Programming Practices of Chinese Code Farmers

Articulations, Technology, and Alternatives

PING SUN

ABSTRACT: Built on the theoretical framework of articulation and assemblage, this article explores programming practices among grassroots programmers in contemporary China. Using data obtained from ethnographic fieldwork in Shenzhen, Guangdong Province, it provides an account of the information technology practices in contemporary China at the nexus of the Beijing government, IT corporations, and individual programmers. Through examining how programming is articulated in both China's advocacy for "a creative society" and grassroots programmers' daily practices in the process of China's informatization, this article has mapped myriad articulations such as engagement, communication, discourse, and practice that have made and unmade grassroots programmers' programming assemblage. We argue that technology for Chinese programmers is a mixed blessing. As a means of survival, technology exacerbates the precariousness and marginalisation of grassroots programmers in China, while the capability of technology production also enables the remaking of subjectivity and social change. The findings of this study thus advocate a deeper and dialectical understanding of the interaction between technology, labour, and empowerment.

KEYWORDS: Programmers, articulation and assemblage, technology, Maker Movement, China.

Introduction

At the meetings of the 2017 National People's Congress (NPC) and Chinese People's Political Consultative Conference (CPPCC), the Beijing government reiterated the "Internet Plus" action plan ("*Hulianwang+" xingdong jihua* 互联网+"行动计划). According to Chinese Premier Li Keqiang, "Internet Plus" entails the integration of mobile Internet, big data, cloud computing, and the Internet of Things (*wu lianwang* 物联网) with manufacturing. The aim of this policy is to promote innovation-driven development, foster new industries, cultivate entrepreneurial individuals, and upgrade China from a "big industrial country" to a "creative country."

As China's economic growth slows after two decades of rapid expansion, Beijing is concertedly seeking strategies to bolster the economy. Information and Communication Technologies (ICT) fits this bill. Through a clarion call to combine Internet technology with modern manufacturing, the Chinese Government is trying to foster a broad "Creative Society" (*chuangxin xing shehui* 创新性社会) and realise its "Chinese Dream" (*Zhongguo meng* 中国梦). With the government's relentless support for the IT industry over the past three decades, China is currently at the acme of its tech boom. The world has witnessed the breakneck emergence of an "IT dragon" that is largely supported by a huge cheap labour force and a massive churning out of copycat products. As China connects itself closer to the global digital revolution, the global open source movement and maker movement have also had a great impact on its IT prosperity. Transnational IT companies, hackerspaces, and open source communities are cross-pollinating with China, making the IT vista even more diverse.

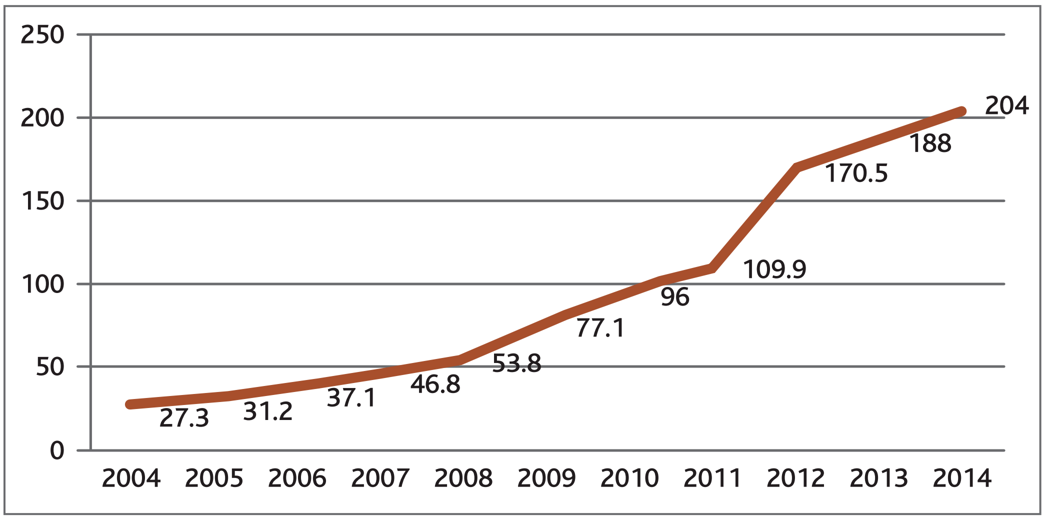
By the end of 2014, China's ICT market output reached US$204 billion (see Figure 1), accounting for 1.9% of China's GDP. Software has become a fast-growing sector as China establishes itself as one of the most important countries for ICT production and consumption. In 2013, software production in China accounted for 25% of the total ICT industry and 13.4% of the global software market. Based on Gartner's report, [1] China has been the world's second largest software producer since 2013.

The convergence of technology production, political discourse, and general public entrepreneurship enables new areas of research. Previous literature has addressed China's ICT development on the macro level (e.g., Segal 2003; Zhao 2007), while a mesi- or micro-lens on individual and contextualised IT practices has received less attention. To fill this gap, this paper examines China's ICTs by particularly focusing on individual programmers' work practices. Programmers, especially those from small companies in China, are also referred to as "*manong*," which literally translates as "code farmers" in English. In Chinese, *ma* (码) means code, while *nong* (农) refers to traditional Chinese farmers who have limited education, who are frequently dispossessed of power, and are frequently overlooked. When combined, the word "*manong*" depicts a position related to programming and coding that is also subject to frequent exploitation and high pressure.

Through the lens of an ethnographic study in Shenzhen, Guangdong Province, this article explores digital experience of Chinese *manong* in IT companies. While it addresses IT practices in a more detailed way, the paper does not overlook the inseparable relations among individual programmers, social institutions, and technology. By introducing stories and discourses from the fieldwork in Shenzhen IT companies, this article illustrates the mixed role technology plays in programmers' technological experience as well as in remaking their subjectivity. The paper argues that technology

1. "Enterprise Software Markets, Worldwide, 2008-2013, 4Q09 Update," Gartner Forecast: Online Report, https://www.gartner.com/technology/research/ (accessed on November 2016).

Figure 1 – Growth of ICT Market Output in China by Billion USD



Source: Compilation based on IDC Research Report (2004-2014).

itself cannot be fully understood without considering its essential sociopolitical and socioeconomic character.

