

Systems Biology: A Paradigm Shift?

Devdatt P. Dubhashi
Department of Computer Science
Chalmers University
SE 412 96 SWEDEN
`dubhashi@cs.chalmers.se`

September 13, 2006

Too soon

John Bothwell [4] examines skeptically the recent claims made about systems biology in terms such as “a revolution” and “paradigm shift”. Indeed in a recent book [17], the noted physiologist Denis Noble even goes off the edge, calling for fundamental shifts in our philosophical and even linguistic processes to incorporate oriental . Bothwell on the other hand, argues that Systems Biology “should neither be called a paradigm shift nor a revolution”, and cautions against jumping onto the bandwagon. Bothwell’s main point is that the idea that function is an emergent property of a system has long been part of the standard tenets of reductionism, and as such, systems biology is just a step in the same programme, albeit an important one. I agree with Bothwell that it may be too soon to describe systems biology as a “revolution” or a “paradigm shift”, but I fear he, in turn, has been too hasty to dismiss at least the potential for both.

Two views of Science

Two historians, Peter Galison and Thomas Kuhn, have explored in depth the process of scientific discovery in the modern age. Galison’s great work, *Image and Logic*, was published in 1997. Kuhn’s *The Structure of Scientific*

Revolutions appeared thirty-five years earlier. Bothwell discusses Kuhn's concept of "paradigm shift" at some length in the introduction, and examines systems biology in that light. He does not discuss Galison at all. Dyson [9] compares the two views of science, Galisonian and Kuhnian:

Their views of the history of science are totally different. Their two books have almost nothing in common. Galison's book contains hundreds of pictures of scientific apparatus; Kuhn's book contains only words. For Galison the process of scientific discovery is driven by new tools, for Kuhn by new concepts. Both pictures are true and neither is complete. The progress of science requires both new concepts and new tools.

Elsewhere he has said

[the Kuhnian model] applies to Physics in the 1920s ... it doesn't really describe biology. Galisonian science which is driven by tools is at least as important. Biology is like that.

I want to argue that systems biology has the potential to form a "revolution" or "paradigm shift" in both views, but the the Galisonian view especially. In particular, I believe concepts and tools from Computing Science can help crucially in realizing this potential. Of course, the "-omics" technologies have seen computers contributing vitally all along – they are indispensable in dealing with large data. Yet, I believe this contribution is but the tip of the iceberg, playing more a "service" function. Computing Science concepts and tools have the potential to make a much deeper and far reaching impact on systems biology.

Coping with Complexity

It is well recognized that the single biggest challenge facing Systems Biology is the tremendous complexity of biological systems. Through "-onomics" technologies, much information is now available about constituent parts of living systems but nobody knows much about how to put them together. In an entertaining and a seemingly facetious article entitle "Can a biologist repair the radio?". Yuri Lazebnik [16] makes a very important point. He argues compellingly that the notations, concepts and tools that biologists use

to study systems today are too primitive and crude to cope with the complexity of biological systems. Biologists could gainfully look at how engineers cope with complex systems, he argues.

Engineering has been concerned since the beginning with techniques to cope with complexity, and recently many efforts have originated around the world to bring engineering approaches to bear on biology. MIT for instance now has a Department of Biological Engineering. Drew Endy (from this Department) [12] identifies two of the challenges limiting the engineering of biology as “(1) an inability to avoid or manage biological complexity, (2) the tedious and unreliable construction and characterization of ... biological systems.” He then proposes that the three key ideas for engineering biology are (1) standardization, (2) decoupling and (3) abstraction.

Wonder of wonders! These are precisely the three principles taught in even the first undergraduate programming courses in Computer Science. One may even say that they characterize the (true) paradigm shift in the programming idiom from the early days of Fortran-style “imperative” programming to the more modern functional and object-oriented idioms. It turns out not to be an accident - in the acknowledgments, Endy thanks (among others) G. Sussman, author of the very influential computer science text [1] for “teaching me about the theory and practice of engineering, especially as it relates to biology”. Good modern introductory texts today are [14] in the functional paradigm and [3] in the object-oriented one.

Today Software Engineering is a major area within Computer Science devoted to principles of engineering complex software systems. A notation that is now becoming standard in both academia and industry for the specification of systems is the Unified Modelling Language (UML) [21]. Recently, such notations are becoming better known in the biological community, for instance the introduction of the Systems Biology markup Language (SBML). See also [19, 22] for how UML could be used to model biological systems and [6] for other CS languages used to describe biological systems.

Another major concern in designing complex software systems is to ensure their dependability and robustness to errors. *Formal Methods* is often the generic name given to the set of concepts and techniques designed to do this. The basic idea is for the requirement specifications of a system to be given in a formal language, often a derivative of some appropriate mathematical logic. Then one constructs a program that automatically checks if the given system (say in Java code) satisfies the specification. If so, this is an iron clad guarantee in the form of a mathematical proof that the system works

dependably. There are basically two approaches to this process of verification: *model checking* and *theorem proving*. In model checking, one conducts an exhaustive exploration of the state space of the system. Of course doing so directly would be utterly infeasible for even very small systems because of the explosion in the size of the state space. Sophisticated mathematical techniques are employed to reduce the size of the state space to be explored, and often one does it one module at a time. In the second approach, an automatic theorem prover is invoked to provide a formal proof of the assertion that the system satisfies the specification. In practice, this is usually only partially automated and is driven by the user's understanding of the system to validate. Both methods have been used with moderate success in industry scale problems. For a recent commentary on the state of the art, see [2, 11].

