Talking about bugs: Male computer scientists and their emotional relationship with errors

Martina Heßler

For some digital technologists today, it is a matter of course to speak euphorically about their failed projects. They are proud of failing because they see it as an expression of their willingness to take risks, ambition, and originality. Failure qualifies them as innovative inventors. As Lisa Nakamura noted in a short essay, "[f]ailure has an exalted status in new and digital media culture."[1] However, this pride in failure is usually accompanied by a success story. As Nakamura also emphasized, it is primarily those who were ultimately successful who present themselves as proud of their failures.[2] In Silicon Valley, embracing failure has become a mantra, with variations such as "fail fast and often," "fail better," and "fail forward." In an article in *Forbes* magazine, Rob Ashgar stated: "embracing failure makes for a trendy mythology, especially for the aspiring heroes of innovation." In reality, however, it is "lip service, while they scramble hysterically to avoid failure at all costs." In fact, there is a fear of "any kind of failure."[3] These brief remarks touch on the emotions tied to failure and mistakes, which range from openly expressed enthusiasm and pride to a hidden fear of failure. Obviously, failure is deeply intertwined with emotions.

This chapter explores the relationship between emotions, gender, and technological errors and failures. However, it does not focus on enthusiastic project developers in Silicon Valley, but rather on a contrasting phenomenon: concerned software developers who were frightened by software errors and malfunctioning products. The chapter highlights a specific historical moment when software developers discussed software fallibility and sought solutions, namely during two NATO conferences: one in Garmisch-Partenkirchen (Germany) in 1968 and the other in Rome in 1969.

In the 1960s, a growing awareness emerged that software development was challenging, error-prone, and costly, yet increasingly vital to many aspects of society's functioning. At the beginning of 1967, the NATO Science Committee established a

1 Nakamura, "Plug and Pray," 87.

2 Nakamura, 87.

3 Ashgar, "'Fail Fast' Mantra."

Study Group on Computer Science, which began organizing the first conference.[4] This Science Committee was part of NATO's civil science program, aimed at promoting nonmilitary cooperation. Established in 1958, its goal was to encourage scientific cooperation within the Alliance.[5] Scientific cooperation and research across nearly all fields of science were supported. Therefore, it was no surprise that computer science became part of those efforts. By spring 1968, the conference program had been designed, and "key figures in academia, in the computer industry, in the emerging software houses had been invited."[6]

This chapter is mainly based on the reports of the two NATO conferences, each of which is approximately 130 pages long. Both reports document the lectures as well as the discussions that ensued.[7] These conferences have already been a focus of historical scholarship, mainly because they popularized the term "software crisis." Donald MacKenzie wrote a key essay on the subject and highlighted: "The historical significance of the Sonnenbichl [the conference hotel in Garmisch] meeting was the way it provided the single most influential early diagnosis of one of the two fundamental problems that are the obverse of hardware's success: the deeply intractable difficulty of achieving dependable software."[8] However, historians have criticized the term "crisis" and questioned its reality. Thomas Haigh pointed out that it exists more in historians' imaginations than in actual programming practice. He remarked critically that it was an "actor category" that historians should not adopt lightly.[9] Janet Abbate summarized the criticism that historians have directed at this term.[10] Most of them argue that talk of a crisis is exaggerated. This criticism is also supported by the fact that one participant in the first NATO conference remarked, "I do not like the use of the word 'crisis'. It is a very emotional word."[11] However, Abbate emphasizes that the term signaled a genuine concern among software producers,[12] since three problems, in particular, were repeatedly discussed in the computing literature of the 1960s and 1970s: "shortage of programmers; production problems that caused projects to be late, over budget, and full of bugs; and a supposedly undisciplined labor force."[13]

Following Abbate's remark that software developers expressed genuine concerns at that time, the categories and concerns of the actors are taken seriously in what fol-

4 Naur and Randell, *Software Engineering*, 8.
5 "NATO Science Committee." *Nature* 181 (1958): 738–40.
6 MacKenzie, "View from Sonnenbichl," 98.
7 Naur and Randell, *Software Engineering*; Buxton and Randell, *Software Engineering Techniques*.
8 MacKenzie, "View from Sonnenbichl," 97n.
9 Haigh, "Crisis, What Crisis?"
10 Abbate, *Recoding Gender*, chap. 3.
11 Naur and Randell, *Software Engineering*, 71.
12 Abbate, *Recoding Gender*, 89.
13 Abbate, 89.

lows. Reading the conference documentation, it is clear that the participants were outwardly emotional, as the participant quoted above rather critically remarked. However, the conference documents have not yet been examined regarding how participants felt when dealing with problems in software development and what that meant for the history of software. These emotions are at the center of this chapter, which aims to combine the history of emotions, gender history, and failure studies.

The first section situates the chapter within these research fields to clarify the potential insights that can be gained from their combination. The second and third sections examine male emotions in relation to the potential for failure, mistakes, and technological unreliability. These sections analyze the feelings during the two conferences, showing how a strong but short-lived emotional community formed during the first conference completely fell apart during the second conference, just one year later. The chapter concludes with a summary of its findings and some considerations about the role of emotions in the digital age.

Emotions, gender, and failure: New research perspectives

As William Reddy pointed out, historians cannot directly reconstruct feelings because they always result from an interaction between internal experience and the expression of feelings.[14] The historiography of emotions has strongly emphasized this point. Reddy introduced the term "emotive" to describe expressions of internal experiences.[15] He claimed that these expressions are used to navigate emotions. In the following, it will be examined which emotions participants in the two conferences expressed using which terms, and thus which emotives they generated. Following Reddy, it will be described how emotives were a way for the attendees of the conference to navigate their feelings. As will be shown, they thus built an emotional community at the first conference. Furthermore, I consider how the participants themselves evaluated their feelings and what significance it had for the history of software development.

