

'We want to do for biology what Intel does for electronics': re-factoring biology as a software engineering enterprise.

Adrian Mackenzie

CESAGen - Centre for Social and Economic Aspects of Genomics

Lancaster University, LA1 4YD, UK

a.mackenzie@lancaster.ac.uk

Abstract

The paper describes certain trends in the field of synthetic biology or 'synbio' from the perspective of software engineering practices. The nascent field of synbio is relying heavily on software engineering approaches such as modularity, platforms, registries, libraries, standards and re-factoring to develop technologies such as biofuels, drugs, assays, biosensors and crops. The idea is that shared codifications of techniques and processes of biological engineering will lead to an accelerating rate of invention in biology. As well as invoking software engineering as a design philosophy to be 'ported' into biology, the everyday practices of synbio are saturated by web software cultures of collaboration and participation (wikis, blogs) as well as relying on web-based technical services (such as DNA synthesis, sequences database and searching tools). One analytical question for sociologies of synbio would be: does the model of software engineering and design abstraction begins to break down in synbio? Based on a small case study of two different synbio projects, this paper will sketch a preliminary answer to that question.

Introduction: programmable wetware?



How would you program this? This is the annual dogwood, or *artemisia annua*, a plant from northern China. The Chinese name is qinghao, green wormwood. This plant is very important as an anti-malarial treatment. An extract from its leaves, artemisinin is now used globally in treating chloroquine-resistant malaria. The problem is that the almost all varieties of annual dogwood, including commercial hybrids, produce relatively

little artemisinin.

Hence, people are programming yeast to produce artemisin. Amyris Technologies is industrializing the production of artemisinin by reprogramming bacteria. What is interesting about this from the perspective of programming is that they engineered whole new metabolic pathways from yeast into the common bacteria *e.coli*, thereby allowing to produce high concentration of artemisinin. (Use 2003 reference, Keasling).

In showing you annual wormwood, I want to approach programming, programmers, and programmability from a slightly unfamiliar angle: contemporary biology, and in particular, the emerging field of synthetic biology. So, rather than software, or hardware, I'm going to focus on 'wetware' (as used in the sense of living things). Of course, the programmability of wetware, of living things, is inseparable from chips and code anyway. The question, today, is: how and in what

ways will the programming of living things be implemented? In the last 3 years (2005-8), synthetic biology or 'synbio' has been emerging with high media visibility at the intersection between life sciences and information technology. Although the boundaries of the field are still very much in flux, it uses techniques and processes to reconfigure microbial lifeforms for attractive ends such as cheap biofuels or drug synthesis. It seeks to convert the massive growth in genomic knowledges and techniques into designed products. The anti-malarial compound artemisinin is one of the first practical, commercial products of synthetic biology. Synbio is often said to be the 'next technological revolution'.

We can understand synbio as an attempt to extend programming and software culture in general into biology and biotechnology. At the moment, synbio feels a bit like the RadioShack electronics circuit kits that I used to buy when I was a kid. Using a small set of electronic components, these hobby kits did things like turn a set of LEDs on and off, or play some simple tunes or act like a metronome. Contemporary synbio projects most do things like get some bacteria to flash lights on and off, or act like boolean logic gates. However, they are also doing some serious things like synthesising antimalarial drugs. I'll have more to say about what is happening in synthetic biology in a minute, but to start I want to focus on a single quote, the quote in the title: 'we want to do for biology what Intel does for microelectronics.' The quote comes from George Church, a Professor of Genetics at Harvard Medical School. Along with Walter Gilbert, he invented the first direct genomic sequencing method in 1984. He helped initiate the Human Genome Project in 1985. He recently co-founded with Drew Endy at MIT a company called Codon Devices. This is a company that does gene synthesis:

In late 2004, multiple breakthrough technologies gave rise to Constructive Biology™, a fundamentally new era of gene design and construction. With its proprietary BioFAB™ platform, Codon Devices designs and constructs engineered devices—such as full-length genes, operons, pathways, and ultimately genomes—quickly, accurately and at low cost.

