Leveraging Learning Collectives: How Novice Outsiders Break into an Occupation

Ece Kaynak
Bayes Business School
City, University of London
Ece.kaynak@city.ac.uk

Abstract

Existing research depicts occupational learning as predominantly happening through formal education or situated learning, or a combination of the two. How career switchers might develop occupational skills outside of these established learning pathways is understudied. This paper examines how novice outsiders break into a skilled occupation by looking at the case of aspiring software developers attending coding bootcamps. Drawing on 17-months of fieldwork in the San Francisco Bay Area, I find that bootcamps did not resemble either schools or workplaces, the two institutions that facilitate occupational learning. Instead, bootcamps scaffolded *learning collectives*—groups composed of peers and near-peers who learn collaboratively and purposefully to reach a shared goal. Within learning collectives, aspirants progressed from novice outsiders to hirable software developers despite limited access to proximate experts to learn from or legitimate peripheral participation opportunities. Three scaffoldings facilitated learning at bootcamps: First, peer team structures turned what is normally a solitary activity—writing code—into a collaborative endeavor and facilitated peer to peer knowledge exchange. Second, near-peer role structures engaged recent graduates in teaching and mentorship relationships with novices so that aspirants could access knowledge quickly and easily. Third, bootcamps encouraged aspirants to self-learn by reaching out to the expertise of the broader occupational community. This third scaffolding prepared aspirants for learning beyond the bootcamp curriculum and socialized them for an occupation with high learning demands. The outcome of this process was that novices pursuing an alternative mode of occupational entry developed both occupational skills and new self-conceptions as software developers.

Keywords: Occupations and professions, expertise, learning, careers, ethnography

Acknowledgements

I would like to thank Ruthanne Huising and three anonymous reviewers for their invaluable comments in developing this paper. Throughout this project, I gratefully acknowledge the support and guidance of Steve Barley, John-Paul Ferguson, Aruna Ranganathan, Melissa Valentine, Bob Sutton, and Hatim Rahman. I would also like to thank Arvind Karunakaran, Emily Truelove, Daisy Chung, Joelle Evans, Kate Kellogg, and Michel Anteby for their feedback. Previous versions of this paper were presented at the Academy of Management, Wharton People and Organizations Conference, Oxford Said Professional Service Firms Conference and emLyon Ethnography Workshop. I gratefully acknowledge a generous Cyber Initiative Grant from Stanford MediaX. Finally, I would like to thank my informants who were so generous with their time.

Leveraging Learning Collectives: How Novice Outsiders Break into an Occupation

The past few decades have seen significant changes in the nature of work and careers (Barley and Kunda 2001; Barley, Bechky and Miliken 2017). Changes of note include the computerization of work (Zuboff 1989; Levy and Murnane 2012); rise of virtual and remote working (Johns and Gratton 2013); proliferation of nonstandard employment relationships such as contract, part-time and gig work (Cappelli 1999; Kalleberg 2000 and 2009; Barley and Kunda 2004; O'Mahony and Bechky 2006; Halpin & Smith, 2017); algorithmic management of workers (Rahman 2021; Kellogg, Valentine, Christin 2020; Rosenblat 2018); and boundaryless careers spanning different organizations and even different occupations (Arthur and Rousseau 1996; Bidwell and Briscoe 2010). While our understanding of work and careers evolve, our understanding of how people learn occupational skills and enter different lines of work has remained relatively unchanged.

Entry into occupations is traditionally understood as a process of learning overseen by occupational incumbents (Van Maaneen 1973, 1978; Trice 1993; Anteby, Chan and DiBenigno 2016). In this process, novices learn both the task area of the occupation (Abbott 1988), as well as the particular behaviors, attitudes and worldviews shared by the occupational community (Van Maanen and Barley 1982). Learning happens through formal education (Moore 1976; Schleef 2006), or situated learning on the job (Lave and Wenger 1991; Brown and Duguid 1991), or a combination of the two (Pratt, Rockmann, and Kaufmann 2006). This is the case not just for highly skilled occupations and professions (Becker, Geer, Hughes, Strauss 1961), but also for middle skill occupations and craft work (Van Maaneen, 1973; Riemer 1977; Fine 1996). Novices who pursue established occupational learning pathways are considered legitimate newcomers in the eyes of the occupational community and external audiences.

However, in practice, we know that occupational learning does not always unfold unproblematically, or along established pathways. For example, not all situated learning opportunities are organized to ensure skill development. Novices can experience difficulty observing or partaking in professional practice, and fail to develop necessary skills (Lave and Wenger 1991; Bharatan, Swan, and

Oborn 2022). For example, in his study of surgical residents trying to learn robotic surgery, Beane (2019) finds that the technological mediation of work limited novices' access to professional practice and only a few opportunistic learners were able to develop robotic surgical skills by following shadow learning strategies. Developing occupational skills is also problematic for freelancers, who do not have employers to sponsor their learning (Barley and Kunda 2004). In their study of freelancers trying to expand their skillsets into new domains, O'Mahony and Bechky (2006) find that freelancers had to bluff about their existing skills and experience, and discount their rates in order to obtain stretchwork that would allow them to try their hands at a new technology and learn on the job. These studies challenge and add to our traditional understanding of occupational learning and skill development. Yet, in all of these cases, the learners in question are insiders to the occupational community within which they pursue alternative learning strategies. How *outsiders* might develop occupational skills in order to break into an occupation without following established learning pathways—without going back to school or obtaining situated learning opportunities—is understudied. This phenomenon deserves scholarly attention as more and more workers switch occupations in the course of their careers (Arthur 2008; Hall 2004; Baruch and Vardi 2016; Tolbert 1996), and may not choose to or not be able to go back to school for several years, or obtain low-paid or unpaid internships, as these learning pathways are costly and time consuming. Little is known about the alternative learning strategies that career switchers pursue in these circumstances, especially when trying to enter a skilled occupation. This is despite the fact that under-institutionalized career transitions are becoming increasingly common in contemporary careers (Ibarra and Obodaru 2016; Demetry 2017 Petriglieri, Ashford and Wrzesniewski 2019).

I examine how novice outsiders break into a skilled occupation through a study of aspiring software developers attending coding bootcamps, which are relatively novel, short-term vocational training programs that promise to transform novices into hirable occupational entrants. My preliminary observations suggested that bootcamps did not resemble either of the two institutions that facilitate occupational learning: Schools, which are sites for formal education; or workplaces, which allow for legitimate peripheral participation opportunities (Becker et al. 1961; Moore 1976; Fine 1985; Anteby 2013; Schleef 2006).

Bootcamps did not resemble schools because they offered minimal expert teaching. Bootcamps did not resemble or necessarily lead to legitimate peripheral participation opportunities either, because these internships and new graduate roles were reserved for newcomers pursuing established entry pathways into the occupation[1]. Given these challenges, how did aspiring occupational entrants develop necessary skills to become hirable software developers?

To answer this question, I conducted 17-months of fieldwork in the San Francisco Bay Area, including 80 interviews and 8 months of nonparticipant observation at two coding bootcamps. I found that bootcamps scaffolded *learning collectives*, which I define as groups composed of peers and near-peers who learn collaboratively and purposefully to reach a shared goal. The term near-peer refers to learners who are several months ahead in their learning journey and who can therefore help and teach more junior learners, even though they are not yet experts themselves (Topping 1996; Whitman 1988). Within the learning collectives I studied, near-peers were aspirants who had recently completed the bootcamp curriculum and who stayed on to teach and hone their skills before applying to jobs. The goal shared by both novices and their near-peers was thus to progress from outsider to hirable software developer.

Three scaffoldings enabled occupational learning at bootcamps: peer team structures, near-peer role structures, and encouragement to self-learn[2]. Peer team structures, in the form of pair programming and group projects, made coding a conversational and collaborative activity and facilitated knowledge exchange between peers. Near-peer role structures, in the form of near-peer instructor, teaching assistant and mentor roles, engaged recent graduates in teaching relationships with novices, thus allowing novices to access help quickly and easily. Finally, aspirants were encouraged to learn how to learn programming on their own by reaching out to the expertise of the broader occupational community. This final scaffolding socialized aspirants for an occupation with constant learning demands, and prepared them to continue

[1] At the time of fieldwork, a very limited number of apprenticeships were offered by tech companies for bootcamp graduates in an attempt to diversify their talent pipeline. However, these apprenticeships were so few in number that they did not constitute a normal stage of learning after bootcamp. The overwhelming majority of bootcamp graduates had to search for entry level software developer jobs in competition with formally trained engineers.

[2] I thank the anonymous reviewer who helped me coin the terms that describe the scaffoldings.

building their expertise beyond the bootcamp curriculum[1]. The outcome of this learning process was that aspirants were able to develop both occupational skills and new self-conceptions as software developers.

In what follows, I first provide an overview of the different conceptualizations found in the literature of how novices acquire occupational expertise: occupational socialization, situated learning, and shadow learning. I show how, in all of these prior conceptualizations, novices have insider access to proximate experts and other learning resources of the occupational community by virtue of pursuing an established learning pathway. This leaves unexplained how career switchers pursuing alternative learning pathways, and who therefore do not benefit from the same insider status, might develop occupational skills. After this review of the literature, I introduce my research setting before delving into the findings.

HOW NOVICES ACQUIRE OCCUPATIONAL EXPERTISE

Occupational Socialization

One of the primary ways that expertise is organized in our society is through occupational communities (Abbott 1988, 1991). Occupational communities owe the reproduction of their task area expertise, as well as their common meaning systems, first and foremost to the particular ways in which new entrants to the occupation are trained and socialized (Van Maaneen 1973, 1978; Anteby, Chan and DiBenigno 2016). Studies of entry into skilled occupations and professions depict socialization as a long and elaborate process, a well-established pathway to membership guarded by occupational incumbents who control access to practice (Wilensky 1964; Abbott 1991; Freidson 1986). Abbot (1988: 84) notes,

> It is common to create rigid entry standards, coupling extensive education with several levels of examination prior to formal entry into the profession. This is part of a structure of control… It protects recruitment, controls professional numbers (and consequently professional rewards), and guarantees a minimum standard of professional ability.

The occupational entry process for skilled occupations and professions thus involves a period of formal training, often coupled with a period of internship, apprenticeship or residency where learning continues on the job and becomes more relevant to actual practice (Trice 1993; Bailey and Barley 2011).

In the course of occupational socialization, novices learn both the task domain of the occupation, as well as the patterns of thought and action that are shared by the occupational community (Van Maanen and Barley 1984; Michel 2011). Incumbents act as 'socialization agents' in this process (Van Maanen 1973; Saks and Gruman 2012, Ranganathan 2017). They transfer knowledge and model appropriate behaviors and attitudes (Anteby 2013). Novices emulate incumbents, who represent models for desirable future selves (Ibarra 1999). The interactions between novices and incumbents also provide feedback for novices on their progression from novice to expert (Pratt et al. 2006). Through such feedback, incumbents provide social validation for novices' newly forming self-conceptions as members of the occupation (Ashforth, 2001). Novices are not merely passive recipients in the socialization process either. They actively engage in information seeking behaviors to understand the demands of their new role (Morrison 1993; Saks and Ashforth 1997). Through the transmission of knowledge and worldviews between incumbents and newcomers, new generations of occupational entrants reproduce the expertise that is the jurisdiction of their occupation (Van Maanen and Schein 1979; Fine 1985, 1996; Schleef 2006).

Importantly, studies of occupational socialization all describe learning processes where novices have unproblematic access to proximate experts to learn from and emulate. Whether we are talking about the training of a doctor in the classroom and at the hospital (Becker et al. 1961), or the policeman at the academy and on patrol (Van Maanen 1973, 1978), the assumption is that novices have access to incumbents who sponsor their learning and socialization (Ranganathan 2017). It is thought that only through such close contact can incumbents shuffle newcomers through the cognitive and behavioral changes necessary to 'become' a member of an occupation (Anteby et al. 2016).

Situated Learning Through Legitimate Peripheral Participation

A second stream of research that offers insight into how novices develop occupational expertise is the literature on situated learning. Situated learning refers to the process by which newcomers learn the expertise and ways of being of an occupational group by participating in their shared practice (Lave and Wenger 1991; Brown and Duguid 1991). The central idea is that learning and doing cannot be separated. Newcomers to an occupation learn relevant skills through participation as they work among incumbents within a community of practice (Wenger 1998). Novices initially partake in peripheral tasks and slowly acquire the necessary skills and knowledge that lead to mastery (Brown and Duguid 1991; Jordan 1989). Lave and Wenger (1991) proposed the term 'legitimate peripheral participation' to describe this relationship between novices and expert practice.

Inspired by the model of craft apprenticeship, situated learning theory deemphasizes formal teaching, and instead highlights the importance of learning through imitation and through examples, or 'occasions of use,' as practitioners communicate with each other in the course of shared practice (Jordan 1989; Orr 1996). Situated learning theory argues that, for any given occupation, the primary site for learning is not the school but the workplace. For example, a medical intern or resident learns how to practice medicine by observing senior doctors at work. As the novice's understanding and skill increases, they are invited to assist more and more in the core professional activity until they become experts themselves. Jordan (1989) explains that "the activities to which the apprentice is a witness and, by stages, a contributor, are organized around work to be done, and whatever teaching or learning may happen is coincidental to that overriding concern" (p. 933). Many occupations are either learned completely through situated learning (e.g. the crafts) or the entry process involves some period of situated learning following formal education (e.g. law enforcement, medicine, engineering).

A key outcome of occupational learning and skill development is a change in one's conception of self. Both occupational socialization and situated learning literatures, in fact, talk about changing self-conceptions (e.g. Hughes 1958, 1971), or the development of 'identities of mastery' (Lave and Wenger 1991), or identification with an occupation (e.g. Becker and Carper 1956) as an important aspect of learning.