Why are technicians such as programmers important? From a historical perspective, the narratives of technology tend to regard its development through isolated physical artefacts or visible things, while little attention is paid to the people behind it. This compelling discourse may risk becoming technological determinism as it ignores the interactions among people, institutions, and society. Secondly, previous literature on technology attaches importance to technology use while less consideration is paid to technology production (Ensmenger 2012; Light 1999). Taking China as an example, while IT professionals such as programmers constitute a critical part of its "digital empire," little attention has been paid to technological practices such as how technology is deployed and constructed in specific contexts. In other words, how the invisible but no less significant programs, codes, and algorisms are embedded into technological narrations is of importance in understanding techno-social relations.

The article is structured as follows. First, a theoretical framework of articulation and assemblage is introduced, which presents the programming practices in an interconnected and contextualised way. Second, the methods of data collection and data analysis are explained. In this part, experience is addressed as an effective way to illustrate the social meaning of technological practice among grassroots programmers in China. In the data analysis, the universal scramble for IT jobs in Shenzhen is described, illustrating how technology is understood and serves as a means of survival for grassroots programming labour. Following this, an alternative analytical lens is proposed by discussing how Chinese and English are used differently in programmers' coding work. The focus then shifts to examine how programmers gain empowerment through teamwork, algorithms, and the fledging maker movement. The possibility for remaking programmers' subjectivity and social change is discussed in the conclusion.

Theorizing IT practices: Programming as articulation and assemblage

This article employs the theory of articulation and assemblage as an exploration of the technological practices of Chinese programmers. In the theory of "double articulations," Deleuze and Guattari (1988) maintain that one form of articulation can be "the plane of expression," which refers to signifying semiotics or discursive practices, including but not limited to elements such as categories, discourses, forms, and identities. According to Deleuze, assemblage is "a multiplicity which is made up of heterogeneous terms and which establishes liaisons, relations between them, across ages, sexes and regions—different natures" (Deleuze and Parnet 2002, p. 69). It can therefore be argued that assemblage refers to the network of articulations that link formations among heterogeneous elements. The elements of assemblage may include "discourses, words, 'meanings,' and non-corporeal relations that link signifier with effects" (Stivale 2014, p. 94). Assemblages can be systems of signs, things, actions, or semiotics. Delanda argues (2010), however, that the relation between articulations and assemblage is not simply whole and parts, unity and elements. Rather, the two levels of scale (articulations and assemblage) can be both constraining and enabling to each other. The relationship can be manifested in a top-down direction, in which individual entities can only remain active and functional within a social assemblage. Alternatively, it can also be a bottom-up relationship, with individual entitles at different levels (persons, communities, organizations) interacting with each other and thus forming forces to be reckoned with by assemblages.

This paper will examine the interactive relations between articulations and assemblages in individual programmers' work experience in contemporary China. As a theoretical framework, articulations and assemblage have been persuasive in explaining techno-social relations (e.g. Wallis 2015). The articulations and assemblage framework captures technology not as static or unified infrastructures, but as an ongoing system that is composed of, and interacting with, "social practices, discursive statements, ideological positions, social forces or social groups" (Slack 1996, p. 331). In the politics of technology in China, both individuals and society are transformed by the informatization agenda. In order to build a theoretical connection between individual entities at different levels and the overall ever-changing IT industry, I argue that China's informatization assemblage is shaped and re-shaped by the interconnected physical, ideological, and subjective activities in the IT industry. To be specific, the assemblage in China is the programming practices constructed by Chinese programmers, which can be divided into two main aspects. The first is how programmers get involved in the process of China's informatization. The second is the role of programmers in the international division of digital labour (Fuchs 2014). Articulations are also divided into two parts: expressional and practical. I argue that programmers use both discursive and collective experiences to build their social identities and work performance.

The rationale of the articulation and assemblage framework presumes that any articulation or assemblage is historically and contextually contingent. Assemblage creates territories, not in a static way, but always in making or unmaking. It is noted that articulations are made, maintained, or transformed in particular social practices, while assemblages are connected or disconnected, embedded or dis-embedded in social cultures or relations. As will be demonstrated in this paper, programming can be reviewed as an ongoing set of articulations and linked assemblages. Programming assemblages are not just collections of physical hardware, gadgets, or algorithms, but also take up particular languages, discourses, communication, and social meanings in programmers' lived practices. Programming assemblage refers not only to the networks or linkages among different elements in programmers' daily practices, but also to the asymmetric power relations Chinese ICTs have generated between individuals, institutions, and technology. Viewing programming as articulation and assemblage, this article aims not to complicate its theoretical abstraction, but to employ it as an utilizable, yet also malleable and changeable framework in exploring the technological experience of Chinese programmers. In other words, using articulation and assemblage does not place the emphasis on the predetermined narratives on technology, but rather embraces "conceptual openness to unexpected possibilities and resolutions" (Ong 2006, p. 17) in which experience, discourses, and communications are expressed and negotiated in a specific context.

In tackling articulation and assemblage in programming, this paper particularly focuses on the experience of programmers in Shenzhen. Based on previous literature, experience can be defined as the engagements people have made to relate themselves to the world, which includes feelings, thoughts, expressions, and actions (Scott 1992; Bruner 1986; Pickering 1997). This study uses what Burner terms "the anthropologist of experience" as the entry point to Shenzhen's fieldwork (Burner 1986, p. 9). Rather than just including snippets of discourses and narrations of the participants, the anthropologist of experience regards participants as active agents and takes their expressions such as memoirs, narratives, and other forms of discourse as experience. By doing this, "we leave the definition of the unit of investigation up to the people, rather than by the anthropologist as alien observer" (Burner 1986, p. 9). In this research, the programming experience includes a wide range of social activities and discourses such as code development, interactive communication, discursive expressions, and organisational practices.