<http://www.codondevices.com/science.aspx?id=58>

The ultimate goal here is fabricating genomes to order. Making genomes is a bit like making semiconductor chips. They are expensive to make. Once genomes can be made to order, biology becomes hackable or programmable. The programming of living things, if it happens (and it seems likely) poses very interesting and challenging problems for design thinking and engineering.

Let's return to the material conditions of the programmability of computers. We can program today because of the rise of microelectronics. Intel Corporation epitomises this rise. By the early 1990s, Intel's 16-bit microprocessors came to dominate the installed base of personal computers. In 1981, Intel had around 2% of the installed base. By 1992, it had over 75% of the installed base (Campbell-Kelly 2003, 238). The story of what happened in those 10 years has been often told: the IBM compatible PC standard appeared and took hold of the software industry. At the same time, Moore's Law has a hold on microelectronics: the number of transistors on integrated circuit chips doubles every month, including on Moore's own companies chips, Intel 8000 series.

So Church, Endy and others involved in synthetic biology at universities and companies around the world want to do for biology and biotechnology, what Intel did for microelectronics and hence for software. However, this project of doing for biology what Intel did or does for microelectronics cannot happen yet. As Church says:

It's like we're in the '50s in the sense that we're just beginning to get comfortable with our parts the way in the '50s they were getting comfortable with transistors and capacitors and what-not. We're very primitive. But then, we're also kind of even with the electronics-computing industry in that we have some of the amazing recursiveness going already that they didn't get until recently i.e. computer-aided design (CAD), where you used computers to design electronics, and compilers to help make compilers

http://www.edge.org/3rd_culture/church06/church06_index.html

On the one hand, parts and components are just beginning to take shape in biology. The techniques of fabrication of these parts are just starting to be automated. On the other hand, Church suggests, synbio is level with contemporary microelectronics in that it has 'some of the amazing recursiveness going already.' He is referring to the ways in which chip-designers use software to help design semiconductors for computers. This combination of being 'primitive' and 'recursive' makes synbio an interesting challenge to think about in terms of design, engineering, and ultimately, in terms of notions of programmability.

In introducing you perhaps for the first time to synbio, I will make use of just two main examples here. They are both from 2007. There is a tension between these two synbio examples that is interesting in its own right. However, for the purposes of this paper, I will focus on what it means for *programmability*. Under what conditions will biology become programmable? My focus here is not purely technical. Rather I am interested in how the programmability of biology is being configured from a number of different perspectives simultaneously. Further down the track, I'm interested in what makes it difficult for biology to take on programming or even software engineering as a model.

My paper draws on a background of a dozen policy, ethics and governance papers and reports published by various participants. But it focuses on two examples: firstly, a paper published in *Science* in January 2008 describing the first completely synthetic genome, and secondly, *ElectroEcoBlu*, a prize-winning project to build an environmental biosensor carried out by a UK university for the International Genetically Engineered Machine (iGEM) 2007 competition. The very-high profile publication of the first construction of a synthetic genome by researchers comes from the J. Craig Venter Institute (JCVI). The genome is called JCVI-1.0. By contrast, *ElectroEcoBlu* is a project from Glasgow University entered in iGEM 2007, the International Genetically Engineered Machine competition. It detects petrochemical pollutants.

The design ambitions of synbio

Like many other biosciences and biotechnologies today, synbio inhabits an intersection between network cultures and a biotechnical sensibility. It is an attempt to configure that intersection via a combination of design processes, commercial services, models and platforms. We need to analyse synbio, I would argue very much in the context of ICTs and the internet, both as infrastructures and tools, and as a oft-cited model for synbio itself.