As novices observe incumbents, they learn not just how to do the work of the occupation, but how to think of themselves and conduct themselves as members of that occupation. By observing incumbents, novices learn what it means to be a carpenter, teacher, doctor, or policeman in society, and start to see themselves in that light (Hughes 1958). This change in one's sense of self is a key part of the transition from layperson to occupational member (Van Maanen and Barley 1984; Ibarra 1999; Pratt et al. 2006).

Similar to studies of socialization, situated learning theory argues that, for effective learning to take place, novices need unproblematic access to proximate incumbents. Novices are able to learn both tacit and esoteric knowledge through close observation of and participation in expert work. In fact, Lave and Wenger (1991) in their seminal book talk about the importance of access in situated learning, arguing that access is so fundamental to effective learning that "in a sense, all we have said so far is about access" (p.101) and without it, novices fail to learn (Marshall 1972). Contu and Willmott (2003), similarly point to the importance of power relations that can impact who has access to key learning opportunities.

Shadow Learning

What happens when access to key learning opportunities is obstructed? Recent studies have started to explore various unconventional learning strategies that novices employ, for example, to overcome barriers to legitimate peripheral participation opportunities. One such unconventional learning strategy involves norm- and policy-challenging shadow learning practices (Beane 2019). When robotic surgery, compared to open surgery, prevented surgical residents from working alongside incumbents for long periods of time and learning via participation in professional work, Beane (2019) finds that most surgical residents failed to learn robotic surgical skills. However, a few residents leveraged alternative learning resources within the occupational community, and managed to develop robotic surgical skills outside of their residency programs. These novices, Beane (2019) argues, acted opportunistically and in isolation, for example by going to work alongside a lower status attending physician who would be more willing to trust novices from a prestigious residency program to work the robotic surgery device, or by working with a superstar attending physician who ran multiple parallel surgeries and hence could not supervise residents

as closely, giving them more access to the device. Successful learners also engaged in abstract rehearsal by using simulators, rather than learning from incumbents directly in the course of surgery. These shadow learning practices demonstrate that situated learning arrangements should not be conceived of as being tightly bounded, and that novices can potentially exercise agency and shape their own opportunities for legitimate peripheral participation. In a similar line of research. Bharatan and colleagues (2022) studied how newcomer navy cadets negotiated access to situated learning opportunities on merchant ships that would enable them to progress from cadets to officers, when such learning opportunities were not ordinarily offered to them as part of their everyday training. While the authors note that these learning practices were not necessarily counter-normative, they were nonetheless strategic and opportunistic, and allowed a small group of cadets to access learning opportunities beyond what was available to their broader newcomer cohort.

Comparison of Different Conceptualizations of Occupational Learning

Table-1 lays out some of the key similarities and differences between the different conceptualizations of occupational learning—occupational socialization, situated learning and shadow learning—that lead to entry into skilled occupations and professions.

---Insert Table 1 Here----

In terms of the role of formal instruction, occupational socialization literature considers both formal education and on the job training as important aspects of learning. In fact, many studies of socialization trace novices from the classroom to the workplace (e.g. Van Maanen 1973; Fine 1985). Situated learning theory, on the other hand, deemphasizes formal instruction. However, in the case of skilled occupations, there is an underlying assumption that novices come from the same formal educational background, which is a prerequisite to access situated learning opportunities. A person cannot become a surgical resident without a degree in medicine, nor can a novice undertake clinical training in nursing without being part of

a formal nursing program. This assumption of a shared educational background applies to shadow learning as well, since the concept is an extension of situated learning theory.

In each of the three conceptualizations of occupational learning, novices are considered insiders to the occupational community from day one, albeit in a neophyte role. As legitimate newcomers, their learning is sponsored by incumbents. Socialization literature envisions a more active teaching role for incumbents, involving the explicit transfer of knowledge (Saks and Ashforth 1997), while situated learning assumes that learning takes place in the course of practice, without explicit teaching (Orr, 1996). However, in both cases, the status of insider implies that novices enjoy legitimate access to the learning resources of the occupational community. Shadow learning happens when access is obstructed. However, once again, due to their insider status, novices acting strategically and opportunistically can leverage alternative learning resources within their occupational community (e.g. opportunities to work alongside lower status experts, or in the operating rooms of superstar physicians, or by using a simulator device at the hospital) (Beane, 2019).

Finally, the outcome of all three conceptualizations of occupational learning is that novices are given license to practice, sometimes even irrespective of skill development (e.g. Beane 2019), as a result of partaking in an established learning pathway designed and sponsored by occupational incumbents. These studies do not consider how novice outsiders might develop the necessary skills to move through the inclusion boundaries of an occupation without following established learning pathways. One way to investigate this phenomenon is to look at the case of career switchers who cannot afford to or chose not to go back to school or take internships or apprenticeships. Such noncodified career transitions call for alternative learning strategies that are understudied (Ibarra 2003; Ibarra and Obodaru 2016). Especially in the case of skilled occupations, we know very little about how aspirants might obtain expertise outside of an occupation's established learning pathways. This paper investigates this 'breaking in' journey by analyzing the case of aspiring software developers attending coding bootcamps.

RESEARCH SETTING AND METHODS

Coding bootcamps offer a unique setting to study how novice outsiders break into a skilled occupation. Bootcamps emerged in 2011, during the second technology boom, as an alternative pathway for entering careers in software development. They are short term vocational training programs focusing on web application development, and sometimes mobile application development. The vast majority of learners at bootcamps have four-year college degrees in unrelated disciplines, and wish to switch careers into software development without going back to university. The idea and promise behind the bootcamp model of occupational entry is to transform novices with no background in programming into hirable occupational entrants through a short period of intense learning (Waguespack, Babb, and Yates 2018). In that sense, bootcamps resemble immersion-based language programs (Scott, Holzman, Ris and Biag 2017). The past decade has seen a proliferation of bootcamps not just for coding, but also for data science and UX/UI design, among other topics, in North America and other parts of the world.

How Bootcamps Compare to Established Pathways for Occupational Learning

While coding bootcamps and other nontraditional workforce development programs have been hailed by policymakers as a way to address the STEM skills gap in the United States (Caren, Bentz, Cataldi and Sanders, 2019) these new and expedited pathways for occupational learning remain controversial (Thayer and Ko, 2017). Bootcamps represent a stark contrast to computer science degree programs that are often considered the desired pathway for entry into careers in software development. Like other established pathways of entry into skilled occupations and professions, computer science (CS) degree programs involve the transmission of abstract, formalized knowledge from experts to novices. For the development of job specific skills, degree programs require or encourage summer internships, which are opportunities for legitimate peripheral participation in the occupational community. By the time a typical student graduates with a CS degree, she has not only accumulated a knowledge of basic engineering, computer science, and programming, but has also gained experience working alongside professional developers in industry. The

credential received through this formal training is recognized by both employers and the occupational community.

In contrast, coding bootcamps offer training in a very limited and specialized body of knowledge pertaining to full-stack web application development. The curriculum of a typical web development bootcamp covers programming fundamentals such as database modelling and object relational mapping, working with application programming interfaces, understanding the model-view-controller framework, and deploying applications on the web (Mulas, Paradi-Guilford, Allende Letona, and Dalphond 2017: 25). Computer science theory is covered only superficially for preparation for technical interviews. Learning at bootcamps is thus vocational, intended to help novices develop practical skills sufficient to become hirable junior occupational entrants. As one learner explained:

> It was the only time I have been educated in a way that was specifically targeted to being able to do a certain thing. They give you the exact skillset you need to be able to go out and do web development. There is some theory that goes along with it. They mix it in at certain times to try to increase your understanding. But three months is not really enough time to get a whole theoretical understanding of computer science, the logic behind it… [After bootcamp] it is a very specific set of skills that you are going to walk away with, as long as you stay on top of it. Hopefully the idea is that once you get a job, you're going to learn most of what you do on the job (Clark, Aspirant, Digital Academy, Interview).

My preliminary observations and interviews with aspiring developers like Clark revealed that bootcamps offered minimal formal, expert teaching. While most full-time bootcamps were 12-to-13-week long programs, only about 9 weeks involved coding instruction. The rest of the time was devoted to building projects and job search related activities. Moreover, most of the learning I observed at bootcamps happened not between experts and novices, but among peers, and between novices and their near-peers who were only a few months ahead in their learning journey. These near-peers were recent bootcamp graduates who stayed on to teach at bootcamp while they honed their software development skills before applying for jobs. Near-peer teaching is an established practice in higher education where advanced trainees teach more junior trainees (Topping 1996; Whitman 1988). While there is not a precise definition in the literature, in the context of medical education Bulte et al. (2007: 583), for example, define near-peer as "a trainee one or

more years senior to another trainee." Within the context of coding bootcamps, near-peers who completed the curriculum and who were thus several months ahead in their learning journey could assist more junior learners going through the curriculum. Whereas in formal educational settings near-peer teaching is used to supplement expert teaching, at bootcamps I found that it was not uncommon for near-peer teaching to replace expert teaching.

Finally, upon completing bootcamp, aspirants did not receive a credential, degree or certificate, that was recognized by employers and the broader occupational community. It is important to note, of course, that software development has historically been a meritocratic occupation where skills are more important than credentials (Ensmenger 2010). A 'self-taught developer' is a legitimate occupational member category, and can command the respect of both their peers and employers. However, self-taught developers have professional experience to attest to their expertise, even though the learning process by which they initially obtained expertise and job opportunities is unknown. Aspiring developers at coding bootcamps not only did not have credentials, but they also lacked professional experience. Legitimate peripheral participation opportunities available to novices in the form of internships and new graduate roles were largely not available to aspirants, because these roles required candidates to be either students or recent graduates of CS or other relevant engineering degree programs. As a result, when novices coming from bootcamps entered the job market, they had neither the credentials nor the professional experience that are used in external labor markets to support one's claim to being a legitimate job candidate. Despite these obstacles, I found that many of my informants progressed from being novice outsiders to hirable software developers, or were able to find other coding related employment. I explore how this learning occurred.

Data Collection

There was very little academic research on coding bootcamps when I embarked on this project (Arbeit, Bentz, Cataldi, and Sanders, 2019). I employed inductive, ethnographic research methods, which are particularly valuable for building theory about novel phenomena (Glaser and Strauss 1967; Strauss and Corbin 1990). My goal was to develop a grounded understanding of the bootcamp path of occupational

entry from the eyes of career switchers (Spradley 1979). I was in the field for 17 months, from August 2016 to December 2017. I spent 8 months of my fieldwork conducting nonparticipant observation at two coding bootcamps. I also conducted a total of 80 semi-structured interviews with bootcamp graduates, founders and bootcamp staff in the San Francisco Bay Area. I conducted another 30 interviews in the Midwest to be able to understand career switchers' experiences in different regional labor markets. The data utilized in this paper are based on observations and interviews conducted in the San Francisco Bay Area.

Non-participant Observation. I initially conducted observations at over 20 learn-to-code meetups organized by bootcamps and other learn-to-code organizations in the San Francisco Bay Area. These meetups gave me an understanding of the broader field of learn-to-code organizations, and enabled me to make contacts in the bootcamp industry. I visited the campuses of eight different bootcamps. I collected information on their program structures and curricula.

I first gained access to CodeCamp (pseudonym) which is one of the oldest bootcamps in the United States, and spent two to three days a week at their San Francisco campus from December 2016 to May 2017, following two cohorts of students. CodeCamp was my primary field site, and where I conducted the majority of my observations. Afterwards, I sought a second field site in the Bay Area to assess if my observations were unique to CodeCamp, or if there were common learning processes in place across coding bootcamps (Bechky and O'Mahony, 2015). For my second field site, I gained access to DevHouse (pseudonym), which is another well-established bootcamp in the Bay Area, similar to CodeCamp in terms of age and size. Both bootcamps had several campuses across the United States. They were both in-person, full-time, 12-to-13-week long programs considered "immersive" learning experiences. The main difference between the two bootcamps was their selectivity and rigor, with DevHouse being the more selective and rigorous one. I spent 2 months observing learners at DevHouse, during July and August 2017. My data sources are summarized in **Table-2.**

----Insert Table 2 Here----

I was able to measure student outcomes for the two bootcamps and found that DevHouse was more successful than CodeCamp in facilitating career transitions both in terms of time-to-job, and initial salary. However, the two bootcamps had similar curricula and very similar approaches to learning. In fact, I was surprised to find that both bootcamps provided the same three scaffoldings to facilitate learning.

In terms of selection, both bootcamps had an admissions process that assessed the fit and aptitude of applicants, involving a coding test that aspirants were given materials to prepare for. DevHouse's admissions process was known to be very selective, whereas CodeCamp was not considered a selective bootcamp. A number of my informants from CodeCamp mentioned that they had initially applied to DevHouse, but attended CodeCamp after failing to pass DevHouse's interview and live coding challenge. DevHouse students were required to complete a total of three projects during their program, whereas CodeCamp students completed one. Overall, DevHouse was known in the industry as a more rigorous bootcamp.

