Apoptotic Computing: Programmed Death by Default for Computer-Based Systems

Roy Sterritt, *University of Ulster*

Inspired by the cellular self-destruct mechanisms in biological apoptosis, apoptotic computing offers a promising means to develop self-managing computer-based systems.

At the 2009 International Joint Conference on Artificial Intelligence, researchers warned that the nightmare scenarios depicted in sci-fi films such as *2001: A Space Odyssey*, the *Terminator* and *Matrix* series, *Minority Report*, and *I, Robot* could come true. "Scientists fear a revolt by killer robots" proclaimed the UK's *Sunday Times*,[1] which highlighted alarming findings at the conference that mankind might lose control of computer-based systems that carry out a growing share of society's workload, from chatting on the phone to waging war, and have already reached a level of indestructibility comparable with the cockroach. For instance, unmanned predator drones, which can seek out and kill human targets, have already moved out of the movie theatre and into the theatre of war in Afghanistan and Iraq. While presently controlled by human operators, these drones are moving toward more autonomous control. Similar devices may also soon appear above city streets to carry out domestic surveillance. Samsung, the South Korean electronics giant, has developed autonomous sentry robots with "shoot to kill" capability to serve as armed border guards.[1]

To provide for this future, the Apoptotic Computing project has been working since 2002 toward the long-term goal of *programmed death by default* for computer-based systems.[2-6] Motivated by the apoptosis mechanisms in multicellular organisms, apoptotic computing can be considered a subarea of bio-inspired computing, natural computing, or autonomic systems. Two example applications are autonomic agent-based environments and swarm space exploration systems.

BIOLOGICAL APOPTOSIS

Developing a self-managing computer system is the vision of autonomic computing.[7-9] As the "Autonomic System Properties" sidebar explains, an autonomic computing system is analogous to the biological nervous system, which automatically maintains homeostasis (metabolic equilibrium) and controls responsiveness to external stimuli. For example, most of the time you are not consciously aware of your breathing rate or how fast your heart is beating, while touching a sharp knife with your finger results in a reflex reaction to move the finger out of danger.[10]

If you cut yourself and start bleeding, you treat the wound and carry on without thinking about it, although pain receptors will induce self-protection and self-configuration to use the other hand. Yet, often the cut will have caused skin cells to be displaced down into muscle tissue.[11] If the cells survive and divide, they have the potential to grow into a tumor. The body's solution to this situation is cell self-destruction (with mounting evidence

AUTONOMIC SYSTEM PROPERTIES

The general properties of an autonomic, or self-managing, computing system consist of four objectives that represent broad system requirements, and four attributes that identify basic implementation mechanisms.[1,2]

An autonomic system has the following objectives:

- *Self-configuration*. The system must be able to readjust itself automatically, either to support a change in circumstances or to assist in meeting other system objectives.
- *Self-healing*. In reactive mode, the system must effectively recover when a fault occurs, identify the fault, and, when possible, repair it. In proactive mode, the system monitors vital signs to predict and avoid health problems, or to prevent their reaching undesirable levels.
- *Self-optimization*. The system can measure its current performance against the known optimum and has defined policies for attempting improvements. It can also react to the user's policy changes within the system.
- *Self-protection*. The system must defend itself from accidental or malicious external attacks, which requires an awareness of potential threats and the means to manage them.

To achieve these self-managing objectives, a system must be

- *self-aware*—aware of its internal state;
- *self-situated*—aware of current external operating conditions and context;
- *self-monitoring*—able to detect changing circumstances; and
- *self-adjusting*—able to adapt accordingly.

Thus, to be autonomic a system must be aware of its available resources and components, their ideal performance characteristics, and current status. It must also be aware of interconnection with other systems, as well as rules and policies for adjusting as required. Operating in a heterogeneous environment requires relying on open standards to communicate with other systems.

