


50 & 25 YEARS AGO



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



MARCH 1973

<https://www.computer.org/csdl/mags/co/1973/03/index.html>

On the Passing of MOBIDIC-B; M.D. Abrams and R. Rosenthal (p. 10) "The history, design, hardware configuration, instruction set and operating system of the MOBIDIC-B used in the National Bureau of Standards (NBS) computer research facility is described. ... Two fundamental objectives were established early in the design phase: 1. extremely high reliability and ease of maintenance, and 2. portability. ... MOBIDIC (Mobile Digital Computer) was built into a standard Army van with its own power supply." [Editor's note: The article describes in detail the functions of the system, which has been phased out at the time of this writing. It is surprising how many functions found in much later designs were found already in MOBIDIC. However, note that a truck was needed to ensure the portability of the system!]

PMS: A Notation to Describe Computer Structures; Mario Barbacci et al. (p. 19) "The PMS [Editor's note: processor-memory-switch] notation was developed to describe the physical structure of computer systems in terms of a small number of elementary components for textbook (Bell and Newell, McGraw-Hill 1971). There are seven basic component types, each distinguished by the kinds of operations (function) it performs: Memory, Link, Control, Switch, Transducer, Data-Operation, and Processor." [Editor's note: The article is from a textbook to introduce computer architectures. The notation has not been used widely outside the educational environment of that time.]

ISP: A Notation to Describe a Computer's Instruction Sets; Mario Barbacci et al. (p. 22) "The ISP (for Instruction Set Processor) notation was developed for a text [Bel and Newell, McGraw-Hill 1971] to precisely describe the programming level of a computer in terms of its Memory, Instruction Format, Data Types, Data Operations, Interpreting a Specific

Instruction Set." [Editor's note: PMS together with ISP were developed together for a textbook, but, again, ISP has not found wide acceptance. Both articles are interesting from the point of view that they attempt to introduce a stringent/normative notation to describe the architecture and behavior of computers.]

Cache-Based Computer Systems; K.R. Kaplan et al. (p. 30) "This paper treats the design problems of a computer system employing a cache memory hierarchy. The arguments for use of a cache are reviewed, and the basic design parameters defined. New measures of cache performance are proposed, numerical results are given derived from an extensive data base, and examples of their application are discussed." (p. 32) "Cache Design Parameters: The size of a cache is generally a dominant factor in both performance and cost - the more blocks of information that can be stored locally, the more likely any desired information is to be found there. ... An important design parameter for cache systems is the manner in which processor writes to memory are handled. ... The size of a block can also be important; we shall see that for a given size of cache, certain block sizes are performance-optimum, and large departures from them degrade performance substantially." [Editor's note: The article provides an interesting and detailed analysis and discussions about the benefits of using caches to enhance computer performance. This has, of course, been settled long ago, but, 50 years ago, it was a hot topic.]

Attacking on the Flank; Gene Amdahl (p. 37) "Now why would Gene Amdahl decide to form a company whose product appears to be in direct competition with his former employer? Especially in December of 1970 during the heart of the electronics industry recession when even well-capitalized companies were going belly-up with alarming frequency." [Editor's note: This very interesting interview-like article explains Amdahl's rationale and approach for developing high-performance computers that would outpace IBM and Control Data Corporation. At that time, no computer was yet delivered, but, as we now know, the company was very successful but finally was acquired by Fujitsu in 1997.]

MARCH 1998

<https://www.computer.org/csdl/mags/co/1998/03/index.html>

Changing Database Market Hurts Major Vendors; John

Edwards (p. 10) "The four major vendors are facing a shrinking market for the relational databases they have traditionally sold. ... Early last year, many industry observers predicted a bright future for the major database vendors—IBM, Informix, Oracle, and Sybase. ... Many industry observers say the Internet's rising popularity has slowed sales of relational databases. ... The major vendors are taking steps to regain market share, such as offering hybrid object-relational databases." (p. 11) "Thus, a key for database companies, particularly the major ones, will be flexibility and adaptability." [Editor's note: As we now know, this flexibility and especially the move into service-oriented solutions have led to new success stories for the database field. However, of the four main vendors mentioned here, Informix and Sybase both were acquired, and new players, such as the Microsoft SQL server, came along.]

