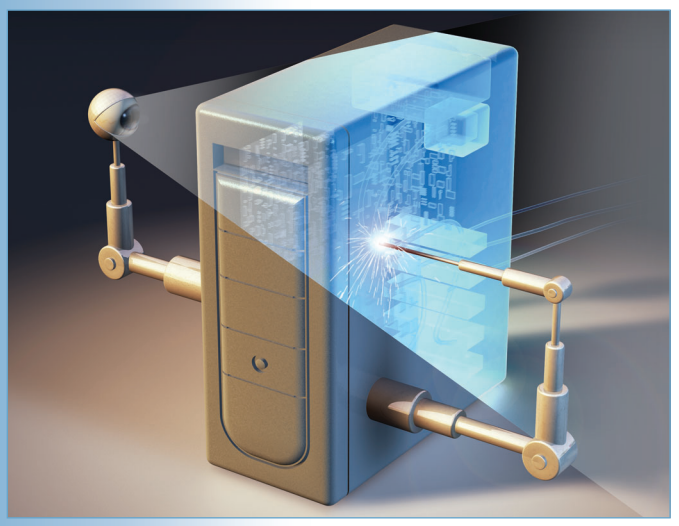

Autonomic Computing

Daniel A. Menascé
George Mason University

Jeffrey O. Kephart
IBM Research

In October 2001, Paul Horn, IBM's senior vice president of research, coined the term *autonomic computing* to describe a solution to the ever-growing complexity crisis that threatens to thwart IT's future growth.[1] In this vision, systems manage themselves in accordance with high-level behavioral specifications from administrators — much as our autonomic nervous system automatically increases our heart and respiratory rates when we exercise.

In five years, autonomic computing has evolved into a new subdiscipline of computer science. Academics and industry professionals attend dozens of conferences and workshops and write hundreds of papers each year; universities around the world are offering dozens of courses on the subject; government agencies in Europe and the US are supporting a variety of autonomic computing research projects; and several industry labs have substantial research and development efforts in autonomic computing.

Spurred by this momentum, the worldwide community has progressed on many fronts. Dozens of products from both well-established vendors and start-ups offer hundreds of autonomic features that ease the administrator's burden. Systems-management standards efforts such as Oasis's Web Services Distributed Management (WSDM) are under way, aiming to facilitate the creation of multivendor autonomic computing systems. However, we've by no means reached our destination. Although many autonomic components have been developed and are proving useful in their own right, no one has yet built a large-scale, fully autonomic computing system — comprising multiple components that work together to satisfy high-level business goals — that exhibits the ability to configure, heal, optimize, and protect itself.

Autonomic Data Center

What might a truly autonomic computing system look like? As an example, consider a hypothetical autonomic data center of the future that hosts several applications or services on behalf of multiple customers. From time to time, a customer will subscribe to a new service and specify preferred levels of performance,

1089-7801/07/$25.00 © 2007 IEEE

availability, and security in a service-level agreement (SLA). The autonomic data center will determine which resources are needed to satisfy the SLA, automatically provision and configure the appropriate physical or virtual server, database, storage, and networking resources in accordance with those objectives, and install and configure various application modules on those resources. Then, after the service is instantiated, the autonomic data center will monitor it, sensing faults, bottlenecks, and attacks, perhaps even anticipating impending ones. It will localize problems to specific databases, servers, routers, application modules, Web services, or virtual machines and work around those problems as necessary. It might even diagnose and fix them *in situ* by rebooting a failed component, identifying and applying a suitable patch, switching quickly to a backup component, or provisioning and configuring a new copy. It will protect itself by continually checking and upgrading its components to ensure compliance with security regulations, and by detecting intrusions and automatically taking measures to contain them and minimize their impact. As objectives change, services are added and removed, and workloads fluctuate, the autonomic data center will dynamically tune itself in myriad ways, exploiting any and all parameters at its disposal (such as load-balancing weights, thread priorities, CPU shares, clock frequencies and voltages on individual processors, and virtual machine size and placement) to best meet both the SLAs and any additional objectives the administrator has deemed important, such as minimizing power consumption.