These mechanisms do not exist independently. For example, to successfully survive an attack, the system must exhibit self-healing abilities, with a mixture of self-configuration and self-optimization. This not only ensures the system's dependability and continued operation but also increases self-protection from similar future attacks. Self-managing mechanisms must also ensure minimal disruption to users.

References

1. R. Sterritt, "Towards Autonomic Computing: Effective Event Management," *Proc. 27th Ann. IEEE/NASA Software Eng. Workshop* (SEW 02), IEEE CS Press, 2002, pp. 40-47.
2. R. Sterritt and D. Bustard, "Autonomic Computing—A Means of Achieving Dependability?" *Proc. 10th IEEE Int'l Conf. and Workshop Eng. of Computer-Based Systems* (ECBS 03), IEEE CS Press, 2003, pp. 247-251.

that some forms of cancer are the result of cells not dying fast enough, rather than multiplying out of control, as previously thought).

Biologists believe that cells are programmed to commit suicide through a controlled process known as *apoptosis*.[12] The term is derived from the Greek word for "to fall off," in reference to dead leaves falling from trees in autumn; likewise, cells "fall off" the living organism and die. As Figure 1a shows, a cell's constant receipt of "stay alive" signals turns off the self-destruct sequence.[3] When these signals cease, the cell starts to shrink, internal structures decompose, and all internal proteins degrade; thereafter, the cell breaks into small, membrane-wrapped fragments to be engulfed by phagocytic cells for recycling. Figure 1b contrasts apoptosis, also known as "death by default,"[11] with *necrosis*, the unprogrammed death of a cell due to injury—inflammation and the accumulation of toxic substances.[13]

Recent research indicates that cells receive orders to kill themselves when they divide.[12] The reason appears to be self-protection. An organism relies on cell division for maintenance and growth, but the process is also dangerous: if just one of the billions of cells in a human body locks into division, the result is a tumor. The suicide and reprieve controls can be likened to the dual keys of a nuclear missile: the suicide signal (first key) turns on cell growth but at the same time activates a sequence that leads to self-destruction, while the reprieve signal (second key) overrides the self-destruct sequence.[14]

AUTONOMIC AGENTS

Autonomic computing depends on many disciplines for its success; not least of these is research in agent technologies. There are no assumptions that an autonomic architecture must use agents, but agent properties—adaptability, autonomy, cooperation, and so on—complement the paradigm's objectives. In addition, there are arguments for designing complex systems with multiple agents,[15] providing such systems with inbuilt redundancy and greater robustness,[16] and for retrofitting legacy systems with autonomic capabilities that may benefit from an agent-based approach.[17]

Figure 2 shows a basic *autonomic element* (AE), which consists of a *managed component* (MC) and an *autonomic manager* (AM).[18] The AM can be a stationary agent—for instance, a self-managing cell[19] that contains functionality for measurement and event correlation and provides support for policy-based control. AMs communicate via means such as self-* event messages.

Mobile agents can also play a role in autonomic systems. Their ability to reduce network load, overcome network latency, encapsulate protocols, execute asynchronously and autonomously, adapt dynamically, reflect natural heterogeneity, and maintain robustness and fault tolerance can make it easier for AMs within different systems to cooperate.

APOPTOSIS IN AGENT-BASED AUTONOMIC ENVIRONMENTS

Michael S. Greenberg and colleagues first proposed agent destruction to facilitate security in mobile-agent

systems.[20] They described an event in which network operators decommissioned a computer named omega.univ.edu and moved its work to other machines. A few years later, the operators assigned a new computer the old name and, to everyone's surprise, e-mail arrived, much of it three years old; the mail had survived "pending" on Internet relays waiting for omega.univ.edu to come back up. Greenberg's team considered a similar scenario in which mobile agents—not rogue agents but ones carrying proper authenticated credentials—carried out work that was out of context rather than the result of abnormal procedures or system failure. In this circumstance, the mobile agents could cause substantial damage—for example, deliver an archaic upgrade to part of the network operating system, bringing down the entire network.

