

Expanding the horizons of software engineering education: integrating autonomic computing into the curriculum

Gilda Pour

San José State University
San José, United States of America

ABSTRACT: Increasing size and complexity of software systems, rising cost of administering complex systems, lack of sufficient supply of trained system administrators, numerous system failures and outages caused by human errors, and major difficulties in dealing with changes that affect development, integration, deployment and management of complex systems make it impossible to rely on human intervention and administration. This has motivated the development of a new paradigm of autonomic computing. Inspired by the human body's self-regulatory nervous system, autonomic computing systems will be self-aware and able to self-manage (ie self-configure, self-optimise, self-heal and self-protect). It is envisioned that autonomic computing systems monitor the environment and resources to be managed, analyse the collected information, and use it to develop and execute plans to manage the system and the resources. Autonomic computing is crucial to the success of pervasive computing – the emerging next-generation computing. This calls for software engineering education reform to integrate autonomic computing into the curriculum. In this article, the author presents a set of key strategies for such an education reform.

INTRODUCTION AND MOTIVATION

Increasing the size and complexity of software systems, rising cost of administering complex systems, lack of sufficient supply of trained system administrators, numerous system failures and outages caused by human errors, and major difficulties in dealing with changes that affect the development, integration, deployment and management of complex systems make it impossible to rely on human intervention and administration.

In order to address this key challenge, IBM Senior Vice President and Director of Research, Paul Horn, introduced the vision for the new paradigm of autonomic computing in his keynote speech at the National Academy of Engineering Conference, held at Harvard University in March 2001. He stated:

> [autonomic computing systems are] *computer systems that regulate themselves much in the same way our autonomic nervous system regulates and protects our bodies.*

Autonomic computing has been derived from the human body's autonomic nervous system, which governs human body's involuntary vital functions, such as breathing, respiration, heart beat (rate) and body temperature. Inspired by the human body's self-regulatory nervous system, an autonomic system is self-aware and able to self-manage.

> *Our bodies have great availability. I have soft errors all the time: my memory fails once in a while, but I don't* crash. *My whole body doesn't shut down when I cut a finger.*

This analogy was made by IBM Vice President of Personal Systems and Storage and Director of IBM's Almaden Research Centre, Robert Morris.

The new paradigm of autonomic computing shifts the fundamental definition of the technology age from one of computing to another defined by data. Access to data from multiple, distributed sources in addition to traditional centralised storage devices will allow users to transparently access information when and where they need it. Autonomic computing requires the industry to change its focus on processing speed and storage to developing distributed networks, which are mainly self-managing, self-diagnostic and transparent to the user [1].

Autonomic computing is crucial to the success of pervasive computing – the emerging next generation of computing. Autonomic computing systems are in high demand in a wide variety of mission critical systems in important application domains, such as healthcare, elderly care, defence, homeland security, space and planetary exploration, air traffic control, transportation, finance, e-commerce, e-business, manufacturing and many others that affect safety and security.

INTEGRATING AUTONOMIC COMPUTING INTO THE SOFTWARE ENGINEERING CURRICULUM

It is critically important to integrate the new paradigm of autonomic computing into software engineering education so that software engineers will have a good understanding of autonomic computing, as well as the technical and professional skills and experience to deal with the associated challenges and opportunities.

The suggested key strategies to integrate autonomic computing into software engineering education include the following:

- Redesigning the software engineering curriculum to incorporate autonomic computing into the curriculum;
- Systematically integrating autonomic computing research into education;

- Providing students with applied and experimental research opportunities;
- Cultivating industry-academic partnerships in research and education;
- Providing institutional support for multidisciplinary collaborations in research and education;
- Fostering life-long learning;
- Systematically updating the contents and structure of software engineering curricula.

Table 1 presents the list of suggested key topics for incorporating autonomic computing into the software engineering curriculum and redesigning the curriculum.

Due to a rapidly evolving and highly diversified area of autonomic computing, it is critically important to systematically integrate research into education in order to increase the effectiveness of students' learning experiences. Engaging students in applied and experimental research provides major educational opportunities for software engineering students to acquire invaluable experience that cannot be gained by attending lectures and reading technical articles [2-7].

Thus, it is crucial to provide students with opportunities to experience the activities required for developing innovative engineering solutions and building demonstration prototypes. Such a beneficial experience helps students become familiar with and appreciate applied and experimental research that is behind research papers that present software prototype development and case studies.