At both bootcamps I introduced myself as a doctoral student. I was given access to the aspirant experience: I could come and go as I pleased and attend all events. I opted to be a nonparticipant observer because I realized early on that trying to learn to code myself would consume all of my attention and not allow for detailed observations and note taking. As an observer, I sat between rows of learners and observed them working alone and in pairs. I sat in on the lectures. I observed novices working in teams when they were building larger scale projects towards the end of their bootcamp experience. I joined them for lunch time activities, as well as several evening meetups. I attended panels where alumni gave talks to current students and advised them on their job search. I attended lectures relating to the job search process offered by career coaches at the bootcamps. These lectures covered topics such as how to build a resume, how to build an online presence, how to network through LinkedIn, and similar topics. I observed aspirants practice technical interviewing with each other or help improve each other's resumes. I hung out in the alumni spaces at CodeCamp and DevHouse and engaged in countless informal conversations with job seekers. I attended several alumni events where job seekers broke into groups to commiserate over the job search process, share anecdotes as well as leads. I observed bootcamp graduates work on new projects together to

improve their portfolios. Finally, I followed my informants to tech industry networking events, one job fair, and two hackathons. I tried to limit my observations to four to five hour episodes a day so as to be able to accurately record my observations. When days ran longer due to evening meetups or events, I recorded my observations the next day.

Ethnographic Interviews. I conducted a total of 80 semi-structured interviews in the San Francisco Bay Area. Eleven of the interviews were with the founders and directors of different bootcamps and non-profit learn-to-code organizations. These interviews gave me an understanding of the industry dynamics, as well as the challenges facing bootcamps and their graduates in the job market. I then conducted 69 interviews with aspiring software developers who completed bootcamps. I recruited the majority of my interview participants (count=58) from my two main field sites. I interviewed 31 CodeCamp graduates and 27 DevHouse graduates roughly 6 months after graduation. My goal in these interviews was to develop an in-depth understanding of the occupational learning and career transition experience of my informants. I asked them about their backgrounds, how they decided to learn coding, how they decided to pursue a bootcamp training instead of a more established learning pathway, their experience at bootcamp, as well as their job search experience afterwards, and (if they were already employed) their adjustment to their first jobs. I expanded my sample with 11 interviews with graduates of five other bootcamps in the San Francisco Bay Area. I recruited these interviewees via LinkedIn or through the technology industry networking events I attended. These interviews were conducted anywhere between one month to two years after the informant had finished bootcamp.

Backgrounds of Aspirants

Most of the aspirants I met at bootcamps had recently decided to learn coding, tried their hands at it briefly to see if they liked the work and if they could do it, and applied to a bootcamp soon afterwards. At each bootcamp I visited there were a handful of CS graduates who were there to improve their coding skills, however they were a small minority and the bootcamp curriculum was not designed for them. The

majority of aspirants were career switchers who had a 4-year college degree, who had worked for a few years and were pursuing a career change to obtain better jobs. Their educational backgrounds ranged from English to Mechanical Engineering, and their professional backgrounds ranged from low-wage service work to white collar professional work. A few of the learners in the cohorts I observed were recent college graduates from unrelated majors with no work experience. These recent graduates also considered themselves as career switchers because they were trying to move from one knowledge domain to another for the purposes of seeking employment. One near-peer instructor explained that in each cohort he oversaw, "the vast majority had basically never touched code in their life before bootcamp" (Luke, Aspirant, DevHouse, Interview). Another aspirant reflected on the diversity of his peers and the learning environment this created:

> Bootcamps attract a really diverse and interesting student body. There was a woman who used to be a nanny… There was a guy who worked at Wal-Mart, stocking shelves… There was a 55-year-old veteran who wanted to start his life over. There were other people who were English majors like me. (Dan, Aspirant, DevHouse, Interview)

For an example of what student composition looked like at coding bootcamps, see **Table-3** which shows the backgrounds of aspirants who graduated in February and March of 2017 from my two field sites. As can be seen, 84% of aspirants were complete outsiders trying to move through the inclusion boundaries of the occupation. My analysis centers on the experiences of these career switchers.

----Insert Table 3 Here----

Format of Training

At both CodeCamp and DevHouse, the curriculum was divided into two parts: a teaching curriculum which lasted roughly 2 months, and time for projects and interview preparation, which occupied the last three to four weeks of bootcamp. During the teaching phase, students followed a curriculum that walked them through learning basic algorithms, how to store and retrieve data from a database, how to use

several backend and frontend languages to build full-stack web applications from scratch, and how to deploy these applications on the Internet. Bootcamps utilized a flipped classroom approach whereby students were given tutorials to watch, and readings and exercises to complete on their own the night before. The following day, classroom instruction time was spent answering questions, clarifying key concepts, and live coding parts of the assignment that were particularly important. At both CodeCamp and DevHouse lectures were brief; lasting less than an hour a day. The majority of the day was spent working on coding assignments in pairs. Students were paired with a different person from their class each day, and worked jointly on the day's coding assignments. When aspirants needed help, they could ask questions to instructors and teaching assistants who were "on the floor" all day for the purpose of answering questions.

At both CodeCamp and DevHouse formal hours were from 9:00am to 6:00pm, when everyone was required to be in the open office space. Many learners stayed into the late hours of the evening to finish up the day's work, or to prepare for the next day, or to attend an evening meetup. Some came in on weekends to utilize the space for teamwork. Once the program was over, aspirants were encouraged to continue coming to bootcamp and use the area dedicated to alumni for their job search and interview preparation activities. This coworking space function that bootcamps offered their alumni was very useful for my research purposes, as it gave me access to informants during their job search journey, as well as meeting rooms where I could conduct interviews.

Analytic Approach

I analyzed my data using ATLAS.ti qualitative coding software, following the principles of inductive analysis and grounded theory building (Glaser and Strauss 1967; Strauss and Corbin 1990). I initially open-coded my field notes and interview transcripts to identify the range of themes present in my data. This process yielded 250 codes across more than 150 documents. The overarching themes that emerged were aspirants' divergent career histories, the underlying motivations they had for wanting to learn coding and become software developers, their unconventional learning process, their constant self-comparison to computer science graduates, and finally their strategies for finding jobs after bootcamp. With

these themes in mind, I began iterating between my initial set of codes and existing literature. The learning processes I observed at bootcamps appeared as the most unique aspect of the bootcamp experience. Neither occupational socialization, situated learning nor shadow learning literatures predicted what I was seeing in my data.

Since I could not find qualitative studies describing the learning and socialization process of software developers, I spoke to a member of faculty in the computer science department at my institution who is in charge of the introductory programming course to understand the learning dynamics within the occupation. I presented to him what I was seeing in my data and sought his opinion. I also shared my findings with practitioners, asking for their input. I read bootcamps related discussion threads on HackerNews, which is an online discussion forum used exclusively by the developer community, to understand what the broader occupational community thought about bootcamps as a way of training novices. After this process of comparing my findings with the literature, and with what is known about learning programming, I went back to my data with a fresh set of eyes. I reread my fieldnotes, as well as parts of my interview transcripts where aspirants described their learning process at DevHouse and CodeCamp.

During my second pass through the data, I focused on the three major learning processes I observed at bootcamps that enabled breaking into software development: peer to peer learning, learning from near-peers, and self-learning by reaching out to the broader occupational community. I aggregated various sub-codes under these key learning processes. For example, initial codes such as ‘pair programming’ and ‘group project work’ were aggregated under ‘peer to peer learning’. Similarly ‘self-learning through documentations/tutorials/forums’ and ‘Googling’ were aggregated under the code ‘self-learning by reaching out to the broader occupational community’.

At this stage of the analysis it became apparent that three scaffoldings that bootcamps put in place facilitated these occupational learning processes: Pair programming and group projects constituted *peer team structures*, and facilitated peer to peer learning. *Near-peer role structures* in the form of instructor, teaching assistant and mentor roles for recent graduates facilitated learning from near-peers. Finally, the

instructional approach of bootcamps constituted *encouragement to self-learn*. The flipped classroom format during the teaching curriculum emphasized the primacy of self-learning over learning from an expert teacher. The constant socialization to become "self-sufficient", and the expectation that aspirants should learn and implement new technologies for their final projects all constituted *encouragement to self-learn*.

This second pass through the data also revealed that certain scaffoldings were more versus less important at different stages of the aspirants' learning journey. Peer team scaffolds were important in facilitating peer to peer learning throughout the bootcamp experience. Near-peers role structures, on the other hand, were important during the teaching curriculum, but their importance waned towards the end of bootcamp when aspirants began learning technologies beyond the curriculum, and, as a result, beyond the expertise of their near-peers. At this stage, self-learning from documentations, tutorials, forums and other resources became predominant.

The *learning collective* construct arose from this analysis to describe the aggregation of the three scaffoldings, and to emphasize the collaborative and peer and near-peer centered learning dynamics I observed at bootcamps. Whereas learning within formal educational institutions is led by teachers; and in situated learning environments learning happens as novices work alongside incumbents; learning at bootcamps was scaffolded as a collaborative activity undertaken by a group of peers, with the help of near-peers, undergirded by self-learning. After this round of analysis, I shared my key findings, and my learning collective construct with five of my informants in the course of follow up conversations and solicited their feedback. These informants validated my depiction of bootcamps as learning collectives.

In the presentation of my findings, I don't compare but pool my data from CodeCamp and DevHouse because I observed the same scaffoldings in both settings (Bechky and O'Mahony, 2015). These two organizations operated very similarly, I later learned during one of my founder interviews, because they were both modeled after one bootcamp that was seen as the originator of the industry in the San Francisco Bay Area. I also use supplementary data from interviews with aspirants from other, similar bootcamps in the region.

FINDINGS

Neither CodeCamp nor DevHouse resembled traditional educational institutions that are designed to facilitate the transmission of abstract knowledge between experts and novices. The two bootcamps housed few experts, and offered minimal formal teaching. The two bootcamps didn't resemble or lead to internships or apprenticeships either, where newcomers could work side by side with incumbents and learn through legitimate peripheral participation. Instead, bootcamps scaffolded learning collectives. In what follows I first elaborate the concept of a learning collective, and how diverse groups of learners united by a shared purpose were motivated to collaborate with each other in learning programming. I then discuss the three scaffoldings that bootcamps provided for learning collectives, and how these facilitated occupational learning. I end by showing that, as an outcome of this learning process, aspirants not only developed occupational skills but they also formed new self-conceptions as software developers.

Bootcamp as a Learning Collective

Bootcamps brought together diverse learners united by a shared aspiration to break into software development relatively quickly, without going back to school. This shared goal motivated aspirants to collaborate in learning, despite their divergent backgrounds and their lack of a common knowledgebase to build upon. One aspirant, comparing bootcamps to higher education, reflected on the consciousness he felt of a shared goal among his peers and how this created a collaborative community of learners:

> Compared to people in university with their parents paying their money, who are still a little amateur... most people at bootcamp were self-funding it. They had worked somewhere for a couple of years and they wanted to switch careers. Everyone was spending their own money, and they were working super hard. Everyone was very, very excited about learning the technology. Everyone was helping each other out. It was a very inclusive community of people trying to learn together. (George, Aspirant, CoderSpace, Interview)

Compared to higher education settings where learners either don't have an immediate concern about getting hired within the discipline they were studying, or where learners might have different interests

and specializations, aspirants came to bootcamps with the purpose of breaking into software development. One aspirant explained, “In college you've got your buddies. You go out to career fairs, but you're not all necessarily in the same field, so you all go off and do different things.” (Carl, DevHouse). At CodeCamp and DevHouse, however, there was a strong collective sense that aspirants were going through a difficult learning journey together, working towards a prized goal. For the majority of aspirants, attending bootcamp was a high stakes career switching process that would lead to better jobs. Some had quit their jobs to be there. Many had moved from different cities or states. One aspirant explained how the high stakes environment also contributed to a sense of shared purpose:

> You are working with a lot of people who are passionate and exciting… There was a lot of pressure and worry about, ‘I quit my job for this. What’s going to happen if I’m not understanding a topic and I fail and then this is all for nothing?’… Everyone was pretty committed and kind of on the same page. (Armin, Aspirant, DevHouse, Interview)

Aspirants worked in the same open office space throughout the day, and were available to each other for the purposes of collaborating on coding assignments and projects, building resumes and preparing for job interviews. Learning at bootcamps largely took place within this community of peers, with the help of near-peers, through self-study. One aspirant noted that “the value in the program is largely contained in the peers you meet and the way they motivate you to accelerate your learning over, for example, self-study.” (Luke, Aspirant, DevHouse, Interview). I propose the term *learning collective* to describe these groups of peers and near-peers who learn together collaboratively and purposefully to reach a shared goal. In this context, the shared goal was to progress from novice to hirable software developer. In the following sections, I examine the three scaffoldings that bootcamps put in place to support learning collectives: peer team structures, near-peer role structures, and encouragement to self-learn.

Peer Team Structures

While a shared aspiration of breaking into software development motivated aspirants to collaborate in learning, this collaboration was made possible through peer team structures. At both CodeCamp and

DevHouse, when aspirants were introduced to a new topic, it was followed by a coding assignment or small project to facilitate learning by doing. However, aspirants rarely worked by themselves. Bootcamps put in place two forms of peer team structures to facilitate collaboration and knowledge exchange between peers: a pair programming setup that was enforced early on at both CodeCamp and DevHouse, and team projects that were required towards the end of the bootcamp curriculum. Both of these scaffoldings supported peer to peer teaching and learning.

Pair programming. During the early, teaching phase of the curriculum (roughly the first 2 months at CodeCamp and DevHouse), aspirants worked on coding assignments in pairs. Pair programming involves two developers sitting side-by-side at a single computer terminal and writing code together. In industry, pair programming as a technique is used to enhance code quality, because one developer can catch the mistake of his peer or suggest a more efficient way of writing a program. However, this is an expensive and rare practice, as it involves two engineers working on the same piece of code. At bootcamps, pair programming was used as a pedagogical tool to facilitate learning among peers.

