50 & 25 YEARS AGO

EDITOR **ERICH NEUHOLD**
University of Vienna
erich.neuhold@univie.ac.at

FEBRUARY 1969

In the early years, *Computer* was published only bimonthly. Therefore, we will have to skip our interesting and informative extractions for this month. The next column will appear in the March issue of *Computer*, and we hope you will eagerly wait for its publication.

FEBRUARY 1994

Is 1994 1984 All Over Again? (p. 8) "This year, we face the prospect of yet another profound development. Instead of micro, it is a new era of multimedia telecommunications—the convergence of micros with communications networks. ... 1994 marks the beginning of a roller-coaster ride into yet another era of computing. We at *Computer Magazine* plan to be at the forefront of this activity with articles and columns that aim squarely at the core of these technologies. Our new "Computing Milieu" section tackles the sociological and philosophical underpinnings of this new era, our "Computing Practices" section illustrates how to apply the new technologies, and our regular research articles give you a look at what is to come."

Exploiting the Parallelism Available in Loops (p. 13) "An efficient approach for extracting this potential parallelism is to concentrate on the parallelism available in loops. Since the body of a loop may be executed many times, loops often comprise a large portion of a program's parallelism. A variety of parallel computer architectures and compilation techniques have been proposed to exploit this parallelism at different granularities." (p. 14) "We're interested in parallelism limitations due to data dependences, since, at least in theory, we could eliminate resource dependences with additional hardware. For simplicity, we'll limit the analysis to singly nested loops in which the loop index has been normalized to vary from 1 to *n* with a stride (increment between iterations) of 1." (p. 16) "Fine-grained parallel architectures exploit loop parallelism at the instruction set level by issuing several instructions or operations in a single cycle. At runtime, dependences between operations must be checked either statically by the compiler or dynamically by the hardware to ensure that only independent operations are issued simultaneously." (p. 19) "While fine-grained parallel architectures exploit parallelism at the instruction level, coarse-grained architectures exploit it by scheduling entire iterations on separate processors. On the shared-memory multiprocessor shown in Figure 10, for example, the independent tasks to be scheduled are the individual instantiations of the iterations, each with a unique value of the loop index." *(Editor's note: The article evaluates various parallelization strategies for both coarse-grained and fine-grained approaches. However, it restricts its analysis to situations that appear in scientific programming, for example, matrix-based situations. In today's situations, statistical analysis of large amounts of data—for example, for information mining—are important parts of parallelization; therefore, novel distribution techniques had to be found.)*

Defining Requirements for a Standard Simulation Environment (p. 28): "In this article, we define a reference model for general-purpose discrete-event simulation environments as well as the associated requirements for the model's functional layers, to be used as the basis for a future standard." (p. 30) "Reference model ... The model consists of five distinct layers. The top layer, or application layer 4, can access all layers so that developers can add application-specific constructs to their environments. The lower layers include properties that enable implementation of similar features at higher levels. The lowest layer 0 provides the basic language-level support for the environment and can be accessed by all other layers. Layer 1 defines the requirements for model specification. Layer 2 deals with model knowledge management. Layer 3 is the system design layer." *(Editor's note: The article proposes a reference architecture for simulation systems that, despite looking attractive, never managed to become* the *reference model in simulation. Simulink from MathWorks is widely used, but other approaches still play important roles.)*

Digital Object Identifier 10.1109/MC.2019.2897920
Date of publication: 22 March 2019

A Methodology for Design, Test, and Evaluation of Real-Time Systems (p. 35): "This methodology can reduce project costs and shorten schedules because it requires performance evaluation and integration testing early, when problems are generally easier and less costly to correct. ... This article presents a methodology that is suitable for use as part of either a prototyping approach or a component-reuse approach. This methodology integrates modeling and simulation as well as developmental and operational testing over the life cycle. The type of systems or components we address operate in real time." (p. 36) "This methodology consists of an analytical approach for representing (1) a system and its environment, and (2) a hardware and software architecture. Both the methodology and the modeling-and-simulation/test-and-evaluation suite based on this architecture are called ASSET (an acronym for aerospace systems simulation, engineering, and test tool)." *(Editor's note: The publication analyzes in detail a design/simulation/implementation approach for an airborne interferometer processing system.)*

