



Forty Years of Software Engineering

Brian Randell



Computing in the Late 1960s

- Computers were mainly very big (room size) and expensive - mainly used by rich organisations for large scale commercial data processing and scientific calculations.
- The first mini-computers were starting to appear in well-funded laboratories.
- The idea of a personal computer had not yet arrived.
- Data bases, networks and distributed systems were also yet to come.
- Local area networks were unknown, though work had started on implementing ARPANET, the Internet's main predecessor.



Software in the late 1960s

- The term “software” had come into use, though as yet systems software was usually provided “free” with the hardware.
- Shifting to a different computer normally meant users having to abandon or rewrite all existing application programs.
- There were increasingly ambitious applications and systems software projects - involving “man-millenia” efforts.
- Organizations found themselves becoming very dependent on large and complex systems, and suffering cost, schedule, performance and reliability problems.
- The summary: “*underestimates and over-expectations.*”
- Only, in 1969, when IBM “unbundled” its software did software become a commodity; and a recognisable software industry start to come into existence.
- (The NATO conference title “*software engineering*” was essentially provocative.)



Fritz Bauer in
1968

ICSE, 14 May 2008, Leipzig



The 1968 Garmisch Conference

- The aim was a report that would be widely circulated to officials and governments, not just to the technical community.
- A tremendously excited and enthusiastic atmosphere developed at the conference as participants came to realize the degree of common concern about what some were even willing to term the “*software crisis*.”
- Many participants admit it changed their future research plans.
- By the end Peter Naur and I had been provided with a detailed proposed report structure.
- The following week we added quotations from participants’ documents, and taped discussions, to create the full report.
- The result was memorably described by Doug McIlroy as: “*A triumph of misapplied quotation.*”



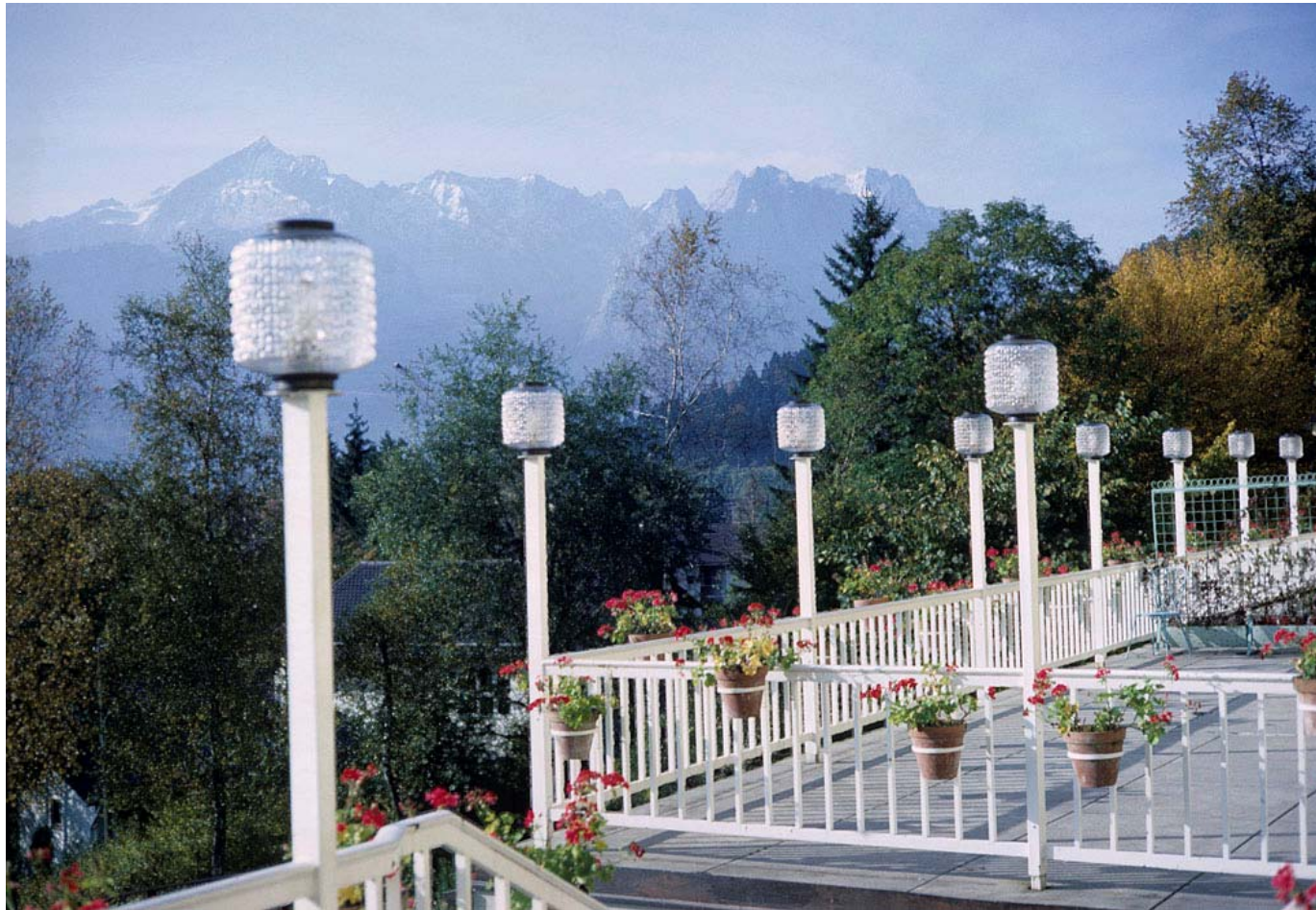
Golf Hotel, Sonnenbichl, Garmisch-Partenkirchen