It is important to note that coding as an activity is not naturally conversational or collaborative. It is in fact a solitary activity that requires deep concentration by a lone practitioner. There is collaboration at the macro scale as different branches of code written by individuals need to be merged. However, professional developers largely work alone[3]. Pair programming made the solitary activity of writing code a conversational, collaborative endeavor that facilitated learning among peers. Pair programming was utilized not only at my two field sites, but also at many of the other bootcamps I visited in the San Francisco Bay Area.

In a pair programming setup, one person (the driver) is in charge of typing the code, while the other person (the navigator) decides the overall direction of the work and dictates the driver what to type.

[3] In normal professional practice, conversation occurs when developers ask for help from each other, give updates to each other during coordination meetings, or in the course of code reviews. Coding itself is not a conversational activity.

The driver and navigator, who are both novices in this case, discuss as they code, and take turns driving or navigating. Because of this conversational element, at any hour of the day a constant hum could be heard throughout the expansive open office space at both CodeCamp and DevHouse. Sometimes partners progressed in lock-step as they thought out loud. For example, I was observing Adam and Tim pair programming one day, working on an assignment that asked them to parse a CSV file using Ruby, which was the first coding language they learned. They needed to figure out how to search the file for people with the same email domain name.

> Tim and Adam are staring at the screen. Adam remains quiet for some time.
> Tim: What is your thinking right now?
> Adam explains.
> Tim: Okay, that sounds reasonable.
> Adam starts typing. Tim is following and approving.
> Adam: Do we delete most of this? [He selects a line of code with his cursor and looks at Tim]
> Tim: Yeah we can delete most of it… Can you go back to the debugger real quick?
> Adam opens the debugger and they read it together.
> Adam: This part is fine.
> Tim: Yeah

As can be seen in this short vignette, both aspirants were able to follow what the other was doing and offer suggestions as they tried to figure out how to solve a problem in their code. In other cases, one person could have a better grasp of the day's topic than their partner, and this could create a temporary teacher-learner dynamic. One aspirant explained,

> The driver should not be typing anything that the navigator didn't navigate them to do, but there's obviously flexibility there. If I felt the person was not really competent or prepared for the day, I would usually talk more even if I was driving. For example, I'd say, 'We need something that will do this.' (Sal, Aspirant, DevHouse, Interview)

Aspirants at both DevHouse and CodeCamp were advised to help each other and teach each other when they noticed their partner was struggling with a topic. A career coach at DevHouse, for example, told a new cohort of aspirants to "practice being a patient teacher" to each other. He said,

> Sometimes the person you are working with will want to charge through the material. Sometimes it feels like your partner is not prepared, not getting stuff, and you feel like they're holding you back… It is your job to make sure your partner understands what you are doing. If you are the one who is more comfortable with the material, practice being a patient teacher. Communicate your knowledge. (Chris, Staff, DevHouse, Fieldnotes)

This culture of sharing knowledge and of teaching each other was echoed at CodeCamp. One staff member explained to aspirants that the norm at CodeCamp was to collaborate in learning. She said,

> Our learning environment here is very different than a traditional school environment… In traditional education we learn to be competitive… Here we don't compare ourselves to each other and we don't compete. We have a collaborative learning environment where teaching each other is the norm. If you feel you know something, teach it to another person… We are all learning a new skill here. (Amy, Instructor, CodeCamp, Fieldnotes)

Socialized in this manner, aspirants learned to see each other as sources of knowledge. If they had knowledge that others lacked, they offered it voluntarily; and similarly, they didn't shy away from asking help from their peers. Engaging in such a teaching/learning dynamic was viewed as being beneficial to both parties. The learner benefited from obtaining quick and easy access to knowledge from someone sitting close to them rather than struggling with the material on their own. The person in the teacher role, on the other hand, benefited from verbalizing and trying to explain what they knew. Many aspirants reported in interviews, without being prompted, that their comprehension improved when teaching, because it made them retrieve and communicate newly obtained information.

The norm of teaching one's peers and learning from peers was first established in the course of pair programming, as this setup forced aspirants to work together on the same task in a conversational manner and explain to each other their thought processes. One aspirant explained how pair programming was beneficial in his learning journey because it enabled such knowledge exchange between peers at a time when both parties didn't have a good grasp on the expertise:

> I think it is valuable especially earlier on, because for one, if you are behind, then you can then sit by your partner and he is teaching you. I mean they have Teaching Assistants but, you know, as far as how many there are versus students, it's good

> to let people teach each other. Plus, if I happen to be a little bit more on top of the material than my partner, then I'm basically teaching him or her, and that's great for me. That helps me learn better. You learn, I think, a lot better as you teach something. And maybe you think you know something and you think you're teaching but then they correct you. So that's helpful too. (Sam, Aspirant, DevHouse, Interview)

In the course of conversation, as aspirants worked on the same piece of code together, gaps in knowledge were exposed and addressed collectively. Aspirants worked in a constant stream of conversation which made them verbalize their knowledge, make connections between different pieces of information, and recognize and fix knowledge gaps and misunderstandings. I observed such an interactional, collaborative working style being established in the first two months as aspirants pair programmed every day. Afterwards, when aspirants stopped pair programming and moved on to building larger projects alone or in teams, the highly interactional working style continued. Aspirants continued to rely on their peers for quick access to knowledge.

Group projects. A second peer team structure that facilitated peer to peer learning was the group projects that aspirants completed. For these projects, aspirants were asked to build web applications from the ground up. Some built projects that were based on novel ideas, while others built clones of popular web applications such as Facebook, Twitter or Soundcloud. During their group projects, aspirants continued the practice of talking through problems they faced and exchanging knowledge with their peers. Thus, the practice of assessing situations in terms of who has the required knowledge and either seeking or offering that information continued.

Aspirants took on a 'teacher' role towards their peers casually and situationally when the need arose, and stepped out of the role at the end of a teaching encounter (Goffman 1961; Bailey and Barley, 2011). The below exchange shows how aspirants took on teaching roles towards one another depending on who was more comfortable with the topic at hand. I was observing four aspirants in the early weeks of bootcamp trying to plan how to build a short team project, an application similar to Wikipedia, using Ruby on Rails, which is a web application framework written in Ruby—a coding language that many bootcamps

taught novices as their first coding language due to its similarity to plain English. The group huddled around a whiteboard and decided that their first task was to decide how to organize the information that would be contained in the application's database, using their knowledge of associations and Active Record, which is a tool that simplifies how a programmer works with the application's database. James felt he understood the topic and, without hesitation, went up to the whiteboard and picked up the marker. He started writing down the associations, in other words the linkages between different classes of objects such as 'user', 'article', and 'article category' that they would need in a Wiki application. After a few minutes of listing how these objects should be associated with each other, his friend Adam interfered,

> Adam: I think we're at a good point with Active Record. Let's go on to user stories [*User stories are a method for representing the different ways in which a user can interact with the application*]
> James: Okay. I'm not so good at that. Will someone else take the marker?
> While waiting for someone to step up, Adam started thinking out loud.
> Adam: As creator, I should be able to create or edit [a wiki entry].
> James is still at the whiteboard and, next to Creator, he writes down "create" and "edit". Adam looks around, then defers to Kelly whom they know has prior experience with user stories. James encourages Kelly to speak up.
> James: Is this how we make user stories?
> Kelly, from where she is sitting, begins teaching.
> Kelly: One good example of a user story is user login. The way I like to do it is to do a wireframe. You get the routes out of the way too.
> (CodeCamp, Field notes)

This episode of peer-to-peer teaching occurred when aspirants were building mini projects using material covered in the curriculum. They had all studied Active Record that week and were learning how to apply their knowledge through a short project building a wiki application. James took on a teacher role because he felt he had a better grasp on the topic. The team however had not learned how to create user stories before, and for that knowledge they relied on Kelly, who had worked as a Project Manager in her previous career and had some experience with user stories. Even though her knowledge was far from perfect, it was a starting point that the group could build on.

Group projects and pair programming were formal structures put in place to facilitate peer to peer learning in all of the bootcamps I visited. Individual bootcamps could also have more idiosyncratic ways

of encouraging peer-to-peer teaching and learning. Two other bootcamps I visited, for example, encouraged aspirants to organize short lecture sessions to teach their peers a new technology they had learned. One bootcamp had a formal evening hour dedicated to such peer-to-peer teaching sessions.

Aspirants talked about the cognitive and emotional benefits of peer-to-peer teaching in interviews without being prompted. Besides the cognitive benefits of information retrieval, of making connections between different pieces of information, and having others affirm or fix your understanding, aspirants talked about how teaching one's peers could be a source of motivation and a boost to confidence. As one aspirant put it, "[When you teach] you feel knowledgeable yourself, and by saying it out loud you realize how much you know" (Kim, Aspirant, CodeCamp, Field notes).

Reliance on peer-to-peer learning dynamics at coding bootcamps could benefit some learners more than others. Sometimes an aspirant who was ahead of her peers could find herself in an asymmetrical exchange relationship to the community, where she offered more help than she was able to get. One student at CodeCamp who was more knowledgeable when she started bootcamp, for example, realized that she quickly progressed ahead of her peers and found herself rarely in the learner role in these knowledge exchanges. She said, "Bootcamp helped me with confidence. At the bootcamp I helped a lot of people more than I was helped. So that made me feel, 'Oh I do understand stuff...'" (Jill, Aspirant, CodeCamp, Interview). As this quote suggests, while Jill expressed benefiting from teaching her peers from an emotional perspective, she found herself in an asymmetrical exchange relationship. Another aspirant who completed an online, self-paced bootcamp where the community conversed through an online chatroom explained this asymmetry as happening between learners who progressed faster in the curriculum and left their peers behind:

> I definitely noticed that the further I got in the curriculum—even though we're finishing TechSpace I'm very much a beginner—as I got towards the end I became one of the experts. So, as you progress… the ability for other people to help me, the pool from which I could ask for help kept getting smaller, and smaller, and smaller. (Dathan, Aspirant, TechSpace, Interview).

These quotes suggest there may be drawbacks to peer-to-peer learning for aspirants who outpace their peers, and find themselves disproportionately in teacher, as opposed to learner roles in knowledge exchanges. For questions that could not be resolved within peer-team structures, aspirants turned to their near-peers, who helped and taught novices in the roles of mentor, teaching assistant and instructor.

Near-Peer Role Structures

Another way in which CodeCamp and DevHouse supported and structured learning collectives was by creating near-peer role structures that put recent graduates in teaching and mentorship relationships with newcomers. Expert instruction, whether explicitly done or subsumed in the practice of work, is central to our traditional understanding of occupational learning. However, expert teaching played a minimal role within learning collectives. First of all, the flipped classroom format deemphasized teaching. Lectures every day were brief. Most of teaching happened as students asked questions to instructors and teaching assistants while working on coding assignment and projects. Moreover, many instructors and teaching assistants were not experts, but rather near-peers who had recently completed the bootcamp curriculum. I found that it was common practice in the bootcamp industry to hire recent graduates to fulfill teaching roles. These near-peers were only several months ahead of the novices in question, but as a result of having gone through the curriculum, could teach newcomers even though they were not yet experts themselves. Near-peers' reason for choosing to stay at bootcamp rather than starting their job search immediately was to start earning a salary while continuing to hone their skills as they helped more junior learners go through the curriculum.

At DevHouse—which had very successful student outcomes—almost all instruction was delivered by near-peers. At CodeCamp, teaching staff included two full-time instructors and two part-time mentors who were professional programmers. The rest of the teaching staff consisted of half a dozen instructors and teaching assistants who were near-peers. Despite having more staff who could be called experts, CodeCamp's student outcomes in terms of time to job and average starting salary were not as good as DevHouse, where near-peer instruction was the norm.

The minimal role that formal, expert teaching played at bootcamps came up in interviews

unsolicited. However, only occasionally was this presented as a disadvantage of the bootcamp environment. Students were socialized to think of software development as an occupation that required learning how to self-learn. At CodeCamp they were told by staff that, "this is not a school" and that "you are responsible for your own learning" (Fieldnotes). I was surprised to hear the same words repeated by a member of staff to a new cohort of aspirants at DevHouse. The staff member said, "Initially this place might feel like school. But it's not. This is a career switching process… Part of what we are trying to cultivate in you is self-reliance, which is so much a part of being a working engineer" (DevHouse, Fieldnotes). Socialized in this manner, aspirants normalized the minimal role that expert teaching played in their learning, and the fact that lectures were often delivered by near-peers. Laura summarized the teaching she received at Digital Academy, another reputable bootcamp, as follows:

> [At bootcamp] you are learning from other people who are learning themselves. There is not a lot of hands-on instruction from people who know what they are doing. If you need help you can get help, but you are getting help from people who went through the course three months ago. (Laura, Aspirant, Digital Academy, Interview)

The people that Laura was getting help from were near-peer teaching assistants who spent the entire day answering questions "on the floor" (Fieldnotes), referring to the open office space where aspirants sat together, huddled around terminals. Laura went on to comment that the prevalence of near-peer teaching made the target expertise seem more achievable to her. She said, "It really helped solidify the idea that this isn't unobtainable knowledge. This is stuff you can figure out on you own. [The bootcamp will] give you the tools that you need to figure out how to do this on your own." (Laura, Aspirant, Digital Academy, Interview)

The emphasis on learning among a group of peers with the assistance of near-peers, and self-reliance in the learning process reduced Laura's learning anxiety by making her view the target expertise as obtainable through a nontraditional learning pathway. Others treated near-peer teaching simply as a matter of fact. When I confronted one DevHouse graduate in a follow-up interview about the fact that most instruction was delivered by near-peers, he said, "I didn't even think of them as recent graduates, for the

most part. They just were people who knew how to do this stuff that I was trying to do" (Sal, Aspirant, DevHouse, Interview).