This hypothetical autonomic data center exhibits several important self-management properties that will characterize all fully autonomic computing systems: self-configuration, self-healing, self-optimization, and self-protection.[1] (Indeed, autonomic computing systems are often referred to as *self-** systems.) These are by no means orthogonal dimensions of self-management; there is overlap among the various self-* capabilities and in the techniques used to achieve them — any real system will provide them in a uniform, integrated manner. It might be difficult to distinguish, for example, whether a component fails to respond to a request due to a component failure, a lack of resource, an overload condition, or a denial-of-service attack. Regardless, the means for detecting these various problems will require applying analyses to monitored data streams, and the response might entail isolating or shutting down a system component, rebooting it, patching it, or provisioning a similar one.

A survey of autonomic computing literature indicates that none of the capabilities ascribed to the hypothetical autonomic data center are entirely hypothetical — academia and industry are already developing many of the critical technologies underlying each of the main facets of self-* systems. To cite just one example, computer scientists and engineers have developed, prototyped, and evaluated several techniques for autonomic resource allocation in small-scale data centers.[2,3] Note also the diversity of techniques that they've used to build systems with some self-* properties: control theory,[4] queuing models combined with heuristic search techniques,[5] and machine learning.[3] Important gaps remain, however, and integrating all of these capabilities together in a single system remains a daunting challenge.[6]

It's easy (and true) to say that more research is needed to realize the autonomic computing vision.

Realizing the Vision

It's easy (and true) to say that more research is needed to realize the autonomic computing vision, but we can place certain emphases on that research to help move toward that vision more expeditiously:

- *General techniques for autonomic components.* Develop practical and general techniques that we can apply across a broad range of autonomic components. This includes tailoring and extending planning, optimization, control theory, and machine-learning techniques to a systems-management context, and making them work well together.
- *General techniques for autonomic systems.* Develop general algorithms, interfaces, and frameworks that support cooperative and coherent interactions among multiple autonomic components that are geared toward satisfying a common system-wide objective.

Resources

Readers interested in background literature on autonomic computing should check out these additional resources:

- For a book that presents a diverse set of contributions from leading experts in autonomic computing, see S. Hariri and M. Parashar, eds., *Autonomic Computing: Concepts, Infrastructure, and Applications*, CRC Press, 2007.
- For a book that combines many good contributions to the field of autonomic computing, see O. Babaoglu et al., eds., *Self-Star Properties in Complex Information Systems,* LNCS 3460, Springer-Verlag, 2005.
- You can find recent developments in autonomic computing in the *Proceedings of the IEEE International Conference on Autonomic Computing* (ICAC; 2004, 2005, and 2006); www.autonomic-conference.org.
- Information and links to many current autonomic computing projects and resources are available at www.autonomiccomputing.org.
- For IBM's activities on autonomic computing, visit www.ibm.com/autonomic and www.research.ibm.com/autonomic.
- HP's activities on its adaptive enterprise project are available at www.hpl.hp.com/research/ssrc/services/adaptive/.
- Microsoft's self-tuning and self-administering databases activities can be found at http://research.microsoft.com/dmx/autoadmin/.
- For Motorola's activities on autonomic networking, visit www.motorola.com/content.jsp?globalObjectId=6652-9277.
- A good collection of recent articles on autonomic computing is available in Intel's *Technology Journal*, "Special Issue on Autonomic Computing," vol. 10, no. 4, 2006; www.intel.com/technology/itj/.
- Specific journals on autonomic computing include *The Journal of Autonomic and Trusted Computing* (JoATC; www.joatc.org), *American Scientific Publishers*, and *ACM Transactions on Autonomous and Adaptive Systems* (TAAS; www.acm.org/pubs/taas).

- *Prototypes*. Build prototype autonomic computing systems to help understand the nature of the remaining gaps and determine which frameworks work best.

The articles in this issue, each of which exemplifies two or three of these themes, represent a small cross-section of efforts of this nature that are occurring worldwide.

"Reinforcement Learning in Autonomic Computing," by Gerald Tesauro, advocates reinforcement learning (RL) as a broadly applicable machine-learning technique for autonomic components and systems that avoids an important knowledge bottleneck: the need for human experts to create performance models for complex distributed systems. He argues on general principles that RL is an inherently powerful way to develop management policies that explicitly take into account long-range consequences of actions (which is difficult to achieve with expert-crafted policies). Tesauro presents a hybrid approach that allows RL to bootstrap from existing management policies, which substantially reduces learning time and costliness, and he demonstrates hybrid RL's effectiveness in the context of a simple data-center prototype.

