AESTHETICS AND PROGRAMMING

Prof. Dr. Peter Molzberger
University of the German Armed Forces, Munich

Abstract

The paper at hand is based on interviews with a total of eight so-called "superprogrammers", software people, who show exceptional performance quantitatively as well as qualitatively. It becomes apparent that these people do not experience programming as a purely rational activity, but that for them it possesses strong intuitive components.

Programs are visualized wholistically as three-dimensional structures. In this, aesthetics plays a special part: the structure must please optically, be elegant -- then it is functionally acceptable. Logical mistakes manifest themselves as interfering with this aesthetics.

The author suggests that in the area of software as well there is something like the absolute beautiful: perfect solutions with a maximum of transparence beyond all rivaling design parameters. He has a feeling that the faculties described in this paper are widespread and may open up a totally new dimension in programming.

A Philosophical Introduction: What is Aesthetics?

The word aesthetics comes from the Greek language and means "perception". In the initial sense, a plastic (spatial) perception is meant by that. In a very similar way the word "to grasp" contains a reference to something wieldy, therefore spatial.

In his work, "Critique of Pure Reason" /1781/, Kant states in the end that pure reason is "architectonic", that y is, three-dimensional, plastic. Our thinking seems to take place in three-dimensional (four-dimensional, if you add time) pictures. As a rule, this is not conscious to us, that is, we do not know what our thoughts look like.

We are used to giving the attribute "beautiful" to a program, if we like it (and by that we do not at all mean the optic impression of its printout on a piece of paper). Is this purely a subjective judgment? Without doubt our aesthetic feeling is individually different and to a large extent dependent on culture. On the other hand, like a red thread the idea runs through the occidental tradition of thinking -- starting with Plato and Aristotle -- that beyond this, there had to be something like the "absolute beautiful": there is an original ability of all people to spontaneously differentiate between "beautiful" and "ugly". Why should this not be true for software as well? Is there not beyond all programming languages, individual styles and fashion trends, something like the absolute beauty or elegance in programming? And the beautiful is -- according to another philosophical idea -- at the same time the good, the true. This age-old thesis as well seems to prove itself surprisingly, as we will see, within the framework of our modern software development.

1. "Strange People Doing Strange Things."

In his guest speech, at the first conference of this kind in Gaithersburg*, Weinberg urged the participants: "There are strange people somewhere, doing strange things. Look for them." The following true story popped up into my mind:

In 1977/78 my company, a software house, had an order to develop a complicated real time system which had to be written for the most part in assembly language. When the tasks of specification and design were completed, the project had fallen far behind in time. A catastrophe was threatening. Among the five assembler programmers of the project team there was one who one night sat down at the terminal, got glassy eyes, and slipped into a mental state in which he could not be talked to any more.

The next morning he had completed a difficult piece of code. This event happened repeatedly and within six months the man wrote 45.000 assembler statements and 10.000 macros! Among his colleagues the man was called the "trance programmer". He once commented to me: "You could fire a cannon next to me and it would not bother me." His coding was written adhering strictly to the rules negotiated in the team: an exemplary, that is, a well-balanced,

*Conference on Human Factors in Computing Systems, March 15-17, 1982, Gaitherburg, MD.

cleanly and efficiently constructed product. What was the most amazing, however, was that the system with more than a hundred terminals kept operating at full capacity for months without a breakdown.

Weinberg strongly encouraged me in a private talk to follow up on the incident of the "trance programmer". That motivated me to do a study interviewing programmers I had noticed in the last ten years of my vocational practise as being extremely efficient. What came out of it was fascinating for me, because it contradicts the current concept of programming in every way /MOLZBERGER 1983/. In the following paragraphs I will especially concern myself with one aspect, which turned up in the interviews time and again, the role of aesthetics in programming. The reproduced dialogues are excerpts from original notes (in German).

