Jurnal Teknologi

Full paper

A Review of CADs, Languages and Data Models for Synthetic Biology

Narjeskhatoon Habibi[a*], Siti Zaiton Mohd Hashim[a], Cesar A. Rodriguez[b], Mohd Razip Samian[c]

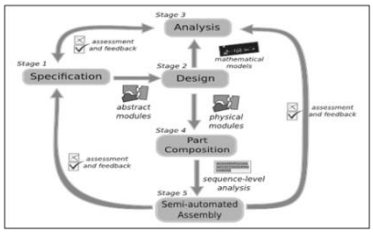
[a]*Soft Computing Research Group, Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia*
[b]*Autodesk Incorporation, San Francisco, CA,USA*
[c]*School of Biological Sciences, Universiti Sains Malaysia,Pinang, Malaysia*

*Corresponding author: hnarjeskhatoon2@live.utm.my

Article history

Received :25 March 2013
Received in revised form :
4 June 2013
Accepted :15 June 2013

Graphical abstract



Abstract

Synthetic biology is to apply engineering standards to the biology. It has two main aims; first to design and construct biological systems in order to extend the understanding of how they work, and second, to design biological systems for performing valuable tasks like medications synthesis and biofuels production. Due to the complex nature of the biological systems, and similar to the engineering fields, computational methods and computer aided design (CAD) tools, are required to handle this complexity. In this review, existing CADs, languages and data models for synthetic biology are explored. The goal is to provide an insight for the synthetic biologists to take advantage of the available computational tools and to find out the gaps to work on.

Keywords: CAD; BioCAD; synthetic biology; bioinformatics; language; data model

Abstrak

Biologi sintetik adalah untuk memohon piawaian kejuruteraan biologi. Ia mempunyai dua matlamat utama iaitu pertama untuk mereka bentuk dan membina sistem biologi untuk melanjutkan memahami bagaimana ia berfungsi, dan kedua, untuk mereka bentuk sistem biologi untuk melaksanakan tugas-tugas yang berharga seperti sintesis ubat-ubatan dan pengeluaran biofuel. Oleh kerana sifat kompleks sistem biologi, dan sama dengan bidang kejuruteraan, kaedah pengiraan dan reka bentuk bantuan komputer (CAD) alat, diperlukan untuk mengendalikan kerumitan ini. Dalam kajian ini Cads yang sedia ada, bahasa dan model data untuk biologi sintetik akan diterokai. Matlamatnya adalah untuk memberi gambaran untuk biologi sintetik untuk mengambil kesempatan daripada alat-alat yang ada pengiraan, dan untuk mengetahui jurang untuk bekerja.

Kata kunci: CAD; BioCAD; biologi sintetik; bioinformatik; bahasa; model data

1.0 INTRODUCTION

Synthetic biology is applying engineering standards to the biology. It has two main aims; first to plan and construct biological systems in order to enlarge the current comprehension of how they work, and second, to design biological systems to perform valuable tasks like microbes engineering to execute cancer tumors, medications synthesis economically by metabolic engineering, biofuels production, chemical material creation, bioremediation, business design, and chemical catalyzers design for novel reactions.[1,2]

Due to multiple scales, versatile and dynamic nature of biological systems and materials, computational methods and computer aided design (CAD) tools are required to handle this complexity. In the recent years, several CADs and programming languages for synthetic biology have been developed to support the design of new systems. These tools need to access the libraries of parts and query them.[1]

Because the synthetic biology is an emerging field, there is only very few reviews on its computational tools. It means that, it is difficult for a researcher in the field to find out what kinds of tools are available and how to take advantage of them.

R. Weis *et al.*[3] explored the computational methods for synthetic biology. D. Chandran *et al.*[4] investigated the CADs. The existing programming languages in the field were reviewed by J. Beal *et al.*[5] The data models were discussed in M. Galdzicki *et al.* work.[6] A survey has been done by L. Kahl *et al.* to find out the technologies which are used by the synthetic biologists in their projects.[7]

In this review, existing CADs, languages and data models for synthetic biology are explored. The goal is to provide a broad view of the current available computational facilities for synthetic biologists who intend to use them, as well as for the researchers who seek to propose new tools.

The advantage of the currecnt review, compared with the similar works, is that it integrates discussion on the CADs,

languages and data models in one place. In addition, it aims to review more computational tools than the previous ones.

2.0 SYNTHETIC BIOLOGY

Synthetic biology, a subfield within biological design concerned with engineering organisms to perform novel functions and develop the means by which organisms can be engineered easily and robustly, is a relatively young field with a large potential for growth. Much of the work in the field includes designing and constructing genetic circuitry, joining biological "parts" (e. g. Promoters, ribosome binding sites, and translated sequences) to form the basis for biological "devices," organisms that have an engineered, well-specified input-output behavior.

