

Supporting Life Scientists via End User Programming

Luke Church

*Computer Laboratory
University of Cambridge
luke@church.name*

Jasmin Fisher

*Computational Biology Group
Microsoft Research, Cambridge
jasmin.fisher@microsoft.com*

Katinka Apagyí

*Department of Biochemistry
University of Cambridge
ka312@cam.ac.uk*

Abstract

We discuss the application of techniques and theories from End User Programming to life sciences in general, and Executable Biology in particular. We briefly outline some motivations, current progress and challenges to that the field faces. We conclude by arguing that the application of End User Programming techniques will be mutually beneficial for both life sciences and computing.

1. What is End User Programming (EUP)?

EUP is the study of the special requirements of people who aren't professional software developers to engage in programming. Examples include accountants producing spreadsheets, astrophysicists building computational models of stars, and home users programming heating controllers. The population of end user programmers is very substantial, current estimates are that by 2012 there will be between 13 and 55 million End User Programmers in the US compared to only 3 million professional programmers (Scaffidi et al. 2005). Research in EUP¹ includes studies of how and when people start programming (Blackwell, 2002), what strategies they use (Subrahmaniyan et al, 2008), how to increase the reliability of end user software, and tradeoffs in the design of programming languages (Green, 1989)

EUP is being applied to a wide variety of problems, ranging from security usability (Church, 2008) to configuration of home networking appliances (Blackwell & Hague, 2001). Successful applications of EUP methodologies can substantially improve existing systems, for example making spreadsheets easier to debug (Abraham & Erwig, 2007), or help users in creating new ones by, for example, inferring types used in locating people after hurricane Katrina (Scaffidi, 2006).

Possible applications of EUP to biology might include improving the scripting languages that are commonly used for processing data to support better sharing and reuse of scripts, or more ambitiously, to supply designing tools for non-professional programmers to directly model and analyse biological systems. We shall concentrate on the later possibility; EUP techniques will allow us to make such systems easy to learn and use.

2. Executable Biology: Life as Computation

There has been a recent move to view biological systems as computational artefacts. Perhaps the archetypical case of this is Executable Biology (Fisher & Henziner, 2007), the study of biological processes as computer programs. An example from (Fisher et al. 2005) is the modelling of vulval precursor cell fate in *C. elegans* as a reactive model, this study provided new, lab verified, understanding of temporal patterning in *C. elegans*.

Executable Biology sets itself apart from mathematical modelling by contrasting models that can be directly executed with mathematical models that require the development of simulation algorithms. Formalisms currently in use for modelling include, for example, Boolean networks, Petri nets, process calculi, and interacting state machines. A typical usage of executable biology is that an executable model is constructed and executed. Its results are then compared to laboratory data. If the model and the data agree, then the model may suggest further experiments of interest, if the model and the data do not agree then the model may be updated.

Executable biology is showing considerable promise in supporting the understanding of complex biological processes both through better models and through focusing the use of valuable experimental resources.

¹ There are a number of related communities, which for simplicity we shall include within EUP, these include The Psychology of Programming Interest Group and the Visual Languages community.

3. How Executable Biology can benefit from EUP

As described above, executable biology involves describing a model of a biological process to the computer. This is a form of programming, which is non-trivial for people who are not professional programmers; consequently, executable biology is currently often done by having a biologist work with a computational modeller. To support adoption and the generation of new insights we would prefer executable biology languages and tools to be sufficiently easy to use and learn that biologists could work with them directly. Design and testing of languages for ease of use and learning is one of the key research areas of EUP.

For example, we used parts of the Cognitive Dimensions of Notations framework (Green, 1989), to guide the design of a tool for modelling qualitative networks (Doman, 2008), which had positive results in the subsequent very small user study. We propose to continue to apply these usability techniques to our tools, and to develop specialised versions for assisting others in designing executable biological languages.

EUP can also be of benefit to biology more generally, as witnessed by the large number of ‘professional end user programmer-biologists’ who routinely create scripts for data processing. However, where professional end user programmers work as tool builders (including in biology), they are often not afforded an appropriate status: they are inappropriately recruited, underappreciated in their contribution to the scientific process and denied promotion (Segal, 2008). This will limit the adoption of computational methods and tools.

It is notable that such problems don’t seem to occur in fields where the computational models are recognised as of objects of scientific value in themselves (Segal, personal communication). Supporting the adoption of executable biology would create a biological community for whom the models are of scientific value and therefore offers a vector by which the use of computation within biology generally may gain more appropriate recognition.

Challenges: Biologically Meaningful, Computational Tractable, Psychologically and Culturally Acceptable Abstractions

One of the major challenges in the development of an EUP environment for executable biology is to *identify abstractions* with which we can describe biological systems. By analogy to electronic engineering; systems are designed in general in an abstraction hierarchy, with a model of a transistor scaling to a model of a logic gate, then a functional unit etc. Ideally this allows the understanding and use of the behaviour of a functional block without having to know very much about the way it is designed. We are addressing the question: ‘can we understand biology in the same way, as a hierarchy of abstractions?’ Such a hierarchy of abstractions would then be the key to the development of languages and tools to be used by end-user programmers for the construction and communication of biological models.

