Proceedings of the 5th IEEE International Conference on
Autonomic Computing, Chicago, Illinois, June 2008.

Autonomic Software Development Methodology Based on Darwinian Evolution *

Benjamin E. Beckmann, Laura M. Grabowski, Philip K. McKinley, and Charles Ofria
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
E-mail: {beckma24,grabow38,mckinley,ofria}@cse.msu.edu

Abstract

Natural organisms are remarkably well adapted to their environment. Through the process of evolution those organisms that exhibit beneficial traits have prospered due to natural selection. As software developers we strive to create systems as well adapted to a virtual environment as natural organisms are to their physical environment. Leveraging Darwinian evolution, we propose a software development methodology capable of producing self- software. Employing this methodology we present an example behavioral concept from inception to fruition on physical hardware.*

Introduction. To design robust and resilient computational systems, one can take inspiration from nature. Living organisms exhibit an amazing ability to adapt to a changing environment, both in the short term (phenotypic plasticity) and in the longer term (Darwinian evolution). For this reason, many researchers have recently turned to biologically-inspired methods for constructing highly adaptive, robust systems [2, 3]. However, purely biomimetic approaches seek to imitate the *results* of evolution, but do not account for the process of natural selection that optimizes a system to fit its environment.

A fundamentally different approach to building robust computational systems is *digital evolution* [6]. In this method, a population of computer programs exists in a user-defined computational environment and is subject to mutations and natural selection. These "digital organisms" are provided with limited resources whose use must be carefully balanced if the organisms are to survive. Over generations, these organisms will evolve to optimize resource usage and thrive if they are able. The most widely used digital evolution platform is AVIDA [6]. Over the past several years, AVIDA has been used to conduct pioneering research in the evolution of biocomplexity, with an emphasis on understanding the evolutionary design process in nature [4]. However, digital evolution also provides a means to harness the power of evolution and apply it to a wide variety of problems in science and engineering [5], often revealing unintuitive solutions.

In this work we investigate the application of digital evolution to the design of software exhibiting self-* properties. Specifically, we propose a software development methodology that uses digital evolution to synthesize robotic controllers, based on high-level goals provided by the developer. In particular, we evolve behaviors in AVIDA and automatically translate them into C code, which is compiled and loaded onto iRobot™ Create mobile robots. Experiments demonstrate that the robots exhibit the desired problem solving behaviors.

Development Model. As described by Kephart and Chess [3], autonomic computing focuses on creating computer systems that manage themselves according to an administrator's goals. Our approach uses digital evolution to help design computational behavior needed in such systems. However, rather than addressing only runtime behavior, we also apply autonomic computing concepts to help guide the software *development* process. Specifically, we evolve behaviors in virtual entities whose capabilities closely match those of a target platform, potentially producing well adapted solutions that can be directly mapped into a target environment.

To create such a system, physical, biological, and evolutionary principles must be packaged into a single development methodology. The system must also enable verification and validation of the resulting software, with a feedback loop to the developmental process. We propose a three-stage development model, depicted in Figure 1, to meet this need. The stages are: cultivation, evaluation, and deployment. During cultivation, digital organisms with capabilities similar to a target platform are evolved in the AVIDA digital evolution system. Effectively, AVIDA provides a "digital Petri dish" for evolving new computational behaviors based on high-level human guidance. Thus the solutions are produced *autonomically* – once initiated, the population evolves without human interaction. By abstracting development in this manner, more time can be focused on the desired behavior of the system rather than low-level implementation. During cultivation, the digital organisms' genomes – programs coded in a specially designed language – evolve to fit their environment. During the evaluation stage a genome is translated and evaluated in simulation and hardware. Any shortcomings observed in either of these stages are fed

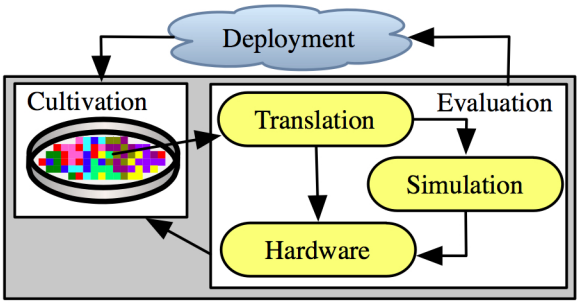


Figure 1. Development process

* This work was supported in part by the U.S. Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744, National Science Foundation grants ITR-0313142, CCF 0523449, and CCF 0750787 Cambridge Templeton Consortium Emerging Intelligence grant, and a Quality Fund Concept grant from Michigan State University.

back into the cultivation stage to refine the evolved model. Additionally, a developer can integrate existing tools such as model checkers into the cultivation stage, in order to verify that the software adheres to specified properties. That aspect of our group's research is discussed in an accompanying paper [1].

