FULFILLING THE VISION OF AUTONOMIC COMPUTING

Simon Dobson, *University of St. Andrews, UK*
Roy Sterritt, *University of Ulster, Northern Ireland*
Paddy Nixon and Mike Hinchey, *Lero—the Irish Software Engineering Research Centre*

Efforts since 2001 to design self-managing systems have yielded many impressive achievements, yet the original vision of autonomic computing remains unfulfilled. Researchers must develop a comprehensive systems engineering approach to create effective solutions for next-generation enterprise and sensor systems.

In 2001, IBM researchers predicted that by the end of the decade the IT industry would need up to 200 million workers, equivalent to the entire US labor force, to manage a billion people and millions of businesses using a trillion devices connected via the Internet.[1,2] Only if computer-based systems became more autonomic—that is, to a large extent self-managing—could we deal with this growing complexity, and they accordingly issued a formal challenge to researchers.

We have reached 2010, and, much like the Y2K problem, the situation clearly is not as extreme as anticipated. So was it all hype, or has the IT industry had a very productive decade? Have we met IBM's challenge, or have we simply performed another heroic effort without solving the underlying problem?

BACK TO THE FUTURE

In its autonomic computing call to arms, IBM compared what the IT industry faced in 2001 to what occurred in the US telephony industry in the 1920s. At that time, the rapid expansion and infiltration into daily life of the phone aroused serious concern that there would not be enough trained operators to work the manual switchboards. Analysts predicted that by the 1980s, half the country's population would have to become telephone operators to meet demand. AT&T/Bell System's implementation of the automated switching protocol and other technological innovations averted this crisis.

In 2001, unfilled IT jobs in the US alone numbered in the hundreds of thousands, even in uncertain economic conditions, and global demand for IT workers was expected to increase by more than 100 percent in the next five years. Today's actual employment numbers are hard to determine, as government statistics do not explicitly capture system administration, IT maintenance, and other related functions. However, crude data from the Bureau of Labor Statistics suggests that there are approximately 260,000 IT workers in the US, with employment in the industry declining slightly but steadily over the past decade[3] despite the enormous increase in computing power available. This trend suggests that consolidation of computing power, which will increase alongside the use of cloud computing and Web 2.0, reduces the amount of management per unit of service.

The story is not that simple, of course.

Seven years ago, Jeffrey Kephart and David Chess published "The Vision of Autonomic Computing" in *Computer*,[4] setting forth IBM's autonomic computing manifesto[1] in the specific context of enterprise systems management. This article has been wildly influential, with more than 1,100 direct citations according to Google Scholar. Moreover, the study of autonomic systems has become a significant component of systems research, with its own dedicated journals, conferences, and IEEE Computer Society technical committee (TCAAS), as well as a substantial presence in mainstream computing and networking venues.

The most widely recognized elements of autonomic systems are the so-called self-* properties.

The vision of autonomic computing represents a surprising combination of revolution and retrenchment. By focusing on total costs of ownership for enterprise systems, Kephart and Chess highlighted the central impact that IT systems can have on the core economics of modern businesses. Indeed, the deployment, maintenance, and evolution of enterprise systems often require enormous efforts by extremely valuable staff, whose successes add little visible business value but are nevertheless vital and whose failures can be catastrophic for the whole enterprise. Autonomic computing, in its broadest sense, seeks to reduce the need for such heroic efforts and their consequential risks.

To what extent is the vision set forth by Kephart and Chess being fulfilled? What is the status of autonomic computing systems research in its current realization, and how has it influenced research thinking?

THE BROADENING VISION

The increasing use of information systems to collect, analyze, locate, collate, summarize, and otherwise process information has had an immense impact on modern life. That so much of this change has occurred in back offices makes it easy to underestimate the extent to which the design, construction, and especially maintenance of these systems challenge our capabilities as engineers. Feature interaction is a major cause of system failures, and its avoidance is a major cost for system administrators deploying new features.

In some minds autonomic computing today remains closely associated with the original IBM initiative, but to the IEEE and other organizations the term more broadly describes the application of advanced technology to the management of advanced technology. Similar proposed visions are clearly related: organic computing, bio-inspired computing, self-organizing systems, ultrastable computing, autonomous and adaptive systems, to name a few. We use the term *autonomic* to encompass all of these initiatives.