To better understand this, I draw on the concept of emotional communities, developed by Barbara Rosenwein.[16] This concept allows for analyzing how software developers commonly expressed fear, concern, and unease, and how these shared feelings became the basis for further action and problem-solving. This concept will be elaborated on further in the second section of this chapter, within the empirical context.

14 Reddy, *Navigation of Feeling*.

15 Reddy.

16 Rosenwein, *Emotional Communities*.

Notably, this emotional community was exclusively male, with no women among the participants. "Over 50 experts from various fields related to software problems attended, including representatives from computer manufacturers, universities, and software companies."[17] However, only men were invited to participate. Not even computer scientist and mathematician Grace Hopper was included, although she was highly respected at the time

The history of computing has repeatedly demonstrated how women faced discrimination and marginalization.[18] To their credit, women's significant presence and influence in the early history of software have been recognized. From a gender perspective, the history of computing has involved the devaluation of women's activities, sexual segregation, and the process of masculinization. The 1968 software conference took place when many women still worked in computing; however, the professionalization and subsequent masculinization had already begun, as Nathan Ensmenger has shown.[19] The fact that only men were invited can be seen as part of this process of masculinization through professionalization. The conference aimed to address problems in software development and ensure the production of reliable software. At that time, no one seemed to believe women could contribute as well. It is worth noting that one member of the organizing committee explained that "the invitation list was carefully contrived."[20] He emphasized that the committee tried "to pick up the leading figures in their countries."[21] Excluding women from the event reflected the belief that only men were "leading figures" and capable of solving key problems in software development. This aligns with the widespread view that qualified and important work was a male domain.[22]

From a gender perspective, it is crucial to analyze male emotional culture and how it became part of a professionalization process that marginalized women. Notably, the historiography of computing and gender has mostly ignored emotions, focusing, if at all, on the feelings of users.[23] In contrast, the emotions of technology developers, especially software developers, remain a missing piece.

This chapter examines male emotions related to failure, malfunction, and errors. Thus, its goal is to contribute to the emerging field of failure studies and

17 Naur and Randell, *Software Engineering*, 8.

18 Abate, *Recoding Gender*; Ensmenger, "Making Programming Masculine,"; Ensmenger, The Computer Boys; Light, "When Computers Were Women,"; Hicks, "Sexism is a Feature,"; Misa, *Gender Codes*.

19 Ensmenger, "Making Programming Masculine".

20 Valdez, "Gift from Pandora's Box," 176.

21 Valdez, 176.

22 See Hicks, "Sexism is a Feature," 135–58.

23 See Berth, "Zwischen Hoffnung, Stolz und Wut," 229–49; Corn, *User Unfriendly*; Geuenich, "...gibt auch mal ein Küsschen," 271–90; Norman, "Computer Rage." Compare also Heidi Schweickert's chapter in this volume.

highlight the potential insights that can be gained from these perspectives. Neta Alexander coined the term "failure studies"[24] in an article and further developed this concept in a book co-authored with Arjun Appadurai.[25] In essence, failure studies incorporate approaches from various fields, including infrastructure studies, media studies, feminist theory, disability studies, and queer studies. According to Alexander, failure studies "include the analysis and mapping of noise, rupture, disconnection, and the limitations of human perception, knowledge, sensorium, and agency."[26] Alexander "seeks to foreground the inherent failures and limitations of digital technologies."[27] Steven Jackson had already argued for a "broken world" thinking in a 2014 article, calling for a shift beyond the focus on innovation and recognizing that technology often malfunctions.[28] He emphasized the importance of repair and maintenance, which are usually invisible yet vital for functionality and society, yet tend to be undervalued. Two aspects will be added to the emerging field of failure studies in this chapter. First, these studies could benefit from a historical perspective, which clarifies how the evaluation of failure and the approach to dealing with failure and technological errors have evolved and why. Second, failure studies focus on the role of users; thus, the technology developers and their handling of malfunctions remain absent. However, we cannot understand technological development without understanding the fundamental role of malfunctioning and dealing with errors in technological developments.

Ultimately, this chapter aims to contribute to the history of emotions in the digital age by examining the emotions of male developers regarding malfunctions and potential failures at a specific point in software history, namely a moment in which errors and failures of software systems became a heated topic within the software community.

Concerns, fears, and new beginnings: Emotional communities as a self-reflective view on software problems at the conference in Garmisch in 1968

Technological errors and malfunctions have become increasingly common since the latter part of the twentieth century. Engineer Henry Petroski noted in 1982 that he lived in a time when technology was running amok, with technological errors

24 Alexander, "Rage against the Machine." 1–24.
25 Appadurai and Alexander, *Failure*.
26 Alexander, "Rage against the Machine," 15.
27 Alexander, 4.
28 Jackson, "Rethinking Repair," 221–39.

happening "left and right."[29] He observed many "horror stories" about technology. Technological errors, of course, are not unique to modern societies. Even in ancient times, every fourth ship was wrecked.[30] Boiler explosions, mining disasters, and industrial accidents were the much-maligned side effects of industrialization. Nevertheless, the types, scales, and consequences of technological errors have undergone significant changes with the development of software. Computer scientist John Licklider stated in 1969:

> The essential facts are that all complex programs contain programming errors, that no complex program is ever wholly debugged, and that no complex program can ever run through all its possible states or conditions to permit its designers to check that what they think ought to happen actually happens.[31]