Misuse involving mobile agents can occur in several forms. Agents can accidentally or unintentionally misuse hosts due to, say, race conditions or unexpected emergent behavior in those agents. In addition, external bodies acting upon agents, either deliberately or accidentally, can lead to their misuse by hosts or other agents—for example, due to damage, breaches of privacy, harassment, social engineering, event-triggered attacks, or compound attacks.

Encryption can prevent situations in which portions of an agent's binary image—monetary certificates, keys, information, and so on—could be copied when visiting a host. However, agent execution requires decryption, which provides a window of vulnerability.[20] This situation is analogous to the body's vulnerability during cell division.[3]

Figure 3 shows a high-level view of a simple autonomic environment with three AEs (a typical system has hundreds, thousands, or even millions of AEs). Each AE is an abstract view of Figure 1, and in this case the MCs represent self-managing computer systems. These AEs can have many other lower-level AEs—for example, an autonomic manager for the disk drive—while at the same time residing within the scope of a higher-level AM such as a system-wide local area network domain's AE.

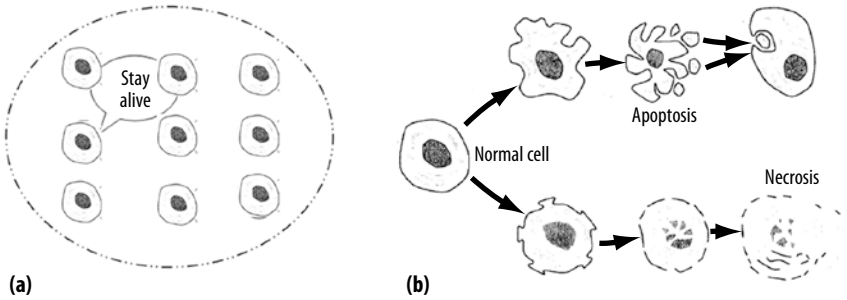

Figure 1. Biological apoptosis. (a) A cell's constant receipt of "stay alive" signals turns off its programmed self-destruct sequence. (b) Apoptosis versus necrosis due to injury.

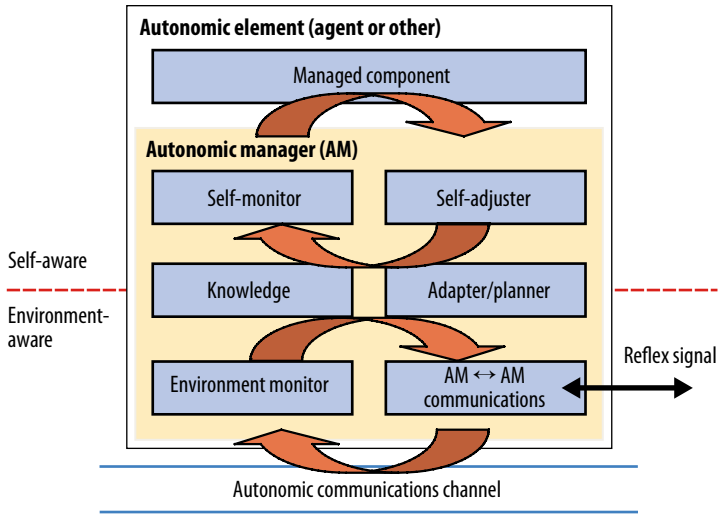

Figure 2. An autonomic element consists of a managed component and an autonomic manager, which can be a stationary agent. The AM ↔ AM communications module includes heartbeat monitoring and pulse monitoring. AMs communicate through an autonomic channel via such means as self-* event messages.

Within each AM, heartbeat monitors (HBMs) send "I am alive" signals to ensure the continued operation of vital processes in the MC and to immediately indicate if any fail. The AM has a control loop that continually monitors and adjusts, if necessary, metrics within the MC, yet vital processes in the MC can also be safeguarded by an HBM that emits a heartbeat signal as opposed to its being polled by the AM, avoiding lost time (time to next poll) by the AM to notice a failure (note in Figure 3 that the left-hand AE has an HBM between the AM and a process on the MC).

