

ERTS – Toulouse

Engineering Software Engineering

M. le Maire, Messieurs et Mesdames:

C'est la deuxième fois que j'ai eu l'honneur de parler dans cette salle si magnifique, cette Salle des Illustres. Mais dans les années qui sont intervenues, j'ai eu peu d'occasion de parler français, et je suis certain qu'il sera mieux pour nous tous si je continue in English.

As you know, I was kindly invited by Jean-Claude Laprie to give a banquet speech marking the occasion of the fortieth anniversary of the 1968 NATO Conference, the conference that started the software engineering bandwaggon rolling.

After such a magnificent banquet, I am sure that you do not want to listen to a lengthy and serious account of anything, leave alone of software engineering and how it has developed over these last forty years.

So I could instead deliver a typical British after-dinner talk - but the trouble is that such a talk is rather like a cross between a religious sermon, and a stand-up comedy act ("*l'acte d'un comique qui fait son one-man show*", *d'un "raconteur de blagues"*?). So the British after-dinner speech is a very peculiar art form, peculiar just to the British, since it does not cross the Atlantic successfully, leave alone the English Channel, as we insist on calling "*La Manche*".

So rather than inflict on you either a scholarly lecture, or a British-style after-dinner talk, I will instead provide some informal reminiscences about those long distant days of the first software engineering conferences.

But first I fear I need to explain my choice, for this banquet speech/after-dinner talk, of the title, "*Engineering Software Engineering*" – in particular to explain that I am making use of two rather different meanings of the verb "*to engineer*".

Assuming I am allowed to quote the Oxford English Dictionary in this the land of *l'Academie Française*:

The Oxford English Dictionary's main definition of To Engineer is: "*To act as an engineer*", which can I presume be regarded as a wholly desirable and respectable activity.

But another meaning the dictionary provides of To Engineer is: "*To arrange, contrive, plan, manoeuvre*" – which, with its connotations of craftiness or deviousness, is very different, and often not at all respectable.

And I need to make use of *both* meanings if I am going to describe to you, as is my plan in this brief talk, some of the behind the scenes activities at not just the first NATO conference, held in 1968 in Garmisch-Partenkirchen in Bavaria, but also the follow-up conference that was held a year later in Rome.

As I indicated, the events I'll talk about are from forty or so years ago – events that don't feel too long ago to me, though they must seem like another era to many of you, in computing terms a positively medieval one.

So let me tell some of you, and remind others, about what the world of computing was like in the late 1960s. Computers were mainly very big (room size) and expensive, and were mainly used by rich organisations for large scale commercial data processing and scientific calculations, though the first minicomputers 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, but work had started on implementing ARPANET, the Internet's main predecessor, though local area networks were unknown.

The term “software” came into use, though as yet systems software was usually provided “free” with the hardware by the computer manufacturer, and applications software was normally designed specially for particular clients and particular computers.

In those pre-Microsoft days, shifting to a new computer normally implied – because of hardware incompatibilities – users having to abandon or rewrite all existing applications programs.

Increasingly ambitious applications and systems software projects were being undertaken, and organizations found themselves becoming much more dependent on large and complex computer systems than had previously been the case.

It was the US military-industrial complex that first started to try and develop very large software systems involving literally thousands of man-years of effort, man-millenia in other words. The secrecy that cloaked such systems' purposes served also to hide the extent to which such projects were often characterised by “*underestimates and overexpectations*”.

This phrase “*underestimates and overexpectations*” was in fact the title of a paper written in 1969 by Licklider, a paper which stated “*At one time, at least two or three dozen complex electronic systems for command, control and/or intelligence operations were being planned or developed by the military. Most were never completed. None was completed on time or within the budget.*”

Licklider's paper was a contribution to the then very active public debate about the Anti-Ballistic Missile System, a project that was clearly going to require immensely sophisticated software. I still remember the ABM debate vividly, and my horror and incredulity that some people really believed that one could and should depend on massively complex software

systems to automate the detonation of one or more H-bombs at exactly the right time and place over New York City to destroy just the incoming missiles, and not the city and its inhabitants.

These various large software projects were almost all one-off projects, developed for specific customers. It was perhaps only when, in 1969, IBM “unbundled” its software by pricing it separately from its hardware that software became a commodity; and a recognisable software industry, and the notion of package software, started to come into existence. (It was in fact only in 1968 that the first software patent was granted.)

It is intriguing to see how software was regarded at this time. Let me quote from the 1968 volume of the very influential book series “Advances in Computers”:

“There is a seductive fascination in software. There is a pride of accomplishment in getting a program to run on a digital computer, a feeling of the mastery of man over machine. And there is a wealth of human pleasure in . . . mental exercises that bear a strong resemblance to such popular pastimes as fitting together the pieces of a jigsaw puzzle, filling out a crossword puzzle, or solving a mathematical brainteaser of widely familiar variety. And what is more, digital computer programming is an individual skill, and one which is relatively easy to learn”

But it was also in 1968 that a much-cited study of the huge variability of programmers was published. This revealed startling differences between the worst and best programmers in the group of employees that was monitored the best programmer coded and tested twenty-five times faster than the worst, and the resulting programs were thirteen times faster!

