The Role of Aesthetics in Code: a Qualitative Interview Study with Professionals

Mohamed Chahchouhi
Hani Al-Ers
M.Chahchouhi@hhs.nl
H.Al-Ers@hhs.nl
The Hague University of Applied Sciences
The Hague, Zuid-Holland

Felienne Hermans
f.f.j.hermans@liacs.leidenuniv.nl
Leiden Institute of Advanced Computer Science
Leiden, the Netherlands

ABSTRACT

A lack of relevant research exists on the topic of aesthetics in programming. This study aims to lay a foundation for future research on the topic by qualitatively researching the relation between functionality and aesthetics in code. Eight programmers from the Netherlands were interviewed for their opinion on the beauty in the code they come across in their day to day life. The answers given by the interviewees were generally convergent. The definition of beauty seemed to change based on the time the code was written in and the nature of the problem that was being solved. Visual properties mainly focused around readability and modifiability like cleanness, compactness, structure, and modularity were seen as important tough often not seen as the true definition of beauty. Beauty in code was mostly influenced by the expertise a programmer had in a language and the level of understanding of the problem that is solved. Information gathered in this study suggests a strong link between aesthetics and functionality when reading and writing code. Judging from the interviews the topic of aesthetics in code seems to be alive and relevant in the programming community and can thus be seen as a promising direction for future research.

KEYWORDS

Aesthetics, Programming Code, Beauty in code, Ugly code, Clean code, Smelly code, Aesthetics Versus Functionality in Code

ACM Reference Format:
Mohamed Chahchouhi, Hani Al-Ers, and Felienne Hermans. 2018. The Role of Aesthetics in Code: a Qualitative Interview Study with Professionals. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In a technical domain that is mainly focused around functionality, personal values like aesthetical preference are often overseen. This is the case with the field of programming. Once a program is running with the intended purpose the goal is reached, no matter how well or poorly the code is written. However, in today's world, we find ourselves working on programs that are running indefinitely and constantly changing. The code that is written will be read and modified by different people in different time-frames. Programmers with different backgrounds and expertise in programming will be reading each other's code and this, in turn, will result in divergent opinions on the code of one other.

The occurrence of an aesthetic experience when reading or writing code is quite common[5]. Programmers themselves have rated aesthetics in code important, mainly when it comes to their own code[5]. Some of the frequent answers on which programmers rate aesthetics in code included Efficiency, Elegance, Familiarity, and Personality[6].

Remarkably, most of these criteria can also be found in another, more explored, scientific field namely, mathematics[2]. This is interesting because every field normally has its field-specific terms to indicate a product of that field is aesthetic. For example, the only overlapping terms to describe aesthetics over eight different fields were 'beautiful' and 'ugly'[1].

In mathematics, some reoccurring criteria that were associated with aesthetics are 'simplicity, 'understandability', and 'interesting topic'[1]. A characteristic that made a mathematical solution less beautiful was a repeating pattern[2].

Research on the functionality of programming code tells us that beyond the fact that the code has to be able to fulfill its intended purpose it also has to meet certain criteria like readability, maintainability, modifiability, and efficiency [3][7]. These last criteria are mentioned all over the internet when searching for a definition of 'good' code, like on big programming forums like *stackoverflow.com* [7].Code that meets all these requirements is known by many programmers as code that is "clean" . Interestingly most of these criteria are in common with the criteria for code to be aesthetic[5].

While a quantitative approach to aesthetics in programming code has been investigated and discussed before , little is still known about the personal opinions behind these aesthetic preferences. Why, for example, is code that is easy to modify associated with aesthetics, why is code that is familiar to the programmer more beautiful than code that is not and why are the requirements for aesthetics in code mostly in common with the requirements for writing clean code.

In this article, we will take a closer look at the underlying aesthetic preferences in relation to the functionality of programming code that form these opinions by answering the question: To what extent do programmers differentiate between aesthetics and functionality in code?

2 METHODOLOGY

2.1 Aim of the study

This study aims to shed light on the importance of aesthetics in code. Because of the lack of relevant research on the topic, a qualitative approach is chosen to serve as a foundation for future research on the topic.

2.2 Research Method

The information in this article was gathered by conducting individual interviews in which eight programmers were asked about their opinions on aesthetics compared to the functionality of programming code. All interviews were recorded, performed and verbatim transcribe. Unstructured open interviewing was used to ensure the interviewee would give unaffected answers. Respondents were contacted at least one week before the interview and were provided with a document that served as a preparation for the interview. The document consisted of descriptions of terms that were important during the interview and a general overview of the structure that the interview would follow.