The intersection of 20th century biology and information systems is hardly new, and has been extensively analysed. **However**, in synbio, this intersection is being re-staged at various levels, and with heightened specificity. For instance, one framing uses microelectronics industries as a model. George Church, of Harvard Medical School, is quoted as saying:

We want to do for biology what Intel does for electronics. We want to design and manufacture complex biological circuits”, says George Church, professor of genetics at Harvard Medical School. IT is no longer just a tool to design and test biological circuits. For synthetic biologists it is also a major source of inspiration. The comparison with micro electronics is particularly apt. (Rinie van Est 2007, 4)

As we will see, much effort is being directed towards shaping synbio in terms of platforms, components and operating systems in biotechnology. It entails a re-shaping of biotechnology through design practices drawn from the worlds of software and microelectronics. For instance, a current EU project on coordination of synthetic biology in Europe again see synthetic biology in terms of software engineering design:

By applying the tool box of engineering disciplines such as ... computer sciences, including the vigorous application of modelling techniques and organizing the development of novel biological systems along a hierarchical systems architecture with defined and standardized interfaces, Synthetic Biology aims at no less than revolutionizing the way bioengineering is done today. If successful, Synthetic Biology will transform bioengineering into a highly successful and sustainable life science industry. (Emergence 2008)

The ideals of 'hierarchical systems architecture' and 'defined and standardized interfaces' is highly redolent of the design processes and abstractions that have been vital in the development of networks cultures. Many other examples of explicit specific borrowings from the worlds of ICTs, networks and microelectronics could be cited here.

Competing design cultures: infectious agents vs 'snap together'

So, what differentiates synbio from biotechnology more generally or perhaps from genetic engineering in particular? It seems to me that synbio attempts to align biological engineering with design practices and cultures that have been so productive around ICTs and in software in particular. This is my key observation actually: as in the world of software, the design culture in play around synthetic biology does not concern only biological processes itself, but myriad commercial services and products, modes of collaboration and scientific work, forms of regulation and governance. This is a significant shift for biotechnology.

While there is widely shared consensus on the need to frame synbio in terms of design, there is less agreement on how design is to be brought into biology, and what the integration of design into biology might mean. My two exemplars represent two different angles on the problem of designing living things.

The first design axis, represented by the entity JCVI-1.0, a synthetic mycoplasma genitalium genome, treats synbio as a process of building biological platforms by reverse engineering existing organisms. Such platforms then would support accelerated biotechnological experimentation as well as smooth regulation of well-defined social, economic and political issues. For instance, synbio minimises the confusing tangle of interactions that afflict many biotechnologies such as genetic modification of organisms by only relying on specific genes.

The second axis, represented by the entity *ElectroEcoBlu*, also treats design as a process of

abstraction. But it envisages abstraction as creating standard, reusable components that can be put together in an indefinite number of ways, thus allowing **many more** interactions via collections of parts. Collections of biological parts would allow a proliferation of diversely engineered inventions.

| | | |
|--------------------------------|------------------|----------------------------------|
| <u>Synbio entities:</u> | <i>JCVI-1.0</i> | <i><u>ElectroEcoBlu/iGEM</u></i> |
| Siting: | <i>Platforms</i> | <i>Standard parts</i> |
| Generative process: | <i>Iteration</i> | <i>Recursion</i> |

I don't want to say too much about the actual entities JCVI-1.0 and *ElectroEcoBlu*, but rather examine the design process through they come to life. It seems to me that this process is where a lot of the energy in synbio is generated.

Iterating infectious agents A – accelerating the process

In January 2008 *Science* published an article entitled 'Complete Chemical Synthesis, Assembly, and Cloning of a Mycoplasma genitalium Genome.' Gibson and co-authors including J.C Venter described the design and assembly of a complete bacterial genome at the J. Craig Venter Institute in Maryland, USA. Any paper with Craign Venter's name on it has to be read in the long line of Craig Venter-authored genome articles that have appeared since the early 1990s. Venter's articles often report on his achievements in terms of speed and scale, on being the biggest or the fastest. Venter is known for having led the private effort to sequence the human genome in the late 1990s.