This then was the world of software that formed the background to the first NATO conference. In such circumstances, it is hardly surprising that the term “Software Engineering” was essentially unknown and unexpected, when it was chosen as a conference title by NATO’s Study Group on Computer Science. This Group, chaired by Professor Fritz Bauer of the Munich Technical University, had been tasked with organising a conference, and perhaps later an International Institute of Computer Science. (I’ll return later to this ulterior motive of an International Institute.)

It is important to understand that the title ‘software engineering’ was deliberately chosen – I think by Fritz Bauer personally – as being *provocative*, as implying, and I quote: *“the need for software manufacture to be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering”*. There was, I should stress, no attempt by the 1968 conference organisers or attendees to claim that such an engineering field actually existed!

At the time I was working at the IBM Research Center in Yorktown Heights, NY – and was invited by Fritz Bauer to join Peter Naur as co-editor of the planned conference report. (I should mention that right from the start the conference was aimed at producing a report that would be widely circulated to officials and governments, not just to the

technical community.) Peter Naur was already famous as the Editor of the wonderful Algol 60 Report, and we were both members of the IFIP Working Group on Algol.

Many attendees have since described the Garmisch Conference as one of the most stimulating conferences that they have ever attended. A notable aspect of the conference was the willingness of the participants, “*about 50 experts from all areas concerned with software problems – computer manufacturers, universities, software houses, computer users, etc.*”, to admit the extent and gravity of those software problems.

As the late great Edsger Dijkstra notably said: “*The general admission of the existence of the software failure in this group of responsible people is the most refreshing experience I have had in a number of years, because the admission of shortcomings is the primary condition for improvement.*” No wonder then that many of the participants view the 1968 Software Engineering Conference as a turning point, an event that had a significant effect on their attitudes and their subsequent work in the software field.

Many people, such as Dijkstra, were prompted or encouraged to work on the problem of how to create programs that were free from errors, and of how to prove, to prove mathematically, that this was the case. In contrast, and appropriately reflecting my and Dijkstra’s respective levels of programming skill, I became interested in the problem of how to design programs that could be usefully relied upon even if it was admitted that they still contained yet-to-be found bugs.

But I am getting ahead of myself. As I and other participants have since testified, a tremendously excited and enthusiastic atmosphere developed at the conference. This was as participants came to realize the degree of common concern about what some were even willing to term the “*software crisis*”, and general agreement arose about the importance of trying to convince not just other colleagues, but also policy makers at all levels, of the seriousness of the problems that were being discussed.

Thus throughout the conference there was a continued emphasis on how the conference could best be reported. Indeed, by the end of the conference, Peter Naur and I had been provided with a detailed proposed structure for the report that we were committed to produce. This structure was based on a logical sequencing of the topics discussed, rather than the actual way in which the conference’s various sessions had happened to be timetabled.

During an intensive week in Munich immediately following the conference, Peter and I with a number of helpers produced a report by fleshing out this detailed structure with quotations from documents produced by the participants before and even during the conference, and from the taped records we had of every conference session.

The resulting report was widely circulated by NATO. (It is now of course also on the web.) Years later I learned that when Alan Perlis, for many years the very influential director of Carnegie Mellon University’s Computer Science Department, when he received his set of copies of the Report, he took them down to the graduate students’

room and distributed them, saying: “*Here – read this – it will change your life!*” But the comment about our Report that I most enjoyed was that made by another attendee, Doug McIlroy of Bell Labs, who memorably described the report as: “*A triumph of misapplied quotation*”.

The editing activity in Munich was as enjoyable as it was intense, and afforded plenty of opportunity for re-hearing some of the more noteworthy discussions, so that many of these became etched much more deeply into my memory, and had a stronger effect on my subsequent research, than would have been the case had I merely taken part in the conference.

In 1969, after five years in America, I left IBM Research and returned to Britain, to join the University of Newcastle upon Tyne – a move I’ve often described as “*leaving the ivory towers of industry for the sordid commercial reality of a university computing science department!*”

The NATO Committee decided there would be a follow-up conference, to be held in Rome in 1969. Peter Naur and I were again invited to edit the conference report. He wisely refused – I naively agreed, and it was arranged that I would be joined by John Buxton, then of Warwick University, one of the 1968 conference participants.