In 1985, software developer David Parnas warned "that complex software was prone to unreliability for 'fundamental mathematical reasons' that would 'not disappear with improved technology.'"[32] These severe warnings were also echoed in sarcastic jokes. Lisa Nakamura mentions in her essay "Plug and Pray" a joke about Bill Gates and Jack Welch (CEO at GM) from 1998 that highlighted the fallibility of software. Welch claimed that if GM had developed its technology like Microsoft had "for no reason at all, your car would crash twice a day."[33]

At the end of the 1960s, however, the participants in the NATO conferences were not in the mood to laugh. They emphasized that they were facing significant challenges. At that time, as Janet Abbate summarized, "the size, novelty, and complexity of the programs … led to some spectacular failures."[34] From a business perspective, the enormous costs resulting from delayed, erroneous, or failed software projects posed a significant problem. For software developers, however, the potential consequences of unreliable and faulty software were the primary challenge. Software programmers already had extensive experience with failure and malfunction. In her dissertation on the early history of software, María Valdez even concludes that the software developers attending the conference were traumatized. She referred to the IBM 360 and the so-called third-generation computers. The programming of these new computers encountered "enormous problems" and "created an awareness for the first time of the importance of programming."[35] Thus, "the participants brought with them to Garmisch the experience of the third generation, the trauma of the

29 Petroski, *To Engineer is Human*, 3.
30 Wolf, "Schiffsbruch mit Beobachter," 20.
31 Licklider, "Underestimates and Overexpectations," 51.
32 Parnas, "Software Aspects," 1328, quoted in Slayton, *Arguments That Count*, 173.
33 Nakamura, "Plug and Pray," 88.
34 Abbate, *Recoding Gender*, 89.
35 Valdez, "Gift from Pandora's Box," Introduction.

360, the difficulties experienced in implementing large projects, the failure to meet deadlines, the disappointments of the users, the example of disasters or near-disasters, the anxieties of unbundling."[36]

An analysis of the conference documentation reveals the extent to which the software developers were troubled at the time. They shared a common concern, feeling unease and fear. Donald MacKenzie called the conference a "defining moment of self-reflection in the history of software."[37] While the term self-reflection implies a rational effort, the conference documents reveal how emotionally the software issues were discussed. Software developers shared emotions, thereby building a community where self-reflection could occur in the first place.

As mentioned earlier, Barbara Rosenwein coined the term emotional community: "An emotional community is a group in which people have a common stake, interests, values, and goals."[38] These are "groups in which people adhere to the same norms of emotional expression and value."[39] Rosenwein used examples from the Middle Ages to demonstrate that different groups have their particular norms of emotional valuation and expression. The concept has already been widely adopted, used, criticized, and modified in the history of emotions. Therefore, it is essential to define it in each case. In what follows, this term is used to show that the software programmers at the 1968 conference in Garmisch built a community based on shared emotions. However, it was not a stable emotional community of a specific social group in Rosenwein's sense. Instead, it was a temporary emotional community formed during a conference and fell apart at the follow-up conference one year later. Thus, the concept is applied here to demonstrate how shared emotions can be very effective, even if they are short-lived and fragile. The shared emotions united the conference participants in Garmisch, fostering a sense of community. This emotional connection laid the ground for acknowledging and discussing the existence of a fundamental problem with software development. Further, it enabled future efforts to resolve the crisis.

This emotional community in Garmisch in 1968 is historically significant because it allowed participants to express feelings that had previously been largely ignored. Having the opportunity to share negative emotions, such as fear, anxiety, and a sense of losing control in the face of technological errors, turned the software programmers into a community. The ability to openly discuss their concerns and fears distinguished this emotional community. As Dutch computer scientist and mathematician Edsger Dijkstra, also a participant of the conference, remembered a few years later, it had been difficult to talk about problems before the conference: "to talk

36 Valdez, 176.

37 MacKenzie, "View from Sonnenbichl," 99.

38 Rosenwein, *Emotional Communities*, 24.

39 Rosenwein, 2.

about a software crisis was blasphemy."[40] This only became possible in public during and after the conference in Garmisch. Others also remembered this turning point: "This was quite new: before Garmisch, the literature tended to emphasize the 'seductive fascination' of software rather than its failure."[41] The opportunity to discuss problems openly was a hallmark of the emotional community in Garmisch.

The first software conference in Garmisch-Partenkirchen mainly addressed the challenges of software development. Essentially, the software developers acknowledged the demanding and complex nature of software development and the limited capabilities they had in creating reliable software at that time. They discussed a lot of examples, particularly in the field of safety, i.e., aircraft safety, the problem of faulty software in military projects, as well as the "overall health of the industry."[42]

The comments and presentations at the conference had a tone of concern and anxiety. Referring to Reddy, certain emotions repeatedly surfaced, used to express feelings and to help "navigate" them. Comments often began with phrases like "What worries me," "What concerns me," and "What frightens me." The word "problem" appeared in nearly all documented contributions, along with references to "nightmare," "danger," and "disaster."

