# My Thoughts on Software Engineering in the Late 1960s

David Gries

Cornell University

Two years after I received my PhD, while an assistant professor at Stanford, I attended the NATO conference in Garmish, Germany, where the "software crisis" was first openly discussed and the term "software engineering" was brought to the fore. I suppose I was invited because Fritz Bauer of Munich, Germany, one of the organizers of this eventful international conference, was my PhD advisor. I was close to the youngest of the 50-odd participants.

I did my part. I listened attentively, I made a few very small points, and I helped organize and run one of the workshops. Yet I felt small and unsure of myself. I wondered whether I would ever be able to speak on a level with these people (the Algol-60 people like Bauer, Naur, and Perlis, who gave a good keynote speech, and Samelson; Doug McIlroy, who spoke so eloquently about the need for components and reusable software; Dijkstra; Galler; Graham; Ross; and others).

After the conference, I helped transcribe the tapes of the sessions and organize notes for the editors of the proceedings, Peter Naur and Brian Randell. This gave me a nice perspective, for I could go over what the people said at leisure and ponder. I now read some of the comments made during the conference with a chuckle (e.g. "use high-level [languages] for research production, low-level for commercial production"). But as I look back at the proceedings, I am struck by the perceptiveness of the attendees.

I knew that these people were right; we did not know much about programming or software engineering. I was a good programmer myself, a good hacker. But my experiences in teaching introductory programming at Stanford had shown me how little I really knew. I remember quite clearly introducing my own version of "stepwise refinement" and "top-down design," but of course I had neither the experience nor the eloquence of a Wirth or Dijkstra, who wrote such excellent works just a few years later. I remember trying to teach my students how to develop a loop, and realizing that I really didn't know how I developed one myself. I felt funny indeed, in front of the class. And the conference gave me the feeling that most of the other participants felt the same way about programming. They might be good programmers themselves, but they did not know why and they couldn't teach their trade to others.

Of program correctness concerns at that time, I knew nothing. The notion did come up during the conference, but no one had any idea what it meant, really. We were still floundering with the structure and organization of programs. The thought that programs might be treated as mathematical entities was indeed mentioned from time to time (sometimes with dismay).

The conference pinpointed many problems, but few solutions. However, it served its purpose of making people aware of the problems and their significance. And it stimulated the research that has had so much effect in the past twenty years.





©1989 ACM 0270-5257/89/0500/0098$00.75