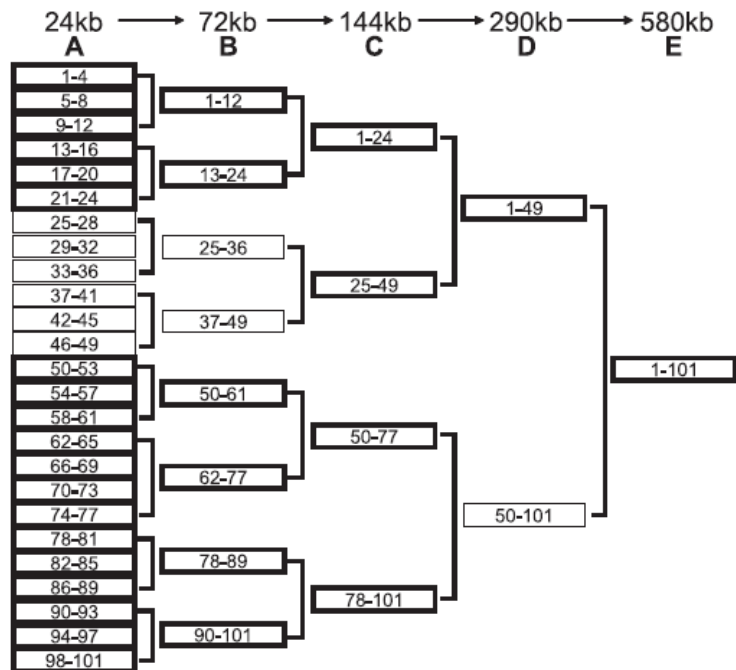


Fig. 2. A plan for the five-stage assembly of the *M. genitalium* chromosome. In the first stage of assembly, four cassettes are joined to make an A-series assembly approximately 24 kb in length (assembly 37-41 contained five cassettes). In the next stage, three A-assemblies are joined together to make a total of eight ~72-kb B-series assemblies (assembly B62-77 contained four A-series assemblies). The eighth-genome B-assemblies are taken two at a time to make quarter-genome C-series assemblies. These assemblies were all made by in vitro recombination (see Fig. 3) and cloned into *E. coli* using BAC vectors. Half-genome and whole-genome assemblies were made by in vivo yeast recombination. Assemblies in bold boxes were sequenced to verify their correctness. For the final molecule, the D-series half molecules were not employed. Rather, we assembled the whole molecule from the four C-series quarter molecules.

In this article, a relatively small bacterial genome became the object of design in several respects. Firstly, an accelerated process of fabricating a whole genome was a key design objective. Whole genome synthesis has not been done previously. Hence JCVI-1.0 is possibly the largest 'chemically synthesized molecule of defined structure' (1219). The process of synthesis itself is the **main** object of design.

First of all, the authors divided a bacterial genome into 'cassettes' that could be outsourced to commercial DNA synthesis services:

Synthesis of DNA the size of our cassettes has become a commodity, so we opted to outsource their production, principally to Blue Heron Technology, but also to DNA2.0 and GENEART. The main challenges in this project were the assembly and cloning of synthetic DNA molecules larger than those previously reported. (Daniel G. Gibson 2008, 1216)

The availability of cheap, web-based DNA synthesis services is a key component of synbio, and highlighted by many different commentators, both affirmatively and negatively. For our purposes, the possibility of designing genomic platforms depends on DNA-synthesis services. These services,

like the fabrication plants for silicon chips, are highly technical, and are constantly speeding up and increasing in capacity. The cost of DNA synthesis is also dropping rapidly (TBA:ref).

After receiving their DNA cassettes back from the DNA synthesizers, the biologists developed a hierarchical process of building 1/8, 1/4, 1/2 and finally a full, exact genome. Everything here is heavily couched in terms of efficiency of speed. Hence, although the process of assembling a complete genome does not yield any new understanding of living systems, or any particular product, it lays down the possibility of producing designed genomes quickly using readily available commercial services.