Enterprise systems are only one member of a class of complicated systems that must function consistently and reliably in the absence of detailed human involvement. Many management tasks can no longer be handled with sufficient efficiency by manual operators, however skilled: The system itself must take responsibility to adapt its own operation in the face of changing conditions. This need for self-adapting behavior characterizes the domains in which autonomic computing ideas are gaining traction.

To take two examples:

- The main cost for the operator of a data center is power, thus the provisioning of systems to match workloads and service-level obligations becomes a critical business success factor. Because workload demands change minute by minute, no human operator can provision services with sufficient efficiency.
- Applications like environmental sensing cause the network to meet the real world in ways that preclude direct human management. The viability of environmental sensing—essential for effective science and policymaking—therefore depends on sensor systems' ability to self-manage in the face of a changing environment.

The most widely recognized elements of autonomic systems are the so-called self-* properties: For systems to be self-managing they should be *self-configuring*, *self-healing*, *self-optimizing*, and *self-protecting* and exhibit *self-awareness*, *self-situation*, *self-monitoring*, and *self-adjustment*. Despite their seeming simplicity, these goals mask a complex interaction between the behaviors of systems and their goals, users, and relationships with the external environment. We can only optimize a system against some external criteria, so self-optimization implies that these criteria are made available in some way to the management system. Moreover, composition and analysis of systems probably imply that the criteria be explicit, symbolic, and machine-readable rather than embedded implicitly into algorithms.

In thinking of systems rather than simply of machines, we must also consider communications a component of the problem space,[5] the most notable omission from Kephart's and Chess's vision. Mikhail Smirnov[6] propounded the notion of autonomic communications based on David Clark and colleagues' call for a *knowledge plane* for the Internet,[7] and it has become an active research topic,[8] especially in Europe, where it has received considerable support from the EU's Framework programs. Considering communications as well as computing naturally leads to an exploration of the interplay of these different aspects.

THE EVOLVING STATE OF THE ART

As the "Autonomic Computing: Biological Inspiration" sidebar describes, the term autonomic suggests an analogy to the biological nervous system, with the self-* properties similar to those of homeostasis and responsiveness, as well as to the more conventional notions of closed-loop feedback and control. Critics have argued that the autonomic computing field simply synthesizes results from other areas, but this view ignores the breadth that comes from a whole-systems focus. Indeed, we believe that autonomic systems research has the potential to provide a broad systems theory for open adaptive systems.

It is perhaps best to start with the driving forces. Systems are exhibiting rapid increases in complexity of construction, evolution, and management. Putting enterprise systems together is difficult; changing them to meet changing business conditions is complicated by unexpected dependencies and limitations imposed by earlier design decisions; and managing a system to maintain adequate quality of service in the face of a dynamic environment tests the abilities of human managers.

Developers considering the evolution and management of systems in terms of self-* properties must take a different perspective—for example, by including programmatic monitoring and management interfaces. Such a perspective, while common in telecommunications in the form of managed components, is unusual in software architectures still based largely on configuration files read only at start-up time.

As Figure 1 shows,[8] providing monitoring and control suggests the application of control theory—expressing a control action derived from a system's observed behavior against a model of intended or expected behavior. Researchers have successfully applied such techniques to, for example, power management,[9] to achieve clear closed-form representations. However, it is less clear whether the techniques can be applied more broadly in areas where the control domain changes dynamically.

There are numerous alternative implementations of control-theoretic ideas. Clark and colleagues' vision of a knowledge plane maintaining a configuration view of a system[7] allows for distributed access to nonlocal information, which in turn can inform control systems built from agents that act to optimize some local aspect. Such systems' richness comes from the interaction of agents within the agent ecosystem, which can make it difficult to predict overall system behavior in any given circumstances.

The analogy of autonomic computing to biology is proving extremely fruitful. Researchers are using swarm and ant-colony models to coordinate robot behavior, sometimes in combination with more traditional formal methods.[10] The "ANTS: A Milestone in Autonomic Computing" sidebar describes one notable project. Stigmergic approaches capture the notion of depositing time-limited information into an environment to influence later computations. Physics provides another source of inspiration—for example, using models derived from electromagnetic field theory to control load and communications balancing.[11]

The Internet has become the de facto pervasive communications system across the world today, providing the means for cloud computing. Its success lies in its generality and heterogeneity, the combination of a simple transparent network (the data plane) with rich end-system functional-

AUTONOMIC COMPUTING: BIOLOGICAL INSPIRATION

IBM's autonomic computing initiative largely derives its inspiration from the biological nervous system.[1] The idea is that built-in regulatory mechanisms in the body that require no conscious thought can suggest the construction of mechanisms that will likewise enable a computer system to become self-managing.