Because each AM is aware of its MC's health via the continuous control loop, it can share this information by sending a pulse signal ("I am un/healthy") to another AM—in Figure 3, for example, from the left-hand AE to the middle AE. This not only allows self-managing options if the machines are, say, sharing workload as a cluster but protects the AM itself as the pulse signal also acts as

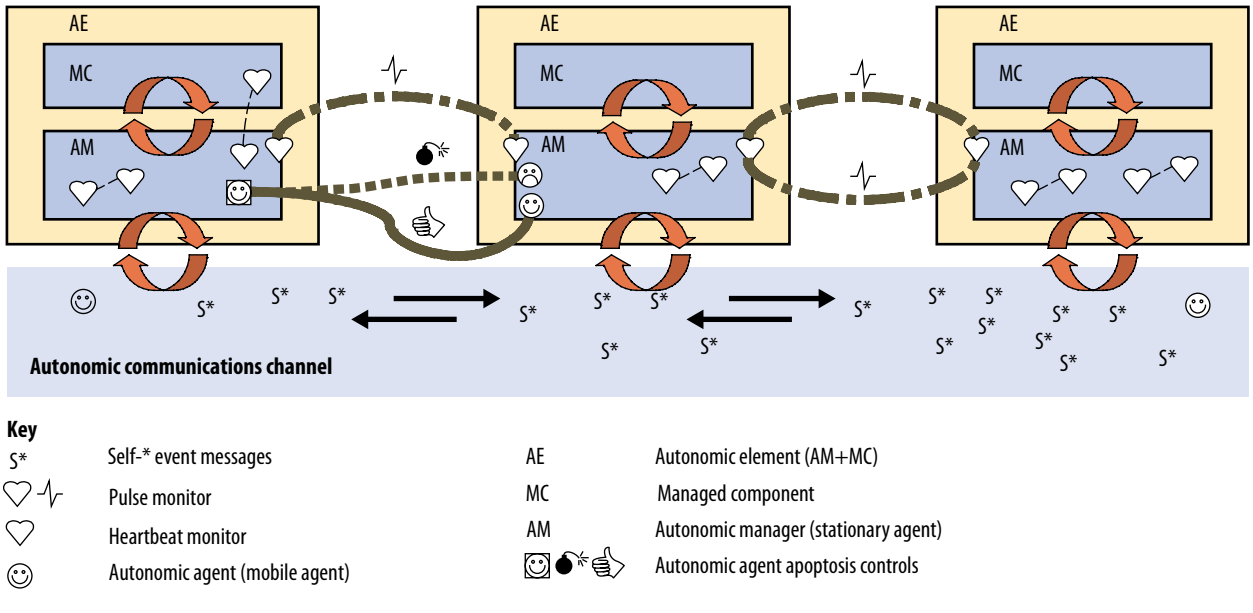

Figure 3. Simple autonomic environment consisting of AEs with autonomic agents (stationary and mobile), heartbeat monitors ("I am alive"), pulse monitors ("I am un/healthy"), and apoptosis controls ("stay alive/self-destruct").

an HBM signal from one AM to another. Thus, if an AE's vital process fails, the neighboring AM will immediately become aware of it and, for example, try to restart the failed AE or initiate a failover to another AM. This pulse signal can also act as a reflex signal between AMs warning of an immediate incident—a more direct solution than the AM's processing numerous event messages to eventually determine an urgent situation.

Because AMs also monitor the external environment (the second control loop), they have a view of their local environment's health. They can encode such information into the pulse signal along with self-health data (just as our hearts have a double beat). The double-pulse signals between the right-hand and center AEs in Figure 3 represent this situation.

AMs can dispatch mobile agents to work on their behalf—for example, to update a set of policies. To help provide self-protection in these situations, AMs can send apoptosis signals ("stay alive/self-destruct") to such agents by either authorizing continued operation or by withdrawing such authorization—for example, if the policies become out of date. Figure 3 depicts both scenarios.