To collect data, I conducted ethnographic fieldwork in Shenzhen from July 2015 to July 2016. By working as a Search Engine Optimizer (SEO) in an IT company called AoC, I conducted participant observations with programmers.[2] After building trustful relationships with programmers in AoC, I then conducted interviews with a total of 45 programmers through snowball sampling. Participants came from more than 20 small companies. I conducted interviews with each of the interviewees one or two times, with each interview lasting 60 to 90 minutes. The interviews were conducted in Chinese, and I translated them into English later. The interviews usually started with the researcher telling the interviewees the purpose of the study and promising confidentiality. Informed consent was agreed or endorsed by interlocutors. The researcher usually started with warm-up questions such as "describe your work experience" or "describe the project you are doing now." Then questions such as, "why do you want to be a programmer?," "do you think of yourself as a *manong* (code farmers), or "what are the impressive things in your work experience?" were asked. Sometimes the author also made the dialogue open so as to find interesting and unexpected data.

Rather than focusing on the upper-class programmers who work in big IT companies such as Tencent, Alibaba, or Huawei, the fieldwork paid particular attention to "grassroots programmers" working in small companies in Shenzhen's Nanshan District. According to Yue (2015), small companies (with a total number of staff less than 50) dominate the Chinese IT market. They account for more than 55% of the total number of software corporations, accommodating 90% of the programming labourers in China. As distinct from high-level programmers in giant companies, "grassroots programmers" are the "new poor" (Wang 2014) as urbanisation and informationalization, which are characterised by deficient education, lower income, and precarious work conditions, continue in China. Graduating from vocational schools, programmers usually cluster in small outsourcing start-ups and maintain unstable work, since small IT companies may go bankrupt at any time.

In Shenzhen, programmers have been part of the "invisible human infrastructure" (Oregalia 2013, p. 24) that keeps the city running. As discussed previously, with China's continuing globalisation, individual programmers' work has not only been integrated into the national movement to build a "creative society," but also contributes to the global labour market. Based in this context, the paper answers the following two questions: How does the assemblage of programming, from either a national or global perspective, interact with the articulations performed by individual programmers in China? What does the work practice of Chinese programmers add to existing knowledge about articulations and assemblage?

Program or be programmed

Over the last four decades, the open door policy and Beijing's relentless pursuit of modernity have paved the way for the emergence of China's informatization. As a country whose modern history has been characterised by backward technology and an ignorance of science, China's efforts to transform its national image have been evident in political rhetoric praising the realisation of informatization and the building of an IT empire. From 2004 to 2014, the ICT market output has increased six-fold, and IT developers tenfold. As the IT industry has roared to life in big cities, a substantial number of migrant workers have been uprooted from traditional manufacturing and have immersed themselves in this new field. Instead of following their migrant parents into the factories, the second generation of migrants regards IT as a more effective way to make a living. Roberto is the first programmer I encountered in my fieldwork. After failing the College Entrance Exam (*gaokao* 高考) three times, he attended Aptech Beida Qingniao (*Beida qingniao* 北大青鸟), a popular IT training school in China, and became a Java programmer in Shenzhen. Regarding his reason for becoming a programmer, he said:

> My fellow villager [*laoxiang* 老乡] told me that I could find a high-salary job in big cities if I learned computer programming there [Aptech Beida Qingniao, the IT training school]. Beida Qingniao also guaranteed me a job if I could complete the training courses.[3]

The large demand of the IT industry requires churning out as many programmers as possible. As a consequence, training organisations have blossomed. At the company where I conducted my internship and gathered fieldwork data, there were LED advertisement boards of IT training schools on the metro ad walls (see Figure 2). These advertisements change every week, reminding passers-by of an "easier and better" way to hunt for a job.

2. The name of the company has been changed due to confidentiality concerns.
3. The interview was carried out with Roberto when he came to Shenzhen for a project in August 2015.

Photo 1 – Advertisement Boards for IT Training in the Taoyuan Metro. © Ping Sun, May 2016.

Training organisations such as Aptech Beida Qingniao, Tsinghua Wanbo (*qinghua wanbo* 清华万博), and ITcast (*chuanzhi boke* 传志博客) are quite familiar to people in big cities. They provide various IT training courses to attract more trainees, but the tuition fee is usually not cheap. At Aptech Beida Qingniao in Shenzhen, staff asked for 15,888 *yuan* (approx. US$2,450) for a three-month course in zero-based Python (one of the programming languages).

The schedules of these IT crash courses are tight. For example, XDL (*xiongdilian* 兄弟连) in Shenzhen is a training organisation that arranges 14 hours' IT training per day for people who come from 8:00 A.M. to 12:00 P.M. Trainees are required to make 24-hour commitment to their courses. Students are accommodated in shabby dormitories near the training organisation, with eight or more sharing one room. Based on different computing languages, they are divided into dozens of classes, with 20 to 80 trainees as a unit. Their main activities during the training were attending class and practicing coding. "Too many things in one day—you can hardly digest them without practicing by yourself," Roberto said. Most of the trainees in these training organizations are youth from rural areas who either dropped out of school or failed the college entrance exams. Under the pressure of high tuition fees and job-hunting, no one skips class. Trainees nevertheless also complain about their training. For instance, in order to drum up business, some training schools guarantee trainees work after finishing their course, but that quite often turns out to be bogus. Meanwhile, when trainees are sent to the labour market, they often find that their education in the training schools is not enough to obtain a programming job. Wu Qing is a programmer who received his training at XDL, but three months after he finished training, he was still looking for a job.

For many rural migrants in China, choosing to become a programmer is articulated as a way to "survive" rather than "thrive." Many factors contribute to the seemingly speculative choice to engage in software programming. Compared with people who work in other fields, programmers in the IT industry have relatively high salaries that few would consider giving up. The burden of supporting families also plays an important role in pushing them to remain programmers. Working as programmers, they are often seen as "important providers" to support their siblings or families. Jane is a .NET (a kind of programming language) developer from Yiyang, Hunan (益阳，湖南). Every month, she sends half of her salary (2,500 *yuan*, approx. US$380) home to pay her younger brother's tuition fees. In China, it is not uncommon for elder working sisters to send money back to support their families, who usually live in the rural areas and are mired in poverty and insufficiency. In the summer of 2015, Roberto left Shenzhen and entered another start-up in Beijing. The reason for doing that, according to him, was for "more money" and to "support his bros." During our third interview, Roberto shared his stories with me. He had to pay a housing loan for his elder brother, who was serving a jail sentence. In addition to the housing loan, he also had to support his parents, both of whom are farmers from a small village in Henan Province. He told me:

> I am not interested in the work I will do in Beijing. (...) But the thing is I can earn more money in this small company. In Shenzhen, I get 12,000 [*yuan*, approx. US$1,860], but here I can get 16,000 [*yuan*, approx. US$2,480], though they are doing some online education systems that are very boring to me.[4]

In explaining the relationship between articulations and assemblage, Delanda (2010) suggests that articulations at a personal level are usually constrained by the overall assemblage. The underground programming training movement ongoing in Shenzhen is more passive than active. More specifically, these would-be programmers are pushed by the top-down tide of informatization and urbanisation in China. Loaded with the burden of supporting rural families, their choices in job seeking become limited, and programming becomes one of very few opportunities left for rural youth to gain social mobility. Takhteyev (2012) mentioned the nerd characteristics of software programmers in Rio de Janeiro, and stressed their childhood passion for computers. It would be wholly inaccurate to assume that China lacks people who are obsessed with programming, but for grassroots programmers in China, their lack of passion and fascination with coding makes programming far from their first choice of professions. In other words, these "coding farmers" (Sun and Magasic 2016) in China do not choose to do software programming; rather, they are chosen for it.

The mother tongue or the better tongue?

Ethnographic fieldwork in Shenzhen gave me more exposure to programming languages. Like most people who do not work with programming, I am completely unfamiliar with these arcane and highly technical codes. It was not until the winter of 2014, when I communicated more frequently with some of my interviewees, that I realised I had to immerse myself in

4. Roberto, Shenzhen, August 2015.

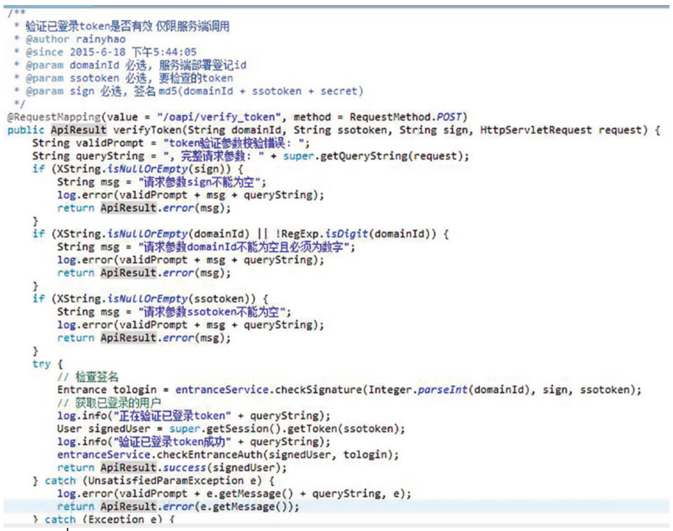
programming language if I wanted to know more about its meaning to the programmers. I started to actively look at the programs my interlocutors wrote, trying to figure out their basic meaning, while also beginning the journey of learning Python, a widely used programming language. I soon discovered an interesting phenomenon when I tried to discuss source code with programmers: most of the codes they wrote were a mix of Chinese and English—or to be more precise, a mix of Chinese and a particular programming language, the latter usually composed of English words (see Photo 2).

As I found during my fieldwork, programmers' discussions and discourses about programming language quite often express inherent contradictions in their daily work practices. In examining the source code, it is common to find an amalgamation of two languages: Chinese and English. The snippet shown on Photo 2 has two types of text: the text enclosed in "/*…*/" (the first six lines) and the two line starting with "//." Such texts are comments, which are usually ignored by Java syntax, and in fact they can be written in any language. The second type of text is the Java program, which converts the readable comments into coded instructions that can be recognised by the computer. It can be surmised that programmers maintain a very limited choice of languages for the second type of text, since the very technology, Java complier, can only be understood and processed in English. The Chinese characters that appear in the Java program are messages for the call end (*zhendui diaoyong duandi tishi xinxi* 针对调用端的提示信息), to help them better know what the codes mean, since this software is for Chinese clients. Of course, the Chinese version can be replaced by any other language, such as Hindi, Spanish, German, or Russian. When I asked Roberto why he used Chinese for comments, he replied:

> Roberto: "It is just more natural and convenient for us to use Chinese. Of course, I mean, it is our mother tongue, right? We use Chinese to remind us what we are doing in this part of the code, or we will forget the content."
> Researcher: "The code content? You can't just read the code?"
> Roberto: "Well, we can, but it is time-consuming for me to read English all the time, you know. I am not good at English." [5]

Programmers mostly use Chinese for the comments, and then they have to follow the instructions of Java to finish the main body of the program with English. In an analysis of software language, Takhteyev (2012) discusses the relationship between English as a global language and Portuguese as a local language, indicating that English is not only associated with the culture of software but is also embedded in its very technology (Takhteyev 2012). Programs are machine-built codes, which can only be recognised by a certain type of language. Here, English being the "normal language" has been widely accepted as a precondition for programmers in China. As the original language of the entire Java programming system, English is irreplaceable and indispensable. For programmers in China, their overall programming skills are confined and performed within English, and coding in Chinese is usually thought of as something impossible or ridiculous.

The articulated conflict between Chinese and English is far from a coincidence. To some extent, it sheds light on the power relations between Chinese as a peripheral language and English as a central language in programming. For programmers in China, being good at or at least making themselves familiar with English is the first step, as all the codes they are typing are in English. Sometimes this can be a challenge for the lower level



```
/**
 * 验证已登录token是否有效 仅限服务端调用
 * @author rainyhao
 * @since 2015-6-18 下午5:44:05
 * @param domainId 必选, 服务端部署登记id
 * @param ssotoken 必选, 要检查的token
 * @param sign 必选, 签名 md5(domainId + ssotoken + secret)
 */
@RequestMapping(value = "/oapi/verify_token", method = RequestMethod.POST)
public ApiResult verifyToken(String domainId, String ssotoken, String sign, HttpServletRequest request) {
    String validPrompt = "token验证参数校验错误: ";
    String queryString = ", 完整请求参数: " + super.getQueryString(request);
    if (XString.isNullOrEmpty(sign)) {
        String msg = "请求参数sign不能为空";
        log.error(validPrompt + msg + queryString);
        return ApiResult.error(msg);
    }
    if (XString.isNullOrEmpty(domainId) || !RegExp.isDigit(domainId)) {
        String msg = "请求参数domainId不能为空且必须为数字";
        log.error(validPrompt + msg + queryString);
        return ApiResult.error(msg);
    }
    if (XString.isNullOrEmpty(ssotoken)) {
        String msg = "请求参数ssotoken不能为空";
        log.error(validPrompt + msg + queryString);
        return ApiResult.error(msg);
    }
    try {
        // 检查签名
        Entrance tologin = entranceService.checkSignature(Integer.parseInt(domainId), sign, ssotoken);
        // 获取已登录的用户
        log.info("正在验证已登录token" + queryString);
        User signedUser = super.getSession().getToken(ssotoken);
        log.info("验证已登录token成功" + queryString);
        entranceService.checkEntranceAuth(signedUser, tologin);
        return ApiResult.success(signedUser);
    } catch (UnsatisfiedParamException e) {
        log.error(validPrompt + e.getMessage() + queryString, e);
        return ApiResult.error(e.getMessage());
    } catch (Exception e) {
```