In order to significantly enhance the opportunities for students to become familiar with software engineering practice, it is necessary to cultivate industry-academic partnerships in research and education. This will offer benefits to students by providing industrial experience with project sponsors and enhance students' hands-on experiences, as well as their technical competences and skills.

Furthermore, the multidisciplinary nature of autonomic computing requires multidisciplinary collaborations in both research and educational activities, which are necessary for success in developing autonomic systems. Therefore, it is required that academic institutions foster multidisciplinary collaborations in research and education to allow students, faculty and other professionals across various fields to engage in collaborative, multidisciplinary projects. This will also help students learn and enhance their engineering knowledge and skills, as well as their professional skills (eg teamwork, written and verbal communications, etc).

On the one hand, collaborative multidisciplinary projects require additional time and effort to ensure productive cooperation among those involved. However, funding structures in academia traditionally do not provide necessary institutional support for multidisciplinary collaborations, particularly when cooperation among different departments and colleges is needed.

Thus, it is crucial that academic institutions make necessary changes to their funding structures and faculty evaluation criteria so as to provide institutional support and funds for such multidisciplinary collaborations. Cultural changes in this regard are also needed in academia.

The nature of software engineering also requires software engineers to be truly life-long learners and keep their technical knowledge, competences and skills current throughout their careers. Autonomic computing is essential to the success of pervasive computing. Thus, life-long learning becomes even more crucial in a pervasive computing world.

To help graduates become self-motivated and life-long learners, it is critical to provide students with opportunities to acquire both the awareness of the importance of life-long learning and the knowledge, skills and abilities to engage in life-long learning.

Last, but not least, it is critically important to make the curriculum flexible and responsive to change, and also to systematically update the contents and structure of the software engineering curriculum to ensure it provides the most effective learning opportunities for software engineering students in the key area of software engineering for autonomic computing.

CONCLUDING REMARKS

The vision for autonomic computing cannot be implemented if software engineering education is not reformed to properly prepare graduates of software engineering programmes for autonomic computing. In this paper, the author discusses the necessity of integrating autonomic computing into software engineering curricula and presents a set of suggested key strategies for integrating autonomic computing into software engineering education.

The suggested strategies include redesigning software engineering curriculum to incorporate autonomic computing into the curriculum; systematically integrating autonomic computing research into education; providing students with applied and experimental research opportunities; cultivating industry-academic partnerships in research and education; providing institutional support for multidisciplinary collaborations in research and education; fostering life-long learning; and systematically updating the contents and structure of software engineering curricula to better prepare students for a challenging career in software engineering.

REFERENCES

1. Pour, G., Pervasive computing reforming software engineering education. *World Trans. on Engng. and Technology Educ.*, 2, **3**, 357-360 (2003).
2. Pour, G., Agent-Oriented Software Engineering (AOSE): its emergence as a cornerstone of enterprise software engineering education. *World Trans. on Engng. and Technology Educ.*, 2, **2**, 225-228 (2003).
3. Pour, G., Component-Based development refining the blueprint of software engineering education. *World Trans. on Engng. and Technology Educ.*, 2, **1,** 45-48 (2003).
4. Pour, G., Engineering curriculum reform to introduce students to security and privacy in the Internet era. *World Trans. on Engng. and Technology Educ.*, 4, **2,** 285-288 (2005).
5. Pour, G., Multi-agent autonomic architectures for quality control systems. *Pervasive Computing* (2006).
6. Pour, G., Exploring multi-agent autonomic architectures for telehealth systems. *Telehealth* (2006).
7. Pour, G., Exploring multi-agent architectures for autonomic systems. *Pervasive Computing* (2005).

8. Kephart, J. and Chess, D., The vision of autonomic computing. *IEEE Computer*, 36, **1**, 41-50 (2003).
9. Waldrop, M., Autonomic Computing: the Technology of Self-Management (2003), http://www.thefutureofcomputing.org/Autonom2.pdf
10. Carnegie Mellon University, Self-Securing Storage, http://www.pdl.cmu.edu/Secure/S4.html
11. Carnegie Mellon University, Self-Securing Devices, http://www.pdl.cmu.edu/Secure/S4.html
12. Berkeley University, OceanStore, http://oceanstore.cs.berkeley.edu
13. Berkeley University, Recovery-Oriented Computing, http://roc.cs.berkeley.edu
14. Cornell University, Astrolabe, http://www.cs.cornell.edu/ken/Astrolabe.pdf
15. Georgia Institute of Technology, Qfabric, http://wwwstatic.cc.gatech.edu/systems/projects/ELinux/
16. IBM Research Projects, http://www.research.ibm.com/autonomic/research/projects.html#ibmresearch

Table 1: The key topics to be incorporated into the software engineering curriculum.