We refer to the absence of a "stay alive" signal resulting in agent self-destruction as *strong* apoptotic computing, or programmed death by default, while *weak* apoptotic computing involves an explicit self-destruct signal—similar in principle to the garbage collection method first used by Lisp and by many languages since or the destructor method in object orientation. The differences in these approaches are subtle but important. Only a built-in default death can guarantee true system safety. For example, you would never rely on a self-destruct signal getting through to an agent containing system password updates in a hostile environment. Likewise, a robot with adaptive capabilities could learn to ignore such a signal. That said, clearly not all circumstances require a death-by-default mechanism. However, we believe that many researchers using *programmed death* under the apoptosis descriptor should be using programmed death by default.

There is a concern that denial-of-service attacks could prevent "stay alive" signals from reaching their target and thereby induce unintentional agent self-destruction. DoS attacks could likewise interrupt terminate signals, resulting in potentially dangerous scenarios. DoS-immune architectures are thus a critical part of next-generation self-managing systems.

SWARM SPACE EXPLORATION SYSTEMS

Space exploration missions by necessity have become increasingly autonomous and adaptable. To develop more self-sustainable exploration systems, NASA is investigating the use of biologically inspired swarm technologies.[3] As Figure 4 shows, the idea is that swarms of small spacecraft offer greater redundancy (and, consequently, greater protection of assets), lower costs and risks, and the ability to explore more remote regions of space than a single large craft.

The Autonomous NanoTechnology Swarm mission (http://ants.gsfc.nasa.gov), a collaboration between NASA's Goddard Space Flight Center and its Langley Research Center, exploits swarm technologies and AI techniques to develop revolutionary architectures for both space-

craft and surface-based rovers. ANTS consists of several submissions:

- The Saturn Autonomous Ring Array consists of a swarm of 1,000 pico-class spacecraft, organized as 10 subswarms with specialized instruments, to perform in situ exploration of Saturn's rings to better understand their constitution and how they were formed. SARA will require self-configuring structures for nuclear propulsion and control as well as autonomous operation for both maneuvering around Saturn's rings and collision avoidance.
- The Prospecting Asteroid Mission (PAM) also involves 1,000 pico-class spacecraft but with the aim of exploring the asteroid belt and collecting data on particular asteroids of interest for potential future mining operations.
- The Lander Amorphous Rover Antenna (LARA) will implement new NASA-developed technologies in the field of miniaturized robotics, which could form the basis of remote lunar landers launched from remote sites, as well as offering innovative techniques to allow rovers to move in an amoeboid fashion over the moon's uneven terrain.

The ANTS architecture emulates the successful division of labor exhibited by low-level social-insect colonies. In such colonies, with sufficiently efficient social interaction and coordination, a group of specialists usually outperforms a group of generalists. To accomplish their specific mission goals, ANTS systems likewise rely on large numbers of small, autonomous, reconfigurable, and redundant worker craft that act as independent or collective agents.[21] The architecture is self-similar in that ANTS system elements and subelements can be structured recursively,[22] and it is self-managing, with at least one ruler (AM) per ANTS craft.

NASA missions such as ANTS provide a trusted private environment, eliminating many agent security issues and enabling system designers to focus on ensuring that agents are operating in the correct context and exhibiting emergent behavior within acceptable parameters.

In considering the role of the self-destruct property inspired by apoptosis, suppose one of the worker craft in the ANTS mission was operating incorrectly and, when coexisting with other workers, was causing undesirable emergent behavior and failing to self-heal correctly. That emergent behavior could put the mission in danger, and ultimately the ruler would withdraw the "stay alive" signal.[3] Likewise, if a worker or its instrument was damaged, either by colliding with another worker or (more likely) an asteroid, or during a solar storm, the ruler would withdraw the "stay alive" signal and request a replacement worker. Another worker would then self-configure to take on the role of the lost worker to ensure optimal balanced coverage of tasks to meet the scientific goals. If a ruler or messenger was similarly damaged, its ruler would withdraw the "stay alive" signal and promote a worker to play its role.