ICSE, 14 May 2008, Leipzig



The Alps from the Hotel



ICSE, 14 May 2008, Leipzig



Doug McIlroy on Software Components, 1968



ICSE, 14 May 2008, Leipzig

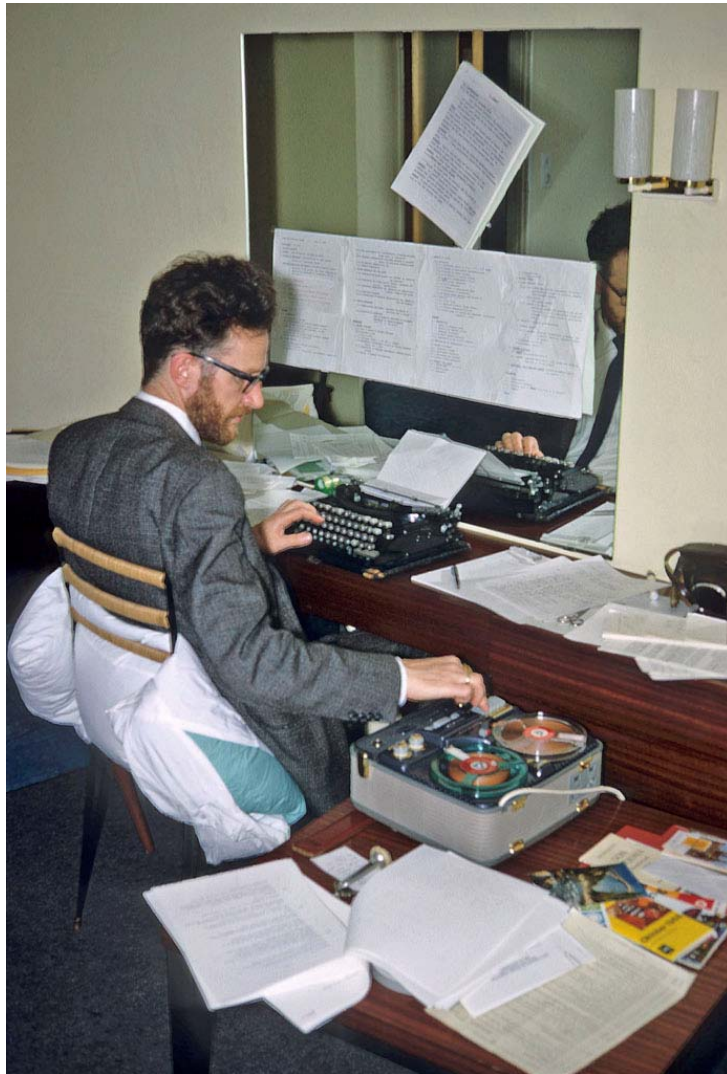


Peter Naur, 1968

ICSE, 14 May 2008, Leipzig



Producing the 1968 Report



ICSE, 14 May 2008, Leipzig



The 1969 Rome Conference

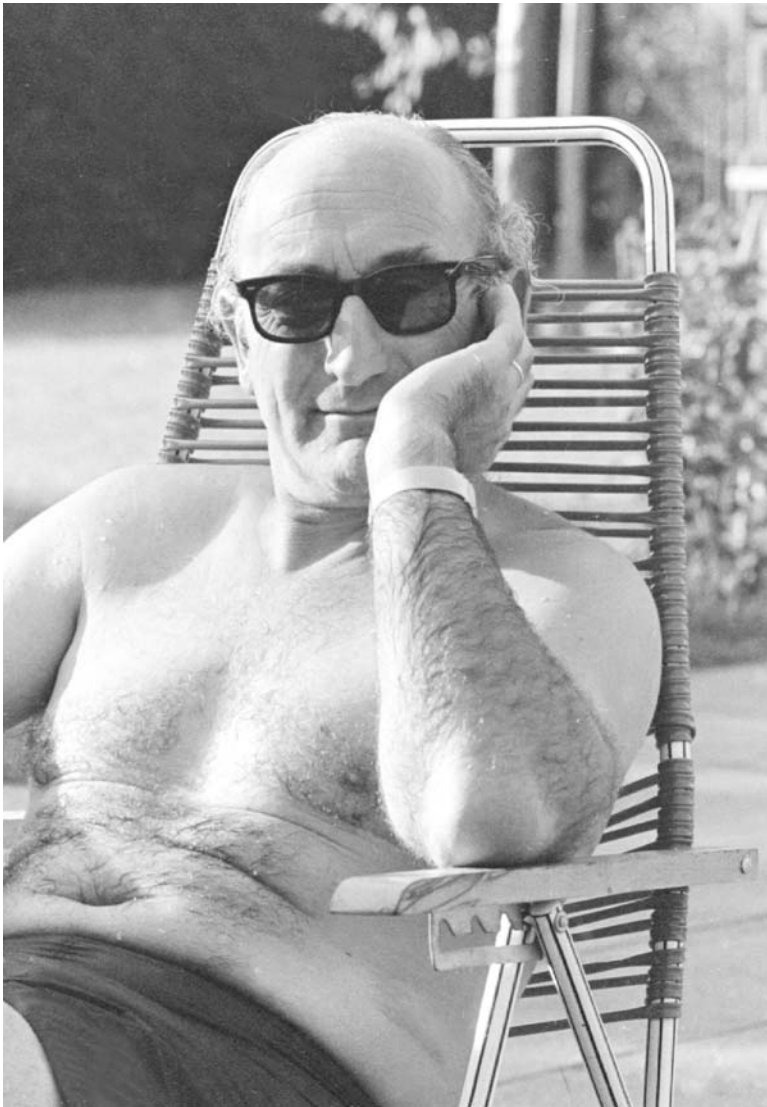
- Peter Naur and I were again invited to edit the conference report. He wisely refused – I naively agreed, and was joined by John Buxton.
- A gulf opened up between the researchers and practitioners, and many discussions were ill-tempered.
- Some participants talked as though “software engineering” already expressed a reality rather than a need.
- The organizers’ hidden agenda promoting creation of an “International Software Engineering Institute” was torpedoed by Tom Simpson’s satire on “*Masterpiece Engineering*”.
- There was no agreement on the intended report’s contents or structure.



Hotel Americana di Roma



ICSE, 14 May 2008, Leipzig



Fritz Bauer in
1969

ICSE, 14 May 2008, Leipzig



Doug Ross, 1969



ICSE, 14 May 2008, Leipzig



Producing the 1969 Report - 1

- My initial (over)confidence – based on the experience of editing the first report and the fact we had two experienced technical writers from ICL and two expert secretaries to help.
- Straight after the conference the six of us, plus luggage, and a very impressive set of typewriters, tape-recorders, boxes of paper and other office supplies, etc., were transported by minibus to Central Rome to the very pleasant hotel (belatedly booked by the conference organisers).
- I still treasure the memory of our arrival and of how we managed to find continual comfort in humour despite the pressure of work and many adversities.