In "Distributed Cooperative Control for Adaptive Performance Management" Mianyu Wang and his colleagues describe a general distributed cooperative control framework for managing performance and other management concerns in autonomic computing systems. The framework, which is based on concepts from optimal control theory, achieves scalability by decomposing an overall management objective into subproblems that individual controllers solve locally, and by requiring the controllers to share a minimal amount of information about the global state and environmental variables. With a simulation that employs real Web site data traces, the authors demonstrate that, by using their control scheme to manipulate processor operating frequencies, they can save substantial amounts of power while still meeting performance objectives.

Finally, "Achieving Self-Management via Utility Functions" by Jeffrey Kephart and Rajarshi Das advocates utility functions[3] as a general paradigm for representing high-level objectives and describes general mechanisms involving modeling and optimization that enable autonomic components to work together cooperatively and coherently to manage autonomic systems toward those objectives. After discussing utility-function policies' relative advantages over more traditional action policies on general principles, they illustrate their approach via a prototype that employs two commercial products into which they've infused their utility-based framework: a workload manager and a provisioner. The authors show how utility information can be transformed and propagated through the system to drive server-

allocation decisions from business value, and relate their experiences with the often tricky road to commercialization.

The theme articles in this issue of *IEEE Internet Computing* illustrate the general flavor of the research that's needed for us to move more quickly toward the ultimate autonomic computing vision; they present general techniques for either autonomic components or autonomic systems (or both). However, they represent only a small proportion of similar efforts to develop and apply other general mechanisms for autonomic components and systems and a very thin slice of the worldwide autonomic computing research effort.

We hope these articles will inspire readers to adopt these ideas and apply them in different domains, or to develop new general mechanisms of their own, perhaps adapted from planning, optimization, machine learning, software agents, or other well-established computer science subdisciplines. Once we begin to see medium- to large-scale prototypes that employ these general mechanisms and exhibit multiple self-* capabilities, we'll know that we've taken a significant step closer to attaining fully autonomic computing systems.

References

1. J.O. Kephart and D. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, 2003, pp. 41–50.
2. M. Bennani and D.A. Menascé, "Resource Allocation for Autonomic Data Centers Using Analytic Performance Models," *Proc. 2nd IEEE Int'l Conf. Autonomic Computing* (ICAC 05), IEEE CS Press, 2005, pp. 229–240.
3. W.E. Walsh et al., "Utility Functions in Autonomic Computing," *Proc. 1st Int'l Conf. Autonomic Computing* (ICAC 04), IEEE CS Press, 2004, pp. 70–77.
4. Y. Diao et al., "Using MIMO Feedback Control to Enforce Policies for Interrelated Metrics with Application to the Apache Server," *Proc. IEEE/IFIP Network Operations and Management Symp.*, IEEE CS Press, 2002, pp. 219–234.
5. D.A. Menascé, R. Dodge, and D. Barbara, "Preserving QoS of E-Commerce Sites through Self-Tuning: A Performance Model Approach," *Proc. 2001 ACM Conf. E-Commerce*, ACM Press, 2001.
6. J.O. Kephart, "Research Challenges of Autonomic Computing," *Proc. Int'l Conf. Software Eng.*, ACM Press, 2005, pp. 15–22.

Daniel A. Menascé is a professor of computer science and the associate dean for research and graduate studies at the Volgenau School of Information Technology and Engineering at George Mason University. He has a PhD in computer science from UCLA. Menascé is a fellow of the ACM, a senior member of the IEEE, and a recipient of the 2001 A.A. Michelson Award from the Computer Measurement Group. Contact him at menasce@cs.gmu.edu.

Jeffrey O. Kephart manages the Agents and Emergent Phenomena group at the IBM T.J Watson Research Center in New York. He and his team have applied ideas inspired by biology and economics to various large-scale distributed computing systems, including computer virus epidemiology and immunology, economic software agents, and autonomic computing. Kephart has a PhD in electrical engineering from Stanford University. He was a finalist in the 1997 *Discover Magazine* Awards for Technological Innovation. Contact him at kephart@us.ibm.com.