Accelerating design

What is the significance of this careful design process described in detail in the article? The genome has become synthetic because it was not made in inside the bacteria. It has designed in a computer system, and then produced mainly by DNA synthesising machines, before being assembled in a laboratory process. Given that trillions of genomes are being synthesised all the time in living systems, without any need for design, computers or commercial synthesis, what has been gained here? Firstly, the genome itself now shows that it has been coded by someone, rather than being the produce of evolution:

Short “watermark” sequences were inserted in cassettes 14, 29, 39, 55 and 61. Watermarks are inserted or substituted sequences used to identify or encode information into DNA. ... They allow us to easily differentiate the synthetic genome from the native genome (Daniel G. Gibson 2008,1215)

The watermarks, as lodged in the Genbank sequence, basically stamped authors names into the sequence:

VENTERINSTITVTE
CRAIGVENTER

HAMSMITH
CINDIANDCLYDE
GLASSANDCLYDE

<http://blog.wired.com/wiredscience/2008/01/venter-institut.html>

While Venter and Ham Smith's names have been inscribed into the genome to allow the synthetic genome to be differentiated from 'native genomes,' they also stand as flags in the sand. Struggles over intellectual property claims nearly always lie in the background of biotechnology. Reading the *Science* article alongside the patent applications suggests that a prime motivation of the design is accelerating the expansion of property claims. Hence, unlike most software engineering, the January 2008 paper needs to be read against the background of several patent applications dating from 2006. The titles of these patents are simply 'Synthetic genome' (Venter, Smith, and Hutchinson III 2007, 1) and 'Installation of genomes or partial genomes into cells or cell-like systems' [REF TBA]. They make huge claims over the very idea of designing and producing genomes. For instance,

Synthetic genomes of the invention may be introduced into vesicles ... to generate synthetic cells. Synthetic genomes of synthetic cells may be used for a variety of purposes.' - US Patent, abstract, (Venter, Smith, and Hutchinson III 2007, 1)

I want to leave outside the intellectual property issues here, although they are very significant in synbio, in the same way the licensing of source code is very important to contemporary software cultures.

Secondly, the synthetic genome is significant because it now *decouples* the way genomes are made from who puts these genomes together. Indeed, according to some views of synbio, “'decoupling' the design of engineered genetic material from the actual construction of the material' is the first 'benefit' of 'synthetic genomics' (Garfinkel et al. 2007, 10). In some ways this is similar to the early history of computing in which the separation between hardware and software was only achieved

over the course of a decade. At the moment, in the form of whole genome synthesis, we are witnessing a relation decoupling of biological platforms from

Of course, there is much more to be said about the synthetic genome. What I want to highlight here is that an early exemplar of synbio focuses on decoupling design from production in a way that is very familiar from ICT. The abstraction of software from hardware has been a key accomplishment, and a pivotal dynamic in information, network and digital cultures. We have programming and software today because of this decoupling that took place in the 1950s. The synthetic genome begins to do something similar since it is meant to act as a complete operating system for cellular hardware that is treated as separate and pre-given platform. The other key aspect here, although it is only implicit in the January 2008 paper, concerns re-factoring. Once synthetic genomes can be produced, then it seems that there is no obstacle to re-factoring them. As you will know,

'Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. Its heart is a series of small behavior preserving transformations. Each transformation (called a 'refactoring') does little, but a sequence of transformations can produce a significant restructuring.' <http://www.refactoring.com/>

A synthetic genome becomes a body of code subject to restructuring. Many synbio projects can be seen as re-factoring processes. While Venter's team has not re-factored *mycoplasma g.* very heavily in JCVI-1.0 very heavily, other synbio work have refactored existing organisms much more extensively. The most salient example here would be T7.1, a refactored version of the bacterial virus or bacteriophage, T7 that was first discovered in the 1940s. As the 2005 paper entitled 'Refactoring Bacteriophage T7' reports,

Here, we converted the genome of a natural biological system, bacteriophage T7, to a more structured design. (Chan, Kosuri, and Endy 2005, 3)

So that is all I am going to say about the synthetic genome announced in January. That such a genome can be made is really all the paper shows. The design work done on JCVI-1.0 is minimal. It has been watermarked, and also a gene for antibiotic resistance was inserted so that JCVI-1.0 can be separated out from wild-type *mycoplasma*. But this demonstration of the design and synthesis of a genome outside the cell opens up a myriad of design possibilities.