The software developers felt uneasy as they created something that was increasingly being applied in everyday life, yet they did not fully understand it. They noted "the problems of achieving sufficient reliability in the data system, which are becoming increasingly integrated into the central activities of modern society."[43] High expectations from users increased the pressure on them, but they were unable to meet them, as they emphasized. Reading the many presentations and statements in the discussions, a recurring feeling of losing control emerges. The ever-increasing size of the programs and the ever-growing number of software programmers involved in the projects made the products prone to errors, and the errors were difficult to find. As the report summarized: "Yet this growth rate was viewed with more alarm than pride."[44] There was repeated talk of "great concern." The software programmers emphasized how overwhelmed they were with the current software development, for example: "The basic problem is that certain classes of systems are placing demands on us which are beyond our capabilities and our theories and methods of design and production at this time. It is large systems that are encountering great difficulties."[45]

Even at the beginning of the 1980s, the size and complexity of maintenance and error problems were still being discussed. It became clear how endless the work of

40 Dijkstra, "The Humble Programmer."
41 Randell, "Software Engineering in 1968," 1–10.
42 Valdez, "Gift from Pandora's Box," 178.
43 Naur and Randell, *Software Engineering*, 3.
44 Naur and Randell, 9.
45 Naur and Randell, 9.

debugging was in software development. One software engineer stated that maintenance is the "ultimate black hole."[46] It gradually became clear that programs are like cabbages, as he described the issue: "If you put them on the shelf and forget about them, they go bad." He commented satirically: "The work of the maintenance programmer is formidable: trying to find original bugs, trying to find the last maintenance programmers' bugs, trying to find the bugs he put in last week himself, making improvements for the user if there is any time left over."[47]

However, it was not just the size of the software systems ("the problem of scale"[48]) that unsettled the programmers. A lack of understanding of the programs, which was connected to their size, also prompted critical and concerned comments:

> Today we tend to go on for years, with tremendous investments to find that the system, which was not well understood to start with, does not work as anticipated. We build systems like the Wright brothers built airplanes – build the whole thing, push it off the cliff, let it crash, and start over again.[49]

It was considered irresponsible to allow software to enter society in this manner. After all, it could be a matter of life and death: "Particularly alarming is the seemingly unavoidable fallibility of large software, since a malfunction in an advanced hardware–software system can be a matter of life and death."[50] Edsger Dijkstra stated that the "massive dissemination of error-laden software is frightening."[51] A summary in the conference report mentioned "growing pains."[52] At the same time, another participant expected "an exponential growth of errors,"[53] another spoke of "obvious dangers" in programming[54] or of "serious errors."[55] At the follow-up conference in 1969, the emotional and psychological strain of producing faulty software was summed up in one sentence. One participant stated: "Programmers call their errors 'bugs' to preserve their sanity; that number of 'mistakes' would not be psychologically acceptable."[56]

By acknowledging this uncertainty and the feelings tied to it, the attendees of the conference began to navigate through it. The words often used at the confer-

46 Feeney, "Management Information System," quoted in Valdez, "Gift from Pandora's Box," 115.
47 Feeney, quoted in Valdez, 115n.
48 Naur and Randell, *Software Engineering*, 39.
49 Naur and Randell, 10.
50 Naur and Randell, 9.
51 Naur and Randell, 9.
52 Naur and Randell, 10.
53 Naur and Randell, 20.
54 Naur and Randell, 28.
55 Naur and Randell, 30.
56 Buxton and Randell, *Software Engineering Techniques*, 17.

ence, such as fear, worry, pain, serious errors, concern, problem, disaster, danger, alarm, trouble, and frightening, clearly show that the software developers wanted to confront the issue directly. They frequently emphasized that they currently had no solution for what they described as alarming and disturbing.

The presence of emotions also becomes visible in striking imagery. At the beginning of the 1970s, after the two conferences, Dijkstra remembered his feelings:

> The difficulties of programming arise partly from size. An increase in size can make a problem incomparably more difficult: one can close one's eyes and imagine how it feels to be standing in an open place, a prairie or a seashore, while far away a gib, reinless horse is approaching at a gallop, one can see it approaching and passing. To do the same with a phalanx of a thousand of these big beasts is mentally impossible: your heart would miss a number of beats out of sheer panic if you could.[57]

And further: "Underestimation of the difficulties arising from size is one of the major underlying causes of the current software failure."[58] He described the feeling of threat and loss of control vividly.

In the late 1960s, with its dominant images of confident male scientists and engineers, it is remarkable that these men spoke so openly about their worries, and fears, and even admitted a loss of control. A few, however, were not satisfied with the dominant focus on errors. While most participants, in line with Steven Jackson, practiced, so to speak, "broken software thinking" and reflected on the development of software from the perspective of errors and problems, one participant pointed out that many applications worked well.[59] Another complained that an "aura of gloom had fallen over the assembly."[60] And yet another participant asked whether they perhaps were exaggerating.[61] However, these were only a few voices. The prevailing sentiment was one of shared concern, uncertainty, and fear. The conference in Garmisch began with an evening lecture, in which the speaker admonished those present and even questioned their guilt for not having addressed the errors earlier: "so the effect that we were all guilty, essentially of concealing the fact that big pieces of software were increasingly disaster areas and we were all sitting around actually worrying internally about it and doing precisely nothing."[62]

---

57 Dijkstra, "Notes on Structured Programming," 2.

58 Dijkstra, 2; Valdez, "Gift from Pandora's Box," 198.

59 Naur and Randell, *Software Engineering*, 70.

60 Naur and Randell, 70.

61 Naur and Randell, 71.

62 Interview with John Buxton, Professor of Information Technology, King's College London, co-editor of the Rome Conference Report, quoted in Valdez, "Gift from Pandora's Box," 175. Valdez interviewed computer scientists in the context of her dissertation, which she com-

The conference now presented an opportunity to change this in a spirit of great unity, as an emotional community. Although it might seem surprising at first glance that the exclusively male software developers openly discussed their struggles with technological problems, thereby undermining the image of the confident technician or scientist, it is worth noting that two factors put the dominance of negative feelings into perspective: first, emphasizing responsibility, and second, taking action.