2. What Does a Program in the Programmer´s Mind Look Like?

What does a program look like, which we can imagine so clearly that we can find mistakes in it? This question, asked in a circle of students, caused confusion. Some believed to have seen strings of characters, which caused hilarity in others. Others made use of their hands to draw loops in the air. Nobody knew for certain! Even our mental images, which we work with daily, are not conscious to us! From top programmers, I got the answers to the question of what they were visualizing in the designing of a program, like:

Edwin: "For me, a program is a three-dimensional structure of stairs around which I can walk and into which I can enter." (Figure 1)

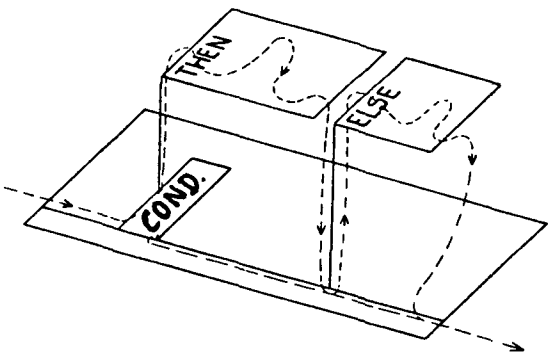

Figure 1: Representation of "IF - THEN - ELSE" as a three-dimensional structure structure

Edwin can do even more with his three-dimensional structures: He can test them by moving through them. He claims to "exist twice at the same time":

(a) in a position outside the program which allows for keeping the general view.

(b) in a movable position inside the coding. "I myself am the processor. I become a point. That is how I run through the program: through loops, jumps, etc. I execute the program. Afterwards I am completely sure that the program is correct. I cannot be wrong, because it has been carried out correctly!"

Another very good programmer spoke of a symphony that he composes when he designs a program. Plastic impressions has Georg, when he says:

Georg: "I am like a sculptor or a potter. I design something! But I cannot see any pots. Conceptualizing a program is not an intellectual, but an emotional performance for me. My difficulty is to express in words what I design. I have problems furnishing an understandable description."

Question: "Do you see statements perhaps?"

Georg: "No. There are no statements. Statements only irritate me. The program is a whole! When you think in statements you lack continuity."

3. Artistic ´State of Consciousness´

Phases of extreme creativity and concentration were marked in the interview partners by symptoms, which deviated distinctly from the normal state. Rather uniformly reported were:

- strongly reduced need for sleep and food
- changed subjective concept of time (up to the factor ten)

On the other hand, there occurred marked differences in regard to the times necessary for reaching the state and in the reactions to interruptions. Thus Georg says:

"Interruptions can be unfortunate. It can take hours after a telephone call! When I am interrupted at an unfavorable point where many threads run together and I am not completely finished yet, I have to start all over. I don´t believe that it has anything to do with a trance. It is simply extreme concentration. It is true that I am in another state of consciousness in doing this. I call this state ´programming´."

Contrary to this, Hans, a freelance consultant, claims:

"When I am interrupted during work I need only a few seconds to switch my state of consciousness around. For this activity I use the expression ´push to stack.´ Indeed this activity has a baffling similarity with the interrupt-handling of the microprocessor 8080."

The duration of the state seems to be varied as well; for all ages, too. Thus the 40-year-old Georg reports:

"Today I hold the state for 12-15 hours, in earlier times up to 36 hours, in which I do not burn myself out."

Obviously, there is a need for all those interviewed to get into such a creative phase from time to time. The state seems to be something that has a very positive effect on the general contentedness with life.

Michael: "I have to like the solution, it has to be aesthetic. I cannot describe what that means. A feeling inside of me has to be satisfied. That is also why I do all that, not for money in the first place."

About feelings during work, however, quotes like this one are more likely:

Question: "Is this state pleasant?"

Georg: "It is nothing at all! Emotions are switched off! I like to do this type of thing, because I see the success. The depth of that state depends on my interest. I become the problem and machine at the same time! I am both simultaneously."

4. Mistakes as Disruption of the Aesthetic Feeling

Logical errors in programs are experienced time and again as a disruption of the aesthetic harmony. Logical correctness and good solutions manifest themselves in aesthetic elegance.

Edwin: "It has to result in an aesthetic picture. If I don´t like it aesthetically, I know that the program will not run. The emerging structure is perfect then. I know that it is completely free of mistakes. I do not write the program down until it is perfect."

Georg: "Before I find a mistake I become aware that something is wrong with the aesthetics. I work very essentially with aesthetics."

It was also remarkable that with several programmers the sleeping phase and their dreams play a role. For instance:

Paul: "When I have been looking in vain for a while for a mistake, the thing takes possession of me. If I don´t find the mistake there is a point at which I know I should stop the thing. When I sleep afterwards and wake up, I have found the mistake. It is simply there!"