'They must snap together' A – valorising openness

By contrast, in my second main example, *ElectroEcoBlu*, the separation between design as abstraction and production of material remains much more unstable. However, in this example, the analogies and borrowings from programming and software development are much stronger and more vivid.

ElectroEcoBlu was the Glasgow University entry into the 2007 iGEM competition. iGEM is the annual synbio competition held at MIT. It relies on much more recursive design processes since it requires contests to use or design standard biological parts call BioBricks. If the JCVI-1.0 ambition aligns with Intel, BioBricks aims to be something more like a public-domain Lego. The injunction 'they must snap together' represents a governing design imperative here.

The Glasgow team won 1st prize in the Environmental Track at iGEM 2007 for an environmental biosensor. Their biosensor is based on a microbe that generates an electrical signal in the presence of a range of organic pollutants such as toluene, benzene, xylenes and other petrochemical derivatives that contaminate industrial land. *ElectroEcoBlu* responds to environmental risks associated with existing industrial processes by designing a living system that signals danger, and therefore functions to protect.

(http://parts.mit.edu/igem07/index.php/Glasgow/Plan#XylR_and_BTEX_Chemical).

In order to compete in iGEM, *ElectroEcoBlu* had to use existing BioBricks, or they had to design

new BioBricks that could be added to the repository of standard parts. The public domain library of standard biological components, BioBrick stored in the Registry of Standard Biological Parts is central to all iGEM projects. In a sense, it could be said that the competition exists to feed the repository. (It is interesting that the parts registry URL changed in 2008 to from parts.mit.edu to the more network-friendly partsregistry.org). The BioBrick component or part is ‘a natural nucleic acid sequence that encodes a definable biological function ... refined in order to conform to one or more defined technical standards’ (Shetty, Endy, and Knight 2008, 2).

BioBricks and the Registry together encapsulate a whole design-philosophy and set of practices translocated from the software industries, electronic engineering and computer science. The BioBrick begins to formalize and abstract from the specific laboratory bench craft skills essential to much biotechnology. It offers researchers from engineering disciplines access to biological entities in a more familiar form of a datasheet or API. It enrolls biologists in design practice drawn from software engineering called ‘functional composition.’ It also brings the whole process of standardisation, which is very familiar to programmers, into the realm of biotechnology. All teams must design and use of these parts to some extent, and in the process of carrying out their own projects, construct and share new parts through the Registry of Standard Biological Parts. In a sense, participation in iGEM means recursively contributing to the collective process of using, and reusing, BioBricks.

Making things using standard components or by designing standard components that others can use can be seen as a recursive design process. Although the process design abstracts away from non-codified craft skills, it does not do this in the interests of speed. Rather, it does it in the interests of collaboration, cooperation or downstream re-use. It begins to move programming of biology away from a machine-code level to a higher-level programming language. The BioBricks are explicitly conceived as an ‘abstraction hierarchy.’ They import into bioengineering that crucial software

engineering separation between use and implementation, between what code does and how it does it. That is, through their modularity, BioBricks allow 'information hiding,' 'separation of concerns,'

At the moment, the abstraction hierarchy of BioBricks hampers participants in iGEM and synbio quite a lot. For instance, a biologist working on the *ElectroEcoBlu* complained:

SR: And so one of the things I found difficult about this project is that because we had to use the BioBrick format, then that was much slower than what I would normally do in my research.

AM: So, was it hard to get things into the BioBrick format?

SR: Yes, it was a pain. Because you have to, the ends of the BioBricks have restriction sites. If you happen to have one of those restriction sites in your gene of interest, you have to modify the gene and remove that restriction site. In some of the things we did, we were dealing with quite big chunks of DNA, so we did have that issue. So we'd have to do site-directed mutagenesis to remove that restriction site before we use that gene. Which is an utter pain in the ass. You would never, you wouldn't do it normally.

In other words, using BioBricks slows the process of research down, at least at the moment. The cost of collaboration is a reduction in speed. It also means that commercial techniques for cutting and pasting DNA cannot be used, since they clash with the BioBricks standard.