Synthetic biology attempts to modularize biology to make it tractable. Hence, it focuses on synthesis, abstraction, and standardization of biological components. Synthesis is creating reusable and modular parts (regularly as DNA segments). Abstraction is the implementation of synthesized parts by function instead of composition (e.g., "this is a promoter" instead of "this is ttgacagctagctcagtcctaggtataatgctagc") which can aid the design of more complex biological systems. Standardization means that every synthesized part and device is well-characterized, reproducible, and exchangeable.[8]

Synthetic biology has the potential to transform how human beings interact with the environment and how they approach health. Traditional genetic engineering approaches to solving complex problems, typically focus on working on one or a few genes. Synthetic biology, by contrast, approaches these problems from a novel, engineering-driven perspective that focuses on wholesale changes to the existing cellular architectures and the construction of elaborate systems from the ground up. Synthetic biology has the potential to fabricate practical organisms that could clean hazardous waste in inaccessible places, to use plants to sense chemicals and respond accordingly, to produce clean fuel in an efficient and sustainable fashion, and to recognize and destroy tumors. Whether addressing an existing problem or creating new capabilities, effective solutions can be inspired by, but need not mimic, natural biological processes. These new designs can potentially be more robust or efficient than the original systems.[3]

3.0 CADS FOR SYNTHETIC BIOLOGY

Comparable to other engineering fields like electrical and mechanical engineering, computer-aided design (CAD) has been proposed for synthetic biology. The goal of a synthetic biology's CAD (bioCAD) is to give a productive design flow to build and analyze biological systems.[4]

However, there is a paramount difference between biology and other disciplines. First, mechanisms of living organism's functions are not understood thoroughly to make prescient tools. Second, there are not accepted engineering methods in biology. Third, the system's modeling must be done based-on the biological parts. Fourth, there is no accepted standard to define a biological part and to construct a system with multiple parts. There are lots of unresolved problems, like forming DNA secondary structures and interfering with host cell mechanisms.

Due to a large number of unknown parameters and lack of full comprehension of biological mechanisms, quantitative modeling is not possible in most of the cases and just qualitative descriptions are provided. Biological system design is mostly exploratory instead of a logical procedure.

A recent review on BioCADs has been done by D. Chandran *et al*. [5]. In their paper, the authors state that the design procedure in a CAD starts with a user-defined specification and continues up to entering the building process. According to their theory, an ideal design methodology includes five main stages: specification, design, analysis, composition of parts, and assembly. It is essential to note that feedback and iteration play important roles in the design process. Figure 1 illustrates this design methodology.

In the following sub-sections, major existing synthetic biology CADs are described briefly. Table 1 summarizes their features.

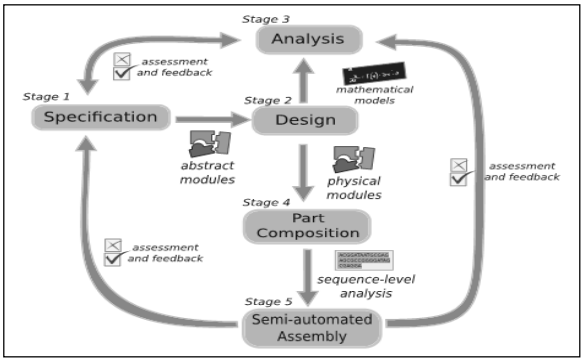


Figure 1 An ideal CAD methodology for synthetic biology[4]

3.1 BioJADE

BioJADE is the first implementation of abstract concept in genetic engineering and a tool to design genetic systems using graphical notations of genetic components. It is inspired by similar tools in electrical engineering. BioJADE key concept is "genetic component prototype": an abstract representation of a set of circuits (e.g. inverter) with similar function. BioJADE is able to translate the abstract representations to phenotypes and to simulate the resulted phenotypes. BioJADE developers built a database for genetic parts which finally evolved to MIT Registry of biological parts.[9]

Despite interesting features, BioJADE does not include the following capabilities:

- Definition of genetic parts.
- Definition of part functional properties.
- Finding the optimal combination of parts in a construct.

3.2 Clotho

Clotho is a design toolset for designing synthetic biological systems. It is inspired by Electronic Design Automation and its architecture is based-on the Platform-based Design (PBD). Clotho's structural modules are ClothoCore, ClothoHubs, ClothoConnections, ClothoData, ClothoAlgorithms and PoBolBindings. The general design flow in Clotho includes the following steps[10,11]:

1. Connect to a DB.
2. Associate fields to a standard semantic definition (PoBol).
3. Process, assemble and analyze.
4. Save the new design as a part to the DB.

