A Biologically Inspired Programming Model for Self-Healing Systems

Selvin George
Department of Computer Science
University of Virginia
Charlottesville, VA
selvin@virginia.edu

David Evans
Department of Computer Science
University of Virginia
Charlottesville, VA
evans@virginia.edu

Lance Davidson
Department of Biology
University of Virginia
Charlottesville, VA
lance_davidson@virginia.edu

ABSTRACT

There is an increasing need for software systems to be able to adapt to changing conditions of resource variability, component malfunction and malicious intrusion. Such self-healing systems can prove extremely useful in situations where continuous service is critical or manual repair is not feasible. Human efforts to engineer self-healing systems have had limited success, but nature has developed extraordinary mechanisms for robustness and self-healing over billions of years. Nature's programs are encoded in DNA and exhibit remarkable density and expressiveness. We argue that the software engineering community can learn a great deal about building systems from the broader concepts surrounding biological cell programs and the strategies they use to robustly accomplish complex tasks such as development, healing and regeneration. We present a cell-based programming model inspired from biology and speculate on biologically inspired strategies for producing robust, scalable and self-healing software systems.

Categories and Subject Descriptors

D.1.0 [**Programming Techniques**]: General; D.2.4 [**Software/Program Verification**] – *reliability*; F.1.1 [**Models of Computation**].

General Terms

Design, Reliability, Experimentation, Security, Languages

Keywords

Biological programming; self-healing systems; amorphous computing.

1. INTRODUCTION

Biology is replete with examples of systems with remarkable robustness and self-healing properties. These include morphogenesis, wound healing and regeneration:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WOSS '02, Nov 18-19, 2002, Charleston, SC, USA.

Copyright 2002 ACM 1-58113-609-9/02/0011 ...$5.00

Morphogenesis. A single cell develops into a full organism following a program encoded in its DNA that evolved over billions of years. Cells perform various actions like division, deformation and growth based on gene actions. The actions of the genes are dictated by the presence of chemical substances. Gene actions coupled with physical forces acting on a cell from its neighboring cells and external environment lead to a developed organism. Even in simple organisms, development is robust to many kinds of local failures and adapts to a wide range of environments. For example, when a cell dies, the neighboring cells sense changes in the environment and adapt their own development to correct the problem [6].

Wound Healing. Almost all complex organisms have some sort of mechanism for healing simple wounds. In humans, when a minor injury happens, an inflammatory response occurs and the cells below the dermis (the deepest skin layer) begin to increase collagen (connective tissue) production. Later, the epithelial tissue (the outer skin layer) is regenerated. The interesting point here is the apparent level of awareness of the cells. Also, cells around the injury are able to adapt to a different function based on the new circumstances [2].

Regeneration. Many organisms can regenerate new heads, limbs, internal organs or other body parts if the originals are lost or damaged. Organisms take two approaches to replacing a lost body part. Some, such as flatworms and the polyp *Hydra*, retain populations of stem cells throughout their lives, which are mobilized when needed. These stem cells retain the ability to regrow many of the body's tissues. Other organisms, including newts, segmented worms and zebrafish, convert differentiated adult cells that have stopped dividing and form part of the skin, muscle or another tissue back into stem cells. When a newt's leg, tail or eye is amputated or damaged, cells near the stump begin an extraordinary change. They revert from specialized skin, muscle and nerve cells into blank progenitor cells. These progenitors multiply quickly to about 80,000 cells and then grow into specialized cells to regenerate the missing part [5].

We observe that nature's approach to programming has the following properties:

1. **Environmental Awareness.** Though the cells may have limited communication capabilities, they act differently in response to sensed properties of the surrounding environment. This enables cells to react to changes in nearby cells, as well as the surrounding environment.

2. **Adaptation.** Many cells have a great amount of adaptability. In many organisms, at the beginning of morphogenesis, if one of the initial cells obtained by the first division of the germ cell dies then the surviving cell is often able to complete the development of the organism. This indicates that enough information is preserved to be able to "backtrack" to a previous state of development. This is due to the fact that all cells run the same cell-program and can hence respond to aberrant behavior from neighbors
3. **Redundancy.** A non-redundant organism would have every cell assigned a fixed role in the development process. Any failure during the development process would produce a defective organism. Typical organisms have many cells devoted to the same function throughout development, so that failures of individual cells are inconsequential. Biological systems also exhibit redundancy of function, where several distinct mechanisms evolve for the same purpose in a single organism.
4. **Decentralization.** There is no global coordination and limited communication for most of the development process. Cells sense properties of their environment, and are most affected by nearby cells. Cells can induce neighboring cells to do a particular action, but there is no centralized control and limited long-distance communication.