The practice of near peer teaching also served to make the learning collective model of occupational entry more cost effective and scalable. Since professional software developers command a high salary in the labor market, hiring many expert instructors was cost prohibitive. Instead, bootcamps hired large groups of near-peers who were present and accessible all day long in the roles of instructor, teaching assistants and mentors. Having a large group of constantly available near-peers allowed novices to access help quickly when they needed it, and not spend long periods of time "being stuck", struggling with technical complexity. Remaining stuck for extended periods of time was seen as being demotivating when trying to learn a new skill. Aspirants were instructed by staff to ask for help and not remain stuck on the same problem for long. For example, at CodeCamp, aspirants were told by staff, "Don't be stuck more than 10 minutes, ask for help" (Field notes).

The amount of help aspirants were able to get from their near-peers varied between the teaching portion of bootcamp, and the last month devoted to projects. Near-peers were more knowledgeable and hence more helpful for subjects covered in the teaching curriculum. To be able to answer any question beyond the curriculum, however, near-peers would need to have come across that information through their own self-studying efforts. As a result of this limitation, I found that near-peer teaching waned towards the end of bootcamp, as aspirants started working on novel technology stacks for their projects. During this project phase of bootcamp, aspirants opted to learn new programming languages or frameworks that were in demand in the labor market. If a near-peer happened to study the same technology in their own time, they could offer assistance. However, this was not always the case. For example, one project team that I was observing at CodeCamp was trying to learn React, which is a front-end JavaScript library that they wanted to learn because it was in high demand in the labor market at the time. Nick and Cindy related how they could not get help from their teaching assistants because they had not studied React either. Eventually they got help from an alumni mentor who was volunteering at the bootcamp in the evening. This recent graduate was able to help Nick and Cindy because she had learned React at her job.

> I find Nick and Cindy in the morning, who are telling me how they overcame the "fear and despair", as they put it, of trying to learn React.
> Nick: Well, I'm exaggerating a bit but, we are using React. And we had a very hard time trying to understand it. We were really bashing our heads.
> Cindy: We are trying to leverage a bunch of React and Redux.
> Interviewer: When did you start trying to learn React?
> Nick: Saturday.
> Cindy: We're out of the really bad weeds right now, and we're in the normal weeds.
> Interviewer: Could none of the instructors help?
> Nick: We've asked but no one really knows React.
> Interviewer: How about the graduates?
> Nick: The evening mentors have been super helpful.
> Cindy: Because they're learning React at their jobs or for interviews.

These two aspirants overcame the initial difficulty of understanding and working with a new technology by finding an alumni mentor who wasn't a part of the regular teaching staff, but who was volunteering at the bootcamp after work. As can be seen from this example, due to the limitations of the teaching staff's expertise, near-peer teaching waned towards the end of bootcamp. Overtime, aspirants began to rely more and more on their own ability to seek information and apply newly obtained information to their work. In the next section, I talk about the final scaffolding that supported learning at coding bootcamps: encouragement to self-learn.

Encouragement to Self-Learn by Reaching out to the Expertise of the Occupational Community

Bootcamps encouraged and provided opportunities for self-learning as the third scaffolding of learning collectives, to push aspirants to learn beyond the bootcamp curriculum, and prepare them for careers in software development. During the teaching phase of bootcamp, both the flipped classroom format, and the minimal classroom teaching that was provided, encouraged self-learning. In the final three to four weeks of bootcamp when aspirants worked on their own projects, they were expected to fully rely on their self-learning skills as they learned and implemented new coding languages and frameworks of their choosing. Aspirants did this by reaching out to the expertise of the broader occupational community.

To access expertise beyond the bootcamp curriculum, aspirants learned to navigate a multitude of knowledge artefacts created and freely shared online by the broader occupational community of software developers. These knowledge artefacts included video tutorials, documentation, question and answer forums (e.g. Stack Overflow), blog posts, and public code repositories (e.g. GitHub). Documentation refers to users' manuals for software tools, typically produced by the creators and maintainers of the tools. When learning a new technology, aspirants initially consulted these documentations. Aspiring developers also heavily utilized Stack Overflow, a question and answer forum that is the most widely used platform for accessing a crowdsourced body of programming knowledge (Vasilescu, Filkov, and Serebrenik 2013; Bosu, Corley, Heaton, Chatterji, Carver and Kraft 2013; Wang and Jiang 2013). Reading through discussion threads on this platform allowed aspirants to tap into conversations between professional programmers. In most cases, aspirants did not need to post a question themselves, but could access the expertise they needed by searching through existing question and answer threads. Aspirants also watched video tutorials produced by other developers when they needed to quickly learn how to accomplish a task. In the course of breaking into software development, aspirants learned how to search through and find needed information among these vast learning resources of the occupational community made freely available to interested outsiders like themselves.

For example, I observed Adam build an application that shows a user at any point in time what restaurants are open for the next two hours within a certain radius. For this simple coding project, Adam needed to learn how to implement Google Maps API (application programming interface), which would integrate his application with Google Maps. He first watched a video tutorial created by a developer advocate from Google to understand the basics of using the API. As he was working, he started searching Stack Overflow and other sources to find information on how to implement various features on the map such as dropping a pin on the location of the user. By the time he completed his Google Maps integration, Adam had accessed the expertise of multiple dispersed professionals and learned from them virtually and asynchronously in order to accomplish his task at hand.

Another aspirant described building projects as a process of "helping yourself" as opposed to getting help from near-peers at bootcamp. She said, "You basically don't get too much technical help at [the project] stage. You basically help yourself and, hopefully, you have learned to help yourself by that point" (Sam, Aspirant, DevHouse, Interview). Another aspirant said, "You're learning on your own and you are just looking up documentation a lot. You're looking up Stack Overflow and researching yourself". (Oliver, Aspirant, DevHouse, Interview)

All of my informants interpreted their bootcamp experience as a process of "learning how to learn new technologies". Aspirants considered this a key skill they developed at bootcamp. As Phil noted, "I think the biggest part [of bootcamp] was learning what to look for. Learning how to ask and how to get an answer" (Phil, Aspirant, CodeCamp, Field notes). Another aspirant explained how near-peers also guided him towards developing self-learning skills. He said,

> The process of learning [at bootcamp] was different. Problem solving, debugging, research skills... Whenever I'd ask a question, instead of just answering, [TAs] would be like, 'Let's say I wasn't here. How would you figure this out?' You start to figure out, I'm going to look at my code. I'm going to use the debugger. I'm going to go online. I'm going to go through all the steps in this process... You learn a lot more when you have to struggle through the process to figure it out. More than you would if someone was like, that's wrong because of this. (Clark, Aspirant, DevHouse, Interview)

In the course of self-learning from online resources made available by the broader occupational community, aspirants cultivated a reliance on themselves as self-learners, and developed a relationship with the dispersed, virtual occupational community of software developers as the source of the expertise they sought. Matt explained,

> The biggest skill I learned [at bootcamp] is how to learn other coding languages. I've gotten really good at Googling problems. When you run into a problem, 99 times out of a 100 someone else has run into the same problem. So when you Google your error message, you find a Stack Overflow entry or a tutorial... It's amazing how many resources are available for free in tech. I am way less scared about striking out on my own [after bootcamp]. (Matt, Aspirant, CodeCamp, Fieldnotes)

While striking out on one's own after bootcamp implied self-reliance, this was made possible by cultivating a reliance on the broader occupational community for expertise, when one needed it. Aspirants recognized this group of developers with whom they interacted virtually, and referred to 'the developer community' in their conversations:

> It is an incredible community. There are about 25 million people in the world who consider themselves to be in software development one way or the other. And these people learned through each other. For example, Stack Overflow–it's all people helping each other figure things out. (Jim, Aspirant, CodeCamp, Interview)

Aspirants talked both about a developer community at large, as well as the smaller user communities formed around particular technologies. One near-peer teaching assistant at DevHouse summarized the process of self-learning I observed at bootcamps, specifically referring to the creators of the knowledge artifacts that aspirants needed to consult in order to continue their self-learning journey. To a group of newcomers he said,

> Google and Stack Overflow are your friend. Learn your way around documentation and communities. These are way better than asking one person-- It's a collection of people's knowledge… If you think you need a tool, someone probably made it already, so first search online. You won't need to handroll it [i.e. build it yourself] most of the time. (DevHouse, Fieldnotes)

The communities that he was referring to were the smaller communicates of practice formed around particular technologies such as Ruby, Python or JavaScript that aspirants at bootcamps were trying to learn. These virtual groups of practitioners were the distant experts whom aspirants at bootcamp were asynchronously learning from when they read documentation, or searched through a Stack Overflow thread relating to that particular technology. While aspirants were self-learning and cultivating self-reliance, they were relying on the knowledge sharing practices of the occupational community they were trying to become a part of. Skill development within learning collectives was thus made possible by the organization of expertise in the target occupational community. Without the democratic knowledge sharing practices of the software development community, this mode of occupational learning would not be possible.

Table-4 summarizes the three scaffoldings bootcamps put in place, and how each scaffolding contributed to occupational learning, and to the learning collective as a whole.

----Insert Table-4 Here----

Progression from Novice to Hirable Software Developer

The three scaffoldings facilitated occupational learning such that novices who were formerly outsiders to the occupational community could develop both occupational skills and new self-conceptions as software developers. In this section, I show how aspirants' sense of self as builders and developers evolved hand in hand with their skill development. I also present job placement outcomes I calculated for the two cohorts of aspirants at my field sites.

Aspirants experience change in self-conceptions. Over the course of several months, I observed aspirants enter bootcamp as laymen who were outsiders to the occupational community, with little familiarity with coding, and towards the end of the bootcamp experience slowly begin to refer to themselves as 'builders' and 'developers' as they learned how to relate to software and build increasingly complicated projects. Aspirants' sense of progression from outsider to hirable software developer—albeit junior ones—depended on their ability to exercise skill, which was marked by success moments in the course of everyday work.

Software development practice lent itself to success moments that are sprinkled throughout the day when code works. Such success moments brought on a feeling of elation that was sometimes expressed out loud by aspirants and shared with their peers. The rest of the time was spent trying to figure out how to get code to work. My informants referred to this latter cognitive and emotive state as "being stuck". All day long, aspirants cycled between being stuck, and experiencing success moments when code worked. This was the rhythm of practice among aspirants learning software development. The below excerpt shows one aspirant, Bill, sharing the pride and elation he felt when his code finally worked after he was stuck for a long time:

> Bill lifts his fist in the air and says, “I did it! I can go home now!” He holds up his computer, showing his friends the rendering of a Google maps on his screen.
> Aaron: Congrats man. What was the problem?
> Bill: I wasn’t giving it a height, because I wasn’t styling it or anything. So I gave it a height and it worked!
> Aaron: Oh yeah man, Google maps doesn’t work if you don’t give it a height and width.
> Bill [smiling, turns to me]: This is the reason to write code. This moment.
> Aaron: Yeah, that high is totally worth it.
> Bill: Today is the best day right now. I worked through all my errors too. So now it’s all good. Until the next thing.
> (DevHouse, Field notes)

By “the next thing,” Bill was referring to the next time he would be stuck, which he expected to be soon. The elation that he expressed (“Today is the best day right now”), or the “high” as Aaron put it, was a common sentiment among aspiring developers that accompanied moments of success when code worked. As can be seen in this exchange, success moments were easily demarcated in the process of learning to code. Aspiring developers began to think of themselves as builders and developers as they experienced success moments in their work, ranging in complexity from writing simple programs that produced desired outputs to building a clone of a popular web application from scratch, to building their own projects and deploying these applications on the Internet for others to see and interact with. Seeing the fruits of their labor on the computer screen served as an undeniable manifestation of technical accomplishment. Over time, as aspiring developers experienced increasingly significant success moments, they began to think of themselves as ‘real developers’ and not just amateurs or outsiders.

For example, I was observing two aspirants from DevHouse pair programming one day when they achieved a new success moment that made them feel like software developers for the first time. Roy and Alice were in the early weeks of bootcamp and were slowly progressing from working on small coding challenges to building simple web applications. On this day, they were trying to build a to-do list application and I observed them struggle to figure out how to connect the backend (or server side) of the application that held the database, with the frontend, which is the part that the user sees and interacts with in the browser. After multiple trial and error attempts, they finally managed to get their code to work. Alice shouted in joy:

> Alice: We connected our frontend and backend!
> Roy: Which had never happened before!
> Alice: I'm pretty sure we *just* became software developers. Like this moment was it! (DevHouse, Field Notes)

In this instance, and in many other such success moments, I observed aspirants express a change in their self-conceptions. Different success moments could produce the feeling of no longer being a layman, and of stepping into a new occupational domain. Jim described the first time he started feeling like a developer:

> I started feeling like a developer when we started dealing with databases and routing, and learning about concepts like RESTful routing as a best practice and the whole HTTP handshake and how to chain together web pages and then start to do that programmatically so you can write little programs that will generate your whole website for you and build out all these different pages. That's when it really clicked and I thought, 'Oh wow, *I can do this!*' All those little pieces that I've been learning now fit together into something that I can say, '*Look, I built this. This is something I can do now!*' (Jim, Aspirant, CodeCamp, Interview, Emphasis added)

Being able to "build" things, to create a web application, validated Jim's sense of self as a developer. In William's case, being able to quickly learn and work with new technologies reinforced his sense of progression from layman to software developer:

> [Before bootcamp] I thought the technology was unattainable by someone like me. I did not study Computer Science. I did not grow up with computers at hand. [At bootcamp] you are shown a new technology every two days. You have to get up-to-speed quickly, within several hours, and then you have to learn how to navigate that technology to build stuff in it. I think that process of learning something quickly... not mastering it, but coming up to a real competency… Going through that a lot of times builds this mental muscle of, "I can build whatever I want." (William, Aspirant, Digital Academy, Interview)

Here William is describing his journey from being an outsider for whom the expertise was "unattainable," to someone who feels he can learn and implement any web related technology in a short span of time. In this way, aspirants' sense of self (*'who I am'*) developed hand in hand with their skills (*'what I can do'*) (Hughes 1958; Nelson and Irwin 2014). The projects that aspiring developers completed at bootcamp were particularly conducive to feeling like a 'builder' and 'developer' because aspirants built

these applications from scratch. Whereas in a professional setting a developer might work on a large codebase, trying to maintain an existing software system and making only small modifications to it, the bottom-up project work that bootcamps asked from their students allowed aspirants to execute one or more web application ideas of their own. Building an application from the ground up allowed aspirants to feel like developers. Their projects were visible, tangible, interactive web applications that came alive on their computer screens. Sam, who loved playing video games, felt this pride when he was able to build a simple JavaScript game for the first time:

> Aside from the learning, I guess that the biggest thing [about the bootcamp experience] was being like 'Wow I can't believe I actually can do these things!' With coding, there are these small pieces that you learn how to do, and then there is also the larger idea that I can take these small pieces and build them into something big and amazing. I'm a PC gamer. Now I could actually build – not those huge millions of dollar budget games but the smaller games, indie games – I could actually code those now! (Sam, Aspirant, DevHouse, Interview)

In these success moments, aspiring developers felt they had a grasp on the expert practice they were trying to break into. Having technologies respond to them in the way they intended was a major accomplishment in their eyes *("I could actually code those now!"*, or *"I could build whatever I want!"*). It was as a result of the accumulation of these success moments that aspirants felt comfortable calling themselves software developers as they entered the labor market in search of jobs where they could continue learning while working alongside professional programmers.