This is a challenging research question, with no obvious clean possible answers. ‘Abstractions’ in biology tend to break down if they are studied in too much detail. For example, even the view of entities as ‘single celled organisms’ is brought into question by context sensitivity effects like quorum sensing (Fuqua et al. 1994) where an organism’s behaviour changes dependent on the density of similar organisms in the environment.

As the theory of computation can tell us how expressive an abstraction framework is, techniques from EUP such as Misfit analysis (Blandford & Green, 2001) and the Attention Investment Model (Blackwell, 2002) describe users’ response to abstractions. Work by Bowker (2008) describes the social processes surrounding the scientific use of abstractions. We propose to use this body of knowledge and techniques to ensure that the hierarchy of abstractions we create are *usable* by biologists in their social context, as well as scientifically valid.

Studying this challenging question will ultimately lead us to a better understanding of the answer to the question of ‘What does it mean for us to understand biology computationally?’

4. Benefits for End User Programming

However, this is not a unidirectional process. Solutions to many of the challenges that the biosciences present would be of substantial value to contemporary debates in software engineering. Such benefits include:

- an understanding of what it means to use programs to communicate behaviour – with a biological model, the program is the point and needs to be shared amongst scientists; better supporting this sharing of behaviour would be valuable to a wide range of domains from security (Church, 2008) to art (Turner et al. 2005)

- a better process for understanding how to design abstractions, in complex, partially understood environments – this would be of use for building systems that don't impose themselves as much as modern computer systems do (Smith 1998, Brown 2001)
- new models and visual formalisms for concurrency – biological systems do not follow traditional computational models of concurrency, we will need to develop usable ways for biologists to express such concurrent behaviour, this would be of value to emerging concerns as to how to support parallel programming.

5. Conclusion

We conclude that the study of biological computation, using techniques from End User Programming reveals some deep challenges for computational biology. Using the approach we are beginning to make progress in addressing these challenges, and are confident that doing so will benefit both computational biology and the end user programming community in general.

6. References

- Abraham, R. & Erwig, M. 2007. GoalDebug: A Spreadsheet Debugger for End Users, in Proc. 29th IEEE/ACM Int. Conf. on Software Engineering, pp 251-260
- Blackwell, A.F & Hague, R. 2001. AutoHAN: An Architecture for Programming the Home. In Proc. IEEE Symposia on Human-Centric Computing Languages and Environments, pp. 150-157.
- Blackwell, A.F. 2002. First steps in programming: A rationale for Attention Investment models. Proc. IEEE Symposia on Human-Centric Computing Languages and Environments, pp. 2-10.
- Blandford, A.E. & Green T.R.G. 2001. From tasks to conceptual structures: misfit analysis. In Proc. IHM-HCI2001 Vol 2.
- Bowker, G. 2008. Memory Practices in the Sciences. MIT Press
- Brown, B. A. T. 2001. Unpacking a Timesheet: Formalisation and Representation, Computer Supported Cooperative Work, 10 3-4, pp. 293-315
- Church, L. 2008. End User Security: The democratisation of security usability, presented at the first international workshop on Security and Human Behaviour
- Doman, C. 2008. Boolean Network Models for Systems Biology, submitted in partial requirements for the Cambridge Computer Science Tripos.
- Fisher, J. & Henzinger, T.A. 2007. Executable Cell Biology., Nature Biotechnology 25(11):1239-1249
- Fisher, J., Piterman, N., Hubbard J., Stern, M. & Harel D. 2005. Computational insights into *C. elegans* vulval development. PNAS 102(6):1951-1956
- Fuqua W.C., Winans & S.C. Greenberg, E.P. 1994 Quorum sensing in bacteria: the LuxR-LuxI family of cell density responsive transcriptional regulators. Journal of Bacteriology; 176: 269-275
- Green, T. R. G. 1989. Cognitive dimensions of notations. In A. Sutcliffe and L. Macaulay (Eds.) *People and Computers V*. Cambridge, UK: Cambridge University Press, pp 443-460
- Segal, J. 2008. Scientists and Software Engineers: A Tale of Two Cultures, Proc. Psychology of Programming Interest Group
- Scaffidi, C., Shaw, M. & Myers, B. 2005. Estimating the number of End Users and End User Programmers, pp. 207-214, IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)
- Scaffidi, C. 2006. Challenges, Motivations and Success Factors in the Creation of Hurricane Katinka 'Person Locator' Web Sites, In Proc. of the 18th Annual Workshop on the Psychology of Programming Interest Group
- Smith, B. C. 1998. On the Origins of Objects. MIT Press
- Subrahmanian, N., Beckwith, L., Grigoreanu, V., Burnett, M., Wiedenbeck, S., Narayanan, V., Bucht, K., Durmmond, R. & Fern, X. 2008. Testing vs. Code Inspection vs... What Else? Male and Female End User Debugging Strategies, Proceedings of ACM CHI, pp 617-626
- Turner, G., Weakley, A. Zhang, Y. and Edmonds, E. 2005 Attuning: A Social and Technical Study of Artist-Programmer Collaborations, In Proc. of the 17th Annual Workshop on the Psychology of Programming Interest Group, pp 106-119.