3.3 DeviceEditor

DeviceEditor is a web-based and graphical bioCAD tool to simulate the visual whiteboard design process which is common in biological laboratories. DeviceEditor's main novelties are visual combinatorial library design, integration with DNA assembly automation, and a graphical user interface for the creation and modification of design specification rules. DeviceEditor provides prototype creation facility using standardized, functional, and visual abstractions. In addition, it is able to integrate with other bioCADs and software platforms. [12]

3.4 iBioSim

iBioSim is a tool for constructing synthetic biological systems which can be used for two purposes: to analysis a natural genetic circuit and to design a new synthetic genetic circuit.[13]

To analysis a circuit, the following procedure must be pursued:

1. Gathering gene expression data (by means of for example microarrays).
2. Analyzing the data and generating the Genetic Circuit Model (GCM).
    a. GCM is a graphical specification language which describes a circuit at a higher level of abstraction than SBML, by including just the important species and resections.
    b. GCM tuple is: <S, P, G, I, Sd>
        i. S: Proteins.
        ii. P: Promoters.
        iii. G: Species that are produced by a promoter.
        iv. I: Influences of species on a promoter (A: activation, R: representation).
        v. Sd: These species only influence other species after forming a dimer.
3. Converting GCM to SBML to be ready for simulation tools.

To design a new synthetic genetic circuit, the user starts with a GCM and modifies it iteratively to reach the desired behavior. IBioSim will generate the DNA sequence of the GCM.

3.5 j5

j5 is a web-based software tool which automates the design of scar-less multipart DNA assembly protocols including SLIC, Gibson, CPEC, and Golden Gate. The key innovations of the j5 design process are cost optimization, using DNA synthesis when it is cost-effective, the enforcement of design specification rules, hierarchical assembly strategies to remove likely assembly errors, and the instruction of manual or automated construction of scar-less combinatorial DNA libraries. These innovations save researchers time and effort, reduce the frequency of user design errors and off-target assembly products, decrease research costs, and enable scar-less multipath and combinatorial DNA construction of scales impractical without computer-aided design[14].

3.6 ProMoT

ProMoT is a tool for modular model design. Genetic circuit design is done by placing biological parts on a canvas and by connecting them by means of "wires" that enable flow of signal carriers. ProMoT supports two different modeling approaches, qualitative and quantitative. The qualitative approach is a description of the system by logical equations where the quantitative method is based on differential algebraic equations (DAEs). The final code associated with a circuit, can be exported into MATLAB or SBML format (Level-1 and Level-2) to be simulated.[15, 16]

3.7 SynBioSS

SynBioSS is a tool for synthetic network construction. It builds a kinetic model of the desired construct, retrieves required kinetic information and simulates the model. The results are probability distributions of dynamic biological phenotype. The components of the SynBioSS are Designer, WIKI and Simulator which are described below:[17-19]

- Designer:
    - The user builds a construct from BioBrick parts.
    - The designer generates all the reactions (kinetic model) representing transcription, translation, regulation, induction and degradation, in SMBL or NetCDF files. It has simple rules of how molecules interact in regulatory networks.
- WIKI: It is a database in wiki format and an extension of MediaWiki software, to store, retrieve, view and edit of species and reaction data.
- Simulator: There are two types of simulations in SynBioSS, both based-on the Hy3S; desktop and supercomputer.

3.8 TinkerCell

TinkerCell is a CAD tool to construct biological networks graphically and analyze their behavior. It takes advantage of third-party programs' functions and has a flexible and extensible design to adapt itself to the future changes in the synthetic biology field. Features of TinkerCell's Design are [20-22]:

- Providing mathematical description of the models (parameters, equations of dynamics of the part, etc.).
- Providing several modeling methods (different ways of defining the dynamics of the model).
- Extracting dynamics of a model automatically.
- Component based modeling. It allows the user to build models by choosing and connecting components from the parts catalogue.
- Ability to reuse existing smaller circuits in order to build larger ones.
- Extensibility. Extensions are programs (C++) which can be added to the TinkerCell without modifying it. They can be removed or replaced. Most of the works are done by extensions in TinkerCell, like driving rate equations and graphical user interfaces.
- Supporting the third-party functions (e.g. Analysis) by providing a rich API which is callable from C and Python programs.
- Layered architecture:
    - Layer 1: Core library.
    - Layer 2: C++ extensions.
    - Layer 3: C and Python extensions.
- Supporting standard data models like SBOL. TinkerCell can be used to query parts DBs, biological models and connect and analysis functions. In

addition, TinkerCell can connect mathematical models to biological parts.