Photo 2 – A Snippet of Java Source Code.
© Ping Sun, November 2016.

manong, since their English training in the vocational schools is lacking. Matthew came to Shenzhen hunting for a job after he graduated from a vocational school in Anhui Province. He greatly valued the ability to use English well. Matthew told me that his English used to be very poor, but in order to continue in his work, he memorised all of the frequently used programming patterns and practiced them by inserting them into his coding. During the interview, many other programmers also expressed their admiration for programmers who were good at English and described them as "high-level" (*gaoduan* 高端). Brown was one of the "high-level" programmers. When I asked him what kind of website he used for programming information searching, he said:

> I usually go to Google or Github to search source code whenever I come across a problem. So, in that case, English can really help you a lot. As long as you can understand it, you will find it is a big, big world, a lot of good codes out there… It's huge. [6]

During the interview, Brown laughed at programmers who searched codes on Baidu and said the Chinese websites were his last choice: "English is better. It shows you're professional." In his discourse, English is not only a tool for programming, but also a kind of cultural capital that can take them to a higher level of social status. Every time I asked my interlocutors about teaching myself Python, they all recommended I go for the original English book rather than the translated Chinese one. Roberto told me:

> Your English is good; go for the English version. It is direct and original. This is your big advantage. [7]

The presence of "English admiration" in the software industry is not limited to the programming language, but also extends into a wider corporate environment. In the IT companies in Shenzhen, it is not uncommon for software programmers to give themselves English names. Some companies even

5. Roberto, Shenzhen, August 2015.
6. The interview was conducted in November 2015 in Nanshan District, Shenzhen. Brown used to work in Shanghai, and he moved to Shenzhen in the winter of 2015.
7. Roberto, Shenzhen, August 2015.

make this a policy. In the daily work routine, they keep calling each other English names, and claim that "everyone is equal this way" (*meigeren dou shi pingdeng de* 每个人都是平等的).

Linguists use "diglossia" to describe a context wherein a particular social group uses two different languages simultaneously, and in which most of the group members are fluent (Ferguson 1959; Grosjean 1982). These two languages have different functions in the system. *Manong* programmers in China are far from a typical diglossic system, since many of them are not proficient in English. In the interaction between Chinese and English, however, the different roles these two languages have played in the identity construction of programmers are clearly evident: English is the "high" language that is used for formal communication or professional programming, while Chinese is much more of a peripheral or "low" language that is used for daily communication and informal chatting.

Programming language has its own hegemony. In the discussion of assemblage and articulation, Delanda (2010, p. 13) notes that there is a tendency wherein assemblage "homogenizes its own components." By making English the standard coding language, programming in the perspective of the international division of labour territorialises its boundaries, and sharpens the distinction between English and Chinese. The use of English is articulated as being synonymous with professionalism, egalitarianism, and even social status. As the IT industry becomes more globally connected, mastering English has become a must for programmers. Here the language, or more specifically English, acts as another social structure, which controls how the codes and programs are created and processed. As a consequence, the diglossic system becomes not just a pragmatic choice, but also a standard for drawing a line between professional and non-professional, bureaucratic and egalitarian, educated and non-educated, or even high social status and low social status.

It is also worth considering the power relations underlying English as a better tongue and Chinese as the mother tongue. As a latecomer to the digital economy, China, and Chinese programmers, struggle within a technical assemblage that has been well established by Western countries. Being disenfranchised by rule designing and process designing, Chinese programmers are located at the bottom of the labour division pyramid, and become even more passive during the process of programming. Here we are not to ignore or disparage the natural competitive advantage of English being a programming language, but move further to think why English can become a factor in explicating software programmers' articulations in China. Here articulations on programming languages not only reflect the hardship and difficulty for China, but also the struggle for the Global South to establish its own coding system in the global programming system.

Articulations of making the alternative

On 25 May 2015, China's largest online travel agency (OTA) Ctrip was hacked: its search engine service was not accessible, and users could not access its page links. According to a report in *The Wall Street Journal*, the Nasdaq-listed company was hacked by "unidentified sources," leading to a loss of more than US$15 million. This incident stunned the general public. This case provides new perspectives to rethink the power of algorithms as well as the "techies" who master them. When the assemblage of programming attempts to homogenise its component articulations, it may confront counter-forces from the articulations, which contain the potential for producing an alternative. Although individual programmers from small companies are confined within a technologically structured framework, this does not mean they are necessarily subject to biased power relations. Rather, fieldwork data indicate that grassroots programmers may construct their identification with technology inside and outside of their work. This dual narrative presents the possibility of individual empowerment within an even wider collaborative solidarity.