2. CELL-BASED PROGRAMMING

Inspired by biological systems, we propose a cell-based programming model that can be used for software systems operation and healing. Our model is similar to the cellular automata that have been studied extensively since von Neumann's early work [4], but differs in that it is more closely related to biological processes. In particular, we support a notion of cell division, a communication model based on chemical diffusion, and a rudimentary model of the physical forces involved. By developing a programming model more like nature's, we believe we will produce more robust programs with natural self-healing properties.

A related approach is amorphous computing, which considers approaches for programming a medium of randomly distributed computing particles. The Growing Point Language [1] and Origami Shape Language [3] both illustrate mechanisms for global self-organization using simple local communication of the agents. Self-healing properties are also being studied using GPL. As with our work, the challenge is to produce programs that generate predictable behavior with a locally unpredictable and non-traditional programming model. Because the underlying execution environment is inherently redundant and decentralized, robustness is practically inevitable if programs are constructed in the right way.

We represent a cell program as an automaton containing discrete states and transitions between these states. Every cell comprising the program is in one of these states. The input to each cell state is the sensed properties of the local environment and the output is a transition to another state, or a division into two (possibly different) states. States are represented by circles and state transitions by directional arrows. Dots represent cell divisions.

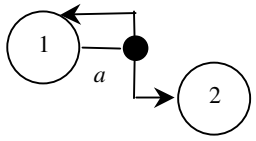
Our cell programming model incorporates:

1. **Cell Division.** A cell can divide into two daughter cells that may be dissimilar in orientation and chemical composition but have the same program (DNA). A cell has an axis called the apical-basal axis. Divisions can be either perpendicular to this cell axis or along the plane containing the axis. The difference in chemical composition and also the different chemicals on their cell walls causes the two daughter cells to behave differently from that point onwards. Cell division is modeled by using a transition from one state to two states.
2. **Cell Actions.** Cells can produce proteins and signaling chemicals depending on what genes are active. Chemicals produced this way affect the environment and neighboring cells through chemical diffusion.
3. **Gene Actions.** Genes can activate or deactivate depending on the presence or absence of a particular protein or a certain degree of chemical concentration. Activation or deactivation of a gene results in cell actions like production of chemicals.

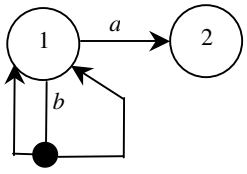
The varying degrees of concentration produced by earlier cell actions (both by the cell and its neighbors) cause gene actions and gene actions cause cell actions; this results in a powerful programming paradigm. Cell actions such as production of chemicals are modeled using messages. Gene actions are modeled using cell state transitions; these are a result of received messages.

A cell program begins with cells in an initial configuration, and all the cells follow transition rules like a finite state machine. Between steps, an environment simulator determines changes in external stimuli. The changes to the environment can be due to operations of the software system, expected input conditions or failure conditions. Since cells can sense their local environment it is possible for them to be able to perform failure recovery (healing) or re-composition of appropriate components (regeneration). Our simulator also provides opportunities to conduct experiments involving random and catastrophic failures.

Two simple examples of cell programs are shown in Figure 1. Automaton A produces a line of cells as long as the input condition *a* exists. The condition *a* may represent the presence of food for growth. Automaton B produces cells to combat intruders as long as it detects unfavorable conditions. This approach creates excess cells so that some may survive the malicious action.