Treatment. The goal of this case study is to evolve control software for robots, of varying capabilities, that allows them to search for and move to a light source. To encourage the evolution of object detection and location behaviors we apply selection pressures that reward organisms for efficiency in reaching the target, a simulated light source. This treatment consists of 20 replicate runs, each containing 500 organisms divided into individual worlds, and lasting for 250,000 time steps (8 hours of compute time). Each individual is allowed to live for 1000 time steps or until it is replaced by an organism who has reached the target.

Experimental Results. Figure 2 shows a three dimensional representation of the gradient used in the experiments. The black lines represent paths that organisms followed relative to the gradient. Depending on the organism's initial facing the beginning of a path may display a few movements that allow the organism to orient itself to face the high point of the gradient, and then move in that direction. Since an organism is rewarded for energy conservation, the more direct a path the better.

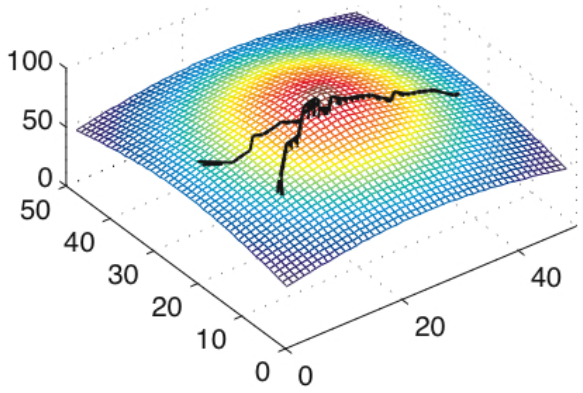
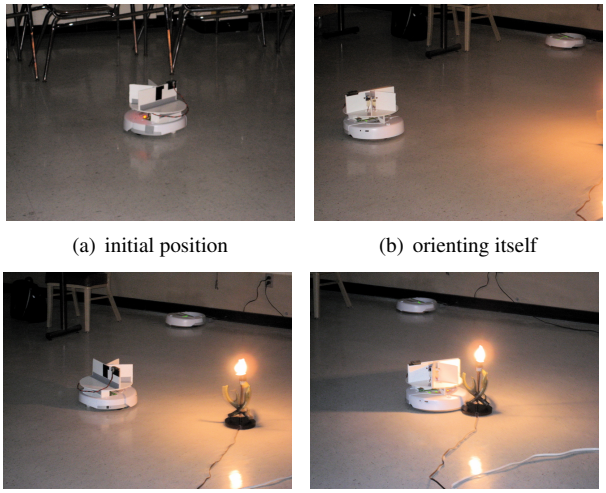


Figure 2. Paths of evolved genomes, superimposed on the illumination gradient.

We translated several evolved genomes into C code, compiled them, and loaded them onto an iRobot™ Create robot. We then placed the robot in a room with a light source and filmed the resulting behavior. Four sample clips from a movie are shown in Figure 3. The images show a sequence of clips from the robot's initial position, orientation, its progress toward the light source, and then finally touching the light.

Conclusions and Future Directions. We have shown that the proposed methodology can be used to successfully model and cultivate solutions that, once translated, can execute directly on real-world hardware. We have also shown that a developer can direct the process by providing high-level requirements for the evolved solution. We have also observed that employing evolution as the "designer" of software has limitations. The complexity of the solution is limited by an organism's virtual CPU and the instruction set. Also, well crafted high-level goals and building blocks that allow evolution to produce the desired results are required.



(a) initial position (b) orienting itself

(c) moving toward higher intensity (d) light source found

Figure 3. Movie clips showing translated code executing on an iRobot Create system.

We intend to extend this work into the production of evolved behaviors for swarm robots. We are beginning to work on problems such as division of labor, communication optimization, and predator avoidance. We also plan to address evolution of the morphology of the target platform along with its controller. By adding this dimension we hope to produce cohesive robot-controller combinations that are competitive with human-designed systems.

Traditional software development methods are being challenged by the ever-increasing complexity of today's software systems. As the cost of developing these systems grows, alternatives that construct adequate solutions with less manpower are appealing. Evolution provides us with a method capable of designing solutions for increasingly complex environments. Enabling software development methods to harness this power, through digital evolution, may provide a means to produce economical software solutions that exhibit the robustness, flexibility, and adaptability that abound in solutions produced by natural evolution.

References

[1] H. J. Goldsby, B. H. C. Cheng, P. K. McKinley, D. B. Knoester, and C. A. Ofria. Digital evolution of behavioral models for autonomic systems. In *Proceedings of the 5th International Conference on Autonomic Computing*, Chicago, Il, June 2008.

[2] S. Johnson. *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. Scribner, 2002.

[3] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[4] R. E. Lenski, C. Ofria, R. T. Pennock, and C. Adami. The evolutionary origin of complex features. *Nature*, 423:139–144, 2003.

[5] P. McKinley, B. Cheng, C. Ofria, D. Knoester, B. Beckmann, and H. Goldsby. Harnessing digital evolution. *Computer*, 41(1):54–63, January 2008.

[6] C. Ofria and C. O. Wilke. Avida: A software platform for research in computational evolutionary biology. *Artificial Life*, 10:191–229, March 2004.