Table 1 Synthetic biology CADs and their features

#	CAD	Description
1	**BIOJADE**	- The first tool to design genetic systems using graphical notations of genetic components, inspired by similar tools in Electrical engineering. - Its key concept: "genetic component prototype"; an abstract representation of a set of circuits (e.g. inverter) with similar function. - Being able to translate the abstract representations to phenotypes and simulate the resulted phenotypes. - Including a database for genetic parts which finally evolved to MIT Registry of Biological Parts.
2	**CLOTHO**	- Inspired by Electronic Design automation. - Having "Platform-based Design (PBD)" architecture. - Including the following structural modules: ClothoCore, ClothoHubs, ClothoConnections, ClothoData, ClothoAlgorithms and PoBolBindings.
3	**DEVICEEDITOR**	- A web-based and graphical bioCAD to simulate the visual whiteboard design process. - Being able to integrate with other bioCADs and software platforms. - Its main novelties: o Visual combinatorial library design. o Integration with DNA assembly automation. o A graphical user interface for the creation and modification of design specification rules.
4	**IBIOSIM**	- A tool to construct synthetic biological systems. - Circuit analysis steps: o Gathering gene expression data (e.g. By means of microarrays). o Analyzing the data and generating the Genetic Circuit Model (GCM) which is a graphical specification language to describe a circuit at a higher level of abstraction than SBML. o Converting GCM to SBML to be ready for simulation tools. - Procedure to design a new synthetic genetic circuit: o Starting with a GCM. o Modifying GCM to reach the desired behaviour. o Generating the DNA sequence of the GCM by iBioSim.
5	**J5**	- A web-based software tool to automate the design of scar-less multipart DNA assembly protocols including SLIC, Gibson, CPEC, and Golden Gate. - The key innovations: o Cost optimization. o Using DNA synthesis when it is cost-effective. o The enforcement of design specification rules. o Hierarchical assembly strategies to remove likely assembly errors. o The instruction of manual or automated construction of scar-less combinatorial DNA libraries.
6	**PROMOT**	- A tool for modular model design. - Providing genetic circuit design by placing biological parts on a canvas and by connecting them by means of wires that enable flow of signal carriers. - Supports two modeling approaches: qualitative and quantitative. - Being able to export a circuit's code into MATLAB or SBML format for deterministic and stochastic simulation.
7	**SYNBIOSS**	- A tool for synthetic network construction. - Its components: o Designer: ▪ The user builds a construct from BioBrick parts. ▪ The designer generates all the reactions (kinetic model) representing transcription, translation, regulation, induction and degradation, in SMBL or NetCDF files. o WIKI: is a database in wiki format to store, retrieve, view and edit of species and reaction data. o Simulator: There are two types of simulations in SynBioSS based-on the Hy3S; desktop and supercomputer.
8	**TINKERCELL**	- A tool to construct biological networks graphically and analyze their behavior. - Its features: o Providing mathematical description of the models. o Providing several modeling. o Extracting dynamics of a model automatically. o Component based modeling. o Ability to reuse existing smaller circuits in order to build larger ones. o Extensibility by C++ programs. o Supporting the third-party functions by providing a rich API which is callable from C and Python programs. o Layered architecture. o Supporting standard data models like SBOL. o Being able to query parts DBs, biological models and connect and analysis functions.

4.0 LANGUAGES

The idea behind high-level languages is automating problem solving by connecting sub-problems' solutions. High-level languages hide details of programmers to bring accessibility (requiring less knowledge), scalability and reliability to the programming task.[5]

A high-level language for bio-molecular systems is any system description language where the choice of actual parts may be left unspecified. There are a few developed languages, each one focuses on different aspects of design and automates different stages. Selecting a grammar for describing biological sequences is a trade-off between expressivity and the compilation complexity.

Despite their usefulness, high-level languages for synthetic biology have two main drawbacks as well:

1. The programmer gives up control of some aspects of the system.
2. The results are less efficient relative to hand-tuned designs.

The most mature and well-known languages for synthetic biology are (from the lowest level to the highest level) GenoCAD, Eugene, GEC and Proto. Beside mentioned languages, there are other ones which are mentioned in a review on synthetic biology's languages by Beal J. *et al*.[5] Piegon is another recent language developed by S. Bhatia *et al*.[23]

The major languages are described in the following sub-sections. A summary of the languages' features can be found in Table 2.

4.1 Antimony

Antimony is a simple, modular and text-based language which allows the user to create and combine biological models. LibAntimony is a library which provides the facility for other softwares to transfer the Antimony modules to their own formats. Antimony syntax is based-on the Jarnac [24]. Its features include:[25]

- Chemical reactions
  - Example:
    *J0:S1+S2□S3; K1*S1*S2; K1=21;*
- Modularity
  - Example 1: Using "is" to associate two variables together.
    import "model1.xml"
    import "model2.xml"
    model multiModule()
    A: test1(); //from model1.xml
    B: test2(); //from model2.xml
    A.PYR is B.pyruvade;
  - Example 2: Using input and output to associate two variables together.
    test1(glucose,pyruvate);
    test2(pyruvate,co2);
- Genetic Networks
  - The elements are listed in order:
    --P1--RBS1--G1--stop--
  - The rate Law of each element could be defined or calculated based-on the up-stream elements.
  - If a module has a part inside, it can be used in a new network.
  - Example:
    F2620 : bba_f2620(AHL);
    -- F2620--GFP--
- Other features: Ability to define variables, constants, interactions, components, events, rate rules and assignment rules.

