Report on the 10th IFIP Congress

September, 1986
Doc. 860911 fpb
TR86-021

Frederick P. Brooks, Jr.

Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina 27514

Report on the 10th IFIP Congress

Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill

Summary: Conclusions and Recommendations

This Congress clinched for me the conclusion the triennial IFIP Congresses have outlived their usefulness – indeed, that all general scope computer conferences (including the NCC) have outlived their usefulness. Hence I recommend that

1. NSF should decide now not to use its funds to subsidize attendance at the 11th Congress in San Francisco in 1989, and

2. NSF should so notify the IFIP Organizing Committee now, while the plans are still fluid, lest there be unfulfilled expectations.

Discussion

The 10th triennial international Congress of IFIP was held at Trinity College, Dublin on September 1-5, 1986. Both the city and the university were near-ideal hosts, providing unusually convenient accommodation, excellent and compactly located meeting halls, and most attractive surroundings for the Congress social program and pre-Congress and post-Congress sight-seeing. The program was carefully crafted and included many notable speakers, whose papers were, by and large, well-aimed at the broad interests of the computer professional, rather than narrowly at the specialist.

Nevertheless, the Congress attracted only 1200 people, which I contrast with the 15,000 or so who attended the quite specialized SIGGRAPH '86 at Dallas two weeks before. I found SIGGRAPH to be dense with helpful new information; I found IFIP to be interesting but not really worth a week of my time. I will not go again.

One of the prime justifications of IFIP activities is the contact among senior professionals from different nations and continents. I think it is clear that the specialized IFIP working groups, workshops, and conferences are in fact meeting this need. The IFIP Congress is not; most of the senior scholars in Europe were not there, even though the Congress was in Europe. They caught on, one or two Congresses sooner than I did, that the broad-scope Congress is not as useful an investment of time and money as the more specialized meetings.

The question of whether the broad-scope Congress has outlived its usefulness was, I understand, debated long and intently in the IFIP General Assembly meetings held

in conjunction with the Congress, and, surprisingly, the General Assembly seemed very open to the possibility of ending such Congresses. I think NSF should give them every encouragement to do so.

Papers, etc.

I have learned that panels tend to have low content, so I attended none. All the assessments I heard from others confirmed the wisdom of this decision.

Papers I found notable, and whose reading I would recommend are:

1. Cheriton, in "Software Technologies and Paradigms", nice report.

2. Gurd, "Dataflow Computers", a very good overview.

3. Maibaum, "Role of Abstraction in Program Development", thoughtful.

4. Barron, "OCCAM and the Transputer", a truly great paper after the model of the classical FJCC, SJCC "new computer introduced" papers.

The Transputer is a new microprocessor architecture especially designed to be an element in fine-grained multiprocessor systems – "thousands to tens of thousands". The novelty lies chiefly in the inter-processor communication facilities, which were designed by C. A. R. Hoare and provide built-in primitives for his Communicating Sequential Processes protocols. Each chip has four inter-processor channels (plus other I/O). Each such channel provides for two-point connection. At SIGGRAPH, a start-up company, Meiko Incorporated, exhibited a graphics engine made of up to 157 Transputers.

The Transputer designers, after much thought, decided *not* to build a RISC machine. The arguments are worth reading. The most telling two are (1) that protection and synchronization demand atomic multi-step primitives, and (2) multiplication is so frequent as to justify by itself the provision of an operation subsequencing mechanism. Subsequencing does not necessarily imply microcoding, but the Transputer designers went on that far.

Several implementations exist at various word lengths. There are modest architectural differences among the different-word-length machines.

It looks like an ideal element for implementing Magò's L cells, and a suitable one for quick-prototyping T cells.

5. Balzer, "Living in the Next Generation Operating System", I didn't hear it because it conflicted with Barron's paper, but it is well worth reading. Basically, his next-generation OS is object-oriented throughout.

6. Goldberg, "The Programmer as Reader", reemphasizes that most programming is modification, not *ab initio* creation, and hence promotes the importance of document-management tools especially designed for software building.

7. Buxton, "Future Interface Management Systems", also a well-done overview, richly illustrated from his own experience. More about interfaces than about interface management systems.

8. McDermott, "Making Expert Systems Explicit", a fine overview of the history, status, and problems of knowledge acquisition, which is the central problem today in expert systems.

9. Muira, "Supercomputers in Japan", the best single short recent treatment of supercomputers in general, their historical evolution, and present trends, all illustrated with meaty examples from Fujitsu's experience.

10 H.T. Kung, responder to Muira. He challenged the very foundations of the supercomputer effort, multiprocessors with a small number of elaborate processors. Kung argued that VLSI technology and economics now mean that a large number of simple processors now give orders-of-magnitude better performance-price ratio. He showed how his programmable systolic array overcomes the rigidities of earlier systolic arrays, described a few running applications, and produced some stunning performance numbers.

11. Boehm, "Understanding and Controlling Software Cost", a masterly survey of sweeping scope, offering both broad perspectives and new specific data and analyses.

12. Joy, "The Workstation Approach to Software Engineering", really about status and prospects for workstations – the same lecture he gave here in March. Stunning in sweep, coverage, boldness. Not in the *Participants' Edition* of the *Proceedings.* Did not talk about software engineering at all, despite the title.

13. Turski, "And No Philosopher's Stone, Either". Turski shows that the developing software involves two transformations – the first from the real world of the application to the formal system of the software, and the second from the formal system of the (even abstract) software structure to the formal system of the machine. The former is inherently a *scientific, not* a mathematical endeavor, and is not capable of mathematical verification or proof. The latter is a mathematical task and is subject to proof. A profound insight, and one which makes clear the proper role of formal methods in software building.