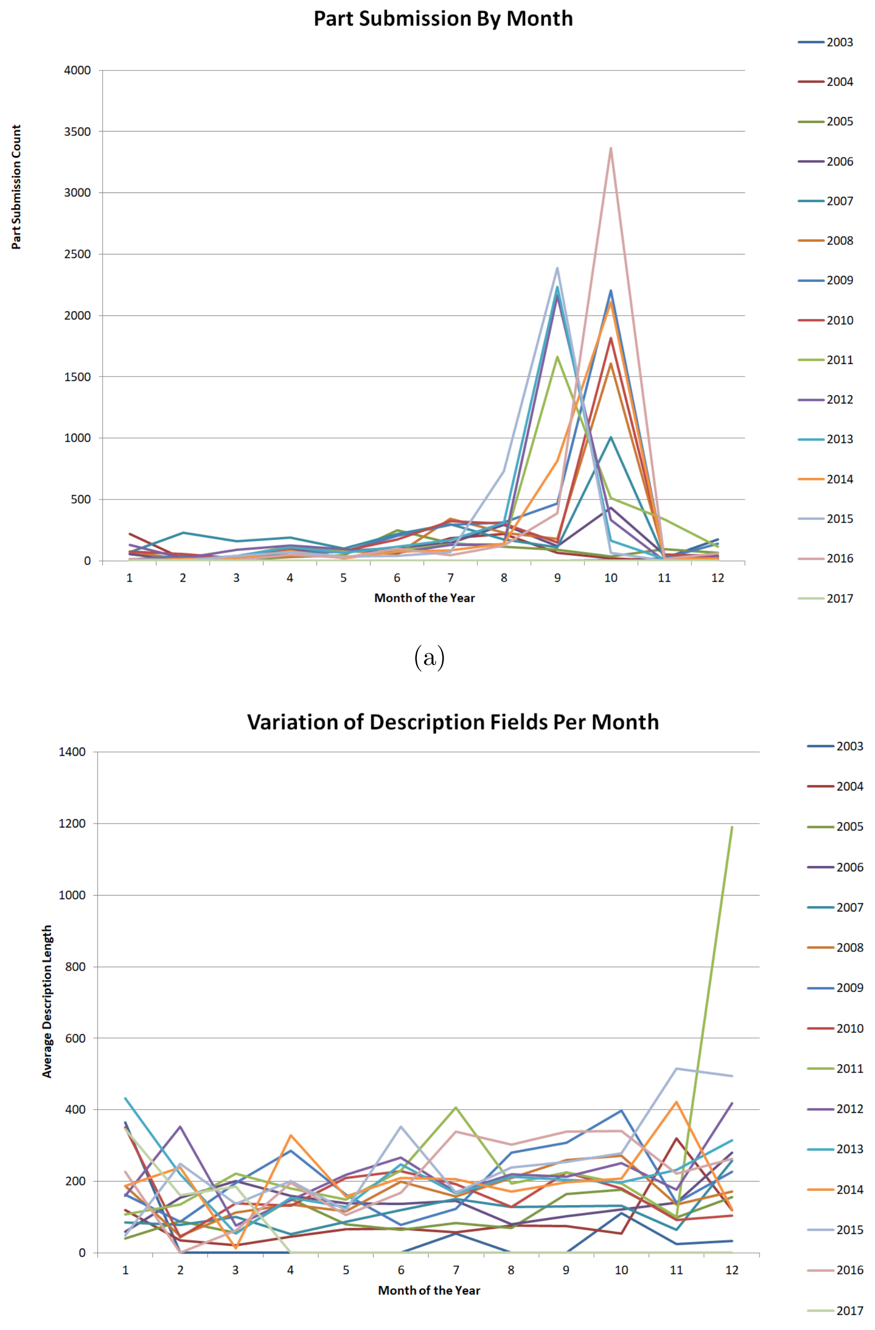


Figure A.1: Variation in different part description field lengths over different iGEM years. The outlier of 2003 is assumed to be due to the very small number of teams at that time.



(a)

(b)

Figure A.2: Variation in part submission and description by month and year. a) The number of submissions being made per month. The spikes indicate the approximate time of the jamboree each year. b) Part Description Variation by Month. There are no clear indicators of correlation between the month of the year, or the time till jamboree so no further statistics were run.