Job Search Outcomes of Aspirants. Despite the short duration and unconventional nature of the occupational learning pathway that they pursued, my data show that a large portion of aspirants attending CodeCamp and DevHouse during the time of fieldwork found jobs in entry level software developer roles or in hybrid roles requiring coding skills. I gathered job placement data for two cohorts of students who graduated during February and March of 2017 from my two field sites. I excluded 13 aspirants who were not complete 'outsiders' to the target occupational community, either because they had a CS degree or because they had worked in IT related employment prior to bootcamp. After excluding these learners, I

calculated job placement outcomes for the remaining 70 aspirants. These outcomes, which are broken down by bootcamp, duration of job search, and type of role, are summarized in **Table-5**.

----Insert Table-5 Here----

Job placement outcomes differed by bootcamp. DevHouse was significantly more successful than CodeCamp. Six months after graduation 71% of DevHouse graduates and 46% of CodeCamp graduates had started programming related employment. Whereas DevHouse graduates were placed predominantly in software developer roles (69%), six months after graduation only 27% of CodeCamp graduates were working as software developers. Another 18% of CodeCamp graduates had started jobs in what my informants called "hybrid roles" that combined coding knowledge with other skillsets, such as Technical Support Engineer or Solutions Engineer. Hybrid roles were less desirable than Software Developer roles because they required less programming expertise and commanded lower salaries. Yet my informants still considered themselves successful when they accepted a hybrid role, because they could not have attained these jobs without knowing how to code, and because these jobs also came with significantly higher salary and benefits than their jobs before bootcamp. I therefor accept the emic definition of success, and consider transitions into hybrid roles also as a successful learning outcome.[4] Nine months after graduation, 83% of DevHouse graduates were working as software developers and 2% were working in hybrid roles. Among CodeCamp graduates, 32% had become software developers and 27% had started work in hybrid roles. For another 23% of CodeCamp graduates, their transition into software development took more than a year.

One way to interpret these results is to consider the fact that DevHouse was widely believed to be one of the most selective bootcamps in the United States, enrolling candidates that it believed had the highest aptitude for learning programming, whereas CodeCamp was considered almost "open access" (Staff

[4] Of the 11 aspirants in total who accepted hybrid roles after bootcamp, I observed three move into software developer roles over time. Four were promoted within their hybrid roles. One remained in a hybrid role without promotion. Two moved into Technical Program Management roles, and one dropped out of coding.

member, CodeCamp, Fieldnotes). Therefore, it could be argued that these job placement outcomes reflect two ends of the spectrum in terms of what learners can hope to achieve by attending a coding bootcamp for the purposes of changing careers. Another factor to consider when interpreting these results is the vastly different attitudes that the two bootcamps displayed once aspirants finished the program. DevHouse's careers team encouraged students to obtain employment quickly after graduation by giving them deadlines and weekly targets for how many jobs they should apply to. CodeCamp applied no such pressure to its graduates. Career services were offered on demand and I observed many aspirants taking breaks after bootcamp, and taking time to improve their coding skills for several months before applying to jobs.

DISCUSSION

To sum up, aspirants trying to break into software development expertise by attending coding bootcamps learned the basics of programming and how to continue learning on their own while working amongst a community of peers and near-peers. The bootcamp curriculum that guided their learning was intense but short, and limited in focus. Aspirants studied the teaching materials on their own, were provided with brief lectures during their first two months of learning, and spent the majority of their time at bootcamp working on coding assignments with their peers, with the help of instructors and teaching assistants, many of whom were their near-peers. The three scaffoldings that facilitated this learning process were 1) peer team structures in the form of pair-programming and group projects, 2) near-peer role structures that engaged more advanced learners in teaching and mentorship relationships with more junior learners, and 3) encouragement to self-learn, which undergirded the entire occupational learning process, and prepared aspirants for learning beyond the bootcamp curriculum.

The concept of a *learning collective* is proposed in this paper to describe the collaborative, peer and near-peer focused learning environment at bootcamps that helped aspirants develop occupational skills. As a prerequisite to collaboration, aspirants shared an understanding that they were working towards a common, valued goal: breaking into a skilled occupation in a relatively short amount of time. Collaborative

learning as a practice was established in the early stages of learning at bootcamps through peer team structures, and continued into aspirants' job search process as they collaborated in preparing for behavioral and technical interviews.

I found that being able to collaborate in the act of learning, exchanging information with peers, and receiving help from near-peers allowed aspirants to more easily overcome the difficulty associated with trying to break into a foreign and technically complex domain of expertise. Compared to learning on one's own, which is an established, albeit opaque mode of entry into the occupation of software development (Ensmenger 2010), this collaborative mode of learning enabled aspirants to access knowledge quickly and easily from other aspirants around them. Instead of 'being stuck' for long periods of time when they didn't understand a concept, or the reason for an error in their code, aspirants could turn to a peer sitting next to them, or a near-peer teacher who could help them get 'unstuck.'

Besides the motivational benefits of not being stuck for long, of accessing help quickly and easily, being part of a learning collective gave aspirants opportunities to teach their peers what they knew, which improved comprehension. Both pair programming and group projects made coding a conversational, collaborative activity. Aspirants stepped into teaching roles towards their peers situationally, and in the course of teaching they retrieved information, made connections between different pieces of information, and addressed gaps in their knowledge that became apparent in the course of knowledge exchange. The potential benefits of peer to peer and near-peer teaching are widely recognized in the educational psychology literature (Whitman 1988; Topping 1996; Fiorella and Meyer, 2014; Rashid, Sobowale and Gore 2011). For the peer or near-peer acting as teacher, there is considerable evidence suggesting that teaching will help consolidate and improve their knowledge (Roscoe and Chi 2007&2008). The potential benefits for the learner in these exchanges are that they are learning from someone who can potentially better tailor their explanations to the learners' needs as a result of being in the same position, or having been in the same position only recently (Lockspeiser, O'Sullivan, Teherani, Muller 2008). Bootcamps as learning collectives made extensive use of such peer and near-peer teaching dynamics.

Finally, while learning at bootcamps happened within a community of peers and near-peers, it was undergirded by self-learning. Aspirants learned how to learn programming themselves by navigating the vast resources made available online by the occupational community. So even though aspirants did not always have access to proximate experts they could consult, they were able to consult the expertise of the broader, virtually organized occupational community and learn from them. Thus, the organization of expertise and the norms of knowledge sharing within the target occupational community made the learning collective mode of occupational entry possible.

Implications for Occupational Learning

The main theoretical contribution that this study makes to our understanding of occupational learning is to show one process by which outsiders can obtain occupational expertise: by participating in learning collectives. Prior conceptualizations of how novices acquire occupational expertise predominantly describe cases where novices pursue the established learning pathways of an occupation, and by virtue of doing so, are granted insider status from the start of their learning journey (e.g. Becker et al. 1961; Van Maaneen, 1973; Pratt et al. 2006). Novice insiders' learning is then sponsored by incumbents who make learning opportunities available to them (Schleef 2006; Ranganathan 2017). Learning is conceived as happening predominantly between incumbents and the novices whose occupational entry they are assisting. Even though peers and near-peers are mentioned in the description of occupational communities (Lave and Wenger, 1991), their role in teaching novices does not receive sufficient attention.

Aspirants at bootcamps, in contrast, started their learning journey as outsiders pursuing a contested learning pathway and desiring to move through the inclusion boundaries of their target occupation. They did not enjoy the same level of access to experts and expert practice that novices who are insiders enjoy. As a result, their learning process differed from prior conceptualizations of occupational learning identified in the literature. **Table-6** offers a comparison between skill development within learning collectives, and other conceptualizations of occupational learning found in the literature: occupational socialization, situated learning and shadow learning.

----Insert Table 6 Here----

Whereas novices who are considered insiders to an occupational community learn from proximate incumbents who sponsor their learning, aspirants at bootcamps were primarily surrounded by other aspirants—their peers and near-peers—who were also trying to break into software development. When aspirants had questions, help often came from these peers and near-peers. The lack of access to proximate experts was not a function of the status or quality of the bootcamp either. DevHouse, which was a more successful bootcamp, had fewer expert teachers than CodeCamp. Given minimal access to proximate experts, aspirants at bootcamps acquired occupational expertise through engaging in knowledge exchange with their peers and near peers, and through self-learning from the knowledge resources made available by the occupational community.

In terms of access to legitimate peripheral participation opportunities, which are an indispensable part of entry into skilled occupations (e.g. Brown and Duguid 1991), aspirants pursuing the bootcamp path of learning could not apply to new graduate positions or software engineering internships, because these roles were reserved for novices following the established learning pathways of the occupation. At the time of research, bootcamps were still a new industry and not considered a legitimate credential by most employers. As a result, workplaces did not commonly sponsor legitimate peripheral participation opportunities for aspirants pursuing the bootcamp path of learning. In order to obtain access to professional practice and the opportunity to learn from working with practitioners, bootcamp graduates needed to convince employers that they were professional developers in the first place. In other words, their initial admittance into professional practice had to be in the role of a 'full participant' (Lave and Wenger, 1991), which aspirants were granted only if they could demonstrate skill.

The paradox of having to demonstrate skill in order to obtain opportunities for skill development, which is common in interorganizational careers (O'Mahony and Bechky 2006), plagued aspiring developers attending bootcamps. However, the paradox was more pronounced for aspirants attending bootcamps

compared to the contractors that O'Mahony and Bechky (2006) studied. The contractors in their study engaged in stretchwork, in other words "work that largely fits with an individual's previous work experience but introduces a small novel element" (O'Mahony and Bechky, 2006: 919), in order to extend their skills in a new direction. An example would be a Java developer learning C++ or, in a more extreme case, a front-end developer learning back-end development. These contractors were staying within the boundaries of their occupation when seeking stretchwork. Aspirants at bootcamps, on the other hand, were coming from diverse educational and professional backgrounds and trying to move from one domain of knowledge and associated practice into an unrelated domain. Without any formal education or professional experience to attest to their expertise, aspirants used only their projects to showcase their skills, and tried to obtain opportunities for further skill development in entry level roles.

These findings can generalize to a variety of other contexts. New organizing practices coupled with new technologies are not only changing the nature of work (Barley, Bechky, and Miliken 2017), they are also changing how people learn new skills and take on new occupational roles as they navigate their careers (Barley and Kunda 2004; Smith 2010; Halpin and Smith 2017). For contemporary workers, it is becoming increasingly common to cross not just organizational, but also occupational boundaries in the course of one's career (Tolbert 1996). Many career switchers who are unable to go back to school or take on internships and apprenticeships may find themselves in the position of novice outsiders trying to break into an occupation. For people in this situation, learning collectives could facilitate entry into many skilled occupations where credentials are not a requirement to practice, and where the target expertise can be learned through self-study, in a DIY manner. These occupations have lower levels of social closure (Van Maanen and Barley 1984; Weeden 2002). Many skilled occupations today potentially could be entered via DIY learning. There are no formal educational requirements, other than having a four-year college degree in most cases, to entering UX research, UX design, data science, marketing, or some roles within finance, for example. If sufficient learning resources are shared by the target occupational community with interested outsiders, then learning collectives could be formed to utilize these resources and facilitate occupational learning.

The knowledge sharing practices of the target occupational community would therefore be an important factor affecting the success of occupational entry via learning collectives. While Internet native occupations such as programming appear to be at the forefront of peer-to-peer sharing of occupational expertise online (Wasko and Faraj 2000; Wasko and Teigland 2004; Vasilescu, Filkov, and Serebrenik 2013; Schwartz 2018), many other occupations are migrating their expertise sharing practices to the Internet, in effect making them accessible to lay audiences and opening up pathways for DIY entry into occupations (Susskind and Susskind, 2015). In general, boundaries around professional expertise are becoming more permeable as "the new knowledge economy enables new ways of acquiring and sharing expertise that do not conform to professionalization theory" (Croidieu and Kim 2017: 3). Learning collectives can be thought of as a more structured and community supported version of DIY learning processes, as aspirants follow a shared curriculum and benefit from collaborative learning dynamics—facilitated by peer team structures and near-peer role structures—to support and accelerate self-learning. Thus, any occupation that can be entered through DIY learning processes is an occupation that can be entered, potentially more easily, through leveraging learning collectives.