The body's sympathetic nervous system deals with defense and protection ("fight or flight"), and the parasympathetic nervous system deals with long-term health ("rest and digest"), performing vegetative functions such as circulation of the blood, intestinal activity, and hormone secretion. An autonomic computing system similarly tries to ensure its continued health and well-being by sending and monitoring various signals.

An autonomic system has four objectives—to be self-configuring, self-healing, self-optimizing, and self-protecting. These represent broad system requirements. To achieve these objectives, the system must be aware of its internal state (self-awareness) and current external operating conditions (self-situation), detect changing circumstances (self-monitoring), and accordingly adapt (self-adjustment). These four attributes constitute implementation mechanisms. The system must therefore have knowledge of its available resources as well as its components, their desired performance characteristics, their current status, and the status of interconnections with other systems, along with rules and policies of how these may be adjusted.[2]

The self-managing mechanisms in an autonomic computing system are not independent entities. For instance, a successful attack on the system will necessitate self-healing actions and a mix of self-configuration and self-optimization, initially to ensure dependability and continued system operation and later to increase self-protection against similar future attacks. They should also minimize disruption to users by avoiding significant delays in processing.

References

1. M.G. Hinchey and R. Sterritt, "99% (Biological) Inspiration…," *Proc. 4th IEEE Int'l Workshop Eng. of Autonomic and Autonomous Systems* (EASe 08), IEEE CS Press, 2007, pp. 187-195.
2. R. Sterritt and D.W. Bustard, "Autonomic Computing—A Means of Achieving Dependability?" *Proc. 10th IEEE Int'l Conf. and Workshop on the Eng. of Computer-Based Systems* (ECBS 03), IEEE Press, pp. 247-251.

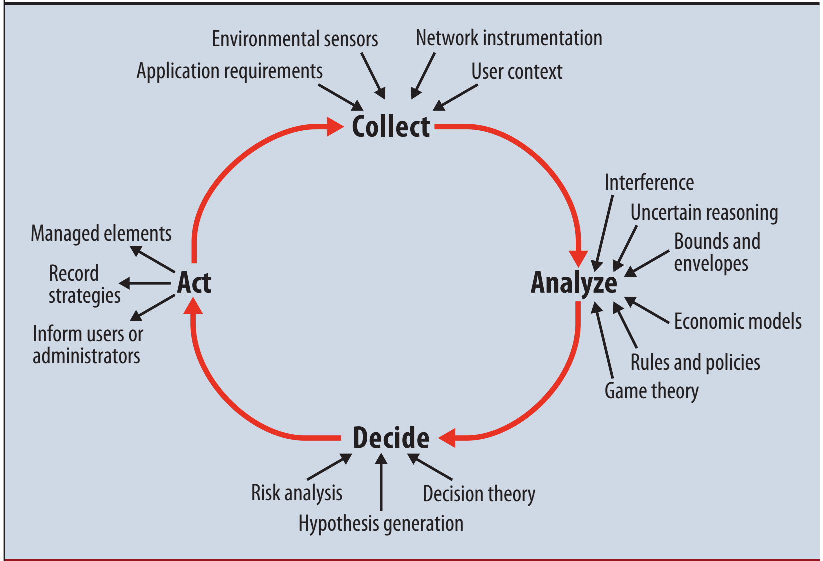

Figure 1. Technologies applied to the four stages of the autonomic control loop. Although inspired by control theory, this structure encompasses symbolic and other techniques within a common framework as well as aspects of both computing and communications.

ity. However, it has high manual configuration, diagnosis, and design costs, and problems become apparent when something fails. Because the Internet has a simple and transparent core and intelligence lies at the edge, the network carries data without knowing its nature or purpose. If a combination of events prevents data from reaching the edge, the core may recognize a problem but has no idea what should be happening.