4.2 Eugene

Eugene is a programing language for synthetic biology which is inspired by languages like Verilog and VHDL. Main capabilities of Eugene are[26-28]:

- Flexible part and device specification and composition.
- Design space exploration using a rule system.
- Interfacing with other simulation and assembly tools.

Eugene is composed of primitives, constructs, rules and functions:

- Primitive: txt, num, Boolean, txt[], num.
- Construct
  - Property: A name and a primitive type: *Property strength(num);*
  - Part
    - Definition: Part Promoter(ID, sequence, strength);
    - Declaration:
      Promoter P1(1,"TATATA", 30);
  - Device: Including parts and other devices. Devices are ordered 5' to 3':
    Device BBa_1(P1,GFP);
- Rule
  - Declaration: Describing the constraints on devices using rules operators: Rule R1(P1 BEFORE GFP);
  - Assert and Note:
    - Assert: Throws an exception: *Assert(R1);*
    - Note: Prints error messages: *Note(R2);*
    - Header file: Contains properties and part definitions, as well as part instantiations.
- Function: For instance print and permute. Permute generates all the permutation of a device by changing each part with its other instances.

At present, small molecule interactions cannot be defined in Eugene. But if sequences of parts are proper, these interactions will occur naturally in the physical device.

4.3 GEC

GEC is a language to represent the interactions between (potentially unspecified) proteins and genes in a modular way. Using GEC, the designer only should know the basic part types like RBS and determine the constraints (for example "at this point I need a negatively regulated promoter"). The compiler determines the actual parts from a DB. If there are several design options, GEC will find them and user can simulate them to select the best one and to refine the model by adding some constraints. Syntax and semantic definition of GEC is independent of choice of part types and properties, but translation to reactions is based on the part types and properties.[29]

GEC has a database of parts with their properties and a database of reactions. Reactions in GEC are represented by Language for Biochemical Systems (LBS).[30] In addition, stochastic and deterministic simulations are available (using Systems Biology Workbench or third-party tools). A graphical

tool is developed which includes a database editor, a GEC editor and a GEC compiler.[29]

GEC's basics are summarized as follows:

- Part types:
  - Example*:*
    r0040:prom; b0034:rbs; c0040:pcr; b0015:ter
- Part variables and properties:
  - Example: Rewriting the previous example which results in 4 designs:
    X1: prom<neg(Y)>;
    X2:rbs;
    X3:pcr<codes(Y)>;
    X4:ter;
- Parameterized modules.
- Comportments and reactions: Reactions as additional constraints on parts and compartments (location of parts like a specific cell) can be defined in GEC.

4.4 GenoCAD

GenoCAD [31-37] is a framework, based-on the attribute grammars and is developed to:

- Represent the biological functions of genetic parts.
- Formalize the dependency of parts' functions on their context.
- Translate a DNA sequence to a model to predict their behavior.

GenoCAD includes a formal semantic model which represents the dynamics of the DNA sequence using attribute grammars. The proposed compiler translates the syntactic information coded by DNA sequences to a dynamic model of phenotype. The workflow of GenoCAD is as follows:

1. Converting the sequence to a series of genetic parts by lexer (scanner).
2. Checking the structural consistency of the sequence with the syntax by parser and generating pars trees for valid sequences.
3. Translating the sequence to a mass action model of the molecular reactions using the parse tree, attributes (properties of individual parts or combination of parts) and semantic actions (associated with the production rules).

4.5 Proto

Proto has proposed an automatic compilation technique to convert a high-level description into an abstract genetic regulatory network. The compiler optimizes the network and generates a simulation of it. For this purpose, Proto includes motif-based compilation. The networks are organized into a set of promoter-genes-terminator functional units. Each unit has a known input-output relation. Parts in units can be selected from a database in a way that are compatible with each other and with their input-output characteristics. Proto currently uses ODE for mathematical modeling, but in the future, stochastic modeling will be added.[38-40]

Regarding the motif-based compilation, each primitive has an associated motif. To convert the graph to an abstract network, the compiler first replaces each operator/primitive with its motif and each edge and variables (of motifs) to a regulatory protein. Then, all are connected together. Choice of molecules and sequences is left which can be made by GEC, Eugene or MatchMaker.

Proto relation with other similar tools can be summarized as follows:

- It can integrate with "assembly-level languages" like SBML, SBOL and CellML.
- It simplifies model building of antimony, Little B and ProMoT.
- It uses Eugene [26-28] and GenoCAD [31-37] to improve the analyzing and converting networks to physical implementation.