Boundary Conditions and Directions for Future Research

Several characteristics of software development practice and the occupational community surrounding it enabled the breaking in process that I observed at bootcamps. First of all, software development, despite being a skilled occupation, has many self-taught members (Ensmenger, 2010). Breaking into software development is much more acceptable than breaking into law, medicine or civil engineering where the lack of a degree would disqualify an aspirant from consideration completely (Abbott 1988). Therefore, these findings could not generalize to job areas where credentials or licensing are a prerequisite for practice.

Another factor that made software development expertise accessible to novice outsiders was the occupational community's culture of knowledge sharing. Software developers, more so than many other occupational groups, share an ethos of open access to knowledge (Levy 1984; Rheingold 2000; Lakhani

and von Hippel 2003; O'Mahony and Ferraro 2007). This was reflected in the abundance of knowledge artefacts such as tutorials, public code repositories, and knowledge sharing platforms like Stack OverFlow (Lakhani & von Hippel, 2003; Wang, Lo, and Jiang, 2013) that aspiring developers were able to access as amateurs and outsiders. Without such abundant resources that are designed for the use of not just seasoned practitioners but also beginners, it would be difficult for an occupational community to be amenable to DIY entry, or entry by leveraging learning collectives.

Finally, the immateriality of software was another factor enabling the learning processes that I observed. Software as an object of practice could be inexpensively set up outside of a professional context. All that aspiring developers needed to practice coding and build projects was a personal computer and software programs to set up a development environment, and these they could download for free. Moreover, the immateriality of software meant that errors were relatively costless, and novices could engage in countless cycles of trial and error in their learning process. My findings may not generalize to other occupations where either the tools necessary for professional practice are expensive to setup, or errors are costly.

Further studies are needed to understand the wider range of alternative learning processes that are occurring today within and at the borders of occupational communities, and what this means for expertise more broadly. Importantly, in the case of skilled occupations, further research is needed to investigate whether aspirants who break into a skilled occupation without following established learning pathways suffer any disadvantages later in their careers, or if they are able to overcome the shortcomings of their initial learning process and build on their expertise over time, either through self-study or through gaining experience on the job. If the latter were not possible, then it can be argued that the process of breaking into an occupation can lead to a subcategory of occupational members who are stuck in the lower tiers of their occupation, unable to progress in their careers. In the case of aspiring software developers attending coding bootcamps, the answer can be found by following aspirants five or ten years into their careers.

Lastly, this study points to some implications for employers and practitioners. If employers are interested in developing alternative talent pipelines into programming and related jobs (e.g. Colby, Ma,

Robinson, and Yee2016) then there appears to be a need for designing situated learning opportunities for this purpose. At the time this fieldwork took place, there were a very limited number of apprenticeships reserved for bootcamp graduates at technology companies. These apprenticeships were designed to accommodate what employers called 'nontraditional job candidates' to diversify their talent pipelines. Making such situated learning opportunities involving more mentorship a part of industry practice would be helpful in integrating aspirants coming from nontraditional backgrounds. Finally, employer tailored bootcamp programs can be another path forward, where employers communicate their hiring needs to bootcamps that in turn design tailored curricula to develop needed skillsets.

Bibliography

Abbott, A. 1988. *The system of professions: An essay on the division of expert labor*. Chicago: University of Chicago Press.

Abbott, A. 1991. The Future of Professions: Occupation and Expertise in the Age of Organization. In P. S. Tolbert and S. R. Barley (Eds.), *Research in the Sociology of Organizations,* 8:17-42. JAI Press Inc.

Anteby, M. 2013. *Manufacturing Morals: The Values of Silence in Business School Education*. Chicago, IL: University of Chicago Press

Anteby, M., Chan, C. K., and DiBenigno, J. 2016. Three Lenses on Occupations and Professions in Organizations: Becoming, Doing, and Relating. *The Academy of Management Annals*, 10,(): 183-244.

Arbeit, C. A., Bentz, A., Cataldi, E. F., and Sanders, H. 2019. *Alternative and Independent: The Universe of Technology-Related Bootcamps*. RTI Press. https://doi.org/10.3768/rtipress.2018.rr.0033.1902

Arthur, M. B. 2008. Examining contemporary careers: A call for interdisciplinary inquiry. *Human Relations*, 61(2): 163-186

Arthur, M. B., and Rousseau, D. M. 1996. *The boundaryless career: A new employment principle for a new organizational era*. New York: Oxford University Press.

Ashforth, B. E. 2001. *Role transitions in organizational life: An identity-based perspective*. Mahwah, N.J: Lawrence Erlbaum Associates.

Bailey, D. E. and Barley, S. R. 2011. Teaching-learning ecologies: Mapping the environment to structure through action. *Organization Science*, 22: 262-285.

Barley, S.R., Bechky, B. A., and Miliken, F. J. 2017. The Changing Nature of Work: Careers, Identities and Work Lives in the 21st Century. *Academy of Management Discoveries*, 3(2): 111-115

Barley, S. R. and Kunda, G. 2001. Bringing work back in. *Organization Science*, 12(1): 76–95

Barley, S. R. and Kunda, G. 2004. *Gurus, Hired Guns, and Warm Bodies: Itinerant Experts in a Knowledge Economy.* Princeton: Princeton University Press.

Baruch, Y. and Vardi, Y. 2016. A Fresh Look at the Dark Side of Contemporary Careers: Toward a Realistic Discourse. *British Journal of Management*, 27: 355–372

Bharatan, I., Swan, J., and Oborn, E. 2022. Navigating Turbulent waters: Crafting learning trajectories in a changing work context. Human Relations. 75(6): 1084-1112

Beane, M. 2018. Shadow Learning: Building Robotic Surgical Skill When Approved Means Fail. *Administrative Science Quarterly*. 64(1): 87-123

Bechky, B.A. and O'Mahony, S., 2015. Leveraging comparative field data for theory generation. In K.D. Elsbach & R.M. Kramer (Eds), *Handbook of qualitative organizational research: Innovative pathways and methods*, pp.168-176. New York: Routledge

Becker, H. S., and Carper, J. 1956. The Elements of Identification with an Occupation. *American Sociological Review*, 21(3): 341-348.

Becker, H. S., Geer, B., Hughes, E. C., and Strauss, A. L. 1961. *Boys in White: Student Culture in Medical School*. Piscataway, NJ: Transaction.

Bidwell, M.J., and Briscoe. F. 2010. The Dynamics of Interorganizational Careers. *Organization Science*. 21(5): 1034-1053

Bosu, A., Corley, C. S., Heaton, D., Chatterji, D., Carver, J. C., and Kraft, N. A. 2013. Building reputation in StackOverflow: An empirical investigation. IEEE Working Conference on Mining Software Repositories (MSR 2013). 89-92.

Brown, J. S., and Duguid, P. 1991. Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation. *Organization Science*, 2: 40–57

Bulte, C., Betts, A., Garner, K., Durning, S. 2007. Student teaching: Views of student near-peer teachers and learners. Medical Teacher, 29(6):583–590.

Cappelli, P. 1999. *The new deal at work: Managing the market-driven workforce.* Boston, Mass: Harvard Business School.

Colby, S., Ma, H., Robinson, K., and Yee, L. 2016. What it will take to make the tech industry more diverse. Harvard Business Review. 03/15/2016

Contu, A. and Willmott, H. 2003. Re-embedding Situatedness: The Importance of Power Relations in Learning Theory. Organizations Science. 14(3): 283-296

Croidieu G., and Kim, P. H. 2017. Labor of Love: Amateurs and Lay-expertise Legitimation in the Early U.S. Radio Field. *Administrative Science Quarterly*. 63(1): 1-42.

Demetry, D. 2017. Pop-Up to Professional: Emerging Entrepreneurial Identity and Evolving Vocabularies of Motive. *Academy of Management Discoveries*, 3(2): 187-207.

Ensmenger, N. 2010. *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*. Cambridge, MA: MIT Press

Fine, G. A. 1985. Occupational aesthetics: How trade school students learn to cook. *Urban Life*. 14: 3-32

Fine, G. A. 1996. Justifying work: Occupational rhetorics as resources in restaurant kitchens. *Administrative Science Quarterly*. 41: 90-115

Foirella, L. and Mayer, R. E. 2014. Role of expectations and explanations in learning by teaching. *Contemporary Educational Psychology*. 39(2): 75-85

Freidson, E. 1970. *Profession of Medicine: A Study of the Sociology of Applied Knowledge*. New York: Dodd, Mead.

Freidson, E. 1986. *Professional powers: A study of the institutionalization of formal knowledge*. Chicago, Illinois: Univ. of Chicago Press.

Glaser, B. G., and Strauss, A. L. 1967. *The discovery of grounded theory: Strategies for qualitative research*. New York, NY: Aldine de Gruyter.

Hall, D. T. 2004. Protean Career: A quarter-century journey. *Journal of Vocational Behavior*. 65: 1-13

Halpin, B. W. and Smith, V. 2017. Employment Management Work: A Case Study of Theoretical Framework. *Work and Occupations*. 44(4): 339-375

Hughes, E. C. 1958. *Men and their work*. London: Forgotten Books.

Hughes, E. C. 1971. *The sociological eye: Selected papers.* Chicago: Aldine-Atherton

Ibarra, H. 1999. Provisional selves: Experimenting with image and identity in professional adaptation. *Administrative Science Quarterly*, 44: 764–791.

Ibarra, H. 2003. *Working Identity: Unconventional Strategies for Reinventing Your Career*. Cambridge, MA: Harvard Business School Press.

Ibarra, H. and Obodaru, O. 2016. Betwixt and between identities: Liminal experience in contemporary careers. *Research in Organizational Behavior*. 36: 47–64

Johns, T., and Gratton, L. 2013. The third wave of virtual work. *Harvard Business Review*, 91(1): 66.

Jordan, B. 1989. Cosmopolitical obstetrics: Some insights from the training of traditional midwives. *Social Science and Medicine*, 28: 925–937

Kalleberg, A. L. 2000. Nonstandard employment relations: Part-time, temporary and contract work. *Annual*

Review of Sociology, 26: 341-365

Kalleberg, A. L. 2009. Precarious work, insecure workers: Employment relations in transition. American Sociological Review, 74(1): 1–22

Kellogg, K. C., Valentine, M. A., & Christin, A. 2020. Algorithms at work: The new contested terrain of control. *Academy of Management Annals*, 14(1): 366-410

Lakhani, K. R., and von Hippel, E. 2003. How Open Source Software Works: "Free" User-to-User Assistance. *Research Policy,* 32(6): 923–943

Lave, J., and Wenger, E. 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge: Cambridge University Press.

Levy, S. 1984. *Hackers: Heroes of the Computer Revolution*. New York: Dell Publications

Lockspeiser, T., O'Sullivan, P., Teherani, A., Muller, J. 2008. Understanding the experience of being taught by peers: the value of social and cognitive congruence. *Advances in Health Science Education*, 13:361-372.

Marshall, H. M. 1972. Structural constraints on learning: Butchers' apprentices. *American Behavioral Scientist*, 16: 35–44.

Michel, A. 2011. Transcending Socialization: A Nine-Year Ethnography of the Body's Role in Organizational Control and Knowledge Workers' Transformation. *Administrative Science Quarterly*, 56 (3): 325-368

Moore, W. E. 1976. *The Professions: Roles and Rules*. New York: Russell Sage Foundation

Morrison, E. W. 1993. Newcomer Information Seeking: Exploring types, modes, sources and outcomes. *Academy of Management Journal*, 36(3): 557-589.

Mulas, Victor; Paradi-Guilford, Cecilia Maria; Allende Letona, Elene; Dalphond, Zhenia Viatchaninova. 2017. *Coding bootcamps: building future-proof skills through rapid skills training*. World Bank Group. Washington, D.C.

Nelson, A. J. and Irwin, J. 2013. Defining what we do all over again: Occupational identity, technological change and the librarian/Internet-search relationship. *Academy of Management Journal*.

O'Mahony, S., and Bechky, B. 2006. Stretchwork: Managing the career progression paradox in external labor markets. *Academy of Management Journal*, 49: 918–941

Orr, J. E. 1996. *Talking about Machines: An Ethnography of a Modern Job.* Ithaca: Cornell University Press

Petriglieri, G., Petriglieri, J.L., and Wood, J.D. 2017. Fast tracks and inner journeys: Crafting portable selves for contemporary careers. *Administrative Science Quarterly*. 63(3): 479-525.

Pratt, M. G., Rockmann, K. W., and Kaufmann, J. B. 2006. Constructing professional identity: The role of work and identity learning cycles in the customization of identity among medical residents. *Academy of Management Journal*, 49: 235–262.

Rahman, H. A. 2021. The invisible cage: Workers' reactivity to opaque algorithmic evaluations. Administrative Science Quarterly, 66(4): 945-988.

Ranganathan, A. 2017. Train Them to Retain Them: Work Readiness and the Retention of First-time Women Workers in India. *Administrative Science Quarterly*, 63(4): 879-909.

Rheingold, H. 2000. *The virtual community: Home steading on the electronic frontier*. MIT Press

Riemer, J. W. 1977. Becoming a Journeyman Electrician: Some Implicit Indicators in the Apprenticeship Process. Sociology of Work and Occupations, 4(1): 87-98.

Roscoe, R. D., and Chi, M. T. H. 2007. Understanding tutor learning: Knowledge building and knowledge-

telling in peer tutors' explanations and questions. *Review of Educational Research*, 77(4), 534–574.