Georg, too, uses his aesthetic feeling in a calculated way to test programs, when he says:

"What is, well-designed is, at once aesthetical, that is, elegant, optimal and intelligible. When I see such a program I simply know that it is ´waterproof´. The feeling of familiarity does not only hold true for one´s own programs, but generally when a program is well-written. There are programs which give me stomach aches at first sight. They appear unfamiliar, although they may be correct. I rely heavily on my emotions! There are programs which practically cannot be tested (for instance interrupt control on SPL). There I simply have to look at them and know that they are o.k. When I see that, I´m convinced of their correctness."

The notion that the program is perfect in the mind was voiced by several programmers. Mistakes come about in the translation into the statements of a programming language, in destroying the wholeness. This ability was labeled by Georg first as "spooling out".

Question: "Do you mean that it is pure routine work?"

Georg: "What I have called spooling out of a program is not dumping off, but a highly concentrated, creative occupation. It is true that I have completely finished the basic structure of the program beforehand. But the flesh has to be put in still."

5. Programming is Beautiful

The key question is whether there is such a thing as the "absolute beautiful" in software as well: programs which (respecting rivaling parameters of interpretation and fully fulfilling the desired functions) unite two demands which seem paradoxical:

- the absolute maximum of clarity (and with it the possibility of maintenance)
- the unmistakably personal flair of a great artistic creation

I believe that such a thing is indeed possible. Let us remember Georg´s statement: "The feeling of familiarity does not only hold true for one´s own programs, but generally when a program is well-written."

There is certainly something that a person familiar with art intuitively understands as a statement in a work of art, independent of the individuality of the artist. I claim that the same thing is true for a really good program: It contains elements, beyond the individual style, which allow the expert to grasp essential parts intuitively.

I found remarkable parallels in an old neighboring field, mathematics. It is reported again and again that after someone´s long and intensive concern with the subject, the theorem is there first -- at times in geometrical shape and as an aesthetically faultless structure. And only then the hard work of proving begins. According to Hamming /1980/ the speculation exists that more than half of the 200.000 theorems published annually are substantially correct, although their proofs are wrong!

In his article published as early as 1914, "Mathematical Invention", Poincare describes many of the elements which turn up in my interviews: "intuition", "feelings of absolute certainty", "sudden illumination", "feelings for mathematical beauty." The process -- stated in a lecture at the Societe de Psychologie in Paris -- is quoted by Arthur Koestler /1960/ in his documentation "The Art of Creation". In the same source, references to similar experiences by Hadamard, Ampere, Polya, and Gauss can also be found.

Conclusion

In discussing a great number of topics, especially the results of the interviews, with software people, I could markedly differentiate three different kinds of reactions:

1. Positive: great interest up to enthusiasm. Recommendations for a utilization of the results, especially in education.
2. Negative: "Cases like those are, if they are true at all, exceptional. Our problems lie not with the top man, but with the average programmer."
3. "So what": Surprisingly I met a series of software people for whom the statements were completely self-evident throughout the interviews. They could not imagine at all, how someone could program well without commanding such abilities.

This arouses the suspicion in me that the abilities are by no means commanded only by some strange people, but that they are -- similar to the ones of mathematicians -- widely spread and perhaps easily learned (or activated) by many people. Because they oppose the current paradigm of software engineering, (to get rid of the artist), these capabilities have not been discussed openly until now. The strong emotions accompanying these reactions make me feel that the time has come to open up the non-rational side of programming as a source of effectiveness and reliability.

References

Hamming, R.W.: The Unreasonable Effectiveness of Mathematics. The American Mathematical Monthly. Vol. 87, 1980.

Kant, I.: Kritik der reinen Vernunft. Königsberg 1781.

Koestler, A.: The Act of Creation. New York 1966.

Molzberger, P.: Und Programmieren ist doch eine Kunst, in: Schelle, H.; Molzberger, P.: Psychologische Aspekte der Software-Entwicklung. Oldenbourg, Munich 1983.

Poincaré, H.: Die mathematische Erfindung. Leipzig 1914, reprinted in: Ulman, G. (ed.): Kreativitatsforschung. Köln 1973.