2.3 Participants

The study was performed on eight (ex)programmers partly recruited using the snowball sampling method [4] in which the respondent recommends future participants. Respondents included had to work or study computer programming in the Netherlands. All interviews were conducted in Dutch. The experience level of the participants varied from beginners to experts, people with more than ten years of serious involvement in the field.

2.4 Data analysis

The data collected from the interviews were analyzed according to the grounded theory method of data analysis [8]. Collected data were organized into different themes, which were then determined on relevance to the topic. Parts deemed not relevant were crossed out. This was followed by an analysis of the remaining data consisting of three steps: open coding, axial coding, and selective coding. Open coding consisted of simultaneously listening and reading the interviews and an in-depth analysis of the data. Followed by labeling the data based on their contents and joining synonyms together. In axial coding, the labels are categorized based on similarity in sub-topic. In selective coding, the categories were put together to identify the core categories.

Participant	Gender	Age	Expertise
Bart	M	49	25
Moemen	M	22	3
Anonymous	F	35	17
Mohammed	M	30	14
Xander	M	45	20
Ton	M	72	45
Dinesh	M	31	14
Erik	M	45	20

3 RESULTS

Even though the eight respondents interviewed had different backgrounds and experience, it was quite easy to group their statements under common code labels. There was quite a bit of convergence in what they said, but each person added a nuance to what was already mentioned. Based on the eight interviews, the following three categories were developed:

- Different definitions of beauty
- Clean code
- The solution

3.1 Different definitions of beauty

In the interviews, a reoccurring answer seems to be given: There are different definitions of beauty in code. One factor that seemed to play a role in the beauty of code is the time that it is written in. Where twenty years ago there was a big emphasis on optimization of code in the form of memory usage this emphasis seems to be non-existent in today's world because of the use of much better CPU's. This constraint in memory affected the view on aesthetics of programmers twenty years ago. This is what Mohammed, programmer and the CEO of a software engineering company, said: *"In the past, for example, we found it beautiful when code used as little memory as possible. In that time your CPU might have had thirty megahertz which simply meant you had to be very thoughtful when writing code. It was almost a competition to see who could cram the most functionality in as little memory as possible and such pieces were beautiful to look at. I no longer experience that kind of beauty today because most systems have a minimum of sixteen gigabytes of memory making memory optimization unnecessary for most programmers."*

This change in programming has been verified by the majority of the senior programmers including Ton, a retired programming teacher that has been programming his whole life. He said the following: *"If a present-day programmer would read the code I would write back then he would definitely get confused. So much has changed in the world of programming since then. A lot of things don't make sense to do anymore."*

3.2 Clean code

In the desk research done on the aesthetics in code, it became apparent that the criteria for writing clean code most of the time equaled the criteria that made code beautiful. This was also noticeable in the interviews. All interviewees found clean code or parts of it i.e. readability, maintainability, good commenting, etc., to be beautiful or at least nice to look at. However, when explicitly asked whether clean code was beautiful the participants gave interesting answers. Bart, a programming teacher at the University of Groningen, had the following answer: *"Yes and no, I think that code must be clean before it can even be regarded as beautiful"*. He later added that clean code was pleasing to look at but that beauty in code also had a deeper meaning that lies in the solution of the problem. In Bart's eyes, the cleanness of the code is rather a condition before code can even be regarded as beautiful. This aspect of beauty, or in his word's neatness, could be observed by most programmers and even people that never learned how to code. This was convergent to the answer of given by one of the interviewees that wished to remain anonymous. When she was asked about her perception of beauty in code, she, like the deeper meaning of beauty Bart was talking about, said that beauty in code contained different levels. She was talking about one of the levels when she said this: *"If you pick someone of the street that never coded before that person could still point out where the duplicates are in a chunk of code, thus on a different level of beauty than for example the algorithm used in code"*

These answers tell us that clean code definitely contributes to aesthetics in code but is not precisely the part that made the code itself beautiful. This can be compared to this example of a room. The cleanness of the code can be seen as the cleanness of the room. Let's imagine a room that is beautifully furnished. By many, this room is seen as beautiful. Now imagine the same room only when it is badly maintained. Clothes lying around, leftover food that has not been cleaned up and the furniture is moved around to a point where it does not make sense. The same beauty from before is now barely noticeable. The same can be said of code, one might be writing code that is incredibly beautiful but if you leave "print f" statements lying around, don't use proper indentation or cluster functions together that have nothing to do with each other not much of the original beauty is left to be seen. Another benefit of the well-maintained room is its functionality. Working in a room that is a mess often leaves you distracted from the initial goal you wanted to reach. You will instead end up cleaning the room. The same goes for code, this is what Mohammed said: *"The more readable code is the easier it is to maintain. Let's say my code is a mess and there's a colleague who works on the code after me. If my messy code contains a bug and he tries to debug it he won't understand anything of what I wrote. What he will probably end up saying is 'Sorry Mo, but I'm rewriting the code so I can understand what is being said"*