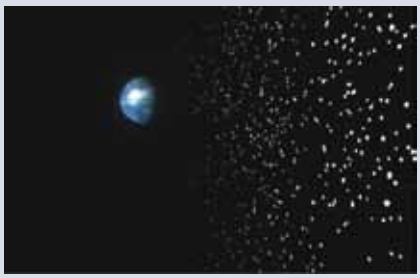

Figure 4. NASA's new space exploration paradigm calls for missions involving thousands of small spacecraft rather than a single large craft. Image courtesy of NASA.

THE EVOLVING STATE OF THE ART

Several researchers have investigated the apoptotic computing concept and its potential applications.

Christian Tschudin initially suggested using apoptosis in highly distributed systems.[23]

James Riordan and Dominique Alessandri proposed apoptosis as a means to automatically counter the increasing number of security vulnerabilities that hackers publish and exploit before systems administrators can close them.[24] They described an apoptosis service provider that, should a system vulnerability be found, could release a message into the environment to trigger various preconfigured responses to shut down the system or warn a responsible party.

Leszek Lilien and Bharat Bhargava argued for apoptosis as a means to secure atomic bundles of private data, in which the process is activated when detectors determine a credible threat exists to the bundle by any host, including the bundle's destination.[25]

In drawing parallels between biology and computing, Steve Burbeck proposed four interconnected principles for managing evolving systems, one of which is apoptosis.[26] As an example of this principle, he cited the Blue Screen of Death, a programmed response to an unrecoverable error. Burbeck argued that a computer, like a metazoan cell, should be able to sense its own rogue behavior, such as downloading uncertified code, and disconnect itself from the network.

M.M. Olsen, N. Siegelmann-Danieli, and H.T. Siegelmann developed a multiagent system called HADES that can protect itself via "life" protocols—which control the replication, repair, movement, and self-induced death of each agent—and a "rescue" protocol.[27]

Madihah Mohd Saudi and colleagues researched apoptosis with respect to security systems, focusing on

network problems, and later applied it specifically to worm attacks.[28,29]

Finally, David Jones implemented apoptotic self-destruct and "stay alive" signaling while investigating memory requirements in inheritance versus an abstract-oriented approach.[30]

The majority of these applications fall into the weak apoptotic computing (programmed death) category, and would likely benefit from, instead, utilizing a strong (programmed death by default) approach. They also highlight the strong need for standards and trust requirements, with the immediate challenge of developing a DoS-resistant architecture.

The human body regulates vital functions such as heartbeat, blood flow, and cell growth and death, all without conscious effort. We must develop computer-based systems that can perform similar operations on themselves without constant human intervention.

Promising apoptotic computing applications have been developed for data objects, highly distributed systems, services, agent systems, and swarm systems. However, more applied work is needed in other areas, and researchers must address the challenges around trust—until then, users are not likely to embrace a system with self-destruct capabilities.

The case has been made that all computer-based systems should be autonomic.[31] Likewise there is a compelling argument that all such systems should be apoptotic, especially as computing becomes increasingly pervasive and ubiquitous. Apoptotic controls should cover all levels of human-computer interaction from data, to services, to agents, to robotics. With recent headline incidents of credit card and personal data losses by organizations and governments, and scenarios once relegated to science fiction becoming increasingly possible, programmed death by default is a necessity.

We are rapidly approaching the time when new autonomous computer-based systems and robots should undergo tests, similar to ethical and clinical trials for new drugs, before they can be introduced. Emerging research from apoptotic computing could guide the safe deployment of such systems. C

Acknowledgments

The author is supported by the University of Ulster's Computer Science Research Institute and School of Computing and Mathematics. Some of the research described in this article is patented and patent-pending by Roy Sterritt and Mike Hinchey (Lero—the Irish Software Engineering Research Centre) through NASA and assigned to the US government. In memory of our dear friend and colleague Scott Hamilton.