Introduction & Fundamental Concepts of Autonomic Computing	It is envisioned that an autonomic computing system: • Monitors the environment and resources to be managed, analyses the collected information, and uses it to develop and execute plans to manage the system and resources; • Has the ability to manage and dynamically adapt itself to changes in accordance with the policies and standards in an application domain, and it can perform such activities based on the situations that the system observes or senses in its environment. The computing paradigm will change from one based on computational power to one driven by data. The way computing performance is measured will change from processor speed to the immediacy of the response. Individual computers will become less important than more granular and dispersed computing attributes. The economics of computing will evolve to better reflect actual usage – what IBM calls e-sourcing [1]. Autonomic computing requires development of computer systems, software, storage and support to exhibit: • Accessibility: System's availability to provide service as required by the nature of the autonomic system; • Flexibility: System's ability to filter and examine data via a platform- and device-agnostic approach; • Transparency: System's ability to perform tasks and adapt to a user's needs while shielding the user from the complexity of the system.
Self-Awareness in Autonomic Computing	System's ability to know its environment and the context surrounding its activity, and act accordingly. This requires the system to find and generate rules for its interaction with other systems (eg negotiating with other systems about the use of resources) and adapting to its environment [1][8-12].
Self-Management in Autonomic Computing	S*elf-Configuration*: System's ability to automatically and seamlessly adapt itself to dynamically changing environments and conditions by changing its own configuration (ie configuring and reconfiguring its components based on high-level policies, under varying conditions and without service disruption).
	Self-Optimisation: System's ability to monitor its state and performance; proactively adjust itself to improve its performance and maximise the use of resources to ensure predetermined system goals are achieved.
	Self-Healing: System's ability to automatically and continuously detect, diagnose, act/react to prevent any disruption, and recover from errors, damages and routine and extraordinary events that may cause some parts of the system to malfunction, or the system to fail.
	Self-Protecting: System's ability to automatically and continuously detect, identify, protect itself against various types of attacks and prevent system-wide failures to maintain overall system security and integrity.
Autonomic Computing Systems [1][8-10]	Autonomic architectures [1][8-11].
	Autonomic servers.
	Autonomic networks: • Completely transparent to users; • Entirely heterogeneous environment, platform agnostic and dependent on open standards.
	Autonomic intelligent middleware: • The core of autonomic computing system; • Must support all modes of access to it (eg cell phones, PDAs, Web-connected appliances, embedded devices, PCs); • Is intended to support transparent access to the autonomic system. The servers that make up this system will be more cellular and distributed, similar to the human nervous system. There will be systems and networks of single-chip cells that integrate processors, memory and communications. These cellular architectures will make it feasible for the computer to get its power out to where the data is, thereby eliminating some of the latency characteristics of current architectures.
	Autonomic storage systems.
	Autonomic federated systems.
	Autonomic capabilities in access devices.
	Autonomising legacy systems.

Open Standards & Technologies for Autonomic Computing	Open standards and new technologies are necessary for autonomic systems to interact effectively, to enact pre-determined business policies more effectively, and to be able to protect themselves and *heal* themselves with a minimal dependence on traditional IT support [1][8][9].
The Challenges of Autonomic Computing	Multidisciplinary collaborations: • Developing autonomic computing systems is overwhelmingly difficult as it requires extensive multidisciplinary collaborations; • Collaborations among not only researchers from multiple technical and scientific disciplines, but also different companies and research and academic institutions to share a sense of urgency and purpose are required [1][8][9].
	Development of open standards and new technologies for autonomic computing
	Research challenges of autonomic computing: • Development; • Integration; • Deployment; • Management.
	Emerging new business models
Case Studies and Examples	• Carnegie Mellon University: Self-Securing Storage [13]; • Carnegie Mellon University: Self-Securing Devices [14]; • NASA, SJSU: Multi-Agent Autonomic Architectures [8-10]; (co-sponsors: IBM, HP): • Berkeley University: OceanStore [15]; • Berkeley University: Recovery-Oriented Computing [16]; • Cornell University: Astrolabe [17]; • Georgia Institute of Technology: Qfabric [18]; • IBM Autonomic Computing Research Projects [19].