Ian Hugo,
Ann Laybourn,
Margaret Chamberlain
and John Buxton

ICSE, 14 May 2008, Leipzig



Producing the 1969 Report - 2

- By mid week, almost all of the original typewriters and tape recorders were broken, and we were threatening to decamp to NATO Headquarters in Brussels.
- Half the report had to be typed by Ann Laybourn on a totally-unfamiliar German-keyboard typewriter.
- Despite such problems, whose impact would have been much less had we had the promised local assistant, the report was completed by the Friday, in time for a celebration dinner.
- Nearly all the restaurant waiters in Rome chose that moment to go on strike!
- And we were forced to omit “*Masterpiece Engineering*” (which we’d wanted as the concluding chapter).
- But the report was well received– and described as “*much better than the conference it allegedly records*”.



Some of the Censored Text

“ . . . The next natural step to take was, of course, to double the number of painters but before taking it they adopted a most interesting device. They decided to carry out some proper measurement of productivity. Two weeks at the Institute were spent in counting the number of brush strokes per day produced by one group of painters, and this criterion was then promptly applied in assessing the value to the enterprise of the rest. If a painter failed to turn in his twenty brush strokes per day he was clearly under-productive.

Regrettably none of these advances in knowledge seemed to have any real impact on masterpiece production and so, at length, the group decided that the basic difficulty was clearly a management problem. One of the brighter students (by the name of L. da Vinci) was instantly promoted to manager of the project, putting him in charge of procuring paints, canvases and brushes for the rest of the organisation. . . .”

From Tom Simpson's: *Masterpiece Engineering*



Since the NATO Conferences

- The “bespoke” software industries are embarrassingly recognisable successors to the 1960s software world – attempting ever larger and more complex tasks, but having continuing challenges re cost, schedule, performance and reliability.
- In contrast there are now software “package” industries – tending to “natural monopolies”, and experiencing Darwinian-type evolution:
 - These industries provide a huge and wonderful marketplace of very useable and useful software
 - This software’s validation and refinement is helped by vast numbers of (unwilling?) users
 - But technical mono-culturalism (allied to the growth of networking) has spawned a large computer crime industry (unforeseen at the NATO conferences)



Three Continuing Topics

- Among the many looming technical developments discussed enthusiastically at the NATO conferences, two now stand out as still of great interest and constituting a continuing challenge:
 - Software components (subject of the keynote speech by Doug McIlroy at Garmisch).
 - Software development tools and environments.
- As does a third which was already under way, though not featuring large at the conferences:
 - Multiprocessor system design.
- My view: progress in all three has been rather disappointing.



1 – Software Components

- The one undoubted success – UNIX pipes and filters – was already identified by McIlroy in 1968.
- A more doubtful success – the way large applications are built nowadays using a few very large components: an operating system, a browser, a database transaction processor, and a development environment.
- However, as Butler Lampson has pointed out: *“You use only a small fraction of the features of each component, and your program consumes 10 or 100 times the hardware resources of a fully custom program.”*



2 – Software Development

- Software development is fragmented across a ridiculous number of different languages, methodologies, platforms, and “standards”.
- The HOPL interactive register of programming languages currently lists 8512 languages!
- Quoting from “*Professionalism in IT*” by Les Hatton –
 - “*We actively encourage students to work on real projects for industry and as a result one of my colleagues this year had students submitting projects in C, C#, C++, Java, PHP, MySQL, XML, HTML, XHTML, VB.Net on XP, Mac OS X, Linux and even Vista with Eclipse, Netbeans, Ant, JWSDP, Glassfish, DreamWeaver, Developer Studio, .Net with maybe even some Etruscan.*”
- There are many overly-distinct R&D communities.



3 – Multiprocessor System Design

- Amdahl's Law, which governs the speed-up obtainable through exploiting multiple processors in parallel, is (like Murphy's Law) as valid in today's multi-core world as it was 40 years ago.
- High performance (i.e. linear speedup) from parallelism works well only for:
 - small numbers of processors
 - or for so-called "embarrassingly parallel" problems - ones for which no particular effort is needed to segment the problem into a very large number of parallel tasks, and there is no essential dependency between those tasks
 - though there are many such problems, it seems they often need separate investigation/development

To conclude: I wonder (i) whether others share my disappointment regarding the progress that has been made in these three areas, and (ii) what future progress can be realistically expected.