Ada System Dependency Analyzer Tool (p. 49): "Strongly typed languages like Ada promote significant extensibility and reusability because they support safe, error-free interfaces between different software packages. At the same time, this ability to embed diverse systems within an application, particularly commercial off-the-shelf (COTS) software, often unintentionally adds to architectural complexity. ... With large, complex software systems, automated tools are indispensable for identifying the architectural components, the structure that interconnects them, and other subtle dependencies. This article describes the construction of an Ada System Dependency Analyzer (SDA), a software architecture analysis tool that generates a quantitative snapshot of an Ada application's software architecture. The SDA can process thousands of Ada source files during a single run and report on them as a group of files comprising a single Ada system." (p. 54) "Our object-oriented design method provides the additional benefit of letting us add new features to the SDA in a straightforward manner. For example, to add a new capability, a handle routine is added to the parser, providing a point of call for the new construct when it is detected in the grammar." *(Editor's note: The authors claim that the approach can easily be adopted to other high-level programming environments but do not indicate how this could be done, which is especially interesting in light of the ADA idiosyncrasies mentioned in the next article.)*

Implementing a Software Configuration Management Environment (p. 56) "Configuration management is concerned with maintaining a product's integrity. To do so, a successful CM environment requires three basic capabilities: (1) version control, (2) a check-out/check-in facility, and (3) a buffered-compare program." (p. 59) "There are a few other items to consider when developing a software CM environment: (1) establishing a standards-checking program, (2) implementing an automated problem-reporting system, (3) automating the generation of configuration status accounting reports, and (4) providing an automated metrics acquisition and reporting capability." *(Editor's note: The article develops and describes a software configuration management system not so different from others existing in 1994, but it also relates it to the ADA environment developed in the previous article.)*

Computer Science for the Many (p. 62) "Traditional literacy courses were developed some years ago and are still taught at many universities throughout the world. These courses emphasize learning the vocabulary of computing; gaining some experience with software packages, such as word processing, spreadsheets, and database systems; and studying the history and social impact of computing. ... Unfortunately, literacy courses address the syntax and form of the field rather than the structure and ideas. They enable students to use machines, but they do not engage their intellects in the real excitement of computing. Memorized vocabulary will not last unless tied to a real understanding, and the applications packages students learn will soon be out of date." (p. 63) "The philosophy is that if students develop sufficient knowledge of what computers can do, and learn how to get started doing those things, they will have gained knowledge and skills of lasting value. The course teaches this material by introducing the students to programming and by teaching them the fundamental mechanisms of computer hardware and software." (p. 69) "Artificial intelligence. This topic divides into two parts, knowledge representation and reasoning. The knowledge representation lectures introduce several common representation schemes. ... The reasoning lectures show several search methodologies and applications of problem solving. ...

"In concluding the artificial intelligence study, instructors try to characterize the field's current level of success and warn against over optimism about the future. Students are told that almost any intelligent phenomenon—learning, problem solving, creativity, natural language processing, vision, and so forth—can be simulated to a very limited extent. However, no artificially intelligent phenomenon of this nature has been exhibited to a great degree, nor is this likely in the foreseeable future." *(Editor's note: I consider this curriculum proposal to be well thought out and well argued. Not knowing the U.S. scene for such basic computer courses in academia, I am somewhat surprised that the subjects covered apparently are not contained in such courses. In Germany and Austria, such content was and is more or less standard.)*

Former IBM Chief T.J. Watson, Jr., dies (p. 84) "In 1977, when he received an honorary Doctor of Civil Law degree from Oxford University, Watson summed up concerns that had occupied him in both his business and his public service roles: 'One of the most important problems we face today, as techniques and systems become more and more pervasive, is

the risk of missing that fine, human point that may well make the difference between success and failure, fair and unfair, right and wrong. … No IBM computer has an education in the humanities. … There isn't a single one with moral standards, conscience, soul, or heart. … I say that for every step forward in the direction of more so-called scientific management, we must take one or more steps toward improved preservation of human values." *(Editor's note: A statement I fully subscribe to. However, more than 40 years later, I sometimes have the strong impression that we have forgotten this fact and trust computers, clouds, big data, and deep learning more than our own human values, intelligence, and ethics.)*

The Open Channel: Natural Talent for Computing (p. 120) "Two extremes. Today, I am teaching in the laboratory again. I'm moving between people who are convinced that they know everything about computers and people who are convinced that they will never know anything about computers. Somewhere in here are future computer science stars, but they're not so easy to spot anymore. I'm thinking, maybe we need fewer gun nuts and fewer computer nuts, and more people who can hit what they're aiming at. I'm beginning to believe that natural talent is a matter of a fresh, patient, and open mind." *(Editor's note: What a true statement and one that is often completely disregarded by our "modern" recruiting methods.)*

Digital Object Identifier 10.1109/MC.2019.2901911