Salary bargaining is a case in point. Empowered by their programming skill, programmers' requests for salary increases are considered more seriously. Evan is a front-end programmer for data visualisation in the company where I completed the internship. He shared with me regarding his success in negotiating a higher salary over the last two years at AoC:

> The first time I asked for a salary rise was half a year after I came to AoC. (...) I finished the first project well, which was rather satisfying. (...) I talked with our boss, and he agreed to my request. My salary increased from 4,500 (*yuan*, approx. US$700) to 6,000 (*yuan*, approx. US$930).[8] (...) The company recruited another guy, Simon. I had to teach Simon everything; meanwhile, I also had to finish my work. I asked for a salary rise again. The tightwad (a nickname for the boss that company staff used in private conversation) refused. I didn't give up. (...) When I told him I was considering leaving the company and looking for another job, he agreed. [laughing] He needs me, and he cannot afford for me to leave.[9]

Being technically professional can allow individual empowerment (Lehdonvirta 2016; Lindtner 2014), as it can increase programmers' bargaining power in wage or promotion negotiations. Small companies such as AoC tend to compromise on capable employees' requests, since it is not always easy to find the right replacement. In contemporary China, there is a significant mismatch between market demand and the nurturing of higher education IT talent, making it hard for small companies to find the right programmer. The cost of training and maintaining employees cannot be ignored. In this regard, companies may take technicians' requests more seriously.

In the daily practices of IT programmers, teamwork is another important tool to show their empowerment. In IT companies, a team is initially formed as a project management strategy, but beyond this, teams also constitute a fundamental social framework for solidarity among programmers. Team can be of different sizes, usually more than three people, and there is a team leader who is responsible for coordination, communication, and making schedules. Through working together, programmers in a team become familiar with each other; develop shared opinions, and form a sense of belonging. Sometimes these relationships can extend from work into their private life. For example, they may hang out and play computer games together. Matteo is a Java programmer in a medium-sized IT company, with his first job specialising in remote router development (*yaokong luyouqi kaifa* 遥控路由器开发). As one of the first group of programmers in the company, he joined the programming group and worked there for three years. When talking about teams, he shared his experiences with me:

> We formed a very nice team. Our group leader was intelligent and funny. He taught us from scratch. Everybody learned very fast and

8. The Chinese *yuan* was converted to US dollars based on the fixed exchange rate on 10 August 2016.
9. The interview was conducted in Hong Kong in September 2015, when Evan came to Hong Kong to buy a Macbook for himself.

> fought for the same project. We never concealed information or techniques from one another. We shared everything. We worked together for three years… But then he [the leader] left the company because there was a disagreement between him and the boss, my previous boss, over the product design. We all knew he [the leader] was right, but the boss rejected his idea and a gap grew between them… After one month, he resigned, taking almost all of the brothers in the developing group with him. The boss lost his support and became very angry.[10]

Controlled by corporate management, which rationalises homogenisation and standardisation, individual programmers' articulations can become a resistant force to the controlling assemblage. As Wang Jing (2015) indicated, technology and community-building are equally indispensable in ICT4D. During the interview, Matteo characterised the team as a community full of consciousness, identity, solidarity, and power. Empowered by technique and skills, programmers in corporations form solid articulations that have become a force to be reckoned with (Ensmenger 2012). The story ended with the team leader taking his group to another company, where they were better paid. In the articulations of Matteo, it becomes evident that both technology and community-building are integrated into and contribute to their collective solidarity. Compensating and supporting each other, those two aspects provide programmers in China with the alternative to gain more bargaining power.

On 4 January 2015 the Premier of China, Li Keqiang, visited Chaihuo Makerspace (*chaihuo chuang ke kongjian* 柴火创客空间) in Shenzhen,[11] where he expressed his appreciation for the maker community and endorsed the creative group. This gave China's nascent makerspace a footing in the state agenda, and ignited the further deployment of makerspaces in Shenzhen. In June, when the International Maker Week was held in Shenzhen, the local government introduced two official documents: the Trial regulations to promote maker culture development (*guanyu cujin chuangke fazhan de ruogan cuoshi (shixing)* 关于促进创客发展的若干措施(试行)) and Three-year action plans to promote maker culture (2015-2017) (*cujin chuangke fazhan sannian xingdong jihua* 促进创客发展三年行动计划) to further encourage and support the maker movement in Shenzhen. The two documents state the intention to open more makerspaces, encourage new maker projects, and give financial support to maker fairs and other maker activities. It also lists 14 maker communities as key platforms for governmental support, including Chaihuo, Huaqiangbei Makerspace Centre (*huaqiang bei guoji chuangke zhongxin* 华强北国际创客中心), Seed Studio Makerspace (*xidi chuangke* 矽递创客), and Luohu Makerspace (*Luohu chuangke kongjian* 罗湖创客空间).

In Chinese, "maker" translates as "*chuangke*" (创客), a word coined by combining both "maker" and "hacker." It refers to a technology-based DIY (do-it-yourself) culture that is more concerned with physical objects and the innovation of new devices. Maker culture intersects with hacker culture and shares similar ideals and ethics such as access, networks, shared learning, and free and open information flow. The main difference is that maker culture focuses strongly on learning practical skills and carrying out technology production (Lendtner 2014; Thomas 2002). According to Wang, the distinction is important because it is through maker communities that online and offline activities converge, humanity and technology meet, and "real changes can be measured" (Wang 2015, p. 39).

As makerspaces mushroom in China, the constitution of the movement takes on a diverse and pluralistic maker scene. More "grassroots makers," like blue collar workers, freelancers, grassroots NGOs, and the wider general public, are joining makerspaces. Maker movements gain momentum when this kind of cross-connection and collaboration happens. Centred on IT and technology, makerspaces also provide grassroots programmers with good opportunities to become involved in the massive innovation movement. Facilitated by start-up toolkits and how-to guides in makerspaces, incubators, and accelerators, individual programmers take the first steps in escalating themselves to entrepreneurs or project leaders. Programmers passionately participate in activities such as start-up weekends, hackathons, co-working programs, and pitch contests, trying to carve out a path to their future advancement.