The other significant aspect of the design processes embodied in BioBricks is a certain commitment to others. All BioBricks are in the public domain. While open source principles are not openly invoked, and while open source licences cannot apply directly to BioBricks since they are subject to copyright law, this aspect of synbio's design process is very explicitly oriented towards ongoing participation. If JCVI-1.0 represents a proprietary platform that can only be used by those who own it or those who license it from its owners, BioBricks represent a commitment to allowing others to reuse what has been done before without too much cost.

Of course there are great uncertainties as to how others could reuse what has been before. For

instance, if parts are not validated, that is, if their sequence is not verified, then they can cause problems for other users downstream. In other words, they have bugs. Furthermore, there was, at least in 2007, a mismatch between the design philosophy of refactoring biological components and the practices of reuse them. Many people were not reusing code. As one of the Glasgow teams commented:

DG: The size of the machines that we saw were being built [by other teams] might have been five or six bricks. Now people were getting brownie points for depositing 170 bricks. It was never said “And they used so many bricks.” There were no brownie points given for reuse, which is what you would expect. So basically people were just making their own thing new, and then blanging them into the repository.

So the fact that BioBricks are produced under competitive conditions affects the nature of the BioBricks. Appearing to contribute many bricks, it seems, could count more than using them, no matter how creatively or well. Here the conditions under which many BioBricks are drawn up – an international student competition – cuts across the design process in ways that merit further analysis.

Indeed, preliminary analyses of the Registry of Standard Biological Parts suggest that the abstraction hierarchies themselves contain anomalies – Parts sometimes contain others Parts; some Parts contain Devices, which are higher in the abstraction hierarchy (Peccoud et al. 2008, 4). (This would have implications for any attempt to design synbio integrated development environments.)

Conclusion: the price of programmability

This paper has not really been an introduction to synbio. Rather it is an initial survey of some of the design issues that arise when biology is reconfigured as an engineering problem along the lines of information technology. I have suggested that programmability is a crucial consideration here.

Synbio is often presented as 'DNA programming.' What allows programming to become a metaphor

for biology or biotechnology? Programmability relies on a set of separations: between hardware and software, between matter and form, and thing and idea. I think this is what JCVI-1.0 represents: the attempt to effectively separate platform and application. The metaphor of programming also represents a way to managing hyper-complexity by re-structuring a situation around a set of abstractions, or an abstraction hierarchy. *ElectoEcoBlu* and BioBricks represent this: the introduction of the techniques of encapsulation, information hiding, and re-factoring into biological constructs. So we could say that certain strands of the history of software development, with all ways of reconfiguring programming practices and software cultures, is being fairly explicitly re-played in biology.

Programmability can be understood in various ways. There is a formal mathematical sense to programmability that was first addressed by Alan Turing in his response to Hilbert's *Entscheidungsproblem*. As is well-known to any computer-science undergraduate, Turing's 1936 paper demonstrated that there are some mathematical problems whose solutions cannot be guaranteed in advance. Importantly, Turing's 'halting problem' framed a notion of programmability that continues to be important. That notion can be posed quite simply as a question: given a program and some data, will the program produce an output? Turing mathematically demonstrated a limit on programmability. There could be no general way of knowing that in advance. This is one sense of programmability that is to synthetic biology at some level.

But other less formal senses of programmability relate more directly to what is happening in synbio. The key issue in programming over the last 50 years, even since the very verb 'to program' first became meaningful in the 1950s, has been how to couple what programmers want a program to do to the output of a program in use. In many ways, programmability has been achieved by increasing the gap between what programmers do and what programs do. Hierarchies of abstraction mean that we don't need to think very often about CPU cycles or logic gates. Many of the design and

engineering processes associated with software and programming try to deal with this problem. These range from engineering and quasi-mathematical abstractions embedded in programming languages or design philosophies. Examples here would include functional programming languages such as LISP vs procedural languages such as C or Java. Design processes also deeply affect the organisation of programming work. Object-oriented programming or extreme programming are implicitly two different notions of programmability. So gaps between programmers and code are useful. They allow programmability.