A. Creating a line of Cells



B. Robustness through replication

a – Condition favorable to continued cell production

Numbered Circle – State
Arrow – Transition
Dot – Cell division

a – Favorable conditions

b – Unfavorable conditions

Figure 1. Example Cell Programs

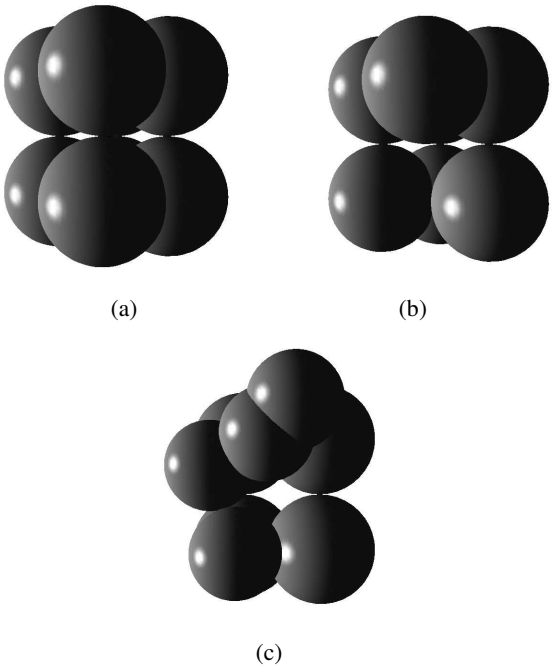
3. SIMULATING CELL PROGRAMS

Using a simulator, we have conducted simulations of different cell programs. The simulator simulates a cell program on a simulation configuration. The simulation configuration is used to introduce new cells, chemical concentrations or failures. Our simulator is available at http://swarm.cs.virginia.edu/cellsim.

A sample program for creation of a self-healing blastula is shown below. A blastula is a spherical structure that is the first stage of development of many large organisms. A sufficient number of cells are needed before organism development proceeds to the next stage.

```
state s1 {
    emits (sig, 0.1)
    transitions
      (0 <= sig <= 0.375) -> (s2, s2) axis;
                          -> (s1);
}
state s2 {
    emits (sig, 0.1)
    transitions
      (0 <= sig <= 0.375) -> (s3, s3) normal-X;
                          -> (s2);
}
state s3 {
    emits (sig, 0.1)
    transitions
      (0 <= sig <= 0.375) -> (s1, s1) normal-Y;
                          -> (s3);
}
```

In the above cell automaton there are three cell-states – *s1*, *s2* and *s3*. They are similar in that they emit the same signaling chemical *sig* and divide into two cells each if they sense that the concentration of *sig* is less than 0.375. The cell remains in its current state if the concentration of *sig* is above 0.375.



(a) (b)

(c)

Figure 2. Simulated Blastula Program. (a) Blastula in 8-cell stage – starting from one cell; (b) damaged blastula – after killing one cell (c) after the blastula regenerates.

This self-healing blastula has the property that if a few cells are killed, it will automatically heal itself by producing the required number of cells. Figure 2 shows a simulation of the blastula program for four steps, after which one of the cells was killed to observe the self-healing behavior. The surviving cells regenerate additional cells to continue the process. The principle behind this type of healing is that the once a cell was killed, it stopped producing the particular chemical that was being sensed by its neighbors. Note that nothing in the cell program explicitly deals with healing and regeneration. The neighbors of a failed cell just follow a different path in the cell-program due to the changed environmental conditions.

4. TOWARDS SELF-HEALING SYSTEMS

Although our initial experiments have focused on mimicking simple biological processes and generating basic geometric structures, our long-term goal is to develop techniques that can be used to produce robust, self-healing systems designed to perform a complex task. Developing complex programs using state diagrams, however, is infeasible. A high-level programming abstraction for cell-based programs is needed in which a programmer can describe desired processes at a high-level and a cell-program compiler will produce the steps of the automaton. An important design issue is which operations or abstractions should be part of the language and which can be composed of the elementary operations and hence can be kept outside the language. We are currently working on programming abstractions based on the biological cell model. If successful, programs described in this way will have intrinsic robustness, scaling and self-healing properties. We hope that our experiments with biological programs will provide insights into how to build more robust computer systems.

5. ACKNOWLEDGMENTS

This work was funded in part by grants from the National Science Foundation (CCR-0092945 and EIA-0205327) and NASA Langley Research Center.

6. REFERENCES

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman and R. Weiss, *Amorphous Computing,* Communications of the ACM, Volume 43, Number 5, p. 74-83. May 2000.

[2] Mary Y. Mazzotta. *Nutrition and wound healing*. Journal of the American Podiatric Medical Association. Volume 84, Number 9, p. 456–62. September 1994.

[3] Radhika Nagpal, *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics,* PhD Thesis, MIT Department of Electrical Engineering and Computer Science, June 2001.

[4] John von Neumann, *Theory of Self-Reproducing Automata.* University of Illinois Press, 1966 (Originally published in 1953).

[5] Helen Pearson, *The regeneration gap*, Nature Science Update. 22 November 2001.

[6] Lewis Wolpert, Rosa Beddington, Peter Lawrence, Thomas M. Jessell, *Principles of Development*, Oxford University Press. 2002.