Willis is a Python software developer. After searching around Beijing, Shanghai, and Wuhan for a job, he was attracted by the favourable policies toward micro-enterprise (*xiaowei qiye* 小微企业) in Shenzhen. He came to Shenzhen, trying to start his own company. He registered his company at the Shenzhen Industrial and Commercial Bureau for free, and joined the makerspace there. He quickly found partners and started a program to manufacture infrared adapters (*hongwaixian shipeiqi* 红外线适配器). He was full of excitement when describing their ideas, as well as the somewhat complicated mechanism of the infrared adapter:

> It is a cool idea since everyone is familiar with Bluetooth, and IR blazer (红外发射器) has been forgotten. But if we can make an infrared remote controller that can help people control their TV, air conditioner, and fridge through their mobile phone, isn't that such a cool idea? What we need to do is to make an infrared adapter, and also develop an App for people's phones. (…) I like the project. You can feel the power and motivation inside of you in the project, and you think you can change the world (…). Our team argues and also cooperates with each other. It's an invaluable experience…[12]

Lindtner (2014) implies that Chinese makers see their technology production as a kind of individual empowerment. In Willis's case, programmers not only utilise their techniques, but also network themselves to hardware, finally creating their own start-ups. As programmers' engagement within the maker movement proliferates, many have moved beyond their original environment to create their own subjectivity and community solidarity. Programmers stress notions such as making, sharing, giving, learning, playing, participating, supporting, and changing. Although makerspaces adhere to openness, sharing, and access, grassroots programmers do not see themselves as isolated from the market and business. Rather, their motivation is to find a sense of community, to improve networking, and to brainstorm their ideas.

The data above indicate that the assemblage of programming in China is constructed on at least two different lines of articulation. First, it is boosted by nationwide political discourse and economic logic. In order to re-energise economic growth and build a creative society, the Central Government has been making great efforts to bridge the massive labour force with an innovation movement. Second, the assemblage of programming is also com-

10. The interview was conducted with Matteo in Shenzhen's Futian District during afternoon tea on 19 September 2015.
11. Kun Cao, Lei Li, "Prime Minister Li Visit Chaihuo Maker," *People.cn* (website), http://politics.people.com.cn/n1/2017/0220/c1001-29094476.html (accessed on 6 July 2016).
12. Willis was an IT entrepreneur in Shenzhen, and the interview was conducted in February 2016 in Shenzhen.

prised of individual programmers whose articulations involve the potential to create an alternative. Although controlled by biased employment relations, programmers are empowered by their knowledge of technology, skill, and cooperative form of work, enabling them to resist the programming assemblage that prioritises standardised management while ignoring individual welfare. By joining makerspaces and co-working spaces, grassroots programmers break the isolation between the virtual platform and the real world, and unite with each other. In another sense, the involvement of grassroots programmers extends the denotation of makers through collaboration and social participation, and transcends not only technical and professional barriers but also social class and division of labour.

Concluding thoughts

Through examining grassroots programmers' daily practices in the process of China's informatization, this article has mapped the myriad articulations such as engagement, communication, discourse, and practice that make and unmake programmers' programming assemblage. In reference to the original research questions, programmers, as the main producers of information and technology in contemporary China, have been involving themselves in the grand narrative of China's informatization, as well as the construction of a "Creative Society." Fieldwork uncovers some "hidden realities" that may be different from what people typically imagine of "working as a techie." For example, the relentless pursuit of IT crash-courses at the grassroots level, and the mixed use of Chinese and English in their coding practices. These "things [that] do not fit" (Becker 2008) call for critical reflections upon an uneven programming assemblage formed by both the national and global technology production system.

Technology by nature contains inherent biases. As Innis and Watson (2008) imply in *The Bias of Communication*, technology and communication media can be either space-biased or time-biased to serve the power centre through imposing dominant cultural standards and political ideology. As latecomers to the programming world, Chinese code techies are confronting "biased technology." Chinese programmers must follow the rules premade by Western countries, including the use of English-dominant coding frameworks, operating systems, and programming languages. Being insufficient in English and other social resources, programmers in China are disconnected from decision-making, community engagement, and the expression of opinions in global IT communities. In this sense, technology demarcates the line between the included and the excluded, and places Chinese programmers at the risk of being marginalised. This bias facilitates the social hierarchy among Chinese programmers: educational attainment, proficiency in English, and other necessary forms of social capital are all integrated into the hierarchy of technology, leaving grassroots programmers mired in unskilled and repetitive work.

The data also indicate that technology is a mixed blessing for Chinese programmers. As a means of survival, it exacerbates the precariousness and marginalisation of grassroots programmers who are far from the power centre, but the capability of technology production also offers grassroots programmers the possibility of remaking their subjectivity and of social collaboration. It shows that while overwhelmed by a web of political domination and technical hierarchy, programmers can still carve out some bargaining and breathing space through the empowerment of algorithms and codes. By working as a team and joining the globally prevalent maker movement, programmers form collective solidarity, win bargaining power, and craft their subjectivity. Revolving around a series of articulations such as peer production, creative making, and open source, grassroots programmers are embedding themselves into the state's policy of building a "creative country" that maintains "massive entrepreneurship and innovation" (*dazhong chuangye wanzhong baoxin* 大众创业，万众创新), but are also generating articulations that conflict with the overall regulated programming assemblage. This resistance to the homogenising assemblage prepares themselves for alternatives. As Lindtner (2014) and Wang (2015) have suggested, "forming subjectivity and making the change from within" can be an alternative analysis framework for a transforming contemporary China. The alternative articulations performed by grassroots programmers in China, dialectically speaking, are an attempted sublation of the capital-labour antagonism. In other words, programmers' technological practices extend beyond a binary perspective – it is not a predominant resistance versus control, or activism versus subjugation model. On the contrary, the alternative articulation produced by programmers is a process of becoming from within. This articulation shows how grassroots software engineers in China try to overcome the social antagonism between software capital and software labour.

In this article, articulations of making and unmaking the programming assemblage in modern China consist of three themes: migrant workers' involvement in a programming movement promoted by the political discourse, the technological gap between China as the periphery and the West as the centre, and the bottom-up expressions and practices to establish solidarity and form co-operatives.

Grassroots' programmers' work practices in this article highlight the enabling and constraining relations between articulations and assemblage. In other words, individual programmers' experience constitutes, while also changing, the spectacle of informatization and globalisation underway in China. As Wise (2005) argues, every assemblage is made up of "affects and effectivity." This article has sought to explain how the articulations of programmers produce particular power relations and constructions of communication, practice, and empowerment. A dialectical relation can be found in the articulations and assemblage of Chinese programming labour. Programmers' individual communication, experience, and expression constitute the IT performance on a corporate, national, and global level. Grassroots communities, however, also develop counterforces that not only refuse to be homogenised but also advocate for redevelopment and reconstruction. This self-contradiction performed by programming labour demonstrates the alternative narrative of software labourers in China trying to overcome the labour-capital antagonism. ICT has long been invoked as a site of resistance that can bring people together and make a change (Qiu 2009; Yang 2009). The mastery of technique means making a difference, either through creating the possibility of collaboration or through forming community solidarity. As part of the critical forces of social development and change, programming itself also becomes an alternative discourse that is open, expressive, and collective.

