The Craft of Code: Practice and Knowledge in the Production of Software

Pierre Depaz

January 2021

1 Introduction

Software development as a practice has been developing for the past sixty years, emerging as a corollary from the field of computer science. In the 21st century, the importance of code in everyday lives has been highlighted from general-audience journalism (Paul Ford, What is code) to government initiatives (Obama, Code). However, the distinction between programming and computer science is a blurry one. While a computer science degree does provides the appropriate formation for a programmer, successful programmers do not necessarily need a computer science background in order to be competent at their jobs. Indeed, while programming has historically emerged as an occupation which responded to the *ad hoc* requirements of computing as a developing field (Chun, Software Wants to Be Overlooked), developing further into a codified and identified practice (Djikstra), programming has nonetheless often been approached in media- and software-studies with a focus on the phenomenon of computation (notably, Galanter, Katherine Hayles, McKenzie), rather than on the reality of programming.

Starting from this observation, this article proposes to investigate a different tradition than that of the sciences to highlight the specificities of programming as a practice—the tradition of craftsmanship. Indeed, code isn't

just code, but rather a myriad of socio-technical *assemblages* composed of programming languages (e.g. Ruby, C, Julia, JavaScript), operating systems (e.g. Linux, BSD, OSX, Windows) and tools (e.g. IDEs, debuggers, compilers and processors). Those assemblages are in turn used within a cultural context made up of stories, sayings and texts, both from academic and folklore origins. This approach relies on a shift from a conceptual perspective of code, in which the word *code* encapsulates varieties of activities (Brian Hayes, Cultures of Code), to a more practical one, in which the variety of practices, self-identifications and narratives from programmers themselves is put at the forefront.

While links between craftsmanship and programming have existed as self-proclaimed ones by programmers themselves, as well as incidentally by academics (Sennett, Chandra), they have not yet been elucidated in a comprehensive manner. Indeed, craftsmanship as such is an ever-fleeting phenomenon, a practice rather than a theory, in the vein of Michel De Certeau's *tactics*, bottom-up actions designed and implemented by the users of a situation, product or technology as opposed to *strategies*, in which ways of doing are prescribed in a top-down fashion. It is this approach that this article chooses, that of the *practice*, the informal manners and standards of working, in order to provide an additional, cultural studies perspective, to its media and software studies counterparts.

A comparative approach of a broad mode of economic and cultural activity (craftsmanship) with a narrower technical know-how (software development) asks us first to verify to what extent such an approach is even valid. How, then, is the designation by software developers of their own practice as craftsmanship relevant? Where is the comparison a productive one, and where does it show its limits? How can such comparison enrich both our understanding of code as a practice, as well as craftsmanship practices within the highly networked environment that has become the backdrop of the 21^{st} century? Particularly, how does it re-present processes of knowledge acquisition and aesthetic judgment?

The article proceeds in a comparative fashion, mobilizing texts about craftsmanship as well as sources from the field of programming, describing programmers' self-identification with craftsmanship. From those, I analyze how those references by programmers enter into a productive dialogue with our historical and cultural conception of craftsmanship. To do so, I approach these questions through three different, contiguous topics. First, the focus is set on the historical unfoldings of craftsmanship and software development. After inquiring into the modes of organization and the economical development of both craftsmanship and software development, this section focuses on a particular comparison: that of programming with building, and in particular its relationship with architecture. Second, we shift our focus from a broad view to the specific practices of knowledge acquisition and production. There, we highlight the similarities in terms of tacit and personal knowledge, as well as the means of education and information available both to traditional craftspeople and software developers. This, in turn allows us to disucss the differences of learning environments by taking into account the networks of environment that have developed exponentially with the Internet. Finally, the third section turns to aesthetic judgment of the crafted product. Building upon discussions of code as art and code as literature, the focus here is on the standards which allow practicioners to ascribe beauty to a piece of software; particularly, this section discusses the materiality of code not in terms of bits, bytes and languages, but rather as a material that is worked with and worked through.

These incursions in the practices of software development via the lens of craftsmanship as a cultural practice concludes on the productive differences and similarities in terms of knowledge circulation and materialities when programming code. Ultimately, they result in new understandings of both craftsmanship and software development as mutually influencing practices.

2 Social and historical developments

This section starts by providing an overview of the various perspectives and realities on craftsmanship, starting from the Late Middle-Ages until the 20^{th} century. This allows us to highlight some initial important features of craftsmanship: social organization, the nature of work and the transmission of knowledge. Jumping to the history of software development, starting in the mid-20^{th} century, it looks at the claims that programmers make about themselves in relation with the term and concept of craftsmanship. By examining formal and informal texts, we focus on the fact that software developers ground their practice in passion, know-how and myths. Finally, we inspect the place of architecture, both in historical craftsmanship and contemporary software development, in order to qualify further the relationship between theory and practice in those two fields.

2.1 Craftsmanship throughout history

a specific time: late middle ages/renaissance, before automation happened, after the times in which craftsmanship was disregarded socially

- middle-ages, builders, architects, guilds - renaissance: optics, computers, tacit knowledge - re-organization of work: artificiers

2.2 Software developers as craftsmen

- early days: hackers and myths - compulsivity in practice: weizenbaum - uncle bob: clean code

2.3 Architecture between craftsmanship and programming

- the cathedral and the bazaar - patterns of software, patterns of space - parallels with guilds and displacement of the hand-made (?) - the devel-

opment of the architect, as a parallel with the computer scientist

3 Knowledge acquisition and production

After having elucidated in what broad respects software development might be like craftsmanship, we highlight one specific aspect which asserts interesting tensions (both similarities and differences): knowledge acquisition and transmission.

3.1 Knowledge in traditional craft

- short-circuit - learning by doing and observing - apprentices/masters - cookbooks - practice and repetition

3.2 Knowledge in software development

- ad hoc - somewhat formalized, a lot self-taught - learning by doing, not so much observing (paul graham, hackers & painters) - - snippets of code on stack overflow? - relationship to IP law? crafts and wares were signed, code as well. however, code can be hidden much more effectively. - the bus factor - the problem of documentation

3.3 Diffuse knowledge and hobbyists

- the computer is a cheap tool - diffuse knowledge, resulting maybe in more hobbyism than expertise (the network society, cscw studies) - the role of clubs (TMRC, hacker conf, meetups) - still specific commununity online

4 Material aesthetics

at the heart of knowledge transmission and acquisition stands the *practice*, and inherent in the practice is the *good practice*, the beautiful one. this section investigates the aesthetics of code within the broader context of the aesthetics of craftsmanship, highlighting how code can act as a material.

4.1 The Aesthetics of Craftsmanship

- pye - sennett - harold osborne

4.2 The Aesthetics of Code

- functional beauty (also in osborne: the difference between fine arts and craftsmanship is selflessness, practical utility) - clear, simple, elegant - no ornament?

4.3 Code as material

- essential complexity - code smells, spaghetti, etc. - the role of tools (terminal)

5 Conclusion

there are some self-proclaimed similarities, but the most reliable/salient ones come from architecture and aesthetics, making the case for code as material.