A conference planning meeting was held at the NATO Headquarters in Brussels where, based on my experience of the Munich report writing task, I listed a set of requirements for carrying out the report editing, chief among which was the need for us to have a local minder – someone who spoke Italian and who could help us if, or rather when, anything went wrong.

My main memory of that NATO Headquarters meeting is that it took place in an office that was dominated by a very large and impressive safe – a safe which to my amusement was revealed to be completely empty when our embarrassed host, who had to leave before the end of the meeting, had to unlock and open it so as to put away the bottles from which he had earlier served us drinks.

Unlike the first conference, at which it was fully accepted that the term software engineering expressed a need rather than a reality, in Rome there was a tendency on the part of some participants to talk as if the subject already existed. An even greater contrast to the Garmisch conference was the gulf that opened up between the researchers and practitioners, and the ill-tempered nature of many of the discussions.

And it became clear during the conference that the organizers had a hidden agenda, namely that of persuading NATO to fund the setting up of an International Software Engineering Institute. However things did not go according to their plan. The discussion sessions, which were meant to provide evidence of strong and widespread support for this proposal, were instead marked by considerable scepticism, and led one of the participants, Tom Simpson of IBM, to write and present a splendid short satire on “*Masterpiece Engineering*”.

John and I later decided that Tom Simpson's text would provide an appropriate, albeit somewhat irreverent, set of concluding remarks to the main part of the Rome report. However we were in the event persuaded by the conference organizers to delete this text from the report. This was, I am sure, solely because of its sarcastic references to a "Masterpiece Engineering Institute". I have always regretted that we gave in and allowed our report to be censored in such a fashion. So, by way of atonement, I'd like to read Tom's text to you now.

You may be interested in an experience I had last night while I was trying to prepare some remarks for this address. I was walking outside in the garden attempting to organize my thoughts when I stumbled over a stone in the ground. To my surprise as I picked myself up I saw that it had an inscription chiselled into it. With some difficulty I deciphered it; it began

"Here on this spot in the year 1500 an International Conference was held".

It seems that a group of people had gotten together to discuss the problems posed by the numbers of art masterpieces being fabricated throughout the world; at that time it was a very flourishing industry. They thought it would be appropriate to find out if this process could be "scientificized" so they held the "International Working Conference on Masterpiece Engineering" to discuss the problem.

As I continued walking round the garden, now looking a little closer at the ground, I came across the bones of a group, still in session, attempting to write down the criteria for the design of the "Mona Lisa". The sight reminded me strangely of our group working on the criteria for the design of an operating system.

Apparently the Conference decided that it should establish an Institute to work in more detail on production problems in the masterpiece field. So they went out into the streets of Rome and solicited a few chariot drivers, gladiators and others and put them through a five week (half-day) masterpiece creation course; then they were all put into a large room and asked to begin creating.

They soon realized that they weren't getting much efficiency out of the Institute, so they set about equipping the masterpiece workers with some more efficient tools to help them create masterpieces. They invented power-driven chisels, automatic paint tube squeezers and so on but all this merely produced a loud outcry from the educators: "All these techniques will give the painters sloppy characteristics", they said.

Production was still not reaching satisfactory levels so they extended the range of masterpiece support techniques with some further steps. One idea was to take

a single canvas and pass it rapidly from painter to painter. While one was applying the brush the others had time to think.

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.

Well, for all I know, the Institute may still be in existence. I leave you with one thought: in a few hundred years, somebody may unearth our tape recordings on this spot and find us equally ridiculous.

Needless to say, Tom Simpson's satire killed the subject of an International Software Engineering Institute stone dead, much to the ill-concealed disappointment of the conference organisers, who realised that their attempt at "Engineering Software Engineering" had failed quite spectacularly.

Writing the second report was also a very different experience, not least because – unlike the 1968 conference – the 1969 conference failed to agree on the contents and structure of the report that John Buxton and I were supposed to create. Frankly we had the task of "making a silk purse from a sow's ear", to use a well-known English idiom.

My initial confidence – or rather overconfidence – regarding the report editing task was in part because I thought I'd obtained the conference organisers' agreement on a set of improvements that would be made to the editorial processes and their support, compared to how we had organised and performed the task in Munich. (I was not to know that the organisers were going to fail to live up to their promises.)

But my confidence was also due to the fact that this second time around, John and I had been offered the full time services of two experienced technical writers from ICL, the British computer company that is now part of Fujitsu, namely Ian Hugo and Rod Ellis, and John and I had each arranged to be accompanied to Rome by an expert secretary, Margaret Chamberlain and Ann Laybourn, respectively.

On the Saturday morning following the conference the six of us, plus all our 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 (hurriedly booked at the last moment by the conference's organisers), a hotel which was situated just across from the main entrance to the Roman Forum.

In fact we arrived rather too early for the hotel, since only the small suite we were to use as an editorial office was available, our bedrooms not yet having been vacated and cleaned. We thus had to accept the hotel receptionist's suggestion that we all be initially installed in this one suite until our own rooms were ready.