Researchers recognize that a new construct is needed for next-generation communication networks, a pervasive system within the network that builds and maintains high-level models of what the network is supposed to do to provide services and advice to other network elements. The knowledge plane would function as a global, decentralized overlay to the transparent network that aggregates global information, observations, assertions, requirements, constraints, and goals.[7] In terms of fault detection and isolation, it would facilitate cross-correlation assessment, with diagnoses traveling up to the knowledge plane and conclusions being passed down. Knowledge plane proponents argue that it can apply machine-learning algorithms to garner knowledge and increase self-awareness.

How to achieve the knowledge plane is an open question, although given the uncertainties and complexities involved it would likely rely on AI and cognitive system tools rather than traditional algorithmic approaches—possible building blocks include epidemic algorithms for distributing data and Bayesian networks for learning. Because the knowledge plane resides in a different space than the data and control planes, it does not move data directly, nor is it responsible for management functions such as accounts.[7,12-13] Researchers are exploring the use of mobile and static agents to provide network knowledge points.

A SYSTEMS THEORY FOR ADAPTIVE SYSTEMS

Autonomic computing techniques provide sophisticated and often extremely impressive solutions to problems that until recently would have been intractable without the intervention of a skilled human operator. However, researchers still lack an understanding of the broader software engineering aspects of autonomic system development. The International Conference on Software Engineering's SEAMS (Software Engineering for Adaptive and Self-Managing Systems) workshop and the IEEE's EASe (Engineering of Autonomic and Autonomous Systems) conference and workshops provide forums to explore fundamental questions about the requirements, specification, and verification of such systems.

Some might argue that adaptive systems do not really differ from other systems: They map an input space to an output space of behavior and actuation. However, the input space may include elements of the environment in which the system operates. In adaptive provisioning, for example, the "environment" includes estimates of expected tasks extrapolated from the previously observed workload. We know from the development of traditional control systems that such feedback requires careful design if the system is not to diverge or exhibit other undesirable dynamics. We also know that this is difficult to accomplish in systems in which the behavior is underspecified and expected to vary over time.

Further, what does it mean for a system to be "correct" when its behavior is expected to change over time? Perhaps a better question is, How can we describe the range of acceptable possibilities for a system's behavior as well as the preferred behavior at any given instant, and over any given sequence of events? Rather than accepting that systems must simply be "point-correct" in response to a situation at any given time, they must also be "process-correct" by responding correctly to changing situations.[14]

In addition, system management is not simply a combination of independent choices, but rather the balancing of a range of possibilities to obtain the best overall result. It is not enough to state, for example, that an autonomic power management subsystem can reduce power requirements: We must be able to state the bounds within which the power demand will vary, its impact on response times, and its interaction with subsystems that may affect load or

ANTS: A MILESTONE IN AUTONOMIC COMPUTING

Autonomous Nanotechnology Swarm[1,2] is a concept NASA mission that represents a significant achievement in autonomic computing.[3] In one ANTS submission, the Prospecting Asteroid Mission (PAM),[4] a transport ship launched from Earth will travel to a point in space where gravitational forces on small objects are all but negligible. From this point, termed a Lagrangian, the transport ship will launch 1,000 pico-class spacecraft assembled en route into the asteroid belt. It is expected that as much as 60 to 70 percent of the craft will be lost during the mission, primarily due to collisions with each other or with an asteroid during exploration operations, since, having only solar sails to provide thrust, their ability to maneuver will be severely limited. Because of their small size, each spacecraft will carry just one specialized instrument to collect data from asteroids in the belt.

Approximately 80 percent of the spacecraft will be *workers* that carry the specialized instruments—a magnetometer or an x-ray, gamma-ray, visible/infrared, or neutral mass spectrometer—to obtain specific types of data. Some will be *rulers* that decide the types of asteroids and data the mission is interested in and that will coordinate the workers' efforts. Finally, *messengers* will manage communication between the rulers and workers, and between the swarm and the Earth ground station.

As Figure A shows, the swarm will form subswarms under the control of a ruler, which contains models of the types of science that it wants to perform. Each worker uses its individual instrument to collect data on specific asteroids and feeds this information back to the ruler, which will determine which asteroids are worth examining further. If the data matches the profile of an asteroid of interest, the swarm will send an imaging spacecraft to ascertain its exact location and create a rough model for other spacecraft to use when maneuvering around the asteroid. Other teams of spacecraft will finish mapping the asteroid to form a complete model.