Mobile Wireless Internet Technology Faces Hurdles;

Tammy Parker (p. 12) "Even with improvements in technology, there is no killer app for mobile wireless Internet. ... To date, however, wireless data communications have been too slow and too expensive for most users to adopt. Now, companies are spending considerable sums of money on technological developments that would enable faster, less expensive, and more reliable wireless Internet access." (p. 13) "COSTS AND BENEFITS: ...For example, mobile wireless technology would be impractical for recreational Internet browsing or business transactions that are not time-sensitive." [Editor's note: The article then proceeds to discuss possible improvements in transmission speed, device performance, and price to make wireless Internet a success. However, in no way does it foresee the tremendously rapid worldwide acceptance of multimedia smart phone technology with its innumerable apps that, let us not forget, is only about 20 years old!]

Scripting: Higher-Level Programming for the 21st

Century; John K. Ousterhout (p. 23) "Increases in computer speed and changes in the application mix are making scripting languages more and more important for the applications of the future." (p. 24) "Scripting languages such as Perl,⁴ Python,⁵ Rexx,⁶ Tcl,⁷ Visual Basic, and the Unix shells represent a very different style of programming than do system programming languages. Scripting languages assume that a collection of useful components already exist in other languages. They are intended not for writing applications from scratch but rather for combining components." (p. 30) "Scripting languages represent a different set of trade-offs than system programming languages. They give up execution speed and strength of typing relative to system programming languages but provide significantly higher programmer productivity and

software reuse." [Editor's note: This very interesting article carefully analyzes the advantage of using scripting languages but does not ignore that system programming languages will remain to have their place. The innumerable apps on the Internet are not imaginable without the use of app platforms and scripting languages.]

The Emergence of Distributed Component Platforms;

David Krieger et al. (p. 43) "Distributed component platforms (DCP) isolate much of the conceptual and technical complexity involved in constructing component-based applications. ... Sun's JavaBeans has emerged as the leading rival to Microsoft's DCOM (Distributed Component Object Model), ... With DCPs, businesses can assign their few highly skilled programmers to component construction and use less sophisticated developers to carry out the simpler assembly tasks. ... However, Internet and Object Management Group (OMG) component standards are emerging that will likely impact both the content and status of these two DCPs. We also discuss component frameworks, which extend DCPs to provide more complete application development solutions." [Editor's note: This interesting article, which is still worth reading today, discusses in detail the properties of DCPs and the frameworks that ease the use of such platforms. It concentrates on JavaBeans and DCOM but does not ignore other efforts like those of the OMG.]

Enterprise Knowledge Management; Daniel E. O'Leary

(p. 54) "As employees turn over in today's overheated job market, organizations are likely to lose access to large quantities of critical knowledge. Can we create a system that will capture company-wide knowledge and make it widely available to all its members?" (p. 55) "In many companies, one of the first KM tools is a data warehouse. ...Rather than the kind of quantitative data typical of data warehouses, knowledge warehouses are aimed more at qualitative data." (p. 56) "Knowledge discovery is a method that includes different tools and approaches to analyze both text and numeric data." (p. 60) "Ultimately, KM systems require a strong leadership that instills a culture of knowledge sharing. ... The types of incentives and the ability to measure contributions to KM generally are contingent on the level or function in the organization and the particular application to which the KM system is being put." [Editor's note: This very interesting article discusses many of the techniques that were used in 1998 to handle "knowledge" explicitly entered into the system. However, it does not foresee the wide usage of knowledge extraction from all kinds of data with or without knowledge of the information provider. It has become big business to provide/sell extracted knowledge widely, thereby ignoring privacy concerns of both industry and individuals.]

Surviving Network Partitioning; Peter Michael Melliar-Smith et al.

(p. 62) "A distributed application consists of many processes, spread across a number of processors.

The processes cooperate to perform the tasks of the application, and some of the data may be replicated in several processes. ... Maintaining the consistency of replicated data also depends on information about the configuration—the processors and the topology of the network—and, for each process group, the membership—the processes that are members of that group.” (p. 65) “Normally, however, a transaction commits and generates a further transaction, called a fulfillment transaction, which records what has been done by the transaction and why. ... When communication is restored and the components remerge, the fulfillment transactions inform the processes in the primary component of the actions taken in a nonprimary component.” [Editor’s note: This approach requires human involvement to design the fulfillment transactions and has not been successful in commercial systems.]