First, concerns, anger, fear, and uneasiness were connected to the responsibility that software developers have. The participants strongly emphasized their responsibility, referring to themselves as reliable people who had come together to solve the problem. They did not portray themselves as despondent and hopeless, but as rightly concerned and therefore responsible. This was accompanied by outrage and accusations against the industry, which was perceived, at least implicitly, as a significant part of the problem. Companies were even accused of being "fraud" because they raised false expectations among customers: "You may be right in blaming users for asking for blue-sky equipment, but if the manufacturing community offers this with a serious face, then I can only say that the whole business is based on a big fraud " (laughter and applause).[63] Industry and managers were often accused of overselling the abilities of software designers.[64] Consequently, the economic conditions that caused time pressure and fueled false expectations, putting software developers in a position where they could not meet these expectations, were denounced.

This highlights the intertwining of emotions with morality, norms, and values that defined this emotional community. The emotional debates at the two conferences involved moral and political judgments, such as frustration with economic conditions or marketing practices. Emotions are closely tied to criticism of financial pressure, as well as moral demands, responsibility, and political attitudes. Dijkstra even called for humility and acknowledgement of human limitations, both in later publications and elsewhere. He even spoke of his "small head" that limited him.[65] It becomes clear how strongly emotions were linked to norms, values, and (political) attitudes toward the world.

Second, in this emotional community, which combined concern, unease, and fears with a sense of responsibility, the need for action was highlighted. The conference attendees saw themselves as responsible and, ultimately, as action-oriented and problem-solving. The conference did not stop at simply discussing concerns together. Instead, it succeeded in transforming negative feelings into positively evaluated and promising action. Many regarded the conference as a turning point from

pleted in 1988. She conducted semi-structured interviews, from which she quotes. She does not provide any further details about when exactly these interviews were conducted.

63 Naur and Randell, *Software Engineering*, 7.

64 Naur and Randell, 71.

65 Dijkstra, "Notes on Structured Programming," 3.

not addressing problems to taking action and seeking solutions. Thus, a second set of feelings emerged that characterized this emotional community: euphoria and hope, confidence and a call for action. By beginning to formulate the problems together and exchanging ideas, solutions seemed within reach, and the concerned software developers turned into active and responsible problem-solvers. The conference was seen as a new beginning. Dijkstra put it this way: "The meeting in Garmisch Partenkirchen was very exciting. For me, it was the end of the Middle Ages. It was very sunny. The meeting was a success."[66]

Other participants later emphasized similar points in interviews. They described "an enormous atmosphere of enthusiasm; we felt we might actually achieve something and solve some problems [...]; the main thing from the first conference was that we were openly talking about the software crisis and that something needed to be done."[67] At the conference, problems were "attacked," as one participant put it.[68] There was also a sense of humor. One participant asked, quite seriously: "How many errors should we be prepared to accept in a system containing one million instructions?" Another participant replied: "Seven [laughter]."[69] Despite the significant concerns and fears, the mood was therefore positive overall: "And the result then was an immensely enthusiastic week."[70]

Thus, the emotional community of software developers at the first conference in Garmisch was characterized by a specific mixture of concerns and fears, along with an emphasis on personal responsibility. This did not lead to resignation, but rather served as the starting point for hope and the expectation of regaining control. The conference can thus be interpreted as part of a professionalization strategy of software development, which fostered the process of masculinization. Male software developers were frightened, but first and foremost, they interpreted their worries as a sense of reliability and a determination to act. They built a male community of responsible problem solvers. When the conference ended,

> the majority ... left ... with a feeling of relief, some even in a state of great excitement: It had been admitted at last that we did not know how to program well enough. I myself and quite a few others had been waiting eagerly for this moment because now at last something could be done about it.[71]

66 Interview with Edsger Dijkstra, Professor of Computer Sciences, University of Texas, formerly Professor of Mathematics at the Technical University of Eindhoven and Burroughs Fellow, quoted in Valdez, "Gift from Pandora's Box," 175.

67 Interview with Douglas Ross, SofTech Inc., formerly of the Servo Mechanisms Laboratory, Massachusetts Institute of Technology, quoted in Valdez, 176.

68 Naur and Randell, *Software Engineering*, 22.

69 Naur and Randell, 41.

70 Interview with John Buxton, quoted in Valdez, "Gift from Pandora's Box," 175.

71 Dijkstra, "On the Interplay," 3–5.

With these expectations, the second NATO conference was scheduled to take place in Rome in 1969. At the start of this follow-up conference, the "sense of urgency in the face of common problems" that had prevailed at the conference in Garmisch was reiterated.[72]

The disintegration of the emotional community one year later: Anger, frustration, hurt feelings

However, as Valdez has already demonstrated in her dissertation, the follow-up conference resulted in great disillusionment.[73] The participants emphasized that even the venue and its atmosphere had been disappointing. "The place where the conference was held was sterile – I mean it was too modern, too – it didn't bring people together with feeling."[74] It is worth noting that the participant considered feelings important for the conference and criticized the venue for not fostering an emotional atmosphere. Actually, the emotional community started to disintegrate.

The unity that formed through a shared, concerned, and anxious understanding of the problem, accompanied by a sense of relief and expectation that solutions would be found, disintegrated at the moment discussions about possible solutions began. The second conference in Rome was marked by negative feelings, culminating in mutual recriminations. It is also interesting to note that the open approach to errors was no longer welcomed so unanimously. When asked whether errors should be discussed, it was pointed out that this might be honorable, but it could be detrimental to one's career. One commentator suggested introducing a "pathology" of software into the curriculum. He compared discussing mistakes in software education to medical students learning about "disease structures." Software programmers should do the same, even if it is "very painful and unfortunately … not glamorous."[75] However, the reactions to the suggestion were muted.