New approaches to space exploration missions such as ANTS augur great potential but also pose many challenges. The missions will be unmanned and necessarily highly autonomous; to assist in survivability, the swarms will be self-protecting, self-healing, self-configuring, and self-optimizing.[5,6] Many ANTS missions will be sent to parts of the solar system beyond the reach of manned spacecraft and to where the round-trip communications delay exceeds 40 minutes, meaning that responses to problems and undesirable situations must be made in situ rather than from ground control on Earth. Future swarm-based missions may employ additional techniques and self-* paradigms.[7]

References

1. W. Truszkowski et al., "NASA's Swarm Missions: The Challenge of Building Autonomous Software," *IT Professional*, vol. 6, no. 5, 2004, pp. 51-56.
2. W.F. Truszkowski et al., "Autonomous and Autonomic Systems: A Paradigm for Future Space Exploration Missions," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 36, no. 3, 2006, pp. 279-291.
3. M.C. Huebscher and J.A. McCann, "A Survey of Autonomic Computing—Degrees, Models, and Applications," *ACM Computing Surveys*, article no. 7, vol. 40, no. 3, 2008.
4. P.E. Clark et al., "PAM: Biologically Inspired Engineering and Exploration Mission Concept, Components, and Requirements for Asteroid Population Survey," *Proc. 55th Int'l Astronautical Congress* (IAC 04), Int'l Astronautical Federation, 2004; http://ants.gsfc.nasa.gov/documents/Clark308IAC2004.pdf.
5. 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, 2005, pp. 262-270.
6. R. Sterritt and M. Hinchey, "Engineering Ultimate Self-Protection in Autonomic Agents for Space Exploration Missions," *Proc. 12th Int'l Conf. and Workshops Eng. Computer-Based Systems* (ECBS 05), IEEE CS Press, 2005, pp. 506-511.
7. W. Truszkowski et al., *Autonomous and Autonomic Systems: With Applications to NASA Intelligent Spacecraft Operations and Exploration Systems*, Springer, 2010.

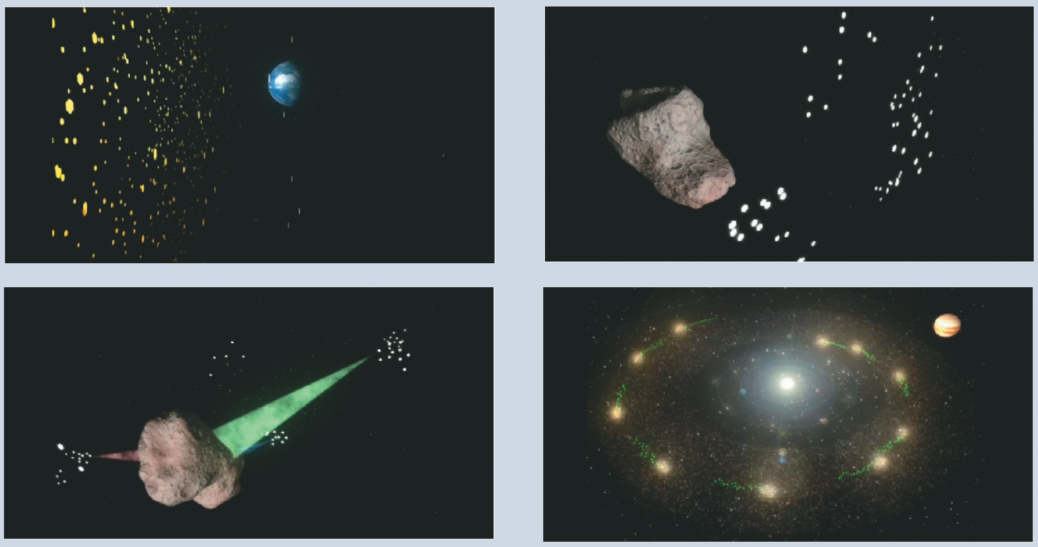

Figure A. Autonomous Nanotechnology Swarm visualization. ANTS is a concept NASA mission that, among other things, will launch highly autonomous swarms of pico-class spacecraft to explore asteroids.

communications, as these factors influence other choices within the management system. In other words, autonomic management requires autonomic mechanisms that are broadly self-describing in terms of the impacts they have across the range of system concerns.