I still treasure the memory of our arrival, which was watched open-mouthed by the various hotel staff and guests in the lobby. This was not just because of our number and our mountain of luggage, and the small army of porters – just one of whom had a door key in his hand – that were being employed to transport this mountain.

The open-mouthed reactions were undoubtedly also due to the interesting appearance the six of us must have made – in particular the fact that Margaret Chamberlain was wearing an extremely short miniskirt. In 1969 this fashion had yet to spread from London to Rome, where it was still regarded at least by all the Italian men as quite sensational. (Walking down a Rome street behind Margaret, watching Italian men's faces, not just Margaret, was quite an experience, I can tell you!)

And Rod Ellis was wearing a splendid long black leather jacket and the sort of thick-soled suede shoes that at that time were known, in Britain at least, as "brothel-creeper".

But most memorable of all was John Buxton's remark when the last of the porters had bowed himself out of our suite, and the six of us were standing around our luggage mountain wondering what to do first. He suddenly said, "*I've had a great idea! Let's phone down to the front desk and ask for two thousand foot of colour film and a stronger bed, please.*"

This provided a wonderful start to a week in which we managed to find continual comfort in humour despite the pressure of the work and the many adversities we had to face. For example, by mid week, almost all of the original typewriters and tape recorders were no longer operational, and we were threatening to abandon Rome and to move to Brussels in order to complete the work at NATO Headquarters. Even the stapler had broken.

As Ian Hugo has since reminded me, "*the suite had a bathroom which was surplus to requirements and the bath became the final resting ground for dead typewriters, tape recorders, etc; by the end of the week it was full to overflowing!*"

However we soldiered on, though in the end half of the Report had to be bravely typed by Ann Laybourn on a totally-unfamiliar German-keyboard typewriter that we had managed to borrow ourselves from the hotel.

All these adversities – whose impact would have been much less had we had the promised local assistant – all these adversities in fact helped to bind us together as a

team. Rod Ellis' brilliant gift for mimicry also helped by providing many welcome moments of general hilarity as, suiting his choice to the topic at hand, he switched effortlessly in conversations with us between the voices of Edsger Dijkstra, Fritz Bauer, and many of the other participants whose conference comments had been captured for posterity by our tape recorders.

We did in fact finish the report by early on the Friday evening – in good time for a final celebration dinner, once Rod and Ian had returned from the University of Rome where they had made copies of the draft report (and, rather fittingly, had broken the University's photocopier).

It was in keeping with the rest of the week, though, that nearly all the restaurant waiters in Rome chose that moment to go on strike – indeed, we saw a large procession of them march right past our hotel windows shouting and waving banners; so we had to content ourselves with dinner – in fact an excellent dinner – in the hotel.

Once the report was printed and published – using a very early computer-typesetting system at Newcastle University – I very deliberately distanced myself from the topic of software engineering. I greatly disliked the way in which lots of people started to claim, quite unjustifiably in my view, that the software-related activities they were pursuing merited being called “engineering”.

So for nine years, I refused all invitations, and there were many, to talk or write on the subject – only relenting when the International Software Engineering Conference decided to celebrate the 10th Anniversary of the first NATO Conference, and I agreed to be one of their four keynote speakers.

My task was to talk on “*Software Engineering as it was*” – the three other keynote speakers, all three of whom definitely can be classed as “*Illustres*”, were Barry Boehm on “*Software Engineering as it is*”, Wlad Turski on “*Software Engineering as it will be*”, and Edsger Dijkstra on “*Software Engineering as it should be*”.

Let me suggest that this set of talks might prove very interesting to you all now, even thirty years further on, as indeed might be the original 1968 NATO Report.

In fact, though you might have a very different opinion of this after-dinner talk, I personally will regard my talk as a success if it has motivated even just one or two of you to go and read the 1968 Report, and to reflect on its possible still-continuing relevance.

But I'm thinking just of the 1968 Report, not the report on the ill-fated Rome Conference. So I hope that, you never, ever, find yourself (perhaps at a future ERTS session) so appalled by the presentations you've just had to endure that you are tempted to compose and deliver your own equivalent to Tom Simpson's satirical essay on Masterpiece Engineering.

On the other hand, let me express the hope that you too will one day manage to engineer matters so that you find yourself in a situation where you are tempted to call the hotel front desk to order two thousand feet – that’s about six hundred meters – of colour film and a stronger bed!

But I am in danger of slipping into “*religious sermon/stand-up comedian mode*”, so it is time for me to stop – and to repeat my thanks to the conference organisers for the honour that they did me by inviting me to give this banquet speech, and to thank you all for your kind attention.

Footnote:

The NATO Reports, and other material, can currently be found at:

<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/>