What was more decisive in the disintegration of the emotional community was that it now became clear that the shared talk of software issues, problems, and even crises had different meanings for different groups of software developers. One could assume that this could be explained by the fact that the participants were not exactly the same. However, there was a significant overlap of participants. As one participant complained: "People were saying their own version in less compelling terms of what had been covered the year before."[76] Rome can be seen as typical of

72 Buxton and Randell, *Software Engineering Techniques*, 7.
73 Valdez, "Gift from Pandora's Box."
74 Interview with Douglas Ross, quoted in Valdez, 183n.
75 Buxton and Randell, *Software Engineering Techniques*, 48.
76 Interview with Douglas Ross, quoted in Valdez, "Gift from Pandora's Box," 183n.

the international software community of the time, which was not in harmony about how to program. There was a fundamental disagreement about what constituted an adequate solution for problems, which they even defined differently. Two groups emerged, who engaged in a heated debate: academic computer scientists and the so-called industrialists, i.e., software developers from an industrial background. Heavy conflicts arose over the appropriate solution strategies, and it soon became clear that this conference lacked harmony and shared concerns. The emotional community, which had united software developers a year earlier, no longer existed. Essentially, the debate centered on whether a mathematical-theoretical approach or a more pragmatic, test-based approach was the most suitable path forward. The positive atmosphere and sense of solving problems collectively as a community of software developers who shared a common concern had vanished, replaced by a dispute over the correct way to address the issue. Taking action became disintegrating. As one speaker summarized: "The truth of the matter is that we tend to look with doubt and suspicion at the other side."[77] The industrialists would say: "Well, there's nothing we can get out of computer science: look at all this rubbish they are talking about." The academics, on the other hand, said: "Goodness me; what rotten techniques they use and look: they all fail!"[78] One speaker urged everyone to ask how they could assist each other. The current situation made him "sad." The academics were "disgracefully arrogant in neglecting and denying the existence of large problems."[79]

Positions on the proper way to program were exchanged directly. As one participant recalled, people from industry felt like "monkeys to be looked at by the theorists." On the other hand, the theorists felt isolated, believing "they were not allowed to say anything."[80] The report of the Rome conference shows what can be described as hardened fronts, which were very emotionally charged and often perceived as a devaluation of one's position. The descriptions and assessments of the respective opposing positions contained terms such as "trouble," "problem," "neglecting," and "collapsing." There was an atmosphere of hurt feelings and mutual disparagement. Overall, disappointment and frustration were so overwhelming that some attendees even suggested the conference would have been better off never having taken place. One participant summed up that most of the participants were left with "an enormous sensation of disillusionment."[81]

While an emotional community had formed at the first conference in Garmisch, with the software developers united by shared feelings of responsibility, concern,

77 Buxton and Randell, *Software Engineering Techniques*, 9.

78 Buxton and Randell, 9.

79 Buxton and Randell, 8.

80 Buxton and Randell, 9.

81 Interview with John Buxton, quoted in Valdez, "Gift from Pandora's Box," 184.

and fear, which were ultimately transformed into an enthusiastic spirit of optimism and hope, this emotional community fell apart when it came to concrete problem-solving. Many negative feelings were expressed at the second conference. However, these were no longer unifying but divisive. They no longer led to a shared, euphoric strategy for action. Instead, when the discussion about the right action started, participants expressed feelings of disappointment, frustration, and a lack of recognition for their positions, which now had a separating effect. At this conference, the participants did not form an emotional community. Factions had emerged, and emotions now had a divisive rather than a unifying effect. Thus, emotions were strongly unifying at the first conference and firmly separated the participants at the second conference.

Conclusion

The all-male software developers were very emotional at both conferences. It is remarkable, first, how clearly and openly feelings of fear, worry, and loss of control were expressed at the first conference, seemingly counteracting the image of a confident, male, in-control software developer. What is striking is the relief felt by many participants in finally being able to discuss the problems, their emotions, and concerns they faced. However, this admission of fear, uncertainty, and discomfort was linked to an emphasis on personal responsibility. Expressing concern thus became part of a sense of responsibility that stylized feelings of anxiety as part of a solution. The openness with which male participants spoke about their fears and concerns, which seemed unusual at first glance, was quickly linked to the great responsibility they bear, which obliges them to address the problems and to express their worries, above all, to find solutions. In this way, negative feelings of fear and concern were transformed into positive emotions. By the end of the conference, these men were confident that they could regain control and resolve the issues. It was thus clear that the software developers were not despondent critics of technology, but saw themselves as responsible technicians who were raising serious concerns and seeking solutions.

Second, the debates on how to solve the problems were part of a male-dominated professionalization strategy. The fact that only male participants were invited effectively meant that women were excluded from a conference where the solution to software problems was discussed openly and in depth. This was in line with the contemporary "recoding" of gender, the attribution of "important topics" as male concerns, and the associated marginalization of women in the software development field.[82]

82 Abbate, *Recoding Gender*, chap. 3; Ensmenger, "Making Programming Masculine."

Third, the 1968 NATO conference represents a brief moment in history where a temporary, very short-lived emotional community openly discussed problems, errors, malfunctions, and potential failure and, importantly, reflected on them emotionally. The intertwining of emotions and reflection aligns with key insights of the historiography of emotions, which has criticized the constructed opposition between emotions and rationality. During the first conference, reflection on software issues became possible within a setting where the same emotions were shared. In any case, technology developers were not purely rational actors, confident in their work and decision-making. They shared their concerns, fears, and uncertainties openly. They expressed their worries, highlighted their sense of responsibility, and sought solutions directly. This was especially evident at the second conference, where they seemed offended, frustrated, and disappointed; they felt somewhat belittled and disrespected during debates about the best solution. Thus, at both meetings, the crucial role of emotions in technology development was evident.