Although it would be tempting to argue that the maker movement in China is a new attempt to make social relations and rebuild "the program of society," it is nevertheless also essential to realise that the formation of maker movements is still at the very beginning of social resistance, and that the grassroots programmers involved lack structural solidarity. Through the deployment of technology and the building of makerspaces, a social networking community that revolves around IT techies is emerging. In his discussion of the distinctions of the digital age, Douglas Rushkoff (2002, p. 7) writes: "In the highly programmed landscape ahead, you will either create the software or you will be the software. It is really that simple: Programmed or be programmed." It would be incorrect to conclude that the maker movement is the way for social transformation, but it does show the dialectical possibility and provides the potential. The subversive effect created by technology and social relations entails continuous imagination.

Ping Sun is Assistant Professor at the Department of Journalism and Communication at the Chinese Academy of Social Sciences. Chinese Academy of Social Sciences, Building No. 9, Panjiayuan Dongli, Beijing, Chaoyang District (sophiesunping@gmail.com).

Manuscript received on 23 May 2017. Accepted on 26 September 2017.

REFERENCES

BECKER, Howard S. 2008. *Tricks of the Trade: How to Think about Your Research While You're Doing It.* Chicago: University of Chicago Press.

BRUNER, Edward M. 1986. "Experience and its expressions." In Turner Victor Witter and Edward M. Bruner (eds.), *The Anthropology of Experience.* Champaign: University of Illinois Press. 3-32.

DE LANDA, Manuel. 2010. *Deleuze: History and Science.* New York, Dresden: Atropos Press.

DELEUZE, Gilles, and Claire PARNET. 2002. *Dialogues II.* New York: Columbia University Press.

DELEUZE, Gilles, and Félix GUATTARI. 1988. *A Thousand Plateaus: Capitalism and Schizophrenia.* London: Bloomsbury Publishing.

ENSMENGER, Nathan L. 2012. *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise.* Cambridge: MIT Press.

FERGUSON, Charles A. 1959. "Diglossia." *Word* 15(2): 325-340.

FUCHS, Christian. 2014. *Digital Labour and Karl Marx.* Abingdon: Routledge.

GROSJEAN, François. 1982. *Life with Two Languages: An Introduction to Bilingualism.* Cambridge, MA: Harvard University Press.

INNIS, Harold Adams. 2008. *The Bias of Communication.* Toronto: University of Toronto Press.

LEHDONVIRTA, Vili. 2016. "Algorithms that Divide and Unite: Delocalization, Identity, and Collective Action in 'Microwork'." In Jörg Flecker (ed.), *Space, Place and Global Digital Work.* London: Palgrave-Macmillan. 53-80.

LINDTNER, Silvia. 2014. "Hackerspaces and the Internet of Things in China: How Makers Are Reinventing Industrial Production, Innovation, and the Self." *China Information* 28(2): 145-167.

ONG, Aihwa. 2006. *Neoliberalism as Exception: Mutations in Citizenship and Sovereignty.* Durham: Duke University Press.

OREGLIA, Elisa. 2013. *"From farm to Farmville: Circulation, adoption, and use of ICT between urban and rural China."* PhD dissertation in Philosophy, University of California, Berkeley.

PICKERING, Michael. 1997. *History, Experience, and Cultural Studies.* London: Macmillan.

QIU, Jack Linchuan. 2009. *Working-class Network Society: Communication Technology and the Information Have-less in Urban China.* Cambridge: MIT Press.

RUSHKOFF, Douglas. 2010. *Program or Be Programmed: Ten Commands for a Digital Age.* New York: OR Books.

SCOTT, Joan. 1992. "Experience." In Judith Butler and Joan Scott (eds.), *Feminists Theorize the Political.* New York: Routledge. 22-40.

SEGAL, Adam. 2003. *Digital Dragon: High-technology Enterprises in China.* Ithaca, New York: Cornell University Press.

SLACK, Jennifer Daryl. 1996. "The Theory and Method of Articulation in Cultural Studies." In David Morley and Kuan-Hsing Chen (eds.), *Stuart Hall: Critical Dialogues in Cultural Studies.* London and New York: Routledge.

STIVALE, Charles J. 2005. *Gilles Deleuze: Key Concepts.* Montreal: McGill-Queen's University Press.

SUN, Ping, and Michelangelo MAGASIC. 2016. "Knowledge Workers, Identities, and Communication Practices: Understanding Code Farmers in China." *TripleC: Communication, Capitalism & Critique* 14(1): 312-332.

TAKHTEYEV, Yuri. 2012. *Coding Places: Software Practice in a South American City.* Cambridge: MIT Press.

THOMAS, Douglas. 2002. *Hacker Culture.* Minneapolis: University of Minnesota Press.

WALLIS, Cara. 2015. *Technomobility in China: Young Migrant Women and Mobile Phones.* New York: NYU Press.

WANG, Hui. 2014. "Two Kinds of New Poor and Their Future: The Decline, Reformation of Class and the Dignity Politics of the New Poor." *Open Times* 6(1): 1-4 [in Chinese].

WANG, Jing. 2015. "NGO2.0 and Social Media Praxis: Activist as Researcher." *Chinese Journal of Communication* 8(1): 18-41.

WISE, J. M. 2005. "Assemblage." In Charles Stivale (ed.), *Gilles Deleuze: Key Concepts.* Montreal: McGill-Queen's University Press. 77-87.

YANG, Guobin. 2009. *The Power of the Internet in China.* New York: Columbia University Press.

YUE, Bin. 2015. "The129 Cententration of IT Software in China." *Modern Economy* 18: 36-37 [in Chinese].

ZHAO, Yuezhi. 2007. "After Mobile Phones, What? Re-embedding the social in China's 'digital revolution'." *International Journal of Communication* 1(1): 92-120. http://ijoc.org/ojs/index.php/ijoc/article/view/5/20 (accessed on 11 November 2016).