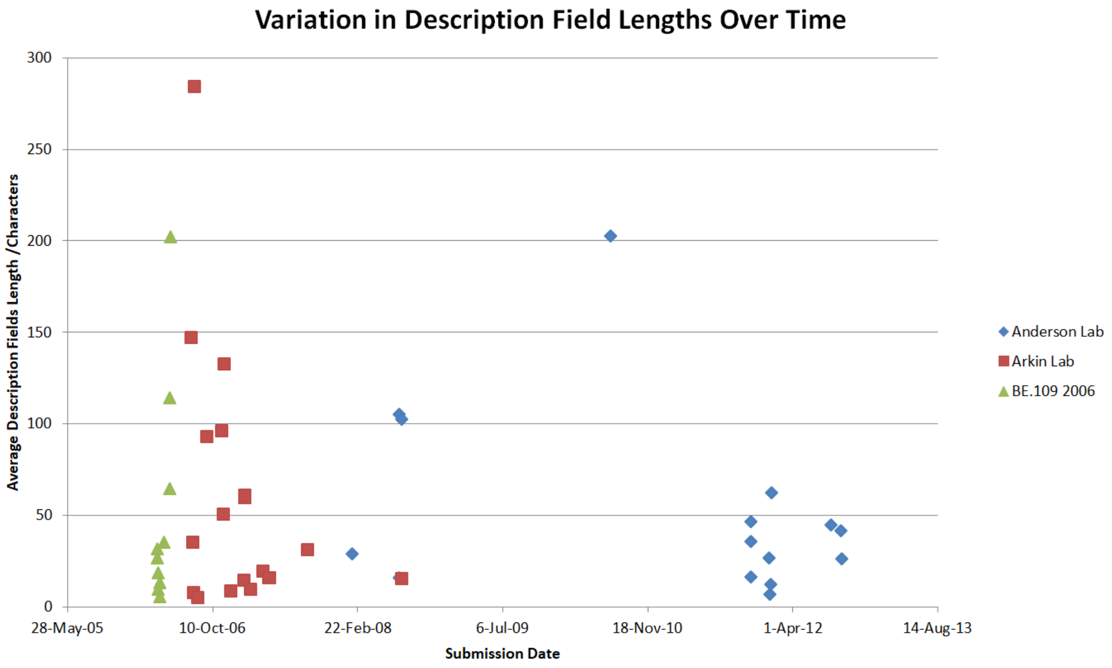


Figure A.3: Variation in the average part description field lengths over time for three different iGEM submission groups. No clear pattern is visible here or in any of the simple scatter plots. Further analysis is needed to make statements about correlation. The full data for all groups is available in the supplemental information.

Appendix B

GitHub Repositories

A list of GitHub Repositories used in this work.

B.1 Plugins

- Test:
  - ∗ `https://github.com/SynBioHub/Plugin-Submit-Test`
  - ∗ `https://github.com/SynBioHub/Plugin-Visual-Test`
  - ∗ `https://github.com/SynBioHub/Plugin-Visual-Serve-Test`
  - ∗ `https://github.com/SynBioHub/Plugin-Download-Test`
  - ∗ `https://github.com/SynBioHub/Plugin-Curation-Test`
  - ∗ `https://github.com/SynBioHub/Plugin-Search-Test`
  - ∗ `https://github.com/SynBioHub/Plugin-Index-Test`
  - ∗ `https://github.com/SynBioHub/Plugin-Link-Test`
  - ∗ `https://github.com/SynBioHub/Plugin-Submit-Test-js`
  - ∗ `https://github.com/SynBioHub/Plugin-Visual-Test-js`
  - ∗ `https://github.com/SynBioHub/Plugin-Visual-Serve-Test-js`
  - ∗ `https://github.com/SynBioHub/Plugin-Download-Test-js`
- Docker and Postman:

  - ∗ https://github.com/SynBioHub/Postman
  - ∗ https://github.com/SynBioHub/Docker-base-python

- Submit:
  - ∗ https://github.com/SynBioHub/Plugin-Submit-ShortBOL
  - ∗ https://github.com/SynBioHub/Plugin-Submit-Snapgene

- Visualization:
  - ∗ https://github.com/SynBioHub/Plugin-Visual-ProteinStructure
  - ∗ https://github.com/SynBioHub/Plugin-Visual-Igem
  - ∗ https://github.com/SynBioHub/Plugin-Visual-VisBOL-Js
  - ∗ https://github.com/SynBioHub/Plugin-Visual-Seqviz
  - ∗ https://github.com/SynBioHub/Plugin-Visual-Component-Use

- Download:
  - ∗ https://github.com/SynBioHub/Plugin-Download-iBioSim
  - ∗ https://github.com/SynBioHub/Plugin-Download-ShortBOL
  - ∗ https://github.com/SynBioHub/Plugin-Download-Snapgene

- Curation:
  - ∗ https://github.com/SynBioHub/Plugin-Curation-Synbict

B.2 Excel-SBOL

- https://github.com/SynBioDex/Excel-to-SBOL
- https://github.com/SynBioDex/SBOL-to-Excel
- https://github.com/SynBioHub/Plugin-Submit-Excel2SBOL

- https://github.com/SynBioHub/Plugin-Download-SBOL2Excel
- https://github.com/SynBioHub/Plugin-Submit-Excel-Composition
- https://github.com/SynBioHub/Plugin-Submit-Excel-Library
- https://github.com/SynBioHub/Plugin-Visual-Flapjack

B.3 Post-hoc Curation

- iGEM work: https://github.com/JMante1/iGem-Data-Cleaning
- ACS work:
    - https://github.com/synbioks/SynBioBERT
    - https://github.com/synbioks/ACS-XML-to-text
    - https://github.com/synbioks/SPARQL-queries-ACS-Synbio
    - https://github.com/synbioks/sequence_supplementals
    - https://github.com/synbioks/Text-Mining-NLP
- Addgene Work: https://github.com/JMante1/Addgene-Annotation

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346 USA