Roscoe, R. D., and Chi, M. T. H. 2008. Tutoring learning: The role of explaining and responding to questions. *Instructional Science*, 36, 321–350.

Rosenblat, A. 2018. *Uberland: How Algorithms Are Rewriting the Rules of Work*. University of California Press

Saks, A. M., and Ashforth, B. E. 1997. Organizational Socialization: Making Sense of the Past and Present as a Prologue for the Future. *Journal of Vocational Behavior*, 51(2): 234-279.

Saks, A. M., and Gruman, J. A. 2012. Getting newcomers on board: A review of socialization practices and introduction to socialization resources theory. In S. Schmidt (Ed.), *The Oxford Handbook of Organizational Socialization*: 27-55. New York : Oxford University Press

Schleef, D. J. 2006. *Managing elites: Professional socialization in law and business schools*. Lanham: Rowman and Littlefield.

Schwartz, D. 2018. Embedded in the Crowd: Creative Freelancers, Crowdsourced Work, and Occupational Community. *Work and Occupations*. 45(3): 247-282.

Scott, W. R., Holzman, B., Ris, E. and Biag, M. 2017. The Changing Ecology of Higher Education in the San Francisco Bay Area. In W. R. Scott, and M. W. Kirst (Eds.), *Higher Education and Silicon Valley: Connected but Conflicted*. Baltimore: Johns Hopkins University Press

Smith, V. 2010. Enhancing employability: Human, cultural, and social capital in an era of turbulent unpredictability. *Human Relations*, 63(2), 279–300

Spradley, J. P. 1979. *The Ethnographic Interview*. New York: Holt, Rinehart and Winston

Strauss, A., and Corbin, J. 1990. *Basics of qualitative research: Grounded theory procedures and techniques*. Newbury Park, CA: Sage Publications

Susskind, R., and Susskind, D. 2015. *The future of the professions: How technology will transform the work of human experts*. Oxford, UK: Oxford University Press.

Tolbert, P. S. 1996. Occupations, organizations, and boundaryless careers. In M. B. Arthur and D. M. Rousseau (Eds.), *The boundaryless career*: 331-349. NewYork, NY: Oxford University Press

Topping, K. 1996. The effectiveness of peer tutoring in higher and further education: A typology and review of the literature. *Higher Education*. 32 (3): 321-345

Trice, H. M. 1993. *Occupational subcultures in the workplace*. Ithaca, NY: ILR Press.

Van Maanen, J. 1973. Observations on the making of policemen. *Human Organization*, 32(4): 407-418.

Van Maanen, J. 1978. The asshole. In: Manning, Peter K. and Van Maanen, John, eds. Policing: A View from the Street. Los Angeles: Goodyear Press

Van Maanen, J., and Barley, S. R. 1984. Occupational communities: Culture and control in organizations. *Research In Organizational Behavior*, 6: 287-365.

Van Maanen, J., and Schein, E. H. 1979. Toward a theory of organizational socialization. In B.M. Staw (Ed.), Research in organizational behavior (Vol. 1, pp. 209-264). Greenwich, CT: JAI Press.

Vasilescu, B., Filkov, V., and Serebrenik, A. 2013. StackOverflow and GitHub: Associations between software development and crowdsourced knowledge. Proceedings of the 2013 International Conference on Social Computing, pp. 188–195.

Waguespack, L., Babb, J. S., and Yates, D. J. 2018. Triangulating coding bootcamps in IS education: Bootleg education or disruptive innovation? Information Systems Education Journal, 16(6): 48-58

Wang, S., Lo, D., and Jiang, L. 2013. An Empirical study on developer interactions in StackOverflow. SAC '13 Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 1019-1024

Wasko, M., and Faraj, S. 2000. It Is What One Does: Why People Participate and Help Others in Electronic Communities of Practice. The Journal of Strategic Information Systems, 9, 155-173.

Wasko, M. M. and R. Teigland. 2004. Public Goods or Virtual Commons? Applying Theories of Public Goods, Social Dilemmas, and Collective Action to Electronic Networks of Practice. *The Journal of Information Technology Theory and Application*, 6(1): 25-41

Wenger, E. 1998. *Communities of practice: Learning, meaning, and identity*. Cambridge, U.K: Cambridge University Press.

Whitman, N. A. 1988. *Peer Teaching: To Teach Is to Learn Twice*. ASHE-ERIC Higher Education Reports. Washington.

Wood, D., Bruner, J. S., and Ross, G. 1976. The role of tutoring in problem solving. *Journal of Child Psychiatry and Psychology*. 17: 89-100.

Wilensky, H. L. 1964. The Professionalization of Everyone? *American Journal of Sociology*, 70(2): 137-158.

Table 1. Comparison of different conceptualizations of occupational learning

	Occupational Socialization	Situated Learning	Shadow Learning
Role of the novice in relation to the occupational community	Insider	Insider	Insider
Role of formal instruction	Formal instruction occurs during schooling	Importance of formal instruction is downplayed. Novices learn on the job, working alongside incumbents. For skilled occupations, novices often share formal educational background	
Access to legitimate peripheral participation opportunities	Full access	Full access	Limited access. Organization of work and associated technologies hinders effective legitimate peripheral participation
Locus of learning	In the classroom, and during engagement with professional practice in the workplace	Engagement with professional practice in the workplace	Successful learners opportunistically create their own learning opportunities across different professional and informal spaces
Outcome of learning	Occupational skill and identity development License to practice	Occupational skill and identity development License to practice	Only shadow learners develop necessary skills for robotic surgery Both successful and unsuccessful learners are licensed to practice

Table 2. Summary of data collection

Type	Description
Interviews	31 CodeCamp graduates 27 DevHouse graduates 11 graduates of five other coding bootcamps 11 bootcamp founders and directors
Ethnographic Observations	Observations at 20 learn-to-code meetups in the San Francisco Bay Area (Sept - Dec 2016) Observations at CodeCamp (Dec 2016 - May 2017) Observastions at DevHouse (July - Aug 2017)
Archival Data	Artifacts created by coding bootcamps and other learn-to-code organizations (informational brochures, syllabi, career guides etc.) Blog entries created by bootcamp students / graduates about their learning journeys

Table 3. Backgrounds of aspirants at DevHouse and CodeCamp

Aspirant backgrounds	
Career switcher	60 (72%)
New graduate (non-CS) career switcher	10 (12%)
New graduate (CS)	5 (6%)
Background in IT related work	8 (10%)
TOTAL	83 (100%)

Table 4. How bootcamps scaffolded occupational learning

Scaffolds	Enabled Practices	Examples	Contribution to Occupational Learning	Contribution to the Learning Collective
Peer Team Structures	Aspirants collaborate on learning Aspirants situationally take on teacher role towards each other	**Pair programming:** "Pair programming is amazing for learning things whether you are less knowledgable or more knowledgeable. If you know more, you get to explain to others. If you know less, you get to learn from the person who knows more." (Kim, Aspirant, CodeCamp, Fieldnotes) Brian and Jake are pairing for the day. They are talking amongst each other, trying to understand what a reducer does. There is some back and forth between them, then Brian says, "So the reducer decides what needs to be done and tells the store what to change." Jake nods and says, "Makes sense". Brian is the driver so he starts typing. Jake is following his code as he types. He says, "Don't forget we need to freeze state". Brian says, "Oh right. I guess this is how we do it" and goes on to type "Object.freeze(state)" (DevHouse, Fieldnotes) Tim points the curser to several lines of code and says, "I don't get what's going on here". Adam explains and says, "That's my understanding." (DevHouse, Fieldnotes) **Group projects:** "I didn't know Python or Raspberry Pi until this (group) project… I actually started late with Python because I was trying to get the Raspberry Pi working. So when I started learning Python, the other guys had already been grappling with it, so I asked them. They showed me what they learned." (Phil, Aspirant, CodeCamp, Interview) All group members are sitting in one room, trying to get the Ruby on Rails and React marriage working. Trevor is struggling with a problem and finds a possible solution on Stack Overflow. He announces to the group the error and the possible solution. One of his teammates says he's not sure. Another teammate, Gary, says "I'm working on the same problem, give me a couple of minutes." Trevor waits for Gary to share what he learned. (CodeCamp, Fieldnotes)	Aspirants access knowledge quickly and easily Time spent 'being stuck' is minimized and motivation is maintained Teaching improves comprehension for aspirant in teacher role	Asking for help and teaching each other are normalized Learning becomes a collaborative, community-based activity
Near-Peer Role Structures	Near-peers teach novices in formal instructional roles	"A lot of the teaching assistants had graduated from the previous cohort" (Scott, Aspirant, DevHouse, Interview) "I think the precedence of [near-peers] served as encouragement that in just a couple weeks, a given student could go from learning the material for the first time to being in the role of teacher to students that started after them." (Jim, Aspirant, CodeCamp, Interview) "[Before starting bootcamp] I was really wary of the whole idea that many people who teach are people who have gone through the program...But what was cool about it, was a lot of people who had stayed to TA really loved the teaching aspect of it...Maybe their first cohort, they're a little rusty, but (with every cohort), they are reiterating on it and learning more...The actual curriculum that you go through there is just sort of grilled into the TA's so much that they're masters of it." (Oliver, Aspirant, DevHouse, Interview)	Aspirants access knowledge quickly and easily Time spent 'being stuck' is minimized and motivation is maintained Learning from near-peers can reduce learning anxiety Teaching improves comprehension for near-peer	Learning collective becomes more cost efficient and scalable

Scaffolds	Enabled Practices	Examples	Contribution to Occupational Learning	Contribution to the Learning Collective
Encouragement to Self-Learn	Aspirants teach themselves using knowledge resources created by the occupational community	"The way I usually describe the curriculum we went through to people not in the know is usually in the lines of, 'We didn't learn how to program. We learned how to learn how to program.'... The curriculum was much more focused on the pair programming and tackling the coding challenges of the day with minimal time learning directly from an instructor." (Adam, Aspirant, CodeCamp, Interview) "Strive towards independence...You will have support during the first two months, but you'll get less and less support as time goes on. When you come across a bug, I know you might think I'll ask a TA, they'll see it faster than me. Don't do that. Read documentation. Use Google and Stack Overflow." (Staff member, DevHouse, Fieldnotes) "They want you to learn how to learn on your own because that's how the industry is." (CodeCamp, Fieldnotes) "If you have a problem, Google the issue first before asking a TA… You are in charge of yourself to a large extent. When you are on the job, in your new position, you'll be given a codebase, you'll be given a list of tickets to solve, and you're just going to have to figure it out. That's how the real world works. So flex those muscles now". (Staff member, DevHouse, Fieldnotes)	Aspirants continue to learn beyond the bootcamp curriculum Aspirants are socialized for an occupation with constant learning demands Aspirants develop a relationship to the broader occupational community as a source of expertise	Learning collective is reliant on the knowledge sharing practices of the target occupational community

Table 5. Job search outcomes broken down by time to job, role, and bootcamp

Duration of Job Search	New Role	DevHouse	CodeCamp	Total Number of Aspirants
< 6 months	Software Developer	34 (69%)	6 (27%)	40 (57%)
	Hybrid Role	1 (2%)	4 (18%)	5 (7%)
6-9 months	Software Developer	7 (14%)	1 (5%)	8 (11%)
	Hybrid Role		2 (9%)	2 (3%)
>9 - 12 months	Software Developer	3 (6%)		3 (4%)
	Hybrid Role	1 (2%)	2 (9%)	3 (4%)
>12 months	Software Developer	1 (2%)	5 (23%)	6 (9%)
	Hybrid Role		1 (2%)	1 (1%)
Did not accomplish career transition		1 (2%)	1 (2%)	2 (3%)
TOTAL		48	22	70*

*This number excludes the 5 computer science majors, and the 8 aspirants who had prior experience in an IT related field

Table 6. How skill development within learning collectives compares to prior conceptualizations of occupational learning

	Occupational Socialization	Situated Learning	Shadow Learning	Skill Development within Learning Collectives
Role of the novice in relation to the occupational community	Insider	Insider	Insider	Outsider, aspiring to cross the inclusion boundaries of the occupation
Role of formal instruction	Formal instruction occurs during schooling	Importance of formal instruction is downplayed. Novices learn on the job, working alongside incumbents. For skilled occupations, novices generally share formal educational background		Minimal formal instruction Aspirants come from diverse educational backgrounds
Access to legitimate peripheral participation opportunities	Full access	Full access	Limited access. Organization of work and associated technologies hinders effective legitimate peripheral participation	Very limited access. In vast majority of cases, aspirants need to get hired as 'full member' to access professional practice*
Locus of learning	In the classroom, and during engagement with professional practice in the workplace	Engagement with professional practice in the workplace	Successful learners opportunistically create their own learning opportunities across different professional and informal spaces	Learning happens at bootcamp and in other informal spaces through self-study and project work, in collaboration with peers and near-peers
Outcome of learning	Occupational skill and identity development License to practice	Occupational skill and identity development License to practice	Only shadow learners develop necessary skills for robotic surgery Both successful and unsuccessful learners are licensed to practice	Aspirants learn basic occupational skills, and how to continue learning Aspirants experience change in identity No license to practice necessary. Yet aspirants' lack of credentials and lack of professional experience make job search difficult. Aspirants need to demonstrate skills through projects to convince employers of their hirability

*Novices could potentially participate in open source software development projects, which would constitute a case of legitimate peripheral participation, but I did not observe any such cases.
Also, at the time of fieldwork, a very limited number of apprenticeships designed for bootcamp graduates were offered by technology companies. Since these apprenticeships were rare and did not constitute a usual part of the breaking in experience of aspirants, I do not report on them in this paper.