xCoAx - PhD Symposium

Communication

Purpose of the research and its importance to the field

The purpose of this research is to identify the role that aesthetics play in the writing and reading of source code. At first somewhat counter-intuitive -why should something ultimately understood by a machine as binary voltage current be deemed "beautiful" or "ugly", "aesthetic" or not?- the fact that source code can have pleasing formal properties is almost universally acknowledged by programmers, digital artists and hobbyists alike. This implicity consensus about the possibility for code to be "beautiful" however falls short of providing identifiable and consistent formal aesthetic features specific to written source code. Taking the definition of aesthetics as a set of formal properties eliciting a sensory-emotional reaction in the creator and the beholder, this research project consists in looking at source code through the field of literary theory, by treating it as literary text, asking questions about form, authorship, readership, distribution and reception in the context of written scripts. The purpose of identifying the aesthetic properties of source code is to see to what extent these relate to, or differ from, the aesthetic properties of written language, and what is their relationship with the necessities of executing code. In particular, by looking at the tension between what is at the same time a machine-readable text and a human-readable text, I aim at highlighting the role that aesthetics play in both machine- and human-understandings of source code.

This research intends to open up new readings of source code, using the lens of aesthetics in order to better understand how code functions as a specific semantic apparatus. It intends to bypass approaches of code as a purely theoretical notion—"code" as a term encompassing, and thus erasing, the variety of programming languages that make up its reality—, or as an effective procedure—code as a compiled binary and executed software. This definition of source code as limited to written code thus excludes graphical programming languages, although it is possible that the conclusions of the current project could apply to those, as well as executed code, thus drawing influences from, but not directly contributing to, the field of electronic literature. Particularly, this research contributes to an approach of aesthetics as a functional component in the creation, understanding, and re-use of cognitive objects.

Brief survey of background and related work

Overall, I intend to draw from two separate fields of research which address aesthetics in programming. The first starts with the literature on software development and software engineering as technical practice, as illustrated by the works of Djikstra, Knuth, Kernighan and others. Most of the publications in this field connects aesthetics, as well as the related fields of "art of computer programming" (knuth), "craft of programming" (Hunt), or "beautiful code"

(Oram), to the reality of writing and reading understandable code—in a sense being concerned with "applied aesthetics". If one of the greatest hurdles of programming is to be able to comprehend what a piece of software's potential is, aesthetics is here presented as an essential step towards overcoming that hurdle.

A second field stems from media studies sotware studies. It starts from broad theoretical work establishing a definition of software as a socio-technical and cultural object (MacKenzie), towards close, deliberate readings of source code in more recent work (Montfort et. al., Paloque-Berges), complemented by an approach from the field of literary studies (Marino, Hayles). The object of source code is more and more the focus of researchers, particularly in terms of *what code means*, how it represents the world, and how it is represented in the world (Sondheim). However, little attention has been given to what kind of intrinsic formal aesthetic properties it could exhibit, or to what standards can source code be held to as an aesthetic object. More than regular source code -taken from, say, the Linux kernel-, some types of source code appear recurrently in these studies.

While dealing with the same object, the corpuses studied are clearly separated from one another. One the one side, there exists a well-established corpus of commented source code circulating in the professional field of software development. These include best practices, code demonstrations, commentaries and text books, ranging from Djikstra's *programming pearls* to the commentary of the Unix version 6 source code. The intent of this research is to further improve the quality of the software made by industry practicioners. References to other fields (such as linguistics, literary studies, or cognitive psychology) are therefore rare but not completely absent (**insert refs here**). Most of the contributions of those texts is to establish concrete best practices when writing and reading source code, yet only accidentally linking it back to an aesthetic practice and literary concepts such as *simplicity*, *clarity*, and *elegance*. On the other side, the body of texts of self-defined *source code poetry* has been given close attention, but yet again mostly in terms of socio-cultural practices, and what they mean within the communities of programmers within which they were written and read. While often addressing the *poetics* of code (i.e. the world-making, storytelling potential of software), some of these **refs**) focus particularly on formal exercises such as Perl Poetry, the Obfuscated C Code Contest and well-known esoteric languages and investigate to what extent these writings could qualify as "code". Within the realm of "artistic code", Paloque-Berges's work is closest when it comes to analyzing the specific forms that source code poetry takes, and what kinds of relationships those forms have with natural languages. For instance, she does so both by highlighting syntactical tokens that make Perl uniquely suited to this kind of endeavour, but also by providing larger theoretical frameworks, based on the idea of fiction (Goodman) and of practice (De Certeau).

The most relevant and interesting research today has been coming from the field of *Critical Code Studies* (CCS) and digital rhetoric. These link directly the syntactic and the semantic components of software: reading source code

is essential to understanding what that code is intended to do. It is in this trajectory of code rhetoric that I place this research project, particularly in terms of untangling the relationship between specific formal manifestations and overall conceptual understanding. In that sense, I will be investigating this specific overlap by linking both the analysis of software as a technical practice as well as a socio-economical object.

Description of the proposed approach

The proposed methodology is threefold: empirical, theoretical and experimental. The basis of this research will be constituted of primary sources: source code available online, published in hobbyist magazines, commented in textbooks or published as artworks. The analysis of these texts, along with the discourses around those texts (comments, reviews, discussions) will help develop a heuristic to understand when and how the criteria of the beautiful, as well as the ugly, is summoned among communities of programmers. This corpus encompasses all ranges of project sizes, languages and periods, from the Apollo IX landing module procedure, to the snippets published in fanzines, or the examples shown in style guides, as well as the more obvious recent publications of *source code poetry*. The wide variety of texts integrated into the corpus will allow me to identify to what extent the aesthetic standards identified are context-free—i.e. which aspects vary between high-level languages and low-level languages? between professional and amateur codebases? between educational and artistic settings?

The second component of this process will be to examine the findings from the analysis of this corpus at the light of the somewhat more traditional concepts of literary theory and linguistics. Recurring concepts in programming discourses, such as *clarity*, will be further examined in the light of concepts such as authorship, reception, rhetorical figures, style, voice and layout, among others. This cross-disciplinary approach will establish a framework which takes into account both the appreciations intrinsic to the communities of practice of source code (e.g. software developers), as well as further highlight some unique aesthetic properties of code via literary methods of analysis. For instance, there could be an exploration of the concept of *simplicity* by comparing the programming paradigm *DRY* (Don't Repeat Yourself) and Barthes's *writing degree zero*, or a re-assessment of *explicitness* in the light of Mikhail Bakhtin's *dialogism*.

Having sketched a conceptual framework for approaching source code as a text with potentially aesthetic manifestations, the last part of this research project will consist in confronting these conclusions with case studies. In line with the empirically starting point and the close-reading practice of CCS, I intend to examine three texts, ranging in variety in terms of source code quality (professional, amateur, artistic), in order to confirm or infirm, and in any case further elaborate on, the framework established in the previous phase.

Expected contributions

The expected contributions of this research project will be as follows:

- formalization of "ambient knowledge" existing in the field of software development
- further qualification of the role of aesthetics as a functional one (telic, rather than autotelic)
- exploration of "code as thought as material" and the socio-cultural consequences that this draws in terms of:
  - the role of linguistics in aesthetics
  - the role of social circles in aesthetics
- and essentially making a bit more explicit the interplay between object-making (craftsmanship), language (literary theory) and thought (linguistics)[^1]

[^1] I'm not sure whether lit is more related to language or to thought, or vice-versa for linguistics