References

1. J. Arlidge, "Scientists Fear a Revolt by Killer Robots," *The Sunday Times*, 2 Aug. 2009; http://technology.timesonline.co.uk/tol/news/tech_and_web/article6736130.ece.
2. R. Sterritt and M. Hinchey, "SPAACE IV: Self-Properties for an Autonomous & Autonomic Computing Environment—Part IV: A Newish Hope," *Proc. 7th IEEE Int'l Conf. and Workshops Eng. of Autonomic and Autonomous Systems* (EASe 10), IEEE CS Press, 2010, pp. 119-125.
3. R. Sterritt and M. Hinchey, "Apoptosis and Self-Destruct: A Contribution to Autonomic Agents?" *Proc. 3rd Int'l Workshop Formal Approaches to Agent-Based Systems* (FAABS 04), LNCS 3228, Springer, 2004, pp. 262-270.
4. R. Sterritt and M. Hinchey, "Engineering Ultimate Self-Protection in Autonomic Agents for Space Exploration Missions," *Proc. 12th IEEE Int'l Conf. and Workshops Eng. of Computer-Based Systems* (ECBS 05), IEEE CS Press, 2005, pp. 506-511.
5. R. Sterritt and M. Hinchey, "From Here to Autonomicity: Self-Managing Agents and the Biological Metaphors That Inspire Them," *Proc. 8th Int'l Conf. Integrated Design and Process Technology* (IDPT 05), Soc. for Design and Process Science, 2005, pp. 143-150.
6. R. Sterritt and M. Hinchey, "Biologically-Inspired Concepts for Autonomic Self-Protection in Multiagent Systems," M. Barley et al., eds., *Safety and Security in Multi-Agent Systems: Research Results from 2004-2006*, LNCS 4324, Springer, 2009, pp. 330-341.
7. J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing," *Computer*, Jan. 2003, pp. 41-52.
8. M.G. Hinchey and R. Sterritt, "Self-Managing Software," *Computer*, Feb. 2006, pp. 107-109.
9. S. Dobson et al., "Fulfilling the Vision of Autonomic Computing," *Computer*, Jan. 2010, pp. 35-41.
10. R.A. Lockshin and Z. Zakeri, "Programmed Cell Death and Apoptosis: Origins of the Theory," *Nature Reviews Molecular Cell Biology*, July 2001, pp. 542-550.
11. Y. Ishizaki et al., "Programmed Cell Death by Default in Embryonic Cells, Fibroblasts, and Cancer Cells," *Molecular Biology of the Cell*, Nov. 1995, pp. 1443-1458.
12. J. Klefstrom, E.W. Verschuren, and G. Evan, "c-Myc Augments the Apoptotic Activity of Cytosolic Death Receptor Signaling Proteins by Engaging the Mitochondrial Apoptotic Pathway," *J. Biological Chemistry*, 8 Nov. 2002, pp. 43224-43232.
13. M. Sluyser, ed., *Apoptosis in Normal Development and Cancer*, Taylor & Francis, 1996.
14. J. Newell, "Dying to Live: Why Our Cells Self-Destruct," *Focus*, Dec. 1994; http://members.fortunecity.com/templarseries/Yahoo/Omegaman/apoptosi.html.
15. N.R. Jennings and M. Wooldridge, "Agent-Oriented Software Engineering," J. Bradshaw, ed., *Handbook of Agent Technology*, AAAI/MIT Press, 2000.
16. M.N. Huhns, V.T. Holderfield, and R.L.Z. Gutierrez, "Robust Software via Agent-Based Redundancy," *Proc. 2nd Int'l Joint Conf. Autonomous Agents and Multiagent Systems* (AAMAS 03), ACM Press, 2003, pp. 1018-1019.
17. G. Kaiser et al., "Kinesthetics eXtreme: An External Infrastructure for Monitoring Distributed Legacy Systems," *Proc. Autonomic Computing Workshop—5th Ann. Int'l Workshop Active Middleware Services* (AMS 03), IEEE Press, 2003, pp. 22-30.

