50 & 25 YEARS AGO

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

NOVEMBER/DECEMBER 1971

www.computer.org/csdl/mags/co/1971/06/index.html

Distributed Intelligence in Terminal Systems; Robert V. Dickinson (p. 17) "One trend that became clearly visible during the course of the workshop is the increasing tendency to distribute intelligence throughout a terminal system rather than concentrating it at its center. This can be done through the use of small stored program processors as front end communication processors and remote concentrators and the use of stored program controllers within the terminals themselves. ... Although there is not total agreement that the distribution of intelligence is the best system design solution, it is clear that the next several years will see many systems based on this philosophy." *[Editor's note: "Terminals," of course, came to include more and more processing power, but they never became part of a really distributed system. Instead, they were replaced by minicomputers as parts of networks.]*

An Outlook for the Terminal Industry in the United States; Roy M. Salzman (p. 18) "Many vendors in the terminal industry believe—not too illogically—that their route to success lies in finding out from the user community what kind of terminal characteristics they desire for their projected applications and then simply building a terminal to suit the most prevalent of those needs." (p. 21) "Basic input to a terminal has been and probably always will be primarily through a manual keyboard of some form. ... *Visual display methods* are rapidly changing to give users more capacity, economy, and esthetic appeal. The use of television technology for digital display has been an important advance which could well be refined further to provide a good display capability at an extremely low cost." (p. 25) "While up to now, we have been able to speak of terminals as teletypes, CRT's, remote-batch devices, etc., and generally are able to envision what such a device consists of, we expect that by 1975, this picture will become a great deal more fuzzy and distinct terminal types may not be recognizable as such." *[Editor's note: The article foresees the wealth of different input and output devices that would appear in later years but of course misses some that are now common in, for example, smartphones such as cameras.]*

Distributed Intelligence in Data Communications Networks; Stanford R. Amstutz (p. 26) "Today, however, it *is* desirable to distribute the power of large, fast, and expensive computers to many terminals, but the variety of installed terminals makes their administration difficult." (p. 28) "Figure 2 illustrates a transformation of the Figure 1 network by the addition of programmable processors, represented by squares, at three different levels in the network; *Level 1*—the central sites, *Level 2*—remote sites between the central sites and the terminals, and *Level 3*—at the terminals." (p. 32) "The kinds of communication functions which can be performed by a communications processor have already been discussed above in the central site context. ... We have discussed the particular ways in which a minicomputer can be useful in reducing the operating costs of a computer/communications network, by use at central sites and at remote sites." *[Editor's note: It is interesting to note that this lengthy article discusses quite a number of ways to interconnect computers and terminals into networks but does not explain/utilize the bus concepts that were rather new in 1971.]*

The Rationale for Smart Terminals; L.C. Hobbs (p. 33) "Viewed from a strictly functional standpoint, there is no justification for centralizing the computing and processing functions, but frequently there is a strong functional reason for centralizing the data base. In general, the question of whether a particular computation or processing operation is carried out in the local terminal or in a large central computer system is an economic one." (p. 35) "Future trends: These technology trends can also reasonably be expected to provide a larger $50,000 to $200,000 smart terminal including: a 32,000 to 64,000 word minicomputer, a 9600 baud modem, a keyboard, a character serial printer or other hard copy device,

Digital Object Identifier 10.1109/MC.2021.3055916
Date of current version: 22 October 2021

0018-9162/21©2021IEEE

a magnetic tape cassette, a graphic cathode ray tube display, a light pen." *[Editor's note: Just think of it and compare it to today's networks and computation powers.]*

Database Management in a Multi-Access Environment; Arthur J. Collmeyer (p. 36) "Viewed from a strictly functional standpoint, there is no justification for centralizing the computing and processing functions, but frequently there is a strong functional reason for centralizing the data base." (p. 37) "The basic elements of a Database Management System are described in Figure 1. Two logically distinct user interfaces are provided. *The first is database definition.* This interface, defined implicitly in the Database Definition Language (DDL), enables the creation of a (dataless) database. *The second user interface is the interface to an existing database.* This interface provides the means for the manipulation of data ... the Database Manager (DBM)." *[Editor's note: This article continues to describe the various concepts that have to be considered in a multiuser database environment, mostly using its own terminology. It is interesting to note that the article refers only once to the CODASYL database standard and not at all to the papers on relational databases that had been published, staring with Ted Codd's Association for Computing Machinery paper of June 1970.]*