The more and closer one looks at biological systems, the more one is struck by the parallels in organizational principles between them and computing systems, especially modern software systems – abstraction, modularity, reuse of components and algorithmic information-driven behaviour. In biological systems, these are reflected at many different levels - say the basic architecture of the body skeleton [6][Ch. 4] to the reuse of components in the master genetic toolkit [6, 15]. It stands to reason that modelling approaches, notations and concepts for biological systems should parallel those for computer systems. This basic insight was expressed way back in 1976 by Richard Dawkins in the same year as the book that made him famous, *The Selfish Gene*:

If a computer is doing something clever and life-like, say playing chess, and we ask how it is doing it, we do not want to hear about transistors, we accept them ... we need *software explanations* of behaviour ... just as the lowest level of explanation is not always the most appropriate for the computer, no more is it for an animal. Animals and computers are both so complex that something on the level of software explanation must be appropriate for both of them. [8]

Bridging Two Cultures

To quote Freeman Dyson again,

Most of the recent scientific revolutions have been tool-driven, like

the double-helix revolution in biology and the big-bang revolution in astronomy

I hope to have argued that there is potential for a tool (and concept)-driven revolution in systems biology. But there are formidable challenges to overcome for this to happen. A half century ago, C.P. Snow [20] bemoaned the gap between “The Two Cultures”. Today we [13] bemoan a similar gap between the biological sciences on the one hand, and engineering sciences on the other. I feel the gap is particularly egregious with respect to computing science. Of course, it is true that biologists now find the computer indispensable as a data-cruncher, a database, graphical visualizer etc. But they still see computing science as a “push-button” service provider, and remain almost totally ignorant of its deeper concepts and tools. On the other hand, computing scientists are shaped by their traditional distaste for descriptive biology and balk at its messy complexity. They should instead view biology as providing the best challenge for their wares, remembering the sculptor Théophile Gautier who said: “L’ouvre sort plus belle, d’une forme au travail rebelle vers”, roughly translated as “The work is more beautiful from a material that resists the process”.

References

- [1] H. Abelson and G. Sussman, *Structure and Interpretation of Computer Programs*, (2nd edition) MIT Press 1996.
- [2] J.P. Bowen and M.G. Hinchey, “Ten Commandments of Formal Methods ... Ten Years Later”, *IEEE Computer* 39:1, pp. 40–48, 2006.
- [3] D. Barnes and M. Kolling, *Objects First with Java: A Practical Introduction Using BlueJ*, Prentice Hall 2004.
- [4] John H.E. Bothwell, “The Long Past of Systems Biology”, *New Phytologist*, 170, pp. 6–10, 2006.
- [5] L. Cardelli, “Abstract Machines of Systems Biology”, *Transactions on Computational Systems Biology*. III, LNBI 3737, pp 145-168, Springer 2005.

- [6] S. Carroll, *Endless Forms Most Beautiful: The New Science of Evo Devo*, W.W. Norton and Co. 2005.
- [7] Edmund M. Clarke, Orna Grumberg and Doron A. Peled, *Model Checking*, MIT Press 2000.
- [8] Richard Dawkins, “Hierarchical organization: a candidate principle for ethology”, In P.P.G. Bateson and R.A. Hinde (eds), *Growing Points in Ethology*, Conference sponsored by St. John’s College and King’s College, Cambridge, Cambridge University Press, 1976.
- [9] Freeman Dyson *The Sun, the Genome and the Internet: Tools of Scientific Revolution*, Oxford University Press 2000.
- [10] Freeman Dyson, “Clockwork Science”, review of Peter Galison, *Einstein’s Clocks, Poincar’s Maps: Empires of Time*, *New York Review of Books*, Volume 50, Number 17 November 6, 2003.
- [11] “Building a better bug-trap”, *Economist*, June 19, 2003.
- [12] D. Endy, “Foundations for engineering biology”, *Nature* 438, 449-453 (24 November 2005)
- [13] Evelyn Fox-Keller, *Making Sense of Life Explaining Biological Development with Models, Metaphors, and Machines*, Harvard University Press 2002.
- [14] P. Hudak, *The Haskell School of Expression : Learning Functional Programming through Multimedia*, Cambridge University Press 2000.
- [15] M. W. Kirschner and J. C. Gerhart, *The Plausibility of Life: Resolving Darwin’s Dilemma*, Yale University Press 2005.
- [16] Y. Lazebnik, “Can a biologist fix a radio?—Or, what I learned while studying apoptosis”, *Cancer Cell*. 2002 Sep;2(3):179-82.
- [17] Denis Noble, *The Music of Life*, Oxford University Press 2006.
- [18] P.A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*, The MIT Press, Cambridge, MA 2000.

- [19] M. Roux-Rouquié, N. Caritey, L. Gaubert and C. Rosenthal-Sabroux, “Using the Unified Modelling language (UML) to guide systematic description of biological processes and systems”, *BioSystems* 75, pp 3–14, 2004.
- [20] C.P. Snow, *The Two Cultures*, Cambridge University Press Reissue edition (July 30, 1993).
- [21] P. Stevens with R. Pooley, *Using UML: Software Engineering with Objects and Components*, Addison-Wesley, 2006.
- [22] K. Webb and T. White, “UML as a cell and biochemistry modelling language”, *BioSystems* 80, pp. 283–302, 2005.