Do the fears expressed by technology developers and their emotional debate about the best solution strategy in the early days of software development now hold greater historical importance? Was this a short and temporary phase of openness about errors, potential failure, and uncertainties that over time gave way to a sense that things were more or less under control? It is undoubtedly the case that fear, concern, and uncertainty have diminished due to the various error control strategies in software development. Without doubt, it is still not possible to produce error-free software systems, but it is possible to produce more or less dependable software.

Nevertheless, the early phase of programming clearly shows that the image of euphoric and self-assured technology developers enthusiastically contributing to progress needs a more nuanced view. Today, in the realm of AI, discussions about its limitations, unreliability, and errors are resurfacing. Warnings are also being expressed in the AI context. Some of these warnings come from Silicon Valley heroes themselves, while others are from well-known AI researchers.[83] However, the hype about AI's potential and successes remains dominant. It's essential to consider the concerns and fears that software developers expressed early on, as this helps to differentiate between the perception of an overly optimistic culture in Silicon Valley and the obvious problems and failures surrounding AI. A culture of risk-taking and embracing failure may prevail in Silicon Valley. However, focusing on these media-savvy, loud, and self-congratulatory project leaders obscures essential questions of how to handle fallibility and uncertainties in technology development – topics that are also filled with concern, fear, and caution. Discussing errors and failures in a way that emphasizes failure not only as a means of success but also as a possible danger

83 United Kingdom Government, *The Bletchley Declaration*; Metz, "Godfather of A.I."

paves the way for a more realistic societal debate. After all, errors, malfunctions, failures, and fears are all part of the digital age.

Epilogue: The digital age and anxiety?

Neta Alexander stated that anxiety "is the dominant reactive affect of the 21st century."[84] She references a manifesto-like essay entitled "We Are All Very Anxious," written by a British collective of artists, scholars, critical thinkers, and anti-capitalists called Plan C.[85] The collective explains the shift from the Fordist system to the present day. With full-time jobs, welfare, job security, and mass consumption, the Fordist era was characterized by a sense of boredom. Jobs were simple, repetitive, and dull. This is no longer the case: "In contemporary capitalism, the dominant reactive affect is *anxiety*."[86] Anxiety is the basis of our attitude toward life, as the collective claims.

From a historical perspective, this dichotomous historical caesura is clearly too simplistic and focused on the Global North. Still, it raises a compelling question: What role does anxiety play in digital societies? A German newspaper recently reported that 16% of the German population would like to see digitalization reversed, and 37 % often feel overwhelmed. A study highlighted the numerous issues people face when using technology correctly, updating it, and maintaining its functionality.[87] When dealing with technology, many often feel uncertain, anxious, and frustrated. Determining whether it is appropriate to call it an "age of digital anxiety" requires more research. Understanding the feelings of technology developers is an essential part of exploring what role anxiety plays. Digital anxiety is a significant social issue that does not receive enough attention.

Bibliography

Abbate, Janet. Recoding Gender: Women's Changing Participation in Computing. The MIT Press, 2017. https://doi.org/10.7551/mitpress/9014.001.0001.

Alexander, Neta. "Rage against the Machine: Buffering, Noise, and Perpetual Anxiety in the Age of Connected Viewing." *Cinema Journal* 56, no. 2 (2017): 1–24. https://dx.doi.org/10.1353/cj.2017.0000.

Appadurai, Arjun, and Neta Alexander. *Failure.* Polity, 2019.

84 Alexander, "Rage against the Machine," 22.

85 Weareplanc, "We Are All Very Anxious."

86 Weareplanc.

87 Herbold, "Speicher voll."

Asghar, Rob. "Why Silicon Valley's 'Fail Fast' Mantra Is Just Hype." *Forbes*, July 14, 2014. https://www.forbes.com/sites/robasghar/2014/07/14/why-silicon-valleys-fail-fast-mantra-is-just-hype/.

Berth, Christiane. "Zwischen Hoffnung, Stolz und Wut: Die emotionale Aneignung des Telefons in Mexiko, 1930er bis 1980er Jahre." In *Technikemotionen*, edited by Martina Heßler. Schöningh, 2020. https://doi.org/10.30965/9783657703456_011.

Buxton, John, and Brian Randell, eds. *Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee, Rome, Italy, 27th to 31st October 1969*. NATO Science Committee, 1970.

Corn, Joseph. *User Unfriendly: Consumer Struggles with Personal Technologies, from Clocks and Sewing Machines to Cars and Computers*. Johns Hopkins University Press, 2011.

Dijkstra, E. W. "On the Interplay between Mathematics and Programming." In *Program Construction*, edited by F. L. Bauer and M. Broy. Springer, 1979. https://doi.org/10.1007/BFb0014653.

Dijkstra, E. W. "Notes on Structured Programming." In *Structured Programming*, edited by O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare. Academic Press, 1972.

Dijkstra, E. W. "The Humble Programmer." Manuscript EWD 340. E. W. Dijkstra Archive, University of Texas at Austin. Reprint of article published in *Communications of the ACM* 15, no. 10 (1972): 859–66. https://www.cs.utexas.edu/ EWD/transcriptions/EWD03xx/EWD340.html.