18. R. Sterritt and D. Bustard, "Towards an Autonomic Computing Environment," *Proc. 14th Int'l Workshop Database and Expert Systems Applications* (DEXA 03), IEEE CS Press, 2003, pp. 694-698.
19. E. Lupu et al., "AMUSE: Autonomic Management of Ubiquitous Systems for e-Health," *Concurrency and Computation: Practice and Experience*, Mar. 2003, pp. 277-295.
20. M.S. Greenberg et al., "Mobile Agents and Security," *IEEE Comm. Magazine*, July 1998, pp. 76-85.
21. P.E. Clark et al., "ANTS: A New Concept for Very Remote Exploration with Intelligent Software Agents," *Eos, Trans., American Geophysical Union*, vol. 82, no. 47, 2001.
22. S. Curtis et al., "ANTS (Autonomous Nano Technology Swarm): An Artificial Intelligence Approach to Asteroid Belt Resource Exploration," *Proc. 51st Int'l Astronautical Congress, Int'l Astronautical Federation*, 2000; http://ants.gsfc.nasa.gov/documents.d/iaf2000-ants.pdf.
23. C. Tschudin, "Apoptosis—The Programmed Death of Distributed Services," J. Vitek and C. Jensen, eds., *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, LNCS 1603, Springer, 1999, pp. 253-260.
24. J. Riordan and D. Alessandri, "Target Naming and Service Apoptosis," *Proc. 3rd Ann. Workshop Recent Advances in Intrusion Detection* (RAID 00), LNCS 1907, Springer, 2000, pp. 217-225.
25. L. Lilien and B. Bhargava, "A Scheme for Privacy-Preserving Data Dissemination," *IEEE Trans. Systems, Man, and Cybernetics, Part A: Systems and Humans*, May 2006, pp. 503-506.
26. S. Burbeck, "Complexity and the Evolution of Computing: Biological Principles for Managing Evolving Systems," v2.2, white paper, 9 Apr. 2007; http://evolutionofcomputing.org/Complexity%20and%20Evolution%20of%20Computing%20v2.pdf.
27. M.M. Olsen, N. Siegelmann-Danieli, and H.T. Siegelmann, "Robust Artificial Life via Artificial Programmed Death," *Artificial Intelligence*, Apr. 2008, pp. 884-898.
28. M.M. Saudi et al., "An Overview of Apoptosis for Computer Security," *Proc. Int'l Symp. Information Technology* (ITSim 08), IEEE Press, 2008, pp. 2534-2539.
29. M.M. Saudi et al., "An Overview of STAKCERT Framework in Confronting Worms Attack," *Proc. 2nd IEEE Int'l Conf. Computer Science and Information Technology* (ICCSIT 09), IEEE Press, 2009, pp. 104-108.
30. D. Jones, "Implementing Biologically-Inspired Apoptotic Behaviour in Digital Objects: An Aspect-Oriented Approach," MSc dissertation, Open University, UK, 2010; http://www.apoptotic-computing.org/media/Apoptotic-Computing-MSc-Dissertation.pdf.
31. R. Sterritt and M. Hinchey, "Why Computer-Based Systems Should be Autonomic," *Proc. 12th IEEE Int'l Conf. Workshops Eng. Computer-Based Systems* (ECBS 05), IEEE CS Press, 2005, pp. 406-414.

Roy Sterritt *is a faculty member in the School of Computing and Mathematics and a researcher in the Computer Science Research Institute at the University of Ulster, Northern Ireland. His research focuses on the engineering of computer-based systems, in particular self-managing/autonomic systems. Sterritt received a BSc in computing and information systems and an MA in business strategy from the University of Ulster. He is a member of IEEE and the IEEE Computer Society. Contact him at r.sterritt@ulster.ac.uk.*