Moreover, we must be able to prove that self-* properties are maintained; analyze the effects of changes on constraints through "what if?" scenarios; and perform these operations on systems composed from individual, independently developed subsystems.

These issues, which are vitally important to the continued development and deployment of autonomic systems, require a comprehensive systems theory for adaptive systems. No such theory has yet emerged, but elements of one are surfacing within the literature, and we can certainly sketch desiderata for any candidate theory.

An autonomic system functions as part of a wider environment and exists to fulfill some externally defined purpose. In describing the system, we must not lose sight of this purpose, suggesting that we capture its requirements and constraints within the description. These in turn provide an envelope within which we can adapt the system.

Adaptation is a dynamic process, not simply a functional response to a change in some variables. There may be many acceptable choices, from which we choose a particular one to exhibit. The different choices may emphasize one system aspect over another. The choices, and the way they vary over time, provide different dynamics through the space of possibilities, a formulation similar to those physicists use with dynamic systems.

For example, suppose the purpose of a wireless sensor network is to sense a pollutant concentration over an extended time. To accomplish this, we must balance the system's behavior between two extremes. At one pole, the system senses and communicates constantly and expends available power quickly; at the other, it performs no sensing or communication at all and expends virtually no power. Exactly where in the spectrum the system lies is a dynamic decision that might use a baseline of infrequent sensing overlaid with more rapid activity when the network observes "interesting times" and with intensified communications as the battery runs down, to extract the last value from a sensor node.

We may want to select different adaptations depending on observed values of the pollutant as well as first and second time derivatives—for example, start sampling much more rapidly if pollution is increasing rapidly, and then decay sampling more slowly. Any choice will impact the nodes' longevity and the network overall, and there may be additional constraints on nodes acting as communications gateways or providing other specialized services. The point is that the system description, purpose, and dynamics are essentially one piece and must be stated, analyzed, and evolved as such.

Of course, the mathematics of dynamic systems is nowhere near rich enough to describe all the features of modern enterprise computing. In physics, researchers typically model an already-existing system; in computing, researchers seek to design a system with certain properties, to build complexity by combining subsystems, and to predictably evolve that system over time. Put another way, we need an approach for critical systems that combines adaptive systems analysis with evolving software engineering techniques.

Efforts since 2001 to create self-managing systems have yielded many improvements, yet IBM's original vision of autonomic computing remains unfulfilled. Indeed, the need for progress is actually greater: IT departments are scrambling to put out fires as technologies such as VoIP and new applications converge, adding even more complexity to our systems.

How we as humans manage an event depends on multiple factors: our mood; the time, place, and circumstances; what happened to us earlier in the day; what we plan to do later; and so on. The sheer complexity of influences on a moment in time and an evolving situation are mind-boggling. Have we as engineers therefore set an impossible goal for ourselves? A computer-based system can be programmed to do tasks but has no emotion, no passion, no soul; how can we create true autonomy from a collection of programs?

Over the years, the AI field has fallen victim to unrealistic expectations, and we see similar warning signs in the autonomic computing field. Yet from the beginning there has been a successful focus on evolutionary research, tightly linked to applied industrial problems. Additional funding and industrial collaboration are crucial to future success, but something more is required: Researchers must develop a long-range, overarching strategy to realize the vision propounded by Kephart and Chess.

That vision has enriched the landscape of systems research, and has in turn been enriched by insights gained within the past decade. It has resulted in a substantial roster of achievements, especially in enterprise and cloud computing but also in communications, sensor networks, and other fields. The idea that computer-based systems can and should be self-managing is having an enormous impact on system design and evaluation.

Yet in some ways that success is deceptive. Researchers have devised innovative autonomic solutions to individual problems, but the larger, more difficult task of combining these point solutions into autonomic systems remains. More consideration must be given to integrating solutions, and to choosing solutions from the range of possibilities—to *autonomic systems engineering*, in other words. Without such an approach, we will simply rediscover the risks of feature interaction at a higher level, and in a way that is so dynamic as to be resistant to debugging and testing. We are confident, however, that the foundation exists to construct a systems theory and practice from which we can engineer autonomic solutions for the next generation of enterprise and sensor systems. C

Acknowledgments

This work is partially supported by Science Foundation Ireland under grant number 03/CE2/I303_1, "Lero—the Irish Software Engineering Research Centre," and the University of Ulster's Computer Science Research Institute (CSRI). The authors also wish to acknowledge the ideas and support of Dave Bustard, Kevin Ryan, Brian Fitzgerald, Lorcan Coyle, M.A. Razzaque, Norah Power, Ron Black, Jim Rash, Chris Rouff, Walt Truszkowski, and Manila Rhythm.