NOVEMBER 1996

www.computer.org/csdl/mags/co/1996/11/index.html

Survivability in the Age of Vulnerable Systems; Mario Barbacci (p. 8) "As we all become more connected, system survivability, an issue that used to concern mostly business or governments is now routinely covered in the mainstream press. Survivability is defined as a system's capacity to complete its mission in a timely matter, even if significant portions are incapacitated by attack or accident. ... For example, Microsoft plans to integrate the multimedia capabilities of the World Wide Web with its Windows 95 operating system. Microsoft's new paradigm will abandon files and folders kept in local storage in favor of stand-alone Web pages. Every document, everywhere, would potentially be accessible through hypertext links. ... The good news is that there is a great deal of research on issues related to survivability. The bad news is that different researchers don't hear from each other. This is a major problem, because the concepts and practices associated with system survivability span almost the entire range of computer science and engineering." *[Editor's note: As we know, all this research did not eliminate the vulnerability of our systems and networks. It actually looks like the frequency of wide-ranging, serious attacks—denial of service, identity theft, extortion locking, social media distortion, and so on—is rising, not falling.]*

System Test and Reliability: Techniques for Avoiding Failure; Rohit Kapur et al. (p. 28) "Digital systems are a combination of hardware and software. Errors in either of these components could cause a failure. Though software and hardware are very different from each other, the basic concepts (viewed at a higher level of abstraction) used to test failures and tolerate the errors are the same." (p. 29) "Testing for failures ... If the output does not match the expected response, the component is declared faulty and discarded or repaired. ... Tolerating failures ... Once a failure has been detected, the error must be masked out. This is usually performed by ensuring that the redundant information outweighs the error." *[Editor's note: The articles following this guest editor's introduction analyze different aspects of handling errors in IT systems. As we know, of course, in the intervening years, progress has been made, but errors in IT systems exist and even enable attacks that damage system functions beyond what an error itself can cause. Apparently, in 1996, this was not an issue of very high concern, as none of the following articles mention it.]*

Testing ICs: Getting to the Core of the Problem; Brian T. Murray et al. (p. 32) "This tutorial examines the market and technology trends affecting the testing of integrated circuits with emphasis on the role of predesigned components—cores and built-in self-test. ... Here we explain manufacturing testing, as opposed to design testing, which happens before manufacturing, and on-line testing, which happens after." (p. 37) "Testing is a major contributor to the cost of manufacturing and maintaining digital ICs. Well-developed fault models and test generation methods for such circuits are known, and are widely supported by design tools. However, their applicability to today's increasingly fast and complex circuits is limited by practical cost considerations. Design-for-test techniques, especially scan design and built-in self-test, can provide a satisfactory solution in many instances." *[Editor's note: This article provides an interesting analysis of the challenges faced with the rapid increase in IC size and complexity.]*

Built-In Self-Test: Assuring System Integrity; Bernd Konemann et al. (p. 39) "Today's complex electronic products are harder to test using traditional external methods. BIST can frequently be used without significantly increasing a product's size, cost, and production time." (p. 40) "Now, semiconductor technology lets you implement comprehensive chip-level BIST features for very little additional circuitry-related cost, while hardware synthesis technology for BIST integration has caused design related costs to drop. ... Economy dictates that the BIST-related stimulus-generation and response processing functions that we encapsulate into product components be very compact." (p. 43) "Complex electronic products must be reliable, available, and serviceable in the field. Consequently, many products contain extensive hardware test and diagnostic support functions that can be executed in the field. The development of higher level hardware diagnostics is greatly simplified by encapsulating comprehensive tests into each chip." *[Editor's note: In this article, various basic self-test principles are explained. Today's complex systems would not work reliably if self-testing was not included from the early design stages on.]*