Table 2 Synthetic biology languages and their features

#	Language	Description
1	**ANTIMONY**	- Simple, modular and text-based. - Allowing the user to create and combine biological models. - Providing LibAntimony, a library for other softwares to transfer the Antimony modules to their own formats. - Based-on the Jarnac's syntax. - The capability to define: o Chemical reactions o Genetic Networks o Variables, constants, interactions, components, events, rate rules and assignment rules.
2	**EUGENE**	- Inspired by languages like Verilog and VHDL. - Composed of primitives, constructs, rules and functions. - Main capabilities: o Flexible part and device specification and composition. o Design space exploration using a rule system. o Interfacing with other simulation and assembly tools.
3	**GEC**	- A language to represent the interactions between potentially unspecified proteins and genes in a parameterized modular way. o The designer only should know the basic part types like RBS and determine the constraints (e.g. a negatively regulated promoter). The compiler determines the actual parts from a DB. o If there are several design options, GEC will find them and user can simulate them to select the best one. - Independence of the GEC's syntax and semantic definition of choice of part types and properties (but the translation to reactions is based on the partition types and properties). - Including a database of parts with their properties and a database of reactions (represented by LBS). - Including stochastic and deterministic simulations using Systems Biology Workbench or third-party tools. - Including a graphical tool: a database editor, a GEC editor and a GEC compiler.
4	**GENOCAD**	- A framework based-on the attribute grammars which is designed to:

			○ Represent the biological functions of genetic parts. ○ Formalize the dependency of parts' functions on their context. ○ Translate a DNA sequence to a dynamic model to predict their behavior.
5	**PROTO**	-	A technique to convert a high-level description into an abstract genetic regulatory network. The compiler optimizes the network and generates a simulation of it. - Including motif-based compilation: ○ Each primitive has an associated motif. ○ The compiler first replaces each operator/primitive with its motif and each edge and variables (of motifs) to a regulatory protein. ○ Choice of molecules and sequences is left which can be made by GEC, Eugene or MatchMaker. ○ It uses ODE for mathematical modeling (Stochastic modeling will be added later). - Proto relation with other similar tools: ○ It can integrate with "assembly-level languages" like SBML, SBOL and CellML. ○ It simplifies model building of antimony, Little B and ProMoT. ○ It uses Eugene and GenoCAD to improve the analyzing and converting the networks to physical implementation.

5.0 DATA MODELS

Because of the diversity and size of biological data (large number of components, interacting physically and chemically at multiple time and spatial scales) standardization is required to design and analysis synthetic biological circuits by means of computational tools. The case is similar to Bioinformatics field which can be a reference to learn from its successes and failures.[6] In the following sub-sections, available data models for synthetic biology are explored and a summary is presented in table 3.

5.1 Visual Representation Standards

Visual standards are required in the fields which use diagrams to exchange information. There are two approaches to define the required symbols [6]:

1. Selecting the used symbols in the community (e.g. SBGN (systems biology graphical notion) [41]).
2. Creating new symbols (e.g. SBOLv [42]).

5.2 Software Data Models

Most of the synthetic biology's softwares, have their own data model and use import and export functions to support standard formats. Some of them have standards which allow adding customs information (e.g. GeneBank and SBML). However, there are some softwares with completely different data models, so they cannot use standard formats. Examples are TinkerCell (diagrams) and GenoCAD (grammars).[6]

5.3 SBOL

SBOL is proposed by "Synthetic Biology Data Exchange Group". The goal is describing data in a standard but extensible outline to make the electronic information exchange possible. SOBL is the successor of ProBol (Provisional BioBrick Language), an initial attempt at defining a minimal data model for BioBrick repository entries.[1, 6] SBOL includes two main projects: SBOL-Semantic and SBOL-Visual.[42]

SBOL-Semantic uses information technology for data on the web. Its base is a core ontology which is a set of fundamental synthetic biology concepts and their relationships including Part, Sequence Features and Assembly Standards, as well as how they connect to each other. The ontology conforms to the W3C recommended technology for semantic web (RDF/OWL) and it is written in OWL, the W3C standard for ontology definition.

SBOL-Visual (SBOLv) [42] is a symbolic representation based on symbols which are already in use in the community. The most important feature of a standard is reducing the ambiguities such as synonymy (multiple terms for the same concepts) and homonymy (one symbol for multiple concepts). The key contribution of SBOL is limiting the number of symbols for a concept. The metric for the success of a standard is the amount of community's acceptance. Regarding the use of SBOL in tools like Clotho, DeviceEditor, GenoCAD, SynBioSS and TinkerCell, it could be considered successful.

Table 3 Synthetic biology data models and their features

#	Data Model	Description	Example(s)
1	**Visual representation standards**	Visual standards in order to use diagrams for exchanging the information.	SBOLv
2	**Software data models**	Data models which are designed specifically for each software.	TinkerCell' diagrams, GenoCAD's grammar
3	**SBOL**	The successor of ProBol and a modeling approach to describe data in a standard but extensible outline.	SBOL-Semantic, SBOL-Visual (SBOLv)

6.0 DISSCUSSION

Regarding the data models for synthetic biology, it seems that tools designers are shifting gradually from software-specific data models to the standard models like SBOL. It makes interoperability between softwares even more possible.