Ensmenger, Nathan. "Making Programming Masculine." In *Gender Codes: Why Women Are Leaving Computing*, edited by Thomas J. Misa. John Wiley & Sons, 2010. https://doi.org/10.1002/9780470619926.ch6.

Ensmenger, Nathan. *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*. The MIT Press, 2010. https://doi.org/10.7551/mitpress/9780262050937.001.0001.

Feeney, J. M. "Management Information Systems – The Failure of Technology." In *Business Information Systems*, Infotech State of the Art Report 9, no. 7. Pergamon Infotech, 1981.

Geuenich, Michael. "'… gibt es auch mal ein Küßchen auf das Lenkrad': Anthropomorphisierung von Technik und die fragile Black Box Automobil." In *Technikemotionen*, edited by Martina Heßler. Schöningh, 2020. https://doi.org/10.30965/9783657703456_013.

Haigh, Thomas. "'Crisis, What Crisis?' Reconsidering the Software Crisis of the 1960s and the Origins of Software Engineering." Paper presented at the Second Inventing Europe/Tension of Europe Conference, Sofia, June 17–20, 2010. https://www.tomandmaria.com/Tom/Writing/SoftwareCrisis_SofiaDRAFT.pdf.

Herbold, Astrid. "Speicher voll." *Die ZEIT*, no. 35/2025, August 17, 2025. https://epaper.zeit.de/article/124e6251e0618cf51faabd202fcec177412a73eb23ed0d62c9402a2cf3d9fa6f.

Hicks, Mar. "Sexism is a Feature, not a Bug." In *Your Computer is on Fire*, edited by Thomas S. Mullaney, Benjamin Peters, Mar Hicks, and Kavita Philip. MIT Press, 2021. https://doi.org/10.7551/mitpress/10993.003.0011.

Jackson, Steven J. "Rethinking Repair." In *Media Technologies: Essays on Communication, Materiality and Society*, edited by Tarleton Gillespie, Pablo Boczkowski, and Kirsten Foot. The MIT Press, 2014. https://doi.org/10.7551/mitpress/9042.003.0015.

Licklider, J. C. R. "Underestimates and Overexpectations." *Computers and Automation* 18, no. 9 (1969): 48–52.

Light, Jennifer S. "When Computers Were Women." *Technology and Culture* 40, no. 3 (1999): 455–83.

MacKenzie, Donald, "A View from Sonnenbichl: On the Historical Sociology of Software and System Dependability." In *History of Computing: Software Issues*, edited by Ulf Hashagen, Reinhard Keil-Slawik, and Arthur L. Norberg. Springer, 2002. https://doi.org/10.1007/978-3-662-04954-9_9.

Metz, Cade. "'The Godfather of A.I.' Leaves Google and Warns of Danger Ahead." *New York Times*, May 5, 2023.

Misa, Thomas J., ed. *Gender Codes: Why Women Are Leaving Computing*. Wiley; IEEE Computer Society, 2010. https://doi.org/10.1002/9780470619926.ch1.

Nakamura, Lisa. "Plug and Pray: Performance of Risk and Failure in Digital Media Presentations." *The Velvet Light Trap* 64 (2009): 87–89.

"NATO Science Committee." *Nature* 181 (1958): 739–40. https://doi.org/10.1038/181739b0.

Naur, Peter, and Brian Randell, eds. *Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968*. Scientific Affairs Division, NATO, 1969.

Norman, Kent L. "Computer Rage: Theory and Practice." Presentation, HCIL-V and the Department of Psychology, October 21, 2004. Archived at Internet Archive (Wayback Machine), November 19, 2015. https://web.archive.org/web/20151119181900/http://129.2.36.150/trons/hcil22oct2004/hcilbbl_10_22_2004.pdf.

Parnas, David. "Software Aspects of Strategic Defense Systems." *Communication of the ACM* 28, no. 12 (1985): 1326–35.

Petroski, Henry. *To Engineer is Human: The Role of Failure in Successful Design*. Vintage, 1992 (First published 1982).

Randell, B. "Software Engineering in 1968." In *Proceedings of the 4th International Conference on Software Engineering (ICSE '79)*. IEEE Press, 1979.

Reddy, William M. *The Navigation of Feeling: A Framework for the History of Emotions*. Cambridge University Press, 2001. https://doi.org/10.1017/CBO9780511512001.

Rosenwein, Barbara H. *Emotional Communities in the Early Middle Ages*. Cornell University Press, 2006.

Slayton, Rebecca. *Arguments that Count: Physics, Computing, and Missile Defense, 1949–2012*. MIT Press, 2013. https://doi.org/10.7551/mitpress/9234.001.0001.

United Kingdom Government. *The Bletchley Declaration by Countries Attending the AI Safety Summit, 1–2 November 2023*. Policy paper, updated February 13, 2025. https://www.gov.uk/government/publications/ai-safety-summit-2023-the-bletchley-declaration/the-bletchley-declaration-by-countries-attending-the-ai-safety-summit-1-2-november-2023.

Valdez, María Eloína Peláez. "A Gift from Pandora's Box: The Software Crisis." PhD diss., University of Edinburgh, 1988.

Weareplanc. "We Are All Very Anxious." *Plan C* (blog). April 4, 2014. https://www.weareplanc.org/2014/04/we-are-all-very-anxious/.

Wolf, Burkhardt. "Schiffbruch mit Beobachter. Zur Geschichte des nautischen Gefahrenwissens." In *Die Unordnung der Dinge: Eine Wissens- und Mediengeschichte des Unfalls*, edited by Christian Kassung. transcript, 2009. https://doi.org/10.1515/9783839407219-002.