The move to activity-centered views of software development

Christiane Floyd
MESHS Lille,
June 5, 2019

In this talk, I will reflect on the concern for richer views on software development, as evidenced in three papers which all originated around 1985. Although the authors knew each other well, the three papers were written independently and do not refer to one another. The papers are:

- *Programming as Theory Buildin*g (1985) by Peter Naur, editor of the Algol 60 report, co-founder of software engineering and expert on programming methodology,
- *Program Development as a Social Activity* (1986) by Kristen Nygaard, co-author of Simula 67, co-inventor of the class concept and pioneer in systems development with users,
- *Outline of a Paradigm Change in Software Engineering* (1987) by myself.

Since this is a historic approach, I feel the need to give some information on my background. At the time, I was a young professor in Software Engineering at the Technical University of Berlin. My previous experience had included 6 years of practice in programming and in method development for German software companies. My aim was to develop and teach methods that would on one hand be viable for guiding the work of software professionals in teams and on the other hand facilitate the development of software, adequate or suitable to meet the needs of users. Entering academia in 1978, I found a department with a strong focus on programming languages and formal methods, reflecting the then prevalent academic outlook of the emerging discipline of software engineering, which was in the 1970s almost exclusively devoted to formalization of processes and products.

My struggle to find my place as a player in this arena involved several efforts in parallel, which mutually influenced one another:

1. to understand the relevance of formal methods; this eventually led to the TAPSOFT (Theory and Practice of Software Development) conference that I organized together with my colleague Hartmut Ehrig at the TU Berlin in 1985;
2. to develop with my group our approach STEPS (Software Technology for Evolutionary, Participatory Systems Development) which combined communication, user participation, prototyping and incremental development with technical concerns for quality software;
3. to establish epistemological foundations beyond logical positivism and the rationalist tradition for these richer approaches in software engineering, culminating in the conference *Software Development and Reality Construction* (1988, book published in 1992).

In the first years, an important ally in this struggle was Peter Naur, who had co-authored the report of the founding conference for software engineering (1968) in Garmisch-Partenkirchen and became a leading critic of the formalization movement soon afterwards. The title of the present conference session implicitly refers to his book *Computing: A Human Activity*, featuring a rich collection of his papers that view the computer as tool for problem solving and programming as a comprehensive activity, integrating everything from understanding the problem to documenting the result. In the paper chosen here, the author draws on his own experience and argues (1) that the programmer builds a 'theory' of the program in the course of the sum total of his activities; (2) that the programmer 'owns' this theory, which can not be explicated completely in documents; (3) that the 'life' and 'death' of programs depend on the availability of this theory and therefore on the programmers, and (4) that the role of programmers should not be reduced to software production, but they should be seen as consultants to their clients in information matters.

In 1982, I turned to Kristen Nygaard in my search for allies in user-oriented design. He was the founder of an emerging Scandinavian School for Participatory Design and Systems Development

and very much the leader of this movement, which eventually was recognized and influential world-wide. His paper of 1986 is encompassing: he makes it clear how he synthesizes the individual contributions of a rather large number of his followers, and programmatic: ranging from basic definitions in the field of informatics to political measures ensuring work-oriented legislation in the field of computing. For a richer view of software development, I will focus on the following aspects: (1) 'perspectives' which, for him, are basic to understanding different views, for example those of software users; (2) 'systems', which he considers as being viewed as wholes by a person or a group of people for some time and for a specific purpose, thereby questioning the claim to 'objectivity' underlying classical software development; and (3) the interplay between 'process' and 'structure', here described as dialectic, which is related – though not identical – to my own approach.

To proclaim a 'paradigm change' in software engineering in the 1980s was not unproblematic for me. I wanted to put forth a substantial argumentation, but not offend anyone. It was not my intention to put down the mainstream – which I termed the 'product-oriented' perspective – but to create a platform that would also accommodate the 'process-oriented' perspective, which considered software in the context of human learning, work and communication arising in software development and use. I did this by considering several basic concepts in the field, such as 'program', 'system', 'method' and 'quality', from both perspectives. My basic point was that product- and process-oriented perspectives are complementary and interleaved – but hardly anyone read the paper this way, most everyone sided with one perspective or the other. My real problem, however, was: Can you 'proclaim' a paradigm change or can you only diagnose it after it happened? Fortunately, history was on my side – nowadays I am simply considered as a precursor of agile methods in the software engineering community.

All three papers make reference to philosophical schools to support their argumentation. In Peter Naur's case it is the philosophy of Gilbert Ryle and to some extent Sir Karl Popper, Kristen Nygaard is clearly rooted in dialectics, while I have an obvious connection to Thomas Kuhn and, less obvious, I am fascinated with constructivist epistemology.

In closing, I would like to mention that there were several other publications questioning he rationalist tradition in computing in the mid-1980s, most notably the highly influential book *Understanding Computers and Cognition* (1986) by Terry Winograd and Fernando Flores, which unfortunately must remain outside the scope of this talk, due to time constraints.