According to the mentioned design methodology (Figure 1), the tools and languages explored in this paper, can be categorized based-on the stage that they can be used for. Table 4 shows this kind of categorization.

The advantages of existing CADs and languages and are presented in table 5 and 6 respectively.

Table 4 Categorization of the tools based-on the design stage they can be used for it

Function	CAD/Language
Stage 1: Specification	Eugene, DeviceEditor.
Stage 2: Design	Antimony, BioJADE, Clotho, DeviceEditor, GEC, GenoCAD, iBioSim, ProMoT, Proto, SynBioSS, TinkerCell.
Stage 3: Mathematical Analysis	No specific tool for synthetic biology. Usually tools from "Systems Biology" are used.
Stage 4: Biological Part Composition	GenoCAD, j5, TinkerCell
Stage 5: Assembly	J5

Table 5 Comparing the CADs for synthetic biology

#	CAD	Advantage(s)
1	**BIOJADE**	- Included a database for genetic parts which finally evolved to the MIT Registry of Biological Parts.
2	**CLOTHO**	- Having "Platform-based Design (PBD)" architecture. - Connecting users to repositories of biological parts. - Making it easier to share data.
3	**DEVICEEDITOR**	- A web-based tool. - Being able to integrate with other bioCADs and softwares. - Visual combinatorial library design. - Integration with DNA assembly automation. - Including a GUI for the creation of design specification rules.
4	**IBIOSIM**	- Converting its internal data model representation into SBML to be ready for simulation tools.
5	**J5**	- A web-based tool. - Supporting several scar-less multipart DNA assembly protocols. - Cost optimization. - The enforcement of design specification rules.
6	**PROMOT**	- Supports two modeling approaches: qualitative and quantitative. - Being able to export a circuit's code to MATLAB or SBML format for simulation.
7	**SYNBIOSS**	- A web-based and desktop-based tool.
8	**TINKERCELL**	- Providing mathematical description of the models. - Providing several modeling. - Ability to reuse existing smaller circuits in order to build larger ones. - Extensibility by C++ programs. - Supporting the third-party functions. - Layered architecture. - Supporting standard data models like SBOL. - Being able to query parts DBs, biological models and connect and analysis functions.

Table 6 Comparing the languages for synthetic biology

#	Language	Advantage(s)	Limitation(s)
1	**ANTIMONY**	- Simple. - Modular. - Providing LibAntimony to transfer the Antimony modules to other software formats.	- Text-based.
2	**EUGENE**	- Design space exploration using a rule system. - Interfacing with other simulation and assembly tools.	- Text-based. - Small molecule interactions cannot be defined in Eugene.
3	**GEC**	- Modular. - Graphical. - Syntax and semantic independence of choice of parts. - Including a database of parts and reactions. - The designer only should know the basic part types and determine the constraints. The compiler determines the actual parts from a DB.	- Being dependent on the availability of detailed chemical reaction models with precisely quantified rate constants.
4	**GENOCAD**	- Proving a web-based user interface. - Using the attribute grammars to formalize the biological circuit definition.	- Being unfamiliar to non-computer Science users.
5	**PROTO**	- Choice of molecules and sequences can be made by GEC, Eugene or MatchMaker. - Integration capability with some other languages.	- Relatively hard to learn.

■7.0 CONCLUSION AND FUTURE DIRECTION

Computational approaches and tools are vital for the success of the emerging field of synthetic biology. Several tools have been proposed and are already in use in the community. In-depth understanding of such kind of tools provides opportunities for the synthetic biologists to fully take advantage of them, and for the researchers and developers to find the gaps which need to be filled.

The major difficulty in proposing and designing computational tools for biology is that there are not recognized methods in biology itself as well as full understanding of the biological behavior. Hence collaborative efforts in the community could be effective to tackle such difficulties.

Acknowledgement

This work was supported by the Ministry of Higher Education of Malaysia [Grant No. KPT.B.600-18/3 (115) to N.Habibi]; and Universiti Teknologi Malaysia.