References

1. P. Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology," 15 Oct. 2001, IBM Research; www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf.
2. E. Mainsah, "Autonomic Computing: The Next Era of Computing," *Electronics & Comm. Eng. J.*, vol. 14, no. 1, 2002, pp. 2-3.
3. US Dept. of Labor Bureau of Labor Statistics, "Current Employment Statistics, Table B-1: Employees on Nonfarm Payrolls by Major Industry Sector, 1959 to Date;" ftp://ftp.bls.gov/pub/suppl/empsit.ceseeb1.txt.
4. J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing," *Computer*, Jan. 2003, pp. 41-50.
5. R. Sterritt, "Towards Autonomic Computing: Effective Event Management," *Proc. 27th Ann. NASA Goddard Software Eng. Workshop* (SEW 02), IEEE CS Press, 2002, pp. 40-47.
6. M. Smirnov, *Autonomic Communication: Research Agenda for a New Communications Paradigm*, tech. report, Fraunhofer FOKUS, 2004.
7. D.D. Clark et al., "A Knowledge Plane for the Internet," *Proc. 2003 Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm.* (Sigcomm 03), ACM Press, 2003, pp. 3-10.
8. S. Dobson et al., "A Survey of Autonomic Communications," *ACM Trans. Autonomous and Adaptive Systems*, vol. 1, no. 2, 2006, pp. 223-259.
9. D. Kusic et al., "Power and Performance Management of Virtualized Computing Environments via Lookahead Control," *Proc. 2008 Int'l Conf. Autonomic Computing* (ICAC 08), IEEE CS Press, 2008, pp. 3-12.
10. C.A. Rouff et al., "Towards a Hybrid Formal Method for Swarm-Based Exploration Missions," *Proc. 29th Ann. IEEE/NASA Software Eng. Workshop* (SEW 05), IEEE CS Press, 2005, pp. 253-264.
11. M. Mamei and F. Zambonelli, "Programming Stigmergic Coordination with the TOTA Middleware," *Proc. 4th Int'l Joint Conf. Autonomous Agents and Multiagent Systems* (AAMAS 05), ACM Press, 2005, pp. 415-422.
12. J.M. Agosta and S. Crosby, "Network Integrity by Inference in Distributed Systems," *NIPS Workshop on Robust Communication Dynamics in Complex Networks*, NIPS Foundation, 2003; www.agosta.org/pubs/NIPS03/Agosta.pdf.
13. R. Sterritt, S. Dobson, and M. Smirnov, eds., *Proc. IJCAI Workshop on AI and Autonomic Comm.*, IJCAI, 2005.
14. J. Coutaz et al., "Context Is Key," *Comm. ACM*, vol. 48, no. 3, 2005, pp. 49-53.

Simon Dobson *is a professor in the School of Computer Science at the University of St Andrews, UK. His research focuses on the design, programming, and analysis of autonomic and sensor systems. Dobson received a DPhil in computer science from the University of York, UK. He is a senior member of the IEEE and the ACM and a fellow of the British Computer Society (BCS). Contact him at sd@cs.st-andrews.ac.uk.*

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 from the University of Ulster. He is a member of the IEEE and the IEEE Computer Society. Contact him at r.sterritt@ulster.ac.uk.*

Paddy Nixon *is a research area leader at Lero—the Irish Software Engineering Research Centre. and Science Foundation Ireland Research Professor in Distributed Systems at University College Dublin. His research interests lie in the areas of pervasive and autonomic computing. Nixon received a PhD in computer science from the University of Sheffield, UK. He is a chartered engineer and a member of the IEEE, the ACM, and the BCS. Contact him at paddy.nixon@ucd.ie.*

Mike Hinchey *is scientific director of Lero—the Irish Software Engineering Research Centre and a professor of software engineering at the University of Limerick, Ireland. His research interests include self-managing software and formal methods for system development. Hinchey received a PhD in computer science from the University of Cambridge. He is a senior member of the IEEE and currently chairs the IFIP Technical Assembly. Contact him at mike.hinchey@lero.ie.*