He later explained that one of the benefits of clean code was that it makes the life of a programmer easier. This, according to him, also influences the perception of a programmer on code. This opinion is shared by the anonymous interviewee, here's what she said: *"What I think is that it costs more effort to read it. Because one form of beauty is that it costs the reader low effort to understand."*

3.3 The solution

In the interview with the anonymous interviewee the following theory was presented by her: *"Readable and reusable- or 'clean' code could be referred to so often by programmers because it's the only word that the general programmer knows when it comes to talking about code. Most programmers never learned how to look at code from an aesthetic point of view."*

She later adds that most programmers don't learn about beauty in code in the first place. She later adds that little focus is set on the aesthetic side of code. Classes that focus on aesthetics, like reading and discussing each other's code, are rarely given at schools. Much emphasis is put on writing code that does not strictly contains a good solution but much rather a solution that is "clean" and thus understandable for future colleagues. Bart says something similar when he is talking about the way he looks at the coding work of his students: "The first thing I do is check whether the code compiles, then I look at whether it does what it should do and lastly I check whether it's clean or not."

Bart tells us that he looks at the solution before looking at the cleanness of the code, though he is not talking about the beauty in the solution but rather about whether the code achieves the purpose it's written for. Bart, in the following sentence, gives more insight into what beauty in code looks like for him: *"One could also look a level above that like whether for example it's a creative, exciting or even funny solution and that is a really fun experience. Usually, you won't see that in the work of first-year programming students, but in the work of last year students I sometimes think this is a solution I wouldn't come up with, and there is an aesthetic moment in that event."*

Bart tells us that he finds beauty in the specific way the solution to a problem is presented. This can be done in many ways which in turn brings up different emotions like excitement or joy. The opinion that beauty lies in the solution is shared by many of the participants among which Xander. Xander is a veteran programmer that researches improving the quality of code. He said something interesting about beauty in code: *"Before I started working at the company I currently work for, I once read code that was chaotic and completely unreadable. But when reading it I did think the code was beautiful, somebody wrote a smart solution. It was almost like a piece of art where you had to look at it for some time to see the beauty."*

Xander explains that he, too, thinks that code does not have to be clean or visually appealing to be aesthetic. He even goes as far as telling us that code can be ugly, from a visual perspective, but still be beautiful because of how smart the solution to a problem is presented. When asked to be a bit more specific about the beauty of the solution he was talking about Xander added that it's almost like an admiration for the work. Factors like the proper usage of the standard functions in a language, uniqueness and an element of surprise played a role, this is what he said: *"What he did was alternately addressing memory locations. To do that you need to have good knowledge about the specific problem you're trying to solve since it was a somewhat obscure instruction in that programming language. By doing this, commands could be written very concisely, and could be executed very quickly. When reading that I taught, wow somebody did something very special here, this is really beautiful."*

As Xander noted good knowledge about the problem and notable expertise with the language that is used to answer the problem play a big role in the beauty of the solution. This answer is similar to what Bart is trying to say. Let us remember what Bart said earlier, specifically the last sentence: *"Usually you won't see that in the work of first-year programming students, but in the work of last year students I sometimes think 'this is a solution I wouldn't come up with', there is certainly an aesthetic moment in that event."*

Bart makes a clear distinction between first and last year programming students. The expertise the students gained in the four years of programming had a positive impact on the aesthetic appearance of their code. This could be because of what Xander said, an experienced programmer is more likely to find an effective and often special solution to the problem he or she is trying to solve.

Other than the expertise of the writer, the expertise of the reader also plays a role in the aesthetic opinion of a programmer. This is what Dinesh, a teacher, and retired banking programmer, said about one of his ex-colleagues (for privacy reasons the name Jeff is picked): *"Jeff's code was complex because it made too much use of the possibilities the tool had. Through the years tools further developed,* *Jeff liked to follow these developments very closely and thus used the newest functions. By using Jeff's coding style only one line was needed instead of six for example. With Jeff's method, the solution was suddenly there. In my solution, you could see what's happening more clearly."*

Because of the differences in knowledge about the utility's a tool contained it was hard for Dinesh and his other co-workers to follow what Jeff was coding. Dinesh did have a certain degree of respect for Jeff which was apparent by the phrase: *"I had a colleague who probably was 15 times smarter than I was."*

Despite the respect Dinesh had for Jeff and his method of coding he did not find it practical nor beautiful. Something similar is said by Erik, a programmer that works with data visualization. Erik knows his way around JavaScript, he was talking about JavaScript experts when he said: *"I sometimes read code written by a JavaScript expert and I think it's beautiful because it's so compact. But what's written is complicated and thus hard to read for somebody like me. I like to write out my code a bit more which makes it more readable."*

Notable was that Erik also had a certain degree of respect for the code written by an expert in a given language. But he, like Dinesh, preferred to write his code a bit longer making it easier to read and maintain. Both Erik and Dinesh did see the beauty in writing code that was compact, but both noted that understandability, in their eyes, was more important than writing compact code.