Dicaf: A Distributed Architecture for Intelligent Transportation; Noppanunt Utamaphethai (p. 78) Wouldn’t it be great if you knew how traffic was shaping up on your alternative routes? Even better, wouldn’t it be great if your car knew and could map out the best route, suggesting changes in course, even while you were driving, to accommodate for changes in congestion? ...Dicaf (distributed scalable architecture for IVHS [Editor’s note: intelligent vehicle highway systems] using a continuous-function congestion measure) is a new system now being proposed for ITS [Editor’s note: intelligent transportation system].” (p. 80) “In the Dicaf framework, automobiles would communicate with the local DTMC [Editor’s note: distributed traffic management center] via cellular, infrared, or radio technology. ... The navigation systems in the vehicles—which would contain a detailed map of the US highway system—would then use that information to determine the fastest route. The DTMCs could also provide information on weather conditions and the availability and hours of restaurants, banks, and other services along the route.” [Editor’s note: This very interesting article predicts many features now commonly found in our car navigation systems but proposed the not-yet-existing intercar communication instead of GPS and the ever-present wireless phone and Internet facilities.]

Rescuing La Scala’s Music Archives; Goffredo Haus (p. 88) “Over the course of nearly five decades, La Scala has produced more than 5,000 tapes (analog open reels from 1951 to 1990 and digital audio tapes since 1991). ... Unfortunately, most of the tapes have not been well preserved and are deteriorating at a rapid pace. ... Once the tapes are of a satisfactory quality for transferring, the contents are digitized at the standard sampling rate of 44.1 kHz with a 20-bit analog-to-digital converter.” [Editor’s note: This commendable effort is described in detail using the digitization technology of 25 years ago. It is not known to this editor whether further transfers have been executed, as preservation relies on technology that is still in use.]

None of Us Is as Smart as All of Us; Warren Bennis et al. (p. 116) “With poetic economy, it summed up so much of what we discovered when we looked at remarkable creative collaborations. ... Above all, each was a refutation of one of America’s most treasured cultural myths—that of the Lone Ranger. ... We still tend to think of our institutions as lengthened shadows of a Great Man or Great Woman. ... The leaders of Great Groups worship talent, and they know where to find it.” (p. 117) “Great Groups crackle with excitement. No one needs to be told to work harder because everyone is addicted to the work.” [Editor’s note: This is a very interesting and very true article. It reflects very well what many of us know implicitly about successful cooperation.]

Design Patterns and Language Design; Joseph (Yossi) Gil et al. (p. 118) “Abstraction, a fundamental objective of good software development, is the process of identifying commonalities and then capturing them in abstraction mechanisms. ... Viewed from this perspective, each language is a toolbox of abstraction mechanisms. Object, class, and inheritance, for example, are the fundamental abstraction mechanisms in object-oriented languages.” (p. 119) “Design patterns enhance the power of object-oriented mechanisms by capturing non-trivial relationships and interoperations. ... A software engineer can use an entire toolbox of abstraction mechanisms that can be manually adapted to various circumstances. What is saved is the up-front expense of a well-integrated set of general-purpose mechanisms.” [Editor’s note: This article actually goes beyond patterns and also discusses concepts like clichés, idioms, and cadets.]

Web-Based Knowledge Management; Hermann Maurer (p. 122) “It has become increasingly clear that current Web structures are simply not well designed for customized information access. ... But it isn’t too late. ... Yet storing documents without meta-information is quite common practice for most Web applications. ... Using metatags is becoming slightly more popular due to initiatives like the ‘Dublin Core.’ ... We need to group documents together, and group the groups together, too. ... Grouping information would also allow developers to create different information views for different user groups.” (p. 123) “If this document is deleted, it’ll introduce dangling links, which we’d all like to have deleted automatically. ... These developments over the past few years all point to a new kind of Web paradigm—let’s call it Web-based knowledge management—that will shape the future of the Internet.” [Editor’s note: This very interesting article discusses issues that plague the web even today and actually have become a more serious problem when looking for “certifiable” information. In 2001, the introduction of the Semantic Web by Tim Berners Lee tried to add some of the asked-for features into the web design, but even those concepts are not widely utilized when creating web content.] ■