Multiprocessor Validation of the Pentium Pro; Deborah T. Marr et al. (p. 47) "Validation was even more challenging because Intel wanted to deliver a production system within a year of first silicon. Presilicon validation was crucial because it let Intel detect and fix problems before they made it into silicon. It is easier to isolate problems in a simulation model, where we can control events and also 'look' inside the processor and chipset, than on silicon." (p. 50) "We used our presilicon RTL test methodology on our postsilicon test platforms. These platforms had no operating system and consisted of the processors and the chipset on a board with supporting ASIC chip. ... We discovered and quickly fixed a few complex problems that had been difficult to hit in the RTL simulation mode." *[Editor's note: This article provides an excellent example of the many difficulties encountered when a real-world processor has to be tested and the many obstacles that have to be conquered.]*

Safety-Critical Systems Built With COTS; Joseph A. Profeta III et al. (p. 54) "At the same time, competitive pressure has led to the increased use of COTS (commercial, off-the-shelf) equipment in safety-critical systems, making it imperative that we extend proven safety techniques to COTS-based systems as well. ... The key technologies in this framework are formal methods, information redundancy, a proprietary data format, and a concurrent checking scheme." (p. 57) "There are three distinct pieces of software that must be considered in evaluating and proving the correctness of program execution: the application code, the compiler, and the runtime kernel. ... The proof-of-correctness mentioned above does not cover the graphical compiler because it is difficult, if not impossible, to quantify the compiler's safety as we can the application's. Instead, we apply formal methods to prove the correctness of the graphical compiler invocations." *[Editor's note: The method employed here combines, in an interesting way, formal techniques and others, such as built-in redundancy and logging.]*

Software-Reliability-Engineered Testing; John D. Musa (p. 61) "The standard definition for software reliability is the probability of execution without failure for some specified interval, called the mission time. This definition is compatible with that used for hardware reliability, though the failure mechanisms may differ. In fact, SRET is generally compatible with hardware reliability technology and practice." (p. 64) "Engineer reliability strategies. There are three principal reliability strategies: fault prevention, fault removal, and fault tolerance." *[Editor's note: This is an interesting article that covers reliability issues encountered in telephone systems. In these extremely distributed systems, many errors (racing conflicts, denial of service, and so on) play a role that goes beyond software and hardware faults. These issues are even more important today, and therefore this article is worth reading.]*

Predicting Software Reliability; Alan Wood (p. 69) "We collected defect occurrence times during system test and statistically correlated the test data with known mathematical functions, called software reliability growth models. If the correlation is good, then the function can be used to predict future failure rates, or the number of residual defects in the code. We found that the correlation with a simple exponential model was good and that this model can reasonably predict the number of residual defects in our delivered software." *[Editor's note: The article analyzes quite a number of different models based the actual failures encountered in a sequence of releases of Tandem software. The in-depth discussion of the models makes the article worthwhile reading even today.]*

Measuring Software Quality: A Case Study; Thomas Drake (p. 78) "To ensure cost-effective delivery of high-quality software, NSA has analyzed effective quality measures applied to a sample code base of 25 million lines. This case study dramatically illustrates the benefits of code-level measurement activities." (p. 79) "We use two primary measurement activities to derive our code-level release criteria. The first is code metrics analysis: We measure development productivity indicators, predictability measures, maintainability indicators, essential quality attributes of the code, and 'hot spots,' and we identify overly complex modules that need additional work. ... The second measurement activity is coverage analysis, which centers on 'inside-the-code' analysis (or decision-level metrics), testability indicators through executable path analysis, and predictive performance analysis based on the number of segments per path." *[Editor's note: This is an interesting article not only because it analyzes numerous methods for the two measurement activities it mentions but also because it shows that the reengineering of problematic code led to vast improvement in reliability and execution time.]*

Why Higher Education Needs an Advanced Internet; William H. Graves (p. 93) "They do not wish to lose the conversational and social aspects of learning, which allow for rich sensory cues and spontaneous give-and-take. Face-to-face, we switch tasks and modes of communication seamlessly, but today's computers and network services do not support an integrated, seamlessly rich palette of communication and application capable of supplanting proximity. ... We must also recognize the distinction between instruction aimed at learning particular skills and bodies of knowledge, and instruction supporting the residential under-graduate experience with its goals of socialization and learning how to learn." *[Editor's note: Despite this fact, the article proceeds to propose necessary technology for a "virtual" university as a solution to higher education. Our latest experience with virtuality (COVID-19) has shown that some (how much?) physical contact is necessary not only for higher education but in practically all environments for the proper functioning of society.]*