Yet there are also gaps between code and execution that elude control. The gap between code and execution continues to change. Programmability is constantly being contested in various ways. Today, we could say that programmability is in crisis, if not in decline, in several different senses. What have been the big changes in software in the last twenty years? If we listen to the accounts of software development written by industry participants, the big changes are related to personal computers and the Internet. The advent of network programming languages, patterns of collaboration, and a general intensification in the circulation of source code as well as executable code - all of these can be related to PCs and the Internet. They trigger profound changes in institutions, media, cultures, and economies associated with these changes.

What does it mean, however, when the platforms and logics of programming change? The changes I have just discussed do not affect the limits of programmability itself. What is programmed, even in the latest mobile media gadgets, is microelectronic. Programming is heavily entangled with software and code that runs to lesser or greater degrees, on microelectronic platforms. No doubt these platforms differ widely, especially by virtue of their locations and functions in globalized systems of transaction, communication and control. But they share certain generic traits, properties and potentials that stem from their architecture. That architecture comprises logic gates and logical operations. The layers of abstraction, information-hiding, modularisation, protocols, and interfaces

that mark out the edges of our encounters with programming and programs are all pervaded by this architecture and the materials it comprises. What difference would it make to programming if the substrate shifted from microelectronics to biological systems such as bacteria, yeasts or other kinds of cell?

This paper approaches this question from the perspective of synthetic biology. What kinds of abstraction, what kinds of coding, and what kinds of programming are on the horizon here? The fact that Google Inc, Microsoft, as well as any number of software industry events (such as O'Reilly) are discussing and funding synthetic biology suggests that the convergences of programming and bioengineering, and in particular, in terms of design processes. The interest for software studies here might be still quite speculative, but now is a good to start thinking about how lessons of programming and programmability could be brought to bear on synbio.

References

Campbell-Kelly, Martin. 2003. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. Cambridge, MA: MIT Press.

Chan, Leon Y, Sriram Kosuri, and Drew Endy. 2005. Refactoring Bacteriophage T7. *Mol Syst Biol*.

Daniel G. Gibson, Gwynedd A. Benders, Cynthia Andrews-Pfannkoch, Evgeniya A. Denisova,

Holly Baden-Tillson, Jayshree Zaveri, Timothy B. Stockwell, Anushka Brownley, David W.

Thomas, Mikkel A. Algire, Chuck Merryman, Lei Young, Vladimir N. Noskov, John I. Glass, J.

Craig Venter, Clyde A. Hutchison, III, Hamilton O. Smith* 2008. Complete Chemical Synthesis, Assembly, and Cloning of a *Mycoplasma genitalium* Genome. *Science* 319 (5867):1215 - 1220.

Emergence. 2008. *Emergence. A Foundation for Synthetic Biology in Europe* 2008 [cited 12 June 2008]. Available from <http://www.emergence.ethz.ch/>.

Garfinkel, Michele S., Drew Endy, Gerald L. Epstein, and Robert M. Friedman. 2007.

SYNTHETIC GENOMICS Options for Governance. Craig J. Venter Institute.

Peccoud, Jean, Megan F. Blauvelt, Yizhi Cai, Kristal L. Cooper, Oswald Crasta, Emily C. DeLalla, Clive Evans, Otto Folkerts, Blair M. Lyons, Shrinivasrao P. Mane, Rebecca Shelton, Matthew A. Sweede, and Sally A. Waldon. 2008. Targeted Development of Registries of Biological Parts. *PLoS One* 3 (7).

Rinie van Est, Huib de Vriend, Bart Walhout. 2007. Constructing Life. The World of Synthetic Biology. Rathenau Institut.

Shetty, Reshma P., Drew Endy, and Thomas F. Knight. 2008. Engineering BioBrick vectors from BioBrick parts. *Journal of Biological Engineering* 2 (5).

Venter, J. Craig, Hamilton O. Smith, and Clyde A. Hutchinson III. 2007. Synthetic genomes. edited by USPTO. U.S.A.