References

[1] M. Galdzicki, C. Rodriguez, D. Chandran, H.M. Sauro, J. H.Gennari. 2011. PLOS ONE. 6(2).
[2] J. T. MacDonald, C. Barens, R. I. Kitney, P. S. Freemont, G. B. Stan. 2011. *Integr. Biology (Camb).* 3(2): 97–108.
[3] P. E. M. Purnick and R. Weiss. 2011. *Nature Reviews Molecular Cell Biology.* 10: 410–422.
[4] D. Chandran, F. T. Bergmann, H. M. Sauro and D. Densmore. 2011. *Design and Analysis of Bio-molecular Circuits*. Springer-Verlag. 203–224.
[5] J. Beal, A. Phillips, D. Densmore, and Y. Cai. 2011. *Design and Analysis of Bio-molecular Circuits.* Springer-Verlag.
[6] M. Galdzicki, D. Chandran, J. H. Gennari and H. M. Sauro. 2011. *Design and Analysis of Bio-molecular Circuits.* Springer-Verlag. 281–293.
[7] L. Kahl and D. Endy. 2013. *J Bio Eng. 10*. 7(1): 13.
[8] G. H. McArthur and S. S. Fong. 2010. *Journal of BioMed and BioTech.*
[9] J. A. Goler. 2004. CSAIL-MIT.
[10] D. Densmore, A. V. Devender, M. Johnson and N. Sritanyaratana. 2009. ACM.
[11] B. Xia, S. Bhatia, B. Bubenheim, M. Dadgar, D. Densmore, and J. C. Anderson. 2011. *Methods in Enzymology*. 498.
[12] J. Chen, D. Densmore, T. S. Ham, J. D. Keasling and N. J. Hillson. 2012. *Journal of Biological Engineering*. 28: 6(1):1
[13] C. J. Myers, N. Barker, K. Jones, H. Kuwahara, C. Madsen, N. P. Nguyen. 2009. *Bioinformatics.* 25(21): 2848–9.
[14] N. J. Hillson, R.l D. Rosengarten and J. D. Keasling. 2012. *ABC Synthetic Biology.* 1(1): 14–21.
[15] M. A. Marchisio and J. Stelling. 2008. *Bioinformatics*. 24(17). 1903–1910.
[16] S. Mirschel, K. Steinmetz, M. Rempel, M. Ginkel and E. D. Gilles. 2009. *Bioinformatics.* 25(5): 687–689.
[17] A. D. Hill, J. R. Tomshine, E. M. B. Weeding, V. Sotiropoulos and Y. N. Kaznessis. 2008. *Bioinformatics.*
[18] E. Weeding, J. Houle, Y. N. Kaznessis, 2010. *Briefing in Bioinformatics.* 11(4): 394–402.
[19] Y. N. Kaznessis. 2011. *Methods in Enzymology*. 498.
[20] D. Chandran, F. T. Bergmann and H. M. Sauro. 2009. *Biol Eng*. 3: 19.
[21] D. Chandran, H. M. Sauro and D. Densmore. 2010. *Bioeng Bugs*. 4: 274–281.
[22] D. Chandran and H. M. Sauro, 2012. *ACS Synthetic Biology*. 1: 353−364
[23] S. Bhatia and D. Densmore. 2013. *ACS Synthetic Biology.*
[24] H. M. Sauro. 2000. Proceedings of the 9th International Meeting on BioThermoKinetics. Stellenbosch University Press.
[25] L. P. Smith, F. T. Bergmann, D. Chandran and H. M. Sauro. 2009. *Bioinformatics.*
[26] D. Densmore, J. T. Kittleson, L. Bilitchenko, A. Liu and J. C. Anderson. 2010. IEEE.
[27] L. Bilitchenko, A. Liu., S. Cheung, E. Weeding, B. Xia, M. Leguia, J. C. Anderson and D. Densmore. 2011. PLOS ONE. 6(4): e18882.
[28] L. Bilitchenko, A. Liu and D. Densmore. 2011. *Methods in Enzymology*. 498.
[29] M. Pedersen and A. Phillips. 2009. *Journal of the Royal Society.*
[30] M. Pedersen and G. D. Plotkin. 2010. *Trans. on Comput. Syst. Biol.* XII, LNBI 5945. Springer-Verlag. 77–145.
[31] Y. Cai, B. Hartnett, C. Gustafsson and J. Peccoud. 2007. *Bioinformatics.* 23(20): 2760–2767.
[32] J. A. Goler, B. W. Bramlett and J. Peccoud. 2008. *Trends in Biotechnology*. 26(10): 538–44.
[33] M. J. Czar. Y. Cai and J. Peccoud. 2009. *Nucleic Acids Research.* 37.
[34] Y. Cai, M. W. Lux, L. Adam and J. Peccoud. 2009. PLOS Computational Biology. 5(10): e1000529.
[35] Y. Cai, M. L. Wilson and J. Peccoud. 2010. Nucleic Acids Research. 38(8): 2637–2644.

[36] Y. Cai. Ph.D. Dissertation. Virginia Polytechnic Institute and State University. 2010.
[37] M. L. Wilson, R. Hertzberg, L. Adam and J. Peccoud. 2011. *Methods in Enzymology*. 498.
[38] J. Bealand J.Bachrach. 2009. IEEE.
[39] J. Beal, T. Lu and R. Weiss. 2011. PLOS ONE. 6(8): e22490.
[40] J. Beal, R.Weiss, F. Yaman, N. Davidsohn and A. Adler. 2012. MIT-CASIL.
[41] http://www.sbgn.org.
[42] J. Quinn, J. Beal, S. Bhatia, P. Cai, J. Chen, K. Clancy, N. Hillson, M. Galdzicki, A. Maheshwari, U. P, M. Pocock, C. Rodriguez, GB. Stan, and D. Endy. 2013. DOI: 1721.1/78249.