Several topics have been covered during the interviews conducted in this study. From the answers, the following questions were answered.

4 CONCLUSIONS

Several topics have been covered during the interviews conducted in this study. From the answers, the following questions were answered.

What types of aesthetic exist in code? Two different types of aesthetic could be differentiated based on the reactions given in the interviews. "Low level aesthetic", an objective and measurable type of aesthetic. The characteristics that were given with this type of aesthetic hinted at "clean code, often used when working in teams in a professional setting. This type of beauty is directly linked to readability and re-usability. Secondly "high level aesthetic", a more subjective type of aesthetic. This type of aesthetic was by some associated with different kinds of emotions such as excitement, joy, and surprise. Interviewees had a hard time describing this type of beauty relative to the objective type of beauty mentioned above. Interestingly despite often being revered to as beautiful, compact and efficient code was not always well received because of the complexity it brought with it. Though this complexity, for others, had an element of beauty to it.

In their view, what are the characteristics of aesthetic code? Characteristics associated with the first type of aesthetic included: correct white spacing, good commenting, modular, neat structure, and overview. A notable similarity between these characteristics is measurability. A characteristic that was commonly associated with this type of beauty was low effort. With the second type of aesthetic, there were two important characteristics. The expertise of the writer and their understanding of the problem that was being solved.

How do the aesthetics and functionality of code relate to each other? Both types of aesthetic seem to link directly to functionality. The programming world is built around functionality. Interestingly the programmers interviewed mostly were mixing the two words when they were asked about beauty in code. When asked whether functional code was beautiful in their eyes most programmers answered yes. Clean code, for example, is seen by many programmers as beautiful. The main goal of clean code is to keep code readable and maintainable, both characteristics of functionality. When the interviewees were talking about characteristics like structure and modularity, they quickly added that it made life easier for them. Another example is the usage of CPU's. Programmers in the early years of programming had to be very careful with CPU usage when writing code. This resulted in a shared opinion among programmers that writing code that was low in CPU usage was beautiful. Since the technology of the CPU has greatly increased this beauty has completely faded.

5 DISCUSSION

For this study only programmers from the Netherlands were asked for their opinions and thus might not be a accurate representation of the general programmer in the world. This study does, however, show the impact aesthetics has on the programming community. Most of the interviewees have had coding related aesthetic experiences in their day to day life and were therefore eager to talk about their opinions and insights. From the results of this study, one could conclude that aesthetics do reflect the functionality of a given piece of programming code. No research is done on why the programming community is so fixated on functionality and does not leave space for topics like aesthetic. Future research on this topic would be very interesting and may be useful.

ACKNOWLEDGMENTS

A special thanks to Jacco van Uden from the lectorate change management at the Hague University. We furthermore would like to thank all 8 participants of the interviews for providing the data used in this study.

REFERENCES

[1] M Dorothee Augustin, Johan Wagemans, and Claus-Christian Carbon. 2012. All is beautiful? Generality vs. specificity of word usage in visual aesthetics. *Acta Psychologica* 139, 1 (2012), 187–201.
[2] Astrid Brinkmann. 2009. MATHEMATICAL BEAUTY AND ITS CHARACTERISTICS-A STUDY ON THE STUDENT'S POINT OF VIEW. *The Mathematics Enthusiast* 6, 3 (2009), 365–380.
[3] Joe Ferris. 2012. *What is good code?* Retrieved March 2, 2005 from https://thoughtbot.com/blog/what-is-good-code
[4] Leo A Goodman. 1961. Snowball sampling. *The annals of mathematical statistics* (1961), 148–170.
[5] Aaron Kozbelt, Scott Dexter, Melissa Dolese, and Angelika Seidel. 2012. The aesthetics of software code: A quantitative exploration. *Psychology of Aesthetics, Creativity, and the Arts* 6, 1 (2012), 57.
[6] Peter Molzberger. 1983. Aesthetics and programming. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 247–250.
[7] rzetterberg. 2009. *Are you concerned with the aesthetics of your code.* https://stackoverflow.com/questions/1079645/are-you-concerned-about-the-aesthetics-of-your-code
[8] Anselm Strauss and Juliet Corbin. 1994. Grounded theory methodology. *Handbook of qualitative research* 17 (1994), 273–85.